Design, Implementation and Analysis of Efficient Hardware-based Security Primitives

By

N. NALLA ANANDAKUMAR

Under the supervision of

Dr. Somitra Kumar Sanadhya

Dr. Mohammad S. Hashmi

Indraprastha Institute of Information Technology Delhi

September, 2019

Design, Implementation and Analysis of Efficient Hardware-based Security Primitives

By

N. NALLA ANANDAKUMAR

Submitted

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

to the

Indraprastha Institute of Information Technology Delhi

September, 2019

CERTIFICATE

This is to certify that the thesis titled "**Design, Implementation and Analysis of Efficient Hardware-based Security Primitives**" being submitted by **N. Nalla Anandakumar** to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of **Doctor of Philosophy**, is an original research work carried out by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

September, 2019

Dr. Somitra Kumar Sanadhya
Associate Professor & Head
Department of Computer Science & Engineering
IIT-Ropar, India

Dr. Mohammad S. Hashmi
Associate Professor
Department of Electronics & Communication Engineering
IIIT-Delhi, India

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisors Dr. Somitra Kumar Sanadhya and Dr. Mohammad S. Hashmi for their tremendous support, guidance, and encouragement during the past four years. I would like to thank them for giving me the opportunity to pursue a Ph.D. in their research groups and to carry out cutting-edge research in the field of hardware security. Thanks to the positive attitude towards work and life that the professors have provided, the past four years have been an unforgettable time in my entire life that has given me a tremendous development, both in research career and personally. I could not have imagined having excellent and resourceful advisors as my mentors.

I would also like to show gratitude to the annual review committee, including Dr. Donghoon Chang and Dr. Sujay Deb for their patient hearing and suggestions. I would like to add a special note of thanks to all the reviewers and co-authors of my PhD research articles.

I would like to extend my gratitude to my thesis committee members Dr. Anupam Chattopadhyay and Dr. Shivam Bhasin for helping me with their valuable feedback and time to finish my successful thesis.

I would also like to thank all my dear friends from the IIIT-Delhi, especially, Amit Kuamr Chauhan, Dinesh Rano, Hemanta Kumar Mondal, Rahul Gangopadhyay and Deepayan Banerjee for their valuable friendship and providing pleasant academic atmosphere. They had major contribution in making my PhD life at IIIT-Delhi easy going and memorable. I would also appreciate much help from IIIT-Delhi administrative staffs, especially, Ms. Priti Patel.

I am very grateful to Professor R. Balasubramanian (ex Executive Director at SETS, Chennai) and Mr. S. Thiagarajan (ex Registrar at SETS, Chennai) who allowed me to start my doctoral program at IIIT-Delhi. I would like to thank Dr. N. Sarat Chandra Babu, Executive Director, SETS, Chennai for his heartfelt support, without which all this was not possible. I also wish to thank my colleagues from the SETS, Chennai for creating a good working environment.

I would like to thank my family: my parents N. Nachimuthu and N. Sellathal for their love, encouragement and perpetual support. I would like to thank my sister N. Indira, my brother-in-law A. Mahesh Kumar and my In-laws for their support around over the years. And my two sons N. Aadhish and N. Aadahv, who helped

in their own charming ways to reduce the stress during the years of my graduate career. Above all, I thank my wonderful wife N. Nandhini for all her sacrifices, support, and encouragement. It is my greatest honour to dedicate this work to them.

I also express my regards to all of those who supported me in any respect during the completion of my research. Last but not the least, to all other people I forgot to mention here: You know I owe you, thanks!

N. Nalla Anandakumar
September 2019

"Destiny is not a matter of chance. It is a matter of choice. It is not a thing to be waited for, It is a thing to be achieved."

– William Jennings Bryan

Dedicated

To
My Family

ABSTRACT

Internet of Things (IoT) is a vast and rapidly growing technology right now in the world of innovation. Billions of new electronic devices are going to be connected to the internet in wide-ranging applications. With this massive increase in adoption and utilization of new technology, security vulnerabilities are growing exponentially as well. Traditionally, conventional cryptographic primitives are used in order to provide security of these devices. The security of the cryptographic protection relies on the secrecy of the key. Typically, secret keys, which are used as device identification (IDs), are stored in non-volatile memories (NVMs), and combine cryptographic primitives to implement information encryption and authentication. However, through such traditional technique, secret keys are vulnerable to various kinds of attacks and can be easily obtained or cloned. Further, maintaining such secrets in NVMs is difficult and expensive. In addition, random key generation and key exchange are also very challenging in secure IoT applications.

Physically Unclonable Function (PUF) promises to be a critical hardware security primitive to provide an alternative method to create unique signatures (IDs) from complex physical characteristics of ICs rather than storing the IDs in non-volatile memories. Eventually these IDs can be used to authenticate devices and also to generate secret keys for cryptographic functions. A True Random Number generator (TRNG) is another important hardware security primitive that generates high entropy random numbers (keys) from a physical process for use in key exchange/agreement, encryption, and digital signature, etc. The IoT infrastructure adopts a large number of these hardware-based security primitives in order to securely exchange data in an effective and resource efficient manner. Furthermore, one of the major requirements of PUF and TRNG intended for IoT applications is that the device area must be efficiently utilized. Unfortunately, the huge area consumption of many PUF and TRNG implementations on Field-Programmable Gate Arrays (FPGAs) made them infeasible in IoT environments. Therefore, we undertake the study and development of new techniques to design, develop and implement highly efficient PUFs and TRNG for FPGAs in the context of IoT applications in this thesis.

In the first part of this thesis, we study different techniques for improving performance characteristics of PUFs. In this context, we carry out the design,

development, implementation and evaluation of four major types of PUFs has for IoT security. These PUFs fall in three categories: memory based, delay based or hybrid PUFs. The first design we study is RS-Latch based which is a memory based PUF. Next two designs are Ring oscillator and Arbiter based, and fall in the category of delay based PUF. The fourth design is a hybrid of RS Latch and Arbiter PUF designs. All the four designs have been thoroughly tested on FPGA devices. The enhancement in performance of the new designs is achieved through the incorporation of various novel techniques. Performance metrics of these designs have been presented and compared to the state of the art PUFs. It has also been shown that the proposed designs yield the most area-efficient conventional and hybrid PUFs reported so far. Moreover, the proposed PUFs are resistant to temperature, supply voltage, and correlated process variations making them attractive for IoT applications.

In the second part of this thesis, we design and develop a ring oscillators based true random number generation on FPGA. The quality of generated true random bits can be improved by employing different new techniques. Subsequently experimental evaluation and comparisons with existing techniques are presented. Further, our proposed implementation provides a very good area-throughput trade-off and high entropy rate of the produced output bits when compared to the existing state-of-the-art.

Lastly, in the third part of this work, we focus on efficient FPGA implementation of elliptic curve based authenticated key agreement protocol for IoT devices using PUF and TRNG. In this context, we design and develop a novel hardware architecture for Binary Edwards Curve (BEC) point multiplication. Subsequently, an FPGA design of elliptic curve based key agreement protocol (ECMQV) using PUF and TRNG is presented. The obtained implementation results show that the proposed architecture yields a better performance when compared to the existing state-of-the-art.

List of Related Publications

1. **N. Nalla Anandakumar**, Mohammad S. Hashmi and Somitra Kumar Sanadhya. "Compact Implementations of FPGA-based PUFs with Enhanced Performance". in *30th IEEE International Conference on VLSI Design 2017 (VLSID 2017)*, India, pages 161-166

2. **N. Nalla Anandakumar**, M. Prem Laxman Das, Somitra Kumar Sanadhya and Mohammad S. Hashmi. "Reconfigurable Hardware Architecture for Authenticated Key Agreement Protocol Over Binary Edwards Curve". in *ACM Transactions Reconfigurable Technology and Systems (TRETS)*, vol. 11, pages 12:1–12:19. 2018.

3. **N. Nalla Anandakumar**, Somitra Kumar Sanadhya and Mohammad S. Hashmi, "FPGA-Based True Random Number Generation Using Programmable Delays in Oscillator-Rings". *IEEE Transactions Circuits and Systems (TCAS)*, 2019 (Accepted, available online in the IEEE xplore).

4. **N. Nalla Anandakumar**, Mohammad S. Hashmi and Somitra Kumar Sanadhya, "Design and Analysis of FPGA-Based PUFs with Enhanced Performance for Hardware Oriented Security". *ACM Transactions on Embedded Computing Systems (TECS)* (Under Review).

5. **N. Nalla Anandakumar**, Mohammad S. Hashmi and Somitra Kumar Sanadhya, "Efficient and Lightweight FPGA-based Hybrid PUFs with Improved Performance". *International journal of Microprocessors and Microsystems, Elsevier* (Under Review).

Contents

List of Figures

List of Tables

Abbreviations

IoT :	Internet of Things
ID :	Identity
IC :	Integrated Circuits
NVMs :	Non-volatile Memories
FPGAs :	Field-Programmable Gate Arrays
PUFs :	Physically Unclonable Functions
CRPs :	Challenge-Response Pairs
TRNGs :	True Random Number generators
A-PUF :	Arbiter-based PUF
RO-PUF :	Ring Oscillator-based PUF
RS-LPUF :	RS Latch-based PUF
ROs :	Ring Oscillators
LFSR :	Linear Feedback Shift Registers
ASICs :	Application-Specific Integrated Circuits
PDLs :	Programmable Delay Lines
TMV :	Temporal Majority Voting
CMOS :	Complementary Metal-Oxide-Semiconductor
HDL :	Hardware Description Language
LUTs :	Look-Up Tables
SCA :	Side-Channel Attacks
ECC :	Elliptic curve cryptography
PKC :	Public-Key Cryptography
BEC :	Binary Edwards Curve
CPU :	Central Processing Unit

DH :	Diffie-Hellman
ECDH :	Elliptic Curve Diffie-Hellman
ECMQV :	Elliptic Curve Menezes, Qu and Vanstone
I/O :	Input/Output
PM :	Point Multiplication
PC :	Personal Computer
UART :	Universal Asynchronous Receiver-Transmitter
LSBs :	Least Significant Bits
MSBs :	Most Significant Bits
PRNG :	Pseudo Random Number Generator
CI :	Confidence Interval
HD :	Hamming distance
SEM :	Standard Error of the Mean
CTW :	Context-Tree Weighting
NIST :	National Institute of Standards and Technology
SVM :	Support Vector Machine
LR :	Logistic Regression
ES :	Evolution Strategies
XOR :	Exclusive OR (logical)
CLBs :	Configurable Logic Blocks
NBTI :	Negative-Bias Temperature Instability
BER :	Bit Error Rate
ML :	Machine Learning
EM :	Electromagnetic
FFs :	Flip-Flops
SRAM :	Static Random Access Memory

Chapter 1

Introduction

In recent years, there has been a tremendous growth in the adoption of Internet of Things (IoT). A variety of devices, such as smartphones, home appliances, sensors, medical devices and other network devices are now ubiquitous and widely used. IoT market is estimated to be worth 11 trillion dollars and 75 billion devices are likely to be connected by 2025 [62]. The security of communication between these devices is becoming increasingly important. Providing this security relies on well established cryptographic primitives for key generation, key agreement, identification and authentication, data confidentiality, etc. Therefore, these cryptographic primitives are paramount for IoT devices to be able to perform operations and critical tasks in a low-cost, highly efficient, and secure way. However, in many IoT applications, resources like CPU, memory, and battery power are limited which limits their adoption of classical cryptographic security solutions since many of these solutions are resource demanding. Many conventional approaches (e.g. key generation and storage, key agreement, encryption, and digital signatures) are expensive, slow, resource inefficient, and increasingly vulnerable to physical attacks. Hardware-based security primitives such as PUFs and TRNGs can overcome these limitations and provide true random numbers in order to establish security and trustworthiness in IoT systems.

PUFs rely on the fact that every chip that is manufactured has its unique physical characteristics. This is similar to fingerprints of human beings. The PUFs

exploit unavoidable and uncontrollable random process variations that exist in IC manufacturing to generate their own unique hardware characteristics, which can be used to create unique IDs or to generate unique secret keys. PUFs can significantly increase physical security by generating secret IDs/keys that only exist in a digital form when a chip is powered on and running. Further, these secret IDs/keys are unique and extremely hard or impossible to clone. IDs/Keys generated by PUFs can resist tampering attacks because the underlying nano-scale structural disorder will most likely be damaged during the course of physical tampering of a PUF device. Authentication, identification, and key generations are the most common use cases for PUFs.

The TRNGs are another heavily used primitive in security applications. These are common in cryptographic applications such as key generation, random padding bits, and generation of challenges and nonces in authentication protocols. There effectiveness of a TRNG depends on the quality of random numbers it produces. In fact, the security properties of many algorithms are crucially dependent on the ability of the TRNG to produce unpredictable and unbiased bits in a cost effective manner.

It can be inferred that the PUF and TRNG should provide cost effective and efficient trustworthiness of the physical hardware platforms. Therefore, PUF and TRNG are promising security primitives for Internet of Things.

1.1 Hardware Security Primitives

Modern cryptographic methods rely on the existence of keys to perform any secret exchange of information. The keys may be long-term keys, session keys for block ciphers or signature keys, or ephemeral keys for public key algorithms. The key is to be kept secret and is an interesting target for adversaries. Typically, secret keys, which are used as device identification (IDs), are stored in non-volatile memories (NVMs), and combine cryptographic primitives to implement information encryption and authentication. However, keys stored in

such media are vulnerable to various kind of attacks and can be extracted or cloned. Further, maintaining such secrets in NVMs is difficult and expensive. Thus, we need alternative cryptographic protocols based on hardware security primitives that do not require an explicit secret key/ID storage. Moreover, random numbers are necessary to secure several cryptographic protocols and are crucial to the security of the protocols. Therefore, hardware-based security primitives such as PUFs and TRNG play important roles in protecting and securing a system in IoT applications.

1.1.1 Physically Unclonable Functions (PUFs)

A PUF is a function that is realized by a physical system and is easy to evaluate but the physical system is hard or impossible to clone [120]. PUFs possess the ability to extract unique signature, from the intrinsic process variability of silicon devices, using a challenge and response mechanism. Eventually these unique signatures (IDs) can be used to authenticate devices and also to generate secret keys for cryptographic functions. The basic functionality of a PUF can be described mathematically as a mapping $R = PUF(C)$, where C is a challenge (an external stimulus) applied and R is the response (output) generated by the PUF (Fig. 1.1a). Thus, the considered PUF must satisfy the following security related properties, presented in [113] (Fig. 1.1b-g):

Reproducible: The responses generated from the same PUF on the same challenge should always be very similar during multiple evaluations (Fig. 1.1b).

Unique: The responses generated from different PUFs on the same challenge should always be largely different (Fig. 1.1c).

Unclonable: It should be impossible to create a physical copy of any PUF instance (Fig. 1.1d).

One-way: Predicting the challenge from a given response should be infeasible (Fig. 1.1e).

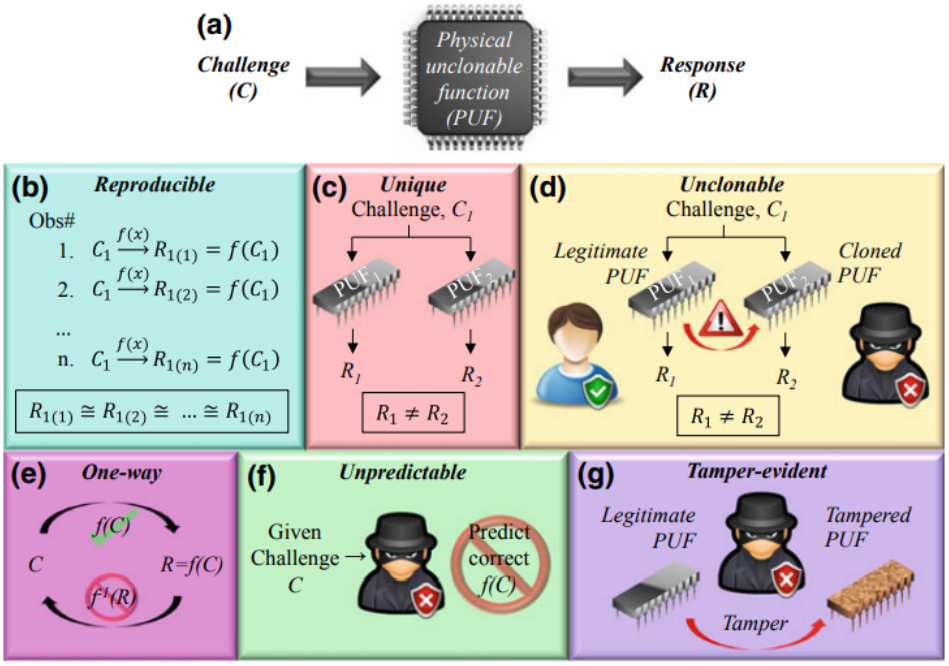

FIGURE 1.1: Schematic representations of essential features of PUF [113]

Unpredictable: The adversary should not be able to compute the response of a PUF to a challenge efficiently, given a limited set of other challenge response pairs (CRPs) (Fig. 1.1f).

Tamper-evident: The PUF should be tamper-evident, i.e. the PUF system should permanently change its functionality when a physical attack on the PUF is carried out (Fig. 1.1g).

1.1.1.1 Concept of PUFs

PUFs challenge-response relationship is determined by manufacturing nano-scale process variations in the logic and interconnect of an integrated circuit (IC) chip. From a circuit design perspective the process variations can be generally divided into two major groups: *die-to-die variations* and *within-die variations* [110].

As shown in Fig. 1.2, *die-to-die variations* have a variation radius larger than the die size including within wafer, lot to lot, wafer to wafer, and fab to fab variations. These variations affect all the circuits within the die equally. On the other hand, *within-die variations* refer to the variations that occur between various

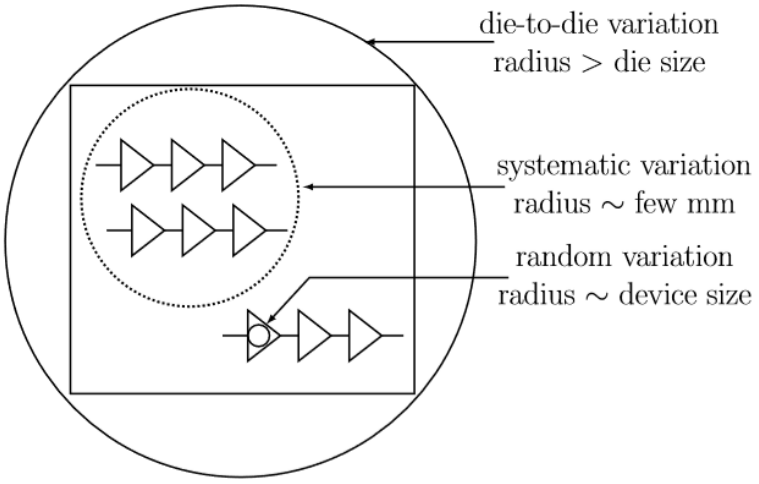

FIGURE 1.2: Classification of silicon process variations [110]

circuit elements of the same die. They can be further divided into *systematic* and *random* variations. The radius of *systematic* variation is on the order of few millimeters. Moreover, the *systematic* variations are the component of variation which is caused by imperfections in the fabrication process and they normally show spatial correlation behavior. Finally, *random* variations are non-systematic which is caused by sources such as random concentration of dopants in transistors and variation in gate oxide thickness. These are intrinsic to the silicon material and cannot be controlled. Moreover, the radius of this variation is comparable to the sizes of individual devices, so each device can vary independently. This latter type of variation (i.e. random) is the main cause for producing random responses in a PUF which is unpredictable and unclonable. It is very difficult, almost impossible, to build an exact identical copy of a PUF instance. No successful physical cloning has been reported so far even when given the exact manufacturing process that produced it.

Functionality of a PUF can be generally divided into two parts: namely, enrollment and evaluation. During the enrollment process, a PUF is characterized to extract a reference response R_{ref} corresponding to a given challenge. The CRP is stored in a secure database by an authorized entity. After the enrollment process is over, the PUF is deployed in applications. For example, it may serve the purpose of device authentication or may be used as a secret key in a cryptographic operation. During evaluation, the PUF is provided with the challenge and generates a corresponding

response R. If $R_{ref} = R$ then the device is authenticated successfully otherwise an encryption/decryption occurs.

1.1.1.2 Classification of PUFs

PUFs can be classified based on fabrication method and security strength, as shown in Fig. 1.3

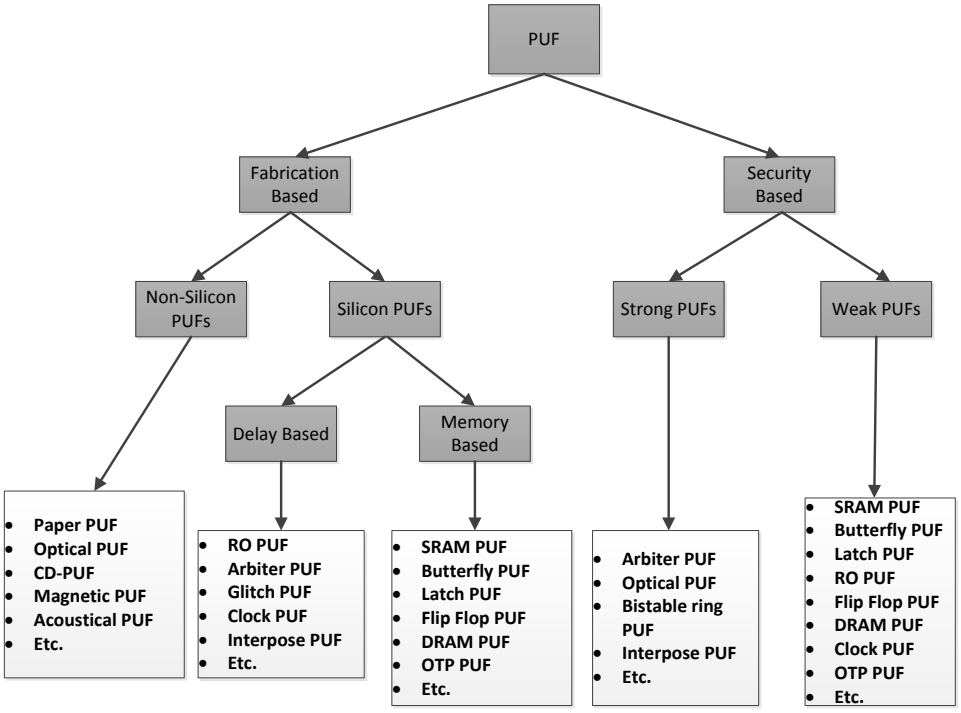

FIGURE 1.3: Different types of PUFs

Based on fabrication: PUFs are generally categorized into two major groups based on the fabrication [172]: Silicon and Non-Silicon PUFs.

- Non-Silicon PUFs: Generally these type of PUFs can be constructed using a non-silicon material. Some examples of this type of PUFs are Optical PUF, paper PUF, acoustical PUF, magnetic PUF, etc. [103, 172].
- Silicon PUFs: These PUFs utilize the uncontrollable manufacturing variations to generate a unique signature for each IC. According to the different source of variation, silicon PUFs can be mainly categorized

as delay-based PUFs and memory-based PUFs. Delay-based PUFs exploit race conditions (e.g. Arbiter-PUF [143]) and frequency variations (e.g. RO-PUF [143]) found in integrated circuits. On the other hand, Memory-based PUFs are based on the instability of volatile memory cells. Some examples of these type of PUFs are SRAM-PUF [50], RS-LPUF [164], Flip-Flop PUF [90], Butterfly PUF [78], DRAM [149], etc. For more details one may refer to [103].

Based on security (Based on CRPs): PUFs can also be classified into two major classes based on the size of their challenge-response space [48, 50, 126, 174]: Weak PUFs and Strong PUFs. Weak PUFs have limited number number of CRPs and they are more suited to applications such as key generation in cryptographic functions, seeding a pseudo random number generator (PRNG), and identity generation. Strong PUFs have a large CRP space in contrast to the weak PUFs.

An attacker cannot access the response of Weak PUFs because these responses must be kept private (never leave the chip) and are only accessed as required. The SRAM-PUF, Ring Oscillator-based PUF (RO-PUF) and RS Latch-based PUF (RS-LPUF) are typical examples of weak PUFs [172]. Alternatively, Strong PUFs have a large set of CRPs and can be used directly for authentication without additional cryptographic hardware. It is assumed that the attacker may have access to the CRPs; however, it should be impossible for them to attack the CRPs in a reasonable time frame (for example, a few days or weeks). Typical examples of Strong PUFs are Arbiter-based PUF (A-PUF) and Bistable Ring PUF [172]. It is pertinent to note that the prefixes "strong" or "weak" are not used to imply security level of a PUF in the literature. The classification is based only on the size of the CRP space.

1.1.1.3 Attacks Against PUF

The adoption of a PUF as key storage and generation avoids some of the shortcomings of non-volatile memory-based approaches. It is generally harder

to perform an attack aimed to read out, predict or derive responses from a PUF than to gain access to a non-volatile memory. However, PUF implementations may be subjected to some attacks. In this subsection, we will discuss a set of possible attacks that can be attempted against PUF circuits and the countermeasures to them in detail.

1) *Modeling Attacks:* Modeling attacks are the most efficient and powerful attack for strong PUFs, where the adversary builds an accurate model (clone) of the PUF and intentionally collects a large number of CRPs to train the model. An adversary who creates a well-trained model can accurately predict the responses of unknown challenges. A successful modeling attack can replace a legitimate PUF and steal sensitive information or get access to restricted resources. Ruhrmair et al. reported that arbiter PUFs are weak against machine learning attacks [127]. They further analyzed the PUFs in the context of security protocols in [128]. We note that previous works have most often utilized machine learning based models as a way to attack many PUF circuits [32, 43, 127]. The model of a PUF instance is built based on CRPs or some side channel information by using Logistic Regression (LR) [127], Support Vector Machine (SVM) [55] or Evolutionary Strategies (ES) [43]. In order to resist modeling attacks, many defense mechanisms are proposed. These strategies can broadly be divided into two categories: CRP obfuscation [31, 33, 44, 88, 108, 169], and adding non-linearity to the PUF structures [173].

2) *Physical attacks:* Physical attacks correspond to gate level or transistor level characterization such as delay, leakage, or some other device metric [118]. Tajik et al. proposed a photonic emission analysis mechanism to characterize an arbiter PUF [147]. In a semi-invasive manner, from the IC's backside, Nedospasov et al. [111] stimulate the inverters of an SRAM PUF cell with a near-infrared laser to reveal the SRAM cell contents. Likewise, Tajik et al. [146] use a laser to iteratively disable all but one arbiter chains that constitute an XOR PUF to learn each individual chain of the XOR PUFs. It is possible to prevent invasive active attacks on control logic of the PUF circuit. One possible strategy is to enclose the control logic by the delay wires of the PUFs [46]. These wires normally introduce path delays that the PUF circuit uses to determine its response. Therefore, if

invasive attacks attempt to probe the control logic then the PUF secret will be altered and damaged.

3) *Side-Channel Attacks (SCA):* A PUF may be attacked passively by using side-channel information such as execution time, power consumption or electromagnetic radiation emanating from a chip containing a PUF. In 2010, Karakoyunlu et al. [69] successfully attacked the software implementation of error correction of the PUF using a power measurement based attack. In 2013, Merli et al. [105] successfully attacked an RO-PUF using an EM attack that directly targeted the PUF and not the error correction. Mahmoud et al. [92] proposed to combine modeling attacks with power side-channel attacks to better characterize PUFs. In 2014, Becker and Kumar [11] measured power traces in order to model Arbiter PUF. Moreover, the authors in the above works have pointed out potential countermeasures to their proposed attacks.

4) *Cloning attacks:* In a cloning attack, an adversary makes the exact physical copy (or clone) of the original PUF with identical challenge-response behavior. However, in practice, no successful physical cloning attacks on PUF has been reported so far. The main reasons for it is that a large number of chips are manufactured using the same layout mask, and it is very difficult to measure the complex delay characteristics of each of them. Further, it is hard to control the manufacturing variation. Due to these reasons we believe that it is extremely difficult to successfully clone a PUF.

1.1.1.4 PUF Quality Metrics

PUFs are typically evaluated from the collected CRPs of several PUF instances of the same type on a specific platform. The common performance metrics are *uniqueness*, *reliability*, *uniformity*, and *bit-aliasing* [94].

1) *Uniqueness* measures the variation of responses obtained from different chips for the same set of challenges. If X_i and X_j are the n-bit responses of ith and jth chips respectively for the same challenge C, then the *uniqueness* (HD_{INTER})

is expressed as the average inter-chip hamming distance (HD) among k devices given by (1.1),

$$Uniqueness = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{\mathsf{HD}(X_i, X_j)}{n} \times 100\% \tag{1.1}$$

where $\mathsf{HD}(X_i, X_j)$ is the Hamming distance between n bit strings X_i and X_j, and k is the number of chips (devices). The ideal value of *uniqueness* is 50%.

2) *Reliability/Stability/Steadiness/Repeatability* determines how efficiently a PUF can generate the same response at different operating conditions (ambient temperatures or supply voltages) over a period of time for a given challenge. For the i^{th} chip, the average intra-chip HD is estimated using (1.2). Then the *reliability* of a PUF is defined by (1.3). Here, X_i is the reference response of the i^{th} chip, $X_{i,t}$ is the response generated by it at time t, and s is the number of responses of same set of challenges.

$$HD_{INTRAi} = \frac{1}{s} \sum_{t=1}^{s} \frac{\mathsf{HD}(X_i, X_{i,t})}{n} \times 100\% \tag{1.2}$$

$$Reliability_i = 100\% - HD_{INTRAi} \tag{1.3}$$

The ideal value of *reliability* is 100% (i.e., ideal value for HD_{INTRA} is 0%) and the *average reliability* of k chips can be calculated using (1.4).

$$Average\ Reliability = \frac{1}{k} \sum_{i=1}^{k} Reliability_i \tag{1.4}$$

3) *Uniformity* determines how uniform the proportion of $0's$ and $1's$ are in the PUF response and is calculated by (1.5).

$$Uniformity_i = \frac{1}{n} \sum_{j=1}^{n} r_{i,j} \times 100\% \tag{1.5}$$

where $r_{i,j}$ is the jth bit of n-bit response of i^{th} chip. The ideal value of *uniformity* is 50%.

4) *Bit-aliasing* happens when different chips may produce nearly identical PUF responses, which is an undesirable effect. The *bit-aliasing* is calculated using (1.6):

$$Bit\text{-}aliasing_j = \frac{1}{k}\sum_{i=1}^{k} r_{i,j} \times 100\% \tag{1.6}$$

where $r_{i,j}$ is the jth bit of n-bit response of i^{th} chip and k is the number of chips. The ideal value of *bit-aliasing* is 50%.

The evaluation metric of PUF structures may vary in different application scenarios [24]. For example, the metrics of uniqueness, reliability, and uniformity may have different importance in different PUF usage scenarios such as identification, authentication or encryption as explained below:

- Identification: The PUF can be used to generate a "serial number" to identify and/or track parts through manufacturing. Uniqueness is the most important metric in this situation. Further, reliability is not a major concern in the scenario of identification as long as the Bit Error Rate (BER) is relatively small. A large BER may lead to unacceptably high probability of uniqueness which reduces the number of unique IDs that can be generated and used.

- Authentication: The PUF is used to securely identify the chip in which it is embedded to an authority through corroborative evidence. In this scenario, randomness (uniformity) is the most critical metric. In addition, the PUF should also have good uniqueness properties. Moreover, a very large CRP space is necessary to prevent adversaries from reading all the responses and building a clone. The large CRP space also prevents ML attacks against the SCA adversaries.

- Encryption: The PUF is used to generate key for symmetric encryption algorithms and to generate a random nonce that can be used to select

specific public-private key pair for asymmetric encryption. The randomness, reliability, and uniqueness are critical in such applications. However, a large CRP space is not necessary in cases where only a few keys need to be generated over the lifetime of the chip. In this case, BER must be zero which may require error correction.

1.1.2 True Random Number Generations (TRNGs)

TRNGs are widely used in cryptographic applications such as key generation, random padding bits, and generation of challenges and nonces in authentication protocols. Moreover, TRNGs also find use in lottery drawings, computer games, gambling and probabilistic algorithms. The TRNGs must fulfill strict statistical requirements, be unpredictable, and generate truly random numbers by making use of a physical source that is non-deterministic. In general, a poor random number generator often leads to decrease in the complexity of attacking a system using such a generator. For example, insecure pseudo-random number generator (PRNG) used on Mifare Classic tags reduced the attack complexity and allowed attackers access to the smart card's secret key [112]. These issues can be addressed by utilizing secure TRNGs to produce the needed random bits in cryptographic systems. The quality of randomness of a TRNG is usually assessed through statistical test suites such as Diehard [101] and NIST [10] while the unpredictability is verified by estimating entropy-per-bit through a stochastic model [135].

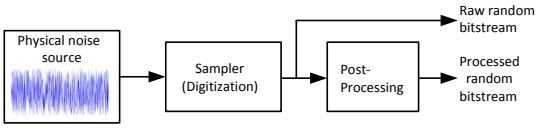

FIGURE 1.4: Block diagram of True Random Number Generator

Typical TRNGs use a single source of entropy and post-processing operation. Fig. 1.4 shows the block diagram of a typical TRNG architecture. Randomness is first extracted from the physical noise source and then this is interpreted into raw random bit-stream using a sampler (digitization). In practice, the raw random

bit-stream usually does not bear good quality of randomness. Therefore, auxiliary post-processing operations, such as Neumann corrector [152] or hash function [67] is required to improve the quality of the output TRNG bit stream and increase its randomness.

A key resource for a TRNG design is the entropy source used in the design. Commonly used sources of entropy are thermal noise, metastability, timing jitter in circuits, chaos circuits, quantum effects etc. These are briefly explained next.

1) *Thermal Noise:* It is also known as Johnson–Nyquist noise [114]. This is the electronic noise generated by the thermal agitation of the charge carriers inside an electrical conductor at equilibrium. This agitation takes place regardless of whether an external voltage is applied or not. This type of entropy source is normally suitable for application specific integrated circuits (ASICs) implementations. A well-known TRNG from this family was introduced by Intel [67]. In this design, first the Johnson thermal noise over a resistor is amplified and then digitized by using high-speed oscillator. Finally, a von Neumann post-processing operation is used to improve the statistical properties of the generated random numbers. More recent TRNGs from this family can be found in [25, 116].

2) *Metastability:* Metastability is the most commonly used entropy source for both FPGA and ASIC TRNGs. In early designs, the metastable operation of latches was utilized as the entropy source [13]. In 2001, Walker et al. evaluated the metastability of a DFF (D flip-flops) circuit for random number generation in [155]. Utilizing the write collisions in memory blocks as entropy source for random number generation was presented in [52]. The last passage time of ring oscillators is utilized as the noise source in [123]. The metastability of flip-flops was used for generating true random numbers [99]. The metastability was achieved in [99] by using Programmable delay lines (PDLs) that accurately equalize the signal arrival times to flip-flops. The metastable operation of RS latches was utilized as a entropy source for generating true random numbers in [54]. A TRNG, which utilizes a cross-coupled NAND gates, was recently proposed by Li et al. [81].

3) *Timing Jitter:* In electronic systems, timing jitter is defined as the variation in a signal's timing from its nominal value. It is popular to use the timing jitter of free-running ring oscillators (ROs) or Phase-locked loops (PLLs) as entropy sources for both FPGA and ASIC TRNGs. PLL-based TRNG was introduced in [37], and the optimization of PLL-based TRNG design was presented in [1]. A TRNG based on multiple ROs was introduced in [144], and was improved by Wold et al. in [159]. RO based TRNG designs were presented by utilizing multiple edges of an RO as the randomness source [167] and by employing a tetrahedral oscillator with large jitter to realize the TRNG [85]. Johnson et al. [65] presented an efficient and tunable TRNG based on the principle of beat frequency detection. A timing jitter based TRNG which utilizes a self-timed ring oscillator was introduced in [26] and its stochastic model was given by [27].

4) *Chaos Circuits:* Chaotic circuit denotes a simple electronic circuit that exhibits classic chaotic behavior. They amplify small changes at the initial states to produce significantly different future outputs. This type of entropy source is normally suitable for ASICs implementations. A TRNG using a double-scroll attractor was presented in [163]. A simple chaotic discrete-time systems based TRNG was introduced in [124]. The proposed TRNG relies on a simple mathematical model called discrete maps to generate chaotic signals. Beirami et al. present a framework for investigating the performance of chaotic-map TRNGs [12]. Analog-to-Digital Converters (ADCs) are also used for chaos-based TRNG. Exploiting nonlinear signal processing and chaos to design a TRNG was used by Callegari et al. in [19]. Another TRNG design based ADC which internally exploits a pipeline ADC modified to operate as a set of interleaved chaotic maps was presented by Pareschi et al. in [115].

5) *Quantum Effect:* Various quantum effects can be utilized as a randomness source. John walker [154] presented a TRNG based on radioactive decay as a source of entropy. A popular type of Quantum Random Number Generators (QRNGs) are based on the detection of a single photon between two outputs of a beam splitter. Typical sources of photons are single-photon emitters, lasers and light emitting diodes. There are various TRNG designs in this family, ranging from

academic literature [141] to commercial products [129]. In [64], Jennewein et al. presented a physical QRNG with a throughput of 1Mbps. A QRNG, which utilizes a 16x16 array of detector pairs, was recently proposed by Massari et al. [161]. Arbiter modules were implemented to determine which detector in the pair of two received a photon first. The proposed design achieved a throughput of 128Mbps.

1.1.3 Possible Application Areas for Security Primitives

As discussed in previous sections, one can use these hardware besed security primitives for the purpose of securing physical devices to protect sensitive data and functionality from an attacker. In the IoT context, PUFs can be used to assign unique IDs to billions of connected devices. Eventually these IDs can be used for device identification and/or to generate secure keys for cryptographic algorithms and protocols such as PUF-based block cipher [7], PUF based hardware authentication [34, 143], PUF based key exchange, and key agreement protocols [18, 51]. Further, TRNGs are widely used security primitives used as, for example, key generators for symmetric as well as asymmetric key cryptosystems (e.g. AES and ECC). In some algorithms (e.g. digital signatures) or protocols (e.g. key agreement, zero-knowledge), random numbers are intrinsic to the computation [104]. Due to their being inexpensive with low hardware footprint, PUFs and TRNGs can also be used in securing IoT environments.

1.2 Related Works: FPGA based PUFs and TRNG

Various PUF and TRNG implementations on both ASICs and FPGAs have been reported. In this thesis, we focus on design and analysis on many FPGA based PUFs and TRNGs with enhanced performance for hardware oriented security. Compared to ASIC, the design of PUF primitives on FPGAs are preferred owing to their inherent flexibility, quick turn-around, and reconfigurability. Two

major FPGA manufacturers namely Intel (formerly Altera) [42] and Xilinx [119] have recently announced PUF implementations in their respective products for security applications. Some interesting applications of FPGA-based PUFs can be found in secure key generation [55, 143], IP protection [50, 125], fingerprint identification [57], IC counterfeit detection [168], and IoT security [86, 100].

There exist many delay-based FPGA PUFs that include the Arbiter-PUF [80], RO-PUF [80], glitch PUF [80], HELP [80], Intrinsic Personal PUF (IP-PUF) [80] etc. Among these, the RO-PUF and the Arbiter-PUF are the most popular ones. Arbiter PUFs fall in the category of strong PUFs. Many FPGA based arbiter PUFs were proposed in [4, 58, 98, 130]. The work [58] defined performance metrics and presented statistical evaluation results for FPGA based arbiter PUFs. To improve the PUF performance metrics, Majzoobi et al. [98] introduced an arbiter PUF using programmable delay line (PDL) implemented by lookup tables (LUTs). In [4] the authors also constructed PDL based compact arbiter PUF. In [130] a design methodology to implement bias-free PDL based arbiter PUF utilizing the hard macro feature of Xilinx CAD tools was described. To enhance the modeling robustness, Feed-Forward APUF [84], XOR APUF [143] and Lightweight PUF [97] were proposed with additional non-linear component in the FPGA based Arbiter PUF designs.

The RO-PUF [96] was proposed by Suh et al. in 2005 (Fig. 1.2). The PUF response in this case is derived from the difference in the oscillator frequencies of selected pairs of identically designed ROs. Note that RO-PUF is a weak PUF [171], since there are a limited number of challenge bits that can be configured for operating the RO PUF. Moreover, several earlier works have reported FPGA implementations of RO PUFs [4, 30, 86, 96, 107, 165, 170, 171]. The notion of configurability in RO PUF (CRO) has been introduced by Maiti and Schaumont [96] to reduce noise in the PUF responses. Similar implementations and improvements have been reported in [30, 107, 170]. XRRO (XOR-based Reconfigurable RO) PUF [86] is evolution of RO PUF that uses XOR gates instead of inverters. In [4], the authors constructed PDL based compact RO-PUF on FPGA. The RO-PUF reliability is

significantly improved by using frequency offset method [171] and phase calibration process [165].

Further, a number of FPGA prototypes of TRNG with post-processing designs have been proposed in the literature [16, 35, 54, 95, 99, 117, 144, 159]. These designs derive entropy from the jitter of Ring Oscillators (RO) [16, 35, 95, 117, 144, 159] or the metastability of flip-flops [54, 99] which is caused by setup or hold time violations of flip-flops (FFs). Researchers have investigated several ways for improving the performance of PUFs and TRNG. However, the existing solutions, although enhance PUF and TRNG performance, are still inferior when compared to the ideal desired metrics.

1.3 Problem Statement

After an extensive research survey on PUF primitive designs, we identify that the existing state-of-art techniques have severe limitations in most of the performance metrics namely reliability, uniqueness, and randomness. Another problem facing the wide scale deployment of hardware-based security primitives such as PUFs and TRNGs for IoT applications is that there is a high demand for low-cost resource efficient solutions. However, most of the current state of the art PUF and TRNG primitives are expensive for low area implementations. It is, therefore, imperative to investigate and propose novel solutions to address these pressing problems in the existing design approaches. This thesis attempts to address some concerns regarding design, development and implementation of highly efficient PUFs and TRNG for FPGAs with enhanced performance in IoT security systems.

1.4 Thesis Outline and Contributions

This section gives an overview of the structure of the thesis and highlights the personal contributions.

In Chapter 2, we have developed three major types of area efficient PUF designs and improving their qualities. One is a memory based PUF: RS-Latch based design. The second and third are delay based PUF: Ring oscillator and Arbiter based designs. These three designs have been thoroughly tested on FPGA devices. The enhancement in performance is achieved through the incorporation of various techniques such as internal variations of FPGA Look-Up Tables (LUTs) in terms of coarse and fine Programmable Delay Lines (PDLs), Temporal Majority Voting (TMV) scheme, and hard macro techniques for routing and placements of PUF units. Performance metrics of these designs have been presented and compared to the state of the art PUFs.

In Chapter 3, we present an area efficient hybrid PUF design on FPGA. Our approach combines units of conventional RS Latch-based PUF and Arbiter-based PUF which is then augmented by the PDLs and TMV for performance enhancement. The measured results on the FPGA demonstrate PUF signatures exhibits good uniqueness, reliability, and uniformity with no occurrence of bit-aliasing.

In Chapter 4, we design and developed ROs based true random number generation on FPGA. The programmable delay of FPGA LUTs has been used to achieve random jitter and to reduce correlation between several equal length oscillator rings, and thus improve the randomness qualities. Moreover, our proposed implementation provides a very good area-throughput trade-off and high entropy rate of the produced output bits when compared to the existing state-of-the-art.

In Chapter 5, we focus on efficient FPGA implementation of authenticated key agreement protocol for IoT devices using BEC, PUF and TRNG. In this context, A novel hardware architecture of binary Edwards curve (BEC) point multiplication using mixed w-coordinates of the Montgomery laddering algorithm has been developed. Subsequently, an FPGA design of elliptic curve based key agreement protocol using PUF and TRNG is presented. The key agreement protocol uses PUF for the unique long term secret key generation, TRNG for short term random

secret key generation, BEC for generating the public key corresponding to the secret key, and ECMQV for generating the shared secret key and key exchange. The obtained implementation results show that the proposed architecture yields a better performance when compared to the existing state-of-the-art.

In Chapter 6, In this final chapter, conclusions are drawn and some directions for future work are discussed.

■

Chapter 2

Design and Analysis of FPGA Based PUFs with Enhanced Uniqueness and Reliability

This chapter focuses on three FPGA based PUF designs that involves different strategies to achieve better uniqueness and reliability characteristics with very competitive area-throughput trade-offs.

2.1 Motivation

The performance of PUFs are defined in terms of *uniqueness*, *reliability*, *uniformity*, and *bit-aliasing*. These are often dependent on several internal and external factors. For example, factors such as the systematic or correlated process variation and the environmental noise caused by the voltage and temperature variations degrade the *uniqueness* and the *reliability* of PUF responses as well as the resiliency to external attacks [96]. The performance of PUFs are enhanced through Temporal Majority Voting (TMV) scheme [7, 102], hard macro techniques [96], Programmable Delay Lines (PDLs) [98], and combining its outputs [143, 162]. For example, the combination of PUF responses from multiple

PDLs require an aggregate function (XOR operation or crypto-algorithms) of PUFs existing in the system to enhance uniformity and security [143, 162]. Furthermore, high-resolution PDLs implemented by a single lookup table (LUT) on the FPGA can significantly improve the number of independent response bits by partially alleviating the problem of systematic design bias [98]. The TMV concept aids in mitigating the variability issues and in achieving more stable results by averaging N sequential measurements [7]. The hard macro technique, provided by standard design tools, is commonly used to enhance uniqueness and bit-aliasing of RO-PUF [96], A-PUF [130], and RS-LPUF [53]. However, the existing solutions, although enhance PUF performance, are still inferior when compared to the ideal desired metrics.

This work advances the state-of-the-art in the domain of FPGA-based PUF primitives. This has been achieved by incorporating the TMV scheme, hard macro techniques, and coarse or fine delay lines in conjunction with conventional PUF modules concurrently.

The key contributions of this chapter are:

- Area-efficient RO-PUF, RS-LPUF, and A-PUF designs on Xilinx Spartan-6 FPGAs.
- Demonstration of increase in the number of independent responses through the use of fine and coarse programmable delay lines (PDLs) of FPGA LUTs.
- Proposal of more stable (i.e., better reliable) PUFs by incorporating of TMV scheme in the conventional PUFs.
- Achievement of better PUF performance in terms of uniformity and bit-aliasing by XORing PUF responses together from multiple PDL configurations and utilizing the hard macro design technique (i.e., placement strategy) to make all the PDLs identical in terms of placement and routing.
- Detailed analysis of the three proposed PUF designs in terms of entropy, correlation resistance, and aging effects.

The organization of the chapter is as follows. Section 2.2 briefly discusses the Xilinx Spartan-6 FPGAs and PDLs. The implementation details of the proposed PUFs are presented in Section 2.3. Possible attacks on the proposed architectures and the countermeasures are discussed in Section 2.4. Experimental validation of the proposed design is given in Section 2.5, and discussion on implementation results along with area and processing speed comparisons with the existing techniques is given in Section 2.6. Finally, conclusions are presented in Section 2.7.

2.2 Preliminaries

2.2.1 Xilinx Spartan-6 FPGA Structure

In this work, Spartan-6 FPGA from Xilinx has been used to prototype the three proposed PUF designs. This FPGA is organized as a grid of interconnected Configurable Logic Blocks (CLBs) which can be further subdivided into two logical components called slices. In this FPGA, three different types of CLB slices, namely SliceM, SliceL, and SliceX, as shown in Fig. 2.1 exist. Each CLB consists of two slice types i.e., first is SliceM or SliceL and the other is always a SliceX. The SliceX (logic only), the most basic type, consists of four 6-input lookup-tables (LUTs) and eight flip-flops (FFs). The SliceL (logic and arithmetic only) is similar to SliceX but with additional multiplexers and carry chain whereas SliceM (logic, arithmetic and memory) also includes additional memory component. In this work, the 6-input LUT primitives [160] has been instantiated in Hardware Description Language (HDL) to represent logic gates in the proposed PUF instances to avoid the asymmetry caused by software synthesis. For example, Listing 2.1 give the pieces of code to represent inverter by using LUT-6 directly. The initial value of LUT-6 for inverter with input I5 should be `64'h00000000FFFFFFFF`.

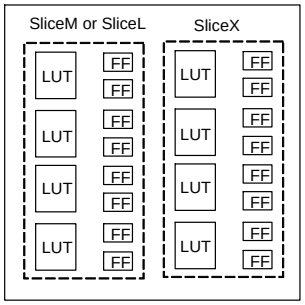

FIGURE 2.1: Slices per CLB of Spartan-6 FPGAs

```
module Inverter (input a, output z);
 LUT6 #(.INIT(64'h00000000FFFFFFFF))
    Not (
         .O(z),
         .I0(1'b0),
         .I1(1'b0),
         .I2(1'b0),
         .I3(1'b0),
         .I4(1'b0),
         .I5(a)
         );
endmodule
```

LISTING 2.1: Verilog source for Inverter

2.2.2 Programmable delay lines (PDLs)

The internal variations of FPGA Look-Up Tables (LUTs) can be generated from changes in the LUTs propagation delays under different inputs [98, 99].

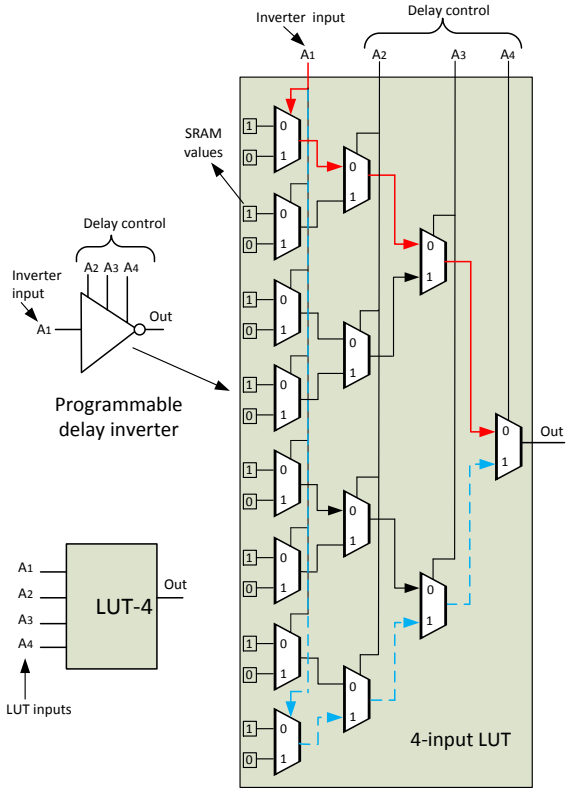

FIGURE 2.2: PDL using a 4-input LUT.

For example, the LUT in Fig. 2.2 is programmed to implement an inverter whose LUT output (0) is always an inversion of its first input (A_1). Other inputs A_2, A_3,

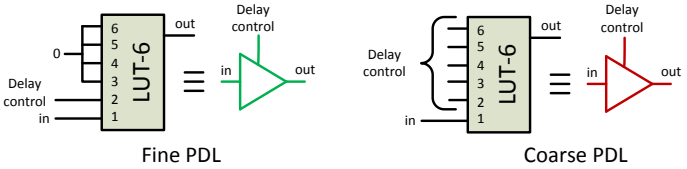

FIGURE 2.3: Fine and coarse PDLs implemented by a single 6-input LUT [99].

and A_4 act as "don't-care" bits but their values affect the signal propagation path from A_1 to the output (0). In this context, it has been shown, in Fig. 2.2, that the signal propagation path from A_1 to the output (0) is shortest for $A_2A_3A_4 = 000$ (marked with solid red line) and longest for $A_2A_3A_4 = 111$ (marked with dashed blue lines) for 4-input LUTs. Thus, a programmable delay inverter with three control inputs can be implemented by using one LUT. For the PDL, the first LUT input A_1 is the inverter input and the rest of the LUT inputs (three) are controlled by $2^3 = 8$ discrete levels. However, in this work, Xilinx Spartan 6 devices having 6-input LUTs which allow realization of fine and coarse PDLs are used. Therefore, use of one LUT enables implementation of a programmable delay inverter/buffer with five control inputs. The generic configurations of fine and coarse PDLs on the 6-input LUTs is depicted in Fig. 2.3. For the fine PDL, the first LUT input A_1 is the inverter input and the LUT inputs A_3 to A_6 are fixed to 0 whereas the only input that controls the delay is A_2 [99]. For the coarse PDL, the first LUT input A_1 is the inverter input and the rest of the LUT inputs (five) are controlled by $2^5 = 32$ discrete levels.

2.3 Design and Implementations

The optimized implementations of RO-PUF, A-PUF and RS-LPUF along with their performance assessments on Xilinx Spartan-6 FPGAs is presented in this section. This FPGA, developed in $45nm$ CMOS (complementary metal-oxide-semiconductor) technology, is specifically appropriate for embedded applications. The designs are developed using Xilinx ISE design suite 14.5 and coded in VerilogHDL whereas Matlab is utilized for communication between the PUF instances and the PC using UART interface.

2.3.1 RO-PUF

In this PUF, the response is derived from the difference in oscillator (RO) frequencies of selected pairs of ring-oscillators [96]. The earlier reported RO-PUF [96] implemented on Xilinx Spartan-3E device has each RO placed in four slices of one CLB. The RO-PUF, Fig. 4, on Spartan-6 devices incorporates PDLs and realizes 2 ROs inside a single CLB but each RO is placed in a single slice, each with a different color scheme, as shown in Fig. 2.5. Each RO is realized using 3 inverters and 1 AND gate as can be seen in Fig. 2.4 (black dashed lines). The design of inverters and AND gate use three LUTs and one LUT respectively. Improper routing and placement of ROs in FPGA introduces bias in the PUF response bits and affects the uniqueness of PUF responses [96]. Though our LUT placement constraints have fixed the internal routing path of each RO, the other parts are routed by Xilinx ISE automatically. Moreover, the delay LUTs contain no logic, 'keep' attributes is used in the design to stop logic optimization by the synthesis tool. In order to eliminate design-induced bias, hard macro technique has been used with the FPGA Editor in Xilinx ISE design tool, to place the ROs at selected locations in order to make all the ROs identical. In this design, 32 ROs are configured at the center of a chip in a 4×4 matrix of CLBs as shown in red in Fig. 2.6.

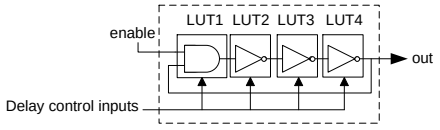

FIGURE 2.4: Configurable one RO cell

For the generation of programmable delays inside the 6-input LUT to achieve fine PDLs, one of the inputs to the LUT is used for ring connection and the other input is configurable. The rest of the LUT inputs are fixed to zero. For the coarse PDLs, one LUT-input is used for ring connection, another one is the challenge bit, while the remaining LUT inputs are configured with $2^4 = 16$ discrete levels (configuration from 0000 to 1111). In addition, one of the inputs to the LUT of AND gate is used for enabling the RO as evident in Fig. 2.7.

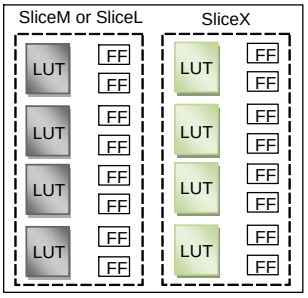

FIGURE 2.5: Implementation of 2 ROs per CLB.

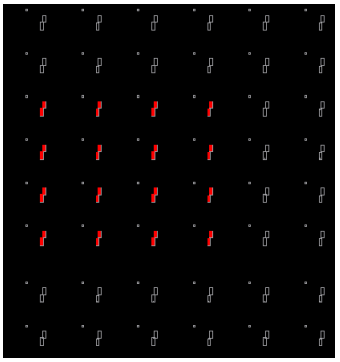

FIGURE 2.6: PUF array configuration of 32 ROs

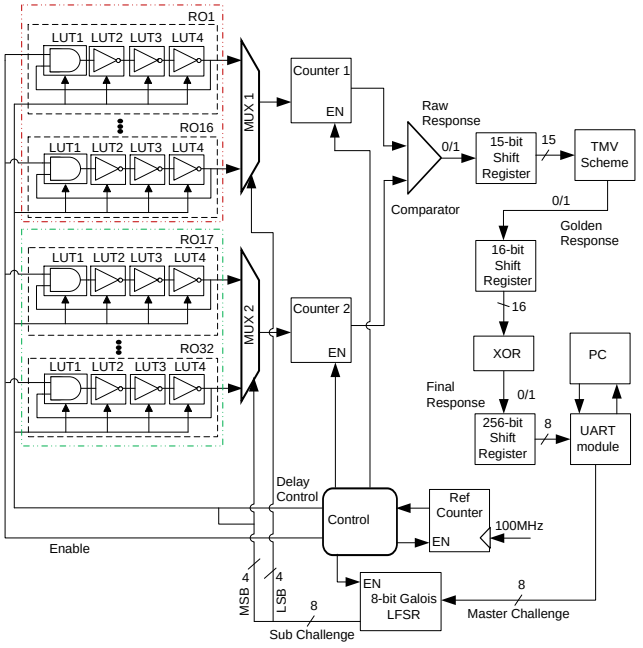

FIGURE 2.7: The proposed design of RO-PUF

Algorithm 1 describes the process of generating response bits by the proposed RO-PUF design. As shown in Fig. 2.7, implementation of the RO-PUF starts with the initiation of 8-bit master challenge from a user PC (Matlab) through a UART interface to the 8-bit Galois LFSR [41] (having maximum cycle length) to generate 256 subsequent challenges. Then two different ROs are chosen from each of these sub-challenges for comparison. In the sub-challenges, the 4 least significant bits (LSBs) select one of the 16 ROs in group 1, first 16 ROs marked with gray dashed lines, through the multiplexer 1 while the 4 most significant bits (MSBs) of the sub-challenge select one of the 16 ROs in group 2, the last 16 ROs marked with green dashed lines, through the multiplexer 2. The frequency of the

ALGORITHM 1: Pseudocode for response generation from the proposed RO-PUF design using coarse PDLs

```
Input: 8-bit master challenge
Output: 256-bit final response
/* Pseudocode:  Steps for Response bits generation                    */
Generate 256 sub-challenges from the 8-bit master challenge
/* applied to the delay control inputs for each sub-challenge         */
for sub-challenge = 1 to 256 do
    for Delay control inputs = 0 to 15 do
        /* each delay control inputs is applied 15 times              */
        apply each delay control inputs ← 0
        while apply each delay control inputs ¡ 15 times do
            if reference counter ¡ maximum value then
                Counter 1 ← frequency of selected one of the 16 ROs in group 1
                Counter 2 ← frequency of selected one of the 16 ROs in group 2
            end
            else if reference counter = maximum value then
                if Counter 1 ¿ Counter 2 then
                    Raw response bit 1
                end
                else
                    Raw response bit 0
                end
            end
            15-bit shift register ← Raw response bit
            apply each delay control inputs ← apply each delay control inputs + 1
        end
        /* TMV concept is applied on 15-bit shift register            */
        if more than half of the generated raw responses are 1s then
            Golden response bit 1
        end
        else
            Golden response bit 0
        end
        16-bit shift register ← Golden response bit
    end
    /* Final response generation                                      */
    Final response ← XORing the sixteen 1-bit golden responses
end
256-bit shift register ← final response bit
return 256−bit final response
```

selected ROs in group 1 and group 2 are then obtained and fed into the 32-bit respective counters. A crystal 100 MHz clock signal generated by an on-board oscillator drives the 8-bit reference counter. The counter 1, counter 2, and the reference counter start counting at the same time and are forced to stop when reference counter hits its maximum value. Then, the comparison of counter 1 and counter 2 values generates a response bit 0 or 1 for this RO pair depending on which counter had the higher value.

For the fine PDLs, 0 and 1 are applied to the delay control inputs for each sub-challenge. On the other hand, for the coarse PDLs, 2^4 (= 16) discrete levels to the delay control inputs are applied for each sub-challenge. Similar strategy is also adopted in RS-LPUF and A-PUF designs later in the work. In this work, we

employed concept of majority voting before the XOR operation in order to reduce the response instability and increase the attack complexity [157]. Each discrete level is applied to the delay control inputs 15 times and the generated raw responses are stored in a 15-bit shift register. The TMV concept is subsequently applied on them to determine the "golden response". The golden response is considered 1 if more than half of the generated raw responses are 1s, otherwise it is considered 0. These generated golden responses are stored in a 16-bit shift register (i.e., whereas for the fine PDLs, these generated golden responses are stored in a 2-bit shift register) and a 1-bit "final response" is generated by XORing the sixteen 1-bit golden responses (i.e., by XORing the two 1-bit golden responses for the fine PDLs). A total of 256 final response bits are generated for each 8-bit master challenge and these are stored in a 256-bit shift register. The final response bits are sent to the PC using a UART 8-bit interface for the PUF performance analysis. The controller circuitry is responsible to start and stop the ROs using the enable input, selection of the the ROs, count the oscillations, and return the responses based on their comparisons.

2.3.2 RS-LPUF

This generates each response using a metastable value of a latch composed of cross-coupled logic gates [164]. The recently reported RS-LPUFs [164], [53] also utilize Spartan-6 FPGA. In [164], PUF response determines the final state of the output patterns (all zeros, all ones, or a combination of zeros and ones) of each latch by applying consecutive rising edges. The design in [53] determines the PUF response using the exact number of oscillations at the output of each latch during the metastable state by applying rising-edge at a control signal. On the other hand, the proposed PUF response is derived from the difference in counting the numbers of 1's of the selected pairs of RS-LPUFs by applying consecutive rising edges. For the RS latch cell, Fig. 2.8, the output is 1 when input is 0 in a stable state. As the input of the RS latch changes from 0 to 1 (i.e., the rising edge), the

RS latch temporarily enters a metastable state. It then enters a stable state with either output 0 or 1.

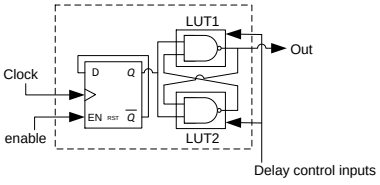

FIGURE 2.8: Configurable RS latch cell

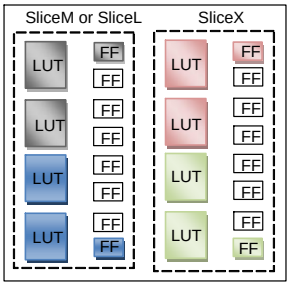

FIGURE 2.9: Implementation of 4 SR latches per CLB

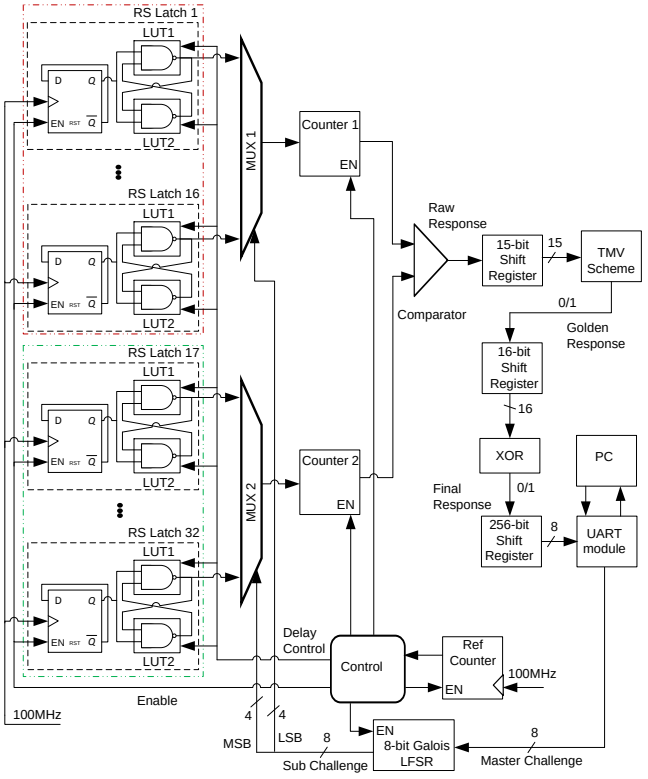

FIGURE 2.10: The proposed design of RS-LPUF

The FPGA implementation in [164] requires two NAND gates and one FF (for each SR-latch) in same kinds of slices on different CLBs whereas the implementation in [53] requires two NAND gates and two FFs (for each SR-latch) on different kinds of slices on the same CLB. In our work, two NAND gates and one FF (for each SR-latch) are implemented in a single slice on one CLB as shown in

Fig. 2.9. We use two LUTs and one flip-flop (FF) to create one SR-latch. The LUTs are used to create NAND gates while the FF in front of the two NAND gates, in Fig. 2.8, is used to reduce the clock skew. Furthermore, four latches are implemented inside a single CLB whereas two RS latches are implemented inside similar type of CLB slice using hard macro with proper placement, shown in Fig. 2.9, each with a different color scheme. In summary, our work uses only 32 RS latches for generating 256 response bits whereas [164] and [53] require 128 RS latches and 512 latches respectively.

Subsequently, we employ the PDL concept in the PUF design for improving the random response. For fine PDL, we use 2 inputs of each LUTs for connection i.e., the flip flop output and the NAND gate output. One of the LUT inputs is configurable bit while the rest of the LUT inputs are fixed to zero. In the implementation of coarse PDL, we again use 2 inputs of each LUTs for connection i.e., the flip flop output and the NAND gate output. However, in this case all the other remaining LUT inputs are configured with $2^4 = 16$ discrete levels (configuration from 0000 to 1111).

For the proposed design, shown in Fig. 2.10, 32 RS latches are configured at the center of FPGA in a 4×2 matrix of CLBs. A 100-MHz clock signal generated by an on-board oscillator is applied to each FF, which divides it into a 50-MHz clock signal that is applied to the corresponding RS latches. The D-type FF acts as frequency divider by feeding back the output from Q to the input terminal D, the output pulses at Q have a frequency that are exactly one half that of the input clock frequency as can be seen in Fig. 2.10. The RS latches are stopped using the enable signal. The flip-flop (FF) is reset before application of the enable signal to ensure that the latches always start with the same initial state.

Once again, the generation of 256 challenges and selection of 2 SR latches are done in 2 groups based on the subsequent challenge inputs. In every clock cycle, the respective outputs of selected RS latches in group 1 and group 2 are fed into the 8-bit counters 1 and 2. Then both counters are incremented every time the selected RS latches output 1's. Counters 1 and 2, and the 8-bit reference counter starts

counting simultaneously and the counting is terminated as soon as the reference counter hits its maximum value. At that instance, values in counter 1 and counter 2 are compared to generate a response bit 0 or 1, depending on which counter had the higher value for this RS latch pair. Subsequently, the final responses are generated by employing the PDLs and TMV scheme and stored in 256-bit shift register before performance analysis.

2.3.3 A-PUF

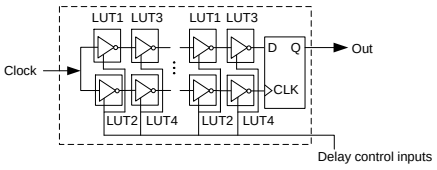

FIGURE 2.11: Configurable one PUFI cell

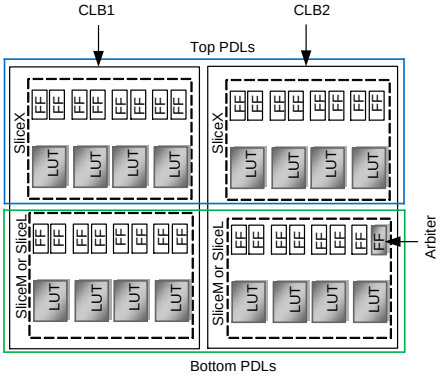

FIGURE 2.12: Implementation of one PUFI on the 2 CLBs.

It is composed of two identically configured delay paths stimulated by an activating signal. The proposed RO-PUF design concept is used for the development of PDL based A-PUF on Spartan 6 FPGA. It is significantly different from earlier reported PDL based A-PUF implemented on different set of FPGA (i.e., Virtex 5) [98]. In our design, the PUF response is derived from the difference in counted 1's between the selected pairs of PUF instances (PUFI), shown in Fig. 2.11, from 32 PUFI by applying rising clock edges. We have implemented these PUFIs in 32×2 matrix of FPGA CLBs, in Fig. 2.12, in such a way that each PUFI consists of 8 stages of inverter and one arbiter which connects the last stage of the two paths shown in Fig. 2.11. Each PUFI uses 16 LUTs for inverters and one FF for arbiter and are implemented with 2 CLBs. The PDLs are implemented by 2 LUTs each acting as inverter and the 2 LUTs are implemented in different slices on the same CLB, Fig. 2.12, marked by identical but independent paths in blue and green boxes.

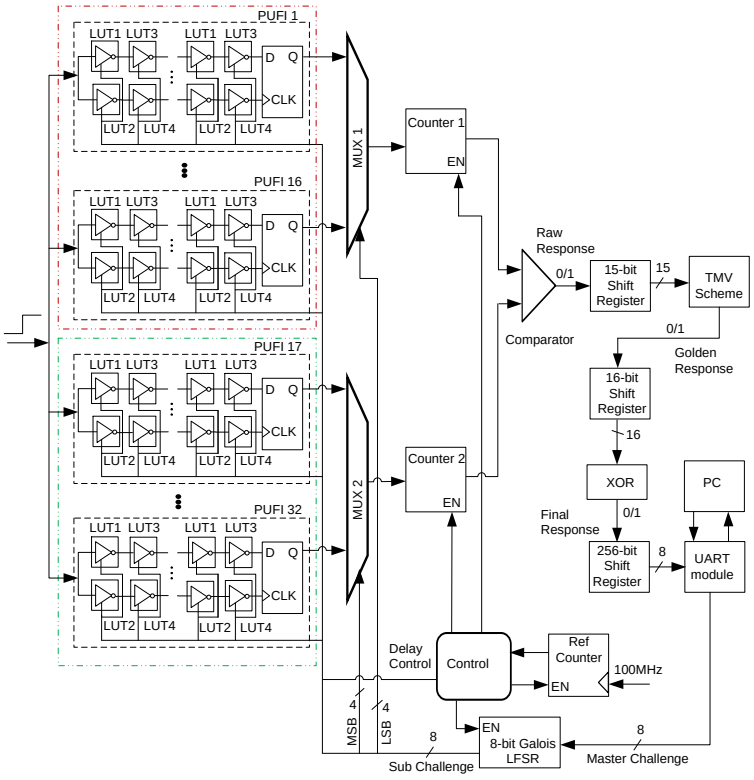

FIGURE 2.13: The proposed design of A-PUF

The slices are placed using hard macro with predefined location and connections to eliminate design bias between each PUFI paths. For fine PDL, one of the LUT inputs is used for connection, one for the configurable bit, and the rest of the input bits are fixed to zero. For the coarse PDL, one LUT input is used for the connection, one is the challenge bit, while the remaining LUT inputs are configured with $2^4 = 16$ discrete levels (configuration from 0000 to 1111).

In the proposed design, in Fig. 2.13, the generation of 256 subsequent challenges and selection of 2 PUFIs are done in 2 groups based on the subsequent challenge inputs similar to the proposed RO-PUF design. Each generated challenge is applied to the configured delay paths of the PUFIs. A 100-MHz clock signal generated by an on-board oscillator is also applied to the PUFIs. In every clock cycle, the outputs of a selected PUFI in group 1 and group 2 are fed into the 8-bit counters 1 and 2 respectively. Then values of both the counters are incremented every time the output of the selected PUFI is 1. The counters 1 and 2, and the 8-bit reference counter start counting simultaneously and the counting gets terminated

no sooner the reference counter hits its maximum value. Next, a response bit 0 or 1 for this PUFIs pair is generated by comparing the values of counters 1 and 2. If counter 1 has the higher value then the response bit is set to 1 else to 0. For the generation of final response, the fine and coarse PDLs in conjunction with the TMV scheme is applied. Eventually, the generated 256 final responses are stored in a 256-bit shift register and then sent to PC for PUF response analysis.

2.4 Security Analysis

Silicon PUFs have received a lot of attention and they have been adopted by industry for many hardware-oriented cryptography applications [48]. However, several attacks have been reported that break the security guarantees of PUFs successfully. A general description of attacks and countermeasures against PUF designs was already presented in Chapter 1 (see Section 1.1.1.3). Possible attack vectors against our proposed PUF designs are given in Table 2.1.

2.4.1 Machine Learning (ML) based Modeling Attacks

ML-based modeling attacks are the powerful attack for strong PUFs. In this work, we employed few mechanisms in order to increase the ML attack such as generating internal challenges from the external challenges, concept of majority voting before the XOR operation [157], discrete PDL configurations, and XORing multiple individual responses to form a single response bit [55]. However, the ML-based modeling attacks need a huge amount of PUF CRPs during the learning phase. Therefore, this attack will not be effective to weak PUFs such as the SRAM PUF, our three proposed PUF designs and similar architectures (i.e., since our proposed A-PUF uses limited number number of CRPs following our RO-PUF design philosophy). This is primarily due to the fact that there is no external access to the response for an attacker and thus she does not have very large CRP space.

2.4.2 Reverse Engineering (RE) Attack

The proposed PUF architectures also provide reasonable security against reverse engineering attack. For example, in our proposed RO-PUF architecture, an attacker may try to gain knowledge of the RO frequencies or raw responses or golden responses to construct final responses. At the input, even if an adversary knows the value of the external challenge, the corresponding internal challenges, raw/golden responses with the PDL configuration, and XORing golden responses are calculated within the FPGA chip. Since they never come out of the chip, it is difficult to get the internal challenges or PDL configuration values or raw/golden responses. Since these values are not accessible, reverse engineering of the original RO frequencies from raw/golden responses values becomes very hard.

TABLE 2.1: Attack levels of the proposed PUF designs against several attacks

Design	Modeling Attacks	RE Attack	Invasive Attacks	SCA
Our RO-PUF	Not effective	Very hard	Hard	Medium (need further investigation)
Our RS-LPUF	Not effective	Very hard	Hard	Medium (need further investigation)
Our A-PUF	Not effective	Very hard	Hard	Medium (need further investigation)

2.4.3 Invasive Attacks

An attackers can open up the package of the secure processor and attempt to read out the secret when the processor is running or attempt to measure the PUF delays when the processor is powered off [150]. Probing the delays with sufficient precision (the resolution of the latches, LUTs) is extremely difficult. Furthermore, interaction between the probe and the circuit may affect the delay. Damage to the layers surrounding the PUF delay paths should alter their delay characteristics (disturb the underlying nano-scale structure) changing the PUF outputs, and destroying the secret [45, 142]. It is possible to prevent invasive

attacks on our proposed architectures by enclosing the control logic by delay wires of the PUFs [46]. These wires normally introduce path delays that the PUF circuit uses to determine its response. Therefore, if invasive attacks attempt to probe the control logic then the PUF secret will be altered and damaged or destroyed.

2.4.4 Side-Channel Attacks (SCA)

SCA statistically analyses the execution time, power consumption or electromagnetic radiation of the PUF devices to gain knowledge about intermediate secrets. There is a possibility of SCA such as power consumption or electromagnetic radiation on our designs because our proposed designs use counters, comparator, XOR operation, and intermediate storage registers. Though we did not consider SCA in this work, we consider it as an important security issue and a part of future work.

2.5 Performance Analysis and Discussion

The number of FPGA testbeds used for performance evaluation of PUFs vary significantly in the related literature. These numbers range from five to ten [48, 130, 162, 170], above ten to fifty [100, 143, 171], and above hundreds [56, 96, 156]. The performance evaluation in terms of uniqueness, uniformity, bit-aliasing, and reliability for the three proposed PUF designs have been carried out through implementations on 10 Spartan-6 (XC6SLX45) FPGAs.

2.5.1 Uniqueness (UQ)

It is measured by calculating the inter-chip Hamming distance (HD) between different PUF devices using (1.1). To investigate uniqueness in the generated response of the three proposed PUF designs, $k = 10$ (10 FPGAs) and $n = 256$ (response bit length) is used. This provides a total of 10 responses from 10

FPGAs, one response per FPGA, at core supply voltage of 1.2V and standard temperature of 25°C for each proposed PUF designs. To confirm the effectiveness of XORing responses while varying the PDL inputs, we developed and evaluated the proposed designs using the concept of fine (utilize only 1-bit control LUT input) and coarse PDLs (utilize all the configurations to LUT control inputs). The histogram of normalized inter-chip HD between two arbitrary responses among the 10 responses, i.e., $\binom{10}{2} = 45$ combinations, of proposed designs using fine and coarse PDL concepts are shown in Figs. 2.14 and 2.15 respectively. The horizontal axis represents the percentage HD and the vertical axis represents the number of occurrences of a specific HD between any two PUF responses. The best fit ideal binomial curves to the histogram diagram of the three proposed PUF designs are also plotted in Figs. 2.14 and 2.15. The mean (μ) and the standard deviation (σ) of the proposed RO-PUF, RS-LPUF and A-PUF implementations using both the fine and the coarse PDLs are given Table in 2.2. For a 256-bit response, the ideal (50%) average HD is 128 bits ($\mu = 50\%$) and the expected standard deviation is 8 bits ($\sigma = 3.125\%$).

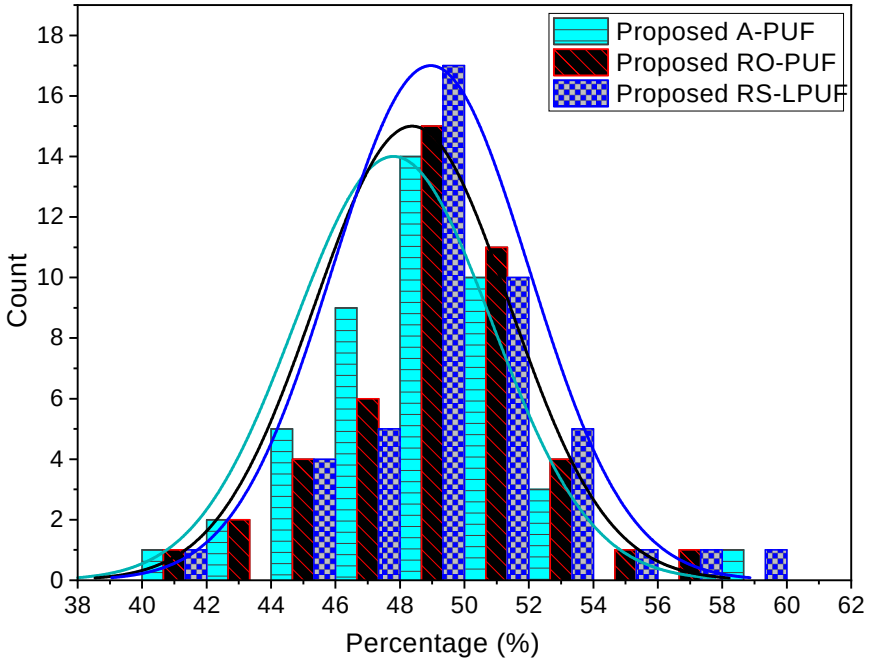

FIGURE 2.14: Uniqueness: Inter-chip HD distribution (Fine PDL) from the 256-bit response.

95% confidence interval (CI): We then estimated the interval for the uniqueness metric for 95% CI proposed in [58]. The CI estimates an interval within which

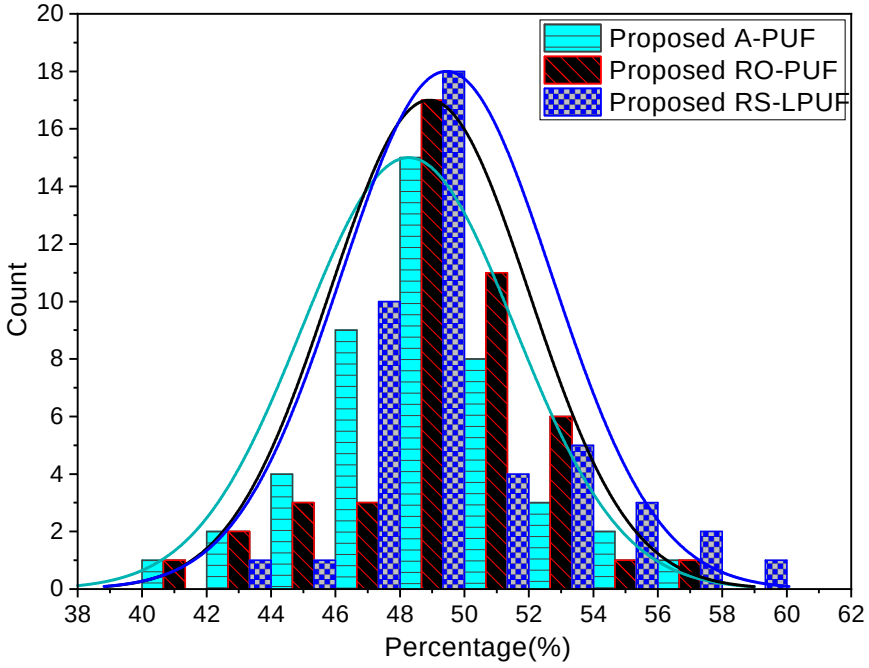

FIGURE 2.15: Uniqueness: Inter-chip HD distribution (Coarse PDL) from the 256-bit response.

TABLE 2.2: Mean (μ), Standard deviation (σ), SEM and 95% Confidence Interval (CI) of μ for the three proposed designs uniqueness (UQ) metric

Design	Metric	PDL	μ (%)	σ (%)	SEM (%)	95% CI of Mean [Lower, Upper]
Our RO-PUF	UQ	Fine	48.38	3.02	0.45	[47.48, 49.29]
		Coarse	48.91	3.10	0.46	[47.97, 49.86]
Our RS-LPUF	UQ	Fine	48.96	3.04	0.45	[48.02, 49.89]
		Coarse	49.44	3.27	0.49	[48.44, 50.44]
Our A-PUF	UQ	Fine	47.80	3.06	0.46	[46.86, 48.74]
		Coarse	48.27	3.30	0.49	[47.25, 49.27]

the estimated value of a population lies with some confidence. If μ and σ are the respective mean and standard deviation of the uniqueness then the lower value for 95% confidence interval is $\mu - t_{0.025;(N-1)} \times (\frac{\sigma}{\sqrt{N}})$ and the upper value is $\mu + t_{0.025;(N-1)} \times (\frac{\sigma}{\sqrt{N}})$, where $t_{0.025;(N-1)}$ denotes the 2.5 percentile in the t-distribution with $N-1$ degrees of freedom [58]. Here, $N = 45$, the total number of response combinations, $(\frac{\sigma}{\sqrt{N}})$ is standard error of the mean (SEM) and $t_{0.025;(N-1)} = 2.0154$. The SEM and 95% CI of μ for the proposed implementations using the fine and coarse PDLs are given in Table 2.2.

It can be seen that the proposed designs possess enhanced uniqueness (Table 2.2). This has been achieved by incorporation of fine and coarse PDLs, XORing responses from the PDL configurations, and the placement strategy. The XORing and placement strategies are used to eliminate the influence of biased responses. In particular, the process (internal) variation and the propagation delay of PUF cells are the major deciding factors in the resulting count. The internal variations of FPGA LUTs of each PUF cell can be generated from changes in the propagation delay of the LUTs under different inputs. Due to the internal variations of FPGA LUTs, we can generate different outputs/responses under varying inputs for each sub-challenge. To further improve uniqueness of the PUF, we combined (XORed) these outputs and generated a 1-bit final response for each sub-challenge. A total of 256 final response bits are generated for each 8-bit master challenge (256 sub-challenges) by applying the same procedure. The uniqueness of the designs incorporating coarse PDLs was observed to be marginally higher when compared to the designs using fine PDLs. This is due to the fact that coarse PDLs utilize all the configurations (i.e. one LUT input is the challenge bit while the remaining LUT inputs are configured with 16 discrete levels) to LUT control input lines for the determination of the final response obtained by XORing the golden responses from varying the PDL inputs. On the other hand, fine PDLs utilize only 1-bit control. Thus, the effect of XORing responses with varying the PDL inputs is clearly identified from the difference of uniqueness results of the designs incorporating the fine and coarse PDLs. However, in order to judge the effect of varying the PDL inputs, we have also investigated the effect of varying the PDL inputs with/without TMV. For example, the uniqueness results of PUFs incorporating the coarse PDLs with/without TMV are tabulated in Table 2.3. Apparently, the uniqueness results are not impacted in any of the designs with/without TMV and thus it can be concluded that the PUFs with coarse PDLs perform marginally better in eliminating the bias of the response bits and possess better uniqueness.

TABLE 2.3: The experimental results of the proposed designs with/without TMV

Design		Only PDL (without TMV) (%)	both PDL and TMV (Including TMV) (%)
Our RO-PUF	UQ	48.91	48.91
	UF	49.55	49.62
	BA	49.55	49.62
	RE	97.91	99.39
Our RS-LPUF	UQ	49.44	49.44
	UF	50.91	50.83
	BA	50.91	50.83
	RE	98.43	99.53
Our A-PUF	UQ	48.27	48.27
	UF	51.09	51.05
	BA	51.09	51.05
	RE	98.52	99.50

2.5.1.1 Entropy Estimation:

A number of methods have been proposed for estimation of entropy of a PUF. The context-tree weighting (CTW) algorithm is employed to estimate the upper bound of entropy (i.e. best case) [59, 151]. Min-entropy is another metric widely employed to evaluate the lower bound of entropy (i.e. worst case) [58, 151]. In this work, we make use of the lossless compression algorithm (CTW) [59] to find out whether PUF responses can be compressed. If compression is possible, the PUF responses do not have full entropy. In order to perform the CTW compression test, we first concatenate all the bit strings (10 responses) into a single bit string of length $10 \times 256 = 2560$ bits. Compressing this string with CTW gives results in Table 2.4. Clearly, very little compression is achieved by CTW and this indicates that the proposed PUFs have good compression resistance. Besides the compression factor, it is also possible to estimate the min-entropy of PUFs. It is the worst-case (i.e. lower bound of the unpredictability of the response) measure of uncertainty for a random variable.

We use the method described in NIST specification 800-90 [10] for evaluating the min-entropy of a binary source. The n-bit responses of k devices have an

TABLE 2.4: CTW Ratio and Min-entropy Results

Design	Devices	PDL	Original size (bits)	Comp ressed size (bits)	CTW Ratio	Min-entropy per bit
Our RO-PUF	10	Fine	2560	2512	98.13%	0.691
		Coarse	2560	2518	98.36%	0.698
Our RS-LPUF	10	Fine	2560	2532	98.91%	0.706
		Coarse	2560	2541	99.26%	0.711
Our A-PUF	10	Fine	2560	2495	97.46%	0.681
		Coarse	2560	2507	97.93%	0.687

occurrence probability at each bit of p_1 and p_0 for the values of 1 and 0 respectively. The p_1 and p_0 are calculated by $\frac{HW_i}{k}$ and $1 - \frac{HW_i}{k}$ respectively, where HW_i is counting the number of $1's$ in k devices. When $p_{i\,max}$ is the maximum value of these two probabilities (i.e., $p_{i\,max} = \max(p_1, p_0)$), the definition for min-entropy of each individual bit (binary source) is given by (2.1). Here, $p_{i\,max}$ is given by (2.2).

$$H_{min,i} = -\log_2(p_{i\,max}). \tag{2.1}$$

$$p_{i\,max} = \begin{cases} \frac{HW_i}{k} & \text{if } HW_i > \frac{k}{2} \\ 1 - \frac{HW_i}{k} & \text{otherwise} \end{cases} \tag{2.2}$$

The total min-entropy of the design is given by (2.3) and is calculated by averaging the estimated min-entropy of each bit.

$$(H_{min})_{average} = \frac{1}{n}\sum_{i=1}^{n} H_{min,i} \tag{2.3}$$

We estimate the average min-entropy per bit of the proposed PUF implementations using (2.1) and (2.3) for 10 FPGAs. The obtained results in Table 2.4 clearly conveys that the proposed PUF designs with incorporated fine and coarse PDLs possess sufficient entropy which are close to 0.70. The values found by this test for the proposed PUFs are conservative estimates as it will likely increase with

increasing number of FPGAs. It can be concluded that the entropy per bit for the proposed PUFs lie between 0.68 (from min-entropy) and 0.99 (based on the compression test). To get meaningful entropy value of PUF designs, a very large scale analysis with many boards is needed. Even though the largest conducted experiment consisted of more than 100 boards [56, 96, 156], it is not really enough to accurately determine the entropy. In general, how to determine the exact entropy of the proposed PUF responses is another very important open research problem.

It is now apparent that the three proposed PUF implementations are able to generate sound uniqueness, close to 50%, of signatures. Moreover, the spread of the 95% CI of μ is at most $\pm 2\%$ and the entropy possess sufficient amount of randomness. These results indicate that the proposed implementations generate highly unique responses and this makes them excellent candidates for unique identification and key generation applications.

2.5.2 Uniformity (UF)

It estimates how uniform the proportion of $0's$ and $1's$ are in the PUF response and is calculated using (1.5). It is important for security perspective as it prevents an attacker from guessing if response of a particular device is biased towards any particular value. For PUF uniformity, the experiments have been performed on ten Spartan-6 FPGAs i.e.one response per FPGA. It can be inferred from the obtained results in Fig. 2.16 that the proposed implementations incorporating fine and coarse PDLs have good uniformity values (close to 50%). Moreover, the PUFs with the coarse PDLs possess better uniformity when compared to the designs utilizing fine PDLs. Once again this can be attributed to the use of all the configurations to the LUT control input lines and XORing more single golden responses to determine the final responses. This eliminates the influence of the biased responses and produces better uniformity.

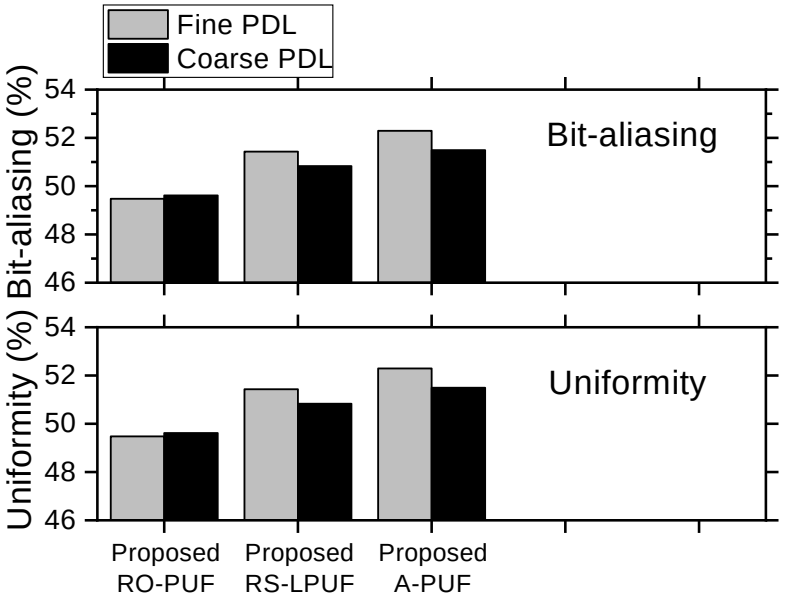

FIGURE 2.16: Uniformity and Bit-aliasing of the proposed designs using the fine and coarse PDLs

2.5.3 Bit-aliasing (BA)

It may give us information about any systematic spatial effect across devices. The presence of bit aliasing leads to the situation when different chips will produce similar responses. As a consequence, it becomes relatively easier for an attacker to guess the response. An effective PUF response generator should not exhibit bit-aliasing when implemented on different FPGAs. The bit-aliasing calculated using (1.6) is measured by estimating the bias of a particular response bit across several chips. The proposed designs with the fine and coarse PDLs schemes is now implemented on ten ($k = 10$) Spartan-6 FPGAs to find out bit-aliasing. The obtained results in Fig. 2.16 clearly shows that the proposed designs with fine and coarse PDLs exhibit good bit-aliasing values which are close to the ideal value of 50%. This can be attributed to the incorporation of hard macros (i.e., the placement strategy) during the design of PUFs.

2.5.4 Correlation Between Bits

If the bits are highly correlated then an attacker might be able to predict response to an unknown challenge from a set of known challenge-response pairs. The auto-correlation test can be used to detect correlation between bits of a response. Systematic aspects to process variation may show up as significant correlation at particular intervals. The responses are extracted from a common fabric and hence it is possible for spatial correlation to appear. To check the presence of correlation in the test chip, the auto-correlation function R_{xx} is used. Here, x is the n-bit response being observed, and R_{xx} is evaluated at lag j. This value tends towards 0.5 for uncorrelated bit-strings and towards 0 or 1 for correlated bit-strings.

$$R_{xx}(j) = \frac{1}{n}\sum_{i=1}^{n} x_i \oplus x_{i-j} \tag{2.4}$$

The minimum R_{xx} among ten different 256-bit responses ($k = 10$ FPGAs) for the proposed RO-PUF, RS-LPUF and A-PUF implementations with fine PDLs are 0.42, 0.43 and 0.41 while the maximum R_{xx} are 0.59, 0.58 and 0.59. The corresponding minimum R_{xx} with the coarse PDLs are 0.42, 0.44 and 0.43 while the maximum R_{xx} are 0.58, 0.57 and 0.58. The average R_{xx} of the proposed RO-PUF, RS-LPUF, A-PUF implementations using fine PDLs are 0.493%, 0.505% and 0.496%. The corresponding numbers utilizing coarse PDLs are 0.505%, 0.504% and 0.494%. This indicates that the proposed PUFs are resistant to correlation analysis (close to 0.5) and hence no spatial correlation exists within PUF responses.

2.5.5 Reliability (RE)

The PUF responses are affected by external factors such as temperature variation, supply voltage fluctuation, and thermal noise and these lead to issues in their reproducibility. Reliability measures the temporal stability of PUF responses generated by the same master challenge in different environments and is calculated using (1.3) for one chip whereas (1.4) is used for the calculation of the average

reliability of k chips. The three proposed PUF designs with fine and coarse PDLs schemes were implemented on ten ($k = 10$) Spartan-6 FPGAs for temperature variation from $0°C$ to $85°C$ (the rated temperature range of the Spartan-6 commercial-grade FPGA) using a Southern Scientific EC Environmental Chamber [60] to control the temperature while the core supply voltage is varied within $1.14 - 1.26$V (the rated voltage range of Spartan-6 FPGAs) to find the reliability in responses. The reference response of 256 bits (for each chip) was generated at the standard temperature of $25°C$ and standard supply voltage of 1.2V. Afterwards, other responses were generated for analysis under varying operating conditions.

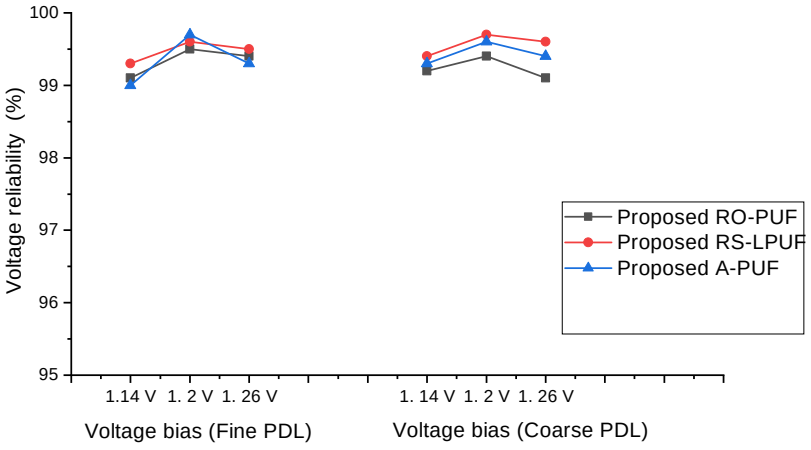

FIGURE 2.17: Reliability (single FPGA) with respect to supply voltage bias at $25°C$ (Fine PDL and coarse PDL)

TABLE 2.5: Mean (μ), Standard deviation (σ), SEM and 95% Confidence Interval (CI) of μ for the three proposed designs reliability (RE) metric using the fine and coarse PDLs

Design	Metric	PDL	μ (%)	σ (%)	SEM (%)	95% CI of Mean [Lower, Upper]
Our RO-PUF	RE	Fine	99.44	0.28	0.03	[99.36, 99.51]
		Coarse	99.39	0.38	0.05	[99.28, 99.48]
Our RS-LPUF	RE	Fine	99.46	0.27	0.03	[99.38, 99.53]
		Coarse	99.53	0.32	0.04	[99.44, 99.61]
Our A-PUF	RE	Fine	99.55	0.26	0.03	[99.47, 99.62]
		Coarse	99.50	0.25	0.03	[99.43, 99.56]

Fig. 2.17 shows reliability results of our PUF designs for one FPGA obtained with respect to supply voltage bias at $25°C$. Fig. 2.18 shows the average reliability

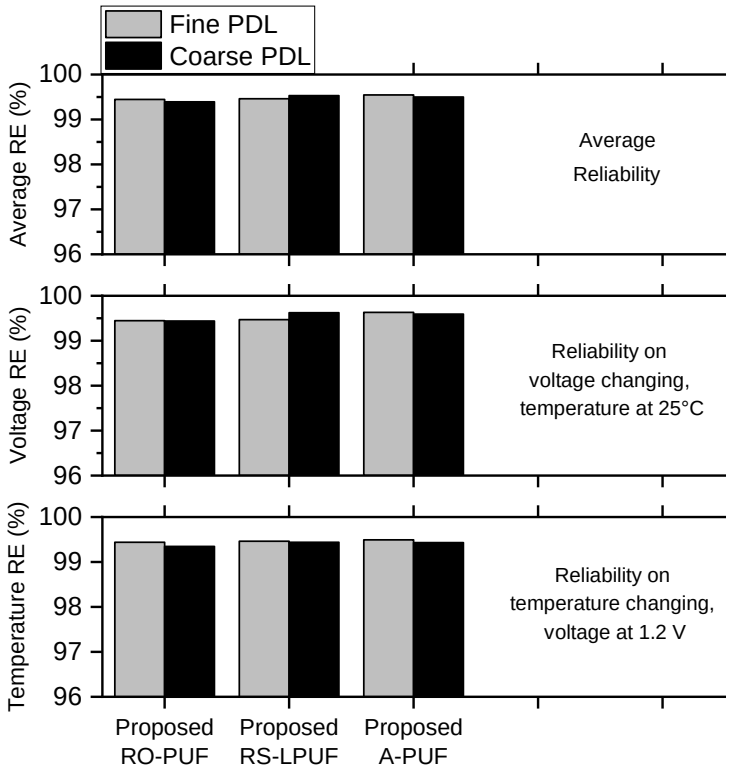

FIGURE 2.18: Temperature RE, Voltage RE, Average RE of the proposed designs using the fine and coarse PDLs

results of our PUF designs for 10 FPGAs with fine and coarse PDLs. The mean reliability on temperature and voltage variations for 10 FPGAs of the proposed designs with fine and coarse PDLs are deduced from these plots and tabulated in Table 2.5. It also contains the standard deviation (σ), SEM, and 95% CI of μ for the proposed designs with fine and coarse PDLs.

Furthermore, Table 2.3 also shows that the reliability result of the proposed designs improve significantly using a TMV scheme. The TMV technique is to extract a statistical bias in the presence of noise for PUF responses [7, 102]. The basic principle is to repetitively test (say, 15 times) the PUF using the same challenge and then take the majority value of the responses as the final output. The multiple evaluation of the PUF's response would give the true statistical bias of the PUF response (bias towards 0 or 1). It is also clear from the results in Table 2.5 that the reliability of the proposed designs with fine and coarse PDLs are very close to the theoretical maximum of 100% and the spread of the 95% CI is at most $\pm 0.5\%$. These results imply that the proposed PUFs are not affected across

the range of temperatures and voltages, and compare favourably to the reference temperature of $25°C$ at supply voltage of 1.2V. Extra error-correction circuitry [55] can be employed to correct the small errors when desired. The details are given in subsection 2.6.1.

2.5.6 Aging Effect

Aging is an extremely important issue for silicon devices that must remain in operation for decades. Over the course of the lifetime of a PUF, its stability degrades due to the transistor threshold voltage shift induced by phenomena such as hot carrier injection, electromigration and negative-bias temperature instability (NBTI) [93]. To consider the aging effect, accelerated aging tests [93] were performed by raising the temperature to $85°C$ and the supply voltage to 1.26V, i.e. +5% variation in the nominal voltage, for one week. The PUF outputs were recorded every 3 hours using a Southern Scientific EC Environmental Chamber [60]. This is sufficient to induce more than 5 years of aging [93]. These responses were then compared with the reference ones generated at $25°C$ and 1.2V. The Intra-chip HD (1.2) can be used to calculate the bit errors. The bit error rate (BER) of the proposed RO-PUF, RS-LPUF and A-PUF implementations using fine PDLs are 3.9%, 3.4% and 3.1%. The corresponding numbers with coarse PDLs are 3.5%, 3.9% and 3.4%. However, our results show that the proposed PUF designs are still usable as the BER is below 4% during the test and this can be corrected using error correcting codes [55]. The details are given in subsection 2.6.1.

2.5.7 Performance Comparisons with Previous PUFs

In order to assess the performance, the proposed architectures are implemented on Xilinx Spartan-6 FPGAs and the obtained results are compared with the existing state-of-the-art in terms of uniqueness, uniformity, bit-aliasing and reliability in Table 2.6. Following key observations can be deduced from these tabulated results:

TABLE 2.6: Performance Comparisons with Previous FPGA based PUFs

Design		Uniqueness	Reliability	Uniformity	Bit-alias	Target FPGA
	Ideal Value	50%	100%	50%	50%	
RO-PUF	In 2011 [94]	47.24	99.14	50.56	50.56	Spartan-3E (XC3S100E)
	In 2012 [170]	47.00	—	—	—	
	In 2011 [96]	44.10	99.00	—	—	
	In 2018 [171]	49.33	95.45	49.5	—	Virtex-5
	In 2018 [100]	55.00	—	—	—	Spartan-6 (XC6SLX16)
	In 2016 [30]	49.97	98.41	—	—	
	In 2019 [86]	48.76	97.72	—	—	
	In 2019 [23]	49.83	99.35	—	—	Artix-7 XC7A35T
	In 2019 [165]	—	99.33	50.05	—	Kintex-7
Our RO-PUF (Section 2.3.1)	Fine PDL	48.38	99.44	49.48	49.48	Spartan-6 (XC6SLX45)
	Coarse PDL	48.91	99.39	49.62	49.62	
SRAM PUF	In 2008 [50]	49.97	88.00	—	—	—
PUF ID	In 2008 [48]	45.60	99.42	—	—	Spartan-6
RS-LPUF	In 2018 [6]	49.32	98.80	44.65	44.65	Spartan-3
	In 2013 [164]	49.00	99.14	—	—	Spartan-6 (XC6SLX45)
	In 2015 [53]	49.24	98.87	—	—	
Our RS-LPUF (Section 2.3.2)	Fine PDL	48.96	99.46	51.43	51.43	
	Coarse PDL	49.44	99.53	50.83	50.83	
A-PUF	In 2011 [94]	18.37	99.76	55.69	19.57	Virtex-5 (XC5VLX30)
	In 2015 [130]	45.25	95.93	48.10	—	Spartan-3 (XC3S400)
Our A-PUF (Section 2.3.3)	Fine PDL	47.80	99.55	52.30	52.30	Spartan-6 (XC6SLX45)
	Coarse PDL	48.27	99.50	51.05	51.05	

- The proposed RO-PUF design with fine and coarse PDLs outperforms existing RO-PUF designs [86, 94, 96, 100, 170] in terms of uniqueness. However, the obtained uniqueness are slightly inferior when compared to other RO-PUF designs [23, 30, 171]. Moreover, the proposed RO-PUF design using fine and coarse PDLs outperform all the previous RO-PUF designs in terms of reliability. Additionally, the proposed RO-PUF design using coarse PDL improves the bit-aliasing and uniformity metrics when compared to similar other designs [94, 171].

- The results also convey that the proposed RS-LPUF design using coarse PDLs exhibits better uniqueness when compared to existing RS-LPUF designs [6, 53, 164] and memory based PUF (PUF ID) [48] but performs slightly poorly when compared to SRAM PUF [50]. The design with fine PDLs leads to enhanced uniqueness when compared to memory based PUF (PUF ID) [48] but poorly compares to SRAM PUF [50] and the existing RS-LPUF designs [6, 53, 164]. Moreover, the proposed RS-LPUF design

improves the bit-aliasing and uniformity metrics when compared to other RS-LPUF design [6]. Furthermore, the proposed RS-LPUF design using fine and coarse PDLs outperform all the previous RS-LPUF designs in terms of reliability.

- The proposed A-PUF design using fine and coarse PDLs exhibits substantially improved uniqueness, uniformity, and bit-aliasing whereas the reliability is very close to the ideal value and is comparable to the other A-PUF designs [94, 130].

In summary, the statistical analysis of measured data demonstrates significantly better performance of the proposed designs in terms of uniqueness, reliability, uniformity, and bit-aliasing. Therefore, our proposed PUFs designs can be used for device ID generation and key generation. However, using a PUF itself may not be suitable for chip authentication scenario because our three proposed PUF designs are not having very large CRP space. On the other hand, chip authentication can be achieved by using a cryptographic function, and a secret key which can be generated from the proposed PUF designs (having a limited number of CRPs).

2.6 Hardware overhead analysis

The proposed architectures implemented on Xilinx Spartan-6 FPGAs are compared with the existing state-of-the-art in Table 2.7. It is apparent that the proposed RO-PUF, RS-LPUF, and A-PUF designs are area efficient when compared to the existing RO-PUF [51, 91], RS-LPUF [53, 164] and A-PUF [49, 70] respectively on similar Xilinx Spartan-6 FPGAs, and Zynq-7000 FPGA (it has 4 LUTs, 8 registers and 3 MUXes in each slice, the same as the Spartan-6 FPGA used in this work). However, the proposed implementations with/without TMV have poor throughput compared with the conventional RO-PUF [51, 91] but is still better than the RS-LPUF [53]. Moreover, our three proposed PUF implementations outperform all the previous works in terms of the resources consumed per response bit. In summary, it can be deduced that the proposed architectures provide very competitive area-throughput trade-offs.

TABLE 2.7: Comparison of FPGA implementation results for the our RO-PUF, RS-LPUF and A-PUF with State-of-the-Art

PUF Design		Area (Total Slices)†	Process time (msec)	Res. bit length	Slices / Res. bit	Target FPGA Device
RO-PUF	In 2018 [51]	849	1.68	128	6.6	Zynq-7000 (XC7Z045)
	In 2012 [91]	952	4.59	49	19.4	Spartan-6 (XC6SLX45)
Our RO-PUF (Section 2.3.1)	WOTMV	93	10.5*	256	0.36	
	WTMV	107	158.3‡	256	0.42	
RS-LPUF	In 2015 [53]	324	5120	256	1.27	Spartan-6 (XC6SLX45)
	In 2013 [164]	256	—	256	1	
Our RS-LPUF (Section 2.3.2)	WOTMV	71	10.5*	256	0.28	
	WTMV	86	158.3‡	256	0.34	
	In 2016 [49]	8192	—	256	32	Spartan-6 (XC6SLX45)
	In 2011 [70]	4288	—	64	67	
Our A-PUF (Section 2.3.3)	WOTMV	194	10.5*	256	0.76	
	WTMV	209	158.3‡	256	0.82	

† The total slices for the PUF with control logic (without UART).
* The total number of clock cycles (at 100 MHz) required to generate a response without TMV = total sub challenges × (Ref.counter counts × delay evaluations) + control logic.
‡ Number of clock cycles required to generate a response with TMV = total sub challenges × (Ref.counter counts × (delay eval. × TMV)) + control logic.

2.6.1 Error Correction Codes (ECC)

In general, the construction of error correction code in hardware is very expensive in terms of area. To reduce the area costs, the TMV post-processing scheme is already applied before error correction [7, 102] in the proposed designs. However, removal of the noise (error) from PUF response in the field is very important because in a encryption/decryption algorithm, even the slightest change to the secret key will change the cyphertext dramatically. This means that in order to use a PUF for secret key generation, the PUF's CRPs need to be consistent during temperature variation, supply voltage fluctuation, thermal noise and aging effect.

For error correction, we propose a novel method which is inspired by the design described in [53]. This method is based on the use of Bose, Chaudhuri, and Hocquenghem (BCH) code. The BCH code is a popular error correcting code that is widely used for error correction in PUF response [51, 53, 55, 68, 91, 125, 138, 142, 153]. We have implemented a BCH code with the following parameters: BCH (n=255, k=139, t=15) code. In this format, n=255 is the output block size, k=139 is the input block size (in our case, 139 bits which is randomly chosen from

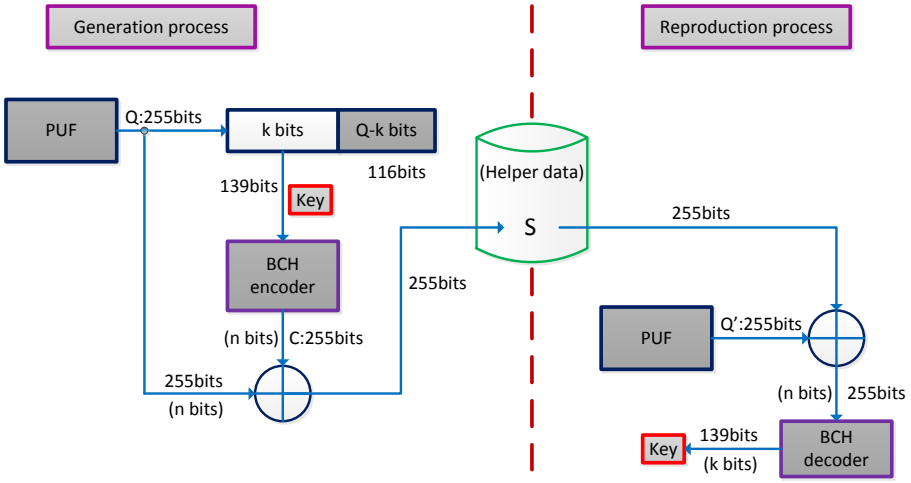

FIGURE 2.19: Error correction scheme.

the PUF response), and t=15 is the number of errors that can be corrected by this BCH code. We chose these parameters according to our reliability and aging effects of the PUF outputs. As shown in Fig. 2.19, During the generation process, a k-bit string (139 bits) is randomly chosen from the noisy PUF data Q itself and encoded to an n-bit (255 bits) BCH codeword (C) by using the BCH encoder. The code-word is offset by XOR with the n-bit PUF output Q and the result is stored as the helper data S. During the reproduction process, the helper data is used to regenerate the key K from a noisy PUF response $Q^{'}$. In this case, the helper data S is offset by XOR with the n-bit noisy PUF data $Q^{'}$ and the result is decoded by using BCH decoder, which is then used to regenerate the key k. The BCH (255, 139, 15) design consumes 418 slices (encoder: 36 slices, decoder: 382 slices) and results in a latency of 831 clock cycles (encoder: 255 clock cycles, decoder: 576 clock cycles).

2.7 Summary

Efficient designs and FPGA implementations of RO-PUF, RS-PUF, and A-PUF with significantly enhanced performances have been reported in this chapter. Through statistical analysis of the obtained results, it has been shown that the incorporation of fine and coarse PDLs of FPGA LUTs significantly advances the

architecture and implementation scheme of PUF technology, and thereby enhances the uniqueness and randomness in the PUF responses. In addition, incorporation of the TMV scheme improves the reliability significantly. It has been shown that the proposed designs achieve better hardware efficiency and also yield the most area-efficient RO-PUF, RS-PUF and A-PUF reported so far. The proposed architectures are also resistant to temperature, supply voltage and correlated process variations.

■

Chapter 3

Efficient Hybrid PUF Design with Enhanced Uniqueness and Randomness

3.1 Motivation

It has been identified that factors such as the systematic or correlated process variation and the environmental noise caused by the voltage and temperature variations degrade the uniqueness and the reliability of PUF responses as well as the resiliency to external attacks [96]. It is shown that PUF primitives can be combined to enhance the uniqueness, reliability, unpredictability [72, 89, 102, 131]. For example, an XOR operation combines the responses of double Arbiter PUFs to enhance uniqueness and unpredictability compared to a single PUF instance [89]. Similarly, hybrid and composite PUFs also enhance performance metrics [72, 102, 131]. Moreover, a high resolution programmable delay line (PDL) can also be used to enhance PUF performance [98]. There was also report of increased reliability of PUFs through the incorporation of TMV scheme [102]. However, most of these reported designs of PUF suffer from insufficient uniqueness, uniformity, and reliability when implemented on FPGAs, or may consume

excessive resources. Therefore, we propose an efficient and lightweight hybrid PUF primitive, specifically for FPGAs, with significantly enhanced performance.

The key contributions of this chapter are:

- Design and implementation of the most compact hybrid PUF primitive reported in the literature.
- The proposed design has the highest randomness in the PUF response for the hybrid PUFs achieved through the use of the PDLs of FPGA LUTs.
- The proposed design exhibits higher reliability through the introduction of TMV scheme and better uniformity as well as bit-aliasing by the incorporation of hard macro design techniques.

In the subsequent sections, the implementation details of Hybrid RO-RSLatch PUF are discussed. Then we present implementation results along with a detailed analysis for its entropy estimation and correlation resistance. Finally, experimental results and the associated validation is presented.

3.2 Design and Implementations

The design is developed using Xilinx ISE design suite 14.5 and coded in VerilogHDL whereas Matlab is utilized for communication between the PUF instances and the PC using UART interface. First, a brief description of the conventional PUF cells are presented which is then followed by the detailed design of the proposed hybrid PUF.

3.2.1 Hybrid PUF Cells

3.2.1.1 RS-Latch

The SR-latch based PUF implementations on Xilinx Spartan-6 FPGA device are reported in [53] and [4]. The design in [53] includes two NAND gates and two FFs (for each SR-latch) implemented on different kinds of slices on the same CLB. The design proposed by us in [4] has two NAND gates and one FF (for each SR-latch) implemented on same kinds of slices on the same CLB and this enabled implementation of 4 SR-latch in one CLB. However, we implement two NAND gates and one FF (to reduce the clock skew) in different kinds of slices on one CLB as highlighted in pink in Fig. 3.3 for use in the proposed hybrid-PUF. Furthermore, the proposed approach employs the concept of PDL in the design of PUF for improving the random response. For the implementation of the PDL, we again use 2 inputs of each LUTs for connection i.e., the flip flop output and the NAND gate output. However, in this case all the other remaining LUT inputs are configured with $2^4 = 16$ discrete levels (configuration from 0000 to 1111). In our previous design [4], the PDLs utilize only 1-bit control.

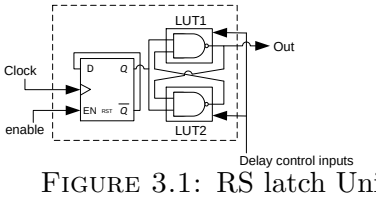

FIGURE 3.1: RS latch Unit

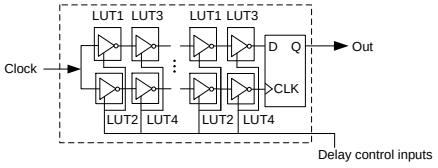

FIGURE 3.2: PUFI Unit

The output of RS latch, architecture in Fig. 3.1, is 1 when input is 0 in a stable state. When input of the RS latch changes from 0 to 1, i.e., the rising edge, the RS latch temporarily enters a metastable state. It then enters a stable state with either output being 0 or 1.

3.2.1.2 A-PUF instances (A-PUFI)

A PDL based A-PUF on Xilinx Virtex 5 FPGA was reported [98]. In this design, non-swapping path based switches instead of path-swapping switches are used to

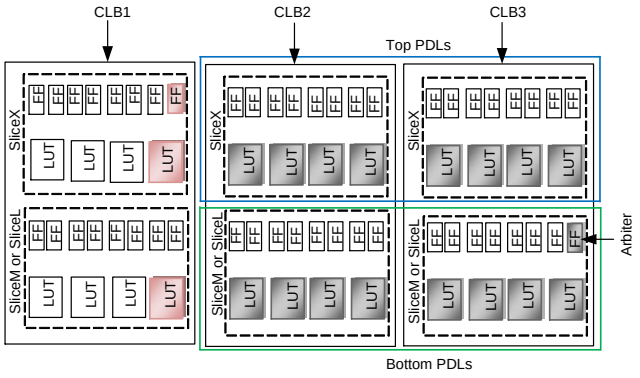

FIGURE 3.3: Implementation of one hybrid RS-Arbiter instance on the 3 CLBs.

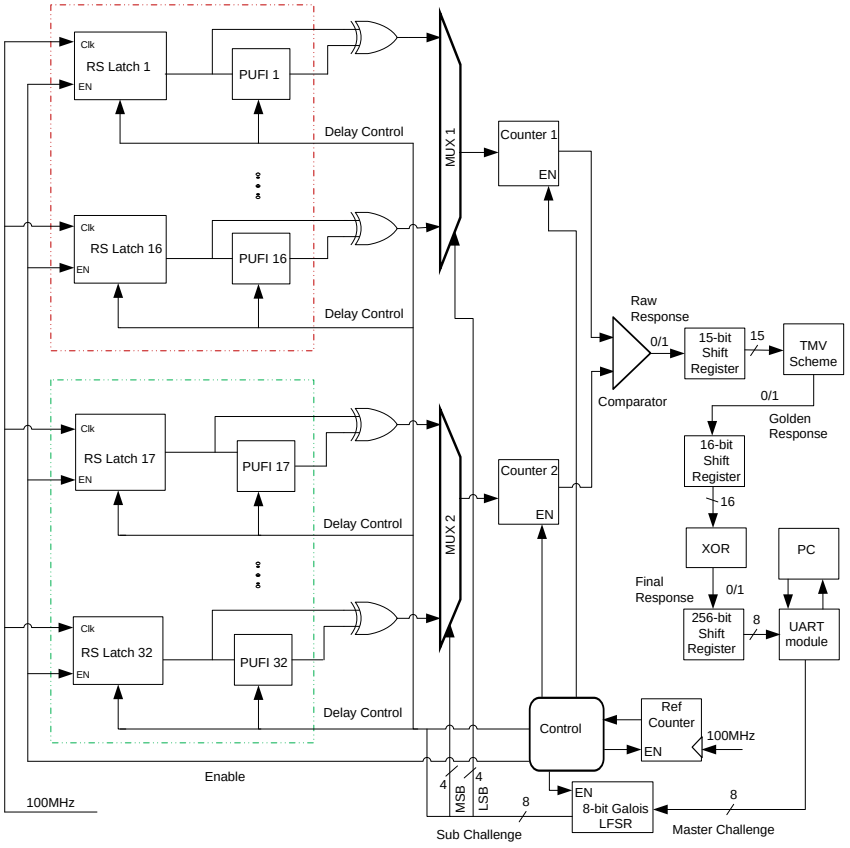

FIGURE 3.4: The proposed design of Hybrid RS Latch-Arbiter PUF.

cancel out the delay skews caused by asymmetries in routing on FPGAs. In that design, the PDL (top and bottom part of switch) is implemented by 2 LUTs each acting as an inverter. The 2 LUTs are implemented in same kinds of slices on different CLBs. It incorporated 16 PUF instances in parallel on the FPGA to produce 16 bits of responses per challenge, where each PUF instance consists of 64 stages of PDLs. In our earlier design [4], the PDL based A-PUF was realized on Xilinx Spartan-6 FPGA and consisted of 8 stages of PDLs for each PUF instance (PUFI). In that design, each PUFI uses 16 LUTs (i.e., 2 CLBs) for buffers and one FF (i.e., 1 CLB) for arbiter using 3 CLBs. Therefore, for the proposed hybrid-PUF, each PUF instance (PUFI) consists of 8 stages of PDLs but each PUFI uses 16 LUTs for inverters and one FF for arbiter using 2 CLBs. The PDL is implemented by 2 LUTs each acting as inverter and the 2 LUTs are implemented in different slice on the same CLB as can be seen in gray in Fig. 3.3. Each PUFI consisting of 8 stages of inverter and one arbiter connects the last stage of the two paths as shown in Fig. 3.2. Additionally, the proposed approach employs the concept of PDL in the design of PUF for improving the random response. For the PDL, one LUT input is used for the connection, one is the challenge bit, while the remaining LUT inputs are configured with $2^4 = 16$ discrete levels (configuration from 0000 to 1111). Note, in our earlier design [4], the PDLs utilize only 1-bit control.

3.2.2 The Proposed Design of Hybrid RS-Arbiter PUF

The main idea of the proposed Hybrid PUF structure is to combine two different sources of randomness in a PUF to improve uniqueness and randomness properties. In the hybrid RS-Arbiter PUF, the output of each RS-Latch cell is XORed with each PUF instance of A-PUF (A-PUFI) as shown in Fig. 3.4. In this design, the output of each RS-Latch cell is also connected to the input of each A-PUFI. Each hybrid RS-Latch cell uses 3 CLBs as shown in Fig. 3.3. Improper routing and placement of PUF cell in FPGA introduces bias in the PUF response bits and affects the uniqueness of PUF responses [96]. Though our LUT placement constraints have fixed the internal routing path of each hybrid RS-Arbiter PUF

cell, the other parts are routed by Xilinx ISE automatically. Moreover, the delay LUTs contain no logic, and the 'keep' attribute is used in the design to stop logic optimization by the synthesis tool. In order to eliminate design-induced bias, hard macro technique has been used with the FPGA Editor in Xilinx ISE design tool, to place the RS Latches and A-PUFIs at selected locations to make all the RS Latches and A-PUFIs identical. In our work, 32 RS-Latches and 32 A-PUFIs were configured at the center of FPGA in a 32×3 matrix of CLBs. This design also incorporates PDLs for enhancement of random response and incorporates TMV scheme for improving the reliability metric.

In our design, shown in Fig. 3.4, the output of each RS-Latch cell and the propagation delay by the Arbiter-PUF instances are the major deciding factors in the resulting count because the output of each RS-Latch cell is XORed with each A-PUF instance. We combined two different sources of randomness in this design to improve uniqueness and randomness properties. The output of each RS-Latch cell depends on the propagation delay plus the process variation in the two NAND gates, and the propagation delay of the interconnection lines. Similarly, the output of each A-PUF instance depends on the propagation delay plus the process variations of the 16 inverters, and the propagation delay of the interconnection lines. A 100-MHz clock signal generated by an on-board oscillator is applied to each FF to divide it into a 50-MHz clock signal before being applied to the corresponding RS latches and the 100 MHz clock signal also drives the 9-bit reference counter. In this design, in Fig. 3.4, starts with the initiation of 8-bit master challenge from a user PC (Matlab) through a UART interface to the 8-bit Galois LFSR, having maximum cycle length, to generate 256 subsequent challenges. Then two different RS-Arbiter instances are chosen from each of these sub-challenges for comparison. In the sub-challenges, the 4 least significant bits (LSBs) select one of the 16 RS-Arbiter instances in group 1, first 16 RS-Arbiter instances marked with gray dashed lines, through the multiplexer 1 while the 4 most significant bits (MSBs) of the sub-challenge select one of the 16 RS-Arbiter instances in group 2, the last 16 RS-Arbiter instances marked with green dashed lines, through the multiplexer 2. In every clock cycle, the respective outputs

of selected RS-Arbiter instances in group 1 and group 2 are fed into the 8-bit counters 1 and 2. Then both counters are incremented every time the selected XORing RS-Arbiter instances output 1's. Counters 1 and 2, and the 9-bit reference counter starts counting simultaneously and the counting is terminated as soon as the reference counter hits its maximum value. At that instance, values in counter 1 and counter 2 are compared to generate a response bit 0 or 1, depending on which counter had the higher value for this RS-Arbiter instance pair.

For the PDLs, 2^4 (= 16) discrete levels to the delay control inputs are applied for each sub-challenge. In this work, we employed concept of majority voting before the XOR operation in order to reduce the response instability and increase the attack complexity [157]. Each discrete level is applied to the delay control inputs 15 times and the generated raw responses are stored in a 15-bit shift register. The TMV concept is subsequently applied on them to determine the "golden response". The golden response is considered 1 if more than half of the generated raw responses are 1s, otherwise it is considered 0. These generated golden responses are stored in a 16-bit shift register and a 1-bit "final response" is generated by XORing the sixteen 1-bit golden responses. A total of 256 final response bits are generated for each 8-bit master challenge and these are stored in a 256-bit shift register. The final response bits are sent to the PC using a UART 8-bit interface for the PUF performance analysis.

3.3 Performance Analysis and Discussion

The uniqueness, uniformity, bit-aliasing, and reliability for the proposed hybrid PUF has been evaluated on 10 Spartan-6 (XC6SLX45-CSG324C) FPGAs.

3.3.1 Uniqueness (UQ)

The ideal value of uniqueness is 50%. The uniqueness is calculated using expression (1.1) [94]. We experiment on 10 FPGAs (k=10) with 256 bit response

length ($n = 256$) at a core supply voltage of 1.2V and standard temperature of $25°C$ for the proposed PUF. The histogram of normalized inter-chip HD between all combinations of pairs of responses (i.e. $\binom{10}{2} = 45$) of proposed PUF is shown in Fig. 3.5. The best fit ideal binomial curve to the histogram of the proposed PUF is also plotted in Fig. 3.5. The achieved mean (μ) is 49.31%, while the standard deviation (σ) is 3.0%. The ideal average HD value is $\mu = 50\%$ and $\sigma = 3.125\%$ for 256 bit response.

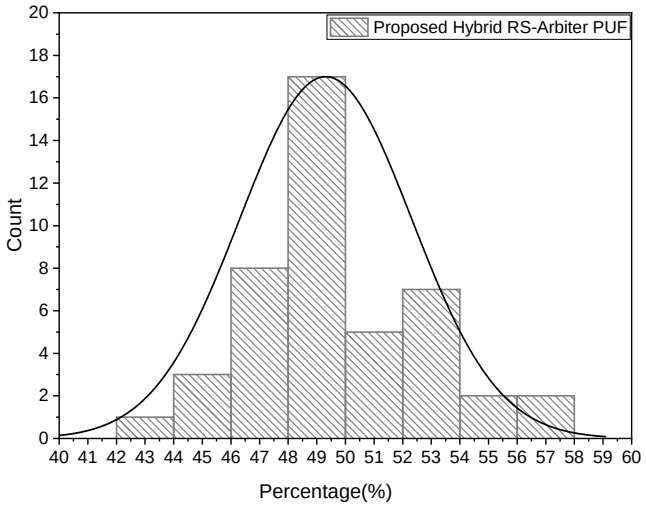

FIGURE 3.5: Inter-chip HD distribution from the 256-bit response.

TABLE 3.1: The experimental results of the proposed PUF design with/without PDL and TMV

Design		without PDL and TMV (%)	with PDL (without TMV) (%)	with PDL and TMV (Including TMV) (%)
Proposed RS-Arbiter	UQ	48.62	49.31	49.31
	UF	49.21	50.17	50.09
	BA	49.21	50.17	50.09
	RE	97.96	97.82	99.22

It can be seen that the proposed PUF possess enhanced uniqueness, (Table 3.1), and this has been achieved by the placement strategy, combining PUF primitives and incorporation of PDLs. In particular, the process (internal) variations and propagation delay of RS-Latch cells and Arbiter-PUF instances are the major deciding factor in the resulting count because the output of each RS-Latch cell is XORed with each A-PUF instance (A-PUFI). The internal variations of FPGA LUTs of each RS-Latch cell and each A-PUF instance can be generated from changes in the LUTs propagation delays under different inputs. Due to the internal

variations of FPGA LUTs, we can generate different outputs/responses under different inputs for each sub-challenge. Further improving the uniqueness, we combined (XORdd) these outputs and generated a 1-bit final response for each sub-challenge. A total of 256 final response bits are generated for each 8-bit master challenge (256 sub-challenges) by applying same procedure. To confirm the effectiveness of XORing the responses while varying the PDL inputs, we developed and evaluated the proposed PUF using the concept of two ways: not utilizing control LUT inputs (i.e., only 0 is applied to the delay control inputs) and utilizing all the configurations to LUT control inputs (i.e., $2^4 = 16$ discrete levels). The achieved uniqueness of the design incorporating the PDLs is higher when compared to the design without using the PDLs. In order to judge the effect of varying the PDL inputs, we also investigated the effect of varying the PDL inputs with/without TMV. The obtained results, in Table 3.1, conveys that the uniqueness results are not impacted in any of the designs with/without TMV. Thus, it can be concluded that the PUF with PDLs achieve only marginally enhanced uniqueness.

3.3.1.1 Entropy Estimation:

In order to perform the CTW compression test [59], as mentioned in Chapter 2, we first concatenate all the bit strings (10 responses) into a single bit string of length $10 \times 256 = 2560$ bits. The compression of this string using CTW results in outcome given in Table 3.2. Clearly, very little compression is achieved by CTW and indicates that the proposed PUF have good compression resistance.

TABLE 3.2: CTW Ratio and Min-entropy Results

Design	Devices	Original size (bits)	Compressed size (bits)	CTW Ratio	Min-entropy per bit
Proposed RS-Arbiter	10	2560	2546	99.45%	0.73

Min-entropy is another metric widely employed to evaluate the lower bound of entropy (i.e. worst case) [47]. It is the worst-case (i.e. lower bound of the unpredictability of the response) measure of uncertainty for a random variable. We use the method described in NIST specification 800-90 [10] for evaluating the

min-entropy of a binary source. We make use of (2.1) and (2.3) to estimate the average min-entropy per bit of the proposed PUF for the 10 FPGAs. The obtained results in Table 3.2 clearly conveys that the proposed PUF and incorporation of PDLs possess sufficient entropy which are close to 0.73. The values found by this test for the proposed PUF are conservative estimates as it will likely increase with increasing number of FPGAs [47]. It can be concluded that the entropy per bit for the proposed PUF lie between 0.73 (from min-entropy) and 0.99 (based on the compression test). Compared to our earlier reported result (see Table 2.4 in Chapter 2) conventional A-PUF, RO-PUF and RS-LPUF, the proposed hybrid PUF improved the entropy slightly. This is due to fact that combining PUF primitives. It is now apparent that the proposed PUF is able to generate sound uniqueness, close to 50%, of signatures and the entropy possess sufficient amount of randomness. These results indicate that the proposed PUF generate highly unique responses and this makes them excellent candidates for security applications in particular which is used to generate device IDs and secret keys.

3.3.2 Uniformity (UF) and Bit-aliasing (BA)

The ideal values of both uniformity and bit aliasing is 50%. The uniformity is calculated using expression (1.5) [94]. Furthermore, as mentioned earlier, an effective PUF response generator should not exhibit bit-aliasing when implemented on different FPGAs. It is calculated using expression (1.6) [94] for a particular response bit across several chips. The proposed PUF with the PDLs is then implemented on ten ($k = 10$) Spartan-6 FPGAs to find out uniformity and bit-aliasing. The uniform bit-aliasing values of the proposed PUF implementation are 50.09% (which are close to the ideal value of 50%). Moreover, the proposed PUF with the PDLs exhibit good uniformity and bit-aliasing values when compared to the design not utilizing PDLs.

3.3.3 Correlation Between Bits

To check the presence of correlation in the test chip, the auto-correlation function R_{xx} (2.4) is used [94]. The minimum R_{xx} among ten different 256-bit responses ($k = 10$ FPGAs) for the proposed PUF is 0.41 while the maximum R_{xx} is 0.59. This indicates that the proposed PUF is resistant to correlation analysis (close to 0.5) and hence no spatial correlation exists within PUF responses.

3.3.4 Reliability (RE)

The ideal value of *reliability* is 100% (i.e., ideal value for HD_{INTRA} (1.2) is 0%). it measures the temporal stability of PUF responses generated by the same master challenge in different environments and can be calculated for one chip using (1.3) [94] and *average reliability* of k chips using (1.4). The proposed PUF is now implemented on five ($k = 5$) Spartan-6 FPGAs for temperature variation from $0°C$ to $85°C$, the rated temperature range of the Spartan-6 commercial-grade FPGA, using a temperature controlled chamber [60] while the core supply voltage is varied within $1.14 - 1.26$V, the rated voltage range of Spartan-6 FPGAs, to find the reliability in responses. Here, one 256-bit response (from each chip) is generated as the reference at the standard temperature of $25°C$ and standard supply voltage of 1.2V, and then other responses are generated for analysis under varying operating conditions. The obtained results of average reliability with respect to temperature bias at 1.2V and voltage bias at $25°C$ for 5 FPGAs. The mean (μ) reliability on temperature and voltage variations for 5 FPGAs of the proposed PUF is 99.22%, while the standard deviation (σ) is 0.42%. Furthermore, Table 3.1 also shows that the reliability result of the proposed PUF significantly improve using a TMV scheme. The TMV is one of the popular techniques to extract statistical bias (bias towards 0 or 1) in the presence of noise [7, 102]. The basic principle of the TMV is to repetitively test (say, 15 times) the PUF using the same challenge and then take the majority value of the responses as the final output. The multiple test of the PUF's response would give the true statistical

bias of the PUF response. It is also clear from the results in Table 3.1 that the reliability of the proposed PUF is almost close to the theoretical maximum of 100%. It means that the proposed PUF is not affected across the range of temperatures and voltages, and compare favourably to the reference temperature of 25°C at supply voltage of 1.2V. Additional error-correction circuit can correct the small errors (the details are given in subsection 2.6.1).

3.4 Comparison of resource consumption and metrics of different PUF designs

The proposed architecture has been implemented and evaluated performance on Xilinx Spartan-6 FPGAs and the obtained results are compared with the existing state-of-the-art in terms of uniqueness (UQ), reliability (RE), uniformity (UF), bit-aliasing (BA), area and processing time (PT) in Table 3.3. Following can be deduced from the obtained results:

- It can be inferred that the proposed hybrid RS-Arbiter PUF design outperforms than all the previous conventional RS-LPUF designs [4, 6, 53, 140, 164] in terms of uniqueness and reliability. Furthermore, the proposed PUF outperforms in terms of uniformity [4, 6] and bit-aliasing [6], and both the area and processing time [53].

- Proposed hybrid PUF is occupies more area and low reliability than other reported conventional A-PUF [89] but outperforms in terms of resources consumed per response bit, UQ, UF and BA. Furthermore, the proposed hybrid PUF design outperform the previous conventional A-PUF designs [4, 94, 130] in terms of uniqueness, reliability, uniformity and bit-aliasing.

- Compared to our earlier reported conventional A-PUF [4] and RS-LPUF [4], the proposed hybrid PUF occupies more area but exhibits substantially improved performance in all aspects.

TABLE 3.3: Performance Comparisons with Previous PUFs

Design Design Ideal Value	Uniqueness 50%	Reliability 100%	Uniformity 50%	Bit-alias 50%	Area (Slices†)	Process time (msec)	Resp. bit length	Slices/ Res. bit	Target FPGA Device*
RS-LPUF [164]	49.00	99.14	—	—	—	—	256	—	S6
RS-LPUF [53]	49.24	98.57	—	—	324	5120	256	1.27	
RS-LPUF [4]	48.10	99.19	50.2	—	54	—	256	0.21	
RS-LPUF [140]	34.73	92	–	—		—	128		
RS-LPUF [6]	49.32	98.80	44.65	44.65	—	—	127	—	S3
A-PUF [89]	04.70	99.32	54.78	—	177	—	128	1.38	V5
A-PUF [94]	18.37	99.76	55.69	19.57	—	—	—	—	
A-PUF [4]	44.30	96.00	48.45	—	234	—	256	0.91	S6
A-PUF [130]	45.25	95.93	48.10	—	—	—	—	—	S3
Composite PUF [89]	50.24	88.20	53.94	—	436	—	128	3.41	V5
Composite PUF [131]	49.04	97.48	50.07	—	1051	—	—	—	S3
Hybrid PUF [148]	32.52	96.96	55.66	—	—	—	256	—	S6
Hybrid PUF [72]	39.63	93.63	48.30	25.20	—	153.6	128	—	V2P
Proposed Hybrid PUF	49.41	99.22	50.09	50.09	257	31.5‡	256	1.0	S6

† The total slices for the PUF with control logic (without UART).
‡ The total number of clock cycles required to generate a response = Total sub challenges × (Ref.counter counts × (delay evaluations × TMV)) + control logic.
* Xilinx Spartan-6 (S6), Xilinx Spartan-3 (S3), Xilinx Virtex-2 Pro (V2P) and Xilinx Virtex-5 (V5)

- For the proposed Hybrid RS-Arbiter PUF, much enhanced performance in all aspects when compared to other composite [89, 131] and hybrid PUF designs [72, 148] in terms of uniqueness, uniformity, reliability and bit-aliasing. Furthermore, new proposed design outperforms in terms of area [89, 131], throughput [72] and resources consumed per response bit [89].

In summary, the statistical analysis of measured data demonstrates significantly better performance of the proposed designs in terms of uniqueness, reliability, uniformity, and bit-aliasing. Therefore, our proposed PUF design can be used to generate device IDs and secret keys for device identification, encryption and IP protection applications.

3.5 Summary

Efficient design and implementation of FPGA based hybrid RS-Arbiter PUF with significantly improved performances has been reported in this chapter. Detailed

statistical analysis of the obtained results convey that the incorporation of PDLs of FPGA LUTs significantly advances the architecture and implementation scheme of PUF technology, and thereby enhances the uniqueness and randomness in the PUF responses. In addition, incorporation of the TMV scheme improves the reliability significantly. It has also been shown that the proposed design yields the most area-efficient composite and hybrid PUFs reported so far.

■

Chapter 4

Efficient TRNG Design and Implementation

This chapter presents a new and efficient method to generate true random numbers on FPGA by utilizing random jitter of free-running oscillator rings as a source of randomness. The free-running oscillator rings incorporate programmable delay lines to generate large variation of the oscillations and to introduce jitter in the generated ring oscillators clocks. The main advantage of the proposed TRNG utilizing programmable delay lines is to reduce correlation between several equal length oscillator rings and thus improve the randomness qualities of the produced bit stream.

4.1 Motivation

There are different problems that might arise in the construction of a TRNG based on oscillator rings implemented in FPGAs [158]. For example, the entropy of the output bit sequence from the TRNG would be drastically reduced when equal length oscillator rings configured in FPGAs are highly correlated with each other due to identical delays. To address this issue, we use programmable delay lines (PDLs) in the oscillator rings in the work described in this chapter. This

creates higher variation in RO oscillations between cycles and hence causes jitter in the generated RO clocks. Further, the output of the RO's are not correlated with each other due to the incorporation of the PDLs in the oscillator rings for each sampling clock which produces higher randomness [5]. In addition, Von Neumann corrector is used as post-processor for improving statistical properties of the bitsteam produced by the proposed TRNG. We implement the base TRNG as well as the Von Neumann corrector on the same Xilinx Spartan-3A FPGAs (XC3S400A-4FTG256).

The key contributions of this chapter are:

- Proposal of an FPGA-based TRNG that uses PDL-induced random jitter in the clocks of free-running ROs as the source of randomness.
- Demonstration of effectiveness of the Von Neumann corrector as a post-processor for bias elimination.
- Experimental validation of the proposed TRNG and demonstration that it passes all tests in the NIST suite.
- The hardware evaluation results demonstrate high throughput-per-area and high entropy rate (i.e., true randomness) of the produced output bits.

The following sections briefly discuss some existing RO based TRNGs, and the details of the proposed TRNG. Subsequently experimental evaluation and comparisons in terms of area and throughput with existing techniques, and quality comparison by using the NIST statistical test suite are presented.

4.2 Ring Oscillator Based TRNG

As mentioned earlier, a number of RO based TRNG designs have been reported in literature [16, 35, 95, 117, 144, 158, 159]. Typically, jitter is accumulated in the free-running RO's consisting of odd number of inverters or delay elements

connected in ring configuration. This causes digital value of the oscillators output to change with a period of approximately $2DL$ where D is the delay of a single inverter and L is the number of inverters in an oscillator. Period of these oscillations vary from cycle to cycle causing jitter of the rising and falling edges of the generated RO clocks as shown in Fig. 4.1. Such oscillations, and hence jitter, in digital circuits can occur due to power supply variations, cross talks, semiconductor noise, temperature variations, and propagation delays. These jitters can be used to generate a stream of truly random bits using D flip-flop (DFF) based sampler for sampling the output of a high frequency oscillator as illustrated in Fig. 4.2.

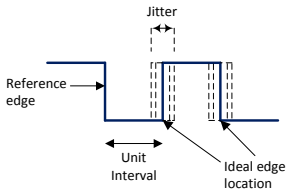

FIGURE 4.1: Jitter in clock signals

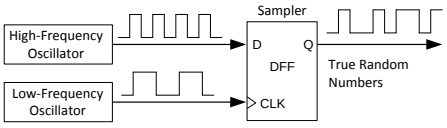

FIGURE 4.2: Basic oscillator-based TRNG

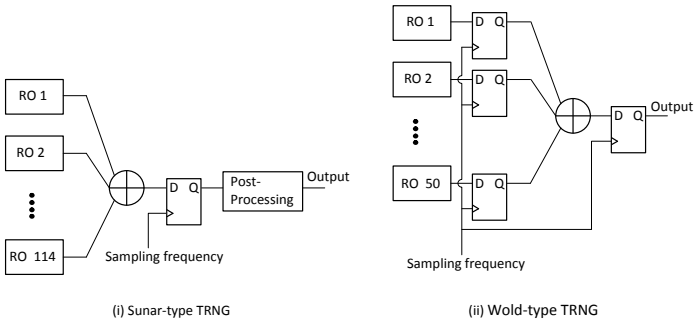

FIGURE 4.3: Block diagram of the original TRNG (a) [144], and the modified TRNG (b) [159]

The quality of generated true random bits can be improved by employing multiple free-running RO's [144]. This is achieved by feeding the outputs to an XOR tree (i.e., a multi-input XOR) and then sampling the XOR tree by a reference clock with a fixed frequency using a DFF to generate the random bit stream as shown in Fig. 4.3. However, it is very challenging for the XOR-tree and the sampling DFF

to handle high number of switching activity from the free-running RO's in such designs [35]. It is due to the fact that high number of transitions during a sampling period due to parallel RO's place stringent setup and hold-time requirements. This aspect can be addressed to a small extent by incorporating a sampling DFF at the output of each free-running RO as shown in Fig. 4.3 [159]. This design passes the NIST statistical tests without post-processing and employs reduced number of RO's. However, it has similar mathematical complexity to the original design [144] and hence similar associated problems such as mutual dependence of rings [16] and correlation between the rings [158], which cause a lack of entropy at the output of the TRNG. The randomness qualities of the original TRNG [144] can also be improved at the cost of higher hardware resources [95] or a lower throughput [117].

4.3 The Proposed TRNG Architecture

For prototyping of the proposed TRNG, Spartan-3A FPGA from Xilinx is employed and it can be considered as a grid of interconnected Configurable Logic Blocks (CLBs) subdivided into four logical components called slices. It has two pairs of CLB slices, SLICEL and SLICEM, with each slice containing two 4-input Look-Up Tables (LUTs) and two flip-flops (FFs). In brief, SLICEL is the most basic type of slice and supports logic only while SLICEM supports both logic and memory functions (including RAM16 and SRL16 shift registers). The LUT based primitives are instantiated in hardware description language (HDL) as described in Spartan-3 Libraries Guide for HDL Designs.

It is imperative to note that the RO based TRNGs, although exciting, are extremely limited in terms of randomness when identical RO's are employed [158]. Equal length oscillator rings configured in FPGA are highly correlated with each other due to identical delays and therefore the XOR of the output from these rings returns mostly zeros. This leads to poor randomness from the design. We show in this work that the a TRNG incorporating PDL's can overcome this problem. Previously, the metastability of flip-flops was used for generating true

random numbers [99]. They achieved metastability by using PDLs that accurately equalize the signal arrival times to flip-flops. On the other hand, our work uses random jitter of free-running oscillators for generating true random numbers. We employ the PDL's (See Fig. 2.2) in oscillator rings to generate large variation of the oscillations and to introduce jitter in the generated RO's clocks. The main advantage of the proposed TRNG utilizing PDL's is to reduce correlation between several equal length oscillator rings. For example, this can be achieved by variable RO outputs for each sampling clock by incorporating PDL as shown in Fig. 4.4. Moreover, the variation in RO oscillations from cycle to cycle (CTC) is also introduced by each oscillator ring due to inverter-delay. As a result, the XOR operation significantly improve the randomness qualities.

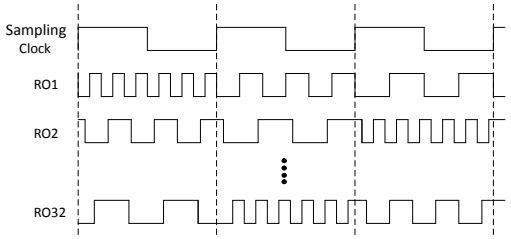

FIGURE 4.4: RO outputs for each sampling clock by using PDL (variation in oscillations from CTC are not shown for clarity)

The implementation of the proposed TRNG along with the post-processing module on a low cost Xilinx Spartan-3A FPGAs is described in this section. Subsequently, Xilinx ISE design suite 13.4, the PyUSB-1.6 software, and the FT2232D USB interface is used for experimental evaluation of the proposed technique.

4.3.1 Design Overview

The proposed TRNG architecture is shown in Fig. 4.5. Here, each RO is realized using 3 inverters and 1 AND gate marked by black dashed boxes. The role of the AND gates is to enable the respective RO's. The design of the inverters and the AND gate require three and one LUT on the FPGA, respectively. In order to generate programmable delays inside the 4-input LUT, one of the LUT inputs is the ring connection while the other three inputs are configured with $2^3 = 8$ discrete

levels. In case one uses 6-input LUTs (which are common in high-end FPGAs), one can use one input for ring connection and allow $2^5 = 32$ delay configurations for the remaining LUT inputs. This allows configuring 32 RO's without any constraints on the placement of the inverters.

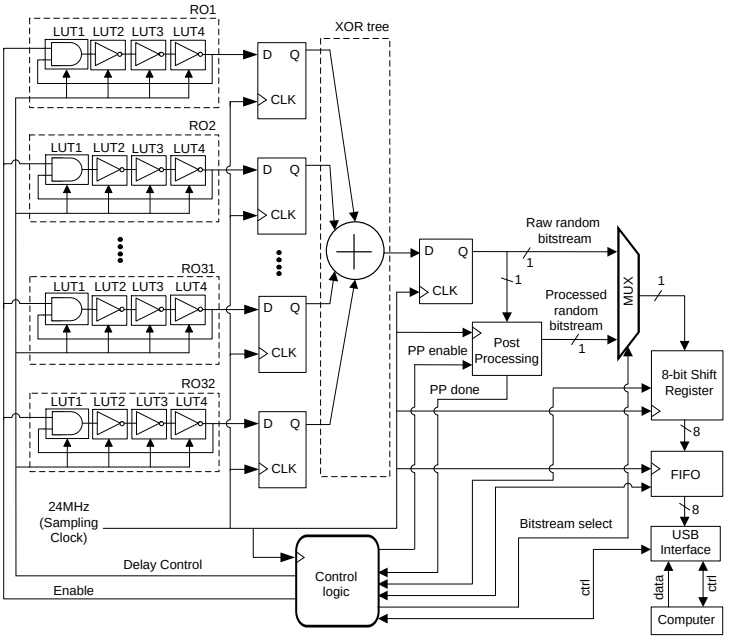

FIGURE 4.5: Architecture of the proposed TRNG.

The proposed architecture consists of 32 RO's, XOR tree, DFF's, shift register, FIFO (First-In, First-Out), and a post-processing unit. First, the control circuitry starts the 32 RO's simultaneously using the 'enable' input. The RO outputs are then combined by the XOR tree and sampled at the frequency clock of 24 MHz. If higher operating frequency is used for sampling then a frequency divider may be needed as well. Then, for the generation of PDLs, 2^3 discrete levels are arbitrarily applied to the delay control inputs for each sampling clock. Subsequently the sampled bits are either fed to the post-processor unit or directly sent to the FIFO without post-processing. Thus the output is either raw random bitstream or processed random bitstream selected via control input of the multiplexer and collected in blocks of 8 bits using the 8-bit shift register. Finally, each byte is stored in a FIFO of 64 byte width (i.e. 512 bits) and sent to PC through a USB interface for the TRNG statistical analysis. The FIFO allows reading of raw/processed random bitstream without flow interruption. The control logic module enables

the start and stop of the RO's, FIFO, 8-bit shift register, post-processing unit, and selection of the raw/processed random bitstream for transfer to the PC.

4.3.2 Post-Processing

The proposed implementation employs a simple post-processing unit based on a Von Neumann corrector [152] for enhancing the entropy and for removing any bias in the generated random bits. The post-processor also provides robustness of the TRNG output sequence. The employed Von Neumann corrector post-processing scheme is depicted in Fig. 4.6. We read two bits at a time from the raw TRNG output and discard them if both of them are same (i.e. we eliminate 00 and 11 patterns). If the two bits are different (i.e. 01 or 10) then we take the first bit and discard the second bit. On an average, the post-processing unit requires 512 bits of raw input to generate 128 bits of post-processed output.

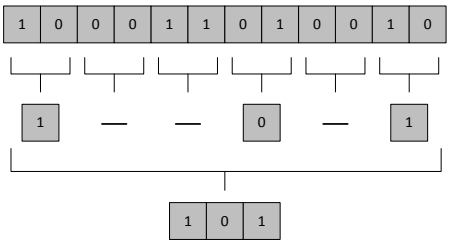

FIGURE 4.6: Principle of operation of the Von Neumann.

4.4 Analysis and Discussion

4.4.1 Hardware overhead analysis

The results from the proposed TRNG architecture implemented on Xilinx Spartan-3A FPGAs are compared with recently reported TRNGs on Xilinx FPGAs in Table 4.1. It is apparent that the proposed design outperforms earlier design [54] in terms of area. However, the throughput in our design is lower as we use a 24 MHz operating frequency (O.F) whereas the design in [54] uses a 100 MHz

O.F. Furthermore, compared to previous works [99, 117, 166], our design exhibits higher throughput but occupies more area. The designs [99, 117, 166] employ more advanced FPGA configurations and hence achieves better area performance. On the other hand, our design outperforms [95] in terms of both the area and throughput although both are implemented on similar type of low cost FPGA. In summary, it can be concluded that the proposed TRNG architecture stands out as a potential candidate for lightweight secure applications considering that it provides very competitive area-throughput trade-offs.

TABLE 4.1: Comparison of the proposed method with other existing TRNGs implemented on Xilinx FPGAs

TRNG Types	Area*			T.put (Mbps)	O.F. (MHz)	Platform
	LUTs	FFs	Slices			
This work	528	177	270	6	24	Spartan-3A (XC3S400A)
RO-based [95]	712	753	—	3.2	—	Spartan-3E (XC3S500E)
RO-based [166]	10	5	—	1.15	100	Spartan-6
RO-based [117]	521	131	—	2.57	—	Spartan-6
Meta stability [54]	—	—	580	12.5	100	Virtex-4 (XC4VFX20)
Meta stability [99]	128	—	—	2	—	Virtex-5 (XC3S500E)

* The Area for the TRNG with control logic (without USB interface).

4.4.2 Correlations Test

In order to investigate correlation between the rings used in the proposed TRNG, we computed the sample Pearson's correlation coefficient [158] between all combinations (i.e., $\binom{32}{2} = 496$) of the 32 RO's outputs. We generated 50,000 consecutive sampled bits from each of the RO's and used it for the test. The outputs of all these rings were sampled with an external sampling frequency of 24 MHz and 8 discrete levels were arbitrarily applied to the delay control inputs for each sampling clock. The correlation coefficient was observed to be ± 0.06 for all combinations. This allows us to conclude that the rings used in the proposed

TRNG are not correlated with each other (i.e., correlation coefficient is close to 0). Moreover, we generated 80 million random bits from our proposed design and used it for the auto-correlation test specified in AIS-31 (T5) [135]. This test checks the bitwise correlation. The generated bit sequence passed the test and hence our proposed design has no correlation and dependency problems.

4.4.3 Measures of the Quality of Randomness

As per the standard practice in the domain, entropy test as the objective criterion of randomness is carried out. Followed by this, the restart test is carried out to provide evidence that the output, before post-processing, is distinct after restarting the system multiple times under the same conditions. Finally, the quality of the bitstream produced by the TRNG is tested using the NIST statistical test suite.

4.4.3.1 Entropy Estimation

The expected entropy per bit is 1 for an ideal true random number generator because the proportion of '0's and '1's is ideally 0.5. To estimate the entropy rate for the generated numbers, test $T8$ of the procedure B of the AIS-31 [135], available to test raw random numbers, is utilized in this work. Considering a sequence of length N, test $T8$ splits the sequence in $Q+K$ disjoint L-bit words. The suggested parameters for the test $T8$ are $L = 8$, $Q = 2560$ and $K = 256000$. The minimum sequence of length required for this test is 7200000 bits. We generated 80 million random bits from our implementation and used it for the test. The bit sequence passed the test and achieved entropy of 7.9946 bits per byte (i.e., 0.9993 per bit). This is better than the minimum entropy requirement of 7.976 per byte (i.e., 0.997 per bit). Therefore, the proposed TRNG successfully fulfills the criteria for procedure B of the AIS-31 and can be used for applications that require high security. Moreover, Our proposed RO-based TRNG achieves higher entropy per bit when compared to earlier designs [117] (0.999 per bit) and [166] (0.997 per bit).

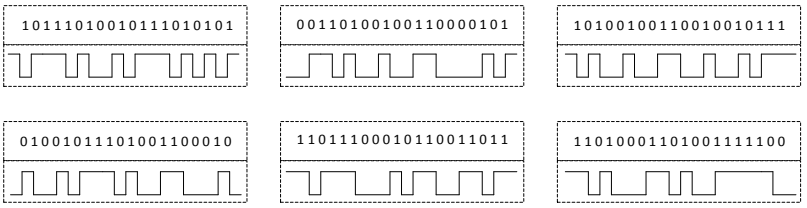

FIGURE 4.7: 6 output bitstreams captured after restarting the TRNG.

4.4.3.2 Restart Experiment

A test was carried out to verify the start up sequences for 6 restarts, from identical starting conditions, for the proposed design. First 20 sampled bits after every restart were observed and plotted as shown in Fig. 4.7. For a pseudorandom signal, the restart experiment produces similar graphs when the TRNG is repeatedly started from identical starting conditions. However, in our case, these were observed to be different each time and thus discard any pseudo randomness possibilities. Works [85, 159] and [35] used similar procedure to distinguish the amount of true randomness contained in a pseudorandom oscillating signal.

4.4.3.3 Statistical Evaluation of the Output

A set of randomness tests is standardized by NIST for evaluation of the quality of randomness of bitstreams [10]. The NIST statistical test suite is the most comprehensive publicly available tool. In essence, there are 15 type of tests which are typically carried out to assess the performance of TRNGs. The *frequency* test checks the proportion of ones and zeros for the bit sequence, and passing it requires that the bits are roughly equally divided between 0's and 1's. Similarly, the *block frequency* test determines the proportion of 1's within M-bit blocks. The *Runs* test checks that strings of consecutive 1's and 0's in the sequence is as expected in a truly random sequence, while the *longest run* test checks the long strings of 1s within M-bit block. The *Rank* test checks for linear dependence among fixed length sub-strings of the bit-sequence, and the *Fast Fourier transform (FFT)* test checks the repetitive patterns in the tested sequence that would indicate a deviation from the assumption of randomness. The *non-overlapping template* test

is to determine the number of occurrences of pre-specified target strings. The *Overlapping template* test is similar, but when the pattern is found, the window slides only one bit before the next search. The *Universal statistical* test determines the number of bits between matching patterns, and the *Linear complexity* test attempts to find the length of a linear feedback shift register and checks its complexity. The *Serial* test determines the frequency of all possible overlapping m-bit patterns across the bit-sequence, while the *Approximate Entropy* test is similar to *Serial* test but checks overlapping blocks of two lengths. The *Cumulative Sums (Cumsum)* test checks if the 1's and 0's occur in large numbers at early stages or at later stages in the sequence or they are intermixed evenly across the entire sequence. The *Random Excursions* determine the number of cycles having exactly K visits in a cumulative sum random walk, while the *Random Excursions Variant Test* determine the total number of times a particular state is occur in a cumulative sum random walk.

In this work, the parameters of each test were set as per the NIST recommendations [10]. The significance level α was chosen to be the default of 0.01 as it indicates 99% confidence interval. To examine the distribution of the P-values (randomness measure), 1000 times (sequences) of 10^6 bits both before and after post-processing were acquired. The statistical results of P-values and proportions (before and after post-processing) are given in Table 4.2. If P-value is larger than 0.01 then the sequences are approximately uniformly distributed. Furthermore, a test is considered to have passed when the range of acceptable proportions are between 0.98056 and 0.99943 for $\alpha = 0.01$ and sequences $= 1000$ [10]. The Proportion (Prop.) column indicates the number of P-values that were above the 0.01 confidence interval. The minimum proportion (minimum pass rate) for the tests is 0.982 before post-processing, and 0.987 after post-processing. The minimum pass-rate of sequences for our design (0.987) is higher in comparison to [85] (i.e., 0.97) and [54] (i.e., 0.986). The TRNG is then tested under two different temperature values of $55°C$ and $75°C$ using a temperature-controlled chamber [60] while keeping the core supply voltage at 1.2 V. The minimum proportion for the tests is 0.981 at both these temperatures before post-processing. However, after

post-processing, the minimum proportion becomes 0.983 and 0.982 at 55°C and 75°C respectively. In summary, it is evident from the results that the proposed TRNG exhibits better randomness (P-value is greater than 0.01 with greater proportion) properties at low hardware overhead and high throughput.

TABLE 4.2: NIST randomness test results for 1-Gbits record that passed all tests (room temperature at 1.2 V).

Statistical Test	Before Post-processing			After Post-processing		
	P-value	Prop.	Result	P-value	Prop.	Result
Frequency	0.5592	0.990	PASS	0.6199	0.992	PASS
Block-Freq.	0.6621	0.994	PASS	0.7441	0.991	PASS
Cumsum	0.3776	0.987	PASS	0.4138	0.987	PASS
Runs	0.4590	0.992	PASS	0.4276	0.993	PASS
Long-Run	0.7791	0.989	PASS	0.7011	0.990	PASS
Rank	0.7197	0.985	PASS	0.7634	0.988	PASS
FFT	0.5141	0.990	PASS	0.6593	0.994	PASS
Overlapping	0.0962	0.987	PASS	0.0805	0.991	PASS
Approx-Entropy	0.0553	0.988	PASS	0.2145	0.992	PASS
Serial	0.3345	0.991	PASS	0.4698	0.992	PASS
Linear-Complex	0.0088	0.989	PASS	0.2703	0.989	PASS
Nonperiodic	0.1573	0.990	PASS	0.4213	0.993	PASS
Universal	0.0909	0.982	PASS	0.2771	0.988	PASS
Rand-Excur	0.3394	0.986	PASS	0.3781	0.990	PASS
Rand-Variant	0.2587	0.984	PASS	0.3347	0.987	PASS

4.5 Summary

In this chapter, a new design of RO-based TRNG is described and its implementation on Xilinx Spartan-3 FPGA is presented. The programmable delay of FPGA LUTs has been used to achieve random jitter and to enhance the randomness. It has been demonstrated that the proposed implementation provides a very good area-throughput trade-off. Effectively, it is capable of producing a throughput of 6 Mbps after post-processing with a low hardware footprint. In addition, the restart experiments show that the output of the proposed TRNG behaves truly random. It achieves high entropy rate and successfully passes all NIST statistical tests.

■

Chapter 5

Authenticated Key Agreement Protocol Using ECC, PUF and TRNG

Establishing a secure connection between a device and a host processor (or between two devices) requires a secure protocol. As we know, symmetric cryptography requires that a copy of the same key is used on both ends of the communication channel, which can be achieved by a key agreement protocol. In this chapter, we investigate secure and efficient FPGA implementation of a key agreement protocol for IoT devices using elliptic curve, PUF and TRNG with very competitive area-throughput trade-offs. The key agreement protocol using PUF is meant for establishing a unique long-term secret key, the TRNG is for short-term random secret key generation, BEC is for generating the public key corresponding to the secret key, and ECMQV is used for generating the shared secret key and key exchange.

5.1 Motivation

Elliptic curve cryptography (ECC) is a type of public-key cryptography (PKC) that was introduced by Miller [106] and Koblitz [73] in the mid-1980s. Elliptic

curve-based protocols are the preferred choice for key agreement and signatures due to their lower key sizes [124, 137] than other public key based schemes that achieve the same level of security. Importance of public key cryptosystems is well-known and these are extensively used in softwares. However, software based implementations lead to slower designs and downgrade the overall system performance. This is due to the use of complicated mathematical operations like multiplication and exponentiation on groups. Therefore, efficient implementation (in terms of the area-time product) of such protocols on suitable hardware platforms is considered a better alternative [40, 75, 132, 136].

In this context, hardware implementation of an elliptic curve based protocol depends on the choice of the model of the curve. Commonly used models of elliptic curves are Weierstrass, Montgomery, Edwards, Hessian and Huff. About a decade ago, Bernstein et al. presented an alternate model called Binary Edwards Elliptic Curve (BEC) [15]. Moreover, several works discuss Field Programmable Gate Array (FPGA) based implementations of group law on BEC [8, 9, 20, 21, 38, 39]. The BEC provides a unified formula for computing point addition and point doubling operations on an elliptic curve, that is, the same sequence of operations are used for both these computations. This property of BEC helps make the point addition and point doubling operation indistinguishable to an attacker. The BEC unified formula is thus intrinsically resistant to the side channel attacks (SCA). Furthermore, BEC allows slightly faster point addition in comparison to other methods [87, 139].

In a seminal work, Diffie and Hellman [36] proposed the first two-party key agreement protocol that can be used for deriving a shared key. Elliptic curves can be used to instantiate this protocol. However, this protocol does not provide authenticity of the shared key. On the other hand, most common use-cases of such a protocol require authenticity of the shared key material to each party in order to avoid man-in-the-middle attacks. One of the ways to include authenticity in Diffie-Hellman (DH) protocol is to use some authentication mechanism on top of the protocol [76]. In a different approach, it is possible to ensure authenticity of the shared key in the protocol itself. The protocol given by Menezes, Qu and Vanstone

(MQV) [79] follows this approach and supports authenticated key agreement. An elliptic curve version of the MQV protocol has also been studied in [79] and is referred to as the ECMQV protocol. Certain security flaws in this protocol were pointed out and fixed in [77, 133]. Our goal is to study the overhead incurred in implementing a representative, authenticated version of DH. We wish to point out that our implementation of ECMQV can be used as a starting point of any elliptic curve based authenticated key agreement protocol. In this chapter, we propose efficient FPGA implementation of authenticated key agreement protocol in IoT using BEC, PUF and TRNG modules.

The main contributions of this chapter are:

We present a high-performance hardware architecture of an ECMQV protocol using 251 bit BEC [15], PUF and TRNG to address some of the constraints in the existing state-of-the-art.

Firstly, we discuss our choices for finite field multiplier module. In this work, we choose the bit-parallel Karatsuba multiplier [121] for performing finite field multiplications in BEC arithmetic. We use the same finite field as used in Bernstein's work on BEC [15]. The details are given in Section 5.2.

In [38], the authors discuss affine coordinates based BEC implementation using the bit-serial multiplier (at the finite field level). Projective coordinates based BEC implementation using the bit-parallel multiplier has been reported in [20, 21, 39]. Further, w-coordinate based BEC implementations using digit-level multiplier have been reported in [8, 9]. In this work, we investigate the performance of affine, projective and mixed w-coordinate based BEC arithmetic when using a bit-parallel multiplier approach. Our w-coordinate based BEC implementation occupies significantly lower chip area and has higher computing speed when compared to our affine and projective coordinates based point addition. The details are given in Section 5.3.

Next, we also analyze the performance of point multiplication (scalar multiplication) on BEC using Montgomery ladder. Unlike earlier designs, we

concentrate on a high speed hardware implementation of this algorithm which is resistant to SPA attacks. We use mixed w-coordinates with bit parallel multiplier for finite field multiplication. Our implementation of BEC scalar multiplication is performed in $21.32\mu s$ using 15223 slices on Virtex-4 FPGA and $16.24\mu s$ using 7511 slices on Virtex-5 FPGA. The mathematical properties of BEC and usage of Montgomery ladder ensure that the proposed design resists SPA attack. We perform SPA on the Montgomery ladder module and experimentally demonstrate its resistance to such attacks. The details are given in Section 5.4.

We achieve high-performance for FPGA implementation of the BEC scalar multiplication by: (1) employing the mixed w-coordinate differential formulations for point multiplication [15], (2) placing the registers efficiently in FPGA, (3) usage of efficient scheduling mechanisms to reduce the area consumption and processing time for the BEC arithmetic implementations, and (4) using bit parallel Karatsuba multiplier [121] for field multiplication. Moreover, the multiplication operation is done in one clock cycle; while the addition and squaring operations are done in parallel (no extra clock cycle is required for the addition and squaring operations).

Finally, we discuss an FPGA architecture for performing key exchange over BEC and discuss the performance of our implementation. We utilize our PUF and TRNG modules with the above components to implement an ECC based authenticated key exchange protocol, namely, the ECMQV protocol. Our PUF and TRNG primitives (described in the previous chapters) are used for the long term and short term secret keys generation in the key agreement protocol. Our implementation of ECMQV requires 180 $msec$ using 32102 slices on Virtex-4 FPGA and 151 $msec$ using 15495 slices on Virtex-5 FPGA. To the best of our knowledge, ours is the first work which discusses efficient FPGA design of the ECMQV protocol using binary Edwards curves, PUF and TRNG. The details are given in Section 5.5.

All our proposed implementations have been done using VerilogHDL, and the targeted Xilinx FPGAs were Virtex-4 XC4VFX140 and Virtex-5 XC5VLX110. We have used Xilinx ISE 13.4 tool for design synthesis and post-implementation.

The organization of the chapter is as follows. We describe our finite field routines in Section 5.2. Details and comparison of BEC implementations using various coordinate systems are given in Section 5.3. Point multiplication implementation and its SPA attack resistance are discussed in Section 5.4. ECMQV implementation details are presented in Section 5.5. Finally, we conclude the work in Section 5.6.

5.2 Field Arithmetic over $\mathbb{GF}(2^{251})$

In this section, we present finite field (FF) operations over the $\mathbb{GF}(2^{251})$ which was also used in Bernstein's work on BEC [14]. The main operations used in BEC arithmetic are field addition, field squaring, field multiplication, and field inversion.

Addition over $\mathbb{GF}(2^{251})$ is simply a bitwise *xor* operation which is a carry free operation. Square operation is performed as interleave with zeros followed by a reduction modulo $f(x)$, where $f(x) = x^{251} + x^7 + x^4 + x^2 + 1$. Multiplication is carried out by computing a multiplication of polynomials in $\mathbb{GF}(2^{251})$ followed by a reduction modulo $f(x)$. We have adapted a hybrid Karatsuba bit parallel multiplier from [121] to the 251-bit field. We use simple Karatsuba decomposition and accumulation up to the multiplicand size of 30-bit and general Karatsuba for 15-bit and 16-bit multiplications. The main advantage of hybrid Karatsuba bit-parallel multiplier is the sub-quadratic complexity of the Karatsuba algorithm coupled with efficient utilization of Look Up Table (LUT) resources in FPGA's. Further, the bit-parallel scheme requires lower clock cycles compared to bit-serial multipliers [38] and digit-level multipliers [8, 9]. We refer to our ACM TRETS 2018 publication [3] for field inversion details.

5.3 BEC Arithmetic: A Comparative Study

Binary Edwards Curves have a unified and very efficient group law. Many coordinate systems can be used for representing points and performing arithmetic. We compare the affine and projective coordinate representation based addition on BEC. While affine system uses two coordinates for point representation (group law involves inversion), projective system uses three coordinates. We refer to our ACM TRETS 2018 publication [3] for more details.

In this thesis, We mainly study the w-coordinate system based differential addition formulae. "Differential addition" means computing $Q + P$ given Q, P, and $Q - P$ i.e., addition of points with known difference [15]. One of the goals of our work is to benchmark Montgomery ladder and hence key agreement. The w-coordinate formulae are most suitable for this purpose. These are analogous to the *x-coordinate only* arithmetic on the Montgomery curves. We implement these formulae and present our FPGAs statistics in Subsection 5.3.3.

5.3.1 The Binary Edwards Curve Equation

In this section, we describe the elliptic curve used in our implementation. We used the same parameters as Bernstein's work on Batch Binary Edwards [14], i.e. the base field is a 251-bit FF and the generating polynomial is $f(x) = x^{251} + x^7 + x^4 + x^2 + 1$. We use the same polynomial for all the implementations in this work. The binary Edwards curve is

$$E : d(x + x^2 + y + y^2) = (x + x^2)(y + y^2),$$

where $d = x^{57} + x^{54} + x^{44} + 1$. Since the trace of d is 1, this curve has complete addition law. This curve also has near prime order $4p_1$, where p_1 is a 250 bit prime. The works [14, 15] provide unified addition formula on BECs in affine, projective and w-coordinates.

5.3.2 Affine vs. Projective Coordinates: A Comparison

We have tabulated the resources required for implementing BEC across affine and projective coordinate systems in Table 5.1. We refer to our ACM TRETS 2018 publication [3] for implementing details. Our projective coordinate implementation clearly requires much smaller area and performs better than our affine coordinate implementation.

TABLE 5.1: FPGA implementation results of BEC Point addition using affine and projective coordinate systems

Point Addition	**Field size**	**Area**			**Freq** (MHz)	**Clock Cycles**	**T** (μs)	**AT**	**FPGA Device**
		Slices	LUTs	FFs					
affine coordinates	251	29914	57385	4966	108.84	80	0.735	0.042	Virtex-4
		15773	45288	4912	138.2	80	0.579	0.026	Virtex-5
projective coordinates	251	17521	33427	4348	107.5	23	0.214	0.007	Virtex-4
		7678	26389	4573	133.4	23	0.172	0.005	Virtex-5

5.3.3 Mixed w-Coordinates

For point addition in affine coordinates, finite field inversion dominates the computation. Projective coordinate addition does not require inversion but an needs an extra register to store the Z coordinate. In the affine w-coordinate system, an affine point x, y is used as (w), where $w = x + y$.

The Montgomery ladder is commonly used for secure computation of nP, where n is a scalar. However, in order to eliminate inversion step in the addition and doubling operations, it is convenient to work with projective coordinates (X, Y, Z). Therefore, we use mixed w-coordinate system in which the point is represented as (w, Z) where w is $X + Y$. The Y coordinate is not needed explicitly since it can be derived from the w-coordinate. For further details, one may refer to [14, 15].

We use the so-called 'differential addition formula' which has the interesting property of allowing addition and doubling computations simultaneously. Given the intermediate points Q and R, this formula computes the points $2Q$ (or $2R$) and $Q + R$ in the same step. Further details can be seen in [14, 15].

TABLE 5.2: Scheduling for batch BEC differential addition in w co-ordinates

Clock	Operations	RTL decription
1	$C = W_2 \times (W_2 + Z_2)$	temp[1] $\leftarrow W_2 \times (W_2 + Z_2)$
2	$m_2 = W_3 \times (W_3 + Z_3)$	temp[2] $\leftarrow W_3 \times (W_3 + Z_3)$
3	$V = m_2 \times C$	temp[2] $\leftarrow$ temp[1] $\times$ temp[2]
4	$m_4 = Z_2 \times Z_3$	temp[3] $\leftarrow Z_2 \times Z_3$
5	$m_5 = \mathrm{d} \times (Z_2^2)^2$	temp[4] $\leftarrow \mathrm{d} \times (Z_2^2)^2$
6	$m_6 = \mathrm{d} \times (m_4^2)$	temp[3] $\leftarrow \mathrm{d} \times temp[3]^2$
7	$m_7 = (V + m_6) \times w_1$	temp[5] $\leftarrow (temp[2] + temp[3]) \times w_1$
Final outputs are:	$W_4 = C^2$	$W_4 \leftarrow temp[1]^2$ at the end of step 2
	$Z_4 = m_5 + C^2$	$Z_4 \leftarrow$ temp[4] $+ temp[1]^2$ at the end of step 5
	$W_5 = V + m_7$	$W_5 \leftarrow$ temp[2] + temp[5] at the end of step 7
	$Z_5 = V + m_6$	$Z_5 \leftarrow$ temp[2] + temp[3] at the end of step 6

The differential addition formula in mixed w-coordinates for BEC consists of 7 field multiplications and 4 field squarings (two α^{2^1} and one α^{2^2}). For finite field squaring, we use two precomputed powerblocks α^{2^1} (i.e., 1 time $\times$ α^2) and α^{2^2} (i.e., 2 time $\times$ α^2). Powerblocks and addition are performed in parallel with an appropriate multiplication in one clock cycle. Detailed scheduling with RTL description for the operations of differential addition in mixed w-coordinates for BEC is provided in Table 5.2. Our implementation uses only 5 temporary registers for storing intermediate values. We can execute this differential addition in 7 clock cycles each of which consists of one binary field multiplication. Additionally, 3 clock cycles are needed for starting BEC addition, storing current point addition/doubling result, and to reset all the signals and flip flops. Hence, a total of 10 clock cycles are required to perform one BEC addition in mixed w-coordinates.

Note that the a BEC addition implementation in [8] takes 23 clock cycles using 5 registers for storing intermediate values; and the work [9] uses 12 intermediate registers and requires 20 clock cycles to achieve the same. Our scheduling mechanism requires significantly lower number of intermediate registers than [9] and also requires lower clock cycles in comparison to [8, 9]. Our FPGA implementation results are given in Table 5.3. As can be seen from these results, the cost of the w-coordinate based unified formula is much lower in terms of covered area when compared to affine and projective coordinates based point addition, but the computation speed is higher (see Table 5.1).

TABLE 5.3: FPGA implementation results of differential addition formulae in mixed w-coordinates

Differential Addition	**Field size**	**Area**			**Freq** (MHz)	**Clock Cycles**	**T** (μs)	**AT**	**FPGA Device**
		Slices	LUTs	FFs					
mixed w-coordinates	251	14576	27899	3011	118.2	10	0.085	0.0024	Virtex-4
		7021	22455	2914	153.7	10	0.065	0.0015	Virtex-5

5.4 Point Multiplication on BEC using Montgomery Ladder

We now describe efficient computation of nP for a scalar n and a BEC point P. In [109], Montgomery defined non-binary elliptic curves $v^2 = u^3 + a_2u^2 + u$ and used what is known as the 'differential addition'. Using this addition formula, Montgomery suggested a fast algorithm for scalar multiplication on this form of elliptic curves. It has a uniform double-and-add structure, which provides a natural resistance against simple side channel attacks [17, 63, 66]. We recall the algorithm in Table 5.4. For more details one may refer to [29, 109].

TABLE 5.4: Montgomery Ladder Algorithm

Input: A point P on an elliptic curve and a positive number $n = (n_{l-1}, \cdots n_0)_2$.
Output: The point $[n]P$, which is equal to $P + \ldots + P$ added n-times.

1. Initialize $Q \leftarrow P$ and $R \leftarrow P + P$
2. for $(i = l - 2)$ downto 0 do
   if $n_i = 1$ then
   $Q = Q + R$ and $R = R + R$
   else
   $R = Q + R$ and $Q = Q + Q$
3. Return Q

5.4.1 Architecture of BEC Point Multiplication

Analogous to Montgomery curve laddering algorithm, a point multiplication may be performed on BEC using the mixed w-coordinate formulae discussed in Section 5.3.3. We give the architecture of the BEC point multiplication block in mixed w-coordinates which executes left-to-right Montgomery ladder algorithm (described in Table 5.4) in Figure 5.1.

Point Addition Block consists of a bank of registers containing 5 temporary registers to store intermediate results (see in Section 5.3.3), 5 registers to store internal input parameters, 2 registers to store coordinates of input point P (i.e., P_x and P_y), 1 register to store scalar n, 4 registers to store outputs, and 1 register for the curve parameter d. The number of registers used in the design is optimized through careful data flow analysis of the algorithm.

First, the point $P = (P_x, P_y)$ is converted to the mixed w-coordinates by computing $w_1 = P_x + P_y$ and setting $W_2 = w_1$ and $Z_2 = 1$. Initially, the Q_x, Q_y registers are initialized with coordinates w_1, 1. Then, R_x, R_y registers are initialized with doubling results (i.e., $=(w_1,\ 1)+(w_1,\ 1)$) of corresponding coordinates. Control circuitry of the *initialize* block determines selector pins of the multiplexer result.

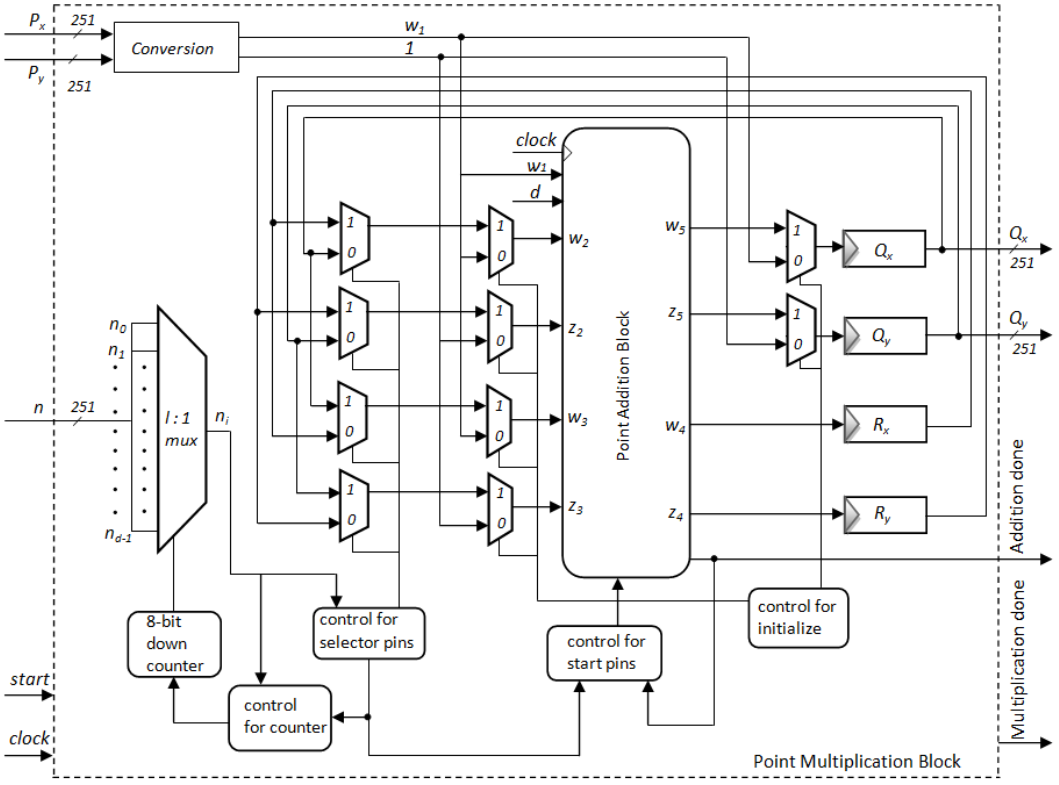

FIGURE 5.1: Architecture of BEC Point Multiplication

There is an 8-bit counter i which counts from $l-2$ down to zero. At every iteration this counter selects the corresponding bit of scalar n. If n is 1 (resp. 0) then the coordinates of point addition result $(Q+R)$ are stored in Q_x and Q_y (resp. R_x and

R_y) registers, while the coordinates of point doubling result $(R + R)$ are stored in R_x and R_y (resp. Q_x and Q_y) registers. The selector pins of the multiplexer results from a control circuitry made of $n[i]$.

There are two sets of input points (W_2, Z_2) and (W_3, Z_3) which are fed to the *Point Addition Block* at every iteration. These two inputs are fed back from the output (see in Figure 5.1). *Point Addition Block* also consists of a single multiplier sub-block where the input multiplicands are fed through a multiplexer. The inputs to the multiplexer are a bank of registers and the output signals of some combinatorial circuits which are used to perform addition and powering operations. Both the multiplication and powering operations are followed by the reduction sub-block. The outputs of the multiplier block after reduction are stored in one of the five temporary registers as defined in our RTL Table 5.2. The *start* pin of the *Point Addition Block* results from a control circuitry of the *initialize*, *selector* pins and *addition done* signals. At the end of the iteration, the *multiplication done* signal is enabled and the resulting outputs (i.e., W_5, Z_5) are stored in registers Q_x and Q_y, i.e., $Q = (Q_x, Q_y)$.

5.4.1.1 Comparison with other BEC Implementations:

Implementation (post place and route) results of the proposed BEC point multiplication are compared with the existing state-of-the-art in Table 5.5. In this table, occupied area is measured in LUTs, maximum clock frequency (Freq) in MHz and the computation time (T= ClockCycles $\times$ 1/Freq) in μs. We also adapt the product of area and computation, i.e. area-time (AT) as an efficiency metric.

As can be seen in Table 5.5, our work provides the smallest area, faster design and yields a better performance (AT) than most other BEC designs [9, 20, 21, 38, 39]. Our design requires more slices compared to the BEC implementation of [8]. However, our design does not require any block RAM (BRAM) while [8] requires 6 BRAMs. The number of LUTs in our design are larger due to the fact that we use an finite field size of 251 bits while [8] uses an finite field size of 163

TABLE 5.5: The proposed FPGA implementation results (after place and route) of Point Multiplication and Comparisons

Work	**Coordinate System**	**Multiplier approach**	**Field size**	**Area**		**Freq** (MHz)	**Clock Cycles**	**T** (μs)	**AT**	**FPGA Device**
				Slices	LUTs					
Our Approach	Mixed w	HK bit-parallel	251	15223	29041	118.2	2520	21.32	0.619	Virtex-4
				7511	23736	155.2	2520	16.24	0.385	Virtex-5
BEC [21]	Projective	HK bit-parallel	233	21816	35003	47	—	190	6.650	Virtex-4
BEC [20]	Projective	HK bit-parallel	233	21816	35003	50	—	170	5.950	Virtex-4
BEC [38]	Affine	bit-serial	233	15804	—	308	—	1000	15.80	Virtex-5
BEC [9]	Mixed w	pipelined digit-level HD	233	29252	48577	198.4	7212	36.3	1.763	Virtex-4
BEC [39]	Projective	Karatsuba bit-parallel	233	—	40793	67	3277	49	1.999	Virtex-4
				—	32874	132	3277	25	0.822	Virtex-5
BEC [8]	Mixed w	pipelined digit-level	163	12834, 6 BRAMs	22815	—	—	23.3	0.532	Virtex-4
				6536, 6 BRAMs	17432	—	—	16.9	0.294	Virtex-5
BKC [83]	Mixed LD	pipelined bit-parallel finite-field	233	5616, 1 BRAM	—	240	—	4.09	—	Virtex-5
			283	7738, 1 BRAM	—	213	—	5.81	—	Virtex-5
BGC [122]	Projective	pipelined HK bit-parallel	233	13620	23147	154	—	12.5	0.289	Virtex-4
				5644	18097	156	—	12.3	0.223	Virtex-5
BGC [82]	LD Projective	pipelined bit-parallel finite-field	233	4762, 1 BRAM	15296	244	1926	7.9	0.121	Virtex-5
			283	6286, 1 BRAM	20256	213	2329	10.9	0.221	Virtex-5
BGC [145]	Projective	digit-serial	233	6615	25129	188.7	3357	17.8	0.447	Virtex-5
			283	7069	25030	189	6347	33.6	0.841	Virtex-5
BGC [71]	Projective	pipelined full-precision	163	11777	42192	113	450	3.99	0.168	Virtex-5
BHC [22]	Projective	HK bit-parallel	233	20437	39034	81	—	73	2.849	Virtex-5

LD: López-Dahab, BRAM: Block RAM, HK: hybrid Karatsuba, HD: hybrid-double
BGC: binary generic curve, BKC: binary Koblitz curve
BEC: binary Edwards curve, BHC: binary Huff curve.

bits. Furthermore, our proposed architecture achieves higher operating frequency as compared to other BEC architectures using bit-parallel multiplier [20, 21, 39]. The architectures [20, 21, 39] introduce many extra primitives into the critical path, apart from the FF multiplier, which results in longer critical path delay and lower clock frequency. For example, Ayantika et al. [20, 21] introduce one quad block (14 cascaded quad circuits), one addition (XOR), one 8:1 MUX and one 2:1 MUX into the critical path apart from the FF multiplier. This greatly increases the critical path delay of their design and causes lower clock frequency. In [39], the authors introduce one 8:1 MUX, two FF multipliers, two FF squares and three FF adders (XOR) into the critical path. These lead to substantial increase in the critical path delay and hence in the significantly reduced clock frequency. The critical path in our architecture uses three 2:1 MUX, one FF square (only one α^{2^1} circuit), one FF quad (only one α^{2^2} circuit), and one addition (XOR), apart from the FF multiplier. This ensures that the critical path delay is greatly reduced and we are able to achieve higher clock frequency.

5.4.1.2 Comparison with Binary Generic, Kobliz and Huff Curve Implementations:

As illustrated in Table 5.5, the proposed point multiplication over BEC is faster, consumes lesser area and achieves better area-time performance than binary Huff curve (BHC) [22] and binary generic curve (BGC) [145] designs. The proposed BEC implementation is slower when compared to the work in BGC [71, 82, 122] and binary Koblitz curve (BKC) [83] designs in terms of speed. This is due to the fact that most of these designs utilize pipelining and parallelism techniques to improve the working frequency and to reduce the clock cycles of their implementations. Compared to these designs, our implementation consumes lower area than [71] and larger area than [82, 83, 122]. The number of slices in our design are larger as we use a finite field of size 251 bits while [122] uses a finite field size of 233 bits. Furthermore, our design does not use any BRAM while [82, 83] utilize BRAM in their designs.

5.4.2 Side Channel Attack Resistance of BEC Point Multiplication

Side channel attacks are a class of attack that exploit information leaking from physical implementation of a cryptosystem [74]. Simple Power Analysis (SPA) attacks against point multiplication are based on variations in the power consumption of the cryptosystem during the execution of an operation. Any method which performs different set of operations depending on the value of a secret bit will leak information about that bit in terms of power consumption. We evaluated the side channel resistance of our implementation of Montogomery ladder with BEC mixed w-coordinates. Usually, the scalar n is the secret key during the computation of nP. We performed experiments to determine if SPA could be used to recover the scalar while the device is computing a point multiplication.

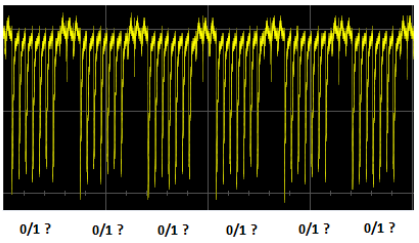

FIGURE 5.2: SPA result of BEC arithmetic

We used XC5VLX50-1FFG324 FPGA device of the Side-channel Attack Standard Evaluation Board (SASEBO-GII) [2, 134] to perform the SPA. Figure 5.2 shows a power trace during an example execution of $[n]P$ on BEC using unified formula. The representative figure is shown for n which has alternating 0's and 1's in its binary representation. The power consumption for executing unified formula in mixed w-coordinates for BEC is mostly due to the execution of 7 finite field multiplications as shown in Table 5.2. The power trace during point addition (PA) and point doubling (PD) operations consists of 7 peaks for executing multiplications. However, no observable power consumption difference, which depends on whether a bit of n is 0 or 1, is seen during the execution of unified addition formula. This leads to the conclusion that, as expected, our BEC point multiplication design (due to proper scheduling) is resistant against the SPA attack.

5.5 Architecture of ECMQV Protocol

In this section, we discuss FPGA implementation of the ECMQV protocol [79]. We use mixed w-coordinate based Montgomery ladder for performing the BEC arithmetic and uses PUF and TRNG for generating static (long term) and dynamic (short term) private keys, respectively. The description of the architecture and the utilized resources for our implementation are discussed next.

5.5.1 The ECMQV Protocol

The elliptic curve MQV [79] key agreement method is used to establish a shared secret between parties who already possess trusted copies of each other's static (long term) public keys. Both parties still generate dynamic public and private keys and then exchange public keys. However, upon receipt of the other party's public key, each party calculates a quantity called an *implicit signature* using their own private key and the other party's public key. The shared secret key is then generated from the *implicit signature*. The term implicit signature is used to indicate that the shared secrets do not agree if the other party's public key is not employed, thus giving implicit verification that the remote secret was indeed generated by the claimed party. An attempt at interception will fail as the shared secrets will not match because the adversary's private key is not linked to the trusted public key of either party.

Let E be an elliptic curve defined over the finite field $\mathbb{F}_q$ and G be a cyclic group of order n of elliptic curve points generated by P. The cofactor $h = \#E(\mathbb{F}_q)/n$ where $\#E(\mathbb{F}_q)$ is the order of the elliptic curve E and n is the order of the base point P. The two parties Alice and Bob have long term secret and public pairs (w_a, W_A) and (w_b, W_B) respectively. Note that $W_A = [w_a]P$ and $W_B = [w_b]P$. Further, for a given point $Z \in G$, let $\bar{Z}$ denote the string of $f = \lfloor(\log_2 n + 1)/2\rfloor$ most significant bits of the x-coordinate of Z. The protocol is described in Table 5.6. For more details, one may refer to [79].

TABLE 5.6: The Two-pass MQV Algorithm

Given: Two parties Alice and Bob with key pairs (w_a, W_A) and (w_b, W_B)
Output: A common key K
1. Alice chooses a random integer $r_A \in \{1, \ldots, n-1\}$, computes $R_A = [r_A]P$, and sends it to Bob
2. Bob chooses a random integer $r_B \in \{1, \ldots, n-1\}$, computes $R_B = [r_B]P$, and sends it to Alice
3. Alice performs a embedded key validation of R_B, and if the validation succeeds, computes her *implicit signature* $s_A = (r_A + \bar{R}_A w_a) \bmod n$ and computes $K = hs_A(R_B + \bar{R}_B W_B)$
4. Bob performs a embedded key validation of R_A, and if the validation succeeds, computes his *implicit signature* $s_B = (r_B + \bar{R}_B w_b) \bmod n$ and computes $K = hs_B(R_A + \bar{R}_A W_A)$
5. K is the shared secret key

5.5.2 Implementation Details

5.5.2.1 Static (Long-lived) Private Key Generation Using PUF

A long-term private/secret key is one that is stored somewhere, either on a computer disk, flash memory, or fuse. The key is intended to be used at multiple points in time. For example, using the key to encrypt some secret file today, and using it again to decrypt a secret next week. However, through such traditional storing key techniques, static secret keys are vulnerable to various kind of attacks and can be easily extracted or cloned. Further, maintaining such secrets in NVMs is expensive. Therefore, we derived static (long term) secrets from the PUFs rather than storing the secrets in non-volatile memories. PUFs can significantly increase physical security by generating secret keys that only exist in a digital form when a chip is powered on and running, which is unique, extremely hard or impossible to clone. In this work, we used our RS Latch based PUF (described in Section 2.3.2) for generating long term secret key. When compared to other our proposed PUFs (described in Chapter 2), the RS-LPUF using coarse PDL is more efficient in terms of speed and area in hardware and also achieves better performance. Moreover, the reliability of the proposed RS-LPUF responses is 100% stable. This is achieved by employing TMV with more votings. Therefore, there is no need for extra error-correction circuitry. Our RS Latch based PUF can be performed in 171 *msec* using 138 slices on a Virtex-4 FPGA and 142 *msec* using 91 slices on a Virtex-5 FPGA. In the protocol (ref. Table 5.6), the two parties Alice and Bob are generating long term secret keys (w_a, w_b) from their corresponding devices using the RS Latch based PUFs and then exchange their public keys (W_A, W_B) before performs steps of the MQV algorithm (ref. Table 5.6).

5.5.2.2 Ephemeral Private Key (Short-lived) Generation Using TRNG

Realizations of MQV protocol requires a random number generator, which is used to generate the ephemeral/session key. A ephemeral key is one that is not

intentionally stored, and is not re-creatable. Ephemeral keys are used only for communications protocols, never for storage purposes. In this work, we use RO based TRNGs (described in Section 4.3) for generating session keys such as r_A and r_B during the operation of the protocol (ref. Table 5.6). Our RO based TRNG can be performed in 7.8μs using 257 slices on a Virtex-4 FPGA and 6.4μs using 193 slices on a Virtex-5 FPGA.

5.5.2.3 The Initiator Circuit.

The two-pass version of ECMQV has two entities, namely, initiator (i.e., Alice) and responder (i.e., Bob). The initiator sends the first protocol message to the responder. The responder verifies the received protocol message and replies to the initiator. Finally, both the parties arrive at a common key (K) (ref. Table 5.6) through computation and communication.

The block diagram of the initiator circuit of ECMQV which performs *step* 1 and 3 of the MQV algorithm is shown in Figure 5.3. It consists of 4 block modules: *TRNG* (described in Section 4.3), *Point Multiplication Block* (described in Section 5.4.1), *Point Addition Block* (described in Section 5.3.3), and *Field Multiplication Block* (described in Section 5.2). First, the initiator circuit performs *step 1* of the MQV algorithm. A random integer r_A is generated using the *TRNG module* and the result is stored in register r_A. The *start* pin of the *TRNG Block* result from a control circuitry made of the *TRNG done* signal.

Then, the initiator circuit computes R_A using the *Point Multiplication Block* which reads values P_x, P_y and r_A from the coordinates of input points (i.e., P_x, P_y) and the register r_A. After the completion of the R_A computation, *Ready-*R_A signal is enabled and the resulting outputs (i.e., R_{Ax}, R_{Ay}) are sent to the responder (i.e., Bob) circuit. The *start* pin of the *Point Multiplication Block* results from a control circuitry made of the *PM selector* pins, *n selector* pins and the *multiplication done* signals.

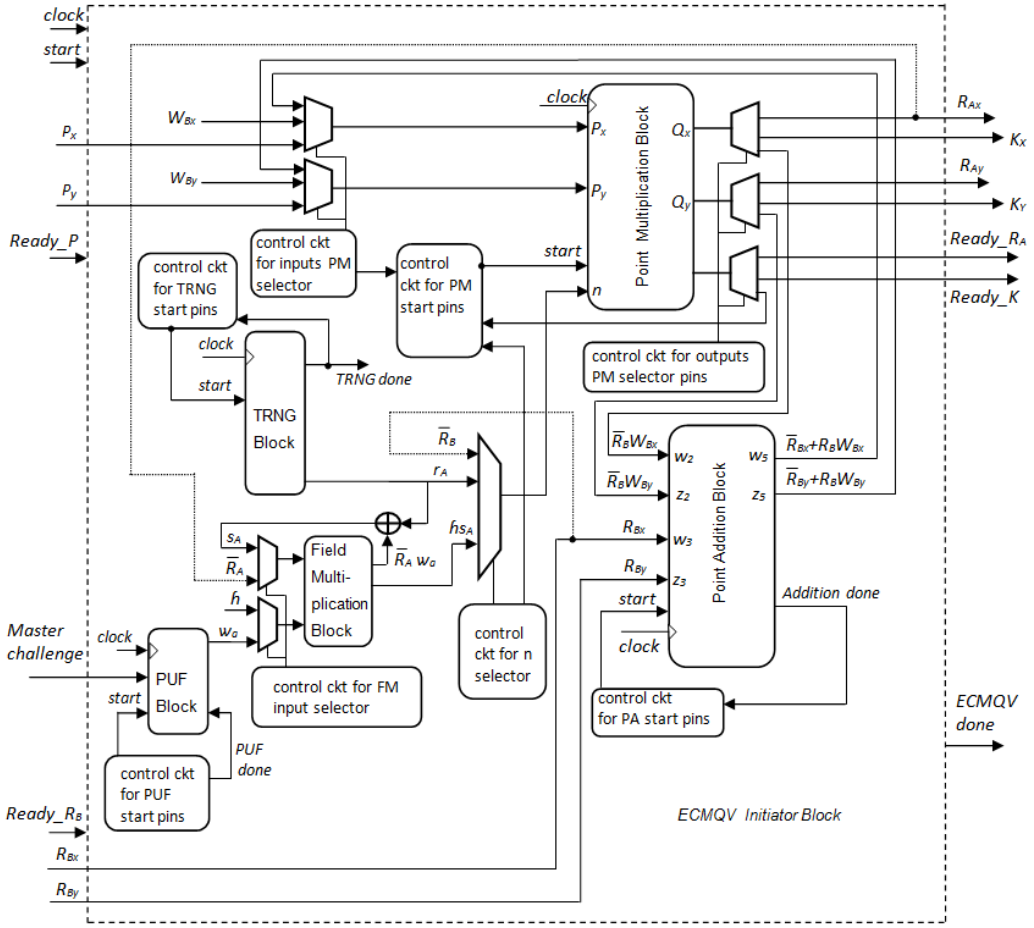

FIGURE 5.3: Architecture of ECMQV Protocol

Similarly, after the ready signal *Ready-*R_B is high (enabled) and the resulting outputs R_{Bx} and R_{By} are received from the responder after *step 2*, the initiator circuit performs *step* 3 of the MQV algorithm to compute s_A, hs_A and common secret key K. The circuit first computes $\bar{R_A}$ and $\bar{R_B}$ from R_{Ax} and R_{Bx}, and then the results are stored in registers $\bar{R_A}$ and $\bar{R_B}$. Then s_A (*implicit signature*) and hs_A are computed using *Field Multiplication module* with corresponding inputs and the results (s_A and hs_A) are stored in the corresponding registers. The control pins of the multiplexer select corresponding inputs to compute $\bar{R_A} w_a$ and hs_A. Next, the values WB_x, WB_y and $\bar{R_B}$ are fed to the *Point Multiplication Block* to compute $\bar{R_B} W_B$ and the resulting outputs are stored in the corresponding registers. Then, the values R_B and $\bar{R_B} W_B$ are fed to the *Point Addition Block* to perform the computation $(R_B + \bar{R_B} W_B)$, and the results are stored in the corresponding registers. The *start* pin of the *Point Addition Block* results from

a control circuitry made of the *Ready-*R_B signal, *Multiplication done* signal and *Addition done* signals. Finally, the resulting values hs_A, $(R_{Bx} + \bar{R_{Bx}}W_{Bx})$ and $(R_{By} + \bar{R_{By}}W_{By})$ are fed to the *Point Multiplication Block* to compute the shared secret key K (i.e., K_x, K_y). After computing the common key, the *ECMQV done* signal is enabled and the resulting outputs K_x and K_y are stored in K_x and K_y registers, respectively.

5.5.2.4 Input/Output.

Due to the choice of the finite field, the base point coordinates P_x, P_y and the R_B point coordinates such as R_{Bx}, R_{By} are all of length 251 bits. Hence, the total input pins required are at least $251 \times 4 = 1004$ and the total number of output pins required are at least $251 \times 4 = 1004$. Therefore a total of $1004 + 1004 = 2008$ I/O pins are needed for R_A (i.e., R_{Ax}, R_{Ay}) and K (i.e., K_x, K_y) coordinates. Because of the limitations of the I/O pins on the available FPGA, we send the input parameters through two 32-bit ports on the FPGA. Hence, to input two 251-bit numbers, 8 clock cycles are required for base point coordinates (i.e., P_x, P_y) and 8 clock cycles are required for R_B coordinates (i.e., R_{Bx}, R_{By}). Similarly, for the output display on two 32-bit ports, 8 clock cycles are required for two coordinates of R_A (i.e., R_{Ax}, R_{Ay}) computation and 8 clock cycles are required for two coordinates of the resultant point of a K (i.e., K_x, K_y) computation.

In Table 5.7, we present the FPGA implementation results of ECMQV protocol using BEC, PUF and TRNG. Notably, the ECMQV protocol can be performed in 180 *msec* using 32102 slices on Virtex-4 FPGA and 151 *msec* using 15495 slices on a Virtex-5 FPGA. Therefore, our proposed ECMQV design can be used as a cryptographic accelerator [28, 40, 61] for IoT security applications in data center security (data center is central to the IoT as it processes data from millions of devices), intelligent automation, smart grid security. As can be seen from Table 5.7, our ECMQV provides faster design and yields a better performance (AT). When compared to the ECDH implementation [40], our design requires more LUTs and slower in terms of speed. However, our design does not require

any RAMs while [40] requires 2.1 kB. The number of LUTs in our design are larger due to the fact that we use a finite field of size 251 bits while [40] uses a finite field of size 233 bits. Note that we did not employ any serialized/pipelining/parallelism techniques in the presented design. We believe these techniques are very important to achieve better performance in terms of the area-time product over other designs and it is a part of our future work.

TABLE 5.7: FPGA implementation results (post place and route) of ECMQV-Protocol

Work	**Field size**	**Area**			**Freq** (MHz)	**T** (*msec*)	**AT** $\times 10^3$	**FPGA Device**
		Slices	LUTs	FFs				
Proposed ECMQV	251	32102	60536	10923	111.28	180	10.89	Virtex-4
	251	15495	48356	10429	142.11	151	7.301	Virtex-5
ECDH [40]	233	—	35102	—	125	8.88	0.311	Virtex-5

5.6 Summary

In this chapter, we presented a high-performance implementation of ECMQV key agreement protocol using BEC, PUF and TRNG. We showed performance results of our design on Virtex-4 and Virtex-5 FPGAs. We first provided FPGA implemenation of the unified BEC formula in affine, projective and w-coordinates. Compact scheduling mechanisms were applied to improve the performance of point-addition and point-doubling computations. In particular, we proposed an FPGA design of point multiplication on BEC in w-coordinates and also analyzed its resistance against the SPA attack. According to our implementation results, this design is faster and achieves the smallest area in comparison to the previous FPGA implementations of BEC. Finally, we realized an FPGA implementation of ECMQV key agreement protocol using BEC, PUF and TRNG. Our implementation shows that the entire protocol can be performed in less than 180 *msec*. To the best of our knowledge, this is the first implementation of ECMQV key agreement protocol using binary Edwards curves, PUF and TRNG in FPGA till date.

■

Chapter 6

Conclusion

With the increasing deployment and necessity of security in IoT systems, there is a need of various security solutions. Despite tremendous achievements in this field in the recent decade, there is scope for further improvement in the cost, performance, and security level of these systems. The trade-off becomes significant since increasing number of interconnected devices within IoT regime can not support very strong security algorithms with large computational complexity. For such scenarios, a low area algorithm/primitive may be more suitable. Therefore, we must consider all aspects and find a trade-off between performance and cost. Thus, PUF and TRNG are potentially useful security primitives with strong advantages.

In this thesis, we have presented several novel designs and architectures for PUF and TRNG primitives and their practical usability in key agreement protocol. All of our contributions are FPGA compatible. The major contributions of the thesis are summarized ahead.

6.1 Summary of Contributions

Efficient PUF design and Implementations with Enhanced Performance

In chapters 2 and 3, efficient designs and implementations of RO-PUF, RS-PUF, A-PUF and Hybrid RS-Arbiter PUF with significantly enhanced performance have

been reported. Detailed statistical analysis of the obtained results convey that the incorporation of PDLs of FPGA LUTs significantly advances the architectures and implementations scheme of PUFs technology, and thereby enhances the uniqueness and randomness in the PUF responses. In addition, incorporation of the TMV scheme improves the reliability significantly in all our proposed PUFs. It has also been shown that the proposed designs yield the most area-efficient conventional, composite and hybrid PUFs reported so far. This feature makes it attractive for secure IoT applications because the proposed PUFs are resistant to temperature, supply voltage and correlated process variations.

Efficient TRNG design and Implementation

In chapter 4, we presented a RO based TRNG on FPGA. The programmable delay of FPGA LUTs has been used to achieve random jitter and to enhance the randomness. It has been demonstrated that the proposed implementation provides a very good area-throughput trade-off. In addition, the restart experiments show that the output of the behavior of the proposed TRNG is indistinguishable from a truly random source. The design achieves high entropy rate and successfully passes all NIST statistical tests. It can actually be inferred that the proposed design has the potential to be a good candidate for secure IoT applications.

Elliptic Curve based Key Agreement Protocol using PUF and TRNG

In chapter 5, we presented a practical design for an area efficient authenticated key agreement protocol between two IoT devices using BEC, PUF and TRNG. First, a novel hardware architecture of BEC point multiplication using mixed w-coordinates of the Montgomery laddering algorithm has been developed. Then, we presented an FPGA design of elliptic curve based key agreement protocol using PUF and TRNG. The obtained implementation results show that the proposed architecture yields a better performance when compared to the existing state-of-the-art.

6.2 Future Work

This thesis showed the efficiency and usability of hardware based security primitives on FPGAs. However, the research presented in this thesis seems to have raised more questions that it has answered. There are several lines of research arising from this work which can be pursued in the near future. This section presents some of these directions. The following ideas could be tested:

- A general question not solved by the PUF community yet is the exact entropy estimation of a PUF response. The current work on entropy estimation that was presented in chapters 2 and 3 was by using a set of 10 PUF instances based on the available boards. As an extension of the work, we plan to work on the entropy analysis the PUF designs with a much larger set of PUF instances (above 100 boards).

- The accelerated aging tests that were done in chapter 2 could be extended by more stress designs that would accelerate other types of aging mechanisms. Furthermore, a larger number of devices can be tested for more reliable results. Due to the relatively small climate chamber, only few devices were aged in this work.

- As an extension of the work, we plan to employ serialized/pipelining/parallelism techniques in the presented protocol design 5 to further achieve better performance in terms of the area-time product.

- Moreover, If PUFs and TRNGs are ever to be used in high security systems then they should resist side channel attacks (SCA). So far, very little has been done in this area. Clearly, this is an area in which a lot of progress can still be made.

■

Bibliography

[1] E. N. Allini, O. Petura, V. Fischer, and F. Bernard. Optimization of the PLL configuration in a PLL-based TRNG design. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1265–1270, March 2018.

[2] N. Nalla Anandakumar. SCA Resistance Analysis on FPGA Implementations of Sponge Based MAC-PHOTON. In *8th International Conference, SECITC 2015, Bucharest, Romania, June 11-12*, pages 69–86, Cham, 2015. Springer International Publishing.

[3] N. Nalla Anandakumar, M. Prem Laxman Das, Somitra K. Sanadhya, and Mohammad S. Hashmi. Reconfigurable Hardware Architecture for Authenticated Key Agreement Protocol Over Binary Edwards Curve. *ACM Trans. Reconfigurable Technol. Syst.*, 11(2):12:1–12:19, November 2018.

[4] N. Nalla Anandakumar, Mohammad S. Hashmi, and Somitra Kumar Sanadhya. Compact Implementations of FPGA-based PUFs with Enhanced Performance. In *30th International Conference on VLSI Design and 16th International Conference on Embedded Systems (VLSID)*, pages 161–166, Jan 2017.

[5] N. Nalla Anandakumar, Somitra Kumar Sanadhya, and Mohammad S. Hashmi. FPGA-Based True Random Number Generation Using Programmable Delays in Oscillator-Rings. *IEEE Transactions on Circuits and Systems II: Express Briefs*, pages 1–1, 2019.

[6] Amir Ardakani, Shahriar B. Shokouhi, and Arash Reyhani-Masoleh. Improving performance of FPGA-based SR-latch PUF using Transient Effect

Ring Oscillator and programmable delay lines. *Integration*, 62:371 – 381, 2018.

[7] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions. In *Advances in Cryptology – ASIACRYPT 2009*, pages 685–702, 2009.

[8] R. Azarderakhsh and A. Reyhani-Masoleh. Efficient FPGA Implementations of Point Multiplication on Binary Edwards and Generalized Hessian Curves Using Gaussian Normal Basis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(8):1453–1466, Aug 2012.

[9] R. Azarderakhsh and A. Reyhani-Masoleh. Parallel and High-Speed Computations of Elliptic Curve Cryptography Using Hybrid-Double Multipliers. *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1668–1677, June 2015.

[10] E. Barker and J. Kelsey. NIST Special Publication 800-90: Recommendation for random number generation using deterministic random bit generators (revised). Technical report, March 2007.

[11] Georg T. Becker and Raghavan Kumar. Active and Passive Side-Channel Attacks on Delay Based PUF Designs. *IACR Cryptology ePrint Archive*, 2014:287, 2014.

[12] A. Beirami and H. Nejati. A Framework for Investigating the Performance of Chaotic-Map Truly Random Number Generators. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(7):446–450, July 2013.

[13] M. J. Bellido, A. J. Acosta, M. Valencia, A. Barriga, and J. L. Huertas. Simple binary random number generator. *Electronics Letters*, 28(7):617–618, March 1992.

[14] Daniel J. Bernstein. Batch Binary Edwards. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 317–336. Springer, 2009.

[15] Daniel J. Bernstein, Tanja Lange, and Reza Rezaeian Farashahi. Binary Edwards Curves. In Elisabeth Oswald and Pankaj Rohatgi, editors, *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 244–265. Springer, 2008.

[16] Nathalie Bochard, Florent Bernard, Viktor Fischer, and Boyan Valtchanov. True-Randomness and Pseudo-Randomness in Ring Oscillator-Based True Random Number Generators. *Int. J. Reconfig. Comp.*, 2010:879281:1–879281:13, 2010.

[17] Eric Brier and Marc Joye. Weierstraß Elliptic Curves and Side-Channel Attacks. In *Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002*, pages 335–345, 2002.

[18] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically Uncloneable Functions in the Universal Composition Framework. In *Advances in Cryptology – CRYPTO 2011*, pages 51–70. Springer Berlin Heidelberg, 2011.

[19] S. Callegari, R. Rovatti, and G. Setti. Embeddable ADC-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos. *IEEE Transactions on Signal Processing*, 53(2):793–805, Feb 2005.

[20] Ayantika Chatterjee and Indranil Sengupta. FPGA Implementation of Binary Edwards Curve Using ternary Representation. In *Proceedings of the 21st Edition of the Great Lakes Symposium on VLSI*, GLSVLSI '11, pages 73–78. ACM, 2011.

[21] Ayantika Chatterjee and Indranil Sengupta. Design of a high performance Binary Edwards Curve based processor secured against side channel analysis. *INTEGRATION, the VLSI journal*, 45(3):331–340, 2012.

[22] Ayantika Chatterjee and Indranil Sengupta. High-Speed Unified Elliptic Curve Cryptosystem on FPGAs Using Binary Huff Curves. In *Progress in VLSI Design and Test - 16th International Symposium, VDAT 2012, Shibpur, India, July 1-4, 2012. Proceedings*, volume 7373 of *Lecture Notes in Computer Science*, pages 243–251. Springer, 2012.

[23] A. S. Chauhan, V. Sahula, and A. S. Mandal. Novel Randomized Biased Placement for FPGA Based Robust Random Number Generator with Enhanced Uniqueness. In *32nd International Conference on VLSI Design (VLSID)*, pages 353–358, Jan 2019.

[24] W. Che, M. Martinez-Ramon, F. Saqib, and J. Plusquellic. Delay model and machine learning exploration of a hardware-embedded delay PUF. In *IEEE Inter. Symposium on Hardware Oriented Security and Trust (HOST)*, pages 153–158, April 2018.

[25] W. Chen, W. Che, Z. Bi, J. Wang, N. Yan, X. Tan, J. Wang, H. Min, and J. Tan. A 1.04 W Truly Random Number Generator for Gen2 RFID tag. In *2009 IEEE Asian Solid-State Circuits Conference*, pages 117–120, Nov 2009.

[26] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet. A Self-Timed Ring Based True Random Number Generator, year=2013. In *2013 IEEE 19th International Symposium on Asynchronous Circuits and Systems*, pages 99–106, May.

[27] Abdelkarim Cherkaoui, Viktor Fischer, Laurent Fesquet, and Alain Aubert. A Very High Speed True Random Number Generator with Entropy Assessment. In *Cryptographic Hardware and Embedded Systems - CHES 2013*, pages 179–196. Springer Berlin Heidelberg, 2013.

[28] Aaron E. Cohen and Keshab K. Parhi. Fast Reconfigurable Elliptic Curve Cryptography Acceleration for $GF(2^m)$ on 32 bit Processors. *Signal Processing Systems*, 60(1):31–45, 2010.

[29] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren, editors. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman and Hall/CRC, 2005.

[30] Y. Cui, C. Wang, W. Liu, Y. Yu, M. O'Neill, and F. Lombardi. Low-cost configurable ring oscillator PUF with improved uniqueness. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 558–561, May 2016.

[31] F. Dan, Y. Xu, Z. Li, J. Wen, B. Liu, S. Chen, and B. Li. A Modeling Attack Resistant R-XOR APUF Based on FPGA. In *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*, pages 577–581, 2018.

[32] J. Delvaux. Machine-Learning Attacks on PolyPUFs, OB-PUFs, RPUFs, LHS-PUFs, and PUF–FSMs. *IEEE Transactions on Information Forensics and Security*, 14(8):2043–2058, Aug 2019.

[33] Jeroen Delvaux, Roel Peeters, Dawu Gu, and Ingrid Verbauwhede. A Survey on Lightweight Entity Authentication with Strong PUFs. *ACM Comput. Surv.*, 48(2):26:1–26:42, October 2015.

[34] Martin Deutschmann, Lejla Iriskic, Sandra-Lisa Lattacher, Mario Münzer, and Oleksandr Tomshchuk. A PUF Based Hardware Authentication Scheme for Embedded Devices. Technical report, Technical report, Technikon Forschungs-und Planungsgesellschaft mbH . . . , 2018.

[35] Markus Dichtl and Jovan Dj. Golić. High-Speed True Random Number Generation with Logic Gates Only. In *Cryptographic Hardware and Embedded Systems - CHES 2007*, pages 45–62. Springer Berlin Heidelberg, 2007.

[36] Whitfield Diffie and Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[37] Viktor Fischer and Miloš Drutarovský. True Random Number Generator Embedded in Reconfigurable Hardware. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 415–430. Springer Berlin Heidelberg, 2003.

[38] Apostolos P. Fournaris and Odysseas G. Koufopavlou. Affine Coordinate Binary Edwards Curve Scalar Multiplier with Side Channel Attack Resistance. In *2015 Euromicro Conference on Digital System Design, DSD 2015, Madeira, Portugal, August 26-28, 2015*, pages 431–437. IEEE Computer Society, 2015.

[39] Apostolos P. Fournaris, Nicolas Sklavos, and Christos Koulamas. A High Speed Scalar Multiplier for Binary Edwards Curves. In *Proceedings of the Third Workshop on Cryptography and Security in Computing Systems, CS2@HiPEAC, Prague, Czech Republic, January 20, 2016*, pages 41–44, 2016.

[40] Apostolos P. Fournaris, Ioannis Zafeirakis, Christos Koulamas, Nicolas Sklavos, and Odysseas G. Koufopavlou. Designing Efficient Elliptic Curve Diffie-Hellman Accelerators for Embedded Systems. In *2015 IEEE International Symposium on Circuits and Systems, ISCAS 2015, Lisbon, Portugal, May 24-27, 2015*, pages 2025–2028. IEEE, 2015.

[41] Fpga4fun. LFSR counters: LFSR reaches all possible states. https://www.fpga4fun.com/Counters3.html.

[42] Steve Gabriel. Altera Partners with Intrinsic-ID to Develop World's Most Secure High-End FPGA, 2015. https://newsroom.intel.com/news-releases/altera-partners-intrinsic-id-develop-worlds-secure-high-end-fpga.

[43] Fatemeh Ganji, Shahin Tajik, Fabian Fäßler, and Jean-Pierre Seifert. Having no mathematical model may not secure PUFs. *Journal of Cryptographic Engineering*, 7(2):113–128, Jun 2017.

[44] Y. Gao, G. Li, H. Ma, S. F. Al-Sarawi, O. Kavehei, D. Abbott, and D. C. Ranasinghe. Obfuscated challenge-response: A secure lightweight authentication mechanism for PUF-based pervasive devices. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 1–6, March 2016.

[45] Y. Gao, H. Ma, S. F. Al-Sarawi, D. Abbott, and D. C. Ranasinghe. PUF-FSM: A Controlled Strong PUF. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(5):1104–1108, May 2018.

[46] Blaise Gassend, Marten Van Dijk, Dwaine Clarke, Emina Torlak, Srinivas Devadas, and Pim Tuyls. Controlled Physical Random Functions and Applications. *ACM Trans. Inf. Syst. Secur.*, 10(4):3:1–3:22, January 2008.

[47] C. Gu, W. Liu, N. Hanley, R. Hesselbarth, and M. O'Neill. A Theoretical Model to Link Uniqueness and Min-Entropy for PUF Evaluations. *IEEE Transactions on Computers*, 68(2):287–293, 2019.

[48] Chongyan Gu, Neil Hanley, and Máire O'neill. Improved Reliability of FPGA-Based PUF Identification Generator Design. *ACM Trans. Reconfigurable Technol. Syst.*, 10(3):20:1–20:23, 2017.

[49] Hongxiang Gu, Teng Xu, and Miodrag Potkonjak. An Energy-Efficient PUF Design: Computing While Racing. In *Proc. of the Inter. Symp. on Low Power Electronics and Design*, ISLPED '16, pages 142–147. ACM, 2016.

[50] Jorge Guajardo, Sandeep S. Kumar, Geert-Jan Schrijen, and Pim Tuyls. FPGA Intrinsic PUFs and Their Use for IP Protection. In *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727, pages 63–80. Springer, 2007.

[51] Onur Günlü, Tasnad Kernetzky, Onurcan Iscan, Vladimir Sidorenko, Gerhard Kramer, and Rafael F. Schaefer. Secure and Reliable Key Agreement with Physical Unclonable Functions. *Entropy*, 20(5):340, 2018.

[52] T. Güneysu. True random number generation in block memories of reconfigurable devices. In *2010 International Conference on Field-Programmable Technology*, pages 200–207, Dec 2010.

[53] Bilal Habib, Jens-Peter Kaps, and Kris Gaj. Efficient SR-Latch PUF. In *Proceedings of the Applied Reconfigurable Computing - 11th International Symposium, ARC 2015, Bochum, Germany, April 13-17, 2015*, volume 9040, pages 205–216. Springer, 2015.

[54] Hisashi Hata and Shuichi Ichikawa. FPGA Implementation of Metastability-Based True Random Number Generator. *IEICE Transactions on Information and Systems*, E95.D(2):426–436, 2012.

[55] C. Herder, M. Yu, F. Koushanfar, and S. Devadas. Physical Unclonable Functions and Applications: A Tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, 2014.

[56] R. Hesselbarth, F. Wilde, C. Gu, and N. Hanley. Large scale RO PUF analysis over slice type, evaluation time and temperature on 28nm Xilinx FPGAs. In *IEEE Inter. Symp. on Hardware Oriented Security and Trust (HOST)*, pages 126–133, April 2018.

[57] D. E. Holcomb, W. P. Burleson, and K. Fu. Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Transactions on Computers*, 58(9):1198–1210, Sept 2009.

[58] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh. Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs. In *International Conference on Reconfigurable Computing and FPGAs*, pages 298–303, Dec 2010.

[59] T. Ignatenko, G. j. Schrijen, B. Skoric, P. Tuyls, and F. Willems. Estimating the Secrecy-Rate of Physical Unclonable Functions with the Context-Tree Weighting Method. In *IEEE International Symposium on Information Theory*, pages 499–503, July 2006.

[60] Southern Scientific Lab Instruments. Model EC Environmental Chamber, 2018. http://www.southernscientific.in/.

[61] Intel. IoT (Internet of Things) SoC and FPGA Solutions, 2017. https://www.intel.com/content/www/us/en/internet-of-things/products/programmable/overview.html.

[62] IoT Statistics: Number of Enabled Devices & Growth, 2019. https://ipropertymanagement.com/iot-statistics/.

[63] Tetsuya Izu and Tsuyoshi Takagi. A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks. In *Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002*, pages 280–296, 2002.

[64] Thomas Jennewein, Ulrich Achleitner, Gregor Weihs, Harald Weinfurter, and Anton Zeilinger. A fast and compact quantum random number generator. *Review of Scientific Instruments*, 71(4):1675–1680, Apr 2000.

[65] A. P. Johnson, R. S. Chakraborty, and D. Mukhopadhyay. An Improved DCM-Based Tunable True Random Number Generator for Xilinx FPGA. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 64(4):452–456, April 2017.

[66] Marc Joye and Sung-Ming Yen. The Montgomery Powering Ladder. In *Proceedings of the 4th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2002, Redwood Shores, CA, USA, August 13-15, 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 291–302. Springer, 2002.

[67] Benjamin Jun and Paul Kocher. The Intel random number generator. Cryptography Research, Inc., April 1999.

[68] H. Kang, Y. Hori, T. Katashita, M. Hagiwara, and K. Iwamura. Cryptographie key generation from PUF data using efficient fuzzy extractors. In *16th International Conference on Advanced Communication Technology*, pages 23–26, 2014.

[69] D. Karakoyunlu and B. Sunar. Differential template attacks on PUF enabled cryptographic devices. In *IEEE International Workshop on Information Forensics and Security*, pages 1–6, 2010.

[70] Stefan Katzenbeisser, Ünal Koçabas, Vincent van der Leest, Ahmad-Reza Sadeghi, Geert-Jan Schrijen, Heike Schröder, and Christian Wachsmann. Recyclable PUFs: Logically Reconfigurable PUFs. In *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 1, pages 374–389. Springer, 2011.

[71] Z. U. A. Khan and M. Benaissa. High-Speed and Low-Latency ECC Processor Implementation Over GF(2^m) on FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(1):165–176, Jan 2017.

[72] Sasan Khoshroo. Design and Evaluation of FPGA-based Hybrid Physically Unclonable Functions. In *Electronic Thesis and Dissertation Repository*, 2013.

[73] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[74] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology - CRYPTO '99, Santa Barbara, California, USA, August 15-19, 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[75] Philipp Koppermann, Fabrizio De Santis, Johann Heyszl, and Georg Sigl. Fast FPGA Implementations of Diffie-Hellman on the Kummer Surface of a Genus-2 Curve. Cryptology ePrint Archive, Report 2017/814, 2017. https://eprint.iacr.org/2017/814.pdf.

[76] Hugo Krawczyk. SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, pages 400–425. Springer Berlin Heidelberg, 2003.

[77] Hugo Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer, 2005.

[78] S. S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, and P. Tuyls. Extended abstract: The butterfly PUF protecting IP on every FPGA. In *IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 67–70, June 2008.

[79] Laurie Law, Alfred Menezes, Minghua Qu, Jerome A. Solinas, and Scott A. Vanstone. An Efficient Protocol for Authenticated Key Agreement. *Des. Codes Cryptography*, 28(2):119–134, 2003.

[80] J. W. Lee, Daihyun 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 applications. In *IEEE Symposium on VLSI Circuits. Digest of Technical Papers*, pages 176–179, June 2004.

[81] C. Li, Q. Wang, J. Jiang, and N. Guan. A metastability-based true random number generator on FPGA. In *2017 IEEE 12th International Conference on ASIC (ASICON)*, pages 738–741, Oct 2017.

[82] L. Li and S. Li. High-Performance Pipelined Architecture of Elliptic Curve Scalar Multiplication Over GF(2^m). *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(4):1223–1232, April 2016.

[83] L. Li and S. Li. High-Performance Pipelined Architecture of Point Multiplication on Koblitz Curves. *IEEE Transactions on Circuits and Systems II: Express Briefs*, PP(99):1–1, 2017.

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

[85] D. Liu, Z. Liu, L. Li, and X. Zou. A Low-Cost Low-Power Ring Oscillator-Based Truly Random Number Generator for Encryption on Smart Cards. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 63(6):608–612, June 2016.

[86] Weiqiang Liu, Lei Zhang, Zhengran Zhang, Chongyan Gu, Chenghua Wang, Maire O'neill, and Fabrizio Lombardi. XOR-Based Low-Cost Reconfigurable PUFs for IoT Security. *ACM Trans. Embed. Comput. Syst.*, 18(3):25:1–25:21, April 2019.

[87] Julio López and Ricardo Dahab. Fast Multiplication on Elliptic Curves over GF(2^{m}) without Precomputation. In *Proceedings of the First International Workshop on Cryptographic Hardware and Embedded Systems (CHES'99), Worcester, MA, USA, August 12-13, 1999*, volume 1717 of *Lecture Notes in Computer Science*, pages 316–327. Springer, 1999.

[88] Qingqing Ma, Chongyan Gu, Neil Hanley, Chenghua Wang, Weiqiang Liu, and Máire O'Neill. A Machine Learning Attack Resistant multi-PUF Design on FPGA. In *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, ASPDAC '18, pages 97–104. IEEE Press, 2018.

[89] Takanori Machida, Dai Yamamoto, Mitsugu Iwamoto, and Kazuo Sakiyama. A New Arbiter PUF for Enhancing Unpredictability on FPGA. *The Scientific World Journal*, 2015(13), 2015.

[90] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Intrinsic PUFs from Flip-flops on Reconfigurable Devices. In *3rd Benelux Workshop Information and System Security*, page 17, 2008.

[91] Roel Maes, Anthony Van Herrewege, and Ingrid Verbauwhede. *PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator*, pages 302–319. Springer Berlin Heidelberg, CHES 2012.

[92] Ahmed Mahmoud, Ulrich Rührmair, Mehrdad Majzoobi, and Farinaz Koushanfar. Combined Modeling and Side Channel Attacks on Strong PUFs. *IACR Cryptology ePrint Archive*, 2013:632, 2013.

[93] A. Maiti and P. Schaumont. The Impact of Aging on a Physical Unclonable Function. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(9):1854–1864, Sep. 2014.

[94] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. *A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions*, pages 245–267. Springer New York, New York, NY, 2013.

[95] Abhranil Maiti, Raghunandan Nagesh, Anand Reddy, and Patrick Schaumont. Physical Unclonable Function and True Random Number Generator: A Compact and Scalable Implementation. In *Proceedings of the 19th ACM Great Lakes Symposium on VLSI*, GLSVLSI, pages 425–428. ACM, 2009.

[96] Abhranil Maiti and Patrick Schaumont. Improved Ring Oscillator PUF: An FPGA-friendly Secure Primitive. *J. Cryptology*, 24:375–397, 2011.

[97] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Lightweight secure PUFs. In *2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 670–673, Nov 2008.

[98] Mehrdad Majzoobi, Farinaz Koushanfar, and Srinivas Devadas. FPGA PUF using programmable delay lines. In *IEEE International Workshop on Information Forensics and Security, WIFS*, pages 1–6. IEEE, 2010.

[99] Mehrdad Majzoobi, Farinaz Koushanfar, and Srinivas Devadas. FPGA-Based True Random Number Generation Using Circuit Metastability with Adaptive Feedback Control. In *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917, pages 17–32. Springer, 2011.

[100] C. Marchand, L. Bossuet, U. Mureddu, N. Bochard, A. Cherkaoui, and V. Fischer. Implementation and Characterization of a Physical Unclonable Function for IoT: A Case Study With the TERO-PUF. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):97–109, Jan 2018.

[101] G. Marsaglia. Diehard: A Battery of Tests of Randomness, 1996.

[102] S. K. Mathew, S. K. Satpathy, M. A. Anders, H. Kaul, S. K. Hsu, A. Agarwal, G. K. Chen, R. J. Parker, R. K. Krishnamurthy, and V. De. 16.2 A 0.19pJ/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm CMOS. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 278–279, 2014.

[103] Thomas McGrath, Ibrahim E Bagci, Zhiming M Wang, Utz Roedig, and Robert J Young. A Puf Taxonomy. *Applied Physics Reviews*, 6(1):011303, 2019.

[104] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[105] D. Merli, J. Heyszl, B. Heinz, D. Schuster, F. Stumpf, and G. Sigl. Localized electromagnetic analysis of RO PUFs. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 19–24, June 2013.

[106] Victor S. Miller. Use of Elliptic Curves in Cryptography. In *Proceedings of the Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985*, pages 417–426, 1985.

[107] Mingze Gao, Khai Lai, and Gang Qu. A highly flexible ring oscillator PUF. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2014.

[108] M. S. Mispan, H. Su, M. Zwolinski, and B. Halak. Cost-efficient design for modeling attacks resistant PUFs. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 467–472, March 2018.

[109] Peter L. Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, 48:243–264, 1987.

[110] A. Mutlu, K. J. Le, M. Celik, D. Tsien, G. Shyu, and L. Yeh. An exploratory study on statistical timing analysis and parametric yield optimization. pages 677–684, March 2007.

[111] D. Nedospasov, J. Seifert, C. Helfmeier, and C. Boit. Invasive PUF Analysis. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 30–38, Aug 2013.

[112] Karsten Nohl, David Evans, Starbug Starbug, and Henryk Plötz. Reverse-engineering a Cryptographic RFID Tag. In *Proceedings of the 17th Conference on Security Symposium*, pages 185–193. USENIX Association, 2008.

[113] Nafisa Noor and Helena Silva. *Phase Change Memory for Physical Unclonable Functions*, pages 59–91. Springer Singapore, Singapore, 2020.

[114] H. Nyquist. Thermal Agitation of Electric Charge in Conductors. *Phys. Rev.*, 32:110–113, Jul 1928.

[115] F. Pareschi, G. Setti, and R. Rovatti. A Fast Chaos-based True Random Number Generator for Cryptographic Applications. In *2006 Proceedings of the 32nd European Solid-State Circuits Conference*, pages 130–133, Sep. 2006.

[116] C. S. Petrie and J. A. Connelly. A noise-based IC random number generator for applications in cryptography. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(5):615–621, May 2000.

[117] O. Petura, U. Mureddu, N. Bochard, V. Fischer, and L. Bossuet. A survey of ais-20/31 compliant trng cores suitable for fpga devices. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–10, Aug 2016.

[118] M. Potkonjak and V. Goudar. Public Physical Unclonable Functions. *Proceedings of the IEEE*, 102(8):1142–1156, Aug 2014.

[119] Graham Prophet. Xilinx to Add PUF Security to Zynq Devices, 2016. https://www.eenewseurope.com/news/xilinx-add-puf-security-zynq-devices-0.

[120] Pappu Srinivas Ravikanth. Physical one-way functions. In *PH.D. THESIS*. Massachusetts Institute of Technology, Mar 2001.

[121] Chester Rebeiro and Debdeep Mukhopadhyay. High Speed Compact Elliptic Curve Cryptoprocessor for FPGA Platforms. In *Proceedings of 9th International Conference on Cryptology in India, Progress in Cryptology - INDOCRYPT 2008, Kharagpur, India, December 14-17, 2008*, pages 376–388, 2008.

[122] Chester Rebeiro, Sujoy Sinha Roy, and Debdeep Mukhopadhyay. Pushing the Limits of High-Speed GF(2^m) Elliptic Curve Scalar Multiplication on FPGAs. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, pages 494–511, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[123] S. Robson, B. Leung, and G. Gong. Truly Random Number Generator Based on a Ring Oscillator Utilizing Last Passage Time. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 61(12):937–941, Dec 2014.

[124] A. Rodriguez-Vazquez, S. Espejo-Meana, J. L. Huertas, and J. D. Martin. Analog Building Blocks for Noise and Truly Random Number Generation in CMOS VLSI. In *ESSCIRC '90: Sixteenth European Solid-State Circuits Conference*, pages 225–228, Sep. 1990.

[125] Debapriya Basu Roy, Shivam Bhasin, Ivica Nikolić, and Debdeep Mukhopadhyay. Combining PUF with RLUTs: A Two-party Pay-per-device IP Licensing Scheme on FPGAs. *ACM Trans. Embed. Comput. Syst.*, 18(2):12:1–12:22, March 2019.

[126] Ulrich Rührmair, Heike Busch, and Stefan Katzenbeisser. *Strong PUFs: Models, Constructions, and Security Proofs*, pages 79–96. Springer Berlin Heidelberg, 2010.

[127] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling Attacks on Physical Unclonable Functions. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 237–249. ACM, 2010.

[128] U. Rührmair and M. van Dijk. PUFs in Security Protocols: Attack Models and Security Evaluations. In *2013 IEEE Symposium on Security and Privacy*, pages 286–300, May 2013.

[129] ID Quantique SA. Quantis AIS 31 certified random number generator (RNG), Accessed on October, 2019. https://www.idquantique.com/random-number-generation/products/quantis-ais-31/.

[130] D. P. Sahoo, R. S. Chakraborty, and D. Mukhopadhyay. Towards Ideal Arbiter PUF Design on Xilinx FPGA: A Practitioner's Perspective. In *Euromicro Conference on Digital System Design*, pages 559–562, 2015.

[131] Durga Prasad Sahoo, Sayandeep Saha, Debdeep Mukhopadhyay, Rajat Subhra Chakraborty, and Hitesh Kapoor. Composite PUF: A New Design Paradigm for Physically Unclonable Functions on FPGA. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2014.

[132] Augustin Sarr. *Authenticated Key Agreement Protocols: Security Models, Analyses, and Designs.* Theses, Université Joseph-Fourier - Grenoble I, October 2010.

[133] Augustin P. Sarr, Philippe Elbaz-Vincent, and Jean-Claude Bajard. A Secure and Efficient Authenticated Diffie-Hellman Protocol. Cryptology ePrint Archive, Report 2009/408, 2009. http://eprint.iacr.org/.

[134] Side-channel attack standard evaluation board, 2012. http://satoh.cs.uec.ac.jp/SASEBO/en/board/sasebo-g2.html.

[135] Werner Schindler and Wolfgang Killmann. A proposal for: Functionality classes for random number generators, 2011.

[136] A. Sghaier, M. Zeghid, and M. Machhout. Fast hardware implementation of ECDSA signature scheme. In *International Symposium on Signal, Image, Video and Communications (ISIVC 2016)*, pages 343–348, Nov 2016.

[137] Masaaki Shirase and Yasushi Hibino. An architecture for elliptic curve cryptograph computation. *SIGARCH Computer Architecture News*, 33(1):124–133, 2005.

[138] B. Srinivasu, P. Vikramkumar, A. Chattopadhyay, and K. Lam. CoLPUF : A Novel Configurable LFSR-based PUF. In *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 358–361, Oct 2018.

[139] Martijn Stam. On Montgomery-Like Representations for Elliptic Curves over GF(2^k). In Yvo Desmedt, editor, *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography - PKC 2003, Miami, FL, USA, January 6-8, 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 240–253. Springer, 2003.

[140] A. Stanciu, M. N. Cirstea, and F. D. Moldoveanu. Analysis and Evaluation of PUF-Based SoC Designs for Security Applications. *IEEE Transactions on Industrial Electronics*, 63(9):5699–5708, Sep. 2016.

[141] André Stefanov, Nicolas Gisin, Olivier Guinnard, Laurent Guinnard, and Hugo Zbinden. Optical quantum random number generator. *Journal of Modern Optics*, 47(4):595–598, 2000.

[142] G. E. Suh, C. W. O'Donnell, and S. Devadas. Aegis: A Single-Chip Secure Processor. *IEEE Design Test of Computers*, 24(6):570–580, Nov 2007.

[143] G. Edward Suh and Srinivas Devadas. Physical Unclonable Functions for Device Authentication and Secret Key Generation. In *Proceedings of the 44th Design Automation Conference, DAC 2007, USA, June 4-8, 2007*, pages 9–14. IEEE, 2007.

[144] B. Sunar, W. J. Martin, and D. R. Stinson. A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks. *IEEE Transactions on Computers*, 56(1):109–119, Jan 2007.

[145] G. D. Sutter, J. P. Deschamps, and J. L. Imana. Efficient Elliptic Curve Point Multiplication Using Digit-Serial Binary Field Operations. *IEEE Transactions on Industrial Electronics*, 60(1):217–225, Jan 2013.

[146] S. Tajik, H. Lohrke, F. Ganji, J. Seifert, and C. Boit. Laser Fault Attack on Physically Unclonable Functions. In *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 85–96, Sep. 2015.

[147] Shahin Tajik, Enrico Dietz, Sven Frohmann, Helmar Dittrich, Dmitry Nedospasov, Clemens Helfmeier, Jean-Pierre Seifert, Christian Boit, and Heinz-Wilhelm Hübers. Photonic Side-Channel Analysis of Arbiter PUFs. *Journal of Cryptology*, 30(2):550–571, 2017.

[148] T. Tanamoto, S. Yasuda, S. Takaya, and S. Fujita. Physically Unclonable Function Using an Initial Waveform of Ring Oscillators. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 64(7):827–831, July 2017.

[149] F. Tehranipoor, N. Karimian, W. Yan, and J. A. Chandy. DRAM-Based Intrinsic Physically Unclonable Functions for System-Level Security and

Authentication. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(3):1085–1097, March 2017.

[150] Assia Tria and Hamid Choukri. *Invasive Attacks*, pages 623–629. Springer US, 2011.

[151] Vincent van der Leest, Geert-Jan Schrijen, Helena Handschuh, and Pim Tuyls. Hardware Intrinsic Security from D Flip-flops. In *Proceedings of the Fifth ACM Workshop on Scalable Trusted Computing*, pages 53–62. ACM, 2010.

[152] John von Neumann. Various techniques used in connection with random digits. In *Monte Carlo Method*, pages 36–38. National Bureau of Standards Applied Mathematics Series, 12, 1951.

[153] S. Vyas, N. K. Dumpala, R. Tessier, and D. E. Holcomb. Improving the efficiency of PUF-based key generation in FPGAs using variation-aware placement. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4, Aug 2016.

[154] John walker. HotBits: Genuine random numbers, generated by radioactive decay , 1996.

[155] S. Walker and S. Foo. Evaluating metastability in electronic circuits for random number generation. In *Proceedings IEEE Computer Society Workshop on VLSI 2001. Emerging Technologies for VLSI Systems*, pages 99–101, April 2001.

[156] A. Wild, G. T. Becker, and T. Güneysu. A fair and comprehensive large-scale analysis of oscillation-based PUFs for FPGAs. In *27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–7, Sep. 2017.

[157] Nils Wisiol, Christoph Graebnitz, Marian Margraf, Manuel Oswald, Tudor A. A. Soroceanu, and Benjamin Zengin. Why Attackers Lose: Design and Security Analysis of Arbitrarily Large XOR Arbiter PUFs. In

PROOFS@CHES 2017, volume 49 of *EPiC Series in Computing*, pages 68–83, 2017.

[158] K. Wold and S. Petrović. Security properties of oscillator rings in true random number generators. In *2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, pages 145–150, April 2012.

[159] K. Wold and C. H. Tan. Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings. In *Int. Conf. on Reconfigurable Computing and FPGAs*, pages 385–390, Dec 2008.

[160] Xilinx. Spartan-6 Libraries Guide for HDL Designs, 2009. https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/spartan6_hdl.pdf.

[161] H. Xu, D. Perenzoni, A. Tomasi, and N. Massari. A 16 × 16 Pixel Post-Processing Free Quantum Random Number Generator Based on SPADs. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 65(5):627–631, May 2018.

[162] Xiaolin Xu, Ulrich Rührmair, Daniel E. Holcomb, and Wayne Burleson. Security Evaluation and Enhancement of Bistable Ring PUFs. In *Radio Frequency Identification*, pages 3–16. Springer, 2015.

[163] M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle. True random bit generation from a double-scroll attractor. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(7):1395–1404, July 2004.

[164] Dai Yamamoto, Kazuo Sakiyama, Mitsugu Iwamoto, Kazuo Ohta, Masahiko Takenaka, and Kouichi Itoh. Variety enhancement of PUF responses using the locations of random outputting RS latches. *Journal of Cryptographic Engineering*, 3(4):197–211, Nov 2013.

[165] W. Yan, C. Jin, F. Tehranipoor, and J. A. Chandy. Phase calibrated ring oscillator puf design and implementation on fpgas. In *27th International*

Conference on Field Programmable Logic and Applications (FPL), pages 1–8, Sep. 2017.

[166] Bohan Yang, Vladimir Rožic, Miloš Grujic, Nele Mentens, and Ingrid Verbauwhede. Es-trng: A high-throughput, low-area true random number generator based on edge sampling. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018:267–292, Aug. 2018.

[167] K. Yang, D. Fick, M. B. Henry, Y. Lee, D. Blaauw, and D. Sylvester. 16.3 A 23Mb/s 23pJ/b fully synthesized true-random-number generator in 28nm and 65nm CMOS. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 280–281, Feb 2014.

[168] Kun Yang, Domenic Forte, and Mark M. Tehranipoor. CDTA: A Comprehensive Solution for Counterfeit Detection, Traceability, and Authentication in the IoT Supply Chain. *ACM Trans. Des. Autom. Electron. Syst.*, 22(3):42:1–42:31, April 2017.

[169] J. Ye, Y. Hu, and X. Li. RPUF: Physical Unclonable Function with Randomized Challenge to resist modeling attack. In *2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, pages 1–6, Dec 2016.

[170] H. Yu, P. H. W. Leong, and Q. Xu. An FPGA Chip Identification Generator Using Configurable Ring Oscillators. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(12):2198–2207, Dec 2012.

[171] J. Zhang, X. Tan, Y. Zhang, W. Wang, and Z. Qin. Frequency Offset-Based Ring Oscillator Physical Unclonable Function. *IEEE Transactions on Multi-Scale Computing Systems*, 4:711–721, 2018.

[172] Ji-Liang Zhang, Gang Qu, Yong-Qiang Lv, and Qiang Zhou. A Survey on Silicon PUFs and Recent Advances in Ring Oscillator PUFs. *Journal of Computer Science and Technology*, 29(4):664–678, Jul 2014.

[173] Jiliang Zhang and Lu Wan. CMOS: dynamic multi-key obfuscation structure for strong pufs. *CoRR*, abs/1806.02011, 2018.

[174] Y. Zheng, Y. Cao, and C. Chang. A PUF-Based Data-Device Hash for Tampered Image Detection and Source Camera Identification. *IEEE Transactions on Information Forensics and Security*, 15:620–634, 2020.