5-21-2013 12:00 AM

Design and Evaluation of FPGA-based Hybrid Physically Unclonable Functions

Sasan Khoshroo, *The University of Western Ontario*

Supervisor: Arash Reyhani-Masoleh, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering

DESIGN AND EVALUATION OF FPGA-BASED HYBRID PHYSICALLY UNCLONABLE FUNCTIONS

by

Sasan Khoshroo

Graduate Program in Electrical and Computer Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering Science

The School of Graduate and Postdoctoral Studies
Western University
London, Ontario, Canada

Abstract

A Physically Unclonable Function (PUF) is a new and promising approach to provide security for physical systems and to address the problems associated with traditional approaches. One of the most important performance metrics of a PUF is the randomness of its generated response, which is presented via uniqueness, uniformity, and bit-aliasing. In this study, we implement three known PUF schemes on an FPGA platform, namely SR Latch PUF, Basic RO PUF, and Anderson PUF. We then perform a thorough statistical analysis on their performance. In addition, we propose the idea of the Hybrid PUF structure in which two (or more) sources of randomness are combined in a way to improve randomness. We investigate two methods in combining the sources of randomness and we show that the second one improves the randomness of the response, significantly. For example, in the case of combining the Basic RO PUF and the Anderson PUF, the Hybrid PUF uniqueness is increased nearly 8%, without any pre-processing or post-processing tasks required.

Two main categories of applications for PUFs have been introduced and analyzed: authentication and secret key generation. In this study, we introduce another important application for PUFs. In fact, we develop a secret sharing scheme using a PUF to increase the information rate and provide cheater detection capability for the system. We show that, using the proposed method, the information rate of the secret sharing scheme will improve significantly.

Keywords: FPGA, Hardware Security, Information Rate, Message Authentication Code, Physically Unclonable Functions, Robust Secret Sharing, VHDL

Acknowlegement

The author would like to thank Dr. Arash Reyhani-Masoleh and Dr. Shahriar B. Shokouhi for their valuable support and help.

Contents

List of Figures

List of Tables

List of Abbreviations

BCH	Bose-Chaudhuri-Hochquenghen
CLB	Configurable Logic Block
CPUF	Controlled Physically Unclonable Function
CRP	Challenge-Response Pair
ECC	Error Correcting Code
EEPROM	Electrically Erasable Programmable Read-Only Memory
FPGA	Field Programmable Gate Array
HD	Hamming Distance
HW	Hamming Weight
LUT	Lookup Table
MAC	Message Authentication Code
PAR	Place And Route
PUF	Physically Unclonable Function
RO	Ring Oscillator
ROM	Read-Only Memory
RTL	Register-Transfer Level
SRAM	Static Random-Access Memory

List of Notations

b	The ECC syndrome bit-size
q	The secret key bit-size
r	The PUF response
c	The PUF challenge
R	The controlled PUF output (response)
C	The controlled PUF input (challenge)
A^*	The additional arbiter in the feed-forward Arbiter PUF
N	Number of PUF units in the challenge-response system
u	Number of inverters in the RO feedback loop
Ref_Counter	The reference counter in the RO PUF design
Ref_Clock	The reference clock of the RO PUF design
γ	Number of ring oscillators in each RO block in the 1-out-of-γ masking scheme
G_i	The i-th select bit in Maiti's configurable RO
bx_i	The i-th select bit in Xin's configurable RO
sel_i	The i-th select bit in Xin's configurable RO
d	Number of different supply voltage values in the multi-voltage RO PUF design
α	Number of ROs in each RO batch in Maes's RO PUF
β	Number of RO batches in Maes's RO PUF
h	Number of response bits generated in Maes's RO PUF after each β simultaneous frequency measurements
h'	The Lehmer-Gray Encoder output bit-size in Maes's RO PUF
$N2$	The top carry chain multiplexer output in the Anderson PUF
n	Number of players in the secret sharing scheme
p_i	The i-th player in the secret sharing scheme
P	The set of all players in the secret sharing system
s	The secret to be shared using the secret sharing scheme
s'	The reconstructed secret using the secret sharing scheme
k	The threshold value in Shamir's secret sharing scheme
ρ	Information rate of a secret sharing scheme
S	The set of secrets to be shared using the secret sharing scheme
S_p	The set of the shares given to the player p
t	Number of fake shares submitted by dishonest players

key_j	The authentication key received by the player p_j in the Rabin and Ben-Or scheme
λ	Robustness measure of a robust secret sharing scheme
Ω	Lower bound function
$\mathbb{F}$	Finite field
y	A random element chosen from the finite field $\mathbb{F}$
z	An element of the finite field $\mathbb{F}$
Γ	An authorized set
$\mathcal{H}$	A monotone access structure
$\oplus$	Bit-wise XOR
W	Helper data
$\|x\|$	The bit-size of x
$\max$	Maximum function
$\log$	Logarithm function
avg	Average function
l	Number of players in an authorized set
$\|\|$	Concatenation
$flag_i$	The cheater flag corresponding to the player p_i
m	Number of response samples
a	Number of response bits
g	Number of PUF instances

Chapter 1

Introduction

The traditional approach to prevent passive physical attacks such as counterfeiting, cloning, reverse engineering and the insertion of malicious components include cryptographic primitives such as encryption/decryption algorithms, digital signature schemes, and authentication codes. The problem with these types of security measures is that, they rely on the protection of the secret keys which are stored in non-volatile memory such as EEPROMs or fuses. The sensitive data stored in such memory can be read out directly through invasive attacks [1]. To counter this issue, expensive protective coatings are applied but still, the devices are vulnerable to sophisticated physical attacks.

Therefore, physically unclonable functions (PUFs) are introduced to address such problems. A PUF is a *challenge-response* primitive which is used in a physical system to provide the required security measures [1, 13]. Instead of storing the secret key in a memory, it can be intrinsically generated by the PUF. In fact, a PUF generates a *response* to a given *challenge*. The idea behind the PUF is that, the output response is totally random and unpredictable. It is also unique for different instances, even if the two instances are exactly the same and use the exact same components. This is because the PUF response depends solely on the unique and random characteristics of physical devices, such as gate delays. In fact, the very important feature of a PUF is its unclonability, i.e., even if an attacker has access to the circuit and builds an exact same copy of it using the same components, the

response of the new device to a given challenge would be different from that of the original device to the same challenge.

1.1 PUF Constructions

A variety of PUF constructions have been introduced during the past ten years [14]. Non-electrical PUFs include Optical PUFs [15, 16], Acoustical PUFs [17], and Coating PUFs [18]. Optical PUFs use an optical micro-structure which is built by mixing microscopic refractive glass spheres on a tiny transparent epoxy plate [15]. This micro-structure is called a token. When a laser beam is applied to the token, it will generate a random pattern that can be further processed to produce the PUF response. The pattern generated by the token will substantially change even with a slight change in the laser beam, in terms of its wavelength, angle, or focal distance. Acoustical PUFs are built upon the acoustical delay lines. An alternating electrical signal is transformed to a mechanical vibration using a transducer. This vibration propagates through a solid medium (acoustical line) which includes random scatterers. At the other end of the line, the wave is transformed back to an electrical signal. The produced electrical signal has unique properties which depend on the random physical characteristics of the acoustical line. Therefore, this electrical signal can be used as the unique PUF response. In Coating PUFs, a protective coating material is inserted onto the device using random dielectric particles which have random properties in size, shape and location. In fact, in Coating PUFs, a random element is purposely inserted into the device in order to provide more strength against physical attacks. Therefore, Coating PUFs are different from other *intrinsic* PUFs in which the random element is intrinsic to the device.

In addition, Electrical PUFs are categorized as Memory-based PUFs and Delay-based PUFs. Memory-based PUFs include SRAM, SR Latch, Flip-Flop, Butterfly, and Buskeeper PUFs [19–24]. The idea behind memory-based PUFs is to bring a bistable memory element (which can contain only 1 bit of information) into its

metastable state where it is not clear to which stable state it will fall back. This settling state is totally random and unpredictable for different memory elements due to random physical variations.

Moreover, delay-based PUFs consist of Arbiter PUFs [2, 25], Ring-Oscillator PUFs (RO PUFs) [3,8,13,26–28], and Glitch PUFs [8,9,29]. The random element used in delay-based PUFs to produce response bits is the gate delay. For example, in Ring Oscillator PUFs, two identical ring oscillators produce two clocks with different frequencies. The frequency of each RO depends on the delay of the inverters used in the feedback loop of the ring oscillator. Therefore, the frequencies can be compared to each other to produce one response bit, based on which ring oscillator is faster. In this study, we focus on electrical PUFs and discuss their characteristics in more details in Chapter 2.

1.2 PUF Performance Metrics and Properties

Some of the important performance metrics of PUFs include reliability, uniqueness, uniformity, and bit-aliasing [6, 30]. Reliability of a PUF is a measure of its reproducibility. The reliability of an ideal PUF is 100%, i.e., the PUF generates the same response to a given challenge at different instances of time and under different environmental conditions. Uniqueness is a measure of *inter-distance* variations of the response bits among different PUF instances. In other words, if a specific challenge is applied at the same time and under the same conditions to two identical PUF instances, the response of the two PUFs should be different. Ideally, this value should be 50%. Uniqueness is one of the most important features of PUFs and represents the randomness of the PUF response bits [30]. Also, uniformity of a PUF measures the ratio between the number of 1's and the total number of response bits. Uniformity of an ideal PUF is 50% meaning that, 50% of the response bits are 1 and 50% are 0, and therefore, the PUF response does not have a biased behavior towards a specific bit value. Another important factor of a PUF which also represents the randomness of the PUF response is bit-aliasing.

Bit-aliasing of a given bit position in the PUF response is its percentage Hamming Weight (HW) across several PUF instances. Again, this value should be ideally 50% for all response bit positions. The definition of these properties along with their formulations are provided in details in Chapter 3.

1.3 PUF Applications

Two main applications have been introduced for PUFs: device authentication and secret key generation [1]. Authentication is performed in two steps. First, in the enrollment phase, a trusted party (authentication authority) records a number of challenge-response pairs (CRPs) in a database. Then, in the verification phase, a random challenge chosen from the database is applied to the PUF under verification and if the generated response is ”close enough” to the recorded response, the PUF is verified to be authentic. Figure 1.1 shows a general PUF-based authentication scheme [1].

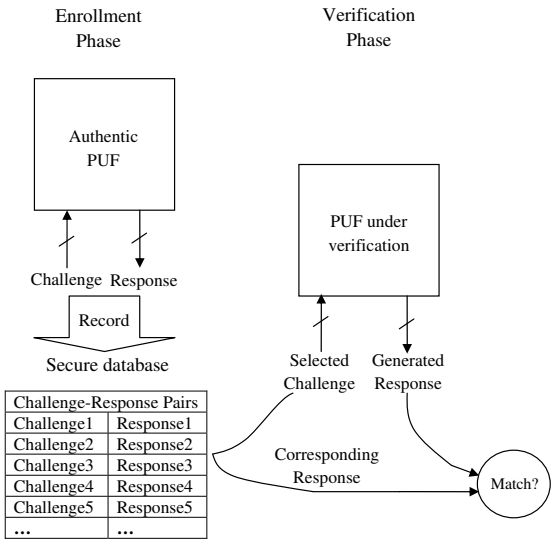

Figure 1.1: A general PUF-based authentication scheme [1]

One of the most important requirements of a practical PUF which is utilized in a device authentication process is a large set of challenge-response pairs (CRPs).

Because the CRP which is chosen by the authentication authority is transferred over an insecure channel, an attacker can capture and reuse this information to attack the authentication system. In order to prevent such replay attacks, each challenge-response pair should be used only once during the authentication process. Therefore, the utilized PUF should provide a large number of challenge-response pairs so that a device can be authenticated a significant number of times before the CRP set is exhausted.

In secret key generation, on the other hand, a specific key should be regenerated for unlimited number of times. In other words, because the secret key is not stored in the system, and the PUF circuit produces it whenever it is needed, the regenerated response (key) should be 100% noise-free. As proposed by Suh et al., the secret key generator based on PUF works as follows [1]: in the initialization phase, a specific challenge is applied to the PUF and a response is generated, as shown in Fig. 1.2. Then, using an error correcting code such as BCH, the error correcting syndrome (called Helper Data) for that response is computed. The applied challenge and the syndrome are then stored publicly on a chip or a server. In the reconstruction phase, the same challenge is applied to the PUF and the noisy output will be corrected using the computed syndrome to produce the same response as the secret key. Note that, the publicly stored syndrome reveals information about the PUF response and thus the secret key. Therefore, if a q-bit secret key is needed and the syndrome bit-size is b, the number of PUF response bits should be at least $|r| = b + q$ [1].

1.4 Thesis Outline and Contributions

In Chapter 2, we discuss different memory-based and delay-based PUFs in more details. More specifically, the construction and properties of SRAM, SR Latch, Flip-Flop, Butterfly, Buskeeper, Arbiter, Ring Oscillator, and Glitch PUFs are discussed. Additionally, more details about the PUF applications are provided. The first contribution of this thesis is introduced in Chapter 3. It includes proposing a novel Hybrid PUF structure to improve the randomness of the generated response.

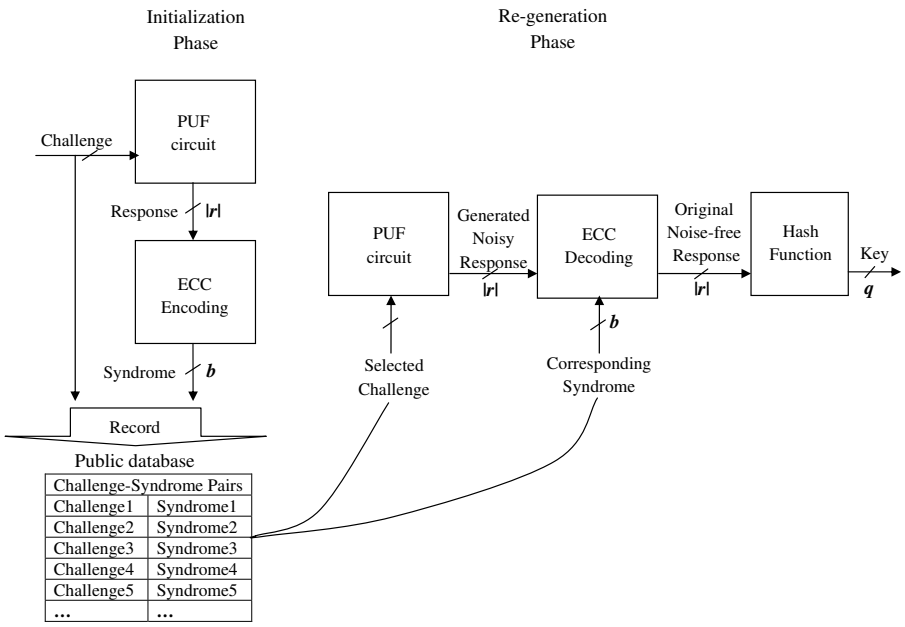

Figure 1.2: A secret key generation scheme using PUF [1]

In fact, two known PUF schemes are combined in a way to take advantage of both of them. We analyze the performance of the proposed scheme in terms of reliability, uniqueness, uniformity, and bit-aliasing and compare it with other implemented schemes. The second contribution is discussed in Chapter 4. We propose an efficient secret sharing scheme based on PUF properties. The information rate which is the main factor in assessing the efficiency of a secret sharing scheme is increased using our method. Our proposed scheme also provides cheater detection capability for the system. Finally, conclusion and future work are briefly discussed in Chapter 5.

Chapter 2

Literature Review

In this chapter, we briefly review the proposed schemes in the literature related to both PUF applications and PUF design. As mentioned in the previous chapter, both memory-based and delay-based PUFs are chosen, which include SRAM, SR Latch, Flip-Flop, Butterfly, Buskeeper, Arbiter, Ring Oscillator, and Glitch PUFs. We describe the PUF structures and explain how the response bits are generated. Also, each structure's advantages and disadvantages are mentioned.

2.1 PUF Applications

In this section, we briefly review the proposed works published in the open literature regarding the applications of PUFs.

2.1.1 Authentication

As discussed before, in device authentication, there is no need to generate 100% noise-free response bits. In fact, if the generated response is "close enough" to the one stored in the database, the PUF under verification is authenticated. Therefore, the failure rate of the authentication system which is defined as the device misidentification probability [30], depends on the reliability and uniqueness properties of the utilized PUF. It also depends on the number of PUF response bits.

In other words, a longer PUF response can authenticate a bigger population of devices with less failure rate [6]. In addition, the resilience of the authentication system against the replay attacks depends on the number of challenge-response pairs provided by the PUF. Thus, all the works which are proposed to either improve the reliability, uniqueness, number of response bits, and/or number of CRPs, can be considered as works related to the authentication application.

2.1.2 Secret Key Generation

The main building block in a secret key generator scheme using PUF is the error correcting code (ECC) which is used to produce a 100% noise-free response. The use of 2D Hamming codes for error correction is suggested in [12]. Also, using a more realistic model of PUFs noisy properties, Suh et al. suggested the use of BCH codes as the ECC [31]. In addition, a new syndrome coding scheme that restricts the amount of leaked information by the PUF error-correcting codes is proposed in [32].

A fuzzy extractor implementation on FPGAs is proposed in [33] to generate uniformly distributed and noise-free cryptographic keys. The proposed fuzzy extractor has two stages; the first stage generates a noise-free key using an ECC, and the second stage transforms the response using a universal hash function to achieve a uniform or any other required distribution of keys. A 128-bit secret key using an RO-PUF is proposed in [13] using a fuzzy extractor which includes a BCH(255,37,45) error correcting code. In addition, Maes et al. proposed a practical low overhead secret key generation called PUFKY, which can generate a 128-bit secret key with a failure rate of 10^{-9}, in 5.62 ms, and with low area overhead [7].

2.2 Memory-based PUFs

As discussed before, a bistable memory cell which has 2 stable states (0 and 1), goes to the metastable state for a short period of time and then settles in one of

the 2 states. This settling state is random and unpredictable for each memory cell. Therefore, this random behavior is used to build a PUF which produces random response bits. For example, as shown in Fig. 2.1, an SRAM memory cell consists of 2 cross-coupled inverters at its core. The transient behavior of an SRAM cell when it is powered up is what an SRAM PUF is built upon [19]. Typical SRAM cells are designed to have perfectly matched inverters. However, due to uncontrollable process variations, the *strength* of the inverters will not match in an SRAM cell. Based on which inverter is stronger, the memory cell will settle in one of the stable states. If the difference between the strength of the inverters is significant, the produced response bit (which is the settling state of the cell) will be stable. On the other hand, if the inverters are somehow equally strong, the settling state on each *power-up* will be different due to noise effects, resulting in an unstable bit.

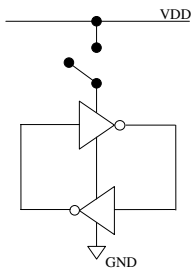

Figure 2.1: SRAM cell logic circuit

The power-up state of 8190 bytes of SRAM from different memory blocks on different FPGA boards are collected in [19]. The uniqueness is reported to be 49.97% and the reliability is shown to be 96.43% at normal conditions and 88% for higher temperature conditions. The main drawback of this PUF scheme is that, the response bit is generated only on the power-up state of the circuit. In other words, the response bit cannot be re-generated while the circuit is operational. This drawback makes the SRAM PUF an impractical PUF because for each sample of the response bit, one has to turn the circuit off and on again.

Other memory-based PUFs are depicted in Fig. 2.2. The basic principle of these PUFs is the same as that of the SRAM PUF: random mismatch between

nominally matched cross-coupled devices. For example, in the SR Latch PUF, 2 cross-coupled NAND (or NOR) gates constitute a simple SR Latch.

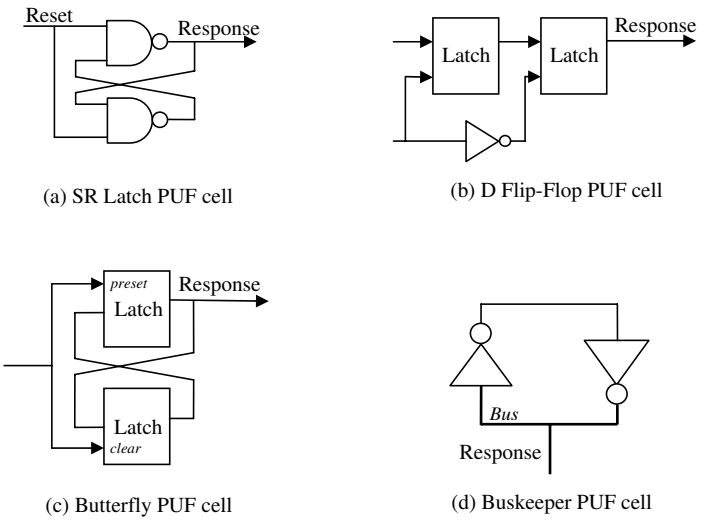

Figure 2.2: Different memory-based PUF structures

In the NAND-based SR Latch, when the *Reset* signal is 0, the output bit is 1. When the reset is released, the output bit will converge to either 0 or 1 depending on the internal mismatch between the 2 gates. 128 NOR-based SR Latches are instantiated on 19 ASICs manufactured in 130 nm CMOS technology [20]. The uniqueness and reliability at nominal conditions are reported as 50.55% and 96.96%, respectively. The main advantage of the SR Latch PUF is that the PUF response bits can be re-generated at any time when the circuit is powered and operational. In fact, we can take many samples from the response bit to analyze the PUF performance by connecting a *clock* to the Reset signal. In addition, a majority voting technique can be applied on the samples to generate more reliable bits. Flip-Flop, Butterfly, and Buskeeper PUFs behavior and principle are basically similar to those of the SRAM PUF. Like the SRAM PUF, the response bits generated by these PUFs are obtained only on the device power-up state. The power up states of 4096 Flip-Flops on 3 different FPGA boards are measured in [21]. After applying simple majority voting techniques on the output bits, the uniqueness and reliability are estimated as 50% and 95%, respectively. Also, implementation of

64 Butterfly cells on 36 FPGA boards yields a uniqueness of approximately 50% and a reliability of 95% [22]. Finally, a 8192-bit Buskeeper PUF has been implemented on an ASIC platform in [34]. The uniqueness is estimated as 48.88%. At normal conditions, the reliability is reported as 95.84% and under higher temperature conditions, it is shown to be approximately 83%.

2.3 Delay-based PUFs

2.3.1 Arbiter PUF

Figure 2.3 depicts a basic Arbiter PUF design proposed in [2]. The basic idea of this scheme is to let a rising-edge signal travel through two different delay paths. At the end of the delay paths, an *arbiter* circuit decides which path is the winner of the race. The arbiter circuit has 2 inputs and 1 output. If the rising edge arrives at the first input before it arrives at the second input, the output will be 1, and 0 otherwise. The delay paths are implemented as a chain of *switch boxes*. Each switch box has a *select* signal which determines the connection between the 2 inputs and the 2 outputs. If the *select* is 0 the connection is straight and if it is 1, the connection is crossed. As shown in the figure, the switch box can be implemented using two $2 - to - 1$ multiplexers. Since there are a number of switch boxes in the chain, the set of *select* signals can be considered as the PUF challenge bits, and the outputs for each configuration can be considered as the PUF response bits. The structure of the utilized switch boxes, and thus the delay lines must be nominally perfectly symmetrical so that the output bits depend only on the random physical and manufacturing variations. Also, the arbiter circuit must be completely fair, i.e., it must not have a biased behavior towards a specific bit. As suggested by Lin et al. in [35], a basic SR latch is the best option for a fair arbiter because of its symmetric construction.

There is a non-trivial chance that, both delay lines are almost identical. In this case, the rising edge arrives at the 2 inputs of the arbiter at nearly the same time.

Therefore, the arbiter goes into its metastable state and after a short period of time, it will settle down in one of its 2 stable states. Although the output in this case is totally random, it is not static for each device and therefore, it is the main cause of unreliability in an Arbiter PUF.

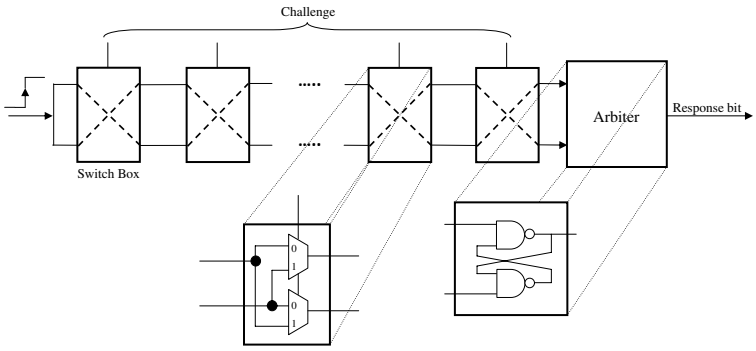

Figure 2.3: A basic Arbiter PUF design [2]

Gassend et al. [36] implemented the basic Arbiter PUF design on a set of FPGA chips. This implementation lacks low-level control over the placement and routing of the delay lines. The reliability is reported to be 99.9% under normal conditions, which is a high value. However, the uniqueness of this scheme has an extremely low value: 1.05%. Hence, this Arbiter PUF implementation is very biased which is a result of non-symmetric delay line design. Another implementation is performed by the same group on ASIC platform which controls the placement and routing of the switch boxes. The uniqueness is shown to improve significantly: 23%. But, it is still far from the ideal 50%. The reliability is also shown to be very high: 99.3% under normal conditions and 95.18% under high temperature conditions.

The most important issue with the basic Arbiter PUF is its weak resilience against modeling attacks. The digital delay line is additive by nature, meaning that, the total delay of the delay line paths is the sum of the delay of the switch boxes in the chain. Therefore, an attacker will be able to predict unknown responses as accurately as possible after monitoring a specific number of challenge response pairs. It is shown in [25] that the basic Arbiter PUF scheme is 96.45%

predictable after observing 5000 CRPs. Hence, it is easily broken through modeling attacks. *Feed-forward* Arbiter PUF is thus proposed in [2] to increase the resilience of Arbiter PUFs against modeling attacks. The idea of feed-forward Arbiter is shown in Fig. 2.4. As we can observe, the *select* signal of a switch box in the main delay path is determined by the inserted arbiter A^*. The implementation results on ASIC platform indicate that the uniqueness of the new Arbiter PUF is increased to 38%, while its reliability is decreased to 90.16% under high temperature conditions. The reliability is decreased because the number of arbiters are increased in the design and as discussed before, each arbiter can go to a metastable state which results in noisy outputs. This scheme is also shown to be vulnerable against modeling attacks [37, 38]. In fact, the feed-forward Arbiter PUF is shown to be 97.5% predictable after observing 50000 CRPs.

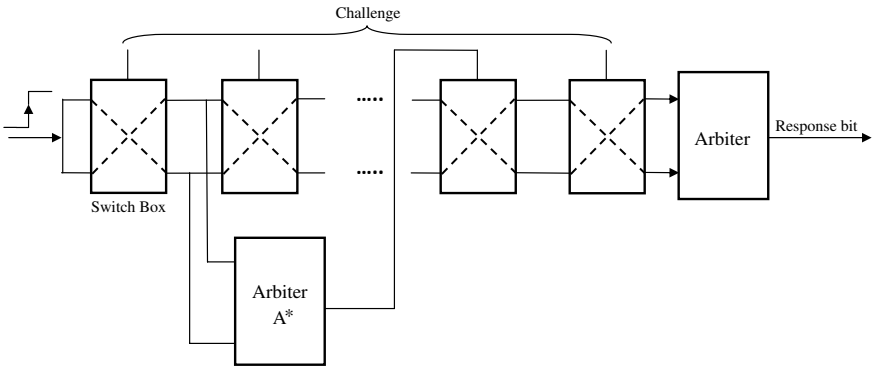

Figure 2.4: The feed-forward Arbiter PUF design [2]

Majzoobi et al. proposed a more advanced technique to make Arbiter PUFs resilient against modeling attacks in [39]. In this technique, multiple arbiter PUFs are used in parallel and their outputs are XOR'ed to generate the response bits. Although this technique makes modeling attacks much harder, it is still shown that modeling attack against such scheme is feasible. Rührmair et al. show that the new scheme with 64 switching boxes and 3 parallel arbiters is 99% predictable with 60000 challenge-response pairs being observed [38].

2.3.2 RO PUF

Figure 2.5 shows a basic RO PUF structure proposed in [1]. It includes N identical u-stage ring oscillators shown in Fig. 2.6. Note that, the number of stages in a ring oscillator is the number of inverters in the feedback loop. The ring oscillator generates a clock signal, the frequency of which is directly related to the delay of the inverters.

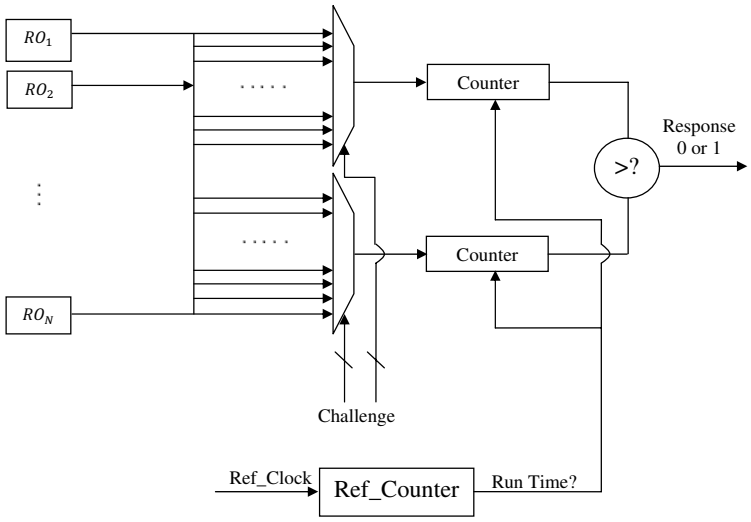

Figure 2.5: An architecture of an RO PUF [1]

The outputs of the ring oscillators are connected to the inputs of two $N-to-1$ multiplexers. A $2\log_2 N$-bit challenge selects a pair of ring oscillators, the outputs of which will be connected to the clock inputs of the two counters.

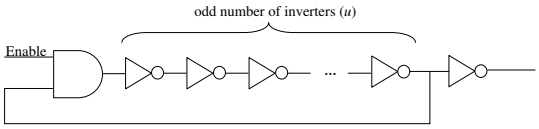

Figure 2.6: A basic ring oscillator circuit

The two counters will start counting at the same time and after a specific period of time (determined by the *Ref_Counter* as *Run Time*), the counter outputs are compared. If the upper counter has a greater value, the response bit will be 1, otherwise 0. Theoretically, the oscillation frequency of all the ring oscillators

should be the same because they are exactly identical. However, due to the inherent inter-chip and intra-chip process variations, as well as the environmental conditions, the delays of the inverters will vary across different ring oscillators, thus affecting the oscillation frequency of the ROs [28]. Note that, the pair of ring oscillators that generate two oscillation frequencies which differ more, will produce a more reliable response bit, because the environmental changes will less likely reverse the relation between their frequencies. In other words, the reliability of a PUF depends greatly on the difference between the oscillation frequencies of any RO pair. Additionally, one of the advantages of the RO PUF is that, the ring oscillator can be implemented as a hard macro and instantiated as many times as needed in the top-level design. Using this technique, all the ROs will be identical in terms of placement and routing. A large scale characterization of RO PUF is provided in [26]. The uniqueness is shown to be 47.31% and the reliability is measured to be 99.14% under normal conditions.

In order to improve the reliability of an RO PUF, a 1-out-of-γ masking was introduced in [1]. In this scheme, the RO pair that has the maximum frequency distance among other pairs are selected and their frequencies are compared to produce the response bit. The reliability of this PUF scheme implemented on 15 FPGA chips shows a uniqueness of 46.15% and a reliability of 99.52% under normal conditions [1]. The main drawback of this scheme is the huge area overhead. In fact, γ times more area is used to produce the same number of response bits. Maiti et al. addressed this drawback by proposing and constructing a *configurable RO* [3]. Figure 2.7 depicts their proposed 3-stage configurable RO, each stage of which can fit into 1 SLICE.

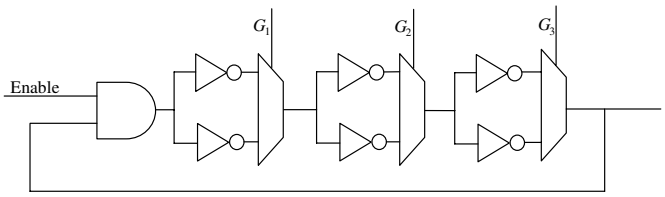

Figure 2.7: Maiti's Configurable RO [3]

Despite the basic RO (Fig. 2.6) that has only one path in the feedback loop, in this configurable RO the three signals, G_1,G_2, and G_3 can select the inverters to be included in the loop. This provides us with 8 different ring oscillators (because of the delay variations of different LUTs and wires within the FPGA), while it occupies the same amount of area (1 CLB) compared to a basic RO. Similar to the 1-out-of-γ masking scheme, a configuration for the one pair of configurable ROs which has the maximum frequency distance among the 8 configurations can be selected in order to improve the PUF reliability. In summary, the configurable ROs can be used in a 1-out-of-γ manner (where $\gamma = 8$), while occupying the same amount of area. Another important advantage of the Maiti's configurable RO is its ability to create 8 response bits instead of a single response bit. Implementation results for 64-, 128-, and 256-RO PUFs under varying voltage and temperature shows that, using the 1-out-of-8 scheme with the configurable RO improves the PUF reliability while maintaining a high value of uniqueness.

In addition, Xin et. al improve Maiti's configurable RO by increasing the number of possible configurations to 256 [4]. Figure 2.8 shows their proposed configurable 3-stage RO which can also fit into 1 CLB. As we can see, similar to Maiti's design, each stage is implemented in 1 SLICE. However, a latch is inserted in all SLICES and the signal *sel* determines whether or not a latch should be included in the path coming from the preceding stage. Because the delay associated with each latch is random and unpredictable due to manufacturing variations, it can be considered as another random factor in the PUF design that helps enhance the PUF unclonability. Note that, the other select signals, bx_i, have the same functionality as select signals, G_i, in Maiti's RO.

Because there are 8 configuration signals in the ring oscillator, namely $sel[3..0]$ and $bx[3..0]$, 256 different RO configurations are available, each of which can generate different oscillation frequencies. Thus, this scheme is able to generate even more response bits for a given challenge while occupying the same amount of area. It is shown that the reliability of this RO PUF design with 128 ROs is

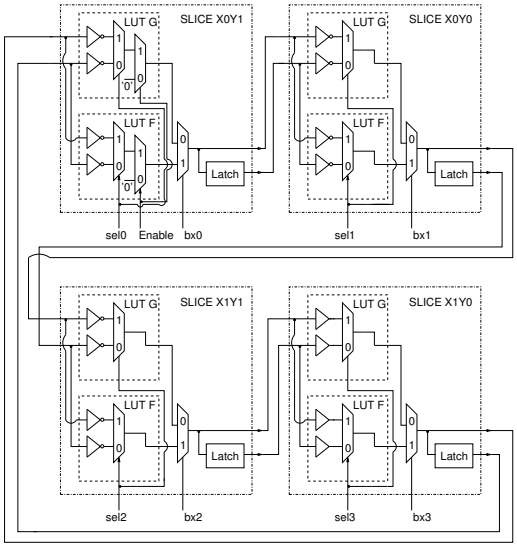

Figure 2.8: Xin's Configurable RO in One CLB [4]

98.98% and the uniqueness is reported to be 40% for the same number of ROs.

In addition, a multi-voltage RO PUF is proposed in [5] as depicted in Fig. 2.9. The idea behind this scheme is the dependency of the combinational path of digital cells delay on the supply voltage. As we can see in Fig. 2.9, the supply voltage of each column of inverters is different and can be selected among d different values. Because the oscillation frequency of each RO depends on the delay of the inverters included in its feedback loop, and the delay of the inverters depends on process variations as well as the supply voltage, different ROs generate clock signals with different frequencies. The authors claim that this new RO PUF can produce a higher number of response bits, consumes less amount of area, and is more reliable in case of temperature variations. It can generate a higher number of response bits because the supply voltage of the different columns is considered as another random factor that can directly influence on the oscillation frequencies of the ROs. Therefore, by changing the supply voltages of the inverter columns, each pair of ROs can generate a set of different response bits. However, one of the important drawbacks of this scheme is that, the inverters used in different columns are not identical any more. Thus, if an attacker gains access to the supply voltage configuration of a chip through an invasive attack, they would most likely estimate

the most probable response bits.

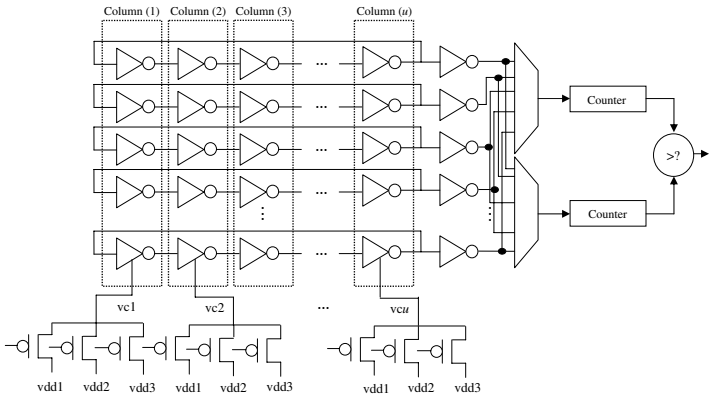

Figure 2.9: Multi-voltage RO PUF [5]

The 1-out-of-γ masking scheme proposed in [1] is a good example of PUF *post-processing*. In fact, an additional processing is performed on the ring oscillator frequencies to provide more reliable response bits. In addition, the previous schemes suffer from the fact that, the number of PUF response bits and the number of challenge-response pairs provided by the PUF are limited to the area. The idea of RO frequencies post-processing is further investigated in [6]. In this study, an *identity-mapping function* along with a quantization process are applied on the RO frequencies in order to increase the number of challenge-response pairs. The proposed scheme is shown in Fig. 2.10.

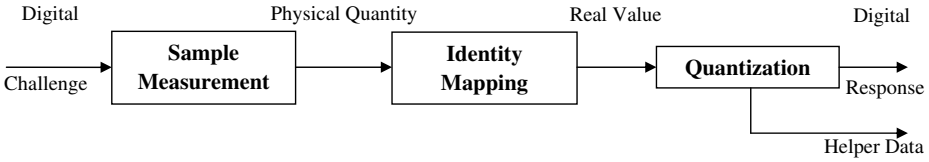

Figure 2.10: The RO PUF with identity-mapping [6]

In the sample measurement phase, the challenge, c, selects each ring oscillator one at a time and the selected RO frequency is measured and recorded. Therefore, the ”Physical Quantity” in the figure refers to the RO frequencies. In the identity-mapping phase, any subset of RO frequencies whose cardinality is greater than 2 is selected and a corresponding Q-value is computed for each subset. The ”Real

Value" in the figure refers to this Q-value because it is not a digital value and it can have more than 2 values. Therefore, a quantization process is required to transform these real values to digital strings which are used as the PUF response bits. It is shown that, the proposed scheme can provide upto $2^N - N - 1$ response bits, where N is the number of ring oscillators. Therefore, it is observed that, with a small number of ring oscillators (and thus, a small area cost), a large number of response bits and a large set of CRPs can be produced. The only expense that is paid is the additional post-processing applied on the RO outputs. Experimental data obtained from an implementation on 125 FPGAs shows a uniqueness of 49.99% which is nearly ideal. Also, the reliability is demonstrated to be 90% under high temperature conditions (70 $^\circ$C).

Another good example of post-processing on the generated RO frequencies is the one proposed by Maes et al. in [7] and shown in Fig. 2.11. There are β batches of ring oscillators where each batch contains α ring oscillators. In total, there are $\beta \times \alpha$ number of ROs. The design of each batch is similar to the basic RO structure shown in Fig. 2.5, i.e., all α ring oscillators are fed into an $\alpha - to - 1$ multiplexer and the output of the multiplexer is connected to the clock input of a counter. The counter counts for a specific period of time which is determined by a reference counter. The count value after this run time represents the frequency of the selected RO. The frequencies of β ROs selected from each batch are measured simultaneously and an h-bit response is generated based on the ordering of the measured frequencies. Therefore, the total number of generated response bits is equal to $h \times \alpha$. The process of encoding the β frequency measurements and transforming them into an h-bit response is performed in 3 steps. First, the measured frequencies are normalized by removing the oscillator-dependent structural bias. The bias value is shown to be the mean value of the RO frequency which is estimated by averaging the frequency over many measurements on many devices. The estimated mean value for different ROs are called the normalization terms which need to be computed only once and can be stored in a ROM for later use. Subtracting this mean value from the measured frequency results in the normal-

ized frequency. The normalized frequencies are then transformed into an h'-bit vector based on the order of the frequencies using the proposed Lehmer-Gray Encoder. It is shown that, some bits among the generated h' bits are biased and/or dependent to each other. Therefore, in order to increase the entropy and thus, the randomness of the response bits, a simple compression is performed on the h' bits. In fact, the bits which suffer the most from the bias and/or dependencies are XOR'ed with each other to produce an h-bit response, where $h' \leq h$.

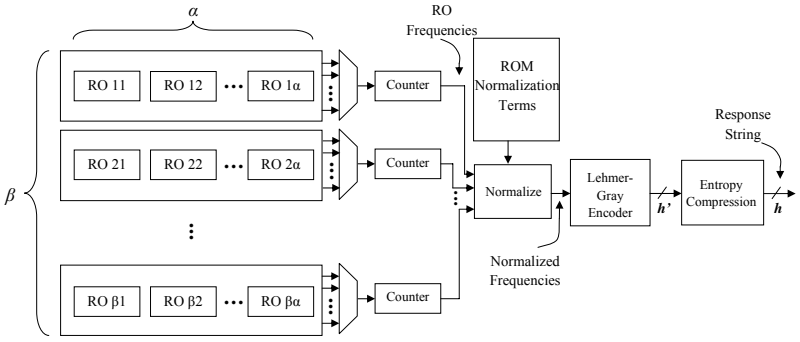

Figure 2.11: The RO PUF structure proposed in [7]

Possible modeling attacks that can be applied on the Basic RO PUF are discussed in [38]. It is mentioned that, if an attacker can select the challenge-response pairs adaptively, they can sort the RO frequencies in a specified order without knowing the exact frequency of each RO. Then, the attacker will be able to predict the responses with a correctness rate of 100% because the absolute value of the RO frequency does not have any effect on the generated response. In fact, the response is produced based on the ranking of the RO frequencies. Maiti et al. investigate the security of their proposed RO PUF with identity-mapping ([6]) against this modeling attack. Since the produced response of this scheme does not solely depend on the frequency ranking of the ROs, it is shown to be resilient against this attack. In other words, because the RO frequencies are first transformed into Q-values, and then the Q-values are transformed into binary strings using the quantization function, the sorting technique proposed in [38] will not work against this scheme. Additionally, five other cases are considered to analyze

the security of the RO PUF with identity-mapping in [6]: uniformity of response, response conditioned by challenge, inter-response dependency test, differential attack, and reverse engineering attack. The proposed scheme is shown to be resilient in all cases [6].

Finally, the effect of the FPGA chip *aging* on the Basic RO PUF is investigated in [40]. Aging is considered to be an *irreversible* temporal change that has the potential of affecting the reliability and randomness of the PUF response and thus, making the PUF unsuitable for authentication and secret key generation applications. It is shown that, the reliability of the RO PUF is reduced by 6% with aging. However, the uniqueness and entropy of the RO PUF do not seem to be affected by this parameter. Therefore, the security of the RO PUF is not compromised with aging.

2.3.3 Glitch PUF

Any combinatorial logic has a glitch behavior. The occurrence, the number and the shape of the glitches on the output of the combinatorial logic is partially random and device-specific depending on the random process variations. The glitch behavior of such circuit can thus be converted into random response bits. In other words, Glitch PUFs produce response bits from the unwanted glitches in the circuit.

Anderson PUF proposed in [8] is an example of Glitch PUFs. As discussed before, in order to have a set of identical ring oscillators in terms of placement and routing, one should create a ring oscillator as a hard macro and instantiate it as many times as needed in the top level PUF design. The drawback of this approach is that, the design flow becomes too complicated with the use of hard macros. In fact, the designer must work at a lower level of abstraction than Register-Transfer Level (RTL). Also, routed hard macros tend to cause longer run times in the Place and Route (PAR), and might even cause PAR to crash. The Anderson PUF addresses these issues. It does not need the use of hard macros and can be easily

embedded in a design's HDL. Figure 2.12 depicts the proposed PUF circuit.

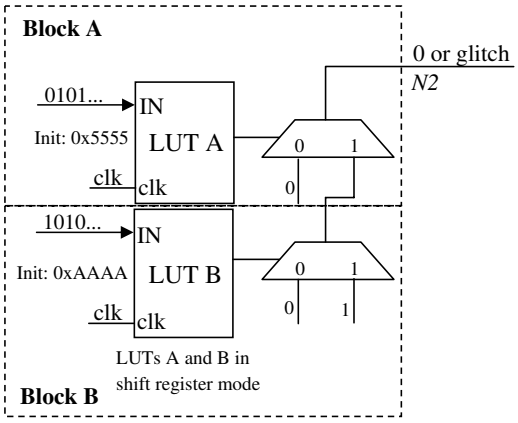

Figure 2.12: The proposed Anderson PUF circuit [8]

Two LUTs within a slice are used as 16-bit shift registers. LUT A is initialized with $0x5555$ and LUT B is initialized with $0xAAAA$. Therefore, shift register A generates a bit stream of $0101...$ and shift register B generates a bit stream of $1010...$.Note that, these two bit streams are complement of each other. Because the delays associated with the shift registers and the multiplexers they drive are different due to process variations, the output $N2$ can be either a constant 0 or a short positive spike. The presence or absence of a positive spike on $N2$ is utilized to decide the response bit. This process is shown in Fig. 2.13. The response bit is 1 if a spike is applied to the asynchronous preset input of the flip-flop, and 0 otherwise. The PUF circuit shown in Fig. 2.12 generates only 1 response bit. This circuit can be instantiated as many times as needed to create a multi-bit response.

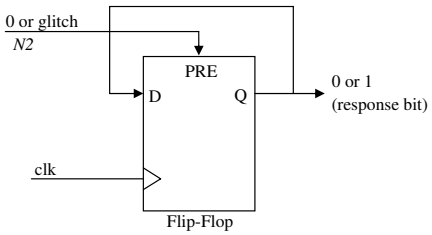

Figure 2.13: PUF response bit generation [8]

The aforementioned design along with a pulse width tuning approach are implemented on a Virtex-5 65 nm FPGA and the performance is analyzed under

temperature variation. It is shown that, on average, 3.6% of signature bits are unstable under high temperature conditions, which is in line with other published PUF circuits. Also the uniqueness is reported 48.28%. Another implementation of a 64-bit Anderson PUF on 5 Spartan-6 FPGAs in [41] shows a uniqueness of 45.62%.

Another example of Glitch PUFs is the one proposed in [29] and later improved in [9]. Figure 2.14 depicts the Glitch PUF proposed in [9]. The generated response bit is the parity of the number of glitches that occur during a specific period of time. In fact, the output of the combinatorial logic, which is chosen to be the AES S-Box as an example, is connected to a toggle flip-flop. If the number of glitches is odd, the response will be 1 and if it is even, the response will be 0. To improve the reliability of the proposed scheme, the unstable bits are identified in a pre-processing stage and the information about them is stored in the system. These unstable bits are ignored when the PUF is actually used in practice. This technique is called *bit-masking* and adds extra overhead to the system but it is shown to improve the reliability, significantly. The proposed scheme along with the bit-masking technique is implemented on 16 FPGA chips and the reliability is reported to be 98.7% under normal conditions. However, it is shown that the applied bit-masking technique ignores almost 38% of the response bits. This demonstrates that, the proposed Glitch PUF without the bit-masking technique suffers from a substantial instability. Since the bit-masking technique is a general technique and not specific to Glitch PUFs, it is concluded that the proposed Glitch PUF does not show a suitable practical behavior. In addition, the uniqueness is reported to be 35%.

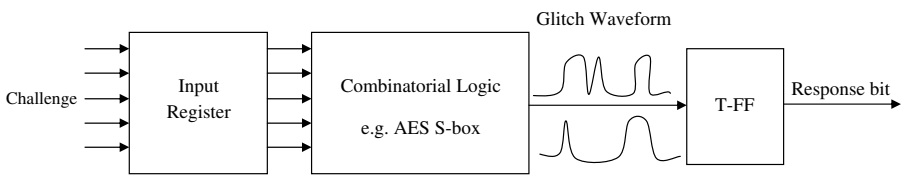

Figure 2.14: The Glitch PUF [9]

Table 2.1: Comparing the performance of different PUF schemes in the literature in terms of uniqueness and reliability (%)

PUF Scheme	Uniqueness	Reliability		Platform	Number of Boards	Any Processing Applied?
		Normal Conditions	Higher Temperature Conditions			
Basic RO [26]	47.31	99.14	96	Xilinx Spartan 3E FPGA	125	No
RO with 1-out-of-γ masking [1]	46.15	99.52	N/A	Xilinx Virtex 4 LX25 FPGA	15	Yes
Configurable RO [4]	40	98.98	N/A	Xilinx Spartan 3E FPGA	4	Yes
RO with identity-mapping [6]	49.99	99	90	Xilinx Spartan 3E FPGA	125	Yes
RO with Lehmer-Gray Encoder [7]	48.4	98	91	Xilinx Spartan 6 XC6SLX45 FPGA	10	Yes
Anderson [8]	48.28	N/A	96.4	Xilinx Virtex 5 FPGA	36	No
Anderson [41]	45.62	N/A	N/A	Xilinx Spartan 6 XC6SLX45 FPGA	5	No
SR Latch [20]	50.55	96.96	N/A	130 nm CMOS ASIC	19	No
SR Latch [34]	37.01	96.6	87.29	65 nm CMOS ASIC	192	No

Table 2.1 summarizes the performance of the PUFs discussed in this chapter in terms of uniqueness and reliability. Note that, a fair comparison between different PUFs performance can be done only when they are all implemented on the same platform, under the same conditions, and even designed by the same developer.

Chapter 3

Implementation Results

In this chapter, we provide the implementation results of different PUF schemes in terms of reliability, uniqueness, uniformity, and bit-aliasing. First, the formal definitions and formulations of these PUF performance metrics are presented. Then more details on the implemented schemes, design parameters, and measurement system are provided. Finally, the implementation results are presented and discussed.

3.1 PUF Performance Metrics

In this section, four important PUF characteristics including reliability, uniqueness, uniformity, and bit-aliasing are discussed. The *randomness* of a PUF response is determined by its *entropy*. However, it is very difficult to estimate and calculate the entropy of a PUF response because one cannot learn the complete details about the statistical distribution of the PUF responses which is generally determined by very complex and even chaotic physical processes [34]. So, the randomness of the PUF responses is truly indicated by uniqueness, uniformity, and bit-aliasing [6].

3.1.1 Reliability

Reliability of a given PUF instance is a measure of stability of the PUF response bits to a given challenge at different times and under different conditions [3]. Ideally, the value of reliability is 100%, meaning that the PUF under study generates the exact same response to a given challenge at different times and under different conditions, such as different temperatures or different supply voltage values. It is defined as [3]:

$$Reliability = (1 - \frac{2}{m \times (m-1)} \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \frac{HD(r_i, r_j)}{a}) \times 100\% \tag{3.1}$$

where m is the number of response samples, a is the number of response bits, and HD is the Hamming distance between two response samples r_i and r_j. So, we basically take m number of samples of the response of a given PUF instance to a specific challenge, calculate the Hamming distance between any two responses (where the total number of unique comparisons between m responses is $\frac{m \times (m-1)}{2}$), and calculate the average number of unstable bits among a response bits. This value represents the average instability or intra-distance of the given PUF instance. Finally, reliability is derived by reducing this value from 100%.

3.1.2 Uniqueness

Another important feature of a PUF is its uniqueness. Uniqueness is a measure of inter-distance variations of the response bits of different PUF instances. In other words, if a specific challenge is applied at the same time and under the same conditions to two identical PUF instances, the response of the two PUFs should be different. Ideally, this value should be 50%. It is calculated as [3]:

$$Uniqueness = \frac{2}{g \times (g-1)} \sum_{i=1}^{g-1} \sum_{j=i+1}^{g} \frac{HD(r_i, r_j)}{a} \times 100\% \tag{3.2}$$

where g is the number of PUF instances under study, a is the number of response bits, and HD is the Hamming distance between two response samples r_i and

r_j. The same challenge is applied to g identical PUF instances and the average Hamming distance between the response bits of any two PUF circuits is calculated (where the total number of unique comparisons between g different PUF circuits is $\frac{g\times(g-1)}{2}$).

3.1.3 Uniformity

Uniformity is the measure of uniform distribution of 0's and 1's in the response of a single PUF instance. It is defined as [14, 42]:

$$Uniformity = \frac{1}{m \times a} \sum_{i=1}^{m} \sum_{j=1}^{a} r_{i,j} \times 100\% \tag{3.3}$$

where m is the number of response samples, a is the number of response bits, and $r_{i,j}$ is the j-th bit of the i-th response sample. Ideally, uniformity should be 50% meaning that 50% of the response bits are 1 and 50% are 0.

3.1.4 Bit-aliasing

Another important indicator of a PUF randomness and unclonability is bit-aliasing. The bit-aliasing of the j-th response bit is the average Hamming weight of that bit position across several PUF instances. Ideally, this value should be 0.5 for all bit positions in the PUF response. It is defined as [26]

$$Bit - aliasing_j = \frac{1}{g} \sum_{i=1}^{g} r_{i,j} \tag{3.4}$$

for all j, $0 \leq j \leq a$, where g is the number of PUF instances and $r_{i,j}$ is the j-th bit of the i-th PUF instance response.

3.2 Design concepts: Basic PUFs

Among the memory-based PUF schemes, the SR Latch PUF is chosen because of its ability to re-generate the response bits when the circuit is powered and opera-

tional. We also implement the Basic RO and Anderson PUFs and we evaluate and compare their performance through statistical analysis.

3.2.1 SR Latch PUF

A NAND-based SR Latch PUF is implemented as shown in Fig. 3.1. A *clock* is connected to the reset signal and whenever the level of the clock is 1, the output bit is read and recorded. Note that, if we need to obtain an N-bit response, we have to instantiate the SR Latch unit N times and obtain 1 bit from each unit.

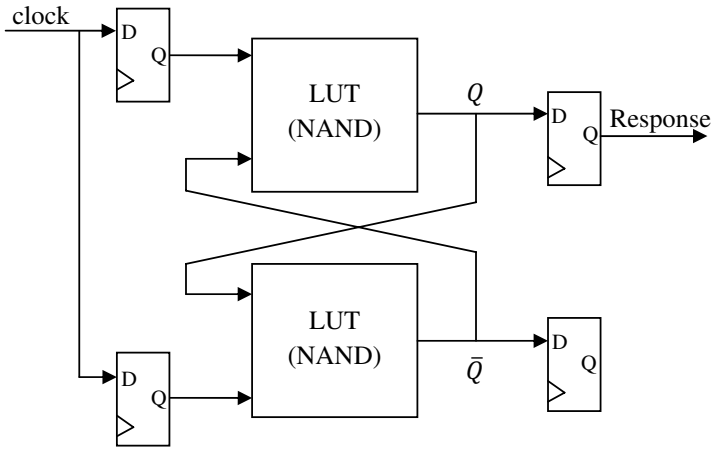

Figure 3.1: The NAND-based SR Latch [10]

In order to achieve the best results for the SR Latch PUF in terms of randomness of the response bits, some details related to the implementation of the SR Latch have to be considered [10]. First of all, the flip-flops used on the input side of the latch are necessary to reduce the skew of the clock signal. The flip-flop used on the output bit (Q signal) is used to balance the capacitive load of the Q signal with the capacitive load of the $\overline{Q}$ signal [10]. Note that, our platform in this study is a Virtex II Pro FPGA evaluation board. Each CLB of a Virtex II Pro FPGA has four SLICEs, and each SLICE contains two lookup tables and two flip-flops. Therefore, each lookup table (NAND gate) along with its input and output flip-flops are implemented in a single SLICE. So, the SR Latch shown in Fig. 3.1 requires only two SLICEs which can be placed in a single CLB. However, in or-

der to guarantee the symmetrical implementation of the Q and $\overline{Q}$ signals, the two SLICEs have to be placed in two separate CLBs. Even the distance between the two CLBs is important and has direct effect on the PUF performance. A thorough analysis is performed on the effect of the placement of the CLBs on the randomness of the generated response bits in [10]. Based on our initial implementations and experiments, we obtained the best results when the distance between the two CLBs is 1 CLB. In other words, if one of the lookup tables along with the corresponding flip-flops is fitted in the SLICE `X0Y0`, the other lookup table and its flip-flops are implemented in the SLICE `X0Y4`.

3.2.2 Basic RO PUF

As discussed before, an RO PUF generates the response bits by comparing the frequencies of two different ring oscillators (Fig. 2.5). Note that, the counter size and run time should be carefully selected because, as we can observe, the response of the PUF relies on the difference between the oscillation frequency of different ring oscillators. Thus, the run time should be long enough to differentiate between the oscillation frequency of different ring oscillators. On the other hand, it should not be too long to cause a counter overflow. Also, the counter size should be big enough to prevent a counter overflow. A more detailed discussion on how to select the counter size and run time is provided in [13]. Also note that, in order to generate an M-bit response, M different challenges (where each challenge is $2\log_2 N$ bits wide) have to be applied to make M comparisons between different ROs. So, one of the disadvantages of this design is the low ratio of the number of response bits to the number of challenge bits ($\frac{1}{2\log_2 N}$).

The maximum number of possible comparisons between N different ROs is equal to: $\frac{N\times(N-1)}{2}$. However, not every comparison will result in an uncorrelated response bit. For example, if A is greater than B and B is greater than C, then A will be greater than C. Therefore, the comparison between A and C is correlated to the comparisons between A and B, and B and C. It is shown in [3] that, selecting and comparing the adjacent RO pairs (i.e., comparing RO_1 with RO_2, RO_2 with

RO_3, RO_3 with RO_4, etc.) eliminates the effect of this correlation. Thus, we perform only $N-1$ comparisons out of the total number of comparisons, resulting in a response which is $N-1$ bits wide.

3.2.3 Anderson PUF

As mentioned in Chapter 2, Anderson PUF design is an example of the Glitch PUFs which produce response bits from unwanted glitches in the circuit (Fig. 2.12 and 2.13). The response bit is decided based on the presence or absence of a glitch signal on the output of the top carry-chain multiplexer (*N2*). The most important factor that determines the quality of this design in terms of randomness is the width of the produced glitch. If the glitch is too narrow, it will be damped while it propagates through a wire which acts like a low pass filter. Thus, even if a glitch is produced, the response will be 0 because the glitch is not *seen* by the flip-flop. If the 2 shift register-multiplexer blocks, A and B, are located very close to each other, the produced glitch will be too narrow and the response bits will always be 0. In fact, 100% of the response bits will be 0 in this case. On the other hand, if the 2 blocks, A and B, are located in a way that they are too far from each other, a glitch will always be present on *N2* and therefore, the response bit will always be 1. The concept of *tuning the glitch width* is utilized in the proposed PUF design to address this issue [8]. The idea is to widen the produced glitch so that it is seen by the flip-flop. This is accomplished by inserting some *intermediate blocks* between the blocks A and B, as shown in Fig. 3.2.

Note that, the shift registers in the intermediate blocks are initialized with all 1's. So, they act like a simple wire. They only cause the transitions from B's output to take a little longer and therefore, the glitch will be widened. Now, if too many intermediate blocks are inserted between A and B, the response bits will be 1 with higher probability. So, all the possible number of intermediate blocks should be tested in order to achieve the best result in terms of uniformity and randomness. The best tuning was shown to be 5 intermediate blocks (shown in Fig. 3.2) in [8]. Based on our tests, the tuning which resulted the best uniformity

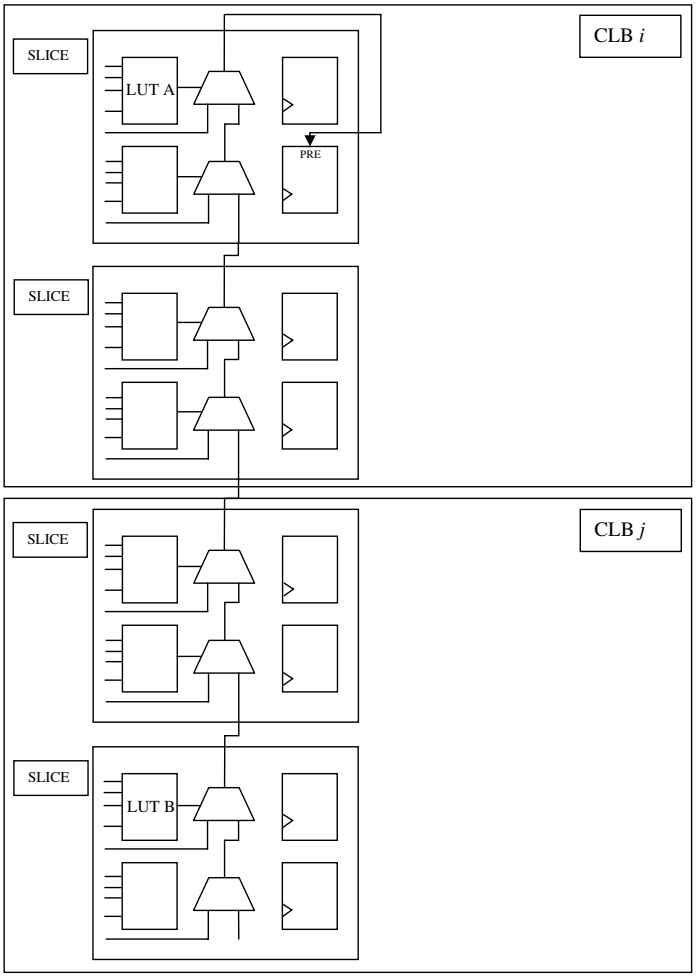

Figure 3.2: Tuning the glitch width in the Anderson PUF on a Virtex II platform [8]

was also 5 intermediate blocks.

3.3 Design Concepts: The Proposed Hybrid PUF

In this section, we propose the idea of the Hybrid PUF structure. The idea is to combine two (or more) available sources of randomness in a way to improve the uniqueness while maintaining other important performance metrics. This structure is based on the Basic RO PUF which can be combined with any other PUF unit that produces a random bit. The RO PUF and the Anderson PUF are combined using two methods and it is shown that the second method results in better

performance. Therefore, we combine the RO PUF with the SR Latch PUF as another example using the second method only. Finally, the implementation results are presented and discussed in Section 3.5.

3.3.1 RO/Anderson Hybrid PUF, Method 1

As previously discussed, the RO PUF uses the randomly generated RO frequencies to produce the response bits, and the Anderson PUF uses the shift register-multiplexer delay as its random parameter to generate the response bits. We try to combine these two ideas to increase the PUF randomness. As shown in Fig. 2.12, the Anderson PUF works in a clocked manner and the clock is the same for all instances. Also note that, the inverters in the RO circuit are implemented using lookup tables in the FPGA. Thus, an inverter in the RO circuit and a 1-bit buffer can be used as LUT A and LUT B in the Anderson PUF design. Figure 3.3 shows the proposed scheme.

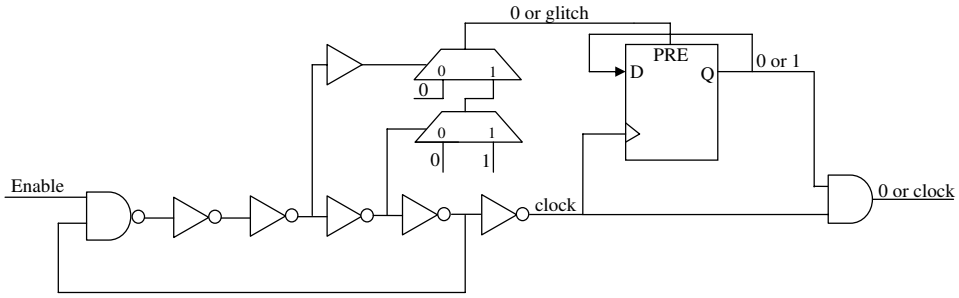

Figure 3.3: The proposed RO/Anderson Hybrid PUF structure, method 1

Note that, the output of the inverter and the buffer are complement of each other which is a requirement of the Anderson PUF. In fact, the proposed scheme is similar to the Anderson PUF except the fact that, the clock of the system is generated using a ring oscillator which can be different across different instances. So, there are two sources of randomness in the scheme, one is the random clock frequency generated by the RO and the other one is the lookup table-multiplexer delay. The output of the flip-flop is AND'ed with the RO clock. Therefore, the output of this scheme is either a 0 or a clock. Similar to the RO PUF structure

(Fig. 2.5), this block is instantiated as many times as needed and the counter values are compared after a specific run time to produce the required number of response bits. Note that, if we have N number of blocks and if we compare only the adjacent pairs, as discussed earlier, the number of response bits will be $N - 1$. Let us consider two instances of this scheme, *instance i* and *instance j*. Because each instance can have 2 different outputs, 0 or a random clock, there are 4 different scenarios shown in Table 3.1. As we can see, the response bit produced in scenario 1 is 1, scenarios 2 and 3 will produce 0, and the response bit of the scenario 4 will be either 0 or 1 (with probability of 0.5 for each of them). Note that, the response bit is the comparison result of the two instances outputs after a specific run time, similar to the Basic RO PUF scheme. Because the occurrence probability of all scenarios are theoretically equal to 0.25, it is expected to have 37.5% ($= (0.25 \times 1 + 0.25 \times 0.5) \times 100\%$) of the response bits to be 1 and 62.5% ($= (0.25 \times 1 + 0.25 \times 1 + 0.25 \times 0.5) \times 100\%$) to be 0. This means that the proposed method has a biased behavior toward the bit 0 and does not improve the randomness of the response bits. The implementation results presented in Section 3.5 verify this fact.

Table 3.1: RO/Anderson Hybrid PUF method 1 different scenarios

	Scenario 1	**Scenario 2**	**Scenario 3**	**Scenario 4**
Instance *i* output	Clock	0	0	Clock
Instance *j* output	0	0	Clock	Clock
Response bit	1	0	0	0 or 1

3.3.2 RO/Anderson Hybrid PUF, Method 2

Figure 3.4 shows method 2 for combining the RO PUF and the Anderson PUF. The Anderson PUF implementation is exactly the same as the original Anderson in terms of tuning and in the sense that the input clock to all instances are the same. The Anderson output bit which is 0 or 1 (with theoretical probability of 0.5 for each of them), is connected to the *select* signal of the $2 - to - 1$ multiplexer in the ring oscillator circuit. If the *select* signal is 0, the ring oscillator will have 5

inverters in its feedback loop and therefore, produces a clock with the frequency of f_5. If the *select* signal is 1, there will be only 3 inverters in the feedback loop which results in a clock with the frequency of f_3. Note that, f_3 is always greater than f_5 because the frequency of a ring oscillator depends on its feedback loop delay. A higher number of inverters in the feedback loop results in a larger delay and a higher clock period and thus, a lower frequency. So, if there are two instances of this block, *instance i* and *instance j*, there will be 4 different scenarios based on the Anderson output bit. These scenarios along with their produced response bits are listed in Table 3.2. Similar to Table 3.1, the produced response bits are results of comparison between two instances frequencies.

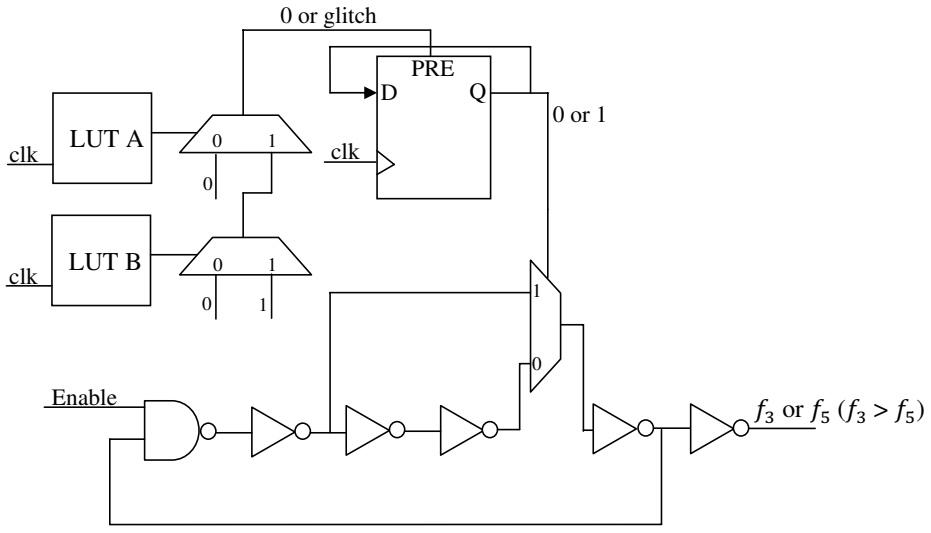

Figure 3.4: The proposed RO/Anderson Hybrid PUF structure, method 2

Table 3.2: RO/Anderson Hybrid PUF method 2 different scenarios

	Scenario 1	**Scenario 2**	**Scenario 3**	**Scenario 4**
Instance *i* Anderson output	1	0	0	1
Instance *j* Anderson output	0	0	1	1
Response bit	1	0 or 1	0	0 or 1

The response bit produced in scenario 1 is 1, scenarios 2 and 4 will produce either 0 or 1 (with probability of 0.5 for each of them), and the response bit of scenario 3 is 0. Because the occurrence probability of all scenarios are theoretically equal to 0.25, we expect to have 50% ($= (0.25\times1+0.25\times0.5+0.25\times0.5)\times100\%$)

of the response bits to be 1 and 50% ($= (0.25\times1+0.25\times0.5+0.25\times0.5)\times100\%$) to be 0. This means that the proposed method maintains the 50% uniformity of the response. It also has a potential to improve the uniqueness of the PUF response because it produces the response bit using two sources of randomness instead of one. The implementation results presented at the end of this chapter verify this fact.

An interesting fact about the proposed scheme (Fig. 3.4) is that, even if the occurrence probabilities of the scenarios are not equal to each other, the theoretical uniformity of the response will still remain 50%. The occurrence probability of each scenario is determined by the probability of 1 and 0 generated by the Anderson PUF (or any other utilized PUF, the output of which is connected to the *select* signal of the multiplexer). Let us denote $Prob_1$ as the probability of the generated *select* signal to be 1 and $Prob_0$ as its probability of being 0. Note that $Prob_1 = 1 - Prob_0$. If the uniformity of the utilized PUF is 50%, we will have $Prob_1 = Prob_0 = 0.5$ and thus, the occurrence probability of all scenarios will be equal to 0.25, as discussed before. Here we assume the general case in which $Prob_1$ is not necessarily equal to $Prob_0$. In this case, the occurrence probabilities of scenarios 1 and 3 are equal to $Prob_1 \times Prob_0$, the probability of scenario 2 is equal to $Prob_0^2$, and the probability of scenario 4 is equal to $Prob_1^2$. Therefore, the uniformity of the produced response will be equal to $Prob_1\times Prob_0\times1+Prob_0^2\times 0.5+Prob_1^2\times0.5 = Prob_0\times(1-Prob_0)+0.5\times Prob_0^2+0.5\times(1-Prob_0)^2 = Prob_0-0.5\times Prob_0^2+0.5+0.5\times Prob_0^2-Prob_0 = 0.5$. Thus, even if the utilized PUF which generates the *select* signal does not have a uniform distribution of 0's and 1's, the theoretical uniformity of the proposed PUF scheme will still remain 50%. Note that, the provided probability analysis is true only if the Basic RO PUF is assumed to generate 1 or 0 with equal probability of 0.5. The implementation results in Section 3.5 confirm that this assumption is true.

As another example shown in Fig. 3.5, the RO PUF and the SR Latch PUF are combined using method 2. The Anderson block is simply replaced with the SR Latch unit. Similar to the Anderson PUF, each SR Latch unit generates 1 bit

that can be connected to the select signal of the multiplexer. Note that, a flip-flop is inserted on the *enable* input of the RO PUF so that both PUF units (RO and SR Latch) are enabled synchronously.

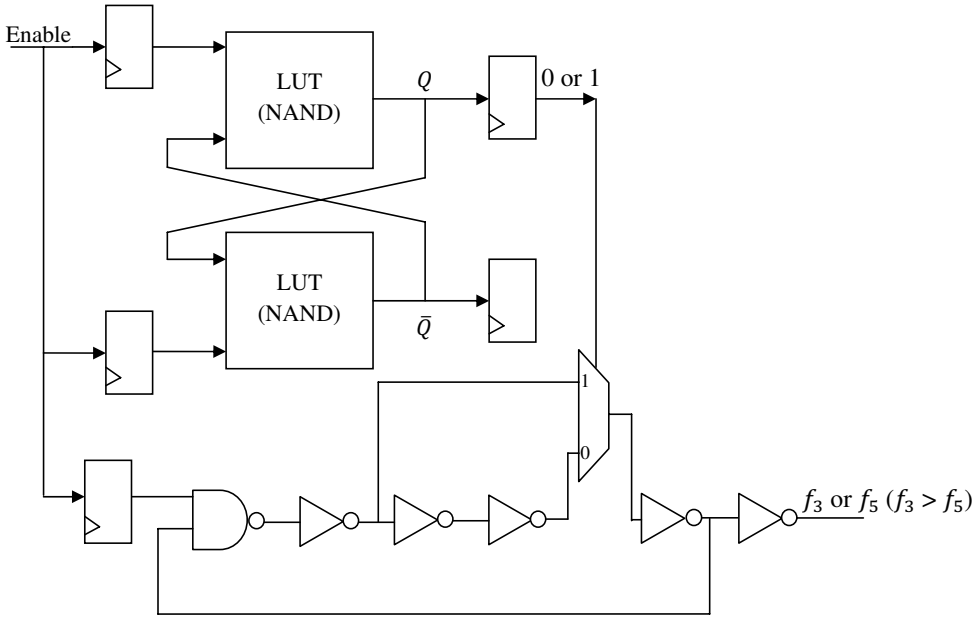

Figure 3.5: The proposed RO/SR Latch Hybrid PUF structure

3.4 Implementation Details and the Measurement System

In this section, the PUF design parameters and the response measurement system are discussed.

3.4.1 Design Parameters

The design parameters used in this study are listed in Table 3.3. The number of PUF units is determined by the required number of response bits and is only limited by the area constraints. The number of PUF units is set to 128 in all structures but it can be much greater than this value. So, the number of response bits is 128 in the Anderson and the SR Latch PUFs. Also, the number of response bits is 127

in the Basic RO PUF, RO/Anderson Hybrid PUF method 1, RO/Anderson Hybrid PUF method 2, and the RO/SR Latch Hybrid PUF.

Table 3.3: PUF implementation parameters

Parameter	Symbol	Value
FPGA chip	N/A	Xilinx VIRTEX II Pro XC2VP100
Number of PUF instances (FPGA boards)	g	4
Ref_Counter clock frequency	Ref_Clock	25 MHz
Number of PUF units in the challenge-response system	N	128
Number of stages in each RO	u	5
Number of intermediate blocks in Anderson PUF	N/A	5
Distance between the 2 NAND gates in the SR Latch	N/A	1 CLB
Number of response bits	a	128 in Anderson PUF 128 in SR Latch PUF 127 in Basic RO PUF 127 in RO/Anderson Hybrid PUF method 1 127 in RO/Anderson Hybrid PUF method 2 127 in RO/SR Latch Hybrid PUF
Counter run time	Run Time	30000 clock cycles
Counter size	N/A	32 bits
Number of response samples	m	50
PUF unit placement	N/A	2-Dimensional

Also, the number of stages in the ring oscillator design is set to 5 which means that 5 inverters are used in the feedback loop. Each ring oscillator uses 5 SLICEs in 2 configurable logic blocks (CLBs). Note that, the ring oscillator is created as a hard macro and is instantiated as many times as needed (128) in the top-level PUF design. Figure 3.6 shows four identical ring oscillators implemented as hard macros. All the ring oscillators are identically placed and routed. Thus, the only random parameter which influences on the oscillation frequency is the delay of the inverters. We have also used the hard macro technique in the SR Latch PUF and the RO/SR Latch Hybrid PUF. Note that we cannot use the hard macro technique in the other PUF structures, Anderson PUF, RO/Anderson Hybrid PUF method 1,

and RO/Anderson Hybrid PUF method 2, because in these structures, the PUF unit requires a power (VCC/GND) component. Power components are no longer supported in the 5.1i FPGA Editor and later when creating hard macros. Therefore, a similar technique proposed in [8] is used to make sure that the placement and routing of all units are identical. This technique uses the `rloc` and `AREA_GROUP` physical constraints. Using these constraints, the designer can force the placement and relative location of the components.

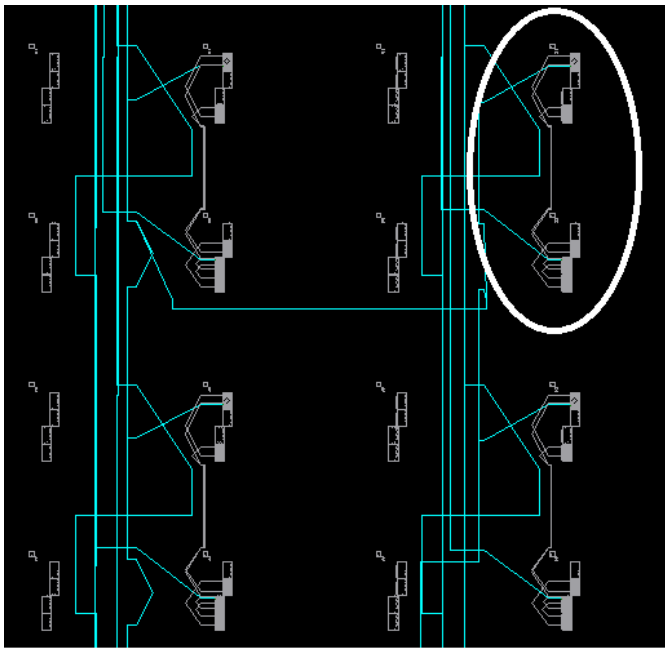

Figure 3.6: Four identical ring oscillators implemented as hard macros: the logic in the white ellipse represents 1 ring oscillator.

Another important design parameter is the placement of the PUF units in the chip. It is shown in [43] that, a systematic variation in the components delays in a die exists in an FPGA chip. In other words, the correlated intra-die variation causes a systematic pattern of the frequencies of several ring oscillators. As suggested by Maiti et al. in [3], the PUF units should be placed as close as possible to each other to eliminate the effect of correlated or spatial intra-die variation. Several placement strategies are analyzed in [13]. Based on our initial experiments on different placement strategies, we consider only the 2-Dimensional placement strategy depicted in Fig. 3.7. In fact, this strategy shows the highest reliability

and uniqueness among other strategies. All the PUF units in all PUF structures are placed in the `SLICE_X56Y88:SLICE_X87Y183` range so that the comparison between different structures is fair.

Finally, after implementing the RO/SR Latch Hybrid PUF scheme, it was noticed that the SR Latch PUF is so sensitive to the surrounding logic. In other words, the high frequency clocks generated by the RO units affect the performance of the SR Latch and, as shown in Section 3.5, the reliability of this scheme is the worst among all other implemented schemes. Also, the uniqueness of this scheme does not seem to improve significantly.

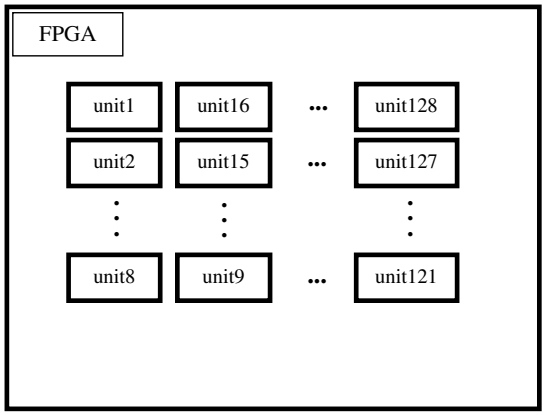

Figure 3.7: 2-Dimensional placement of PUF units

So, we separate the RO and SR Latch units as shown in Fig. 3.8 to reduce the effect of the surrounding logic. It is shown in Section 3.5 that, separating the RO and SR Latch units improves the reliability and uniqueness of the RO/SR Latch Hybrid PUF scheme, significantly. The same method was applied on the RO/Anderson Hybrid PUF method 2 but the improvements were negligible. It is concluded that the Anderson PUF is not affected by the surrounding logic as much as the SR Latch PUF.

3.4.2 Measurement System

Figure 3.9 depicts our measurement system block diagram. The user sends a *start* signal via keyboard. It is transmitted to the FPGA board through the serial port

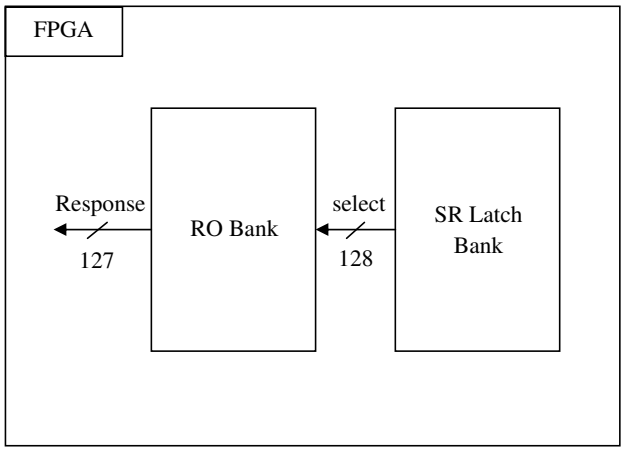

Figure 3.8: Separating the RO/SR Latch PUF units

of the computer. After receiving the start signal, the PUF starts generating the response bits and when the response is ready, it is transmitted to the computer through the serial transmitter. Note that, in this system, the user does not provide the challenge to the PUF; instead, the challenges are applied internally to the PUFs to produce the response bits. In addition, in the Anderson PUF and the SR Latch PUF structures, all the 128 bits of response are obtained at the same time. However, in a practical PUF, the user applies a specific challenge and checks the response to verify the PUF instance. The challenge-response system implemented in this work is only intended to analyze the performance of the PUF structures in terms of reliability, uniqueness, uniformity, and bit-aliasing.

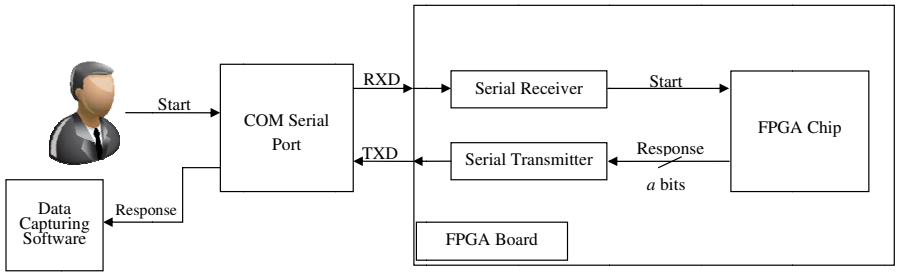

Figure 3.9: The measurement system block diagram

3.5 Results and Discussion

In this section, we provide the implementation results of the seven PUF structures: SR Latch PUF, Basic RO PUF, Anderson PUF, RO/Anderson Hybrid PUF method 1, RO/Anderson Hybrid PUF method 2, and two implementations of the RO/SR Latch Hybrid PUF. We compare the performance of all structures in terms of reliability, uniqueness, uniformity, and bit-aliasing. The reliability of all structures are also analyzed under different environmental conditions such as different FPGA chip ambient temperatures and supply voltage values. The schemes under study are also compared with each other in terms of the area consumption of their PUF units. Table 3.4 presents the implementation results in terms of reliability, uniformity, and uniqueness. In addition, figures 3.10 and 3.18 compare the uniqueness and bit-aliasing of different structures, respectively.

As shown in Table 3.4, the average reliability is almost the same for the SR Latch, Basic RO, and Anderson PUFs. Also note that, all the Hybrid PUF schemes show a lower reliability level than other basic schemes (SR Latch, Basic RO, and Anderson PUFs). This is because there are two sources of instability in the Hybrid schemes. The source of instability in the Basic RO PUF is the unstable frequency of a single ring oscillator. In other words, even the frequency of the clock generated by a single RO is not constant. If the frequencies of two adjacent ROs which produce 1 bit of response are so close to each other, the generated response bit will be unstable due to noise effects. The same scenario happens in the Anderson PUF, where a response bit is produced based on the relation between the two shift register-multiplexer delays. If these delays are so close to each other, the produced response bit will be unstable. In the case of the SR Latch PUF, if the difference between the strength of the NAND gates is small, the produced response bit (which is the settling state of the cell) will be unstable.

As mentioned before, the reliability of the RO/SR Latch Hybrid PUF is the worst among all schemes. We separate the RO and SR Latch units to investigate the effect of the surrounding logic on the SR Latch PUF behavior. As shown in

Table 3.4, the reliability of this scheme is significantly improved when the RO and SR Latch units are separated. The same method was applied on the RO/Anderson Hybrid PUF method 2. However, the improvements were not noticeable. Thus, it is concluded that, the Anderson PUF behavior is not affected by the surrounding logic, compared to the SR Latch PUF.

Table 3.4: Implementation results in terms of reliability, uniformity, and uniqueness (%)

PUF Scheme	**Reliability**		**Uniformity**		**Uniqueness**
	Average	Variance	Average	Variance	
SR Latch	98.8291	0.07458	36.5156	53.10403	36.1979
Basic RO	99.0249	0.53151	50.0551	6.16962	39.895
Anderson	98.9927	0.12851	64.1328	372.50034	39.974
RO/Anderson Hybrid PUF method 1	98.444	0.20538	45.0669	17.30003	36.0892
RO/Anderson Hybrid PUF method 2	98.4757	0.44152	45.4016	8.51688	48.5564
RO/SR Latch Hybrid PUF	93.6325	7.09239	48.3071	5.02041	39.6325
RO/SR Latch Hybrid PUF with separated RO/SR Latch blocks	97.5325	1.58409	49.4094	18.75504	46.063

In terms of uniformity, the Basic RO PUF and the RO/SR Latch Hybrid PUF show the best performance among all schemes. The value of 50.0551% for the uniformity of the Basic RO PUF confirms that the assumption made in the probability analysis provided in Section 3.3 is true. Additionally, the RO/Anderson Hybrid PUF schemes maintain an acceptable level of uniformity, although the Anderson PUF does not show a good value for this performance metric. This behavior is totally reasonable based on the probability analysis provided in Section 3.3. The problem with the Anderson PUF scheme is the necessary tuning process. We observed in our experiments that, when we tune the Anderson PUF for 1 FPGA board, other boards do not seem be tuned at all. For example, the uniformity of the tuned board is measured to be 50.6875% which is close to ideal. Now, when the exact same design is implemented on another board, the uniformity shows to be 78.8594%. The high value of variance for uniformity of the Anderson PUF is due to this variation among different boards. In addition, the SR Latch PUF does not show a good value for uniformity. The average uniformity of 36.5156% represents a biased behavior of this PUF toward the bit 0, which is not good for practical purposes.

One of the most important indicators of a PUF randomness is the uniqueness of its produced response. An ideal PUF has a uniqueness of 50%. As we can see in Table 3.4, the RO/Anderson Hybrid PUF method 2 and the RO/SR Latch Hybrid PUF with separated RO/SR Latch blocks show the best performance in terms of uniqueness. This shows that, the proposed method takes advantage of both sources of randomness properly. Figures 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, and 3.17 show the distribution histogram of the Hamming distance between any pair of PUF instances, and Fig. 3.10 compares the distribution histograms of different structures.

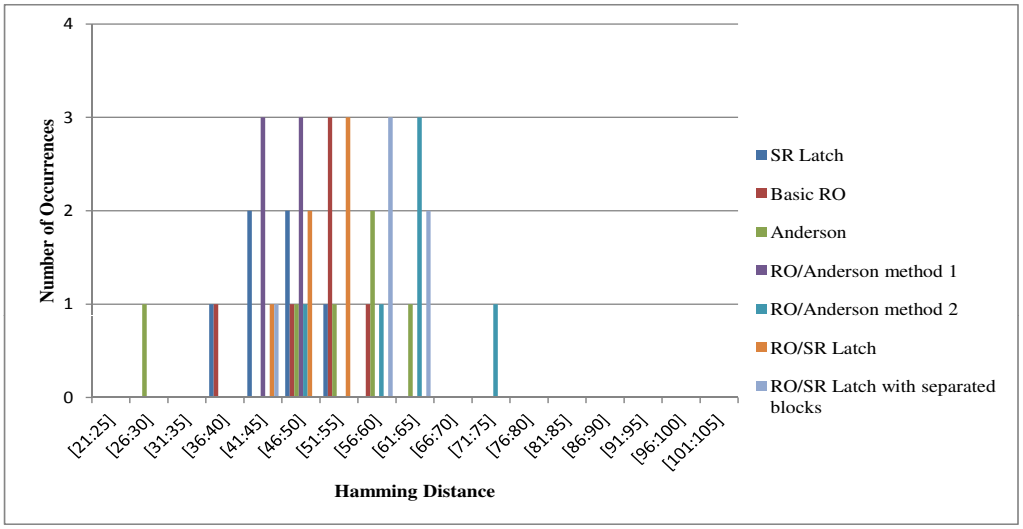

Figure 3.10: Comparison between different schemes in terms of Uniqueness

The best non-ideal PUF would have a normal distribution with the mean value of 0.5. As shown in these figures, the RO/Anderson Hybrid PUF method 2 and the RO/SR Latch Hybrid PUF with separated RO/SR Latch blocks show the best Hamming distance distribution among other structures. Note that, the value of 0.5 is presented using the range [61:65] in the figures because we have 128 (or 127) bits of response in different schemes and $0.5 \times 128 = 64$ (or $0.5 \times 127 = 63.5$).

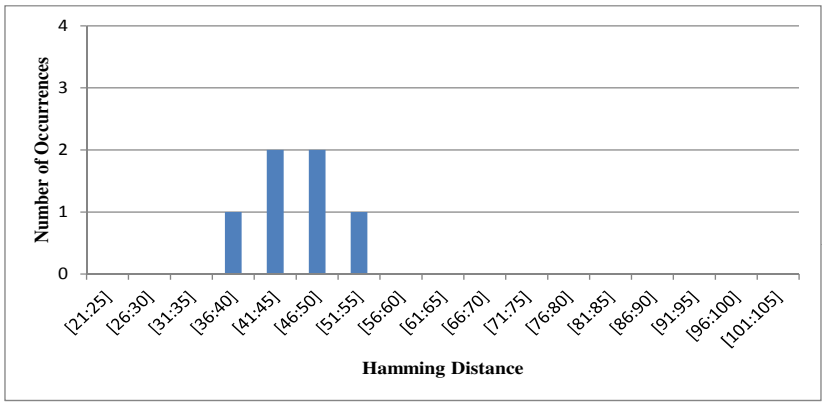

Figure 3.11: Uniqueness of the SR Latch PUF structure

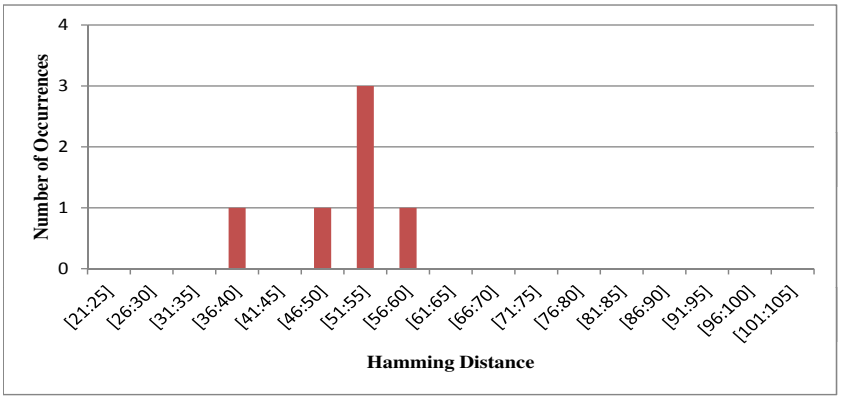

Figure 3.12: Uniqueness of the Basic RO PUF structure

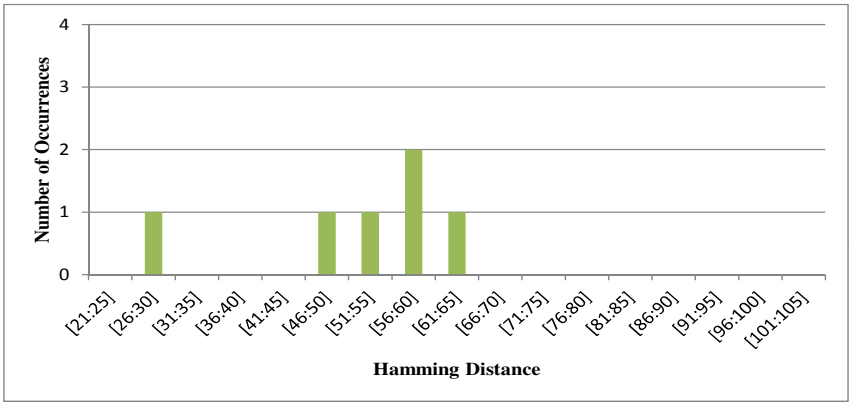

Figure 3.13: Uniqueness of the Anderson PUF structure

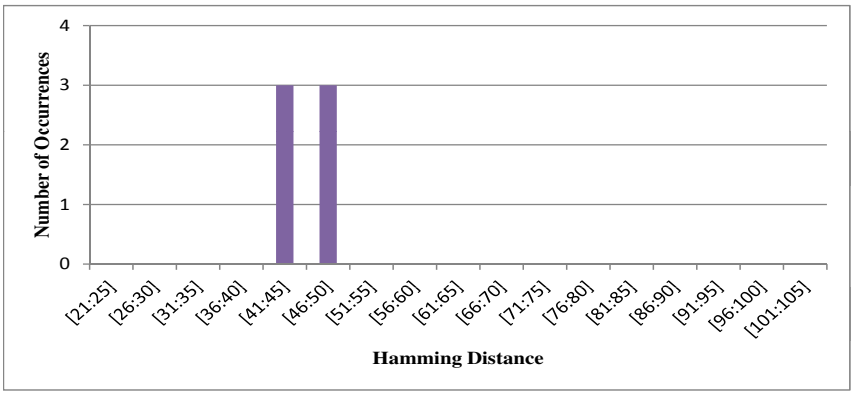

Figure 3.14: Uniqueness of the RO/Anderson Hybrid PUF structure, method 1

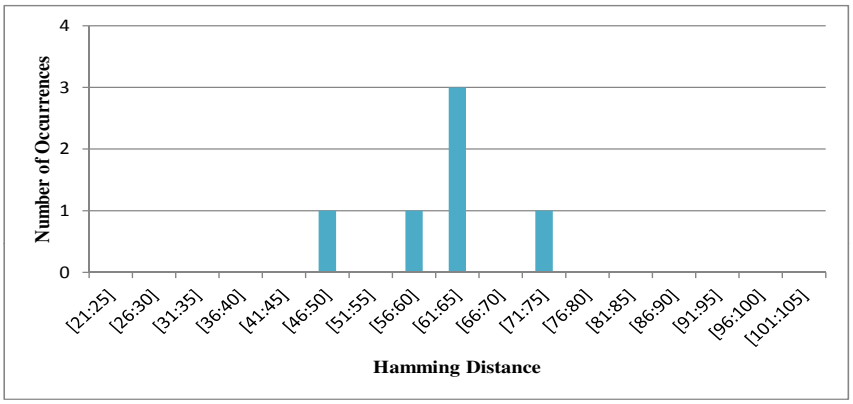

Figure 3.15: Uniqueness of the RO/Anderson Hybrid PUF structure, method 2

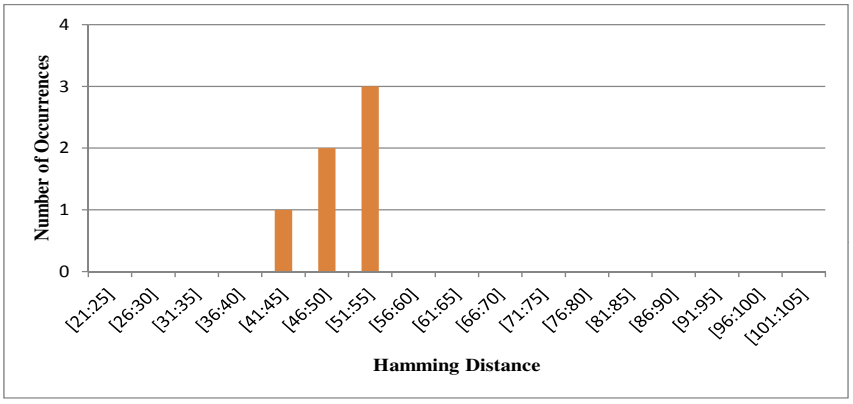

Figure 3.16: Uniqueness of the RO/SR Latch Hybrid PUF structure

Moreover, bit-aliasing of a PUF response is an important factor in assessing the practicality and unclonability of a PUF structure. The ideal PUF has a bit-aliasing of 0.5 for all bit positions. Figures 3.19, 3.20, 3.21, 3.22, 3.23, 3.24,

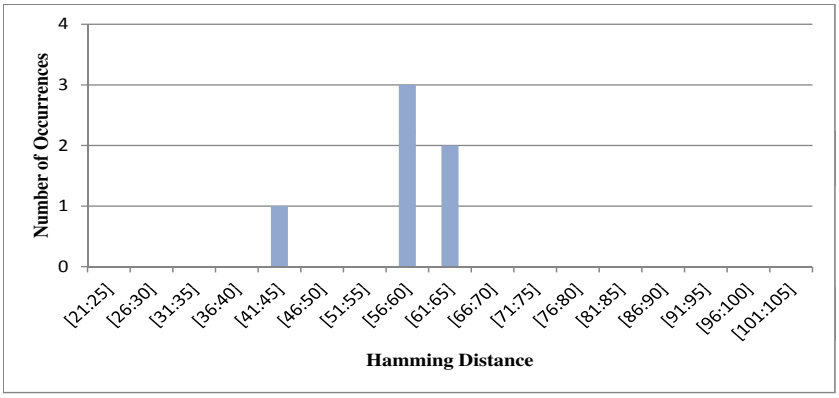

Figure 3.17: Uniqueness of the RO/SR Latch Hybrid PUF structure with separated RO/SR Latch blocks

and 3.25 show the distribution histogram of the average Hamming weight across different PUF instances, and Fig. 3.18 compares the distribution histograms of different structures. The best non-ideal PUF would have a normal distribution with the mean value of 0.5. So, the best schemes in terms of bit-aliasing among the schemes under study are the RO/Anderson Hybrid PUF method 2 and the RO/SR Latch Hybrid PUF with separated RO/SR Latch blocks. This also verifies the suitability of the proposed Hybrid scheme as a practical PUF.

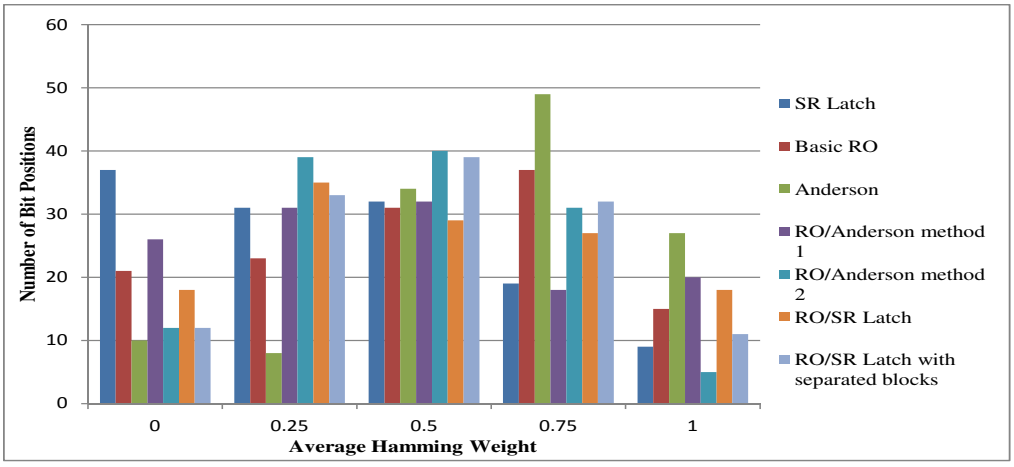

Figure 3.18: Comparison between different schemes in terms of bit-aliasing

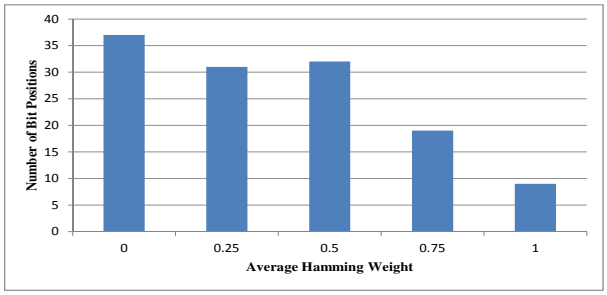

Figure 3.19: Bit-aliasing of the SR Latch PUF structure

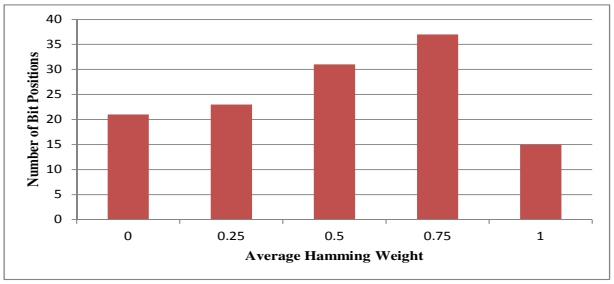

Figure 3.20: Bit-aliasing of the Basic RO PUF structure

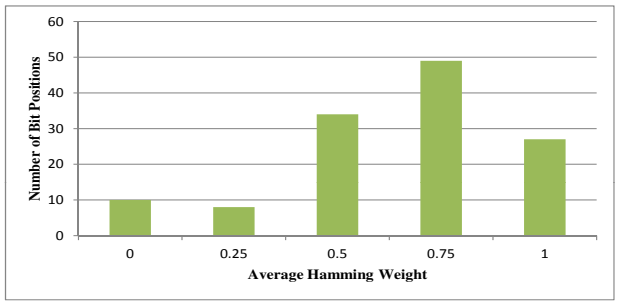

Figure 3.21: Bit-aliasing of the Anderson PUF structure

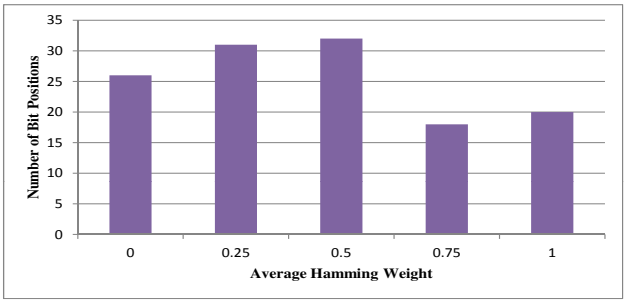

Figure 3.22: Bit-aliasing of the RO/Anderson Hybrid PUF structure, method 1

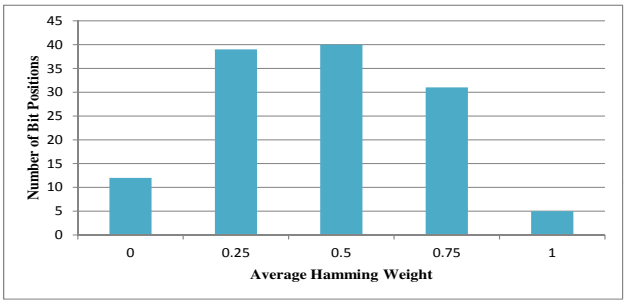

Figure 3.23: Bit-aliasing of the RO/Anderson Hybrid PUF structure, method 2

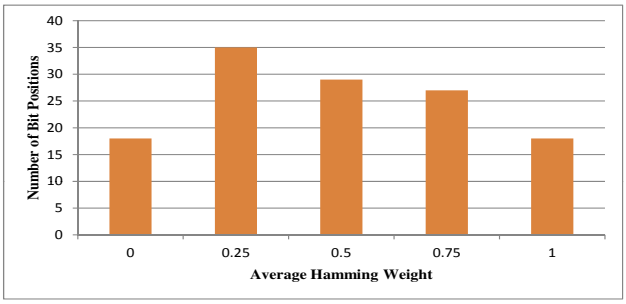

Figure 3.24: Bit-aliasing of the RO/SR Latch Hybrid PUF structure

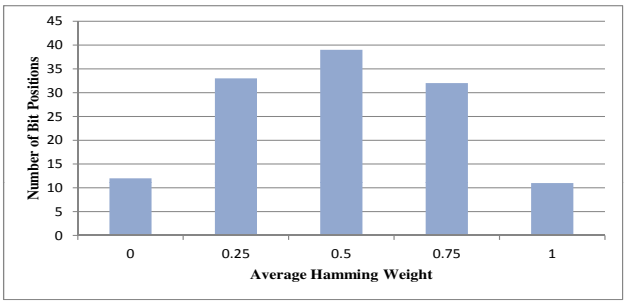

Figure 3.25: Bit-aliasing of the RO/SR Latch Hybrid PUF structure with separated RO/SR Latch blocks

Additionally, the effect of the FPGA board ambient temperature on the reliability of different PUF structures is investigated in this study. The ambient temperature is increased up to 70 degrees (°C) and the reliability of different schemes is recorded for the temperatures 35, 40, 50, 60, and 70 °C. Figure 3.26 shows the results of this experiment. Note that, since the RO/SR Latch Hybrid PUF structure has the worst reliability among all other schemes, the reliability behaviors of other structures are not distinguished well in Fig. 3.26. This is why we have also provided Fig. 3.27 which is basically the same as Fig. 3.26 with the RO/SR Latch Hybrid PUF structure removed so that the range of the graph distinguishes between different schemes, well. Changing the ambient temperature (in the studied range) does not seem to have a significant influence on the reliability of different structures. Since all the PUF structures under study produce response bits based on the mismatch between any two devices (e.g. the NAND gates in the SR Latch PUF, the shift register-multiplexer in the Anderson PUF, RO frequency in the Basic RO PUF, etc.), the temperature influences on the pair of devices almost equally and in the same direction. Thus, the stability of the PUF response bits is not affected by changing the ambient temperature, significantly.

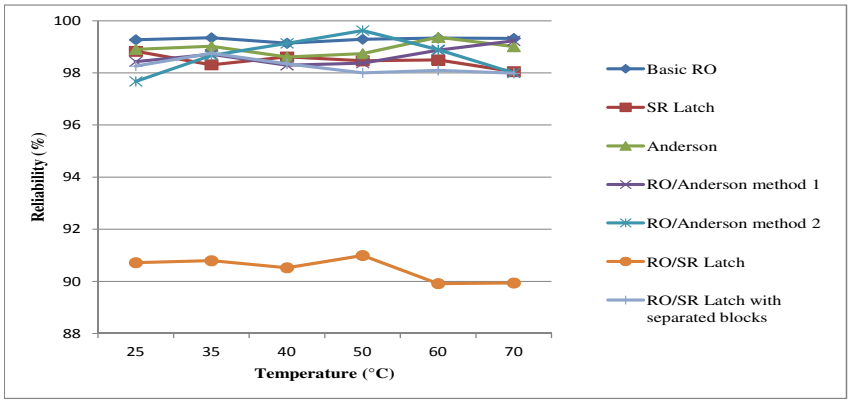

Figure 3.26: The effect of the ambient temperature on the reliability of different PUF structures

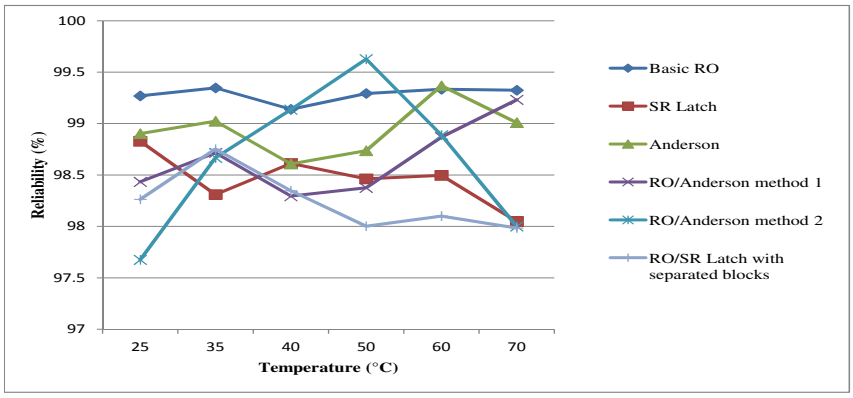

Figure 3.27: The effect of the ambient temperature on the reliability of all PUF structures except the RO/SR Latch Hybrid PUF structure

Moreover, we analyze the effect of changing the FPGA chip supply voltage on the reliability of different PUF schemes. The nominal supply voltage of the chip is 1.5 V and we change it to 1.13, 1.22, 1.37, and 1.72 V. The result of this experiment is depicted in Fig. 3.28. Note that, the exact measured value for the nominal voltage is 1.52 V. It can be observed that, changing the chip supply voltage does not significantly influence the reliability of the Basic RO PUF, SR Latch PUF, Anderson PUF, RO/Anderson Hybrid PUF method 2, and RO/SR Latch Hybrid PUF with separated blocks. All these schemes maintain their reliability at a high level even with the decrease or increase in the chip supply voltage. On the other hand, the RO/SR latch Hybrid PUF shows an interesting trend for its reliability. In fact, the nominal voltage surprisingly results in the worst reliability among other values for the chip supply voltage. In addition, the reliability of the RO/Anderson Hybrid PUF method 1 with the chip supply voltage of 1.13 V is equal to 71.8663%, which is the worst among all cases. The reliability of this scheme shows the highest sensitivity to the chip supply voltage, since it increases significantly when the supply voltage is increased from 1.13 V to 1.22 V (increases to 94.9889%). It also shows a noticeable growth (from 94.9889% to 99.6362%) when the supply voltage is changed from 1.22 V to 1.37 V.

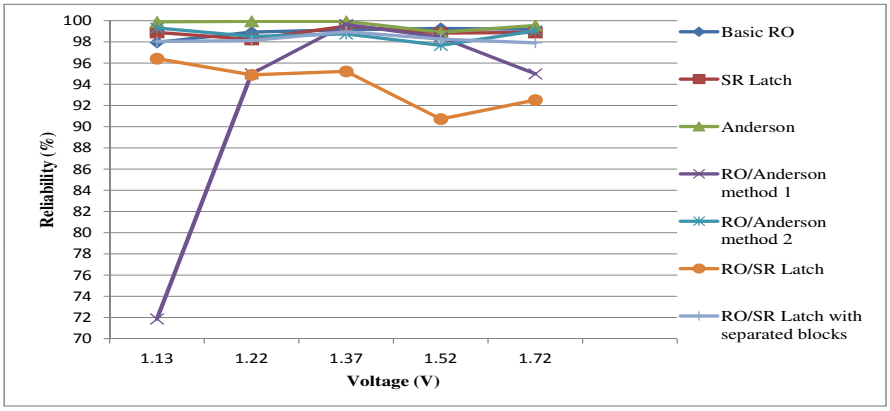

Figure 3.28: The effect of the FPGA chip supply voltage on the reliability of different PUF structures

Table 3.5: Comparison between different structures in terms of area consumption, placement tuning requirement, and time per response bit

PUF Scheme	Area			Tuning Required?	Time per bit (μs)
	No. of CLBs	No. of SLICEs	No. of LUTs		
Basic RO	2	5	6	No	1200
Anderson	5	6	8	Yes	0.08
SR Latch	2	2	2	Yes	0.04
RO/Anderson Hybrid PUF method 1	6	12	7	Yes	1200
RO/Anderson Hybrid PUF method 2	5	12	15	Yes	1200
RO/SR Latch Hybrid PUF	4	9	9	Yes	1200

Finally, Table 3.5 compares different implemented structures in terms of area consumption, placement tuning requirement, and required time per response bit. Note that, except the Basic RO PUF, all other PUF units need a tuning process in order to achieve the best randomness among the response bits. As we can notice, all Hybrid PUF structures consume more area and logic resources than the basic structures. Regarding the required time to produce a response bit, the Basic RO PUF and all Hybrid schemes which are based on RO PUF consume much greater time than the Anderson and SR Latch PUFs. This is because of the *Run Time* that needs to be passed so that the produced response bit is more stable.

In summary, the proposed method of combining different PUF schemes is shown to improve the uniqueness and thus, the randomness of the response, significantly. The added area overhead in our scheme is very small compared to

other available methods in the literature. Also, our proposed Hybrid PUF does not require any pre-processing or post-processing tasks performed on the input or output of the PUF units. In addition to the uniqueness being increased significantly, other important PUF performance metrics are maintained at an acceptable level. The proposed method is a general approach and can be further studied and analyzed using other PUF schemes.

Chapter 4

Secret Sharing Based on Physically Unclonable Functions

As previously discussed, two main categories of applications for PUFs have been introduced and analyzed: authentication and secret key generation. In this chapter, we introduce another important application for PUFs. In fact, we develop a novel and efficient secret sharing scheme using a PUF to increase the information rate and provide cheater detection capability for the secret sharing system.

4.1 Introduction

Secret sharing is a fundamental cryptographic primitive which is used in numerous applications such as secure information storage, Byzantine agreement [44], threshold cryptography [45], secure multiparty computations [46–48], access control [49], attribute-based encryption [50, 51], and generalized oblivious transfer [52, 53]. Secret sharing was invented independently by Shamir [54] and Blakley [55] in 1979. In a basic scheme of secret sharing there are n players and a dealer who has a secret s. The dealer divides the secret into n shares and gives a share to each of the n players p_i, $1 \leq i \leq n$. This step is called the share computation phase. Let us denote P as the set of all players in the system. An authorized set is defined as any subset of P that can reconstruct the secret, in the

secret reconstruction phase, only when all the players in that subset are present. In fact, if at least one player in the authorized set is not present, other players in that authorized set will have no information about the secret. The collection of all the authorized sets is defined as the access structure of the scheme. Shamir's secret sharing scheme is based on the polynomial interpolation. In this scheme, the dealer builds a random polynomial $f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{k-1}x^{k-1}$ by selecting $k-1$ random coefficients, a_1, a_2, ..., a_{k-1}. Also, a_0 is equal to the secret to be shared, s. Then, the dealer obtain n points on the polynomial, $(i, f(i))$, where $i = 1, 2, \ldots, n$. Each player receives a point as his share. Because the polynomial is of degree $k-1$, at least k points are needed to calculate all coefficients and reconstruct the polynomial. Once the coefficients are computed, a_0 will be saved as the secret, s. This scheme is called (n, k)-threshold secret sharing scheme, meaning that, any subset of n players whose cardinality is equal to or greater than a threshold, k, will be considered as an authorized set. In other words, any k players will be able to reconstruct the secret, while less than k players will have no information about the secret. Ito et al. proposed and constructed secret sharing schemes for general access structures in [56]. Even more efficient schemes were proposed in [57–61].

Information rate is an important efficiency metric of a secret sharing scheme and is defined as [62]:

$$\rho = \frac{\log |S|}{\max \log |S_p|} \tag{4.1}$$

where S is the set of secrets, $|S|$ is the bit-size of the set S, and for any $p \in P$, the share of the player p is taken from the set S_p. In addition, $\max$ represents the maximum function. In fact, the information rate, ρ, of a secret sharing scheme is the ratio between the bit-size of the set of secrets and the maximum bit-size of the corresponding shares given to the players. A secret sharing scheme is called ideal if $\rho = 1$. Improving the information rate of a secret sharing scheme is an important concern [63], which is addressed in this study.

Another important issue associated with secret sharing schemes is dealing with dishonest players. In such scenarios, if there is at least one player who fakes his share, then other honest players cannot gain access to the secret. With some of the shares being faked, the reconstructed secret, s', will be different from the original secret, i.e., $s' \neq s$. A robust secret sharing scheme is a scheme which can recover the shared secret even with the existence of some incorrect shares [64]. In fact, if up to t of the shares submitted by the players are fake, a robust secret sharing scheme can still retrieve the original secret. Some known robust schemes are covered in the next subsection. The robustness can be provided by attaching additional redundancy to the shares given to the players. Adding redundancy to the shares reduces the information rate, significantly. In this work, we address this problem and propose, for the first time, a structural model of an efficient secret sharing scheme with cheater detection capability, based on physically unclonable functions. This scheme can be a new application of PUFs, in addition to authentication and secret key generation. Note that, with cheater detection capability, the secret sharing scheme will not have to reconstruct the secret even with some fake shares because there is always the possibility that the fake share is provided by an illegitimate player. In this case, the scheme can identify and simply exclude the cheaters from the authorized set.

4.2 Related Work

In this section, we briefly discuss the known robust secret sharing schemes. As mentioned in [64], the robust secret sharing is impossible if the number of fake shares is equal or greater than half of the players, i.e., $t \geq \lceil n/2 \rceil$. There are two main classes of secret sharing schemes which are robust in the case of $t < \lceil n/2 \rceil$ [64]. The first one is the one proposed by Rabin and Ben-Or in [65] based on an unconditionally secure message authentication code. In this scheme, each player, p_j, $1 \leq j \leq n$, receives an authentication key, key_{ji}. The player p_j can use his authentication key, key_{ji}, to verify if the share provided by player p_i, $1 \leq i \leq n$ and $i \neq j$, is correct. In other words, each player can verify the correctness of all

the shares with the help of his authentication keys. When a share is verified to be correct by at least $t + 1$ players, it will be used to reconstruct the secret. As we can see, each player receives $\Omega(n\lambda)$ bits in addition to their Shamir share, where $2^{-\lambda}$ is the required failure probability in reconstructing the correct secret [64], n is the number of players, and Ω declares the lower bound notation.

The second scheme is proposed by Cramer, Damgård, and Fehr [66]. In this scheme, the dealer shares the secret s, which is an element of the finite field $\mathbb{F}$, i.e., $s \in \mathbb{F}$ among all the players using the standard Shamir scheme. In addition to the original secret, the dealer shares a randomly chosen field element $y \in \mathbb{F}$ and their product $z = s \cdot y \in \mathbb{F}$ among the players. In the reconstruction phase, the reconstructor performs the following for every subset of $t + 1$ players: he reconstructs s', y', and z' which are supposed to be the secret, the random element, and their product, respectively. Then, if the equation $s' \cdot y' = z'$ holds, it outputs s' to be the original secret. In fact, using the redundant information, y and z, given to the players in addition to the actual secret, s, the reconstructor can retrieve the secret with possibly partly incorrect shares of these $t + 1$ players. Compared to Rabin and Ben-Or scheme, this scheme adds much less redundancy to the actual share. However, the running time of this scheme is exponential in n, because the reconstructor has to loop over all possible subsets of size $t + 1$ [64]. A new robust secret sharing scheme that has the advantages of both the above schemes is proposed in [64]. In fact, this scheme has the same low share size as Rabin and Ben-Or scheme and yet, its share computation and secret reconstruction phases run in polynomial time. The important problem associated with these schemes is that they are not efficient schemes in terms of information rate. In this study, a general method is proposed to build the existing secret construction schemes using physically unclonable functions to improve efficiency.

4.3 Preliminaries

We construct our robust secret sharing scheme based on Ito, Saito, and Nishizeki's construction [56]. Note that, with a slight modification, our proposed model can

be built based on other constructions, as well. In addition, the PUF which is used in our design is a controlled PUF. Therefore, in this section, we briefly describe and review Ito et al.'s secret construction and controlled PUFs.

4.3.1 Ito, Saito, and Nishizeki's Constructions

A *monotone* access structure is an access structure in which, any subset that contains an authorized set, is also an authorized set. In Ito et al.'s secret construction, the dealer shares the secret, s, independently for each authorized set $\Gamma \in \mathcal{H}$, where $\mathcal{H}$ is any monotone access structure. Let us assume that Γ includes l players. The dealer chooses $l-1$ random strings of bit-size equal to that of the secret, denoted as $r_1, r_2, \ldots, r_{l-1}$. The required length of these strings is determined by the secret bit-size. The dealer then computes $r_l = s \oplus r_1 \oplus r_2 \ldots \oplus r_{l-1}$, where $\oplus$ is the bitwise XOR operation. Next, the dealer gives the share r_i to the player $p_i \in \Gamma$. Note that, the random strings selected by the dealer should be independent for each player of each set $\Gamma \in \mathcal{H}$. The reconstruction of the secret can be done only when all the players in the set Γ pool their shares and compute $s = r_1 \oplus r_2 \ldots \oplus r_{l-1} \oplus r_l$. On the other hand, any unauthorized set of players which misses at least one player from each authorized set will have no information about the secret. For the case where at least one player from the authorized set fakes his share, the reconstructed secret will be different from the original secret. Although in a general access structure, any player can be a member of more than one authorized set, in this study, we consider the case in which each player is included in only one authorized set, for simplicity. However, this scheme can be extended for the general case, as well. In this case, each player receives only one random string as their share and because the bit-size of the random strings should be equal to that of the secret in the basic scheme, the information rate will be 1 which indicates an ideal secret sharing scheme.

4.3.2 Controlled PUFs

The idea of controlled PUFs is introduced in [11, 12]. As shown in Fig. 4.1, the idea is to apply an error correcting code (ECC) on the output of the PUF to

improve its reliability. In addition, two hash functions are applied on the challenge bits and the ECC output bits in order to restrict the attacker's direct access to the PUF challenge and response bits.

The actual challenge which is applied to the PUF and the PUF actual response are c and r, respectively. However, the challenge and response that are exposed to the outside world are C and R, and because a hash function is a one-way function, the actual challenge and response (c and r) cannot be accessed from the outside world. Of course, if the utilized hash algorithm is known to public, given the challenge C, one can easily compute the actual challenge c ($c = hash(C)$). But because hash functions are one-way functions, given the response R, one cannot obtain the actual response r. In addition, some of the PUF response bits are erroneous or unstable due to noise effects and environmental variations. The error correcting code (ECC) can detect and correct these noisy bits with the help of the Helper Data (W). In fact, for each challenge, C, the actual response is fed into an ECC encoder to produce the helper data. This helper data along with the challenge, C, are used to produce a 100% noise-free response, R, i.e., $R = CPUF(C, W)$.

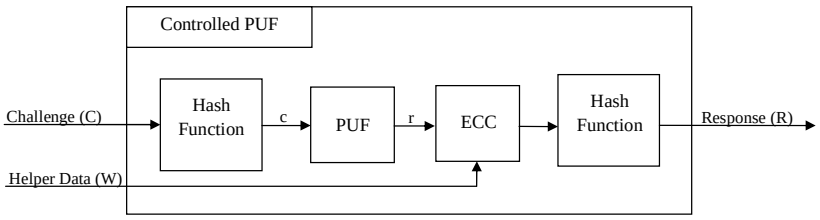

Figure 4.1: The basic idea of a controlled PUF [11, 12]

4.4 Our Proposed Model

4.4.1 Basic Scheme

In this subsection, we discuss our secret sharing scheme using a controlled physically unclonable function (CPUF) based on Ito et al.'s secret construction scheme.

This scheme has three phases, initialization, share computation, and secret reconstruction. In the initialization phase, the trusted dealer applies different challenges with different bit-sizes to the controlled PUF and stores the corresponding challenge-response pair in a secure database. It is explained in the security analysis subsection why different bit-sizes are needed. Also, a general rule on how to decide the challenge bit-sizes is explained in that subsection. The dealer also produces the helper data (W) from each actual response (r) and stores it in the database along with the corresponding challenge-response pair. In fact, the dealer stores the challenge-response-helper data (C,R,W) in the database. Note that, the database should be a part of the dealer and it is assumed that the dealer cannot be hacked, because if its security is compromised, the secret can be read out. Also note that, the hash function input can be of variable bit-size while its output bit-size is fixed. Therefore, the challenges applied to the CPUF can have different bit-sizes. In the share computation phase, the dealer chooses $l-1$ different responses from the database, $R_1, R_2, \ldots, R_{l-1}$ and computes $R_l = s \oplus R_1 \oplus R_2 \ldots \oplus R_{l-1}$. Then, he gives the corresponding challenges that generate the chosen responses along with the helper data to the players p_i, $1 \leq i \leq l-1$. In other words, the player p_i receives C_i and W_i, where $R_i = CPUF(C_i, W_i)$, for $1 \leq i \leq l-1$. The dealer also gives R_l to the player p_l.

In the reconstruction phase, the players $p_1, p_2, \ldots, p_{l-1}$ apply their challenges to the CPUF and provide the helper data, and the corresponding responses are XOR'ed with one another and the share provided by p_l. This scheme is summarized in Fig. 4.2. Note that, because the CPUF is a one-way and more importantly, an unclonable function, the players will not have access to their actual shares. Therefore, even if all the players are hacked and their stored shares are read out by an attacker, the attacker will not be able to construct the secret unless he has access to the original CPUF. In other words, in the existing schemes, if an attacker has access to all players' shares, he can retrieve the secret at his convenience. That is why most studies assume a limitation on the capability of the attacker on how many players he can hack [62, 64]. However, it is not the case

for our scheme. More importantly, if the attacker clones the CPUF circuit, he will generate a wrong secret using all the shares. In addition, if one of the players of the authorized set is not present in the reconstruction phase, other players will have no information about the secret. This scheme, similar to the original scheme, suffers from having dishonest players among the players in the authorized set. In the next subsection, we propose a modified scheme which has cheater detection capability. But before that, the information rate and the security of the proposed scheme is analyzed briefly.

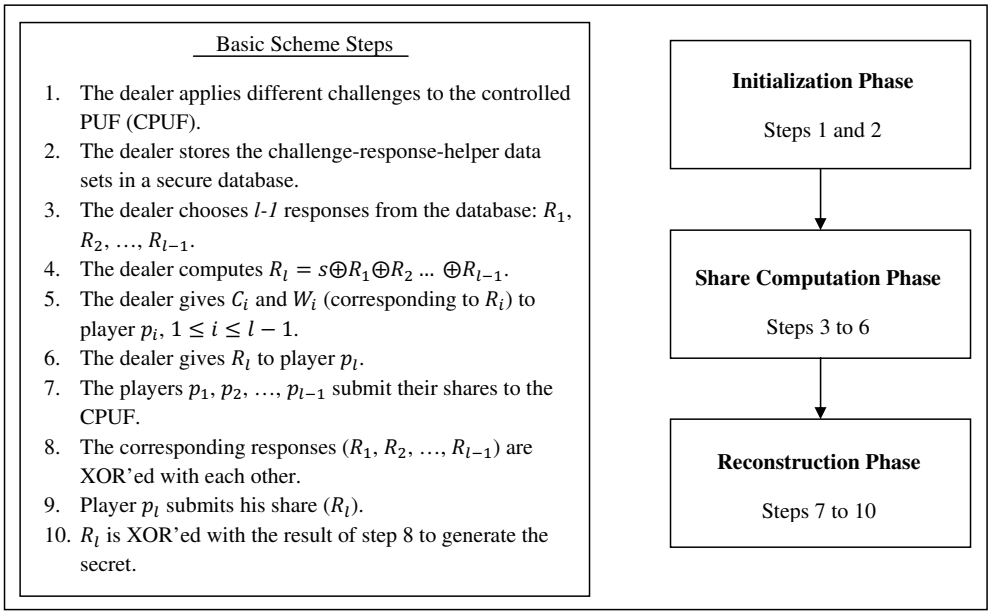

Figure 4.2: The Basic Scheme design steps

Information Rate

In this subsection, the information rate of the proposed scheme is computed and compared with the information rate of the original Ito et al.'s construction scheme [56]. It is shown that, the information rate of the proposed scheme can be even more than 1 while maintaining the required security level. Let us define the response to challenge ratio for a given PUF as the ratio of the generated response bit-size to the challenge bit-size, i.e., $|r|/|c|$. Usually, the bit-size of both challenge and response of a given PUF are fixed. However, in the controlled PUF

scheme shown in Fig. 4.1, the bit-size of the challenge, C, can be variable, because the input to a hash function can be of variable length while its output length is fixed. Also, the PUF should be designed in a way that, its response bit-size is the same as that of the secret, i.e., $|R| = |s|$. Therefore, the share bit-size of $l-1$ players in this scheme will be equal to $|C_i| + |W_i|$, and the share bit-size of player p_l is equal to the secret bit-size which is equal to $|R_i|$. Note that, $|C_i|$ for each player can be different but the bit-size of the helper data (W_i) which is generated from the actual PUF responses is fixed for all players. The bit-size of W_i is determined by the ECC algorithm used in Fig. 4.1. Now, if the following condition is assumed to be true:

$$|C_i| + |W_i| < |R_i| \tag{4.2}$$

for all $1 \leq i \leq l-1$, the information rate will be equal to 1 based on Eq. (4.1), i.e., ideal scheme, because the maximum share bit-size among all the players belongs to the player p_l. The inequality (4.2) will be satisfied only if the number of actual response bits is greater than the number of the corresponding challenge, $|r_i| > |c_i|$ for $1 \leq i \leq l-1$. This condition is a necessary condition but it is not sufficient. The reason is that, the bit-size of the output response, R_i, is less than that of the actual response, r_i, because the input bit-size of the hash function is greater than the output bit-size. Also, because the helper data, W_i may leak some information about the actual response bits, the bit-size of the ECC output is less than $|r_i|$. For the same reason, the input challenge bit-size, $|C_i|$, is greater than the actual challenge bit-size, $|c_i|$. In summary, we have $|C_i| > |c_i|$ and $|R_i| < |r_i|$ for each player p_i, $1 \leq i \leq l-1$. Therefore, having a response to challenge ratio greater than 1 does not necessarily imply $|C_i| + |W_i| < |R_i|$. So, we should make sure that we design the scheme in a way that it satisfies the inequality (4.2).

Up to now, it is shown that, both the original Ito et al.'s scheme and the proposed scheme have an information rate of 1 based on Eq. (4.1). However, our proposed scheme has a very important difference from the original scheme. In

the original scheme, the share bit-size of all the players are equal to that of the secret. However, in our proposed scheme, only one player has a share of bit-size equal to the secret bit-size, and the other players have smaller shares. In fact, if the information rate is defined as [63]:

$$\rho = \frac{\log |S|}{avg \log |S_p|} \tag{4.3}$$

where avg is the average function, the information rate of our proposed scheme will be equal to: $\frac{l \times |R|}{|R| + \sum_{i=1}^{l-1} |C_i| + |W_i|}$ which can be even more than 1 if $|C_i| + |W_i| < |R_i|$ for any $1 \leq i \leq l-1$, as discussed before. In other words, it is a more efficient scheme than the original Ito et al.'s construction scheme and all but one of the players will receive smaller shares. Note that, the information rate of the original scheme is still 1 based on the new equation (4.3).

Security Analysis

For security analysis of the proposed basic scheme, we consider a case where all but one of the players pool their shares and try to retrieve the secret. In the original scheme, because the bit-size of the shares of all players are the same and are equal to the secret bit-size, the complexity of the brute-force approach to guess the remaining share is equal to the complexity of the brute-force approach to guess the secret, i.e., $2^{|s|}$. In the proposed scheme, the share bit-size of all but one of the players is equal to $|C_i| + |W_i|$ which is ideally less than $|R_i| = |s|$, to provide an information rate of more than 1. Therefore, the complexity of the brute-force approach to guess the remaining share is equal to $2^{|C_i|+|W_i|}$ which is less than $2^{|s|}$. However, because the bit-size of the challenge applied to the controlled PUF, $|C_i|$, can be variable, different players might have different share bit-sizes and therefore, in this case, the $l-1$ players will be able to guess the remaining share with a small probability, which depends on the number of different challenge bit-sizes generated by the trusted dealer. For example, let us assume that the secret length is 500 bits and the shares given to the players can be of any length between

200 and 500 bits. When all but one of the players cooperate with each other in order to guess the remaining share and to eventually retrieve the secret, they would notice that their own shares are of different bit-sizes. Therefore, the probability that they will guess the length of the last share correctly will be equal to $1/300$. Note that, this probability is valid only if the players are aware of the range of the valid share lengths. If this is not the case, the probability will be even less than $1/300$. Given this small probability and the complexity of $2^{|C_i|+|W_i|}$, we can claim that, the security of the system is not compromised. The share bit-sizes can be chosen completely randomly. There is no specific requirement on how they are chosen. Also, the range of the bit-sizes is not of critical importance as long as they meet the information rate requirements. However, the number of bit-sizes generated by the trusted dealer should be high enough to increase the resilience of the system against brute-force attack.

Moreover, as previously discussed, even if an attacker obtains access to the share of all the players in the authorized set, he will not be able to construct the secret at his convenience because of the unclonability of the CPUF-based reconstructor.

4.4.2 Modified Scheme With Cheater Detection Capability

In this subsection, we propose a modified scheme, shown in Fig. 4.3 to provide cheater detection capability for the basic scheme. In this scheme, the share bit-size given to each player is exactly the same as in the basic scheme. Therefore, the information rate is the same as that of the basic scheme. However, additional processing is performed by the dealer during the share computation phase and also, extra memory and run time is added to the reconstruction phase. The system works as follows: during the initialization phase, the trusted dealer applies different challenges with different bit-sizes (as in the basic scheme) to the controlled PUF, generates a helper data from each actual PUF response, and stores the corresponding challenge-response-helper data (C_i, R_i, W_i) set in a secure database.

This procedure is similar to the initialization phase of our basic scheme explained in the previous subsection. In the share computation phase, similar to the basic scheme, the dealer chooses $l-1$ responses, $R_1, R_2, \ldots, R_{l-1}$, and gives the corresponding challenge along with the generated helper data to each player. He also gives $R_l = s \oplus R_1 \oplus R_2 \ldots \oplus R_{l-1}$ to the player p_l. The only difference between the modified and the basic schemes is that, in the modified scheme, the dealer must choose different challenge bit-sizes for different players. This requirement is highly recommended in the basic scheme, but in the modified scheme it is necessary. This is because the share bit-size is used as the players' ID and therefore, no redundant information is attached to the original shares to identify the players. This way, we will have a set of valid IDs in the authorized set. Before sending the share to each player, the dealer computes the message authentication code (MAC) of the shares using a MAC algorithm and stores them along with the player's ID in a memory in the controlled PUF system which is used in the reconstruction phase. Note the difference between this memory in the reconstruction system and the database which is part of the dealer. The memory in the reconstruction system can be have public access while the database of the dealer must be kept secure. The dealer then distributes the shares among the players. In the reconstruction phase, each player submits his share to the new system using the controlled PUF shown in Fig. 4.3. In the figure, "Share ($C_i||W_i$)" represents the share submitted by player p_i which is the concatenation of C_i and W_i.

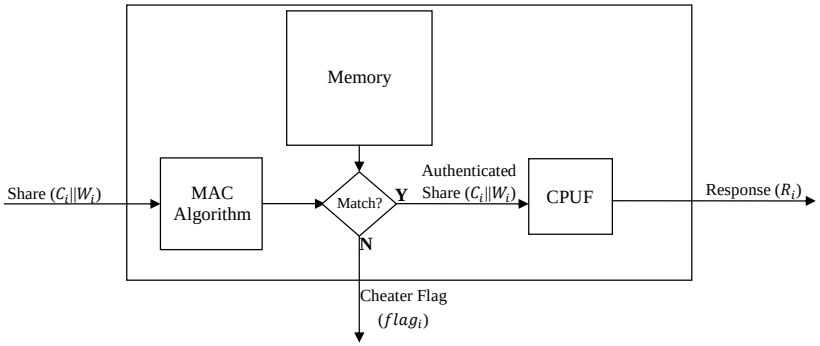

Figure 4.3: The proposed modified scheme with cheater detection capability

The system first computes the MAC of the share and compares it with that stored in the memory. As mentioned before, the system uses the share bit-size to identify the player. If the computed MAC of the share is matched with that stored in the memory, the share (the challenge along with the corresponding helper data) will be applied to the CPUF to generate the corresponding response. This process is performed for all the $l-1$ players and the responses will be bit-wise XOR'ed. Finally, the XOR result will be XOR'ed with the share provided by the player p_l to reconstruct the secret. This scheme is summarized in Fig. 4.4. It is noted that, if one of the players of the authorized set is not present in the reconstruction phase, other players will have no information about the secret. In addition, the cheater detection capability is added to the system using the MAC procedure. If the computed MAC of the share is not matched with that stored in the memory, a flag will be set and the system will identify the cheater and the secret will not be reconstructed. Now, depending on the application and its policies, the cheater can be treated in different ways. He might receive a warning and will have a chance to provide his share again, or his trustability level will be reduced, or he will be removed from the authorized set permanently. Note that, the cheater detection capability of the system depends on the preimage resistance or the one-way property of the hash function used in the MAC algorithm.

If the player fakes his share in a way that, the bit-size of the share is also changed, there will be 2 different cases. In the first case, the system identifies the player as a cheater because the faked share bit-size is not a valid bit-size. In the second case, the system will identify the player as another player in the authorized set (this will happen if the faked share bit-size is a valid bit-size). Then it will compare the MAC of the faked share with the one stored in the memory. If they do not match (which will be the case because of the second preimage resistance property of hash functions), the misidentified player, which could be an honest player, will be identified as a cheater. This problem can be resolved by having the players submit their shares sequentially in a pre-specified order to identify the cheaters correctly. Another way to address this problem is to attach a random ID

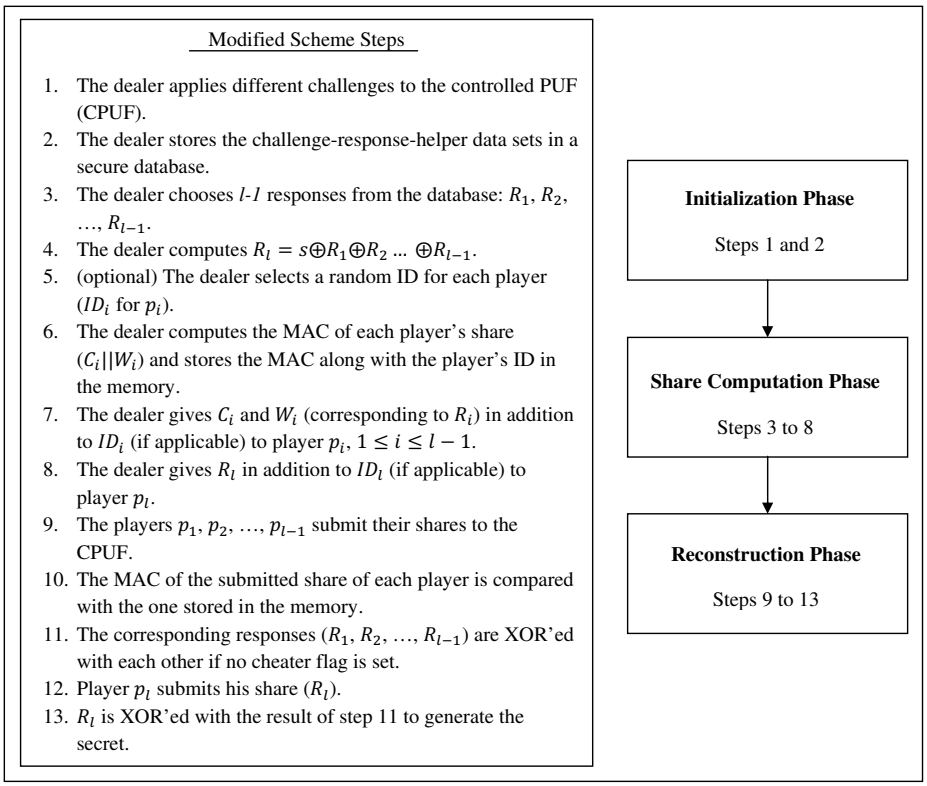

Figure 4.4: The Modified Scheme design steps

to the share given to each player (ID_i given to the player p_i) and storing that ID in the system memory along with the share MAC. In this case, although we lose efficiency and the information rate will be reduced, we can identify such cases if the share bit-size and the random ID submitted by the player do not match. Therefore, the information rate will be equal to: $\frac{l \times |R|}{|R|+|ID_l|+\sum_{i=1}^{l-1}|C_i|+|ID_i|+|W_i|}$, where $|ID_i|$ is the bit-size of the i-th player ID.

Note that, if the dishonest players are able to read out the data stored in the reconstruction system memory via invasive physical attacks, they will not be able to compromise the security of the system because of the second preimage resistance property of hash functions. In other words, the cheater cannot fake his share in a way that, its MAC matches the one stored in the memory. Also note that, it is assumed that the communication between the dealer and the players is secure in all existing schemes, because if it is not, the shares can be seen by other players and the secret sharing scheme will be of no use.

The most important issue in the proposed approach is the case in which the player p_l fakes his share. In fact, when all other players submit their shares correctly and thus, no flag has been set, i.e., $flag_i = 0$ for $1 \leq i \leq l-1$, the player p_l will submit his share to reconstruct the secret. We define this player's share as a *critical* share because of the following reason: if this player fakes his share, the generated secret will be wrong, $s' \neq s$. The fact that the generated secret is wrong and no cheater flag has been set helps the system to identify p_l as the cheater. This problem also exists in the original secret construction scheme. However, in our proposed model, we can absolutely detect this player, p_l as a cheater. Therefore, in the share computation phase, we should make sure that we give this critical share to a player with the highest trustability level. If this player cheats, we can remove him from the authorized set permanently or we can reduce his trustability level, depending on the application policies.

The important issue is that, the player p_l can fake his share so that the generated secret is wrong. Then, he can use his share and the generated wrong secret to compute the correct secret at his convenience. In addition, the dealer can compute the MAC of R_l before giving it to the player p_l and store the MAC in the memory. In the reconstruction phase, the authenticity of the share of this player will also be verified using the same procedure and a flag will be set if it is not matched with the one stored in the memory. Again the bit-size of the share which in this case is equal to that of the secret can be used to identify the player. Because the proposed model helps the system to identify such case, further actions can be made to shut down the system, for example, to re-compute the shares, and to build a new system with a new secret.

Finally, it should be mentioned that, the main contribution of the proposed scheme is its efficiency in terms of information rate and the shares' bit-size. In fact, the idea of computing the MAC of the shares and using the MAC to verify the authenticity of the submitted shares, can be applied to the original Ito et al.'s construction scheme to provide cheater detection capability. However, in that case, the information rate will be 1 at its best. It could be even less than 1 in the

case that, the ID of the players are attached to their actual shares to distinguish between different players. In contrast, in our proposed scheme, the information rate can be more than 1 even in the latter case. We achieve this efficiency at the expense of extra hardware for the CPUF, extra memory for the secure database, and the extra initialization phase where the dealer should apply the challenges and store the challenge-response-helper data sets in its database. Another important contribution of this study is that, the ability of an attacker on hacking the players is not required to be limited to a specific number of players. In other words, an attacker can hack all the players in the access structure, and yet, he will not be able to compute the secret at his own convenience due to the unclonability of the PUF-based reconstructor.

Security Analysis

In subsection 4.4.1 we performed a security analysis for the proposed basic scheme and showed its resilience against the brute-force approach. The same analysis can be performed for the modified scheme against the same attack. In this subsection, another important scenario is considered, in which all but one of the players try to obtain information about the secret by submitting false shares. In other words, dishonest players submit false shares in order to deceive the honest player. Then, based on the obtained incorrect secret and using their correct shares, they will obtain information about the correct secret. This is the main attack on secret sharing schemes and a secure scheme is defined to be resilient against this attack [62]. We can observe that, our scheme is also resilient against this attack because this attack works only for the schemes which cannot detect the cheaters. Our proposed scheme offers cheater detection capability to the system and identifies the dishonest player by setting a cheater flag. In fact, the honest players will not submit their shares when a cheater flag is set. Therefore, dishonest players cannot gain information about other honest players' shares to compute the secret at their own convenience.

4.4.3 PUF requirements

In this subsection, the features of a good PUF that can be used in the proposed model to meet the requirements of our secret sharing scheme are discussed. Besides a high reliability and a close-to-ideal uniqueness which represents the randomness of the response bits, the utilized PUF should have a large set of challenge-response pairs. This allows each challenge-response pair to be used only once to prevent modeling attacks against the PUF design. Also, the number of response bits should be large enough and equal to the secret bit-size. Another important factor of the utilized PUF is the challenge-response ratio which ideally should be much greater than 1, i.e., $|r|/|c| \gg 1$, so that the information rate of the proposed secret sharing scheme can be even more than 1.

Chapter 5

Conclusion and Future Work

In this study, we have implemented three known PUF structures, SR Latch PUF, RO PUF and Anderson PUF, on four identical FPGA boards and have investigated their performance in terms of reliability, uniqueness, uniformity, and bit-aliasing. We have also proposed a Hybrid PUF scheme in which two PUF schemes are combined with each other to improve the randomness of the response. The performance of this scheme is investigated using two examples. In the first one, RO PUF and Anderson PUF are combined with each other; and in the second example, the SR Latch PUF is combined with RO PUF. Implementation results show that, the proposed Hybrid PUF scheme improves the uniqueness and thus, the randomness of the produced response, significantly.

In addition, we have proposed an efficient secret sharing scheme with cheater detection capability based on physically unclonable functions (PUFs). The PUF can generate the random strings that are required for the secret construction scheme. Additionally, the one-way property and the response to challenge ratio of the PUF is used to build our scheme and analyze its efficiency in terms of information rate. It is shown that, under one condition, the information rate can be even more than 1, meaning that the players will receive smaller shares. The proposed scheme can detect the cheaters while maintaining the required efficiency. Also, the security of the proposed scheme against brute-force attack and two other known scenarios is analyzed to show the resilience of the proposed scheme against these attacks.

The results presented in this work are all obtained by implementing the schemes on only four FPGA boards. The statistical analysis will be more valid if a larger number of PUF instances are used. Also, the proposed Hybrid PUF can be implemented in an authentication and/or secret key generation scheme in order to verify its ability to be used in such applications. Additionally, as discussed previously, the proposed Hybrid PUF is a general method and its characteristics can be further investigated using other combinations. Regarding the proposed secret sharing scheme based on PUFs, detailed security analysis can be performed against mush elaborated attacks. Finally, the proposed method can be applied to other existing secret sharing schemes in order to verify its contribution.

Bibliography

[1] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual Design Automation Conference*, ser. DAC '07. New York, NY, USA: ACM, 2007, pp. 9–14.

[2] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. V. Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits with identification and authentication applications," in *In Proceedings of the IEEE VLSI Circuits Symposium*, 2004, pp. 176–179.

[3] A. Maiti and P. Schaumont, "Improving the quality of a Physical Unclonable Function using configurable Ring Oscillators," in *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, Sept. 2009, pp. 703 –707.

[4] X. Xin, J.-P. Kaps, and K. Gaj, "A Configurable Ring-Oscillator-Based PUF for Xilinx FPGAs," in *Proceedings of the 2011 14th Euromicro Conference on Digital System Design*, ser. DSD '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 651–657.

[5] S. S. Mansouri and E. Dubrova, "Ring oscillator physical unclonable function with multi level supply voltages," *CoRR*, vol. abs/1207.4017, 2012.

[6] A. Maiti, I. Kim, and P. Schaumont, "A robust physical unclonable function with enhanced challenge-response set," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 333–345, 2012.

[7] R. Maes, A. V. Herrewege, and I. Verbauwhede, "Pufky: A fully functional puf-based cryptographic key generator," in *CHES*. Springer, 2012, pp. 302–319.

[8] J. H. Anderson, "A PUF design for secure FPGA-based embedded systems," in *15th Asia and South Pacific-Design Automation Conference (ASP-DAC)*, Jan. 2010, pp. 1–6.

[9] K. Shimizu, D. Suzuki, and T. Kasuya, "Glitch puf: Extracting information from usually unwanted glitches," *IEICE Transactions*, vol. 95-A, no. 1, pp. 223–233, 2012.

[10] H. Hata and S. Ichikawa, "Fpga implementation of metastability-based true random number generator," *IEICE Transactions*, vol. 95-D, no. 2, pp. 426–436, 2012.

[11] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Controlled Physical Random Functions," in *Proc. of the 18th Annual Computer Security Applications Conference*, December 2002.

[12] B. Gassend, "Physical Random Functions," Master's thesis, Massachusetts Institute of Technology, Jan. 2003.

[13] D. Merli, F. Stumpf, and C. Eckert, "Improving the quality of ring oscillator pufs on fpgas," in *Proceedings of the 5th Workshop on Embedded Systems Security*, ser. WESS '10. New York, NY, USA: ACM, 2010, pp. 9:1–9:9.

[14] A. Mills, S. Vyas, M. Patterson, C. Sabotta, P. Jones, and J. Zambreno, "Design and Evaluation of a Delay-Based FPGA Physically Unclonable Function," in *30th IEEE International Conference on Computer Design*, Montreal, Canada, Oct. 2012, pp. 143–146.

[15] Srinivasa Ravikanth Pappu, "Physical one-way functions," PhD Thesis, Massachusetts Institute of Technology, 2001.

[16] S. R. Pappu, B. Recht, and N. Gershenfeld, “Physical one-way functions,” Technical Report, Science, vol. 297, no. 5589, pp. 2026–2030, 2002.

[17] Serge Vrijaldenhoven, “Acoustical physical unclonable functions,” Master’s Thesis, TECHNISCHE UNIVERSITEIT EINDHOVEN, 2004.

[18] P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters, “Read-proof hardware from protective coatings,” in *Proceedings of the 8th international conference on Cryptographic Hardware and Embedded Systems*, ser. CHES’06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 369–383.

[19] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, “Fpga intrinsic pufs and their use for ip protection.” in *CHES*, ser. Lecture Notes in Computer Science, vol. 4727. Springer, 2007, pp. 63–80.

[20] “A 1.6pJ/bit 96% Stable Chip-ID Generating Circuit using Process Variations,” in *International Solid-State Circuits Conference (ISSCC)*. IEEE, Feb. 2007, pp. 406–611.

[21] R. Maes , P. Tuyls, and I. Verbauwhede, “Intrinsic PUFs from Flip-flops on Reconfigurable Devices,” in *Benelux Workshop Information and System Security (WISSec 08)*, Eindhoven, The Netherlands, Nov. 2008.

[22] R. Maes, and I. Verbauwhede, “Physically Unclonable Functions: a Study on the State of the Art and Future Research Directions,” In Towards Hardware-Intrinsic Security, Security and Cryptology, D. Naccache, and A. Sadeghi (eds.), 2010.

[23] Martin Deutschmann, “Cryptographic Applications with Physically Unclonable Functions,” Master’s Thesis, Alpen-Adria-Universitat Klagenfurt, Nov. 2010.

[24] E. v. d. S. Peter Simons and V. van der Leest, “Buskeeper PUFs, a Promising Alternative to D Flip-Flop PUFs,” in *IEEE International Symposium on*

Hardware-Oriented Security and Trust (HOST), San Francisco, CA, Jun. 2012.

[25] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005.

[26] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A large-scale characterization of RO-PUF," in *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Anaheim, CA, Jun. 2010.

[27] S. Morozov, A. Maiti, and P. Schaumont, "An Analysis of Delay Based PUF Implementations on FPGA," in *Reconfigurable Computing: Architectures, Tools and Applications*, ser. Lecture Notes in Computer Science, P. Sirisuk, F. Morgan, T. El-Ghazawi, and H. Amano, Eds. Springer Berlin / Heidelberg, 2010, vol. 5992, pp. 382–387.

[28] V. Vivekraja and L. Nazhandali, "Circuit Level Techniques for Reliable Physically Unclonable Functions," in *HOST*, 2009, pp. 30–35.

[29] D. Suzuki and K. Shimizu, "The glitch puf: a new delay-puf architecture exploiting glitch shapes," in *Proceedings of the 12th international conference on Cryptographic hardware and embedded systems*, ser. CHES'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 366–382.

[30] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions," *IACR Cryptology ePrint Archive*, vol. 2011, p. 657, 2011.

[31] G. E. Suh, "AEGIS: A Single-Chip Secure Processor," Master's thesis, Massachusetts Institute of Technology, 2005.

[32] M.-D. M. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions." *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 48–65, 2010.

[33] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient helper data key extractor on fpgas," in *CHES*, 2008, pp. 181–197.

[34] Roel Maes, "Physically Unclonable Functions: Constructions, Properties and Applications," Ph.D. Thesis, Katholieke Universiteit Leuven, 2012.

[35] L. Lin, D. Holcomb, D. K. Krishnappa, P. Shabadi, and W. Burleson, "Low-power sub-threshold design of secure physical unclonable functions," in *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, ser. ISLPED '10. New York, NY, USA: ACM, 2010, pp. 43–48.

[36] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, "Identification and authentication of integrated circuits: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 16, no. 11, pp. 1077–1098, Sep. 2004.

[37] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Testing techniques for hardware security," in *ITC*, 2008, pp. 1–10.

[38] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *ACM Conference on Computer and Communications Security*, 2010, pp. 237–249.

[39] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable pufs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, pp. 5:1–5:33, Mar. 2009.

[40] A. Maiti, L. McDougall, and P. Schaumont, "The impact of aging on an fpga-based physical unclonable function," in *FPL*, 2011, pp. 151–156.

[41] S. Gören, O. Ozkurt, A. Yildiz, H. F. Ugurdag, R. S. Chakraborty, and D. Mukhopadhyay, "Partial bitstream protection for low-cost FPGAs with physical unclonable function, obfuscation, and dynamic partial self reconfiguration," *Computers & Electrical Engineering*, Nov. 2012.

[42] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh, "Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs," in *Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs*, ser. RECONFIG '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 298–303.

[43] N. P. Sedcole and P. Y. K. Cheung, "Within-die delay variability in 90nm FPGAs and beyond," in *FPT*, 2006, pp. 97–104.

[44] M. O. Rabin, "Randomized byzantine generals," in *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, ser. SFCS '83. Washington, DC, USA: IEEE Computer Society, 1983, pp. 403–409.

[45] Y. Desmedt and Y. Frankel, "Shared generation of authenticators and signatures (extended abstract)," in *CRYPTO*, 1991, pp. 457–469.

[46] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for noncryptographic fault-tolerant distributed computations," 1988, pp. 1–10.

[47] D. Chaum, C. Crepeau, and I. Damgård, "Multiparty unconditionally secure protocols," 1988, pp. 11–19.

[48] R. Cramer, I. Damgård, and U. Maurer, "General secure multi-party computation from any linear secret-sharing scheme," in *Proceedings of the 19th international conference on Theory and application of cryptographic techniques*, ser. EUROCRYPT'00. Berlin, Heidelberg: Springer-Verlag, 2000, pp. 316–334.

[49] M. Naor and A. Wool, "Access control and signatures via quorum secret sharing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 9, no. 9, pp. 909–922, 1998.

[50] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *ACM Conference on Computer and Communications Security*, 2006, pp. 89–98.

[51] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," *IACR Cryptology ePrint Archive*, vol. 2008, p. 290, 2008.

[52] B. Shankar, K. Srinathan, and C. P. Rangan, "Alternative protocols for generalized oblivious transfer," in *Proceedings of the 9th international conference on Distributed computing and networking*, ser. ICDCN'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 304–309.

[53] T. Tassa, "Generalized oblivious transfer by secret sharing," *Des. Codes Cryptography*, vol. 58, no. 1, pp. 11–21, Jan. 2011.

[54] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[55] G. R. Blakley, "Safeguarding cryptographic keys," 1979, pp. 313–317.

[56] M. Ito, A. Saito, and T. Nishizeki, "Secret sharing schemes realizing general access structures," 1987, pp. 99–102.

[57] J. C. Benaloh and J. Leichter, "Generalized secret sharing and monotone functions," in *CRYPTO*, 1988, pp. 27–35.

[58] M. Bertilsson and I. Ingemarsson, "A construction of practical secret sharing schemes using linear block codes." in *AUSCRYPT*, ser. Lecture Notes in Computer Science, J. Seberry and Y. Zheng, Eds., vol. 718. Springer, 1992, pp. 67–79.

[59] E. F. Brickell, "Some ideal secret sharing schemes," in *EUROCRYPT*, 1989, pp. 468–475.

[60] M. Karchmer and A. Wigderson, "On span programs," in *In Proc. of the 8th IEEE Structure in Complexity Theory*. IEEE Computer Society Press, 1993, pp. 102–111.

[61] G. J. Simmons, “Geometric shared secret and/or shared control schemes,” in *CRYPTO*, 1990, pp. 216–241.

[62] S. Cabello, C. Padró, and G. Sáez, “Secret sharing schemes with detection of cheaters for a general access structure,” in *Proceedings of the 12th International Symposium on Fundamentals of Computation Theory*, ser. FCT ’99. London, UK, UK: Springer-Verlag, 1999, pp. 185–194.

[63] A. Beimel, “Secret-sharing schemes: a survey,” in *Proceedings of the Third international conference on Coding and cryptology*, ser. IWCC’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 11–46.

[64] A. Cevallos, S. Fehr, R. Ostrovsky, and Y. Rabani, “Unconditionally-secure robust secret sharing with compact shares,” in *EUROCRYPT*, 2012, pp. 195–208.

[65] T. Rabin and M. Ben-Or, “Verifiable secret sharing and multiparty protocols with honest majority,” in *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, ser. STOC ’89. New York, NY, USA: ACM, 1989, pp. 73–85.

[66] R. Cramer, I. Damgård, and S. Fehr, “On the cost of reconstructing a secret, or vss with optimal reconstruction phase,” in *CRYPTO*, 2001, pp. 503–523.

Appendix A

Response Samples

In this appendix, we provide the estimated response of each implemented PUF structure instance. We have seven different structures, SR Latch PUF, Basic RO PUF, Anderson PUF, RO/Anderson Hybrid PUF method 1, RO/Anderson Hybrid PUF method 2, and two implementations of the RO/SR Latch Hybrid PUF which are described in details in Chapter 3. We have also four identical FPGA evaluation boards which use the Xilinx Virtex II Pro XC2VP100 FPGA chip. We implement each PUF structure on each board and we take 50 samples of the produced response of each PUF instance. Based on these samples, we estimate the response of each PUF instance. Note that, the number of response bits of the SR Latch and Anderson PUF is 128 while other PUFs produce 127-bit responses. The responses are represented in the hexadecimal format.

SR Latch PUF:

Board #1: `06 07 8C 81 44 06 06 80 D6 4F CC 94 17 17 18 6A`

Board #2: `4E 84 0C 44 04 18 CE 15 DE 17 4D 1C 87 9D 30 70`

Board #3: `45 87 9F 4D 46 CF 86 41 07 17 05 0E 04 D4 21 69`

Board #4: `15 16 06 05 07 06 44 51 45 10 06 01 86 08 40 21`

Basic RO PUF:

Board #1: `28 88 5B 57 28 82 BD 7F 4A 4C CE F5 58 A8 BD 6B`

Board #2: `4A C8 F5 2D 9D 51 C5 BE AA AA 4D A7 08 A5 B4 AF`

Board #3: `00 44 EA 7E 04 A6 3F DD 4A 4C D7 7D 58 A5 6C EF`

Board #4: `08 A4 D7 EF 45 0A 66 AD 68 84 C5 DF 14 04 72 AB`

Anderson PUF:

Board #1: `FB FF FB 7B DA 77 F3 FF F9 FF DB 7D E9 FF BF DF`

Board #2: `7A 7A B8 7C F2 24 03 F9 4A AF A2 74 D1 7A 4F 14`

Board #3: `FA FD EB BF FB B7 F3 FD 7A FD F1 BF FB 7E DE DF`

Board #4: `5B A0 31 B6 82 A1 80 64 A8 F7 29 25 7A 35 8C 9E`

RO/Anderson Hybrid PUF, method 1:

Board #1: `14 52 C8 53 B2 12 80 9A 85 4E D1 4A D2 5D 58 7B`

Board #2: `15 25 E4 59 FA 2A 89 36 CD 55 D2 5B 90 D5 B3 6B`

Board #3: `10 8B EC 18 B2 02 94 0F 80 57 81 2F 80 57 05 F7`

Board #4: `12 21 F2 16 CB 2A C0 9A 91 55 91 33 91 12 6E F7`

RO/Anderson Hybrid PUF, method 2:

Board #1: `16 18 E2 B3 4A 2B DA 96 42 DA 91 97 D2 EB 6A D4`

Board #2: `11 54 A9 5D 19 0D E0 1E 92 1D 52 64 D5 99 53 5A`

Board #3: `18 23 CA CC A6 6A D2 4A 96 43 32 56 A2 95 92 BE`

Board #4: `12 54 9A A6 A2 17 17 2A 84 36 C0 4A A1 54 96 AF`

RO/SR Latch Hybrid PUF:

Board #1: `14 F1 37 6F 53 DB AB B3 14 0E 43 6A D4 4A 2B EA`

Board #2: `15 14 A2 E6 D5 0D 2A 6A 5C 94 32 6F 94 95 33 43`

Board #3: `14 17 69 6F 14 CD 37 72 C5 95 02 6E 04 17 0A 6F`

Board #4: `55 10 E3 72 99 54 49 D3 95 34 E6 C3 C5 12 B7 6F`

RO/SR Latch Hybrid PUF with separated RO/SR Latch units:

Board #1: `32 2B 39 5C A1 8F AB 5E A2 43 17 5A D2 E3 7A 54`

Board #2: `74 38 A6 95 1B 8A 81 B4 C2 95 56 8A A2 AD 81 A2`

Board #3: `46 63 8F AD 5A AA 55 BE A6 93 1E DB 82 95 C8 DB`

Board #4: `37 92 4D 63 22 86 C5 72 AA 4C A6 DF 23 22 D6 BF`

Appendix B

Hybrid PUF VHDL Code

In this appendix, we provide the VHDL codes for the RO/Anderson Hybrid PUF method 1 and RO/Anderson Hybrid PUF method 2. Note that, only the PUF units source codes are provided here. Other codes related to the control unit, challenge-response system, and the measurement system are not provided. Also, note that, part of the code which is related to the Anderson PUF implementation is obtained (and modified for our platform) from Dr. Anderson's personal web page.

RO/Anderson Hybrid PUF method 1:

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

entity Hybrid1 is
    Port (enable : in  STD_LOGIC;
          output : out  STD_LOGIC
 );
end Hybrid1;

architecture Behavioral of Hybrid1 is

signal OUT_INT  : STD_LOGIC;
signal OUT_INT2  : STD_LOGIC;
signal OUT1 : STD_LOGIC;
signal OUT2 : STD_LOGIC;
signal OUT3 : STD_LOGIC;
signal Int_O1  : STD_LOGIC;
```

```
signal CARRY_BW  : STD_LOGIC;
signal CARRY_BW2 : STD_LOGIC;
signal CARRY_BW3 : STD_LOGIC;
signal CARRY_BW4 : STD_LOGIC;
signal C  : STD_LOGIC_vector (4 downto 0);

attribute keep  : string;
attribute keep of C : signal is "TRUE";
attribute S : string;
attribute S of C : signal is "TRUE";

--controlling the placement of the components
attribute rloc: string;
attribute rloc of FDCPE_inst : label is "X-2Y4";
attribute rloc of FDCPE_inst2  : label is "X-2Y-1";
attribute rloc of LUT1_inst_Buf  : label is "X0Y4";
attribute rloc of LUT1_inst_Inv  : label is "X0Y0";
attribute rloc of MUXCY_inst  : label is "X0Y4";
attribute rloc of Int_MUXCY_inst  : label is "X0Y3";
attribute rloc of Int_MUXCY_inst2  : label is "X0Y2";
attribute rloc of Int_MUXCY_inst3  : label is "X0Y1";
attribute rloc of MUXCY_inst2  : label is "X0Y0";
attribute rloc of inst_inv2  : label is "X-2Y2";
attribute rloc of inst_Inv  : label is "X-1Y1";
attribute rloc of inst_nand  : label is "X-1Y2";
attribute rloc of inst_Inv3  : label is "X-2Y1";
attribute rloc of inst_and  : label is "X-2Y3";

begin

inst_nand : LUT2
generic map (
INIT => X"7") -- initialized as a NAND
port map (
O => C(0), -- LUT general output
I0 => enable, -- LUT input
I1 => C(4) -- LUT input
);

inst_Inv : LUT1
generic map (
INIT => "01") -- initialized as a NOT
port map (
O => C(1), -- LUT general output
I0 => C(0) -- LUT input
```

```
);

inst_Inv2 : LUT1
generic map (
INIT => "01")
port map (
O => C(2), -- LUT general output
I0 => C(1) -- LUT input
);

LUT1_inst_Buf : LUT1
generic map (
INIT => "10") -- initialized as a Buffer
port map (
O => out1, -- LUT general output
I0 => C(2) -- LUT input
);

LUT1_inst_Inv : LUT1
generic map (
INIT => "01")
port map (
O => out2, -- LUT general output
I0 => C(2) -- LUT input
);

inst_Inv3 : LUT1
generic map (
INIT => "01")
port map (
O => C(4), -- LUT general output
I0 => out2 -- LUT input
);

inst_Inv4 : LUT1
generic map (
INIT => "01")
port map (
O => out3, -- LUT general output
I0 => C(4) -- LUT input
);

MUXCY_inst : MUXCY
port map (
O => OUT_INT, -- Carry output signal
```

```
CI => CARRY_BW4, -- Carry input signal
DI => '0', -- Data input signal
S => out1 -- MUX select
);

Int_MUXCY_inst : MUXCY --intermediate multiplexers
port map (
O => CARRY_BW4, -- Carry output signal
CI => CARRY_BW3, -- Carry input signal
DI => '0', -- Data input signal
S => '1' -- MUX select
);

Int_MUXCY_inst2 : MUXCY
port map (
O => CARRY_BW3, -- Carry output signal
CI => CARRY_BW2, -- Carry input signal
DI => '0', -- Data input signal
S => '1' -- MUX select
);

Int_MUXCY_inst3 : MUXCY
port map (
O => CARRY_BW2, -- Carry output signal
CI => CARRY_BW, -- Carry input signal
DI => '0', -- Data input signal
S => '1' -- MUX select
);

MUXCY_inst2 : MUXCY
port map (
O => CARRY_BW, -- Carry output signal
CI => '1', -- Carry input signal
DI => '0', -- Data input signal
S => out2 -- MUX select
);

-- this FF captures the glitch
FDCPE_inst : FDCPE
generic map (
INIT => '0') -- Initial value of register (0 or 1)
port map (
Q => OUT_INT2, -- Data output
C => out3, -- Clock input
CE => '0', -- Clock enable input
```

```
CLR => '0', -- Asynchronous clear input
D => OUT_INT2, -- Data input
PRE => OUT_INT -- Asynchronous set input
);

FDCPE_inst2 : FDCPE
generic map (
INIT => '0') -- Initial value of register (?0? or ?1?)
port map (
Q => Int_O1, -- Data output
C => out3, -- Clock input
CE => enable, -- Clock enable input
CLR => '0', -- Asynchronous clear input
D => OUT_INT2, -- Data input
PRE => '0' -- Asynchronous set input
);

inst_and : LUT2
generic map (
INIT => X"8") -- initialized as an AND
port map (
O => OUTPUT, -- LUT general output
I0 => Int_O1, -- LUT input
I1 => out3 -- LUT input
);

end Behavioral;
```

RO/Anderson Hybrid PUF method 2:

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

Library UNISIM;
use UNISIM.vcomponents.all;

entity Hybrid2 is
    Port (clk : in std_logic;
  enable : in  STD_LOGIC;
          output : out  STD_LOGIC
);
end Hybrid2;

architecture Behavioral of Hybrid2 is

signal sel : STD_LOGIC;
```

```
signal C: STD_LOGIC_vector (5 downto 0);
signal OUT_INT : STD_LOGIC;
signal OUT_INT2 : STD_LOGIC;
signal O1, O2 : STD_LOGIC;
signal Int_O1, Int_O2, Int_O3, Int_O4 : STD_LOGIC;
signal CARRY_BW : STD_LOGIC;
signal CARRY_BW1 : STD_LOGIC;
signal CARRY_BW2 : STD_LOGIC;
signal CARRY_BW3 : STD_LOGIC;
signal CARRY_BW4 : STD_LOGIC;

attribute keep : string;
attribute keep of C : signal is "TRUE";
attribute S: string;
attribute S of C: signal is "TRUE";

attribute rloc: string;

attribute rloc of inst_nand: label is "X-2Y0";
attribute rloc of inst_Inv: label is "X-2Y1";
attribute rloc of inst_Inv2: label is "X-1Y0";
attribute rloc of inst_Inv3: label is "X-1Y1";
attribute rloc of Multiplexer0: label is "X-2Y2";
attribute rloc of inst_Inv4: label is "X-2Y3";
attribute rloc of inst_Inv5: label is "X-1Y2";
attribute rloc of SRL16E_inst: label is "X0Y5";
attribute rloc of Int_SRL16E_inst: label is "X0Y4";
attribute rloc of Int_SRL16E_inst2: label is "X0Y3";
attribute rloc of Int_SRL16E_inst3: label is "X0Y3";
attribute rloc of SRL16E_inst2: label is "X0Y2";
attribute rloc of MUXCY_inst: label is "X0Y5";
attribute rloc of Int_MUXCY_inst: label is "X0Y4";
attribute rloc of Int_MUXCY_inst2: label is "X0Y3";
attribute rloc of Int_MUXCY_inst3: label is "X0Y3";
attribute rloc of MUXCY_inst2: label is "X0Y2";
attribute rloc of FDCPE_inst: label is "X-2Y4";
attribute rloc of FDCPE_inst2: label is "X-2Y1";

begin

inst_nand : LUT2
generic map (
INIT => X"7") -- initialized as a NAND
port map (
O => C(0), -- LUT general output
```

```
I0 => enable, -- LUT input
I1 => C(5) -- LUT input
);

inst_Inv : LUT1
generic map (
INIT => "01") -- initialized as a NOT
port map (
O => C(1), -- LUT general output
I0 => C(0) -- LUT input
);

inst_Inv2 : LUT1
generic map (
INIT => "01")
port map (
O => C(2), -- LUT general output
I0 => C(1) -- LUT input
);

inst_Inv3 : LUT1
generic map (
INIT => "01")
port map (
O => C(3), -- LUT general output
I0 => C(2) -- LUT input
);

Multiplexer0 : MUXCY
port map (
O => C(4), -- Carry output signal
CI => C(1), -- Carry input signal
DI => C(3), -- Data input signal
S => sel -- MUX select
);

inst_Inv4 : LUT1
generic map (
INIT => "01")
port map (
O => C(5), -- LUT general output
I0 => C(4) -- LUT input
);

inst_Inv5 : LUT1
```

```
generic map (
INIT => "01")
port map (
O => output, -- LUT general output
I0 => C(5) -- LUT input
);

SRL16E_inst : SRL16E -- the "top" shift register instance
generic map (
INIT => X"5555")
port map (
Q => O1, -- SRL data output
A0 => '1', -- Select[0] input
A1 => '1', -- Select[1] input
A2 => '1', -- Select[2] input
A3 => '1', -- Select[3] input
CE => enable, -- Clock enable input
CLK => CLK, -- Clock input
D => O1 -- SRL data input
);

Int_SRL16E_inst : SRL16E -- Intermediate blocks
generic map (
INIT => X"FFFF")
port map (
Q => Int_O1, -- SRL data output
A0 => '1', -- Select[0] input
A1 => '1', -- Select[1] input
A2 => '1', -- Select[2] input
A3 => '1', -- Select[3] input
CE => enable, -- Clock enable input
CLK => CLK, -- Clock input
D => '1' -- SRL data input
);

Int_SRL16E_inst2 : SRL16E
generic map (
INIT => X"FFFF")
port map (
Q => Int_O2, -- SRL data output
A0 => '1', -- Select[0] input
A1 => '1', -- Select[1] input
A2 => '1', -- Select[2] input
A3 => '1', -- Select[3] input
CE => enable, -- Clock enable input
```

```
CLK => CLK, -- Clock input
D => '1' -- SRL data input
);

Int_SRL16E_inst3 : SRL16E
generic map (
INIT => X"FFFF")
port map (
Q => Int_O3, -- SRL data output
A0 => '1', -- Select[0] input
A1 => '1', -- Select[1] input
A2 => '1', -- Select[2] input
A3 => '1', -- Select[3] input
CE => enable, -- Clock enable input
CLK => CLK, -- Clock input
D => '1' -- SRL data input
);

SRL16E_inst2 : SRL16E -- the "bottom" shift register instance
generic map (
INIT => X"AAAA")
port map (
Q => O2, -- SRL data output
A0 => '1', -- Select[0] input
A1 => '1', -- Select[1] input
A2 => '1', -- Select[2] input
A3 => '1', -- Select[3] input
CE => enable, -- Clock enable input
CLK => CLK, -- Clock input
D => O2 -- SRL data input
);

MUXCY_inst : MUXCY
port map (
O => OUT_INT, -- Carry output signal
CI => CARRY_BW4, -- Carry input signal
DI => '0', -- Data input signal
S => O1 -- MUX select
);

Int_MUXCY_inst : MUXCY
port map (
O => CARRY_BW4, -- Carry output signal
CI => CARRY_BW3, -- Carry input signal
DI => '0', -- Data input signal
```

```
S => Int_O1 -- MUX select
);

Int_MUXCY_inst2 : MUXCY
port map (
O => CARRY_BW3, -- Carry output signal
CI => CARRY_BW2, -- Carry input signal
DI => '0', -- Data input signal
S => Int_O2 -- MUX select
);

Int_MUXCY_inst3 : MUXCY
port map (
O => CARRY_BW2, -- Carry output signal
CI => CARRY_BW, -- Carry input signal
DI => '0', -- Data input signal
S => Int_O3 -- MUX select
);

MUXCY_inst2 : MUXCY
port map (
O => CARRY_BW, -- Carry output signal
CI => '1', -- Carry input signal
DI => '0', -- Data input signal
S => O2 -- MUX select
);

-- This FF captures the glitch
FDCPE_inst : FDCPE
generic map (
INIT => '0') -- Initial value of register
port map (
Q => OUT_INT2, -- Data output
C => CLK, -- Clock input
CE => '0', -- Clock enable input
CLR => '0', -- Asynchronous clear input
D => OUT_INT2, -- Data input
PRE => OUT_INT -- Asynchronous set input
);

FDCPE_inst2 : FDCPE
generic map (
INIT => '0') -- Initial value of register
port map (
Q => sel, -- Data output
```

```
C => CLK, -- Clock input
CE => enable, -- Clock enable input
CLR => '0', -- Asynchronous clear input
D => OUT_INT2, -- Data input
PRE => '0' -- Asynchronous set input
);

end Behavioral;
```

Curriculum Vitae

Name:	Sasan Khoshroo
Education and Degrees:	- K.N. Toosi University of Technology Tehran, Iran 2005 - 2010 B.S. - University of Western Ontario London, ON 2011 - 2013 M.E.Sc
Honours and Awards:	WGRS-ECE 2011-2013
Related Work Experience:	- Research Assistant The University of Western Ontario 2011 - 2013 - Teaching Assistant The University of Western Ontario 2011 - 2013

Publications:

- S. Khoshroo and A. Reyhani-Masoleh, *"An Efficient Secret Sharing Scheme With Cheater Detection Capability Based On Physically Unclonable Functions"*, Submitted to Selected Areas in Cryptography 2013, under review.

- S. Khoshroo, H. Wang, and Yalin Wang, *"Industrial Wireless Sensor Networks: Applications, Protocols, and Standards"*, chapter Reliable and Robust Communications in Industrial Wireless Sensor Networks, Published by CRC Press, Apr. 2013.

- S. Khoshroo, H. Wang, L. Xing, and D. Kasilingam, *"A Joint Resource Allocation-Channel Coding Design Based on Distributed Source Coding"*, Published in Jour-

nal of Wireless Communications and Mobile Computing, DOI: 10.1002/wcm.2349, Jan. 2013.

- N. Tadayon, S. Khoshroo, E. Askari, and H. Wang, *"Power Management in Energy Harvesting Wireless Sensor Networks"*, Published in Elsevier, Journal of Network and Computer Applications, Feb. 2012.