COMPACT FIELD PROGRAMMABLE GATE ARRAY BASED PHYSICAL UNCLONABLE FUNCTIONS CIRCUITS

by

Yangpingqing Hu

A Dissertation Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

in Engineering

at

The University of Wisconsin Milwaukee

May 2022

ABSTRACT

COMPACT FIELD PROGRAMMABLE GATE ARRAY BASED RING OSCILLATOR PHYSICAL UNCLONABLE FUNCTIONS CIRCUITS

by

Yangpingqing Hu

The University of Wisconsin-Milwaukee, 2022
Under the Supervision of Professor Weizhong Wang

The Physical Unclonable Functions (PUFs) is a candidate to provide a secure solid root source for identification and authentication applications. It is precious for FPGA-based systems, as FPGA designs are vulnerable to IP thefts and cloning. Ideally, the PUFs should have strong random variations from one chip to another, and thus each PUF is unique and hard to replicate. Also, the PUFs should be stable over time so that the same challenge bits always yield the same result. Correspondingly, one of the major challenges for FPGA-based PUFs is the difficulty of avoiding systematic bias in the integrated circuits but also pulling out consistent characteristics as the PUF at the same time. This thesis discusses several compact PUF structures relying on programmable delay lines (PDLs) and our novel intertwined programmable delays (IPD). We explore the strategy to extract the genuinely random PUF from these structures by minimizing the systematic biases. Yet, our methods still maintain very high reliability. Furthermore, our proposed designs, especially the TERO-based PUFs, show promising resilience to machine learning (ML) attacks. We also suggest the bit-bias metric to estimate PUF's complexity quickly.

Index Terms—Field Programmable Gate Array, Physical Unclonable Features, Identification and Authentications, Secure root sources, Hardware security.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

FPGA	Field Programmable Gate Arrays
RO	Ro Oscillator
TERO	Transient Effect Ring Oscillator
PUF	Physical Unclonable Function
SCC	Single-Chip Crypto
HT.	Hardware Trojan
IDF	Isolation Design Flow
HT	Hardware Trojan
PS	Processing System
PL	Programmable Logic
PDL	Programmamble Delay Lines
IPD	Intertwined Programmable Delay
LSB	Least Significant Bit
MSB	Most Significant Bit
LUT	Look up Table
IoT	Internet of Things
IP	Intellectual Property
SRAM	Static Random-Access Memories
ASIC	Application-Specific Integrated Circuit

ACKNOWLEDGEMENTS

First, I would like to express my gratitude to my advisor professor, Dr. Weizhong Wang, who has guided and supported my work and myself during the long journey. He is always responsive and forthcoming, and I can get answered and helped instantaneously.

Also, I would like to thank the other members of the committees: Dr. Yi Hu, Dr. Jun Zhang, Dr. Zeyun Yu, for their help in the academy, and Dr. Robert Turney, who immensely helped me in the industry.

Furthermore, I want to thank Yuqiu Jiang for the discussion and collaboration in our group. I also want to thank many other professors, staff, and graduate students in the electrical engineering department and the College of Engineering & Applied Science for their kindness and help.

Last but not least, I would like to thank my parents for their love and support and my girlfriend for her encouragement.

CHAPTER 1. INTRODUCTION

A. Cybersecurity challenges FPGAs

A field-programmable gate array (FPGA) is a semiconductor device that can be programmed after manufacture to perform a specific application design [31]. NASDAQ OMX Corporate Solutions anticipated that the global FPGA market size would reach USD 18.8 billion by 2027, registering a compound annual growth rate of 9.7% over the forecast period. The increased adoption of FPGA across multi industries, such as networking, data center, electrical vehicles (EV), and the Internet of Things (IoT), is projected to drive the industry growth over the forecast period [33].

As one important application of FPGA, IoT gets involved in many critical infrastructures, like the intelligent power grid and transportation systems, which connect many IoT devices [32]. Therefore, there is an increasing demand for identifying these FPGA devices and blocking malicious identities. FPGA hardware security is the root of trust. Like the malware that corrupts the boot-up sequence, some attacks cannot be detected by software-level countermeasures [24].

Hardware Trojan (HT) is an instance of a hardware-level threat. HT is defined as a malicious, intentional modification of a circuit design that results in undesired behavior when the circuit is deployed [24]. An attacker could inject HT into the device, making affected devices vulnerable and sensitive information leakage.

In the highly globalized semiconductor industry, many profits attract attackers. FPGAs have grown more complex, and the intellectual property (IP) 's value has grown commensurately. Along with this trend, IP piracy has become a threat to IP vendors. SRAM FGPA is the most popular FPGA due to its advantages, such as its higher performance, greater logic density, and improved

power efficiency. Examples for SRAM FGPA include Xilinx 7-Series and Intel Stratix-5. SRAM FPGA requires external nonvolatile memory for the FPGA application program, i.e., bitstreams. The transmission of the program from the nonvolatile external memory to the SRAM FPGA may expose the programming to a potential adversary [31]. Besides, unauthorized copy, theft, and reverse-engineering are other threats to SRAM FPGAs when the bitstream is transmitted from non-volatile storage into the FPGA [34].

B. Countermeasure to the threat.

In earlier days, several methods could be used for protecting bitstreams. One was loading the bitstreams at a secure location. Another solution was to use external memory with a unique identifier. In modern times, FPGA vendors have implemented software, IP cores, etc., to deal with the security threat.

Physical unclonable function (PUF) is among these methods. PUF acts as a device-specific identifier and can be used for FPGA security [31]. PUFs are innovative circuit primitives that extract secrets from the physical characteristics of integrated circuits (ICs) [10]. The secrets come from the random variation during the manufacturing process. Millions of electronic components are fabricated on the die inside an IC, and modern technology has very high precision when fabricating those components. When an IC is being manufactured, these random variations will present unique characteristics. Because of the randomness and uniqueness, it is practically impossible to replicate the same feature on another die. This feature has become a fingerprint for IC. PUF is considered a hardware attestation approach.

One approach of using PUF is single-chip crypto (SCC). With the help of isolation design flow, the whole FPGA design is isolated into several regions. They can be programmed at different times. Some encryption step is done for programming the boot loader as the 1st region into FPGA. Then PUF could be used as the identifier for the secondary encryption key for programming the

rest regions [31]. In [25], an SCC processor architecture called Aegis, equipped with PUF, was introduced.

PUFs have been demonstrated in FPGA fabric (‘‘soft PUF’’) as well as in dedicated logic (‘‘hard PUF’’). Microsemi’s SmartFusion2 includes a hard PUF [31]. This PUF resides in a dedicated SRAM. The random start-up behavior of a 16 Kbit 2 KB SRAM block is used to determine a static secret unique to each device [37].

Xilinx’s high-end MPSoC (Multiprocessor System on Chip) Zynq UltraScale+ has a built-in PUF (Figure 1-1). It can generate a cryptographically strong, device-unique encryption key that can be used in combination with the built-in advanced encryption standard (AES) cryptographic core. This key cannot be read by a user, allowing for a heightened level of key security. When a Zynq UltraScale+ device is provisioned, the PUF’s device-unique encryption key encrypts and decrypts user data [21]. Xilinx has not revealed the detailed structure of the PUF, and it is unknown whether a soft PUF or a hard PUF is used.

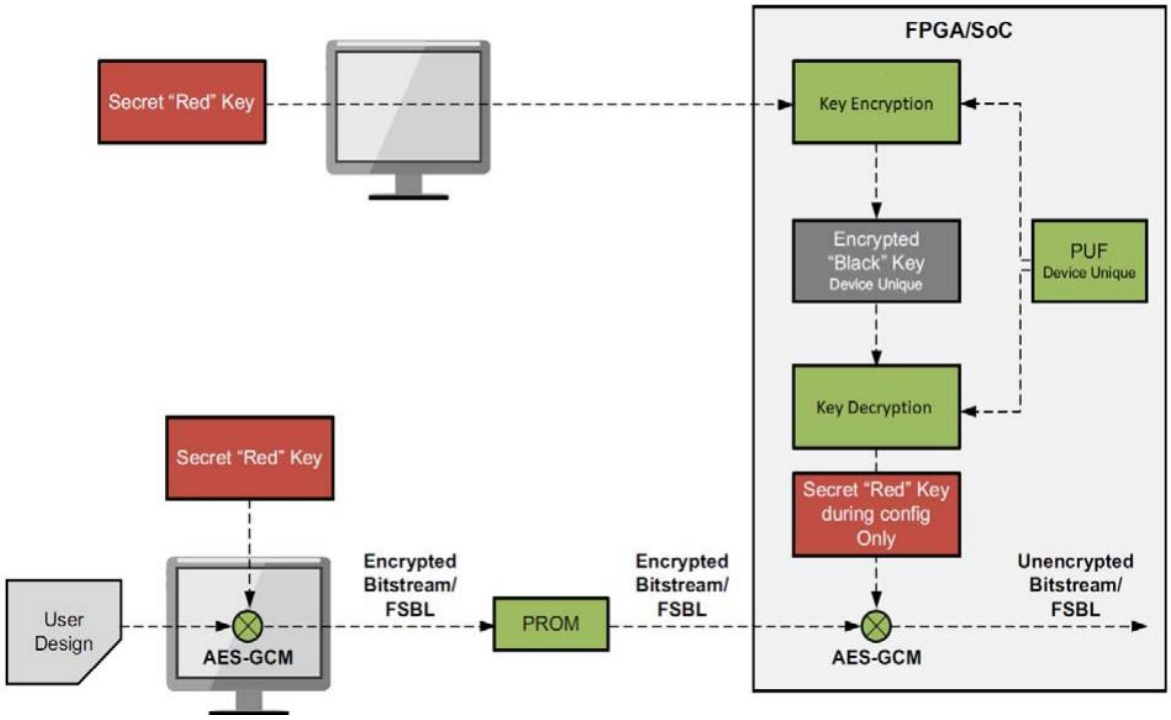


Figure 1-1 Xilinx Zynq UltraScale+ encrypting and decrypting the device key using PUF [21].

Academic research has been more focused on the design of soft PUF, and hardware resources like SRAM, LUT, and flip-flops have been investigated for use by PUFs. The most fundamental challenge for all PUFs is that they must exhibit extreme sensitivity to manufacturing variations, yet they must be deterministic to provide a consistent query response. Therefore, the ideal PUF structures should be free of systematic bias to maximize the entropy due to manufacturing variations among different chips. In the meantime, the desired consistency in the outcomes under various operation conditions mandates that the ideal PUF design should have a mechanism to extract the deterministic hardware variations and suppress temporal random noise in the measurement system.

C. Contributions of This Work and Thesis Outline

The novelty of this work and our main contributions are as follows,

First, this work investigates systematic biases among PDLs inside a LUT structure (Xilinx Artix-7 LUT6). Our experimental results show a significant systematic bias between the two LUT5s in each LUT6. We also found systematic bias among the PDLs within each LUT5.

Second, a novel intertwined programmable delay (IPD) structure is proposed to mitigate the bias found in Xilinx LUTs. Our novel IPD consists of intertwined paths, whose delays are programmable to challenges. A modified 2-pass scheme is proposed to mitigate the biases in the structure of the Xilinx LUT6. The new 2-pass scheme consists of two phases in each pass. The IPD-RO was configured using a challenge that appeared in two phases. Our modified 2-pass scheme significantly reduced the deterministic bias in the LUT5 layout.

Third, PDLs and IPDs are considered as the elements of TERO-PUF for the first time. PUFs are successfully extracted from the subtle differences in PDLs and IPDs, making our TERO-PUFs even more compact. Furthermore, they are proven strong candidates for PUFs resilient to ML

attacks.

Lastly, we propose a novel metric, bit-bias, to quickly evaluate PUF's complexity to ML attacks. Instead of spending a long time training the model and running tests, one can develop a fast and straightforward evaluation to estimate PUF's resilience with the bit-bias metric. Our results show the correlation between bit-bias and NN successful attack rate.

The outline of this thesis is as follows.

Chapter 2 preliminarily introduces the technical background of this thesis. It firstly reviews the development of PUFs and the characteristics of a few PUF designs. Then, this thesis explains canonical characterization metrics frequently used for PUFs for ease of discussion. This chapter presents the PUF implementation and test suite, a Xilinx-based hardware and software development environment.

Chapter 3 investigates the biases in LUTs and PDLs. First, the layouts of LUTs in Intel and Xilinx FPGAs are reviewed. Then, I experimentally investigate and present the biases in Xilinx Artix-7 FPGA LUT6s. This experimental investigation is one of the major contributions. At last, chapter 3 proposes a delay model for the investigated LUT6.

Chapter 4, based on the investigation results and the derived delay model, proposes the novel IPD-RO-PUF. The mitigation of biases is demonstrated, and the PUF is characterized in several ways, including canonical metrics, NIST randomness test suite, entropy, etc.

Chapter 5 discusses the potential of achieving higher reliability in the IPD-RO-PUF with the analysis of the analog measurements of PUFs. Two approaches are exhibited, including using a reference RO and filtering CRPs with margins.

Chapter 6 presents novel TERO-PUFs based on PDLs and IPDs. The potential of PDLs is explored in TEROs, and the biases impacts on TEROs are revealed and analyzed.

Chapter 7 applies the machine learning attacks on the discussed PUFs. The impact of our found systematic biases on NN attacks predictions mainly attracts me. To study this impact, we will consider entropy corresponding to input probabilities and propose a novel metric, bit-bias, to link the biases in PUF to the NN attack prediction.

Chapter 8 presents an application for the compact PUF. In this application, an IPD-RO-PUF replaces the ICAP in Xilinx Artix-7 FPGA and successfully authenticates the reconfiguration of the FPGA.

In the end, Chapter 9 concludes this thesis and discusses the future of our work.

CHAPTER 2. PRELIMINARIES

A. The development of delay-based PUFs

Gassend introduced PUF in 2002 for the first time [40]. For a clear understanding of what PUF is, Gassend made several definitions for PUF.

Definition 1: A PUF is a function that maps challenges to responses that is embodied by a physical device.

Definition 2: A PUF is said to be controlled if it can only be accessed via an algorithm that is physically linked to the PUF in an inseparable way (i.e., any attempt to circumvent the algorithm will lead to the destruction of the PUF). In particular, this algorithm can restrict the challenges presented to the PUF and limit the information about responses given to the outside world.

Definition 3: A type of PUF is said to be Manufacturer Resistant if it is technically impossible to produce two identical PUFs of this type given only a polynomial amount of resources.

Many later works followed these definitions. Almost all PUFs try to discover an excellent way to exploit the silicon variation, and this variation would give a random mapping from challenges to responses. Many sources, roughly classified into memory-based and delay-based, have been exploited to provide such variation [49].

As an instance of the memory-based PUFs, SRAM PUFs, proposed by Guajardo in 2007, extract the initial voltage level in SRAMs [44]. Another example is DRAM-based PUF by Tehranipoor in 2017 [58].

The other type, delay-based PUFs, usually extract the variations in the propagation delays of transistors. Elaborated circuits could exploit delay variation in silicon and provide randomness.

These variations are different from circuit to circuit, and factors like process temperature and pressure during manufacturing are instances causing these variations. This thesis will focus on the study of propagation delay-based PUFs.

1) Arbiter PUF (APUF)

[10] illustrated the concept of arbiter PUF. An arbiter PUF essentially includes a group of delay circuits and an arbiter. There are two competing propagation delay circuits. Delay elements connected to inputs would decide how the propagation delay circuits are. At a time, challenges would set for the propagation delay circuits. The two delay circuits are excited simultaneously. And the signals would propagate to the arbiter. The arbiter determines which delay has faster propagation, thus giving the 1-bit outcome based on the result of the competition. Each of these groups of delay circuits should be implemented, ensuring that the nominal delay is the same. The randomness in the nominal delays gives the randomness of the competition. The number of delay elements determines the size of CRPs.

[42] experimented arbiter PUF with 64 stages. The delay element is based on MUX and arbiter-based latch in this work. Several works proposed some improvement over the traditional arbiter PUF. In [42], a feed-forward arbiter PUF was presented, increasing the inter-chip variation.

2) RO PUF

a) Original RO PUF

Suh discussed RO PUF for the first time in 2007 [10], also called RO bank PUF. Figure 2-1 shows the structure of the traditional RO PUF. There are plenty of identically laid out physical ROs implemented on the hardware. A pair of competing ROs are selected at a time by the challenge. The two competing ROs are triggered at the same time. The outcome of the competition determines the 1-bit outcome as the outcome of the PUF.

In [7], Maiti suggested that his tested RO PUF has better performance in uniqueness and

uniformity than APUF. The uniqueness for APUF was reported only 7.20%, while the tested RO PUF is reported 47.24%. A possible reason for this vast deviation is the difficulty in ensuring routing symmetry in an APUF [45].

One big drawback of this design is that there should be a large number of physical ROs implemented on hardware. The entropy of the RO bank PUF is $n*(n-1)/2$, where n is the number of ROs. To achieve high entropy, the traditional RO PUF needs to consume many hardware resources.

A good example of an implemented traditional RO-PUF was proposed by Maiti in 2010 [12]. The authors implement LUTs as inverters, and each physical RO includes five inverters.

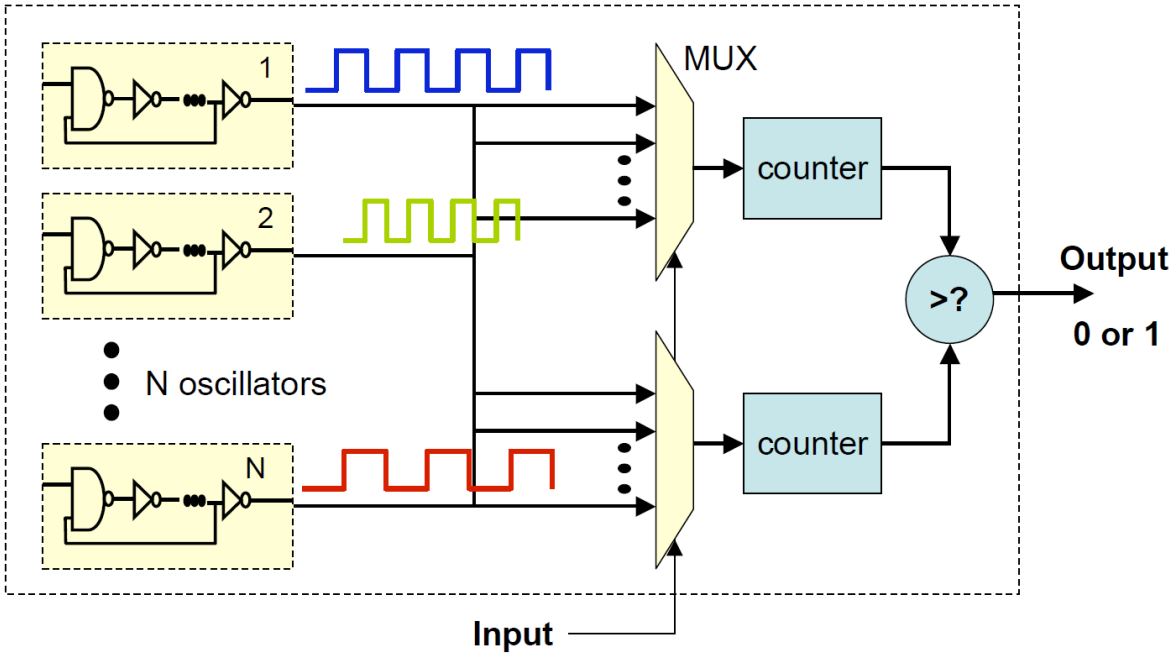


Figure 2-1 Structure of RO bank PUF [10]

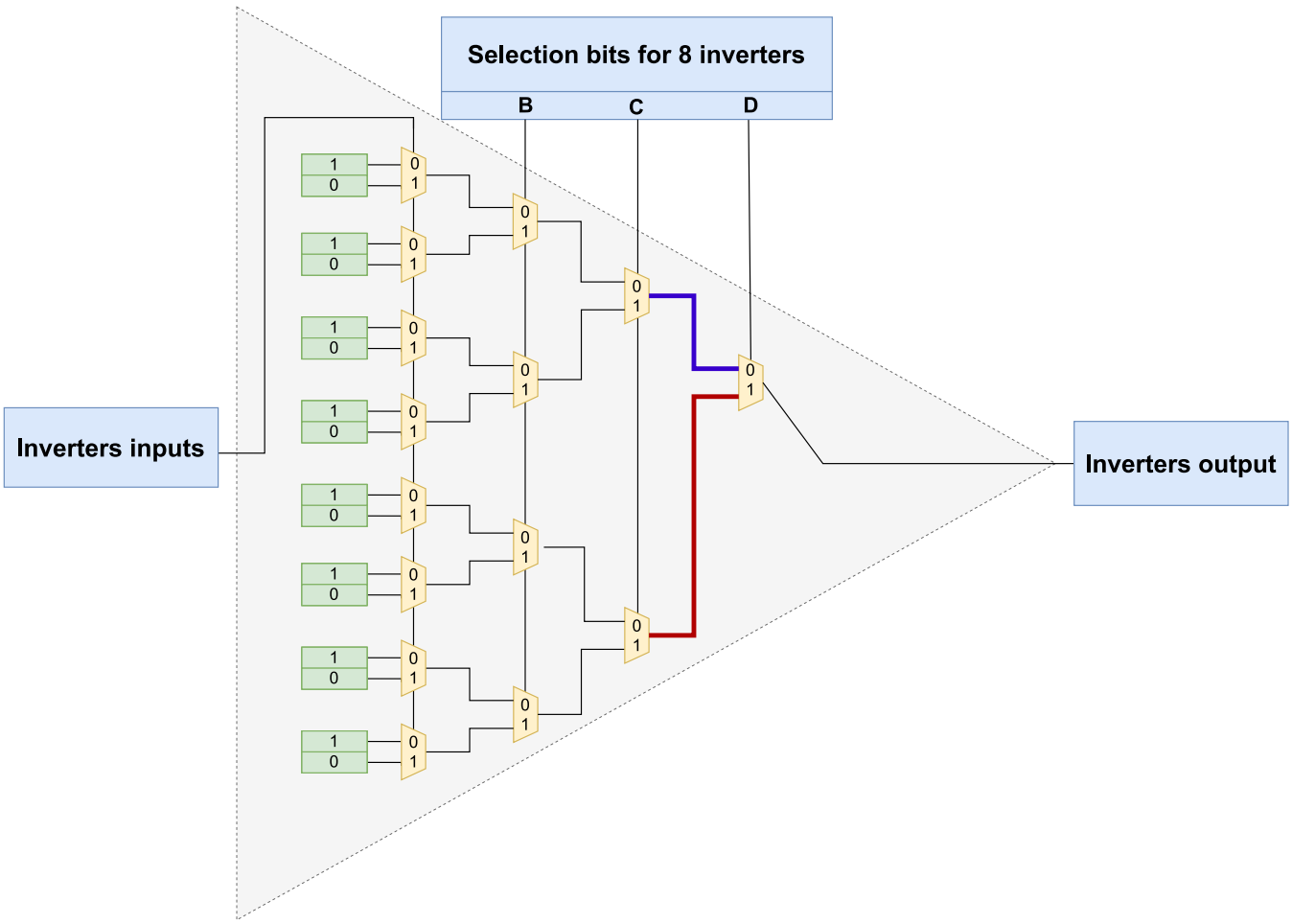


Figure 2-2 LUT4 used as an inverter.

b) Programmable Delay Line (PDL)

In 2010, Majzoobi proposed PDLs and implemented them in an APUF [57]. Figure 2-3 demonstrates the concept of PDL. First, one needs to initialize the content of the LUT logic values correctly. Usually, the LSB of LUT inputs is used as the inverter input. So, the LUT logic values should be initialized in a 2-bit pattern of "01". Thus the output of LUT is always the inverse of the input.

LUTs in FPGAs usually have more than one input. While one of the inputs acts as the inverter input, the other can be used as programmable bits. The use of programmable bits has been studied in many pieces of literature. Two examples will be reviewed next.

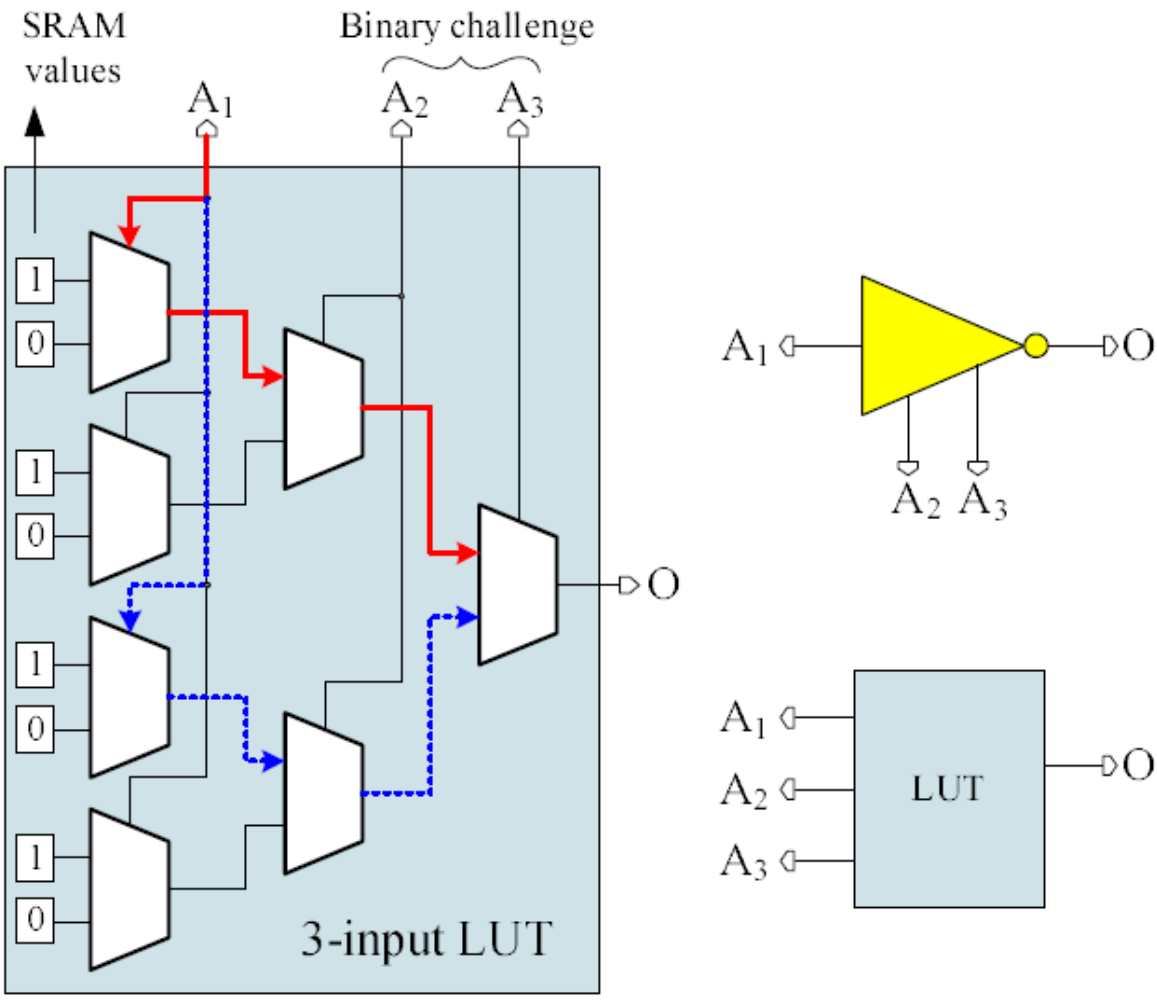


Figure 2-3 The proposed PDL structure in [57]

c) Habib's RO PUF using PDLs

Taking advantage of PDLs, Habib proposed a more compact RO PUF in 2013 [2]. This RO PUF applied PDLs into the traditional RO PUF, and its overall scheme and structure are still very similar to the traditional RO PUF. While one challenge still selects a pair of ROs at a time, multiple responses can be generated thanks to PDLs. One or multiple PUF responses come from comparing all the 8 PDLs configured by the 3-bit input of LUTs. Habib's RO PUF requires less area than traditional RO PUF because multiple PUF responses can be achieved from one pair of ROs.

However, there are a few drawbacks to this design. Firstly, this RO PUF cannot always give multiple responses to one challenge. Only a one-bit response is achieved when all the eight paths

in one RO are faster than the competing RO. Second, the size of configurable bits, 3-bit, is relatively small. In 2019, Zhou proposed a programmable delay RO PUF that increased the number of CRPs [15]. Lastly, Feiten reported the existence of biases of RO implemented on Intel FPGAs [4]. Although Habib's work was based on Xilinx Spartan FPGA, systematic biases are also very likely to present, which is an inevitable characteristic during designing and manufacturing the FPGA chip.

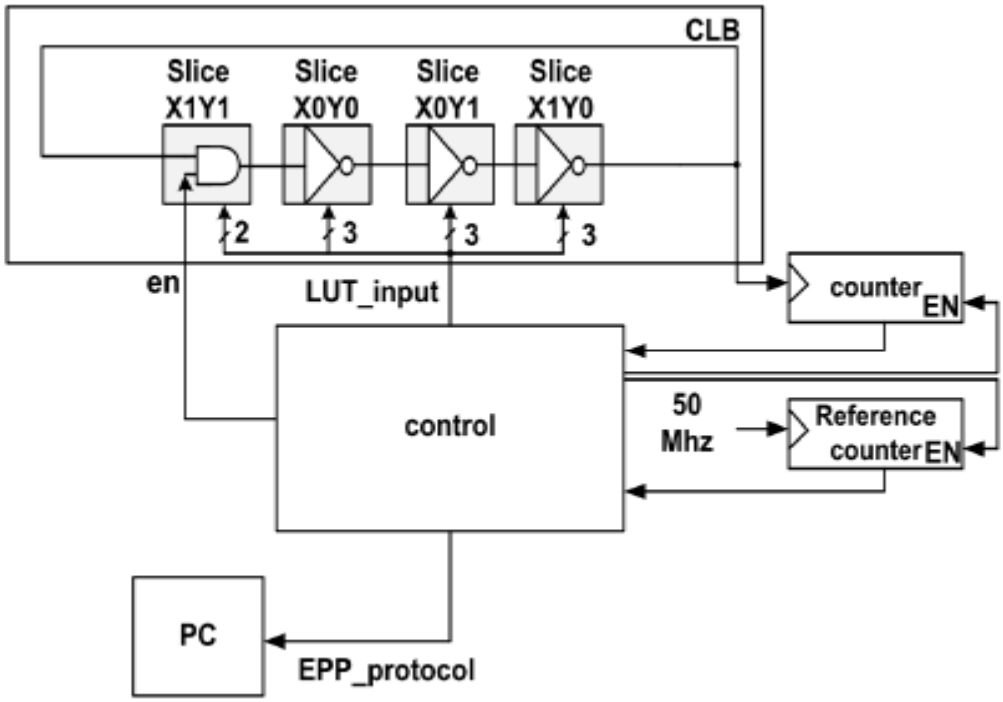


Figure 2-4 RO PUF using PDLs reported by Habib in [2].

d) Feiten's PDL-RO-PUF

Taking advantage of the abundant resources in LUTs, Feiten proposed the PDL-RO-PUF in 2018 [18]. In his work, the disparity in PDL delays is used as a source of entropy. When programmable bits are different, the LUT-based inverter uses different PDLs. Alternatively, one could view it as a LUT that allows selecting inverters. PDL-RO-PUF is solely based on the intrinsic disparity in PDLs in LUTs. When cascading many more LUTs in one RO, many more LUT inputs can be used as programmable bits. Feiten's PDL-RO-PUF is by far the most efficient

at hardware usage. However, real field experiments were lacking in their works. [18] designate the biases in PDLs due to the LUT internal structure. However, they did not continue with the real field experiments. This thesis will fill up this missing work.

The PUF response generation scheme in PDL-RO-PUF is a 2-pass scheme, in which one physical PDL-RO runs twice sequentially, and the comparison of the two passes gives a 1-bit PUF response. One problem with the 2-pass scheme is the variation coming from the sampling at two different times. ROs are very sensitive to the change of system clock, the fluctuation in supplied voltage, and even ambient temperature. The measurements at two different times inevitably suffer from those variations. The effect of those variations is in question.

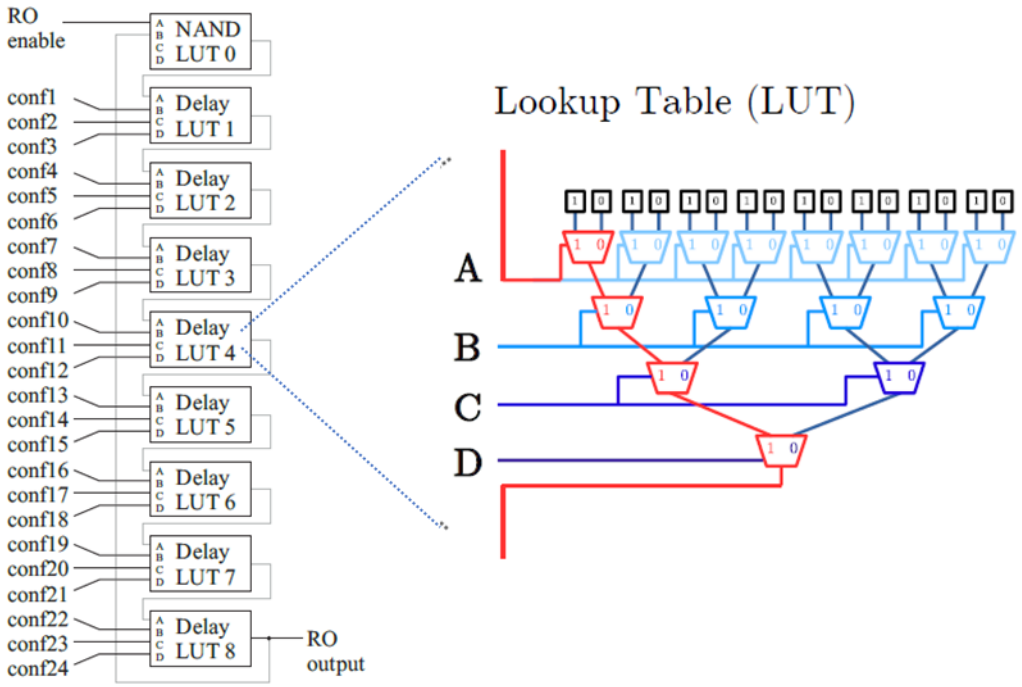


Figure 2-5 VRO reported by Feiten in [18].

3) TERO-PUF

A very different RO structure, TERO, introduced in 2014 by Bossuet [51], and comprehensively studied in 2018 by by Marchand [51], utilizes the transient effect of the two branches in a RO. In conventional ROs, oscillations persist and stay relatively stable. However, the oscillations in TERO die down in the end. Such a transient effect provides a new perspective to extract the

unpredictable physical variation in manufacturing. Furthermore, it has been concluded that TERO is lighter than RO-PUF in the area and power consumption and more robust to electromagnetic attacks [49]. TERO has been found to be a very strong candidate resistant to modeling attacks.

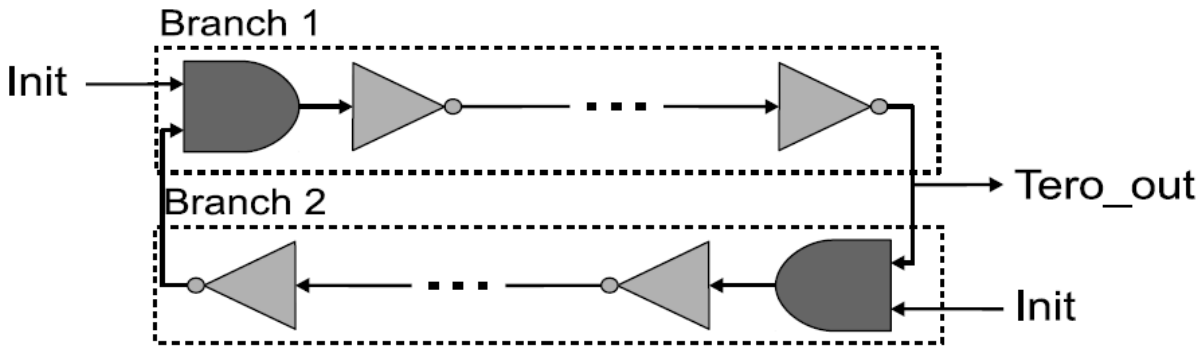


Figure 2-6 Marchand's TERO PUF [49]

B. PUF characterization metrics

Before discussing our proposed PUFs, it is necessary to define and describe the metrics used to characterize PUFs. Maiti has summarized canonical PUF characterization metrics in [7], which is the basis of our characterizations. [7] also defines a 3-dimensional space for PUF, which guides me to define our data dimension, as shown in Figure 2-7.

First, one can define the challenges-axis. Challenges are the inputs to PUFs, and a PUF should generate one or multiple responses to a single challenge. Second, the samples-axis is defined that PUFs are fed with the same challenge, and multiple samples of responses are captured. Last, numerous PUFs tested with the same challenge define the PUFs-axis. Compared to the original three-dimensional space [7], I replaced the original device-axis with the PUFs-axis. Many traditional works implement only one PUF in one device. However, we were able to implement multiple compact PUFs in one device. Furthermore, there has been more interest in the impacts of spatial factors on PUFs [56]. Therefore, our PUFs-axis combines two elements, device and locations.

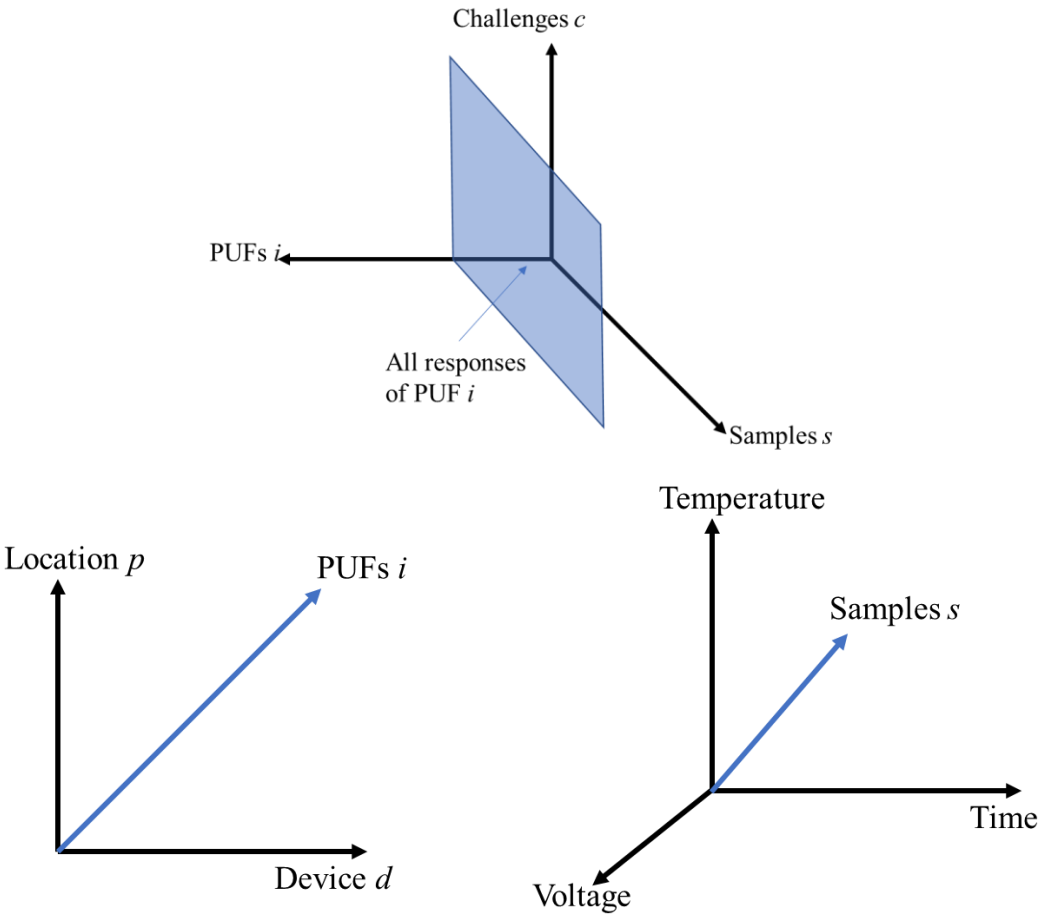


Figure 2-7 PUF dimension.

With the help of the three-dimensional space, we can define the capture of a PUF as,

$$r_{s,p,d,c} \text{ or } r_{s,i,c}$$

s = repeated measurements;

p = index of location in a device;

d = index of device;

i = index of PUFs;

c = challenge/challenge pair.

Frequently, we generally characterize C-bit responses of one PUF. We denote the C-bit response as,

$$R_{s,d,p} \text{ or } R_{s,i}$$

1) Uniformity

For any PUF, it should have the same probability of giving a PUF bit as 0 or 1. Otherwise, the attacker is more likely to be successful by predicting the response with a larger probability. In the PUF *i*, the percentage of 1's out of n PUF bits defines the uniformity, whose ideal value is 50%. Uniformity is determined along with the "Challenge" axis or the plane of "Challenge-Samples".

$$Uniformity_{i,p} = \frac{1}{C}\sum_{c=1}^{C} r_{i,c} * 100\% \quad (2\text{-}1)$$

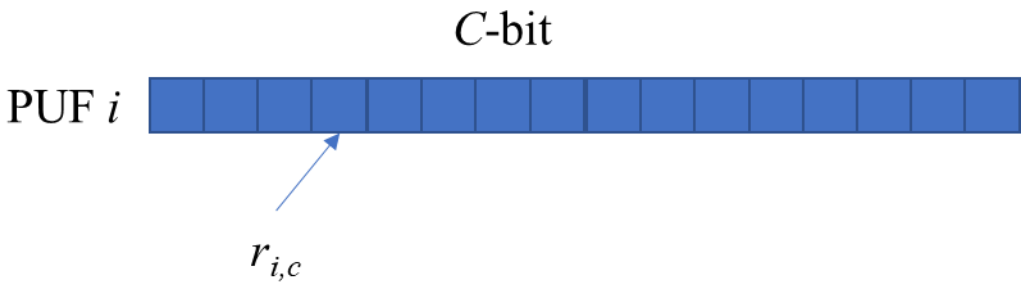


Figure 2-8 Uniformity

2) Bit-aliasing

Bit-aliasing is a straightforward way to characterize the randomness in PUFs, whose principle is demonstrated in Figure 2-9. Bit-aliasing evaluates how likely a group of PUFs generate the same responses when given the same challenges. Ideally, each PUF response is generated based on its unique structure or process. In this situation, the PUF responses are uncorrelated. However, due to biases, PUFs in the same chip or across chips may give the same response to the same challenge. Depending on the scope of the group of PUFs, we would like to discuss three kinds of bit-aliasing.

First, the following equation defines the overall bit-aliasing, where all the PUFs across *D* devices and *P* positions are all in scope.

$$Bit - aliasing = \sum_{i=1}^{I} R_i * 100\% \tag{2-2}$$

Second, inter-device bit-aliasing is calculated based on the following equation. Across D devices, the PUFs at the same position p is in the scope. Many previous works consider this as the only bit-aliasing because each of their device has only one PUF.

$$Bit - aliasing_{Intra}^{p} = \sum_{d=1}^{D} R_{p,d} * 100\% \tag{2-3}$$

Last, we calculate intra-device bit-aliasing in (2-4). In many recent works, there is a growing interest in the spatial correlation of PUFs [25]. Therefore, it is worthwhile to find the bit-aliasing within the same device.

$$Bit - aliasing_{Intra}^{d} = \sum_{p=1}^{P} R_{p,d} * 100\% \tag{2-4}$$

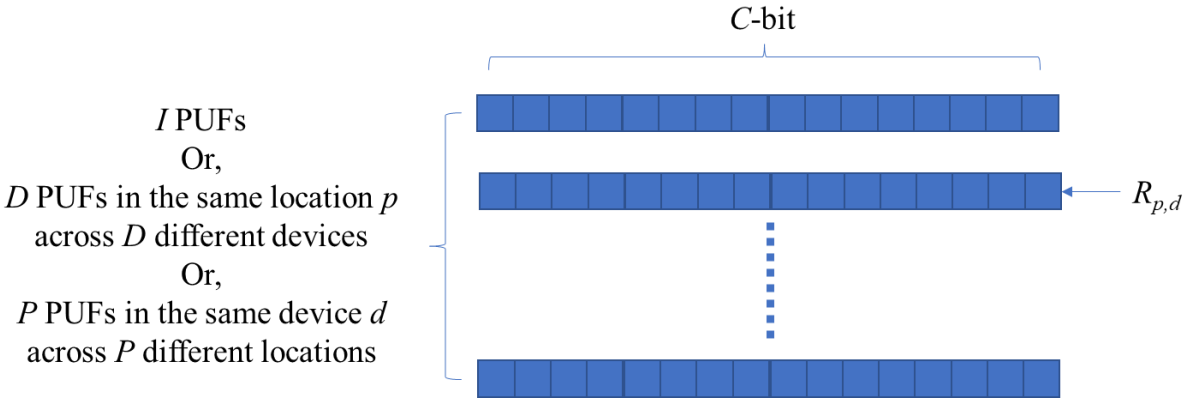


Figure 2-9 Bit-aliasing

3) Uniqueness

In the tests, uniqueness indicates how different a PUF is from another PUF. Hamming distance (HD) evaluates the uniqueness between the *C*-bit responses of PUF *i* and *j*. HD is calculated for all possible PUFs pairs selected from *I* PUFs. Ideally, uniqueness would be 50%.

$$Uniqueness_p = \frac{2}{I(I-1)} \sum_{i=1}^{I-1} \sum_{j=i+1}^{I} \frac{HD(R_i R_j)}{C} * 100\% \quad (2\text{-}5)$$

As mentioned before, the "PUFs *i*" axis combines the device and locations. To better examine the correlation in these two aspects, two variants of uniqueness are defined as follows.

a) *Inter-device uniqueness*: Uniqueness is measured on the PUFs placed on the same location *p* of two devices *d* and *d'*. The following equation measures inter-device uniqueness.

$$Uniqueness_p = \frac{2}{D(D-1)} \sum_{d=1}^{D-1} \sum_{d'=d+1}^{D} \frac{HD(R_{d,p} R_{d',p})}{n} * 100\% \quad (2\text{-}6)$$

b) *Intra-device uniqueness*: In the same device *d*, the PUF responses from RO at different locations *p* and *p'* are considered. The resulted uniqueness is intra-device uniqueness, and this metric could better determine the possible entropy given by a single device *d*.

$$Uniqueness_d^{intra} = \frac{2}{P(P-1)} \sum_{p=1}^{P-1} \sum_{p'=p+1}^{P} \frac{HD(R_{d,p} R_{d,p'})}{C} * 100\% \quad (2\text{-}7)$$

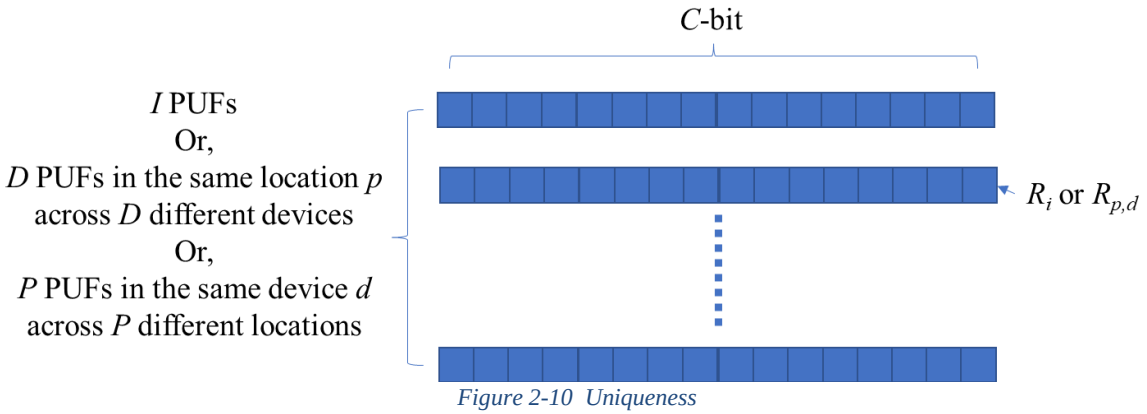


Figure 2-10 Uniqueness

4) Reliability

Reliability is important as PUF should produce consistent responses whenever the they are given the same challenge. It indicates how likely a PUF could reproduce the same PUF bit to the same challenge. Multiple samples are captured on PUF *i* with the same challenge *c*, and HD is calculated over all the samples. The ideal value for reliability is 100%.

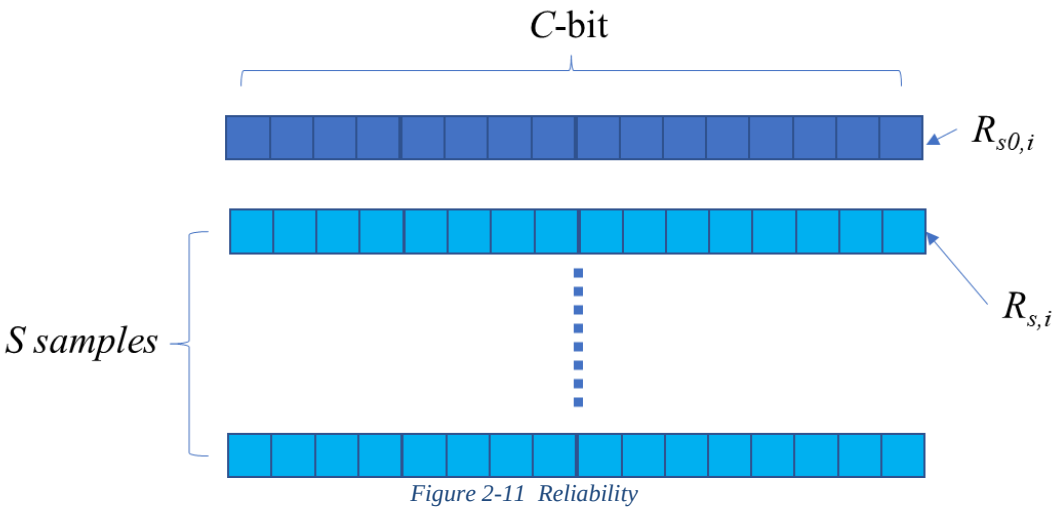


Figure 2-11 Reliability

$$Reliability = \left(1 - \frac{1}{S}\sum_{s=1}^{S}\frac{HD\left(R_{s0,i}, R_{s,i}\right)}{C}\right) * 100\% \quad (2\text{-}8)$$

5) Correlation

The canonical characterization metrics aforementioned are regarding only one CRP. As pointed out in [18][56], in addition to canonical metrics such as uniqueness, the correlation in CRPs must also be investigated to evaluate the guessing complexity of the PUF. The following example can illustrate the undesirable effect of that. Let *m*/2 devices have the signature 11110000, and the other *m*/2 devices have the signature 00001111. *HD* would produce optimal values of 50%, while the signatures are not unique. [47] An attacker can utilize this correlation.

Therefore, a metric called *Correlation Sensitive Metric* (CSM) was proposed in [47] and applied in [18] to evaluate their VRO PUF. CSM is described as follows:

$$cor_{j,k}^{i} = \begin{cases} 1, & if\ R_{i,j} = R_{i,k} \\ -1, & otherwise \end{cases} \tag{2-9}$$

$$cor_{j,k} = \frac{1}{m}\sum_{i=1}^{m} cor_{j,k}^{i} \tag{2-10}$$

C. PUF implementation and test suite

1) Overview

Zedboard, an FPGA development board based on Xilinx Artix-7 FPGA and Zynq microprocessor, is the hardware. On the software side, the major work is on Xilinx Vivado 2019.1 (Vivado) and Xilinx SDK 2019.1 (XSDK). Vivado is a software suite for synthesizing and implementing HDL designs, and Vivado has a built-in simulator, which allows function verification without actual tests. XSDK is the integrated design environment (IDE) for creating embedded applications on Xilinx's microprocessors. Based on the corresponding board support package (BSP), application projects on XSDK complete the implementation suite. The FPGA bitstreams program Zedboard, and the application carries the real field experiment. Data post-processing, including interpreting raw data, PUF characterization, and data analysis, is based on Mathworks Matlab.

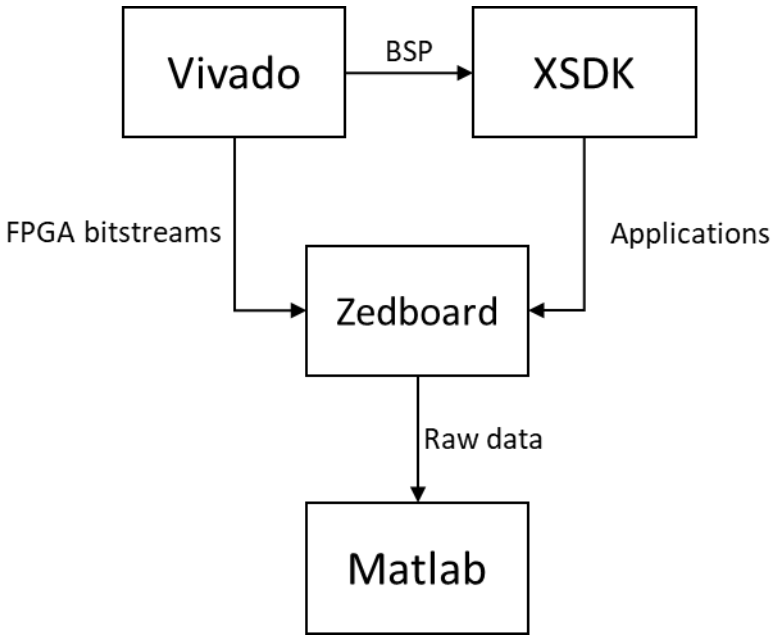
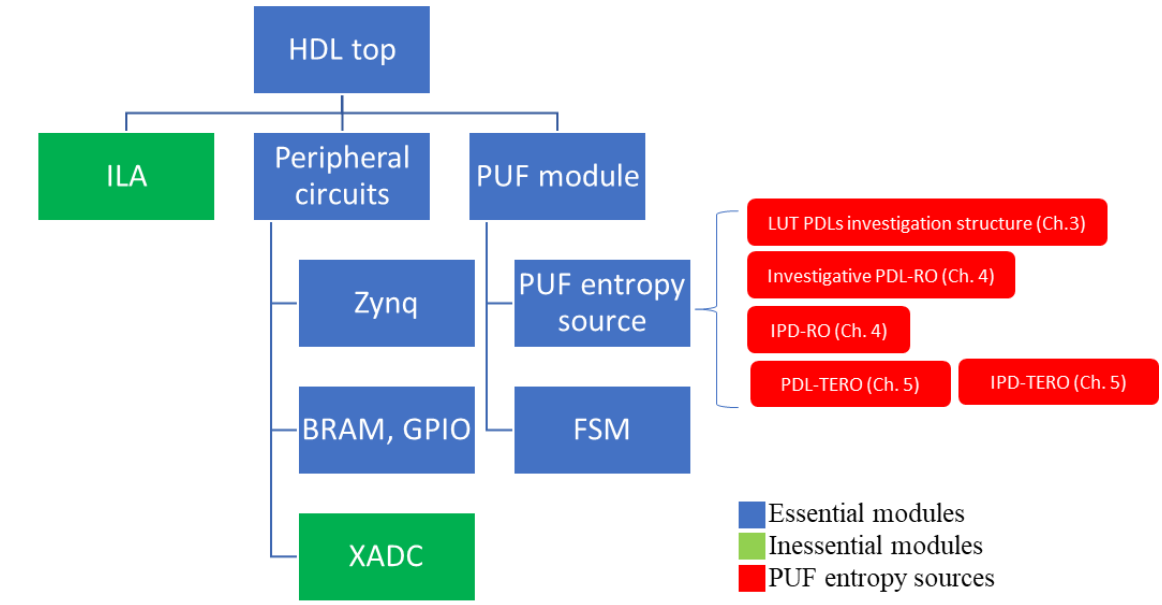


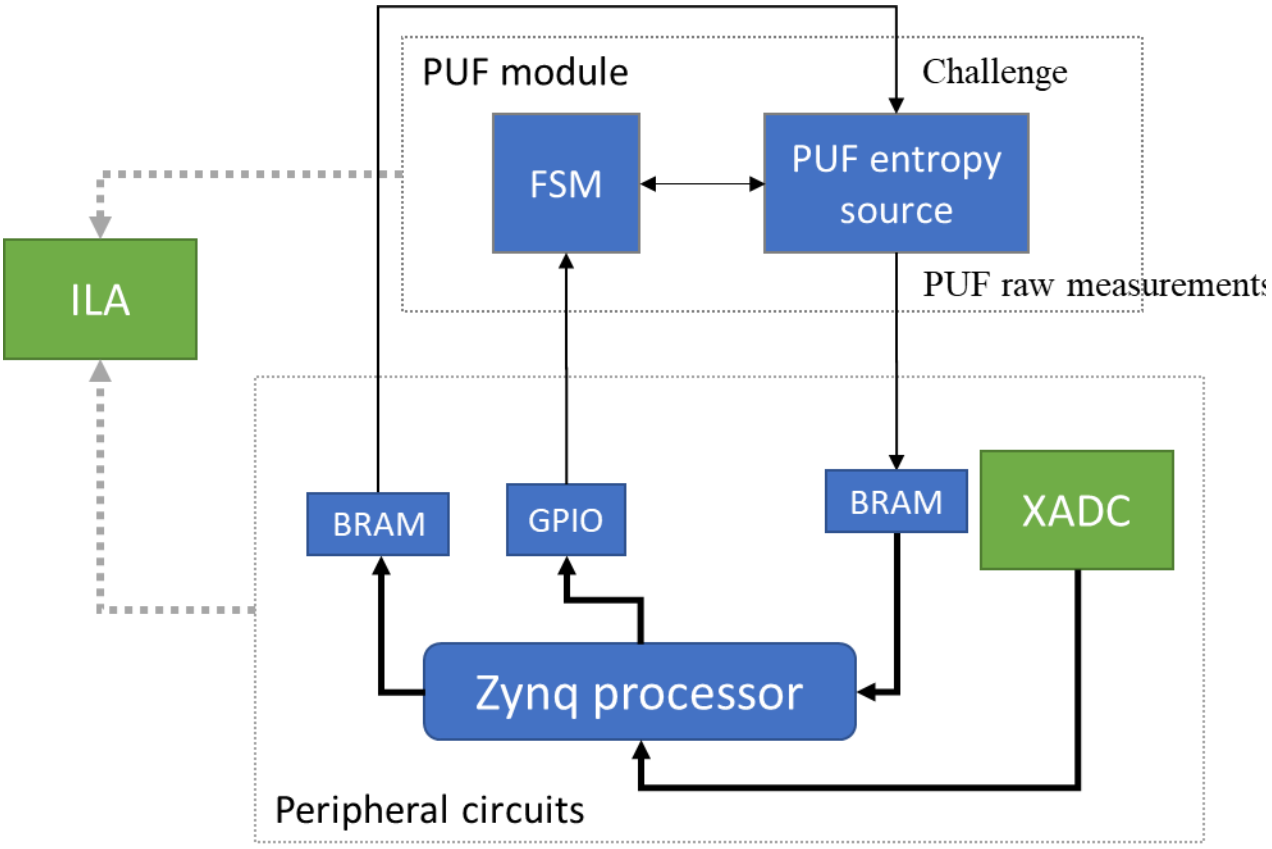
Figure 2-12 Implementation and test suite hierarchy

2) HDL design

The design incorporates HDL design and IP-based block diagram design. Figure 2-13(a) shows the hierarchy of the HDL design project. The two main function blocks are PUF module and its peripheral circuits, and ILA (Integrated logic analyzer) is used when debugging is needed. Figure 2-13(b) shows the PUF test suite diagram. Basically, we operate the suite by the application running on Zynq processor. And Zynq communicates with the PUF module through the GPIOs. BRAMs are the buffers helping Zynq to send the testing challenges to PUFs and receive the raw measurements from PUFs. PUFs under test, e.g., those listed in Figure 2-13(a), are packaged inside the PUF entropy source.



(a)



(b)

Figure 2-13 (a) Vivado project hierarchy. (b) PUF test suites diagram.

a) Peripheral circuits

Peripheral circuits are block design based on Xilinx IP. The most important parts include Zynq

processor and BRAM. BRAMs are connected to Zynq by AXI (Advanced eXtensible Interface) bus. BRAMs are all 32-bit width. In SDK software, control signals would control the enable and reset inputs to control the reading and writing of BRAM. XADC is the sensor in Xilinx 7-series FPGA. It is used for monitoring environmental factors like on-chip temperature and voltage. Further discussion is in CHAPTER 5.

A list of control signals can be found in Table I. InSwitch, InSwitchMode, RepeatCount and Twidth are configuration signals. After FPGA bitstreams are launched on board, SDK application controls them. BRAM_Reset, InSwitch, and Start_cnt are not interfaced to C code. They are only FPGA register signals, used by both peripheral circuits and PUF modules. The FSM controls these control signals.

Table I List of control signals in HDL.

Name	Width	Function	Interface to C code?
BRAM_Reset	1	Reset writing/reading BRAM, FSM	Yes
Module_Enable	1	Controls FSM	Yes
InSwitch	1	*Switch-bit*	No
InSwitchMode	1	Use intertwined structure or not	Yes
Twidth	32	Number of clock cycles that RO run	Yes
Start_cnt	32	The current index of running	No
Flag_ReadDone	1	Controls FSM	Yes

b) PUF module

Several PUF entropy sources are tested in this project, including LUT PDLs investigation structure (Chapter 3), investigative PDL-RO, IPD-RO (Chapter 4), PDL-TERO, and IPD-TERO (Chapter 5).

PUF entropy source is realized mainly using Verilog and some Xilinx built-in IP. LUT6 and LUT5 are instantiated using hard macro. The use of LUT inputs is carefully handled with the help of Xilinx 7-series documentation [16]. In most cases, LUTs are implemented as inverters, so their logic values are initialized as $(5555\ 5555\ 5555\ 5555)_{16}$. In this case, as shown in Figure 2-2, when

the selection bit of a MUX is 1, the output of LUT would be 0; and when the selection bit is 0, the output is 1. An example Verilog code instantiating LUT6 as the inverter can be found below.

```
1.      (* dont_touch = "true" *) LUT6
   #(.INIT(64'b0101010101010101010101010101010101010101010101010101010101010101
   10101)) U1 (
2.          .I0(w[i]),
3.          .I1(CHL[2*(i-0)+0]),
4.          .I2(CHL[2*(i-0)+1]),
5.          .I3(CHL[2*(i-0)+2]),
6.          .I4(CHL[2*(i-0)+3]),
7.          .I5(InSwitch),
8.
9.          .O(w[i+1]));
```

In order to faithfully extract the physical characteristics of the chip, we need to avoid Vivado automated place and routing. Thus, we applied the following constraints in Vivado to ensure the LUTs are true to our HDL. In this example constraint code, line 1 rules how the LUT inputs A1, A2…A5 map to the inputs I0, I1…I5 in the hard macro. Line 2 and 3 lock one of the four LUT6s (A6LUT) in the slice located at X37Y17 used by the specified cell PUF64/ROs[31].RO_Target_i/U[0].U1.

```
1. set_property LOCK_PINS {I0:A1 I1:A2 I2:A3 I3:A4 I4:A5} [get_cells
   PUF64/ROs[31].RO_Target_i/U[0].U1]
2. set_property BEL A6LUT [get_cells PUF64/ROs[31].RO_Target_i/U[0].U1]
3. set_property LOC SLICE_X37Y18 [get_cells
   PUF64/ROs[31].RO_Target_i/U[0].U1]
```

The FSM module works synchronously with the system clock from the Zynq processor. In our design, the system clock is 100MHz. The FSM diagram is shown in Figure 2-14. The control signals trigger the transitions. The default is “0: INITIAL”. Whenever the application starts or any error occurs, FSM is back to this state. In this state, all the control signals are reset to their default values. When PUF modules is enabled, it transits to “1: IDLE”. Some configuration signals will be set accordingly for this run. Once started, the tested entropy sources are activated in the state ‘2: RUN’, whose length is limited by the control signal “Twidth”. Then, FSM transits to “3:

READ_COUNT", where counters read required values and the application reads and stores the reading in SD card. A flag is up once these operations are finished, so the FSM is back to "0: INITIAL" and ready for the next run.

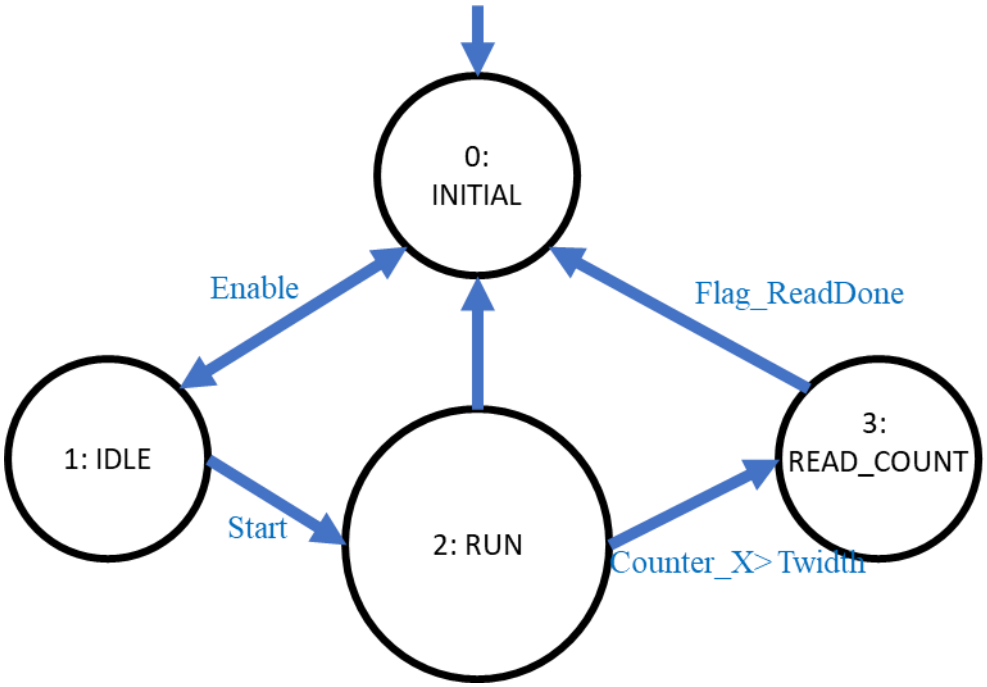


Figure 2-14 FSM diagram

3) SDK application design

XSDK directly interfaces to Vivado embedded hardware design environment. XSDK includes user-customizable drivers for all supported Xilinx hardware IPs and file handling libraries. To enable writing and reading to the SD card, 'xiffs', a generic FAT(File Allocation Table) file system library, is enabled.

Due to the limit of the BRAM, a batch of the testing data points are 2048, i.e., at most 2048 unique challenges can be stored in BRAM at a time, and Zynq from BRAM can read 2048 data points. We defined two levels of loops in the C script to maximize the test efficiency: a) Repetition; b) Cycle. Level (a) repeats the test with the same batch of testing challenges. Level (b) changes the content of input BRAM to another batch of challenges, and repeats the testing. All the raw measurements are stored in SD card for further data post-processing in Matlab until the end.

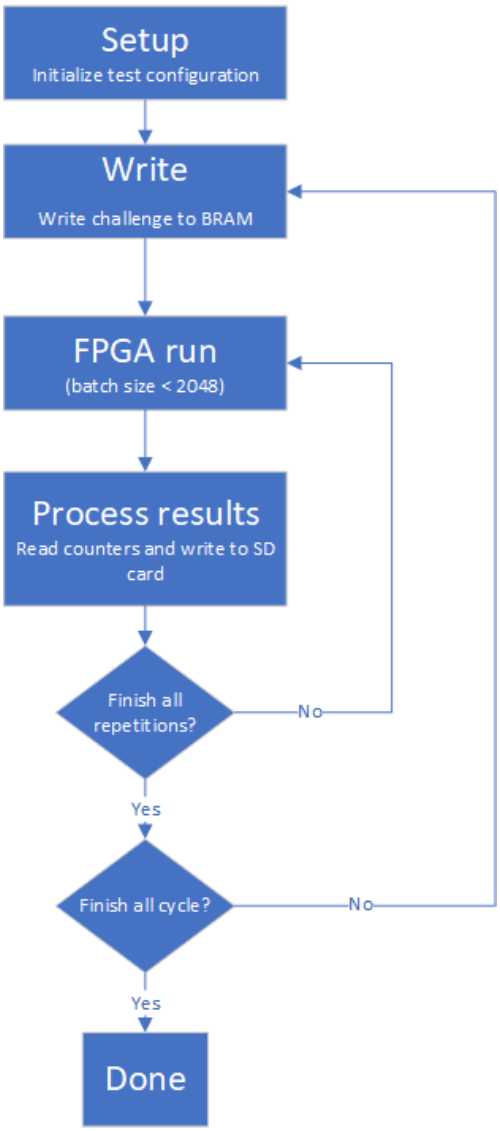


Figure 2-15 C code operation flow chart

4) Debug

Xilinx Vivado provides the integrated logic analyzer (ILA), an IP in Vivado, for developers to debug the HDL design. Configurable probes can touch the questionable signals. The debugging process is briefly described as follows:

1. In XSDK, program FPGA on hardware. Bitstream is transferred via JTAG from PC to board.
2. Launch on hardware.
3. In Vivado, developers open the hardware manager and establish the connection to the debug core in FPGA. When an ILA is instantiated in the design, the hardware manager could detect

the debug core.

4. Set the trigger condition and start the XSDK application.
5. When the trigger condition is fulfilled, ILA would capture probed signals and store them in memory for a period of time that was designated in ILA block.

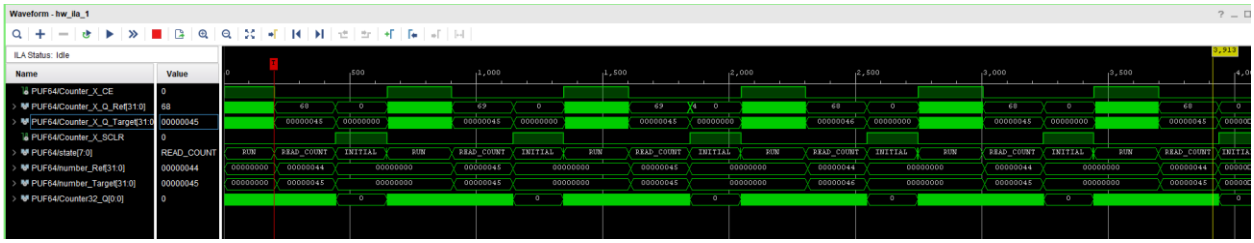


Figure 2-16 ILA debugs the FPGA operation.

As for debugging the application, I utilized XSDK's complete debug tools. When the FPGA bitstreams are already programmed in the development board, I could use 'Debug as-> Launch on Hardware' to start the debug session. The debug tool could monitor the variables in C code. Besides, the on-board memory could also be monitored by the debug tool.

CHAPTER 3. INVESTIGATION OF THE PROGRAMMABLE LUT DELAYS IN XILINX FPGAS

A. Biases in traditional RO and 2-pass RO.

1) Biases in traditional ROs

Traditional RO PUFs [2][12][15][30] are based on the "symmetrical" paths formed by identically designed inverters and interconnect wires. In FPGA implementations, such "symmetrical" paths are instantiated by design software, which are often opaque to circuit designers. In addition, FPGA chip designers and manufacturers usually do not focus on the matching among the symmetrical interconnect wires. FPGA IDE like Intel Quartus and Xilinx Vivado can help users automatically route the wires. Therefore, those "symmetrical" paths often carry systematic bias, i.e., certain paths are faster than others. Such systematic bias is due to (1) predictable wire delay differences; and (2) systematic differences among driving transistors layouts. The systematic bias reduces the randomness or entropy of traditional RO-based PUFs designs.

For instance, as shown in Figure 3-1, interconnect delay A is smaller than B due to the wire difference. In this situation, comparing RO A and RO B responses is likely deterministic, and thus, the PUF response is affected. Feiten has comprehensively investigated the delay biases in Intel Cyclone IV FPGA, due to LUT inputs, ROs' locations, load, etc. [4]. These biases showed impacts on the traditional RO-based PUFs, and they proposed a method to subtract the biases from the average biases found in the known ROs.

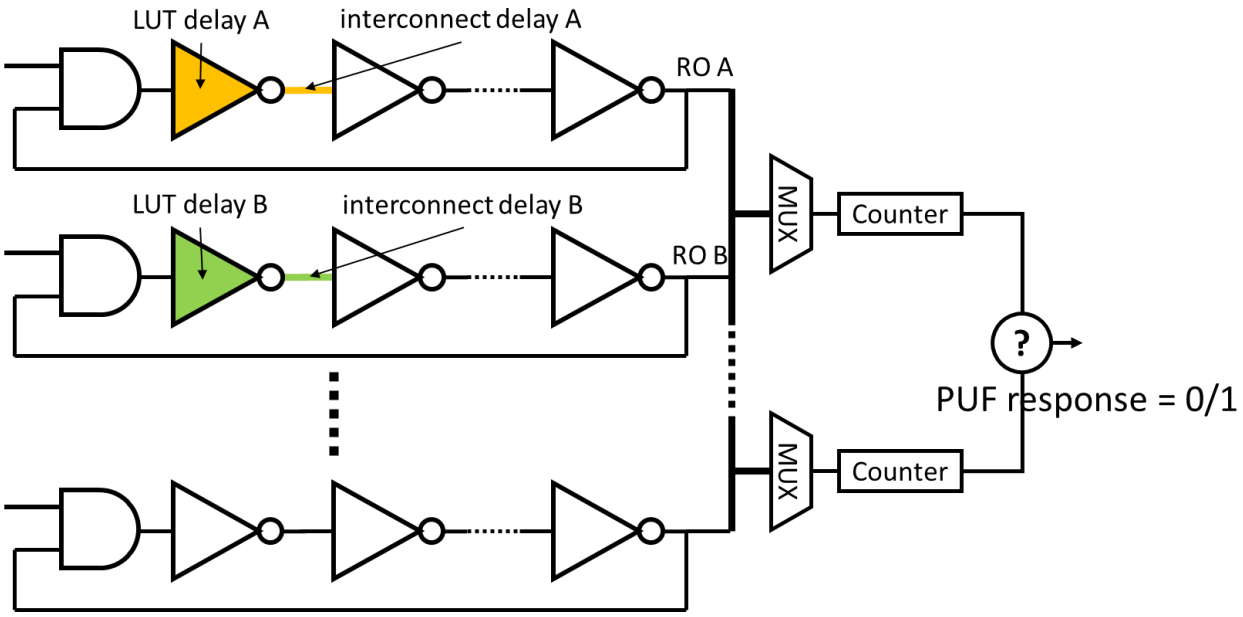


Figure 3-1 In traditional RO PUF, the wiring may carry systemetic biases.

2) Biases in PDL-RO PUF

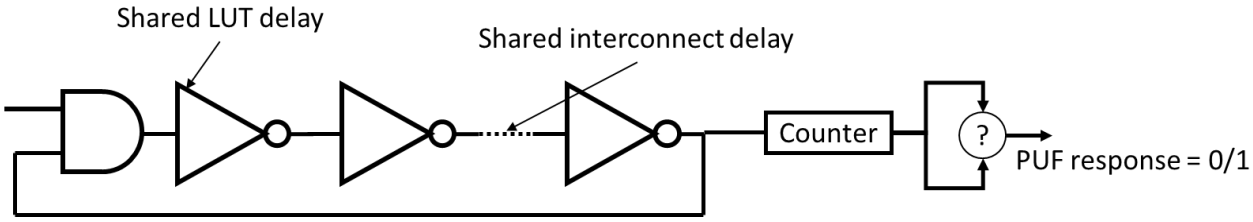


Figure 3-2 Concept of the PDL-based 2-pass PUF architecture

Feiten's PDL-RO-PUF [18] uses a single physical LUT-based RO for PUF response generation. Because the same interconnect is used in two passes, the bias due to interconnect mismatch does not exist. For the same reason, the biases between RO locations reported by Feiten in [4] won't affect PDL-RO-PUF. However, two potential issues remain for Feiten's PDL-RO-PUF.

a) Potential biases in LUT structure

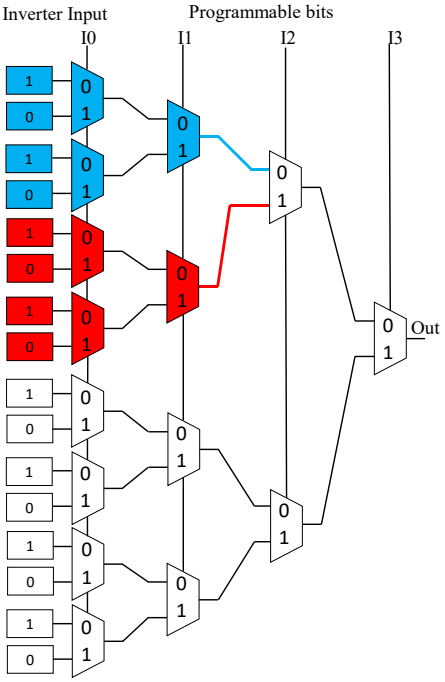


Figure 3-3 Biases in LUT structures.

The internal structure of LUT may also lead to systematic biases. The biases inside the LUT structure are likely due to the layout and manufacturing process. Figure 3-3 shows an example in which input I2 selects between the red and blue inverters. If the delay of the red wire is systematically larger than the corresponding blue wire, the delays of all the PDLs configured by I1,I2,I3=x10 tend to be larger than the ones by I1,I2,I3=x00.

If such systematic biases exist, attackers can guess the PUF delays corresponding to certain challenges based on the measurement results obtained from another chip of the same model. Feiten only selected VRO pairs with equal nominal delays [18]. Consequently, only a sub-set of the challenges can be used, introducing a loss of entropy. Therefore, we are interested in a PUF design that overcomes such difficulty. Identifying the biases in LUT is the key to the mitigation strategy. Therefore, we will experimentally investigate the LUTs in the remaining of this chapter.

b) Environment variation between two passes

The potential system clock timing variation between two passes becomes a new concern because

the system clock is not steady. An unsteady system clock may cause the acquisition times of the two passes to be different. Besides, voltages and temperatures are two main factors impacting the oscillation frequency of ROs. When ROs are active during the operation, voltage and temperature are unlikely to be consistent in the two passes. Thus, the impacts of the oscillations due to these two factors are different, which may cause biases or reliability issues. We will use a reference RO to solve this problem and the discussion is in CHAPTER 5.

B. A brief look at LUT structures in FPGAs

1) Intel FPGAs

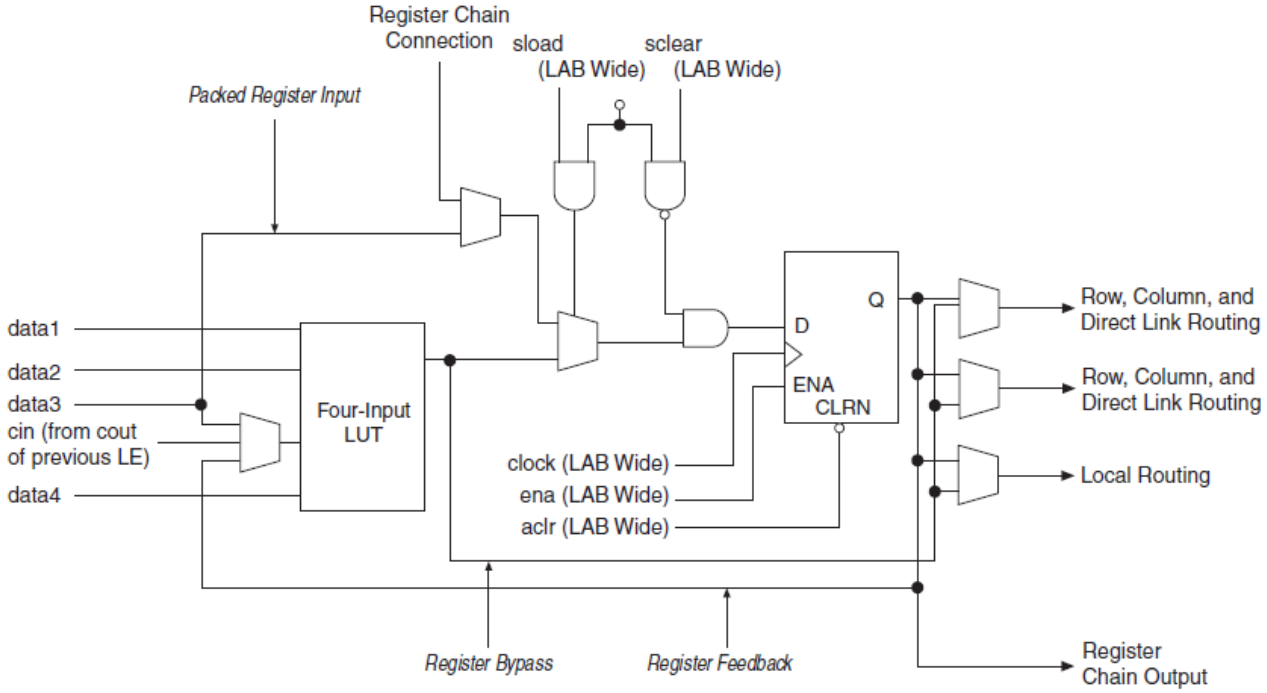


Figure 3-4 Cyclone IV Device LEs in Normal Mode[19]

Figure 3-4 shows the logic element (LE) in Intel Cyclone IV FPGAs. Each LE has one 4-input LUT. The output wire of the 4-input LUT goes through MUX and registers before it exits the LE.

Based on Intel Cyclone IV, [18] presents the experimental result of the nominal delays, as shown in Figure 3-5. Through this result, one could sketch the biases in Cyclone IV FPGA LUT. The pattern is relatively simple. The nominal delays are roughly at two different levels. One is for the

configurable bits 000~011, and the other is 100~111. Theoretically, a 4-input LUT can be formed by two 3-input LUTs. While the internal structure is not revealed, we speculate that two physical 3-input LUTs reside in a 4-input LUT. The internal wires are not well matched. And the internal wiring of the 3-input LUTs is relatively symmetrical.

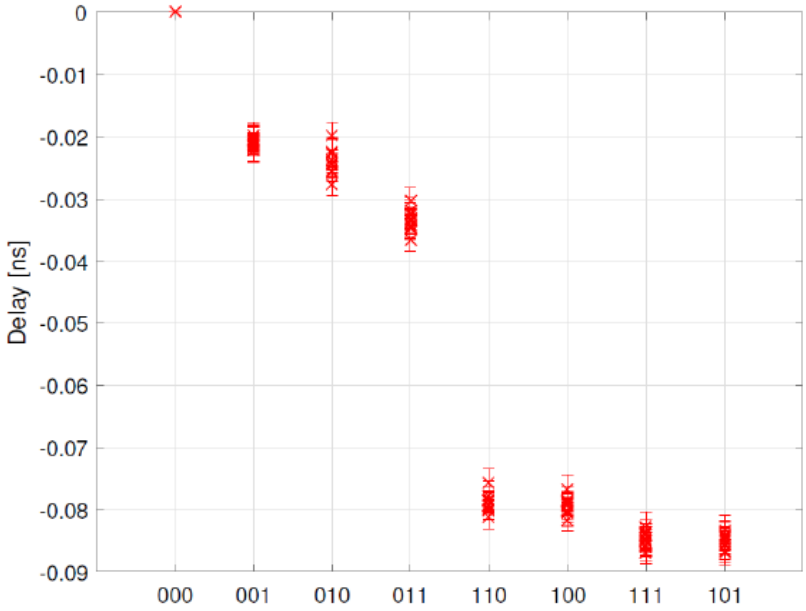


Figure 3-5 Experimental results showing the average (i.e. estimated nominal) delays of all LUT configurations assignments. [18]

Some other Intel FGPAs, like Stratix II, use 6-input LUT [20], which results in greater complexity in the pattern of the biases in this LUT. Figure 3-6 shows the structure of the 6-input LUT in Stratix II. A 6-input LUT can be broken down into two 5-input LUTs. Then, while each 5-input LUT can be broken down into two 4-input LUTs, one 4-input LUT is further broken down into two 3-input LUTs. Intel specifically designs this to achieve some unique properties in LUT. This design would also bring some unique nature in its biases pattern.

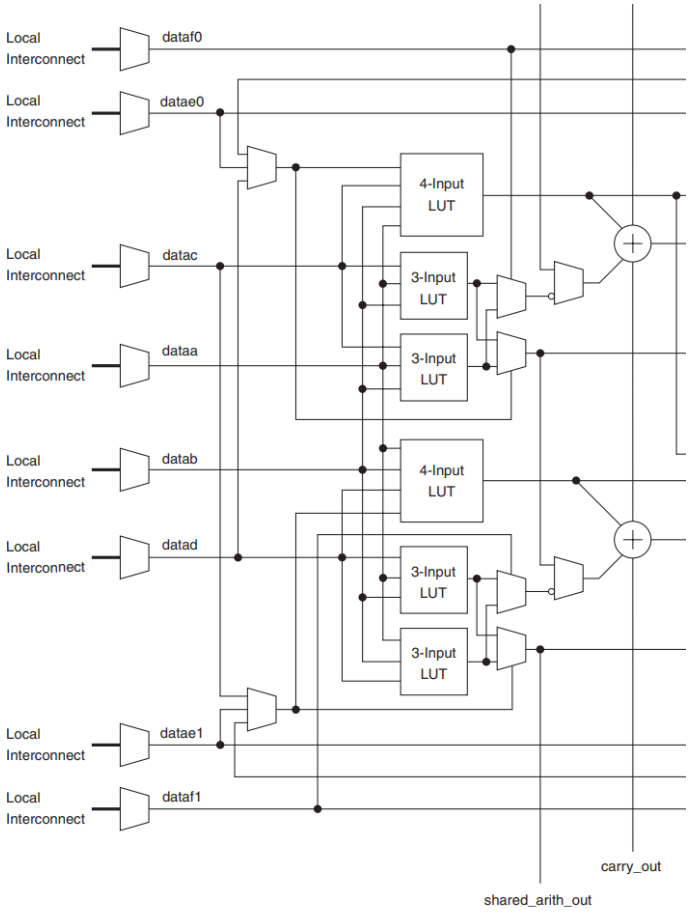


Figure 3-6 Adaptive Logic Module (ALM) used in Intel Stratix II Block Diagram [20]

2) Xilinx FPGAs

A layout view of Xilinx Artix-7 FPGAs is shown in Figure 3-7, and related information can be found in [53]. In Xilinx Artix-7 FPGAs, the fundamental configurable element is called configurable logic block (CLB). A CLB element contains a pair of slices. In each CLB, there are four LUT6s, several storage elements, multiplexers, and other elements.

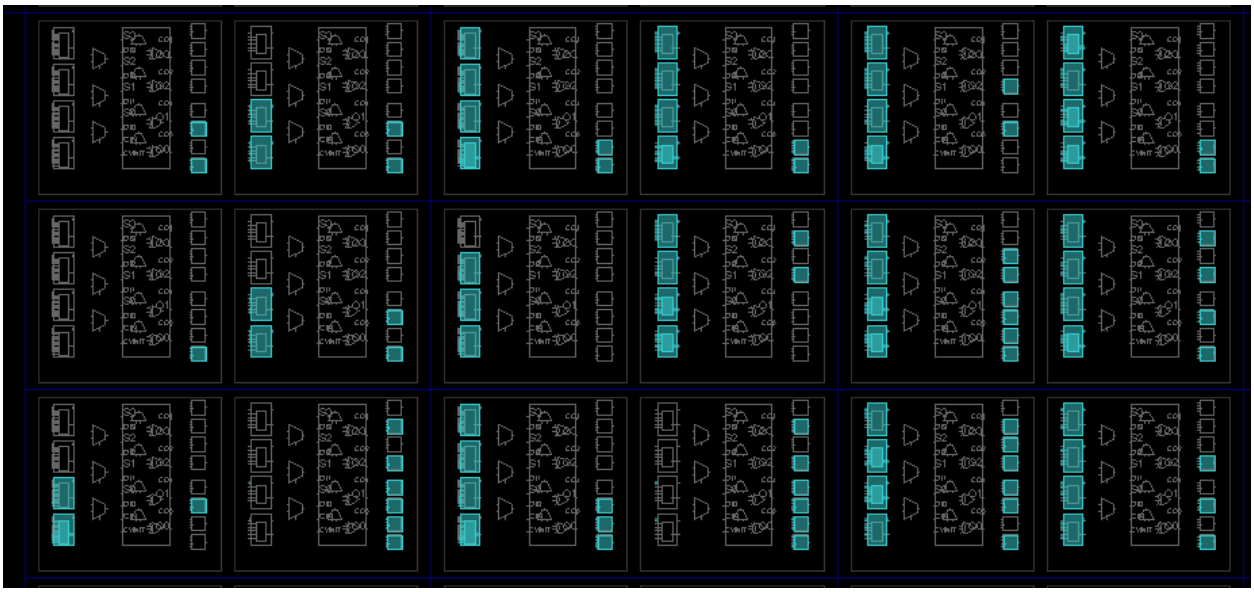


Figure 3-7 Device view of Vivado, showing Xilinx Artix-7 FPGAs CLBs.

CLBs in Xilinx Artix-7 FPGAs have two types: SLICEL and SLICEM. Figure 3-8 shows the diagram of SLICEL. One can find that the outputs of LUT6s go through a long wire to reach the output pin of the CLB, and it also connects to several MUX for logic functions. Depending on the implementation, one may be able to estimate the delay in the wire.

I have the following assumptions about the delay biases in Xilinx Artix-7 FPGA CLB:

1. Due to the design layout, there are systematic delay biases in four LUT6s output wires.
2. The delay of the LUT6 output wire is affected by the use of CLB. If elements in the same CLB is used for other functions, the delay is affected.
3. Due to the manufacturing process, a CLB's location may also cause the deterministic variation in the LUT6 output wire. This assumption is partially supported by the work of Feiten in [4].

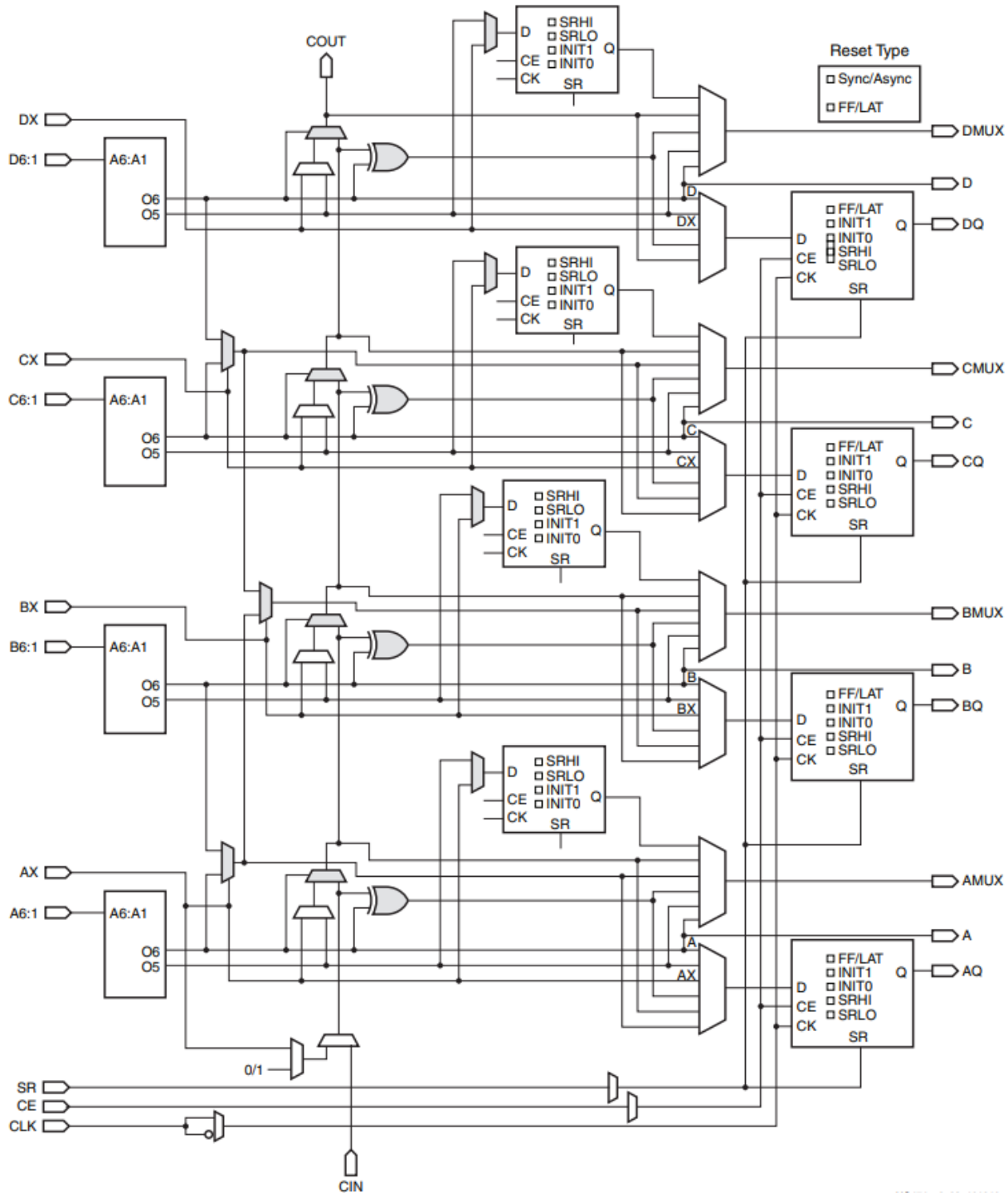


Figure 3-8 Diagram of SLICEL [53].

Then, let's take a look at the internal structure of the LUT6 of Xilinx Artix-7 FPGAs. According to the Xilinx 7 series FPGA datasheet, each LUT6 consists of two physical LUT5 [16]. This LUT6 structure is shown in Figure 3-9. Input I5 of the LUT6 selects the LUT5 that renders the output of

LUT6. As Xilinx's datasheet does not provide any details in LUT6, the knowledge about the LUT6 layout is absent. It was claimed that LUT6 in Xilinx Virtex-5 has the smallest delay when I1~I5 are 00_{16} and the biggest delay when $1f_{16}$ [57]. [2] indicated that there might be a pattern in the Xilinx Spartan LUTs. Indeed, there is no published experimental data for detail. Thus, I'm motivated to investigate the delay biases in LUT structures.

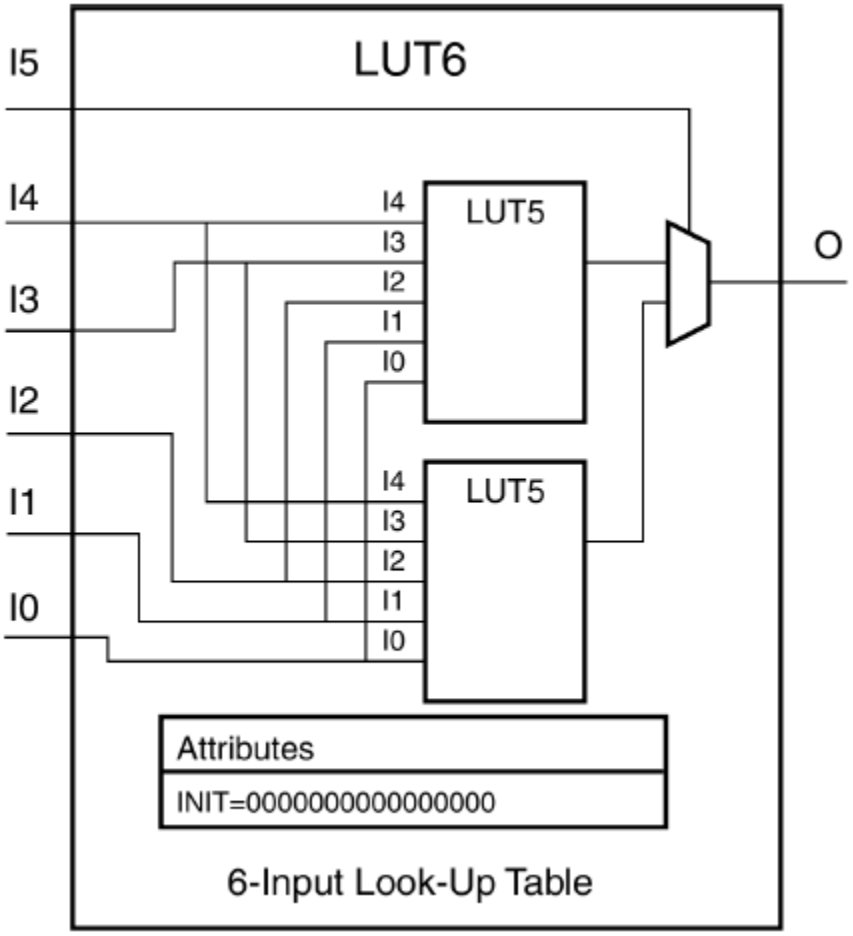


Figure 3-9 Xilinx Artix-7 FPGA LUT6 [16]

C. Experimental investigation on the systematic bias on programmable LUT delays in Xilinx FPGA

1) Experiment Setup and Expectation

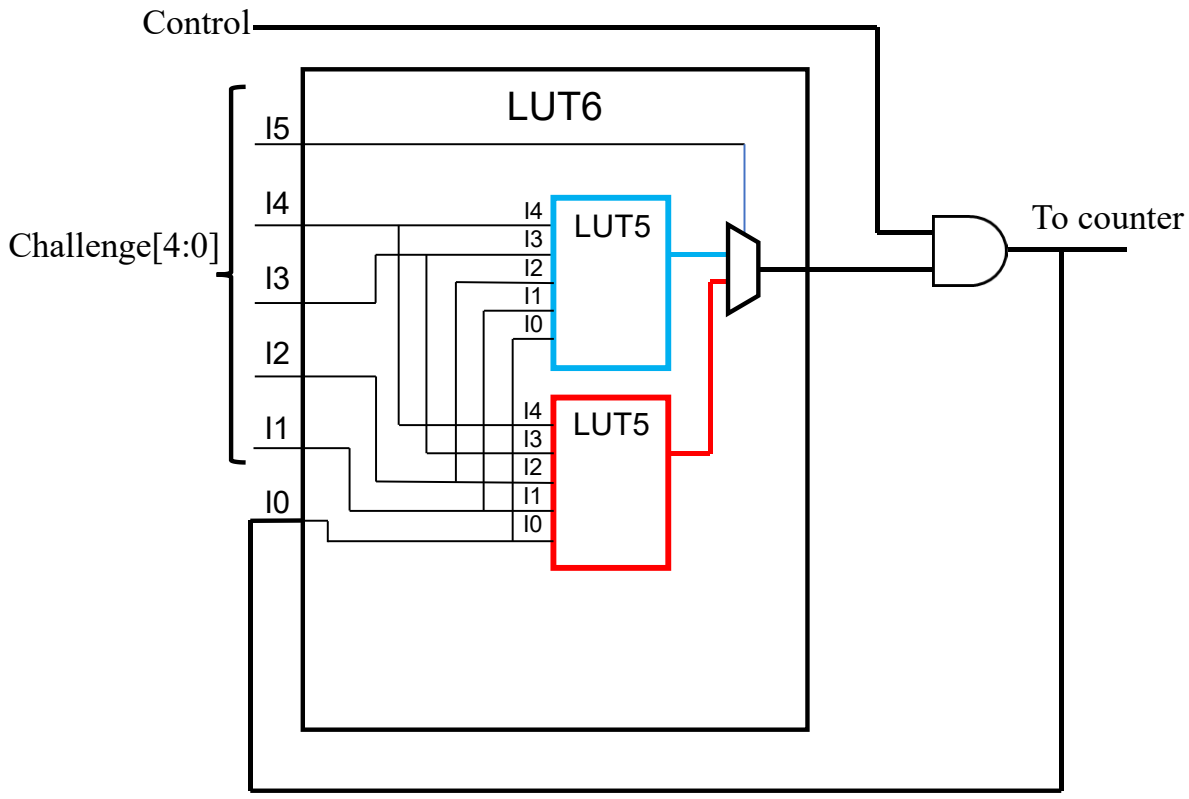


Figure 3-10 Structure of single Xilinx LUT6 4-bit RO. [16]

To investigate the internal delay of LUT6 in Xilinx Artix-7, we experimented with LUT6s by building single LUT6-based ROs. ROs oscillation counts are used to measure the delays of the programmable LUT delay paths. The structure of the experimented RO is shown in Figure 3-10. Each RO consists of an AND gate and a LUT6. Input I0 of the LUT6 is connected to the output of the AND gate, so the loop is formed. Input I1~I5 are fed with 5-bit challenges, picking one of the 32 possible LUT delay paths. The MSB of the challenge is used to determine which LUT5 is used in the specific run.

On each device, 96 CLBs are tested. These CLBs are on different parts of the FPGA. To cover all the cases of LUT6, I tested all the eight LUT6s in each CLB. In total, 96*8=768 LUT6s are tested on each device.

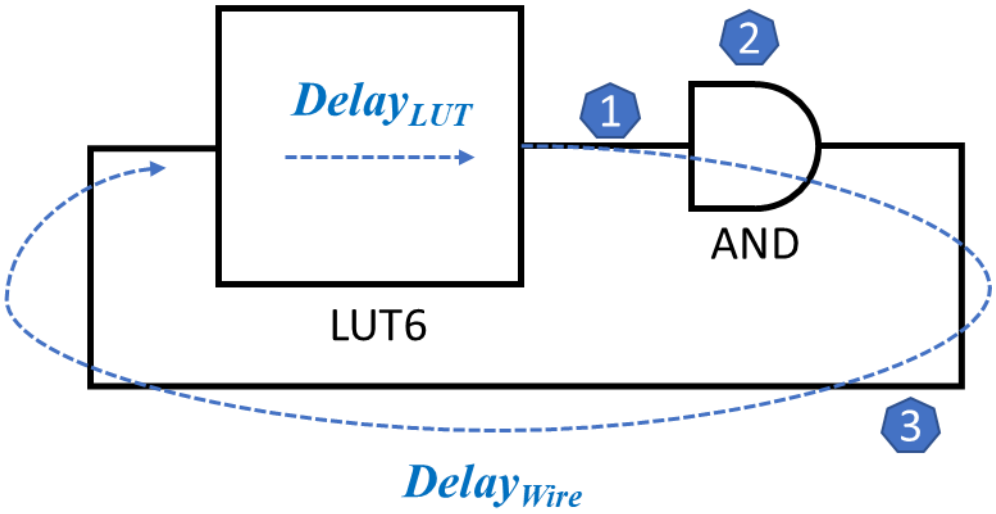


Figure 3-11 The delay diagram of the investigation setup, a single-LUT6-based RO.

Figure 3-11 shows a delay diagram of this experimental setup. The total delay is compromised of two parts: $Delay_{Wire}$ and $Delay_{LUT}$. $Delay_{Wire}$ is made up of three parts: the wires connecting from LUT output to the AND gate, AND internal wire, and the wire from AND gate output back to I0 of LUT. Figure 3-11 marks them with 1, 2, 3, respectively. $Delay_{LUT}$ is the delay of the LUT internal wire. For the different tested PDLs, the $Delay_{LUT}$ is different. For the LUTs at different locations, the wires connecting LUT and AND gate are different. In FPGA IDE tools, designers do not control the placement of the wires. Therefore, when implementing LUT-based ROs at different locations, there is always some mismatch in $Delay_{Wire}$. In this chapter, the difference in the $Delay_{LUT}$ is the biases that we are investigating.

2) Results: Bias between two LUT5s

The delays of tested PDLs are calculated with the raw measurements of the tested ROs. Figure 3-12 shows the number of oscillation cycles measured in the 32 unique PDLs in a LUT6. Each of the LUT programmable bits corresponds to a PDL. For each PDL, we carried out multiple measurements to ensure the accuracy of our investigation, thus each cluster in Figure 3-12 has

some spread. The blue and red clusters are the measurements of the PDLs in two separate LUT5s, respectively. The MSB of the LUT inputs selects one between two LUT5s in a LUT6. It is clearly shown in Figure 3-12 that PDLs in two LUT5 have systematic bias. The delays of the PDLs in one LUT5 (in red) are systematically larger than the PDLs in the other LUT5 (in blue). This systematic bias is similar to the one in Cyclone IV, as shown in Figure 3-5.

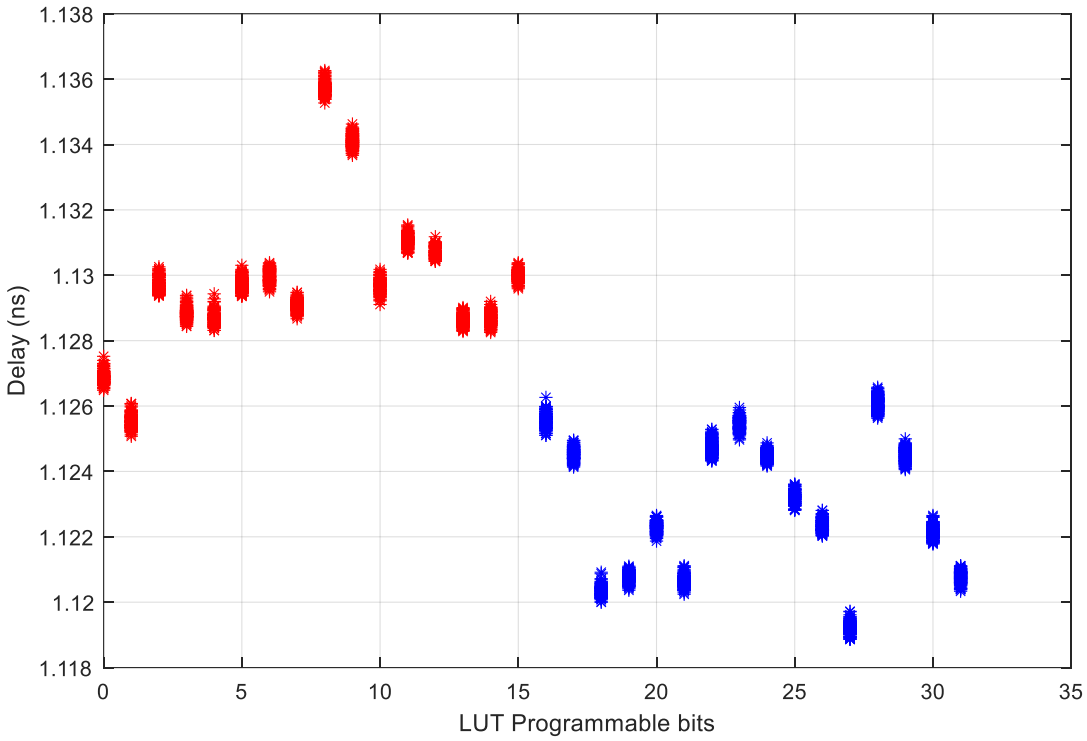


Figure 3-12 Number of oscillation cycles distribution for single LUT6 with 5-bit challenge.

While Figure 3-12 shows the results from one RO, Figure 3-13 shows the results from 32 ROs located at different parts of two FPGA chips. It is clearly demonstrated that systematic bias between corresponding programmable LUT delay paths in two LUT5 within the same LUT6 exists in Xilinx Artix-7 FPGAs.

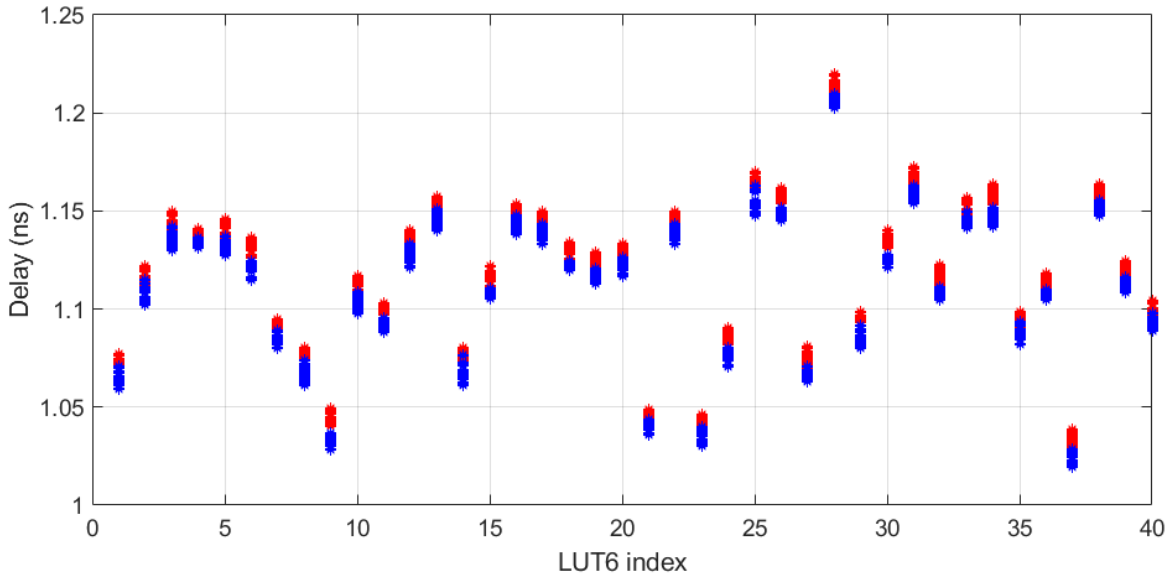


*Figure 3-13 Delay of PDLs in tested LUT6s randomly picked from two devices. Each red/blue cluster includes all the sixteen PDLs in a LUT5, thus having 200(samples)*16(PDLs)=3200 samples.LUT6 indexed from 1 to 20 is from the same device, and the ones indexed from 21 to 40 are from the other device.*

3) Results: The bias pattern in LUT5

The next question is whether systematic biases exist among the PDLs within each LUT5. If not, the PUF can be designed based on any two competing delay paths within the LUT5 cell. Otherwise, a new design is required to mitigate the systematic bias among all PDLs within the LUT5. Therefore, we conducted another experiment, which also measured the delays of the PDLs using LUT-based ROs. We now include eight cascaded LUTs programmed using the same configuration bits to enhance the signal-to-noise ratio in the measurements. We tested 160 such ROs on each of the two devices. In this manner, each measured delay includes the same PDLs in the eight LUT6s.

Figure 3-14 shows that the delays of the thirty-two 8-stage PDLs are approximately 4.9ns. The delay per stage (approximately 4.9/8=0.6125ns) is smaller than the delays shown in Figure 3-12 because there is less delay in the interconnect per LUT stage. Biases between the top and bottom LUT5 are still noticeable. Moreover, there is a repeating pattern for every eight clusters. In addition, the pattern in the red clusters is the same as that in the blue clusters. These patterns prove

the existence of systematic biases in LUT5s. We deduce that the bias is due to the design layout of LUT5s.

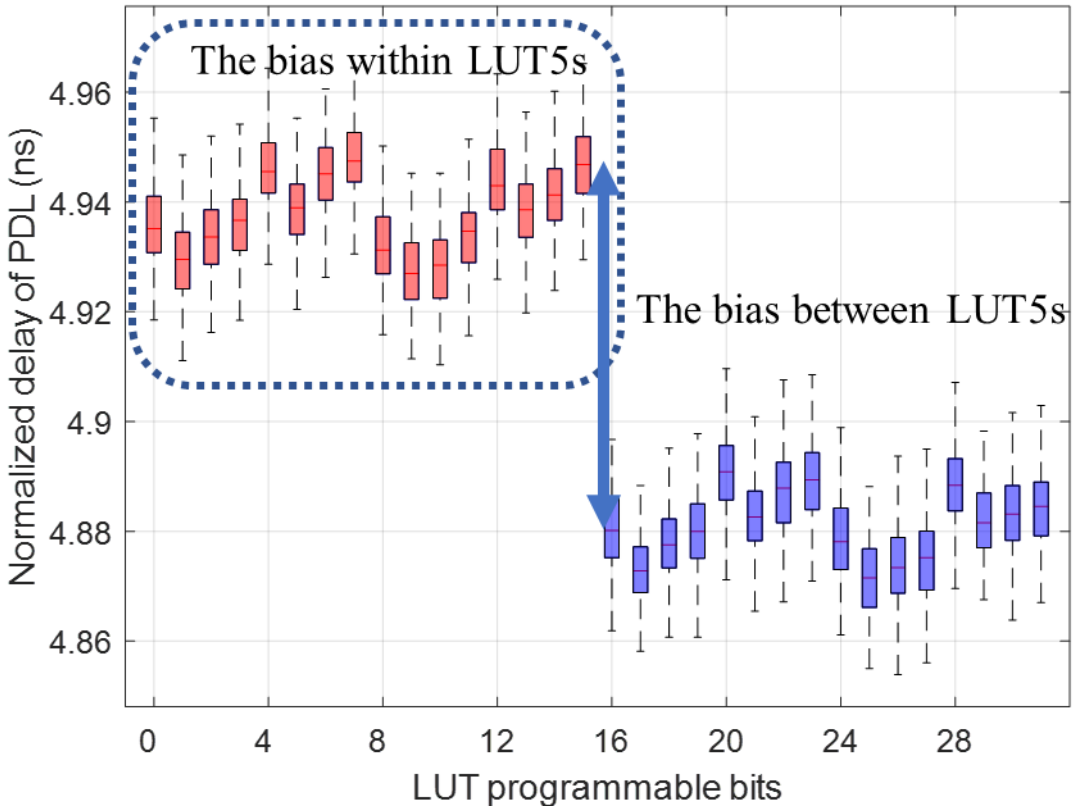


Figure 3-14 Boxplot of 32 instances of 32-bit RO configured by 4-bit repeated LUT programmable bits. Biases between each RO is neglected here.

D. The Delay Model of LUT6 in Xilinx Artix-7

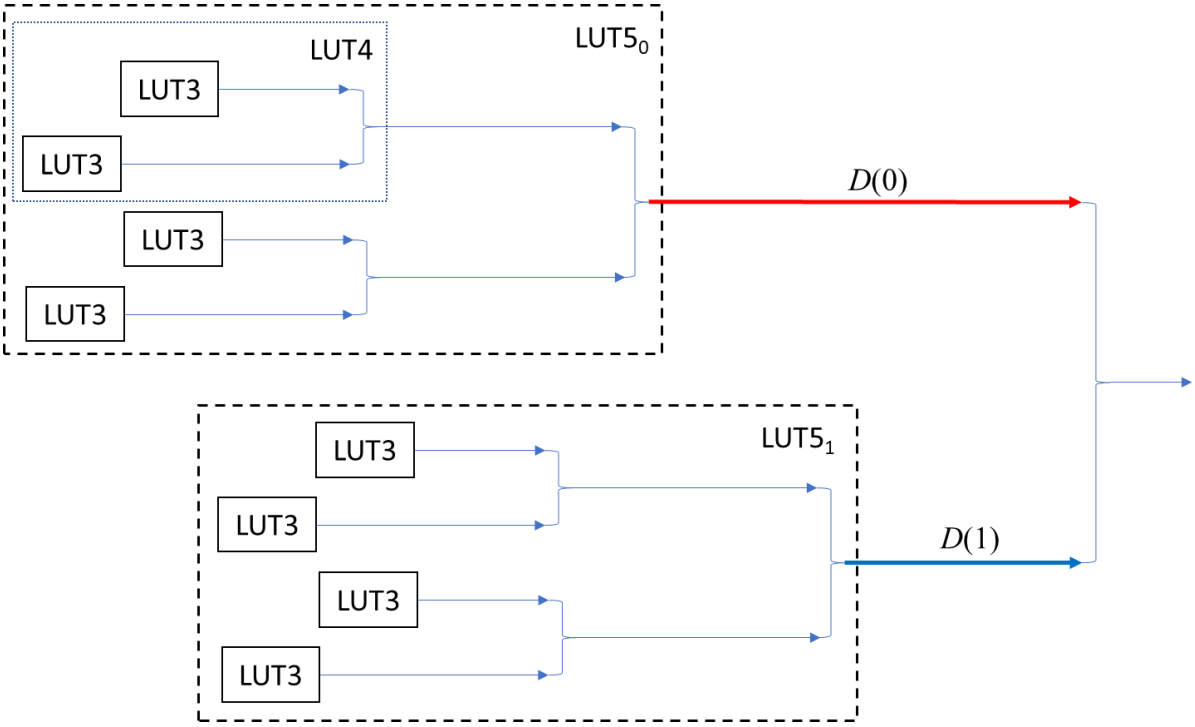


Figure 3-15 Diagram of the delay model of LUT6

The investigation experiments results could derive the delay model for LUT6 in Xilinx Artix-7, whose diagram is shown in Figure 3-15. To distinguish the two LUT5s, the LUT5 that is used when LUT6 input I5 is 0 is called LUT5$_0$, and the other is LUT5$_1$. The colors (red and blue) correspond to the experimental delays shown in Figure 3-14. $D(0)$ is the nominal delay of the propagation wire from LUT5$_0$ to the output of the LUT6, and $D(1)$ is the nominal delay of the propagation wire from LUT5$_1$. From experimental data, it was found $D(0)$ is systematically larger than $D(0)$. Therefore, in Figure 3-15, the propagation wire from LUT5$_0$ is longer than LUT5$_1$. The major systematic delay difference between the two LUT5s is,

$$\Delta D = D(0) - D(1) \tag{3-1}$$

As shown in Figure 3-14, the measured value of ΔD is about 7.77 ps.

As for the 16 PDLs in a LUT5, the first observation is that the delays of the PDLs 0~7 are about the same as those of PDLs 8~15. Therefore, in the delay diagram, the propagation delays from two

LUT4s to the output of LUT5 are the same. Second, the delay patterns are very close in each LUT4, i.e., each group of four clusters in Figure 4-13. The pattern is, the delay of the $LUT3_0$ is systematically smaller than the one of $LUT3_1$.

Since the biases within LUT5 is relatively minor, I describe the delays of PDLs in LUT5s by $\delta(I1{\sim}I4)$, which is in a pattern. In Figure 3-14, $\delta(I1{\sim}I4)$ is approximately 2.50 ps.

For one of the 16 PDLs in a LUT5, the delay is composed of $D(I5)$ and $\delta(I1{\sim}I4)$. Therefore, the delay of a PDL that is chosen by inputs $I1{\sim}I5$ is expressed as,

$$d_{I1\sim5} = D(I5) + \delta^{I5}(I1{\sim}I4) \tag{3-2}$$

We conclude the experimental investigation of the biases here and will propose a structure to mitigate the found biases. The delay model in (3-2) will be used to demonstrate the effectiveness of our mitigation method.

CHAPTER 4. INTERTWINE PROGRAMMABLE DELAYS FOR BIASES MITIGATION

A. From single LUT stage to intertwined LUT stage

1) The Traditional 2-Pass Scheme on Single LUT Stage

First, let's look at how the traditional 2-pass scheme, i.e., two different challenges are used to configure the RO, works with the single LUT stage. This method is the one used in PDL-RO-PUF [18]. Two sequential samplings are compared, so the 1-bit PUF response is determined. The competing elements are defined as:

$$d(chl) = D(I5) + \delta^{I5}(I1{\sim}I4) \tag{4-1}$$

$$d(chl') = D(I5') + \delta^{I5'}(I1'{\sim}I4') \tag{4-2}$$

Therefore, the compared results is:

$$d(chl) - d(chl') = [D(I5) - D(I5')] + \left[\delta^{I5}(I1{\sim}I4) - \delta^{I5'}(I1'{\sim}I4')\right] \tag{4-3}$$

First, the outcome of the difference is heavily dependent on item $D(I5)$-$D(I5')$, which only two MSB of the LUT inputs determine. Second, the term $\delta^{I5}(I1{\sim}I4)$-$\delta^{I5'}(I1'{\sim}I4')$ is also affected by systematic biases. Figure 3-14 shows the pattern in delays of the PDLs in LUT5s. For certain $I1{\sim}I4$ and $I1'{\sim}I4'$ values, the outcome of $\delta^{I5}(I1{\sim}I4)$-$\delta^{I5'}(I1'{\sim}I4')$ is biased reduces the complexity of the PUF response. Therefore, a new architecture is required to eliminate the systematic biases.

2) The intertwined structure in LPUF

Rioul introduced a LPUF structure (Figure 4-1) whose delay stage incorporates two independent delays [48]. Its PUF response was determined based on the sequential samplings of LPUF with the same challenge. Challenge *chl* is used in the 1st pass, and the inverse of challenge, $\overleftarrow{chl}$, is used in the 2nd pass. This scheme ensures that the same number of top and bottom delays are used in two passes. The same number of top and bottom delays is used for any competition of a pair challenges.

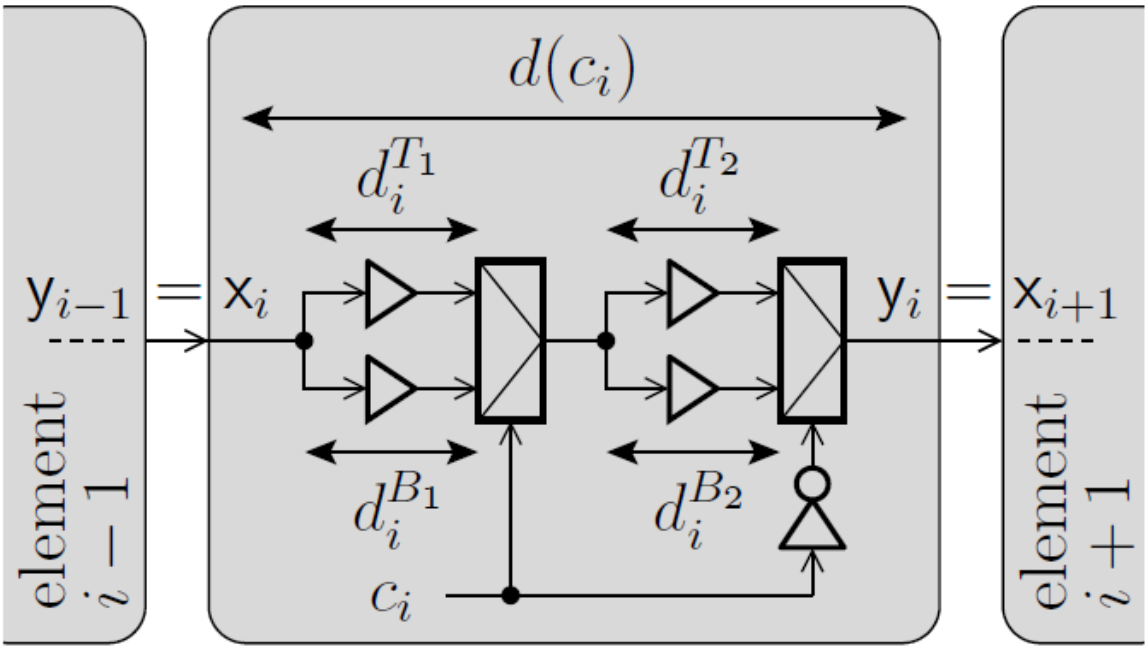


Figure 4-1 LPUF delay structure.

3) Intertwined Programmable Delay (IPD)

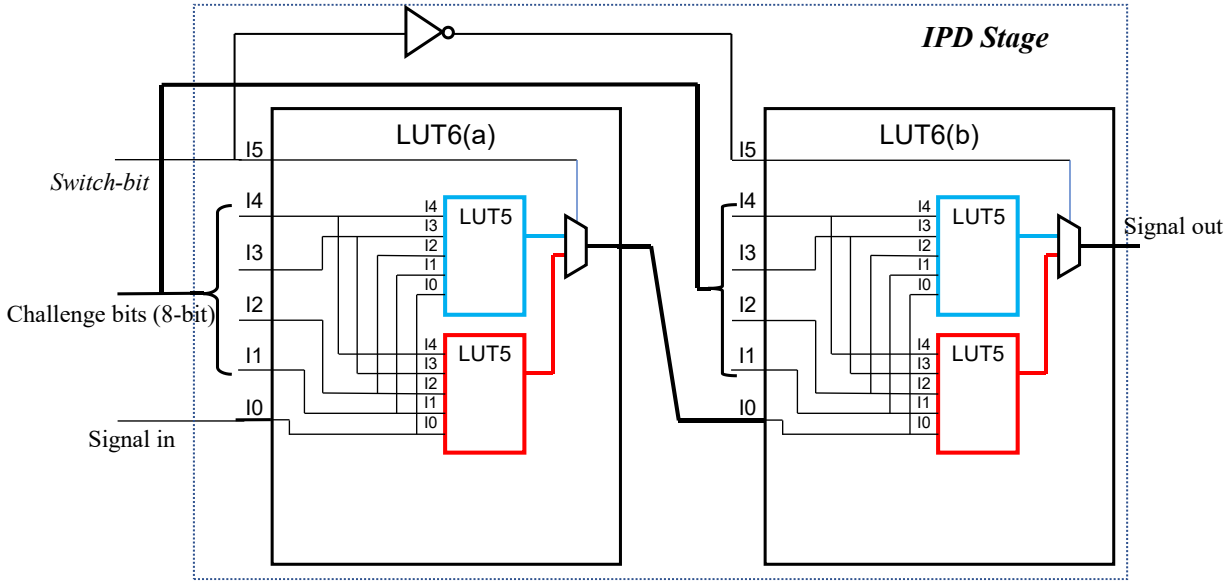


Figure 4-2 Structure of proposed intertwined LUT stage

Inspired by the intertwined structure in Rioul's LPUF structure, we paired two LUT6s into a stage, as shown in Figure 4-2. A big difference between our structure and Rioul's LPUF is that our top and bottom delays are selected by LUT6 input *I*5, and the other LUT6 inputs select the PDLs in

LUT5. While the input *I*5 is dedicated to either pass, the 4-bit inputs of LUT5s, *I*1~*I*4, are programmable. Essentially, we pair two PDLs in an intertwined manner. Thus, the new structure is called intertwined programmable delay (IPD).

Two LUT6's operating as inverters are included in one stage. Input *I*1 of both LUT6s are used as the oscillation signal inputs fed from the previous LUT6. On each LUT6, inputs *I*2~*I*5 are programmable to challenges. One of 16 PDLs inside LUT5 is chosen. Input *I*6 of LUT6(a) takes the *Switch-bit* while I6 of LUT6(b) takes the inversion of *Switch-bit*. As the MSB of LUT6 inputs, *Switch-bit* determines which LUT5 the chosen PDL resides. The output of LUT6(b) acts as the output signal of IPD and is connected to the input of the next IPD.

We'd like to discuss two strategies of using challenges as programmable bits. The first strategy is to use an 8-bit challenge for each IPD, which means a 4-bit challenge for each of the two LUT6s. In this scenario, the nominal delays of two competing paths are:

$$d(chl, 0) = [D_a(0) + \delta_a(chl_a)] + [D_b(1) + \delta_b(chl_b)] \tag{4-4}$$

$$d(chl, 1) = [D_a(1) + \delta_a(chl_a)] + [D_b(0) + \delta_b(chl_b)] \tag{4-5}$$

The difference of these two paths' delays can be divided into two portions:

$$\Delta\big(d(chl, 0) - d(chl, 1)\big) = D_a(0) - D_a(1) + D_b(1) - D_b(0) \tag{4-6}$$

$$\delta\big(d(chl, 0) - d(chl, 1)\big) = \delta_a^0(chl_a) - \delta_a^1(chl_a) + \delta_b^1(chl_b) - \delta_b^0(chl_b) \tag{4-7}$$

(4-6) represents the delay difference of the nominal delay of the whole LUT5. As explained, the intertwined structure mixes the top and bottom LUT5s, thus mitigating the bias in LUT5s.

(4-7) represents the delay difference of the deviation of a PDL from nominal delay of the LUT5. Since we have found that the pattern of delays in the top and bottom LUT5s are the same, the item $\delta_a^0(chl_a)$ and $\delta_a^1(chl_a)$ can mitigate each other.

The second strategy is to use a 4-bit challenge for an IPD, two LUT6s using the identical 4-bit

challenge as the programmable bits. In this scenario, the bias between two LUT5s is also mitigated by the intertwined structure. The situation of the mitigation of biases in LUT5s is different. (4-7) is rewritten as (4-8),

$$\delta\big(d(chl,0) - d(chl,1)\big) = \delta_a^0(chl) - \delta_a^1(chl) + \delta_b^1(chl) - \delta_b^0(chl) \tag{4-8}$$

In this situation, all programmable bits are challenge *chl.* If the top and bottom LUTs have different delay patterns in any circumstance, we still expect that the counterpart LUT5s in LUT6 are the same. Thus, $\delta_a{}^0(chl)$ can mitigate $\delta_b{}^0(chl)$, i.e., the top LUT5 in LUT6(a) mitigates its counterpart in LUT6(b).

B. IPD-RO-PUF architecture

1) IPD-RO structure

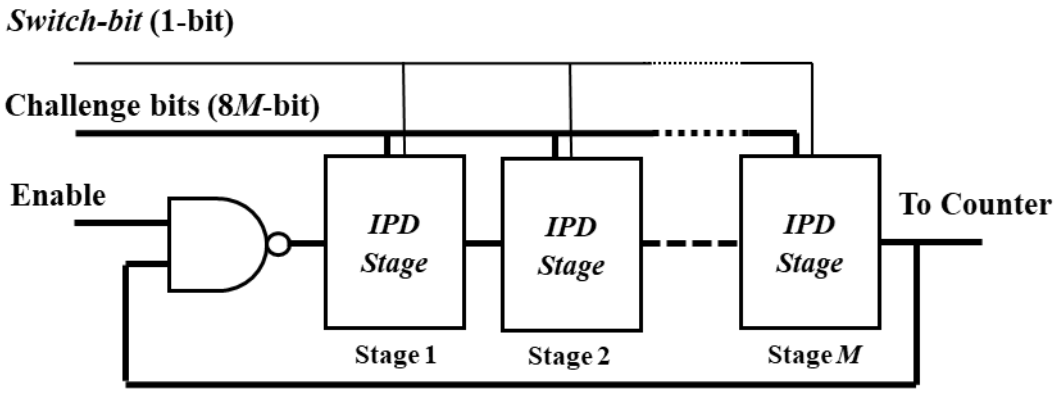


Figure 4-3 Structure of RO implemented with the intertwined LUT stages

Figure 4-3 presents the structure of RO. In addition to the even number stages of inverters, a NAND gate implemented in LUT5 is included. One of its inputs to NAND is connected to a control signal from Zynq to start and stop the oscillation in RO. The other input to NAND is connected to the output of the last inverter stage to complete a loop. The output of the last inverter stage is also connected to a buffer, whose output goes to counter.

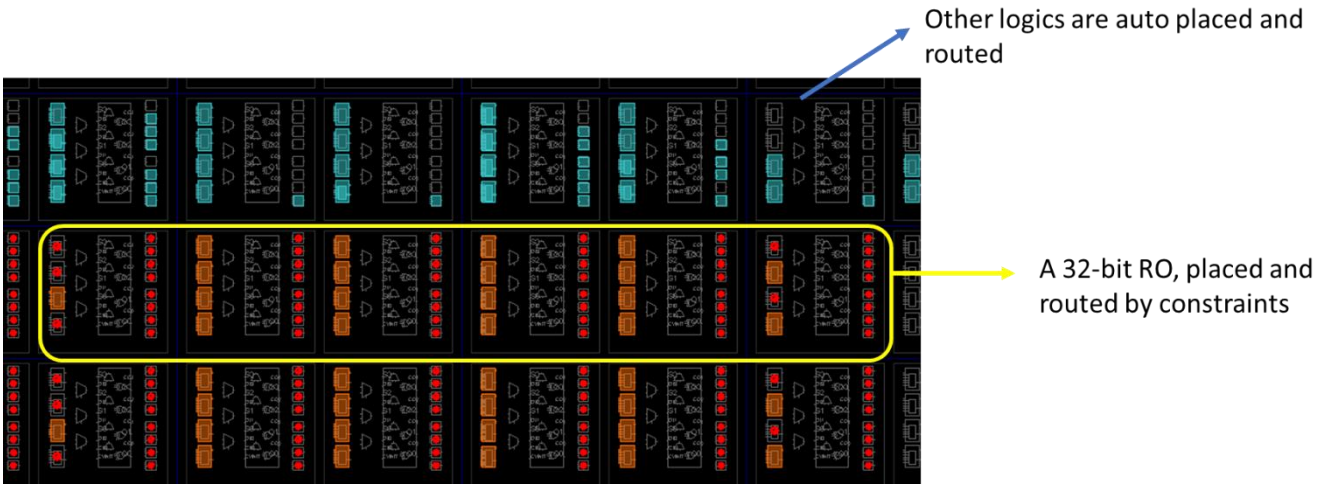


Figure 4-4 ROs should be placed carefully with constraints.

Constraints designatively assign ROs' placement and the LUT6s input pins. Letting Xilinx Vivado automatically route and place would cause RO malfunction. Two major things should be configured manually. An example of the constraints that manually assign the placement and input pints can be found in CHAPTER 2.

2) Testbench architecture

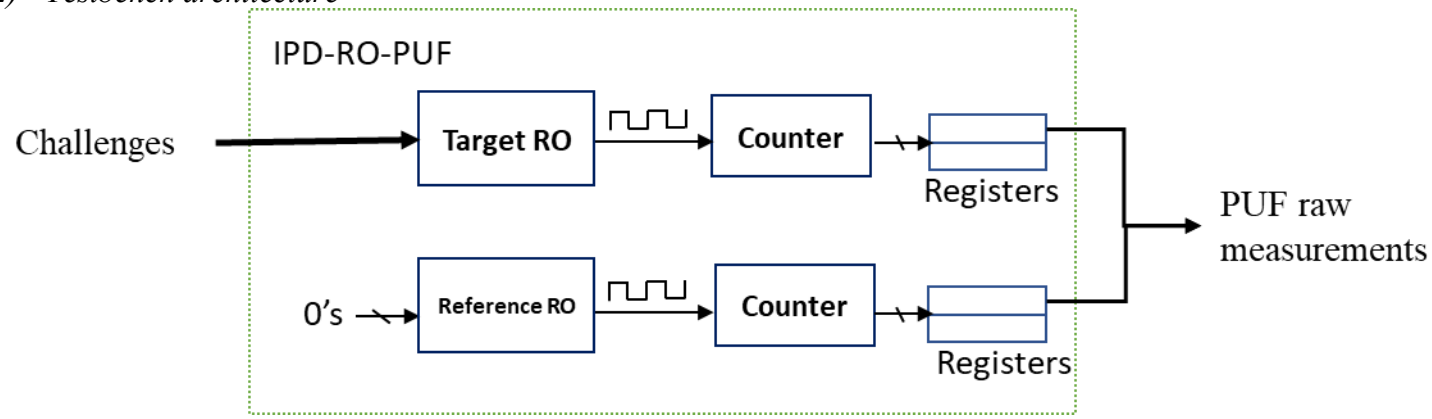


Figure 4-5 Block diagram of IPD-RO-PUF, tested in PUF test suite (Figure 2-13).

The proposed architecture is implemented on Xilinx Artix-7 FPGA, as shown in Figure 4-5, and it is tested in the test suite shown in Figure 2-13. XSDK application on the Zynq processor runs the testbench. Tested challenge bit string programs the target RO. The ROs are turned on for a pre-determined acquisition time measured by the number of system clock cycles. The counters count the number of oscillation cycles of both target RO and reference RO. Register storing the

readings from two counters, and then pass to BRAMs and the Zynq processor.

3) The modified 2-pass scheme

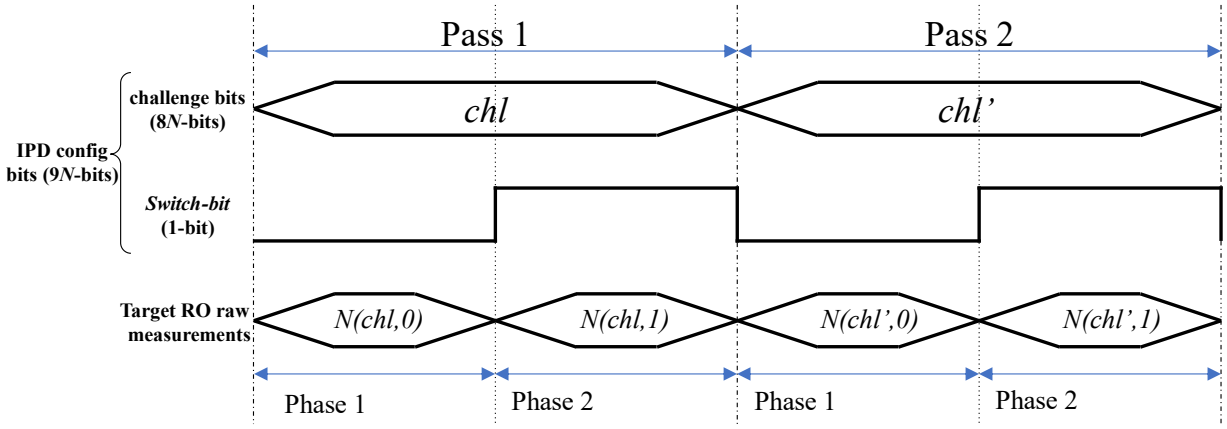


Figure 4-6 The proposed 2-phase 2-pass operation. Each pass involves two phases in which the same challenge configures the IPD-RO.

In the modified 2-pass scheme, the dedicated *Switch-bit* is set 0 and 1 for two phases, respectively. The pattern in each LUT5 could mitigate each other. Before each pass, the challenge bit string is sent from BRAM to ROs. Control circuits set *Switch-bit* to 0 for the 1st phase. At this point, RO paths inside LUTs are determined. Then, the control signal on the NAND gate is set to 1 to start the oscillations. Once the clock counter reaches the pre-set target value, the counters stop counting RO oscillations, and then control circuits stop the oscillation. At this time, the 1st phase of the 1st pass is done. The numbers of oscillations cycles of target and reference RO are read from counters. While the challenge bit string is the same, the same operations repeat for the 2nd phase after the *Switch-bit* is set to 1. After two phases are done, the calibration for the 1st pass corresponding to the 1st challenge bit string chl_1 is calculated by the following equations,

$$calibrate(chl_1) = N_{target}(chl_1, 0) - N_{target}(chl_1, 1) \quad (4\text{-}9)$$

$$calibrate_{ref}^{additive}(chl_1) = \left(N_{target}(chl_1, 0) - N_{ref}\right) - \left(N_{target}(chl_1, 1) - N_{ref}{}'\right) \quad (4\text{-}10)$$

$$calibrate_{ref}^{ratio}(chl_1) = \frac{N_{target}(chl_1, 0)}{N_{ref}} - \frac{N_{target}(chl_1, 1)}{N_{ref}{}'} \quad (4\text{-}11)$$

In the above equations, (4-9) is the scenario where the reference RO is not used, (4-10) is for that the reference RO is considered additive relative to the target RO, and (4-11) for the multiplicative comparison. This chapter will focus on the randomness of IPD-RO-PUF, and (4-9) is used unless otherwise clarified. $N_{target}(chl_1,0)$ is the number of oscillation cycles in target RO configured with challenge bit string chl_1 and *Switch-bit* 0. N_{ref} is the number of oscillations cycles recorded from the reference RO running simultaneously with the target RO. N_{ref} and N_{ref}' reflect the difference caused by the system clock, voltage and temperature variation between the time when $N_{target}(chl_1,0)$ and $N_{target}(chl_2,1)$ are measured.

At this point, the calibrated number of oscillations corresponding to challenge bit string chl_1 is recorded. The above-mentioned process is repeated with another challenge bit string chl_2 for the 2nd pass. The second calibration number is acquired as $calibrate(chl_2)$.

Finally, $calibrate(chl_1)$ and $calibrate(chl_2)$ are compared to generate *diff(chl)*, which is the final calibrated result for challenge *chl*. *diff(chl)* determines 1-bit PUF response bit r. They are described as,

$$diff(chl) = calibrate(chl_1) - calibrate(chl_2) \quad (4\text{-}12)$$

$$r = sign(diff) \quad (4\text{-}13)$$

In the rest of the thesis, *diff(chl)* denotes the calibration giving 1-bit PUF response bit with challenge *chl*, and *chl* is (chl_1, chl_2). $DIFF$ denotes the one giving multiple PUF response bits with multiple challenges. *r* denotes the 1-bit PUF response bit based on *diff(chl)* and *R* for multiple PUF response bits based on *DIFF*.

C. Biases mitigation in IPD-RO-PUF

1) Bias between two LUT5 mitigation

The huge bias between the two LUT5s in one LUT6 is corrected by the intertwined structure. To verify the successful mitigation of systematic biases, we experimented with IPD-RO in the same manner as we did in Chapter 3. Figure 4-7(a) shows the boxplots for the 32 paths configured by 4-bit challenge bit string chl_1 and *Switch-bit*. The red quartiles correspond to the paths when *Switch-bit* equals 0. and In Figure 4-7(a), contrast to Figure 3-14, there is no big gap between the red and blue quartiles. Implementation with mixture of the top and bottom LUT5s neutralize the biases between the two LUT5. Furthermore, the pattern within $d_{chl,0}$ and $d_{chl,1}$ are about the same. When *chl* keeps the same, the quartiles of $d_{chl,0}$ and $d_{chl,1}$ are at a similar level. So, the bias in LUT5 can be further eliminated by taking their difference, which is done by (4-9).

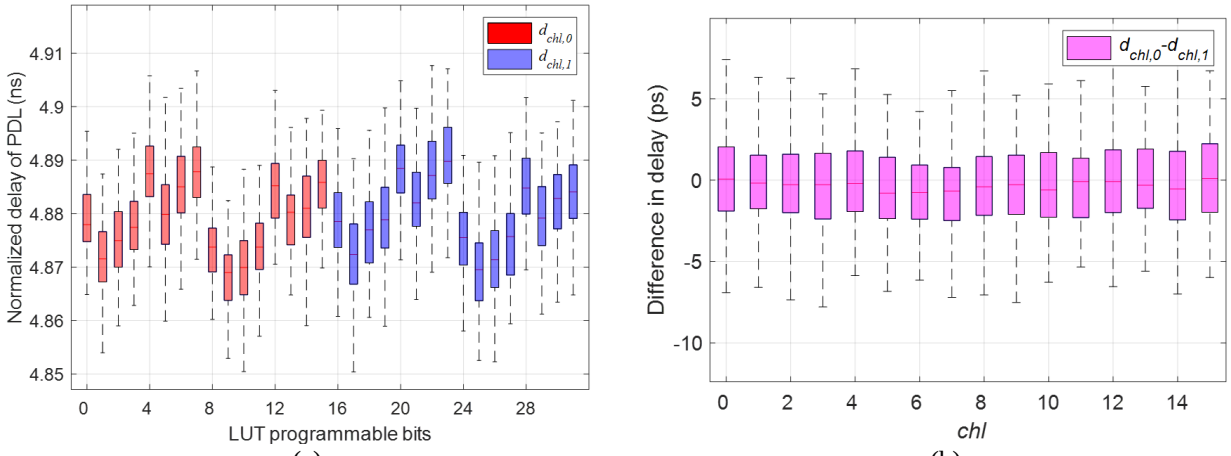


Figure 4-7 RO configured with 4-bit challenge string. (a) distribution of $N_{target}(chl,0)$ and $N_{target}(chl,1)$; (b) distribution of $N_{target}(chl,0)$-$N_{target}(chl,1)$.

2) The pattern in LUT5 mitigation

In Figure 4-7(b), it can be found that the distribution of quartiles for $d_{chl,0}$ - $d_{chl,1}$ does not have any pattern. The biases in the internal structure of LUT5 are canceled. All the quartiles' mean are around 0. In (4-10) and (4-11), N_{ref} is involved in the calculation of the calibration of one pass, *calibrate*(chl_1). N_{ref} just helps to improve the stability to the timing difference. It has very minor

effect on the distribution. *calibrate*(chl_1) has very similar distribution as $N_{target}(chl_1,0)$-$N_{target}(chl_1,1)$. Therefore, with the help of 2-phase operation, the two competing passes' calibration are free of any systematic biases.

D. IPD-RO-PUF Characterization

1) Baseline: Investigative PDL-RO-PUF

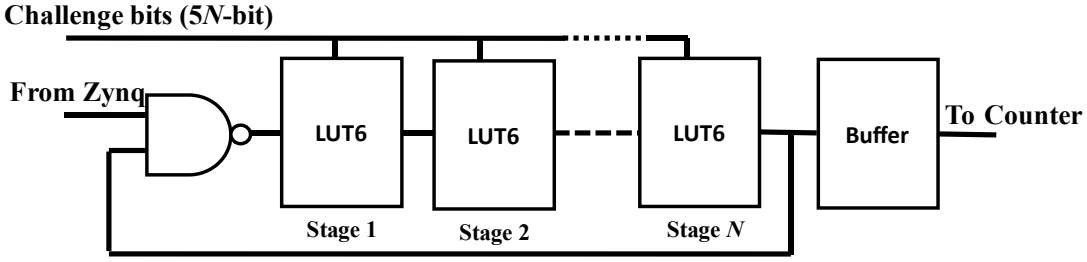


Figure 4-8 Structure of investigation PUF. N is an even number.

Previous 2-pass PUFs lack data on PUF characterization. Implementation was not found in [18][60], and [59] focused on the side-channel analysis of the LPUF. Therefore, no other people have characterized a 2-pass PUF before. To obtain a baseline for the 2-pass PUF, we implemented an investigative PDL-RO-PUF [18] on Xilinx Artix-7. The investigative PDL-RO-PUF can also help to quantify the impacts of systematic biases. Figure 4-8 shows the structure of the investigation PUF. *N* LUT6s plus a NAND gate make up each RO, and the programmable bits of each LUT6 are 5-bit. In total, 5*N* challenge bits configure the RO. For the demonstration in this section, *N*=8.

The sequential sampling method presented in [18] is used to generate the 1-bit PUF response. Two measurements are performed on the PDL-RO configured by a pair of randomly chosen challenges. These two measurements are compared to produce 1-bit outcome as the PUF response. In the experiments, 4000 PUF responses were collected from each of 32 ROs. [18] discussed how the competing pairs should be selected to achieve the best results. Their approach filtered CRPs by delay threshold and disparity threshold, which would discard many challenges. To understand how biases affect the PUF, we keep all CRPs for analysis.

2) Experiments summary

We carried out experiments on two Zedboards. To demonstrate the improvement by the novel IPD structure and the modified 2-pass scheme, we experimented with different FPGA circuits and different PUF responses extraction schemes. Table II summarizes the highlights of all the tested scenarios.

Table II Summary for the experiments for IRO PUF and VRO PUF.

<table>
<tr><td colspan="2">Experimented PUFs</td><td>Ideal PUF</td></tr>
<tr><td>PDL-RO
5-bit configurable bits on each LUT</td><td>IPD-RO
4-bit configurable bits on each LUT</td><td rowspan="3">•Matlab random number generator.
•Size of data is the same as experimental data.</td></tr>
<tr><td>Traditional 2-pass scheme</td><td>a) Traditional 2-pass scheme;
b) LPUF 2-pass scheme;
c) Modified 2-pass scheme</td></tr>
<tr><td colspan="2">•2 Zedboards
•32 ROs on each board
•Challenges selected randomly.
•20 response samples for each challenge</td></tr>
</table>

While many setups and configurations are the same, a major difference between the two FPGA circuits is the number of configurable bits on each LUT. For PDL-RO, each of the LUT6 can fully occupy all five inputs to challenges, while IPD-RO can occupy four inputs to challenges and the MSB of inputs to *Switch-bit*. Since the modified 2-pass scheme is relatively complicated, two traditional 2-pass schemes are also tested. The comparison will show why only the modified 2-pass scheme can mitigate most of the systematic biases in Artix-7 FPGA LUT structures.

3) A close look at bit-aliasing

First, one can intuitively view the raw data of the PUF responses. Figure 4-9 shows the PUF binary responses of investigative PDL-RO-PUF and IPD-RO-PUF. Each graph shows binary responses to 100 unique challenges from 32 PUFs in each of two devices. A conclusion can be quickly made that different investigative PDL-RO-PUFs have many responses in common. However, responses of IPD-RO-PUF are more random.

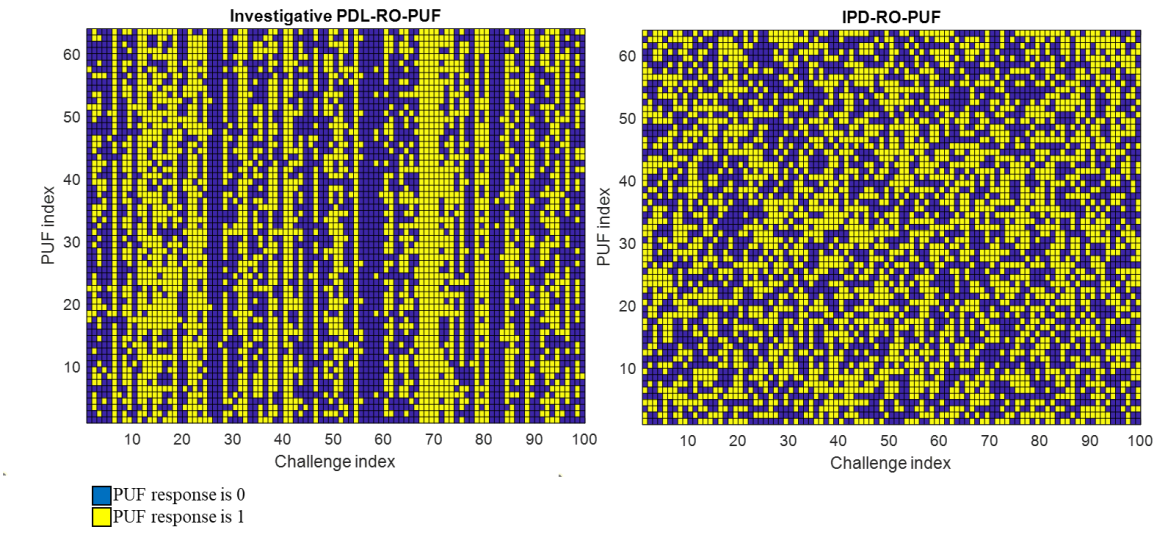


Figure 4-9 Raw data of the PUF binary response

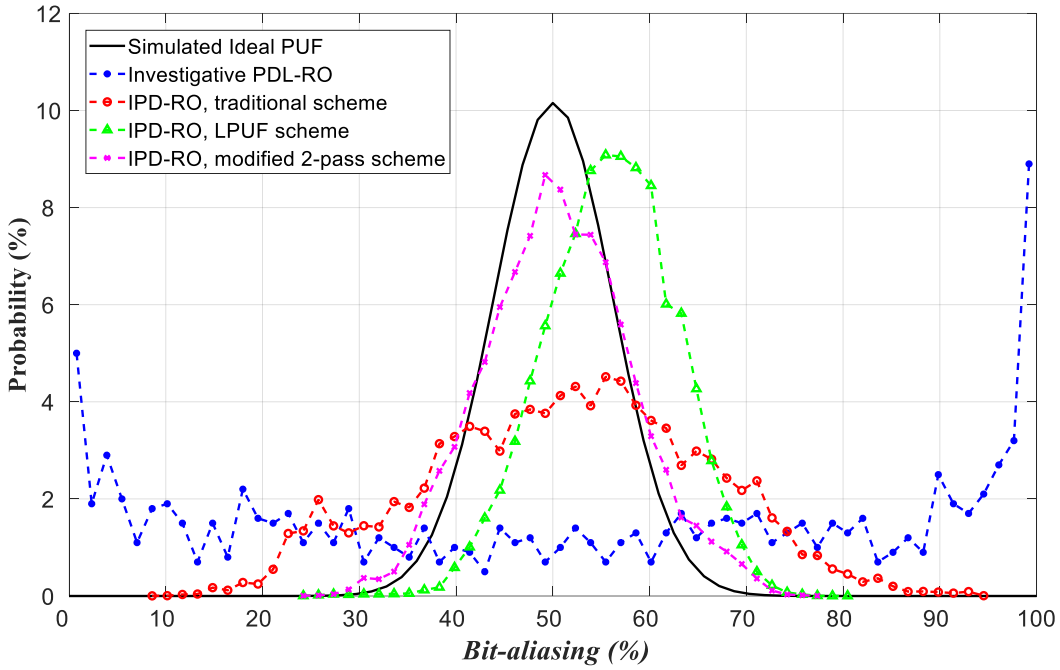
a) Overall and intra-device bit-aliasing

A more analytical but still straightforward way is to look at bit-aliasing and its distribution. First, we will examine the overall bit-aliasing, which is calculated by (2-2). The distribution of the resulted overall and intra-device bit-aliasing, which are quite similar, are shown in Figure 4-10.

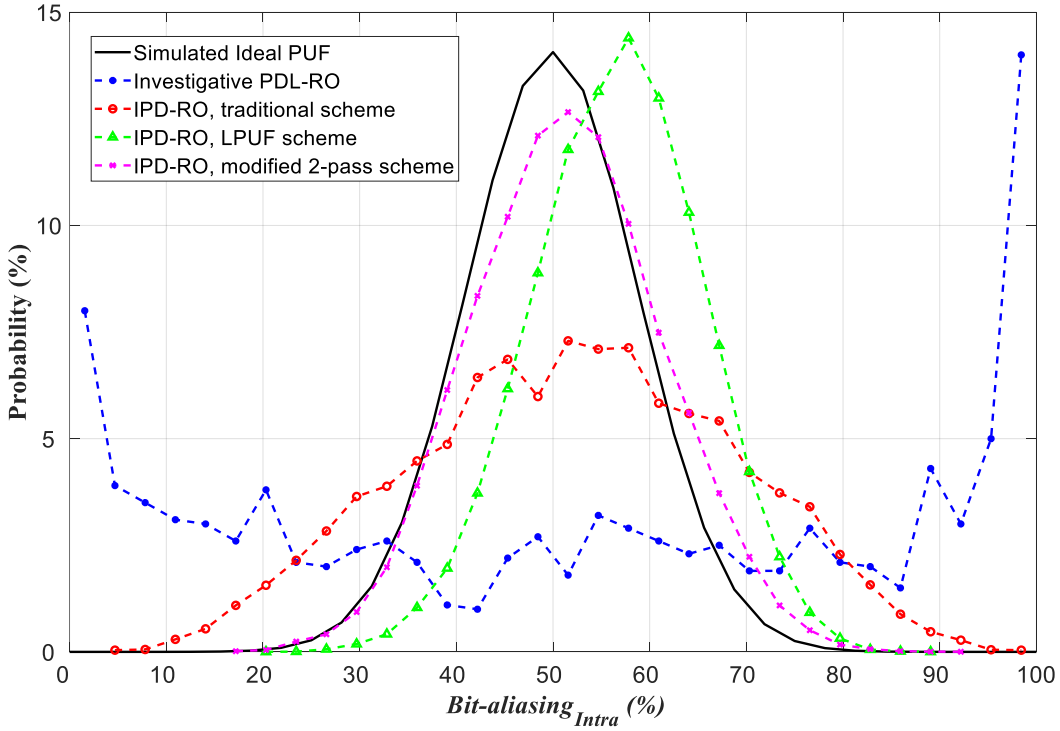
First, the investigative PDL-RO-PUF shows very poor bit-aliasing. Large amounts of CRPs are the same throughout all the PUFs. Especially, about 5% and 7% of total PUF responses are all 0's and 1's.

The distribution is much better when applying the traditional scheme on IPD-RO, thanks to mitigating the strong bias between LUT5s. Using the LPUF scheme shows a skewed distribution whose shape is much closer to the ideal distribution. The skew comes from the remaining unbalanced bias.

The proposed IPD-RO with the modified 2-pass scheme shows the best distribution. Its center locates at 50%, and its shape is very close to the ideal one.



(a)



(b)

Figure 4-10 (a) Overall bit-aliasing distribution; (b) Intra-device bit-aliasing distribution. 100% and 0% mean all 32 ROs in the same device yield the same PUF response, while 50% means PUF responses to the same challenge are random on these ROs.

b) Inter-device bit-aliasing

Figure 4-11 shows the inter-device bit aliasing calculated by (2-3). For only two devices, the

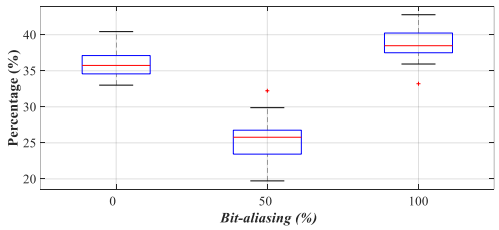
outcomes of inter-device bit-aliasing can only be 0%, 50%, and 100% (x-axis in Figure 4-11). Ideally, the percentage for each case should be 25%, 50%, and 25%, respectively.

For the investigative PDL-RO-PUF, the two tested devices have very poor bit-aliasing. As shown in Figure 4-11(a), the numbers of investigative PDL-RO-PUF are about 36%, 25.5%, and 38.5%.

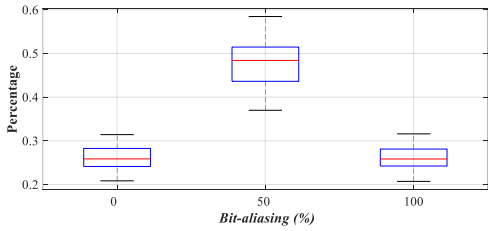
Figure 4-11(b) shows that the inter-bit-aliasing is substantially improved from the one of the investigative PDL-RO-PUF. Using the intertwined pair of LUT6s does mitigate the huge bias between two LUT5s in one LUT6.

In Figure 4-11(c), the probability that ROs in two different FPGA devices but the same location both give 1 is about 28% but only about 19% for giving 0s. This indictes the uneven result in uniformity.

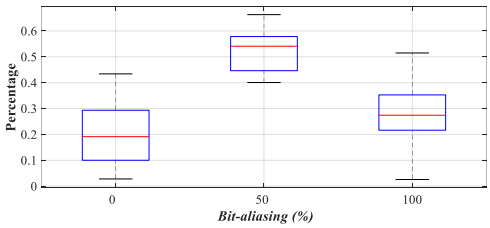
As for the proposed IPD-RO-PUF, as shown in Fig. 14(b), there is a 49.27% chance that ROs at the same location of different devices yield different PUF responses. And the opportunity is 25.20% and 25.52% for both 0's and both 1's, respectively. These numbers are very close to the ideal values.



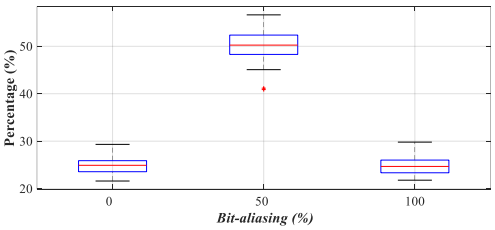
(a) Investigative PDL-RO-PUF;



(b) IPD-RO with traditional 2-pass scheme;



(c)) IPD-RO with LPUF 2-pass scheme;



(d) IPD-RO-PUF

Figure 4-11 Inter-device bit-aliasing the disussed structures and schemes

4) Quantitatively characterization

Two variants of IPD-ROs are implemented (N=2 and N=4). Each variant is experimented with 32 samples of RO (k=32) on each of two instances of board. All measurements were collected after RO running for 15.729ms. The selection of the RO running time was based on reliability and feasibility considerations. A short RO running time would deteriorate reliability because discrete counter counts do not distinguish the difference in the compared calibrations. The RO running time cannot be too long, as it is not feasible for practical applications. An error-correcting code (ECC) can be used to reduce either RO running time or error probability [23]. The experiments were conducted at room temperature and standard voltage. Experiments are conducted at room temperature and normal voltage. The discussion about the ambient conditions' effects will be presented in CHAPTER 5.

The comparison of the characterization of IPD-RO-PUF and other designs in our scope is shown in Table III.

First of all, since all the found biases are not mitigated in investigative PDL-RO-PUF, its uniqueness are severely affected. Inter-device uniqueness is only 26.62%, and intra-device uniqueness is only 26.35%.

As for the proposed IPD-RO-PUF, all the metrics show values very close to ideal values.

Table III IPD-RO-PUF experiment results, with randomly selected challenges. The test for IPD-RO-PUF (M=2) has covered all the challenges. M=4 is not fully covered, but it gives similar results.

	M	**No. of LUTs**	**Challenge length**	**Number of tested challenges**	**Uniformity**	**Overall Uniqueness**	**Inter-device Uniqueness**	**Intra-device Uniqueness**	**Reliability (wo Ref. RO)**
Investigative PDL-RO-PUF (based on [18])	8	9	40	40k	50.64%		26.62%	26.35%	98.79%
IPD-RO-PUF	2	5	16	64k	49.82%	49.82%	49.88%	49.81%	97.83%
	4	9	32	80k	50.09%	49.61%	49.27%	49.77%	97.55%
IPD-RO, with traditional scheme	2	5	16	64k	50.44%	44.86%	45.16%	44.85%	98.47%
	4	9	32	80k	50.36%	46.47%	46.44%	46.53%	98.29%
IPD-RO, with LPUF scheme	2	5	16	128k	57.32%	48.90%	47.31%	48.81%	98.58%
	4	9	32	160k	55.09%	49.39%	47.20%	49.52%	98.47%

Due to the absence of the characterization of other 2-pass PUF, we compared the IPD-RO-PUF

with other PDL-based RO PUFs, as shown in Table IV. We found that the IPD-RO-PUF is more advantageous in terms of both uniformity and uniqueness. Traditional PDL-based RO PUFs that generate a PUF response based on ROs at different locations ([2][15][30]) are affected by the interconnect mismatch. The canonical metrics are insufficient to show PUF response biases, a limitation shown by the correlation result [4]. As shown in Table VII, improvements in uniformity and uniqueness are significant. Uniformity and uniqueness in our proposed work are closer to the ideal values compared to other designs. In addition, the reliability of this work is very close to the highest level.

IPD-RO-PUF is very compact compared to previous RO PUFs. As shown in Table IV, the hardware overhead of IPD-RO-PUF is smaller than previous RO PUFs, only using five LUTs for 16-bit PUF. Existing RO PUFs generate PUF responses based on multiple ROs. The number of ROs determines the number of PUF responses. Therefore, existing RO PUFs require many LUTs to give a large number of PUF responses. A reason for this drawback is that existing RO PUFs did not fully utilize the resource inside LUTs. In [30] and [15], only one and two inputs of each LUT are programmable, respectively. In [2], although all three available inputs in each LUT are programmable, the same three bits are applied to all different LUTs. Taking advantage PDLs, IPD-RO-PUF can generate 32768 PUF responses from only five LUT6s. The comparison of hardware overhead between IPD-RO-PUF and PDL-RO-PUF is shown in Table I. The MSB of the 5-bit inputs of each LUT in IPD-RO-PUF is reserved for Switch-bit. Therefore, the length of the challenge is smaller in IPD-RO-PUF than PDL-RO-PUF if the same amount of LUTs are used. However, as explained, PDL-RO-PUF discards correlated CRPs. It will greatly reduce the number of effective response bits. Also, this operation may require additional hardware resources.

Table IV Compare IPD-RO-PUF with other PDL-based PUFs

	Uniformity	Uniqueness	Reliability	No. of LUTs	Bits of LUT input/programmable inputs	No. of PUF responses
[2] Habib	50.75%	47.67%	98.1%	520	4/3	1032
[15] Zhou	48.96%	47.57%	100%	128	6/2	256
[30] Anandakumar	50.61%	47.13%	99.16%	128	6/1	256
IPD-RO-PUF (*M*=2)	49.99%	49.18%	97.94%	5	6/4	32768

5) Challenge correlation

As pointed out in [18][25], in addition to canonical metrics such as uniqueness, the correlation in CRPs must also be investigated to evaluate the guessing complexity of the PUF. To determine the PUF correlation better, a metric that evaluates the correlation between two PUF responses was proposed in [24] and applied to PDL-RO-PUF in [18]. With different challenges, one would have two 1-bit PUF responses $r_{i,j}$ and $r_{i,k}$ from PUF i. The correlation $cor_{j,k}^{i}$ is defined as follows:

$$cor_{j,k}^{i} = \begin{cases} 1, & if\ r_{i,j} = r_{i,k} \\ -1, & otherwise \end{cases} \tag{4-14}$$

Then, the correlation is summed up over all t PUFs,

$$cor_{j,k} = \frac{1}{t}\sum_{i=1}^{t} cor_{j,k}^{i} \tag{4-15}$$

If $cor_{j,k}$ is zero overall PUFs, then the corresponding challenge pairs are uncorrelated.

We tested the investigative PDL-RO-PUF and IPD-RO-PUF (M=2) with random challenges to calculate the correlation. Due to our data processing capability limitation, we randomly selected 6000 PUF responses. Thus, 6000*5999/2=17,997,000 PUF response pairs were included in the calculation. The test results are shown in Figure 4-12. Without performing any CRP selection/exclusion, the correlation of the IPD-RO-PUF is very close to the distribution of the simulated ideal PUF. The IPD-RO-PUF has more than 9% uncorrelated challenge pairs, that is, $cor_{j,k}$ is zero. In comparison, the distribution of investigative PDL-RO-PUFs shows that only less than 4% of the challenge pairs are uncorrelated. Furthermore, investigative PDL-RO-PUFs have many highly correlated challenge pairs. [18] carefully selected VROs with some criteria, which

discarded a large portion of CRPs. By keeping only 113 out of 8188 PUF responses (i.e., more than 95% of CRPs were discarded), [18] made approximately 8% of the challenge pairs uncorrelated, which is still lower than our result.

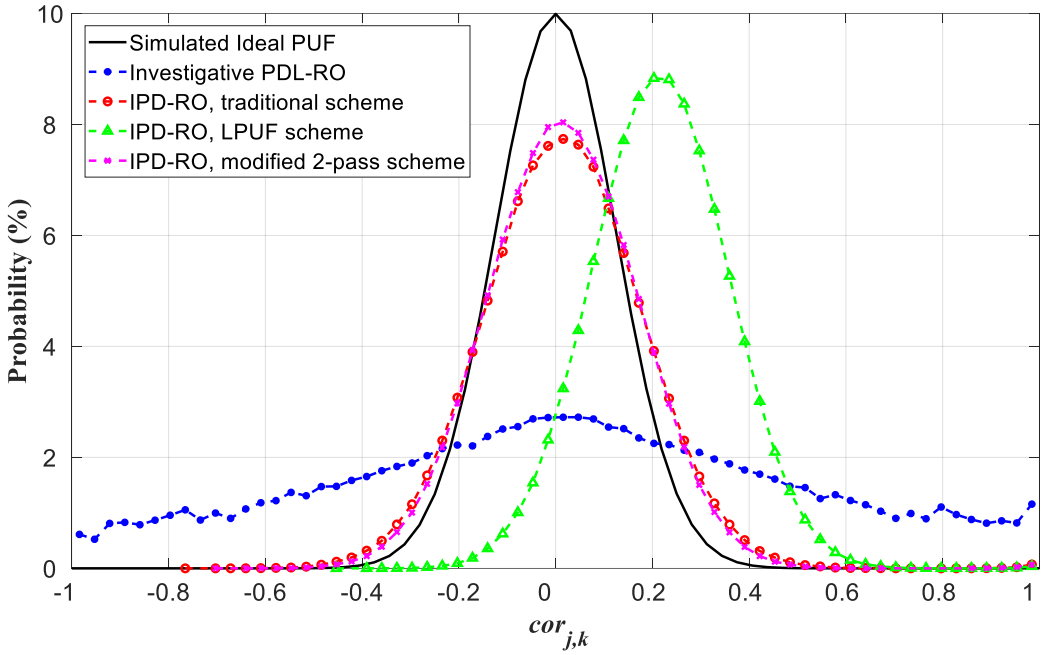


Figure 4-12 $cor_{j,k}$ is plotted for IRO PUF (w=5) and simulated PUF bits based on random number generator.

CHAPTER 5. ELEVATE RELIABILITY OF IPD-RO-PUF.

In the previous chapters, we have characterized several a few PDL-based RO PUFs and schemes and compared them with some earlier works. Only the work reported in [15] could achieve 100% reliability. However, a unique extraction is included in their process. [17] presented a framework for how reliability in PUF could be enhanced. It also demonstrated a mechanism for filtering unreliable CRPs to improve PUF reliability. We will discuss how reliability could be improved in IPD-RO-PUF with this framework.

Schaub introduced BER (bit error rate), and SNR (signal-noise ratio), defined as below [17]:

$$BER = P\big(sign(\delta_C + Z) \neq sign(\delta_C)\big) = Q\left(\frac{|\delta_C|}{\sigma}\right) \tag{5-1}$$

$$SNR = \frac{E[\Delta_C^2]}{E[Z^2]} = \frac{\Sigma^2}{\sigma^2} \tag{5-2}$$

In this chapter, SNR will be used as an additional metric for PUF's reliability and the canonical reliability metric.

A. Mitigation against operation condition variation and noise reduction

First, we will show how the reference RO helps the IPD-RO-PUF's reliability. The purpose of using reference RO is to mitigate the variation caused by system clock and environmental variables, i.e., voltage, temperature, etc. We briefly introduced two methods using the reference RO in CHAPTER 4, and we rewrite them below.

$$calibrate_{ref}^{additive}(chl_1) = \big(N_{target}(chl_1, 0) - N_{ref}\big) - \big(N_{target}(chl_1, 1) - N_{ref}{}'\big) \tag{4-10}$$

$$calibrate_{ref}^{ratio}(chl_1) = \frac{N_{target}(chl_1, 0)}{N_{ref}} - \frac{N_{target}(chl_1, 1)}{N_{ref}{}'} \tag{4-11}$$

Many literatures have considered temperature and voltage as important environmental factors affecting RO PUF [1]. However, what they controlled was ambient temperature. FPGA regular operation inevitably increases the on-chip temperature. Thus, measuring ambient temperature does

not indicate how the RO PUF behaves differently when the temperature is altered. In the Xilinx FPGA, a built-in XADC can be utilized for measuring the on-chip temperature and supply voltage [30]. The XADC includes a dual 12-bit, 1 Mega sample per second (MSPS) ADC and on-chip sensors. With the help of XADC, we can monitor the temperature and voltage change when RO is active. Tests are carried out with IPD-RO-PUF and Figure 5-2 summarizes the situation for one tested PDL in IPD-RO-PUF. We pre-cool down the FGPA boards, and then the board is programmed and activated for tests. Therefore, we recorded the activities when the board got from relatively cold to warm.

First, RO frequency is positively correlated to temperature. Due to its operation, the temperature of the FPGA chip is getting higher, and RO frequency also gets faster. As for voltage, the monitored voltage does not change that much over time, and we cannot detect a clear correlation between voltage and RO frequency.

Figure 5-1(c-f) shows the reference RO's effectiveness to reduce the impact of operation condition variations. Here, for simplicity, $N_{target}(chl_1,0)$ and $N_{target}(chl_1,1)$ are referred as $N_{target}(chl_1,\sim)$. Within 80 s, 20480 measurements $N_{target}(chl_1,\sim)$ from a target RO and 20480 sample measurements of N_{ref} from a reference RO are taken consecutively with a fixed challenge bit string given to target RO. It is clear that $N_{target}(chl_1,\sim)$ fluctuates from time to time. That indicates $N_{target}(chl_1,\sim)$ is sensitive to operation condition variations. N_{ref} has similar fluctuation patterns. Therefore, N_{ref} can be used to calibrate $N_{target}(chl_1,\sim)$ against the operation condition variations. As shown in Figure 5-1(e), compared to $N_{target}(chl_1,\sim)$, the trend of $N_{target}(chl_1,\sim)$-$Nref$ is flattened.

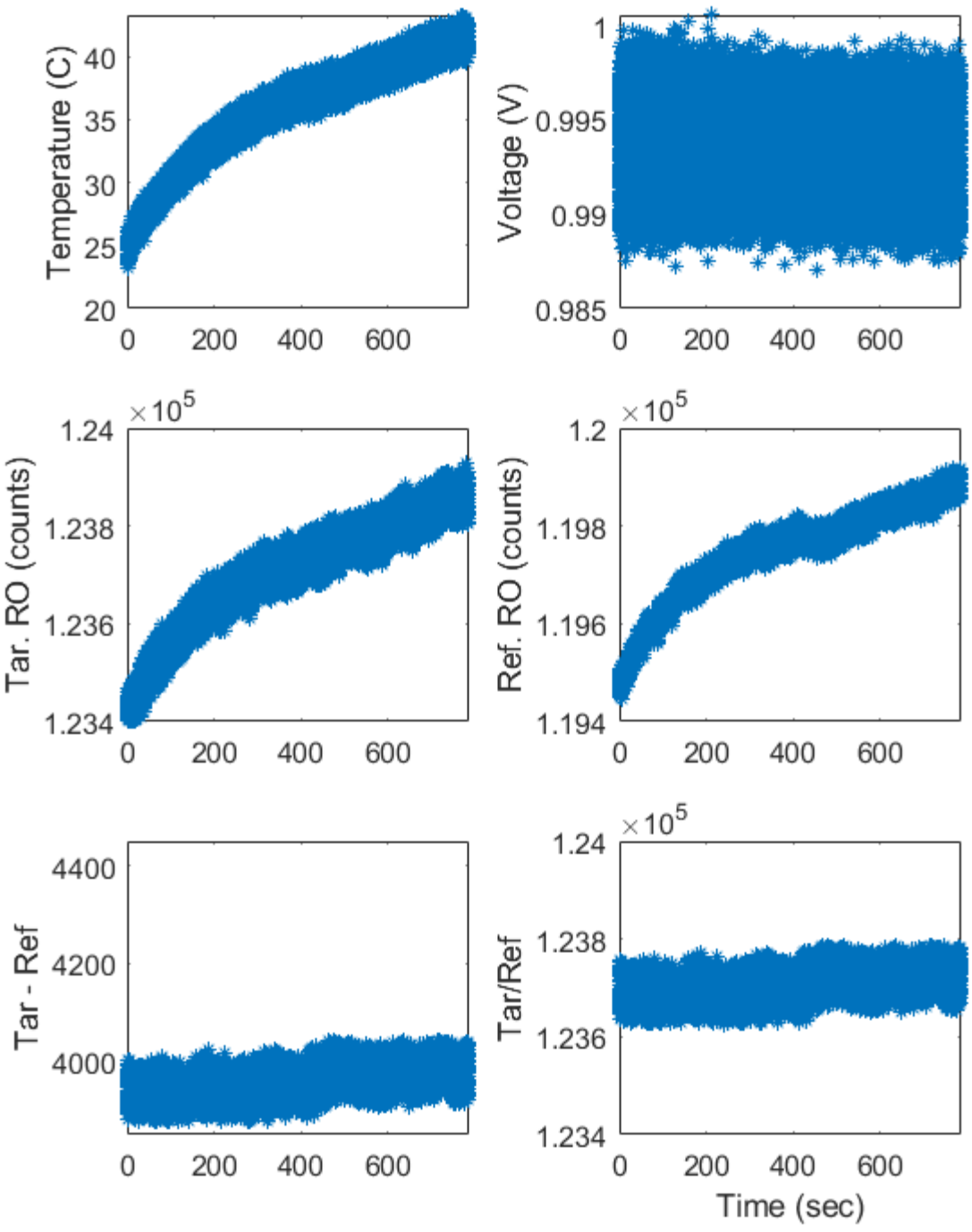


Figure 5-1 Environmental factors (temperature, voltage) and RO frequencies

While $N_{target}(chl_1,0)$ and $N_{target}(chl_1,1)$ have a similar trend, subtracting them does not necessarily cancel the variation within the trend. $N_{target}(chl_1,0)$ and $N_{target}(chl_1,1)$ are taken at different time. If there is a large time interval between their measurements, the system clock, and environmental variables, maybe be quite different. Only the reference RO that runs simultaneously with the target RO suffers common interference.

In , besides the changing trend, it is noticeable that both $N_{target}(chl_1,\sim)$ and N_{ref} have some spread, which comes from FPGA noise. Figure 5-1(c-f) are on the same scale. One can observe that the spread of $N_{target}(chl_1,\sim)$-N_{ref} is smaller than the one of $N_{target}(chl_1,\sim)$ and N_{ref}. This noise reduction can also be seen in *diff(chl)*. To show the improvement by the reference RO, Figure 5-2 shows the σ, Σ, and SNR in (7-1). The reference RO helps push the σ down, and the Σ is not changed much. Therefore, we could see a significant increase in the SNR, from 300 to almost 400 on average. Because most CRPs are intrinsically reliable and only those on the boundary are improved, the reliability is slightly increased.

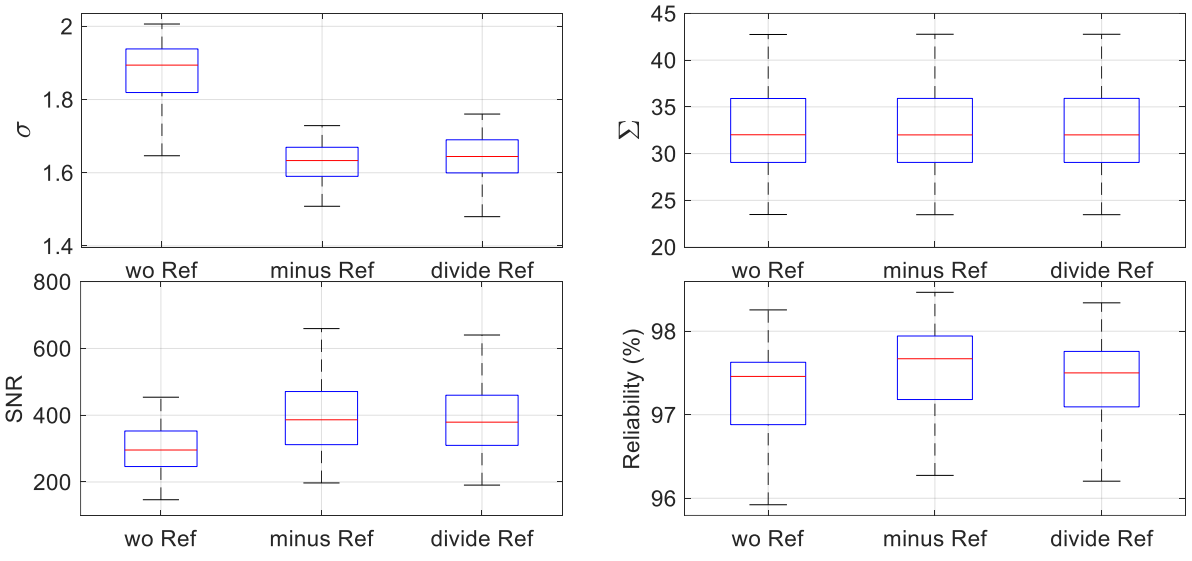


Figure 5-2 $\sigma_{specific}$ for the scenario w/wo reference RO. The smaller the σ is, the more stable the response is.

B. Improve Performance By Increasing RO Running Time.

The raw measurements of ROs follow the Gaussian distribution. In IPD-RO-PUF, *calibrate* and *diff* are based on simple addition and subtraction of the raw measurements. If the involved raw measurements are independent, *calibrate* and *diff* also are Gaussian distribution. Thus, we write *diff* based on RO running time T can be written as $diff_T \sim N(u_T, \sigma_T^2)$. Doubling the running time can be assumed to be the same as adding two $diff_T$ up, thus giving $diff_{2T} \sim N(u_{2T}, \sigma_{2T}^2)$, where u_{2T} equals $2u_T$ and σ_{2T} equals $\sqrt{2}\sigma_T$. In Figure 5-3, the probability that $diff_{2T}$ gives a flipped r is smaller than $diff_T$. Therefore, prolonging RO running time is supposed to improve the reliability of the CRPs that are near decision boundary. For the CRPs having larger distance to the boundary, prolonging could also further improve the reliability.

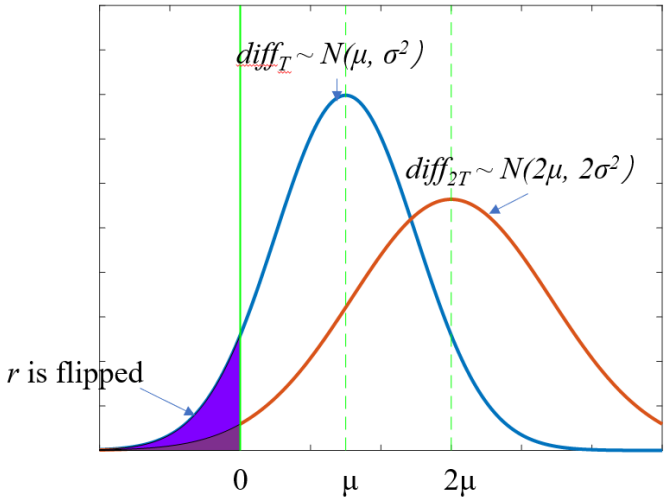


Figure 5-3 Prolonging the running time is expected to lower the probability that r flips.

To study how RO running time affects PUF's reliability in the device, we conduct experiments for various running times on Zedboards. Figure 5-4 compares σ, Σ, SNR, and reliability. Both SNR and reliability achieve higher values with a longer RO running time. However, we can see the marginal effects on the reliability.

For the comparison between T and $4T$. The σ changes from 1.894 to 5.514 and then 18.403. Their ratio are 2.911 and 3.33, respectively. These values are not close to the theoretical value 2.

Apparently, the assumption that raw measurements of ROs are independent does not stand. Besides, with this data, whether the noise in measurement can be simply treated as Gaussian is questionable.

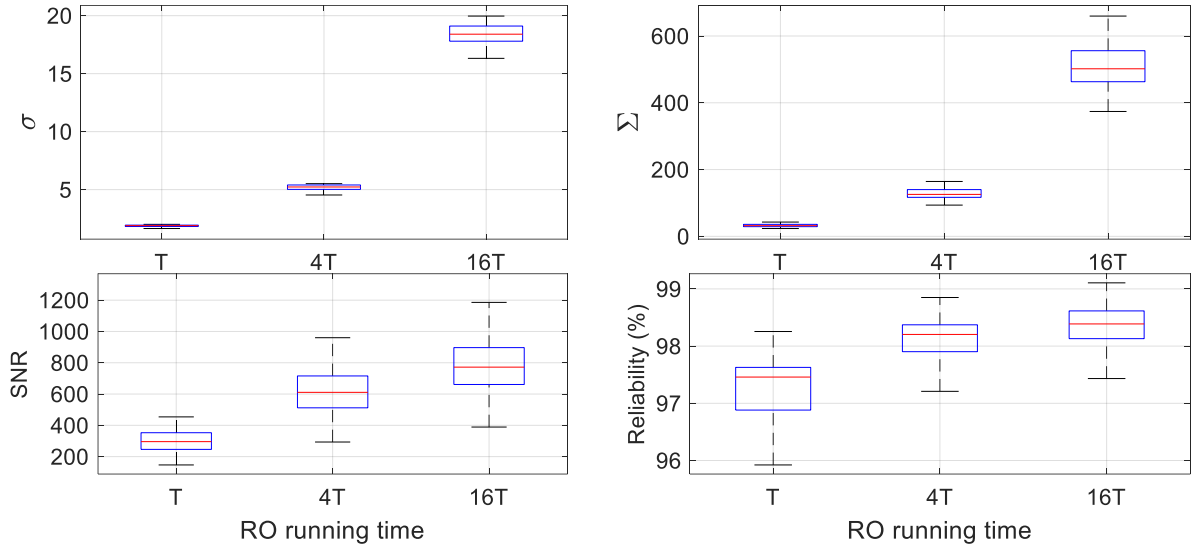


Figure 5-4 Compare the σ, Σ, SNR, and reliability with the change of RO running time.

While increasing RO running does improve the SNR and reliability, the increase in the cost of time is not negligible. When implemented in applications, the IPD-RO-PUF may need to give multiple PUF bits, requiring one RO to run many times. If *16T* is chosen as the RO running time to give 32 PUF bits, one RO may ask for at least 32*15.729ms=503.328 ms, which may not be acceptable for some timing-critical applications. An alternate way is to have multiple IPD-ROs implemented and running in FPGA simultaneously. Therefore, at a time, numerous PUF bits could be generated at a time.

C. Filtering unreliable CRPs based on margin

It is common to use helper data to extract CRPs that satisfy users' demands [15]. In [17], a mechanism filtering unreliable CRPs was presented. For IPD-RO-PUF, as the calibration *diff* and *DIFF* are collected, post-processing can be done in this way to improve the reliability further.

$D_{specific}$ is used to denote the distribution of *diff* and $D_{general}$ is for the distribution of *DIFF*. Experimental data in Figure 5-5 demonstrates that $D_{general}$ and $D_{specific}$ are both close to normal distribution with different standard deviations. Therefore,

$$D_{general} \sim N(\mu_{general}, \Sigma) \quad (5\text{-}3)$$
$$D_{specific} \sim N(\mu_{specific}, \sigma) \quad (5\text{-}4)$$

where $\mu_{general} = \overline{DIFF}$, $\mu_{specific} = \overline{diff}$.

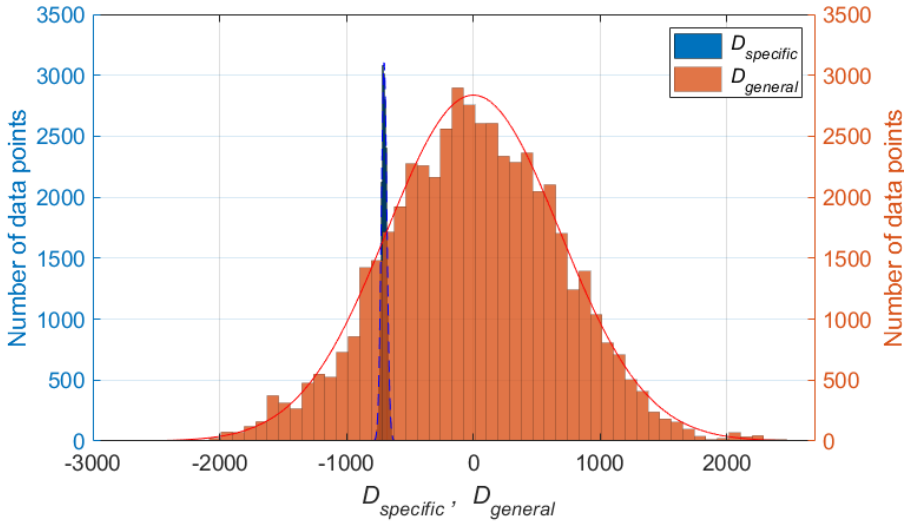
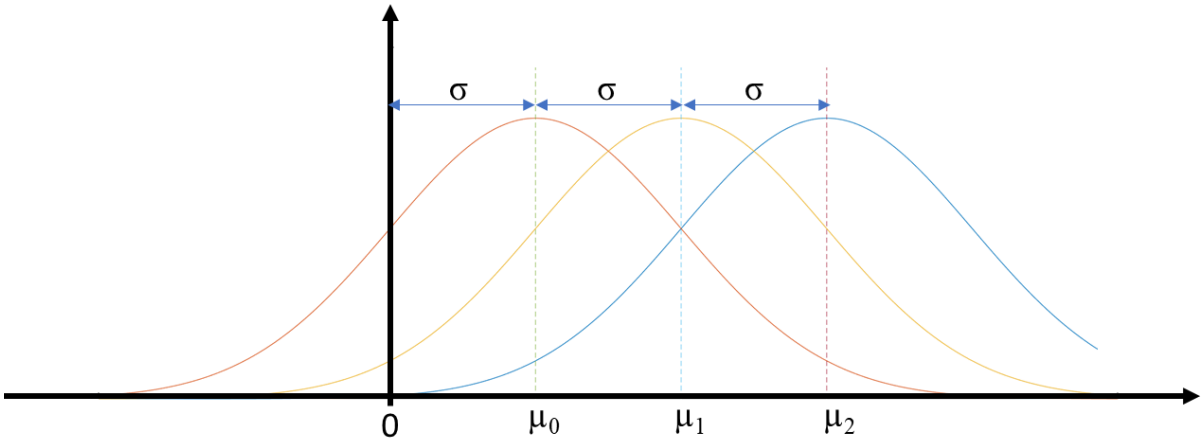


Figure 5-5 Experimental data, with $D_{general}$ (blue) and $D_{specific}$ (black) distribution.

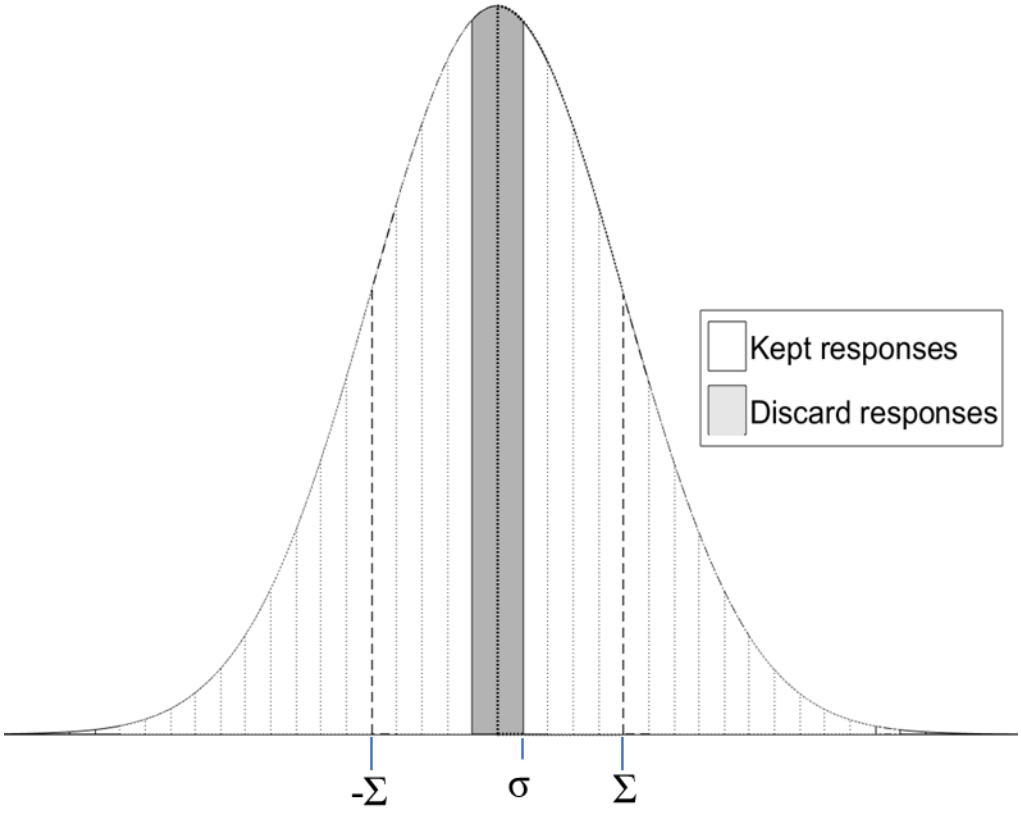
$D_{specific}$ can undesirably fall close to 0. If $|\mu_{specific}|$ is not large enough, the corresponding PUF response may flip when $D_{specific}$ drifts across the decision boundary. Figure 5-6(a) shows some scenarios for such a situation. Here, μ_0, μ_1, μ_2 are all greater than 0, so these responses are more likely greater than zero. However, there is still some probability that they are less than zero. That means PUF response *r* can be sometimes 0 and sometimes 1. The area of this normal distribution on the left side of y axis shows the probability that the response is unreliable. Using error function, one can estimate that, for $\mu_0=\sigma$, there is about 15.8% possibility that $D_{specific} < 0$. This possibility is reduced to about 2.2% for $\mu_1=2\sigma$, and only 0.1% for 3σ. Any $D_{specific}$ that on the right side of μ_2 can achieve higher reliability as there is only less than 0.1% chance that the response is not

consistent.

Data post-processing selects CRPs with higher reliability to avoid unreliable responses. With the knowledge of Σ and σ, we could discard some CRPs whose $|\mu_{specific}|$ is within the range of M times of σ around 0. In Figure 5-6(b), the *DIFF* is in normal distribution. In the middle, the gray part, whose width is 2σ (W=1), shall be discarded, and the others are kept. In this way, it is less likely to pick the unreliable challenge bits. Essentially, the larger W is, the more reliable the responses are. There is a tradeoff between reliability and the number of usable CRPs. In Figure 5-6(b), when M=3, many more CRPs should be discarded. In Figure 5-6(b), Σ is only 5 times of σ, or say its SNR is 25.



(a)



(b)

Figure 5-6 Three scenarios for $D_{specific}$. (b) Among the general responses, responses around 0 with Mσ (M=1) are discard. The Ratio here is 5.

1) PUF performance after CRPs filtering

One concern with the CRP filtering is that it might harm the uniqueness of PUFs. As presented in CHAPTER 3, LUTs in Xilinx FPGA present strong biases pattern. After removing CRPs around

the boundary, the left CRPs may suffer stronger biasing. Removing the CRPs on the border may significantly deteriorate the uniqueness. Table V shows that the investigative PDL-RO-PUF has a deteriorated uniqueness. As for IPD-RO-PUF, its uniqueness is kept very well with any portion of CRPs being removed.

Table V Characterization results of investigative PDL-RO-PUF and IPD-RO-PUF, after CRPs filtering. .

	IPD-RO-PUF (20-bit)			**Investigative PDL-RO-PUF (20-bit)**		
W	**Uniformity**	**Uniqueness**	**Reliability**	**Uniformity**	**Uniqueness**	**Reliability**
0	49.8670%	49.4196%	98.5541%	50.4831%	25.1975%	99.1786%
1	49.8898%	49.4446%	98.9912%	49.7234%	18.2528%	99.3212%
2	50.0230%	49.3524%	99.7644%	49.3603%	12.2105%	99.7915%
3	49.9910%	49.5706%	99.9735%	49.2799%	7.5455%	99.9628%
4	50.0138%	50.2717%	99.9986%	49.2768%	5.0530%	99.9962%
5	50.0527%	48.3871%	99.9975%	49.3034%	3.3122%	99.9958%

2) Remaining CRPs after CRP Filtering

The probability that a normal distribution lies outside the range between $\mu - n\sigma$ and $\mu + n\sigma$ is given by the error function $erfc\left(\frac{n}{\sqrt{2}}\right)$. It is the base when determining whether a CRP should be kept or discarded. In Table VI, the percentage of usable CRPs estimated by using error function and the corresponding *SNR* are listed along with the percentage of usable CRPs measured in experiment. Without discarding any unit of $\sigma_{specific}$ in response, the percentage of usable challenge bits are all above 90%. When some units are discarded, the percentage of usable CRPs is lowered to 70%~85%.

In CHAPTER 5.B, it is presented that the reliability is increased along with the increase of running time. For all the cases, the estimated percentages are very close to the experimental ones, which means *SNR* can be used to accurately estimate how many CRPs satisfy the reliability goal. Discarding challenge bits is not a big obstacle for CRPs selection, while the reliability is essentially elevated.

As presented in Table VI, the *SNR* in the proposed IPD-RO-PUF can be up to about 625. In

contrast, the SNR in [17] are just between 180 and 250.

Table VI Number of usable CRPs estimated by error function and experiment.

Challenge length		**16**			**24**			**32**		
time	**M**	**SNR**	**Err. fn**	**Exp.**	***SNR***	**Err. fn**	**Exp.**	***SNR***	**Err. fn**	**Exp.**
T	1	364.05	95.82%	94.66%	319.69	95.54%	95.59%	94.87	95.19%	92.79%
	2		91.65%	89.64%		91.10%	92.20%		90.39%	87.69%
	3		87.50%	84.69%		86.68%	87.46%		85.63%	82.69%
4*T*	1	407.23	96.05%	94.99%	377.14	95.90%	96.20%	371.99	96.00%	96.07%
	2		92.10%	90.43%		91.80%	92.55%		92.00%	91.99%
	3		88.18%	85.31%		87.73%	88.57%		88.03%	87.99%
16*T*	1	421.89	96.12%	96.14%	406.43	96.04%	96.00%	418.61	96.20%	96.30%
	2		92.24%	92.30%		92.10%	92.06%		92.40%	92.55%
	3		88.39%	88.51%		88.17%	88.16%		88.62%	88.79%

CHAPTER 6. TERO-PUF BASED ON PDL AND IPD

The concept of TERO has been discussed in detail in [49][51]. The fundament of TERO is the two branches in the loop. Generally, two approaches have been used to implement the two branches. In [51], SR flip-flops were used as the two branches. In [49], the chained LUTs were used to form the branches. LUTs were configured as inverters. Each branch has an AND gate, whose two inputs are connected to the trigger signal. Once the trigger is enabled, the rising edge causes the signal to propagate in the loop through the inverters. Depending on how similar the two branches are, the oscillation persists for a different time. Therefore, TERO is a metastable structure. Ideally, if the delays of the two branches are identical, the oscillation would never stop. However, due to any minimal difference in the delay, the oscillation would stop after some time. Depending on how similar the two branches are, the persistent times of the oscillation are different on different TEROs. The electrical behavior of TERO is demonstrated in [51].

Typically, the TEROs are measured at some certain moment, so the capture is used as the source of entropy. Two approaches have been proposed to cope with the capture. The first approach is to use the final state when the TERO is measured [62]; The second one uses number of the oscillations that happened within the acquisition time [49][51].

A. TERO structures

1) Two branches configured by different programmable bits (DPDL-TERO)

To fully take advantage of the resources in LUT, all the LUTs in the TERO structure are configured by different challenges, as shown in Figure 6-1. In the two branches, two different segments of the challenges program the two braches, respectively. Each branch has M stages. Thus, the challenge for the TERO is $10M$-bits.

A potential issue of this structure is that the biases of the delay of the two branches may impact the PUF responses. Chapter 3 investigated the biases in delay of LUTs and presented strong biases

in the Xilinx Artix-7 LUT structure. For instance, a LUT6 configured by $(1xxxx)_2$ has a smaller delay than the ones configured by $(0xxxx)_2$. In Figure 6-1, if all the LUTs in the blue branch are configured by $(1xxxx)_2$ and all the LUTs in the red one are by $(0xxxx)_2$, the total delay in the blue branch is very likely smaller than the delay in the red one.

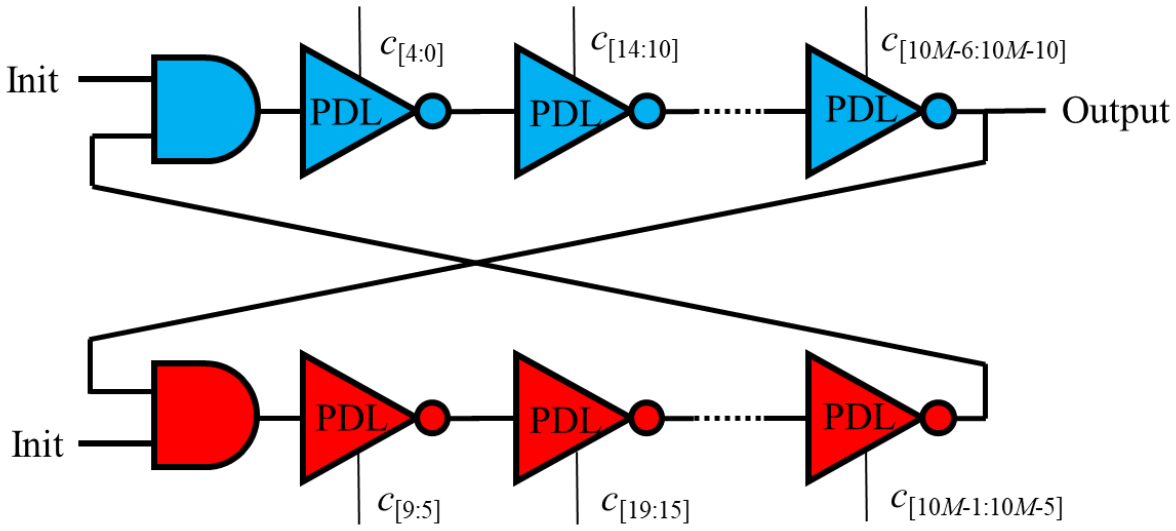


Figure 6-1 DPDL-TERO structure. Different challenges configure the two branches.

2) Two branches configured by the same programmable bits (SPDL-TERO)

In order to minimize the biases' effect, the second structure programmed its two branches by the same challenge. In CHAPTER 3, we found a pattern in the nominal delays of the 32 PDLs in the LUT6. To avoid the biases due to such a pattern, we configure the two branches with the same challenge. This structure is referred to as SPDL-TERO.

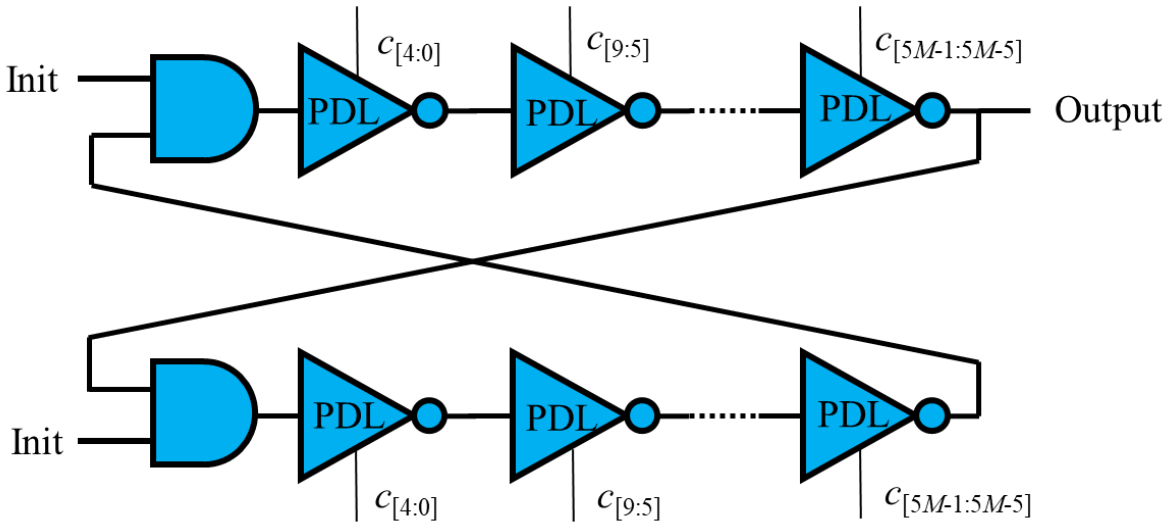


Figure 6-2 SPDL-TERO structure. The same challenges configure the two branches

3) IPD-based TERO

Because of the systematic biases found in CHAPTER 3, we proposed intertwined programmable delay (IPD) to mitigate the major biases in LUT6. Since the experimental results have proved that IPD has significantly reduced the impacts of biases in LUT6, we would like to also build a TERO using IPD.

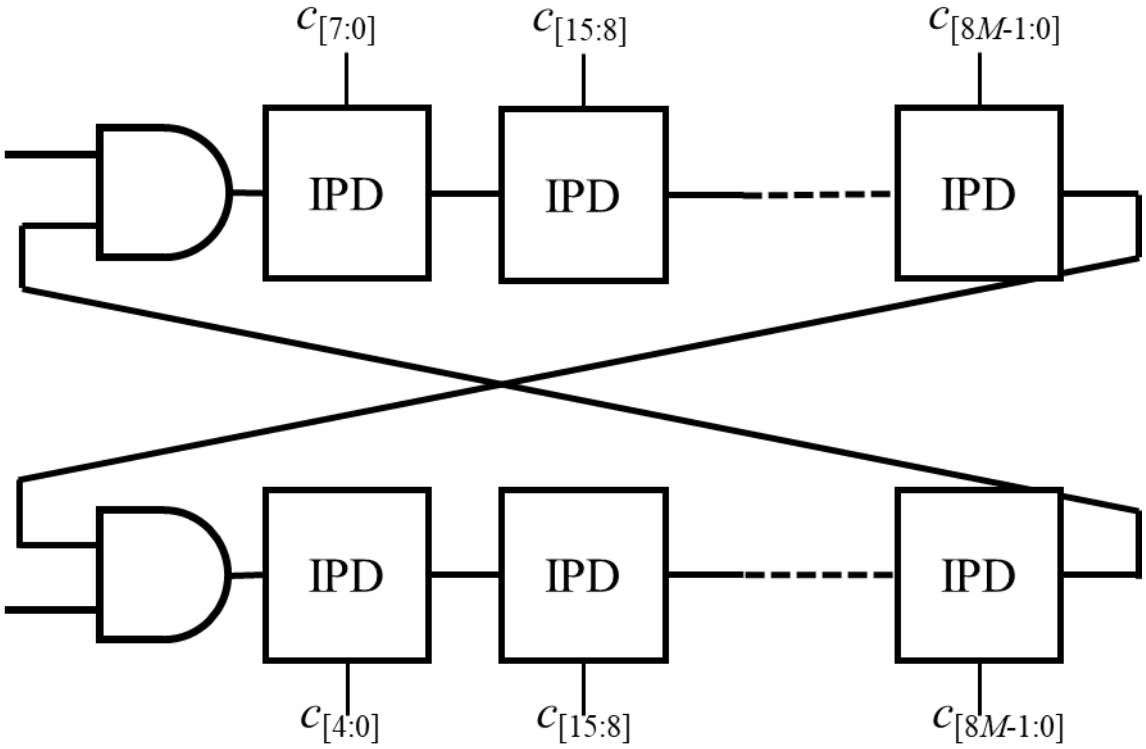


Figure 6-3 IPD-TERO structure.

B. TERO-PUF extraction scheme

1) The final stable state (Method Stable)

Using the final state as the source of PUF response was discussed in [62]. [51] concluded that this PUF response extraction scheme is not useful, since 29% of the tested TERO are unstable. However, since this thesis explores the use of PDLs in TERO-PUF, it is worthwhile to consider the final state as a candidate for PUF extraction schemes.

The binary state can be directly used as the 1-bit PUF response in this scheme. With a challenge *chl*, one could get the PUF response r_S.

2) The length of transient state (Method Transient)

The traditional TERO-PUF [49] has two blocks, each of which has many TEROs. The challenge selects one TERO from each block, and the outcomes of the two selected TEROs are compared. The comparison result determines the one or multiple bits of the PUF response [49][51]. For the

reliability of the PUF responses, the Gray code was used in [49]. The selection of the specific bit of the outcome was discussed in [49][51].

Thanks to PDLs, our work has plenty of signal propagation paths in one physical PDL-based TERO. For DPDL-TERO and SPDL-TERO, we will capture two readings from the same physical TERO and determine 1-bit PUF response based on their comparison outcome. For IPD-TERO, the modified 2-pass scheme mentioned in Chapter 4 will be used.

C. TERO-PUF entropy source

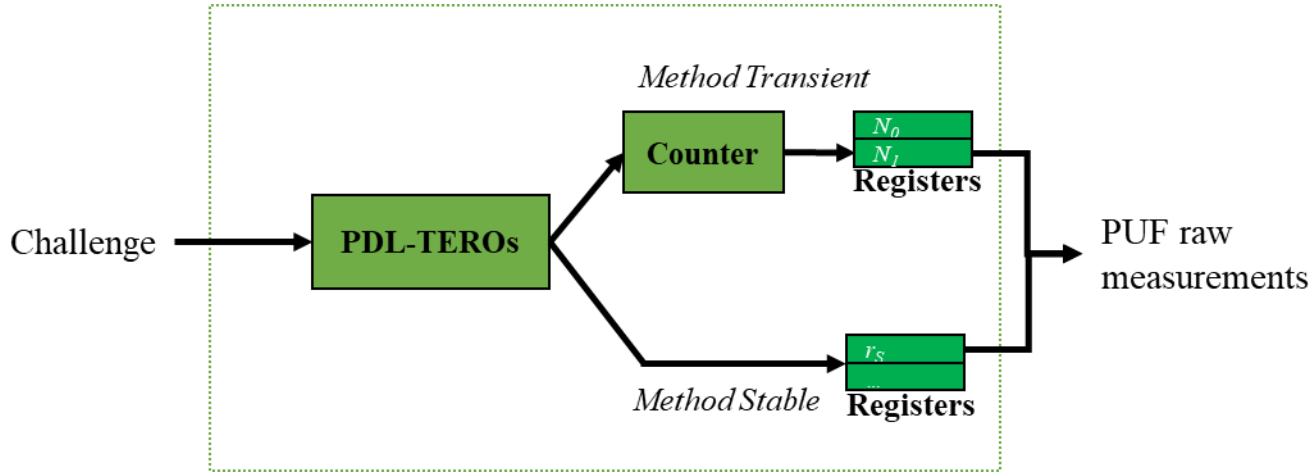


Figure 6-4 Architecture of the PDL-TERO-PUF testbench

Figure 6-4 shows the two methods of testing the PDL-TEROs, tested by the PUF test suites, as shown in Figure 2-13. The tested challenges from BRAMs program the PDL-TEROs under test. The counter counts the positive edge of the PDL-TERO output. After the pre-defined acquisition time, the initialization signal is turned off, and the counter readings are stored in the output registers. For the Method Stable, the final state of the TEROs is stored in registers. These information are the PUF raw measurements to be processed by Zynq processor.

D. TERO Settling Time

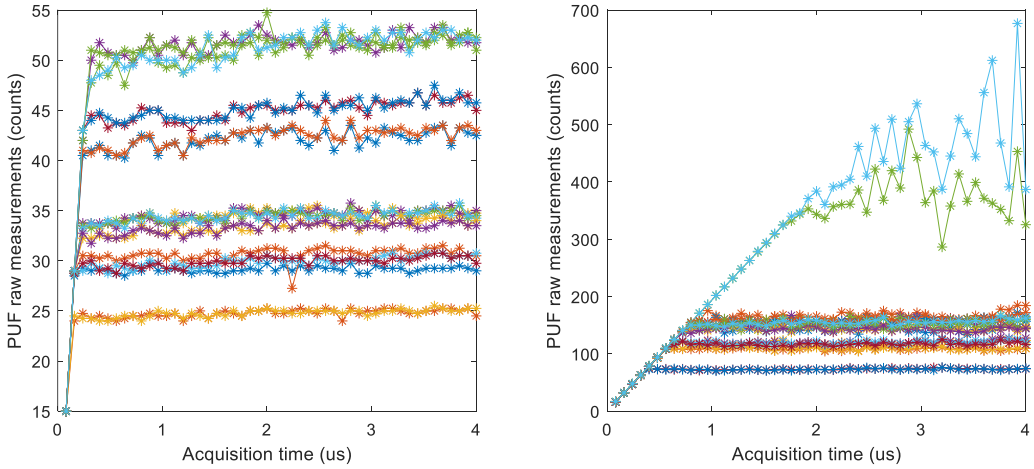


Figure 6-5 TERO requires a sufficient acquisition time.

Before we characterize our TERO-PUFs, let us take a quick look at the unstable state in TERO. Like previous literature, we have also seen many unstable TERO final states in our experiments. Different PDL-TEROs become stable after different times. Therefore, this raises the requirement for the acquisition time that it should be large enough to make enough PDL-TEROs stable. Otherwise, the unstable PDLs tend to give similar outcomes, which is limited by the counter and system clock. Figure 6-5 shows the raw measurements of two DPDL-TEROs, where we can see two very different settling times. All the PDLs in this TERO quickly settle at around 0.3 us for the one on the left. In contrast, the one on the right shows longer settling time. While most PDLs settled within 1 us, two did not stablize at 4 us. The average settling time in our prior tests is about 0.894 us. Therefore, we choose to keep use the same acquisition time (15.729 ms) as Chapter 4. Such acquisition may be overkill for the TEROs, and this may be an advantage of TEROs.

E. TERO-PUF Characterization

1) Extract with Method Stable

We will also characterize and analyze TERO-PUFs with bit-aliasing first. In this chapter, for simplicity, we only find the overall bit-aliasing of each TERO structure. The bit-aliasing of

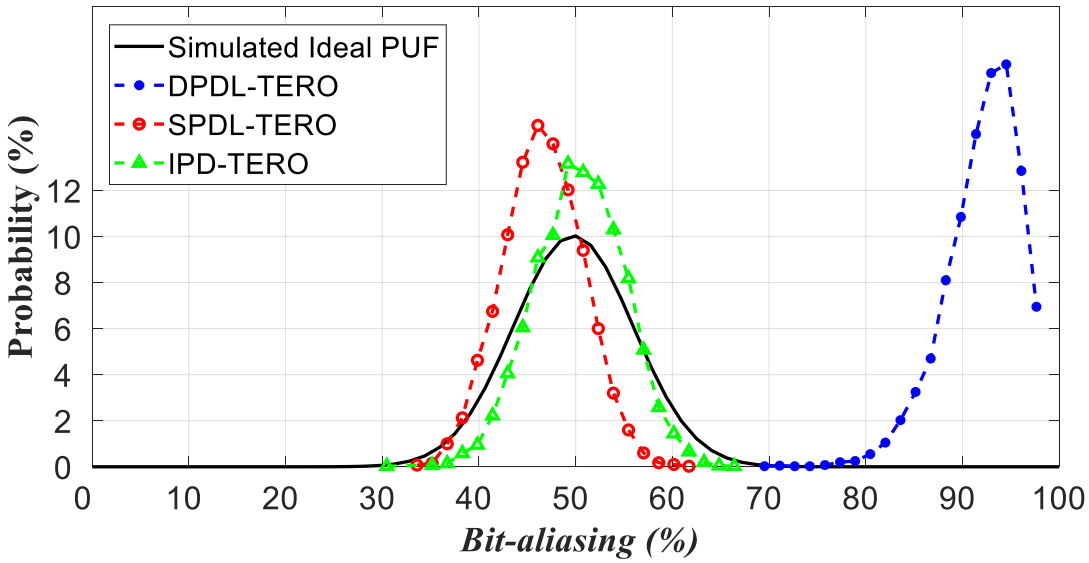
Method Stable is shown in Figure 6-6(a). The distribution of DPDL-TERO lies to the right, which means most PUF responses are 1's. No matter the challenges, it always stabilizes at a high voltage level. The distribution of SPDL-TERO is better, whose center locates roughly at 46%. The bit-aliasing of IPD-TERO seems pretty good, whose shape is quite close to the ideal one.

Figure 6-6(b) shows the correlations of the three structures. First, the correlation of DPDL-TERO is poor, as expected. However, SPDL-TERO and IPD-TERO are also highly correlated. Although PUF responses of these TERO-PUFs are relatively randomly distributed, they are positively correlated to the challenges.

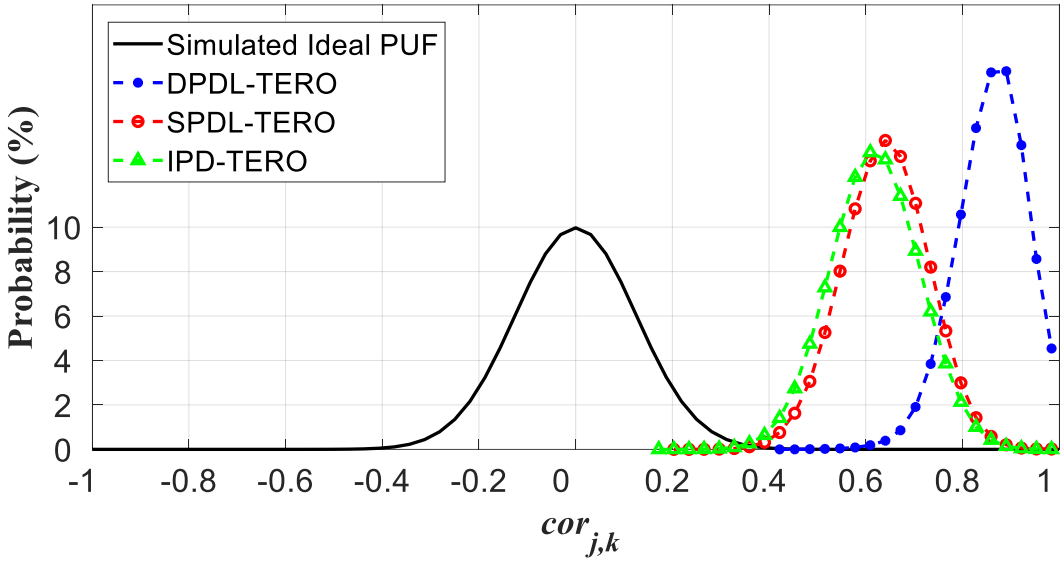
The results of other metrics are shown in Table VII. As shown by bit-aliasing, the majority of PUF responses of DPDL-TERO are 1's, which explains its poor bit-aliasing and correlation. Thus, it uniquenesses are severely affected.

SPDL-TERO has better results. Notably, its inter-uniqueness (31.39%) is much worse than the overall one (50.10%) and intra-uniqueness (49.00%). We deduce that a similar bias in the same locations of different devices has a similar impact on the final stable states. Since we only have two devices, we cannot confirm this assumption.

The higher reliability of DPDL-TERO indicates the bigger difference in its two branches' delays. Different challenges configure the two branches, and thus, their nominal delays are inherently different. As the two branches are more mismatched, the oscillations in DPDL-TERO die down quicker. Therefore, more TERO paths become stable within the acquisition time, making the reliability of DPDL-TERO (99.41%) higher than DPDL-TERO (98.68%). The unreliable TERO paths have two scenarios: 1) Some TERO paths are still unstable after the acquisition time; 2) Some TERO paths' final states are not bound to any specific voltage level.



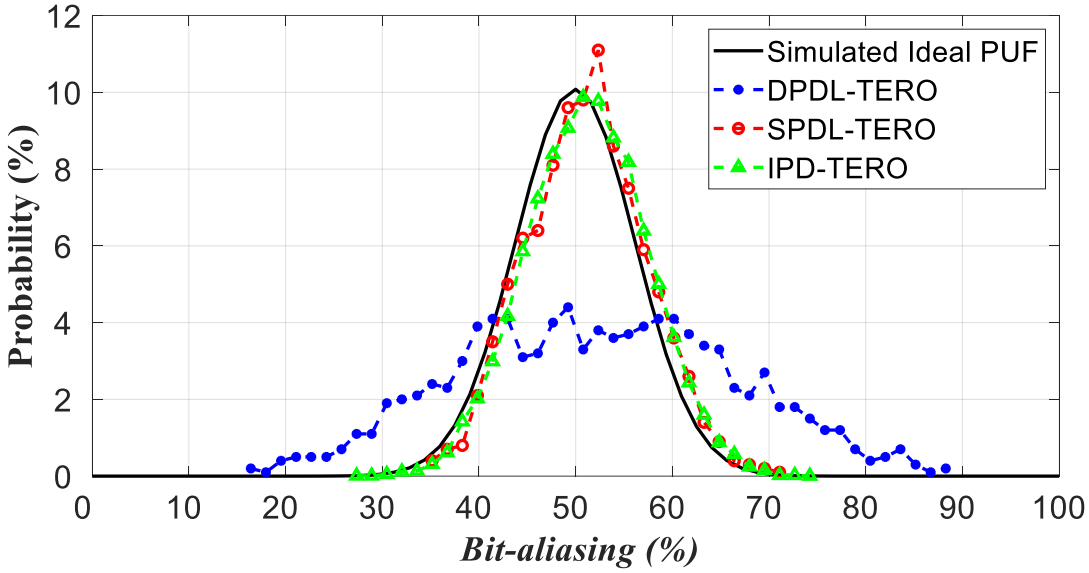
(a)



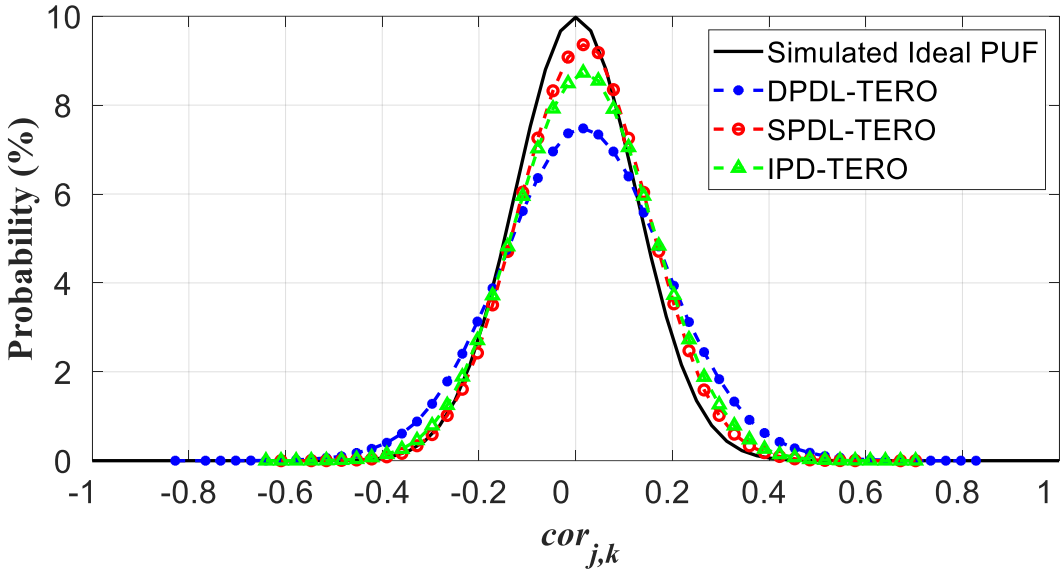
(b)

Figure 6-6 (a) Bit-aliasing distribution. (b) Correlation distributions.

2) Extract with Method Transient



(a)



(b)

Figure 6-7 (a) Bit-aliasing distribution for Method Transient. (b) Correlation distributions.

Method Transient greatly improves bit-aliasing and correlations, as shown in Figure 6-7. SPDL-TERO and IPD-TERO offer close bit-aliasing and correlations to the ideal among the three structures. Furthermore, their correlations are better than the one of IPD-RO-PUF shown in Figure 4-12.

Compared to the others, DPDL-TERO's bit-aliasing is far from close to the ideal. Compared to the results in CHAPTER 4, it is quite close to the one of IPD-RO with the traditional 2-pass

scheme. They two both don't have a mechanism to mitigate the biases pattern. In different ways, the biases pattern impact on the final PUF responses of the two structures. The results prove that the biases in delay greatly influence the outcome of the TERO.

As shown in Table VII, the other metrics of SPDL-TERO are very close to the ideal values. While the reliability is kept at a very high level, the uniqueness is better compared to DPDL-TERO.

Table VII Characterization of PDL-TERO-PUF and other PUFs

	Uniformity	**Uniqueness**	**Uniqueness$_{Inter}$**	**Uniqueness$_{Intra}$**	**Reliability**
Method Stable					
DPDL-TERO	91.22%	16.01%	9.78%	16.11%	99.41%
SPDL-TERO	45.82%	50.10%	31.39%	49.00%	98.68%
IPD-TERO	49.60%	50.37%	34.47%	46.67%	99.06%
Method Transient					
DPDL-TERO	51.27%	46.81%	45.88%	46.76%	96.35%
SPDL-TERO	50.33%	50.05%	49.67%	50.17%	96.17%
IPD-TERO	50.39%	49.98%	49.87%	49.95%	96.24%
TERO-PUF [49]	N/A	48.5%	N/A	N/A	92%

F. Characterization results analysis

Based on the experimental results, we have drawn the following conclusions.

1. Due to the strong biases in LUT structures, the nominal delays of the two branches have to be matched. In both PUF response extraction methods, SPDL-TERO shows stronger performance than DPDL-TERO. That indicates the biases between different PDLs impact both the final stable state and the transient state length.
2. The final stable state is not applicable in any discussed TERO structure. Although the reliability is much higher than the one reported in [51], the other metrics suffer heavy impacts of biases in LUTs. The final stable state is heavily dependent on the relationship between the two branches. This relationship may include but is not limited to the delay and

the initial voltage levels.

CHAPTER 7. ADVANCED PUF ANALYSIS

A. NIST randomness test suite

Although originally developed to evaluate random number generators, many works have applied the National Institute of Standards and Technology (NIST) test suite to PUF. Passing the NIST test suite does not guarantee the randomness of a PUF, but failing it indicates that the tested PUF lacks randomness [54]. Six statistical tests in the suite are applied to PUFs, including:

- Frequency test (T1);
- Frequency test within a block (T2);
- Runs test (T3);
- Test for the longest run of ones in block (T4);
- Approximate entropy test (T5);
- Cumulative sums forward (Cusum-F) test (T6);
- Cumulative sums reverse (Cusum-R) test (T7).

The test suite is open access and can be found in [55]. Table VIII shows the results of the tests on the RO and TERO-based PUFs we discussed before. First of all, among the RO-based PUFs, both the investigative PDL-RO-PUF and IPD-RO-PUF show a high pass rate (all above 95%). It is surprising that the PDL-RO-PUF, showing strong biasing in the characterization, has a very high pass rate in the NIST tests. This result proves that one cannot simply trust NIST randomness tests when evaluating a PUF. In the family of TERO-based PUFs, IPD-TERO-PUF shows the best results. DPDL-TERO and SPDL-TERO show a lower pass rate, which our previous tests expect.

Table VIII NIST randomness test suites results.

	T1	T2	T3	T4	T5	T6	T7
Investigative PDL-RO	100.00%	100.00%	100.00%	96.88%	100.00%	100.00%	100.00%
IPD-RO	96.88%	100.00%	100.00%	100.00%	100.00%	96.88%	96.88%
DPDL-TERO	56.25%	84.38%	71.88%	87.50%	100.00%	56.25%	56.25%
SPDL-TERO	68.75%	93.75%	84.38%	90.63%	100.00%	68.75%	68.75%
IPD-TERO	90.63%	96.88%	96.88%	100.00%	100.00%	90.63%	90.63%

B. Neural Networks attacks

Machine learning attack for PUF has been reported previously for PUF. [27][28][29] Deep learning has been proved to be very efficient modeling APUF and various PUF based on APUF in [29]. Besides, [27] presented successful modeling attack to RO PUF and various PUF based on that with various machine learning techniques including Logistic Regression and Evolution Strategies. Therefore, the resilience to machine learning has become an important criterion for PUF.

All the PUFs proposed in this thesis fall into the type of strong PUF. Strong PUFs usually have no protection mechanisms that restricts Eve in challenging them or in reading out their responses. Their responses are freely accessible from the outside. In [27], the attacked RO PUF was closer to a RO bank PUF, which has very limited number of CRPs. For the proposed PUFs, the number of CRPs are huge. Especially, they can be implemented without discarding any CRPs, thus almost all CRPs can be useful. The huge number of CRPs would be one obstacle for modeling attack.

To analyze the resilience to ML attack, we suppose Eve is able to collect a large number of CRPs. Machine learning attack using Neuron network (NN) is conducted for all the characterized RO-based and TERO-based PUFs. For all PUFs, we consider the structures using 20 programmable bits. Because they all use 2-pass schemes, the length of the challenge is 40-bit. 20,000 CRPs are collected for each PUF. To develop an overview of their resilience to NN attacks, I extensively ran NN training with the collected PUF responses from 32 instances on each of the

two devices.

Figure 7-1 shows an overview of the NN attack results. First of all, the family of TERO-based PUFs shown stronger resilience to NN attacks than the other two. The more complicated process seems to give additional complexity in their PUF responses. The delays in PDL-RO and IPD-RO have additive effects on the raw measurements. These effects are easier for NN to predict. Big improvements in canonical metrics like bit-aliasing, uniqueness have been shown in CHAPTER 4. However, the improvement in NN attack resilience is not that obvious, from about 89% on average to 86%.

In the family of TEROs, the resiliences are quite close. An interesting observation is that although DPDL-TERO was biased, its resilience is quite close to DPDL-TERO. The prediction of IPD-TERO-PUF ranges in a very wide range. Especially, more than 25% of IPD-TERO-PUF have predictions lower than 70%.

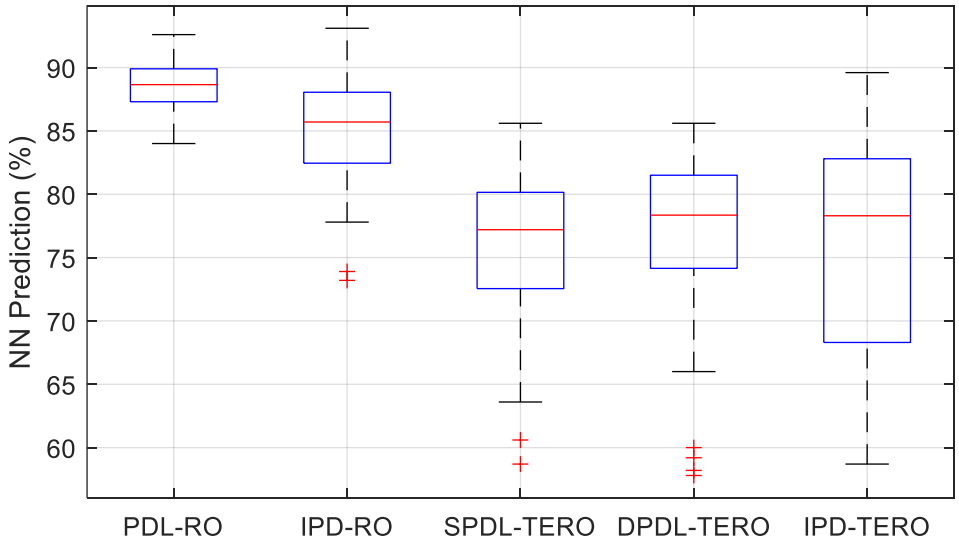


Figure 7-1 Neuron network attack confusion table (a) IPD-RO (b) VRO

C. Entropy and Bit-bias

1) Chain Rule of Entropy and Bit-wise Entropy

Entropy in information theory measure how the level of information in a system. It can also

evaluate the randomness of the system. Shannon defined the entropy H as in the following equation:

$$H(X) = -\sum_{x} P(x) log_2 P(x) \tag{7-1}$$

Where x is the outcome of the system and $P(x)$ is the probability that the system gives x. The chain rule of Shannon entropy is described as:

$$H(X,Y) = H(X) + H(Y|X) \tag{7-2}$$

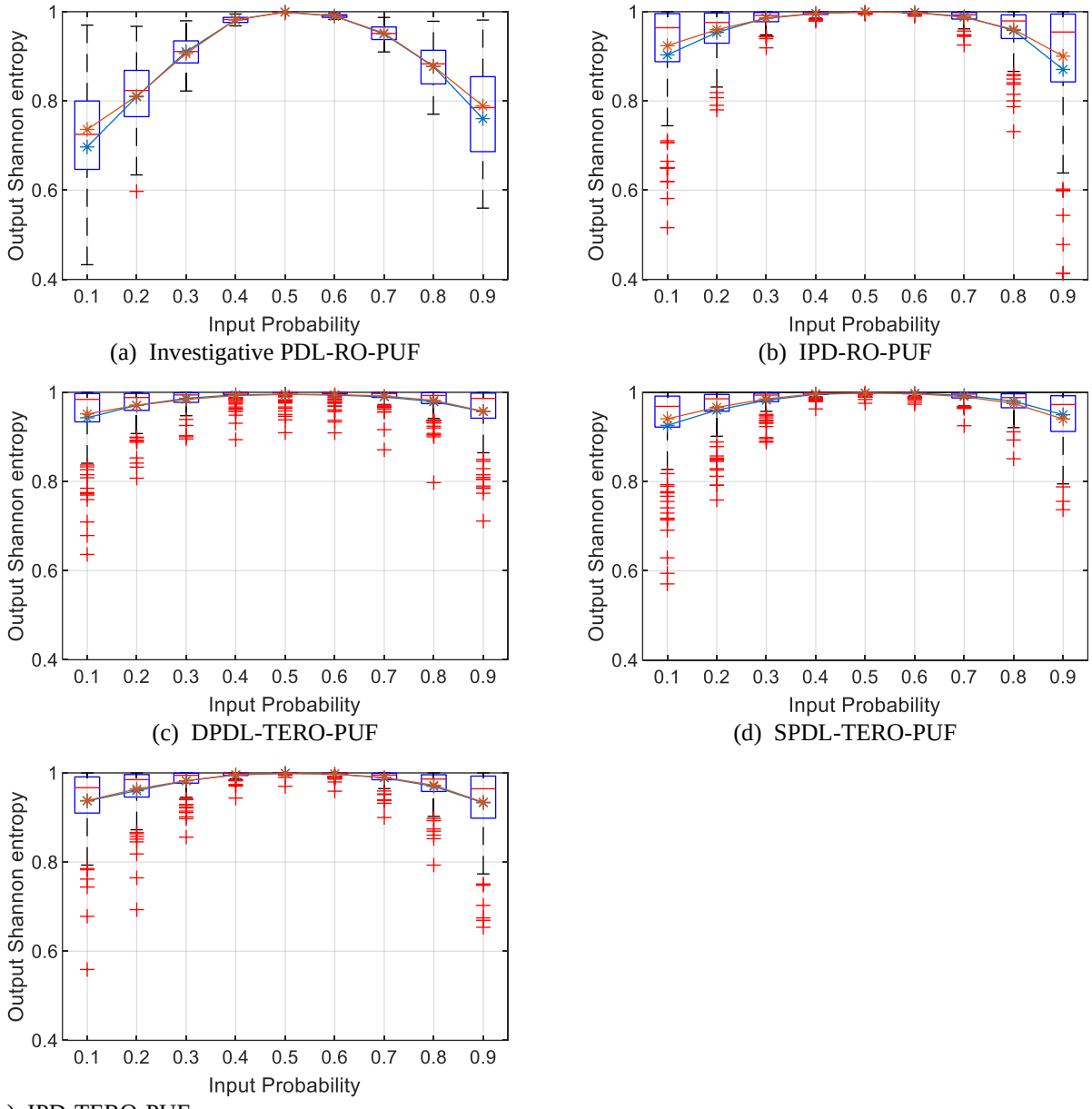
When Y and X are independent,

$$H(X,Y) = H(X) + H(Y) \tag{7-3}$$

For an independent event series X_i

$$H(X_1, X_2 \dots, X_n) \leq \sum_{i=1}^{n} H(X_i) \tag{7-4}$$

For a simple RO PUF with n configurable bits, (7-4) can be used to estimate the entropy of the RO PUF if each bit is independent [26]. However, our designs fully use the PDLs in LUT6s, so each challenge bit is not uncorrelated. Thus, estimating entropy using the joint entropy is not the best approach for us. We then estimated the overall Shannon entropy of each PUF corresponding to the input probabilities, another approach in [26]. Figure 7-2 shows the estimated Shannon entropies, where the input probabilities are the 1's in the input challenge of one pass. First of all, when the input entropy is maximized, i.e., the probability is 0.5, all PUFs achieved very high output entropy. When the inputs are biased (like 0.3 and 0.4), the entropy of PDL-RO-PUF becomes decreasing, and our designs maintain pretty well. When the inputs are extremely biased (0.1 and 0.2), the entropy of PDL-RO-PUF falls down below 0.8, and our PUFs are generally still higher than 0.9.



(a) Investigative PDL-RO-PUF (b) IPD-RO-PUF

(c) DPDL-TERO-PUF (d) SPDL-TERO-PUF

(e) IPD-TERO-PUF

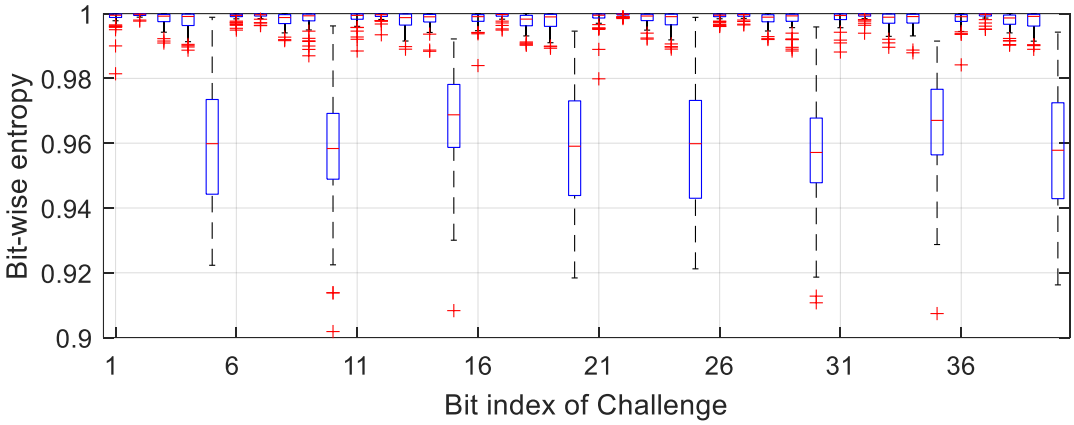
Figure 7-2 Shannon entropy corresponding to challenge bit probability

2) Bit-wise Entropy

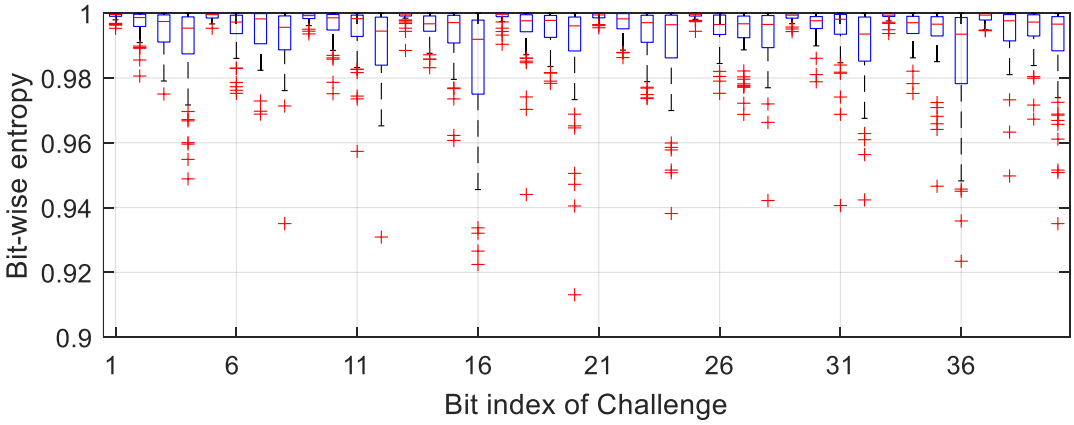
The Shannon entropy only approximates the overall difficulty of attacks. Whereas we are interested to know where the vulnerability of the PUF is. Equation (7-5) defined bit-wise entropy, where x and y are the PUF responses corresponding to challenges c_x and c_y [63], and the Hamming distance of c_x and c_y is always 1 bit. The objective of this metric is to find entropy corresponding to each bit of challenge.

$$H(X|Y) = -\sum_{(x\in X, y\in Y)} p(x,y) log_2\left(\frac{p(x,y)}{p(y)}\right) \tag{7-5}$$

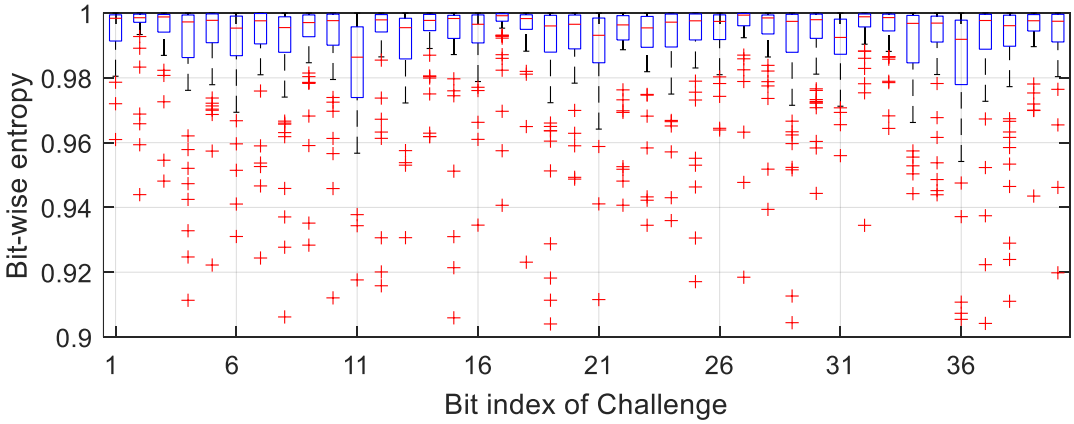
Figure 7-3 shows the bit-wise entropy of the discussed PUFs. 20 bits of each challenge in either pass are considered. Here, we can see the systematic biases in PDLs found in CHAPTER 3. Because there is no mitigation strategy in the investigative PDL-RO-PUF, the found biases, especially the major bias between two LUT5s controlled by every 5th bit, dramatically decrease the bit-wise entropy to around 0.96. These bits may be more vulnerable to ML attacks than other bits. The mitigation in IPD-RO-PUF clears those weak bits. However, we can see relatively low bit-wise entropy every 4th bit. A similar situation can also be seen in DPDL-TERO-PUF and IPD-TERO-PUF. And the one of SPDL-TERO-PUF has the best bit-wise entropy.



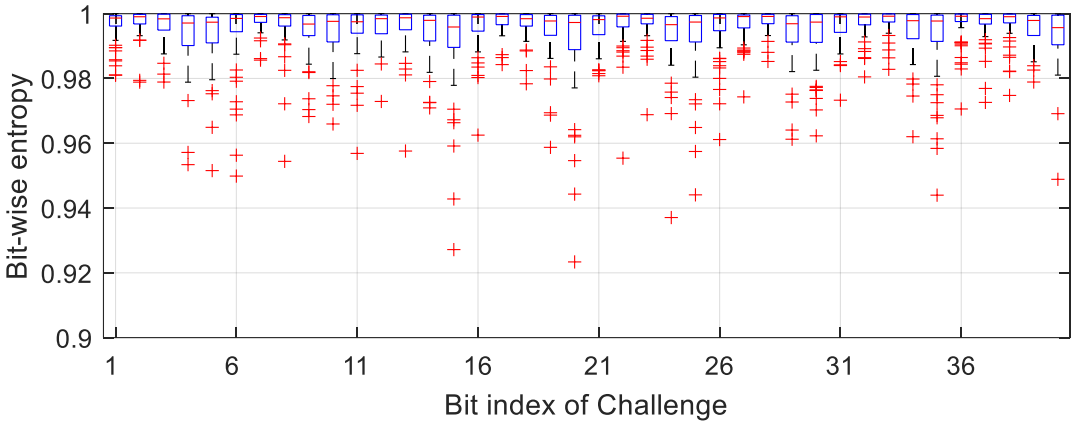
(a) Investigative PDL-RO-PUF



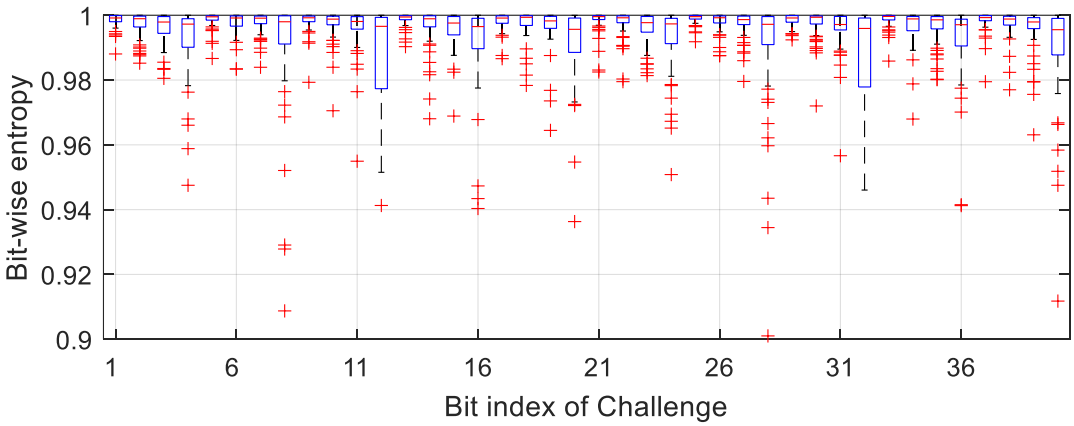
(b) IPD-RO-PUF



(c) DPDL-TERO-PUF



(d) SPDL-TERO-PUF



(e) IPD-TERO-PUF

Figure 7-3 Entropy of 32 ROs (a) Implemented with intertwined LUT stages (b) Implemented with non-intertwined LUT stages.

3) Bit-bias

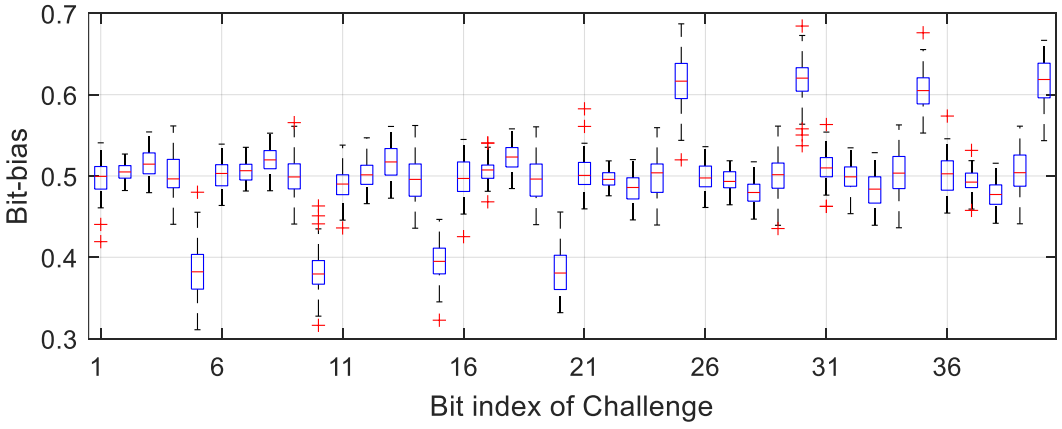
Along with the path of bit-wise entropy, we find that it may be beneficial to explore how each bit of the challenge impacts the PUF response. Therefore, we calculate the bit-bias as the following

equation,

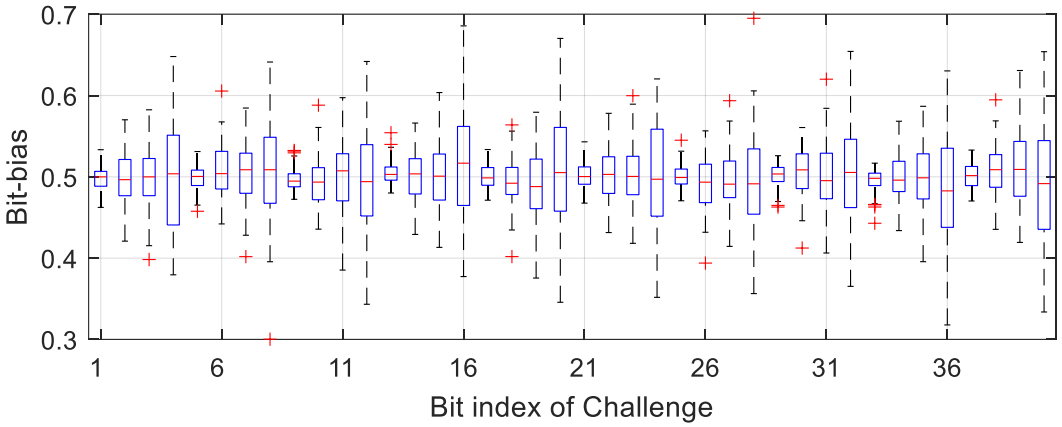
$$Bit - bias = HD(chl, R)$$

Figure 7-4 shows the resulted bit-bias of discussed PUFs. First, we can see the strong bias, especially at 5th bit of each stage. The result of IPD-RO-PUF still shows that it has mitigated the majority of the biases. However, et each of 4th bit of the stage, although not biased, the bit-bias has a relatively larger spread than other bits. The spread means a specific IPD-RO-PUF may still suffer some bias, which might be vulnerable to attackers.

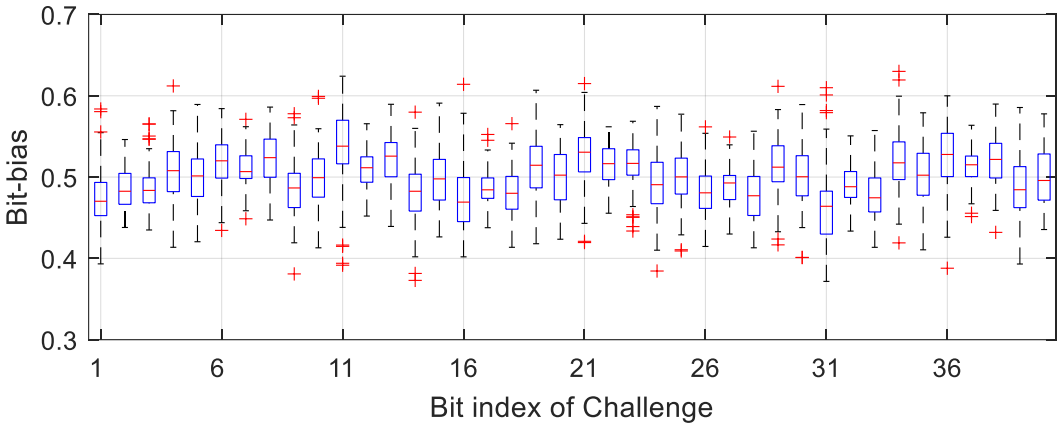
As for the family of TERO-PUFs, we were not able to detect very clear patterns in the bit-bias. Unlike the delay-based PUFs, the competing elements from the transient effect is not simply proportional to the delays. Speaking of the spread, we see that the spread of DPRL-RO-PUF is more extensive, and the ones of SPDL-TERO-PUF and IPD-TERO-PUF are quite comparable. These results are consistent to the ML attack rates. Thus, we think this metric could be a candidate for the fast characterization of PUF's resilience to ML attacks.



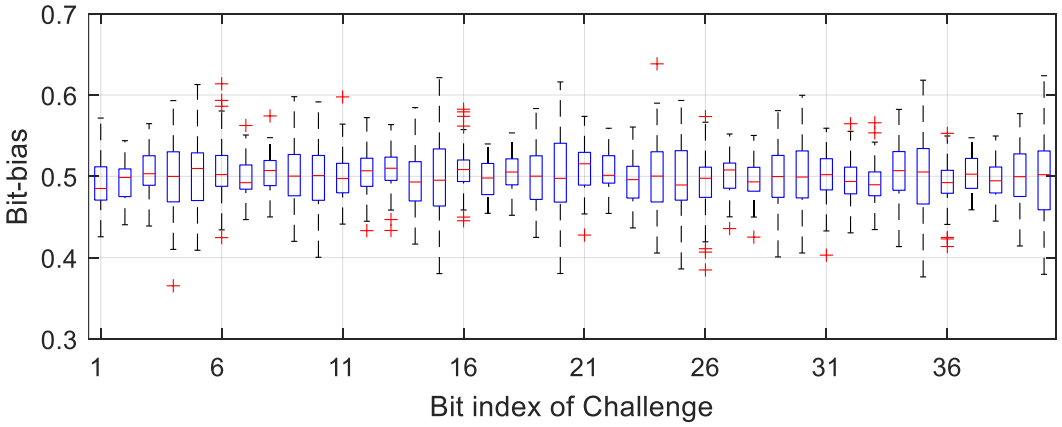
(a) Investigative PDL-RO-PUF



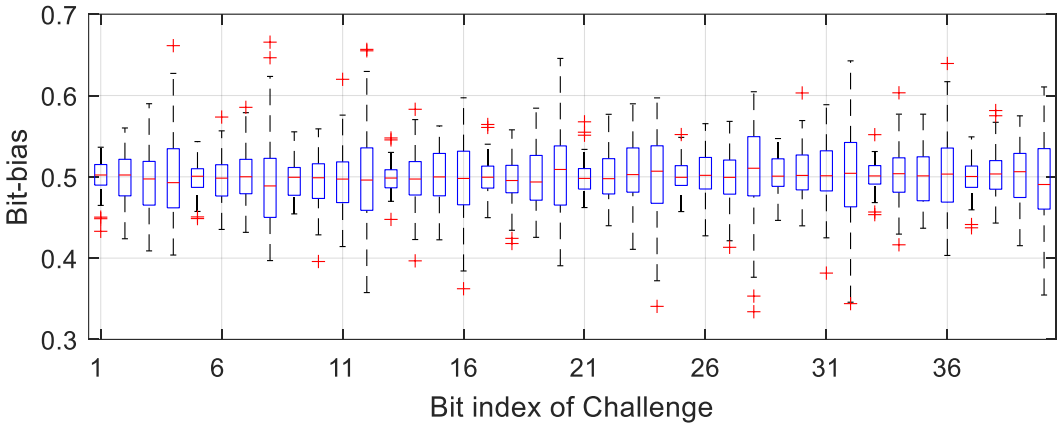
(b) IPD-RO-PUF



(c) DPDL-TERO-PUF



(d) SPDL-TERO-PUF



(e) IPD-TERO-PUF

Figure 7-4 Bia-bias of discussed PUFs

We propose measuring the bit-bias standard deviation to guess the PUF's resilience to ML attacks for effortless comparison. It is easier for ML attacks if one bit of the challenge is more severely biased. Therefore, we can measure the variation of bit-bias to know whether there are severely biased bits. We plot all the 64 instances of each PUF and their NN predictions (was shown in Figure 7-1) in Figure 7-5. We do see the association between bit-bias variation and NN prediction rate, and a transparent lower boundary of the NN prediction rate corresponding to certain values of bit-bias variation. This novel metric can facilitate the study of PUF's resilience to ML attacks. One may be able to derive the best resilience, i.e., the lowest precision rate, with the lower boundary. Furthermore, it saves tons of hardware resources like high-end GPU and long training time.

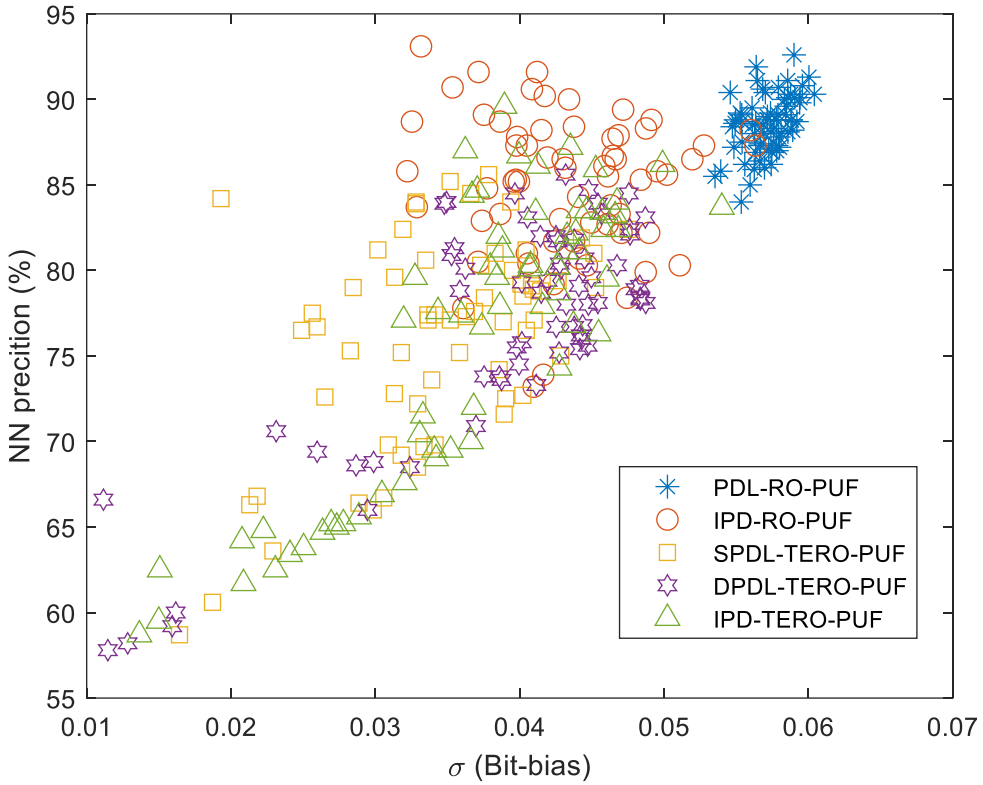


Figure 7-5 NN prediction rate compares with bit-bias variation

CHAPTER 8. APPLICATION: SECURE FPGA BOOT-UP WITH RECONFIGURATION AND PUF

A. Vulnerability in SRAM-FPGA Boot-Up Process

Due to the volatile nature of SRAMs, the SRAM-based FPGAs usually store their encrypted configuration bitstreams in non-volatile flash memories outside of FPGA chips. The conventional boot-up process is shown in Figure 8-1, where the encrypted configuration bitstream gets loaded onto FPGA during the system boot-up process. The bitstreams are decrypted first and then passed on to FPGA fabric to configure blank FPGA fabrics into functional circuits. The FPGA circuits' functionality and associated IPs are stored in the encrypted configuration bitstreams.

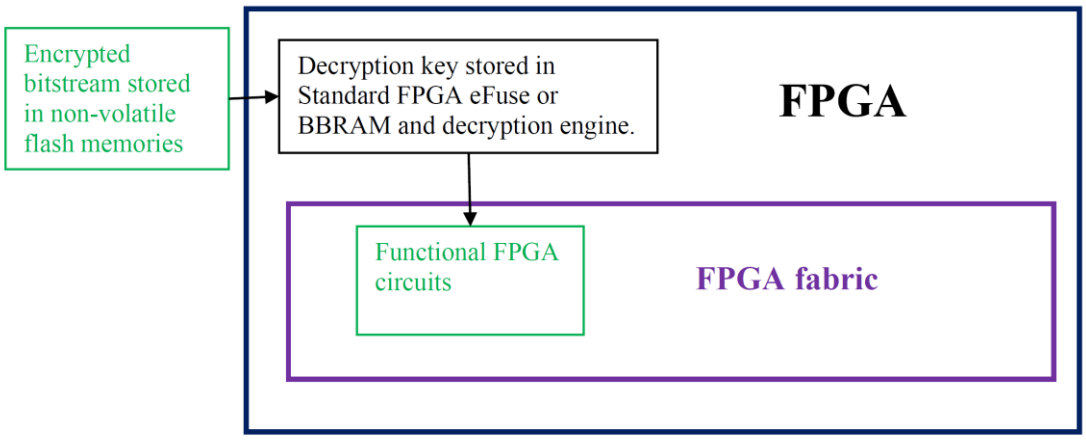


Figure 8-1 The standard FPGA configuration bitstream storage, decryption and programming setup.

Unfortunately, SRAM FPGAs are vulnerable to reverse engineering if the hackers obtain the encrypted configuration bitstreams and corresponding decryption keys. In most applications, inevitably, FPGA chips (where the bitstream decryption keys are embedded) and the non-volatile flash memory (where the encrypted configuration bitstreams are stored) may both be obtained by the hackers [31].

As shown in Figure 8-1, conventional bitstream management practice makes FPGA-based systems more vulnerable than ASIC-based systems in terms of resistance against reverse

engineering. In ASIC, it is very difficult to extract a huge amount of circuit information spread across the entire ASIC chips, even if destructive reverse engineering operations are performed. In contrast, hackers just need to extract decryption keys to unlock the encrypted configuration bitstreams stored in non-volatile flash memories for FPGA systems. Since the decryption keys are usually only hundreds to thousands of bits long, it is much easier to extract decryption keys from FPGAs than to obtain complete circuit information from ASICs.

Furthermore, blank FPGA chips can be easily obtained in the open market. Hackers have plenty of opportunities to learn how to extract decryption keys stored in eFUSE or Battery Back SRAMs (BBSRAMs) through destructive methods. FPGA decryption key extraction has been successful, demonstrated, and published using destructive methods, such as scanning electron microscope (SEM) [31]. The destroyed original FPGA chip can be easily replaced by a new FPGA chip of the same model.

B. Adapt PUF into dynamic partial reconfiguration (DPR)

Dynamic partial reconfiguration (DPR) offers FPGA the ability to add, remove or change functionality during operation time, and many applications have implemented it. Based on Xilinx FPGAs, developers can designate regions in FPGAs as Pblock, which can be reconfigured when FPGA is in operation. An example is Hosny's implementation for multi-standard software-defined radio (SDR) with DPR [36].

Figure 8-2 shows the architecture of our application. Each FPGA design consists of two configuration bitstreams: the functional circuit bitstream (shown in blue) and the configuration bitstream for PUF circuits (shown in green).

The key idea is that the functional circuit bitstream is NOT decrypted or programmed into the FPGA chip through regular FPGA bitstream decryption and programming engine. Instead, the functional circuit bitstream is decrypted using a key generated by a PUF and then programmed into

the FPGA fabric by DPR. In addition, such a description key is NOT stored in standard eFUSE or BBSRAMs inside FPGAs. Instead, it is generated on the fly by the PUF during the boot-up process.

Furthermore, after the functional circuit bitstreams are decrypted and programmed onto the fabric, the PUFs circuits have completed their mission. DRP proactively erase them from the FPGA chip. Therefore, the PUFs circuits and generated decryption keys stay inside FPGA for a brief period during the boot-up process. The PUF circuits and associated decryption keys are already erased by DPR when the boot-up process ends.

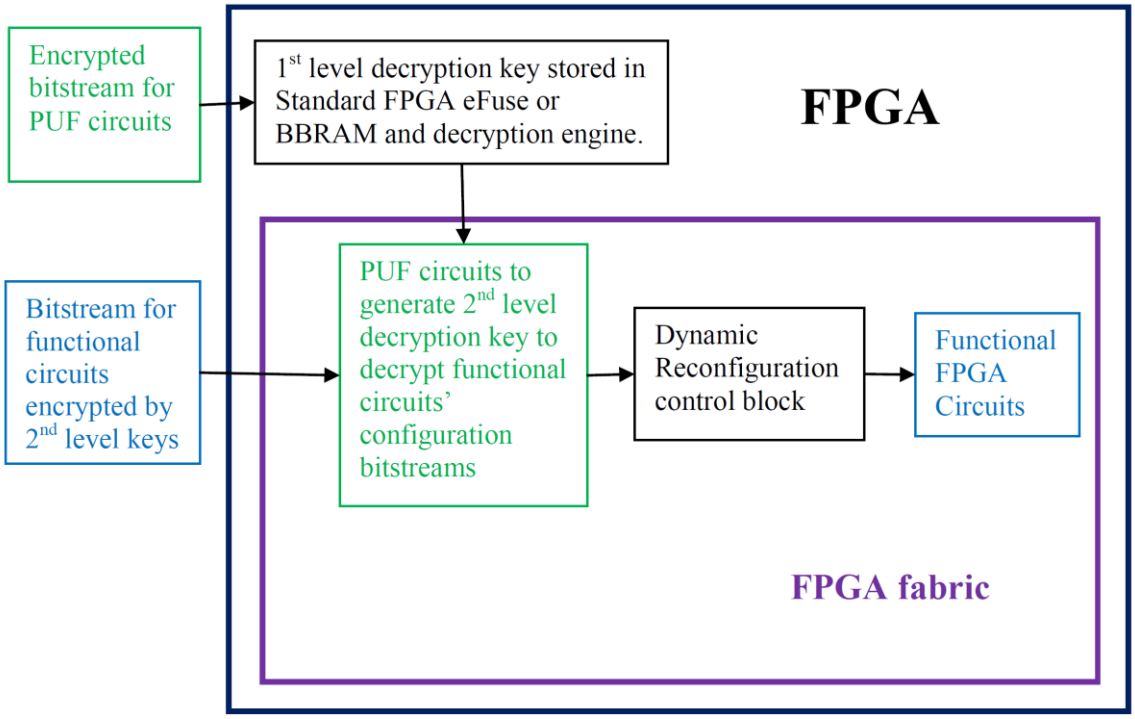


Figure 8-2 Proposed SRAM FPGA zeroization architecture. The blocks in green are the added PUF circuits.

This two-step encryption architecture provides an interlock mechanism to prevent reverse engineering. Hackers need to know the decryption keys generated by the PUF circuits to know the functional circuits. In order to know the PUF generated decryption keys, hackers need to know PUF circuits. They have to extract decryption keys for PUF circuits stored inside the standard

FPGA decryption key storage unit (eFUSE or BBSRAMs). Since modern FPGAs have implemented a mechanism to prevent side-channel decryption key leaks [31], the practical way to extract decryption keys stored in eFUSE or BBSRAM is through destructive reverse engineering. However, the destructive reverse engineering will physically destroy the FPGA chip, and those delicate PUF features embedded in the FPGA chip. Once the specific FPGA chip is destroyed, the decryption key associated with specific PUF features can never be reproduced again. Applying the same PUF circuits onto a different FPGA chip of the same model will not duplicate decryption keys.

C. Boot-up with DPR and PUF

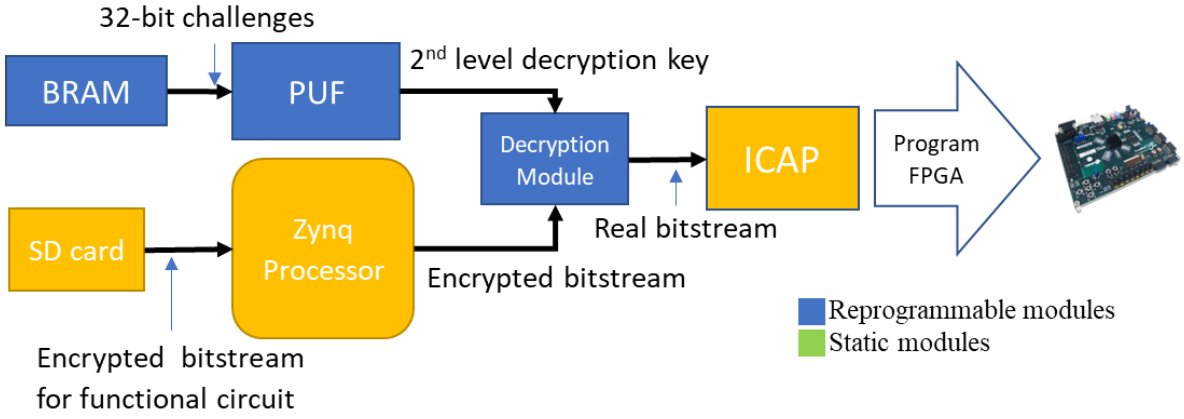


Figure 8-3 PUF based secure partial reconfiguration. Modules in blue are reprogrammable, the ones in yellow always stay in the FPGA.

A prototype has been constructed on Zedboard to demonstrate the feasibility of the proposed two-step interlock mechanism. An IPD-RO-PUF provides the 2nd level key. Figure 8-3 shows the structure of the prototype circuit. BRAM stores challenges. BRAM is intentionally set to stand-alone mode, so there's no AXI bus or other direct connection to the ARM processor. Therefore, BRAM is a pure programmable fabric (PL) part isolated from the processing system (PS). This setup makes the circuit less vulnerable to hacking through PS. The prototype's functional circuit bitstream decryption is just a simple XOR. One can be easily replace it with

other decryption algorithms, such as AES. In [37], Xilinx describes how encryption secures 7-series FPGA with AES. Resource utilization by AES is relatively low [38]. Figure 8-3 only shows the decryption operation. Encryption is done in the circuit design phase, prior application phase. Plain bitstream is encrypted with the 2^{nd} level key. After decryption, the functional circuits are programmed into PL using DPR [39] through ICAP (Internal Configuration Access Port). Although partial reconfiguration is often performed through PCAP (Processor Configuration Access Port), which directly transmits bitstream through the processor [3], we use ICAP to prevent hacking from the software running in the processor. ICAP enables internal readback of the device configuration [31]. Thus, the possible threat from the FPGA and ARM processor integration is reduced. Furthermore, ICAP provides many advantages, like the ability against differential power attacks (DPA) or other side-channel attacks.

The PUF circuits implemented in this prototype come from the IPD-RO-PUF in CHAPTER 4. The PUF circuit consists of ROs and counters, and a 32-bit challenge string is sent to LUTs to configure the IPD-RO. The counter records the number of oscillations within a pre-determined time. In the prototype design, a 32-bit decryption key is generated by sequentially sending 32 different challenge strings to PUF circuits. Alternatively, multiple ROs can be instantiated in FPGA, and they can yield multiple PUF bits simultaneously.

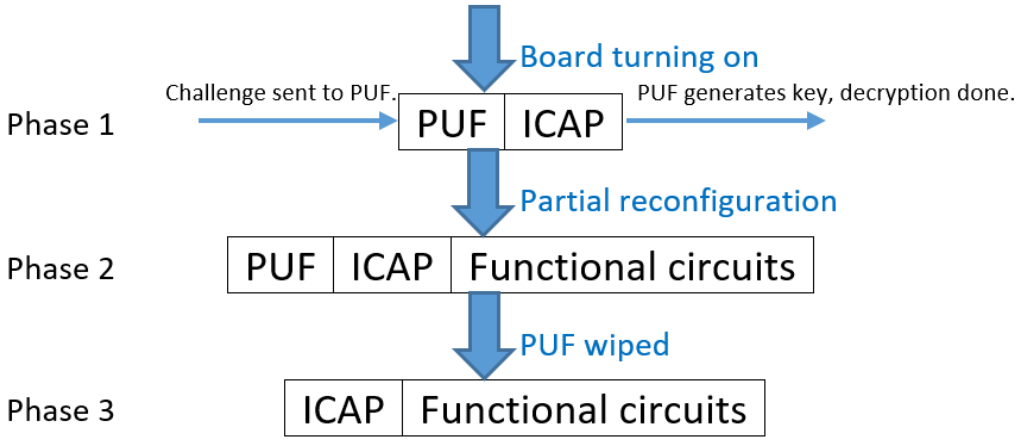


Figure 8-4 Operation flow. Hardware configuring brings the operation from one phase to the next one. Different modules (in blocks) are present on board in different phases.

Figure 8-4 shows the operation flow of the prototype circuit. The entire flow can be broken down into 3 phases. PUF and ICAP are programmed into PL when the board is turned on in Phase 1. Then, 32-bit challenges are sent to PUF, and after a short delay, PUF provides the 32-bit response as 2nd level decryption key. In the meantime, the Zynq processor reads encrypted functional circuit's bitstream from the SD card and sends it to the decryption block. The decryption is done with the generated 2nd level decryption key. After decryption, the plain bitstream is sent to ICAP, and ICAP programs fabrics into functional circuits. At this time, Phase 2 operation ends. The board is equipped with functional circuits. Then, ICAP programs blank modules to replace PUF modules in the Pblock, so that PUF circuits are wiped out. Finally, only the functional circuits and ICAP are left in the FPGA, concluding Phase 3 operation.

D. Implementation Results

Since PUF runs on the fly during the boot-up process, its operation time should not stall the boot-up process of functional circuits. In our prototype design, the majority of the boot-up time is used by PUF to generate 2nd level key. This time is proportional to the length of the 2nd level key, as PUF repeats for the same number of times as the length of the key. In our prototype, each repetition takes about 15.729 ms. This time is consistent with the one in CHAPTER 4, achieving

sufficient readout margin in capturing the delays. The discussion about RO running time can be found in Chapter 0. Since the RO oscillation is independent of the system clock, the clock frequency does not affect this time. In this prototype, the decryption key length is 32-bit, and thus it takes about 503 ms to complete all the 32 runs on PUF. If shortened key generation time is desired, multiple ROs could be implemented to generate multiple bits simultaneously.

PUF and partial reconfiguration circuits are the overhead introduced by this prototype. The extra resources needed are shown in Table IX. The highest utilization comes from the use of 3516 LUTs, i.e., 6.61% of LUT resources available in Zedboard. Although the proposed design is on Zedboard with a Zynq microprocessor, an ARM Cortex processor, the processor is not essential in this implementation as it was only used to fetch decrypted bitstream from SD card. The decrypted bitstream can be stored internal of FPGA, like BRAM. And therefore, the data transfer does not require a processor.

If additional ROs are added to speed up the boot-up process, a few more resources are needed by the additional ROs. The resources consumed by a RO include LUTs building the inverters and the counter measuring the number of RO oscillation cycles. For the prototype design, each additional RO needs 19 LUTs. We can save some resources if we switch to a more compact PUF like DPDL-TERO-PUF.

Table IX Resource utilization on Zedboard (Artix-7, xc7z020clg484-1)

Resources	Utilization	Available	Utilization %
LUT	3516	53176	6.61
FF	3928	106352	3.69
BRAM	6	140	4.29
IO	0	200	0

CHAPTER 9. CONCLUSION AND FUTURE WORKS

This work discusses the implementation of compact PUFs based on the PDLs in LUT6 cells of Xilinx Artix-7 FPGAs. We began with an experimental investigation of the systematic biases of LUT6s. The found biases are categorized into two types: the major systematic bias between the two LUT5s; and the biases within LUT5s. We overcame them by first developing a novel intertwined LUT stage with two LUT6s. We then constructed a 2-phase, 2-pass scheme that mitigated systematic biases within the LUT5 structure. With both types of biases mitigated, the IPD-RO-PUF was solely based on manufacturing variations in LUTs. The characterization of the IPD-RO-PUF shows that our strategy significantly improves the bit-aliasing, uniformity, and uniqueness of PUF responses close to their ideal values. Furthermore, the low correlation in the PUF responses also indicates that random variations can be successfully extracted. This study shows that our IPD-RO can lead to a new generation of strong, compact PUF designs. Environmental variation between the two passes was mitigated by a reference RO, which detects changes in temperature, voltage, etc. Furthermore, with prior knowledge of the IPD-ROs, a filtering can be carried out to discard the marginally reliable CRPs. Both approaches are proven to help to elevate the reliability of the PUF.

Besides, we successfully implemented TERO-PUFs based on PDLs and IPDs, and thoroughly characterized them. The minimal differences in the PDLs and IPDs are ideal for extracting PUF from the transient effect.

Advanced analysis shows that TERO-PUFs based on PDLs and IPDs may be stronger structures to resist ML attacks. We studied the entropy and proposed a novel metric, bit-bias, and its variation, to find the association with ML attacks' success rate.

At last, we presented an application run on Xilinx Artix-7 that dynamically reconfigures the

FPGA circuits on the fly. IPD-RO-PUF protects the dynamic reconfiguration, and this application shows the compactness of the IPD-RO-PUF.

The future works include a further study of the correlation between PUF's structure and its resilience to ML attacks and a practical method to extract PUF from TERO's final state. Also, we are interested in further verifying the bias-bias metric.

REFERENCES

[1] A. Maiti, I. Kim, and P. Schaumont, “A Robust Physical Unclonable Function with Enhanced Challenge-Response Set,” *IEEE Trans. Information Forensics and Security,* Vol. 7, No. 1, Feb. 2012.

[2] B. Habib, K. Gaj, J. Kaps, “FPGA PUF based on Programmable LUT Delays,” *16th Euromicro Conf. Digital System Design*, Sep. 2013.

[3] F. Kodytek, R. Lorencz, “A design of ring oscillator based PUF on FPGA,” *IEEE 18th Int. Symposium on Design and Diagnostics of Electronic Circuits & Systems,* Apr. 2015.

[4] L. Feiten, J. Oesterle, T. Martin, M. Sauer, and B. Becker, “Systemic Frequency Biases in Ring Oscillator PUFs on FPGAs,” *IEEE Trans. Multi-Scale Computing Systems*, Vol. 2, No. 3, Jul-Sep 2016.

[5] U. Rührmair and J. Sölter, "PUF modeling attacks: An introduction and overview," *2014 Design, Automation & Test in Euro. Conf. & Exhib.*, Dresden, Germany, 2014, pp. 1-6, doi: 10.7873/DATE.2014.361.

[6] W. Liu, Y. Yu, C. Wang, Y. Cui and M. O'Neill, "RO PUF design in FPGAs with new comparison strategies," in *2015 IEEE Int. Symposium on Circuits and Systems (ISCAS)*, Lisbon, Portugal, 2015, pp. 77-80, doi: 10.1109/ISCAS.2015.7168574.

[7] A. Maiti, V. Gunreddy, and P. Schaumont, “A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions,” in *Embedded Systems Design with FPGAs*. New York, NY, USA: Springer-Verlag, 2013, pp. 245–267.

[8] C. Herder, M. Yu, F. Koushanfar and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," in *Proc. of the IEEE*, vol. 102, no. 8, pp. 1126-1141, Aug. 2014.

[9] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas, “Silicon physical random functions,” in *Proc. of the 9th ACM Conf. Computer and Communications Security*, CCS 2002, Washington, DC, USA, November 18-22, 2002. ACM, 2002, pp. 148–160.

[10] G. E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *44th ACM/IEEE Design Automation Conf.*, San Diego, CA, 2007, pp. 9-14.

[11] I. Kim, A. Maiti L. Nazhandali, P. Schaumont, V. Vivekraja, H. Zhang “From Statistics to Circuits: Foundations for Future Physical Unclonable Functions.,” in *Towards Hardware-Intrinsic Security. Information Security and Cryptography*, Springer, Berlin, Heidelberg, 2010. https://doi.org/10.1007/978-3-642-14452-3_3

[12] A. Maiti, J. Casarona, L. McHale and P. Schaumont, "A large scale characterization of RO-PUF," in *2010 IEEE Int. Symposium on Hardware-Oriented Security and Trust (HOST)*, Anaheim, CA, 2010, pp. 94-99, doi: 10.1109/HST.2010.5513108.

[13] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Trans. on .VLSI Systems*, vol. 13, no. 10, pp. 1200-1205, Oct. 2005, doi: 10.1109/TVLSI.2005.859470.

[14] A. Maiti and P. Schaumont, "Improving the quality of a Physical Unclonable Function using configurable Ring Oscillators," in *2009 Int. Conf. on Field Programmable Logic and Applications*, Prague, 2009, pp. 703-707, doi: 10.1109/FPL.2009.5272361.

[15] K. Zhou, H. Liang, Y. Jiang, Z. Huang, C. Jiang and Y. Lu, "FPGA-based RO PUF with low overhead and high stability," *Electronics Lett.*, vol. 55, no. 9, pp. 510-513, 2 5 2019, doi: 10.1049/el.2019.0451.

[16] *Vivado Design Suite 7 Series FPGA Libraries Guide*, Xilinx, San Jose, CA, USA 2012, pp. 250-263.

[17] A. Schaub, J. Danger, S. Guilley and O. Rioul, "An Improved Analysis of Reliability and Entropy for Delay PUFs," in 2018 21st Euromicro Conf. on Digital System Design (DSD), Prague, 2018, pp. 553-560, doi: 10.1109/DSD.2018.00096.

[18] L. Feiten, K. Scheibler, B. Becker, M. Sauer, "Using different LUT paths to increase area efficiency of RO-PUFs on Altera FPGAs," *TRUDEVICE Workshop*, Dresden, Mar 2018.

[19] *Cyclone IV Device Handbook*, Vol. 1, Altera, San Jose, CA, USA, 2009, pp. 29-36.

[20] *White Paper, FPGA Architecture,* Altera, San Jose, CA USA, 2006.

[21] Nathan Menhorn, "External Secure Storage Using the PUF", Xilinx, 2018.

[22] Qiaoyan Yu; Zhiming Zhang; Jaya Dofe, "Proactive Defense Against Security Threats on IoT Hardware," in Modeling and Design of Secure Internet of Things , IEEE, 2020, pp.407-433, doi: 10.1002/9781119593386.ch18.

[23] R. S. Chakraborty, S. Narasimhan and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," 2009 IEEE International High Level Design Validation and Test Workshop, San Francisco, CA, USA, 2009, pp. 166-171, doi: 10.1109/HLDVT.2009.5340158.

[24] Hardware Trojans: Lessons Learned after One Decade of Research

[25] G. E. Suh, C. W. O'Donnell and S. Devadas, "Aegis: A Single-Chip Secure Processor," in IEEE Design & Test of Computers, vol. 24, no. 6, pp. 570-580, Nov.-Dec. 2007, doi: 10.1109/MDT.2007.179.

[26] Q. Wang and G. Qu, "A Silicon PUF Based Entropy Pump," in IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 3, pp. 402-414, 1 May-June 2019, doi: 10.1109/TDSC.2018.2881695.

[27] U. Ru¨hrmair, F. Sehnke, J. So¨lter, G. Dror, S. Devadas, and J. Schmidhuber, ''Modeling attacks on physical unclonable functions,'' in Proc. 17th ACM Conf. Comput. Commun. Security, 2010, pp. 237–249.

[28] Y. Ikezaki, Y. Nozaki and M. Yoshikawa, "Deep learning attack for physical unclonable function," 2016 IEEE 5th Global Conference on Consumer Electronics, Kyoto, 2016, pp. 1-2.

[29] J. Huang, M. Zhu, B. Liu and W. Ge, "Deep Learning Modeling Attack Analysis for Multiple FPGA-based APUF Protection Structures," 2018 14th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), Qingdao, 2018, pp. 1-3.

[30] *7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter User Guide*, Xilinx, San Jose, CA, USA July 2018.

[31] Trimberger, S.M., and Moore, J.J.: 'FPGA Security: Motivations, Features, and Applications' Proc. of the IEEE, Aug 2014, 102, (8), pp. 1248-1265, doi: 10.1109/JPROC.2014.2331672

[32] T. H. Szymanski, "Security and Privacy for a Green Internet of Things," in IT Professional, vol. 19, no. 5, pp. 34-41, 2017, doi: 10.1109/MITP.2017.3680952.

[33] Global field programmable gate array market (2020 to 2027) - size, share & trends analysis report by technology, application, region, and segment forecasts. (2020, May 11). NASDAQ OMX's News Release Distribution Channel Retrieved from https://ezproxy.lib.uwm.edu/login?url=https://www-proquest-com.ezproxy.lib.uwm.edu/wire-feeds/global-field-programmable-gate-array-market-2020/docview/2400275707/se-2?accountid=15078

[34] *Zynq UltraScale+ Device Technical Reference Manual*, Xilinx, San Jose, CA, USA, 2020.

[35] *Using SRAM PUF System Service in SmartFusion2 - Libero SoC v11.7*, Microsemi, Aliso Viejo, CA, 2016.

[36] S. Hosny, E. Elnader, M. Gamal, A. Hussien, A. H. Khalil and H. Mostafa, "A Software Defined Radio Transceiver Based on Dynamic Partial Reconfiguration," 2018 New Generation of CAS (NGCAS), Valletta, 2018, pp. 158-161, doi: 10.1109/NGCAS.2018.8572253.

[37] K Wilkinson, 'Using Encryption to Secure a 7 Series FPGA Bitstream', Xilinx, July 2018, https://www.xilinx.com/support/documentation/application_notes/xapp1239-fpga-bitstream-encryption.pdf

[38] Xie Di; Shi Fazhuang; Deng Zhantao; He Wei, "A Design Flow for FPGA Partial Dynamic Reconfiguration," in Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on , vol., no., pp.119-123, 8-10 Dec.2012

[39] Xilinx: 'Partial Reconfiguration Flow on Zynq using Vivado', 2016.

[40] B. Gassend, D. Clarke, M. van Dijk and S. Devadas, "Controlled physical random functions," 18th Annual Computer Security Applications Conference, 2002. Proceedings., Las Vegas, NV, USA, 2002, pp. 149-160, doi: 10.1109/CSAC.2002.1176287.

[41] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In Proceedings of the Computer and Communication Security Conference, November 2002.

[42] J.-W. Lee, D. Lim, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits with identification and authentication applications. In Proceedings of the IEEE VLSI Circuits Symposium, June 2004.

[43] C. Herder, M. Yu, F. Koushanfar and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," in Proceedings of the IEEE, vol. 102, no. 8, pp. 1126-1141, Aug. 2014, doi: 10.1109/JPROC.2014.2320516.

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

[45] S. Morozov, A. Maiti, and P. Schaumont. An analysis of delay based puf implementations on fpga. In P. Sirisuk, F. Morgan, T. El-Ghazawi, and H. Amano, editors, Recongurable

Computing: Architectures, Tools and Applications, volume 5992 of Lecture Notes in Computer Science, pages 382{387. Springer Berlin / Heidelberg, 2010.

[46] M. Majzoobi, G. Ghiaasi, F. Koushanfar and S. R. Nassif, "Ultra-low power current-based PUF," 2011 IEEE International Symposium of Circuits and Systems (ISCAS), 2011, pp. 2071-2074, doi: 10.1109/ISCAS.2011.5938005.

[47] L. Feiten, M. Sauer, and B. Becker, “On metrics to quantify the inter-device uniqueness of PUFs,” in TRUDEVICE Workshop, Dresden, March 2016, https://eprint.iacr.org/2016/320

[48] O. Rioul, P. Solé, S. Guilley and J. Danger, "On the entropy of Physically Unclonable Functions," 2016 IEEE Int. Symposium on Information *Theory (*ISIT*)*, 2016, pp. 2928-2932, doi: 10.1109/ISIT.2016.7541835.

[49] C. Marchand, L. Bossuet, U. Mureddu, N. Bochard, A. Cherkaoui and V. Fischer, "Implementation and Characterization of a Physical Unclonable Function for IoT: A Case Study With the TERO-PUF," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 1, pp. 97-109, Jan. 2018, doi: 10.1109/TCAD.2017.2702607.

[50] L. M. Reyneri, D. Del Corso and B. Sacco, "Oscillatory metastability in homogeneous and inhomogeneous flip-flops," in IEEE Journal of Solid-State Circuits, vol. 25, no. 1, pp. 254-264, Feb. 1990, doi: 10.1109/4.50312.

[51] L. Bossuet, X. Ngo, Z. Cherif and V. Fischer, "A PUF Based on a Transient Effect Ring Oscillator and Insensitive to Locking Phenomenon" in IEEE Transactions on Emerging Topics in Computing, vol. 2, no. 01, pp. 30-36, 2014. doi: 10.1109/TETC.2013.2287182

[52] J. Delvaux, “Security analysis of PUF-based key generation and entity authentication,” Ph.D. dissertation, Shanghai Jiao Tong Univ., Shanghai, China, 2017.

[53] Xilinx, “7 Series FPGAs Configurable Logic Block User Guide”, UG474 (v1.8) September 27, 2016

[54] Mahabub Hasan Mahalat, Suraj Mandal, Anindan Mondal, Bibhash Sen, Rajat Subhra Chakraborty, "Implementation Characterization and Application of Path Changing Switch based Arbiter PUF on FPGA as a lightweight Security Primitive for IoT", ACM Transactions on Design Automation of Electronic Systems, vol. 27, pp. 1, 2022.

[55] A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, https://www.nist.gov/publications/statistical-test-suite-random-and-pseudorandom-number-generators-cryptographic

[56] F. Wilde, B. M. Gammel and M. Pehl, "Spatial Correlation Analysis on Physical Unclonable Functions," in IEEE Trans. on Information Forensics and Security, vol. 13, no. 6, pp. 1468-1480, June 2018, doi: 10.1109/TIFS.2018.2791341.

[57] M. Majzoobi, F. Koushanfar and S. Devadas, "FPGA PUF using programmable delay lines," 2010 IEEE International Workshop on Information Forensics and Security, 2010, pp. 1-6, doi: 10.1109/WIFS.2010.5711471.

[58] F. Tehranipoor, N. Karimian, W. Yan and J. A. Chandy, "DRAM-Based Intrinsic Physically Unclonable Functions for System-Level Security and Authentication," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 3, pp. 1085-1097, March 2017, doi: 10.1109/TVLSI.2016.2606658.

[59] Lars Tebelmann and Jean-Luc Danger and Michael Pehl, “Self-Secured PUF: Protecting the Loop PUF by Masking,” Available: https://eprint.iacr.org/2020/145

[60] O. Rioul, P. Solé, S. Guilley and J. Danger, "On the entropy of Physically Unclonable Functions," 2016 IEEE Int. Symposium on Information Theory (ISIT), 2016, pp. 2928-2932, doi: 10.1109/ISIT.2016.7541835.

[61] *User Guide SmartFusion2 and IGLOO2 FPGA Security and Best Practices*, Microsemi, UG0443_V10.

[62] D. Yamamoto, K. Sakiyama, M. Iwamoto, K. Ohta, M. T. Ochiai, and K. Itoh, ``Uniqueness enhancement of PUF responses based on the location of random outputting RS latches," in Proc. Int. Conf. CHES, 2011, pp. 390.

[63] A. Koyily, C. Zhou, C. H. Kim and K. K. Parhi, "An entropy test for determining whether a MUX PUF is linear or nonlinear," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1-4, doi: 10.1109/ISCAS.2017.8050670.