32-bit and 64-bit CDC-7-XPUF Implementation on a Zynq-7020 SoC

Oğuz Yayla[1][0000−0001−8945−2780] and
Yunus Emre Yılmaz[1,2][0000−0001−5857−5446]

[1] Institute of Applied Mathematics, Middle East Technical University, 06800 Ankara, Turkey
[2] Aselsan Inc., Ankara, Turkey
oguz@metu.edu.tr, yeylmz@gmail.com

Abstract. Physically (Physical) Unclonable Functions (PUFs) are basic and useful primitives in designing cryptographic systems. PUFs are designed to facilitate device authentication, secure boot and firmware integrity, and secure communications. To achieve these objectives, PUFs must exhibit both consistent repeatability and instance-specific randomness. The Arbiter PUF, recognized as the first silicon PUF, is capable of generating a substantial number of secret keys instantaneously based on the input, all while maintaining a lightweight design. This advantageous characteristic makes it particularly well-suited for device authentication in applications with constrained resources, especially for Internet-of-Things (IoT) devices. Despite these advantages, arbiter PUFs are vulnerable to machine learning attacks. Hence, those arbiter PUF designs are improved to achieve increased resistance against such attacks. In this work, a machine-learning-resistant 32-bit and 64-bit component-differentially challenged XOR Arbiter PUF (CDC-XPUF) is implemented based on a design found in the literature. The system is implemented using the ZC702 Rev1.1 Evaluation Board, which features the Xilinx Zynq 7020 SoC, and utilizes a configuration involving three boards for experimental validation. The 32-bit and 64-bit 7-stream CDC-7-XPUFs are evaluated using PUF metrics in the literature, and the utilization ratio of both implementations is also presented. These improvements aim to increase resilience against machine learning attacks while maintaining usefulness and efficiency for IoT applications.

Keywords: PUF · Arbiter PUF · CDC-XPUF · SoC FPGA.

1 Introduction

A Physically Unclonable Function (PUF) is a mechanism that creates a unique relationship between a set of inputs (challenges) and outputs (responses) based on the complex physical properties of a system. This relationship is static and unique to each physical instance. PUFs can only be evaluated through their specific physical systems, and even identical circuits will have different responses due to manufacturing variations [1]. In this thesis, the focus is on silicon PUFs, which leverage timing and delay variations in integrated circuits. These variations arise from inconsistencies in the production process, even when circuits are made with the same design layout.

PUFs enhance security by generating secrets from the complex properties of physical systems, eliminating the need for storing them in memory. Another advantage is that PUFs do not require any specialized manufacturing processes or additional programming and testing steps. PUFs are typically compact and durable, making them ideal for use in devices like RFID tags, smart cards, and other low-cost Internet-of-Things (IoT) devices [2].

For hardware implementation, PUFs can be integrated into application-specific integrated circuits (ASICs) or Field-Programmable Gate Arrays (FPGAs). While ASICs may offer better performance, they are difficult to modify once designed. In contrast, FPGAs allow for flexible reconfiguration, which is particularly useful in hardware development. Modern System-on-Chip (SoC) FPGAs combine programmable logic with processor cores, offering benefits like higher integration, lower power consumption, smaller sizes, and faster communication between the processor and FPGA.

The Arbiter PUF, the first silicon-based PUF, generates numerous secret keys efficiently from input data while maintaining a lightweight design. This makes it well-suited for device authentication in environments with limited resources, such as IoT applications. However, its vulnerability to machine learning attacks highlights the need for enhanced design solutions to improve security.

Consequently, in order to improve resistance to machine learning (ML) attacks, arbiter PUF designs have been enhanced. In this study, an ML attack-resistant component-differentially challenged XOR arbiter PUF (CDC-XPUF) is implemented, following the reference designs from [3] and [4]. The implementation utilizes the ZC702 Rev1.1 Evaluation Board [5], equipped with the Xilinx Zynq 7020 SoC, and a configuration of three such boards for experimental validation. Research in [4] demonstrates that designs with 64-bit or longer challenges and at least 7-stream PUFs are resistant to the most advanced ML attack techniques. Consequently, this work implements a referenced 32-bit CDC-7-XPUF, followed by an improved 64-bit version for enhanced ML attack resilience. The performance results for both the 32-bit and 64-bit CDC-7-XPUFs are presented and compared to the reference design. Additionally, the utilization rates of these designs are evaluated, showing that they are well-suited for IoT systems by providing sufficient space for other software or firmware.

Our work presents two primary contributions:

- We implement the referenced 32-bit CDC-7-XPUF detailed in [3] and [4], and also the machine learning attack-resistant 64-bit version of CDC-7-XPUF. By using the evaluation metrics of steadiness, correctness, diffuseness, uniformity, and uniqueness, which are presented in the literature to evaluate PUFs, it is shown that both PUF design has good scores to use in IoT systems.

- The utilization ratio of these 32-bit and 64-bit CDC-7-XPUF designs are suitable to use in any IoT systems since they provide sufficient space for other software or firmware.

The paper is organized as follows. Section 2 provides the basic background information to explain how both 32-bit and 64-bit CDC-7-XPUF works. In Section 3, the implementation details of both CDC-7-XPUFs are explained. In Section 4, the test results of PUF implementations are presented and compared with the referenced design. In the end, Section 5 concludes the paper and states our future works.

2 Background Information About 32-bit and 64-bit CDC-7-XPUF

2.1 Basics of PUFs

PUF extracts entropy from the physical characteristics of an integrated circuit (IC). Each chip exhibits variations due to the inherent unpredictability in the manufacturing process. PUFs harness static entropy from the fluctuations in the manufacturing process. Once the chip is fabricated, the disparities in the manufacturing process become consolidated and undergo minimal changes throughout the chip's lifespan. Consequently, this form of entropy is termed static entropy [6].

Basically, a PUF generates a sequence (response) of the unique signature by input initial states (challenge), so-called challenge-response pairs (CRPs). Each PUF can be represented as a black box, $R = f(C)$, as illustrated in Fig. 1, where the $f()$ is secret [6].

Challenge
PUF
Response
R = f(C)

Fig. 1. Generic PUF model [6]

In the literature, there are various types of PUFs, and they can be classified with respect to their entropy sources and their CRPs [7]. In this research, an intrinsic and delay-based strong PUF, named Arbiter PUF, is implemented.

It is important to note that the Arbiter PUF is a strong PUF. A Strong PUF can generate a vast number of challenge-response pairs (CRPs), making it impractical to read all possible CRPs within a reasonable timeframe. This property makes them suitable for applications requiring high security due to their extensive challenge-response space.

2.2 Types of Arbiter PUFs

2.2.1 Basic Arbiter PUF

An Arbiter Physical Unclonable Function (APUF) is a robust PUF relying on delay, featuring a race condition between two symmetrical digital paths. In each delay stage, two multiplexers (MUXes) are incorporated, and their operation is governed by challenges (C_0 C_{n-1}).

Upon activation, the APUF initiates its operation with a trigger signal. This signal traverses two paths determined by a pre-input challenge, ultimately reaching an arbiter. The arbiter then determines which of the two paths is faster in generating the binary response that aligns with the black-box model ($R = f(C)$), as it is illustrated in Fig. 1, where C is the challenge and R is the response.

2.2.2 XOR Arbiter PUF (XOR-PUF)

Due to the limited resistance of arbiter PUFs against machine learning modeling attacks, a new PUF design was introduced in [1]. This new design incorporates a non-linear XOR gate into multiple arbiter PUFs to generate the final response and is referred to as the XOR arbiter PUF. An n-XOR-PUF consists of n-component arbiter PUFs (also known as streams or sub-challenges), wherein the responses from all n-component arbiter PUFs are XORed together at the XOR gate to produce a single-bit response. It is important to note that all component arbiter PUFs in an XOR-PUF are supplied with the same challenge bits [4].

2.2.3 Component-Differentially Challenged XOR-PUF (CDC-XPUF)

Component-differentially challenged XOR-PUF (CDC-XPUF) and XOR-PUF share a similar architecture, comprising multiple arbiter PUF components and XOR gates. The key distinction between CDC-XPUF and XOR-PUF lies in the challenge inputs: each component arbiter PUF in a CDC-XPUF receives different challenge inputs, whereas all component arbiter PUFs in an XOR-PUF receive the same challenges [4]. Fig. 2 illustrates the structure of CDC-XPUF.

Studies [8], [9], [10], [11] indicate that applying different challenges to the components of an XOR-PUF can mitigate its vulnerability to ML modeling attacks. Existing ML attack methods on 64-bit CDC-XPUFs with four components achieve a success rate of less than 90% even when utilizing over one million challenge-response pairs (CRPs). Experimental results consistently demonstrate that CDC-XPUFs with four or more components are either unbreakable or prohibitively expensive to breach with current attack methods. Consequently, CDC-XPUFs are considered strong candidates in terms of security performance [4].

In order to generate different challenge bits in [4], a pseudorandom number generator (PRNG) structure is proposed as follows:

$$C_{n+1} = (a * C_n + g) \;\; mod\, m \tag{1}$$

where C is the sequence of the generated random number, a is a multiplier, g is a given constant, and m is 2^K, where K is the number of stages.

2.3 Evaluation Metrics of PUFs

This section outlines a set of PUF characteristics to evaluate the suitability of a PUF design for security applications. Certain statistical properties, such as

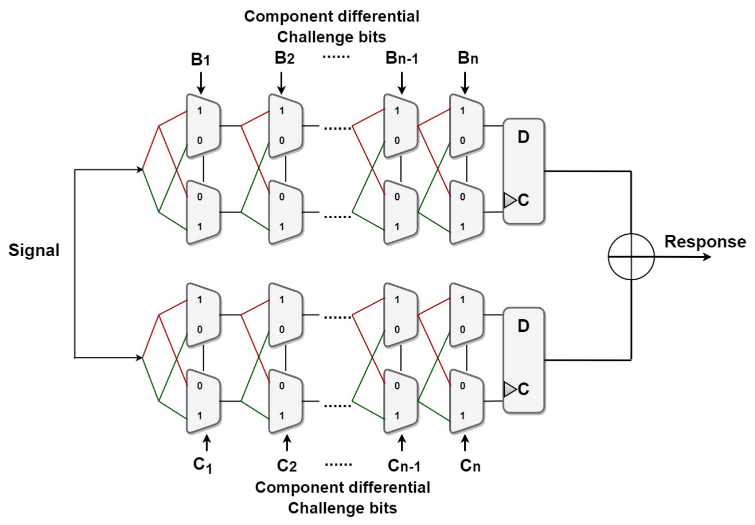

Fig. 2. A CDC-XPUF with 2 sub-streams and n bits of each stream [4]

stability, correctness, diffuseness, uniformity, and uniqueness, can be empirically demonstrated through silicon-based experimentation. Other attributes, including the security vulnerability of PUFs, require computational analysis for thorough assessment.

The first section explains how implemented PUFs are not vulnerable to machine learning (ML) attacks.

In the subsequent chapters following the initial chapter, the evaluation criteria studied and constructed by either Hori et al. [12] or Maiti et al. [13] are explained. They are grouped with respect to three different properties of the responses, and these groups are listed below and explained in detail in the following sections.

The metrics in the first and the second groups evaluate the responses of the same PUFs, although the metrics in the third group evaluate how the responses vary between different devices.

The quality of random numbers is pivotal in cryptography, necessitating a thorough evaluation of their properties. While Hori et al. [12] defines the randomness metric, Maiti et al. [13] defines the uniformity metric. In this work, we think that the uniformity metric is more suitable to use. Because, although randomness in Hori et al. [12] indicates that randomness is evaluated, only some kind of uniformity is evaluated as in [13]. This choice can be understood better by the explanation in Section 2.3.4. In addition to these, as indicated in [14], in general, how to determine the exact entropy of the PUF responses is another very important open research problem. Consequently, for the PUF implementation, only the uniformity and diffuseness metrics are used to evaluate entropy.

2.3.1 Resistance to Machine Learning (ML) Attacks

PUFs are considered secure due to their inherently unclonable architecture. However, several successful studies have demonstrated that PUFs can be mathematically cloned using the additive delay model [15]. Additionally, if adversaries gain access to a sufficient number of silicon CRPs, PUFs may become susceptible to machine learning attacks, as explained in [16], [17], [18], [19]. Therefore, it is imperative for users to ensure that PUFs are resistant to all forms of attacks before deploying them in practical applications.

The study in [4], a comprehensive evaluation of the security of CDC-XPUFs against advanced ML attack methods, utilizing problem-specific parameter values, was conducted to assess the robustness of CDC-XPUFs. Compared to previously reported findings, their study uncovered vulnerabilities in the CDC-XPUF with PUF circuit parameter configurations that were previously not considered insecure. Specifically, they successfully compromised 64-bit CDC-6-XPUFs using approximately 100 million simulated CRPs, and 64-bit CDC-5-XPUFs with 4.5 million simulated CRPs or 2.5 million silicon CRPs. Additionally, they managed to break 128-bit CDC-5-XPUFs with 40 million simulated CRPs, instances that had previously been considered resistant to any existing ML attack methods. Notably, the method in [4] was able to break 64-bit CDC-4-XPUFs using only around 80,000 CRPs, significantly fewer than those used in earlier studies. On the other hand, it also demonstrates that the security of CDC-XPUFs improves substantially as the number of component PUFs increases, with 64-bit CDC-XPUFs featuring seven components proving entirely resilient to the two ML attack methods employed. This finding is particularly encouraging for the IoT security community, as many CDC-XPUFs remain secure, especially those with 64-bit or longer challenges and seven or more component PUFs, which are resistant to the most advanced ML attack methods developed to date. Consequently, the experimental attack study in [4] redefines the boundary between secure and insecure regions within the PUF circuit parameter space, offering valuable insights to PUF manufacturers and IoT security developers for refining the protocols of CDC-XPUF-based applications and mitigating potential risks.

2.3.2 Reliability of Responses From the Same PUFs

PUF responses must be reliable and trusted in real-world applications. A PUF is considered reliable if it consistently generates the same response when the same challenge is applied to the same device. Several factors can affect the reliability of these responses, particularly changes in the operating environment. These factors include, but are not limited to, ambient temperature, humidity, the junction temperature of the circuit, power supply voltage, and circuit aging.

In this work, the environmental variances listed above have not been changed. We have worked at an ambient room temperature of approximately 27^{o}C, stable humidity, and stable core voltage of Zynq SoC.

In terms of the reliability of responses from the same PUFs, steadiness, and correctness are examined in this section.

Steadiness

Steadiness is a reliability metric that is defined by Hori et al. [12]. When generating the same responses multiple times on the same device, it is expected that all responses must be identical. Steadiness indicates how stably a PUF outputs the same responses to the same challenge sets. The steadiness result is 1 if there are no changes in the responses that were recorded during the experiment. Steadiness can be calculated as follows:

$$S = 1 + \frac{1}{N_c}\sum_{k=1}^{N_c}\log_2 max\{\frac{\sum_{j=1}^{N_a} b_{k,j}}{N_a}, 1 - \frac{\sum_{j=1}^{N_a} b_{k,j}}{N_a}\} \tag{2}$$

where N_c denotes the number of different challenges used, N_a denotes the number of times each challenge is applied, and $b_{k,j}$ denotes the j-th response among all N_a responses to the k-th challenge in the set of all N_c challenges. The stable CRPs that pass the steadiness test are known as "Correct ID" [3].

Correctness

This metric is defined by Hori et al. [12] and is almost the same metric as reliability, which is defined by Maiti et al. [13]. The only difference between their equations is the normalization factor. Correctness is normalized by the maximum value of the Fractional Hamming Distance of the responses, while reliability is normalized by the average. Hence, we only computed the correctness value and ignored the reliability. The ideal value of the correctness is 1, which can be calculated as follows:

$$C = 1 - \frac{2}{N_c \times N_a}\sum_{k=1}^{N_c}\sum_{j=1}^{N_a}(b_k \oplus b_{k,j}) \tag{3}$$

where b_k is the "Correct ID". This "Correct ID" is determined by the majority voting of all of the giving responses for the input challenge. N_c is the number of challenges in the dataset. $b_{k,j}$ is the response of the j-th response in the set of all N_a responses to the k-th challenge.

2.3.3 Entropy of Responses From the Same PUFs

A PUF is considered uniform if it generates an equal distribution of zeros and ones in response to a set of challenges. This characteristic is particularly desirable in block and stream cipher processes, as repeated patterns in secret keys are deemed detrimental. In terms of entropy, Hori et al. [12] introduced the diffuseness metric, while Maiti et al. [13] proposed the uniformity metric. Given the close resemblance between Hori's [12] randomness metric and Maiti's [13] uniformity metric, only the uniformity metric is assessed in this context.

Diffuseness

The diffuseness metric, introduced by Hori et al. [12], is an intra-chip metric that assesses the variability of a PUF's responses to different challenges. A PUF is considered to exhibit diffuseness if it produces distinct responses for distinct challenges; for instance, the response to a specific challenge X should differ from the responses generated by other challenges. Diffuseness is quantified by calculating the fractional Hamming distance between the responses produced by the same device in response to a set of challenges. The diffuseness can be computed using the following formula:

$$D = \frac{4}{K^2 \times L} \sum_{l=1}^{L} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} (b_{i,l} \oplus b_{j,l}) \tag{4}$$

where L is the responses' length, counted in bits, and K is the number of such multi-bit responses used in the experimental study.

Uniformity

The uniformity, which was introduced by Maiti et al. [13], of a PUF measures the degree of zeros and ones in the produced responses. Its ideal value is 0.5. The uniformity can be calculated as follows:

$$U = \frac{1}{N_r} \sum_{i=1}^{N_r} b_i \tag{5}$$

where N_r is the response length in a set, and bi is the i-th response bit.

It is stated that the randomness metric, defined by Hori et al. [12], is not used for the evaluation since it is very similar to the uniformity. In order to make this statement more clear, the equations to calculate the randomness are provided below:

$$H = -\log_2 max(p, 1-p) \tag{6}$$

where p is the frequency of '1' in the response set given by:

$$p = \frac{1}{N_r} \sum_{i=1}^{N_r} b_i \tag{7}$$

where N_r is the response length in a set, and bi is the i-th response bit.

It is obvious that the Equation 5 and 7 are nearly the same. Hence, it makes our decision more clear.

2.3.4 Fingerprint Property

Uniqueness

The uniqueness was introduced by Maiti et al. [13], and it can be calculated using a Hamming Distance between two devices' responses. The calculation is as follows:

$$U_k = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{HD(ID_i, ID_j)}{L} \tag{8}$$

where ID_i and ID_j are two L-bit responses of a PUF installed on two different chips (the i-th and j-th chip) to the k-th challenge repeatedly applied L times. The ideal value of the Maiti's uniqueness [13] is 0.5.

In addition to these metrics, the resource utilization rate is a metric for both TRNGs and PUFs.

3 32-bit and 64-bit CDC-7-XPUF Implementation Details

In this study, our aim is to implement an ML-resistant PUF with good cryptographic properties explained in Section 2.3.2 - 2.3.4. As it is stated in [4], 64-bit CDC-7-XPUF is ML-resistant. However, in [4], the cryptographic properties are examined. These are examined in [3], but only for a maximum of 32-bit CDC-7-XPUF. Hence, we decided that firstly, we implemented 32-bit CDC-7-XPUF and showed that the design satisfies good cryptographic properties, as the referenced PUF design does. After that, we implemented the 64-bit, in other words, ML-resistant version of CDC-7-XPUF. In this chapter, we present all of our results with respect to the metrics explained in Section 2.3.2 - 2.3.4 and compare our results to the referenced design [3].

The implementation details of CDC-XPUF are given in both [3] and [4] and also explained in Section2.2.3. The results of the implementation and comparisons with the previous works are presented in Section 4.

The MUX-based CDC-XPUF arbiter structures are implemented using Vivado 2019.1 [20] in VHDL [23].

Since the CDC-XPUF is a delay-based PUF, relying on the calculation of delays incurred by the internal gates and interconnections, the correct placement of its components is crucial. To ensure equal delay lines, the top and bottom of each stage in the CDC XPUF must be precisely aligned in the placement phase of SoC design in Vivado 2019.1.

For generating different challenges for different stages, a PRNG is proposed in Equation 1. Obviously, two PRNGs with two different parameter sets are used for the 32-bit and 64-bit designs.

The implementation setup illustrated in Figure 3 is used. The software developed in Python [22] using Visual Studio 2022 [21] is utilized to calculate the scores for the five evaluation metrics, which are detailed in Section 2.3 from the generated bitstreams.

Using the setup in Figure 3, for the statistical characteristics CRPs, we generated up to 16,000 (challenges) ×32 (iterations) ×128 (response length) ×3 (Zynq 7020 SoCs) CRPs out of each design. The repetition of the CRPs is needed to study the statistical characteristics and investigate related metrics such as correctness and steadiness. The CRPs were captured at an ambient temperature of approximately 27°C, and the core voltage was set to 1.0V. The ambient temperature does not reflect the temperature of the chip, which has changed as long as the experiments continue. Through a dual-access BRAM, CRPs are sent to the PS part. From the PS part via UART, the CRPs are sent to the PC with a baud rate of 230,400 bits/second between the PuTTY [24] terminal and the SoCs.

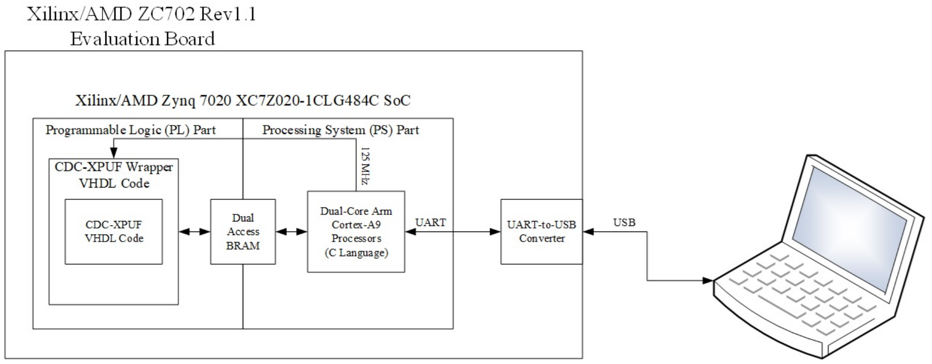
Xilinx/AMD ZC702 Rev1.1
Evaluation Board
Xilinx/AMD Zynq 7020 XC7Z020-1CLG484C SoC
Programmable Logic (PL) Part
Processing System (PS) Part
CDC-XPUF Wrapper VHDL Code
CDC-XPUF VHDL Code
Dual Access BRAM
125 MHz
Dual-Core Arm Cortex-A9 Processors (C Language)
UART
UART-to-USB Converter
USB

Fig. 3. Block Diagram of Implementation Setup of CDC-7-XPUFes

4 32-bit and 64-bit CDC-7-XPUF Experimental Results and Comparisons

The evaluation metrics of PUFs are explained in Section 2.3. As explained in Section 2.3, respectively, the implementation results of steadiness, correctness, diffuseness, uniformity, and uniqueness are presented in Table 1 for the referenced 32-bit CDC-7-XPUF in [3], our implemented 32-bit CDC-7-XPUF, and 64-bit CDC-7-XPUF. In the following sections, the results in this table are analyzed.

4.1 Steadiness

In Equation 2, the steadiness score is calculated between 0 and 1. Hence, we normalize it using percentages to calculate the score in Table 1. As stated before, 32 iterations of 128-bit responses are generated. In the steadiness calculation, these 4096-bit long responses are used.

Our result for the 32-bit is only slightly worse than the reference design in an acceptable range. Also, the 64-bit result is slightly worse than the 32-bit results. As it can be seen in [3], increasing the stage number has a negative effect on the steadiness.

Table 1. Results of CDC-XPUFs with Respect to Evaluation Metrics

Evaluation Metric	**Score of Referenced Work 32-bit CDC-7-XPUF [3]**	**Score of 32-bit CDC-7-XPUF Implementation**	**Score of 64-bit CDC-7-XPUF Implementation**
Steadiness	98.18%	97.09%	96.70%
Correctness	97.63%	96.64%	96.19%
Diffuseness	99.90%	99.96%	99.99%
Uniformity	50.40%	50.94%	49.89%
Uniqueness	17.90%	18.06%	18.96%

4.2 Correctness

In Equation 3, the correctness score is calculated between 0 and 1. Hence, we normalize it using percentages to calculate the score in Table 1.

Our result for the 32-bit is only slightly worse than the reference design in an acceptable range. Also, the 64-bit result is slightly worse than the 32-bit results. As it can be seen in [3], increasing the stage number has a negative effect on the correctness.

4.3 Diffuseness

In Equation 4, the diffuseness score is calculated between 0 and 1. Hence, we normalize it using percentages to calculate the score in Table 1. For the diffusion calculation, "Correct ID"s are used.

Our result for the 32-bit is only slightly better than the reference design. Also, the 64-bit result is slightly better than the 32-bit results. As it can be seen in [3], increasing the stage number has a positive effect on the diffuseness.

4.4 Uniformity

In Equation 5, the uniformity score is calculated between 0 and 1, whose expected score is 0.5. Hence, we normalize it using percentages to calculate the score in Table 1. For the uniformity calculation, "Correct ID"s are used. In [3], the results of the uniformity are not direct, yet they can be inferred from the results of the randomness, as it can be seen from Equations 5, 6, and 7. However, the result of this inference is ambiguous. Since it can not be known that p or $1-p$ is greater, the result in Table 1 for the referenced 32-bit work can be 49.60% also. But it is not important. Because, in terms of uniformity, the proximity of the value to 50% is important, not the percentage value. Hence, in terms of proximity, 50.40% and 49.60% are the same. Consequently, that approach is applied to the comparison in the following paragraph.

Our result for the 32-bit is only slightly worse than the reference design. However, the 64-bit result is slightly better than the result of our 32-bit design and the referenced 32-bit design. Although, as it can be seen in [3], increasing the stage number has a negative effect on the uniformity, in our case, it increased the uniformity.

4.5 Uniqueness

In Equation 8, the uniqueness score is calculated between 0 and 1. Hence, we normalize it using percentages to calculate the score in Table 1. For the uniqueness calculation, "Correct ID"s are used.

Our result for the 32-bit is only slightly better than the reference design. Also, the 64-bit result is slightly better than the 32-bit results. As it can be seen in [3], increasing the stage number has a positive effect on the uniqueness.

4.6 Utilization Results of CDC-7-XPUFs in Zynq-7020 SoC FPGA

For the implementation, we use state machines in the PL part so that we can take the challenges from the PS part, and we can send responses derived from these challenges through the Dual Access BRAM. Although the PL part consists of not only CDC-7-XPUFs but also state machines which are necessary for the implementation, the utilization rate is relatively low, as it can be seen in Table 2 for both 32-bit and 64-bit CDC-7-XPUF implementations.

As expected, the 64-bit design has a higher utilization rate, especially in DSP resources. In order to generate different challenges for each of the streams, we use PRNGs, which multiply 64-bit numbers requiring more DSP than 32-bit design. That relatively low utilization result makes 64-bit CDC-7-XPUF a promising candidate for applications that require a PUF.

Table 2. Utilization Table Generated Using Vivado 2019.1 [20] for 32-bit and 64-bit CDC-7-XPUF Implementations

Resource Type	Avaliable Resource Quantity	Utilization Quantity (Utilization Rate as %) of 32-bit CDC-7-XPUF	Utilization Quantity (Utilization Rate as %) of 64-bit CDC-7-XPUF
LUT	53200	1500 (2.82%)	1740 (3.27%)
LUTRAM	17400	72 (0.41%)	72 (0.41%)
FF	106400	1781 (1.67%)	1933 (1.82%)
BRAM	140	2 (1.43%)	2 (1.43%)
DSP	220	12 (5.45%)	68 (30.91%)
IO	200	8 (4.00%)	4 (100.00%)

5 Conclusion and Future Works

We have thoroughly examined the resilience against machine learning attacks in Section 2.3.1. As discussed in this section and demonstrated in [3], the 64-bit CDC-XPUF designs with 7 streams are resistant to machine learning attacks.

In the following part, evaluation criteria are examined. The PL part of Zynq 7020 has a very similar architecture to Artix-7, which is used to implement the reference design of CDC-XPUF in [3]. Hence, we expected similar results found in [3], and, as expected, we observed similar results as can be seen in the previous sections.

In the last part, the utilization rate of both 32-bit and 64-bit designs are examined, and it is shown that both designs are suitable for an IoT application since they provide a lot of space in the PL part.

As the future works, the following two items can be listed:

- CDC-7-XPUF designs will be implemented in different FPGAs and SoCs using the evaluation criteria given in this work.
- CDC-7-XPUF designs will be tested in various environmental conditions such as varying temperature and varying voltage SoC core voltages.

References

1. Suh, G. E., Devadas, S.: Physical Unclonable Functions for Device Authentication and Secret Key Generation, in 2007 44th ACM/IEEE Design Automation Conference, pp. 9–14 (2007)
2. Ebrahimabadi M., Younis M., Lalouani W., Karimi N.: A Novel Modeling- Attack Resilient Arbiter-PUF Design, in 2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID), pp. 123–128 (2021)
3. Mursi K. T.: From XOR PUF to CDC XOR PUF: Cost-Effectiveness, Statistical Characteristics, and Security Assessment, Ph.D. Thesis (2021)
4. Li G., Mursi K. T., Aseeri A. O., Alkatheiri M. S., Zhuang Y.: A New Security Boundary of Component Differentially Challenged XOR PUFs Against Machine Learning Modeling Attacks, International Journal of Computer Networks & Communications, vol. 14, p. 3 (2022). https://aircconline.com/ijcnc/V14N3/14322cnc01.pdf
5. Xilinx Zynq-7000 SoC ZC702 Evaluation Kit, https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html.
6. Cao Y., Liu W., Qin L., Liu B., Chen S., Ye J., Xia X., Wang C.: Entropy Sources Based on Silicon Chips: True Random Number Generator and Physical Unclonable Function, Entropy, 24(11), ISSN 1099-4300 (2022)
7. Srinivas M. B. R., Elango K.: "Era of Sentinel Tech: Charting Hardware Security Landscapes Through Post-Silicon Innovation, Threat Mitigation and Future Trajectories," in IEEE Access, vol. 12, pp. 68061-68108 (2024). doi: https://doi.org/10.1109/ACCESS.2024.3400624
8. Mursi K. T., Zhuang Y.: Experimental Examination of Component-Differentially-Challenged XOR PUF Circuits, 1729(1), p. 012006 (2021)
9. Mursi K. T., Thapaliya B., Zhuang Y., Aseeri A. O., Alkatheiri M.S.: A Fast Deep Learning Method for Security Vulnerability Study of XOR PUFs, Electronics, 9(10), ISSN 2079-9292 (2020)
10. Wisiol N., Becker G. T., Margraf M., Soroceanu T. A. A., J. Tobisch, Zengin B.: Breaking the Lightweight Secure PUF: Understanding the Relation of Input Transformations and Machine Learning Resistance, Cryptology ePrint Archive, Paper 2019/799 (2019) https://eprint.iacr.org/2019/799
11. Yu M.-D., Hiller M., Delvaux J., Sowell R., Devadas S., Verbauwhede I. M. R.: A Lockdown Technique to Prevent Machine Learning on PUFs for Lightweight Authentication, IEEE Transactions on Multi-Scale Computing Systems, 2, pp. 146–159 (2016)
12. Hori Y., Yoshida T., Katashita T., Satoh A.: Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs, 2010 International Conference on Reconfigurable Computing and FPGAs, Cancun, Mexico, pp. 298-303 (2010). https://doi.org/10.1109/ReConFig.2010.24
13. Maiti, A., Gunreddy, V., Schaumont, P.: A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. In: Athanas, P., Pnevmatikatos, D., Sklavos, N. (eds) Embedded Systems Design with FPGAs. Springer, New York, NY. (2013). https://doi.org/10.1007/978-1-4614-1362-2_11.
14. Anandakumar N. N., Hashmi M., Tehranipoor M.: FPGA-based Physical Unclonable Functions: A Comprehensive Overview of Theory and Architectures, Integration, 81, 07 (2021)

15. Lim D.: Extracting Secret Keys from Integrated Circuits, Master Thesis, Massachusetts Institute of Technology (2004)
16. Alamro M.A., Mursi K.T., Zhuang Y., Aseeri A.O., Alkatheiri M.S.: Robustness and Unpredictability for Double Arbiter PUFs on Silicon Data: Performance Evaluation and Modeling Accuracy. Electronics 2020, 9, 870 (2020) https://doi.org/10.3390/electronics9050870.
17. Alkatheiri M. S., Zhuang Y.: Towards Fast and Accurate Machine Learning Attacks of Feed-Forward Arbiter PUFs, 2017 IEEE Conference on Dependable and Secure Computing, Taipei, Taiwan, pp. 181-187 (2017). https://doi.org/10.1109/DESEC.2017.8073845
18. Aseeri A. O., Zhuang Y., Alkatheiri M. S.: A Machine Learning-Based Security Vulnerability Study on XOR PUFs for Resource-Constraint Internet of Things, 2018 IEEE International Congress on Internet of Things (ICIOT), San Francisco, CA, USA, pp. 49-56 (2018). https://doi.org/10.1109/ICIOT.2018.00014
19. Mursi K. T., Zhuang Y., Alkatheiri M. S., Aseeri A. O.: Extensive Examination of XOR Arbiter PUFs as Security Primitives for Resource-Constrained IoT Devices, 2019 17th International Conference on Privacy, Security and Trust (PST), Fredericton, NB, Canada, pp. 1-9 (2019). https://doi.org/10.1109/PST47121.2019.8949070
20. Xilinx (AMD) Vivado and SDK 2019.1 Design Software for Xilinx (AMD) Adaptive SoCs and FPGAs, https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/archive.html.
21. Microsoft Corporation, Visual Studio 2022 (2022). https://visualstudio.microsoft.com/ Accessed: 2023-01-12.
22. Python Software Foundation, Python Language Reference, version 3.x. Accessed:2023-01-12 https://www.python.org/Accessed:2023-01-12.
23. IEEE Computer Society, IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-2008 (2008)
24. PuTTY - a free and open-source terminal emulator, serial console, and network file transfer application. https://www.putty.org/, accessed: 2023-12-25.