# 32-bit and 64-bit CDC-7-XPUF Implementations 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 (or Physical) Unclonable Functions (PUFs) are basic and useful primitives in designing cryptographic systems. PUFs are designed to facilitate device authentication, secure boot, firmware integrity, and secure communications. To achieve these objectives, PUFs must exhibit both consistent repeatability and instance-specific randomness. The Arbiter PUF (APUF), 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, APUFs are vulnerable to machine learning (ML) attacks. Hence, those APUF designs were improved to achieve increased resistance against such attacks while maintaining usefulness and efficiency for IoT applications, and Component-Differentially Challenged XOR Arbiters (CDC-XPUFs) were proposed. In this work, ML-resistant 32-bit and 64-bit implementations of the Component-Differentially Challenged XOR Arbiter PUF with 7-stream (CDC-7-XPUF) are carried out. These CDC-7-XPUFs are evaluated using PUF metrics from the literature, and the resource utilization ratios of both implementations are also presented. The implementation setup contains the ZC702 Rev1.1 Evaluation Board, featuring the Xilinx Zynq-7020 SoC, and utilizes a configuration involving three boards for experimental validation.

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

## 1 Introduction

A Physically (or Physical) 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, as explained in [1]. In this

research, 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 radio frequency identification (RFID) tags, smart cards, and other low-cost Internet-of-Things (IoT) devices, as described in [2]. More detailed information about PUFs can be found in [5].

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. Furthermore, improvements or modifications in PUF algorithms can be implemented in SoCs or FPGAs more easily than in ASICs, primarily because such changes are not affordable in ASICs. Therefore, an SoC is chosen for the PUF implementation in this work due to these reasons.

The Arbiter PUF (APUF), 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 [6], 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.

### Contribution of this work

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. Then, we evaluate them by using the evaluation metrics of steadiness, correctness, diffuseness, uniformity, and uniqueness. It is shown that both PUF designs have good scores to use in IoT systems, see our summary in Table 1.
- It is also demonstrated that 32-bit and 64-bit CDC-7-XPUF designs can be implemented efficiently in terms of space on an SoC FPGA. Our implementation for designs consumes low amount of resourse in the FPGA, so they leave enough space for other software or firmware. The utilization ratio of these designs in our implementation given in Table 2 shows that these designs are suitable to use in any IoT systems.

**Outline**

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, as described in [7].

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, as stated in [7].


**Fig. 1.** Generic PUF model [7]

In the literature, there are various types of PUFs, and they can be classified with respect to their entropy sources and their CRPs, as explained in [8]. In this research, an intrinsic and delay-based strong PUF, named Arbiter PUF (APUF), is implemented. It is important to note that the APUF is a strong PUF. A strong PUF can generate a vast number of 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 (APUF)

An APUF, which was first defined in [9], 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 APUFs against machine learning modeling attacks, a new PUF design was introduced in [1]. This new design incorporates a non-linear XOR gate into multiple APUFs to generate the final response and is referred to as the XOR arbiter PUF. An n-XOR-PUF consists of n-component APUFs (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 APUFs in an XOR-PUF are supplied with the same challenge bits, as stated in [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 APUF components and XOR gates. The key distinction between CDC-XPUF and XOR-PUF lies in the challenge inputs: each component APUF in a CDC-XPUF receives different challenge inputs, whereas all component APUFs in an XOR-PUF receive the same challenges, as expressed in [4]. Fig. 2 illustrates the structure of CDC-XPUF with 2 sub-streams and n-bit of each stream, which is named n-bit CDC-2-XPUF.

Studies [10], [11], [12], [13] indicate that applying different challenges to the components of an XOR-PUF can mitigate its vulnerability to ML modeling

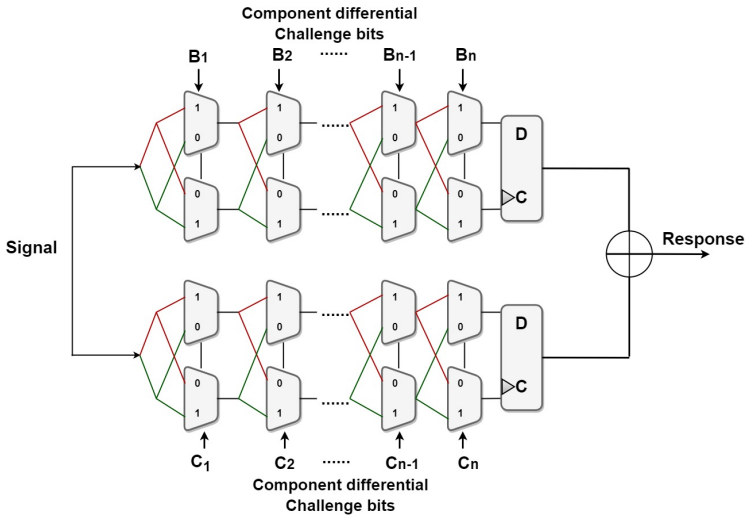


**Fig. 2.** An n-bit CDC-2-XPUF [4]

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, as stated in [4].

In order to generate different challenge bits, a pseudorandom number generator (PRNG) structure is proposed in [4] 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 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 following section explains how implemented PUFs are not vulnerable to ML attacks. In the subsequent subsections after the initial one, the evaluation criteria developed by either Hori et al. [14] or Maiti et al. [15] are

explained. The responses are categorized based on three distinct properties: the reliability and entropy of the responses generated by the same PUFs, as well as the fingerprint characteristic. 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. [14] defines the randomness metric, Maiti et al. [15] 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. [14] indicates that randomness is evaluated, only some kind of uniformity is evaluated as in [15]. This choice can be understood better by the explanation in Section 2.3.4.

### 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, as explained in [17]. Additionally, if adversaries gain access to a sufficient number of silicon CRPs, PUFs may become susceptible to machine learning attacks, as explained in [18], [19], [20], [21]. 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.

As a result, due to the reasons explained in this section, the 64-bit CDC-7-XPUF, whose resistance to ML attacks has been demonstrated in the study by Li et al. [4], has been implemented in this work.

### 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. [14]. 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:

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

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*, as defined in [3].

***Correctness***

This metric is defined by Hori et al. [14] and is almost the same metric as reliability, which is defined by Maiti et al. [15]. 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:

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

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. [14] introduced the diffuseness metric, while Maiti et al. [15] proposed the uniformity metric. Given the close resemblance between Hori's [14] randomness metric and Maiti's [15] uniformity metric, only the uniformity metric is assessed in this context.

***Diffuseness***

The diffuseness metric, introduced by Hori et al. [14], 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 $C_i$ 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:

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

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. [15], 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:

$$U = \frac{1}{N_r} \sum_{i=1}^{N_r} b_i, \tag{2}$$

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

The randomness metric, defined by Hori et al. [14], 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 by Hori et al. [14] are provided below:

$$H = -\log_2 max(p, 1-p), \tag{3}$$

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

$$p = \frac{1}{N_r} \sum_{i=1}^{N_r} b_i, \tag{4}$$

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

It is obvious that the Equations (2) and (4) are nearly the same. These two equations define the same thing actually, and it is the uniformity of the responses. Hori et al. [14] claim that taking this uniformity and using them in (3) calculates the randomness. The approach presented by Hori et al. [14] is not suitable for accurately calculating randomness. Equation (3) can only provide information regarding the percentage distribution of 0s and 1s, which is already captured in the uniformity metric proposed by Maiti et al. [15] in Equation (2). Thus, using this equation does not contribute to a deeper understanding of randomness beyond what uniformity already indicates. As stated in [16], how to determine the exact entropy of the PUF responses is another very important open research problem. Hence, in order to evaluate entropy, we use the uniformity metric introduced by Maiti et al. [15].

#### 2.3.4 Fingerprint Property

***Uniqueness***

The uniqueness was introduced by Maiti et al. [15], and it can be calculated using the Hamming distance between two devices' responses. The calculation is as follows:

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

where $\mathrm{ID}_i$ and $\mathrm{ID}_j$ 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 [15] is 0.5.

Besides these metrics, the resource utilization rate serves as an important indicator for evaluating the suitability of PUFs for IoT systems. If a PUF demands substantial resources within an SoC or FPGA, it is considered unsuitable for integration into IoT environments.

## 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 2.3.1 - 2.3.4. Based on the explanations provided in Section 2.3.1, the 64-bit CDC-7-XPUF is considered resistant to ML attacks and is utilized accordingly. The five evaluation metrics named *steadiness*, *correctness*, *diffuseness*, *uniformity*, and *uniqueness* are explained between Section 2.3.2-2.3.4. 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. Obviously, we also calculated the metrics for this version of the PUF.

In this work, the MUX-based CDC-XPUF arbiter structures are implemented using Vivado 2019.1 [22] in VHDL [25].

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. Hence, we placed the stages carefully by considering this equal delay line approach.

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 Fig. 3 is used. The software developed in Python [24] using Visual Studio 2022 [23] is utilized to calculate the scores for the five evaluation metrics from the generated bitstreams.

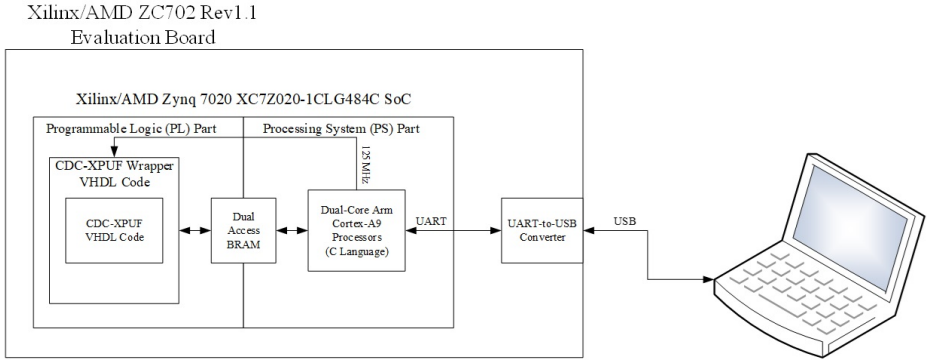


**Fig. 3.** Block Diagram of Implementation Setup of CDC-7-XPUFes

Using the setup in Fig. 3, for the statistical characteristics CRPs, we generated up to 16,000 (challenges) $\times$32 (iterations) $\times$128 (response length) $\times$3 (Zynq-7020 SoCs) CRPs out of each design. The repetition of the CRPs is needed to study the statistical characteristics and investigate related metrics such as correctness and steadiness. The CRPs were captured at an ambient temperature of approximately 27$^o$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 Block Random Access Memory (BRAM), CRPs are sent to the processing system (PS) part from the programmable logic (PL) part. From the PS part via Universal

Asynchronous Receiver-Transmitter (UART), the CRPs are sent to the personal computer (PC) with a baud rate of 230,400 bits/second between the PuTTY [26] terminal and the SoCs.

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

As explained between Section 2.3.2-2.3.4, the implementation result of *steadiness*, *correctness*, *diffuseness*, *uniformity*, and *uniqueness* are presented in Table 1 for both 32-bit and 64-bit CDC-7-XPUF implementations. Also, in this table, the results of the referenced design in [3] are presented. Furthermore, in the following sections, these comparative results in Table 1 are analyzed.

**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% |

### 4.1 Steadiness and Correctness

The steadiness and correctness metrics, which measure stability and error rate, respectively, are both calculated on a scale from 0 to 1. The results for the 32-bit implementation are slightly worse than those of the reference design but remain within acceptable limits. Furthermore, the performance of the 64-bit implementation is marginally lower compared to the 32-bit version. As discussed by Mursi [3], increasing the number of stages negatively impacts both steadiness and correctness.

### 4.2 Diffuseness

The diffuseness metric, which assesses the spread of the generated responses, showed a slight improvement over the reference design for the 32-bit version and further improved with the 64-bit version. This is consistent with [3], which indicate that increasing the stage number generally enhances diffuseness.

### 4.3 Uniformity

Uniformity, which ideally has an expected value of 0.5, was also calculated and normalized as percentages. The results indicate that the 64-bit implementation exhibits slightly better uniformity compared to both our 32-bit design and the reference design, despite Mursi's observation [3] that increasing the number of stages often has a negative effect on uniformity. In this case, however, the increase in the number of stages appears to have led to an improvement in uniformity.

### 4.4 Uniqueness

The uniqueness metric, which evaluates the ability of the PUF to generate distinct responses, also demonstrated improvement with an increase in bit length. The 32-bit results were slightly better than those of the reference design, and the 64-bit results further enhanced the uniqueness, in alignment with Mursi's findings [3], which suggest that increasing the number of stages positively influences uniqueness.

### 4.5 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 digital signal processor (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 [22] 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

In this study, we implemented and evaluated machine learning-resistant 32-bit and 64-bit designs of the Component-Differentially Challenged XOR Arbiter PUF with 7 streams (CDC-7-XPUF), drawing from established designs in the literature. We evaluated these CDC-7-XPUFs using standard PUF metrics and provided an analysis of the resource utilization ratios for both implementations. We have thoroughly examined the resilience against ML 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 ML attacks. In the sections that follow the discussion on ML resilience, five evaluation criteria are detailed. The PL portion of the Zynq-7020 exhibits an architecture closely resembling that of the Artix-7, which was employed to implement the reference design of the CDC-XPUF in [3]. Consequently, we anticipated similar results to those reported in [3], and, as expected, observed comparable outcomes, as presented in Table 1 in Section 4. In addition to these examinations, lastly, 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 for future applications.

As the future works, the following two items would be considered.

- CDC-7-XPUF designs are also applicable to other FPGA and SoC platforms. Hence, this design would be tested over other platforms.
- CDC-7-XPUF designs would be tested in various environmental conditions such as varying temperature and varying voltage.

## 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). https://doi.org/10.1145/1278480.1278484
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). https://doi.org/10.1109/VLSID51830.2021.00026
3. Mursi K. T.: From XOR PUF to CDC XOR PUF: Cost-Effectiveness, Statistical Characteristics, and Security Assessment, Ph.D. Thesis, Texas Tech University (2021). https://ttu-ir.tdl.org/bitstreams/d26a326c-9494-47e4-a9e2-9af57d949e88/download
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. In: International Journal of Computer Networks & Communications, vol. 14, p. 3 (2022). https://doi.org/10.5121/ijcnc.2022.14301
5. Hofer M., Böhm C.: Physical Unclonable Functions in Theory and Practice, Springer New York, NY, 1st edition (2012) https://doi.org/10.1007/978-1-4614-5040-5
6. Xilinx Zynq-7000 SoC ZC702 Evaluation Kit, https://www.xilinx.com/products/boards-and-kits/ek-z7-zc702-g.html, Accessed: 2023-09-01

7. 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. In: Entropy, 24(11), ISSN 1099-4300 (2022). https://doi.org/10.3390/e24111566
8. 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). https://doi.org/10.1109/ACCESS.2024.3400624
9. Gassend B., Clarke D., Dijk M. van, Devadas S.: Silicon Physical Random Functions. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS '02, p. 148–160, Association for Computing Machinery, New York, NY, USA, ISBN 1581136129 (2002) https://doi.org/10.1145/586110.586132
10. Mursi K. T., Zhuang Y.: Experimental Examination of Component-Differentially-Challenged XOR PUF Circuits. In: Journal of Physics: Conference Series, 1729(1), p. 012006. IOP Publishing (2021). https://doi.org/10.1088/1742-6596/1729/1/012006
11. 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. In: Electronics, 9(10), ISSN 2079-9292 (2020). https://www.mdpi.com/2079-9292/9/10/1715
12. Wisiol N., Becker G. T., Margraf M., Soroceanu T. A. A., Tobisch J., Zengin B.: Breaking the Lightweight Secure PUF: Understanding the Relation of Input Transformations and Machine Learning Resistance. In: S. Belaïd and T. Güneysu, editors, Smart Card Research and Advanced Applications, pp. 40–54, Springer International Publishing, Cham, ISBN 978-3-030-42068-0 (2020). https://doi.org/10.1007/978-3-030-42068-0_3
13. 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. In: IEEE Transactions on Multi-Scale Computing Systems, 2, pp. 146–159 (2016). https://api.semanticscholar.org/CorpusID:1355676
14. Hori Y., Yoshida T., Katashita T., Satoh A.: Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs. In: 2010 International Conference on Reconfigurable Computing and FPGAs, Cancun, Mexico, pp. 298-303 (2010). https://doi.org/10.1109/ReConFig.2010.24
15. 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
16. Anandakumar N. N., Hashmi M., Tehranipoor M.: FPGA-based Physical Unclonable Functions: A Comprehensive Overview of Theory and Architectures, Integration, 81, 07 (2021). https://doi.org/10.1016/j.vlsi.2021.06.001
17. Lim D., Lee J., Gassend B., Suh G., Dijk M. van, Devadas S.: Extracting Secret Keys From Integrated Circuits. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 13(10), pp. 1200–1205 (2005). https://doi.org/10.1109/TVLSI.2005.859470
18. 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. In: Electronics 2020, 9, 870 (2020). https://doi.org/10.3390/electronics9050870
19. Alkatheiri M. S., Zhuang Y.: Towards Fast and Accurate Machine Learning Attacks of Feed-Forward Arbiter PUFs. In: 2017 IEEE Conference on Dependable and Secure Computing, Taipei, Taiwan, pp. 181-187 (2017). https://doi.org/10.1109/DESEC.2017.8073845

20. Aseeri A. O., Zhuang Y., Alkatheiri M. S.: A Machine Learning-Based Security Vulnerability Study on XOR PUFs for Resource-Constraint Internet of Things. In: 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
21. 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. In: 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
22. 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, Accessed: 2023-09-01
23. Microsoft Corporation, Visual Studio 2022 (2022). https://visualstudio.microsoft.com/, Accessed: 2023-11-01.
24. Python Software Foundation, Python Language Reference, version 3.x. https://www.python.org/, Accessed: 2023-11-01
25. IEEE Computer Society, IEEE Standard VHDL Language Reference Manual, IEEE Std 1076-2008 (2008)
26. PuTTY - a free and open-source terminal emulator, serial console, and network file transfer application. https://www.putty.org/, Accessed: 2023-09-01