Improving the Reliability of FPGA CRO PUFs

Hayden Cook, Zephram Tripp, Brad Hutchings, and Jeffrey Goeders
Department of Electrical and Computer Engineering
Brigham Young University, Provo, Utah, USA

2023 33rd International Conference on Field-Programmable Logic and Applications (FPL) | 979-8-3503-4151-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/FPL60245.2023.00053

***Abstract*—This paper presents a novel technique that greatly improves the reliability of FPGA-based CRO PUFs. We improve upon existing CRO implementations and increase the number of configurations per CLB tile from $16\,384$ to 1.1×10^{12}. To maximize reliability, each CRO pair must be configured to maximize its frequency difference. This requires using a novel technique that reduces the configuration search space from 1.1×10^{12} to 256.**

Our CRO PUF achieves 100% reliability within the FPGA's maximum rated voltages. We believe that this is the first FPGA PUF that can achieve this level of reliability without the use of post-processing. We also show that in some cases, our CRO may be reliable enough to omit the ECC that is usually required in PUF-based key generation circuits. This allows our CRO PUF to provide the reliability required for key generation while reducing the latency, complexity, and area overhead of ECC algorithms.

I. Introduction

Due to their rising popularity, FPGAs are frequently used to process and handle critical data in security-sensitive industries such as communication infrastructure, government, and national defense. Thus, it is becoming increasingly important to ensure that the cryptographic keys used to secure such critical data are generated and stored in a safe and secure manner [1].

A physical unclonable function (PUF) is a digital hardware primitive that can be used to simultaneously generate and store cryptographic keys. PUFs use the intrinsic randomness of the silicon caused by manufacturing variations to produce a random output that is unique to each chip. Since the output of the PUF is theoretically the same throughout the lifetime of the part, it eliminates the need for key storage as the key can repeatedly be regenerated from the PUF's output.

Unfortunately, due to factors such as environmental variations and silicon aging, a PUF's output almost always experiences noise, making PUFs unfit for key generation by themselves. For successful key generation, the noisy PUF output must be run through an error correcting code (ECC) to reduce the probability of bit flips. However, ECC algorithms add complexity and significant overhead to the latency, area, and memory of the PUF-based key generation circuit.

This work focuses on improving the reliability of ring oscillator PUFs, which are commonly used for FPGAs. In particular, we present an improved configurable RO PUF (CRO PUF) that has a bit error rate below 1×10^{-6}, which is low enough to eliminate the need for ECC during key generation for many applications. This is accomplished by improving upon existing CRO designs to increase the number of possible configurations to 2^{40} configurations per CLB tile. We also implement a novel technique to determine the optimal configuration for each CRO pair in the PUF. The optimal configuration maximizes the frequency difference between CROs in the pair, and minimizes the probability of a bit flip happening, even under varying environmental conditions.

H. Cook, Z. Tripp, B. Hutchings, and J. Goeders are also with the NSF Center for Space, High-performance, and Resilient Computing (SHREC)

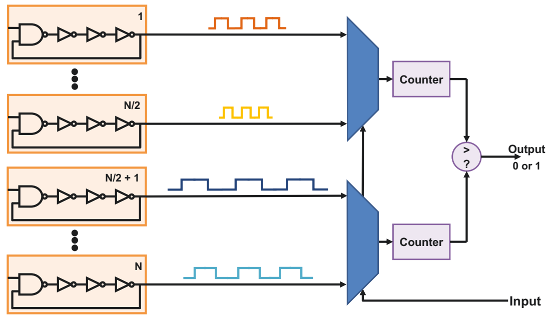


Figure 1: A standard implementation of an RO PUF.

To the best of our knowledge, this is the first work to present a PUF that, for many applications, may be reliable enough to be used for key generation without the need for ECC. We believe this work provides a significant enhancement to PUF-based key generation on FPGAs.

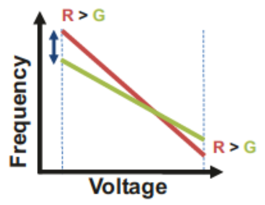
II. Background

A. Physical Unclonable Functions

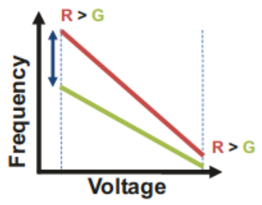
Physical unclonable functions (PUFs) generate a unique output (response) based on an input (challenge) and a source of intrinsic randomness, which allows them to be tamper-resistant and difficult to clone, making them useful for chip authentication and cryptographic key generation.

Ring oscillator PUFs (RO PUFs) use ROs to generate their output. The frequency at which ROs oscillates is dependent on the intrinsic delays of its gates and wires. Due to manufacturing variation, delays will vary between ROs, giving ROs on the same board different frequencies. RO PUFs are one of the most common PUF implementations for FPGAs due to their simplicity and compatibility with FPGA routing.

Suh and Devadas were the first to propose an RO PUF, which is shown in Figure 1. It consists of N ROs, each attached to two multiplexers that are used to select two ROs. These multiplexers are both fed into counters, which are used to measure the frequencies of the pair of selected ROs. The output of the counters is fed into a comparator, which outputs a logic-0 or logic-1 depending on which RO in the pair is



(a)



(b)

Figure 2: Graphs showing what can happen to RO frequencies as environmental conditions, such as voltage, change.

faster [2]. The maximum number of pairs that can be made from N ROs while avoiding correlation between pairs is $N-1$.

Unfortunately, standard RO PUFs can experience bit flips due to aging or changes in voltage or temperature. An example is shown in Figure 2, which plots the frequency of two ROs against the chip voltage for two scenarios. In the first, (Figure 2a), the RO frequencies are close together and eventually cross as the voltage increases, leading to a bit flip. In the second scenario (Figure 2b), the frequencies are far enough apart that they do not cross, preventing a bit flip. Thus the probability of a bit flip can be reduced by increasing the difference in frequency between ROs in a pair.

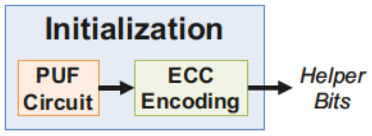
B. PUF Metrics

There are two main metrics used to measure the strength of a PUF's response: reliability and uniqueness. Reliability quantifies the average percentage of bits in a PUF's response that are stable over a range of operating conditions for a single chip. To calculate reliability, the PUF's n-bit base response, R, is extracted from the PUF while the chip is operating at a nominal voltage and temperature. Then, for each operating condition, the same n-bit PUF response, R' is extracted x times. The equation for calculating reliability for a single chip is adapted from [3] and is shown in Equation (1). It relies on the HD between the base response and each of the responses at the operating condition in question. The HD is the number of bits that are different between two responses.

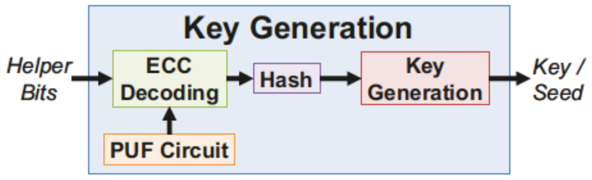
$$R = (1 - \frac{1}{x}\sum_{y=1}^{x}\frac{HD(R, R'_y)}{n}) * 100\% \qquad (1)$$

Uniqueness quantifies the average percentage of bits that change between the PUF's responses of different chips. Ideally, this value is 50%, meaning each bit in each PUF response has an equal probability of being a '0' or a '1'. To calculate the uniqueness of a PUF among k chips, each chip's n-bit PUF response is extracted while the chip is operating at nominal voltage and temperature. Then, for each pair of chips i and j (where $i \neq j$) with responses R_i and R_j respectively, the uniqueness can be calculated from Equation (2) [3].

Imperfect uniqueness is often due to systematic variation which is caused by mask errors, off-axis lithographic focusing errors, etc., and creates the same delay variations across multiple dies. When these systematic variations overcome the random stochastic variations found in the die, it can cause



(a) ECC initialization.



(b) Key generation.

Figure 3: Cryptographic key generation with PUFs. Helper bits are initialized and used for subsequent key generations.

a bias in the output of some of the RO pairs in an RO PUF, so that they are more likely to produce the same value across multiple chips, ultimately decreasing the uniqueness of the PUF's output [4]. This issue becomes more prevalent as transistor sizes continue to scale down [5]. Mismatched routing between ROs in a pair can also create a bias, making it essential that each RO in a PUF has identical routing.

If the PUF's uniqueness is not exactly 50%, then the PUF's output will not have perfect entropy. This can be compensated for by increasing the number of bits the PUF produces at the cost of an increased area footprint.

$$U = \frac{2}{k(k-1)}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\frac{HD(R_i, R_j)}{n} * 100\% \qquad (2)$$

C. PUF Key Generation Model

Best practice requires that cryptographic keys are generated from a source of true randomness to ensure uniqueness and unpredictability. PUFs are a good source of true randomness making them good candidates for key generation. Additionally, keys generated from PUFs are recreated on each bootup which eliminates the need to store the key in memory and restricts an attacker's ability to intervene the key through methods such as side-channel attacks [6], [7].

Unlike authentication, which requires many challenge-response pairs (CRPs), PUFs used for key generation (called *strong PUFs*) require only a single CRPs. However, while PUFs used for authentication can be tolerant of some bit flips in their output, PUFs used for key generation (called *weak PUFs*) must provide the exact same output each time in order to generate the same key. Thus, it is of the utmost importance to ensure that the output of the PUF never changes.

Figure 3 shows the process to generate a key using a weak PUF. The most common technique to ensure a consistent PUF output is to use an error correcting code (ECC) on the PUF output. As shown in Figure 3a, the first time a PUF is run with an ECC, the PUF output is fed through an ECC encoder, which creates a number of helper bits that are commonly referred to as a syndrome. The syndrome is then saved to off-chip, non-volatile memory. Figure 3b shows that subsequent PUF outputs and the associated syndrome are fed through an ECC decoder to ensure that the same output is used each time to generate the cryptographic key [2].

Figure 3b also shows that the RO PUF response is fed into a hash function after being processed by the ECC. Since RO PUFs rarely have perfect entropy, and since the syndrome may reveal some information on the PUF's output [8], PUFs used for key generation are often required to output more bits than the length of the key to maintain the same level of security. The hash function can then be used to compress the PUF's output into the correct size for the key.

For many encryption models, the output of the hash function can be used directly as the key. If the key requires special mathematical properties (such as with RSA), the output of the hash can be used as a seed to generate the key itself.

Additionally, while machine learning attacks can be employed to predict a PUF's CRPs, such attacks are not applicable to weak PUFs, such as the one presented in this paper [9]. Unlike strong PUFs, weak PUFs only have a single CRP, with the response never being exposed to an attacker. Without known CRPs, an machine learning attack becomes impossible.

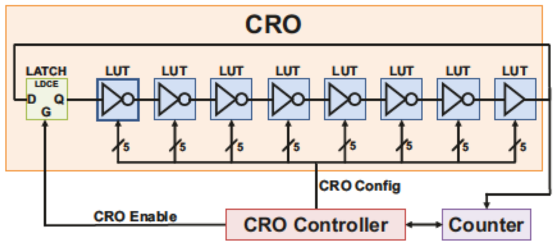
III. Related Works

Suh and Devadas were the first to propose a reliability enhancement for RO PUFs. They used a 1-out-of-k masking scheme where ROs are grouped into groups of k. Only the fastest and slowest ROs of each group are paired together, increasing the frequency difference of each pair. While this improves reliability it also increases the area cost of the PUF by a factor of $k/2$ [2].

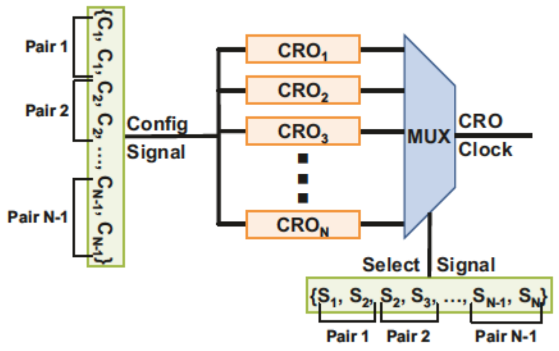
To improve the area efficiency of a reliable RO PUF, Maiti et al. proposed the configurable RO PUF (CRO PUF) for Xilinx FPGAs. It uses a multiplexer to select between two inverters at each stage of the RO. It consists of an AND gate and three pairs of inverters each connected to a multiplexer, giving it a total of eight configurations. For each pair of CROs, all eight configurations are tested to find which one gives the largest frequency difference. This configuration is then saved and used when extracting all future PUF responses. The CRO PUF provides more reliability than a standard RO PUF, without raising the area cost [3].

Habib et al. increase the number of configurations by using the internal LUT multiplexers available in each 4-input LUT of the Spartan 3E. Configuring three LUTs as inverters, and one LUT as an AND gate leaves 11 unused LUT inputs that can be used to select the internal path of the LUTs, allowing their CRO to achieve 2048 challenge-response pairs. However, they use the configurations to create a strong PUF with many challenge-response pairs and do not discuss using it to improve the reliability of a weak PUF used for key generation.

Since Habib et al., others have proposed improvements to RO PUFs, such as new comparison strategies [10], phase calibration [11], and XOR-based CRO PUFs [12], [13]. However, like Habib et al., all of these improvements target strong RO PUFs. To the best of our knowledge, Suh and Devadas and Maiti et al. are the only two works to propose reliability enhancements for weak RO PUFs used for key generation. As such, we will focus on comparing the performance of our PUF with theirs.



(a) Our CRO architecture. which contains 40 configuration lines, giving it a total of 2^{40} possible configurations.



(b) Our CRO PUF architecture. Each CRO in a pair uses the same configuration, which is chosen to maximize the frequency difference between the CROs in the pair.

Figure 4: The architecture of our CRO and CRO PUF.

Table I: Comparison of the number of configurations our CRO has compared to previous implementations of CROs.

Work	Configuration Lines	Configurations Per CLB Tile
Maiti et al. [3]	3	8
Habib et al. [14]	11	2048
Yao et al. [13]	15	16384
This work	40	1.1×10^{12}

IV. Reliable CRO PUF

A. CRO Architecture

The CRO we propose in this paper can be seen in Figure 4a. Our CRO PUF is designed to fit within a single CLB tile for Xilinx 7-Series and UltraScale/UltraScale+ parts. Of the eight LUTS, seven are configured as inverters, and one as a buffer. Each 6-input LUT uses one input (I1) for the ROs feedback line, with the other five inputs (I2-I6) being used as "don't care" inputs that select the internal LUT path.

Our CRO improves upon the one proposed by Habib et al. It nearly quadruples the number of configuration lines from 11 to 40 by doubling the number of LUTs in the CRO and using a newer FPGA architecture that increases the number of LUT inputs from four to six. Additionally, the AND gate is replaced by a latch for the enable signal, providing one additional configuration line.

Table I compares the number of configurations our CRO has with previous work. Our CRO has over double the number of configuration lines compared to the state-of-the-art, which exponentially raises the number of configurations of our CRO from 16384 to 1.1×10^{12} configurations.

B. CRO PUF Reliability Enhancement

To maximize the reliability of our CRO PUF we find the configurations that give each CRO pair the largest frequency difference. However, it isn't feasible to iterate over all 1.1×10^{12} configurations of our CRO. Thus, we had to come up with a technique to find the optimal configuration for each CRO pair without iterating through every configuration.

The optimal configurations for our CRO PUF can be found by assuming that each LUT configuration is independent of all other LUT configurations in the CRO. This assumption is based on the knowledge that configuring one LUT to use a certain internal path does not affect which internal paths are used by any other LUT in the CRO. Thus if we can find the optimal configuration of each individual LUT pair in the CRO pair and concatenate them together, we have found the optimal configuration for the CRO pair.

Figure 5 details the methodology we use to find the optimal configuration for a single pair of CROs.

V. Experiments

A. Experiment Setup

To demonstrate the uniqueness and reliability of our CRO PUF, we perform experiments that test our CRO PUF's response on six different FPGAs throughout a range of voltages and temperatures. The experiments implement our CRO PUF on six Digilent ARTY A7-35T boards. All six boards have been modified to bypass the power regulator and power VCCINT directly. Each board is placed in a thermal chamber with VCCINT connected to a Keysight N6763A power supply, allowing us to perform voltage and temperature sweeps to test the reliability of our CRO PUF.

A CRO is placed on all 4075 tiles of the FPGA. The chain pairing method from [14] is used to create each CRO pair, giving our PUF a 4074-bit response. RapidWright [15] is used to ensure that the placement and routing of each CRO are identical. While extracting the PUF response, each CRO is run individually to ensure that they do not affect each other.

Before running voltage and temperature sweeps, the CRO PUF is first initialized on each FPGA running at a nominal voltage of $1.0\,\mathrm{V}$ [16] and an ambient temperature of $35\,^\circ\mathrm{C}$ to find the optimal configurations for each CRO pair on each FPGA. Once optimal configurations have been found and saved, a voltage and temperature sweep are performed separately. For the voltage sweep, VCCINT is varied by $\pm 20\%$ ($0.8\,\mathrm{V}$ to $1.2\,\mathrm{V}$) in steps of 10% ($0.1\,\mathrm{V}$). For the temperature sweep, the ambient temperature is varied from $25\,^\circ\mathrm{C}$ to $65\,^\circ\mathrm{C}$ in steps of $10\,^\circ\mathrm{C}$. The output of the PUF is extracted three times at each voltage and temperature setting.

Table II: Average reliability of our CRO PUF compared to other reliable-enhanced RO PUFs.

Experiment	0.8 Vdd	0.9 Vdd	Vdd	1.1 Vdd	1.2 Vdd
CRO PUF (This work)	100%	100%	100%	100%	99.997%
CRO PUF (Maiti et al. [3])	94.643%	97.768%	100%	99.554%	99.107%
RO PUF (Suh and Devedas [2])	99.893%	100%	100%	100%	99.964%

B. Experimental Results

1) Reliability: The reliability of our CRO PUF is compared to other reliable RO PUF implementations in Table II. Unlike Maiti et al., Suh and Devadas do not vary voltage by $\pm 20\%$ from nominal while testing reliability. For an accurate comparison, we recreated their reliable RO PUF using the 1-out-of-8 masking scheme and tested its reliability using the same FPGAs and parameters used to test the reliability of our own CRO PUF. The reliability of our CRO PUF is greater than both Maiti et al.'s and Suh and Devadas' PUFs. With our PUF, only one board experiences a bit flip, and only at 20% above nominal, ($1.2\,\mathrm{V}$), which is outside of the Artix-7's maximum rated voltage [16]. While a temperature sweep was also performed to test reliability, no bit flips were experienced by any of the tested PUFs at any temperature.

2) Uniqueness: The uniqueness of our CRO PUF is compared to other RO PUF implementations in Table III. Our PUF has lower uniqueness compared to the other PUFs. When compared to Maiti et al., our lower uniqueness may be partially due to our CRO PUF being implemented on an FPGA with a smaller transistor feature size (see Section II-B) [5]. Suh and Devadas' RO PUF is reimplemented on the same FPGAs as our CRO PUF and should have similar uniqueness. However, the 1-out-of-8 masking scheme used by their PUF increases the spatial variability of the RO pairs, filtering out localized systematic variation and potentially improving the uniqueness of the PUF.

For a more accurate analysis of the impact our reliability enhancement has on the uniqueness of RO PUFs, we also measured the uniqueness of a standard RO PUF that does not use the 1-out-of-k masking scheme [2] (see Figure 1). To ensure a fair comparison, the standard RO PUF was implemented on the same FPGAs and used the same placement and routing as our CRO PUF. As shown in Table III, the output of the standard RO PUF is 2.10% more unique than our CRO PUF, showing that our reliability enhancement does have a minor, but noticeable, impact on uniqueness.

As uniqueness gets further from 50%, entropy is lost. Loss of entropy in bits is calculated using Equation (3), where H is the entropy lost in bits, U is the uniqueness, L is the length of the PUF's response. Loss of entropy can be compensated by increasing the size of the PUF's output. The required length of a PUF's output to compensate for entropy loss is calculated using Equation (4), where L' is the length of the

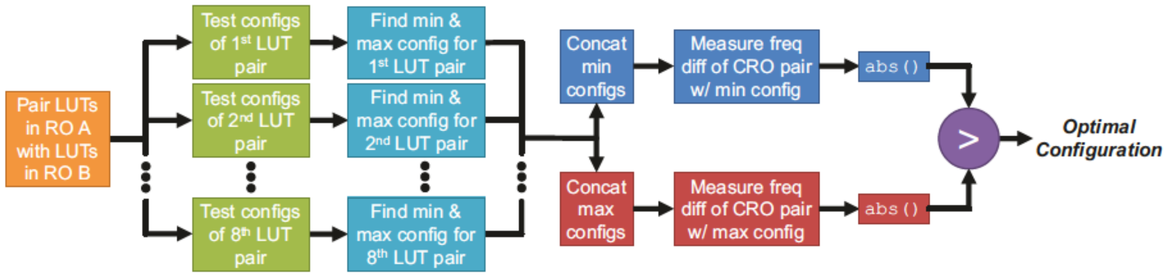

Pair LUTs in RO A with LUTs in RO B
Test configs of 1st LUT pair
Test configs of 2nd LUT pair
Test configs of 8th LUT pair
Find min & max config for 1st LUT pair
Find min & max config for 2nd LUT pair
Find min & max config for 8th LUT pair
Concat min configs
Concat max configs
Measure freq diff of CRO pair w/ min config
Measure freq diff of CRO pair w/ max config
abs ()
abs ()
>
Optimal Configuration

Figure 5: The technique to find the optimal configuration for a pair of CROs.

Table III: Uniqueness of our CRO PUF compared to other RO PUFs. Additionally, the area increase needed to compensate for entropy loss, as well as the area and latency overheads of ECC (if required) are also included for each RO PUF. The ECC area overhead and clock cycle estimation are taken from [17] and assume a 128-bit PUF.

Work	Uniqueness	Area Increase (Entropy Compensation)	Requires ECC	ECC Area Overhead	ECC Clock Cycle Estimate
CRO PUF (This work)	39.72%	25.88%	Likely No	0	0
CRO PUF ([3])	47.31%	5.69%	Yes	21.48%-39.45%	9,000-365,000
Reliable RO PUF ([2])	45.13%	10.79%	Yes	21.48%-39.45%	9,000-365,000
Standard RO PUF ([2])	41.82%	19.56%	Yes	21.48%-39.45%	9,000-365,000

PUF's response after compensating for imperfect uniqueness.

$$H = L * (100\% - 2U) \tag{3}$$

$$L' = \frac{L}{2U} \tag{4}$$

VI. Discussion

To the best of our knowledge, our CRO PUF is the only PUF to achieve 100% reliability within the absolute maximum rated voltage ranges of an FPGA. This means that our CRO PUF may be reliable enough to allow for the removal of the ECC decoder from a key generation circuit (see Figure 3). This would reduce the area overhead of the overall PUF circuit, while also reducing the computational complexity, latency, and memory requirements of the key generation process.

For key generation, it is common to use an ECC that reduces the probability of a bit flip in the corrected PUF response to 1×10^{-6} [17]. To further test the reliability of our PUF, we performed a simple experiment on two FPGAs with one running at 0.8 V and the other at 1.1 V, which is the maximum rated voltage of the Artix 7 [16]. For each board, the 4074-bit response of our CRO PUF is extracted 247 times at an ambient temperature of 35 °C.

No bit flips were observed for either FPGA, showing that the probability of our CRO PUF experiencing a bit flip, even at extreme voltages, is less than 1×10^{-6}. This indicates that our CRO PUF is reliable enough to eliminate the need for an ECC decoder for many applications that require the probability of a bit flip to be under 1×10^{-6}.

While our CRO PUF is as compact as Maiti et al.'s, it does require 20.19% more of an area increase for entropy compensation. However, for applications where our CRO PUF is reliable enough, the omission of the ECC decoder will often offset the area increase (see Table III). For example, when generating a 128-bit key, our CRO PUF must output 162 bits to produce 128 bits of entropy, which requires 26 more tiles than Maiti et al.'s CRO PUF to produce the same number of bits of entropy. In comparison, ECC decoder circuits for a 128-bit key can take anywhere from 27.5 to 50.5 tiles on a 7-Series FPGA, depending on the memory and latency requirements of the key generation circuit [17]. This ECC area overhead does not include the possible size increase of the PUF needed to compensate for leaked information from the ECC's syndrome [18], [19], which may provide our CRO PUF with additional area savings in the case of ECC omission. Overall, we conclude that for applications where the ECC can be omitted, a PUF-based key generation circuit using our CRO is more area efficient than Maiti's CRO PUF while also providing reduced latency and area overhead.

VII. Conclusion

This paper presents a novel CRO PUF for FPGA key generation that is able to achieve 100% reliability within the maximum rated voltage range of the tested FPGAs. It uses an improved CRO that uses internal LUT paths to allow it 1.1×10^{12} different configurations. Additionally, we created a novel technique to find the optimal configuration between two pairs of CROs that gives the pair the maximum frequency difference, maximizing the reliability of our PUF.

We show that our PUF may be reliable enough to perform key generation without the need for an ECC, greatly reducing the area and latency overheads of the key generation circuit. However, regardless of whether ECC is used, our CRO PUF can improve the reliability of any FPGA-based key generation circuit and improve the security of critical FPGA systems.

References

[1] S. M. Trimberger and J. J. Moore, "FPGA Security: Motivations, Features, and Applications," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1248–1265, Aug. 2014, Conference Name: Proceedings of the IEEE, ISSN: 1558-2256. DOI: 10.1109/JPROC.2014.2331672.

[2] G. E. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," in *Design Automation Conference*, Jun. 2007, pp. 9–14.

[3] A. Maiti and P. Schaumont, "Improved ring oscillator PUF: An FPGA-friendly secure primitive," *Journal of Cryptology*, vol. 24, no. 2, pp. 375–397, Apr. 1, 2011, ISSN: 1432-1378. DOI: 10.1007/s00145-010-9088-4. [Online]. Available: https://doi.org/10.1007/s00145-010-9088-4 (visited on 07/18/2022).

[4] K. A. Asha, A. Patyal, and H.-M. Chen, "Generation of PUF-Keys on FPGAs by K-means Frequency Clustering," in *2018 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, Dec. 2018, pp. 44–49. DOI: 10.1109/AsianHOST.2018.8607174.

[5] P. Sedcole and P. Y. K. Cheung, "Within-die delay variability in 90nm FPGAs and beyond," in *2006 IEEE International Conference on Field Programmable Technology*, Dec. 2006, pp. 97–104. DOI: 10.1109/FPT.2006.270300.

[6] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security*, ser. CCS '02, New York, NY, USA: Association for Computing Machinery, Nov. 2002, pp. 148–160, ISBN: 978-1-58113-612-8. DOI: 10.1145/586110.586132. [Online]. Available: https://doi.org/10.1145/586110.586132 (visited on 10/19/2022).

[7] V.-T. Tran, Q.-K. Trinh, and V.-P. Hoang, "Stabilizing On-chip Secure Key Generation Using RO-PUF," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, ISSN: 2162-1233, Oct. 2021, pp. 805–809. DOI: 10.1109/ICTC52510.2021.9621147.

[8] M.-D. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 48–65, Jan. 2010, Conference Name: IEEE Design & Test of Computers, ISSN: 1558-1918. DOI: 10.1109/MDT.2010.25.

[9] M. Kojage, N. Hassan, and U. Chatterjee, "Machine Learning Attacks on Low-Cost Reconfigurable XRRO and XRBR PUF Designs," en, in *Security, Privacy, and Applied Cryptography Engineering*, L. Batina, S. Picek, and M. Mondal, Eds., ser. Lecture Notes in Computer Science, Cham: Springer Nature Switzerland, 2022, pp. 204–224, ISBN: 978-3-031-22829-2. DOI: 10.1007/978-3-031-22829-2_12.

[10] W. Liu, Y. Yu, C. Wang, Y. Cui, and M. O'Neill, "RO PUF design in FPGAs with new comparison strategies," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, ISSN: 2158-1525, May 2015, pp. 77–80. DOI: 10.1109/ISCAS.2015.7168574.

[11] W. Yan, C. Jin, F. Tehranipoor, and J. A. Chandy, "Phase calibrated ring oscillator PUF design and implementation on FPGAs," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, ISSN: 1946-1488, Sep. 2017, pp. 1–8. DOI: 10.23919/FPL.2017.8056859.

[12] L. Zhang, C. Wang, W. Liu, M. O'Neill, and F. Lombardi, "XOR gate based low-cost configurable RO PUF," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, ISSN: 2379-447X, May 2017, pp. 1–4. DOI: 10.1109/ISCAS.2017.8050628.

[13] L. Yao, H. Liang, Z. Huang, C. Jiang, M. Yi, and Y. Lu, "A Lightweight Configurable XOR RO-PUF Design Based on Xilinx FPGA," in *2021 IEEE 4th International Conference on Electronics Technology (ICET)*, May 2021, pp. 83–88. DOI: 10.1109/ICET51757.2021.9451016.

[14] B. Habib, K. Gaj, and J.-P. Kaps, "FPGA PUF Based on Programmable LUT Delays," in *2013 Euromicro Conference on Digital System Design*, Sep. 2013, pp. 697–704. DOI: 10.1109/DSD.2013.79.

[15] C. Lavin and A. Kaviani, "RapidWright: Enabling custom crafted implementations for FPGAs," in *Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Apr. 2018, pp. 133–140. DOI: 10.1109/FCCM.2018.00030.

[16] Xilinx, "Artix-7 FPGAs data sheet: DC and AC switching characteristics (DS181)," p. 64, 2018.

[17] M. Hiller, L. Kürzinger, and G. Sigl, "Review of error correction for PUFs and evaluation on state-of-the-art FPGAs," en, *Journal of Cryptographic Engineering*, vol. 10, no. 3, pp. 229–247, Sep. 2020, ISSN: 2190-8516. DOI: 10.1007/s13389-020-00223-w. [Online]. Available: https://doi.org/10.1007/s13389-020-00223-w (visited on 11/03/2022).

[18] H. Yu, Q. Xu, and P. H. Leong, "Fine-grained characterization of process variation in FPGAs," in *Conference on Field-Programmable Technology (FPT)*, Beijing, China: IEEE, Dec. 2010, pp. 138–145, ISBN: 978-1-4244-8980-0. DOI: 10.1109/FPT.2010.5681770. [Online]. Available: http://ieeexplore.ieee.org/document/5681770/ (visited on 07/18/2022).

[19] G. E. Suh, C. W. O'Donnell, and S. Devadas, "Aegis: A Single-Chip Secure Processor," *IEEE Design & Test of Computers*, vol. 24, no. 6, pp. 570–580, Nov. 2007, Conference Name: IEEE Design & Test of Computers, ISSN: 1558-1918. DOI: 10.1109/MDT.2007.179.