

PA-PUF: A Novel Priority Arbiter PUF

Simranjeet Singh*, Srinivasu Bodapati[†], Sachin Patkar*, Rainer Leupers[§],
Anupam Chattopadhyay[‡], Farhad Merchant[§]

*Indian Institute of Technology, Bombay, [†]Indian Institute of Technology, Mandi,

[‡]Nanyang Technological University, Singapore, [§]RWTH Aachen University, Germany

{simranjeet, patkar}@ee.iitb.ac.in, srinivasu@iitmandi.ac.in, anupam@ntu.edu.sg, {leupers, merchantf}@ice.rwth-aachen.de

Abstract—This paper proposes a 3-input arbiter-based novel physically unclonable function (PUF) design. Firstly, a 3-input priority arbiter is designed using a simple arbiter, two multiplexers (2:1), and an XOR logic gate. The priority arbiter has an equal probability of 0's and 1's at the output, which results in excellent uniformity (49.45%) while retrieving the PUF response. Secondly, a new PUF design based on priority arbiter PUF (PA-PUF) is presented. The PA-PUF design is evaluated for uniqueness, non-linearity, and uniformity against the standard tests. The proposed PA-PUF design is configurable in challenge-response pairs through an arbitrary number of feed-forward priority arbiters introduced to the design. We demonstrate, through extensive experiments, reliability of 100% after performing the error correction techniques and uniqueness of 49.63%. Finally, the design is compared with the literature to evaluate its implementation efficiency, where it is clearly found to be superior compared to the state-of-the-art.

Index Terms—arbiter, priority arbiter, PUF, device authentication, security, LFSR

I. INTRODUCTION

Physical unclonable functions (PUFs) have great prominence in today's secure device authentication and secure communication [1]. A PUF takes advantage of randomness due to manufacturing process variation in integrated circuits and generates a unique response for every device by tapping into the sources of entropy such as variations in path delay or timing [1]. An integrated circuit within the PUF maps the input challenge with a unique response, thus creating a challenge-response pair (CRP). These CRPs are utilized to design various security protocols, ranging from device attestation to data encryption. The concept of a PUF was first reported in [2]. Based on the parameters of how the response is derived, various kinds of PUFs have been proposed in the literature. These parameters can be delay, memory, or the difference between the current in the rails. In the literature, PUF designs are categorized as delay-based and memory-based PUF - of which the prominent examples are Arbiter PUF [3], ring oscillator PUF [4], SRAM PUF [5], Butterfly PUF [6], Glitch PUF [7] and MEMory Cell-based Chip Authentication (MECCA) PUF [8]. These advanced PUFs are being threatened with various attacks [9], [10]. Therefore, we need to keep studying new constructions of PUFs that potentially thwart such attacks with minimalistic overhead.

Arbiter PUF (APUF) is one of the delay-based PUF designs, where an arbiter decides the response of a PUF between the two data paths comprised of cross-coupled multiplexers [11]. For this purpose, the arbiter PUF uses the analog timing difference between the two data paths and decides the output based on the timing difference between the two lines. Several modifications to the classical arbiter PUF are presented in the

literature. In arbiter PUF, it is possible to predict the relation between the challenge and response through software modeling and programs. To prevent the modeling-based attacks, various modifications were proposed, such as multi-arbiter PUF [12], [13] and double arbiter PUF [14]. The multi-arbiter PUF design consists of arbiters in each multiplexer stage and a multiplexer to choose the arbiter response. An alternative is to take the PUF response as XOR of the arbiter outputs, which improves the uniqueness, reliability, and robustness of the PUF.

The arbiter is an electronic circuit that can identify a signal's first occurrence. A simple D flip-flop can be used as an arbiter where one signal is connected as the clock and the other as the data signal. Priority arbiter is typically used in the Network-On-Chip (NoC) [15], in order to determine the priority of the data request among the several requests [15]. Different kinds of arbiters are proposed in literature [16], such as daisy chain arbiter, round-robin arbiter, and dynamic arbiter. Based on the application of the NoC, the arbiters are designed. The concept of priority arbiter from the NoC communication can be adopted into the PUF to improve the design efficiency. The advantage of having a priority arbiter is that the design has more non-linearity compared to the simple arbiter PUF. In this paper, a new priority arbiter is proposed, which demonstrates good uniformity. Based on that, a novel PA-PUF is designed. The major contributions of this work are reported as follows:

- A new PUF using the priority arbiter called PA-PUF is designed. The PA-PUF offers a uniqueness of 49.63 % and uniformity of 49.45% at the output. The non-linearity in the output of the PUF is increased with the use of a priority arbiter.
- We demonstrate the configurability of PA-PUF by varying the number of CRPs. The number of CRPs can be increased by increasing the length of the data path by introducing more feed-forward arbiters.
- The performance of the proposed priority arbiter PUF is studied as a function of the number of feed-forward arbiters in the data path and the length of the data path. For example, the uniqueness of the PUF can be increased by increasing the length of the data path. It offers a reliability of 94.5% for a 128-bit response, which can be increased to 100% by implementing Bose-Chaudhuri-Hocquenghem (BCH) error-correcting codes [17].

The rest of the paper is structured as follows. Section II provides the design insights of the newly proposed PUF. The experimental results are given in section III. Section IV compares the proposed design with state-of-the-art designs

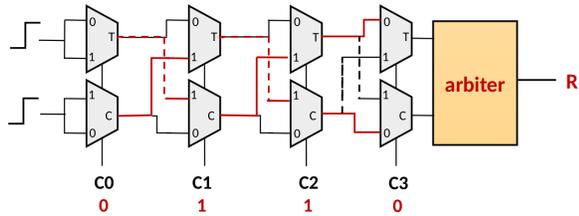


Fig. 1: Classical arbiter PUF design, red lines indicate the data path of a given challenge.

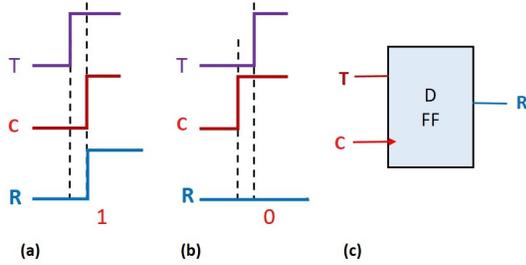


Fig. 2: (a), (b) possible input wave forms, and (c) D flip-flop.

from literature. Finally, Section V provides conclusions and outlook of this work.

II. PROPOSED PRIORITY ARBITER PUF

A classical arbiter PUF with multiplexers and an arbiter is depicted in Fig. 1. The arbiter PUF is a delay-based design and the delay difference is taken from the cross-coupled multiplexers as shown in the design. The challenge of the arbiter PUF is given as select signals of the multiplexers in the data path, a low to high transition given at the input of the multiplexer will be propagated to the arbiter in the path selected by the challenge of the PUF. The arbiter is implemented using a D flip-flop, and the same is presented with the possible operations in Fig. 2, with two signals as Top (T) and Center (C). When the signal C arrives first, then the output of the arbiter is led to '0'; otherwise, the output is '1'. The same is explained in the Fig. 2 (a) & (b). Fig. 1 presents the symmetric path of the circuit through the multiplexers. The multiplexers' select signals (challenge) generate different delay paths, resulting in a unique response for every combination. The major disadvantage with the arbiter PUF is that the uniqueness of the PUF is less. There are several modifications to the classical arbiter PUF by adding a feed-forward path [18] in the design to introduce the non-linearity in the results.

In arbiter PUF, it is possible to predict the relation between the challenge and response through software modeling and programs. To prevent this modeling prediction, various modifications have been proposed in the past. One alternative of arbiter PUF is multi-arbiter PUF [12], [13] and double arbiter PUF [14]. All these methods are proposed to improve the uniqueness, reliability, and robustness of the PUF output. Since the order of non-linearity in the arbiter is less, so

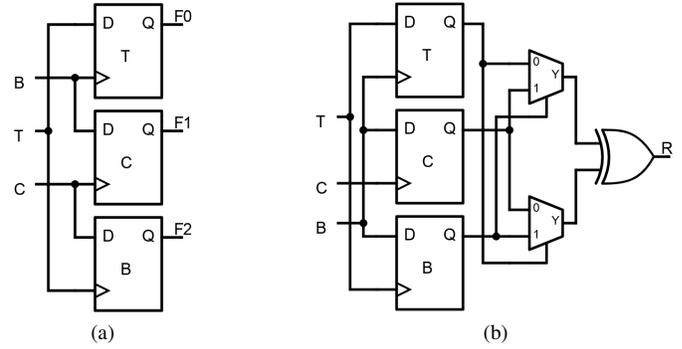


Fig. 3: (a) Feed-forward arbiter for the proposed PA-PUF; T , C and B as the top, center and bottom data lines; $F0$, $F1$ and $F2$ are the three outputs from the feed-forward arbiter (b) Three-input priority arbiter; T , C and B as the top, center and bottom data lines and R as response bit.

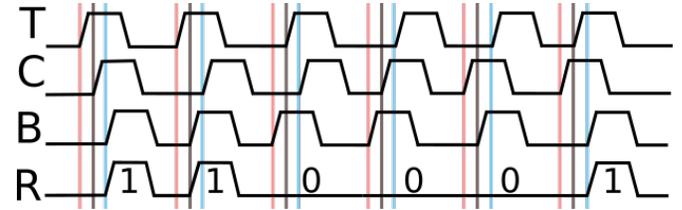


Fig. 4: Priority arbiter possible output (R) combinations with T , C and B as the top, center and bottom data lines.

by considering multi-arbiter designs, the non-linearity can be increased in the design. Next, we will propose a priority arbiter-based PUF design.

A new three-input priority arbiter is proposed to decide the response as per the priority of the input's arrival time. The arbiter used in the classical arbiter is a two-input circuit. Now, an extra input is added to increase the non-linearity in the output. The proposed three-input priority arbiter diagram is shown in Fig. 3b, which is designed with three D-flip flops, two 2-to-1 multiplexers, and an XOR logic gate. The three inputs, T , C , and B (*top, center, and bottom*) are given to the D flip-flops. Next, the select signal and the data line of the multiplexer are taken from the outputs of the D flip-flops. Finally, the output of multiplexers is applied to the input of the XOR gate to generate a response bit.

The operation of the proposed priority arbiter is shown in Fig. 4, where the output is decided based on the priority of the input arrival times. Some of the possible conditions are considered in the plot. Consider a case where the arrival times of these signals are as follows, T , C , and B (first case in Fig. 4). Since T comes first in time, the outputs of the bottom and center D flip-flops (DFF B and DFF C) are '0' and '0', while the top D flip-flop (DFF T) output is '1'. After multiplexer output, the input of the XOR gate will be '1' and '0'. This results in the output of the arbiter being '1'. The Fig. 4 demonstrates the possible conditions $\{T, B, C\}$, $\{T, C, B\}$, $\{B, T, C\}$, $\{B, C, T\}$, $\{C, B, T\}$ and $\{C, T, B\}$ out of these six conditions, the output is '0'

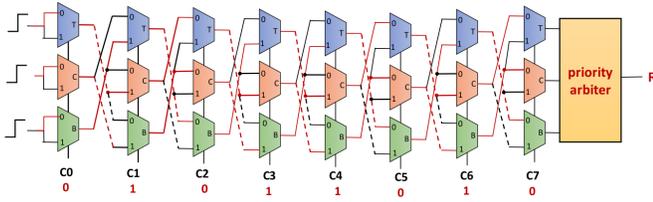


Fig. 5: Schematic of proposed PA-PUF which includes three parallel multiplexer lines (T, C, and B) and a priority arbiter at the end to generate the response bit.

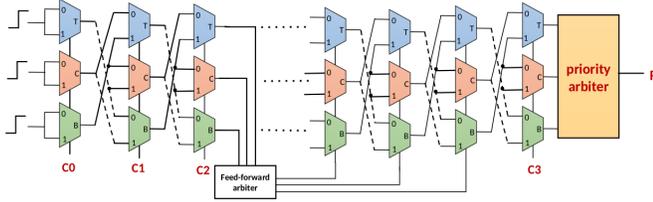


Fig. 6: Proposed PA-PUF with feed-forward arbiter.

in three cases and ‘1’ in three cases. This further results in achieving good uniformity in the output of the priority arbiter. The same circuit with different connections of the inputs will lead to non-uniform outputs (with non-equal probability of 0’s and 1’s).

The PA-PUF is presented in Fig. 5, where the modifications to the classical arbiter PUF have been done by adding a third data path in addition to the two data paths. Fig. 5 shows the working of the PA-PUF for a given challenge (marked in red). This shows that just by increasing the hardware by one-third, the non-linearity in the output can be increased to an extreme. In the case of arbiter PUF, when the output is ‘1’, then it can be understood that signal T arrives before signal C . While in the case of priority arbiter PUF, as explained in Fig. 4, the output is ‘1’, in three cases. Hence, it is difficult to find which signal comes first. Moreover, the design is extended by introducing the feed-forward paths to increase the robustness in the design. The priority arbiter PUF with the feed-forward path is shown in Fig. 6, while the feed-forward arbiter for the priority arbiter PUF is shown in Fig. 3b.

III. EXPERIMENTAL RESULTS

The proposed PA-PUF is designed using Verilog programming and implemented on Nexys Video Artix-7 FPGA board. The responses of the PUF are collected using the universal asynchronous receiver/transmitter (UART) protocol. Next, we will discuss the proposed PA-PUF (with a feed-forward arbiter) and the implications of deriving a security key. For a good PUF design, it should satisfy some important criteria such as *intra-chip Hamming distance*, *inter-chip Hamming distance*. Based on the results of the inter-chip and intra-chip Hamming distance, we can analyse the other important parameters such as *uniformity*, *robustness*, *uniqueness*, *bit-aliasing*, and *reliability*. Intra Hamming distance (HD) is the Hamming distance between the responses of the PUF design within the chip. Ideally, a single bit change in the challenge should result in a 50% Hamming distance in the responses

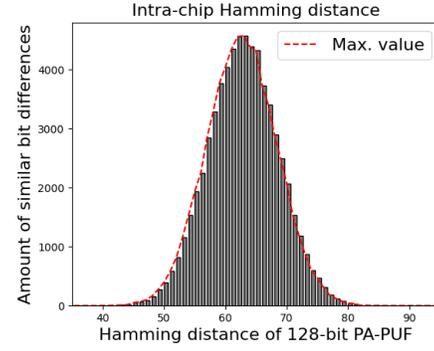


Fig. 7: Intra-chip Hamming distance plot of 128-bit PA-PUF.

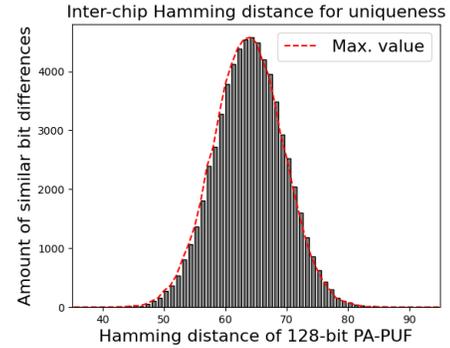


Fig. 8: Inter-chip Hamming distance plot of 128-bit PA-PUF.

bits. The intra HD can be calculated by the formula given in equation 1.

$$Intra\ HD = \sum_{i=1}^k \frac{HD(R_i, R_{i+1})}{n} \times 100 \quad (1)$$

Here, ‘ k ’ is the total number of challenges given to the PUF, R_i and R_{i+1} are the responses to the challenges C_i and C_{i+1} respectively. The challenges are differed by one-bit change, and Fig. 7 shows the Hamming distance in responses of the 128-bit PA-PUF. Fig. 7 also shows the HD has a maximum at the half of the responses, and the plot follows the Gaussian distribution.

Inter HD is the Hamming distance of the responses between two chips of the same family. We have used three FPGAs of the same family/configuration to produce the results on inter HD. Fig. 8 shows the inter HD between responses of two FPGA boards, which is calculated by the formula given in equation 2.

$$Inter\ HD = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^k \frac{HD(R_i, R_j)}{n} \quad (2)$$

Where, i and j are two different FPGAs. R_i and R_j are the responses from $FPGA_i$ and $FPGA_j$ for the challenge C respectively. k is number of PUF designs (3 in our case).

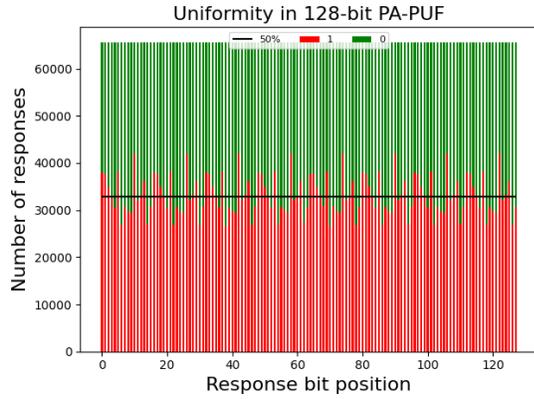


Fig. 9: Uniformity of 128-bit PA-PUF along with 50% probability line.

A. Robustness

Any response bit of the PUF should not be stuck at logic ‘0’ or ‘1’. The stable ‘0’, stable ‘1’ and unstable bits are calculated for various sizes of the PA-PUF design. The results are given in Table I and the results reveal that the design has good robustness.

B. Uniformity and Bit-aliasing

The response should have an equal number of 0’s and 1’s. This can be calculated using the ‘uniformity’ of the PUF. The ideal value of uniformity is 50%, which means the PUF response is uniformly distributed. Fig. 9 shows the distribution of 0’s and 1’s in response bits. Table II gives the values for various sizes of the PUF response. The other parameter of interest is the ‘bit-aliasing’, which is calculated over different chips/devices. The experiment is conducted on three Nexys Video Artix-7 FPGA boards of the same family. The PUF design is placed in the exact physical locations across different boards to find whether any particular bit position is

TABLE I: Robustness results of stable ‘0’ , ‘1’ and unstable bits.

Board Artix-7	Number of bits	Stable ‘0’ %	Stable ‘1’ %	Unstable %
8-bit proposed PA-PUF				
1	0.5×10^6	25.93	24.64	49.43
2	0.5×10^6	27.65	21.71	50.64
16-bit proposed PA-PUF				
1	1×10^6	26.65	24.24	49.11
2	1×10^6	25.07	25.00	49.93
32-bit proposed PA-PUF				
1	2×10^6	26.24	24.92	48.84
2	2×10^6	27.66	23.19	49.15
64-bit proposed PA-PUF				
1	4×10^6	30.52	21.32	48.16
2	4×10^6	28.03	23.01	48.96
128-bit proposed PA-PUF				
1	8×10^6	31.80	19.79	48.41
2	8×10^6	30.49	20.58	48.93
128-bit Ring oscillator PUF				
1	8×10^6	28.1	23.3	48.6
2	8×10^6	27.6	23.6	48.8
Mean		27.85	23.45	48.7
Mean [19]		29.51	30.25	40.22

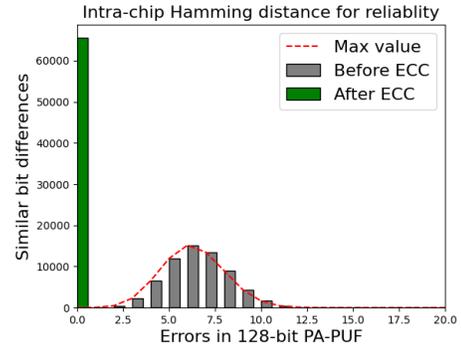


Fig. 10: Reliability plot of 128-bit priority arbiter PUF.

permanently connected to the logic ‘0’ or ‘1’ irrespective of the challenge or a board. The ideal value of the bit aliasing is 50%, which is near to the values recorded in Table II.

C. Reliability and Uniqueness

Reliability and uniqueness are the most important metrics in deciding the design of the PUF. The uniqueness is defined as how uniquely the design can identify from one chip to the other chip of the same family. The ideal value of the uniqueness is 50%; hence the response of the PUF from one chip to the other is differed by half of its length. Uniqueness is calculated using the ‘inter Hamming distance’ of the response, and the calculated values are tabulated in Table III for various sizes of the proposed PUF. Further, the reliability is also given in Table III, which is calculated using the ‘intra-Hamming distance’ of the response by giving the same challenge over million times, and Fig. 10 depicts the Hamming distance plot to calculate the reliability. The error that occurred in the response of the PUF overages can be corrected using the error correction mechanisms [20] and [21]. It has been shown in

TABLE II: Uniformity and bit-aliasing for the proposed PUF.

Parameter	Uniformity	Bit-aliasing
Ideal value	50 %	50 %
8-bit proposed PA-PUF		
Maximum	100 %	52.41 %
Minimum	0 %	42.74 %
Average	49.3 %	49.17 %
16-bit proposed PA-PUF		
Maximum	100 %	64.37 %
Minimum	0 %	39.35 %
Average	49.25 %	48.77 %
32-bit proposed PA-PUF		
Maximum	84.37 %	60.79 %
Minimum	9 %	34.87 %
Average	47.8 %	47.76 %
64-bit proposed PA-PUF		
Maximum	78.75 %	54.82 %
Minimum	21.87 %	32.39 %
Average	47.47 %	47.41 %
128-bit proposed PA-PUF		
Maximum	67.9 %	51.69 %
Minimum	25 %	33.03 %
Average	49.45 %	49.66 %
128-bit Ring oscillator PUF		
Maximum	67.2 %	52.82 %
Minimum	0 %	0 %
Average	48.3 %	47.8 %

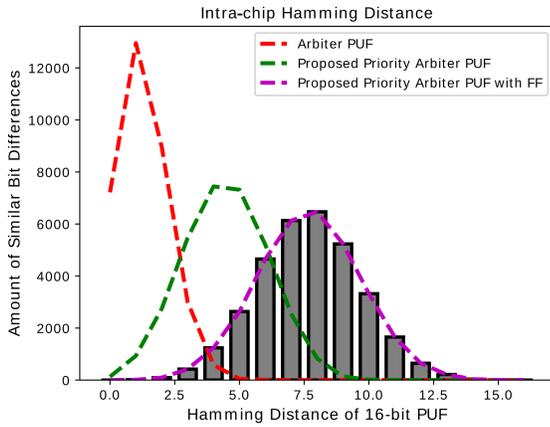


Fig. 11: Hamming distance of the proposed PA-PUF with and without feed-forward arbiters and the classical arbiter PUF.

Fig. 10 that after BCH error correction codes, reliability can increase from 94.5% to 100% in PA-PUF.

D. Machine Learning-based Modelling Attacks

The key purpose of PUFs can be defeated by modeling the PUF structure and by being able to predict its output. In a series of works [22], [9], [10], it has been shown that nearly all variants of delay-based PUFs can be efficiently modeled using machine learning. To further boost security, XOR-based arbiter PUFs are introduced. It was also shown in a recent study [23] that learning XOR-based arbiter PUFs is possible up to a limit on the number of parallel arbiter chains. Complementing this research direction, alternative PUF design frameworks [24] or restricted visibility of the PUF outputs [25] are proposed. Since our proposed design applies to the general studies on arbiter-based PUFs, it will come under the same purview of attacks and resilience presented earlier. Hence, we focus on the lightweight PUF design itself and reserve the study of detailed analysis of modeling attacks for the future. In that context, it will be interesting to juxtapose the priority arbiter design against the XOR-based arbiter chain merger.

IV. COMPARISONS

Based on the performance metrics calculation, the comparisons of the proposed PUF with the existing designs is summarized in Table IV. It is evident from the results that the proposed priority arbiter PUF has good results compared to the existing designs.

TABLE III: Uniqueness and Reliability for various sizes of the proposed PA-PUF.

PUF size	Uniqueness	Reliability
8-bit	50 %	99.45 %
16-bit	50 %	99.05 %
32-bit	50 %	98.84 %
64-bit	48.4 %	98.15 %
128-bit	49.63 %	95.37%

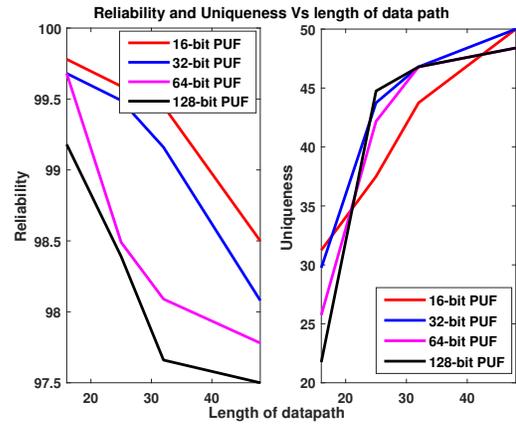


Fig. 12: Comparison of *uniqueness* and *reliability* with respect to data path’s length (16-bit challenge + feed-forward arbiters).

The length of the data path plays a prominent role in deriving the response. In addition to the length of the multiplexer-data path, the number of feed-forward arbiters also plays a key role. Fig. 11 shows the comparison of the classical arbiter PUF, proposed priority arbiter PUF, and the priority arbiter PUF with the feed-forward arbiter. The plot reveals that the classical arbiter PUF is not able to produce a various number of responses while the proposed designs have more number of CRPs. Since the intra-Hamming distance plot has a maximum at half of its response length only for the priority arbiter with the feed-forward arbiter. The number of CRPs in a PA-PUF can be increased by using the feed-forward arbiters.

Further, the number of feed-forward arbiters used as a select signal to the multiplexer in the data path also influences the response. The plot shown in Fig. 12 reveals that as the number of feed-forward arbiters increases in the design, the reliability of the PUF reduces and the uniqueness of the PUF improves. Since the number of CRPs can be increased by increasing the data path length and by introducing more feed-forward arbiters, the PUF is able to generate more variations in the output. Hence, the reliability is going to reduce. So, one can decide the number of feed-forward arbiters as per their requirements of uniqueness and reliability.

V. CONCLUSIONS

This paper proposed a new priority arbiter with a uniform output with an equal probability of 0’s and 1’s. We also proposed a new priority arbiter-based PUF (PA-PUF) with

TABLE IV: Comparison of the performance metrics of the ring oscillator PUF; Ideal value of reliability is 100% while the remaining parameters have an ideal value of 50%.

Parameter %	Proposed PA-PUF	RO PUF	Original	CHAR	CHAR & MAJ
			Design [19]		
Uniqueness	49.63	49.22	48.52	45.60	45.60
Uniformity	49.45	48.5	51.06	50.60	50.54
Reliability	100	99.99	92.00	98.87	99.58
Bit-aliasing	49.66	47.89	43.52	43.52	43.52

three data path lines and a priority arbiter. Further, the feed-forward arbiters are introduced to get new select signals to the multiplexers in the data path to improve the number of challenge-response pairs. Finally, we demonstrated the results of the proposed PA-PUF on Nexys Video Artix-7 FPGA board. The experimental results show that the proposed PA-PUF has a reliability of 94.5%, which can be improved to 100% by implementing error correction techniques. Moreover, the PA-PUF improves the PUF uniformity to 49.45% and uniqueness to 49.63%. We plan to study the attacks on PA-PUF in more detail, practically and theoretically, in the future.

REFERENCES

[1] J. Lee *et al.*, “A technique to build a secret key in integrated circuits for identification and authentication applications,” in *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)*, 2004, pp. 176–179.

[2] R. Pappu *et al.*, “Physical one-way functions,” *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002. [Online]. Available: <https://www.science.org/doi/abs/10.1126/science.1074376>

[3] D. Lim *et al.*, “Extracting secret keys from integrated circuits,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, 2005.

[4] G. E. Suh *et al.*, “Physical unclonable functions for device authentication and secret key generation,” in *2007 44th ACM/IEEE Design Automation Conference*, 2007, pp. 9–14.

[5] J. Guajardo *et al.*, in *FPGA Intrinsic PUFs and Their Use for IP Protection*, ser. Cryptographic Hardware and Embedded Systems - CHES 2007. Springer Berlin Heidelberg, 2007, pp. 63–80.

[6] S. S. Kumar *et al.*, “Extended abstract: The butterfly puf protecting ip on every fpga,” in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 67–70.

[7] D. Suzuki *et al.*, “The glitch puf: A new delay-puf architecture exploiting glitch shapes,” in *Cryptographic Hardware and Embedded Systems, CHES 2010*. Springer Berlin Heidelberg, 2010, pp. 366–382.

[8] A. R. Krishna *et al.*, “MECCA: A Robust Low-Overhead PUF Using Embedded Memory Array,” in *Cryptographic Hardware and Embedded Systems – CHES 2011*, B. Preneel *et al.*, Eds. Berlin, Heidelberg: Springer, 2011, pp. 407–420.

[9] U. Rührmair *et al.*, “Modeling attacks on physical unclonable functions,” in *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, E. Al-Shaer *et al.*, Eds. ACM, 2010, pp. 237–249. [Online]. Available: <https://doi.org/10.1145/1866307.1866335>

[10] U. Rührmair *et al.*, “Puf modeling attacks on simulated and silicon data,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1876–1891, 2013.

[11] D. Ganta *et al.*, “Easy-to-build arbiter physical unclonable function with enhanced challenge/response set,” in *International Symposium on Quality Electronic Design (ISQED)*, March 2013, pp. 733–738.

[12] V. P. Klybik *et al.*, “Use of arbiter physical unclonable function to solve identification problem of digital devices,” *Automatic Control and Computer Sciences*, vol. 49, no. 3, pp. 139–147, May 2015. [Online]. Available: <https://doi.org/10.3103/S0146411615030049>

[13] S. S. Zalivaka *et al.*, “Multi-valued arbiters for quality enhancement of puf responses on fpga implementation,” in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan 2016, pp. 533–538.

[14] T. Machida *et al.*, “A new mode of operation for arbiter puf to improve uniqueness on fpga,” in *2014 Federated Conference on Computer Science and Information Systems*, Sept 2014, pp. 871–878.

[15] G. Dimitrakopoulos *et al.*, “Dynamic-priority arbiter and multiplexer soft macros for on-chip networks switches,” in *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2012, pp. 542–545.

[16] J. Wang *et al.*, “A dynamic priority arbiter for network-on-chip,” in *2009 IEEE International Symposium on Industrial Embedded Systems*, July 2009, pp. 253–256.

[17] R. Bose *et al.*, “On a class of error correcting binary group codes,” *Information and Control*, vol. 3, no. 1, pp. 68–79, 1960. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0019995860902874>

[18] Y. Lao *et al.*, “Statistical analysis of mux-based physical unclonable functions,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 5, pp. 649–662, 2014.

[19] C. Gu *et al.*, “Improved reliability of fpga-based puf identification generator design,” *ACM Trans. Reconfigurable Technol. Syst.*, vol. 10, no. 3, may 2017. [Online]. Available: <https://doi.org/10.1145/3053681>

[20] Y. Dodis *et al.*, “Fuzzy extractors: How to generate strong keys from biometrics and other noisy data,” in *Advances in Cryptology - EURO-CRYPT 2004*. Springer Berlin Heidelberg, 2004.

[21] R. Maes *et al.*, “Pufky: A fully functional puf-based cryptographic key generator,” in *Cryptographic Hardware and Embedded Systems – CHES 2012*. Springer Berlin Heidelberg, 2012, pp. 302–319.

[22] U. Rührmair *et al.*, “Puf modeling attacks: An introduction and overview,” in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2014, pp. 1–6.

[23] F. Ganji *et al.*, “Why attackers win: On the learnability of xor arbiter pufs,” in *Trust and Trustworthy Computing*, M. Conti *et al.*, Eds. Cham: Springer International Publishing, 2015, pp. 22–39.

[24] Y. Wang *et al.*, “Lattice PUF: A strong physical unclonable function provably secure against machine learning attacks,” *CoRR*, vol. abs/1909.13441, 2019. [Online]. Available: <http://arxiv.org/abs/1909.13441>

[25] J. Zhang *et al.*, “Set-based obfuscation for strong pufs against machine learning attacks,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 1, pp. 288–300, 2021.

[26] D. Yamamoto *et al.*, “Uniqueness enhancement of PUF responses based on the locations of random outputting RS latches,” in *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, ser. Lecture Notes in Computer Science, B. Preneel *et al.*, Eds., vol. 6917. Springer, 2011, pp. 390–406. [Online]. Available: https://doi.org/10.1007/978-3-642-23951-9_26

[27] R. Maes *et al.*, “Intrinsic pufs from flip-flops on reconfigurable devices,” in *wissec*, 2008.

[28] D. Merli *et al.*, “Improving the quality of ring oscillator pufs on fpgas,” in *Proceedings of the 5th Workshop on Embedded Systems Security*, ser. WESS ’10. New York, NY, USA: Association for Computing Machinery, 2010. [Online]. Available: <https://doi.org/10.1145/1873548.1873557>

[29] C. Gu *et al.*, “A unique and robust single slice fpga identification generator,” in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 1223–1226.

[30] C. Gu *et al.*, “Ultra-compact and robust fpga-based puf identification generator,” in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 934–937.

TABLE V: Comparisons of different PUF designs; ‘U’ is the uniqueness and ‘R’ is the reliability.

PUF design (response size)	U (%)	R (%)	Platform	Area
Proposed PA-PUF (128)	> 49.63	100	Artix-7	47 slices per PUF
SRAM PUF [5] (128)	49.97	> 88	FPGA	4800 SRAM memory bits
Latch PUF [26] (128)	46	> 87	Spartan-3	2×128 slices
Flip flop PUF [27] (4096)	≈50	> 95	Virtex-2	4096 flip-flops
Butterfly PUF [6] (64)	≈50	94	Virtex-5	130 slices
RO PUF [4] (128)	46.15	99.52	Virtex-4	1024 RO’s
CRO PUF [28] (127)	43.50	96	Spartan-3	64 slices
PUF ID [29] (128)	49.90	93.93	Artix-7	128 slices
Ultra-compact PUF ID [30] (128)	49.93	93.96	Spartan-6	40 slices
Improved PUF ID [19] (128)	45.60	99.42	Spartan-6	128 slices