2022 23rd International Symposium on Quality Electronic Design (ISQED) | 978-1-6654-9466-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISQED54688.2022.9806223

A Lightweight Neighbor-Averaging Technique for Reducing Systematic Variations in Physically Unclonable Functions

Andres Martinez-Sanchez, Deva Borah and Wenjie Che
Klipsch School of Electrical and Computer Engineering
New Mexico State University
Las Cruces, NM, USA
{ amar5150, dborah, wche }@nmsu.edu

***Abstract*—Physically Unclonable Functions (PUFs) are emerging hardware security primitives that leverage stochastic random process variations during chip manufacturing to generate unique secrets. However, the biased systematic variations that exist in the process variations will cause non-random spatial correlations among PUF elements in the layout, which significantly degrades randomness of PUF generated secrets. Existing methods of reducing systematic variations involve operations with high computational complexity and therefore require high implementation overheads. In this paper, we propose a lightweight method based on averaging neighboring PUF values to derive spatial bias and hence reduce spatial variations. Experimental results using RO PUF data from 192 Spartan 3E FPGAs show that the proposed method achieves comparable or even better randomness improvement compared to existing methods. The proposed method also demonstrates advantages of parameter diversity. The proposed method implemented on Xilinx FPGAs shows up to more than 10x lower implementation overhead compared with the existing method.**

Keywords— Physically Unclonable Functions, Systematic Variations, Spatial Correlations, Lightweight PUF Randomness Improvement

I. INTRODUCTION

A growing number of various intelligent Internet-of-Things (IoTs) applications poses a great demand for information processing and storage capability in the model digital systems. However, the increasing complexity of both software and hardware components integrated in such digital systems exposes a much wider attacking surface to adversaries. Moreover, the increasing amount of connected devices to each other and to the Internet has also enlarged the vulnerabilities of compromising data confidentiality, integrity and privacy to attacks. Security mechanisms such as authentication and encryption that are used to achieve security goals typically rely on a secret that is unpredictable by the attackers. Such secrets are conventionally stored in non-volatile memories (NVMs) which have shown to be vulnerable to physical attacks [1][2]. Physically Unclonable Functions (PUFs) have been proposed as a promising alternative for secret generation and storage due to its tamper-evident nature and feature of no-volatile secret generation. A PUF is an integrated circuit that leverages the random process variations during the chip manufacturing process to generate unique and unpredictable secrets for each Integrated circuit (IC) chip. The random and uncontrollable manufacturing variations serve as an entropy source for the PUF to generate unique, reproducible and unpredictable secrets that is "unclonable" by adversaries or even the manufacturer itself.

Unfortunately, process variations during manufacturing not only includes the stochastically random variations but also a second type of systematic variations which demonstrates non-random bias or patterns in the chip physical layout. Such bias or patterns will cause spatial correlations among PUF elements in the physical layout and hence will further reduce the randomness of PUF-generated secrets, posing a threat in PUF-based security applications.

Several works have been proposed to reduce systematic variations including by applying polynomial regressions [5], logarithmic and square root-based operations [6], or by introducing normalized pseudo random number generators [7]. While most of existing works have shown the effectiveness of the methods for reducing systematic variations, they require either complex non-linear mathematical operations or statistical computations which will inevitably occur significant implementation overheads. Such implementation inefficiency would render these methods inappropriate for those resource-constrained IoT applications. Unfortunately, few of existing works has discussed or reported the implementation complexity and overheads of the methods.

In this paper, we propose a lightweight method that only requires simple linear operations to effectively reduce systematic variations in PUFs. The proposed method extracts spatial bias by leveraging information of neighboring PUF elements using a simple averaging operation, which has significantly lower computational complexity and overhead over existing methods. This paper makes the following contributions:

- We propose a lightweight scheme to effectively reduce systematic variations in PUFs by applying a simple linear averaging operation to neighboring PUF cells rather than complex mathematical or statistical methods.
- We evaluated the effectiveness of the proposed method in reducing systematic variations on different types of response generation schemes (coding schemes) using a RO PUF dataset that includes 192 FPGAs. The NIST randomness results show that our scheme has achieved comparative or better improvement on randomness over existing methods.

- We have implemented our scheme in Xilinx FPGAs and report a more than 10x lower implementation overhead over other existing methods. We have also discussed the flexibility/diversity feature of our proposed method provided by its large effective parameter space.

The rest of the paper is organized as follows. Section II presented the related work and Section III introduces the preliminary work. Section IV provides the overall and detailed descriptions of our proposed neighbor-averaging scheme (NAS) and its variants. Experimental evaluations of the effectiveness and overheads are presented in Section V. Section VI concludes the paper.

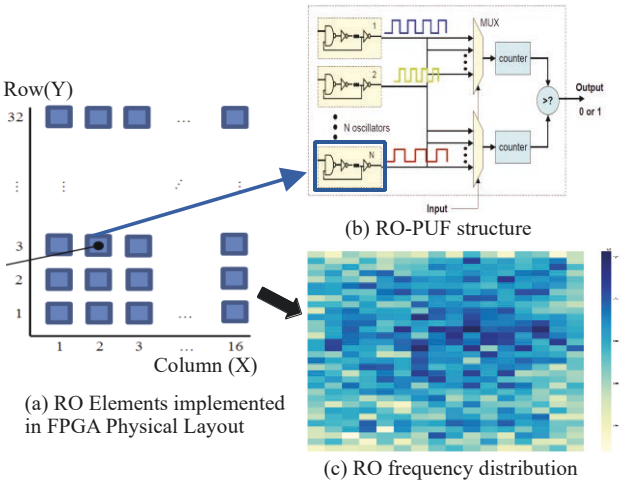
II. Related Work

Several works have proposed methods to reduce systematic variations in PUFs. An early work proposed to select only adjacent RO cells as pairs for generating response bits [11]. The most well-known method in [5] introduced polynomial regressions of varying degrees to approach the systematic bias. Another method uses complex logarithmic base 10 and square root operations which need to be performed using statistical analysis tool like SPSS [6]. Authors in [7] proposed to reduce systematic variations by introducing normalized pseudo random values to the raw RO frequencies which require both costly pseudo random number generators and normalization modules. A recent work introduced a spatial autocorrelation analysis metric for single-challenge PUFs [8]. Different comparison strategies have been proposed to minimize the impact of systematic variations [14] and other works [12][13] improved the PUF entropy by reducing functional correlations through advanced pairing schemes.

III. Preliminaries

A. Basics of RO PUFs

The structure of a conventional RO-PUF is illustrated in Fig. 1(b). The RO-PUF typically consists of *N* identically designed Ring Oscillators that are wired into two *N-to-1* multiplexers (MUXes) so that a pair is selected whose counter (frequency) values are compared. The "Select" signals of the two MUXes are used as the challenge bits to select a particular pair of RO cells whose frequencies are compared. A response bit of '1' (or '0') is generated depending on which frequency is faster.



(a) RO Elements implemented in FPGA Physical Layout

(b) RO-PUF structure

(c) RO frequency distribution

Fig. 1. Systematic variations present in RO frequency distributions of FPGA layout. (a) RO Elements implemented in FPGA layout. (b) RO-PUF structure. (c) RO frequency distribution

B. Systematic variations

Systematic variations refer to the manufacturing bias in the physical layout of chips where groups of PUF elements are implemented. Such physical bias will cause spatial bias present in PUF elements. Fig. 1(a) shows the physical layout of a Xilinx Spartan 3E where 512 RO cells are implemented across 32 rows and 16 column. Fig. 1(c) presents a 2D frequency distribution of these 512 RO cells in the same 2D profile as physical layout. Obvious frequency bias can be observed at the four edges versus the center portion (smaller frequency values are presented at the four edges). Such frequency bias are caused by the systematic variations in the FPGA layout.

C. RO PUF response generation methods (coding schemes)

Similar to other related works, we evaluate several different response generation methods (coding schemes) for RO PUFs in this work. These coding schemes include 1-out-of-8 coding, decouple neighboring coding and T-sequence. Details of these coding schemes are provided in the experimental evaluation section.

IV. Proposed Neighbor Averaging Scheme

A. Overview of the proposed neighbor-averaging scheme

Fig. 2 provides a high-level overview of the proposed neighbor-averaging scheme in comparison with existing polynomial regression based methods. The curves in the boxes on the left and right sides show a region of adjacent RO frequency values before and after applying the method to reduce systematic variations (biased trend), respectively. The middle portion of Fig. 2 illustrates the two major differences of the calculating process between the existing method (top half) and the proposed method (bottom half), i.e., (1) the amount of involved data points in calculation and (2) the complexity of the operations. For the existing regression-based method, a large amount of data points need to be included and the calculation requires complex non-linear operations. For our proposed neighbor-averaging scheme, however, only a small portion of neighboring data points are involved and the calculation only requires simple linear averaging operations on neighboring values. Therefore, the much more simplified calculation process

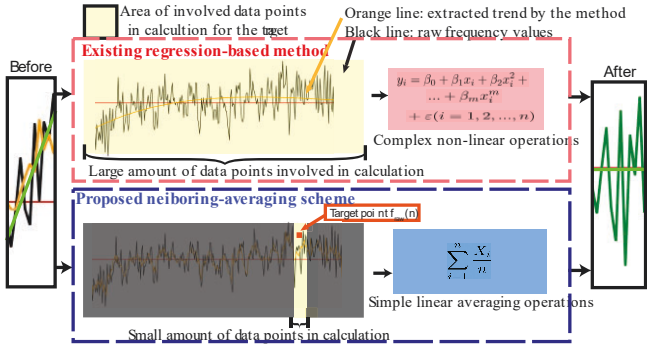


Fig. 2. Overview of the proposed Neighbor Averaging Scheme.

of the proposed scheme brings the advantage of being significantly more lightweight over existing schemes.

The idea of the proposed scheme is to approximate the local trend through calculating local average values. This is achieved by estimating the systematic bias for each individual raw data point through calculating the average of its neighboring data points. These calculated "average values" then are connected to represent the trend line of this data region. The left box of Fig. 2 shows an example. First, the black line connects each raw frequency values in the region. The corresponding systematic bias for each raw value is then computed by averaging its neighboring raw values, obtaining so-called "averaged values". These averaged values are then connected by the orange line to approximate the local trend/bias in this region. To be compared with the actual local trendline represented as the green line, the orange line tracks it very well and therefore can be used to approximate the systematic variations. Finally, the averaged values are subtracted from their corresponding raw values to obtain the new values (shown in the right box of Fig.2) with reduced systematic variations.

B. *Proposed Neighbor-Averaging Scheme (NAS)*

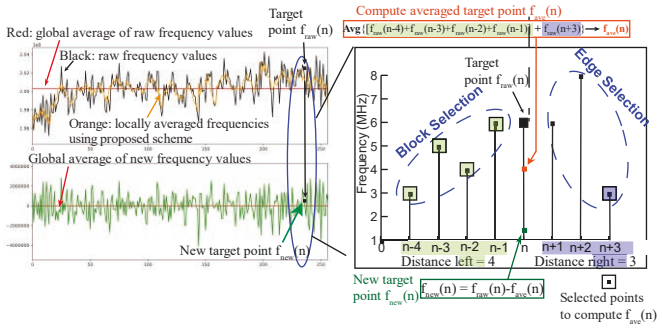


Fig. 3. Explanation of selection method and results of scheme

This section presents a detailed description of the proposed neighbor-averaging scheme. Fig. 3 illustrates the calculation process of approximating the local systematic bias for each specific raw value in the physical layout. Figure 3(a) presents a 1-D representation of the frequency values from 256 RO cells located (16 rows x 16 columns) across the lower half of the Spartan 3E FPGAs, with the upper and lower halves representing the frequency values before (raw) and after (new) applying the proposed scheme respectively. In the upper half of Fig. 3(a), the black line, representing the raw frequencies, shows evident systematic variations as it deviates from the average frequency value (red horizontal line) in a non-uniform manner across sub-regions of the layout. The new frequency values (green line in lower half of Fig. 3(a)) shows that such systematic variations have been significantly reduced, evidenced by the random jumping behavior across the global average line. The reduction of systematic variations in the new frequencies is achieved by subtracting the computed average frequency values (orange line) from the raw frequencies.

For each raw frequency value, a corresponding "averaged value" needs to be calculated for its raw value by averaging a set of selected neighboring raw values. For illustration purpose, we call the raw data point under calculation as the "target point". On each side of the target point, a specific range of neighboring points are selected specified by a distance parameter that indicates the distance range from the target point. For example, a left distance parameter of 4 indicates that up to 4 neighboring points to the left side of the target point are selected. The farthest selected point (4 positions away in this case) from the target point is called the "edge point". In addition to the distance parameters, we also propose two selection methods for neighboring points, i.e., "block selection" and "edge selection". For "block selection", all points between the target point and the edge point (included) are selected, while for "edge selection" only the edge point is selected and all points between are neglected.

Fig. 3(b) illustrates a close-up view of the two selection methods from each side of a target point. First, a target point $f_{raw}(n)$ is selected (indicated by the black square dot). In this particular example illustrated in Fig. 3(b), we use block selection on the left side of the target point with a distance value of 4, which indicates that all 4 points to the left of the target point are selected (denoted by 4 green square dots in Fig. 3(b)). On the right side, the edge selection method is used with a distance parameter value of 3, indicating that only the third point to the right (edge point) of the target point is selected (denoted by the purple square dot in Fig. 3(b)).

Once the neighboring points are selected for a specific target value, they are summed up to calculate the localized average value of the target point, $f_{ave}(n)$. The new target value is then calculated by subtracting the averaged value from the raw target value, i.e., $f_{new}(n) = f_{raw}(n) - f_{ave}(n)$. Fig. 3(b) illustrates this calculation process using an example where the block selection is used on the left side of the target point with a distance value of 4, and edge selection is used on the right side with a distance value of 3. The averaged value $f_{ave}(n)$ is then calculated by summing up and then average 4 data points from the left side and one edge data point to the right side of the target point, shown by the equation on top of Fig.3(b). The calculated average point $f_{ave}(n)$ is denoted as the orange square dot in Fig. 3(b). The final target point value $f_{new}(n)$ is then calculated by subtracting the averaged point $f_{ave}(n)$ from the original frequency point $f_{raw}(n)$, denoted as the green square dot in Fig. 3(b).

C. *Variants and configurations of the proposed method and enhanced flexibility*

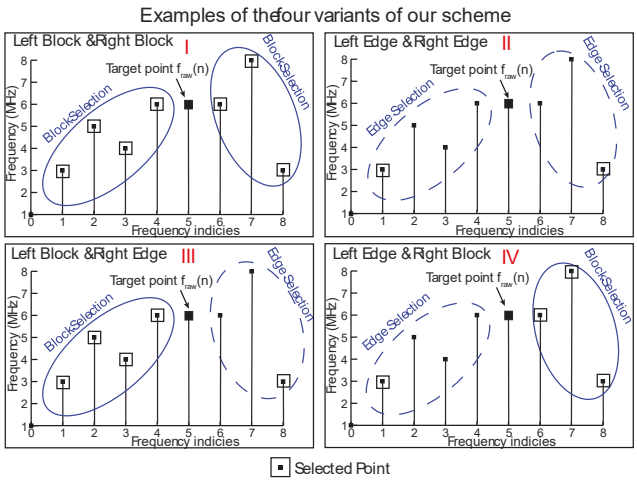


Fig. 4. Illustration of the four variants of our propoesd scheme.

TABLE I. NIST RANDOMNESS TEST RESULTS OF THE PROPOSED NEIGHBORING AVERAGING SCHEME (WITH 3 BEST CONFIGURATIONS) IN COMPARISON WITH DIFFERENT POLYNOMIAL DEGREES OF EXISTING REGRESSION-BASED METHOD

	Scheme	Variants and Parameters	T Test (# of failures out of 9 tests)		1of8 (# of failures out of 9 tests)		Decouple (# of failures out of 9 tests)		Total failures (1of8+Decouple+T)
			Pval	Prop	Pval	Prop	Pval	Prop	
Original	RAW	/	9	9	7	7	1	2	35
Existing methods [5]	Polynomial Regression across all points (in-line)	Order 1	9	9	8	8	1	3	38
		Order 2	8	8	8	8	1	3	36
		Order 3	4	1	8	8	1	3	25
		Order 4	1	1	8	7	2	3	22
		Order 5	0	2	8	7	2	3	22
	Row Based Polynomial Regression	Order 1	8	2	5	1	0	0	16
		Order 2	8	3	5	6	1	0	23
		Order 3	9	5	6	3	0	0	23
		Order 4	9	5	4	2	1	0	21
		Order 5	9	5	4	1	1	3	23
	X-Y based regression in MATLAB	Order 1	9	9	3	0	0	0	21
		Order 2	7	7	3	4	1	0	22
		Order 3	2	2	2	3	1	0	10
		Order 4	0	1	1	0	0	0	2
		Order 5	1	1	0	0	0	0	2
Proposed methods	Best Configurations	Variant II L3/R4	6	2	0	0	2	1	11
		Variant II L4/R5	5	0	3	3	2	0	13
		Variant II L5/R4	4	0	2	2	4	1	13

The edge selection and block selection can be used interchangeably for both sides of the target point, resulting in 4 different combinations. All these 4 combinations construct the 4 variants of our scheme as illustrated in Fig. 4. For each of the 4 variants, a configuration can be determined by specifying the values of the two distance parameters, i.e., the left distance and the right distance. For example, Variant III L4/R3 indicates a configuration that chooses variant III which uses left block selection with a distance of 4, and right edge selection with a distance of 3. The 4 variants as well as the range of left and right distance values provides a large parameter space of the proposed scheme which increases the flexibility of potential adjustment to accommodate PUF designs on different hardware layouts.

V. EXPERIMENTAL EVALUATION

This section presents experimental evaluations on the effectiveness of the proposed method in reducing systematic variations and the implementation overheads. The evaluated results are compared with an existing regression-based method [5].

A. Experimental Setup

We used a public RO PUF frequency dataset [9] that contains 192 Xilinx Spartan 3E FPGA boards. Each FPGA board contains 512 RO frequency values collected from the RO cells implemented as 32 rows by 16 columns in the FPGA layout.

B. NIST Randomness Evaluation

1) Bitstring size for different response generation methods (coding schemes)

We evaluated the randomness of response bitstrings generated by different coding schemes including 1-out-of-8 coding, decoupled neighbor coding and T-Sequence coding schemes.

a) 1-out-of-8 Coding

For the 1-out-of-8 coding, we arrange the frequency values into a 3-bit index from 000,001 up to 111 in the order of slowest to fastest. This allows us to generate 6 bits per row. At 6 bits a row we are able to generate 192 bits per chip and therefore 192 * 192 = 26,864 total bits are generated for 192 chips.

b) Decoupled Neighbor Coding

The decouple neighbor coding is where we look at pairs of frequency values and decide whether the bit is a 1 or 0 depending on which is higher. Using this coding, we are able to generate 8 bits per row. This equates to 8 bits x 32 rows x 192 chips which results in 49,152 bits for our testing sequence.

c) T-Sequence Coding

The T-sequence coding is where we cut each column in half and then compare the respective indices of each half and generate a 1 or 0 depending on which frequency is faster. This allows us to generate 256 bits per chip or 49,152 bits total across all chips. We exclude the S-Sequence coding scheme because most regression-based methods, including our methods, generated abnormal results for S-Sequence coding. Therefore, we only include the T test, 1-out-of-8 test, and the Decoupled Neighbor test for meaningful comparisons.

2) NIST randomness results

Table I reports the results of NIST randomness test for our proposed neighbor-averaging scheme as well as other existing methods and the original scheme with unprocessed raw RO data. In order to fully cover the regression-based methods, we included 3 different variants of the regression-based schemes including the in-line regression, row-based regression and X-Y 2-D based regression. Each scheme has been evaluated using three different coding methods including 1-out-of-8 coding, Decoupled Neighbor coding and the T-sequence coding. We added the last column as the total number of failed NIST tests

across all the three coding methods as an overall performance metric for each scheme.

The third row in Table I shows the NIST randomness results using the original raw RO data before applying any methods. The response bitstring failed most of the T-sequence and 1-of-8 coding schemes and generated a total of 33 failures out of 54 NIST P-value and proportion tests. This is used as the reference to be compared with to show the effectiveness of any method in improving the randomness of PUF response bitstrings.

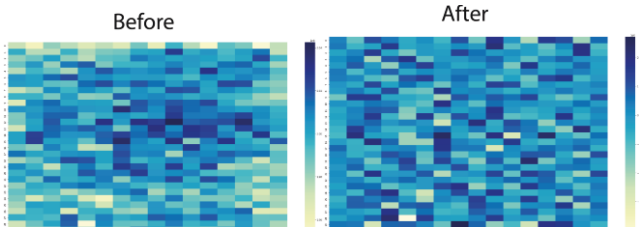


Fig. 5. RO frequency distributions before and after applying our proposed scheme.

The first regression technique, called "in-line" regression, are reported below the original scheme in Table I. The in-line regression applies regressions across all 512 frequency values as a 1-dimensional array. The array is generated by appending each of the 32 rows (each with 16 frequency values) to its previous row. We test this regression method across 6 polynomials degrees. The NIST test results show around 30% improvement over the raw dataset at orders 3 and above. The best result is achieved at polynomial orders 4 and 5 with 22 failures out of 54 tests.

The second type of regression method evaluated is the row-based regression which applies regression to each row of the RO frequency values across 32 rows in the FPGA layout. The best NIST result is achieved at regression order 1 with 16 total failures out of 54 tests.

A third regression method is also evaluated that applies more advanced 2-dimensionsal regression operations to the raw RO frequency values in the layout [3]. The two 1-dimensional regression operations are first applied to the data in the X and Y dimensions respectively followed by a multiplication operation to generate a new 2-D equation *fitsurface(x,y) = p00+p10*x+p01*y + p11*x*y*…. This method is implemented in MATLAB using the CurveFitting Toolbox [10]. As shown in Table I, the higher orders regression (orders 4 and 5) outperforms all other methods in obtaining the least number of failed NIST tests of 2 out of 54 tests. However, the high implementation costs associated with the high degrees of order 4 and 5 (with 7k+ equivalent LUTs reported in the overhead evaluation section) makes this method much less attractive.

We report the NIST results for the best 3 configurations of our scheme across the 4 variants at the bottom of Table I. The best configuration is able to achieve only 11 failures out of 54 NIST tests, which only has as half failures as the best results of 22 failures for the in-line regression method, and has significantly less failures than the best results of row based polynomial regression method of 16 failures. Compared to the least cost-efficient method, 2D regression, we achieved less or comparable number of failures than the lower orders (orders 1 to 3), but more failures than the higher orders of 4 and 5 which occur significantly higher overheads. Our method has also shown dramatic improvement over the 35 failures of the reference design with the unprocessed raw data. A more intuitive demonstration of the effectiveness of our proposed scheme in reducing systematic variations is presented in Fig. 5. The right sub-figure shows a much more uniformly distributed 2D RO frequency distribution after applying our proposed method than that using the reference raw RO frequency values (left figure).

TABLE II. CONFIGURATION DIVERSITY OF THE PROPOSED SCHEME: NUMBER (#) OF DIFFERENT CONFIGURATIONS THAT ACHIEVED NO MORE THAN 2 FAILURES

# of configurations with x Failures (p-value/proportion) out of 9 NIST tests	1of8	Decouple	Both
x=0	1	12	2
x=1	15	52	35
x=2	8	5	28
TOTAL	24	69	65

3) Configuration Diversity of the proposed method

Focusing on 1-out-of-8 and decoupled neighbor coding, we found that our scheme has up to 65 different configurations that are able to achieve no more than 2 failures out of 9 P-value and 9 proportion NIST tests, as shown in Table II. Specifically, as many as 24 different configurations are able to achieve no more than 2 failures for 1-out-of-8 coding, and 69 different configurations achieved for the decoupled neighbor coding.

C. Overhead Evaluation and Comparison

A typical hardware cost of implementing different orders of polynomial regression on Xilinx Virtex-6 platform is reported in lower portion of Table III [4]. The equivalent number of LUTs (including those converted from DSP blocks) is used as the metric to evaluate the hardware cost.

For a comparable comparison, we implemented the 3 configurations with best NIST results (see Table I) of our proposed scheme in hardware (VHDL) on the same Xilinx Virtex-6 FPGA platform. The resource utilization of the proposed method as well as the comparison are reported in Table III. The lower portion of Table III reports the estimated converted resource utilization reported in Fig. 6 [4]. For the 3 best configurations of our proposed method, the overheads are reported in the upper half of Table III with 111 LUTs, 76 FFs and 1 Carry element, respectively. For the regression-based scheme of orders 2-6, however, the number of equivalent LUTs consumed are significantly higher up to several thousands, as reported in the lower portion of table III. For the 2D polynomial regression with degrees of 4 and above that achieved best NIST results of 2 failures (shown in Table I), the amount of equivalent utilized LUTs is as high as 7k+ which is more than 50x of our proposed scheme. For the polynomial degree of 3 that achieve comparable NIST results of 10 failures as our proposed scheme, it also requires 4K+ equivalent LUTs which is more than 10x overhead than the proposed scheme.

TABLE III. IMPLEMENTATION OVERHEAD OF THE PROPOSED SCHEME ON XILINX VIRTEX-6 FPGAS IN COMPARISON WITH EXISTING METHODS

	Configurations of proposed scheme	# LUTs	# FFs	# CARRY
Proposed scheme (3 configurations with best NIST results)	Variant-II, L3R4	111	76	1
	Variant-II, L4R5	110	75	1
	Variant-II, L5R4	110	75	1
Regression-based methods (polynomial degrees) [4]	Degree 2	1k+ ~7k	/	/
	Degree 3	4k+ ~11k	/	/
	Degrees 4-6	7k+ ~25k	/	/

VI. CONCLUSION

We propose a lightweight technique that can effectively reduce systematic variations in the physical layout where PUFs elements are implemented. The proposed method avoids complex computations by applying simple linear operations to a small number of neighboring PUF elements to derive local bias/trend, hence significantly reduces the implementation overheads. Experimental results have shown the effectiveness of our proposed scheme in improving PUF randomness, and hardware implementation cost has been reported as more than 10x lower over existing methods. The flexibility feature provided by the large effective parameter space of the proposed method can be potentially leveraged as a reconfiguration feature that enables dynamic re-enrollment on the same PUF instance.

ACKNOWLEDGMENT

This work is supported in part by the National Science Foundation under Grant 1914635.

REFERENCES

[1] P. Tuyls, G.-J. Schrijen, B. Skoric, J. van Geloven, N. Verhaegh, and R. Wolters, “Read-Proof Hardware from Protective Coatings,” Proc. Eighth Int’l Workshop CHES ’06, vol. 4249, pp. 369-383, Oct. 2006.

[2] S. H. Weingart, “Physical security devices for computer subsystems: A survey of attacks and defenses,” in Proc. Cryptographic Hardware Embedded Syst (CHES)., 2000, pp. 302–317.

[3] Polynomial regression. (2020). In Wikipedia. https://en.wikipedia.org/w/index.php?title=Polynomial_regression&oldid=996324225

[4] S. Xu, S. A. Fahmy, and I. V. McLoughlin, “Square-rich fixed point polynomial evaluation on FPGAs,” in Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA), Monterey, CA, USA, 2014, pp. 99–108.

[5] CE Yin and Q. Gang, “Improving PUF Security with Regression-based Distiller,” Design Automation Conference (DAC), Jun 2013.

[6] M. Mustapa, M. NiamatNovel, “Novel RPM Technique to Dismiss Systematic Variation for RO PUF on FPGA,” Proc. NAECON, 2014.

[7] F. Amsaad, A. Prasad, C. Roychaudhuri and M. Niamat, “A novel security technique to generate truly random and highly reliable reconfigurable ROPUF-based cryptographic keys,” in 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, 2016, pp. 185-190.

[8] F. Wilde, B. M. Gammel, and M. Pehl, “Spatial correlation analysis on physical unclonable functions,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 6, pp. 1468–1480, June 2018.

[9] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, “A large scale characterization of RO-PUF,” in Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST), Jun. 2010, pp. 94–99. https://github.com/patrickschaumont/ropuf_host2010

[10] MathWorks Curve Fitting Toolbox, https://www.mathworks.com/help/curvefit/index.html?s_tid=CRUX_lftnav

[11] A. Maiti and P. Schaumont, ‘‘Improving the quality of a physical unclonable function using configurable ring oscillators,’’ in Proc. IEEE Int. Conf. Field Program. Logic Appl., Aug./Sep. 2009, pp. 703–707.

[12] C.E. Yin and G. Qu, “Lisa: Maximizing RO PUF’s Secret Extraction,” in Hardware Oriented Security and Trust (HOST), pp. 100-105, Jun. 2010.

[13] R. Valles-Novo, A. Martinez-Sanchez, and W. Che, "Boosting Entropy and Enhancing Reliability for Physically Unclonable Functions." Asian Hardware Oriented Security and Trust Symposium (AsianHOST). IEEE, 2020, pp. 1-6.

[14] W. Liu, Y. Yu, C. Wang, Y. Cui, and M. O’Neill, “RO PUF design in FPGAs with new comparison strategies,” in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2015, pp. 77–80.