THE INSTITUTE FOR SYSTEMS RESEARCH

ISR TECHNICAL REPORT 2011-12

A Regression-Based Entropy Distiller for RO PUFs

Chi-En Yin and Gang Qu

ISR develops, applies and teaches advanced methodologies of design and analysis to solve complex, hierarchical, heterogeneous and dynamic problems of engineering technology and systems for industry and government.

ISR is a permanent institute of the University of Maryland, within the A. James Clark School of Engineering. It is a graduated National Science Foundation Engineering Research Center.

www.isr.umd.edu

A Regression-Based Entropy Distiller for RO PUFs

Chi-En Yin and Gang Qu
Department of Electrical and Computer Engineering & Institute for Systems Research
University of Maryland, College Park, USA
{chienyin,gangqu}@umd.edu

***Abstract*—Silicon physical unclonable functions (PUF) utilize the variation during silicon fabrication process to extract information that will be unique for each chip. There have been many recent approaches to how PUF can be used to improve security related applications. However, it is well-known that the fabrication variation has very strong spatial correlation and this has been pointed out as a security threat to silicon PUF. In fact, when we apply NIST's statistical test suite for randomness [1] against the random sequences generated from a population of 125 ring oscillator (RO) PUFs [2] using classic 1-out-of-8 Coding [3], [4] and Neighbor Coding [5], none of them can pass all the test. In the paper, we propose to decouple the unwanted systematic variation from the desired random variation through a regression-based distiller, where the basic idea is to build a model for the systematic variation so we can generate the random sequences only from the true random variation. Applying Neighbor Coding to the same benchmark data [2], our experiment shows that 2^{nd} and 3^{rd} order polynomials distill random sequences that pass all the NIST randomness test, so does 4^{th} order polynomial in the case of 1-out-of-8 Coding, which demonstrates that our method can bolster the security characteristics of existing PUF schemes.**

***Index Terms*—ring oscillator (RO), physically unclonable functions (PUFs), random number generator (RNG), linear regression, variation decomposition**

I. Introduction

A. Overview

One of the most renowned principles for the design of a cryptosystem is Kerckhoff's law: "A cryptosystem should be secure even if everything about the system, except the key, is public knowledge. (1883)" In order to provide a secure storage for cryptographic keys, contemporary tamper-resistant devices such as smart cards arm themselves with a number of countermeasures to defeat various kinds of invasive, semi-invasive and non-invasive physical attacks. Nevertheless, it is still possible for attackers to read, and possibly write, the secret bits in the non-volatile memory through the electron beam of a Scanning Electron Microscope (SEM) once the surface of the chip is exposed by, for instance, Focused Ion Beam (FIB). Physical unclonable functions (PUFs), in contrast, are 'inseparable' because the underlying nano-scale structural disorder will most likely be damaged during the course of physical tampering of the device, so will the keys. Since the first introduction of PUFs, many types of circuitry have been proposed to realize the notion. Most notable are Arbiter PUFs [6], RO PUFs [3] and SRAM PUFs [7], [8]. Many methodologies have been proposed to advance the art in terms of reliability, security, and hardware efficiency. Some researchers further classify PUFs as 'Strong', 'Controlled', and 'Weak' mainly according to the number of challenge-response pairs (CRPs) a PUF can generate [9][1]. Considering that a large number of CRPs can be achieved through a keyed hash function seeded by a Weak PUF, we chooses Weak PUFs as the context of the discussion, though the proposed methodology is expected to work well with Strong PUFs in a similar fashion. Figure 1 outlines the typical workflow of a Weak RO PUF that involves the following steps.

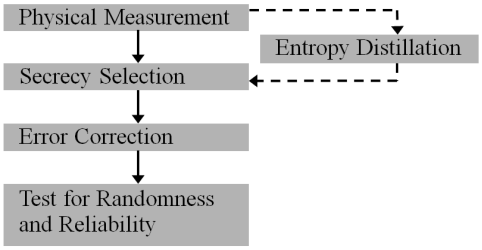

Fig. 1. The typical workflow of a Weak RO PUF

1) Fabrication Variation Extraction: The very first task of PUFs is to measure the unique characteristics endowed from the uncontrollable fabrication process. The analog-to-digital transformation is part of the physical entropy source subject to tests and in our case, this step corresponds to a full frequency characterization of a RO array [2].

2) Secret Selection: This step selects secure and reliable secrecy out of the variation profile measured in the previous step. Existing approaches include the classic 1-out-of-8 Coding [3] and its successor Index-Based Syndrome (IBS) Coding [4], Chain-like Neighbor Coding [5], [2], [10], Temperature-Aware Cooperative (TAC) Coding [11] and Group-Based Coding [12].

3) Error Correction: To further enhance reliability, error correcting code (ECC) may be applied. Codes have been used for RO PUFs include Hamming Code and BCH Code [3].

4) Tests for Randomness and Reliability: The security aspects of a RNG can be judged by the statistical characteristics of the random sequences it produces. The NIST test suite [1] is regarded as an industrial standard to test cryptographic RNGs and thus applicable to our cases. Reliability, on the other hand, can be gauged by placing the device under extreme conditions for secret regeneration and failure rate below 1 part per million

[1]The words *weak* and *strong* are irrelevant to the strength of PUF security [9].

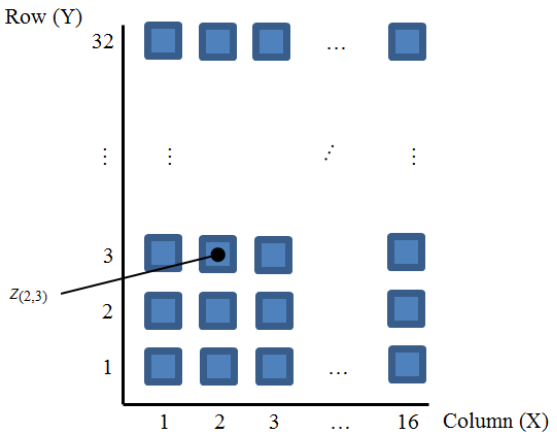

Fig. 2. The placement of 512 ROs as a 16 (columns) by 32 (rows) array; for site $RO_{x,y}$, its running frequency is labeled $z_{x,y}$

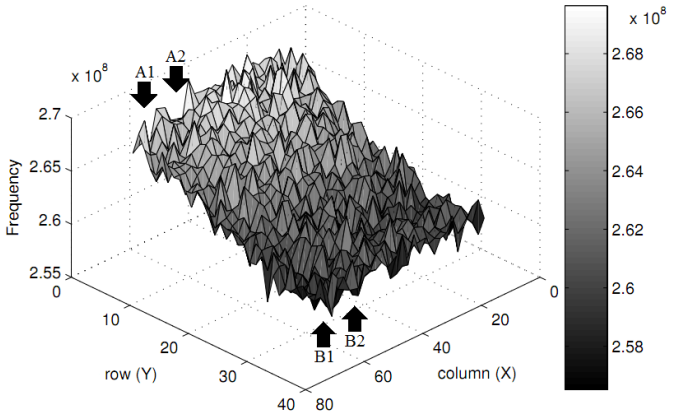

Fig. 3. The across-die frequency topology of a RO array. The roughness of the surface represents the random variation while the slope represents the systematic [13]

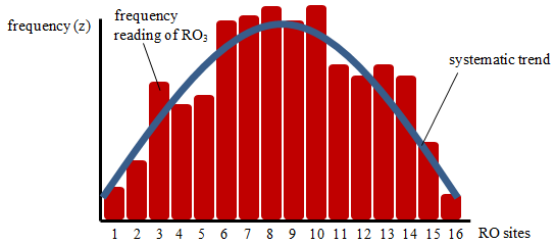

Fig. 4. Illustration of the impact from systematic variation even after pairs are decoupled.

(ppm) has been reported under severe fluctuation of ambient temperature and supply voltage [4].

If we set aside the issue of reliability at a moment, one may think that random sequences generated by well-known secret selection strategies such as 1-out-of-8 Coding and Neighbor Coding should pass the NIST randomness tests. To our surprise, based on the frequency characterization collected from 125 FPGA devices [2], no random sequence actually pass all tests that are applicable to their length, see Table I. The possible causes are discussed next.

II. Security Analysis

A. Failure Cause 1: Chain Dependency

The high failure rate of Chain-like Neighbor Coding can be attributed to the non-independent comparison chain. Take 3 ROs RO_A, RO_B and RO_C for example, two random bits are generated by comparing RO_A with RO_B and RO_B with RO_C. As we know, to pass NIST test for randomness, the random sequence is expected to demonstrate no significant deviation from the probability mass function (p.m.f) of tossing a fair coin twice, i.e., the 4 possible outcomes '00', '01', '10' and '11' are expected to equally likely with probability 1/4. In fact, it is not case for the two bits we generate from the 3 ROs, assuming that the 6 outcomes $RO_A < RO_B < RO_C$, $RO_A < RO_C < RO_B$, $RO_B < RO_A < RO_C$,..., $RO_C < RO_B < RO_A$ are equally likely with probability exactly $1/6$. In turn, the probability of the outcome '00', '01', '10' and '11' of the 2-bit random sequence is $1/6$, $1/3$, $1/3$ and $1/6$ respectively, a clear deviation from the p.m.f of the 'ideal' random sequence; consequently, the min-entropy is actually 1.58 rather than 2, our first thought. One solution is to break up the chain such that all ROs only pair up with its neighbor once, i.e., (RO_A, RO_B), (RO_C, RO_D).... As Table I shows, the decoupling improves the pass rate but still is not good enough to pass all.

B. Failure Cause 2: Spatial Correlation

Generally RO PUFs places its ROs as a 2-D array. The dataset we use lays out 512 ROs as a 16 (columns) by 32 (rows) grid on each of the 125 FPGA devices [2] as illustrated in Figure 2, where $z_{x,y}$ denotes the running frequency of RO at site $RO_{x,y}$. One may conjecture that we can generate 1-bit secrecy out of any RO pair; however, it is not secure if we consider about the underlying spatial correlation. Figure 3 shows how the fabrication variation of the semiconductor process portrays: The roughness of the surface (random variation) is superimposed upon a spatial trend (systematic variation). The systematic component can significantly reduce the min-entropy of the extracted secrecy; for instance, if one generates bit A as '0' if $A1 < A2$, else '1' and similarly, bit B '0' if $B1 < B2$, else '1', spatial correlation would render $p(A = B) >> p(A \neq B)$.

While spatial correlation may explain the reason why none of the coding strategies in Table I passes all tests, it is interesting to note that in fact the threat was acknowledged by Chain-like Neighbor Coding [5]. This is exactly the reason behind its design principles: 1) place ROs as close as possible and 2) pair ROs located adjacent to each other. Their key idea is to let the systematic effect cancel out with each other, extracting secrecy out of the random effect. We see the principles have been accepted by [10]. Nevertheless, to explain those failure cases we postulate that the small remnant of the systematic effect can still be captured by the test. To illustrate, Figure 4 shows a hypothetical frequency characterization of 16 consecutive ROs. If, say, frequency relation $z_i < z_{i+1}$ gives

1-out-of-8		**Chain-like Neighbor**		**Decoupled Neighbor**		
P-VALUE	PROPORTION	P-VALUE	PROPORTION	P-VALUE	PROPORTION	STATISTICAL TEST
0.013689	122/125	0.000072 *	125/125	0.000003 *	115/125 *	Frequency
0.166594	125/125	0.000000 *	125/125	0.050764	120/125	BlockFrequency
0.231636	121/125	0.000000 *	125/125	0.000000 *	119/125 *	CumulativeSums (m-2)
0.059743	122/125	0.000000 *	125/125	0.000000 *	118/125 *	CumulativeSums (m-3)
0.002320	117/125 *	0.000000 *	0/125 *	0.302788	120/125	Runs
0.000603	123/125	0.000000 *	62/125 *	0.000062 *	124/125	LongestRun
0.000001 *	117/125 *	0.000000 *	0/125 *	0.000001 *	119/125 *	ApproximateEntropy
0.004904	124/125	0.000000 *	1/125 *	0.070160	116/125 *	Serial (forward)
0.552185	125/125	0.000000 *	117/125 *	0.192277	123/125	Serial (backward)

TABLE I

NIST TEST RESULTS WITH RESPECT TO THE RANDOM SEQUENCES GENERATED BY **1-OUT-OF-8 CODING**, **CHAIN-LIKE NEIGHBOR CODING** AND **CHAIN-LIKE NEIGHBOR CODING**. FOR **1-OUT-OF-8 CODING**: $M = 32$ FOR BLOCK FREQUENCY TEST, $m = 1$ FOR APPROXIMATE ENTROPY TEST AND $m = 4$ FOR SERIAL TEST. FOR **CHAIN-LIKE NEIGHBOR CODING** AND **DECOUPLED NEIGHBOR CODING**: $M = 32$ FOR BLOCK FREQUENCY TEST, $m = 2$ FOR APPROXIMATE ENTROPY TEST AND $m = 5$ FOR SERIAL TEST. '*' MARKS A FAILURE.

us a '0', else '1', it is likely for the up slope to yield more '0's than '1's and vice versa for the down slope. The same argument can also explain the failures we found in 1-out-of-8 coding strategy, or more generally, 1-out-of-k coding strategy; when $k = 2$, it reduces to the proposed Decoupled Neighbor Coding. Lastly, we stress that the systematic trend can stay undiscovered when one tallies the total number of '0's and '1's or even when one calculates the inter-die uniqueness [3].

C. A Cautious Note on Spatial Correlation

As Figure 3 above, Figure 7 in [7] and the failures in Table I argue, raw PUF measurements cannot be assumed as i.i.d.; nevertheless, assuming PUF output i.i.d. is not uncommon in literature and has been affirmed by statistical results [4]. The key difference appears to be our direct test on raw output as opposed to those obfuscated by a linear feedback shift register (LFSR) and/or an output hash function that de-correlate the challenge and the response [14], [7], [4]. Secondly, the issue of spatial correlation may be more serious than we thought because across-die spatial variation mostly results from deterministic across-wafer variation [15], that is, once process parameters surrounding chips under attack get exposed, the opponent can gain unexpected advantage to model the systematic trend and attack the min-entropy we over-estimate. Next, we describe how to eliminate the systematic variation upfront.

III. SYSTEMATIC VARIATION ELIMINATION

The goal is to model the unwanted systematic variation so that we can decouple it from the desired random variation; most importantly, this section shows how the distillation process in Figure 1 can strengthen PUF output. Due to its simplicity, we apply polynomial regression to model the systematic trend. Indeed, the method can fix all the failures of 1-out-of-8 Coding and Decouple Coding in Table I.

A. The Causes of Process Variation

The main causes of the systematic variation can be attributed to equipment and process nonuniformity such as the focus shift of photolithography, the gradient of thermal annealing, dissimilar interactions between circuit layout and the chemical mechanical polishing (CMP) process. On the other hand, the random component accounts for the difference between those estimates and the observed data; the constituents include unidentified patterns, measurement errors and most importantly, atomic-level stochastic phenomena such as random dopant profiles.

B. Polynomial Regression

Polynomial regression is a form of linear regression in which the relationship between independent variables and a dependent variable is modeled by a polynomial of order k, where k is a non-negative integer. For a RO PUF with its m ROs arranged as r rows by c columns, the Cartesian coordinate (x, y) of ROs is regarded as two independent variables and the oscillating frequency z is the single dependent variable. In such a 2D setting, our polynomial regression model of order k, i.e., $x^p y^q, p + q \leq k, 0 \leq p, q \leq k$, takes the general form of

$$z_{x,y} = \sum_{i=0}^{k} \sum_{j=0}^{i} \beta_{k,i,j} x^{i-j} y^j + \epsilon_{k,x,y} \tag{1}$$

where $1 \leq x \leq c, 1 \leq y \leq r; z, \beta, \epsilon \in \mathbb{R}$. On the right hand side of the equation, the summation term models the systematic variation and the residual term $\epsilon_{k,x,y}$ models the random variation. Equivalently, they can be written in matrix form

$$\boldsymbol{Z} = \boldsymbol{\Omega_k} \boldsymbol{\beta_k} + \boldsymbol{\epsilon_k} \tag{2}$$

where

$$\boldsymbol{\Omega_k} = \begin{pmatrix} \omega_{k,1,1} & \omega_{k,1,2} & \cdots & \omega_{k,1,n} \\ \omega_{k,2,1} & \omega_{k,2,2} & \cdots & \omega_{k,2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{k,m,1} & \omega_{k,m,2} & \cdots & \omega_{k,m,n} \end{pmatrix},$$

$$\boldsymbol{Z} = \begin{pmatrix} z_{1,1} \\ \vdots \\ z_{c,1} \\ z_{1,2} \\ \vdots \\ z_{c,2} \\ \vdots \\ z_{1,r} \\ \vdots \\ z_{c,r} \end{pmatrix}, \boldsymbol{\beta_k} = \begin{pmatrix} \beta_{k,0,0} \\ \beta_{k,1,0} \\ \beta_{k,1,1} \\ \beta_{k,2,0} \\ \beta_{k,2,1} \\ \beta_{k,2,2} \\ \vdots \\ \beta_{k,k,0} \\ \vdots \\ \beta_{k,k,k} \end{pmatrix}, \boldsymbol{\epsilon_k} = \begin{pmatrix} \epsilon_{k,1,1} \\ \vdots \\ \epsilon_{k,c,1} \\ \epsilon_{k,1,2} \\ \vdots \\ \epsilon_{k,c,2} \\ \vdots \\ \epsilon_{k,1,r} \\ \vdots \\ \epsilon_{k,c,r} \end{pmatrix},$$

$m = r \times c$, $n = \frac{(k+1)(k+2)}{2}$ and $\omega_{k,p,q} = x^{i-j}y^j$ in which $x = ((p-1) \mod r) + 1$, $y = \lfloor \frac{p-1}{r} \rfloor + 1$, $i = \lfloor \frac{-1+\sqrt{1+8(q-1)}}{2} \rfloor$, $j = (q-1) - \frac{i^2+i}{2}$, $1 \le p \le m$ and $1 \le q \le n$. For each model of order k, it is an overdetermined system, i.e., $m > n$, and can be solved by the ordinary least squares (OLS) method, which produces the 'best' estimates $\hat{\boldsymbol{\beta}}$ in the sense of minimum sum of squared errors as (3) indicates. By taking partial directives of (4) with respect to each $\beta_{k,i,j}$ and letting each gradient to zero, the solution of OLS can be expressed as (5) in matrix form.

$$\hat{\boldsymbol{\beta}}_{\boldsymbol{k}} = \arg\min_{\boldsymbol{\beta_k}} \left\{ \sum_{x=1,y=1}^{c,r} \epsilon_{k,x,y}^2 \right\} \tag{3}$$

$$= \arg\min_{\boldsymbol{\beta_k}} \left\{ \sum_{x=1,y=1}^{c,r} (z_{x,y} - \sum_{i=0}^{k} \sum_{j=0}^{i} \beta_{k,i,j} x^{i-j} y^j)^2 \right\} \tag{4}$$

$$= (\boldsymbol{\Omega_k^T \Omega_k})^{-1} \boldsymbol{\Omega_k^T Z} \tag{5}$$

C. Model Selection

The higher the order we use, the less the error in the least squares sense results. Therefore, a model in high order is expected to yield less residual terms, that is, increase the difficulties in error control and eventually reduce the efficiency of RO PUFs. A model in low order, on the other hand, may not be able to capture all deterministic variation in the systematic term and in turn weaken the security. Due to these concerns, the goal of our model selection is to find out the minimal (optimal) order of the polynomial-based entropy distiller through which the output bitstrings demonstrate strong randomness. Two tasks remain: i) construct random sequences highlighting underlying spatial correlation, and ii) apply effective tests for randomness on the sequences. For simplicity, we pick an order and find a solution against that model for each PUF device before considering another order.

1) Random Sequence Generation: There are numerous ways to generate a random sequence from the frequency profile of a RO array. As shown earlier, secret selection schemes are design specific. Instead of exhausting all possibilities, we try to devise general-purpose random sequences and use them to gauge the existence of spatial correlation in the distilled random component. For this reason, we purposely pair up ROs at locations far apart, simply opposite to the design principle of Neighbor Coding whose goal is to avoid spatial correlation. Also wanted is the length of sequence to be greater than certain minimum values so that certain tests can bear statistical significance. As a result, two indicator random sequences $\boldsymbol{S}$ and $\boldsymbol{T}$ are formulated according to rules (6) and (7) respectively. Simply put, we cut the array into two subsets in the middle along the axis and form pairs with two elements, one subset each, in distance half the corresponding side length of the given rectangle RO array.

$$\boldsymbol{S} = X_1, \ldots, X_{l_X}, \ldots, X_{L_X} \text{ where } X_{l_X} = \begin{cases} 0 & \text{if } z_{u_X,v_X} \le z_{u_X+\lfloor \frac{c}{2} \rfloor, v_X} \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

$$\boldsymbol{T} = Y_1, \ldots, Y_{l_Y}, \ldots, Y_{L_Y} \text{ where } Y_{l_Y} = \begin{cases} 0 & \text{if } z_{u_Y,v_Y} \le z_{u_Y,v_Y+\lfloor \frac{r}{2} \rfloor} \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

where in (6), $u_X = ((l_X - 1) \mod \lfloor \frac{c}{2} \rfloor) + 1, v_X = \lfloor (l_X - 1)/\lfloor \frac{c}{2} \rfloor \rfloor + 1, 1 \le l_X \le L_X = r \times \lfloor \frac{c}{2} \rfloor$; similarly in (7), $u_Y = \lfloor (l_Y - 1)/\lfloor \frac{r}{2} \rfloor \rfloor + 1, v_Y = ((l_Y - 1) \mod \lfloor \frac{r}{2} \rfloor) + 1, 1 \le l_Y \le L_Y = c \times \lfloor \frac{r}{2} \rfloor$.

2) Tests for Randomness: An 'ideal' random sequence is regarded as the result of consecutive flips of a fair coin: when a head turns out, denote the outcome as, say, '1' and if it is a tail, then '0', while both events have probability exactly $1/2$ and each toss is independent of one another [1]. According to the NIST hypothesis testing, a RNG under test is deemed 'bad' if the statistical properties of its random sequences indicate a clear deviation from the 'ideal', otherwise it is deemed 'good'. In our case, given a polynomial order and a secrecy selection scheme, the random sequence subject to the NIST test comprises $N \times L$ bits, where N is the total number of PUFs and L is the output length for each PUF.

IV. Experimental Results

Our test bench is the frequency characterization of a population of RO PUFs implemented on 125 Xilinx Spartan-3 FPGA devices. For each device, 512 ROs were placed as a 16×32 array and for each RO, 100 frequency readings were measured at room temperature [2]. Readers are referred to Appendix for graphical demonstration of the modeled systematic variations, distilled random variations, and the histograms of the random parts.

A. Model Selection by NIST's Test Suite for Randomness

1) Random Sequences $\boldsymbol{S}$ and $\boldsymbol{T}$: For each RNG, the input file fed to the test suite contains 32000 bits, a concatenation of the 256-bit bitstrings generated from Chip No. 1–125. Tests applicable to this length include Frequency Test, Block Frequency Test, Cumulative Sums Test, Runs Test, Longest Run Test, Serial Test and Approximate Entropy Test. Parameters are chosen according to NIST recommendations, in particular, block length $M = 32$ for Block Frequency Test, block length $m = 2$ for Approximate Entropy Test and block length $m = 5$ for Serial Test. According to [1], empirical results have to

be interpreted in two forms of analysis: First, the proportion of sequences passing a test shall be above a the minimum rate, 0.96 in our case, i.e., to pass 120 sequences out of a sample size of 125 sequences at significance level $\alpha = 0.01$. Secondly, the *P-values* of all the random sequences shall be uniformly distributed. Based on χ^2 Goodness-of-Fit Test, the underlying distribution is deemed uniform if the *P-value* of the *P-values* is equal or greater than 0.0001 given a population of 125 sequences. Whenever either of these two approaches fails, further analysis drawn from a different sample space is necessary to conclude the failure either as a statistical anomaly or a clear non-randomness. The left four columns of Table II summarizes the test results of random sequences $\boldsymbol{S}$ and $\boldsymbol{T}$; see Appendix Table III and IV for complete C1–C10 distribution of 'P-VALUE OF P-VALUES'. As the 0^{th}-order section shows, random sequences generated without entropy distillation fail miserably for both forms of analysis 'PROP. (PROPORTION)' and 'P-VAL. (P-VALUE OF P-VALUES)', where '*' marks a failure. This strongly suggests the existence of systematic variation in the raw data. The failure rate decrease sharply when applied with 1^{st}-, 2^{nd}- or 3^{rd}-order distiller in the case of $\boldsymbol{S}$ and with 2^{nd}- or 3^{rd}-order distiller in the case of $\boldsymbol{T}$. Unfortunately, there is at least one failure with respect to $\boldsymbol{S}$, though the failure is only slightly below the cutting value. In such a boarder case where a weak existence of systematic variation is inferred, further investigation with different dataset is necessary to conclude the RNGs and the corresponding distillers 'good' or 'bad'. If simply taking the sum of failure rates with respect to $\boldsymbol{S}$ and $\boldsymbol{T}$, either 2^{nd}- or 3^{rd}-order distillers can be considered optimal. Moreover, the pass rate of the 'P-VALUE OF P-VALUES' analysis drops when applied with a model in 4^{th} order or beyond, whereas the pass rate of the 'PROPORTION' analysis remains unchanged. The rate drop is due to the clustering phenomenon of *P-values* towards the C10 side, as opposed to the clustering phenomenon towards the C1 side in the low order cases. These can be used as indicators for model over-fitting and model under-fitting respectively.

2) 1-out-of-8 Coding: For 1-out-of-8 Coding, a 3-bit index '000', '001',...,'110', or '111' is generated by pointing to the fastest RO out of 8 consecutive ROs in the same row, resulting a random sequence of length 192 bits per device. Table II shows that distillers of 4^{th} order and beyond are deemed 'good' but it is not clear to us why the failure rate increases until 3^{rd} order and drops suddenly. The performance of 1-out-of-8 Coding is generally worse when applied column-wise for both forms of analysis; failures persist throughout the models we consider in this work.

3) Neighbor Coding: In the case of Chain-like Neighbor Coding, 15 bits are generated per row by pairing up row neighbors and a total 480-bit random sequence results per device. As shown in the fourth column of Table II, none of our distillers can make meaningful improvement. The phenomenon aligns with our expectation for the failures are caused by the intrinsic chain dependencies of the pairing strategy rather than spatial correlation. The argument also draws support from the pass rate of Decoupled Neighbor Coding enhanced by a distiller in 1^{st} order or beyond. Overfitting is mild compared with other coding strategies. Like 1-out-of-8 Coding, pairing column-wise yields worse pass rate regardless chained or not; results are omitted for brevity.

V. Conclusion

The systematic component of fabrication variation has long posted a security threat to RO PUFs. This work provides experimental data to demonstrate that none of the current coding schemes can pass all the NIST randomness tests. To address the issue, we propose a family of entropy distillers based on polynomial regression. We affirm their effectiveness in improving the randomness of the PUF output. Indeed, with our enhancements, the discussed coding strategies can pass all the NIST tests.

References

[1] J. N. M. S. E. B. S. L. M. L. M. V. D. B. A. H. J. D. S. V. Andrew Rukhin, Juan Soto and L. E. B. III, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," *NIST Special Publication 800-22 Revision 1a*, Apr. 2010.

[2] A. Maiti and P. Schaumont, "A large scale characterization of ro-puf," *Proceedings of 3rd IEEE International Workshop on Hardware Oriented Security and Trust (HOST)*, Jun. 2010.

[3] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," *Proceedings of 44th ACM/IEEE Design Automation Conference (DAC) pp. 9–14*, Jun. 2007.

[4] M.-D. Yu and S. Devadas, "Secure and robust error correction for physical unclonable functions," *IEEE Journal of Design & Test Computers, Vol. 27, Issue 1*, Jan. 2010.

[5] A. Maiti and P. Schaumont, "Improving the quality of a physical unclonable function using configurable ring oscillators," *Proceedings of 19th IEEE International Conference on Field Programmable Logic and Applications (FPLA)*, Sep. 2009.

[6] M. v. D. B. Gassend, D. Clarke and S. Devadas, "Silicon physical random functions," *Proceedings of 9th ACM Computer and Communications Security Conference (CCS)*, Nov. 2002.

[7] W. B. D. Holcomb and K. Fu, "Initial sram state as a fingerprint and source of true random numbers for rfid tags," *Proceedings of the Conference on RFID Security 07*, Jul. 2007.

[8] G.-J. S. Jorge Guajardo, Sandeep S. Kumar and P. Tuyls, "Fpga intrinsic pufs and their use for ip protection," *Proceedings of 9th IACR International Workshop on Cryptographic Hardware and Embedded Systems (CHES), LNCS 4727, Springer*, Sep. 2007.

[9] J. S.-G. D. S. D. U. Ruhrmair, F. Sehnke and J. Schmidhuber, "Modeling attacks on physical unclonable functions," *Proceedings of 17th ACM Computer and Communication Security Conference (CCS)*, Oct. 2010.

[10] F. S. Dominik Merli and C. Eckert, "Improving the quality of ring oscillator pufs on fpgas," *Proceedings of the 5th Workshop on Embedded Systems Security*, Oct. 2010.

[11] C.-E. Yin and G. Qu, "Temperature-aware cooperative ring oscillator puf," *Proceedings of 2nd IEEE International Workshop on Hardware Oriented Security and Trust (HOST)*, Jul. 2009.

[12] C.-E. D. Yin and G. Qu, "Lisa: Maximizing ro puf's secret extraction," *Proceedings of 3rd IEEE International Workshop on Hardware Oriented Security and Trust (HOST)*, Jun. 2010.

[13] P. Sedcole and P. Y. K. Cheung, "Within-die delay variability in 90nm fpgas and beyond," *Proceedings of 16th IEEE International Conference on Field Programmable Technology (FPT) pp. 97–104*, Dec. 2006.

[14] M. v. D. B. Gassend, D. Clarke and S. Devadas, "Controlled physical random functions," *Proceedings of the 18th Annual Computer Security Applications Conference*, Dec. 2002.

[15] C. S. K. Q. Lerong Cheng, Puneet Gupta and L. He, "Physically justifiable die-level modeling of spatial variation in view of systematic across wafer variability," *Proceedings of 46th ACM/IEEE International Annual Design Automation Conference (DAC)*, Jul. 2009.

VI. Appendix

	$\boldsymbol{S}$		$\boldsymbol{T}$		**1-out-of-8**		**Chain-like Neighbor**		**Decoupled Neighbor**		
	P-VAL.	PROP.	P-VAL.	PROP.	P-VAL.	PROP.	P-VAL.	PROP.	P-VAL.	PROP.	STATISTICAL TEST
0^{th}-order	0.000000 *	45 *	0.000000 *	38 *	0.013689	122	0.000072 *	125	0.000003 *	115 *	Frequency
	0.000000 *	59 *	0.000000 *	49 *	0.166594	125	0.000000 *	125	0.050764	120	BlockFrequency
	0.000000 *	46 *	0.000000 *	39 *	0.231636	121	0.000000 *	125	0.000000 *	119 *	CumulativeSums (m-2)
	0.000000 *	46 *	0.000000 *	38 *	0.059743	122	0.000000 *	125	0.000000 *	118 *	CumulativeSums (m-3)
	0.000000 *	65 *	0.000000 *	31 *	0.002320	117 *	0.000000 *	0 *	0.302788	120	Runs
	0.000000 *	66 *	0.000000 *	44 *	0.000603	123	0.000000 *	62 *	0.000062 *	124	LongestRun
	0.000000 *	53 *	0.000000 *	23 *	0.000001 *	117 *	0.000000 *	0 *	0.000001 *	119 *	ApproximateEntropy
	0.000000 *	65 *	0.000000 *	25 *	0.004904	124	0.000000 *	1 *	0.070160	116 *	Serial (forward)
	0.000000 *	103 *	0.000000 *	74 *	0.552185	125	0.000000 *	117 *	0.192277	123	Serial (backward)
1^{st}-order	0.166594	124	0.003598	122	0.000949	120	0.000072 *	125	0.130323	124	Frequency
	0.000002 *	120	0.889414	121	0.529142	125	0.000000 *	125	0.056599	122	BlockFrequency
	0.000100	120	0.136969	122	0.063046	120	0.000000 *	125	0.082208	124	CumulativeSums (m-2)
	0.405918	122	0.020616	119 *	0.043046	120	0.000000 *	125	0.034444	123	CumulativeSums (m-3)
	0.082208	124	0.000000 *	68 *	0.192277	124	0.000000 *	0 *	0.096097	122	Runs
	0.048059	120	0.000000 *	90 *	0.000067 *	123	0.000000 *	62 *	0.000274	124	LongestRun
	0.025948	120	0.000000 *	75 *	0.002471	120	0.000000 *	0 *	0.130323	122	ApproximateEntropy
	0.112055	122	0.000000 *	80 *	0.262219	124	0.000000 *	1 *	0.956806	122	Serial (forward)
	0.474938	121	0.000000 *	117 *	0.551044	125	0.000000 *	117 *	0.620686	123	Serial (backward)
2^{nd}-order	0.369588	122	0.012159	121	0.000000 *	115 *	0.001228	125	0.086622	121	Frequency
	0.000782	122	0.422488	122	0.059743	124	0.000000 *	125	0.262219	123	BlockFrequency
	0.020616	120	0.086622	120	0.000000 *	118 *	0.000000 *	125	0.073984	123	CumulativeSums (m-2)
	0.575157	122	0.066516	122	0.000000 *	116 *	0.000000 *	125	0.389809	120	CumulativeSums (m-3)
	0.316158	125	0.915772	122	0.552185	123	0.000000 *	0 *	0.493319	124	Runs
	0.000062 *	122	0.000782	120	0.000000 *	123	0.000000 *	58 *	0.000115	124	LongestRun
	0.750075	124	0.474938	120	0.000000 *	120	0.000000 *	0 *	0.316158	122	ApproximateEntropy
	0.874833	124	0.077998	122	0.000123	123	0.000000 *	0 *	0.643139	121	Serial (forward)
	0.231636	123	0.302788	125	0.457002	124	0.000000 *	110 *	0.262219	123	Serial (backward)
3^{rd}-order	0.011457	125	0.136969	124	0.000000 *	111 *	0.000000 *	125	0.389809	123	Frequency
	0.262219	124	0.551044	125	0.003829	123	0.000000 *	125	0.457002	122	BlockFrequency
	0.002320	125	0.000131	125	0.000000 *	112 *	0.000000 *	125	0.551044	124	CumulativeSums (m-2)
	0.002984	125	0.017315	125	0.000000 *	111 *	0.000000 *	125	0.192277	123	CumulativeSums (m-3)
	0.643139	124	0.529142	121	0.889414	124	0.000000 *	0 *	0.529142	124	Runs
	0.000058 *	123	0.012903	124	0.000000 *	120	0.000000 *	48 *	0.000000 *	123	LongestRun
	0.020616	125	0.422488	123	0.000000 *	114 *	0.000000 *	0 *	0.889414	125	ApproximateEntropy
	0.369588	124	0.915772	123	0.000017 *	121	0.000000 *	0 *	0.493319	123	Serial (forward)
	0.439517	124	0.506075	122	0.575157	124	0.000000 *	114 *	0.439517	125	Serial (backward)
4^{th}-order	0.000001 *	125	0.000000 *	125	0.000407	121	0.000000 *	125	0.192277	124	Frequency
	0.166594	125	0.000051 *	125	0.344248	123	0.000000 *	125	0.807956	124	BlockFrequency
	0.000000 *	125	0.000000 *	125	0.143910	121	0.000000 *	125	0.070160	124	CumulativeSums (m-2)
	0.000011 *	125	0.000000 *	124	0.130323	120	0.000000 *	125	0.117876	124	CumulativeSums (m-3)
	0.316158	125	0.903069	122	0.437182	125	0.000000 *	0 *	0.414457	125	Runs
	0.004904	123	0.045489	125	0.007522	120	0.000000 *	54 *	0.000000 *	123	LongestRun
	0.289860	125	0.265309	122	0.708591	122	0.000000 *	0 *	0.143910	122	ApproximateEntropy
	0.571108	125	0.665311	123	0.571108	125	0.000000 *	1 *	0.457002	123	Serial (forward)
	0.405918	123	0.283039	124	0.551044	125	0.000000 *	110 *	0.825875	124	Serial (backward)
5^{th}-order	0.000029 *	125	0.211194	125	0.316158	124	0.000000 *	125	0.096097	125	Frequency
	0.004074	125	0.000000 *	125	0.493319	124	0.000000 *	125	0.552185	124	BlockFrequency
	0.000000 *	125	0.000000 *	125	0.665311	124	0.000000 *	125	0.043046	124	CumulativeSums (m-2)
	0.000000 *	125	0.000000 *	125	0.166594	123	0.000000 *	125	0.036430	125	CumulativeSums (m-3)
	0.493319	124	0.687147	124	0.036430	125	0.000000 *	0 *	0.192277	123	Runs
	0.006661	124	0.001801	125	0.036430	124	0.000000 *	42 *	0.000000 *	124	LongestRun
	0.059743	125	0.302788	124	0.729586	125	0.000000 *	0 *	0.665311	122	ApproximateEntropy
	0.687147	125	0.304210	121	0.096097	125	0.000000 *	0 *	0.512137	124	Serial (forward)
	0.262219	123	0.789315	123	0.457002	125	0.000000 *	114 *	0.474938	125	Serial (backward)
6^{th}-order	0.000009 *	125	0.001586	125	0.001080	125	0.000000 *	125	0.231636	122	Frequency
	0.000000 *	125	0.000000 *	125	0.086622	125	0.000000 *	125	0.437182	123	BlockFrequency
	0.000000 *	125	0.000000 *	125	0.231636	125	0.000000 *	125	0.091249	123	CumulativeSums (m-2)
	0.000000 *	125	0.000000 *	125	0.050764	124	0.000000 *	125	0.211194	123	CumulativeSums (m-3)
	0.101175	125	0.130323	123	0.422488	124	0.000000 *	0 *	0.529142	122	Runs
	0.000643	124	0.007992	124	0.211194	125	0.000000 *	44 *	0.000017 *	122	LongestRun
	0.000006 *	125	0.643139	124	0.130323	124	0.000000 *	0 *	0.598008	124	ApproximateEntropy
	0.552185	124	0.289860	123	0.329976	124	0.000000 *	0 *	0.277369	123	Serial (forward)
	0.874833	124	0.529142	124	0.405918	124	0.000000 *	113 *	0.843024	121	Serial (backward)

TABLE II

THE RESULTS OF NIST 'P-VAL. (P-VALUE OF P-VALUES)' AND 'PROP. (PROPORTION)' ANALYSES WITH RESPECT TO RANDOM SEQUENCES GENERATED BY $\boldsymbol{S}$, $\boldsymbol{T}$, 1-OUT-OF-8 CODING, CHAIN-LIKE NEIGHBOR CODING AND DECOUPLED NEIGHBOR CODING ACCOMPANIED BY 0^{th}- TO 6^{th}-ORDER DISTILLERS, WHERE '*' MARKS A FAILURE.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
0^{th}-order	93	7	3	4	2	6	1	2	4	3	0.000000 *	45/125 *	Frequency
	95	8	8	3	3	1	1	4	0	2	0.000000 *	59/125 *	BlockFrequency
	95	6	3	5	4	2	3	0	3	4	0.000000 *	46/125 *	CumulativeSums (m-2)
	96	7	3	3	7	1	4	1	0	3	0.000000 *	46/125 *	CumulativeSums (m-3)
	67	5	5	13	9	6	6	4	7	3	0.000000 *	65/125 *	Runs
	82	6	7	11	6	3	1	3	3	3	0.000000 *	66/125 *	LongestRun
	88	10	4	7	3	6	1	2	2	2	0.000000 *	53/125 *	ApproximateEntropy
	81	8	7	5	10	3	1	6	1	3	0.000000 *	65/125 *	Serial (forward)
	41	12	16	6	16	10	8	5	5	6	0.000000 *	103/125 *	Serial (backward)
1^{st}-order	13	19	12	12	13	7	11	7	11	20	0.166594	124/125	Frequency
	29	20	16	16	8	9	8	6	5	8	0.000002 *	120/125	BlockFrequency
	18	26	15	16	4	8	9	9	15	5	0.000100	120/125	CumulativeSums (m-2)
	18	14	17	14	16	10	10	8	9	9	0.405918	122/125	CumulativeSums (m-3)
	12	15	19	14	11	20	9	5	9	11	0.082208	124/125	Runs
	16	17	15	21	13	7	8	13	7	8	0.048059	120/125	LongestRun
	24	14	10	15	9	10	10	15	13	5	0.025948	120/125	ApproximateEntropy
	18	16	13	15	6	11	16	4	14	12	0.112055	122/125	Serial (forward)
	15	15	17	11	10	11	7	12	9	18	0.474938	121/125	Serial (backward)
2^{nd}-order	17	12	16	13	8	11	12	9	10	17	0.369588	122/125	Frequency
	27	14	12	19	7	11	7	10	9	9	0.000782	122/125	BlockFrequency
	13	22	12	12	5	13	14	4	13	17	0.020616	120/125	CumulativeSums (m-2)
	15	16	14	10	9	11	17	9	11	13	0.575157	122/125	CumulativeSums (m-3)
	15	6	10	18	11	16	16	14	8	11	0.316158	125/125	Runs
	9	13	13	30	14	14	9	7	5	11	0.000062 *	122/125	LongestRun
	18	9	12	9	13	13	13	14	13	11	0.750075	124/125	ApproximateEntropy
	13	14	11	15	10	14	16	9	11	12	0.874833	124/125	Serial (forward)
	12	9	22	13	14	11	12	10	15	7	0.231636	123/125	Serial (backward)
3^{rd}-order	4	9	11	15	8	16	22	8	15	17	0.011457	125/125	Frequency
	8	14	14	14	6	16	11	12	17	13	0.262219	124/125	BlockFrequency
	6	5	11	10	11	14	18	8	23	19	0.002320	125/125	CumulativeSums (m-2)
	4	10	10	8	6	18	20	12	18	19	0.002984	125/125	CumulativeSums (m-3)
	9	10	14	14	11	18	11	15	13	10	0.643139	124/125	Runs
	9	5	14	24	24	14	7	5	11	12	0.000058 *	123/125	LongestRun
	5	12	12	7	15	14	20	10	9	21	0.020616	125/125	ApproximateEntropy
	7	17	15	12	9	12	16	12	15	10	0.369588	124/125	Serial (forward)
	12	19	17	12	13	12	14	10	6	10	0.439517	124/125	Serial (backward)
4^{th}-order	2	2	8	26	13	16	21	10	17	10	0.000001 *	125/125	Frequency
	11	7	8	7	11	15	16	16	19	15	0.166594	125/125	BlockFrequency
	2	3	5	11	10	24	19	8	20	23	0.000000 *	125/125	CumulativeSums (m-2)
	2	4	5	15	13	16	16	17	11	26	0.000011 *	125/125	CumulativeSums (m-3)
	11	11	13	5	16	13	10	19	16	11	0.316158	125/125	Runs
	9	6	14	22	20	10	17	5	11	11	0.004904	123/125	LongestRun
	4	11	9	15	13	12	12	17	15	17	0.289860	125/125	ApproximateEntropy
	14	11	6	13	10	15	11	12	14	19	0.571108	125/125	Serial (forward)
	8	16	11	21	11	11	9	13	13	12	0.405918	123/125	Serial (backward)
5^{th}-order	0	5	12	20	13	10	11	20	11	23	0.000029 *	125/125	Frequency
	6	10	8	8	11	16	15	14	11	26	0.004074	125/125	BlockFrequency
	0	4	8	9	14	14	16	10	16	34	0.000000 *	125/125	CumulativeSums (m-2)
	0	8	4	13	10	16	10	12	24	28	0.000000 *	125/125	CumulativeSums (m-3)
	9	13	16	10	10	8	13	11	19	16	0.493319	124/125	Runs
	9	11	18	23	17	7	13	4	12	11	0.006661	124/125	LongestRun
	5	9	11	11	14	15	22	15	15	8	0.059743	125/125	ApproximateEntropy
	15	12	15	13	10	11	11	10	18	10	0.687147	125/125	Serial (forward)
	19	11	17	13	8	11	14	12	11	9	0.262219	123/125	Serial (backward)
6^{th}-order	4	8	2	19	10	8	14	15	22	23	0.000009 *	125/125	Frequency
	2	8	12	9	6	9	10	18	20	31	0.000000 *	125/125	BlockFrequency
	2	5	4	11	7	11	20	7	19	39	0.000000 *	125/125	CumulativeSums (m-2)
	3	9	5	5	5	14	13	9	18	44	0.000000 *	125/125	CumulativeSums (m-3)
	8	18	8	6	19	14	12	17	13	10	0.101175	125/125	Runs
	7	6	16	22	23	9	7	12	8	15	0.000643	124/125	LongestRun
	1	5	16	14	13	14	9	11	13	29	0.000006 *	125/125	ApproximateEntropy
	9	6	12	14	14	13	13	13	16	15	0.552185	124/125	Serial (forward)
	8	14	11	13	10	16	13	13	14	13	0.874833	124/125	Serial (backward)

TABLE III

NIST TEST RESULTS WITH RESPECT TO RANDOM SEQUENCE $\boldsymbol{S}$, WHERE $M = 32$ FOR BLOCK FREQUENCY TEST, $m = 2$ FOR APPROXIMATE ENTROPY TEST AND $m = 5$ FOR SERIAL TEST AND '*' MARKS A FAILURE.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
0^{th}-order	100	6	3	4	2	1	2	2	0	5	0.000000 *	38/125 *	Frequency
	88	9	3	5	4	4	5	1	6	0	0.000000 *	49/125 *	BlockFrequency
	100	5	4	5	1	3	2	2	0	3	0.000000 *	39/125 *	CumulativeSums (m-2)
	100	10	0	6	1	3	0	2	1	2	0.000000 *	38/125 *	CumulativeSums (m-3)
	108	7	4	1	2	0	1	1	1	0	0.000000 *	31/125 *	Runs
	100	4	7	7	3	2	0	1	1	0	0.000000 *	44/125 *	LongestRun
	114	3	2	1	1	1	1	2	0	0	0.000000 *	23/125 *	ApproximateEntropy
	112	5	4	1	1	0	1	1	0	0	0.000000 *	25/125 *	Serial (forward)
	73	11	8	7	10	3	4	2	1	6	0.000000 *	74/125 *	Serial (backward)
1^{st}-order	25	8	10	13	6	11	7	13	13	19	0.003598	122/125	Frequency
	14	8	13	16	12	14	10	13	12	13	0.889414	121/125	BlockFrequency
	22	11	13	7	13	13	15	12	6	13	0.136969	122/125	CumulativeSums (m-2)
	25	14	11	13	12	9	8	6	15	12	0.020616	119/125 *	CumulativeSums (m-3)
	86	14	4	6	4	4	1	2	2	2	0.000000 *	68/125 *	Runs
	66	15	13	16	5	3	3	3	0	1	0.000000 *	90/125 *	LongestRun
	78	15	7	9	2	5	2	2	2	3	0.000000 *	75/125 *	ApproximateEntropy
	80	6	9	6	4	6	6	0	5	3	0.000000 *	80/125 *	Serial (forward)
	40	15	13	14	6	8	6	6	7	10	0.000000 *	117/125 *	Serial (backward)
2^{nd}-order	20	15	14	22	6	9	8	12	7	12	0.012159	121/125	Frequency
	13	11	15	14	10	18	9	9	8	18	0.422488	122/125	BlockFrequency
	20	12	6	18	13	10	11	7	11	17	0.086622	120/125	CumulativeSums (m-2)
	18	15	9	15	9	17	13	2	13	14	0.066516	122/125	CumulativeSums (m-3)
	16	14	14	11	9	10	12	14	13	12	0.915772	122/125	Runs
	18	8	15	18	24	11	6	13	6	6	0.000782	120/125	LongestRun
	19	14	17	12	13	7	9	10	11	13	0.474938	120/125	ApproximateEntropy
	19	6	11	13	11	18	16	8	16	7	0.077998	122/125	Serial (forward)
	16	16	9	10	11	9	11	21	12	10	0.302788	125/125	Serial (backward)
3^{rd}-order	10	7	10	18	10	14	15	10	10	21	0.136969	124/125	Frequency
	10	10	12	16	9	8	13	12	17	18	0.551044	125/125	BlockFrequency
	8	9	8	15	9	10	27	5	14	20	0.000131	125/125	CumulativeSums (m-2)
	8	8	9	10	14	8	17	11	16	24	0.017315	125/125	CumulativeSums (m-3)
	14	13	13	19	11	7	12	12	11	13	0.529142	121/125	Runs
	9	11	11	24	20	10	11	11	12	6	0.012903	124/125	LongestRun
	15	10	11	6	14	11	9	18	15	16	0.422488	123/125	ApproximateEntropy
	11	9	11	15	12	12	15	11	14	15	0.915772	123/125	Serial (forward)
	11	18	15	9	13	12	9	10	12	16	0.506075	122/125	Serial (backward)
4^{th}-order	3	6	18	26	12	10	4	9	12	25	0.000000 *	125/125	Frequency
	6	10	5	5	12	15	8	19	22	23	0.000051 *	125/125	BlockFrequency
	2	5	8	18	15	9	18	8	11	31	0.000000 *	125/125	CumulativeSums (m-2)
	2	6	11	15	12	20	14	4	12	29	0.000000 *	124/125	CumulativeSums (m-3)
	14	13	12	13	8	11	11	13	14	16	0.903069	122/125	Runs
	12	6	23	15	16	8	13	8	13	11	0.045489	125/125	LongestRun
	18	7	11	7	13	19	12	11	12	15	0.265309	122/125	ApproximateEntropy
	13	16	8	13	13	15	13	15	8	11	0.665311	123/125	Serial (forward)
	14	13	13	12	12	13	18	15	10	5	0.283039	124/125	Serial (backward)
5^{th}-order	6	10	11	17	9	14	10	20	15	13	0.211194	125/125	Frequency
	6	5	4	6	7	13	18	13	22	31	0.000000 *	125/125	BlockFrequency
	2	8	8	19	5	9	16	10	19	29	0.000000 *	125/125	CumulativeSums (m-2)
	3	9	5	12	8	14	18	6	14	36	0.000000 *	125/125	CumulativeSums (m-3)
	13	17	9	17	11	10	13	11	11	13	0.687147	124/125	Runs
	11	9	17	25	18	9	13	5	7	11	0.001801	125/125	LongestRun
	9	15	15	7	14	14	15	18	12	6	0.302788	124/125	ApproximateEntropy
	13	11	11	10	10	11	17	11	20	11	0.304210	121/125	Serial (forward)
	12	13	13	16	10	16	11	12	14	8	0.789315	123/125	Serial (backward)
6^{th}-order	2	13	10	14	11	11	20	9	11	24	0.001586	125/125	Frequency
	5	2	4	8	8	8	16	22	19	33	0.000000 *	125/125	BlockFrequency
	1	4	11	10	10	12	12	15	19	31	0.000000 *	125/125	CumulativeSums (m-2)
	2	6	7	10	6	13	17	7	20	37	0.000000 *	125/125	CumulativeSums (m-3)
	23	14	12	14	7	10	13	8	13	11	0.130323	123/125	Runs
	9	12	13	24	19	6	15	8	9	10	0.007992	124/125	LongestRun
	13	17	11	18	12	10	11	12	11	10	0.643139	124/125	ApproximateEntropy
	11	14	11	18	11	13	11	4	17	15	0.289860	123/125	Serial (forward)
	15	9	14	13	11	13	15	14	6	15	0.529142	124/125	Serial (backward)

TABLE IV

NIST TEST RESULTS WITH RESPECT TO RANDOM SEQUENCE $\boldsymbol{T}$, WHERE $M = 32$ FOR BLOCK FREQUENCY TEST, $m = 2$ FOR APPROXIMATE ENTROPY TEST AND $m = 5$ FOR SERIAL TEST AND '*' MARKS A FAILURE.

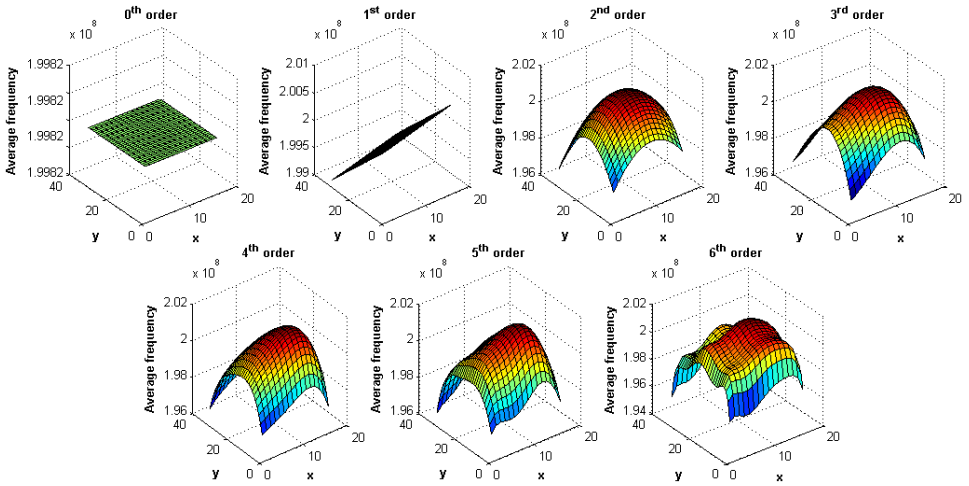

Fig. 5. The modeled systematic variation after applying 0^{th} through 6^{th}-order polynomial regression to the dataset of Chip No. 1.

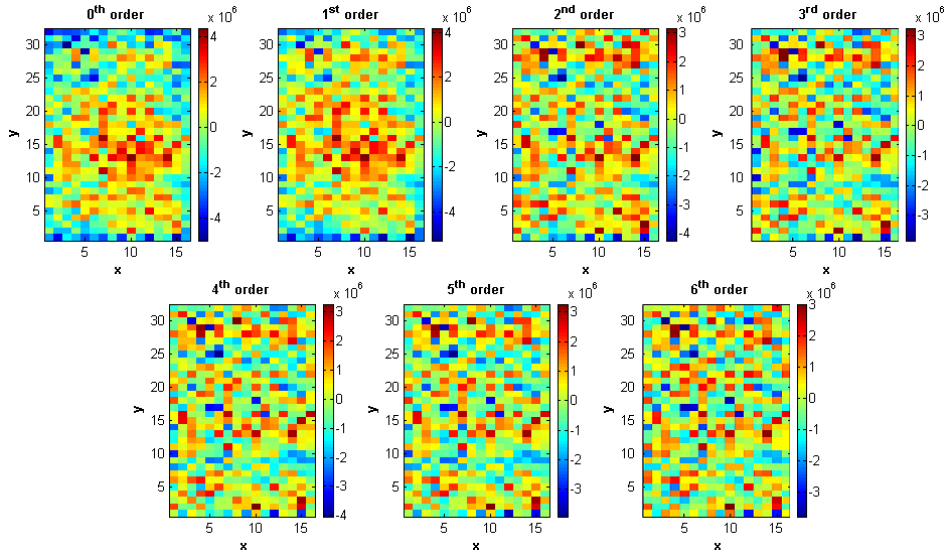

Fig. 6. The distilled random variation after applying 0^{th} through 6^{th}-order polynomial regression to the dataset of Chip No. 1. Notably, we see the 'bull's eye', i.e., the radial pattern close to the center, vanishing in the cases of 2^{nd} order model and beyond.

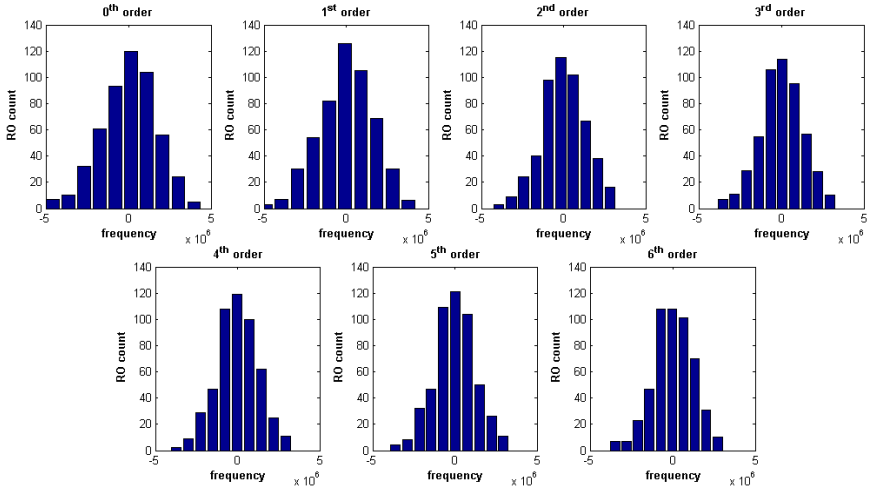

Fig. 7. The histogram of the distilled random variation after applying 0^{th} through 6^{th}-order polynomial regression to the dataset of Chip No. 1. It is difficult to judge simply from the chart which model is the best fit without running NIST tests for all appears normal but with a diminishing variance as the order increases