FPGA PUF based on Programmable LUT Delays

Bilal Habib , Kris Gaj, Jens-Peter Kaps

Electrical and Computer Engineering Department

George Mason University

Fairfax, VA, USA

*Email: {bhabib,kgaj,jkaps}@gmu.*edu

***Abstract*— Strong and efficient techniques are required for chip authentication and secret key generation by integrated circuits (IC). This paper presents a novel approach toward an FPGA friendly Ring Oscillator (RO) based Physical Unclonable Function (PUF). In this design the internal variations of FPGA Look-Up Tables are exploited to generate a PUF response. Statistical tests were performed to study the strength of this PUF. Moreover, stability is compared with the state of the art reported in literature to date. Our design has been tested on 31 Spartan-3e devices and the results are promising with inter-device Hamming distance of 48.3%, Uniformity 50.13%, Bit-aliasing 51.8%, Reliability 97.88%, and Steadiness 99.5%. Furthermore, we also analyzed the frequencies to extract the random variation offered by our design.**

Keywords-Physical Unclonable Function; Programmable LUT Delays; Xilinx FPGAs

I. INTRODUCTION

With an increasing number of communication and computing devices, security challenges are becoming significant. An on-chip PUF (Physical Unclonable Function) can solve these challenges effectively and efficiently. A PUF is a chip-dependant unclonable challenge-response function that can be used to uniquely identify a specific integrated circuit. Furthermore, the PUF itself is tamper resistant against physically invasive attacks. Due to these attributes, a PUF offers security against intellectual property (IP) theft and counterfeiting, and solves issues such as chip authentication, reverse engineering, trusted computing, and secure key generation. The idea of PUFs was first presented in [1]. Since then, the scientific community has profoundly investigated it. Silicon based PUFs use the idea of extracting the maximum variability of the chip manufacturing process. This variation is inherent and results in a unique signature for each chip similar to a biometric thumb impression of humans. Even for the same manufacturing process each chip carries a different signature due to process variations. A strong PUF is classified to be the one that extracts the maximum process variation and is reliable to exhibit this variation under different conditions.

There are strong reasons to design PUFs for FPGAs. Cryptographic functions are implemented using FPGAs for faster execution compared to software execution, and PUFs can provide random keys for these functions. Furthermore, FPGAs are reconfigurable in nature and IP protection during configuration of an FPGA is an important issue from a security point of view.

In Section II, we describe the motivation for FPGA based PUFs, in III we describe the previous work for a better understanding of our study on PUF. Section IV explains the design methodology. We present and discuss implementation details, as well as highlight results in Section V. In VI we highlight the bit-string generation and in VII we do the frequency analysis. The conclusion and future study are described in Sections VIII and IX respectively.

II. MOTIVATION FOR FPGA BASED PUF

FPGAs as opposed to ASICs offer a flexible and secure solution to IP implementations in hardware. The reason for this flexibility is because FPGAs can be configured at any time in the field without any cost associated with it. Similarly FPGAs can offer secure IP implementation. In a practical application, the IP reads the PUF output and compares it with some built-in constant (chip-ID) and if both of them match then it enables the IP to run on this particular FPGA device. This way the IP vendor makes sure that the IP is licensed only for a selected device. The chip ID can be retrieved by the manufacturer during enrollment.

Furthermore PUFs can be employed to verify if the system having an FPGA as one component which came from a genuine source. This can be done by extracting the chip-dependent PUF output in the field and comparing it with the one supplied by a genuine source.

Another motivation of FPGA based PUFs is that FPGAs offer quick product customization as per market demands compared to ASIC, therefore it is important to investigate the security features of FPGA based PUF designs.

III. PREVIOUS WORK

Since 2002, silicon based PUFs have been extensively investigated. The initial proposal of a delay based arbiter PUF was made in [3]. Arbiter PUF was further explored by [4] and [5] to investigate reliability and security features. Although the Arbiter-PUF offers strong PUF properties, it is prone to machine-learning attacks.

In [8], the idea of a Ring-Oscillator (RO) based PUF is presented. In this PUF the challenge is the selection of a pair of ROs. The response is the one bit comparison result of the frequencies of those ROs. A large-scale characterization of RO based PUF has been done in [13]. In [7] the first SRAM-PUF is presented, in which the start-up values of uninitialized Embedded RAMs are used as a PUF response. However, in the current state of the art Xilinx and Altera FPGAs, the start-up values of memory

locations are controlled by the chip manufacturer, which renders SRAM PUF useless for FPGAs. In [9], Butterfly-PUF is presented, which requires symmetric paths between registers for causing metastability. FPGA tools do not offer complete access to symmetric design at the wire level, therefore, routing schemes make it hard to achieve symmetric butterfly design on FPGAs. This fact has been verified by [15] for both Arbiter and Butterfly PUF. In [12] and [20], the concept of programmable delay lines is presented, in which the LUT delays are used to create a metastable condition which is further used to develop a PUF and TRNG respectively. Our approach is to employ the concept of programmable delay lines in ring oscillators based PUF for improving the number of independent response bits. We did not create metastability for randomness. In [11], Maiti et al. presented an RO PUF, in which multiplexers were introduced in the ring to select different paths inside the ring. In this design, only two rings with identical paths were compared at a time because the authors wanted to determine a configuration which resulted in the maximum frequency difference between two ROs. However, in our case, the path outside of the LUTs stays constant. Therefore, we minimize the impact of routing or wire delays on the frequency of ring oscillator. Hence, the randomness is purely due to internal LUT variation. This delay is the significant and deciding factor in the comparison of two ring oscillators. In [17], the number of configurations of ring oscillator has been improved by introducing a latch in the path of a ring, making it impossible to compare a latch-path with no-latch-path. Our approach is completely different because we do not introduce anything in the path of a ring. We extract only the randomness inherent in LUTs, where we have the luxury of comparing any configuration of a LUT with any other configuration i.e., any configuration from '000' to '111' with any other configuration from '000' to '111'. All configurations are applied to LUT input lines. We will show in the next section that the programmable delay offers to generate random and independent bits with a very strong capability to repeat them.

IV. Design Methodology

Our PUF is designed for Spartan xc3s100e devices, where each CLB has four slices, each with two 4-input LUTs. Our PUF design is based on a RO, which has one AND gate and three inverters. This design uses one LUT for an AND gate and three LUTs for inverters. We used one LUT-input to connect in a ring, while the remaining LUT-inputs are varied in order to generate programmable delays, as shown in Fig. 1.

In Fig. 1, three inverter LUTs in a ring are connected to three delay lines each, while the remaining LUT is tied to two lines. Because we wanted to analyze the maximum effect of delay lines on the ring oscillator frequency, therefore we connected all free LUT inputs to our programmable delay controller. All the LUT inputs are locked by using a *LOCK_PINS* attribute provided by Xilinx tools. By doing so we ensured that all interconnects leading to LUT inputs are fixed and cannot be arbitrarily changed by routing tools. Additionally we ensure that all the rings are identical and are subject only to CLB internal routing delays. In order to achieve this, we defined each ring oscillator as a macro and placed them at selected locations. The output of this ring oscillator is fed into a 32-bit counter, which determines the frequency of this oscillator. It is important to mention that the response from our design is solely dependent on the internal variation of FPGA LUTs, and variations due to routing delays are minimized. In particular, our method eliminates any differences in routing paths (and thus routing delays) caused by the tools.

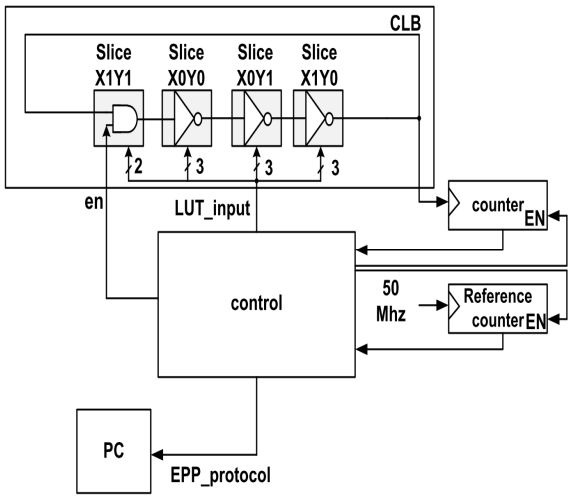

Figure 1. Single Ring-Oscillator with programmable delay lines

Figure 2 shows the placement of 130 ROs that make up our PUF. Each ring oscillator is constrained in a single CLB. In this design, 130 rings are configured at the center of a chip in a 13x10 matrix, The chip under test does not allow us to place 130 rings in a square format placed at the center of a chip. In Figure 2 rings start from the bottom left (R0) corner and the last one is shown at the top-right corner numbered R129.

We selected 130 as a number of rings, because the FPGA devices available to us had 240 CLBs in total. We cannot use all CLBs for rings, because we need some logic resources to use for control purposes as shown in the Table 1.

We could have decreased the slice count by forcing two rings per CLB (as each CLB slice contains two neighboring LUTs), but we intentionally rejected that approach, because the two rings might lock with each other, and hence their frequency affect each other. This phenomenon is also reported in [16].

For data retrieval we used Enhanced Parallel Port (EPP protocol), which has a very small area imprint. On the PC side, Digilent Port Communications (DPC) utilities were used, which are provided with Digilent Adept software.

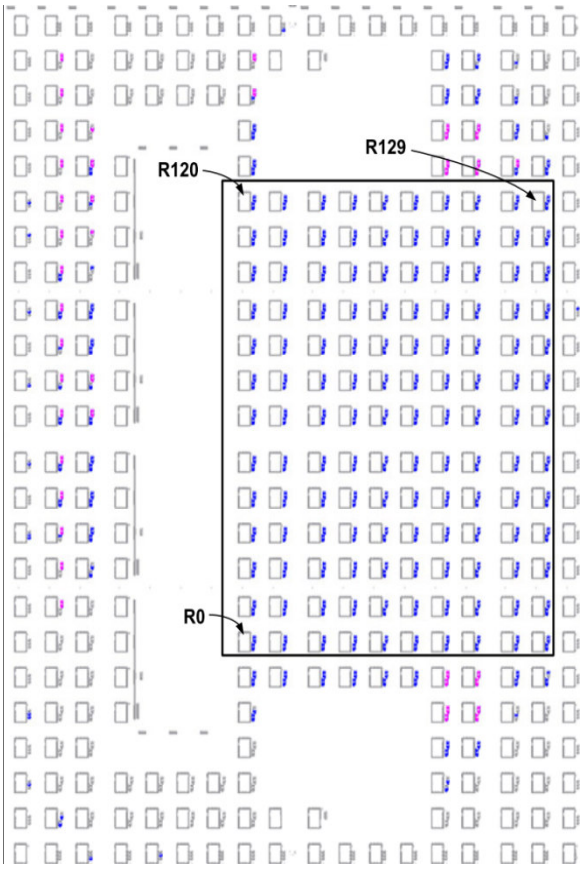

Figure 2. PUF array configuration of 130 RO on Spartan xc3s100e device

Table 1. Area requirements of our design

	Number of slices occupied	Percentage
Slices for rings	4*130 =520	54.2%
Slices for other logic	227	23.6%
Total slices	747	77.8%

V. IMPLEMENTATION DETAILS

LUT_input bits can be varied from '000' to '111', resulting in eight different frequencies of a ring oscillator. Additionally, these frequencies are highly repeatable for any particular ring as shown in Fig. 3. From Fig. 3, it is evident that the frequency varies significantly depending on the LUT_input sequence. In [12] and [20], it is stated that maximum frequency is achieved with '000' and minimum with '111' sequence. However, based on our experiments, this is not always the case.

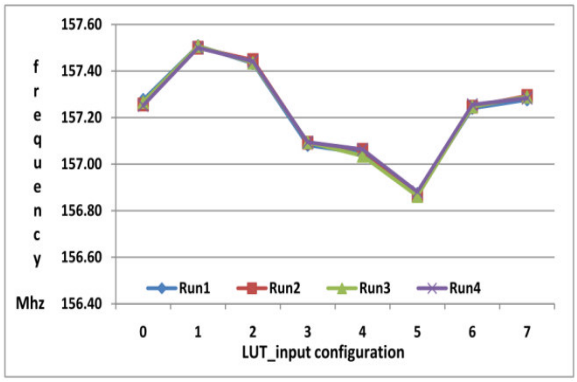

Figure 3. Frequency distribution due to LUT_input bits variation

The pattern presented in Fig. 3 changes completely when we select another ring oscillator. Even a neighboring CLB exhibits a different pattern. One reason might be that we use Spartan-3e devices which are based on 90nm technology while in [12,20], Virtex-5 devices are used which are 65nm technology. We also observed that the standard deviation among 20 samples never exceeds 0.018 % of ring oscillator frequency. By using longer characterization time the noise further decreases. It is important to mention here that we did not test repeatability under different voltage and temperature conditions. We believe that the behavior should not be different from [11], for different voltage and temperature conditions, because Maiti et al. also used Spartan-3e devices in their experiment, which are based on 90nm technology.

We tested our PUF for different characterization times; which is the time required to allow the ring to oscillate freely. Each RO should be enabled for enough time, such that the delay pattern associated with each LUT input value is sufficiently repeatable as shown in Fig. 4. Preliminary investigation revealed that a small characterization time causes huge differences in the pattern. In Fig. 5, the characterization time is reduced from 1sec to 1msec and standard deviation increases among four runs from 0.018% to 0.15%.

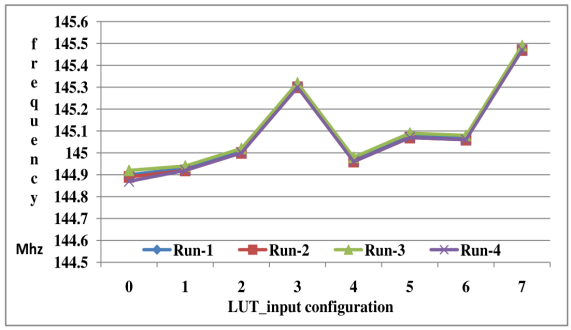

Figure 4. Characterization time equal to 1 sec

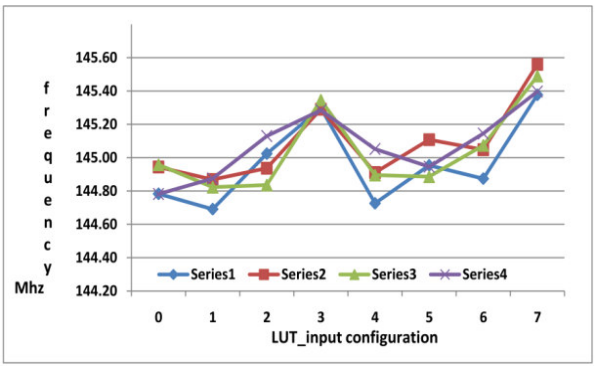

Figure 5. Characterization time equal to 1 msec

We used a characterization time of 1 sec when we extracted data from PUF, implemented on 31 Digilent boards. We did the frequency analysis of each ring and devised two schemes to analyze the improvement in the number of response bits.

Scheme # P1

In this scheme, we compared eight frequencies resulting from eight different values (000 to 111) applied to LUT input of one ring oscillator with eight frequencies of the neighboring oscillator i.e., only ROs under the same configurations are compared. In post-processing, if fr0>fr1, the response is '1' otherwise it is '0'. To remove systematic variation (an unwanted correlated variation due to spatial location of a ring-oscillator on a chip), under each configuration, only the comparisons shown in Fig. 6 are made. Therefore, the PUF will output $8*(r-1)$ bits response for each FPGA, and chip ID is extracted from these 1032 bits.

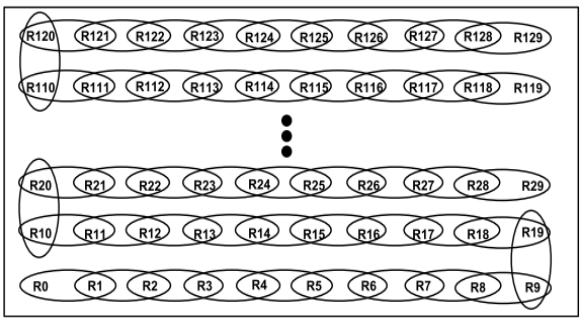

Figure 6. Comparison of rings along the rows

Scheme # P2

In this scheme we compared the frequencies along the CLB columns as shown in Fig. 7, where each circle shows a single comparison, otherwise it is similar to scheme P1. From this scheme, we were able to extract the same 1032 bit response. However, the inter-chip variation decreases to 95.34% from 96.6% as shown in Table 3. One reason is that the first row of the chips (R0-R9) were having the smallest frequencies and resulted in a similar response when compared to the frequencies of the 2nd row (R10-R19). This behavior reduced the HD and subsequently the Inter-chip variation.

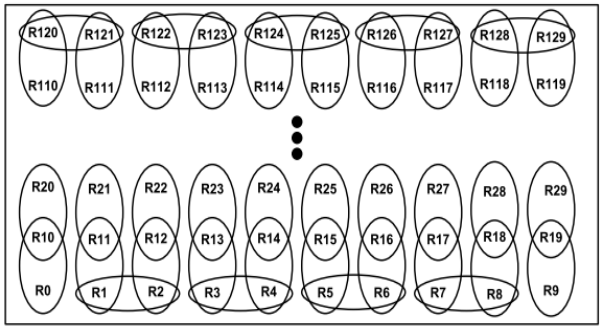

Figure 7. The comparison of rings along the columns

VI. BIT-STRING GENERATION

For 130 ROs, each ring pair is able to generate 8 bits due to 8 LUT configurations, which yields bitstring of length 129*8 = 1032 bits. In this work we only compare the neighboring rings to cancel out the effect of systematic variation. During comparison, if the frequencies of two rings cross, then we record 8 bits, otherwise a single bit is contributed toward the PUF ID. Therefore, during enrollment, we store for each ring number a corresponding crossover bit (i.e., we store pairs: (Ring #, c)). In Fig. 8, a meaning of the crossover is demonstrated.

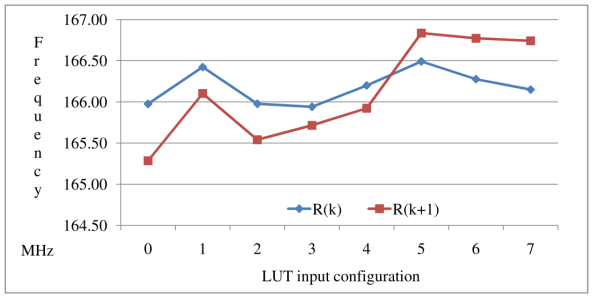

Figure 8. Crossover of two rings

A thresholding technique as explained below is employed to discard those comparisons which are vulnerable to producing "bit-flips".

Thresholding Technique:

If an ID bit is flipped during regeneration then it reduces the reliability. In order to avoid this condition, a thresholding technique is employed which accepts a bit if the difference in frequency for any comparison is greater than 150kHz ($\Delta f \geq 150$ kHz). Otherwise we discard that bit, because it can be flipped by noise during regeneration. But if the frequencies of two rings fail the thresholding condition ($\Delta f \geq 150$ kHz), then only one bit is contributed towards the chip ID, and this bit is generated by the majority vote of 8 comparisons. Furthermore, Δf is fixed at 150 kHz, because we observed that the average standard deviation among 20 samples of each frequency is around 30 kHz for all the rings.

Therefore, we keep it at least 5 times the standard deviation. After applying this condition, the number of strong bits per chip is 283 as shown in Table 2. We call this sequence of bits the Chip-ID. It is important to mention here that 283 bits is the minimum length of bitstring generated by a particular FPGA, all other devices generated more than 283 bits.

The more different a pattern is from another ring oscillator in absolute frequency terms, the stronger 8-bits we will get from their comparison.

Table 2. Details of dataset

	Maiti et al. [22]	**This work**
No. of Chips (N)	193	31
Samples (T)	100	20
No. of ID's (K)	1	1
ID size (L) bits	511	283
Ring oscillators (M)	512	130
FPGA family (Device used)	Spartan-3E (XC3S500E)	Spartan-3E (XC3S100E)

In Tables 2, 3 and 4 we compared our results with the results shown by Maiti et al. in [22]. Five PUF properties are listed in Table 3, they are briefly defined as:

Uniformity: Uniformity of a PUF estimates how uniform the proportion of '0's and '1's are in the PUF response.

$$(\text{Uniformity})_i = \frac{1}{n}\sum_{l=1}^{n} r_{i,l} \times 100\% \quad (1)$$

Uniqueness: It represents the ability of a PUF to uniquely distinguish a particular chip among a group of chips of the same type.

$$\text{Uniqueness} = \frac{2}{k(k-1)}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\frac{HD(R_i,R_j)}{n} \times 100\% \quad (2)$$

Bit-aliasing: If bit-aliasing happens, different chips may produce nearly identical PUF responses, which is an undesirable effect.

$$(\text{Bit}-\text{aliasing})_l = \frac{1}{k}\sum_{i=1}^{k} r_{i,l} \times 100\% \quad (3)$$

Reliability: PUF reliability means how efficient a PUF is in reproducing the response bits.

$$HD_{INTRA} = \frac{1}{m}\sum_{t=1}^{m}\frac{HD(R_i, R'_{i,t})}{n} \times 100\% \quad (4)$$

$$\text{Reliability} = 100\% - HD_{INTRA}$$

Steadiness: Steadiness indicates how stably a PUF outputs the same responses to the same challenge sets.

$$s_n = 1 + \frac{1}{K.L}\sum_{k=1}^{K}\sum_{l=1}^{L} log_2 \max(p_{n,k,l,1} - p_{n,k,l,}) \quad (5)$$

where $p_{n,k,l,1} = \frac{1}{T}\sum_{t=1}^{T} r_{n,k,t,l}$

The following notations are used in equations (1) to (5).

N = total number of chips
n = index of a chip ($1 \le n \le N$)
K = total number of identifiers(IDs) generated per chip
k = index of an ID in a chip ($1 \le k \le K$)
T = total number of samples measured per ID
t = index of a sample ($1 \le t \le T$)
L = total number of response bit in an ID
l = index of a response bit ($1 \le l \le L$)
M = total number of ring oscillators
m = index of an oscillator ($1 \le m \le M$)
r = is the PUF response bit
Ri = Response of chip i
Rj = Response of chip j

Table 3. Comparison with Maiti et al. [22]

	P1	P2	Maiti	Ideal
Uniformity	50.13	50.75	50.56	50%
Uniqueness	96.6	95.34	93.98	100%
Bit-aliasing	51.8	50.75	50.56	50%
Reliability	97.88	98.1	99.13	100%
Steadiness	99.5	99.5	98.9	100%

We measured our PUF responses at room temperature and then generated results shown in Table 3, by running a script available at [23] on our PUF data.

From Table 3 and 4, it is evident that our result set is comparable to Maiti et al. However, we believe that with four times smaller PUF size (in-terms of the number of CLB slices for Ring Oscillators) we were able to generate more than twice as many bits per ring oscillator. Furthermore, our PUF-IDs are more biased towards '1',

resulting in Uniformity greater than ideal by 0.13%. All the five PUF properties are thoroughly explained in [22].

Table 4. Properties of independent strong bits

	This work (P1)	This work (P2)	Maiti et al.[11]
Number of ring oscillators [A]	130	130	512
Average Independent , strong response bits* [B]	283	318	511
Bits per Ring [B/A]	2.17	2.44	~1

*strong bit = When the $\Delta f \geq 150$kHz, [Average of 31 Chips Frequency ≈ 165MHz]

VII. FREQUENCY ANALYSIS

To show the extent of randomness offered by the chip under test and extracted by our design, we analyzed the frequencies of all rings under all configurations.

The frequencies of all 31 chips are shown in Fig. 9. Each frequency is an average of all 1040 frequencies generated by ROs of a given chip. Each ring generates 8 different frequencies and there are 130 rings in total.

The standard deviation in the frequency of 1040 points per chip is shown in Fig. 10.

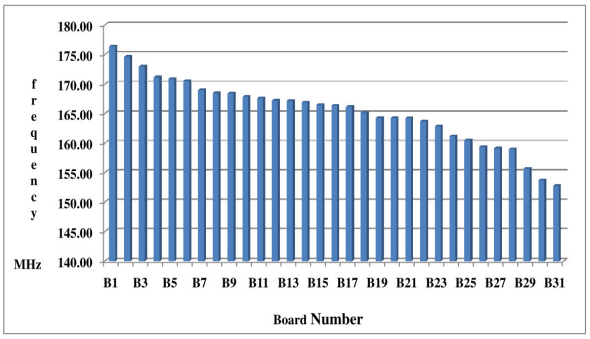

Figure 9. Average frequency of all ring oscillators located on each board

The average frequency of each ring oscillator averaged over all boards is shown in Fig. 11. Each point in this figure is the average of (8 frequencies * 31 boards) = 248 frequencies. From Fig. 11, it is evident that the highest peaks in the frequency occur at the central part of the chip. While the lowest frequencies occur at the edges. Our PUF layout is not placed at the center of the chip as explained in section IV, therefore in Fig. 11 the highest peaks seem off the center.

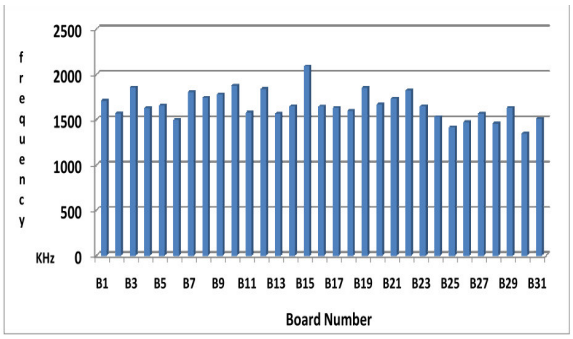

Figure 10. Average standard deviation in frequency of 1040 points per board

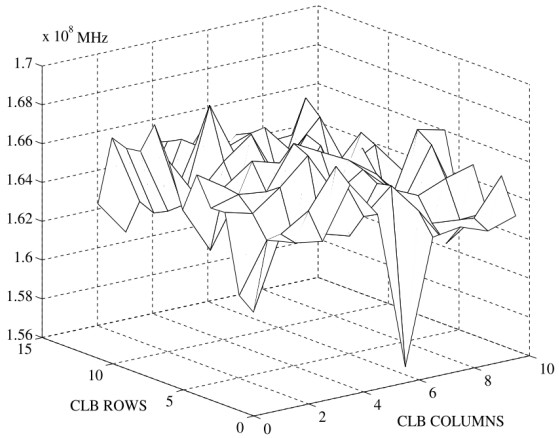

Figure 11. The average frequency of all boards for 130 Ring oscillators depending on CLB locations

Systematic and Manufacturing Variation

The frequency of any ring oscillator consists of both systematic variation and manufacturing process variation. For PUF statistical and qualitative analysis we need to decompose these variations into within-die and die-to-die variations.

Decomposition Methodology

With the introduction of programmable delay lines, we were able to generate eight different frequencies from a single ring oscillator. Therefore we had 1040 total frequencies for 130 ring oscillators. We analyzed 130 rings that make the rectangle shown in Fig. 2 at the central part of the chip. This rectangle is a matrix of 13x10 ring oscillators. We laid these 1040 frequencies in the form of a matrix with 13 rows and 80 columns. Any point in this matrix is denoted by F(x,y), where x ranges from 0 to 79 and y ranges from 0 to 12, i.e., each row contains 80 frequencies from 10 ring oscillators. Other dimensions could also be employed for this experiment. F(0,0) is the frequency of ring oscillator 0, with LUT_input configuration equal to '000'. We call F(0,0) as the nominal frequency in our calculations.

We decomposed the frequency of rings into random and systematic variation components. Our decomposition method follows the method explained in [18], as shown in equation (6).

$$F_{(x,y)} = F_{(0,0)} + R_{WID} + S_{WID(x,y)} + S_{D2D(x,y)} \quad (6)$$

Here, R_{WID} is the random within die variation component, S_{WID} is the systematic within die variation component while S_{D2D} is the systematic die to die variation.

We used Down Sampled Moving Average (DSMA) to extract the random and systematic variation from equation 6. In DSMA we moved the window over 1040 points and we got the average frequency of all the points in a window.

$$DSMA_{(x,y)} = (2z+1)^{-2} \sum_{i=x-z,\, j=y-z}^{x+z,y+z} F_{(i,j)} \quad (7)$$

The window size is 9 with a 3x3 dimension, by setting z=1 in equation (7). We keep z = 1, because with a big window size we will be averaging too many points, which will suppress the randomness due to programmable delay lines.

$$DSMA_{(x,y)} = F_{(x,y)} - R_{WID} \quad (8)$$

From equation 7 and 8, we got

$$R_{WID} = F_{(x,y)} - DSMA_{(x,y)} \quad (9)$$

For 1040 points in total and z=1, we get 858 random values. We normalized it over $F_{(0,0)}$ to get the R_{WID} variation. It has been shown in Fig 12.

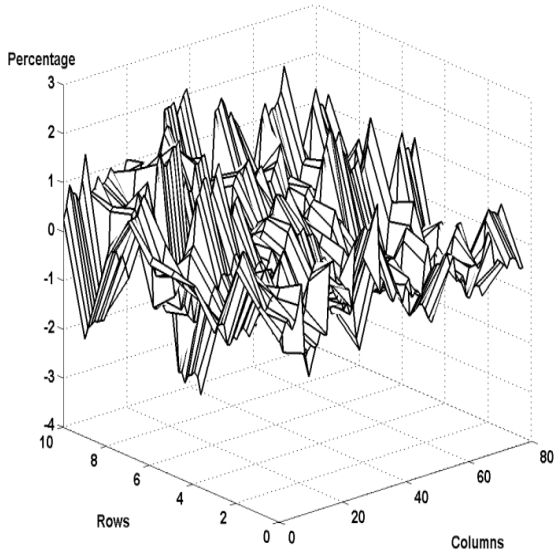

Figure 12. Random within die variation (normalized over F(0,0), shown as a percentage)

Random within die variation has the following properties.

Table 5. Properties of Random with-in die variation normalized over F(0,0)

Mean	0.0
Min	-3.23 %
Max	2.27 %
Peak to Peak	5.5%
Standard Deviation	0.84%

The distribution of this randomness is shown in Fig. 13. The distribution of Random within Die variation is plotted using a histogram. It is evident that the plot is centered at 0.0.

We plan to publish our data for 31 boards at : www.cryptography.gmu.edu for independent verification.

VIII. Conclusion

In this work we presented a novel PUF design, based on ring oscillators, where the programmable delays of FPGA LUTs were used to generate additional bits of an ID. The strength of this design is its ability to generate more than one random frequency per ring oscillator without changing the path of the ring outside LUTs. This solution offers the option to reduce the area requirements of ring oscillator PUFs. To demonstrate the strength of this PUF, it was shown that our design generated more bits per ring oscillator, and these bits are as strong as the ones reported in literature for Configurable Ring Oscillators. The statistical and PUF properties were analysed and were shown to be very strong from a security point of view.

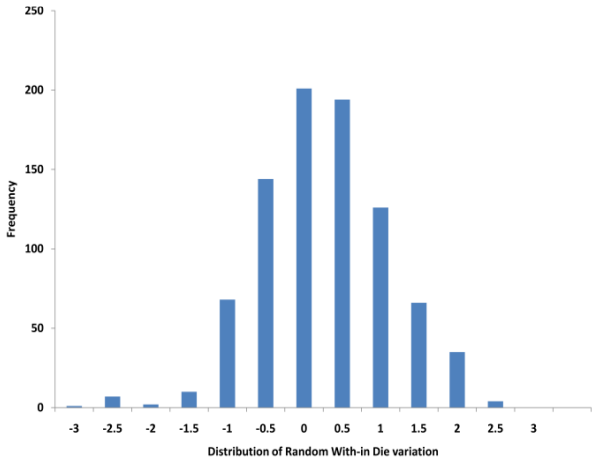

Fig. 13. Distribution of the random within die variation

IX. Future Work

The analysis of the behaviour of LUT programmable delays and patterns under different voltage and temperature conditions will be carried out in the future. Furthermore, mitigation of systematic variations using programmable delay lines will be studied. In addition, the effect of FPGA ageing and the possibility of its compensation will be investigated.

References

[1] R. Pappu: Physical One-Way Functions, PhD Thesis, Massachusetts Institute of Technology, 2001.

[2] T. Kean: Cryptographic rights management of FPGA intellectual property cores. In: ACM/SIGDA tenth international symposium on Field-programmable gate arrays — FPGA 2002, pp. 113–118 (2002).

[3] B. Gassend, D. Clarke, M. van Dijk, S. Devadas, Controlled Physical Random Functions. ACSAC '02: Proceedings of the 18th Annual Computer Security Applications Conference (IEEE Computer Society, Washington, DC, 2002), p. 149

[4] J.W. Lee, D. Lim, B. Gassend, G.E. Suh, M. van Dijk, S. Devadas, A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Application. Proc. Symposium on VLSI Circuits, 2004, pp. 176–159.

[5] D. Lim, Extracting Secret Keys from Integrated Circuits. Master's thesis, MIT, MA, USA, 2004.

[6] P. Sedcole, P.Y.K. Cheung. Within-die Delay Variability in 90-nm FPGAs and Beyond, In Proc. FPT 2006.

[7] J. Guajardo, S. S. Kumar, G. Schrijen, and P.Tuyls. FPGA intrinsic PUFs and their use for IP protection, Workshop on Cryptographic Hardware and Embedded Systems - CHES 2007, Lecture Notes in Computer Science (LNCS), volume 4727, Springer, pp 63–80, 2007.

[8] G.E. Suh, S. Devadas. Physical unclonable functions for device authentication and secret key generation. Proc. 44th Annual Design Automation Conference, New York, 2007.

[9] S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls. Extended abstract: The butterfly PUF protecting IP on every FPGA, Hardware-Oriented Security and Trust – HOST 2008, pp 67–70, 2008.

[10] M. Gora, A. Maiti, P. Schaumont: A Flexible Design Flow for Software IP Binding in Commodity FPGA. IEEE Symposium on Industrial Embedded Systems (SIES 2009), July 2009.

[11] A. Maiti and P. Schaumont: Improved Ring oscillator PUF: An FPGA Friendly Secure Primitive, Journal of Cryptology, volume 24, number 2, pp 375-397, Oct. 2010.

[12] M Majzoobi, F Koushanfar, S Devadas: FPGA PUF using Programmable Delay Lines. Proc. WIFS, 2010.

[13] A. Maiti, J. Casarona, L. McHale, P. Schaumont: A Large Scale Characterization of RO-PUF. Hardware-Oriented Security and Trust (HOST), 2010.

[14] Y. Hori, T. Katashita, A. Satoh, T. Yoshida: Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs Proc. The 2010 International Conference on Reconfigurable Computing and FPGAs, ReConFig 2010.

[15] S. Morozov, A. Maiti, and P. Schaumont: An analysis of delay based PUF implementations on FPGA. Reconfigurable Computing: Architectures, Tools and Applications – ARC 2010. LNCS, vol. 5992, Springer, pp 382–387, 2010.

[16] C. Costea, F. Bernard, V. Fischer, and R. Fouquet: Analysis and Enhancement of Ring Oscillators Based Physical Unclonable Functions in FPGAs. Proc. 2010 International Conference on Reconfigurable Computing and FPGAs, pp 262–267, IEEE, 2010.

[17] X. Xin, J. Kaps, and K. Gaj: A Configurable Ring-Oscillator-Based PUF for Xilinx FPGAs. Proc. 14th EUROMICRO Conference on Digital System Design – DSD'11, pp 651–657, IEEE, Sep. 2011

[18] T. Tuan, A. Lesea , C. Kingsley , and S. Trimberger . Analysis of Within-Die Process Variation in 65nm FPGAs, 12th International Symposium on Quality Electronic Design (ISQED), March, 2011.

[19] A. Maiti, L. McDougall and P. Schaumont: The Impact of Aging on an FPGA-Based Physical Unclonable Function. 21st International Conference on Field Programmable Logic and Applications (FPL 2011), September 2011.

[20] M. Majzoobi, F. Koushanfar, and S. Devadas: FPGA-Based True Random Number Generation Using Circuit Metastability with Adaptive Feedback Control. Proc. CHES 2011.

[21] http://rijndael.ece.vt.edu/puf/detailed_CRO.php

[22] A. Maiti, V. Gunreddy and P. Schaumont: A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions. *IACR ePrint 2011/657*

[23] http://rijndael.ece.vt.edu/puf/script_download.html