Original scientific paper

Informacije MIDEM
Journal of Microelectronics,
Electronic Components and Materials
Vol. 48, No. 2(2018), 121 – 128

Real-Time Random Number Generation with Ring Oscillator Based Double Physically Unclonable Function

Seda Arslan Tuncer

Firat University, Department of Software Engineering, Elazig, Turkey

Abstract: Integrated Circuit (IC) is a semiconductor wafer that is fabricated for millions of components. Although it is accepted that the same chemical properties are observed in the same type of ICs produced within the processes of wafer production, masking, etching, doping, atomic diffusion, ion implantation, metallization and packaging steps, these properties differ at microscopic levels. For example, the propagation delay is different in two logic elements in the same integrated circuit, and the silicon distribution in each LE is not equal. These differences have created the concept of Physically Unclonable Function (PUF). PUF is a function that creates values specific to the physical environment in which it operates. This article presents the implementation of a ring oscillator- (RO) based PUF design in two different ICs to generate random numbers. Two different FPGA (Field Programmable Gate Array) cores were used in the proposed structure, and PUF structures were implemented on these hardware elements. The raw random numbers obtained by the generated system were post-processed to be used in applications (e.g. encryption, game programming). The study used Von Neumann and hash functions for post-processing. NIST and Autocorrelation tests were also administered to check the validity of the obtained random numbers.

Keywords: Random number generation; Field programmable gate arrays; physically unclonable function**.**

Generator naključnih števil v realnem času z dvojno funkcijo nekloniranja na osnovi obročnega oscilatorja

Izvleček: Integrirana vezja so narejena na polprevodniški rezini. Kljub enakim kemijskim postopkom izdelave se lastnosti integriranih vezij na mikro nivoju razlikujejo. Na primer, enaki logični elementi imajo različne zakasnitve in drugačno razporeditev silicija. Te razlike so ustvarile koncept fizikalno neklonirane funkcije (PUF). PUF generira vrednosti glede na okolje delovanja. Članek predstavlja obročni oscilator na osnovi PUF v dveh različnih integriranih vezjih za generacijo naključnih števil. Za demonstracijo sta bila uporabljena dva FPGA jedra s PUF strukturami. Za uporabo v aplikacijah so bila naključna števila post procesirana na osnovi Von Neumann študije. Opravljeni so bili tudi avtokorelacijski testi in NIST.

Ključne besede: generacija naključnih števil; FPGA; fizikalno neklonirane funkcije.

** Corresponding Author's e-mail:satuncer@firat.edu.tr*

1 Introduction

1.1 Background

The physically unclonable function (PUF) generates a response to the given challenge signals. The basic feature of the PUF is that the generated response is random and unpredictable. In other words, the response is random because the characteristics of the basic hardware (such as LE–Logic Element) used in PUF implementations are different from those of the other structures. Even if a PUF design uses the same challenges, it receives a different response, which indicates that the PUF hardware is unclonable [1].

There are two main implementation categories for PUF: authentication and secret key generation [2]. Authentication is performed in two steps. In the first step, the function generates a response against challenges, and stores it in a database. This phase is called enrolment. The second phase is verification. In this step, the response generated by the PUF for a challenge in the database is compared with the one stored in the database. The secret key generation has two stages: initialization and regeneration. In initialization, responses are generated against challenges, and the same challenge is applied to the PUF in regeneration. In the second phase, a key is generated to create the same response using a hash function and a decoding operation like BHC (Bose–Chaudhuri–Hocquenghem) based on the value generated in the first phase.

There are several PUF structures that have been proposed in the relevant literature. Generally, PUFs are investigated under four groups: arbiter, memory-based, optical, and ring oscillator (RO)-based. The underpinning idea of the arbiter PUFs is to measure the delay times between two signals. The Arbiter PUF structure transmits a pulsing signal to the output via two different routes. There are key blocks such as mux on the route. Although the researcher spent effort to design the two routes symmetrically, the delays in these routes have different times due to the production differences and characteristics. The arbiter gives the output 0 or 1 by examining the route that first sends the pulsing signal [3]. The static random-access memory (SRAM) type PUF is the most common memory-based PUF structure. The SRAM cell outputs consist of two inverters that are parallel to each other. The researcher gave the same characteristics to these inverters to the best ability. Due to manufacturing differences, the output value created by the SRAM cell will be different every time it receives power. These properties enabled SRAM cells to be used as PUF [4]. In optical PUF structure, which is also known as Physical One-Way Functions (POWFs), bubble-filled transparent epoxy is applied on the wafer. The pattern that emerges in this process is polished with laser. Because this pattern depends on the wavelength as well as laser angle, wafer material, wafer thickness and the characteristic of epoxy, each integrator has a unique pattern, and accordingly, a unique identity or signature [1, 5]. The RO-based PUF structure was proposed by Suh et al. [2]. This scheme simply generates a response specific to each challenge input based on the signals addressing the input of delay elements. This difference results from different physical and chemical properties of delay elements that emerge during the production.

It is possible to perform key generation in real random numbers with all PUF structures mentioned above. However, there are certain criteria (e.g. reverse engineering, emulation, man-in-the-middle attack, reconfiguration) that must be satisfied for random key generation to be used in computer science [6]. Past studies have proposed many RO-based PUF structures [6–8]. Indeed, the RO-PUF has very favorable statistical properties (output bias, intra-uniqueness and steadiness) when implemented in both FPGA (Field Programmable Gate Array) and ASIC (Application Specific Integrated Circuit) [7]. A relevant study proposed an RO-based PUF that performed number generation with the help of [9] and Gray coding by counting RO outputs with a counter. There are also other RO-based PUF structures in the literature that aim to increase the quality of random numbers by introducing periodic and non-periodic signals to challenge inputs. Chaos-based maps were used for non-periodic signals while Linear Feedback Shift Register (LFSR) structures were used for periodic signals [6, 10]. In this study, the researcher has suggested an alternative grouping based RO-PUFs for the classic RO-based PUF structure. The output generation mechanism of these systems depends on the frequency order of ROs of a group with two phases [11]. Another study presented a transient effect ring oscillator- (TERO) based PUF structure, and proved that this structure guaranteed high stability [12].

The double PUF structures proposed in the literature are based on arbiter PUF [13-15]. A double PUF structure was proposed to improve the unpredictability of responses [13]. In this study, the researcher used N-XOR Arbiter PUFs for unpredictable structures to sample the numbers from each arbiter PUF output with R-S type flip-flops, and then give them as inputs to an n-input XOR circuit. There were different output functions proposed for n = 2, n = 3 and n = 4 values. Steadiness, randomness, and cost were evaluated for each random number sequence. Another study sampled a double arbiter PUF structure using R-S type flip-flops and gave it as input to a two-input XOR circuit [14]. The difference between this study and the study presented in [13] was that this study applied Von Neumann post-processing to the produced numbers. The study also used Diehard test suite to measure the randomness of obtained numbers. A previous study [15] suggested a random number generator that had more than two PUF structures. This study has compared the outputs of each PUF circuit in a separate evaluation unit that did not include post processing, and performed number generation.

1.2 Motivation and contribution

This article, unlike the studies in literature, presents a design for random number generation by a RO-based Double PUF structure. The PUF circuits in the proposed structure were implemented on two separate FPGA cores. It is recommended that the characteristics of the Double PUF structure should be investigated so that the generated random numbers can be used as a key.

Thus, researchers should meet the requirements in Table 1. The study administered Diehard, NIST, and FIPS tests in addition to Scale index and autocorrelation to meet these requirements.

In order to meet these requirements, the study applied Von Neumann and hash-based post-processes to the random numbers that were generated by the new structure. The statistical weaknesses of generated numbers were determined by post-processing applications. NIST test suite and autocorrelation coefficients enabled the researcher to examine the quality of the numbers produced in this study. The contributions of this study to the literature are summarized as follows:

- Random number generation with PUF on two different FPGA.
- Transformation of the RO-based PUF structure into Double PUF structure
- Unlike the multi-PUF structures suggested in the literature, Von Neumann and hash-based post-processing procedures were used and the statistical properties of generated numbers were improved.
- Double PUF structure revealed better performance than the PUF structure.

Table 1: The Necessary Security Requirements for The Random Number Generator (RNG)

R1	Random numbers should not show any statistical weakness.
R2	Knowing the sub-array of random numbers should not allow the calculation or estimation of initial and consecutive random numbers.
R3	If the internal state value of the RNG is known, or even if it is unknown, it should not be possible to calculate previous random numbers with the possibility of being estimated.
R4	If the internal state value of the RNG is known, or even if it is unknown, it should not be possible to calculate next random numbers with the possibility of being estimated.

1.3 Study organization

Section 2 presents the basic components for real random number generation and the RO-based PUF structure. The double PUF structure that is proposed in the study as well as its implementation are presented in Section 3 while Section 4 describes the post processing operations. The results of implementation are given in Section 5. In the last section, the researcher evaluated the results.

2 True Random number generation

In real random number generators, random numbers are generated with the help of a physical noise source. The properties and randomness of random numbers generated by true random number generator (TRNG) depend on the randomness of physical processes. The physical noise source is random and its behavior is unpredictable, both of which reveal that the numbers produced are also random and unpredictable. However, the produced bit array may not show high randomness stability, and it may involve statistical weakness. To eliminate these weaknesses and produce more stable random numbers, the study maintained post-processing on the bit array. Post-processing operations reduce the amount of bit and eliminate weaknesses. The disadvantage is that TRNG is slow and costly, and it depends on random noise sources. However, TRNGs are frequently used in key generation because they are unpredictable and non-reproducible, and also achieve good statistical properties required for cryptographic applications [16]. Figure 1 shows the basic structure for generating real random numbers. First, the structure samples the signal obtained from the noise source, and then generates the pure random number by subjecting it to post-processing. Post-processing component is usually benefited to fill the deficiency of DAS(Digitized Analog Signal) in TRNG systems.

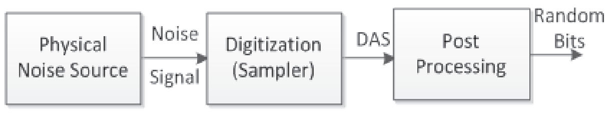


Figure 1: Basic components for TRNG

In RO-based random number generators, the study used propagation delay differences of integrated circuits as the noise source. Figure 2 presents the basic structure of a ring oscillator. RO which consists of an odd number of inverters includes a NAND gate to obtain the delay of the signals given from the input to the output. This structure proposed by Knuth requires a sampling and post processing circuit to produce true random numbers [17]. The quality of random numbers generated by TRNGs depends on the number of oscillators, sampling frequency, and number of inverters in oscillators [17].

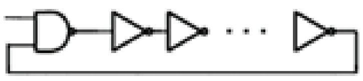


Figure 2: Ring oscillator

Previous studies in the relevant literature proposed RO-featured PUF structures to generate random numbers

[2, 3]. Figure 3 presents the typical RO-PUF content. The system has two 64x1 multiplexers, sixteen counters of 16-bit, one-comparator and one hundred twenty-eight-ring oscillators. Depending on the characteristics of each inverter, such as temperature and propagation delay, inverter outputs have a random quality. Because of these features, ROs create unpredictable variations even when inverters are implemented on the same integrated circuit. If the frequency of RO oscillations is too high, counters may not count oscillations. For this reason, it is necessary to adjust the number of inverters in each RO to ensure proper oscillation frequency. In the RO-PUF design, the frequencies of the output signals with each multiplexer's challenge can be set f_1 and f_2. The oscillation numbers of these frequencies are counted by the *counter*$_1$ and the *counter*$_2$ in the system. As a result of counting, the comparison circuit generates 0 or 1 according to Eq.1 [18].

$$r_{ab} = \begin{cases} 1 & if\ f_1 > f_2 \\ 0 & otherwise \end{cases} \tag{1}$$

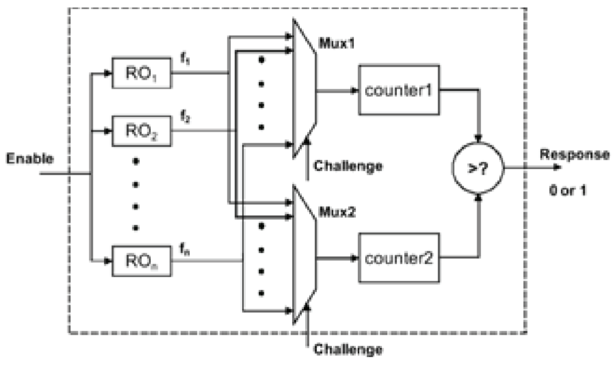


Figure 3: RO-PUF structure

3 Proposed double RO-PUF and implementation

Among all the PUF types, an RO-based PUF structure is most suitable for implementation on an FPGA [17]. Ring oscillator design with inverter and nand gate elements was presented by Sunar el al. Ring oscillators developed for random number generation consist of one NAND gate and 13 inverters. The number of inverters set to 13 so that the entropy of the numbers generated in their designs is high [19]. The 13th inverter output is given to the NAND gate as an input by feedback. The first input of the NAND gate is driven by a multiplexer hardware. With an externally applied excitation signal to the multiplexer's challenge, the RO output becomes a square wave signal. Figure 4 presents the design of a ring oscillator (in a RO-based PUF structure that was explained in Figure 3) in an FPGA environment. Because the propagation delays of the inverters used in each RO are different, each RO output generates a signal at different frequencies. Previous studies implemented the RO-based PUF using 64 or 128 RO. The present study sent the signals obtained from ROs to Mux inputs. The select inputs of the mux are given signals called challenge. The numbers are counted by counters, and the generation of random numbers is completed with a simple comparison.

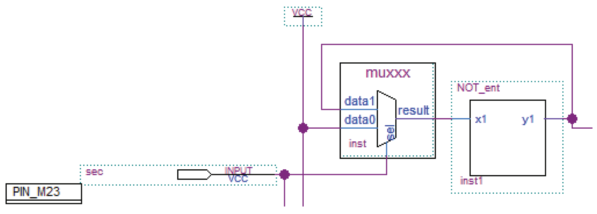


Figure 4: Implementation of RO in the FPGA

In this study, random number generation was implemented in real time using the RO-PUF structure that is described above in general. The proposed system in this study includes two RO-based PUF structures, as shown in Figure 5.a. Each RO-based PUF contains 64 RO and 64x1 Mux as shown in figure 5. b. Each RO-based PUF was implemented on EP4CE115FC7 and EP4CE22 FPGA cores.

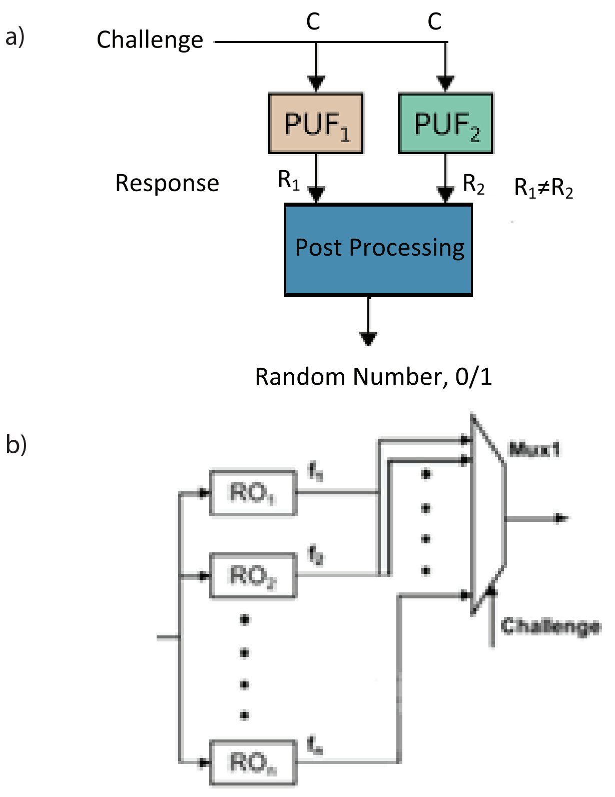


Figure 5: a) The suggested structure b) The PUF1 structure

The frequency of the signals given to challenge inputs enabled sampling the numbers that each PUF would generate at the f_1 and f_2 frequencies. The signals given to the challenge inputs act as the LFSR structures with the characteristic equation $x^{15}+x+1$. The obtained numbers were subjected to post-processing to strengthen their statistical properties and use them as a key. The study administered Von Neumann and hash-based post processing to achieve this goal. The new structure proposed in this study is simple, but very effective. Although the same challenge signals are given to each PUF structure, they produced different responses. The reason of this difference is the chemical properties (e.g. the properties of germanium and silicium elements, and the changes of their atoms in the cubic lattice structure) and the physical properties during integration of the logic elements (the geometry of transistors, oxide thickness, width) that are the main elements of the FPGA hardware.

The researcher recorded the generated numbers in the memory circuit to determine their statistical properties. Also, the system used a 16-bit counter to write the generated numbers on each address of the memory. In each sampling operation, the counter shows the next address of the memory, and a random number is recorded into this address. The sampling frequency and counter frequency in the system were 50 MHz for each. Quartus software allows the system to save the numbers in the memory units into a file with the mif or hex extension. The numbers were recorded in a file with the mif extension. Thus, is was an easy procedure to calculate the statistical properties of these numbers. Figure 6 shows the designed memory hardware.

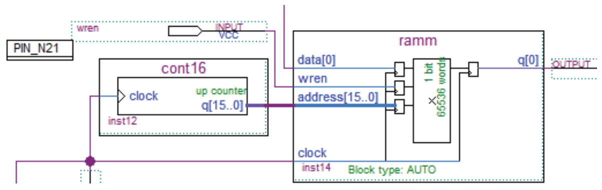


Figure 6: Recording of generated number into the memory unit

4 Post processing

4.1 Von Neumann Post Processing

Post-processing is the oldest and simplest method to eliminate the irregularities in the generated number array. The number generators to which Von Neumann post processing method is applied can generate normally distributed 0 and 1 numbers. The researcher took into consideration the simultaneous pairs that were generated in this post-processing method. If the number is (1, 0), the system generates 1. On the other hand, it generates 0 when the number is (0, 1). (0, 0) and (1, 1) number arrays are ignored. Figure 7 presents the application of Von Neumann post-processing to the generated random numbers. The entropy of the numbers created by this operation is close to the ideal value. However, this method has a disadvantage, which is the slow rate of number generation due to the elimination of the produced bit arrays.

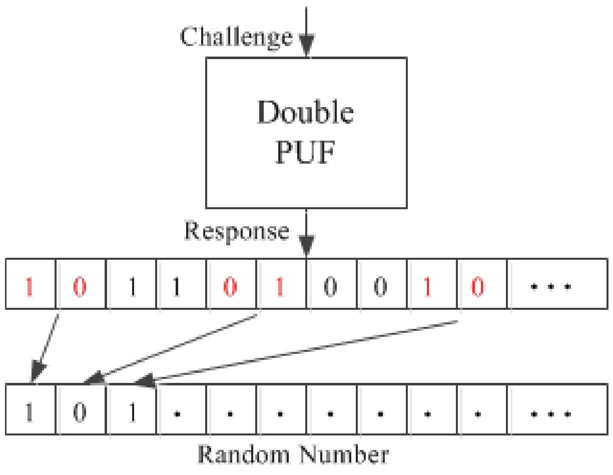


Figure 7: Von Neumann post-processing

4.2 Hash-based Post-processing

The H function is based on XOR gate and a post-processing that achieves more successful results than the XOR. It processes randomly-generated numbers in groups of 16 bits with a base of 2. It subjects the first 8 bits of each 16 bits to the XOR processing. The first eight bits are shifted one left. The result of this operation is subjected to the XOR with the last 8 bits. These operations produce an 8-bit output.

The generated number array for post-processing can be accepted D_0, D_1... D_{15} and considered 16 bit. Then, 16 bit is divided into two blocks; the first 8 bit being A_1 while the last 8 bit being A_2. The A_1 number array is subjected to the XOR by shifting the A_1 by one bit to the left. The resultant array is subjected the XOR with the A_2 number array and the number generation is achieved. Eq.2 shows the hash function.

H(A1,A2)=XOR(XOR(A1,rotateleft(A1,1)),A2) (2)
D=0110111011110100,
A1=01101110, A2=11110100
rotateleft(A1,1)=11011100,
H(A1,A2)= ((11011100 XOR 01101110) XOR 11110100)= 01000110

Figure 8 demonstrates the implementation of Hash post-processing function. The example in this section decreases the number generation rate by 50% with the hash function post-processing.

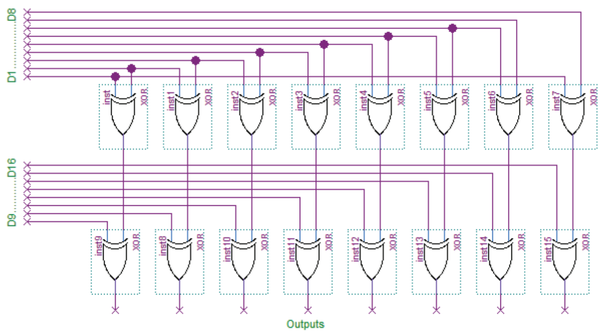
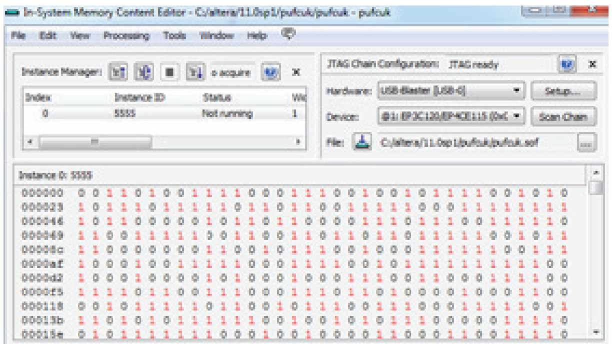


Figure 8: The post-processing of the H function

5 Implementation results

The new double RO-based PUF structure proposed in this study is capable of generating infinite random numbers. The generated numbers were stored in the memory to determine their statistical properties. Figure 9 shows the real-time generated numbers that are stored in the memory as a result of the hash-based post-processing while Figure 10 shows the actual numbers obtained from the output of the memory hardware.


In-System Memory Content Editor - C:/altera/11.0sp1/pufcuk/pufcuk - pufcuk
File
Edit
View
Processing
Tools
Window
Help
Instance Manager:
o acquire
JTAG Chain Configuration:
Setup...
Scan Chain

JTAG ready

Hardware: USB-Blaster [USB-0]

Device: @1: EP3C120/EP4CE115 (0x0

File: C:/altera/11.0sp1/pufcuk/pufcuk.sof

Index	Instance ID	Status	Wi
0	5555	Not running	1

Instance 0: 5555

```
000000  0 0 1 1 0 1 0 0 1 1 1 1 0 0 0 1 1 1 0 0 1 0 0 1 0 1 1 1 1 0 0 1 0 1 0
000023  1 0 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 0 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1
000046  1 0 1 1 0 0 0 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0
000069  1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 1
00008c  1 1 0 0 0 0 0 0 0 1 1 0 0 1 0 1 1 1 0 0 1 0 0 0 1 1 1 1 1 1 0 0 1 1 1
0000af  1 0 0 0 1 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0
0000d2  1 0 0 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 1 1 0 0
0000f5  1 1 1 1 0 1 1 0 0 1 1 1 0 0 0 1 1 1 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
000118  0 0 1 0 1 1 1 1 1 0 1 1 0 0 1 0 1 1 1 0 0 1 1 1 0 0 0 1 1 1 1 1 0 0 1
00013b  1 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0
00015e  0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 1 1 0
```

Figure 9: Real-time generated numbers stored in the memory

Figure 11 shows the resource utilization report obtained for the PUF1 implemented in the FPGA. The total numbers of LUTs consumed are 1110. Table 2 shows the comparison of the proposed approach with the literature according to FPGA resource utilization. The resource consumption of the proposed method according to the table.2 is acceptable average level.

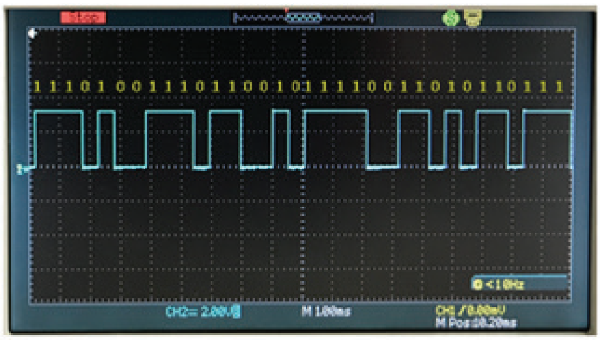
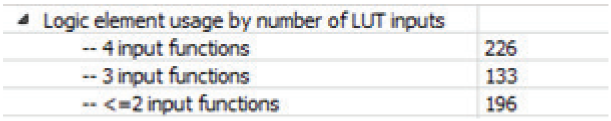


Figure 10: The numbers obtained from memory output



Logic element usage by number of LUT inputs	
-- 4 input functions	226
-- 3 input functions	133
-- <=2 input functions	196

Figure 11: FPGA resource utilization of PUF1

Table 2: FPGA resource utilizations

Reference	Type	FPGA resource utilization
[20]	RO PUF	4096 LUTs
[21]	LUT based on PUF	1280 LUTs
[22]	Arbiter PUF	1428 LUTs
[23]	RO PUF	1288 LUTs
[24,26]	SR Latch PUF	1024 LUTs
[25]	SR Latch PUF	2048 LUTs
[27]	RO PUF	732/1084/1436 LUTs
[28]	RO PUF	1024/1280/2048/ 2560 LUTs
Proposed Double PUF	RO PUF	1110 LUTs

The study investigated the statistical properties of the numbers generated by both Von Neumann and hash-based post-processing according to NIST test suite. P-value is known as randomness measure in the NIST Test suite. If P-value is 0, numbers are not random at all. it is desirable that this value is greater than 0.01 in computer science. With the proposed system according to Table 3, random numbers have produced.

Table 3: The results obtained according to NIST test suite

	Von Neumann	Hash	Raw bits
Frequency Monobit Test	0.715	0.0458	Failure
Frequency Test with a Block	0.709	0.478	Failure

Runs Test	0.141	0.964	Failure
Longest Run of Ones in a Block	0.285	0.715	Failure
Binary Matrix Rank	0.826	0.229	0.642
Discrete Fourier Transform	0.516	0.433	Failure
Non-Overlapp. Temp. Matching	0.611	0.622	Failure
Overlapping Template Matching	0.208	0.738	Failure
Universal Test	0.214	0.696	Failure
Linear Complexity	0.606	0.905	0.393
Serial Test	0.792	0.271	Failure
Approximate Entropy	0.137	0.33	Failure
Cumulative Sum	0.929	0.073	Failure

Moreover, the numbers generated by these procedures were subjected to autocorrelation test. Correlation indicates a linear relationship between two or more variables, and takes values between +1 and –1. If it is equal to or close to 0, there is no linear relationship between the variables. The purpose of this test is to control the correlation between the produced bit array bi and its shift version. When *n* has bit array length, *d* will be a constant integer and $1 \le d \le (n/2)$. The mathematical definitions of the test are given in Eq.3 and 4 [14].

$$A(d) = \sum_{i=1}^{n-d-1} b_i \, xor \, b_{i+d} \tag{3}$$

$$X_5 = \frac{2A(d) - (n-d)}{\sqrt{n-d}} \tag{4}$$

If $\{b_i\}$ is a real random array and $n \to \infty$, this random variable has a normal distribution *N(0, 1)*. If $|X5| < 1.6449$ ($\alpha=0.05$), then the test is successful. Table 4 shows the autocorrelation test results.

Table 4: Autocorrelation Test Results

	d	Double Puf	Result
Autocorrelation	8	0.0249	Passed
	10	0.0316	Passed

6 Conclusion

It is important to use random numbers in authentication and secret key generation. PUF structures are used for low-cost authentication and in the key generation for symmetric and asymmetric encryption. In such applications, random numbers that are generated for key and authentication are the most important parameters that determine the security of the application. With these issues in mind, we proposed a double PUF structure in this study, which is a unique structure in the relevant literature. We used RO-based PUF content for the new structure because it was suitable for being implemented on FPGA. The system was implemented on two different FPGA cores that were manufactured with 65 nm technology. Used FPGA boards have different characteristics because of core manufacturing conditions. In this case, the proposed system will not be cost-effective. To overcome this disadvantage, we suggest that the system should be implemented with two different cores on a single integrated circuit. The study employed Von Neumann and hash-based post-processing to remove the bit irregularities in the generated random numbers. The statistics of the numbers proved that they were successful with both pre-treatments. Considering the features of the PUF structure and its success in statistical tests, the study concluded that the new structure can be used in key generation and authentication applications.

7 References

1. R. Pappu, B. Recht, J. Taylor, N. Gershenfeld, "Physical one-way functions", *Science*, 297(5589): 2026-2030, 2002.
2. G. E. Suh, S. Devadas, "Physical unclonable functions for device authentication and secret key generation", *44th Annual Design Automation Conference, ser. DAC '07*. New York, NY, USA: ACM, , pp. 9–14, 2007
3. J. W. Lee, D. Lim, B. Gassend, G.E. Suh, M. Van Dijk, S.A. Devadas, "Technique to build a secret key in integrated circuits for identification and authentication applications", *In Proc. Symposium on VLSI Circuits. Digest of Technical Papers*,, pp.176-179,2004.
4. D. E. Holcomb, W.P. Burleson, K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers", *IEEE Transactions on Computers*, 58(9): 1198-1210, 2009.
5. P. Tuyls, B. Skoric, S. Stallinga, T. Akkermans, W. Ophey, "An information theoretic model for physical uncloneable functions", *In Proc. International Symposium on Information Theory,* Jun. 2004, p.139.
6. E. Avaroğlu, T. Tuncer, A.B. Özer, B. Ergen, M. Tűrk, "A novel chaos-based post-processing for TRNG". *Nonlinear Dyn.* 1–11 (2015)
7. A. Maiti, P. Schaumont, "Improving the quality of a physical unclonable function using configurable ring oscillators", *In Proc. the 19th International Conference on Field-Programmable Logic and Applications*, Aug. 31-Sept. 2, pp.703-707,2009.

8. J. H. Anderson, "A PUF design for secure FPGA-based embedded systems", *In Proc. the 15th Asia and South Pacific Design Automation Conference*, Jan. pp.1-6,2010.
9. Ş. Baş and M. E. Yalçın, "Key generation and license authentication using physical unclonable functions," *23nd Signal Processing and Communications Applications Conference (SIU),* Malatya, pp. 387-390. 2015.
10. Y. Hori, H. Kang, T. Katashita, A. Satoh, "Pseudo-LFSR PUF: A Compact, Efficient and Reliable Physical Unclonable Function," *2011 International Conference on Reconfigurable Computing and FPGAs*, Cancun, pp. 223-228, 2011.
11. G. Komurcu, A. E. Pusane, G. Dundar, "Dynamic programming based grouping method for RO-PUFs,", *9th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME),* 2013.
12. L. Bossuet, X. T. Ngo, Z. Cherif, V. Fischer, "A PUF Based on a Transient Effect Ring Oscillator and Insensitive to Locking Phenomenon", *IEEE Transactions on Emerging Topics in Computing,* vol. 2, no. 1, pp. 30-36, March 2014.
13. T. Machida, D. Yamamoto, M. Iwamoto, K. Sakiyama, "A New Arbiter PUF for Enhancing Unpredictability on FPGA", *Scientific World Journal,* 864812, 2015.
14. D. Froerer, T. Peterson, "True Random Number Generation with ADouble Arbiter PUF", https://spaces.usu.edu/download/attachments/.../puf.pdf.
15. A. Maiti, V. Gunreddy, P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions" in Embedded Systems Design with FPGAs., New York, NY, USA:*Springer-Verlag,* pp. 245-267, Nov. 2012.
16. F. Özkaynak, "Cryptographically secure random number generator with chaotic additional input", *Nonlinear Dyn.* doi:10.1007/s11071-014-1591-y, 2014.
17. K. Wold, C.H. Tan, "Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Ring", *International Conference on Reconfigurable Computing and FPGAs,* pp.385-390, 2008.
18. A. Maiti, J. Casarona, L. McHale, P. Schaumont, "A large scale characterization of RO-PUF", *Proceedings of the International Workshop on Hardware-Oriented Security and Trust(HOST)*, pp. 94–99, 2014.
19. B. Sunar, W. J. Martin and D. R. Stinson, "A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks," in *IEEE Transactions on Computers*, vol. 56, no. 1, pp. 109-119, Jan. 2007.
20. S. Pei, J. Zhang, R. Wang, "A low-overhead RO PUF design for Xilinx FPGAs", IEICE Electronics Express, Vol.15, No.5, 1–8, 2018.
21. Xu, T., Potkonjak, M. "Robust and Flexible FPGA-based Digital PUF", In: International Conference on Field Programmable Logic and Applications, pp. 1–6, Sept 2014.
22. A. Spenke, R. Breithaupt, R. Plaga, "An arbiter PUF secured by remote random reconfigurations of an FPGA", In International Conference on Trust and Trustworthy Computing, pages 140–158. Springer, 2016.
23. A. Maiti, R. Nagesh, A. Reddy, P. Schaumont, "Physical unclonable function and true random number generator: a compact and scalable implementation", In ACM Great Lakes Symposium on VLSI (2009).
24. B. Habib, J. Kaps, K. Gaj, "Efficient SR-latch PUF", Proc. 11th International Symposium on Applied Reconfigurable Computing, April 15–17, 2015, Bochum, Germany (2015), pp. 205-216.
25. D. Yamamoto, K. Sakiyama, M. Iwamoto, K. Ohta, M. Takenaka, K. Itoh, "Variety enhancement of PUF responses using the locations of random outputting RS latches", Journal of Cryptographic Engineering, vol. 3, pp. 197-211, November 2013.
26. B. Habib J.-P. Kaps, K. Gaj, "Implementation of efficient SR-Latch PUF on FPGA and SoC devices", Microprocessors and Microsystems Volume 53, August 2017, Pages 92-105.
27. L. Parrilla, E. Castillo, D. P. Morales, A. García, "Hardware Activation by Means of PUFs and Elliptic Curve Cryptography in Field-Programmable Devices",Electronics, vol:5, 5, 2016
28. A. Wild, G. T. Becker and T. Güneysu, "On the problems of realizing reliable and efficient ring oscillator PUFs on FPGAs," *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, McLean, VA, 2016, pp. 103-108.

Arrived: 06. 01. 2018
Accepted: 29. 05. 2018