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.

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]. Au-
thentication 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, and the counter, in the system. As a result of counting, the comparison circuit generates 0 or 1 according to Eq.1 [18].

\[
r_{ab} = \begin{cases} 
1 & \text{if } f_1 > f_2 \\
0 & \text{otherwise}
\end{cases}
\] (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

Random Number, 0/1
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 silicon 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, it was an easy procedure to calculate the statistical properties of these numbers. Figure 6 shows the designed memory hardware.

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.

\[
H(A_1,A_2) = \text{XOR}(\text{XOR}(A_1,\text{rotateleft}(A_1,1)),A_2) \quad (2)
\]

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.

\[
D = 0110111011110100, A_1 = 01101110, A_2 = 11110100 \\
\text{rotateleft}(A_1,1) = 11011100, H(A_1,A_2) = (11011100 \text{ XOR } 01101110 \text{ XOR } 11110100) = 01000110
\]
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.

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.

Table 2: FPGA resource utilizations

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

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

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Von Neumann</th>
<th>Hash</th>
<th>Raw bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency Monobit Test</td>
<td>0.715</td>
<td>0.0458</td>
<td>Failure</td>
</tr>
<tr>
<td>Frequency Test with a Block</td>
<td>0.709</td>
<td>0.478</td>
<td>Failure</td>
</tr>
</tbody>
</table>
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 $b_i$ and its shift version. When $n$ has bit array length, $d$ will be a constant integer and $1 \leq d \leq (n/2)$. The mathematical definitions of the test are given in Eq.3 and 4 [14].

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

$$X^2_s = \frac{2A(d) - (n-d)}{\sqrt{n-d}}$$  \hspace{1cm} (4)

If $(b)$ is a real random array and $n \to \infty$, this random variable has a normal distribution $N(0, 1)$. If $|X^2_s|<1.6449$ ($\alpha=0.05$), then the test is successful. Table 4 shows the autocorrelation test results.

**Table 4: Autocorrelation Test Results**

<table>
<thead>
<tr>
<th>d</th>
<th>Double Puf</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autocorrelation 8</td>
<td>0.0249</td>
<td>Passed</td>
</tr>
<tr>
<td>10</td>
<td>0.0316</td>
<td>Passed</td>
</tr>
</tbody>
</table>

## 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


