http://dx.doi.org/10.5755/j01.eee.21.4.12779 ELEKTRONIKA IR ELEKTROTECHNIKA, ISSN 1392-1215, VOL. 21, NO. 4, 2015

Implementation of Duplicate TRNG on FPGA by Using Two Different Randomness Source

Taner Tuncer[1]
[1]*Department of Computer Engineering, Firat University, 23119, Elazig Turkey*
ttuncer@firat.edu.tr

***Abstract*—It is critical in terms of security that numbers used in cryptographic systems are generated by hardware such as Field Programmable Gate Array (FPGA). The most common method to develop high-quality and fast True Random Number Generators (TRNGs) is to use Ring Oscillator (RO). Jitter is mostly used as an approach to provide randomness source in clock signals generated by RO so that random numbers can be generated. Numbers generated by RO based TRNGs are random because jitter generated by RO is random. However, it is a disadvantage that these numbers do not have good statistical properties. Post-processing is applied so as to overcome this disadvantage. In this article, a duplicate system to generate numbers with statistically good properties without post-processing is proposed. The system consists of two TRNGs developed by ROs on two different FPGA boards. The proposed system has six different scenarios with different frequencies and quantity of ROs. It is shown in the proposed system that post-processing is not needed, and numbers are generated in high data rates such as 47.68, 71.52 and 95.37 Mbit/s. The proposed system also successfully passed NIST 800.22 tests.**

***Index Terms*—Hybrid integrated circuit, jitter, random number generation.**

I. INTRODUCTION

Random numbers are widely used in fields of cryptography and data transmission, where key generation is important. Random numbers need to have three essential features in order to be used in the mentioned applications. First feature is that generated numbers should be unpredictable. This feature makes sure that next random numbers to be generated cannot be predicted by examination of previous numbers by a RNG. Second feature is that generated numbers should attain good statistical properties. Lastly, generated number streams are not to be generated again. In other words, numbers are not to be periodically generated. There are basically two groups of random number generators; namely, True Random Number Generator (TRNG) and Pseudo Random Number Generator (PRNG) which are non-deterministic and deterministic, respectively. While TRNGs are expensive and slow for many applications, PRNGs are effective and more appropriate for many applications because they have more simple structure. Besides, PRNGs generate periodic numbers due to being deterministic generators. Thus, the numbers can be easily predicted and they are not suitable for cryptographic applications. Numbers generated by these generators can be used as key in information security systems. However, uncontrolled generation of keys out of security systems reduces reliability of systems. Therefore, system becomes more secure if keys are generated in an integrated circuit (cryptographic system on chip) [1]. Hence, it is getting more and more popular to generate keys by programmable hardware such as FPGA (Field Programmable Gate Array) [2]–[5]. FPGAs are programmable integrated circuits, whose internal structure can be altered multiple times according to aimed functions. FPGA is commonly used nowadays due to the fact that it provides great flexibility in design stage, and it has parallel processing capability.

Quality of numbers in terms of randomness by TRNG-based random number generation is dependent on the type of physical noise. Radioactive decay, thermal noise and ring oscillator take place in literature as physical noise. As shown in Fig. 1, a TRNG consists of physical noise source, digitization and post-processing components [6].

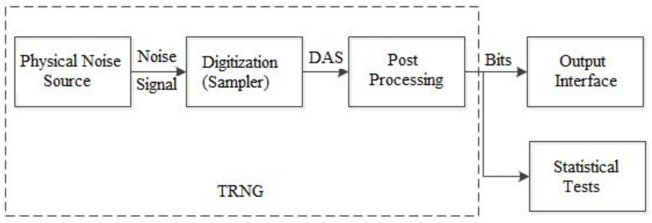

Fig. 1. TRNG components.

The last two components, digitization and post-processing, are generally named as Digitized Analog Signal (DAS) and internal random numbers, respectively. Statistical properties of random numbers generated by ring oscillators are not good. Thus, high quality numbers can be obtained by post-processing of numbers generated by ring oscillators. Post-processing component is usually benefited to fill the deficiency of DAS in TRNG systems.

In this paper, a duplicate structure including two TRNG systems on two different FPGA boards based on RO is proposed. FPGA cores in the proposed system have different features leading to different random signals generated by RO from each core. Instead of only one FPGA core, two different FPGA cores are used so that signal can be more variable. It is a disadvantage that a system with this structure consumes more power. On the other hand, it is an advantage that post-processing is not used, high quality random

Manuscript received November 16, 2014; accepted March 18, 2015.

numbers are generated, and data rate of the system is higher. NIST 800.22 test suit is used for statistical tests of random numbers generated by the system. According to the results of the test suit, the system is suitable for cryptographic applications.

Remaining sections of the study are organized as follows: developments in literature are mentioned in Section II; design parameters and structure of ring oscillator and the proposed system are explained in Section III; development of duplicate structure on FPGA boards for six different scenarios is presented in Section IV. Besides, the structure creating numbers which are generated by the system is explained in Section IV. Statistical comparison of tests applied on generated random numbers is given in Section V. Lastly, results are discussed in conclusion.

II. RELATED WORKS

One of the most effective techniques to generate real random numbers is to use a noise source. Clock signals are obtained from noise source, and deviation of signals from their correct positions occurs. This deviation is also called as jitter. Jitter caused by clock signals can be characterized by a Gaussian distribution [7]. Jitter is an undesired feature in a system. However, being random makes jitter an ideal tool to generate random numbers in systems which are similar to TRNG.

Random numbers are generated on hardware such as FPGA by using Phase Locked Loop (PLL) and ring oscillator [7], [8]. The design proposed by Fischer *et al.* in [8], two PLL systems produce random jitter. This approach has a disadvantage that PLL systems are in limited quantities on FPGA boards. In the RO-based system proposed by Sunar *et al.* in [7], there are 114 ROs and 13 inverters in each RO. In this paper by Sunar *et al.*, the mathematical model (urn model) of TRNG is given. Statistical properties of numbers generated by this system at 40 MHz are not good. Generated numbers by this system are processed by resilient post-processing in order to overcome the statistical disadvantage. As a result, 2.5 Mbit/s data rate is obtained.

The reason for using XOR, LFSR Shuffler or Von Neumann corrector in post-processing is to decrease generated numbers. Consequently, system efficiency is observed to decrease. A new model without post-processing and with good statistical properties was developed by Wold *et al.* [2]. This model has a sampling circuit in each RO apart from the model proposed by Sunar *et al.* in [7]. Usage of sampling circuits enables number of ROs to be reduced to 25. In this way, numbers with higher quality are generated.

Data rate of TRNGs utilizing ROs is generally lower. Metastable ring oscillator to increase data rate was proposed by Vasyltsov *et al.* [9]. This system has two clock frequencies; one is low, and other is high. Low clock frequency is used for sampling process; high clock frequency is used as a multiplexer (MUX), a device selecting input signals, and transmitted to signal sampling unit which is obtained by inverters. The proposed TRNG was tested according to FIPS 140-1/2 and AIS test suits. Data rate of the system was found to be 2.5 Mbit/s.

Fibonacci-based and Galois-based ROs are also included in these studies. These types of ROs consist of XOR logic gates [10], [11]. The TRNG which is developed to be GARO-based has D-type and toggle flip-flops. The system generates true random number in three different processes. D-type flip-flop is used for sampling; toggle flip-flop counts 0-1 passages of GARO. This system results in 6.25 Mbit/s data rate.

In another study, TRNG in Open Loop Structure is proposed in order to use generated numbers in cryptography applications [12]. The biggest advantage of this system is that it has a high data rate, and it does not need special components such as PLL. This system, which contains post-processing, has n delay elements and n D-type flip-flops. System is found to be successful in NIST tests, and data rate is measured to be around 10 Mbit/s.

Apart from mentioned studies, an ASIC application of TRNG is developed [13]. Data rate of this system including Von Neumann corrector is obtained to be 18.5 Mbit/s. Also, it was found out that higher data rate can be reached by increasing number of ROs in the system.

Duplicate structures in random number generation are proposed as well as individual usage of TRNGs. PRNG and chaos-based random number generators are used with TRNG in duplicate structures [14], [15]. The purpose of using these structures is to generate high quality numbers in high data rates. A chaotic-based TRNG system with 1 Mbit/s is obtained with CMOS circuit elements in [15].

Current studies in this field proceed with obtaining high data rate with high quality numbers, and they aim to improve these criteria.

III. THE PROPOSED SYSTEM

A typical TRNG consists of three components as shown in Fig. 1. The main source of randomness is entropy source. Signals resulting from unpredictable physical phenomena such as thermal noise, oscillator jitter or a chaotic signal can be used as entropy source. Periodic sampling circuits, aperiodic sampling circuits or comparator circuits, all of which can be used for sampling, convert discrete signals received from entropy source into digital signals. Numbers generated by entropy source do not have good statistical properties. In this case; Von Neumann, XOR or LFSR Shuffler circuits are used for post-processing. Post-processing leads to numbers with higher quality, but it results in lower rate of number generation.

A common method to generate TRNG is to use ROs. Quality of numbers generated by a RO-based TRNG depends on the following three parameters. These are:

1. Number of oscillators in system
2. Sampling frequency
3. Number of inverters in oscillator

Randomness of numbers generated by RO-based TRNG depends on randomness of jitter signal obtained by each RO. Signal frequency $f_s = 1/(2nt_{inv})$ from RO is proportional to the number of inverters (n) used in the system and delay time (t_{inv}) of inverters implemented by Logic Element(LE) in FPGA. Jitter has more random if inverters are generated in different logic array blocks (LAB) instead of being generated in one LAB by using LE which is main component of FPGAs [2]. In addition, delay value of each gate element is dependent on natural events during generation stage of LEs. In other words, Jitter from ROs depends on following features related with hardware.

1. Power supply
2. Cross-Talk Noise
3. Pink Noise or Flicker Noise
4. Temperature
5. Propagation delay

Hence, features mentioned above will be different in each type of hardware where RO is set. In other words, jitter from ROs on each board will be different. Jitter generated by RO can be represented as a Gaussian distribution with μ mean and σ standard deviation [8]

$$f(x)=\begin{cases}\dfrac{1}{\pm\sqrt{2®}}e^{-\frac{1}{2}\left(\frac{x-«}{\pm}\right)^2} & -\infty<x<\infty,\\ 0 & other.\end{cases} \qquad (1)$$

There are two independent variables of jitter signal generated by two randomness sources and these variables have means $«_1$ and $«_2$ and standard deviations $\pm_1$ and $\pm_2$.

Sum of probability density functions $f_1(x)$ and $f_2(x)$ of two independent random variables with Gaussian distribution has also Gaussian distribution

$$F(x)=\frac{1}{2®(\pm_1\pm_2)}e^{-\frac{1}{2}\left(\frac{x-(«_1+«_2)}{\pm_1\pm_2}\right)^2}. \qquad (2)$$

The Gaussian distribution obtained by convolution has $«_1+«_2$ mean and $\pm_1\pm_2$ standard deviation. Based on this result, we propose a duplicate structure that uses jitter to generate true random numbers and jitter can be represented by a Gaussian distribution with $«_1+«_2$ mean and $\pm_1\pm_2$ standard deviation.

Development of TRNGs on two different kinds of hardware guarantees that generated random numbers will be in higher quality. The main idea behind the proposed method is RO structure given by [7]. In Fig. 2, the proposed system contains two randomness sources, two binary XOR trees and one sampling unit.

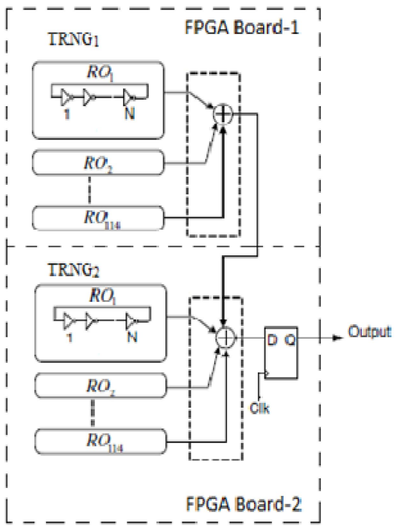

Fig. 2. The proposed system.

ROs set on different kinds of hardware result in randomness in jitter, which eventually leads to randomness in generated numbers. The TRNG in the proposed system is composed of only RO and binary XOR tree logic gates. Random signal received from TRNG1 is used as input signal for binary XOR tree logic element of TRNG2. The biggest advantage of the proposed duplicate system is that the system has higher data rate value than values existing in literature due to the fact that post-processing is not used.

As a result of this design, our contribution is as follows. The usage of TRNG structure implemented on two different FPGA boards is not suitable for cryptographic applications due to security reasons. Each FPGA board has different characteristics because of core manufacturing conditions and operating conditions. We proposed that the proposed system should be implemented on a single integrated circuit with two different FPGA cores.

IV. IMPLEMENTATION

Altera’s FPGA-based 60-nm EP4CE22F17C6 and EP4CE115F29C7 development boards are used so as to measure real performance of the proposed system. The experimental setup is shown in Fig. 3.

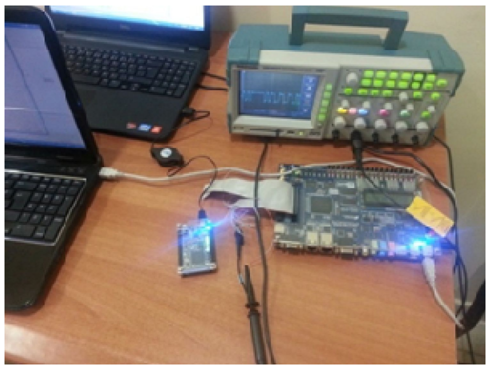

Fig. 3. A view of the experimental setup.

The system is tested for six different scenarios. Clock frequencies having 50 % duty cycle with 50 MHz, 75 MHz and 100 MHz are chosen in order to obtain probability of equal quantities of 1 and 0 in generated numbers for each scenario. Quantities of ROs in designs are 114 and 25 as proposed by [7] and [2], respectively. Also, the inverter numbers used in ROs are 13 and 3, respectively. Only one sampling unit is used for each number generation. All components in the system are modelled by VHDL (Very-high-speed integrated circuit Hardware Description Language) by making use of behavioural and structural models.

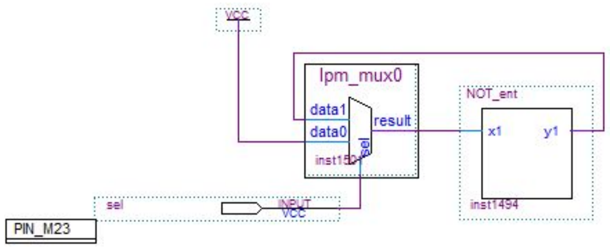

Fig. 4. Ring oscillator structure.

Figure 4 shows each RO schematically. Each RO generates 0 logic value in the beginning. An external signal (sel = 1) from physical environment is given to MUX, and then data1 inputs connected to each inverter outputs are forwarded to inverter input. By this way, delay signals (jitter) are obtained from inverter outputs according to predefined parameters. Figure 5 shows the sampling circuit and developed structure to store generated numbers. Generated numbers are stored in a memory element so that statistical tests can be applied on them. Each generated number is assigned to a RAM address by a counter used in a system. Address space of memory is 216 bits. In other words, 65536 bits are generated by the system and saved in memory element.

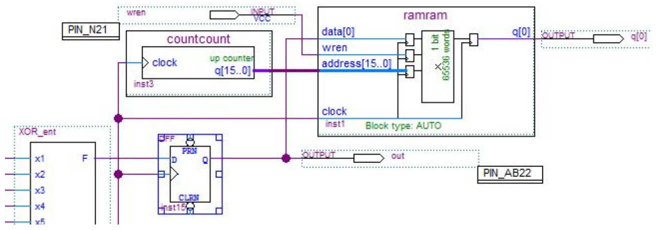

Fig. 5. Sampling circuit.

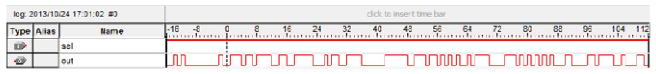

Fig. 6. Sampled numbers in 50 MHz clock frequency.

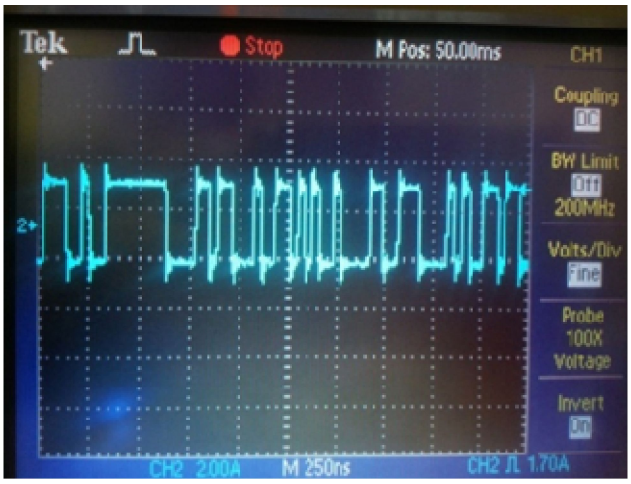

Fig. 7. Variation of numbers generated in 50 MHz sampling frequency.

If the proposed system has 114 ROs, 13 inverters and clock frequency with 50 MHz, Fig. 6 shows variation of bits generated by simulation. Figure 7 shows variation of bits real time generated by duplicate system which has 50 MHz sampling frequency.

V. EXPERIMENTAL RESULTS

Developed true random number generators are subject to randomness tests in order to be safely used in cryptographic applications. There exists many test suites to analyse randomness of generated numbers. One of these test suits is NISTS 800.22 test suit which is statistical-based. There are 15 tests in NIST 800.22 test suit, and parameters of each test explained in detail in [16]. One of the most important parameters used in test is α value which is known as significance level. Selecting significance level as 0.01 indicates 99 % confidence interval. Another parameter is P-value. P-value is known as randomness measure. If P-value is 0, numbers are not random at all. The significance level, α, to be used in cryptographic application must be appropriately chosen. If P-values are greater than or equal to α value for each test, then test is accepted to be successful. Otherwise, test fails. In order words, generated numbers are not random. Significance level is commonly chosen in the range of [0.001, 0.01].

Table I–Table IV show P-values obtained according to NIST 800.22 for six different scenarios. Significance level is 0.01 for each test. Without post processing, TRNG-1 and TRNG-2 fail in tests. Duplicate system with 114 ROs succeed in tests in each frequency. Data rate of the duplicate system is measured to be 47.68 Mbit/s, 71.52 Mbit/s and 95.37 Mbit/s. The duplicate system with 25 ROs succeed in tests in 50 MHz, 75 MHz and 100 MHz sampling frequencies. Data rates of this system are found to be 47.68 Mbit/s, 71.52 Mbit/s and 95.37 Mbit/s.

TABLE I. NIST 800.22 TEST RESULTS FOR 114 RO, 13 INVERTER, 50 MHZ.

	TRNG1	**TRNG2**	**Duplicate**
Frequency Monobit Test	Failure	Failure	**0.818**
Frequency Test with a Block	Failure	Failure	**0.656**
Runs Test	Failure	Failure	**0.723**
Longest Run of Ones in a Block	Failure	Failure	**0.841**
Binary Matrix Rank	0.588	0.647	**0.312**
Discrete Fourier Transform	Failure	0.735	**0.594**
Non-Overlapp. Temp. Matching	Failure	0.145	**0.665**
Overlapping Template Matching	Failure	Failure	**0.473**
Universal Test	Failure	Failure	**0.431**
Linear Complexity	0.465	0.719	**0.647**
Serial Test	Failure	Failure	**0.707**
Approximate Entropy	Failure	Failure	**0.891**
Cumulative Sum	Failure	Failure	**0.634**

TABLE II. NIST 800.22 TEST RESULTS FOR 114 RO, 13 INVERTER, 75 MHZ.

	TRNG1	**TRNG2**	**Duplicate**
Frequency Monobit Test	Failure	Failure	**0.492**
Frequency Test with a Block	Failure	Failure	**0.431**
Runs Test	Failure	Failure	**0.624**
Longest Run of Ones in a Block	Failure	Failure	**0.747**
Binary Matrix Rank	0.316	0.501	**0.554**
Discrete Fourier Transform	Failure	0.695	**0.265**
Non-Overlapp. Temp. Matching	Failure	0.740	**0.615**
Overlapping Template Matching	Failure	Failure	**0.452**
Universal Test	Failure	Failure	**0.395**
Linear Complexity	0.373	0.774	**0.449**
Serial Test	Failure	Failure	**0.668**
Approximate Entropy	Failure	Failure	**0.693**
Cumulative Sum	Failure	Failure	**0.115**

TABLE III. NIST 800.22 TEST RESULTS FOR 114 RO, 13 INVERTER, 100 MHZ.

	TRNG1	**TRNG2**	**Duplicate**
Frequency Monobit Test	Failure	Failure	**0.378**
Frequency Test with a Block	Failure	Failure	**0.301**
Runs Test	Failure	Failure	**0.586**
Longest Run of Ones in a Block	Failure	Failure	**0.364**
Binary Matrix Rank	0.164	0.449	**0.512**
Discrete Fourier Transform	Failure	0.527	**0.198**
Non-Overlapp. Temp. Matching	Failure	0.287	**0.815**
Overlapping Template Matching	Failure	Failure	**0.374**
Universal Test	Failure	Failure	**0.288**
Linear Complexity	0.364	0.619	**0.637**
Serial Test	Failure	Failure	**0.565**
Approximate Entropy	Failure	Failure	**0.471**
Cumulative Sum	Failure	Failure	**0.052**

TABLE IV. NIST 800.22 TEST RESULTS FOR 25 RO, 3 INVERTER, 50 MHZ.

	TRNG1	**TRNG2**	**Duplicate**
Frequency Monobit Test	Failure	Failure	**0.863**
Frequency Test with a Block	Failure	Failure	**0.786**
Runs Test	Failure	Failure	**0.720**
Longest Run of Ones in a Block	Failure	Failure	**0.744**
Binary Matrix Rank	0.266	0.142	**0.295**
Discrete Fourier Transform	0.631	0.656	**0.417**
Non-Overlapp. Temp. Matching	Failure	Failure	**0.345**
Overlapping Template Matching	0.016	0.072	**0.440**
Universal Test	Failure	Failure	**0.237**
Linear Complexity	0.794	0.379	**0.717**
Serial Test	Failure	Failure	**0.436**
Approximate Entropy	Failure	Failure	**0.048**
Cumulative Sum	Failure	Failure	**0.961**

TABLE V. NIST 800.22 TEST RESULTS FOR 25 RO, 3 INVERTER, 75 MHZ.

	TRNG1	**TRNG2**	**Duplicate**
Frequency Monobit Test	Failure	Failure	**0.016**
Frequency Test with a Block	Failure	Failure	**0.576**
Runs Test	Failure	Failure	**0.416**
Longest Run of Ones in a Block	Failure	Failure	**0.584**
Binary Matrix Rank	0.119	0.868	**0.722**
Discrete Fourier Transform	0.191	0.704	**0.315**
Non-Overlapp. Temp. Matching	Failure	Failure	**0.424**
Overlapping Template Matching	Failure	Failure	**0.506**
Universal Test	Failure	Failure	**0.419**
Linear Complexity	0.138	0.314	**0.702**
Serial Test	Failure	Failure	**0.391**
Approximate Entropy	Failure	Failure	**0.515**
Cumulative Sum	Failure	Failure	**0.012**

TABLE VI. NIST 800.22 TEST RESULTS FOR 25 RO, 3 INVERTER, 100 MHZ.

	TRNG1	**TRNG2**	**Duplicate**
Frequency Monobit Test	Failure	Failure	Failure
Frequency Test with a Block	Failure	Failure	**0.324**
Runs Test	Failure	Failure	**0.668**
Longest Run of Ones in a Block	Failure	Failure	**0.279**
Binary Matrix Rank	0.596	0.816	**0.738**
Discrete Fourier Transform	0.931	0.344	**0.954**
Non-Overlapp. Temp. Matching	Failure	Failure	**0.071**
Overlapping Template Matching	Failure	0.011	**0.812**
Universal Test	Failure	Failure	**0.433**
Linear Complexity	0.143	0.396	**0.558**
Serial Test	Failure	Failure	**0.628**
Approximate Entropy	Failure	Failure	**0.178**
Cumulative Sum	Failure	Failure	Failure

According to Tables, TRNG1 and TRNG2 cannot be produced quality number with the use of a single entropy source. The numbers having high quality and good statistical properties were produced by using two different entropy sources in the proposed duplicate system. Moreover, the results showed that, high speed and high quality numbers can be produced without the need for post-processing.

VI. RESULTS

Random numbers are necessary for many cryptographic applications. The TRNG structure implemented on two different FPGA boards is not convenient for cryptographic application due to security reasons. Each FPGA core has different characteristics due to manufacturing and operating conditions. In order to eliminate security problems of the proposed system, we suggest that two different FPGA cores should be used on a single integrated circuit. For this purpose, a structure enabling high quality random number generation in high data rate without post processing is proposed. Jitter obtained from two different hardware sources by ROs is used as randomness source. The system is developed for six different scenarios, and statistical quality of generated numbers is tested according to NIST 800-22 statistical test suite. Despite excluding post processing in the proposed system, first five scenarios successfully passed NIST 800.22 statistical tests. In the sixth scenario; number of ROs decreased from 114 to 25, and frequency was set to 100 MHz. However, this configuration failed in cumulative sum and frequency monobit tests of NIST 800.22 test suite. The system needs more development on this point. It was demonstrated by tests that random numbers can be generated at most with 95.37 Mbit/s data rate without post processing.

REFERENCES

[1] M. Jonathan, C. J. Cerda, C. D. Martinez, H. David, K. Hoe, “Random number generators using cellular automata implemented on FPGAs”, *44th IEEE Southeastern Symposium Son system Theory*, 2012, pp. 67–72.

[2] K. Wold, C. H. Tan, “Analysis and enhancement of random number Generator in FPGA based on oscillator ring”, *Int. Conf. on Reconfigurable Computing and FPGAs*, 2008, pp. 385–390. [Online]. Available: http://dx.doi.org/10.1109/reconfig.2008.17

[3] R. Santoro, O. Sentieys, S. Roy, “On-the fly evaluation of FPGA-based true random number generator”, *IEEE Computer Society Annual Symposium on VLSI*, 2009, pp. 55–60. [Online]. Available: http://dx.doi.org/10.1109/isvlsi.2009.33

[4] E. Erkek, T. Tuncer, “The implementation of ASG and SG random number generators”, *ICSSE 2013*, pp. 363–367. [Online]. Available: http://dx.doi.org/10.1109/icsse.2013.6614692

[5] T. Tuncer, E.Avaroglu, M. Turk, A. BedriOzer, “Implementation of non-periodic sampling true random number generator on FPGA”, *Journal of Microelectronics, Electronic Components and Materials,* vol. 44, no. 4, pp. 296–302, 2014.

[6] W. Schindler, W. Killmann, “Evaluation criteria for true (physcal) random number generators used in cryptographic applications”, in *4th Int. Workshop on Cryptographic Hardware and Embedded Systems*, 2003, pp. 431–449.

[7] B. Sunar, W. J. Martin, D. R. Stison, “Aprovably secure true random number generator with built-in tolerance to active attacks”, *IEEE Transaction on Computers*, vol. 56, no. 1, pp. 109–119, 2007. [Online]. Available: http://dx.doi.org/10.1109/TC.2007.250627

[8] V. Fischer, M. Duratovsky, “True random number generator embedded in reconfigurable hardware”, *LNCS 2523*, 2003, pp. 415–430. [Online]. Available: http://dx.doi.org/10.1007/3-540-36400-5_30

[9] I. Vasyltsov, E. Hambardzumyan, Y. S. Kim, B. Karpinskyy, “Fast digital TRNG based on metastable ring oscillator”, *Int. Workshop on Cryptographic Hardware and Embedded Systems, LNCS 5154*, 2008, pp. 164–180. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85053-3_11

[10] M. Dichtl, J. D. Golic, “High-speed true random number generation with logic gates only”, in *Int. Workshop on Cryptographic Hardware and Embedded System*, 2007, pp. 45–62. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74735-2_4

[11] J. D. Golic, “New methods for digital generation and post processing of random data”, *IEEE Trans. Computers*, vol. 55, no. 10, pp. 1217–1229, 2006. [Online]. Available: http://dx.doi.org/10.1109/TC.2006.164

[12] J. L. Danger, S. Guilley, P. Hoogvorst, “High speed true random number generator based on open loop structures in FPGAs”, *Microelectronics Journal*, vol. 4, pp. 1650–1656, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.mejo.2009.02.004

[13] U. Guler, S. Ergun, “A high speed fully digital IC random number generator”, *Int. Journal of Electronics and Communications*, vol. 66, pp. 143–149, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.aeue.2011.06.001

[14] I. Cicek, A. E. Pusane, G. Dundar, “A novel design method for discrete time chaos based true random number generators”, *Integration, The VLSI Journal*, 2014.

[15] T. Stojanovski, J. Pihl, L. Kocarev, “Chaos-based random number generators part II: practical realization”, *IEEE Trans. Circuits and System I: Fundamental Theory and Applications*, vol. 48, no. 3, pp. 382–385, 2001. [Online]. Available: http://dx.doi.org/10.1109/81.915396

[16] NIST, “NIST Random Number Generation and Testing”, 2006.