This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2019.2919891, IEEE Transactions on Circuits and Systems II: Express Briefs

FPGA-Based True Random Number Generation Using Programmable Delays in Oscillator-Rings

N. Nalla Anandakumar; Somitra Kumar Sanadhya; and Mohammad S. Hashmi, *Senior Member, IEEE*

***Abstract*—True random number generators play a fundamental role in cryptographic systems. This paper presents a new and efficient method to generate true random numbers on field programmable gate array by utilizing the random jitter of free-running oscillators as a source of randomness. The free-running oscillator rings incorporate programmable delay lines to generate large variation of the oscillations and to introduce jitter in the generated ring oscillators clocks. The main advantage of the proposed true random number generator utilizing programmable delay lines is to reduce correlation between several equal length oscillator rings, and thus improve the randomness qualities. In addition, a Von Neumann corrector as post-processor is employed to remove any bias in the output bit sequence. The validation of the proposed approach is demonstrated on Xilinx Spartan-3A FPGAs. The proposed true random number generator occupies 528 slices, achieves 6 Mbps throughput with 0.999 per bit entropy rate, and passes all the National Institute of Standards and Technology (NIST) statistical tests.**

***Keywords*-TRNG, FPGA, Ring Oscillators, Entropy, PDLs.**

I. Introduction

True random number generators (TRNGs) are widely used in cryptographic applications such as key generation, random padding bits, and generation of challenges and nonces in authentication protocols. Moreover, TRNGs also find use in lottery drawings, computer games, gambling and probabilistic algorithms. The TRNGs must fulfill strict statistical requirements, be unpredictable and generate truly random numbers by making use of a physical source that is non-deterministic. In general, a poor random number generator often leads to decrease in the complexity of attacking a system using such a generator. For example, highly insecure pseudo-random number generator used on Mifare Classic tags reduced the attack complexity and allowed attackers access to the smart cards secret key [1]. These issues can be addressed by utilizing TRNGs to produce the needed random bits in cryptographic systems. The randomness of a TRNG is usually assessed through Diehard [2] and NIST [3] statistical tests while the unpredictability is verified by estimating entropy-per-bit through stochastic model [4].

Typical TRNGs use a single source of entropy and post-processing operation. Commonly used sources of entropy are thermal noise [5], metastability [6], [7], clock jitter in circuits [8], [9] and chaos [10]. Randomness is first extracted from the physical noise source and then this is interpreted into raw random bit-stream using a sampler (digitization). In practice, the raw random bit-stream usually does not bear good quality of randomness. Therefore, auxiliary post-processing operations, such as Neumann corrector [11] or hash function [5] is required to improve the quality of the output TRNG bit stream and increase its randomness. In this context, a number of field programmable gate array (FPGAs) prototypes of TRNG with post-processing designs have been proposed [6], [7], [12], [13]. These designs derive entropy from the jitter of Ring Oscillators (RO) [12], [13] or the metastability of flip-flops [6], [7] which is caused by setup or hold time violations of flip-flops (FFs).

There are different problems that might arise in the construction of a TRNG based on oscillator rings implemented in FPGAs [14]. For example, the entropy of the output bit sequence from the TRNG would be drastically reduced when equal length oscillator rings configured in FPGAs are highly correlated with each other due to identical delays. To address this issue, we use programmable delay lines (PDLs) in the oscillator rings in the current work. This creates higher variation in RO oscillations from cycle to cycle and causes jitter in the generated RO clocks. Also, the output of the RO's are not correlated with each other due to the incorporation of the PDLs in the oscillator rings for each sampling clock which produces higher randomness. In addition, Von Neumann corrector is used as post-processor for improving statistical properties of the bitsteam produced by the proposed TRNG. We implement the base TRNG as well as the Von Neumann corrector on the same Xilinx Spartan-3A FPGAs (XC3S400A-4FTG256). The key contributions of this work are:

- Proposal of an FPGA-based TRNG that uses PDL-induced random jitter in the clocks of free-running ROs as the source of randomness.
- Demonstration of effectiveness of the Von Neumann corrector as a post-processor for bias elimination.
- Experimental validation of the proposed TRNG and demonstration that it passes all tests in the NIST suite.
- The hardware evaluation results demonstrate high throughput-per-area and high entropy rate (i.e., true randomness) of the produced output bits.

The subsequent sections briefly discuss the Xilinx Spartan-3A FPGAs and PDLs, some existing RO based TRNGs, and the details of the proposed TRNG. Subsequently experimental evaluation and comparisons in terms of area and throughput with existing techniques, and quality comparison by using the NIST statistical test suite are presented.

N. Nalla Anandakumar is with Society for Electronic Transactions and security (SETS), Chennai and IIIT-Delhi, India (email: nallananth@gmail.com)

Somitra Kumar Sanadhya is with the Department of Computer Science and Engineering, IIT-Ropar, India (email: somitra@iitrpr.ac.in)

Mohammad S. Hashmi is with the School of Engineering, Nazarbayev University, Astana, Kazakhstan, and the Department of Electronics and Communication Engineering at IIIT-Delhi, India. (email: mshashmi@iiitd.ac.in)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2019.2919891, IEEE Transactions on Circuits and Systems II: Express Briefs

II. Preliminaries

A. Xilinx Spartan-3A FPGA Structure

For prototyping of the proposed TRNG, Spartan-3A FPGA from Xilinx is employed and it can be considered as a grid of interconnected Configurable Logic Blocks (CLBs) subdivided into four logical components called slices. It has two pairs of CLB slices, SLICEL and SLICEM, with each slice containing two 4-input Look-Up Tables (LUTs) and two flip-flops (FFs). In brief, SLICEL is the most basic type of slice and supports logic only while SLICEM supports both logic and memory functions (including RAM16 and SRL16 shift registers). The LUT based primitives are instantiated in hardware description language (HDL) as described in Spartan-3 Libraries Guide for HDL Designs.

B. Programmable delay lines (PDLs)

The internal variations of FPGA Look-Up Tables (LUT's) can be generated from changes in the LUT's propagation delays under different inputs [6]. For example, the LUT in

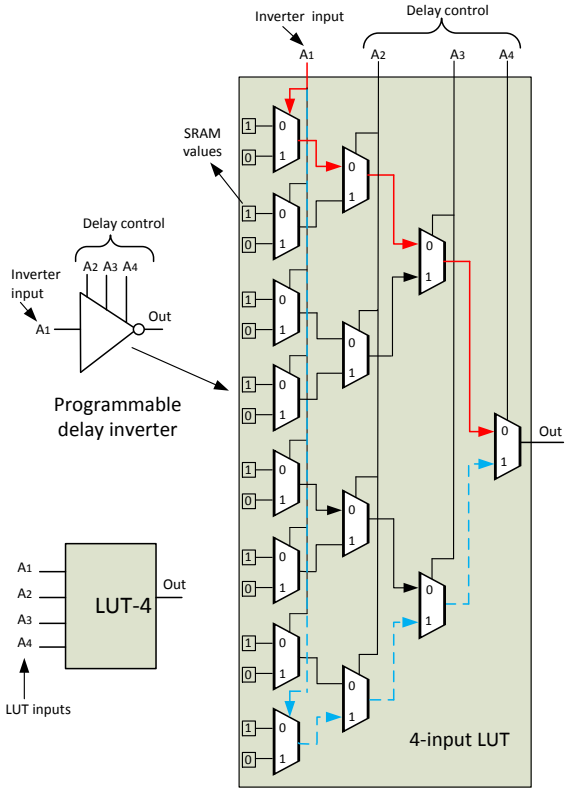

Fig. 1: PDL using a 4-input LUT.

Fig. 1 is programmed to implement an inverter whose LUT output (0) is always an inversion of its first input (A_1). Other inputs A_2, A_3, and A_4 act as "don't-care" bits but their values affect the signal propagation path from A_1 to the output (0). In this context, it has been shown, in Fig. 1, that the signal propagation path from A_1 to the output (0) is shortest for $A_2A_3A_4 = 000$ (marked with solid red line) and longest for $A_2A_3A_4 = 111$ (marked with dashed blue lines) for 4-input LUT's. Thus, a programmable delay inverter with three control inputs can be implemented by using one LUT. For the PDL, the first LUT input A_1 is the inverter input and the rest of the LUT inputs (three) are controlled by $2^3 = 8$ discrete levels.

III. Ring Oscillator Based TRNG

As mentioned earlier, a number of RO based TRNG designs have been reported in literature [12]–[18]. Typically, jitter is accumulated in the free-running RO's consisting of odd number of inverters or delay elements connected in ring configuration. This causes digital value of the oscillators output to change with a period of approximately $2DL$ where D is the delay of a single inverter and L is the number of inverters in an oscillator. Period of these oscillations vary from cycle to cycle causing jitter of the rising and falling edges of the generated RO clocks as shown in Fig. 2. Such oscillations, and hence jitter, in digital circuits can occur due to power supply variations, cross talks, semiconductor noise, temperature variations, and propagation delays. These jitters can be used to generate a stream of truly random bits using D flip-flop (DFF) based sampler for sampling the output of a high frequency oscillator as illustrated in Fig. 3.

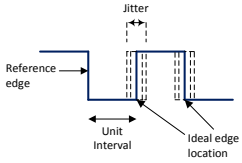

Fig. 2: Jitter in clock signals

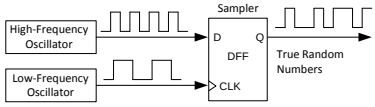

Fig. 3: Basic oscillator-based TRNG

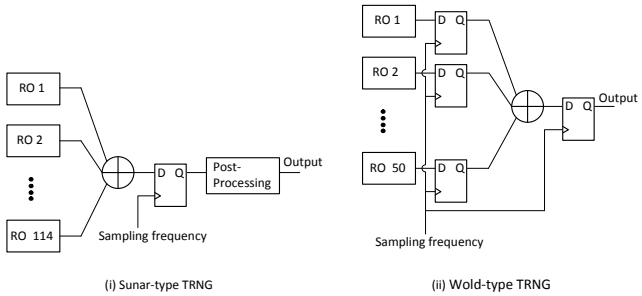

Fig. 4: Block diagram of the original TRNG (a) [12], and the modified TRNG (b) [15]

The quality of generated true random bits can be improved by employing multiple free-running RO's [12]. This is achieved by feeding the outputs to an XOR tree (i.e., a multi-input XOR) and then sampling the XOR tree by a reference clock with a fixed frequency using a DFF to generate the random bit stream as shown in Fig. 4. However, it is very challenging for the XOR-tree and the sampling DFF to handle high number of switching activity from the free-running RO's in such designs [13]. It is due to the fact that high number of transitions during a sampling period due to parallel RO's place stringent setup and hold-time requirements. This aspect can be addressed to a small extent by incorporating a sampling DFF at the output of each free-running RO as shown in Fig. 4 [15]. This design passes the NIST statistical tests without post-processing and employs reduced number of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2019.2919891, IEEE Transactions on Circuits and Systems II: Express Briefs

RO's. However, it has similar mathematical complexity to the original design [12] and hence similar associated problems such as mutual dependence of rings [17] and correlation between the rings [14], which cause a lack of entropy at the output of the TRNG. The randomness qualities of the original TRNG [12] can also be improved at the cost of higher hardware resources [18] or a lower throughput [16].

IV. The Proposed TRNG Architecture

It is imperative to note that the RO based TRNGs, although exciting, are extremely limited in terms of randomness when identical RO's are employed [14]. Equal length oscillator rings configured in FPGA are highly correlated with each other due to identical delays and therefore the XOR of the output from these rings returns mostly zeros. This leads to poor randomness from the design. We show in this work that the a TRNG incorporating PDL's can overcome this problem. Previously, the metastability of flip-flops was used for generating true random numbers [6]. They achieved metastability by using PDLs that accurately equalize the signal arrival times to flip-flops. On the other hand, our work uses random jitter of free-running oscillators for generating true random numbers. We employ the PDL's in oscillator rings to generate large variation of the oscillations and to introduce jitter in the generated RO's clocks. The main advantage of the proposed TRNG utilizing PDL's is to reduce correlation between several equal length oscillator rings. For example, this can be achieved by variable RO outputs for each sampling clock by incorporating PDL as shown in Fig. 5. Moreover, the variation in RO oscillations from cycle to cycle (CTC) is also introduced by each oscillator ring due to inverter-delay. As a result, the XOR operation significantly improve the randomness qualities.

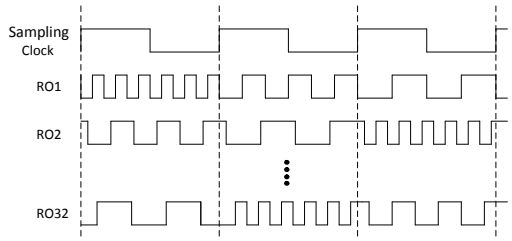

Fig. 5: RO outputs for each sampling clock by using PDL (variation in oscillations from CTC are not shown for clarity)

The implementation of the proposed TRNG along with the post-processing module on a low cost Xilinx Spartan-3A FPGAs is described in this section. Subsequently, Xilinx ISE design suite 13.4, the PyUSB-1.6 software, and the FT2232D USB interface is used for experimental evaluation of the proposed technique.

A. Design Overview

The proposed TRNG architecture is shown in Fig. 6. Here, each RO is realized using 3 inverters and 1 AND gate marked by black dashed boxes. The role of the AND gates is to enable the respective RO's. The design of the inverters and the AND gate require three and one LUT on the FPGA, respectively. In order to generate programmable delays inside the 4-input LUT, one of the LUT inputs is the ring connection while the other three inputs are configured with $2^3 = 8$ discrete levels. In case one uses 6-input LUT's (which are common in high-end FPGA's), one can use one input for ring connection and allow 2^5 = 32 delay configurations for the remaining LUT inputs. This allows configuring 32 RO's without any constraints on the placement of the inverters.

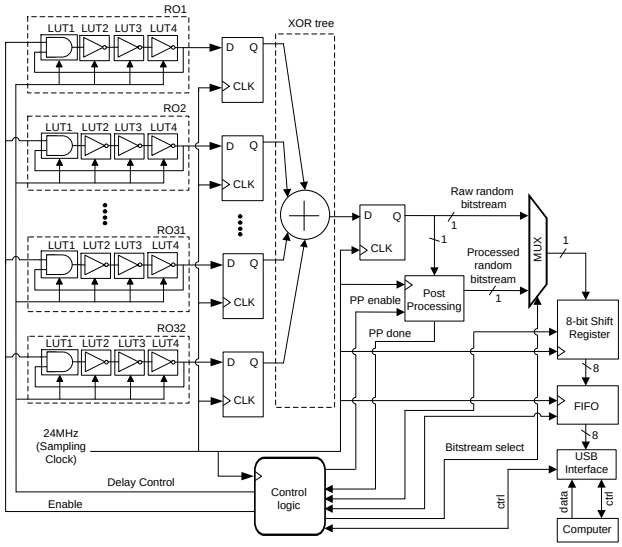

Fig. 6: Architecture of the proposed TRNG.

The proposed architecture consists of 32 RO's, XOR tree, DFF's, shift register, FIFO (First-In, First-Out), and a post-processing unit. First, the control circuitry starts the 32 RO's simultaneously using the 'enable' input. The RO outputs are then combined by the XOR tree and sampled at the frequency clock of 24 MHz. If higher operating frequency is used for sampling then a frequency divider may be needed as well. Then, for the generation of PDLs, 2^3 discrete levels are arbitrarily applied to the delay control inputs for each sampling clock. Subsequently the sampled bits are either fed to the post-processor unit or directly sent to the FIFO without post-processing. Thus the output is either raw random bitstream or processed random bitstream selected via control input of the multiplexer and collected in blocks of 8 bits using the 8-bit shift register. Finally, each byte is stored in a FIFO of 64 byte width (i.e. 512 bits) and sent to PC through a USB interface for the TRNG statistical analysis. The FIFO allows reading of raw/processed random bitstream without flow interruption. The control logic module enables the start and stop of the RO's, FIFO, 8-bit shift register, post-processing unit, and selection of the raw/processed random bitstream for transfer to the PC.

B. Post-Processing

The proposed implementation employs a simple post-processing unit based on a Von Neumann corrector [11] for enhancing the entropy and for removing any bias in the generated random bits. The post-processor also provides robustness of the TRNG output sequence. The employed Von Neumann corrector post-processing scheme is depicted in Fig. 7. We read two bits at a time from the raw TRNG output and discard them if both of them are same (i.e. we eliminate 00 and 11

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2019.2919891, IEEE Transactions on Circuits and Systems II: Express Briefs

patterns). If the two bits are different (i.e. 01 or 10) then we take the first bit and discard the second bit. On an average, the post-processing unit requires 512 bits of raw input to generate 128 bits of post-processed output.

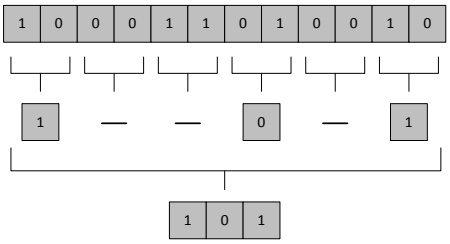

Fig. 7: Principle of operation of the Von Neumann.

V. Analysis and Discussion

A. Hardware overhead analysis

The results from the proposed TRNG architecture implemented on Xilinx Spartan-3A FPGAs are compared with recently reported TRNGs on Xilinx FPGAs in Table I. It is apparent that the proposed design outperforms earlier design [7] in terms of area. However, the throughput in our design is lower as we use a 24 MHz operating frequency (O.F) whereas the design in [7] uses a 100 MHz O.F. Furthermore, compared to previous works [6], [16], [19], our design exhibits higher throughput but occupies more area. The designs [6], [16], [19] employ more advanced FPGA configurations and hence achieves better area performance. On the other hand, our design outperforms [18] in terms of both the area and throughput although both are implemented on similar type of low cost FPGA. In summary, it can be concluded that the proposed TRNG architecture stands out as a potential candidate for lightweight secure applications considering that it provides very competitive area-throughput trade-offs.

TABLE I: Comparison of the proposed method with other existing TRNGs implemented on Xilinx FPGAs

TRNG Types	Area*			T.put (Mbps)	O.F. (MHz)	Platform
	LUTs	FFs	Slices			
THIS WORK	528	177	270	6	24	Spartan-3A (XC3S400A)
RO-based [18]	712	753	—	3.2	—	Spartan-3E (XC3S500E)
RO-based [19]	10	5	—	1.15	100	Spartan-6
RO-based [16]	521	131	—	2.57	—	Spartan-6
Meta stability [7]	—	—	580	12.5	100	Virtex-4 (XC4VFX20)
Meta stability [6]	128	—	—	2	—	Virtex-5 (XC3S500E)

* The Area for the TRNG with control logic (without USB interface).

B. Correlations Test

In order to investigate correlation between the rings used in the proposed TRNG, we computed the sample Pearsons correlation coefficient [14] between all combinations (i.e., $\binom{32}{2} = 496$) of the 32 RO's outputs. We generated 50,000 consecutive sampled bits from each of the RO's and used it for the test. The outputs of all these rings were sampled with an external sampling frequency of 24 MHz and 8 discrete levels were arbitrarily applied to the delay control inputs for each sampling clock. The correlation coefficient was observed to be ± 0.06 for all combinations. This allows us to conclude that the rings used in the proposed TRNG are not correlated with each other (i.e., correlation coefficient is close to 0). Moreover, we generated 80 million random bits from our proposed design and used it for the auto-correlation test specified in AIS-31 (T5) [4]. This test checks the bitwise correlation. The generated bit sequence passed the test and hence our proposed design has no correlation and dependency problems.

C. Measures of the Quality of Randomness

As per the standard practice in the domain, entropy test as the objective criterion of randomness is carried out. Followed by this, the restart test is carried out to provide evidence that the output, before post-processing, is distinct after restarting the system multiple times under the same conditions. Finally, the quality of the bitstream produced by the TRNG is tested using the NIST statistical test suite.

1) Entropy Estimation: The expected entropy per bit is 1 for an ideal true random number generator because the proportion of '0's and '1's is ideally 0.5. To estimate the entropy rate for the generated numbers, test $T8$ of the procedure B of the AIS-31 [4], available to test raw random numbers, is utilized in this work. Considering a sequence of length N, test $T8$ splits the sequence in $Q + K$ disjoint L-bit words. The suggested parameters for the test $T8$ are $L = 8$, $Q = 2560$ and $K = 256000$. The minimum sequence of length required for this test is 7200000 bits. We generated 80 million random bits from our implementation and used it for the test. The bit sequence passed the test and achieved entropy of 7.9946 bits per byte (i.e., 0.9993 per bit). This is better than the minimum entropy requirement of 7.976 per byte (i.e., 0.997 per bit). Therefore, the proposed TRNG successfully fulfills the criteria for procedure B of the AIS-31 and can be used for applications that require high security. Moreover, Our proposed RO-based TRNG achieves higher entropy per bit when compared to earlier designs [16] (0.999 per bit) and [19] (0.997 per bit).

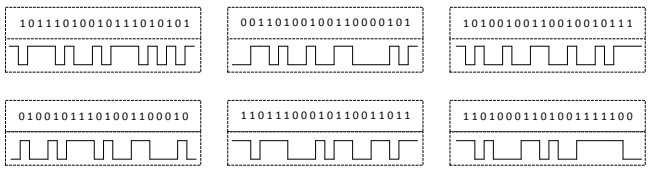

Fig. 8: 6 output bitstreams captured after restarting the TRNG.

2) Restart Experiment: A test was carried out to verify the start up sequences for 6 restarts, from identical starting conditions, for the proposed design. First 20 sampled bits after every restart were observed and plotted as shown in Fig. 8. For a pseudorandom signal, the restart experiment produces similar graphs when the TRNG is repeatedly started from identical starting conditions. However, in our case, these were observed to be different each time and thus discard any pseudo randomness possibilities. Works [9], [15] and [13] used similar procedure to distinguish the amount of true randomness contained in a pseudorandom oscillating signal.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSII.2019.2919891, IEEE Transactions on Circuits and Systems II: Express Briefs

3) Statistical Evaluation of the Output: A set of randomness tests is standardized by NIST for evaluation of the quality of randomness of bitstreams [3]. The NIST statistical test suite is the most comprehensive publicly available tool. In essence, there are 15 type of tests which are typically carried out to assess the performance of TRNGs. In this work, the parameters of each test were set as per the NIST recommendations [3]. The significance level α was chosen to be the default of 0.01 as it indicates 99% confidence interval. To examine the distribution of the P-values (randomness measure), 1000 times (sequences) of 10^6 bits both before and after post-processing were acquired. The statistical results of P-values and proportions (before and after post-processing) are given in Table II. If P-value is larger than 0.01 then the sequences are approximately uniformly distributed. Furthermore, a test is considered to have passed when the range of acceptable proportions are between 0.98056 and 0.99943 for α = 0.01 and sequences = 1000 [3]. The Proportion (Prop.) column indicates the number of P-values that were above the 0.01 confidence interval. The minimum proportion (minimum pass rate) for the tests is 0.982 before post-processing, and 0.987 after post-processing. The minimum pass-rate of sequences for our design (0.987) is higher in comparison to [9] (i.e., 0.97) and [7] (i.e., 0.986). The TRNG is then tested under two different temperature values of $55°C$ and $75°C$ using a temperature-controlled chamber while keeping the core supply voltage at 1.2 V. The minimum proportion for the tests is 0.981 at both these temperatures before post-processing. However, after post-processing, the minimum proportion becomes 0.983 and 0.982 at $55°C$ and $75°C$ respectively. In summary, it is evident from the results that the proposed TRNG exhibits better randomness (P-value is greater than 0.01 with greater proportion) properties at low hardware overhead and high throughput.

TABLE II: NIST randomness test results (uniformity of P-values and proportion of passing sequence) for 1-Gbits record that passed all tests (room temperature at 1.2 V).

Statistical Test	Before Post-processing			After Post-processing		
	P-value	Prop.	Result	P-value	Prop.	Result
Frequency	0.5592	0.990	PASS	0.6199	0.992	PASS
Block-Freq.	0.6621	0.994	PASS	0.7441	0.991	PASS
Cumsum	0.3776	0.987	PASS	0.4138	0.987	PASS
Runs	0.4590	0.992	PASS	0.4276	0.993	PASS
Long-Run	0.7791	0.989	PASS	0.7011	0.990	PASS
Rank	0.7197	0.985	PASS	0.7634	0.988	PASS
FFT	0.5141	0.990	PASS	0.6593	0.994	PASS
Overlapping	0.0962	0.987	PASS	0.0805	0.991	PASS
Approx-Entropy	0.0553	0.988	PASS	0.2145	0.992	PASS
Serial	0.3345	0.991	PASS	0.4698	0.992	PASS
Linear-Complex	0.0088	0.989	PASS	0.2703	0.989	PASS
Nonperiodic	0.1573	0.990	PASS	0.4213	0.993	PASS
Universal	0.0909	0.982	PASS	0.2771	0.988	PASS
Rand-Excur	0.3394	0.986	PASS	0.3781	0.990	PASS
Rand-Variant	0.2587	0.984	PASS	0.3347	0.987	PASS

VI. Conclusion

A new design of RO-based TRNG is described and its implementation on Xilinx Spartan-3 FPGA is presented in this paper. The programmable delay of FPGA LUTs has been used to achieve random jitter and to enhance the randomness. It has been demonstrated that the proposed implementation provides a very good area-throughput trade-off. Effectively, it is capable of producing a throughput of 6 Mbps after post-processing with a low hardware footprint. In addition, the restart experiments show that the output of the proposed TRNG behaves truly random. It achieves high entropy rate and successfully passes all NIST statistical tests. It can actually be inferred that the proposed design has the potential to be a good candidate for lightweight security applications.

References

[1] K. Nohl, D. Evans, S. Starbug, and H. Plötz, "Reverse-engineering a Cryptographic RFID Tag," in *Proceedings of the 17th Conference on Security Symposium*. USENIX Association, 2008, pp. 185–193.

[2] G. Marsaglia, "Diehard: A Battery of Tests of Randomness," 1996.

[3] Bassham III and Lawrence E. et. al, "SP 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," Tech. Rep., 2010.

[4] W. Schindler and W. Killmann, "A proposal for: Functionality classes for random number generators," 2011.

[5] B. Jun and P. Kocher, "The Intel random number generator," Cryptography Research, Inc., April 1999.

[6] M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA-Based True Random Number Generation Using Circuit Metastability with Adaptive Feedback Control," in *Cryptographic Hardware and Embedded Systems – CHES 2011*. Springer Berlin Heidelberg, 2011, pp. 17–32.

[7] H. Hata and S. Ichikawa, "FPGA Implementation of Metastability-Based True Random Number Generator," *IEICE Transactions on Information and Systems*, vol. E95.D, no. 2, pp. 426–436, 2012.

[8] A. P. Johnson, R. S. Chakraborty, and D. Mukhopadhyay, "An Improved DCM-Based Tunable True Random Number Generator for Xilinx FPGA," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 4, pp. 452–456, April 2017.

[9] D. Liu, Z. Liu, L. Li, and X. Zou, "A Low-Cost Low-Power Ring Oscillator-Based Truly Random Number Generator for Encryption on Smart Cards," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 6, pp. 608–612, June 2016.

[10] A. Beirami and H. Nejati, "A Framework for Investigating the Performance of Chaotic-Map Truly Random Number Generators," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 7, pp. 446–450, July 2013.

[11] J. von Neumann, "Various techniques used in connection with random digits," in *Monte Carlo Method*. National Bureau of Standards Applied Mathematics Series, 12, 1951, pp. 36–38.

[12] B. Sunar, W. J. Martin, and D. R. Stinson, "A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks," *IEEE Transactions on Computers*, vol. 56, no. 1, pp. 109–119, Jan 2007.

[13] M. Dichtl and J. D. Golić, "High-Speed True Random Number Generation with Logic Gates Only," in *Cryptographic Hardware and Embedded Systems - CHES 2007*. Springer Berlin Heidelberg, 2007, pp. 45–62.

[14] K. Wold and S. Petrovi, "Security properties of oscillator rings in true random number generators," in *2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, April 2012, pp. 145–150.

[15] K. Wold and C. H. Tan, "Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings," in *Int. Conf. on Reconfigurable Computing and FPGAs*, Dec 2008, pp. 385–390.

[16] O. Petura, U. Mureddu, N. Bochard, V. Fischer, and L. Bossuet, "A survey of ais-20/31 compliant trng cores suitable for fpga devices," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2016, pp. 1–10.

[17] N. Bochard, F. Bernard, V. Fischer, and B. Valtchanov, "True-Randomness and Pseudo-Randomness in Ring Oscillator-Based True Random Number Generators," *Int. J. Reconfig. Comp.*, vol. 2010, pp. 879 281:1–879 281:13, 2010.

[18] A. Maiti, R. Nagesh, A. Reddy, and P. Schaumont, "Physical Unclonable Function and True Random Number Generator: A Compact and Scalable Implementation," in *Proceedings of the 19th ACM Great Lakes Symposium on VLSI*, ser. GLSVLSI. ACM, 2009, pp. 425–428.

[19] B. Yang, V. Roic, M. Grujic, N. Mentens, and I. Verbauwhede, "Estrng: A high-throughput, low-area true random number generator based on edge sampling," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, pp. 267–292, Aug. 2018.