ISSN: 2633-4828 Vol. 5 No. S6, (Oct - Dec 2023)

International Journal of Applied Engineering & Technology

LOW POWER AND AREA EFFICIENT FPGA-BASED TRUE RANDOM NUMBER GENERATION BASEDON LUT VARIANT RING OSCILLATORS

S. Harish, T. Rammohan, C. Iswarya, Jisha Mathew and Anjitha N K
Rathinam Technical Campus, Coimbatore - 641021, Tamil Nadu
Harish.professor2022@gmail.com, dean.ecebme@rathinam.in,
jishamathew.ece@rathinam.in,iswarya.rtc@rathinam.in, anjitha.rtc@rathinam.in

ABSTRACT

Random padding bits, authentication protocol challenges, and key generation are just a few of the cryptographic applications that make extensive use of True Random Number Generators (TRNGs). The suggested approach makes use of the random jitter of free running oscillators as a source of randomness in order to create real random numbers on field programmable gate arrays in an efficient and novel way. Variable LUT causes the total RO's area to decrease and its randomness to increase in various ring oscillators. Programmable delay lines are incorporated into the free-running oscillator rings to produce a wide range of oscillations and jitter in the clocks of the created ring oscillator. The primary benefit of the suggested true random number generator with configurable delay lines is that it improves unpredictability by decreasing correlation between several oscillator rings of identical length. Furthermore, as a post- processor, a Von Neumann corrector is used to eliminate bias from the bit sequence output. The suggested methodology is validated using Xilinx Spartan-3E FPGAs.

Keywords: PDLs, Ring Oscillators, FPGA, and TRNG.

I. OVERVIEW

In cryptographic applications including key generation, random padding bits, and the creation of challenges and nonces for authentication protocols, true random number generators, or TRNGs, are commonly employed. Additionally, TRNGs are used in probabilistic algorithms, computer games, gambling, and lottery drawings. The TRNGs must use a nondeterministic physical source to produce really random numbers, be unexpected, and meet stringent statistical requirements. The complexity of attacking a system with a random number generator tends to reduce when the generator is of low quality.For instance, the extremely risky pseudorandom number generator on Mifare Classic tags made attacks easier to execute and gave hackers access to the secret key on smart cards [1]. By using TRNGs to generate the necessary random bits for cryptographic systems, these problems can be resolved. Diehard [2] and NIST [3] statistical tests are typically used to evaluate a TRNG's randomness, and stochastic model [4] is used to estimate entropy per bit to confirm unpredictability.Standard TRNGs employ a single postprocessing procedure and source of entropy. Thermal noise[5], metastability [6], [7], clock jitter in circuits [8], [9], and chaos [10] are often employed sources of entropy. First, the physical noise source's randomness is collected, and this is then translated into unprocessed, arbitrary bit-stream with a sampler (digitization). In real-world applications, the unprocessed random bitstream typically lacks a high degree of randomness. For this reason, additional postprocessing steps like a hash function [5] or Neumann corrector [11] are needed to enhance the output TRNG bit stream's quality and randomness. Several field programmable gate array (FPGA) prototypes of TRNG with post-processing designs have been put out in this regard [6, 7, 12, 13, 13]. The jitter of Ring Oscillators (RO) [12], [13] or the metastability of flip-flops [6], [7], which is brought on by setup or hold time violations of flip- flops (FFs), are the sources of entropy for these devices. Building a TRNG based on oscillator rings implemented on FPGAs can present a variety of challenges [14]. For instance, when equal length oscillator rings set in FPGAs are closely coupled with each other because of identical delays, the entropy of the output bit sequence from the TRNG would be dramatically lowered. In the current work, we use programmable delay lines (PDLs) in the oscillator rings to address this problem. This produces jitter in the generated RO clocks and increases the variety in RO oscillations from cycle to cycle. Also, because the PDLs are incorporated into the oscillator rings for every sample clock, increasing unpredictability, the output of the ROs are not correlated with one another. Furthermore, the suggested TRNG produces bitsteam that has

improved statistical features with the employment of a Von Neumann corrector as a post-processor. The Von Neumann corrector and the basic TRNG are implemented on the same Xilinx Spartan-3E FPGAs (XC3S400A4FTG256). This work's principal contributions are: A proposal for an FPGA-based TRNG that generates randomness by use of PDL-induced random jitter in the clocks of free-running ROs. The Von Neumann corrector's efficacy as a post-processor for bias removal is demonstrated. The suggested TRNG's experimental validation and proof that it passes every test in the NIST suite. High throughput-per-area and high entropy rate (i.e., real randomness) of the generated output bits are shownby the hardware evaluation findings.

II. PRELIMINARIES A. STRUCTURE OF THE XILINXSPARTAN-3E FPGA

The Spartan-3E FPGA from Xilinx is used for the proposed TRNG's prototyping. It can be seen of as a grid of interconnectedConfigurable Logic Blocks (CLBs) separated into four logical components known as slices. It has two pairs of CLB slices, SLICEL and SLICEM, each of which has two flip-flops (FFs) and two 4-input Look-Up Tables (LUTs). In summary, SLICEMsupports memory functions (such as RAM16 and SRL16 shift registers) in addition to logic. SLICEL, on the other hand, is themost basic sort of slice and just supports logic. As stated in the Spartan-3 Libraries Guide for HDL Designs, the LUT-based primitives are instantited in hardware description language (HDL) as described in Spartan3 Libraries Guide for HDL Designs.

B. DELAY LINES WITH PROGRAMMING (PDLS)

Changes in the propagation delays of the LUT under various inputs can be used to generate the internal variations of FPGA Look-Up Tables (LUTs) [6]. To illustrate, the LUT in

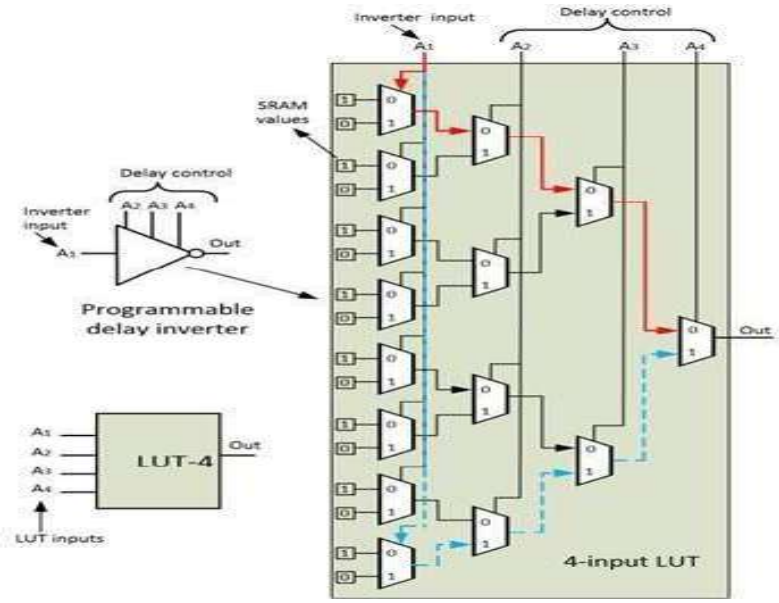


Fig1: PDL using a 4-input LUT.

The inverter shown in Fig. 1 is designed to have a LUT output

(0) that is always the opposite of its initial input (A1). The values of other inputs A2, A3, and A4 influence the signal propagation path from A1 to the output (0) even if they function as "don't-care" bits. In this case, Fig. 1 illustrates that, for 4- input LUTs, the signal propagation path from A1 to the output

(0) is longest for A2A3A4 = 111 (shown with dotted blue lines) and shortest for A2A3E4 = 000 (marked with solid red line). Thus, a three-controller programmable delay inverter One LUT can be used to implement inputs. The inverter input (A1) is thefirst LUT input for the PDL, while the next three LUT inputs are controlled by 23 = 8 discrete levels.

III. TRNG BASED ON RING OSCILLATOR

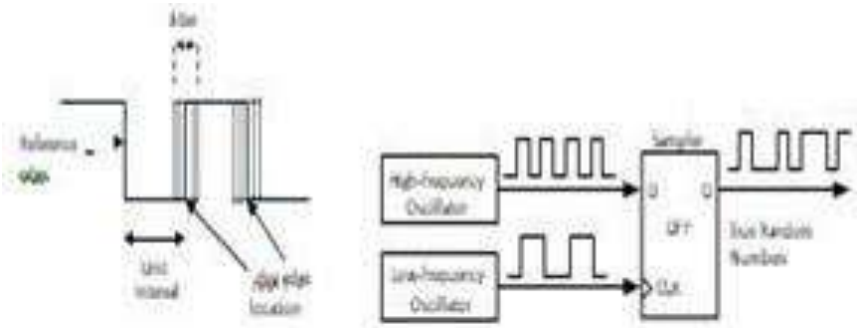
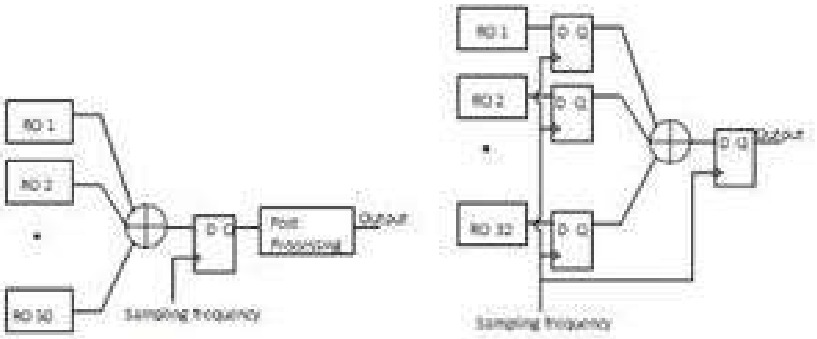


Fig2: Jitter in clock, Fig3: Basic oscillator betweensignals TRNG



i).Sunar type TNRG ii).Wold type TNRG

Fig 4: Block diagram of the original TNRG and the modified TNRG

Number of switching activities from the free-running ROs in such designs presents a significant challenge for the XOR-treeand the sample DFF [13]. It is because parallel ROs impose strict setup and holdtime constraints, which results in a significant number of transitions during a sample period. As illustrated in Fig. 4 [15], this feature can be partially addressed by adding a sample DFF at the output of each free-running RO. This design uses fewer ROs and passes the NIST statistical tests without the need for post-processing. Nonetheless, its mathematical complexity is comparable to that of the original design [12], so 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 TRNG ARCHITECTURE THATIS PROPOSED

It is crucial to remember that, even if they are fascinating, RO- based TRNGs have very little unpredictability when the same ROs are used [14]. Because identical delays cause equal length oscillator rings designed in FPGA to be highly correlated with one another, the output of these rings' XOR operation generally yields zeros. This causes the design's unpredictability to be subpar. In this study, we demonstrate that this issue may be resolved by using a TRNG that incorporates PDLs. Previously, truly random numbers were produced using the metastability of flip-flops [6]. By employing PDLs that precisely equalise the signal arrival timings to flipflops, they were able to achieve metastability. In contrast, our technique generates truly random numbers using the unpredictable jitter of freerunning oscillators. We use PDLs in oscillator rings to produce a wide range of oscillations and jitter in the clocks of the generated ROs. The primary benefit of the suggested TRNG that makes use of PDLs is the decrease in correlation between several oscillator rings of identical length. This can be accomplished, for instance, by adding PDL and varying the RO outputs for each sample clock, as seen in Fig. 5. Furthermore, because of the inverter delay, every oscillator ring also contributes to the change in RO oscillations from cycle to cycle (CTC). Consequently, the randomization features are greatly enhanced by the XOR procedure.

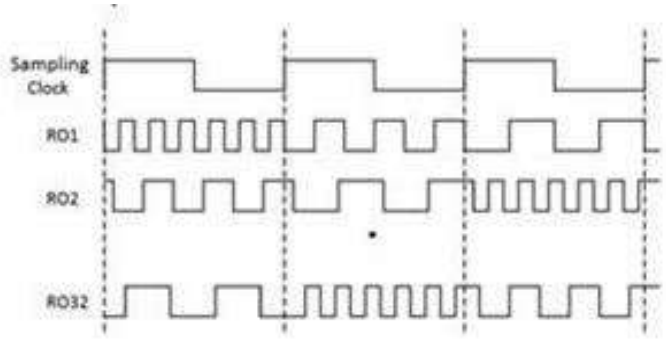


Fig 5: RO Outputs for each sampling clocks by using PDL

This section describes how the suggested TRNG and 4 postprocessing module were implemented on a cheap Xilinx

: Spartan -3E FPGA. The suggested method is then experimentally evaluated using the FT2232D USB interface, The utilisation of numerous free-running ROs can enhance the PyUSB-1.6 software, and Xilinx ISE design suite 13.6. quality of real random bits produced [12]. This is accomplished A. Overview of the Design by feeding the outputs into an XOR tree (a multi-input XOR), Fig. 6 depicts the proposed TRNG architecture. Here, three as illustrated in Figure 4, and then sampling the XOR tree using inverters and one AND gate—designated by black dashed a DFF and a reference clock at a predetermined frequency to boxes—are used to realise each RO.

Enabling the produce a random bit stream. However, handling a large corresponding ROs is the function of the and gates. Three and one LUT on the FPGA are needed, respectively, for the AND gate and inverter designs. One of the four LUT inputs is the ring connection, while the other three inputs are set up with 23 = 8 discrete levels to create programmable delays inside the LUT. If 6-input LUTs—which are frequently seen in high-end FPGAs—are used, For a ring connection, one can use one input, and for the remaining LUT inputs, one can allow 25 = 32 delay configurations. This permits configuring32 ROs with no restrictions on where the inverters must be placed.

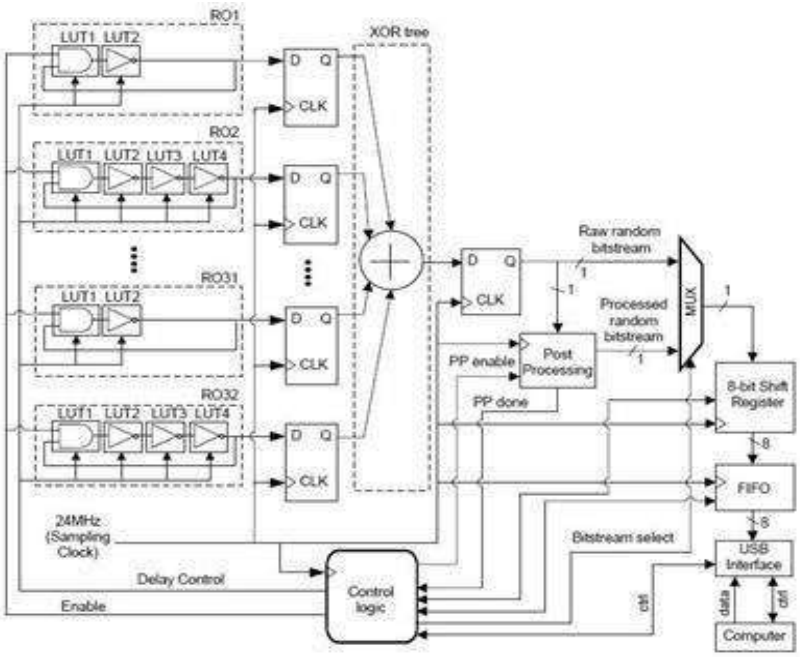


Fig 6: Architecture of the Proposed TRNG

32 ROs, an XOR tree, DFFs, a shift register, FIFO (FirstIn, First-Out), and a postprocessing unit make up the suggested design. First, the control circuitry uses the "enable" input to simultaneously turn on the 32 ROs. After that, the XOR tree combines the RO outputs, and a 24-MHz frequency clock is used to sample the data. A frequency divider might also be required if a greater operational frequency is being used for sampling. Next, 23 discrete levels are applied arbitrarily to the delay control inputs at every sample clock in order to generate PDLs. The sampled bits are then either transmitted straight to the FIFO without postprocessing, or fed to the post-processor unit As a result, the output is either a processed random bitstream chosen by the multiplexer's control input or a raw random bitstream gathered in blocks ofeight bits using the shift register's eight bits. In order to do the TRNG statistical analysis, each byte is then delivered to a PC over a USB port and stored in a FIFO with a 64-byte width, or 512 bits. Without disrupting flow, the FIFO enables reading of raw or processed random bitstream. The control logic module permits the selection of the raw/processed random bitstream for transfer to the PC, as well as the start and stop of the ROs, FIFO, 8-bit shift register, and post- processing unit.

B. Post Processing

A straightforward post processing unit based on a Von Neumann corrector [11] is used in the suggested implementation to increase entropy and eliminate bias from the random bits that are produced. Additionally, the postprocessor ensures that the TRNG output sequence is resilient. Fig. 7 shows the Von Neumann corrector post-processing approach that is being used.We read the raw TRNG output two bits at a time, discarding the first twoif they are identical (i.e. we delete 00 and 11 patterns). We take the first bit and throw away the second if the two bits are different (that is, 01 or 10). Generally speaking, For the post-processing unit to provide 128 bits of post-processed output, 512 bits of rawinput are needed.

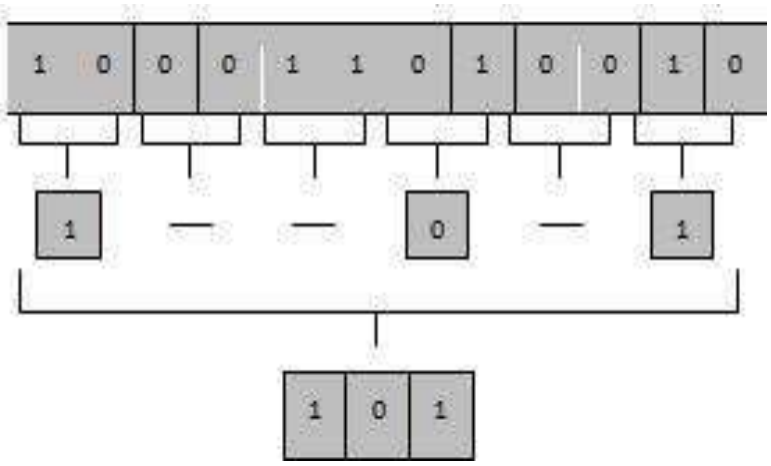


Fig7: Principle of the operation of the von Neumann V. Examining and Talking about A. Analysis of hardware overhead

Table I compares the performance of the proposed TRNG architecture on Xilinx Spartan-3E FPGAs with recently published TRNGs on Xilinx FPGAs. It is clear that in terms ofarea, the suggested design performs better than the previous design [7]. The design in [7] employs an operating frequency (O.F.) of 100 MHz, but ours uses 24 MHz, resulting in a reduced throughput. Furthermore, our approach has a better throughput but takes up more space than earlier works [6, 16, 19, 19]. By using more sophisticated FPGA setups, the designs [6,], [16], 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	379	157	–	8	24	Spartan-3E
RO-based [18]	528	177	270	6	24	Spartan-3E (XC3S400A)
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 (XC4VFX 20)
Meta stability [6]	128	–	–	2	–	Virtex-5 (XC3S500E)

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

B. Correlation Test

We calculated the sample Pearsons correlation coefficient [14] between all combinations (i.e.,) of the 32 RO's outputs in order to look at correlation between the rings employed in the proposed TRNG. For the test, we extracted 50,000 consecutively sampled bits from every RO. All of these rings' outputs were sampled using eight discrete levels and an external sampling frequency of 24 MHz,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 AIS31 (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.

A. 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 testedusing the NIST statistical test suite.

i). Entropy Estimation:

55◦C and 75◦C, respectively, following post-processing. In summary, it is evident from the results that the proposed TRNGexhibits better randomness (*P*-value is greater than 0.01 with greater proportion) properties at low hardware overhead and high

For an ideal true random number generator, the proportion of throughput.

"0"s and "1"s is ideally 0.5, meaning that the predicted entropy per bit is 1. In this work, test T8 of the procedure B of the AIS- 31 [4], which is accessible to test raw random numbers, is used to assess the entropy rate for the generated numbers. Test T8 divides a given sequence of length N into Q + K disjoint Lbit words. For the test T8, the recommended values are L = 8, Q = 2560, and K = 256000. For this test, a minimum sequence length of 7200000 bits is needed. For the test, we took our implementation and created 80 million random bits. The bit sequence met the test's requirements and attained an entropy of 0.9993 bits per bit, or 7.9946 bits per byte. This is superior to the 7.976 per byte (or 0.997 per bit) minimum entropy criterion. Consequently, the suggested TRNG satisfies procedure B of the AIS-31 requirements and is suitable for use in high-security applications. Furthermore, our suggested RObased TRNG achieves better entropy per bit in comparison to previous designs

[16] and [19] (0.997 per bit and 0.999 per bit, respectively).

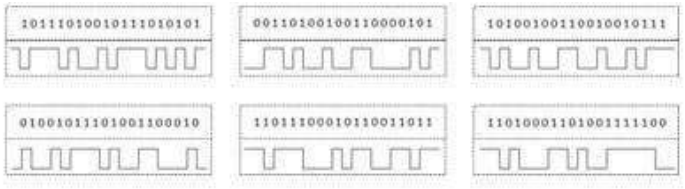


Fig. 8: 6 output bitstreams captured after restarting the TRNG.

2) Restart Experiment:
The suggested design's six restarts from identical beginning points were tested to confirm the startup procedures. Following each restart, the first 20 sampled bits were examined and plotted, as seen in Fig. 8. The restart experiment yields similar graphs for a pseudorandom signal when the TRNG is launched from the same starting points multiple times. But in our situation, these were shown to vary every time, ruling out any possibility of pseudorandomness. Similar techniques were employed in works [9], [15], and [13] to determine the proportion of genuine randomness in a pseudorandom oscillating signal **3) Statistical Analysis of the Output**: NIST has standardized a series of randomness tests to assess bitstreams' level of randomness [3]. The most extensive set of publicly accessible tools is the NIST statistical test suite. To put it briefly, 15 different types of tests are commonly used to evaluate how well TRNGs work. Each test's parameters were chosen in this work in accordance with NIST guidelines [3]. Since 99% confidence interval is indicated by the significance level α, 0.01 was selected as the default value. Before and after postprocessing, 1000 times (sequences) of 106 bits were obtained in order to look at the distribution of the P-values (randomness me in Table II.P-values greater than 0.01 indicate that the sequences have a roughly uniform distribution.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].

In the Proportion (Prop.) column, the count of P-values exceeding the 0.01 confidence interval is displayed. Prior to post-processing, the tests had a minimum proportion (minimum pass rate) of 0.982, while following postprocessing, it was 0.987. Our design's minimal sequence passrate (0.987) is greater than that of [9] (i.e., 0.97) and [7] (i.e., 0.986). Next, using a temperature-controlled laboratory, the TRNG is tested at two different temperatures—55°C and 75°C—while maintaining a core supply voltage of 1.2 V. Prior to post-processing, the tests' minimum proportion at each of these temperatures is 0.981. The minimum proportion, however, changes to 0.983 and 0.982 at

VI. SUMMARY

This study presents a new architecture of RO-based TRNG and details its implementation on Xilinx Spartan-3E FPGA. FPGA LUTs' programmable delay has been utilised to increase the unpredictability and produce random jitter. It has been shown that the suggested implementation offers an excellent trade-off between area and throughput. With a small hardware footprint, it can effectively generate a throughput of 8 Mbps after post-processing. Furthermore, the retry tests demonstrate that the suggested TRNG's output exhibits genuinely random behaviour. In fact, it may be assumed that the suggested approach is a strong contender for applications.

VII. FUTURE WORK

The future work for the topic "Low Power and Area Efficient FPGA-Based True Random Number Generation based on LUT Variant Ring Oscillators" could involve several directions for further research and development. Here are some potential avenues:

- **Performance Optimization**: Further optimization of the design for improved performance metrics such as speed, randomness, and energy efficiency. This could involve refining the architecture, exploring alternative ring oscillator configurations, or optimizing the synthesis and placement algorithms for FPGA implementation.
- **Security Analysis**: Conduct a comprehensive security analysis to evaluate the robustness of the generated

random numbers against various attacks, including statistical tests, cryptographic analyses, and sidechannel attacks. This analysis could identifypotential vulnerabilities and guide further enhancements to the design.

POWER CONSUMPTION:

Parameter	Proposed	Existing
Power(mV)	113	143

REFERENCES

[1] K. Nohl,,¨ "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 MetastabilityBased True Random Number Generator," *IEICE Transactions on Information and Systems*, vol. E95.D, no. 2, pp. 426– 436, 2012.