Design and Implementation of Programmable Delays in Oscillator-Rings for FPGA-Based True Random Number Generation on FPGA

Dr.A.Ranganayakulu [(1)] Mr.D.Satyanarayana [(2)] Kolla Srivalli [(3)] Chakka Sai Harshitha [(4)] Mukkamalla Venkata Bhargavi [(5)]

[1,2] *Krishna Chaithanya Institute Of Technology & Sciences, Ece Department, Markapur, Andhra Pradesh.*

[3,4 5,] *Krishna Chaithanya Institute Of Technology & Sciences, UG Student-ECE, Markapur, Andhra Pradesh.*

Abstract In this project, the use of true random number generators in cryptography systems is essential. This study uses the random jitter of free running oscillators as a source of randomness and proposes a novel and effective approach to create real random numbers on field programmable gate array. Programmable delay lines are incorporated into the free-running oscillator rings to provide a wide range of oscillations and jitter in the ring oscillators clocks. 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. 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-3A FPGAs. The suggested true random number generator uses 528 slices, attains a throughput of 6 Mbps with an entropy rate of 0.999 per bit, and passes every statistical test conducted by the National Institute of Standards and Technology (NIST).

Keywords: TRNG(True Random Number Generator),Communications, FPGA, Verilog HDL, ATPG,BIST,M-Bist.

I.INTRODUCTION

Introduction to True Random Number Generators (TRNG):In the realm of computing and cryptography, the generation of random numbers plays a pivotal role in numerous applications, ranging from secure communications to simulation and gaming. True Random Number Generators (TRNGs) represent a class of devices or algorithms designed to produce random numbers that possess certain essential properties of randomness, such as unpredictability and uniform distribution.

Unlike Pseudo-Random Number Generators (PRNGs), which generate sequences of numbers deterministically based on initial seed values, TRNGs harness physical processes or phenomena to generate randomness inherently. This reliance on physical sources distinguishes TRNGs by providing a higher degree of randomness, often referred to as "true randomness."

TRNGs exploit various physical phenomena to generate random bits, including:

Thermal Noise: Thermal noise, also known as Johnson-Nyquist noise, arises from the random motion of electrons within a conductor at non-zero temperatures. TRNGs can sample this noise to extract random bits, as the fluctuations are unpredictable and exhibit a uniform distribution over time.

Radioactive Decay: The decay of radioactive isotopes occurs randomly and unpredictably over time. TRNGs can utilize the timing of radioactive decay events as a source of randomness, ensuring that the generated numbers are truly random.

Electronic Component Variability: Variations in electronic components, such as resistor values or semiconductor properties, can introduce randomness into electrical signals. TRNGs exploit these variations to generate random numbers by measuring analog signals or digital circuit characteristics.

Photon Arrival Times: The arrival times of photons in a photodetector can be inherently random, especially in low-intensity light conditions. TRNGs can sample these arrival times to generate random bits with high entropy.

Chaos Theory: Chaotic systems exhibit sensitive dependence on initial conditions, leading to seemingly random behavior over time. TRNGs can exploit chaotic systems, such as chaotic oscillators or chaotic maps, to generate random sequences.

The output of a TRNG typically undergoes post-processing techniques, such as whitening or conditioning, to enhance the statistical properties of the generated random numbers. Additionally, TRNGs are often subjected to rigorous testing and certification processes to ensure compliance with recognized standards for randomness, such as the National Institute of Standards and Technology (NIST) Special Publication 800-22.

TRNGs find applications in various domains where high-quality randomness is crucial, including cryptographic key generation, secure communications, numerical simulations, and gambling industries. Their ability to provide true randomness is indispensable for ensuring the security and integrity of sensitive data and systems in today's digital age.

True Random Number Generation (TRNG) with Ring Oscillators

Ring oscillators are widely used in electronic circuits for generating clock signals or oscillating waveforms. However, they can also be leveraged to create true random number generators (TRNGs) due to the inherent noise and instability present in their operation. Here's how TRNGs can be constructed using ring oscillators:

Basic Principle:

A ring oscillator consists of an odd number of inverting stages connected in a loop. The delay through each stage causes the output signal to oscillate.The frequency of oscillation is determined by the propagation delay of each stage, which can be affected by various factors such as temperature, voltage, and process variations.

Noise and Instability:

Due to manufacturing variations and environmental factors, the propagation delay of each stage in the ring oscillator fluctuates randomly.These fluctuations lead to variations in the oscillation frequency and phase, resulting in noise and instability in the output signal.

Randomness Extraction:

The variations in the oscillation frequency can be sampled and digitized to generate random bits.By measuring the time intervals between oscillation cycles or comparing the frequencies of multiple ring oscillators, random bits can be extracted from the inherent noise and instability.

Post-Processing:

The raw random bits generated by the ring oscillator TRNG may undergo post-processing techniques to enhance their statistical properties and remove biases.

Techniques such as whitening, entropy estimation, and conditioning algorithms may be applied to ensure that the output random numbers exhibit uniform distribution and high entropy.

Testing and Validation:

The performance of the ring oscillator TRNG should be thoroughly tested and validated to ensure randomness and security.

Standardized statistical tests, such as the NIST Statistical Test Suite, may be applied to assess the quality of the generated random numbers and verify compliance with recognized randomness standards.

Integration and Application:

The ring oscillator TRNG can be integrated into electronic devices and systems where secure and high-quality random number generation is required.Applications include cryptographic key generation, secure communication protocols, random number seeding for simulations, and gaming applications.

Considerations:

Careful design considerations must be taken into account to mitigate potential vulnerabilities and sources of bias in the TRNG. Environmental factors, such as temperature variations and electromagnetic interference, may impact the performance and reliability of the ring oscillator TRNG and should be carefully managed.

In summary, ring oscillators offer a simple yet effective approach to generating true random numbers by exploiting the inherent noise and instability in their operation. When properly designed and implemented, ring oscillator TRNGs can provide a reliable source of randomness for a wide range of applications in security, cryptography, and data integrity.

2.LITERATURE

FOUNDATIONS on which all secure operations of an integrated circuit (IC) depend is typically defined as a hardware root of trust [55]. High-end roots of trust are usually integrated into silicon as separate, custom-designed security modules—immune from malware attacks—that handle chip and device identities, cryptographic keys and functions, secure boot processes, attestation, authentication, firmware updates,etc. As a security vehicle, the hardware root of trust must be capable of detecting an intrusion, disabling access pending further actions, and/or obfuscating (camouflaging) logic operations of the IC. What lays foundation for a silicon-based fixed-function root of trust is its authentication protocol. It performs a specific set of functions, such as true random number generation, data hashing and encryption, keys validation, and logic locking [59]. As an initial part of the actual challenge-response procedure, an IC creates a random token, commonly known as a challenge or a nonce, and sends it to a secure server. The nonce is typically produced by an on-chip true random number generator (TRNG) that should yield different combinations of 0 and 1 s every time it is activated. It may also contain some individual data from the IC such as its electronic identification number.

In light of the above it is clear that TRNGs became key hardware security primitives capable of producing random sequences by harvesting the randomness present in physical processes, such as the thermal instability and noise [7], [8], [29], [39], [46], metastability [20], [25], [31], [37], [60], [61], [65], edge racing in digital designs [70], chaotic behavior of cellular automata [21], [27], [34], [45], power supply variations [58], stochastic nature of magnetic tunnel junction [64], quantum effects [56], or phase jitters in ring oscillators (ROs) [54]. The latter approach has gained noticeable popularity because it provides a simple yet effective method to build random number generators just by chaining an odd number of inverters into a ring structure. As a result, a wide range of solutions using this principle and its derivatives have been proposed in the contemporary technical literature and industrial practice. For the sake of illustration, let us recall a few exemplary solutions.

The most straightforward mechanism to extract randomness from a jitter is to sample the output of an RO using the output signal of another RO [Fig. 1(a)]. Such coupled oscillators are presented in [2], [8], [18], [51], and [68]. In [18], two ROs are coupled by a nonlinear circuit, whereas in [68] the first RO feeds a programmable delay chain that is sampled by a bit extractor driven by the other RO. If periods of both signals are very close to each other, they form a basis for coherent sampling [32], [44], [69]. Alternatively, one can combine the output signals of several ROs by means of XOR trees [Fig. 1(b)], as shown in [3], [33], [53], [57], [66], or [67]. In particular, the work of [56] provides a thorough mathematical treatment of an approach where combined jitter signals form a source of entropy. A low power scheme where two identical ROs enable, through an XOR gate, a third RO clocking a counter is described in [10]. A reconfigurable TRNG based on transient effect ROs with two different sampling methods is introduced in [1]. In [14], four ROs drive associated LFRSs whose outputs are sampled through a multiplexer driven by yet another RO. ROs with a multistage feedback structure can be XOR-ed to produce true random numbers, as detailed in [15]. Somehow different approach is discussed in [17], [19], [23], and [42]; it implements an RO by replacing a simple circular feedback with a more complex network comprising XOR gates in a way corresponding to conventional Fibonacci or Galois LFSRs. Here, inverters replace memory elements. To enhance the performance of RO-

based TRNG, one can also deploy Muller C-gates instead of inverters. These elements are then interleaved to form an asynchronous pipeline that is capable of propagating several simultaneous voltage events sampled by an XOR tree [12], [13]. Finally, different RO-based TRNGs are compared in [47] to demonstrate how they are amenable to FPGA-based implementations. We also refer the interested readers to other relevant papers, such as [6], [9], [11], [16], [36], [48], [60], and [63].

Besides the schemes recalled above, there are other techniques deployed to produce truly random sequences of bits. Those schemes include the use of chaotic maps [5] with the von Neumann correction algorithm [43], sampling a jitter in a phase-locked loop circuitry [22], extracting a design fingerprint during the power-up of SRAMs [26], or detecting a beat frequency in FPGAs [28].

In addition to its unpredictability, a modern TRNG design is expected to be compact, fast, self-testable, and easily synthesizable by using exclusively digital components [57], i.e., no amplifiers or other analog devices are allowed. Furthermore, additional post-processing steps and the corresponding circuitry to adjust the sampling frequency or to increase the per-bit entropy [53] should be avoided. Consequently, this article proposes a high-performance device that may assume the role of a lightweight all-digital TRNG. Although it was originally designed as a hardware generator of one-time challenges produced for the sake of IC authentication protocols [49], many tests have confirmed that it can be considered as a reliable source of truly random numbers used in a variety of cryptographic or security-related applications.

II.EXISTING SYSTEM AND PROPOSED SYSTEM

The proposed design rests on a ring generator architecture harvesting a source of entropy implemented by a conventional free running RO, and further processing the captured data due to its feedback network.

2.Existing System Block diagram

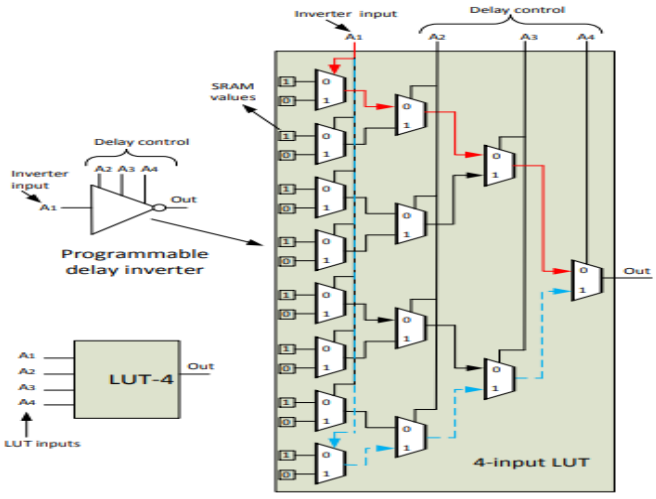


Fig. 1: PDL using a 4-input LUT.

Explanation

True Random Number Generators (TRNGs) are devices or algorithms that generate random numbers from unpredictable physical processes or sources. Unlike Pseudo-Random Number Generators (PRNGs), which are deterministic and repeatable given the same initial conditions, TRNGs aim to produce truly random and unpredictable sequences of numbers. Here are some key aspects of true random number generators:

Characteristics of True Random Number Generators:

Physical Processes:
TRNGs often rely on physical processes that are inherently unpredictable, such as electronic noise, radioactive decay, or atmospheric noise. The idea is to capture randomness from the physical world.

Unpredictability:
True randomness implies that the generated numbers are not influenced by any external factors or patterns. It is difficult, if not impossible, for an external observer to predict the next generated value.

Non-Deterministic:
Unlike PRNGs, TRNGs are non-deterministic, meaning that their output is not determined by a fixed algorithm or seed value. The unpredictability comes from the randomness inherent in the physical process.

Entropy Source:
The physical process that serves as the entropy source is crucial for the quality of randomness. The higher the entropy, the more unpredictable and secure the generated random numbers are.

Bias and Uniformity:
TRNGs strive to produce uniformly distributed random numbers without biases. Any biases or patterns in the generated sequences could be exploited by attackers.

Statistical Tests:
TRNGs are often subjected to rigorous statistical tests to ensure that their output meets the criteria for randomness. Common tests include the NIST (National Institute of Standards and Technology) SP800-22 test suite.
Examples of True Random Number Generator Sources:

Electronic Noise:
The thermal noise or electronic noise in electronic components, such as resistors, diodes, or transistors, can be used as a source of entropy.

Radioactive Decay:
The random decay of radioactive particles can serve as a source of entropy. Geiger-Muller counters or other radiation detectors can be used for this purpose.

Atmospheric Noise:
Atmospheric noise, captured using radio receivers or specialized antennas, provides an unpredictable source of entropy.

Photonic Processes:
Quantum-based TRNGs leverage the randomness inherent in quantum processes, such as the arrival times of single photons.
The TRNG should be designed with countermeasures against potential attacks, including those attempting to manipulate or bias the random number generation process.
Creating a Ring Oscillator-based True Random Number Generator (TRNG) involves constructing a ring oscillator circuit and extracting random bits from the fluctuations in its output. Here's a simplified example of how you might implement such a TRNG:

2.2 Ring Oscillator TRNG Circuit: Ring Oscillator Circuit:

Construct a ring oscillator using an odd number of inverting stages (typically inverters or buffers) connected in a loop.
Each stage introduces a delay, causing the signal to oscillate at a frequency determined by the cumulative delay around the loop.
The number of stages and the properties of the individual inverters/buffers can influence the oscillation frequency and randomness of the output.

Noise Injection:
Introduce sources of noise into the ring oscillator circuit to induce fluctuations in the oscillation frequency. This can be achieved by adding passive components like resistors or capacitors to the feedback path, which can exhibit random variations due to thermal noise or process variations.

Clock Signal Output:
Tap the output of the ring oscillator to obtain the clock signal waveform.
This waveform will exhibit variations in frequency and phase due to the inherent noise and instability in the oscillator circuit.

Sampling and Digitization:
Sample the clock signal at regular intervals to capture the variations in its frequency.
Use a comparator circuit or a threshold detector to convert the analog waveform into digital signals. The timing of the transitions in the digital signal can be considered as random bits.

Post-Processing:
Apply post-processing techniques to the raw digital output to enhance its randomness and remove biases.
Techniques such as entropy estimation, whitening, and conditioning algorithms can be employed to improve the quality of the generated random numbers.

Testing and Validation:
Perform statistical tests and validation procedures to assess the randomness and quality of the generated random numbers.

Ensure that the TRNG meets recognized standards for randomness and security.

Considerations:
Oscillator Stability: Ensure that the oscillator operates within a stable range to maintain reliable oscillation while still exhibiting enough noise for randomness.
Noise Sources: Experiment with different noise sources and circuit configurations to optimize randomness while minimizing bias and non-uniformity in the generated random numbers.
Environmental Factors: Consider environmental factors such as temperature, voltage fluctuations, and electromagnetic interference that may affect the performance and reliability of the TRNG.

Security Considerations: Implement appropriate measures to protect the TRNG against potential attacks and vulnerabilities, such as physical tampering or side-channel attacks.
By carefully designing and implementing a ring oscillator TRNG with appropriate noise sources and post-processing techniques, you can create a reliable and secure source of true random numbers for various applications in cryptography, security, and data integrity.

PROPOSED SYSTEM BLOCK DIAGRAM AND METHODOLOGY

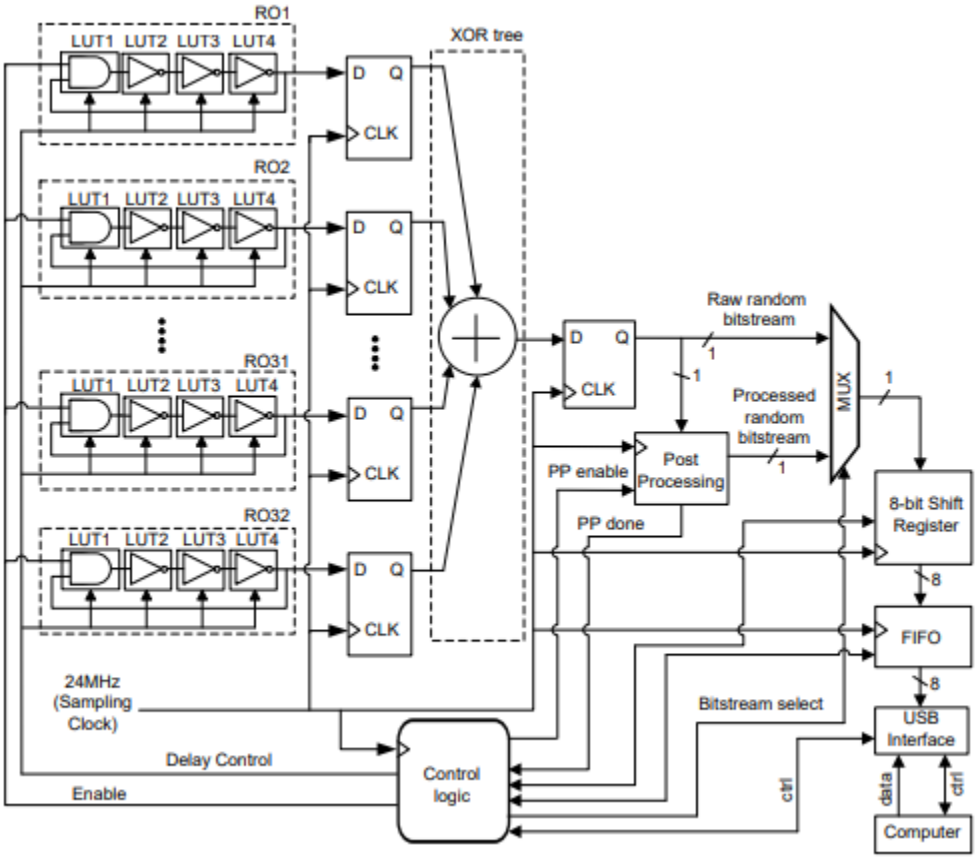


Fig3. Architecture of the proposed TRNG.

The proposed architecture consists of 32 RO's, XOR tree,DFF's, shift register, FIFO (First-In, First-Out), and a postprocessing 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 power 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.

We employ the PDL's (Programmable Delays) 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 PDLs. 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.

The implementation of the proposed TRNG along with the post-processing module on a low-cost Xilinx Ultra Zynq Boards FPGAs Subsequently, Xilinx ISE design suite 2018.1 USB interface is used for experimental evaluation of the proposed technique.

Fig. 2 illustrates the proposed scheme. Its major part is a ring generator [40], [41], i.e., a linear finite state machine obtained by forming a circular shift register, and then by adding gradually feedback connections, which correspond to successive terms of a characteristic polynomial. Given tap xk, the corresponding feedback loop encompasses k adjacent flipflops (FFs), always beginning with the leftmost ones, as shown in Fig. 2 for the polynomial $h(x) = x32 + x27 + x21 + x16 + x10 + x5 + 1$. Two feedback lines cannot cross each other. Instead, they form a very regular rack frame provided a suitable characteristic polynomial is deployed [50]. Since a subset of k adjacent FFs can be chosen in different ways as long as the resultant feedback line does not cross any other feedback line, the ring generator, in a vivid contrast to the corresponding Fibonacci or Galois LFSRs, offers a certain degree of flexibility in forming its structure. Every resultant device will feature a different state trajectory; all of them, however, will remain maximum-length finite state machines producing the same m-sequence, just differently phase shifted in each case [40]. As can be seen, every ring generator is a planar, high-speed circuit that features reduced internal fan-outs and minimal delays on critical paths. As a result, it causes no frequency degradation and lets designers minimize routing complexity, optimize wire sizing, and make the overall layout as compact as possible [41]. Clearly, a synchronous ring generator has a deterministic behavior which renders the generated values as predictable as any other LFSR-produced pseudorandom sequences (vulnerable to statistical attacks). To make the ring generator suitable for TRNG-based applications, it is enhanced by adding an entropy source implemented as a free-running m-stage gated RO, as shown in Fig. 2, where the oscillator is made of a single NAND gate and four inverters chained into a ring. The oscillator internal signals, sampled at the outputs of selected inverters, are subsequently injected into the ring generator via XOR gates placed in the front of its "upper level" FFs. RO operates with a frequency that depends on the IC fabrication process, the number of logic elements it deploys, and the delay of its routing paths.

The purpose of sampling many inverters is to populate a relatively long interval with the timing jitter, hence, maximizing the probability that at least one noisy signal edge is captured in the ring generator. Consequently, the ring generator acts as a special form of a bit extractor processing data collected at several RO stages.

III. RESULTS AND ANALYSIS DISCUSSION

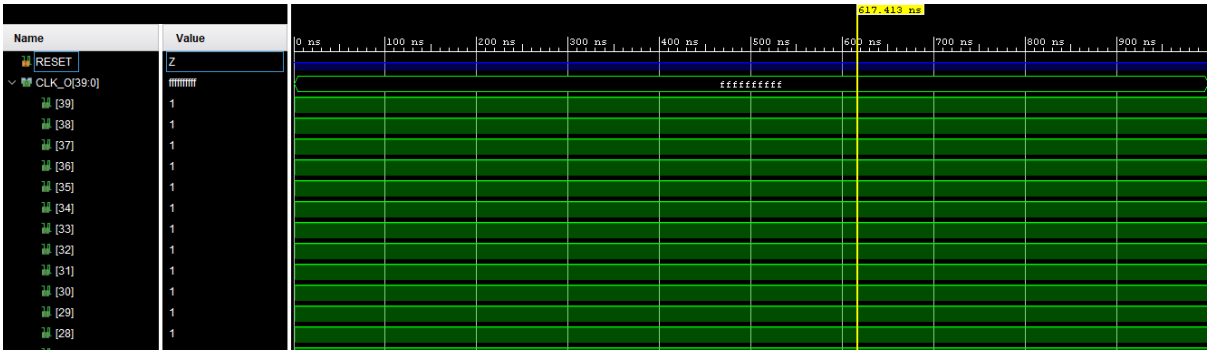


Fig6.1 TRNG with RO simulation output

Fig.6.2.RTL schematic diagram1

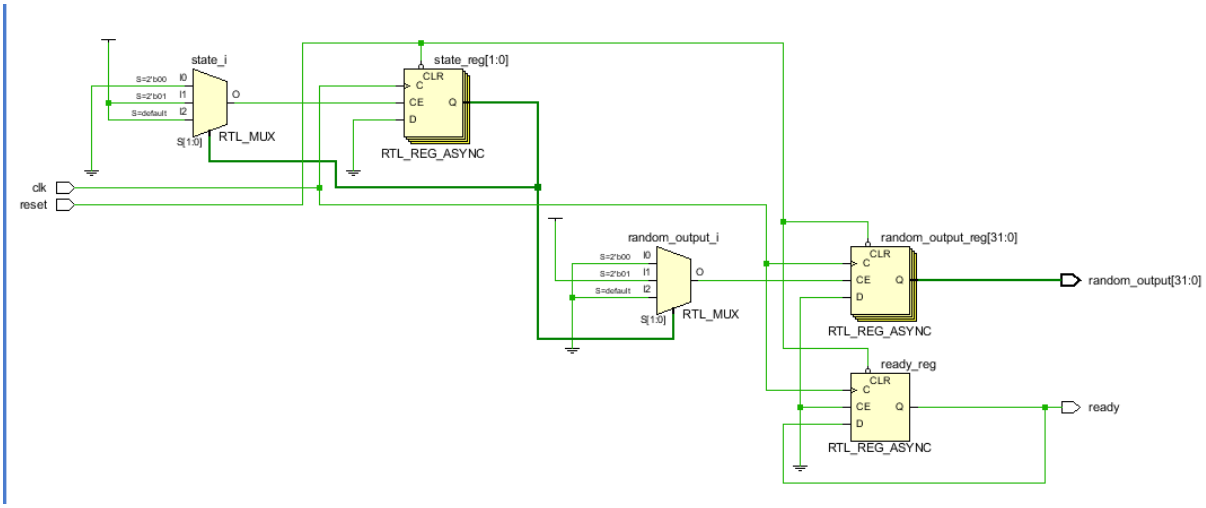


Fig.6.2.RTL schematic diagram2

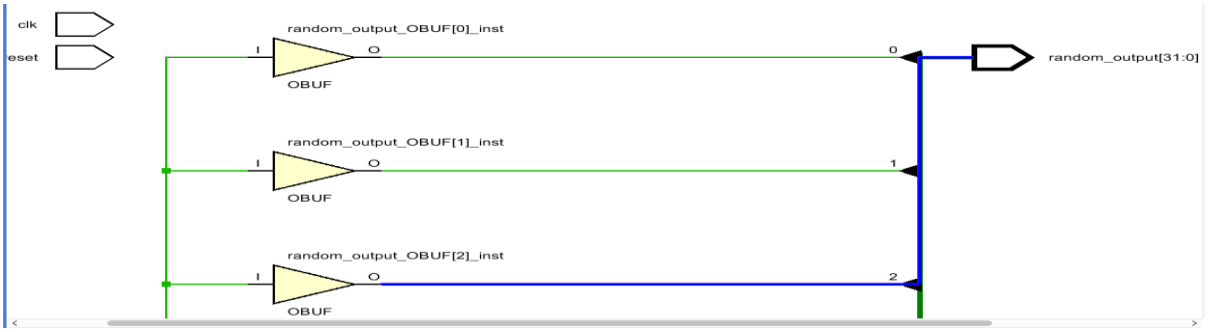


Fig.6.3.TRNG with RO synthesis design

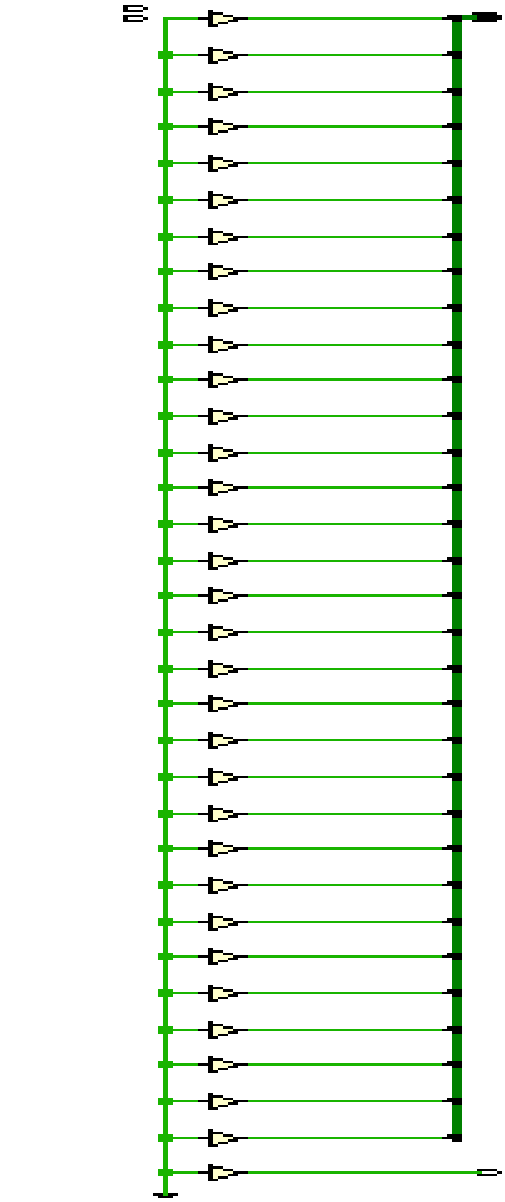
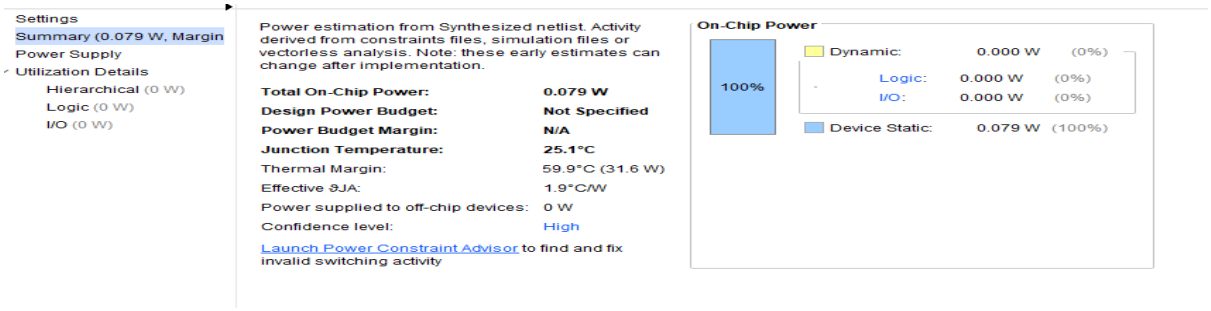


Fig 6.4.32 bit output synthesis design of TRNG.



Fig,6.5.Power report

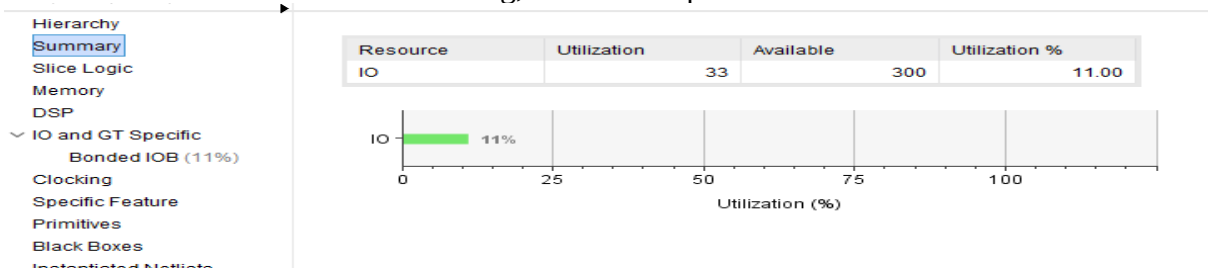


Fig 6.6. Area of LUTs Utilization

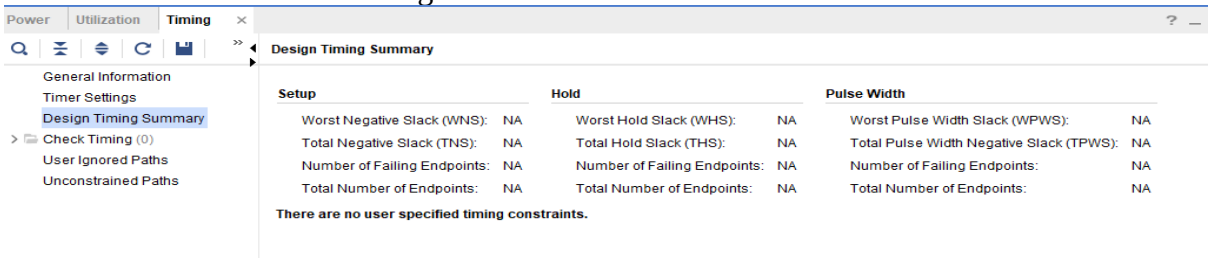


Fig 6.7. Timing Report.

Conclusion

This project introduces a new design of RO-based TRNG is described and its implementation on Xilinx vivado 2018,01 zynq ultra scale FPGA is used. 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 can produce 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 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. Plotz, “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. Golic, “High-Speed True Random Number Genera- ´ tion 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, “TrueRandomness 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.