ENGINEERING IN ADVANCED RESEARCH SCIENCE AND TECHNOLOGY

ISSN 2352-8648
Vol.03, Issue.01
September-2021
Pages: -217-228

POWER OPTIMIZED TRN GENERATOR USING RECONFIGURABLE OSCILLATORS AND CLOCK GATING

[1] YALAMANCHI DEEPA, [2]M.SRIHARI

[1] PG Student, Dept. of ECE, Kakinada Institute of Engineering and Technology for Women, KAKINADA, A.P

[2] Assistant professor, Dept. of ECE, Kakinada Institute of Engineering and Technology for Women, KAKINADA, A.P

ABSTRACT: The paper focuses on generating true random number sequences using hardware, so as to safeguard the keys patterns for digital communications. These sequences are generated using purely digital components supported by an efficient VLSI architecture. 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 protocol. This paper proposes a new and efficient method to generate true random numbers in XILINX by utilizing the random jitter of free running oscillators as a source of randomness. 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. The generated random sequences will further undergo some post processing operations, viz; Von-Neumann correction (VNC) In addition, a Von Neumann corrector as post-processor is employed to remove any bias in the output bit sequence. Clock gating architecture is to limit the switching activity of the address decoder which improves the power efficiency of the proposed number generator. Element structure is adapted to evaluate the clock cycle to the present ring counter block and to release the clock pulse to the next ring counter block. LUT memory accessing does not need write operations, so data lines and read/write lines to the SRAM memory architecture is omitted.

KEYWORDS: True random number generators, Von-Neumann correction, Linear Feedback Shift Register, Look Up table.

INTRODUCTION Computer systems and telecommunications play an important role in modern world technology. The communication and data transfer through computers touches almost every aspect of life, i.e. transferring data, tracking personal data, trading over the internet, online banking and sending emails. As more vital information is transferred through wire or wireless means, the need to safeguard all this data from hackers is growing. All these security concerns emphasize the importance of developing methods and technology for the transformation of data to hide its information content, prevent its modification, and prevent unauthorized use. Random number

generation is a fundamental process for protecting the privacy of electronic communications. It is a key component of the encryption process that protects information from hackers by making it unreadable without the proper decryption process. Since the strength of an encryption mechanism is directly related to the randomness of the binary numbers used in it, there has been an enormous need to design and develop an efficient random number generator that can produce true random numbers to implement a safe and secure cryptographic system. In addition to cyber security, random number generators (RNGs) are a vital ingredient in many other areas such as computer simulations, statistical sampling, and commercial applications like lottery games and slot machines. Random numbers are needed in some areas in computer science, such as authentication, secret key generation, game theory, and simulations. In these applications, particularly numbers should have good statistical properties and be unpredictable and non-reproducible. In modern cryptographic systems, security is based on the statistical quality and on the unpredictability of confidential keys. These keys are generated in random number generators (RNGs) using random physical phenomena that occur in the hardware devices in which the system is implemented. A widespread source of randomness in digital devices is the jitter of the clock signal generated inside the device using free running oscillators such as ring oscillators [SMS07, BLMT11, RYDV15], or self-timed rings [CFAF13]. The statistical quality and unpredictability of the generated numbers depend on the size and quality (e.g. the spectrum) of the clock jitter. It is therefore good practice to continuously monitor this jitter using an embedded jitter measurement method. As required in the document AIS-20/31 published by the German Federal Office for Information Security (German acronym BSI) [KS11], the measured jitter parameters should then be used as input parameters in the stochastic model used to estimate entropy, which characterizes the unpredictability of generated numbers.

LITERATURE REVIEW This section highlights the literature survey which has been done to review the critical points of the related works in recent days. In an irreversible circuit, if one bit information is lost then at least KTln2 joules of energy is dissipated. Where K is Boltzmann's constant and T is absolute temperature. This was stated by Landauer R in 1961. In 1973, Bennett proved that, KTln2 joules of energy dissipated due to information loss in irreversible circuit can be controlled by reversible logic where the reversible circuit allows to reproduce the inputs from output resulting in no information loss. He also showed that reversible systems can do the same computations as the classical or irreversible systems at same efficiency. This leads to the evolution of reversible logic based systems. Any reversible gate should have equal number of inputs and outputs such that, inputs can be recovered uniquely from outputs at any point of time. In paper [3], by Shibinu A.R , Rajkumar, a 4- bit LFSR design using Muller expression is proposed. This paper also gives realization of both edge triggered and level triggered D flip flop using reversible logic. At the end, comparative analysis has been given between conventional LFSR and Reversible LFSR. From this it is observed that, the proposed technique is efficient than conventional technique in implementing LFSR in terms of cost

metrics like power, quantum cost, garbage output and gate count. D. Muthih and A. Arockia Bazil Raj [4] have presented a parallel architecture for designing high speed LFSR and explained that, BCH encoders and CRC operations are normally carried out by using LFSR. A novel approach for high speed BCH encoder is proposed. This paper presents two key points. First, it presents a linear transformation algorithm for converting a serial LFSR into parallel architecture, which can be used for generating polynomials in CRC and BCH encoders. Secondly, a new approach is proposed to amend parallel LFSR into pipelining and retiming algorithm. In paper [5], authors have presented two design approaches for designing reversible D FF with asynchronous set/reset which are optimized in terms of quantum cost, delay and garbage outputs. It also includes the design of 3 bit LFSR using two design approaches. The application of these FF's as LFSR is designed and discussed. The application of LFSR as pseudo random bit sequence generator is proposed.

ARCHITECTURE OF TRUE RANDOM NUMBER GENERATOR:

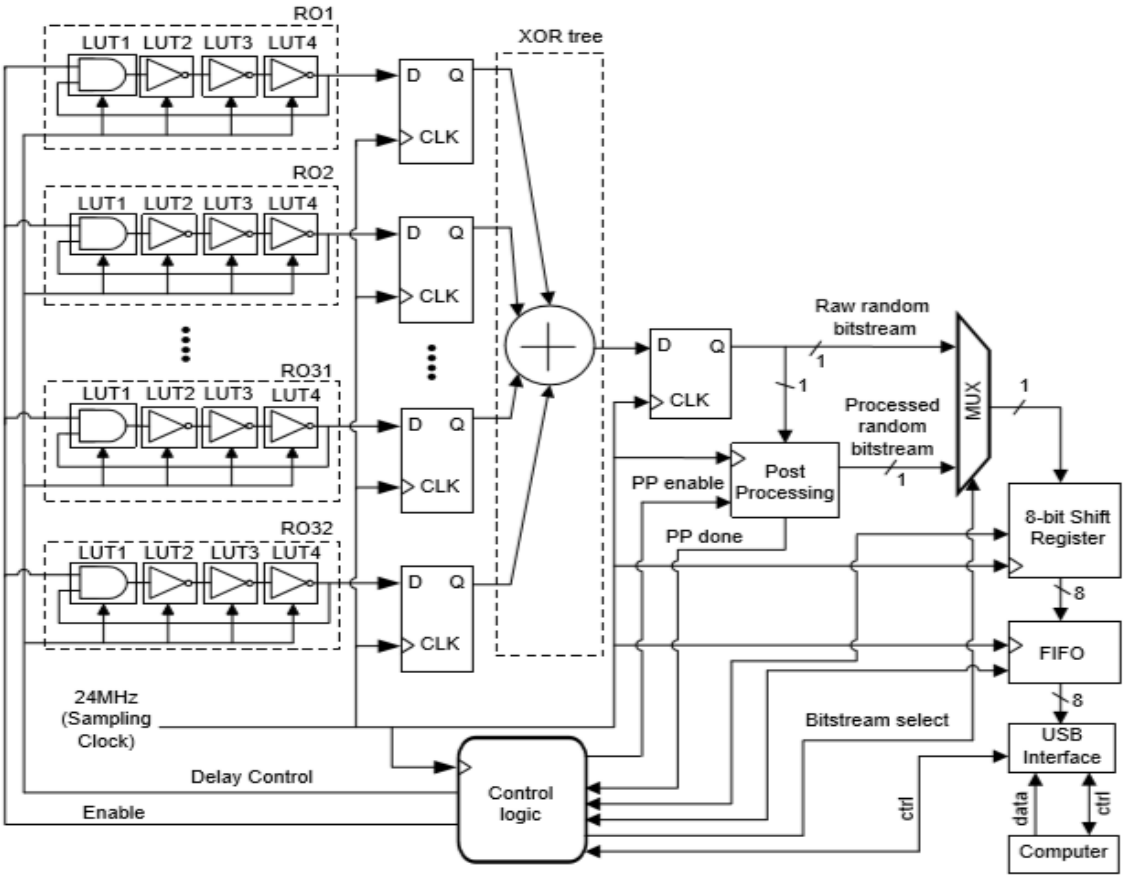

Fig1: Architecture of the TRNG.

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. 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 meta-stability of flip-flops was used for generating true random numbers. They achieved meta-stability 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 existing 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. 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.

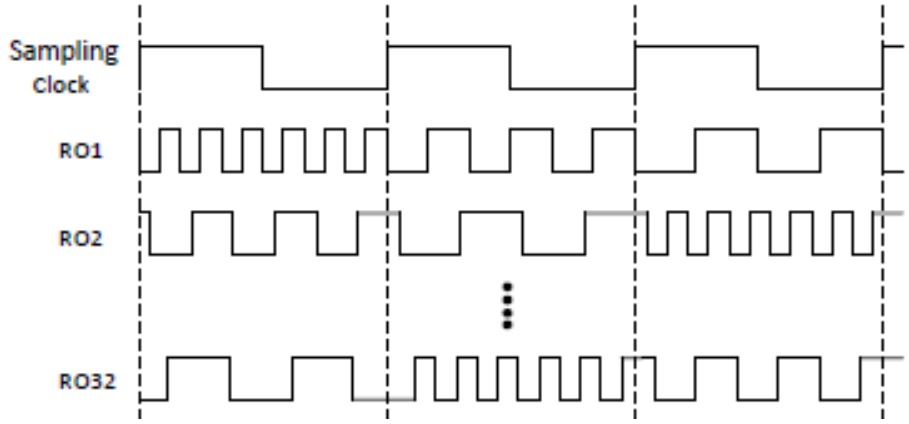

Fig2. RO outputs for each sampling clock by using PDL

The existing TRNG architecture is shown in Fig. 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 23 = 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 25 = 32 delay configurations for the remaining LUT inputs. This allows configuring 32 RO's without any constraints on the placement of the inverters. The existing 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, 23 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 bit stream or processed random bit stream selected via control input of the multiplexer and collected in blocks of 8 bits using the 8-bitshift register. Finally, each byte is stored in a FIFO of 64 by 8 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 bit stream 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 bit stream for transfer to the PC.

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

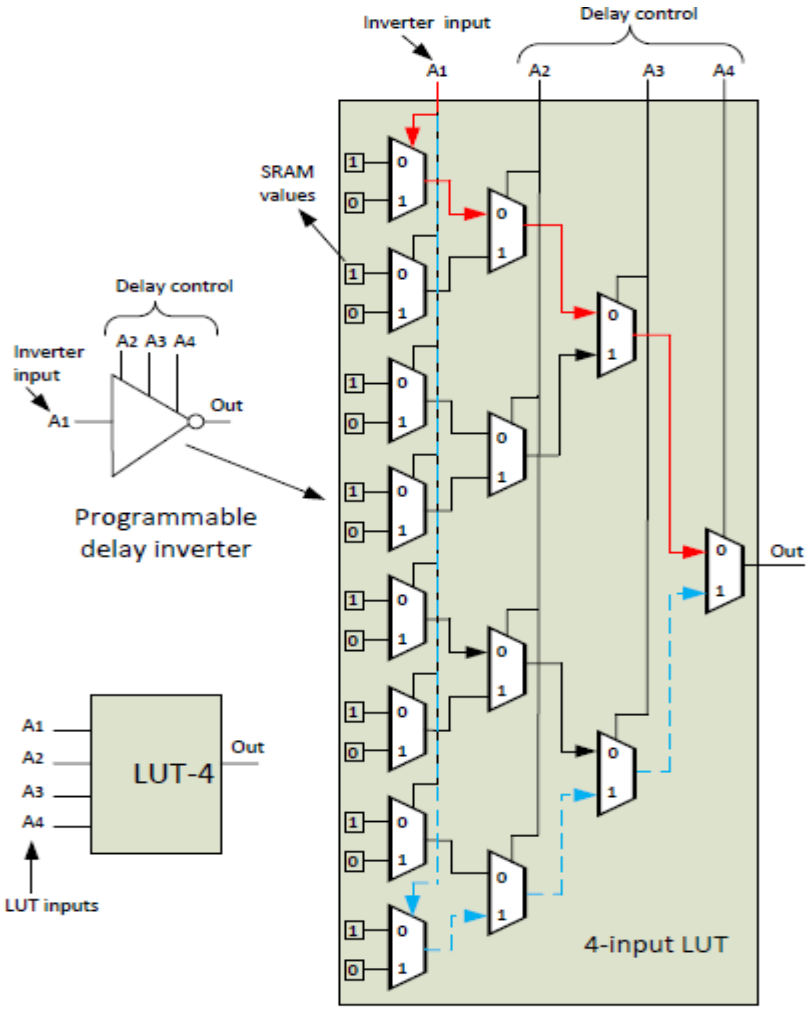

Fig3: PDL using a 4-input LUT

The above Fig is programmed to implement an inverter whose LUT output (0) is always an inversion of its first input (A1). Other inputs A2, A3, and A4 act as "don't-care" bits but their values affect the signal propagation path from A1 to the output (0). In this context, it has been shown in Fig,

that the signal propagation path from A1 to the output (0) is shortest for A2A3A4 = 000 (marked with solid red line) and longest for A2A3A4 = 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 A1 is the inverter input and the rest of the LUT inputs (three) are controlled by 23 = 8 discrete levels. The internal variations of FPGA Look-Up Tables (LUT's) can be generated from changes in the LUT's propagation delays under different inputs. For example, the LUT in

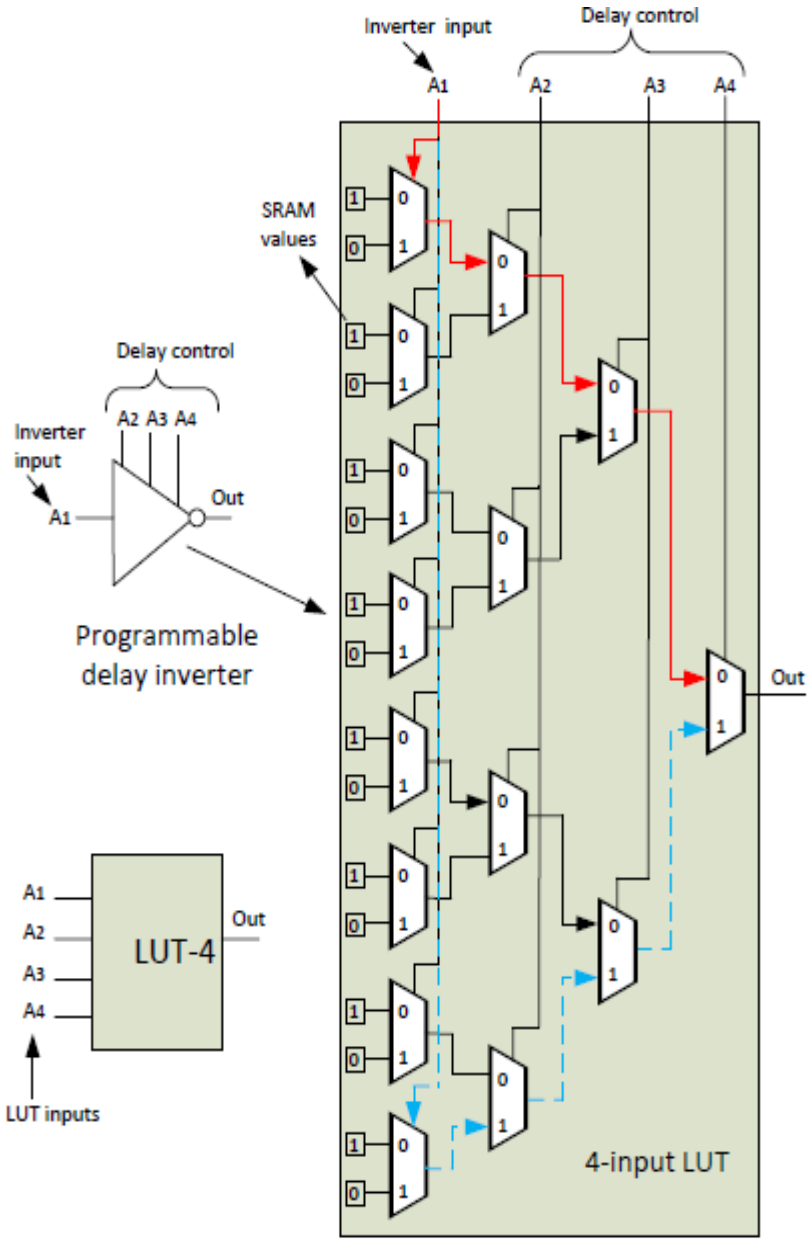

Fig4 : PDL using a 4-input LUT

The above Fig is programmed to implement an inverter whose LUT output (0) is always an inversion of its first input (A1). Other inputs A2, A3, and A4 act as "don't-care" bits but their values affect the signal propagation path from A1 to the output (0). In this context, it has been shown in Fig,

that the signal propagation path from A1 to the output (0) is shortest for A2A3A4 = 000 (marked with solid red line) and longest for A2A3A4 = 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 A1 is the inverter input and the rest of the LUT inputs (three) are controlled by 23 = 8 discrete levels.

MEMORY ARCHITECTURE

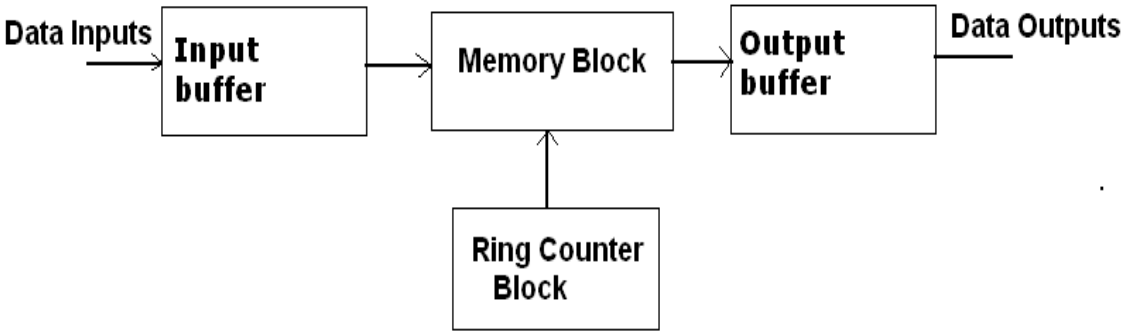

Fig5: Memory Organisation

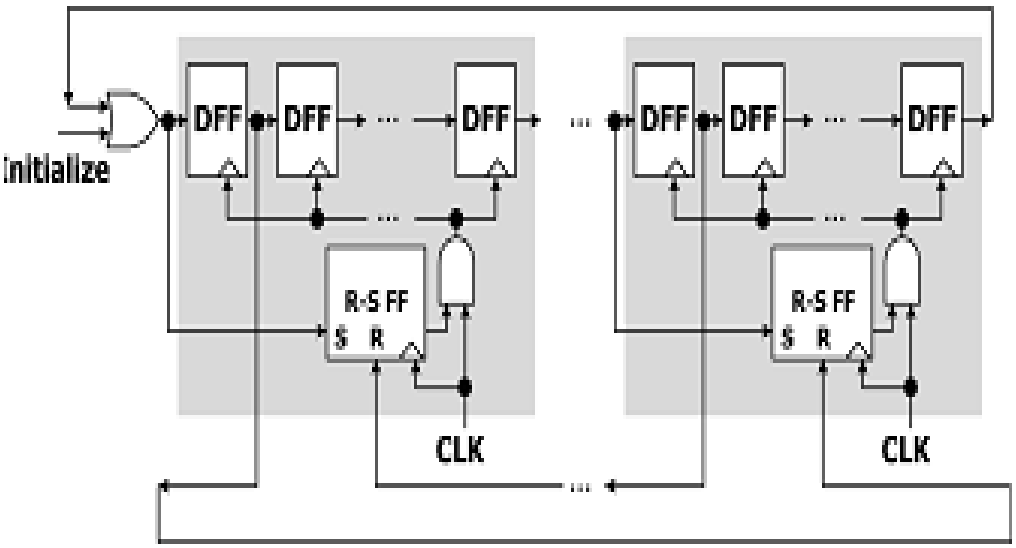

Fig6 : Ring Counter With SR Flip-Flops

The above block diagram shows the power controlled Ring counter. First, total block is divided into two blocks. Each block is having one SR FLIPFLOP controller to reduce constrained parameters.

RESULTS

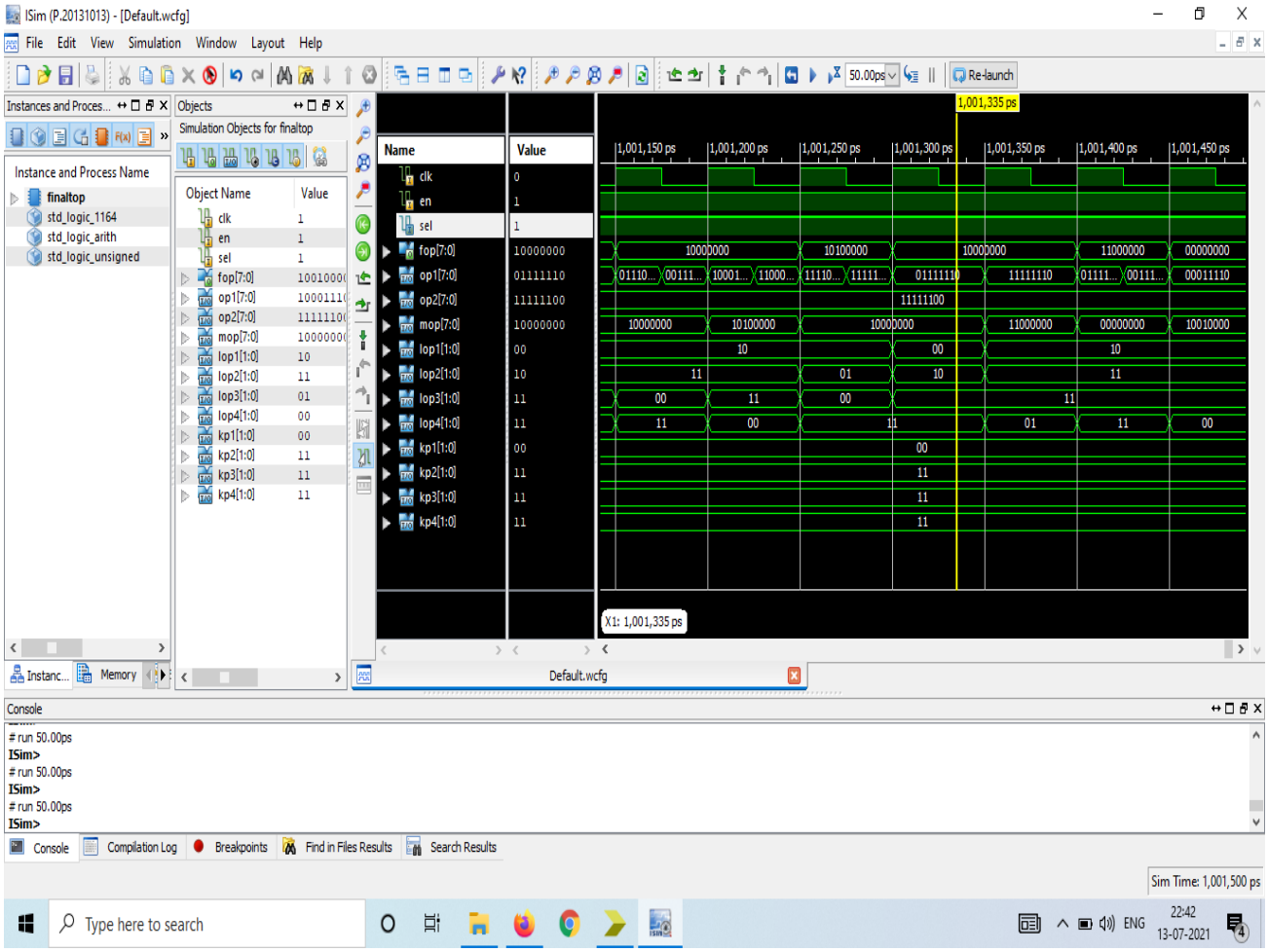

Fig7: Existing simulation output

Above simulation snap shows unique 8 bit patterns using existing method in XILINX ISE14.7 with general memory organisation. Irrespective of the accessed row, power supply will be transferred to each and every row of organised memory (RAM). Final patterns are generated after activating "en" input signal as'1'. Post processing will be done if input 'sel' is forced with '1'. For each and every simulation input signal 'clk' have to be clocked with raising edge as '1' and falling edge as '0'.

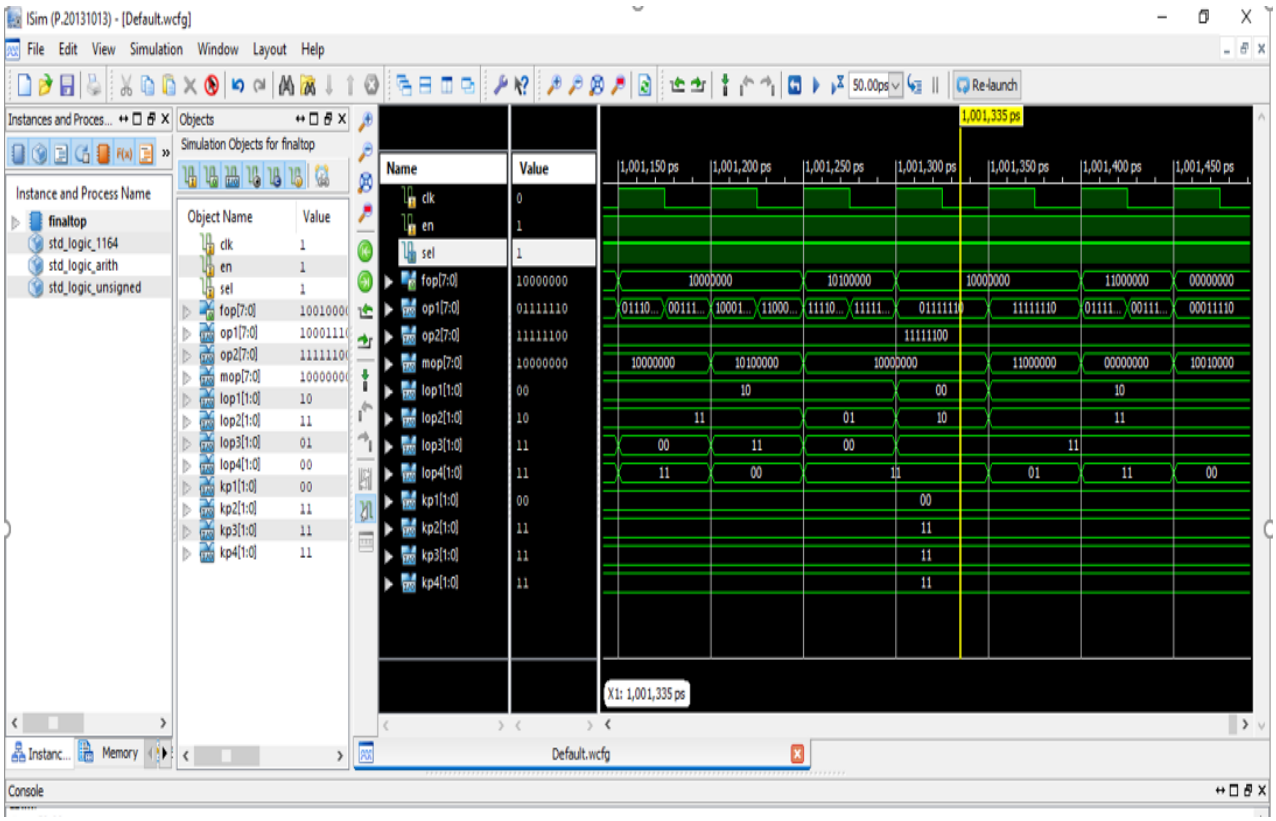

Fig8: Proposed simulation output

Above simulation snap shows unique 8 bit patterns using existing method in XILINX ISE14.7 with clock gating memory organisation. Above simulation snap shows unique 8 bit patterns using proposed clock gating method in XILINX ISE14.7 with modified structured memory organisation. Power supply will be transferred to corresponding block, which is accessed to store pattern of organised memory (RAM). Final patterns are generated after activating "en" input signal as'1'. Post processing will be done if input 'sel' is forced with '1'. For each and every simulation input signal 'clk' have to be clocked with raising edge as '1' and falling edge as '0'.

CONCLUSION and FUTURE SCOPE A new design of RO-based TRNG is described and its implementation on Xilinx is presented in this project. 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 more throughput after post-processing with a low hardware footprint. In addition, the restart experiments show that the output of the proposed TRNG behaves truly random. The bit-swapping LFSR used to generates a random test sequence with low switching power by finding hamming distance between two adjacent patterns and minimizing that distance by using combinational logic. To further reduce the average power, dual threshold voltages are assigned. By using this method and finding out the critical and non-critical paths present in BIST and then assigning a low threshold voltage for critical path, and high threshold voltage for non-critical path, a further reduction in total power, especially leakage power, can be obtained.

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] Shibinu A.R , Rajkumar. et. al, "Implementation of power efficient 4-bit reversible linear feedback shift register for BIST," Tech. Rep., 2010.

[4] D. Muthih and A. Arockia Bazil Raj, "mplementation of high-speed LFSR design with parallel architectures," 2014.

[5] Jayasanthi M, Kowsalyadevi AK, "Low Power Implementation of Linear Feedback Shift Registers ",2019

[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] Harshitha G; Kishore E J; Manoj R; Priyanka R Devarmani; Praveen Kumar Y G; M Z Kurian, “Gate-Diffusion Input based Linear Feedback Shift Register : A Review,” in 2092 .

[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] N. Nalla Anandakumar; Somitra Kumar Sanadhya; and Mohammad S. Hashmi, “FPGA-Based True Random Number Generation Using Programmable Delays in Oscillator-Rings” IIIT-Delhi, 2020 IEEE.