FPGA-BASED TRUE RANDOM NUMBER GENERATION USING PROGRAMMABLE DELAYS IN OSCILLATOR-RINGS

[1]Lingampally Shivprasad,[2]Shazia Fathima,[3]M.Krishna,[4]U.Sai Sindhu
[1234]Assistant Professor
Department Of ECE
Kshatriya College of Engineering

ABSTRACT:
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. In addition, a Von Neumann corrector as post-processor is employed to remove any bias in the output bit sequence. clock gating architecture 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.

1. 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 attackers 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. The number generation in the literature is performed in two different ways as deterministic and nondeterministic [1, 2]. PRNGs (Pseudo Random Number Generators), which are deterministic random number generators, generate numbers with fast, easy, inexpensive, and hardware independent solutions. The statistical qualities of these numbers produced are close to the ideal. PRNGs must meet the requirements specified in Table 1 to be used especially for authentication and key generation [3–5]. Therefore, nondeterministic functions are added to the output functions of PRNGs to guarantee these requirements. TRNGs (True Random Number Generators), which are nondeterministic random number generators, present slower, more expensive, and hardware-dependent solutions compared to PRNGs. Contrary to PRNGs, there is no need to include extra components in the TRNG system designs for R2, R3, and R4 requirements. Because of the unpredictability of random numbers generated by the use of high noise sources with high entropy in TRNGs, it is assumed that the R2 requirement is met. If the R2 requirement is satisfied, then it is assumed that the R3 and R4 requirements are also satisfied. To meet the R1 requirement in TRNGs, post processing techniques are applied on the random numbers obtained by sampling from noise sources. This eliminates the statistical weaknesses of random numbers at the output of the TRNG. In addition, post processing techniques eliminate potential weaknesses and make TRNG designs strong and flexible [6, 7]. Recently, there have been studies performed on random number generation from human-based noise sources [8–12]. Elham et al. showed that two different people would produce different random numbers and that these numbers could be used as biometric signatures [8]. Xingyuan et al. proposed a TRNG structure using a one-dimensional chaotic map based on mouse movements. The proposed structure showed that NIST tests were successful and could be used on personal PCs [9]. Hu et al. performed real random number generation by observing mouse movements of computer users. The statistical properties of the binary number generators generated from mouse movements of three different users were examined by the NIST test suite. Three chaos-based approaches were proposed to eliminate similar motions generated by the same user. Successful results were also achieved with these approaches [10]. Rahimi et al. used two different ECG signals for the cryptographic key generation and suggested two different approaches. The security analyses of keys obtained by both approaches were tested with distinctiveness, randomness, temporal variance, and NIST and successful results were obtained [11]. In the study performed by Chen et al. [12], random number generation was done from ECG signals and the analysis was tested by NIST test suite. It was revealed by the authors that the PRNG-based generated numbers had more successful results in classical PRNG structures. Dang et al. showed the possibility of random number generation from EEG signals. Four different EEG datasets were used to illustrate the use of obtained numbers in cryptography applications and their statistical properties were analyzed with the NIST test suite. In this PRNG-based approach, the samples consisting of EEG signals were transformed into 0 and 1 number generators. Mathematical definitions of the structure using modular arithmetic for the transformation of number generators were given. In the study, it was shown that EEG signals could be used for random number generation. The NIST test suite was used for this purpose and a success of higher than 99% was achieved [13]. In a study carried out by Chen et al. [14], the authors showed that EEG signals agreed with Gaussian distribution and also revealed whether random number could be generated from signals. They used the EEG signals obtained from both healthy and sick people for the PRNG number generation. They used NIST test suite for statistical analysis and they failed some tests. It was shown as a result of the study that the generated numbers could be used as a PRN. In a study done by Chen et al. [15], random number generation was performed by using white noise signals taken from MPEG-1, WEBCAM, and IPCAM video files and it was emphasized that successful results were obtained from statistical tests. Buhanupong et al. proposed a new encoding method for random number generation using EEG signals. This number generator, which can be used in low cost and real applications, is based on TRNG. A success of 99.47% was obtained from statistical tests. It was revealed that it was possible to do simple and fast bit generation by encoding method [16]. 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. The advances in many fields of science have, either directly or indirectly been dependent on the evolution of electronics. The electronic devices and systems are definitely inseparable from our everyday life affecting our lifestyle and life quality. As an example, today's computers with incredible capabilities have control on our life in many ways. In addition, the revolution in communication, media, transportation, etc.

has been due to advances in electronics. It is hard to believe that all of these advances have occurred only in a few decades revolutionizing the human life. The invention of transistors was undoubtedly the starting point of a huge revolution in electronics. The first transistor was invented in 1947 by Bardeen, Brattain and Shockley at Bell Telephone Laboratories. Nine years later, these three scientists received the Nobel Prize in physics for their valuable invention. In 1958, Jack Kilby built the first integrated circuit (IC) at Texas Instruments. He also received Nobel Prize in physics in 2000. In the mid 1960s, CMOS devices were introduced, initiating a revolution in the semiconductor industry. On 19 April 1965, Intel co-founder Gordon E. Moore published his famous paper in Electronics magazine [1] and predicted that the number of integrated components would be doubled every year. This prediction was based on changes in the number of integrated components during 1962-1965. In 1975, Moore amended his prediction to state that the number of transistors would be doubled about every 24 months. As shown in Figure 1.1, interestingly after 40 years, the number of transistors in CPUs manufactured by Intel is following the so-called Moore's law. In 40 years, the technology of IC production has evolved from producing simple chips with a few components to fabricating microprocessors comprising more than one billion transistors. Figure 1.2 shows the first Intel microprocessor 4004 with 2300 transistors clocked at a frequency of 108 KHz along with the new Core™ 2 Quad with 820 million transistors and clocked at frequencies above 3 GHz. True random number generators (TRNGs) harvest physical randomness as entropy sources and are heavily used in cryptography and security [1]. However, their power consumption and design complexity are often high [1-5]. The recently-proposed TRNG using Random Telegraph Noise (RTN) [6-8] has been considered as an ideal solution for future IoT applications (Fig.1), due to its simplicity, low power and robustness against temperature and supply voltage variations. Their practical use, however, is hindered by two major deficiencies: 1) Device Selectivity: RTN-TRNGs is based on one nano-scaled transistor exhibiting clear RTN signal by one preexisting trap. However, the percentage of such devices in one wafer is very low A large transistor array is usually needed in the design, out of which one transistor will be selected manually by tuning the circuit after fabrication. This leads to larger design area and higher cost [10-11]. 2) Low speed: The speed of the RTN-TRNG has strong correlation with τ, the sum of the times to capture and emission (τc, τe) [8]. The opposite voltage dependence of τc and τe imposes an inherent limit for speed. 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 nondeterministic. 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 pseudorandom 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 entropyper- bit through stochastic model [4].

2. LITERATURE REVIEW:

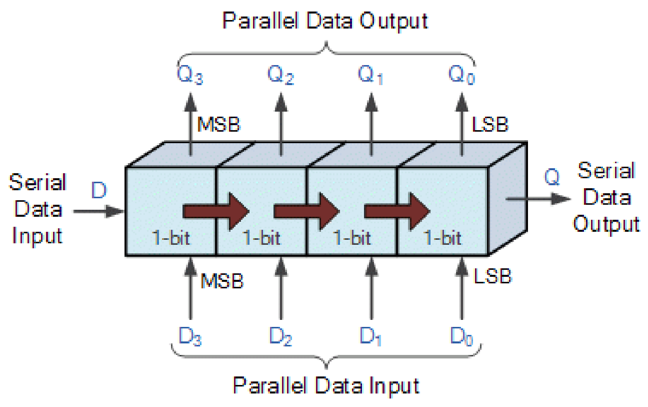
This section highlights the literature survey which has been done to review the critical points of the related works in recent days. In an irreversible circuit, if one bit information is lost then at least KTln2 joules of energy is dissipated. Where K is Boltzmann's constant and T is absolute temperature. This was stated by Landauer R in 1961[1]. In 1973, Bennett[2] 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. The paper is concluded with the comparative analysis of proposed approaches against performance parameters like garbage output, delay and quantum cost. Research paper [6], presents three different automated techniques for implementing LFSR as well as D flip flop so that the layout area and power consumption will be minimized. It is illustrated that LFSR is key component to provide self-test of an Integrated Circuit (IC). This paper implements LFSR upto layout level which will be a key component for low power application. The research explores the LFSR as well as D flip flop using different architecture in a 0.18μm CMOS technology; so that the layout area will be minimized and consumes less power. In paper [7], authors have presented a new approach for data compression and low- power test. This paper highlights the drawback of data compression schemes based on LFSR reseeding which increases the power dissipation and provides a new encoding scheme for reducing the power dissipation. With this background work, this paper mainly aims at design and implementation of an 8 bit LFSR using reversible logic for low power applications. Power dissipation is a crucial factor in testing. Higher power dissipation during the testing raises the temperature of the chip and more current will be drawn from the circuit, which results in damage of the circuit. So reducing the power dissipation in the design is given primary importance in VLSI.

3. LINEAR-FEEDBACK SHIFT REGISTER:

SHIFT REGISTER:

Shift Register

The Shift Register is another type of sequential logic circuit that can be used for the storage or the transfer of binary data.



This sequential device loads the data present on its inputs and then moves or "shifts" it to its output once every clock cycle, hence the name

Shift Register.

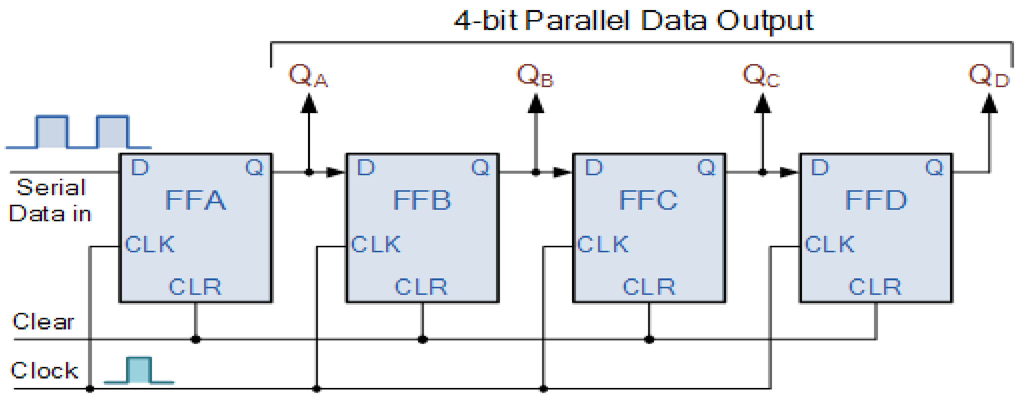
A *shift register* basically consists of several single bit "D-Type Data Latches", one for each data bit, either a logic "0" or a "1", connected together in a serial type daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on.

Data bits may be fed in or out of a shift register serially, that is one after the other from either the left or the right direction, or all together at the same time in a parallel configuration.

The number of individual data latches required to make up a single **Shift Register** device is usually determined by the number of bits to be stored with the most common being 8-bits (one byte) wide constructed from eight individual data latches.

4. SERIAL-IN TO PARALLEL-OUT (SIPO) SHIFT REGISTER

4-bit Serial-in to Parallel-out Shift Register



The operation is as follows. Lets assume that all the flip-flops (FFA to FFD) have just been RESET (CLEAR input) and that all the outputs Q_A to Q_D are at logic level "0" ie, no parallel data output.

If a logic "1" is connected to the DATA input pin of FFA then on the first clock pulse the output of FFA and therefore the resulting Q_A will be set HIGH to logic "1" with all the other outputs still remaining LOW at logic "0". Assume now that the DATA input pin of FFA has returned LOW again to logic "0" giving us one data pulse or 0-1-0.

The second clock pulse will change the output of FFA to logic "0" and the output of FFB and Q_B HIGH to logic "1" as its input D has the logic "1" level on it from Q_A. The logic "1" has now moved or been "shifted" one place along the register to the right as it is now at Q_A.

When the third clock pulse arrives this logic "1" value moves to the output of FFC (Q_C) and so on until the arrival of the fifth clock pulse which sets all the outputs Q_A to Q_D back again to logic level "0" because the input to FFA has remained constant at logic level "0".

TRUE RANDOM NUMBER GENERATOR (TRNG):

A hardware true random number generator is an electronic device that generates truly random and unpredictable binary numbers. The output pattern of a TRNG is arbitrary and non-deterministic in nature, meaning that the output binary numbers cannot be reproduced even if internal design and seed of the generator is known. As complete unpredictability is the key aspect of the true random number generator, a seed given to the TRNG must be random. Fortunately, it is not so difficult to collect true unpredictable randomness by tapping a chaotic world. Some of the examples of physical random sources are, thermal noise, shot noise, atmospheric noise, radioactive decay and clock jitter. A TRNG can be fed a seed from such a physical random process to get true random outputs. Another characteristic of TRNGs is that they are non-periodic in nature (as opposed to PRNGs), meaning that output binary pattern of a TRNGs is never repeated even if the same seed is applied to the generator.

CLOCK JITTER BASED TRNGS:

The period of oscillation for an ideal oscillator is constant, such that the time between the consecutive rise and fall of edges would be same. The period of oscillation for a ring oscillator composed of real elements is not constant, however, because the time between similar edges is not constant. The time period of such a clock is unpredictable. This variation of the time period in a ring oscillator is known as clock jitter (or phase noise). The simplest ring oscillator is made by connecting odd numbers of inverter gates in the form of a ring as shown in figure 8, the output of one gate becomes the input of the other gate. The last inverter output is fed back into the first inverter such that the last output of a chain is the logical NOT of the first input. The feedback provided by the last output to the input of the first inverter causes oscillation. A ring composed of an even number of inverters cannot be used as a ring oscillator; in that case the last output would be the same as the input of the first inverter which would create a stable state, suppressing oscillation.

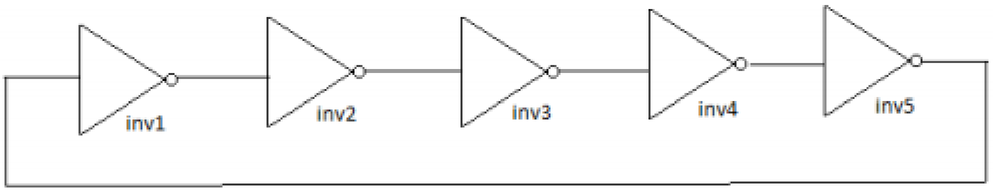


Figure:1. Ring Oscillator using five inverter gates

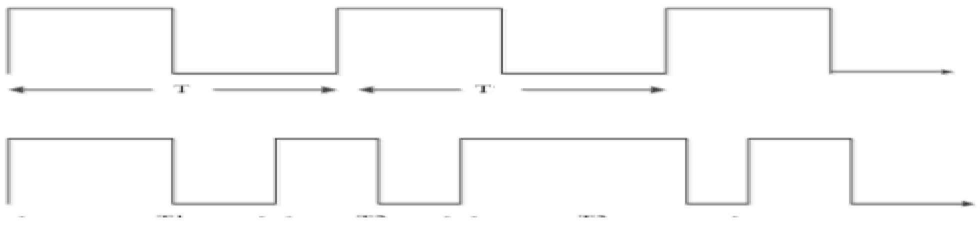


Figure :2. Clock Jitter

A clock jitter property of free running ring oscillator has been used for long time to generate true random number generator as a source of randomness. A very good discussion of building a two ring oscillator based RNG is illustrated in [6]. In this TRNG, a clock jitter in a ring oscillator is used as a source of noise. The TRNG is built with two ring oscillators and three LFSRs 13-bit, 19-bit and 32-bit. One ring oscillator clocks 19-bit LFSR and the other clocks the 13-bit and 32-bit LFSR. Figure from [6] shows the block diagram of the TRNG.

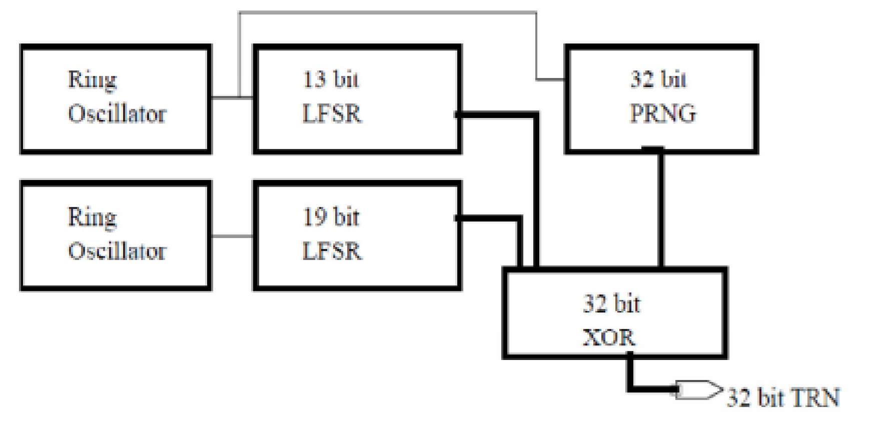


Figure :3. Block Diagram of TRNG [6]

Adding an independent action of split rotation on two LFSRs gives a more uncertain output sequence then the normally operating LFSRs. An extra circuit is added to provide split rotation for the 19-bit LFSR and the 13-bit LFSR. The upper half bits of these LFSRs are rotated left and the lower half bits are rotated right. These two LFSRs are then concatenated to generate the 32-bit long random number sequence, and then XORed with the 32-bit LFSR output to generate the final whitened 32-bit random number. A similar concept is used to design the TRNG of this thesis, however, the differences exist in the design of the ring oscillators and the control registers for LFSRs, such as addition a different independent action in 19-bit and 13-bit LFSR. In this RNG, a 'preloader' is added to reshuffle the 13-bit and 19-bit LFSR bits. Another paper, [7], describes how the randomness related to the unpredictability of the frequency of ring oscillators is used as a source of implementing the true random number generator. This approach uses counters clocked by free-running ring oscillators and sampled at a regular interval. As there is some jitter associated with the clock, there is a 22 small amount of uncertainty about any particular value of the counter, and the output of the counter is random numbers at sample instants. However, the value generated by this generator can be approximated, as path of the generator through the range of possible values is fixed; that is, the generator always counts by incrementing its output every clock. To overcome this weakness, the path traversed by the generator as it counts, is varied by changing the generator operation only at sampling instant. Normally, the function performed by a counter is either incrementing or decrementing the number. If the function is changed to an independent operation like left shifting (doubling) only at the sample instant, then the output value becomes more random. Adding a tertiary independent operation such as transposition at the sampling instant generates even more random output. The generator, which previously had only one path through the range of possible values, now has many more paths throughout the range of possible values. Thus, it can be seen that adding more independent operations to the counter at sample instant increases the randomness of the output sequence. As shown in figure 11 from [7], the RNG is designed with two 16-bit counters to generate a 32-bit random sequence: One counter counts up while the other counts down. Dividing the 32-bit counter into two 16-bit counters decreases the overall time required for generating all possible bit signals. Each counter has a unique ring oscillator. A secondary operation is provided by a 32-bit logarithmic shifter where the 5-bit counter is clocked by its unique ring oscillator that provides shift count for the log shifter. A tertiary operation is provided by adding a transposer unit that is 4:1 multiplexer. It selectively transposes bits in one of four possible patterns. The output sequence coming out from the generator is truly random

5. PROPOSED TECHNIQUE:

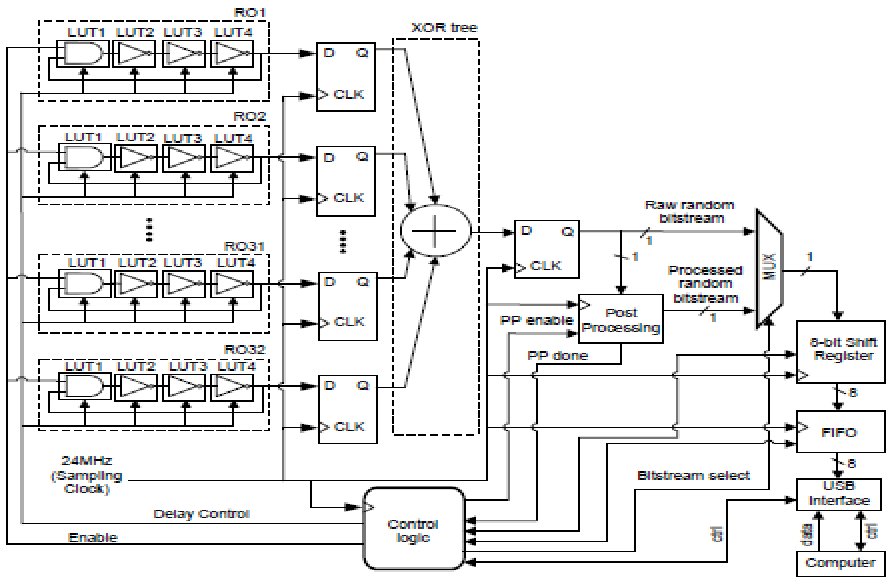


Fig: 3. Architecture of the proposed TRNG.

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 [14]. Equal length oscillator rings configured in FPGA are highly correlated with each other dueto 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 flipflops. On the other hand, our work uses random jitter of freerunning 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. 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.

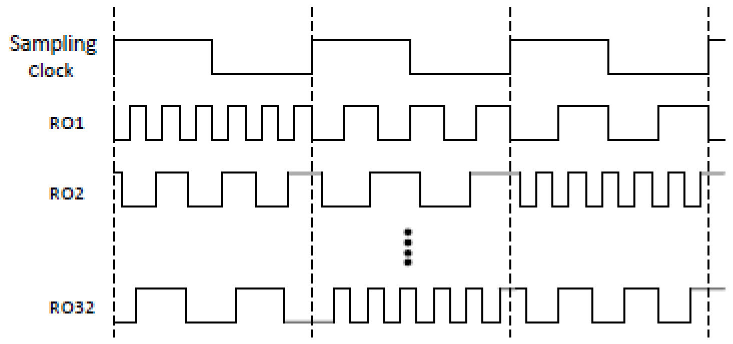


Fig.6. RO outputs for each sampling clock by using PDL

The proposed 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 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, 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 postprocessing. 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 64by 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.

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

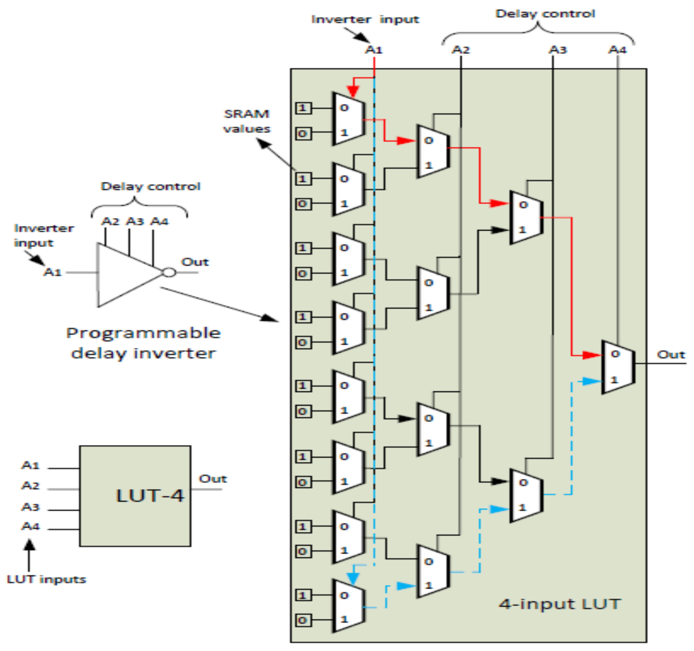
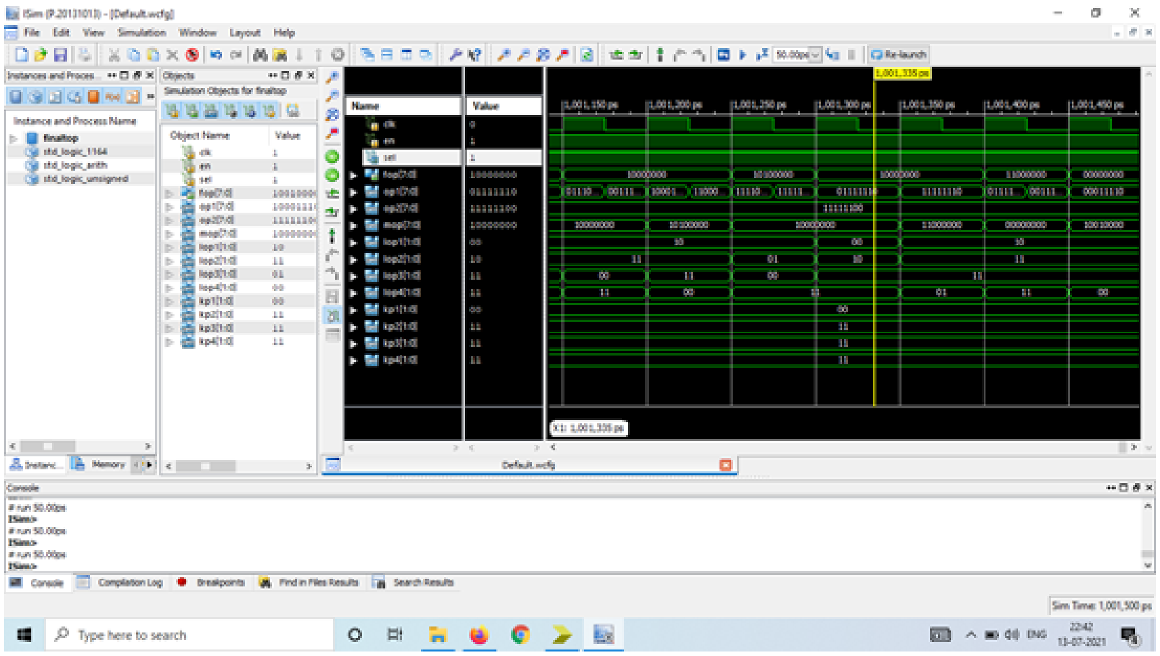


Fig. 7: PDL using a 4-input LUT.

Fig. 1 is programmed to implement an inverter whose LUT output (0) is always an inversion of its first input (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. 1, 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.

6. RESULT:



7. CONCLUSION:

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 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.

FUTURE SCOPE:

The bit-swapping LFSR used by Dhanesh [2] 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] 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