Efficient Pseudo Random Number Generator (PRNG) Design on FPGA

Sonia Akter[†1], Kasem Khalil[*§2], Magdy Bayoumi[¶3]
[†]The Center for Advanced Computer Studies, University of Louisiana at Lafayette, LA, USA
[*]Department of Electrical and Computer Engineering, University of Mississippi, Mississippi, USA
[§]Department of Electrical Engineering, Assiut University, Assiut, Egypt
[¶]Department of Electrical and Computer Engineering, University of Louisiana at Lafayette, LA, USA
Emails: {Sonia.akter1@louisiana.edu, kmkhalil@olemiss.edu, mab0778@louisiana.edu}

2024 IEEE 17th Dallas Circuits and Systems Conference (DCAS) | 979-8-3503-4953-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/DCAS61159.2024.10539915

***Abstract*—Random Number Generators (RNGs) are substantially used in many security domains, providing a fundamental source of unpredictability essential for tasks such as cryptography, simulations, and statistical analyses. The efficiency and quality of an RNG directly impact the reliability and security of diverse applications, making advancements in RNG design, as explored in this study, of significant importance for enhancing computational processes. This paper presents an innovative Pseudo-Random Number Generator (PRNG) that leverages the efficiency of two carefully selected Linear Feedback Shift Registers (LFSRs) and a connecting XOR gate. The investigation of five polynomials identified an optimal pair, resulting in a notable improvement of over 200X in the length of random bit sequences compared to a single LFSR-based PRNG. The Basys3 FPGA board with the xc7a35tcpg236-1 FPGA chip was used to implement and synthesize the proposed design. Two significant findings emerge from this research. Firstly, using variable polynomials demonstrates a huge enhancement in the duration of randomness, outperforming the impact of variable seeds. A noteworthy observation is that employing the same polynomials in different branches does not result in optimal results. Secondly, managing more seeds is associated with an increased area cost, underscoring the efficiency of handling two polynomials.**

Keywords: Random Number Generator, Random Bit, LFSR, PRNG, Security, Integrated circuit, FPGA, Basys3, FLip Flop.

I. Introduction

Security is significant in several applications and is paramount for protecting against cyber threats, ensuring the integrity, confidentiality, and availability of data transmitted and processed by interconnected devices[1–7]. However, PRNG based on LFSR, also has applications in Quantum Key Distribution (QKD) [8]. QKD uses quantum mechanics to establish shared encryption keys between two parties securely. Morever, LFSR-based Random Number Generators (LRNGs) are crucial in low-power applications, particularly in wireless technologies like Bluetooth. They are employed to encrypt frequency hopping spread spectrum systems, enhancing security against signal jamming attacks [9].

The work conducted by Durga et al. [10] propose an RNG based on the Linear Feedback Shift Register (LFSR). To enhance the unpredictability of the RNG, this work incorporates a polynomial modulator. The polynomial Modulator can select different primitive polynomials using a counter, a multiplexer, and a comparator. Furthermore, different primitive polynomials are polynomials with varying bit lengths, including 4-bit, 8-bit, and 16-bit lengths. Although the proposed method successfully passed the NIST 800-22 Test suite, it did not compare the results with similar work in the state-of-the-art literature regarding resource utilization.

The work by Hussain et al. [11] present an LFSR-based pseudorandom number generator to encrypt data. The proposed method uses two LFSRS, LFSR1 and LFSR2, along with a 1-bit comparator. The comparator is used to incorporate inequality in the linear recurrence equation and improve the randomness of the PRNG. The proposed method consumes more power but claims to have more linear complexity compared to traditional 5-bit, 7-bit, and 12-bit LFSR, thus applicable for high-security applications [12, 13].

The research by Bailey et al. [14] present the performance of 4, 8, 16, and 32-bit reversible LFSRs using the Pareek gate approach. All designs were developed using primitive polynomials and implemented in Xilinx Spartan6 FPGA on the Xilinx ISE 14.7 platform. This method reduces the power consumption by the random number generator because reversible LFSRs allow very fast random sequence generation compared to PRNG. On the other hand, PRNG uses linear congruential equations requiring high mathematical operations, resulting in high power consumption. However, this work did not mention the cost paid in randomness while achieving a 10% reduction in power.

Gudla et al. [15] propose a PRNG constructed in Resistance Random- Access Memory chip (ReRAM) to improve the dependability, unpredictability, and unrepeatability of the ReRAM chip. The suggested method used Modified LFSR to generate an ultra-high-throughput random sequence. The idea is to use a ReRAM RTN circuit to generate a random control signal for the MUX. An integrated linear feedback shift register produces an unseen/microscopic quick NOT in the sequence data source. This work used Transmission gate-based 2X1 MUXs to select between signals because transmission gates can transfer input signals to outputs without attenuating threshold voltages. This method might require a high-speed clock source, and the randomness quality may depend on the Digital Clock Manager (DCM) quality.

The work by Zode et al.[16] propose an LFSR-based True Random Number Generator. The idea is to increase the randomness using fly seed change using the FPGA Xilinx IP System monitor. Power supply voltage and on-chip temperature variations were used to generate the random seed. The whole design is implemented on the Xilinx Virtex-5 ML505 board. On-chip resources consume less space and less power. The proposed method passed all NIST Statistical tests. Tests were performed on 218 MB data output from TRNG. The architecture takes only 24 LUTs and 33 slice registers out of 69120, thus making it suitable for resource-constrained devices. Verilog HDL language was used, simulated & synthesized on the ML505 Evaluation board (Device XC5VLX110T). on-board frequency of 33MHz was used.

Han et al. [17] present LFSR-based RNG by introducing the concept of a dynamic polynomial Modulator to avoid predictability. The proposed method can generate 4000 times larger random numbers before repetition than conventional LFSR-based random number generators. The dynamic polynomial Modulator means changing the polynomials by using switches when the output of the LFSR becomes the initial seed to avoid periodic patterns. The proposed method mostly focuses on the quality of the randomness. Tang et al. [18] illustrate configurable LSFR-based PRNG by using the characteristics of the metastable state of the digital circuit. The suggested method can produce pseudo-random 16-bit numbers within the range of 1 to 40,000. Configurable means, seeds, and feedback factors can both be controlled by using a metastable state.

In this academic study, we begin by selecting five polynomials for LFSR design, denoted as traditional design, assessing their performance regarding random sequence length, power consumption, and resource usage. This groundwork was crucial for subsequent designs requiring two high-performing polynomials. We then present two designs to enhance random bit sequence generation compared to traditional LFSR-based RNGs. The proposed method 1, involved creating a PRNG using two LFSRs with the same polynomials but different seeds for each. The proposed method 2 was introduced as an improved solution, providing a notably higher random sequence than the initial proposal.

The paper is structured as follows: In Section II, the design strategies of the proposed Random Number Generator (RNG) are discussed. Following that, Section III presents the implementation details and experimental results. Finally, Section IV concludes the paper.

II. The Proposed Method

In communication systems, linear feedback shift registers (LFSRs) play a vital role in various applications such as scramblers, randomizers, convolutional encoders, and generators of pseudo-random bit sequences (PRBS) [19]. A primitive polynomial, defined as an irreducible mathematical expression, holds significance in generating the longest possible sequence without repetition in specific digital systems like pseudo-random number generators (PRNGs) [20]. These polynomials, being indecomposable into smaller components, serve as fundamental building blocks for creating extensive, non-repeating sequences, especially crucial in cryptographic operations [21]. In Fig. 1, an example of a traditional eight-bit LFSR using the maximum-length polynomial is illustrated. The feedback mechanism of this LFSR involves the XOR combination of the 8th, 6th, 5th, and 4th bits, which are then fed back to the least significant bit (LSB) of the LFSR. An LFSR's feedback function, achieved through a simple polynomial, determines the series of pseudo-random numbers generated by the LFSR.

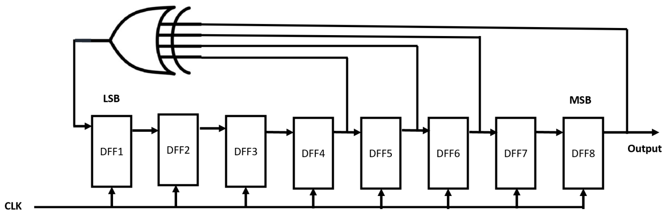

Fig. 1: Basic structure of eight-bit LFSR.

The proposed method 1, incorporates a RNG structure that enhances randomness by employing the XOR operation on the outputs of two identical LFSRs having two different seeds. Fig. 2 illustrates the proposed method 1, where the outputs of the two LFSRs are combined to produce a more robust and unpredictable random sequence. The key feature of the proposed method 2 (Fig. 3) lies in the utilization of two different polynomials for the two LFSR branches, contributing to the diversity and complexity of the generated random numbers. For example, Fig. 3, one LFSR employs the Polynomial-1:

$$x^8 + x^6 + x^5 + x^4 + 1$$

While the other LFSR utilizes the Polynomial-2:

$$x^7 + x^5 + x^4 + x^3 + 1$$

This deliberate choice of distinct polynomials introduces variability in the feedback mechanism of each LFSR, contributing to the overall randomness of the generated sequence. It is noteworthy that, despite the different polynomials employed, the proposed method 2 maintains the unique characteristic of using the same initial seeds for both LFSR branches. The initial seed represents the starting point of the LFSR operation and significantly influences the subsequent sequence of random numbers generated. By employing the same seed for both LFSRs, method 2 introduces a controlled yet intricate interplay between the two branches, aiming to enhance the randomness and statistical properties of the overall RNG output.

The proposed methods, 1 & 2, comprise of traditional method. Moreover, the traditional method is a basic 8-bit LFSR. The traditional method uses four inputs, namely "seed," "enable," "reset," and Clk. According to Fig. 1, feedback bit is calculated by xoring specific bits (8,6,5,4) when using

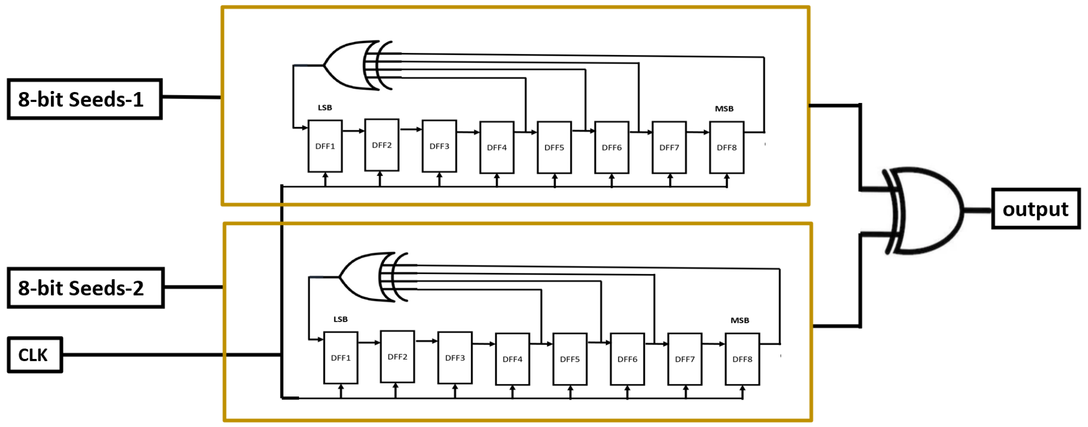

Fig. 2: The first proposed method using two distinct seeds with identical polynomials.

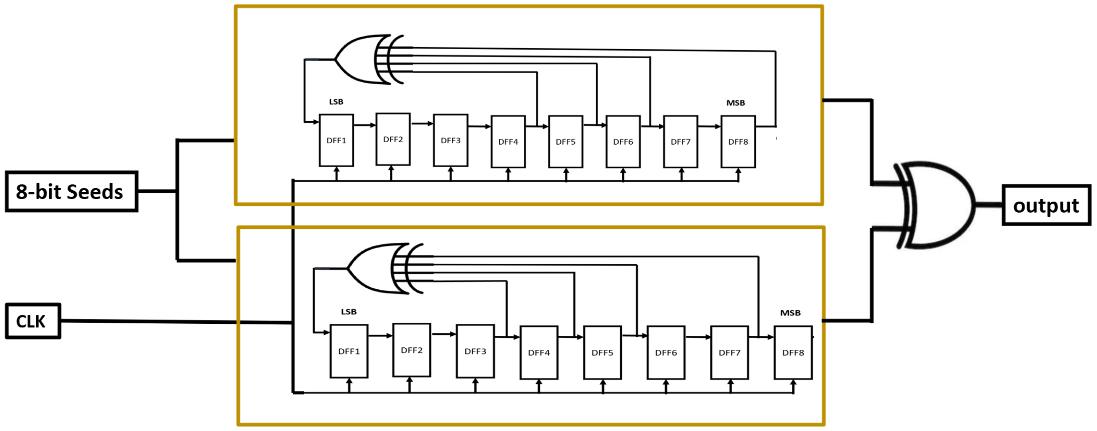

Fig. 3: The second proposed method based on two distinct polynomials and one seed.

polynomisl-1 and feedback to LSB bit of the LFSR. Now, at each positive edge of the clock cycle (posege clk), if reset is low (nonactive) and enable signal is active, 1 bit shifting towards MSB happens, and precalculated bit (by XORring) then will become the new least significant bit. An 8-bit register is used to store the current state of the LSFR. For proposed method 1, two different seeds are used, but two of the same polynomials are used for two branches of the design. On each positive edge of the clock cycle(if enabled), two of the same XOR operations will be performed based on the specified bits in the polynomials, and finally, another XOR operation will be performed based on the output MSB bits of the two LFSRs. Even though the bit number for the XOR operation is the same, the bit value initially depends on the initial seed.

In the proposed method 2, two LFSRs have used two different polynomials. On each positive edge of the clock cycle (if enabled), two different XOR operations will be performed based on the specified bits in the polynomials. Output bits from both LFSRs are again XORed to get new random bits, which is the pseudo-random sequence of the proposed design. The initial seed for method 2 is the same for two LFSR branches.

III. Implementation and Experimental Results

The proposed design was developed and synthesized on the Basys3 FPGA board, featuring the xc7a35tcpg236-1 FPGA chip, utilizing the Xilinx Vivado 2021.1 platform. The Basys3 FPGA board provides a versatile and accessible hardware platform for digital design projects, making it well-suited for the realization of the proposed design. During the simulation phase, a clock frequency of 100MHz was utilized to model the behavior of the designed circuit. The initial seed for traditional eight-bit LFSR and proposed method 2 is Hexa Decimal FF. The proposed method 1 uses two initial seeds: Hexa Decimal FF and Hexa Decimal 55.

A comprehensive overview of five potential polynomials designed for an 8-bit LFSR is shown in TABLE I. We have selected these polynomials from the literature as follows: Polynomial-1 [22], Polynomial-3 [14], Polynomial-4 & Polynomial-5 [23]. However, Polynomial-2 is experimental for this paper. This table details the length of random bit sequences, power consumption, and area consumption associated with each polynomial during its execution in the 8-bit LFSR. Notably, Polynomial-1, Polynomial-3, and Polynomial-5 emerge as the most promising options, demonstrating op-

TABLE I: 8-bit polynomials selection based on their performance based on Fig. 1.

Polynomials	Repetition after(ns)	Power Consumption (W)	Resource Utilization		
			LUTs	Registers	Bonded IOB
$x^8+x^6+x^5+x^4+1$	2615	1.18	34	24	19
$x^7+x^5+x^4+x^3+1$	1335	1.181	34	24	19
$x^8+x^4+x^3+x^2+1$	2615	1.199	34	24	19
$x^8+x^7+x^6+x^5+1$	2225	1.179	34	24	19
$x^8+x^6+x^4+x^3+x^2+x^1+1$	2615	1.252	36	24	19

TABLE II: PRNG results based on the first proposed method

Combination of Polynomials	Repetition after(ns)	Power Consumption (W)	Resource Utilization		
			LUTs	Registers	Bonded IOB
1&1	2615	2.559	67	48	27
2&2	1335	2.566	67	48	27
3&3	2615	2.566	67	48	27
4&4	2225	2.559	67	48	27
5&5	2605	2.673	71	48	27

TABLE III: PRNG results based on the second proposed method

Combination of Polynomials	Repetition after(ns)	Power Consumption (W)	Resource Utilization		
			LUTs	Registers	Bonded IOB
1&2	323905	2.493	58	40	19
1&3	2575	2.499	59	40	19
1&4	553385	2.495	59	40	19
1&5	2565	2.555	61	40	19

timal results in terms of generating a series of unrepeated bits before repetition occurs (at 2615 ns). It is important to note that the clock period is 10 ns, requiring 10 ns to generate one random bit, and it generates 255 bits over 2615 ns, characteristic of max length polynomial, indicating the LFSR is functioning flawlessly. However, The above-mentioned selected polynomials will be further evaluated in the subsequent analyses depicted in Fig.2 (results presented in TABLE II) and Fig.3 (results presented in TABLE III). The objective is to assess and identify a high-performance PRNG based on the criteria of generating the longest series of eight-bit sequences before repetition occurs. The resource consumption and power consumption results for each configuration will be thoroughly examined and subsequently reported to provide a comprehensive understanding of the performance characteristics of the proposed designs.

TABLE II, represents polynomials pair used in two branches of proposed method 1, length of unrepeated bits, power consumption, and resource utilization for each polynomial combination. The proposed method 1 shows how long the unrepeated bits continue to generate while polynomials are the same, but the seeds differ. It is evident that when the Polynomial-1 & Polynomia-1 pair and Polynomial-3 & Polynomial-3 pair used in method 1 provide the highest random bit sequence. However, if TABLE II is compared with TABLE I, random bit sequence is almost the same for the two designs. It is clear that it is better to use the traditional method because it provides almost the same length of randomness with less power and area consumption. Even though proposed method 1 seems inefficient, it revealed two important insights: 1) storing seeds is costlier than running more polynomials, and 2) using different seed pairs dynamically will give almost the same results in terms of randomness if polynomials are the same.

TABLE III, depicts the polynomial combination used in proposed method 2 for the PRNG, length of unrepeated bits, power consumption, and resource utilization for each polynomial combination. When polynomial 1 & 4 is used in proposed method 2, it gives the highest length of randomness, which is more than 200 times (553385 ns) what a single polynomial can generate, compared with the highest length in TABLE I. It can be concluded that using Polynomial-1 and polynomial-4 is the best choice for the proposed design in this work. Using polynomial-1 and polynomial-2 can give the second-best result in terms of random bit sequence. Now comparing TABLE II with TABLE III, it is clear that proposed method 2 consumed less power and less area and provided better results in terms of randomness (also 200X ns length of random bit sequence). The work by Panda et al. [23] implement and investigate the performance of the 8, 16, and 32-bit LFSR on FPGA using Verilog VHDL. This method did not use any new approach to enhance the length of stages with random bits. However, the proposed method 2 in our work significantly increases the size of the random bit sequence length, especially while choosing polynomial-1 & polynomial-4, and Polynomial-1 & polynomial-2.

IV. Conclusion

This study introduces an efficient PRNG employing two carefully selected LFSR joined by an XOR gate. The meticulous investigation of five polynomials revealed that two are optimal for the proposed design, which was synthesized on the Basys3 FPGA board with notable success. The results demonstrate an improvement of over 200 times longer random bit sequences compared to a single LFSR-based PRNG. However, this research also unveils two crucial findings. Firstly, employing variable polynomials significantly enhances randomness, surpassing the impact of using stored variable seeds. However, it is noted that using the same polynomials in different branches is noncritical to achieving optimal results even if the seeds are different. Secondly, the management of more seeds comes at the cost of increased area, emphasizing the efficiency of handling two distinct polynomials. Moving forward, we aim to explore the dynamic application of five polynomials for TRNG, opening up a promising path for further investigation in this field.

References

[1] H. U. Khan, M. Sohail, F. Ali, S. Nazir, Y. Y. Ghadi, and I. Ullah, "Prioritizing the multi-criterial features based on comparative approaches for enhancing security of iot devices," *Physical Communication*, vol. 59, p. 102084, 2023.

[2] S. Akter, K. Khalil, and M. Bayoumi, "Hardware security in the internet of things: A survey," in *2023 IEEE 36th International System-on-Chip Conference (SOCC)*, pp. 1–6, IEEE, 2023.

[3] Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. H. Luan, and X. Shen, "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 319–352, 2022.

[4] K. Khalil, K. Elgazzar, A. Abdelgawad, and M. Bayoumi, "A security approach for coap-based internet of things resource discovery," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pp. 1–6, IEEE, 2020.

[5] A. Vangala, A. K. Das, V. Chamola, V. Korotaev, and J. J. Rodrigues, "Security in iot-enabled smart agriculture: Architecture, security solutions and challenges," *Cluster Computing*, vol. 26, no. 2, pp. 879–902, 2023.

[6] K. Khalil, A. Sherif, M. M. Mamun, M. Elsersy, A. A.-A. Imam, M. Hataba, and M. Mahmoud, "Privacy-preserving and hardware acceleration-based authentication scheme for data collection in e-health applications," in *2023 IEEE 66th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 703–707, IEEE, 2023.

[7] M. Seliem, K. Elgazzar, and K. Khalil, "Towards privacy preserving iot environments: a survey," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–15, 2018.

[8] P. Chandravanshi, J. K. Meka, V. Mongia, R. P. Singh, and S. Prabhakar, "Lfsr based rng on low cost fpga for qkd applications," *arXiv preprint arXiv:2307.16431*, 2023.

[9] A. Ebrahimzadeh, A. Falahati, *et al.*, "Frequency hopping spread spectrum security improvement with encrypted spreading codes in a partial band noise jamming environment," *Journal of Information Security*, vol. 4, no. 1, pp. 1–6, 2013.

[10] R. S. Durga, C. Rashmika, O. N. Madhumitha, D. Suvetha, B. Tanmai, and N. Mohankumar, "Design and synthesis of lfsr based random number generator," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 438–442, IEEE, 2020.

[11] S. Hussain, A. K. Chaudhary, and S. Verma, "Enhancing security in iot devices by using pseudo random number generator based on two different lfsr and a comparator," in *2022 IEEE Delhi Section Conference (DELCON)*, pp. 1–7, IEEE, 2022.

[12] S. Akter, K. Khalil, and M. Bayoumi, "A survey on hardware security: Current trends and challenges," *IEEE Access*, 2023.

[13] A. Meza, F. Restuccia, J. Oberg, D. Rizzo, and R. Kastner, "Security verification of the opentitan hardware root of trust," *IEEE Security & Privacy*, 2023.

[14] K. Bailey *et al.*, "Fpga implementation of reversible lfsr with primitive polynomial using verilog hdl," in *2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, pp. 1–5, IEEE, 2022.

[15] V. V. Gudla and V. S. S. S. S. Jyothi, "Design and implementation of digital clock manager based pseudo-true random number generator," in *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)*, pp. 1–5, IEEE, 2022.

[16] P. Zode, P. Zode, and R. Deshmukh, "Fpga based novel true random number generator using lfsr with dynamic seed," in *2019 IEEE 16th India Council International Conference (INDICON)*, pp. 1–3, IEEE, 2019.

[17] M. Han and Y. Kim, "Unpredictable 16 bits lfsr-based true random number generator," in *2017 International SoC Design Conference (ISOCC)*, pp. 284–285, IEEE, 2017.

[18] H. Tang, T. Qin, Z. Hui, P. Cheng, and W. Bai, "Design and implementation of a configurable and aperiodic pseudo random number generator in fpga," in *2018 IEEE 2nd International Conference on Circuits, System and Simulation (ICCSS)*, pp. 47–51, IEEE, 2018.

[19] B. Soreng, *Implementation of WiMAX physical layer baseband processing blocks in FPGA*. PhD thesis, 2013.

[20] K. Mandal, "Design and analysis of cryptographic pseudo-random number/sequence generators with applications in rfid," 2013.

[21] E. Almaraz Luengo, "A brief and understandable guide to pseudo-random number generators and specific models for security," *Statistic Surveys*, vol. 16, pp. 137–181, 2022.

[22] A. Kurt, "Design of 4 bit and 8 bit pseudo noise sequence generators with all zero condition protection circuit," in *2021 13th International Conference on Electrical and Electronics Engineering (ELECO)*, pp. 480–484, IEEE, 2021.

[23] IEEE, *FPGA implementation of 8, 16 and 32 bit LFSR with maximum length feedback polynomial using VHDL*, 2012.