Recent Development of Hardware-Based Random Number Generators on FPGA for Cryptography

Mangal Deep Gupta and R. K. Chauhan

Abstract The most important component in the cryptographic system is the cryptographic keys generator. These keys are generated by a random number generator (RNG) since the security of the cryptographic system depends entirely on the quality of generated keys. This paper summarizes the recent development of FPGA-based hardware efficient and secure RNGs. The main aim of this study is to summarize the knowledge of hardware performance, security strength, and suitability for the cryptographic system from the different classes of RNGs. It discusses the different classes of RNGs, recalls the basic ideas, and provides the details of several well-known RNGs. This work presents a comprehensive discussion on the hardware implementation of RNGs on FPGAs. A complete list of LCGs-based pseudorandom number generators (PRNGs) is presented with deep technical details on their mathematical formation and implementations. Finally, the performance of RNGs with respect to utilization of FPGA resources, frequency, latency, power consumption, security strength using the national institute of statistical testing (NIST), and weaknesses is presented.

Keywords RNGs · Cryptography · VLSI · FPGA

1 Introduction

With the rapid development of information technology, the cryptosystem is used widely to protect the data or information. A various encryption techniques are used to make ensure of information security. So the random number generator (RNG) used in cryptography determines the system security. The RNGs are widely used in various applications related to cryptography such as key generation, encryption/decryption, masking protocols, Internet gambling, and block ciphers [1–4]. A different RNG is proposed by the researchers that enhance the security of information. In the evolutionary development of smart mobile devices that are connected to the internet, the

M. D. Gupta (✉) · R. K. Chauhan
Department of Electronics and Communication Engineering, MMMUT, Gorakhpur, Uttar Pradesh, India
e-mail: mangaldeepgcet@gmail.com

B. Mishra and M. Tiwari (eds.), *VLSI, Microwave and Wireless Technologies*,
Lecture Notes in Electrical Engineering 877,
https://doi.org/10.1007/978-981-19-0312-0_48

information exchange over an insecure network is a major concern over the years. The physical systems like environment monitoring, health care system, advanced metering in smart grids, etc., which are connected over the Internet of things (IoT) that generate a large amount of data that leads to privacy and security issues. To assure the security of associated information over an Internet network, the RNG is the primary requirement [5]. Due to trade-off between different factors (like hardware performance, security, and cost), most of the cryptosystems are unsuitable for real-time implementation on IoT-based resource constraint devices [6]. To accomplish this request, it is required to implement efficient hardware architectures capable of generating pseudorandom bit sequences to provide the public and private keys for effective data cryptography. Therefore, the hardware-based cryptosystem is compulsory in IoT applications for secure information exchange over smart mobile devices that are connected with IoT. So, the primary requirement of a hardware-based cryptosystem is low-hardware complexity, high speed, low-power consumption, secure key generation, and high randomness. In this regard, different RNG methods were proposed for the generation of the random number to satisfy the randomness behavior as required for cryptography.

The FPGA implementation of the RNG is more useful in real tile applications like cryptography, secure communications, etc. The fully digital circuit or/and embedded systems with high-speed and low-power consumption are suitable for IoT, cybersecurity, and Industry 4.0 security applications. The RNGs based on FPGA open the opportunity to use the large number of combinational blocks that are connected through programmable logic. So, this powerful platform is widely used in digital circuit implementations [7].

This research work surveys the large set of FPGA implementations of RNGs. First, summarize the different classifications of RNGs and provide the advantages and weakness of different well-known RNGs both pseudorandom and truly random, while linear and nonlinear system-based generators are discussed in the PRNG case. The LCGs-based PRBGs are explained in detail and also discuss the choices of the RNGs. Finally, present the performance of FPGA-based PRNGs in terms of FPGA resources, timing performance, security strength, and weakness.

The remainder of the article is as follows. Sect. 2 refers to the classification of the RNGs. Sect. 2.1 presents the FPGA implementations of nonlinear PRNGs, whereas the next Sect. 2.2 focuses on linear system-based PRNGs. This section summarizes the mathematical formation and VLSI architectures corresponding PRBGs with a short comparison regarding area resources and timing performance of the FPGA implementations. The FPGA implementation results (in terms of FPGA resources, frequency, throughput, and power consumptions) and security status of linear and nonlinear PRNGs are detailed in Sect. 3. This article ends with a conclusion section that summarizes the review.

2 Classification of the RNGs

RNGs are categorized mainly into two families: true random number generators (TRNGs) and pseudorandom number generators (PRNGs). In TRNGs, the physical process like jitter or thermal noise is used to generate random numbers. Therefore, the TRNGs cannot be used in the encryption/decryption process because they cannot be able to generate the same sequences corresponding to ciphering and deciphering operations [7]. For this problem, there is only one possible solution is generated sequences from TRNGs stored in memory. Additionally, TRNGs also suffer from a low-throughput rate, therefore they cannot be used in high-speed applications.

In general, there are two types of PRNG: (1) linear and (2) nonlinear PRNG. The linear system-based PRNG, linear feedback shift registers (LFSRs) [8, 9], and linear congruential generators (LCGs) [6, 10–17] are used for generating pseudorandom number sequences. In nonlinear PRNGs, the nonlinear output function or nonlinear transition function is used to convert the linear system into a nonlinear [18–30].

2.1 Nonlinear PRNG

In nonlinear PRNGs, the nonlinear output function or nonlinear transition function is used to convert the linear system into a nonlinear. Various PRNGs based on nonlinear system are used in the cryptography for their good randomness properties. Nonlinear dynamical systems consist of simple mathematical equations that can exhibit chaos behavior. So, the cryptographic properties of generated random sequences from the chaotic map are very crucial for the security of encryption algorithms. Chaotic systems generate a pseudorandom sequence, which can be applied in designing cryptographic keys to get their valuable characteristics like random behavior, sensitivity to the initial conditions, and ergodicity.

Mathematically, a hyperchaotic system can be defined as a chaotic system with two or more than two positive Lyapunov exponents. Its dynamic behavior is expended in more than two directions. So, the hyperchaotic attractor has more complex dynamic behaviors as compared to a chaotic system. The expansion of this dynamic behavior happens at the same time in two or more than two directions that make the hyperchaotic system, which shows better performance in many chaos-based applications including technological applications, than chaotic systems. Nowadays, hyperchaos has attracted attention from various scientific and engineering communities. So, the application of hyperchaos is becoming more popular in the field of chaos-based cryptography. Though, the well-known disadvantage of ordinary chaotic attractors for topological applications possesses only a single positive Lyapunov exponent (LE), hence its degree of disorder is not high as compare to hyperchaotic systems.

The recent literature of FPGA-based PRNG using chaotic and hyperchaotic attractors is discussed. The FPGA implementation of six different multiplierless chaotic PRNGs using Chua, Lorenz, Rössler, and the other three systems has been done

in [7]. To increase the randomness as well as prevent the digital chaotic system to fall into short-period orbits of the generated sequences, a PRNG based on the one-dimensional logistic map was implemented on FPGA [23]. In [24], Rezk et al. proposed an FPGA-based PRNG that is using the Lü and Lorenz chaotic attractors. The PRNG based on a hyperchaotic system with a self-shrinking perturbance generator was proposed by Yang Liu et al. in [25]. A new 4D hyperchaotic oscillator was proposed by wu et al. and analyzed its nonlinear dynamic behavior. Furthermore, an analog circuit of this system is implemented on a chip for some relevant engineering applications such as information encryption [26]. A hyperchaotic system and its qualitative properties were discussed by Rajagopal et al. in [27]. This system was also implemented in FPGA to prove that the system is hardware realizable.

2.2 Linear PRNG

The linear PRNG, LCGs, and LFSRs are used for generating pseudorandom number sequences. The linear PRNGs are suitable for high-speed and low-power applications in a hardware-based cryptosystem, but there is some limitation, i.e., limitation of state and a short period of generated bit sequences. To mitigate this limitation, many-related literature surveys are presents in detail thereafter.

2.2.1 LCG-Based PRNGs

The most popular random number generation method is linear congruential on modular arithmetic. An LCG is originated on the system of linear recurrence equations, which is defined as $x_{i+1} = [(a_1 \times x_i) + x_i + b_1] \mathrm{mod} 2^K$, where a (the "multiplier"), b (the "increment"), where $0 \leq a, b \leq 2^{k-1}$ are parameters of the generator. LCG is convenient for high-speed and low-power constraints, but it is not capable to generate more secure pseudorandom numbers. Because of this, many hardware implementations are proposed to increase the security and period.

The authors of [14] proposed the high-secure dual-CLCG algorithm-based PRBG. The dual-coupled-LCG blocks are used to design this architecture, and these blocks are designed by following recurrence relations:

$$x_{i+1} = [(a_1 \times x_i) + x_i + b_1] \mathrm{mod} 2^n \tag{1}$$

$$y_{i+1} = [(a_2 \times y_i) + y_i + b_2] \mathrm{mod} 2^n \tag{2}$$

$$p_{i+1} = [(a_3 \times p_i) + p_i + b_3] \mathrm{mod} 2^n \tag{3}$$

$$q_{i+1} = [(a_4 \times q_i) + q_i + b_4] \mathrm{mod} 2^n \tag{4}$$

$$B_i = \left\{ \begin{matrix} 1 \text{ if} x_{i+1} > y_{i+1} \\ 0 \text{ if} x_{i+1} < y_{i+1} \end{matrix} \right\} \tag{5}$$

$$C_i = \left\{ \begin{matrix} 1 \text{ if} p_{i+1} > q_{i+1} \\ 0 \text{ if} p_{i+1} < q_{i+1} \end{matrix} \right\} \tag{6}$$

$$z_i = B_i, if C_i = 0 \tag{7}$$

Here, constant parameters $(a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4)$ and initial seeds (x_0, y_0, p_0, q_0) are used in recurrence relation as given in corresponding Eqs. (1)–(4). The comparator output, i.e., B_i and C_i is given by Eqs. (5) and (6). The random bit sequences (z_i) are given by Eq. (7).

Authors of [15] optimized the implementation of the dual-CLCG algorithm [14] that involves arithmetic operations such as multiplication. In this architecture, the author uses the logical left shifting rather than multiplication operation, which reduces the hardware complexity of dual-coupling of LCG [14]. Therefore, Eqs. (1)–(7) can be rewritten as

$$x_{i+1} = \left[\left(2^{r1} \times x_i\right) + x_i + b_1\right] \text{mod} 2^n \tag{8}$$

$$y_{i+1} = \left[\left(2^{r2} \times y_i\right) + y_i + b_2\right] \text{mod} 2^n \tag{9}$$

$$p_{i+1} = \left[\left(2^{r3} \times p_i\right) + p_i + b_3\right] \text{mod} 2^n \tag{10}$$

$$q_{i+1} = \left[\left(2^{r4} \times q_i\right) + q_i + b_4\right] \text{mod} 2^n \tag{11}$$

$$B_i = \left\{ \begin{matrix} 1 \text{ if} x_{i+1} > y_{i+1} \\ 0 \text{ if} x_{i+1} < y_{i+1} \end{matrix} \right\} \tag{12}$$

$$C_i = \left\{ \begin{matrix} 1 \text{ if} p_{i+1} > q_{i+1} \\ 0 \text{ if} p_{i+1} < q_{i+1} \end{matrix} \right\} \tag{13}$$

$$z_i = B_i \oplus C_i \tag{14}$$

Gupta and Chauhan of [6] further optimized the implementation of the dual-CLCG algorithm [15], in which LCG blocks are designed using 2-operands modulo adder instead of 3-operands modulo adder. So, the nth bit of final addition will be calculated by XOR between nth bit of 2-operands modulo adder's output, i.e., $(S_{x1i}[n-1], S_{y1i}i[n-1], S_{p1i}[n-1] \text{and} S_{q1i}[n-1])$ and nth bit of the shifted value of variables $(x_i, y_i, p_i \text{and} q_i)$, i.e., $(x_i[0], y_i[0], p_i[0] \text{ and} q_i[0])$ corresponding to each LCG block and given by $\left(S_{x2i}, S_{y2i}, S_{p2i} \text{and} S_{q2i}\right)$ as shown in Eqs. (15)–(18). The values $(S_{x2i}, S_{x1i}[n-2:0])$, $\left(S_{y2i}, S_{y1i}[n-2:0]\right)$, $\left(S_{p2i}, S_{p1i}[n-2:0]\right)$, and $\left(S_{q2i}, S_{q1i}[n-2:0]\right)$ are stored in four registers corresponding to each LCG

block that hold the value for further processing. These register values are assigned to the next iterative values as given in Eqs. (19)–(22).

$$S_{x1i}[n-1:0] = b_1[n-1:0] + x_i[n-1:0],\ S_{x2i} = S_{x1i}[n-1] \oplus x_i[0] \quad (15)$$

$$S_{y1i}[n-1:0] = b_2[n-1:0] + y_i[n-1:0],\ S_{y2i} = S_{y1i}[n-1] \oplus y_i[0] \quad (16)$$

$$S_{p1i}[n-1:0] = b_3[n-1:0] + p_i[n-1:0],\ S_{p2i} = S_{p1i}[n-1] \oplus p_i[0] \quad (17)$$

$$S_{q1i}[n-1:0] = b_4[n-1:0] + q_i[n-1:0],\ S_{q2i} = S_{q1i}[n-1] \oplus q_i[0] \quad (18)$$

$$x_{i+1}[n-1:0] = \{S_{x2i}, S_{x1i}[n-2:0]\} \quad (19)$$

$$y_{i+1}[n-1:0] = \left\{S_{y2i}, S_{y1i}[n-2:0]\right\} \quad (20)$$

$$p_{i+1}[n-1:0] = \left\{S_{p2i}, S_{p1i}[n-2:0]\right\} \quad (21)$$

$$q_{i+1}[n-1:0] = \left\{S_{q2i}, S_{q1i}[n-2:0]\right\} \quad (22)$$

Authors of [16] proposed the high-secure PRBG architecture based on variable-input coupled-LCG. This architecture is designed using the coupling of LCG and input seeds of these LCG blocks are change by another two LCG blocks in each iteration. Each LCG block is defining by recurrence relations. It is given by Eqs. (23)–(26). The Eqs. (2) and (24) are named as variable-input linear congruential generators, were, the variables p_i and q_i are attained from two different LCGs recurrence relations as mentioned in Eqs. (25) and (26). The random bit sequence Z_i is obtained from the inequality condition, which is given in Eq. (27). Here, x_0, y_0, p_0, and q_0 are the initialization values corresponding to each recurrence equation. The b_1 and b_2 are the constant parameter of LCG blocks, as shown in Eqs. (25) and (26) (Figs. 1, 2, 3, and 4).

$$x_{i+1} = \left[\left(2^{r1} \times x_i\right) + x_i + p_i\right] \mathrm{mod} 2^n \equiv f_1(x_i, p_i) \mathrm{mod}\ 2^n \quad (23)$$

$$y_{i+1} = \left[\left(2^{r2} \times y_i\right) + y_i + q_i\right] \mathrm{mod} 2^n \equiv f_2(y_i, q_i) \mathrm{mod}\ 2^n \quad (24)$$

$$p_{i+1} = \left[\left(2^{r3} \times p_i\right) + p_i + b_1\right] \mathrm{mod} 2^n \quad (25)$$

$$q_{i+1} = \left[\left(2^{r4} \times q_i\right) + q_i + b_2\right] \mathrm{mod} 2^n \quad (26)$$

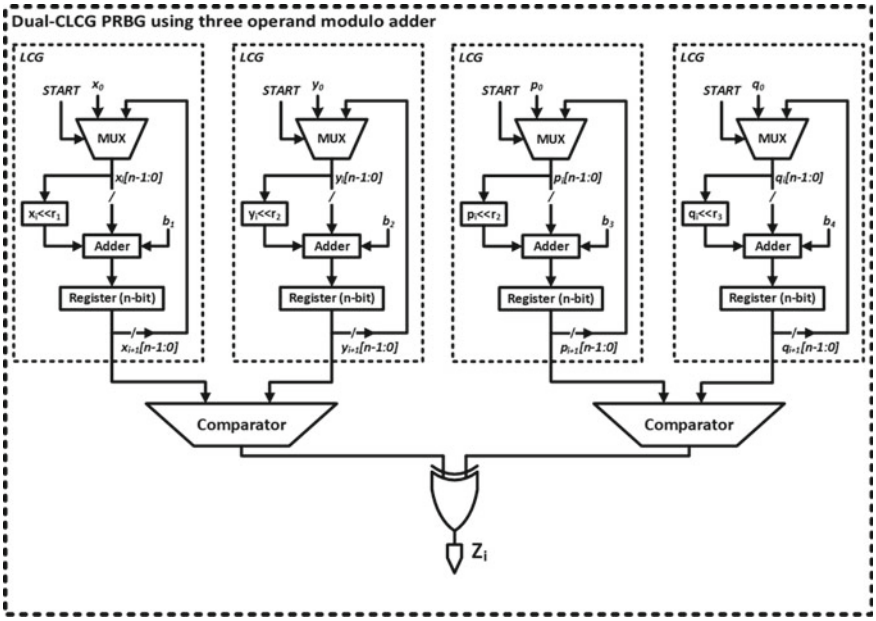

Fig. 1 VLSI architecture of dual-CLCG-based PRBG using 3-operands modulo adder [15]

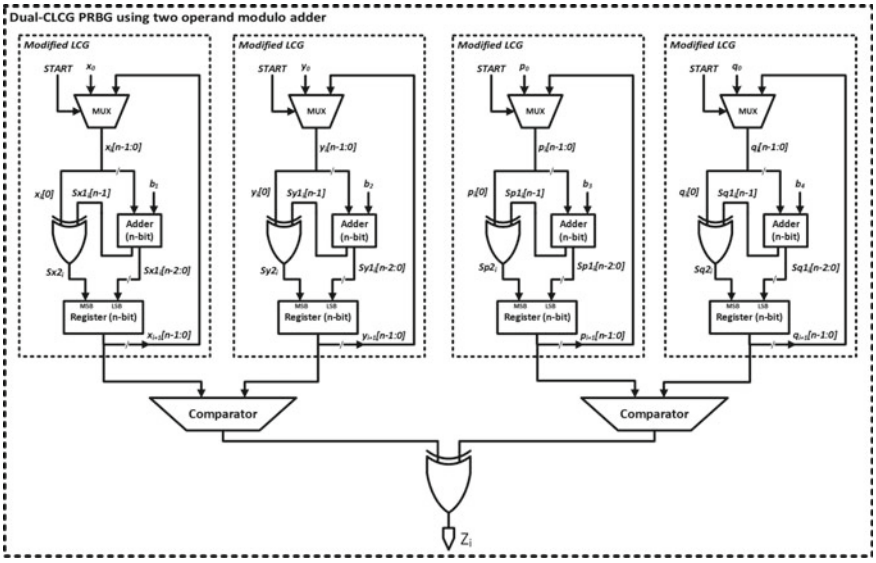

Fig. 2 PRBG architecture of dual-CLCG using 2-operands modulo adder [6]

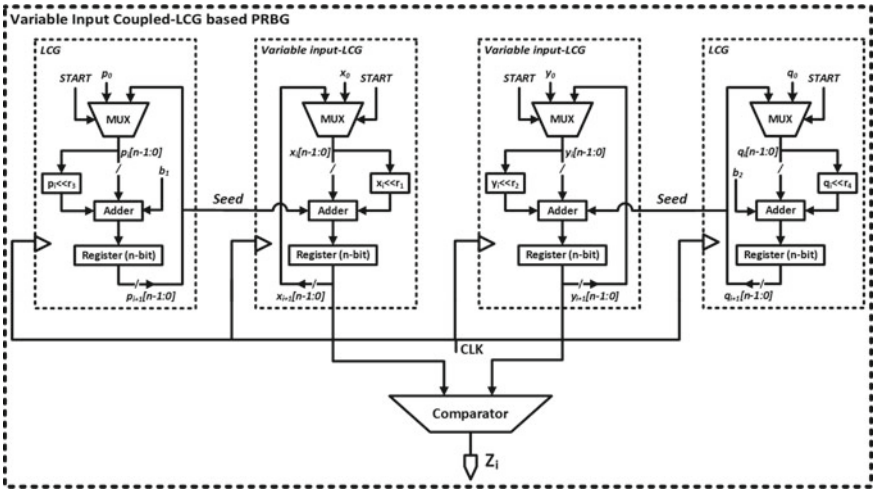

Fig. 3 PRBG architecture is based on the variable-input coupled-LCG [16]

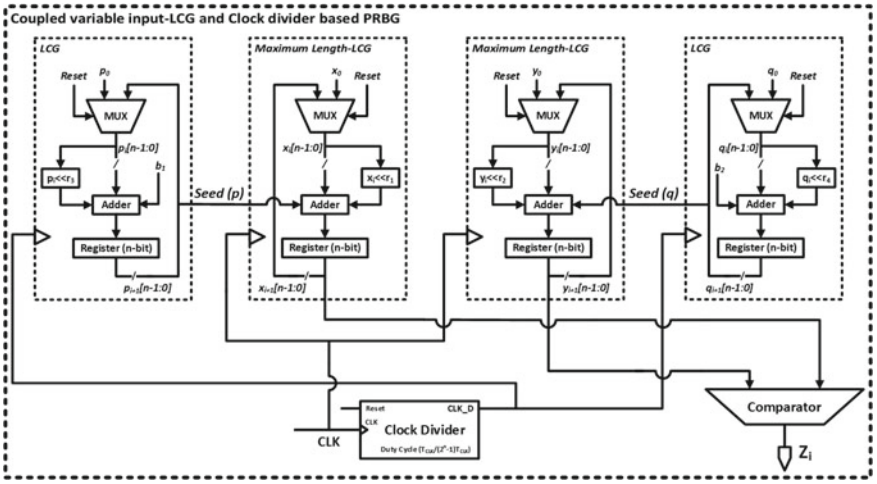

Fig. 4 PRBG architecture is based on the variable-input coupled-LCG and clock divider [17]

$$Z_i = \begin{cases} 1, \text{ if } x_{i+1} > y_{i+1} \\ 0, \text{ otherwise} \end{cases} \tag{27}$$

Gupta and Chauhan of [17] further improve the period length, security, and hardware performance of variable-input coupled-LCG architecture [16]. It is designed using several values of seed, i.e., $p(p_0, p_1, \ldots, p_{2^n-2}, p_{2^n-1})$ and $q(q_0, q_1, \ldots, q_{2^n-2}, q_{2^n-1})$ change periodically instead of changing in every iteration. Benefits of these techniques, the sequence of 2^{2n} maximum elements, i.e.,

$\{x_i\}$ is generated by periodically changing the value of p. In this method, the first 2^n elements are obtained by $x_{i+1} = f_1(x_i, p_0) \bmod m$, the next 2^n elements are obtained as $x_{i+1} = f_1(x_i, p_1) \bmod m$, and so on. After using all values of p, it generates 2^{2n} total elements. All value of p (from initial value) is reused to circularly continuing the process. Similarly, the sequence of 2^{2n} elements, i.e., $\{y_i\}$ is generated by periodically changing the value of q. With this method, the first 2^n elements are obtained by $y_{i+1} = f_1(y_i, q_0) \bmod m$, the next 2^n elements are obtained as $y_{i+1} = f_1(y_i, q_1) \bmod m$, and so on. After using all values of q, it generates 2^{2n} total elements. Now, this sequence of 2^{2n} elements $\{x_i\}$ and $\{y_i\}$ is computed by inequality condition to generate pseudorandom bits sequence of 2^{2n} period, without paying extra resources.

3 Result and Discussion

The performance parameters in terms of FPGA resources (i.e., number of flip-flops (FFs), look-up-table (LUT), slices, and DSP blocks), timing performance (critical path delay, frequency, and bit-rate), power consumption per unit frequency, and security strength are presented in Table 1.

Let us start to conclude the results obtained with different linear and nonlinear PRNGs, as demonstrated in Table 1. It appears demonstrated that the linear system-based PRNGs have the lowest area resources, high throughput, and less power consumable while maintaining the same security strength as compared with the nonlinear system (chaotic and hyperchaotic ordinary differential equations)-based PRNG approaches. These advantages leading the utility of linear system-based PRNG in lightweight IoT enable devices for cryptographic applications, i.e., better and more recommended schemes of PRNG.

The performance parameters of PRBGs belonging to the LCGs category are demonstrated in Table 1. The one that is based on the dual-coupled-LCG using two-operands modulo adder [6] has the lowest area occupation and high throughput. However, some other LCGs PRNGs can be presented as good competitors, namely (1) the variable-input coupled-LCG [16], (2) variable-input coupled-LCG with clock divider [17]-based PRBGs. Regarding the FPGA resources, throughput, the period length of the bit sequence, and security strength, the LCG-PRBG-based on variable-input coupled-LCG with clock divider [17] outperforms other linear PRNGs. In a conclusion, if we compare linear PRNGs to other PRNGs, it can play an important role in high-speed uses due to their parallel and rapidity generation.

4 Conclusion

This manuscript provided a widespread report on recent development in the FPGA implementation of RNGs. We have first recalled the different types of RNGs and

Table 1 Table captions should be placed above the tables

Methods	Size (bits)	FPGA	Total FFs	No. of slices	No. of LUTs	DSP blocks	Maximum frequency (MHz)	Clock period (T_{CLK}) (ns)	Power/freq. (mW/MHz)	NIST status
CLCG [14]	32	Virtex 5	64		311	–	219.78	4.550	0.146	Pass
		Virtex 7	64		294	–	282.64	3.54	0.084	
Dual-CLCG [15]		Virtex 5	128		603	–	219.78	4.550	0.228	Pass
		Virtex 7	128		571	–	282.64	3.54	0.146	
Modified dual-CLCG [6]		Spartan 3E	133	236	453	–	183.051	5.463	–	Pass
VCLCG [16]		Virtex 5	128		480	–	219.35	4.559	0.193	Pass
		Virtex 7	128		440	–	282.64	3.54	0.130	
VCLCG and clock divider [17]	16	Virtex 5	81		304	–	341.384	2.929	7.38	Pass
		Virtex 7	81		243	–	441.014	2.267	2.46	
Reconfigurable chaotic [24]	32	Virtex 5	96	100	276	8	78.149	12.796	–	Pass

discuss their properties in terms of hardware complexity, performance, and security strength. Thereafter, deeply investigate the nonlinear and linear system-based PRNG. Then, a large review of the FPGA-based PRNGs using LCGs systems is presented. For each type of PRNG, a hardware analysis regarding FPGA resources, timing performance, and security strength has been provided. Each RNG technique is discussed in detail. Finally, the performance parameter of FPGA-based PRNGs in terms of FPGA resources, frequency, throughput, power consumption, security strength, and weaknesses is presented.

References

1. Hossain MS, Kong Y, Saeedi E, Vayalil NC (2017) High-performance elliptic curve cryptography processor over NIST prime fields. IET Comput Digit Tech
2. Imran M, Rashid M, Jafri AR, Kashif M (2019) Throughput/area optimised pipelined architecture for elliptic curve crypto processor. IET Comput Digit Tech
3. Canivet G, Maistri P, Leveugle R, Clédière J, Valette F, Renaudin M (2011) Glitch and laser fault attacks onto a secure AES implementation on a SRAM-Based FPGA. J Cryptol 24:247–268
4. Beyne T (2020) Block cipher invariants as eigenvectors of correlation matrices. J Cryptol
5. Alaba FA, Othman M, Hashem IAT, Alotaibi F (2017) Internet of Things security: a survey. J Netw Comput Appl
6. Gupta MD, Chauhan RK (2020) Efficient hardware implementation of pseudo-random bit generator using dual-CLCG method. J Circuits, Syst Comput
7. Rezk AA, Madian AH, Radwan AG, Soliman AM (2020) Multiplierless chaotic Pseudo random number generators. AEU—Int J Electron Commun
8. Babu AS, Anand B (2020) Modified dynamic current mode logic based LFSR for low power applications. Microprocess Microsyst 72:102945
9. Singh SK, Gupta MD, Mani S, Chauhan RK (2020) Design of LFSR circuit based on high performance XOR gate. Int Conf Electr Electron Eng ICE3 2020
10. Gupta MD, Chauhan RK (2020) Design of modified dual-CLCG algorithm for pseudo-random bit generator. Int Conf Electr Electron Eng ICE3 2020
11. Gupta MD, Chauhan RK (2021) Improved VLSI architecture of dual-CLCG for pseudo-random bit generator. Lect Notes Electr Eng
12. Katti RS, Kavasseri RG (2008) Secure pseudo-random bit sequence generation using coupled linear congruential generators. Proc—IEEE Int Symp Circ Syst
13. Katti RS, Srinivasan SK (2009) Efficient hardware implementation of a new pseudo-random bit sequence generator. Proc—IEEE Int Symp Circ Syst
14. Katti RS, Kavasseri RG, Sai V (2010) Pseudorandom bit generation using coupled congruential generators. IEEE Trans Circuits Syst II Exp Briefs
15. Panda AK, Ray KC (2019) Modified dual-CLCG method and its VLSI architecture for pseudorandom bit generation. IEEE Trans Circuits Syst I Regul Pap 66:989–1002
16. Kumar Panda A, Chandra Ray K (2020) A coupled variable input LCG method and its VLSI architecture for pseudorandom bit generation. IEEE Trans Instrum Meas 69:1011–1019
17. Gupta MD, Chauhan RK (2021) Coupled variable input-LCG and clock divider based large period pseudo-random bit generator on FPGA. IET Compute Dig Tech. Accepted on 14 Feb 2021
18. Lambić D, Nikolić M (2019) New pseudo-random number generator based on improved discrete-space chaotic map. Filomat 33:2257–2268
19. Alhadawi HS, Zolkipli MF, Ismail SM, Lambić D (2019) Designing a pseudorandom bit generator based on LFSRs and a discrete chaotic map. Cryptologia 43:190–211

20. Alawida M, Samsudin A, Sen TJ (2020) Enhanced digital chaotic maps based on bit reversal with applications in random bit generators. Inf Sci (Ny) 512:1155–1169
21. Zhao Y, Gao C, Liu J, Dong S (2019) A self-perturbed pseudo-random sequence generator based on hyperchaos. Chaos, Solitons Fractals X 4:100023
22. Tolba MF, Elwakil AS, Orabi H, Elnawawy M, Aloul F, Sagahyroon A, Radwan AG (2020) FPGA implementation of a chaotic oscillator with odd/even symmetry and its application. Integration.
23. Garcia-Bosque M, Perez-Resa A, Sanchez-Azqueta C, Aldea C, Celma S (2019) Chaos-based bitwise dynamical pseudorandom number generator on FPGA. IEEE Trans Instrum Meas 68:291–293
24. Rezk AA, Madian AH, Radwan AG, Soliman AM (2019) Reconfigurable chaotic pseudo random number generator based on FPGA. AEU—Int J Electron Commun 98:174–180
25. Liu Y, Tong X (2016) Hyperchaotic system-based pseudorandom number generator. IET Inf Secur 10:433–441
26. Wu YL, Yang CH, Wu CH (2017) Chip implementation of a new hyperchaotic oscillator. Electron Lett 53:226–228
27. Rajagopal K, Guessas L, Vaidyanathan S, Karthikeyan A, Srinivasan A (2017) Dynamical analysis and FPGA implementation of a novel hyperchaotic system and its synchronization using adaptive sliding mode control and genetically optimized PID control. Math Probl Eng
28. Rajagopal K, Jafari S, Karthikeyan A, Srinivasan A, Ayele B (2018) Hyperchaotic memcapacitor oscillator with infinite equilibria and coexisting attractors. Circuits, Syst Signal Process 37:3702–3724
29. Bonny T (2020) Chaotic or hyper-chaotic oscillator? numerical solution, circuit design, MATLAB HDL-coder implementation, VHDL code, security analysis, and FPGA realization. Circuits, Syst Sig Proc
30. Gupta MD, Chauhan RK (2021) Secure image encryption scheme using 4D-Hyperchaotic systems based reconfigurable pseudo-random number generator and S-Box. Integr the VLSI J, Publisher: Elsevier, 81:137 –159. https://doi.org/10.1016/j.vlsi.2021.07.002
31. Di Patrizio SG, De Marcellis A, Palange E, Faccio M (2019) A true random number generator architecture based on a reduced number of FPGA primitives. AEU—Int J Electron Commun 105:15–23