Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 7, July : 2025

MODIFIED DUAL -CLCG METHOD AND ITS VLSI ARCHITECTURE FOR PSEUDORANDOM BIT GENERATION

[1]DHARAVATH TEJESWARI, [2]K. SNEHALATHA,
[1]M. Tech, VLSI System Design, Student,
Department of ECE,
Email: tejeshwaridharavath14@gmail.com,
[2]Assistant Professor(C), Department of ECE,
Email: snehakatha08@gmail.com,
JNTUH University College of Engineering Sultanpur,
Sangareddy, Telangana, India

ABSTRACT
Pseudorandom bit generator (PRBG) is an essential component for securing data during transmission and storage in various cryptography applications. Among popular existing PRBG methods such as linear feedback shift register (LFSR), linear congruential generator (LCG) coupled LCG (CLCG), and dual-coupled LCG (dual-CLCG), the latter proves to be more secure. This method relies on the inequality comparisons that lead to generating pseudorandom bit at a non-uniform time interval. Hence, a new architecture of the existing dual CLCG method is developed that generates pseudo random bit at uniform clock rate. However, this architecture experiences several drawbacks such as excessive memory usage and high-initial clock latency, and fails to achieve the maximum length sequence. Therefore, a new PRBG method called as "modified dual CLCG" and its very large-scale integration (VLSI) architecture are proposed in this paper to mitigate the aforesaid problems. The novel contribution of the proposed PRBG method is to generate pseudorandom bit at uniform clock rate with one initial clock delay and minimum hardware complexity.

Key Words: PRBG, DUAL-CLCG, VLSI, Verilog HDL.

I. INTRODUCTION

The generation of high-quality pseudorandom numbers is a fundamental requirement in various computing systems, especially those that rely heavily on security, simulation, or probabilistic computation. Pseudorandom number generators (PRNGs) are designed to produce sequences of numbers that mimic the properties of true random sequences, while being generated deterministically by algorithms. The quality of these sequences—measured in terms of uniformity, independence, and unpredictability—greatly influences the reliability, security, and correctness of systems in domains such as cryptography, wireless communications, scientific simulations, and embedded applications [1]. The Linear Congruential Generator (LCG), introduced by Lehmer in the 1950s, is a simple and widely used PRNG defined by a recursive relation involving modular arithmetic. While easy to implement, its linearity and short period lead to statistical weaknesses such as poor distribution and predictability. To overcome these issues, the Combined Linear Congruential Generator (CLCG) merges outputs from multiple LCGs with different parameters, improving randomness and extending the period. However,

CLCGs still present hardware challenges, requiring complex modular units and synchronization logic, which increase area and power consumption. With modern systems demanding low-power and high-speed designs—especially in WSNs, IoT, and embedded platforms—there is a growing need to optimize PRNGs like CLCG for efficient VLSI implementation [2].

OBJECTIVES

- To develop a Modified Dual Combined Linear Congruential Generator (Dual-CLCG) method for efficient pseudorandom bit generation.
- To design and implement a high-performance VLSI architecture for the proposed Dual-CLCG algorithm

II. LITERATURE SURVEY

In recent years, numerous researchers have proposed and analysed various pseudorandom number generation techniques, focusing on enhancing statistical quality, hardware efficiency, and suitability for cryptographic and embedded system applications.

A. REVIEW ON PRNG:

- **Linear Feedback Shift Register (LFSR):** Linear Feedback Shift Registers (LFSRs) are one of the simplest and most widely used PRNGs in hardware due to their low area and high-speed characteristics. Chandravanshi et al. [3] implemented an LFSR-based PRNG on FPGA for quantum cryptographic applications and demonstrated its suitability for low-cost systems. However, LFSRs suffer from inherent linearity, which makes them predictable and vulnerable to cryptanalysis. They also exhibit short linear spans and biased run-lengths, limiting their use in applications demanding high entropy. Hybrid LFSR-based systems have been proposed, but the additional complexity reduces their hardware efficiency [4].
- **Linear Congruential Generator (LCG):** LCGs are mathematically simple and defined by a recursive modular relation. Park and Miller [5] introduced the minimal standard LCG for portable software implementations, and it has been used in many programming libraries. However, Knuth [6] and later researchers identified its statistical flaws—such as poor distribution in higher dimensions and predictability—making it unsuitable for cryptographic purposes. The short period, especially when using small modulus values, and the visible lattice structures in output sequences further restrict LCGs in high-security environments.
- **Combined Linear Congruential Generator (CLCG):** To improve the statistical properties of LCGs, L'Ecuyer [7] introduced the Combined Linear Congruential Generator (CLCG), which combines two or more LCGs with different moduli and parameters. This method

significantly extends the period and enhances randomness, allowing the generated sequences to pass more rigorous tests such as TestU01. However, implementing CLCG in hardware requires multiple modular multipliers and synchronization logic, which increase area and power consumption [8]. Furthermore, despite the improved period, the structure remains linear and thus still susceptible to certain types of statistical attacks.

- **Dual-CLCG and Variants:** The Dual-CLCG approach builds on CLCG by generating two independent pseudorandom sequences and combining them, typically via comparator or modular arithmetic, to reduce correlation and increase entropy. Arif and Zia [9] implemented a dual-LCG structure on FPGA, achieving higher throughput and randomness. More recent work by Rajak et al. [10] proposed a Remodified Dual-CLCG (RDCLCG) with operand reuse and pipeline optimization, resulting in a low-latency, area-efficient hardware design that passed all NIST randomness tests.

B. LIMITATIONS OF THE EXISTING DUAL-CLCG METHOD

While the Dual-CLCG technique offers improved security over traditional LCG and CLCG methods [16], it exhibits several limitations, particularly concerning hardware efficiency and randomness performance:

- The architecture necessitates a dedicated control circuit and a significant number of flip-flops, acting as memory elements. Specifically, to generate k pseudorandom bits, k flip-flops are required. For an n-bit input seed, where $k \approx 2^{n-1}$, this results in substantial memory usage.
- The initial latency—from input seed to the generation of the first output—is determined by the number of clock cycles required to produce k random bits. In maximum-length scenarios, the design incurs an initial delay of 2^n clock cycles.
- The achievable maximum period of the Dual-CLCG output is influenced by the number of zeros in the control sequence (Ci), typically reaching around 2^{n-1} for randomly chosen n-bit seeds.
- The architecture relies on the assumption that the generated maximum-length sequence maintains a balance between the number of zeros and ones, which may not always hold true in practice.

III. PROPOSED METHODOLOGY

The proposed design addresses the limitations identified in the conventional Dual-CLCG method and its architecture, this work introduces a new pseudorandom bit generation (PRBG) approach. The proposed method, termed the Modified Dual-CLCG, enhances the original structure by replacing the core equation of the

existing Dual-CLCG with an improved version. The four equations used in the original Dual-CLCG framework are retained without modification. This refinement aims to improve randomness quality and hardware efficiency while maintaining the core operational principles of the original design.

The Modified Dual-CLCG improves both the randomness quality and hardware efficiency by leveraging a streamlined structure based on four coupled linear congruential equations. The design ensures a one-bit output per cycle, without skipping or buffering bits, thus reducing latency and memory overhead.

The method utilizes four congruential relations defined as follows:

$$x_{i+1} = (a_1 \cdot x_i + b_1) \mod 2^n$$
$$y_{i+1} = (a_2 \cdot y_i + b_2) \mod 2^n$$
$$p_{i+1} = (a_3 \cdot p_i + b_3) \mod 2^n$$
$$q_{i+1} = (a_4 \cdot q_i + b_4) \mod 2^n$$

Each of these equations operates on its own seed and set of constants, where a_1, a_2, a_3, a_4 and b_1, b_2, b_3, b_4 are carefully chosen constants, and x_0, y_0, p_0, q_0 are the respective initial seeds. The modulus 2^n ensures n-bit arithmetic suitable for digital hardware. The output bit Z_i is derived using a modulo-2 summation (XOR) of two intermediate binary signals B_i and Ci, which are computed based on comparison logic as:

$$B_i = 1, \text{ if } x_{i+1} > y_{i+1}; \text{ else } B_i = 0$$
$$C_i = 1, \text{ if } p_{i+1} > q_{i+1}; \text{ else } C_i = 0$$

Then, the final pseudorandom output bit is calculated as:

$$Z_i = B_i \oplus C_i$$

This architecture maintains the full-period behaviour as in the original Dual-CLCG method, achieving a maximum length of 2^n for an n-bit modulus. The XOR logic allows efficient realization in hardware, minimizing area and power usage, and enabling seamless VLSI integration.

IV. Proposed Architecture of the Modified Dual-CLCG Method

A VLSI hardware design for the modified dual-CLCG method, capable of generating one pseudorandom bit per clock cycle in a consistent manner. The top-level architecture is illustrated in Fig. 5, which is constructed by directly implementing four Linear Congruential Generator (LCG) equations, two comparison operations, and a single modulo-2 addition.

At the core of this architecture lies the LCG block, which is enclosed within a dotted rectangle in Fig. 1. Each LCG block is derived from its corresponding LCG equation, as discussed earlier in Section II-A. Rather than using hardware-intensive multiplication, the LCG block employs a logical left shift operation and a three-operand addition, enabling it to produce an n-bit pseudorandom output at every

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 7, July : 2025

clock cycle efficiently. The complete architecture includes four such LCG blocks that independently calculate the sequences x_i+1, y_i+1, p_i+1 and q_i+1 using initial seed inputs x_0, y_0, p_0, and q_0, respectively. The next stage of the architecture consists of two n-bit comparators, each of which compares the corresponding LCG outputs: one comparator compares x_i+1 with y_i+1 and the other compares p_i+1 with q_i+1. These comparators output single bits B_i and C_i representing the comparison results. Finally, the outputs of the two comparators are fed into a single XOR gate, which performs a modulo-2 addition (i.e., bitwise XOR) to generate the final pseudorandom bit Z_i on every clock cycle. Unlike the earlier dual-CLCG architecture that relied on more complex control structures such as memory buffers, data controllers, and flip-flop banks, this optimized architecture requires only a minimalistic XOR logic gate at the output stage. This results in a more area- and power-efficient design.

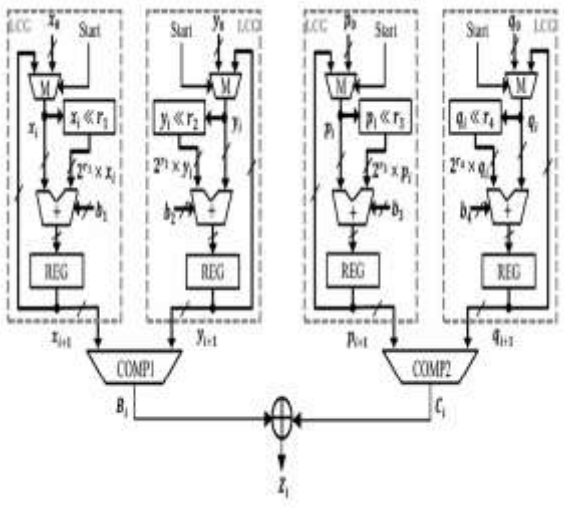

Figure.1 Architecture of Modified Dual-CLCG

V. CONSTRAINTS

- The 8-bit configuration of the proposed PRBG was constructed using the constants:
  a1 = 5, b1 = 1,
  a2 = 5, b2 = 3,
  a3 = 9, b3 = 141,
  a4 = 33, b4 = 79,
  with a modulus m = 2^8, and initialized with seeds (x0, y0, p0, q0) = (1, 2, 14, 3). This setup generates a pseudorandom output bit sequence such as:
  Zi = [0, 1, 0, 0, 1, 1, 0, 1, 0, 0,...]. The generator produces up to $2^8 = 256$ bits before repeating the sequence. One pseudorandom bit is generated per clock cycle, following an initial clock latency.
- The 32-bit architecture used constants:
  a1 = 65, b1 = 117,
  a2 = 16385, b2 = 221,
  a3 = 4097, b3 = 21359,
  a4 = 1025, b4 = 533,
  with modulus m = 2^32, and initial seeds (x0, y0, p0, q0) = (5183, 91356, 39771, 7392). This configuration produced a pseudorandom sequence such as:
  Zi = [1, 0, 0, 1, 1, 0, 1, 0, 1, 0,...]
  A total of $2^{32} = 4{,}294{,}967{,}296$ bits are generated before the sequence repeats.

VI. RESULTS:

The Modified Dual-CLCG PRBG was simulated using Verilog HDL on EDA Playground with Synopsys VCS over. The output bitstream, generated by XORing two

independent LCG sequences and extracting the LSB, showed a balanced distribution of 1s and 0s with high variability and no detectable patterns. Sample outputs confirmed strong randomness. Waveforms validated correct LCG operation, XOR logic, and FSM control with no timing violations.

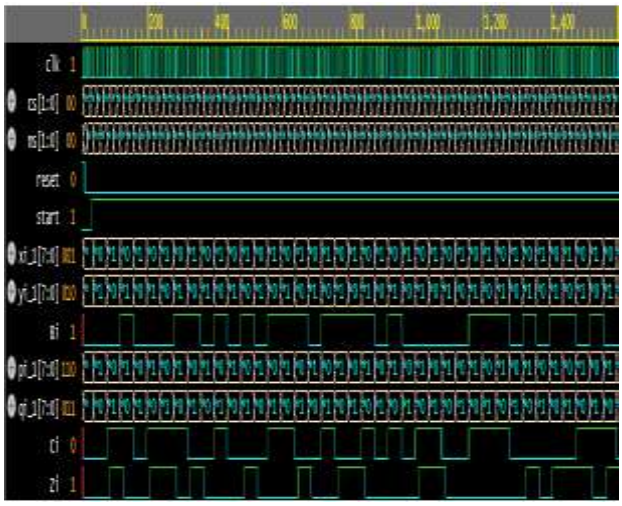

Figure. 2 n=8, Result-1

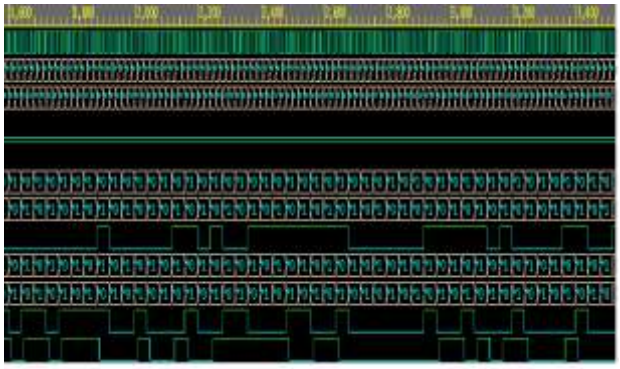

Figure. 3 n=8, Result-2

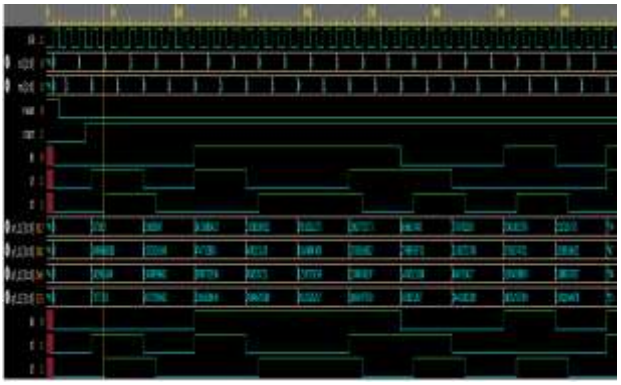

Figure. 4 n=32, Result-1

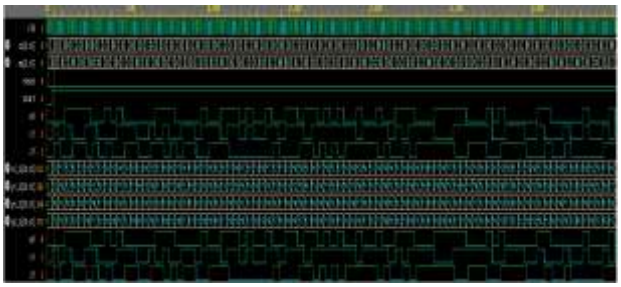

Figure. 5 n=32, Result-2

VII.CONCLUSION

In conclusion, the proposed method successfully implements the Modified Dual-CLCG method for efficient pseudorandom bit generation. By combining two LCGs with enhanced randomness, the method reduces predictability while maintaining low hardware complexity and high-speed performance. Simulation results confirm strong randomness, validating the design's suitability for secure, embedded, and real-time applications. The work effectively bridges algorithmic design and hardware implementation for modern digital systems.

VIII. FUTURE SCOPE

The Modified Dual-CLCG architecture offers a solid base for secure PRBG design and can be enhanced further. Key future directions include ASIC implementation for improved performance and power efficiency, integration with cryptographic algorithms like AES and RSA, and validation using standard randomness tests (e.g., NIST SP 800-22). The design can also be optimized for low-power embedded/IoT devices, scaled for higher bit-widths, and deployed on FPGA for real-time secure applications.

REFERENCES

[1]. W. Stallings, Cryptography and Network Security: Principles and Practice, 7th ed., Pearson, 2017.
[2]. S. B. Wicker and S. Kim, Fundamentals of Codes, Graphs, and Iterative Decoding, Springer, 2003.AES encryption and decryption core using Verilog," in ICCIDS, 2017.
[3]. P. Chandravanshi et al., "LFSR based RNG on low-cost FPGA for QKD applications," arXiv preprint arXiv:2307.16431, 2023.
[4]. B. A. Hameedi et al., "A PRNG based on hybrid LFSR and LCG algorithm," Iraqi Journal of Science, vol. 63, no. 5, pp. 2430–2443, 2022.
[5]. S. K. Park and K. W. Miller, "Random number generators: good ones are hard to find," Communications of the ACM, vol. 31, no. 10, pp. 1192–1201, 1988.
[6]. D. E. Knuth, The Art of Computer Programming, Vol. 2: Seminumerical Algorithms, 3rd ed., Addison-Wesley, 1997.
[7]. P. L'Ecuyer, "Efficient and portable combined random number generators," Communications of the ACM, vol. 31, no. 6, pp. 742–749, 1988.
[8]. P. L'Ecuyer and R. Simard, "TestU01: A C library for empirical testing of random number generators," ACM Transactions on Mathematical Software, vol. 33, no. 4, 2007.
[9]. M. Arif and K. Zia, "High-speed FPGA implementation of dual LCG-based PRNG," in Proc. Int. Conf. on Reconfigurable Computing and FPGAs, pp. 120–125, 2016.
[10]. P. K. Rajak, T. Mandal, and M. L. S. Sai Kumar, "Remodified Dual-CLCG method and its VLSI architecture for pseudorandom bit generation," SN Computer Science, vol. 5, article 423, 2024.
[11]. R. S. Katti, R. G. Kavasseri, and V. Sai, "Pseudorandom bit generation using coupled congruential generators," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 3, pp. 203–207, Mar. 2010.