International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 10, No. 2, July 2021, pp. 109~114
ISSN: 2089-4864, DOI: 10.11591/ijres.v10.i2.pp109-114

Co-simulation of linear congruential generator by using Xilinx system generator and MATLAB Simulink

Suneeta
Department of Electronics and Communication Engineering, Vemana Institute of Technology, affiliated to VTU Belgaum, Karnataka, India

Article Info

Article history:

Received May 12, 2020
Revised Aug 8, 2020
Accepted Jan 10, 2021

Keywords:

LCG
MATLAB
Simulation
VHDL
Xilinx

ABSTRACT

Arbitrary numerals are utilized in a wide range of uses. Genuine arbitrary numeral generators are moderate and costly for some applications while pseudo arbitrary numeral generators (RNG) do the trick for most applications. This paper fundamentally concentrates around the co-simulation of the linear congruential generator (LCG) model utilizing the Xilinx System generator and checking on Matlab Simulink. The design is obtained from the LCG calculation offered by Lehmer. Word lengths decrease strategy has been utilized to streamline the circuit. Simulation has been done effectively. The effective N bit LCG is structured and tried by utilizing demonstrating in MatLab Simulink. The Co-simulation of the model is done by utilizing the Xilinx system generator. This paper conducts an exhaustive search for the best arbitrary numeral generator in a full period linear congruential generator (LCG) with the largest prime numbers.

Corresponding Author:

Suneeta
Department of Electronics and Communication Engineering
Vemana Institute of Technology, Affiliated to VTU
No 1, Mahayogi Vemana Road, Koramangala, Bangalore 56003, India
Email: suneeta@vemanait.edu.in

1. INTRODUCTION

Arbitrary numerals have been utilized in everyday exercises from the time when occasions prior. These days, a little and modest child's puppet holding an arbitrary numeral based circuit in it. For instance, in a model like cell phones will buzz various kinds of tones by pressing the same key more than one time. A few arbitrary numerals of hypotheses have been presented over the most recent quite a few years. LCG (Linear congruential generator) that presented by Lehmer in 1954 is the ancient [1] and generally utilized pseudoarbitrary numeral generator (PNG) [2]. Park and Mill's operators recommend great bounds for LCG [3]. The recommendation is utilized in MatLab for producing similar kind of arbitrary numerals [4]. Numerous new arbitrary numeral generators were suggested and utilized in numerous suplications. Linear feedback shift register, Wichmann-Hill, Blum Blum Shub, Complementary multiply with carry, ISAAC (cipher), Inversive congruential generator, Lagged Fibonacci generator, Mersenne twister, Maximal periodic reciprocals, Naor-Reingold Pseudoarbitrary Function, Multiply-with-carry, Well Equidistributed Long-period Linear, RC4 PRGA, and Xor shift are few familier methods [5-8]. Equipment for creating arbitrary numerals accessible just as its calculation. The equipment has been utilized since 2008. LETech is the quickest among all hardware for generating arbitrary numerals, this method has been used since 2008 [9, 10]. Research for finding the reasonable algorithm of generating an arbitrary numeral is well building upfield as of recently. Numerous specialists utilize FPGA (field programmable gate arrays) for testing their views. At first, the algorithm of LCG joined with the Monte-Carlo method has been utilized for producing non-uniform arbitrary numerals using MatLab [11]. Later, the circuit of arbitrary numeral generator is tested and implemented on FPGA [12]. Here the subtractor block has been removed and presenting the novel structure implemented on MatLab Simulink with the help of Xilinx building blocks for LCG algorithm.

2. LINEAR CONGRUENTIAL GENERATOR

There is a common technique to generate an arbitrary numeral called the linear congruential generator. This knowledge was introduced by Lehmer according to a sequential formula in (1) [1].

$$X_{n+1} = (aX_n + c) mod\ m \tag{1}$$

where ‘a’ is multiplier, ‘m’ is modulus, ‘c’ is increment. Factors a, m, and c must be selected wisely to evade reappearance of like numerals before m [6-8]. The better result can be achieved by selecting c=0 is specified by Park & Miller [3]. The modulus value ‘m’ must be a largest prime number; multiplier ‘a’ will be a number must be less than m-1. The generated arbitrary numerals will be less than or equal to modulus m.

3. LCG CIRCUIT DESIGN

Common Circuit of LCG is shown in Figure 1 LCG activity in all (seed is neglected). It needs adder, multiplier, subtractor and comparator functional units. The multiplier is utilized to find preceding arbitrary value ‘x’ with ‘a’, perorm the addition by incrementing ‘c’. After that, use the value to find the modulus ‘m’. The numeral is reflected as an arbitrary numeral if that numeral is lesser or equal to ‘m’. If the numeral is greater than ‘m’ then it’s subtracted with ‘m’. The resultant of this is considered an arbitrary numeral.

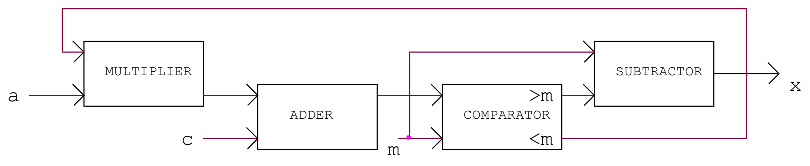

Figure 1. General block diagram of LCG

The block diagram represented in Figure 1 contains operations like addition, multiplication, addition, comparison and subtraction. To simplify the work process; the circuit is intended using the ‘word lengths’ lessening method that has recommended in [13-25]. Then comparator and subtractor blocks can be merged, as shown in Figure 2.

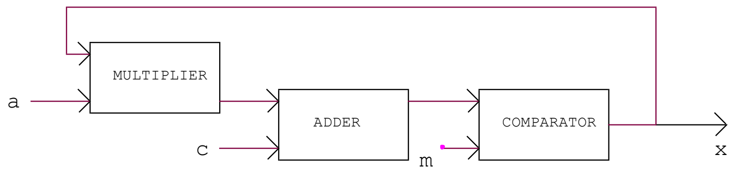

Figure 2. Modified block diagram of LCG

4. CO-SIMULATION OF THE MODEL USING SYSTEM GENARATOR

The fundamental objective is to carry out equipment co-recreation of the Simulink model utilizing an HDL coder for LCG. Figure 3, demonstrates the HDL coder, Xilinx blocks, and Simulink model for equipment coreproduction. The Xilinx Blockset is a public library for a piece, cycle genuine recreation and backups tradition capacities. The HDL coder utilizes Xilinx altered IP centers and creates a synthesizable code. Before ongoing execution of the LCG utilizing FPGA, the model testing of the plan is finished by utilizing a HDL coder. The model testing of the LCG is confirmed effectively by utilizing the VHDL code created from the HDL coder. The framework generator creates a piece record for FPGA and utilized VIRTEX II FPGA board for the execution reason.

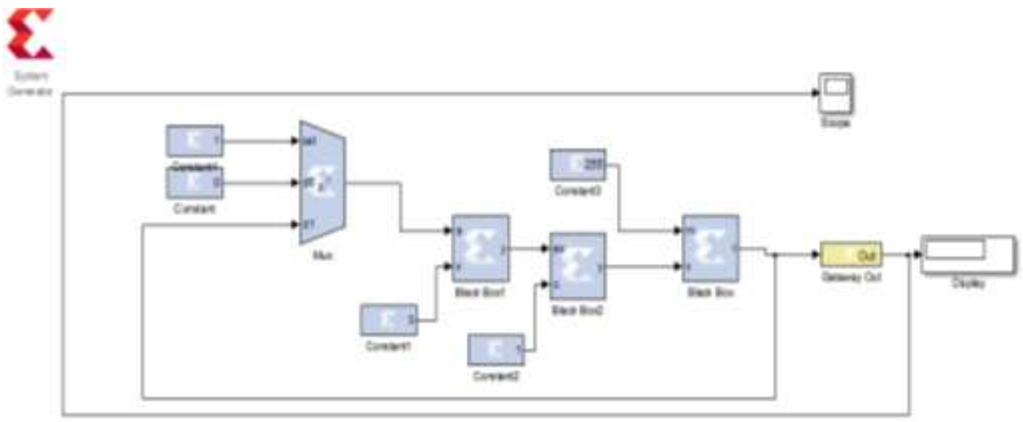

Figure 3. Co-simulation model

5. RESULTS OF HARDWARE CO SIMULATION

Design for PRNGs is based on LCG which can be constructed in such a way as to have optimal statistical and periodical properties using co-simulation. Simulation results are shown in Figure 4 and corresponding signal observed in Oscilloscope is shown in Figure 5 respectively. Simulation results of LCG using Xilinx. Co-Simulation results for 4-bit, 8-bit, 16-bit and 32-bit are shown in Figure 6, Figure 7, Figure 8 and Figure 9 respectively.

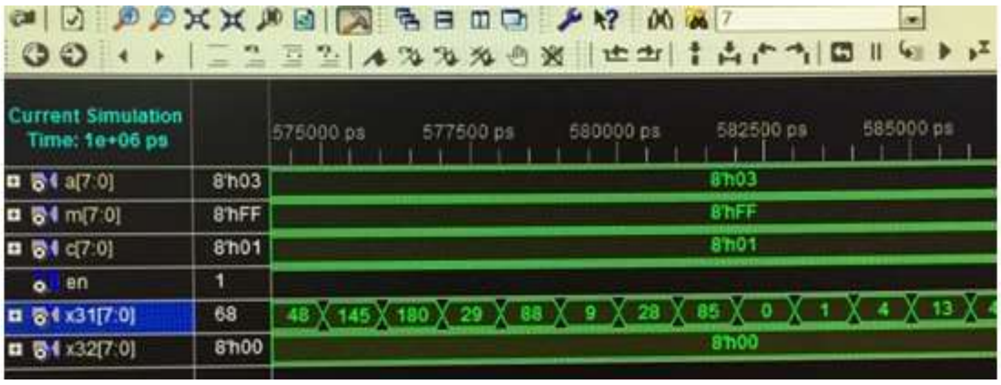

Figure 4. Simulation results for 8-bit LCG.

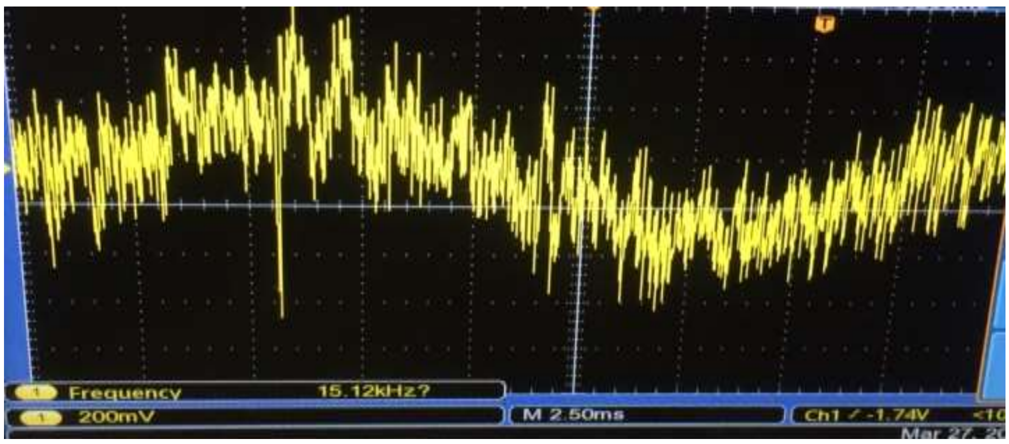

Figure 5. Generated signal from 8-bit LCG code

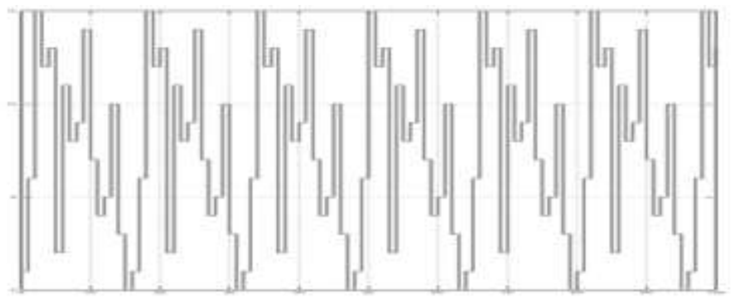
Figure 6. Co-Simulation results for 4-bit

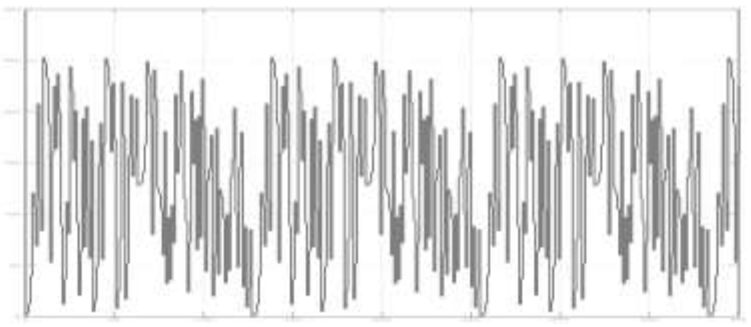
Figure 7. Co-Simulation results for 8-bit

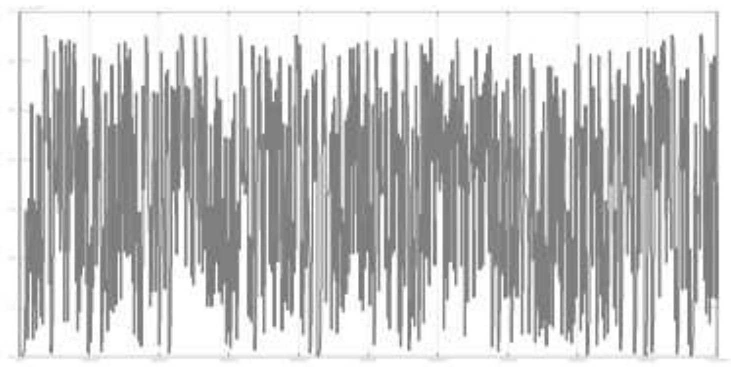
Figure 8. Co-Simulation results for 16-bit

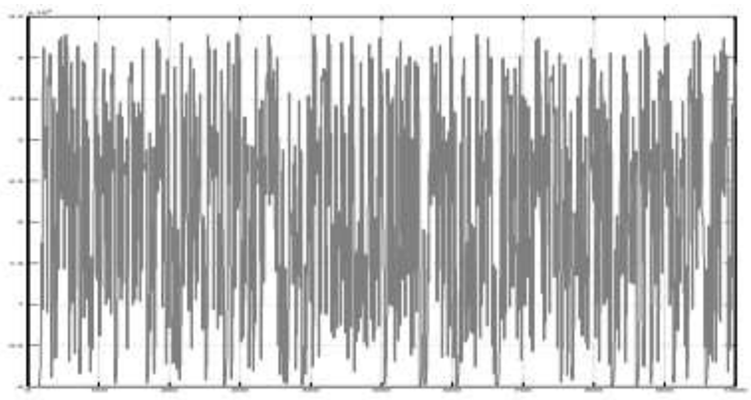
Figure 9. Co-Simulation results for 32-bit

6. CONCLUSION

The simulation results help us to verify the functionality of the model. The model is designed for the LCG, which works fine and performance is better with implementation of FPGA as well as on Co-simulation method. The results that can be obtained are close to the theoretical analysis of specifications chosen for the model. As the numeral of bits increases the complexity increases with Xilinx simulation. This problem is solved using Xilinx system generator based design implementation. It helps us to reduce the time and complexity for writing HDL programs for large data size. The process can be used for larger numeral signals with a higher value of m. Simulation and practical results of 8-bit LCG using HDL code was also recoreded. Co-Simulation results of 4-bit, 8-bit, 16-bit, and 32- bit is verified recorded and compared with HDL based results.

REFERENCES

[1] M. L. Juncosa, (D. H. Lehmer), *Random number generation on the BRL high speed computing machines*, Maryland, United States: Aberdeen Proving Ground, 1953.

[2] Hui-Chin Tang, “An analysis of linear congruential random number generators when multiplier restrictions exist,” *Europian Journal of Operational Research*, vol. 182, no. 2, pp. 820-828, 2007.

[3] S. K. Park and K. W. Miller, “Random number generators: good ones are hard to finnd,” *Communications of the ACM*, vol. 31, no. 10, pp. 1192-2001, 1988.

[4] C. Moler, *Numerical Computing with MATLAB*, SIAM, 2008

[5] David DiCarlo, “Random Number Generation: Types and Techniques,” Thesis, Liberty University, 2012

[6] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, CBMS-NSF Regional Conference Series in Applied Mathematics, Philadelphia, 1992.

[7] C. Dutang and D. Wuertz,” A note on Random number generation,” *Overview of Random Generation Algorithms*, Sep. 2009.

[8] Wolfram,” Random number generation,” Wolfram Language & System Documentation Center, 2008, [Online]. Available: https://reference.wolfram.com/language/tutorial/RandomNumberGeneration.html. [Accessed on May 5, 2021]

[9] Roger D. Peng. “Advanced statistical computing” https://bookdown.org/rdpeng/advstatcomp/random-number-generation.html [Accessed on 17 May, 2021].

[10] A. Yadav, “Design and analysis of digital true random number generator,” Thesis, Virginia Commonwealth University, 2013.

[11] Zulfikar, “Generating non uniform random numbers using residue and rejection methods,” *Journal Rekayasa Elektrika*, vol. 8, no. 2, pp. 58-61, 2009.

[12] Zulfikar, “FPGA implementations of uniform random number based on residue method,” *Journal Rekayasa Elektrika*, vol. 11, no. 1, pp. 15-18, 2014.

[13] Zulfikar, “Novel area optimization in FPGA implementation using efficient VHDL code,” *Journal Rekayasa Elektrika*, vol. 10, no. 2, pp. 61-66, 2012.

[14] V. Fischer, F. Bernard, “True random number generators in FPGAs,” *Security Trends for FPGAS*, pp 101-135, 2011, doi: 10.1007/978-94-007-1338-3_5.

[15] H. Fürst, H. Weier, S. Nauerth, D. G. Marangon, C. Kurtsiefer, and H. Weinfurter, “High speed optical quantum Random number generation,” *Optics Express*, vol. 18, no. 12, pp 13029-13037, 2010.

[16] Priyanka, I. Hussain, A. Khalique,” Random number generators and their applications: A review,” *International Journal of Research in Electronics & Computer Engineering*, vol. 7, no. 2, pp. 1777-1781, Jun 2019.

[17] F. James, L. Moneta, “Review of high-quality random number generators,” *Computing and Software for Big Science*, 2020, doi: 10.1007/s41781-019-0034-3.

[18] F. Yu, *et al.*, "Design and FPGA implementation of a pseudorandom number generator based on a four-wing memristive hyperchaotic system and Bernoulli map," in *IEEE Access*, vol. 7, pp. 181884-181898, 2019, doi: 10.1109/ACCESS.2019.2956573.

[19] A. A. Rezk, A. H. Madian, A. G. Radwan and A. M. Soliman, “Reconfigurable chaotic pseudorandom number generator based on FPGA,” *AEU-International Journal of Electronics and Communications*, vol. 98, pp. 174-180, 2019.

[20] R. S. Hasan, S. K. Tawfeeq, N. Q. Mohammed and A. I. Khaleel, “A true random number generator based on the photon arrival time registered in a coincidence window between two single-photon counting modules,” *Chinese Journal of Physics*, vol. 56, no. 1, pp. 385-391, 2018.

[21] M. Abutaleb, “A novel true random number generator based on QCA nanocomputing,” *Nano Communication Networks*, vol. 17, pp. 14-20, 2018.

[22] Y. F. Wang, “Research and design of true Random number generator,” M.S. thesis, Zhejiang University, 2010.

[23] I. Cicek, A. E. Pusane and G. Dundar, "A new dual entropy core true random number generator," *Analog Integrated Circuits and Signal Processing*, vol. 81, pp. 61-70, 2014, doi: 10.1007/s10470-014-0324-y.

[24] A. Sibidanov, "A revision of the subtract-with-borrow Random number generators," *Computer Physics Communications*, vol. 221, pp. 299-303, 2017, doi: 10.1016/j.cpc.2017.09.005.

[25] A. M. Law and M. D. Kelton, *Simulation modeling and analysis*, 3rd Ed., New York, USA: McGraw Hill Higher Education, 2000.

BIOGRAPHY OF AUTHOR

Dr. Suneeta received her doctorate degree from JNTUK, Kakinada, Andra Pradesh and B E, M. Tech degree from VTU; Belgaum, Karnataka. working as an Associate Professor in VEMANA.IT, Bangalore. Karnataka, India. Guided many undergraduate and post graduate students in VLSI and embedded field. She has number of papers to her credit with national & international journals/conferences and life member for ISTE and IAENG.