International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 04 | April - 2023 Impact Factor: 8.176 ISSN: 2582-3930

Design and FPGA implementation of Lorenz key generator for information security

Nandha kumar S
PG Student, Department of ECE
PSG College of Technology
Coimbatore, India
s.nandhakumar333@gmail.com

Dr. J. Ramesh
Professor,Department of ECE,
PSG college of Technology,
Coimbatore, India
jramesh60@yahoo.com

P. Vivek Karthick
Assistant Professor, Department of ECE,
SONA college of Technology,
Salem, India
vivekmalar@gmail.com

***Abstract*— In recent years, enormous amount of information obtained from digital media, are available through the Internet, satellites, mobiles and other sources making it possible to access these contents. Thus, protecting information from unauthorized access has become a major task for researchers as the usual encryption systems are unable to resist the evolution of hacker attacks. The advantages of Lorenz key is encryption is faster, very secure transmission and storage when compared to the other key. Many cryptosystems based on the generation of Lorenz key for hiding messages, using chaotic systems as a generator for those sequences have been gaining attention recently. The chaos is generated from a nonlinear dynamic system, known for its aperiodicity, randomness and sensitivity with respect to initial values and control parameters. This work presents an approach for a real-time FPGA implementation of a random key, based on Lorenz's chaotic generator for information security. At first, the Lorenz chaotic oscillator model is designed using MATLAB code. The performance is successfully verified, and the model is constructed using MATLAB-Simulink model. Simulink models in MATLAB provide a modelling environment that is well suited for hardware design. Then, the model is converted to Xilinx System Generator model. The Xilinx system generator technology is used for the conception of the Lorenz chaotic system and for generating the code. This code is then dumped to configure the FPGA.**

Keywords*— *Chaos, Lorenz, MATLAB, Simulink, Xilinx System Generator, FPGA

I. INTRODUCTION

During the past decades, there has been huge interest worldwide in the possibility of using chaos in communication systems. Chaos occurs in nonlinear systems that are sensitive to the initial conditions. The behaviour of these systems appears to be random, almost like the behaviour of a system strongly influenced by random noise, even though these systems are deterministic. Chaotic signals are desirable for secure communication because of its broadband nature, sensitive to initial conditions, aperiodic and noise like nature. Many methods had been proposed to realize secure communication system using chaotic signals says Pecora-Carroll [1-2].

In recent years, enormous amount of information obtained from digital media, are available through the Internet, satellites, mobiles and other sources making it possible to access these contents. Thus, protecting information from unauthorized access has become a major task for researchers as the usual encryption systems are unable to resist the evolution of hacker attacks.

There are two types of approaches when using chaotic dynamics in cryptography. The first one used as key streams to mask the plaintext in a manifold of ways. The next one is used as initial state and the cipher text follows from the orbit being generated**.** A new method to implement any type of a chaotic generator was introduced by using Field Programmable Gate Array (FPGA) in the paper given by M.A. Aseeri, M.I. Sobhi and P. Lee [5, 7]. The aim of that method was to increase the frequency of the chaotic signals. A. Abel and W. Schwartz [6]. Starting from general demands for communications, the general communication system structure was introduced by them.

Chaos occurs in nonlinear systems that are sensitive to the initial conditions. The behaviour of these systems appears to be random, almost like the behaviour of a system strongly influenced by random noise, even though these systems are deterministic. These facts were clearly explained by Robert C.Hilborn [9] and M.Lakshmanan, S.Rajasekar [10] in their work.

This work is organized as follows. In section II Flow diagram. In section III Lorenz equations system are explained. Implementation of Lorenz System Using MATLAB code are given in section IV. Implementation of Lorenz System Using Xilinx System Generator are given in section V. In the last section, the FPGA implementation of Lorenz chaotic generator and its simulation results are shown. Finally, the results are summarized in the conclusion.

II. LORENZ EQUATIONS SYSTEM

The Lorenz system, invented for Edward N. Lorenz, The Lorenz system is an example of a non-linear dynamic system these systems are corresponding to the long-term behaviour of the Lorenz system. The Lorenz equation is based on the fundamental Navier-Stokes equation for fluid. The fluid motion and the resulting temperature difference can be expressed in terms of three variables called X(t), Y(t), Z(t); where X is related to time-dependence of the so-called fluid stream function. Y is proportional to the temperature difference between the rising and falling parts of the fluid. and Z is proportional to the deviation from temperature linearity as a function of vertical position. The graphical representation of such system shows how the state of a dynamical system.

The three equations that govern the Lorenz system following:

$$\frac{dx}{dt} = \sigma * (y - x) \quad (1)$$

$$\frac{dy}{dt} = ((r * x) - (y) - (x * z)) \quad (2)$$

$$\frac{dz}{dt} = ((x * y) - (\beta * z)) \quad (3)$$

where, σ, β and r called the control parameters. All control parameters should be greater than zero (σ, r, β > 0), but usually σ = 10, β = 8/3 and r is varied. The system exhibits chaotic behaviour for r = 28. To resolving this Lorenz system.

III. Flow Diagram

At very first, the Lorenz chaotic oscillator model is designed using MATLAB code. Once code is successfully verified after that the model is constructed using MATLAB-Simulink model. Simulink models in MATLAB to provide a modelling environment that is well suited to hardware design and later it is converted to the Xilinx System Generator model. Xilinx system generator model is used to convert Simulink mode into HDL design. The Xilinx System Generator bridges the gap between conceptual architectural design and the actual implementation in a Xilinx FPGA. Xilinx ISE design suite software is used to allows us to generate the FPGA's programming file. Fig 1 shows implementation flow diagram of this work.

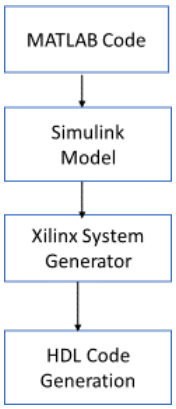

Fig 1: Flow diagram of this work

IV. Implementation of Lorenz System Using MATLAB Code

The generators are first represented by a set of nonlinear equations and a system-based model is developed to represent the equations directly. The systems are first represented by a set of nonlinear equations and a system-based model is developed to represent the equations directly. Here first a system model from state equation was developed using MATLAB software. Fig 2 represent a Lorenz equation model output. Here that output is non-repeated pattern. So, this non-repeated pattern is very use full for the high security purposes. Because these patterns are not easily predictable and unbreakable.

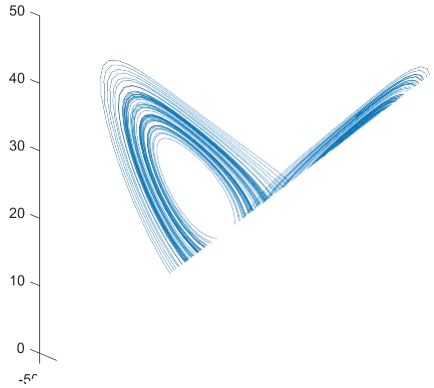

Fig 2: Lorenz equation model MATLAB code output

V. Implementation of Lorenz System Using MATLAB Simulink

Once MATLAB code is successfully verified after that the model is constructed using MATLAB-Simulink model. Here first a system model from state equation was developed using MATLAB/Simulink software. Then the model was simulated to adjust the required frequency. Next step was to convert all models by using Xilinx System Generator block set. In the Lorenz three equations the value σ =10, r=28 and b=8/3 were submitted. Lorenz's equations have two nonlinearities responsible for the chaotic behavior. The products *xz* and *xy* that are performed by two multipliers. The equations are simulated using the Simulink block diagram presented by Fig 3. and Fig 4. The first parts relate to the phase plane (x, y) and (y, z) and final parts relate to the phase plane (x, z).

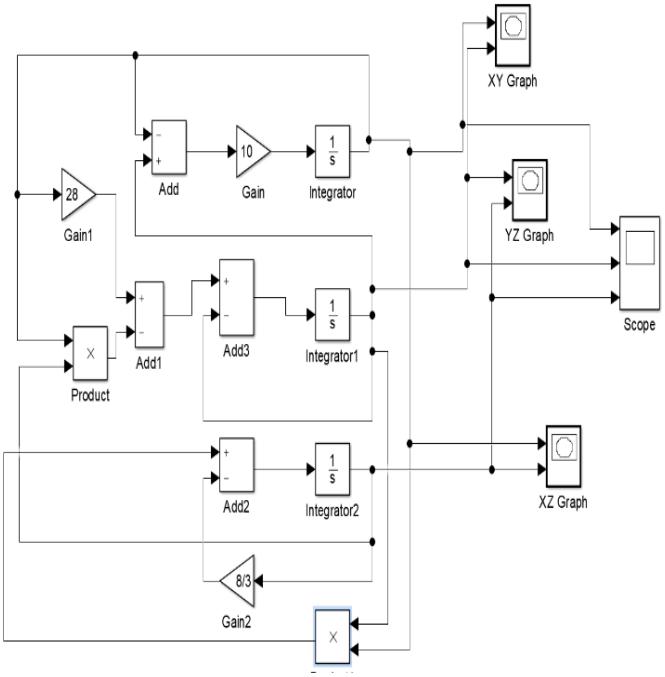

Fig 3: Simulink model of Lorenz Choatic system

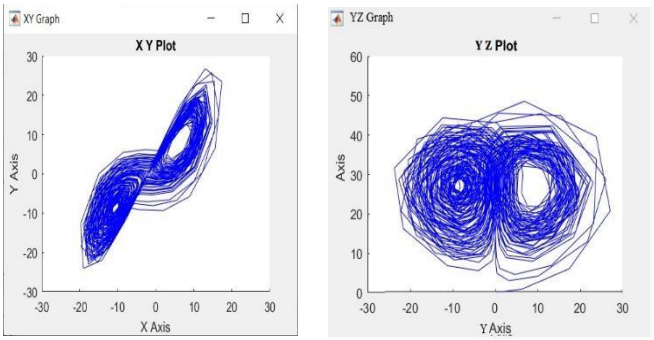

Fig 4: Space diagram of (XY) and (YZ) attractor.

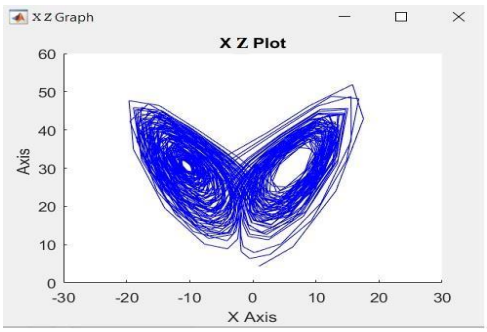

Fig 5: Space diagram of (ZX) attractor.

VI. Implementation of Lorenz System Using Xilinx System Generator

Now the Simulink model was converted to Xilinx System Generator Model using Xilinx block set under the MATLAB. The main problem was there is no integrator within the Xilinx System Generator toolbox, so the integrator block was converted to model as shown in Fig 6. This integrator block was converted to model using the basic numerical operations such as summation and delay blocks. This integrator was used into Lorenz chaotic system to eliminate algebraic loop for the hardware. The block AddSub is composed of one adder and one subtractor. This circuit uses 32-bits words with the binary point position after the bit number 16. The parameter dt presented in the block dt represents the integration step. The initial state of the integrator is stored inside the block Register, in the field Initial Value.

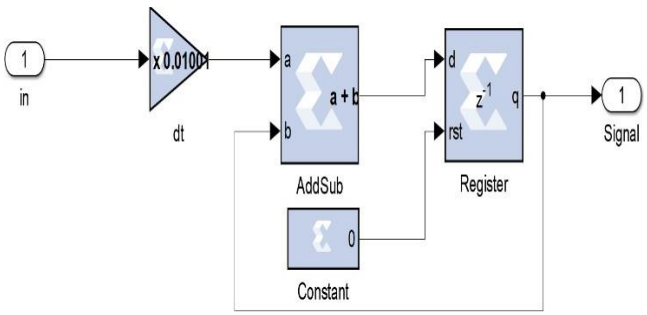

Fig 6: Integrator model using Xilinx system generator

The Lorenz system presented previously is now implemented using Xilinx block set library's elements as can be seen in Fig 7 Three integrators blocks that can be seen in this structure are the same as already presented in Fig 6. From the reference paper [6] the author chooses three different value for different integrators.

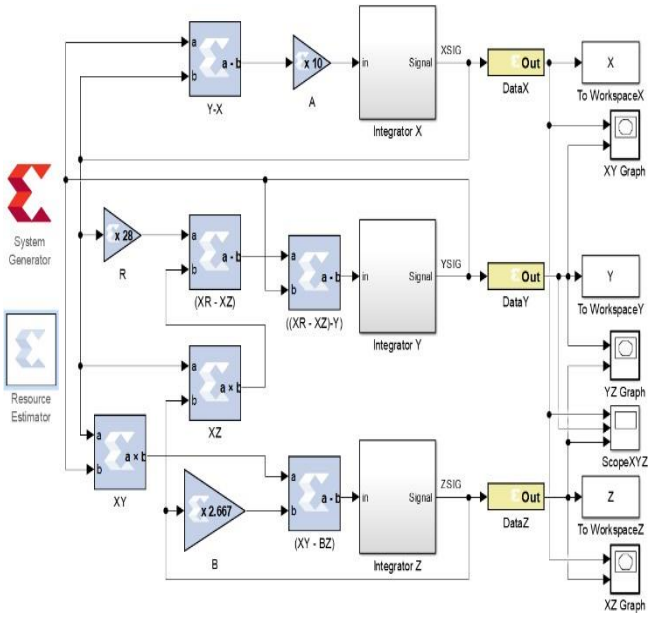

Fig 7: Lorenz chaotic generator using Xilinx System Generator blocks.

The all three Integrators has an initial condition equal to 10. The integration step dt used in this design is equal to 0.01. Many authors choose different initial values (x0, y0, z0) for getting perfect Lorenz output. The perfect output based on control parameters constant values.

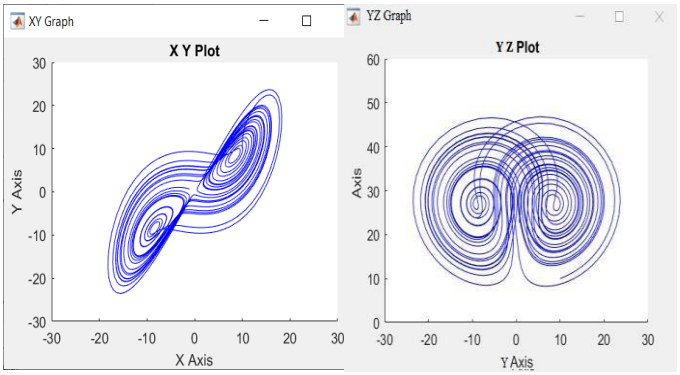

Fig 8: Space diagram of (XY) and (YZ) attractor.

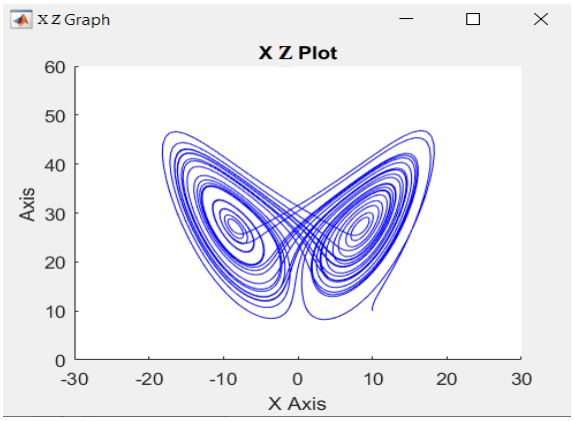

Fig 9: Space diagram of (ZX) attractor.

The Fig 8. The first parts relate to the phase plane (x, y) and second part is (y, z) and final part Fig 9 is related to the phase plane (x, z) of System generator output. Fig 10 shows Signals x, y and z generated by the Lorenz system created using Xilinx system generator. All the control parameter values in every cycle period in Fig 10. The figures below are obtained by fixing the following parameters, σ = 10, β = 8/3 and r = 28. The initial conditions x0 = 10, y0 = 10 and z0 = 10. It seems clearly that the different signals x, y and z have a random behaviour.

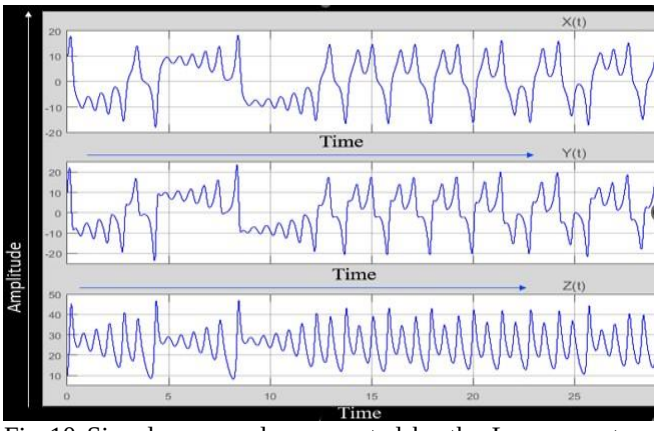

Fig 10 Signals x, y and z generated by the Lorenz system using system generator

TABLE I. NIST RANDOMNESS TEST

Statistical Test	Status	P-value
Frequency	Pass	0.727622
Block Frequency	Pass	0.928730
Cumulative Sums	Pass	0.625562
Long Runs of Ones	Pass	0.685843
Overlapping	Pass	0.122325
Runs	Pass	0.999715

Table I is performed using NIST SP800-22 test that was published by the National Institute of Standards and Technology. There are 13 different testes there. If P-value > 0.01 for each of the 13 tests, the test is considered to have passed.

VII. FPGA IMPLEMENTATION OF LORENZ CHAOTIC GENERATOR

In this section, discuss the implementation of the Lorenz key generators in FPGA. When the performance of Simulink and system generator design are completed, the hardware implementation VHDL code can be generated. We used in our study the Xilinx FPGA ARTIX-7 (xc7a35ticsg236-1L). Which includes,1800K-bit block RAMs, 400K-bit distributed RAMs and 250 user I/Os. It offers 33280 logic cells.

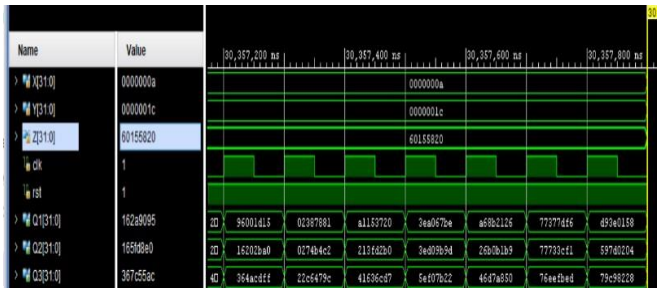

Fig 11 Simulation results using HDL code

The Fig 11 presents a simulation results using HDL code. The design was implemented using ARTIX-7 (xc7a35ticsg236-1L) FPGA and the resource utilization was estimated as shown below.

TABLE II. RESOURCE UTILIZATION TABLE

Logic Utilization	Used	Available	Utilization (%)
Look-up Table (LUTs)	1782	20800	8.57
Flip-Flops	93	41600	0.22
Bonded IOB	98	106	92.45
BUFG	1	32	3.13
DSP	10	90	11.11

The 32-bit floating-point design is optimized in order to meet the timing requirement for 10 ns clock period at 100 MHz system clock frequency. The optimized 32-bit model achieves a maximum frequency of 14.925 MHz with a low latency time of 67 ns.

TABLE III. TIMING REPORT

Clock period Ts(ns)	10 ns
Worst Negative Slack(ns)	1.569 ns
Maximum Frequency (MHz)	14.925 MHz
Total On-chip Power(W)	0.75W
Min clock rate (ns)	67 ns

TABLE IV. RESOURCE UTILIZATION COMPARISION TABLE

Logic Utilization	Artix-7	Spartan-3E[A]	Virtex-II[B]	Kintex-7[C]
Look-up Table (LUTs)	1782	1,912	2057	516
Flip-Flops	93	144	554	256
Bonded IOB	98	14	224	129
BUFG	1	2	NA	1
DSP	10	8	8	14

[A] Reference design in [2]
[B] Reference design in [6]
[C] Reference design in [7]

Compared to reference design [2], [6] and [7] shows in Table IV. The number of logic elements used is reduced in this performance implementation. This work uses a smaller number (93 out of 41600) of flip-flops, LUTs (1782 out of 20800), and BUFG (1 out of 32) to implement this work. So, compared with the Spartan-3E FPGA, Kintex-7, Virtex-II FPGA and Artix-7. Artix-7 will give an efficient and better performance.

VIII. CONCLUSION

This work focuses on the real-time FPGA implementation of a random key, based on Lorenz's chaotic generator for information security. The developed approach consists on using the implemented forth order Rung-Kutta method to resolve the differential equations system of the Lorenz chaotic generator. Lorenz chaotic oscillator model was designed and simulated using MATLAB code, MATLAB-Simulink and Xilinx System Generator. This design approach can be extended for FPGA implementation of other chaotic generator designs. In this experiment, every sequence is generated by the Lorenz Generator and tested by NIST. Chaotic bit stream has been used to generate a truly random key. The design was implemented in ARTIX-7 (xc7a35ticsg236-1L) FPGA and the resource utilization was also calculated. The optimized 32-bit model achieves a maximum frequency of 14.925 MHz with a low latency time of 67 ns. The implementation of the proposed architecture allows a very useful and attractive trade-off between high speed, low area cost and data security transmission for an information security system.

REFERENCES

[1] L.M.Pecora and T.L.Carroll, ”Synchronization in chaotic systems”, Phys. Rev.Let, vol,pp.821-824,Feb.1990.
[2] Merah, L., Ali-Pacha, A., Said, N.H., Mamat, M.: “Design and FPGA implementation of Lorenz chaotic system for information security issues”. Appl. Math. Sci. 7(5), 237–246 (2013).
[3] Zhengguo Li, Kun Li, Changyun Wen, and Yeng Chai Soh; “A new chaotic secure communication system” IEEE Transaction on Communications, Vol.51, No.8, Aug 2003.
[4] L. Zhang, "System generator model-based fpga design optimization and hardware co- simulation for lorenz chaotic generator", 2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS 2017), pp. 170-174, June 2017.
[5] A.Abel and W. Schwartz, “Chaos Communications-Principles, Schemes and Systems analysis’’, Proc. of the IEEE Inst. for Fundamentals of Electr.Eng. & Electron.,Dresden Univ. of Technol, 90. 2002. pp. 691-710.
[6] B N, Aryalekshmi, “FPGA Implementation of Lorenz’s Chaotic Generator”, Universal review, vol viii, March 2019.
[7] Qiang lai, xiao-wen zhao, karthikeyan rajagopal, guanghui xu,Akif akgul, and emre guleryuz “ Dynamic analyses, FPGA implementation and engineering applications of multi-butterfly chaotic attractors generated from generalised Sprott C system”, 2018.
[8] Sadoudi S, Azzaz M.S., Djeddou, M., Benssalah, M. “An FPGA real-time implementation of the Chen’s chaotic system for securing chaotic communicate ons” International Journal, Nonlinear Science. 7, 467–474 (2009).
[9] IEEE Standard for Floating-Point Arithmetic, ANSI/IEEE Standard 754-2008, New York: IEEE, Inc., Aug. 29, 2008.
[10] IEEE standard for binary-floating point arithmetic, ANSI/IEEE Std 754-1985, The Institute of Electrical and Electronic Engineers Inc., New York, August 1985.
[11] M. Aseeri, M. I. Sobhy, and P. Lee “Lorenz chaotic model using field programmable gate array (fpga),” in The Midwest Symposium on Circuits and Systems, Midwest Symposium on Circuit and Systems, 2002, pp. 686-699.
[12] M.Lakshmanan, S.Rajasekar, ”Nonlinear Dynamics Integrability, Chaos and Patterns “ 3rd edition Springer International Edition.