Chaotic Systems with FPGA: Real-time Implementation and Analysis

Ms.Divya K. Shah
Asst professor, Department of Electronics Engineering, Ramrao Adik Institute of Technology, Navi Mumbai, Maharashtra, India. Email: divya.shah@rait.ac.in

Mr.Vishwesh A. Vyawahare
Professor, Department of Electronics Engineering, Ramrao Adik Institute of Technology, Navi Mumbai, Maharashtra, India

Mr. Mukesh D.Patil
Professor, Department of Electronics &Telecommunication Engineering, Ramrao Adik Institute of Technology, Navi Mumbai, Maharashtra, India

Mr.Gaurav Datkhile
Asst professor, Department of Electronics Engineering, Ramrao Adik Institute of Technology, Navi Mumbai, Maharashtra, India

Abstract: Digital implementation of chaotic systems has advantages like high accuracy and ease of integration with embedded systems as compared to analog implementation. This paper presents the implementation of integer order Lü, Liu, Newton-Leipnik, Arneodo, Bhalekar-Gejji and Van der Pol chaotic systems on Field Programmable Gate Array (FPGA). The aforementioned chaotic systems are implemented on Altera FPGA Cyclone IV E (EP4CE11529C7N) chip. FPGA implementation has advantages like high accuracy and hardware optimization. As all the chaotic systems are processed parallel, the FPGA implementation provides real-time flexibility of selection of chaotic system for implementation using a switch. ModelSim simulation results are validated with MATLAB for the proposed chaotic systems. The implementation is carried out on Altera DE2-115 Board. A detailed implementation analysis related to hardware resource utilization on FPGA is presented for each chaotic system.

Keywords: Chaotic System, FPGA, Hardware implementation, Advanced DSP Builder blockset, VHDL.

1 Introduction

In recent years there is an increasing interest in theoretical research and practical implementation of chaotic systems. The non-linear dynamic systems exhibit chaos, and are characterized by sensitivity to initial conditions [1]. In comparison with analog implementation of chaotic system as discussed in [2] for applications like data encryption and secure communications between embedded systems, the digital implementation of chaotic systems has advantages of accuracy and possible integration in embedded applications [3]. The analog implementation has practical difficulties as components value are sensitive to temperature, ageing, whereas digital implementation overcomes the problems. Discrete algorithms can be implemented on various platforms: like Field Programmable Gate Array (FPGA), common microprocessors or special microprocessors like Digital Signal Processor (DSP). The implementation of chaotic system on DSP using fixed point realization seems to be the optimal solution for the real time information processing because of its simplicity, flexibility in operation and low cost. The architecture of DSP allows us to realize the basic algorithm in most effective way such that the software part/code can be written in much convenient way. As

discussed in the [4] DSP implementation of chaotic system has problem like restriction of accuracy, causing transition from infinite set of numbers to a final set etc. DSP is a single chip computer which has finite length variables in computation it becomes necessary to consider the overflow problem and computation error problem. Also implementation of many chaotic systems at a time on DSP becomes difficult as it executes the software part in sequential manner resulting in large computational time and limitations on implementation.

Fractional order chaotic system had been implemented in [5].FPGA implementation consumes less power as compared to implementation using microprocessor and DSP. Low-power usage is due to a lower clock rate and the absence of wasted cycles for the fetch/decode instruction in FP- GAs. In this context, the use of FPGA technology makes it possible to optimize the hardware resources required while allowing for real-time computing. FPGAs provide great trade-off between computational power and the processing flexibility hence they have a great place in design of digital systems. Considerable work has been done so far on the FPGA implementation of basic chaotic systems. Rossler system [6], Novel chaotic oscillator [7], 3D chaotic system [8], chaotic oscillator [9] is implemented using RK-4 method .The Chen system [1], Lorenz system [10, 11, 1], the Hyperchaotic system [12] is implemented using DSP builder design tool. The higher dimensional digital chaotic system [13] is implemented on FPGA.

The novel chaotic system [14], four wing chaotic attractor [15], Lorenz chaotic system [16], new chaotic system for synchronization is implemented using Xilinx system generator tool. Designing of four systems Liu Chen, Lu, Chen, and Lorenz together is shown in [3]. Multiscroll chaotic oscillator [17] is implemented on FPGA for chaotic communication applied to image processing. The implementation of four discrete time chaotic generators [18] is done on FPGA.

The paper presents FPGA implementation of following six chaotic systems:

1. Lü system.
2. Liu system.
3. Newton- Leipnik system.
4. Arneodo system.
5. Bhalekar-Gejji system.
6. Van der Pol system.

The state variable length used for implementation has 32 bit -word length with floating point representation, hence greater accuracy is achieved. The methodology to carry out the implementation of chaotic system is followed from the book [1].

The paper is organized as follows. Section 2 briefly introduces algorithm of implementation and presents the basic block diagram of implementation of chaotic system on FPGA. Section 3 provides insights about different chaotic system implemented. Hardware details are given in Section 4. Section 5 provides details about the DSP builder. Section 6 presents the results of FPGA implementation of chaotic systems. Analysis and comparison in terms of resource utilization for the implemented systems is presented in Section 7. Conclusion is given in section 8.

2 Chaotic systems on FPGA

The block diagram of proposed work is shown in Figure 1. Six different 3D chaotic systems are designed using Advanced DSP builder blockset in MATLAB/Simulink environment. VHDL code is generated by using the Hardware- In-Loop (HIL) block supplied by DSP builder. After debugging, the generated VHDL code is simulated in ModelSim simulator, which is then verified with MATLAB results. Synthesized VHDL code is transferred on FPGA. Multiplexer/ Control unit is designed to select one of the chaotic systems and state variables. Results of Implemented work are

shown on DSO using Coder-Decoder (CODEC) available on FPGA.
The Reset and Clock signals on the inlet of the units have been used to ensure the timing of sub-modules within the units and the synchronization between modules and the systems. The initial conditions required for the chaotic systems are embedded in the code itself. The three output state variables are x(t), y(t) and z(t) are of 32 bits in word length with floating point data type. The laboratory setup for implementation of chaotic system on FPGA is shown in Figure 2.
In comparison with other processors, FPGAs use dedicated hardware for processing logic and therefore not constrained by the complexities of additional overhead, such as an operating system. The latest FPGAs emphasize on low dynamic power performance and hence they are increasingly used for digital signal processing (DSP) applications. The processor based system has layers of abstraction to schedule tasks and share resources among multiple resources. These complexities are not necessary for FPGA based system [1].

1 Chaotic system

This section describes the six chaotic systems used in this work.

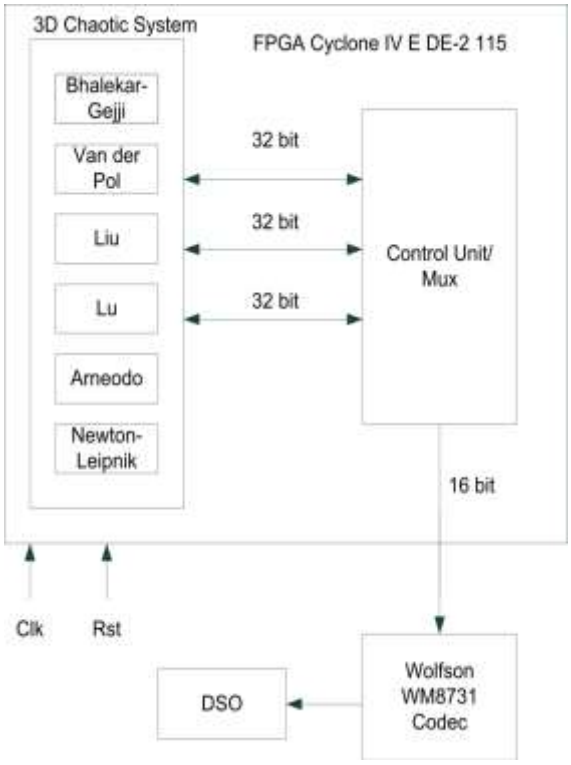


Fig. 1 Basic block diagram of implementation on FPGA.

3.1 Lü system

Recently, Lu¨ reported a new chaotic attractor, which bridged the gap between the Lorenz system and Chen system. It is a typical transition system; Lu¨ system has been analyzed in [3]. This system has a wide range of parameters values in which the system displaces a chaotic attractor of different shapes. The range of values is listed below the system equation.
The system equations are as follows:

$$\begin{aligned} \dot{x} &= a(y(t) - x(t)), \\ \dot{y} &= -x(t)z(t) + cy(t), \\ \dot{z} &= x(t)y(t) - bz(t), \end{aligned} \qquad (1)$$

where $\dot{x}$, $\dot{y}$, $\dot{z}$ represents the time derivative of the variables x, y, z for brevity, the explicit dependent on time will be omitted in the remaining of the paper. The parameters are
a =36, b =3, c =20.
Initial condition of system variables which exhibits chaotic behavior are
$x(0)=10$, $y(0)=20$, $z(0)=30$.

Fig. 2 Laboratory hardware setup for implementation of Chaotic System on FPGA.

3.2 Liu System

Liu System [19] is a three dimensional autonomous system that relies on one multiplier and one quadratic term to introduce the non-linearity necessary for folding trajectories. Reference [20] discusses the control method for Liu system; it can be stabilized at one-equilibrium point or limit cycle surrounding its equilibrium. The system equation for same is as follows:

$$\begin{aligned} \dot{x} &= -ax - ey2, \\ \dot{y} &= by - kxz, \\ \dot{z} &= -cz + mxy. \end{aligned} \qquad (2)$$

The parameters are
a = e =1, b =2.5, k = m =4, c =5.
Initial condition of system variables which exhibits chaotic behavior are
x(0)=10, y(0)=20, z(0)=30.

3.3 Arneodo System

In 1985, Arneodo [21] provided a list of normal forms of ordinary differential equations describing the dynamics of physical systems in condition near to the simultaneous onset of up to three instabilities. The Arneodo has rich dynamical behaviors deserving to be explored. Arneodo system exhibits oscillatory behavior for the integer-order differential equation of at least two. Also to exhibit multiple limit cycle the order of the system must be at least four. The system equation for same is as follows:

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= z, \\ \dot{z} &= -\beta 1x - \beta 2y - \beta 3z + \beta 4x3. \end{aligned} \qquad (3)$$

The parameters are
β1=-5.5, β2=3.5, β3=1, β4=-1.
Initial condition of system variables which exhibits chaotic behavior are

x(0)=-0.2, y(0)=0, z(0)=0.2.

3.4 Newton−Leipnik System

In 1981, Newton and Leipnik constructed a set of differential equations from Euler's rigid body equations which were modified with a linear feedback. Then in 2002, B. Marlin established the existence of closed orbits which were not asymptotically stable for this system. In 2002, Chen et al studied chaos control and synchronization of the Newton−Leipnik system for the first time by using a stable-manifold based method [22]. The System equation for same is as follows:

$$\begin{aligned} x^{\cdot} &= -ax + y + 10yz, \\ y^{\cdot} &= -x - 0.4y + 5xz, \\ z^{\cdot} &= bz - 5xy \end{aligned} \qquad (4)$$

The parameters are
a =0.14, b =0.175.
Initial condition of system variables which exhibits chaotic behavior are
x(0)=0.349, y(0)=0, z(0)=-0.16.

3.5 Bhalekar−Gejji System

A new chaotic system having four parameters is proposed by [23]. It is observed that the system exhibits rich dynamics ranging from chaotic to limit cycle oscillations. It's application for secured communication is described in [24]. The forming of Bhalekar-Gejji chaotic dynamical system is discussed in [23]. The system equation for Bhalekar- Gejji system are as follows:

$$\begin{aligned} x^{\cdot} &= (\omega * x) - y2, \\ y^{\cdot} &= \mu(z - y), \\ z^{\cdot} &= ay - bz + xy. \end{aligned} \qquad (5)$$

The parameters are

ω=-2.667, μ=10, a =27.3, b =1.

Initial condition of system variables which exhibits chaotic behavior are

x(0)=5,y(0)=5,z(0)=0

3.6 Vander Pol system

The Van der Pol oscillator (VPO) represents a nonlinear system with an interesting behavior that exhibits naturally in several applications. It has been used for study and design of many models including biological phenomena, such as the heartbeat, neurons, acoustic models, radiation of mobile phones, and as a model of electrical oscillators (implemented with a tunnel diode, memristor or operating amplifier). The VPO model was used by Van der Pol in 1920 to study oscillations in vacuum tube circuits [25]. Chaotic circuit using Ring Oscillator and Ven der Pol Oscillator is proposed by [26]. The oscillations found in Van der Pol is called relaxation-oscillations and are also known as limit cycle in electrical circuits employing vacuum tubes.

$$\begin{aligned} x^{\cdot} &= y, \\ y^{\cdot} &= -x - (\varrho)(x2 - 1) * y. \end{aligned} \qquad (6)$$

The parameters are
ϱ=1.
Initial condition of system variables which exhibits chaotic behavior are
x(0)=0.2, y(0)=-0.2.

3.7 Discrete approximation of chaotic system: Euler's Method

For digital implementation, need arises to convert the time domain differentiation in discrete domain, hence in the proposed work discrete approximation of integration is used as shown in Figure 3 using Euler's Method. Consider (7)

$$\dot{x}1 = f1(x1, \ , xn),$$
$$\dot{x}n = fn(x1, ..., xn). \quad (7)$$

By using forward Euler's method the (7) can be rewritten as (8)

$$\dot{x}1(t + \delta(t)) = x1(t) + f1(x1(t), \ , xn(t))\Delta t,$$
$$\dot{x}n(t + \delta(t)) = xn(t) + fn(x1(t), ..., xn(t))\Delta t. \quad (8)$$

For further to implement it

on FPGA the above (8) can be defined as (9)

$$xNNew = xn(t) + fn(x1(t),, xn(t))\Delta t,$$
$$xN = xn(t + \delta t). \quad (9)$$

where xNNew and xN are VHDL signals with N = 1, 2,..., n. For D flip- flop to be synchronous with xN the clock period required is δt. The step size required for Euler's method is Δt.

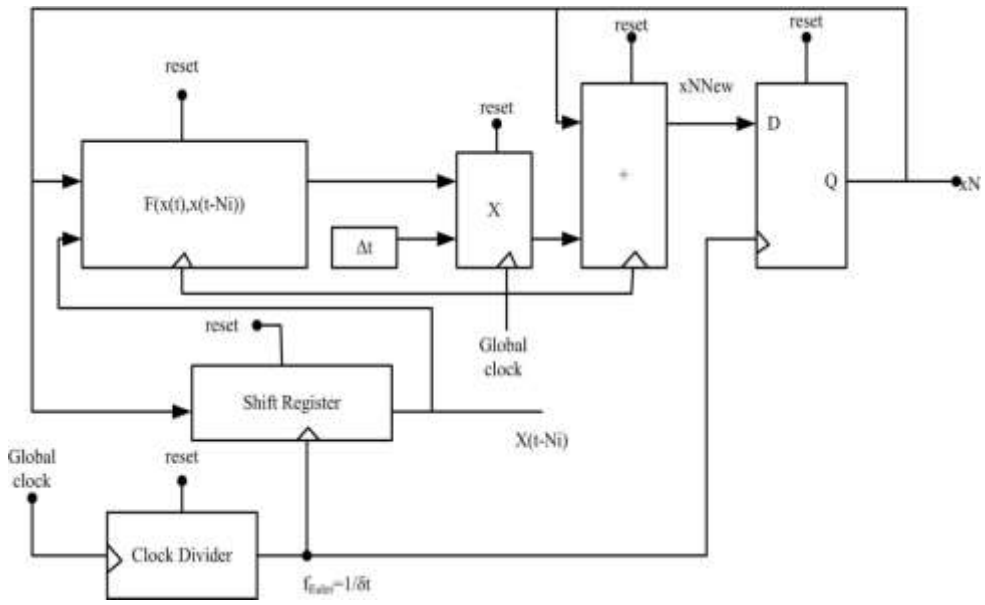


Fig. 3 Euler's Method [1].

4 Hardware Description

The implementation was carried out on DE2-115 board from altera that utilizes a Cyclone IV (EP4CE115F29C7N) FPGA chip [27].

–Cyclone IV devices are ideal for low-cost, small-form-factor applications in the wireless, wire-line, broadcast, industrial, consumer, and communications industries.

–Cyclone IV E offers the lowest power and high functionality
with the lowest cost.

–Cyclone IV EP4CE115F29C7N device has 114,480 logic elements, 300 9*9 bit Multipliers, 266 18*18 bit Multipliers, 3888K Embedded Memory, 4 general-purpose PLLs, 20 Global Clock Networks, 8 User I/O Banks, and 528 Maximum user I/O [27].

5 DSP Builder

DSP Builder is a digital signal processing (DSP) design tool that allows push- button HDL generation of DSP algorithms directly from Math Works Simulink environment.

DSP Builder is used to design algorithm, set desired data rate, clock frequency, and device offering accurate bit and cycle simulation, synthesizing fixed- and floating-point optimized HDL, auto-verify in ModelSim- Altera software, and auto-verify/co-simulate on hardware. This tool adds additional Altera libraries alongside existing Simulink libraries with the Altera DSP Builder Advanced Blockset and DSP Builder Standard Blockset. The proposed methodology is implemented in Advanced DSP builder blockset for obtaining
results on FPGA. DSP Builder is produced by Altera for DSP related development [28], and it can also work in MATLAB and Simulink environment. The procedure of designing with DSP Builder is shown in Figure 4.

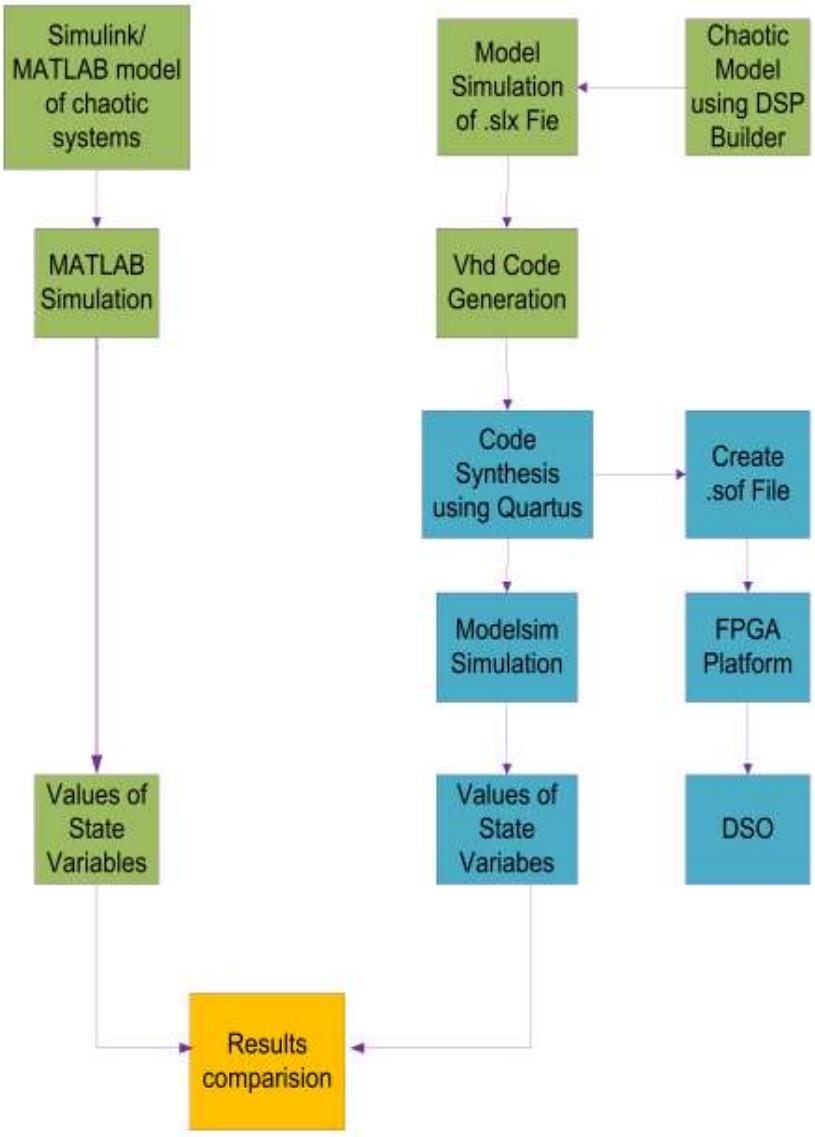


Fig. 4 Steps involved in implementation on FPGA.

The 32 bit floating point representation is used for state variables and 16 bit representation is used of scaled variable to match with voltage range of CODEC. Two separate .vhd programs need to be written for providing the initial conditions for chaotic system and step size, and for providing the appropriate clock.

6. Implementation of Chaotic System

FPGA implementation of chaotic systems is discussed below:

6.1 Lü System

The chaotic system given by (1) in section 3.1 is now implemented using FPGA. With the use of forward Euler's method the ordinary differential equation is solved as given in (7)-(9) in section 3.7 and it is realized in discrete domain, the Lu¨ system looks like: 11-13.

6.1.1 Simulation Results

After the generation of VHDL code from DSP builder it is simulated in ModelSim simulator, the results of same are as shown in Figure 5. As stated earlier, the state variable has word length of 32 bit with floating point representation. In order to match the voltage range of CODEC these state variables are truncated to 16 bit representation. As shown in Figure 5 the variables xout, yout and zout are scaled variables having 16 bits representation. It is also verified that the values obtain from ModelSim for state variables exactly matches with that obtained with MATLAB as shown in x(t) versus y(t), y(t) versus z(t), z(t) versus x(t) phase plots in Figure 6-8. The above procedure is repeated for other implemented chaotic systems as well.

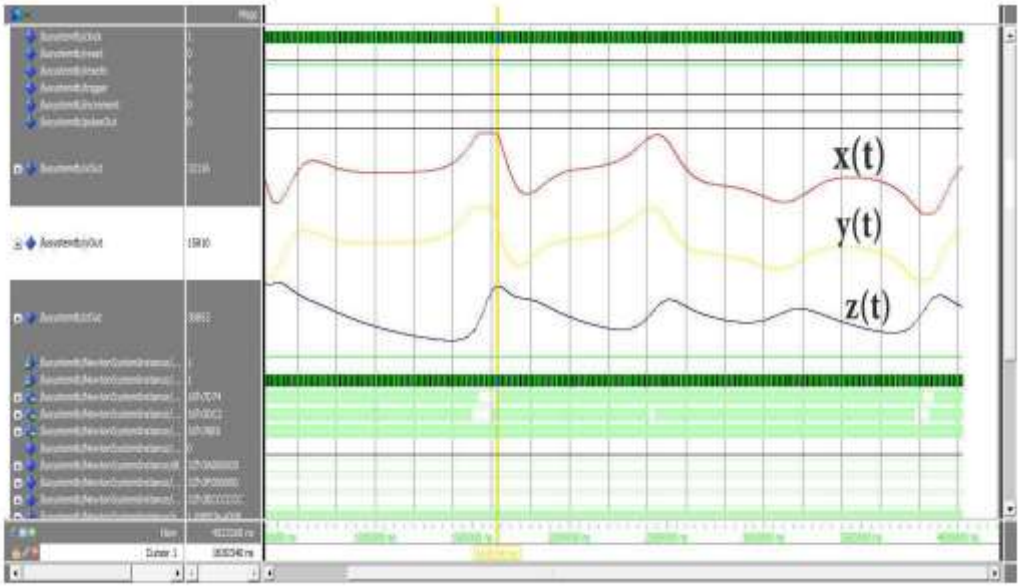


Fig. 5 ModelSim waveform of Lu¨ system simulated for 4ms: chaotic signals x(t), y(t) and z(t)

6.1.2 Implementation Analysis

After generation of VHDL code from DSP builder and after performing the functional simulation of same it is synthesized in Quartus to generate .sof file which is transferred to FPGA via banana cable. The built in CODEC of FPGA produces the analog output which is viewed on DSO. The above steps are followed for other systems as well. The implementation result of Lu¨ System is shown in following Figure 9.

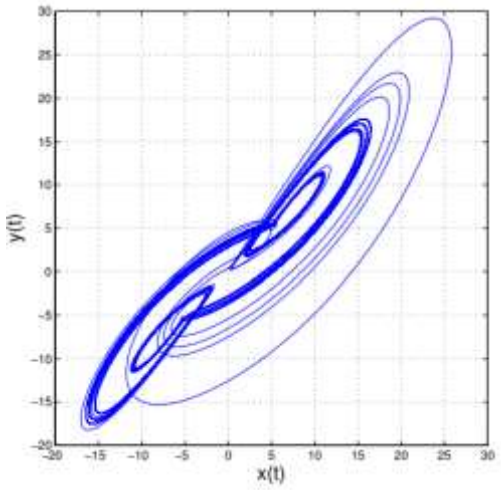


Fig. 6 Phase plot of Lü system obtained from MATLAB: $x(t)$ versus $y(t)$

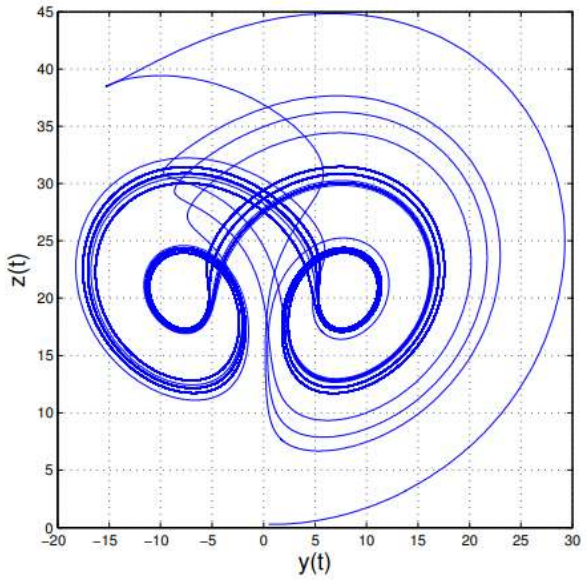


Fig. 7 Phase plot of Lü system obtained from MATLAB: $y(t)$ versus $z(t)$

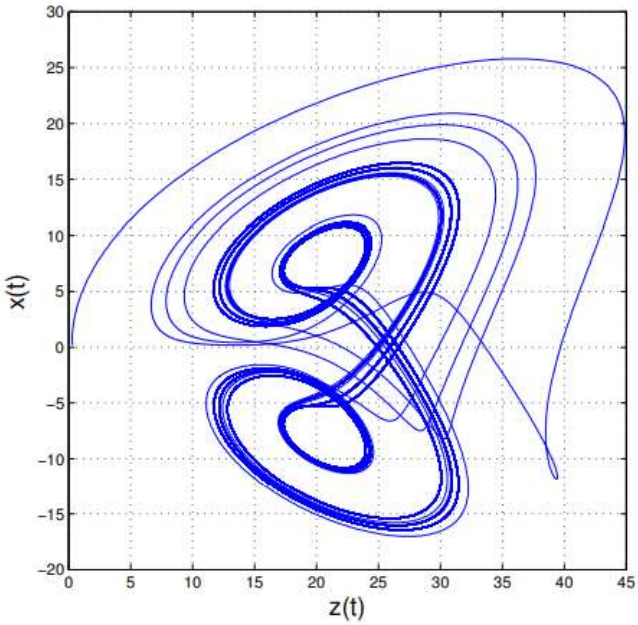


Fig. 8 Phase plot of of Lü system obtained from MATLAB: $z(t)$ versus $x(t)$

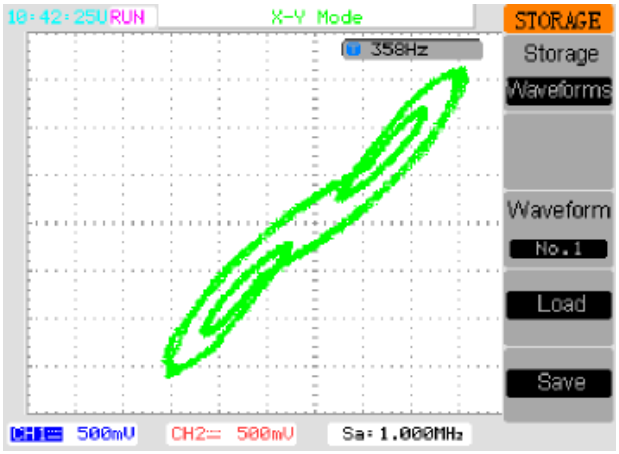


Fig. 9 Real-time Results of the implemented Lü chaotic generator: $x(t)$ versus $y(t)$

6.2 Liu System

The chaotic system given by (2) in section 3.2 is now implemented using FPGA. With the use of forward Euler's method the ordinary differential equation is solved as given in (7)-(9) in section 3.7 and it is realized in discrete domain. The real time implementation result of Liu System is shown in following Figure 10-12.

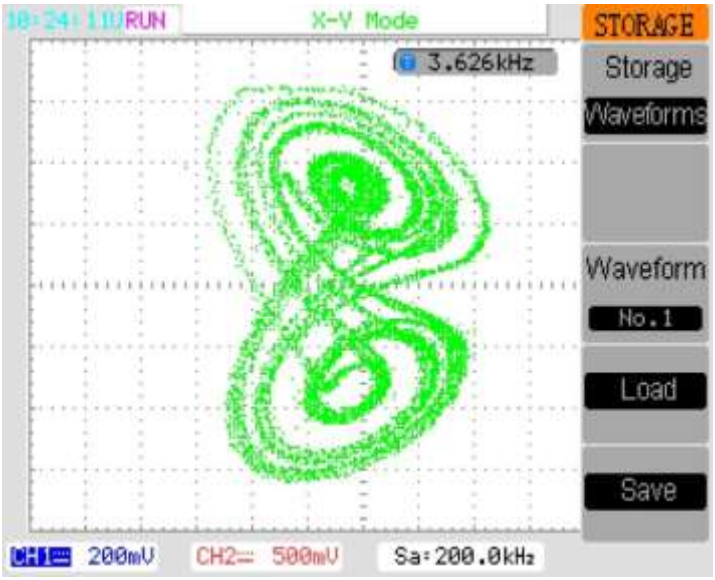


Fig. 10 Real-time Results of the implemented Liu chaotic generator: x(t) versus y(t).

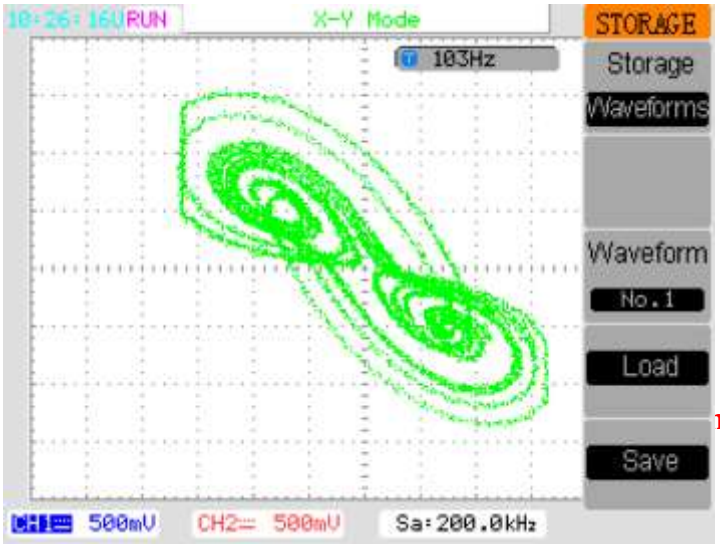


Fig. 11. Real-time Results of the implemented Liu chaotic generator: y(t) versus z(t).

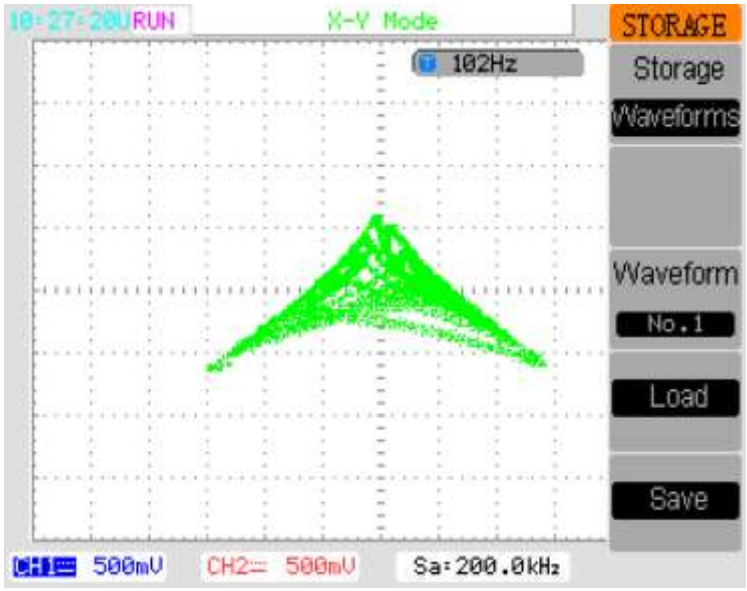
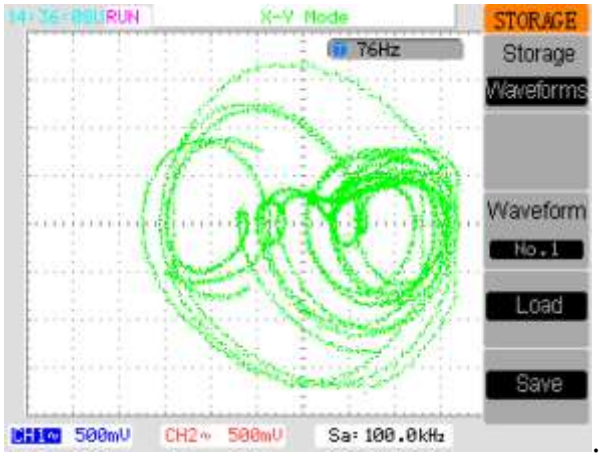


Fig. 12 Real-time Results of the implemented Liu chaotic generator: $z(t)$ versus $x(t)$.

6.3 Arneodo System

The FO system given by (3) in section 3.3 is now implemented using FPGA. With the use of forward Euler's method the ordinary differential equation is solved as given in (7)-(9) in section 3.7 and it is realized in discrete domain. Validation of ModelSim simulation results with MATLAB is done and the real time implementation result of Arneodo System is shown in following Figure 13-15.



.

Fig. 13 Real-time Results of the implemented Arneodo chaotic generator: x(t) versus y(t).

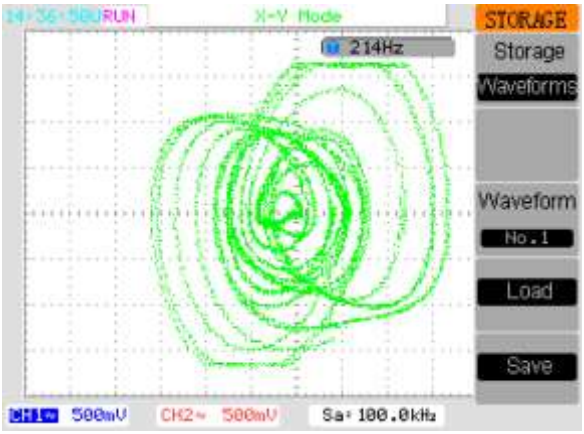


Fig. 14 Real-time Results of the implemented Arneodo chaotic generator: y(t) versus z(t).

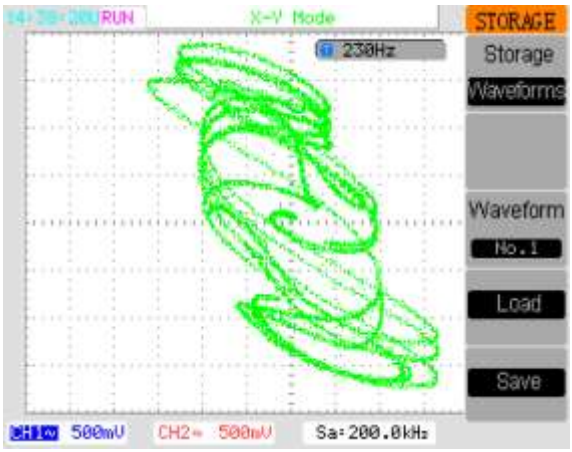


Fig. 15 Real-time Results of the implemented Arneodo chaotic generator: z(t) versus x(t).

6.4 Newton − Leipnik System

The chaotic system given by (4) in section 3.4 is now implemented using FPGA. With the use of forward Euler's method the ordinary differential equation is solved as given in (7)-(9) in section 3.7 and it is realized in discrete domain. Validation of ModelSim simulation results with MATLAB is done and the real time implementation result of Newton − Leipnik System is shown in following Figure 16-18.

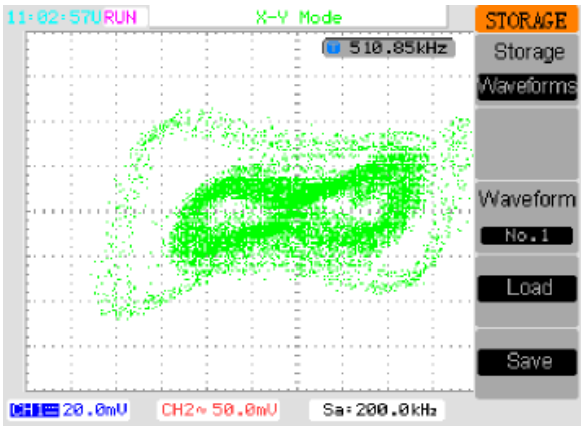


Fig. 16 Real-time Results of the implemented Newton − Leipnik System chaotic generator: x(t) versus y(t).

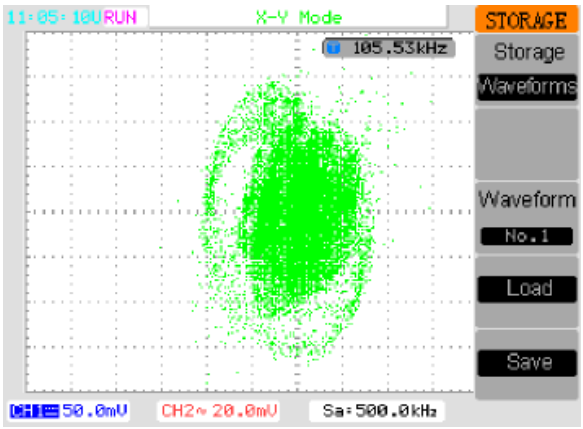


Fig. 17 Real-time Results of the implemented Newton − Leipnik System chaotic generator: y(t) versus z(t).

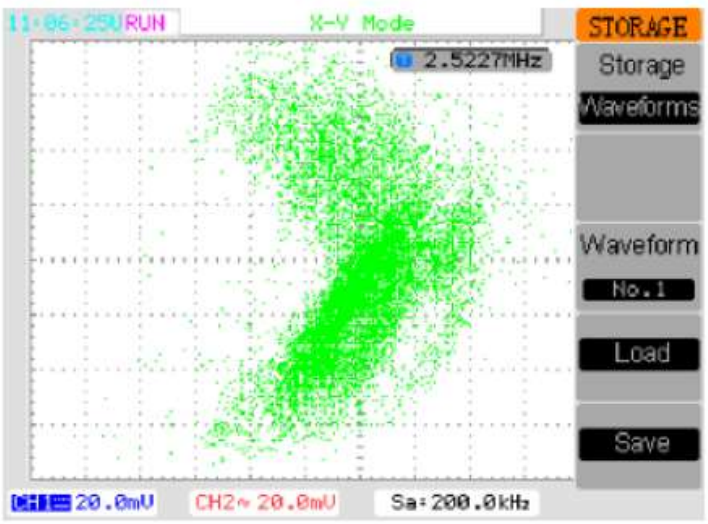


Fig. 18 Real-time Results of the implemented Newton − Leipnik System chaotic generator: z(t) versus x(t).

6.5 Bhalekar − Gejji System

The chaotic system given by (5) in section 3.5 is now implemented using FPGA. With the use of forward Euler's method the ordinary differential equation is solved as given in (7)-(9) in section 3.7 and it is realized in discrete domain. Validation of ModelSim simulation results with MATLAB is shown in Figure 19-21.

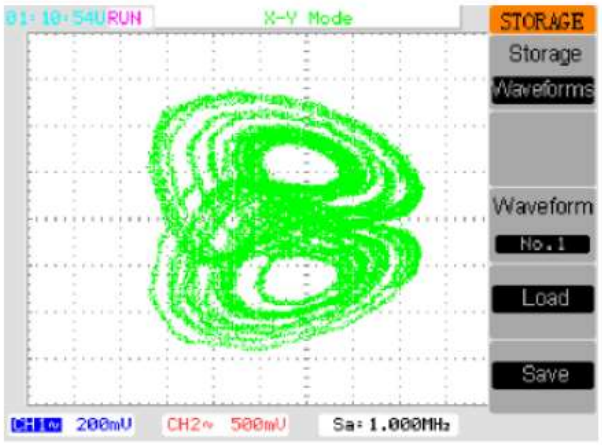


Fig. 19 Real-time Results of the implemented Bhalekar − Gejji System chaotic generator: x(t) versus y(t).

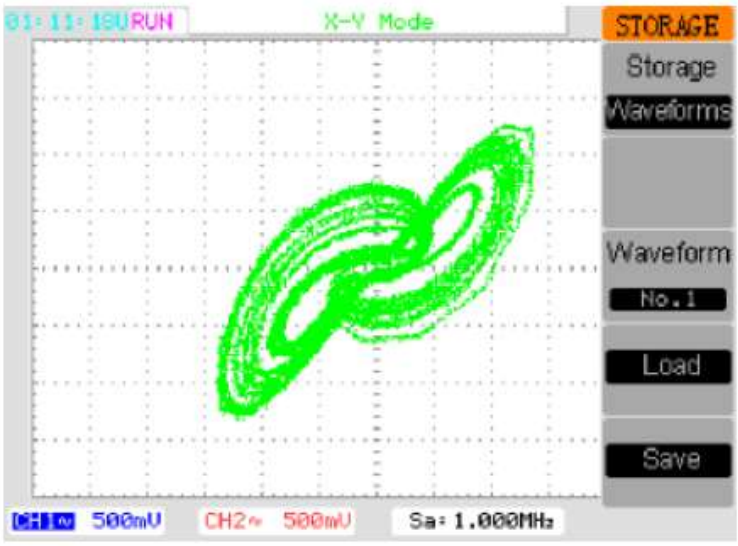


Fig. 20 Real-time Results of the implemented Bhalekar − Gejji System chaotic generator: y(t) versus z(t)

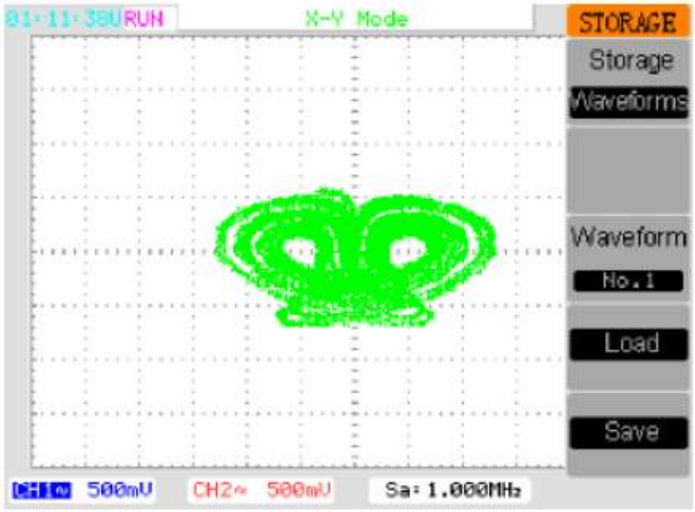


Fig. 21 Real-time Results of the implemented Bhalekar − Gejji System chaotic generator: z(t) versus x(t).

6.6 Van der Pol System

The chaotic system given by (6) in section 3.6 is now implemented using FPGA. With the use of forward Euler's method the ordinary differential equation is solved as given in (7)-(9) in section 3.7 and it is realized in discrete domain. Validation of ModelSim simulation results with MATLAB is done and the real time implementation result of Van der Pol
system is shown in fig 22.

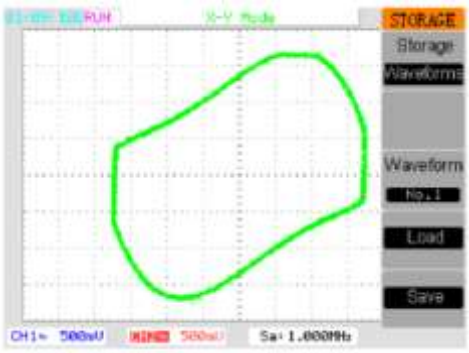


Fig. 22 Real-time Results of the implemented Van der Pol System chaotic generator: x(t) versus y(t)

7. Analysis and Comparison of Synthesis results and implementation performance

The synthesis result obtained for each chaotic generator from Quartus are specified in terms of logic resources required, registers used. Along with these total DSP block, multipliers and PLL used are also listed. The description of each term is discussed below.

7.1 Description of terms:

1. Logic elements are smallest units of logic it consist of a four-input look- up table (LUT) which can implement any function of four variables, a programmable register, a carry chain connection, register feedback support and it has ability to drive local, row, column, direct link interconnects. Total Logic Elements present on Cyclone IV E device is 114,480.
2. Logic utilization reports the percentage calculated from the number of half- adaptive logic modules (half-ALMs) available in the device and the number of half-ALM used in design.
3. Dedicated logic register refers to the two register present in ALM, total register present on Cyclone IV E device is 3528.
4. Total pins show the total number of pins used against the total number of pins available, total pins available on device is 529.
5. DSP block 9-bit elements make up the DSP blocks in a supported device family, total DSP block present on device is 532.
6. Total PLLs shows the total number of Phase-Locked Loop (PLL) used, total PLL present on device is 4.

The table 1 and 2 shows the hardware resources utilized on FPGA in terms of total logic elements, total (Phase Locked Loop) PLLs, total combinational functions, dedicated logic register and embedded multiplier along with maximum frequency. It can be stated from table 1 and 2, the implementation of chaotic system on FPGA utilizes only small percentage of total resources available with great accuracy and speed.

6. Conclusion

The paper presents real-time implementation of some chaotic systems using FPGA environment.

After validating the simulation results with ModelSim and MATLAB, the chaotic systems are implemented on Altera FPGA Cyclone IV E (EP4CE11529C7N) chip. A detailed analysis of various key indices useful in hardware utilization and implementation performance is also carried out. With lesser consumption of hardware resources and guaranteed accuracy, it may be concluded that FPGA hardware has an added advantage over other environments like micro-controllers and DSPs for the real-time implementation of chaotic systems. As a next step, the hardware exercise presented here can be used for real-time implementation of various advanced control strategies for synchronization of chaotic systems.

Table 1 Synthesis results and implementation performance of Lu¨, Liu and Arnedo chaotic systems

Resources	Chaotic Systems		
	Lu¨	Liu	Arnedo
Total Logic Elements (%)	7	8	7
Total Combinational Functions (%)	6	7	6
Dedicated logic Registers (%)	4	5	5
Total pins (%)	40	40	40
Total memory bits (%)	1	1	1
Embedded multiplier 9-bit elements (%)	16	14	22
Total PLLs (%)	25	25	25
Maximum Frequency (MHz)	86.36	87.94	90.3

Table 2 Synthesis results and implementation performance of Newton-Leipnik, Bhalekar-Gejji and Van der Pol chaotic systems

Resources	Chaotic Systems		
	Newton Leipnik	Bhalekar Gejji	Van der Pol
Total Logic Elements (%)	9	8	5
Total Combinational Functions (%)	8	7	4
Dedicated logic Registers (%)	6	5	3
Total pins (%)	40	40	40
Total memory bits (%)	1	1	1
Embedded multiplier 9-bit elements (%)	22	16	19
Total PLLs (%)	25	25	25
Maximum Frequency (MHz)	90.66	85.49	90.35

References

1. B. Muthuswamy and S. Banerjee, A Route to Chaos Using FPGAs. Switzerland: Springer, 2015.

2. K. M. Cuomo and A. V. Oppenheim, “Circuit implementation of synchronized chaos with applications to communications,” Phys. Rev. Lett., vol. 71, pp. 65–68, Jul 1993.

3. M. S. Azzaz, C. Tanougast, S. Sadoudi, R. Fellah, and A. Dandache, "A new auto-switched chaotic system and its FPGA implementation," Communications in Nonlinear Science and Numerical Simulation, vol. 18, no. 7, pp. 1792–1804, 2013.

4. E. Dmitriev, S. Starkov, and S. Yemetz, "Chaotic communication using digital signal processors," in International Symposium on Nonlinear The- ory and its Applications, vol. 3, pp. 1093–1096, 1998.

5. Divya K. Shah, Rohit B. Chaurasiya, Vishwesh A. Vyawahare, Khushboo Pichhode, Mukesh D. Patil, FPGA implementation of fractional-order chaotic systems, AEU - International Journal of Electronics and Communications, Volume 78,2017,Pages 245-257,ISSN 1434-8411,

6. S. Sadoudi and M. S. Azzaz, "Hardware implementation of the Rossler chaotic system for securing chaotic communication," 5th International Conference: Science of Electronic, Technologies of Information and Telecommunications, Tunisia, 2009.

7. K. Ismail, O. A. Turan, and P. Ihsan, "Implementation of FPGA-based real time novel chaotic oscillator," Nonlinear Dynamics, vol. 77, no. 1, pp. 49–59, 2014.

8. C. Tanougast and A. Dandache, Hardware Design of Embedded Systems for Security Applications, ch. 12, pp. 233–260. Croatia: InTech, 2012.

9. L. D. Micco and H. A. Larrondo, "FPGA implementation of a chaotic oscillator using RK4 method," in VII Southern Conference on Programmable Logic (SPL), 2011, pp. 185–190, April 2011.

10. H. Xue and W. bing LI, "Implementation of chaos synchronization on FPGA," Advances in information Sciences and Service Sciences, vol. 4, no. 6, pp. 42–51, 2012.

11. L. Merah, A. Ali-Pacha, N. H. Said, and M. Mamat, "Design and FPGA implementation of Lorenz chaotic system for information security issues," Applied Mathematical Sciences, vol. 7, no. 5, pp. 237–246, 2013.

12. W. Guang-Yi, B. Xu-Lei, and W. Zhong-Lin, "Design and FPGA implementation of a new hyperchaotic system," Chinese Physics B, vol. 17, no. 10, pp. 3596–3602, October 2008.

13. Q. Wang, S. Yu, C. Li, J. L, X. Fang, C. Guyeux, and J. M. Bahi, "Theoretical design and FPGA-based implementation of higher-dimensional digital chaotic systems," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 63, no. 3, pp. 401–412, 2016.

14. R. Rameshbabu, R. karthikeyan, R. balamurali, and A. Prasina, "FPGA implementation of adaptive complete synchronization methodology for novel chaotic systems," Middle- East Journal of Scientific Research, vol. 23, pp. 36–44, 2015.

15. D. Enzeng, L. Zhihan, D. Shengzhi, and C. Zengqiang, "Topological horse- shoe analysis on a four-wing chaotic attractor and its FPGA implementation," Nonlinear Dynamics, vol. 83, no. 1, 2016.

16. Tanougast, Hardware Implementation of Chaos Based Cipher: De- sign of Embedded Systems for Security Applications. Berlin Hedelberg: Springer-Verlag, 2011.

17. E. Tlelo-Cuautle, V. H. Carbajal-Gomez, P. J. Obeso-Rodelo, J. J. Rangel- Magdaleno, and J. C. Nu´n˜ez-Pe´rez, "FPGA realization of a chaotic communication system applied to image processing," Nonlinear Dynamics, vol. 82, no. 4, pp.
1879–1892, 2015.

18. P. Giard, G. Kaddoum, F. Gagnon, and C. Thibeault, "FPGA implementation and evaluation of discrete-time chaotic generators circuits," in 38th Annual Conference on IEEE Industrial Electronics Society IECON 2012, pp. 3221–3224, October 2012.

19. C. Liu, L. Liu, and T. Liu, "A novel three-dimensional autonomous chaos system," Chaos, Solitons and Fractals, vol. 39, no. 4, pp. 1950 – 1958, 2009.

20. M. Wang and X. Wang, "Controlling liu system with different methods, "Modern Physics Letters B, vol. 23, no. 14, pp. 1805–1818, 2002.

21. S. Vaidyanathany, "Adaptive backstepping controller and synchronizer design for Arneodo chaotic system with unknown parameters," International Journal of Computer Science and Information Technology (IJC- SIT), vol. 4, no. 6, pp. 143 – 157, Dec 2012.

22. Z. Ruigin and S. Yunzhong, "Circuit realization of Newton-Leipnik chaotic system via EWB," in Chinese Control and Decision Conference, Yantai, Shandong, pp. 5111–5114, 2008.

23. S. Bhalekar and V. Daftardar-Gejji, "A new chaotic dynamical system and its synchronization," in Proceedings of the international conference on mathematical sciences (A. M. Mathai ed.), Jan 2011.

24. P. P. Singh, J. P. Singh, and B. K. Roy, "Synchronization of chaotic systems using NAC and its application to secure communication," International Journal of Control Theory and Applications, vol. 8, no. 3, pp. 995– 1003, 2015.

25. I. Petras, Fractional-Order Nonlinear Systems, vol. 2 of Nonlinear Physical Science. Berlin Heidelberg: Springer, 2011.

26. Y. Hosokawa and Y. Nishio, "Chaotic circuit using a ring oscillator and a Van der pol oscillator," in International Symposium on Nonlinear Theory and its Applications NOLTA'08, pp. 285–288, Sept 2008.

27. Terasic Corporation Online Cyclone Main Boards, Terasic Altera DE2-115 Board User's Manual on the System CD, 4 2013.

28. Altera Corporation, Altera Corporation DSP Buider, 5 2013.

29. USA: The Mathworks Inc., MATLAB version 7.1 (R14), 2005.

30. Altera Corporation, Altera Corporation Quartus Web Edition, 5 2013.

31. Altera Corporation, Altera Corporation Modelsim-Altera Edition, 5 2013.