IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Bangkok, Thailand, November 2019.

On the Implementation of a Rotated Chaotic Lorenz System on FPGA

Hammam Orabi
Dept. of Computer Science and Engineering
American University of Sharjah
Sharjah, United Arab Emirates
horabi@aus.edu

Mohammed Elnawawy
Dept. of Computer Science and Engineering
American University of Sharjah
Sharjah, United Arab Emirates
b00049112@aus.edu

Assim Sagahyroon
Dept. of Computer Science and Engineering
American University of Sharjah
Sharjah, United Arab Emirates
asagahyroon@aus.edu

Fadi Aloul
Dept. of Computer Science and Engineering
American University of Sharjah
Sharjah, United Arab Emirates
faloul@aus.edu

Ahmed S. Elwakil
Dept. of Electrical & Computer Engineering, University of Sharjah
Sharjah, United Arab Emirates
(also with the *Dept. of Electrical and Computer Engineering, University of Calgary, Canada*) elwakil@ieee.org

Ahmed G. Radwan
Dept. Engineering Mathematics and Physics
Cairo University
Cairo, Egypt
agradwan@ieee.org

***Abstract*— Recent advances in engineering applications of chaos include multimedia security, disturbance modeling in power systems and performance enhancement of electronic circuits among others. This paper discusses the implementation of a rotated Lorenz chaotic system on a Field Programmable Gate Array (FPGA). Unlike most published work, we do not rely on Hardware Description Language (HDL) code generated from MATLAB/SIMULINK using general code conversion tools like HDL coder. Instead, we code the system and all of its modules using the Verilog HDL and highlight the benefits of taking this approach in allowing for a systematic design procedure that tends to minimize the complexity of the design, the number of design errors and simplifies the debugging process. We anticipate that by using Verilog HDL and not relying on converters-generated code, we can improve on hardware resource utilization, synthesis frequency, accuracy, and throughput. The platform used is the DE2-115 development board equipped with Altera Cyclone IV FPGA device.**

Keywords— Lorenz system, Chaotic systems, FPGA

I. INTRODUCTION

In recent years, Field Programmable Gate Arrays (FPGAs) have gained ground as feasible design platforms for prototyping complex digital systems since they prove to perform at an equivalent level with DSPs and microprocessors, especially when considering factors such as flexibility, and true parallelism. FPGAs have been used successfully in numerous designs with impressive performance outcomes [1-3].

Continuous-time chaotic systems and their fractional orders are used in several applications such as communications and encryption schemes [4-6]. Usually, analog circuits are used to realize these systems. However, the reliability of digital systems in general, tempted designers to implement chaotic systems digitally using FPGAs, then convert the system's digital variables to the analog domain using Digital to Analog Converters.

Examples of FPGA realization of chaotic systems include the work discussed in [7-9]. Noticeably, in most of the published work, a combination of MATLAB/Simulink or MATLAB DSP builder toolbox is used during implementation. These designs rely on programs such as HDL coder [10] to convert a MATLAB code or Simulink design to the equivalent hardware description language Verilog HDL or VHDL which in turn is used to synthesize the circuit for FPGA implementations. It is often the case that these programs will generate code that is far from optimal and in turn would lead to excessive consumption of FPGA resources, and sometimes even a delay in the speed with which an implementation can be executed.

In the work presented here, we propose an implementation approach of a rotated Lorenz chaotic system on an FPGA. Based on its specification, a system prototype is developed directly using Verilog HDL to avoid using MATLAB or Simulink during design entry; hence, the need for HDL converters in the development process. Such an approach would yield an optimized and more efficient implementation that uses less resources and therefore run faster when compared to implementations synthesized using a converter-generated Verilog HDL code. Furthermore, compared to Simulink/HDL Coder workflow, traditional forms of using hardware description languages can be highly optimized and offer a greater degree of control over pipelining, reset and enable behaviors [11].

II. BACKGROUND

In [12], it was shown that it is possible to rotate chaotic attractors and that the dynamics do not change as a result of this rotation because the eigenvalues at all equilibrium points remain unchanged. The authors considered first the original Lorenz system of differential equations given by:

$$\begin{aligned} x' &= a\,(y - x) \\ y' &= (b - z)\,x - y \\ z' &= xy - cz \end{aligned} \tag{1}$$

where (a, b, c) are user-specified parameters. Using an Euler discretization technique, this system can be described by:

$$\begin{aligned} x_{i+1} &= x_i + h * a\,(y_i - x_i) \\ y_{i+1} &= y_i + h\,((b - z_i)\,x_i - y_i) \\ z_{i+1} &= z_i + h\,(x_i * y_i - c * z_i) \end{aligned} \tag{2}$$

where h is a constant step.

This model forms the basis for building the rotated attractor that is comprehensively discussed in [12] and is described by the equation set below:

$$
\begin{aligned}
u_{i+1} &= u_i + h\,(a\cos\theta\,(T2 - T1) \\
&\qquad - sin\theta\,((b - w_i)\,T1 - T2)) \\
v_{i+1} &= v_i + h\,(a\sin\theta\,(T2 - T1) \\
&\qquad + cos\theta\,((b - w_i)\,T1 - T2)) \\
w_{i+1} &= w_i + h\,(T1 * T2 - c * w_i)
\end{aligned} \tag{4}
$$

where *u, v* and *w* are the rotated state variables while *T1*and T2 are related to the rotation angle in two-dimensions by:

$$
\begin{aligned}
T1 &= cos\theta * u_i + sin\theta * v_i \\
T2 &= sin\theta * u_i + cos\theta * v_i
\end{aligned} \tag{5}
$$

To code the system in Verilog our basis was the equation given in (2) (switched) and (4) (rotated version).

III. MATLAB PROTOTYPING

Using MATLAB, we built and simulated a model of the system described by equation (4) above. The Chaotic system in [12] was implemented for a finite number of rotation angles, namely, 0, −45, and −90 degrees. The rotation angles 0 and −90 were chosen on purpose to simplify the work needed in writing the Verilog modules to develop their sine and cosine values. This is because sin(0) = cos(−90) = 0 which implies that a sizable segment of the equation will cancel out, and hence we can simplify the system accordingly by avoiding expensive multiplication operations. However, and to illustrate the design complexity, the angle −45 was chosen to demonstrate the results when rotation angles other than 0 or−90 are needed. The values of the coefficients *a*, *b*, and *c* were set to be 10, 28, and 8/3, respectively in all experiments. For illustration, the realized Euler equations in the case of -45 degrees rotation are

$$T1 = 0.707\,(u_i - v_i) \tag{8.a}$$

$$T2 = 0.707\,(v_i + u_i) \tag{8.b}$$

$$u_{i+1} = u_i + 0.071\,(T2 - T1) + 0.007\,(T1\,(28 - w_i) - T2) \tag{8.c}$$

$$v_{i+1} = v_i + 0.007\,(T1\,(28 - w_i) - T2) - 0.071\,(T2 - T1) \tag{8.d}$$

$$w_{i+1} = w_i + 0.01\,(T1 \bullet T2 - \tfrac{8}{3} w_i) \tag{8.e}$$

Eventually, FPGA experimental results are compared to results from MATLAB for verification.

IV. IMPLEMENTATION

For comparison purposes, we first used MATLAB HDL coder to generate the Verilog code that describes the Lorenz attractor. As expected, the generated code was obscure and hard to optimize. Hence, a Verilog code that describes the system was developed from scratch. To further optimize this code, we opted for replacing the trigonometric operations by simple constant multiplications, that can be implemented using adders and shift operations. The design flow to implement the rotated chaotic system on FPGA is illustrated in Fig. 1. Following the design entry, one can verify the functionality of the design at different points, for example behavioral simulation before synthesis and gate level after synthesis. The implementation steps include translation or merging of input netlists and design constraints, mapping allocates the logic to the different FPGA components, and place and route which places and routes the design per the timing constraints. A clear advantage of this flow is the designer's ability to review reports generated at different phases of the implementation such as Map report or Place and Route report and subsequently further improve the design by changing for example source file code or the constraints. Hence, a clear advantage when compared to using MATLAB/SIMULINK is being able to smoothly iterate through several design trials till the most efficient design is reached.

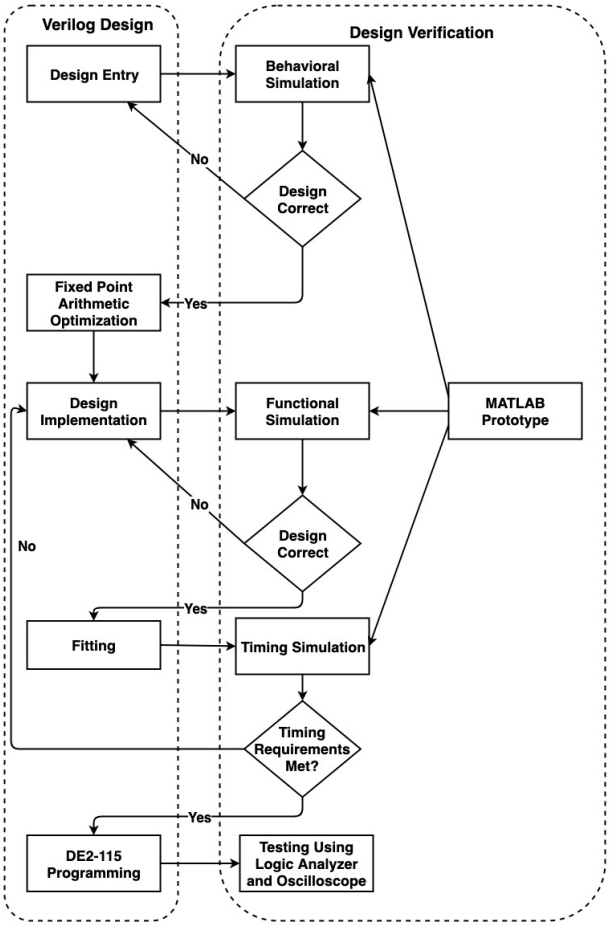

Fig. 1. Flowchart of the development process

A. Verilog HDL Modelling

Three different Verilog modules were developed, one for each rotation angle. Figure. 2 shows a high-level view of the module designed for the $-\pi/2$ rotation angle. The core computational units *Lorenz0*, *Lorenz45* and *Lorenz90*, are all pure combinational circuits, which are responsible for generating the next values of the three state variables u_{i+1}, v_{i+1}, and w_{i+1} given the previous values u_i, v_i, and w_i. The only sequential part in these modules is the register file in the feedback loop. This enables the new outputs corresponding of each set of inputs to develop in one clock cycle only with acceptable latency. Therefore, the outputs of the Lorenz modules are synchronized, and the results can develop and be captured in a predictable manner. At reset, all register files reset to the initial conditions of *u*, *v*, and *w*.

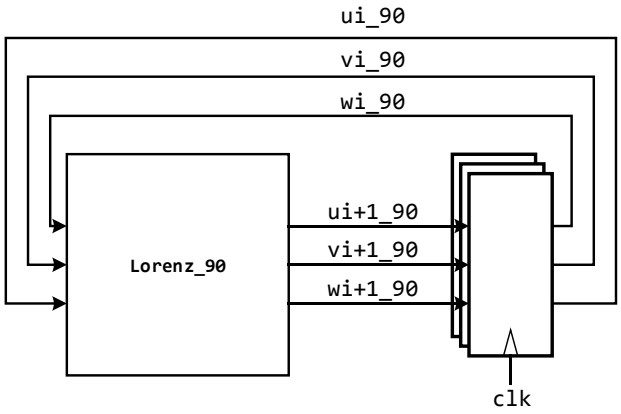

Fig. 2. Rotated Lorenz chaotic system with rotation angle = $-\pi/2$

The three modules are then encapsulated in a higher-level module that consists of a state machine to control the number of iterations of the system. In other words, the state machine limits the number of Euler steps performed towards estimating the solution of the main differential equations. Moreover, the encapsulation module contains multiplexers and selection logic modules to select between the rotated systems' output signals to be passed to the DAC, and eventually being displayed on the oscilloscope. To further clarify the operation of the selection logic, our Lorenz system can produce nine signals in total, *u*, *v*, and *w* of the three rotation angles. Therefore, the selection logic was necessary in order to specify which two signals are passed at a time to the DAC to be visualized on the oscilloscope as an *X*-*Y* two-dimensional plot. This selection logic control module has two modes of operation, the manual mode and the automatic mode. The manual mode enables the user to select only one of the three rotation angles and the two dimensions to be displayed on the scope of that rotation angle. Therefore, the resulting plot of the manual mode is either the *u*-*v* projection, *u*-*w* projection, or the *v*-*w* projection of the specified rotation angle. Whereas, the automatic mode enables the user to select two rotation angles and only one of the three projections. Consequently, the automatic mode enables the alternation between the two rotation angles at 30 frames per second refresh rate such that the same projection is displayed at two different rotation angles on the oscilloscope screen at the same time.

In summary, the selection logic shown in Fig. 3 can route the following pairs to the two DAC outputs (Sig_A, Sig_B) based on the user's input:

- *(u_i,v_i), (u_i,w_i),* or *(v_i,w_i); $\forall i \in [0,-\pi/4,-\pi/2]$,* or
- an alternation between *(u_i,v_i),* and *(u_j,v_j), $\forall i,j \in [0,-\pi/4,-\pi/2]$, @30Hz,* or
- an alternation between *(u_i,w_i),* and *(u_j,w_j), $\forall i,j \in [0,-\pi/4,-\pi/2]$, @30Hz,* or
- an alternation between *(v_i,w_i),* and *(v_j,w_j), $\forall i,j \in [0,-\pi/4,-\pi/2]$, @30Hz.*

B. The Fixed Point Architecture

Throughout the entire design, 64-bit signed fixed-point arithmetic was exclusively implemented. All internal signals and wires were denoted according to *(1.m.n)* convention where *m* is the number of bits for the integer part of the number, and *n* is the number of bits dedicated for the fraction part. Of course, *1+m+n = 64*. The number of bits allocated for *m* and *n*, was decided according to the maximum expected values of *u*, *v*, and *w*, throughout the entire run duration.

To further optimize the design, increase its efficiency, and save on FPGA resources, all multiplications by single constant numbers were reduced to shift operations using the approximation algorithm described in [13]. For example, multiplying by 0.01 is approximately equivalent to multiplying by 41/4096; which is equal to (32 + 8 + 1)/4096 or $(2^5 + 2^3 + 2^0)/2^{12}$. Therefore, instead of using the slow and resource consuming multiplication module, the addition needs two simple binary adders only, while the shift operation does not consume logic, since we have used fixed point notation. Figure 4 shows the data flow graph of the *Lorenz90* module after replacing single constant multiplications with additions and shift operations.

The internal architectures of *Lorenz0* and *Lorenz45* look very similar to that of *Lorenz90*. The dotted lines indicate slicing registers that segregate the inputs from the outputs. Fixed point notation was used instead of floating point since it requires simpler mathematical modules, and is easier in terms of synchronization between the developed results.

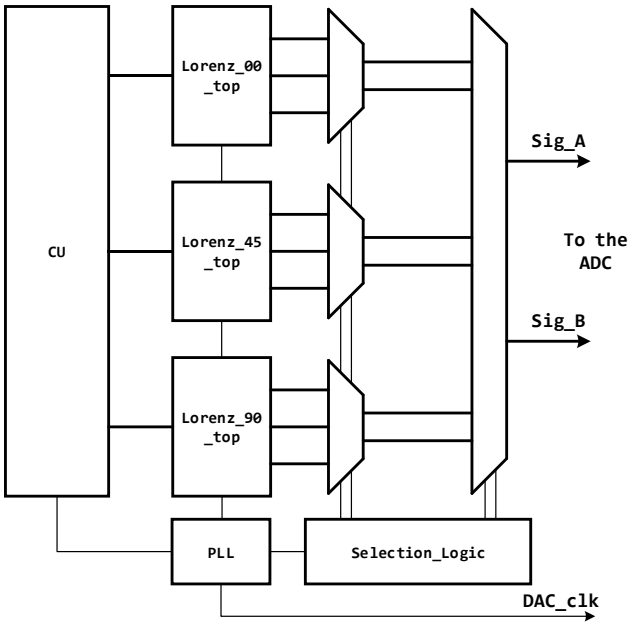

Fig. 3. The top-level module depicting the Lorenz blocks and output selection logic

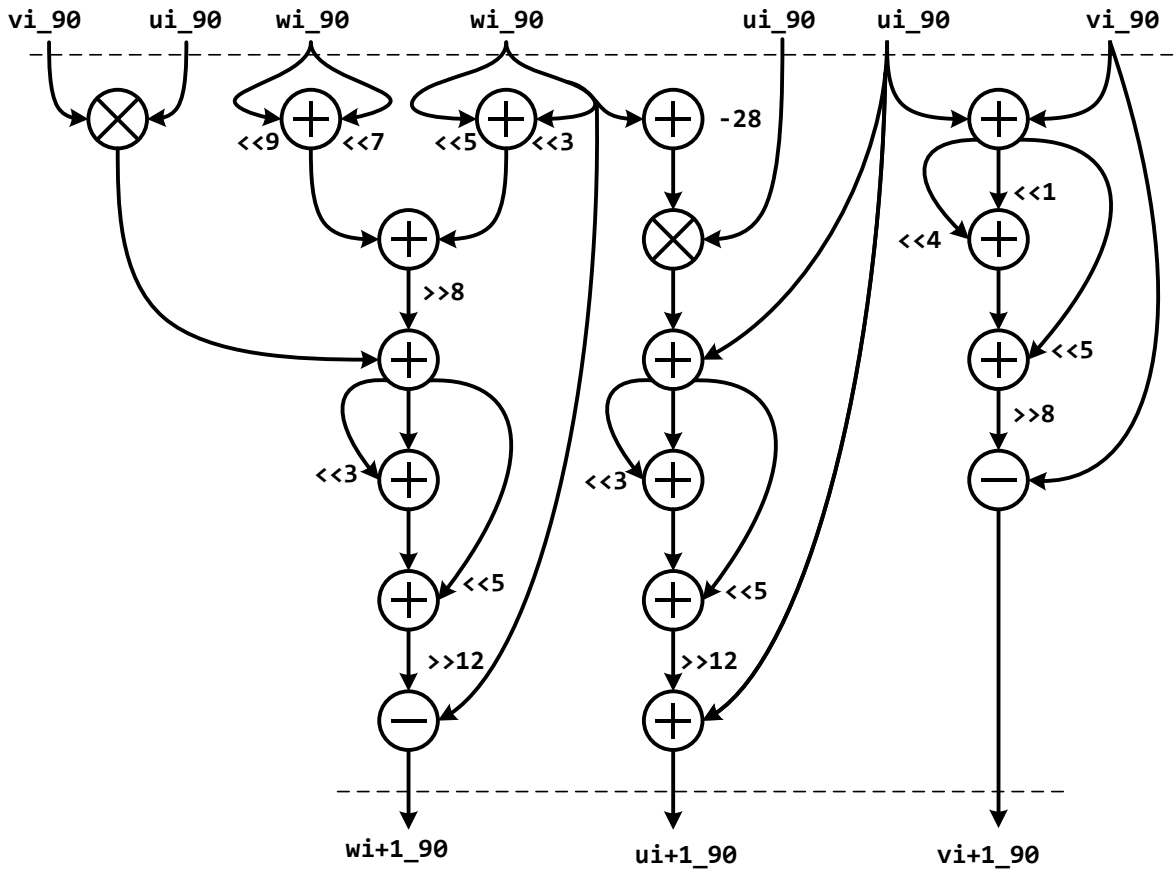

Fig. 4. The internal architecture of *Lorenz90* module, showing the arrangement of adders, multipliers and shift operations

Moreover, floating point operations usually require more than one clock cycle to develop their results, whereas, fixed point computations are usually performed within a single clock cycle. Nevertheless, one of its drawbacks is the fact that it requires careful consideration of the alignment of the location of the binary points in the operands of any operation. For example, this crucial matter is depicted in the *Lorenz0* module where a subtraction operation between two numbers: (1.22.41) and (1.16.49) results in a fixed point number of (1.23.40). As a result, the two operands need to be arithmetically right shifted by 1 bit and 7 bits, respectively.

C. Timing Analysis

Timing analysis of the designed hardware was performed using Timing Quest Timing Analyzer tool in Intel Quartus Prime software [14]. Upon analyzing the behavior of the developed hardware, the input and output constraints of the Lorenz modules were set to a minimum delay of 2ns and a maximum delay of 3ns. After running the timing analysis, the timing analyzer indicated that the maximum allowed frequency to operate the Lorenz modules without violating any timing constraints is 15MHz. A Phase Locked Loop (PLL) was used to generate two different clock signals; the main clock signal runs at 15MHz to synchronize the operation of all the Lorenz modules with the DAC, and a 30Hz clock signal to work as a refresh rate for the oscilloscope display.

V. FPGA PROTOTYPE

The platform used is the DE2-115 development board equipped with Altera Cyclone IV EP4CE115 FPGA device [15]. This platform provides plentiful of IO peripherals, including sliding switches, push buttons, general purpose IO bank, and a High Speed Mezzanine Card (HSMC) connector. These peripherals were used to handle the system's inputs and outputs, for debugging purposes, and to connect the system's digital output to the external Digital to Analog Converter for plotting it on the oscilloscope. A single unit of Texas Instruments high-speed Digital to Analog Converter (TI DAC5672) [16] mounted on Altera's Data Conversion HSMC card. This DAC unit can connect to the DE2-115 board using the HSMC connector [17]. Data is transferred from the motherboard to the daughter board in parallel, hence, eliminating the need of using serial transfer protocols, which in turn speeds up the process of digital to analog conversion and hence speeds up the overall execution of the system. The data conversion card connects to the DE2 development board using the HSMC interface as seen in Fig. 5.

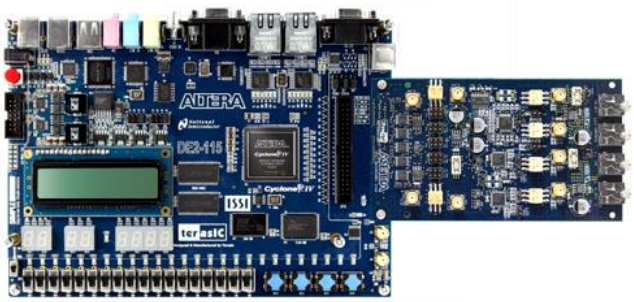

Fig. 5. DE2-115 equipped with the Terasic Data Conversion HSMC Card

A. *Digital to Analog Conversion*

TI DAC5672 is a monolithic, dual channel, 14-bit, high speed Digital-to-Analog converter with on-chip voltage reference. The MATLAB simulation indicated that the maximum and minimum values of the *u*, *v*, and *w* outputs are expected to lie within 6 binary bits. Therefore, the fixed point structure of the delivered output signals was represented as (1.6.7) in (*1.m.n*) notation.

Figure 6 shows how the top-level module of our design, interfaces with the DAC unit.

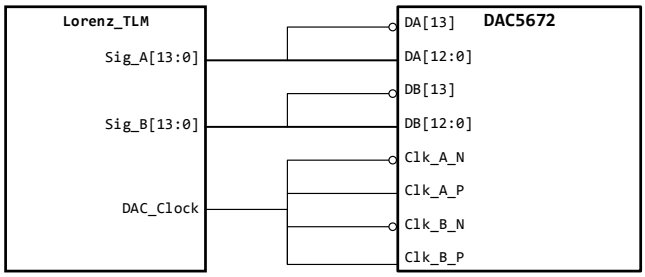

Fig. 3. Top level module connection with the DAC unit

B. *Testing and Results*

In often cases, the design works fine on simulation but behaves differently on hardware. Hence, there arises a need to debug, and monitor buses and signals on FPGA. This task is primarily accomplished by using a logic analyzer. In this work, we tested our design using Tektronix LA6401 logic analyzer and the projections were plotted on Tektronix TDS2002C digital oscilloscope. Figure 7 shows the connection between the FPGA and the Logic Analyzer.

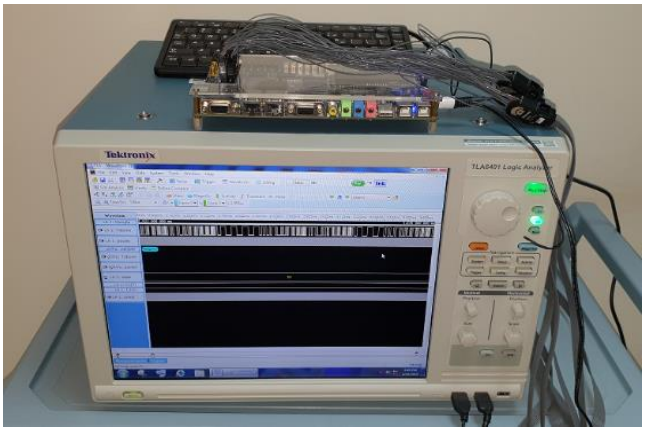

Fig. 4. Logic analyzer setup

The final setup of our implementation including the DE2-115 board, the DAC card, and the oscilloscope is shown in Fig. 8. The resource utilization of the implementation is summarized in Table 1. Table 2 lists the mean absolute error (MAE) between the results captured from the logic analyzer and the corresponding values from MATLAB for the first 5000 iterations.

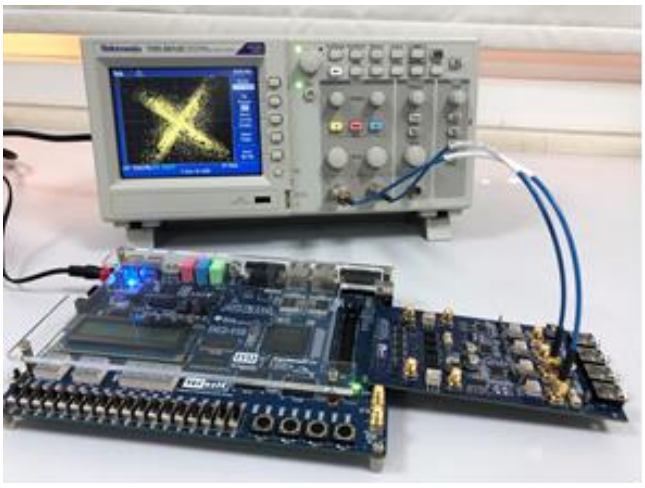

Fig. 5. Testing using the digital oscilloscope

Figure 9 shows the captured plots from the digital oscilloscope's display for the *u*-*v* projection of the different rotation angles using both the manual mode and the automatic mode.

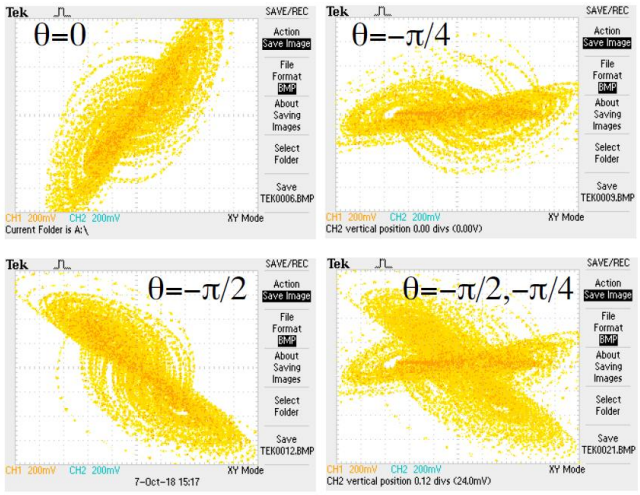

Fig. 6. Oscilloscope pictures of the rotated plots

TABLE I. RESOURCE UTILIZATION

Resources	Our design
Logic Elements	3543/114480 (3%)
Combinational Functions	3490
Registers	736
Embedded multiplier (9-bit)	48/532 (9%)
Pins	41/529 (8%)
PLL	1
Frequency	15 MHz

TABLE II. MAE FOR THE DIFFERENT ROTATIONS

Degree	MAEu	MAEv	MAEw
0	0.0717	0.0695	0.0624
−45	0.0617	0.0571	0.0540
−90	0.0407	0.0435	0.0366

VI. CONCLUSION

In this paper, we discussed an FPGA-based implementation of a rotated Lorenz chaotic attractor. The implemented attractor can rotate the system using three rotation angles 0, $-\pi/4$, and $-\pi/2$. We detailed the hardware

design process spanning the initial prototyping phase passing through the functional and timing simulation of the implemented system and ultimately displaying the obtained outputs on a digital oscilloscope display. We developed the modules using the traditional design entry method using the Verilog Hardware Description Language (HDL) instead of HDL code generation tools like MATLAB's HDL coder. This enables us to optimize the FPGA resource utilization, enhance the overall performance of the generated code and write a simple and modularized HDL code. The designed Lorenz system was implemented on the DE2-115 development board. In testing the designed system, a logic analyzer and a digital oscilloscope were used in order to verify the outcome of the designed hardware compared to the MATLAB simulation. The designed system is capable of running at 15MHZ. Future enhancements of the system include the possibility of implementing any arbitrary rotation angle, and further optimizing the Lorenz attractor to run at a much higher frequency.

REFERENCES

[1] C. Y. Wu, C. Y. Liao, D. Lee, Y. S. Cheng, C. H. Huang, J. Chen, K. H. Hu, and K. T. Hsu, "An FPGA-Based Quench Detector and Data Acquisition System for Superconducting Insertion Devices," *IEEE Transactions on Applied Superconductivity*, vol. 28, no. 3, pp. 1–5, 2018.

[2] C. H. Yang, H. C. Wu, and S. F. Su, "Implementation of Encryption Algorithm and Wireless Image Transmission System on FPGA," *IEEE Access*, vol. 7, pp. 50513–50523, 2019.

[3] X. Cai, M. Zhou, T. Xia, W. H. Fong, W. T. Lee, and X. Huang, "Low-Power SDR Design on an FPGA for Intersatellite Communications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 11, pp. 2419–2430, 2018.

[4] G. Kaddoum, "Wireless Chaos-Based Communication Systems: A Comprehensive Survey," *IEEE Access*, vol. 4, pp. 2621–2648, 2016.

[5] L. Kocarev and S. Lian, "Chaos-Based Cryptography," *Studies in Computational Intelligence*, vol. 354, 2011.

[6] F. C. M. Lau and C. K. Tse, "Techniques for Non-Coherent Detection in Chaos-Based Digital Communication Systems," *Chaos-Based Digital Communication Systems Signals and Communication Technology*, pp. 205–217, 2003.

[7] D. K. Shah, R. B. Chaurasiya, V. A. Vyawahare, K. Pichhode, and M. D. Patil, "FPGA implementation of fractional-order chaotic systems," *AEU - International Journal of Electronics and Communications*, vol. 78, pp. 245–257, 2017.

[8] S. Sadoudi, M. S. Azzaz, M. Djeddou, and M. Benssalah, "An FPGA Real-time Implementation of the Chen's Chaotic System for Securing Chaotic Communications," pp. 467–474, 2009.

[9] Q. Ding, J. Pang, J. Fang, and X. Peng, "Designing of chaotic system output sequence circuit based on FPGA and its applications in network encryption card," *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 2, pp. 449–456, Apr. 2007.

[10] "HDL Coder," *MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/products/hdl-coder.html. [Accessed: 04-Aug-2019].

[11] [V. Y. Sarge, "Evaluating Simulink HDL coder as a framework for flexible and modular hardware description," thesis, 2018.

[12] W. Sayed, A. Radwan, M. Elnawawy, H. Orabi, A. Sagahyroon, F. Aloul, A. Elwakil, H. Fahmy, and A. Elsedeek, "Two-Dimensional Rotation of Chaotic Attractors: Demonstrative Examples and FPGA Realization" Journal Circuits, Systems, and Signal Processing (CSSP), April, 2019; Springer

[13] Vahid, Frank. Digital Design with RTL Design, Verilog and VHDL. John Wiley & Sons, 2010.

[14] *Timing Analyzer Resource Center*, 10-Apr-2019. [Online]. Available: https://www.intel.com/content/www/us/en/programmable/support/support-resources/design-examples/design-software/timinganalyzer/sof-qts-timinganalyzer.html. [Accessed: 04-Aug-2019].

[15] *Cyclone II Device Handbook, Volume 1*, Altera. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyc2/cyc2_cii5v1.pdf

[16] *Texas Instruments*. [Online]. Available: http://www.ti.com/product/DAC5672. [Accessed: 04-Aug-2019].

[17] https://www.terasic.com.tw