SYNCHRONIZATION OF CHAOTIC SYSTEMS: FIELD PROGRAMABLE GATE ARRAY AND NONLINEAR CONTROL FEEDBACK APPROACH

[1]A. Castillo Atoche, [1]G. Solís Perales, [1]A. Matos Gamboa and [2]R. Atoche Enseñat

[1]FACULTAD DE INGENIERÍA
UNIVERSIDAD AUTÓNOMA DE YUCATAN
Mérida Yucatán México, Av. Industrias no contaminantes S/N
Tel: (+52) 999 9410194 ext. 134
[2]INSTITUTO TECNOLOGICO DE MERIDA
Mérida Yucatán México, Av. Tecnológico Km 4.5
Tel: (+52) 999 9448113, 9448122 ext. 163

acastill@uady.mx, perales@uady.mx, jatoche@itmerida.mx

ABSTRACT

In this work an implementation of a geometric nonlinear controller for chaos synchronization in a Field Programmable Gate Array (FPGA) is presented. The Lorenz chaotic system is used to show the implementation of chaos synchronization via nonlinear controller implemented in a Xilinx FPGA Virtex-II 2v2000ft896-4. The main idea is to design a nonlinear geometric controller which synchronizes a slave Lorenz system to a master system and then implement them into the FPGA. A nonlinear geometric controller and the two Lorenz systems have been implemented in the FPGA. The synchronization via control consists of leading the slave trajectories to the master system trajectories. The purpose is to illustrate that a reconfigurable device can perform a complicated operation such as the chaos synchronization. The verification of each Lorenz generator was co-simulated with Matlab/Simulink and Xilinx System Generator.

1. INTRODUCTION

The synchronization of chaotic systems is an interesting research topic, which has attracted much attention in the last two decades. Since Pecora and Carrol presented a paper showing the synchronization in chaotic systems [1], a lot of results have been published see for instance [2,3]. Chaotic systems have a lot of applications, for example, the synchronization in biological systems [4], synchronization applied to secure communications [5], and recently synchronization has been applied to robot synchronization [6].

Since chaotic behavior comes from nonlinear dynamical systems, it is highly dependent on initial conditions and parameter values and a nonlinear controller is adequate for achieve chaos synchronization. The control objective is to obtain a controller that stabilizes a chaotic system at the origin. Such a system represents the dynamic error between the master and slave systems. Therefore, we propose a nonlinear controller obtained from the nonlinear geometric control theory [7,8]. The dynamical error system is obtained by the difference between the two Lorenz generators systems. Figure 1 shows the synchronization problem as a stabilization of trajectories. In this figure the trajectories of the new system (synchronization error system) contained in the square are leaded to the origin via the controller C.

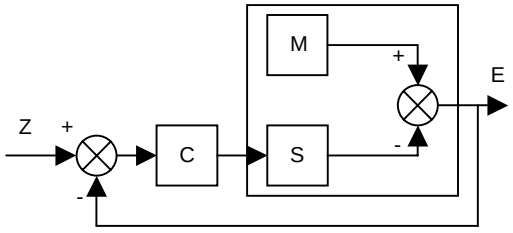

Figure1. Block representation of chaos synchronization as stabilizing trajectories.

There are some important results in literature showing the synchronization of chaotic systems via nonlinear controllers (see for instance [5], [9]), however, the novelty of the present work is the implementation of the controller and both master and slave systems into a reconfigurable architecture.

Advances in VLSI process technology have been applied to the manufacturing of reconfigurable logic including field programmable gate arrays (FPGA) chips and helped their rapid growth in logic capacity,

performance and popularity. The extreme flexibility and the widespread acceptance of hardware description languages such as VHDL and Verilog have made FPGA-based systems the medium of choice for the hardware development and implementation of high-performance compute-intensive applications. The System Generator block is the program that generates VHDL code for the Simulink model that has been created using the Xilinx blocks. Using System Generator digital designers can implement DSP applications into digital designs faster and easier using Simulink and a simple design methodology. This device allows the functional verification with co-simulation [10]. One of the powerful advantages of this device is that the control system parameters and frequencies of the controller and Lorenz oscillators can be really fixed. This FPGA characteristic provides a powerful tool for dynamics control of nonlinear complex systems.

2. SYNCHRONIZATION PROBLEM FORMULATION

As was discussed above, chaos synchronization can be addressed as a stabilization problem. This means that the trajectories of the synchronization error have to be stabilized at the origin. First of all, let us consider the definition for complete chaos synchronization.

Definition 1.[9] Two chaotic systems are completely synchronized if the error $\|x_M - x_S\| \to 0$ as $t \to \infty$, where $x_M, x_S \in \Re^n$.

Note that this definition does not depend on the synchronization technique used; it is a global definition for complete synchronization. Definition 1 means that each state in the slave system is identical or is very close to its corresponding state in the master system along the time. Chaos synchronization as stabilization consists in finding a control command that leads the trajectories of a dynamical error system to the origin. To this end, consider two same order chaotic systems in affine form

$$\sum_M := \dot{x}_M = F_M(x_M), \qquad y_M = h_M(x_M)$$
$$\sum_S := \dot{x}_S = F_s(x_s) + g(x_S)u \qquad y_s = h_s(x_S) \qquad ...(1)$$

Where $y_{k,}$ are smooth real valued output functions, F_k: $D \to \Re^n$ with the domain $D \subset M \subset \Re^n$ are smooth vector fields and M is a smooth manifold, k = M,S, $g(x_S)$ defines the control channel, u is the control command.

Now, the synchronization error system is given by

$$\sum_E = \sum_M - \sum_S := F_M(x_M) - (F_S(x_S) + g(x_S)u) \qquad ...(2)$$

performing the exchange $x_e = x_M - x_S$, we find the so called synchronization error system

$$\dot{x}_e = f_e(x_e) - g(x_e)u, \qquad y_e = h(x_e) \qquad ...(3)$$

where f_e is defined into an open subset $\delta \subset \Delta \subset \Re^n$. x_e is the vector of the state error.

Note that the trajectories must be loaded to a small neighborhood $\partial \subset \delta$ for complete the synchronization. The small neighborhood ∂ is called synchronization manifold and contains the point x_e^o which is the origin. In order to lead the synchronization error trajectories of the synchronization error system to the origin x_e^o via feedback control, Σ_E must satisfy local controllability and local observability [7,8]. In other words, both master and slave systems should be synchronizable (see [8] for details on sinchronizability of similar chaotic systems). Therefore, the complete chaos synchronization between strictly different systems can be solved by means of a control system that stabilizes every state of the synchronization error system at the origin.

3. COMPLETE CHAOS SYNCHRONIZATION

Now from the synchronization error system, we need to calculate a set of controllers that lead the trajectories to the origin x_e^o. Such controllers are determined from the relative degree condition [8,9]. The controllability and observability conditions of dynamical error system (2), is checked via the relative degree.

Definition 2. A nonlinear system in affine form has relative degree Δ at x^o if

(i) $L_g L_{f_e}^k h(x_e) = 0$

for all integer $k < \rho - 1$, and for all x_e in a neighborhood of x^o.

(ii) $L_g L_{f_e}^{\rho-1} h(x_e) \neq 0$

Now suppose that the synchronization error system has relative degree $\rho = n$ the system is completely linearizable by feedback, but if $\rho < n$ at x_e^o the synchronization error system is partially linearizable by feedback. The relative degree condition implies the existence of a diffeomorphic transformation which transforms the synchronization error system into a linearizable equivalent form. Such transformation is based on the Lie derivatives of the synchronization error system output $z_i = L_{f_e}^{i-1} h(x_e)$ with $i = 1,2\ldots\rho$. Therefore, applying the transformation one can obtain the control command

$$u = a(x_e) + b(x_e)v \qquad (4)$$

where $v = \kappa_i(z_i - z_i^*)$ where z^* is the desired stabilization point, and the smooth functions $a(x_e)$ and $b(x_e)$ are given by

$$a = \frac{L_{f_e}^{\rho} h(x_e)}{L_g L_{f_e}^{\rho-1} h(x_e)} \qquad b = \frac{\nu}{L_g L_{f_e}^{\rho-1} h(x_e)} \tag{6}$$

The controllers given by (4) ensure the stabilization of the corresponding output states at the origin. This means that $\|x_M - x_S\| \to 0$ when $t \to \infty$. It is important to note that the controller is designed for the transformed system, however, due to the invertibility of the transformation, the control command can be applied to the error system (2). The controller (4) is such that the states of the chaotic dynamical error remain at the origin. This can be done by choosing the control gains κ such that the closed loop system be stable. The next step is the implementation in the FPGA.

4. IMPLEMENTATION AND SIMULATION OF THE SYNCHRONIZATION SCHEME

Now we illustrate the simulation and implementation of the control system, master and slave chaotic systems using Matlab/Simulink and Xilinx system generator blocks [10,11]. Let us consider the Lorenz equation for the master

$$\begin{aligned} \dot{x}_M &= \sigma(y_M - x_M) \\ \dot{y}_M &= \pi x_M - y_M - x_M z_M \\ \dot{z}_M &= x_M y_M - \beta z_M \end{aligned} \tag{7}$$

And for the slave system we consider the same equations but probably with different parameter values

$$\begin{aligned} \dot{x}_S &= S(y_S - x_S) \\ \dot{y}_S &= P x_S - y_S - x_S z_S + u \\ \dot{z}_S &= x_S y_S - B z_S \end{aligned} \tag{8}$$

Figure 2 shows the block diagram which consists of the Lorenz chaotic generator. In synchronization, two blocks diagrams are constructed, one for the master and other for the slave.

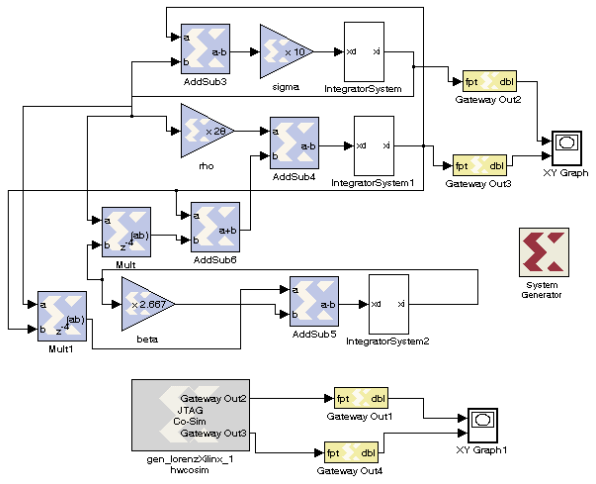

Figure 2. Chaotic Lorenz generator using Matlab/Simulink and Xilinx System Generator blocks

Now following the procedure in previous section, we can obtain the synchronization error system as follows

$$\begin{aligned} \dot{x}_e &= \sigma(y_e - x_e) \\ \dot{y}_e &= \pi x_e - y_e + x_e z_e + x_e z_M - x_M z_e - u \\ \dot{z}_e &= x_e y_e - \beta z_e + x_e y_m + x_M y_e \\ h_e &= x_e \end{aligned} \tag{9}$$

Note that the output signal is given by the difference between the corresponding outputs in each system. It is important to note that chaotic behavior mainly depends on two facts, the initial conditions and parameter values, therefore for simplicity and without loss of generality, consider S = σ, P = π and B = β, and different initial conditions between systems.

From Definition 2 the relative degree of the error system is ρ = 2 and the function b in (6) is defined for all X = [x_e y_e z_e] in the domain D. The transformation is given by Φ(X) = [x_e s(y_e - x_e) z_e]^T which is invertible for all X = [x_e y_e z_e] in the domain D. Therefore the controller is

$$\begin{aligned} u = \frac{1}{\sigma}(&-(-\sigma(y_e - x_e) \\ &+ \sigma(\rho x_e - y_e + x_e z_e + x_e z_M - x_M z_e)) \\ &+ \kappa_1(z_1 - z_1^*) + \kappa_2(z_2 - z_2^*)) \end{aligned} \tag{10}$$

Where κ_1= 1, κ_2 = 2 and z_1^* = z_2^* = 0. Therefore, systems (7), (8) and controller (10) is implemented into the FPGA.

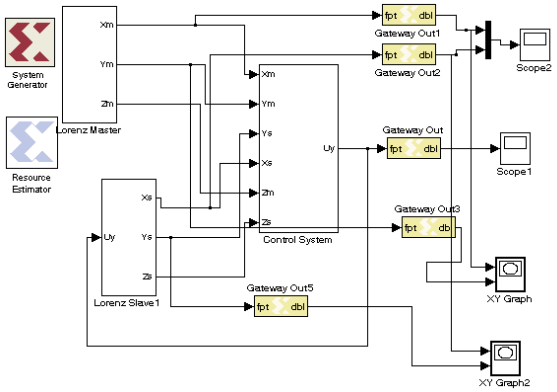

Figure 3. Chaos synchronization diagram, including the controller and both master and slave Lorenz system.

Figure 3 shows the block diagram of three systems. In order to practical realization and achieve synchronization for each Lorenz generator subsystem, all the number of bits was taken equals to 64 and binary point equal 32. This is, the number of bits used to represent the integer part is 64 and the decimal part is 32.

Each block subsystem called Lorenz contains the Lorenz chaotic generator (see Figure 2) designed with the Xilinx System block set. Simulation of Matlab/Simulink of the subsystems in open-loop (no control action) results in the chaotic attractor in Figure 4. Each Lorenz generator design was implemented and Co-Simulated with the System Generator tool of the Matlab/Simulink platform [12]. The Co-Simulation helps to verify in real time the

simulink simulation versus the hardware design. An important part in the co-simulation process is that the gray block represents the digital design that must be downloaded into the FPGA via the JTAG cable. So, in the moment that the simulation begins, we can compare both results in real time. If the JTAG cable of the FPGA is disconnected, then the trajectories of the phase portrait in Figure 4 stops, since the incoming signal from the FPGA is broken.

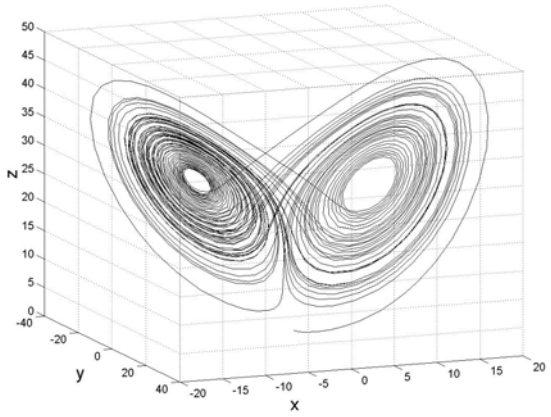

Figure 4. Chaotic attractor for the Lorenz subsystem, generated by the co-simulation process with the FPGA.

Concerning the synchronization, the controller and both master and slave Lorenz generators were implemented into a reconfigurable architecture with an FPGA. This device is capable to run the process at 17MHz. Figure 5 shows system controller implemented with xilinx system generator blocksets.

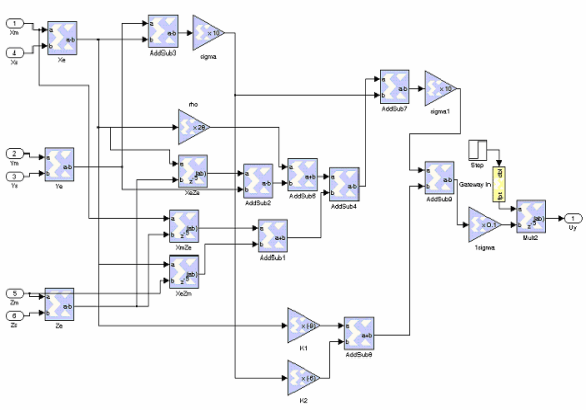

Figure 5. Block diagram for the nonlinear controller used to synchronize both chaotic systems.

Figure 6 shows the synchronization results, where the error states are leaded to a small δ neighborhood which contains the origin. Note that after a small transient behavior, the trajectories go to zero and remain there. This is done by the control action.

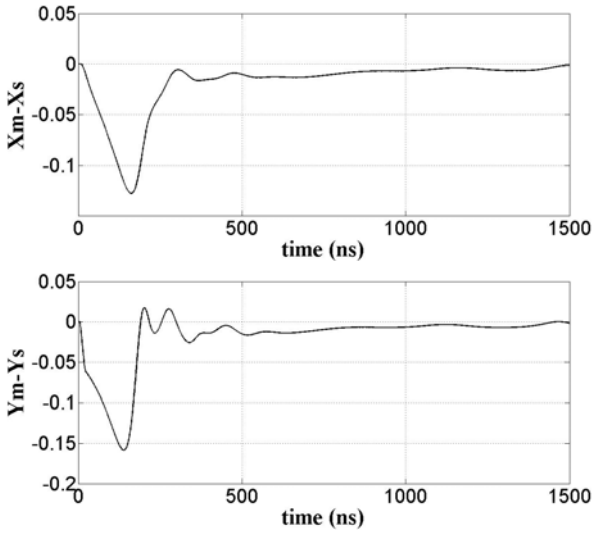

Figure 6. Stabilization of the x_e and y_e error stare at the origin.

Figure 7 shows the synchronization error z_e and the control action applied to the slave Lorenz subsystem. Again the transient takes about 500 ns to disappear and then the synchronization is achieved. The control action U_y is generated by the controller implemented as in Figure 5.

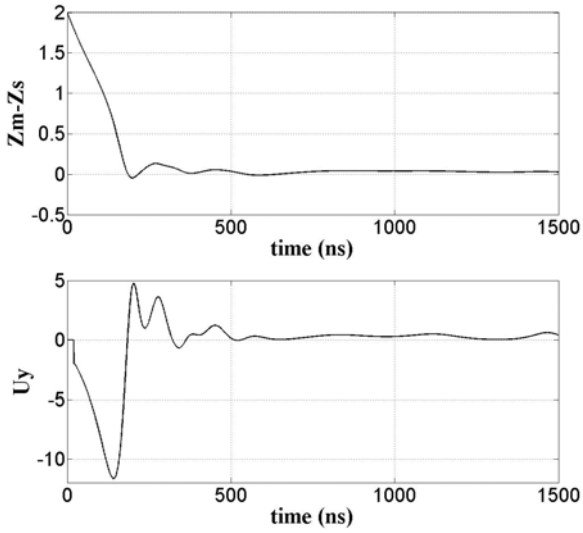

Figure7. Stabilization of synchronization error z_e around the origin and the control action applied to slave system.

The FPGA design was synthesized with Xilinx ISE-FOUNDATION tool. The synthesis process maps the VHDL code into hardware logic gates for a specific technology library [13]. The results obtained in the synthesis were target in a FPGA Virtex-II 2v2000ff896-4. In Figure 8, it can be seen the FPGA resources implemented in the design.

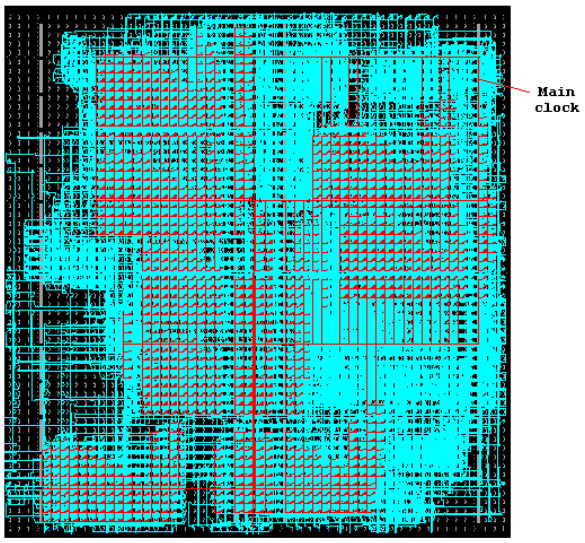

Figure8. Resources implemented in FPGA

In order to be able to carry out a verification of the design, the developed device has been simulated with ModelSim. Functional verification was realized as well. In Table1 and Table2, the synthesis and timing results are presented.

SYntHesis Report **FPGA:** 2v2000ff896-4		
Number of Slices	6055 out of 10752	56%
Number of BUFGMUXs	1 out of 16	6%
Number of IOBs	193 out of 624	30%
Number of GCLKs	1 de 16	6%

Table1. Synthesis Report

TIMING REPORT **CHAOTIC GENERATOR** **FPGA:** 2v2000ff896-4	
Minimum Period	64 ns
Maximum Frequency	16 MHz

Table2. Timing Report

6. CONCLUSIONS

A nonlinear geometric controller and two Lorenz systems were implemented inside a FPGA for synchronization. This is, a nonlinear geometric controller was design to synchronize two Lorenz chaotic systems. The controller performance was such that the synchronization of two nonlinear complex systems is achieved. Moreover, the parameters of the controller and the chaotic systems are time invariant, which provides robustness to the architecture. On the other hand, System Generator provides an efficient way to design dynamic nonlinear control system utilizing a single FPGA board with hardware co-simulation. Therefore reconfigurable devices provide a very powerful tool for the control of complex nonlinear systems, with a high robustness and accuracy. The next step is to implement the synchronization of chaotic systems with different models and more complicated systems with different order, however, results in this direction will be published shortly.

Acknowledgements: G. Solís Perales thanks to PROMEP for supporting part of this investigation under the grant PROMEP-FING-104.5

REFERENCES

[1] Pecora L.M. and Carrol T. “Synchronization in chaotic systems”, *Phys. Rev. Lett.*, 64, 1990

[2]Kapitanik T., Sekeita M. and Ogorzalek M., “Monotone synchronization”, Int. Jour. Of Bifur and Chaos, 6, 1996

[3]Rulkov N. Sushchik M.M and Tsimring L.S., “Generalized synchronization in chaos of directly coupled chaotic systems,” *Phys. Rev. E*, 51, 1995.

[4]Lading B., Mosekilde E., Yanchuk S. and Maistrenki Y., “Chaotic synchronization between coupled pancreatic β-cells,” *Prog. Theor. Phys. Suppl.*, 139, 2000

[5]Femat R. Jauregu-Ortiz R. and Solís-Perales G., “A chaos-based communication scheme via robust asymptotic feedback,” *IEEE Trans. On Circ. And Syst .I*, 48, 10, 2001

[6]Nijmeijer H. and Rodriguez-Angeles A. *Synchronization of Mechanical Systems*, World Scientific, 46, 2003

[7]Nijmeijer H. and van der Schaft A., *Nonlinear Dynamical Control Systems*, Springer-Verlag 1990

[8]Femat R., Alvarez-Ramírez and Fernandez-Anaya G., “Adaptive synchronization of high-order chaotic systems: a feedback with low-order parameterization,” *Physica D*, 139, 2000

[9]Femat R. and Solis-Perales G., “On the chaos synchronization phenomena,” *Phys. Lett. A*, 262 1999.

[10]Aseeri, M.A.; Sobhy, M.I.; Lee, P.; “Lorenz chaotic model using Filed Programmable Gate Array (FPGA)” Midwest Symposium on Circuits and Systems, 2002.

[11]Le Thuyen, Renner Frank, Glesner M., “Hardware in-the-loop Simulation –A Rapid Prototyping Approach for Designing Systems”, Proceedings, 8th IEEE International Workshop, June 1997.

[12]Grega Wojciech, “Hardware-in-the-loop simulation and its application in control education”, 29th ASEE/IEEE Frontiers in Education Conference, November 10 - 13, 1999 San Juan, Puerto Rico.

[13]Clive Maxfield, “The design warrior's guide to FPGA's”, Elsevier 2005.