Turkish Journal of Science & Technology
Volume 13(1), 25-29, 2018

Implementation of Two Cell Non-Autonomous CNN Model on FPGA

Bariş Karakaya*, Vedat Çelik, Arif Gülten
Fırat University, Electrical – Electronics Engineering, Faculty of Engineering, 23119 Elazig, Turkey
*bkarakaya@firat.edu.tr

(Received: 05.01.2017; Accepted: 27.02.2017)

Abstract

This paper presents implementation of a chaotic Cellular Neural Network (CNN) on Field Programmable Gate Array (FPGA). The network has two non-autonomous cells and exhibits chaotic behavior. In the implementation stage, Verilog Hardware Description Language (HDL) is used and discrete time model of the network is coded on Xilinx ISE Design Suite 13.2. It seems that the chaotic attractor can be used as entropy source or short key (seed) of chaos based random number generator design.

Keywords: Cellular Neural Network, Chaos, FPGA, Random Number Generator.

İki Hücreli Özerk Olmayan HYSA Modelinin FPGA'da Gerçeklenmesi

Özet

Bu çalışma, kaotik bir Hücresel Yapay Sinir Ağı'nın (HYSA) Alanda Programlanabilir Kapı Dizileri'nde (FPGA) gerçeklenmesini sunmaktadır. Ağ, özerk olmayan iki hücreye sahiptir ve kaotik davranış sergilemektedir. Gerçekleme aşamasında, Verilog Donanım Tanımlama Dili kullanılmakta ve ağın ayrık zamanlı modeli Xilinx ISE Design Suite 13.2'de kodlanmaktadır. Görülmektedir ki kaotik çeker, kaos tabanlı rasgele sayı üreteci tasarımında entropi kaynağı veya tohum olarak kullanılabilir.

Anahtar Kelimeler: Hücresel Yapay Sinir Ağı, Kaos, FPGA, Rasgele Sayı Üreteci.

1. Introduction

Chaos can be defined as unpredictable behaviors that are sensitive to initial conditions in a nonlinear deterministic system [1]. As chaotic behavior can be undesirable in some applications, there are a lot of application areas which use the chaotic behavior in their structures. In [2], the use of the Field Programmable Gate Array (FPGA) as a controller of a DC-DC boost converter, controlling the output current of a photovoltaic cells and minimizing the effect of the boost converter chaotic behavior on the output voltage are discussed. Micro controllers, Digital Signal Processors (DSP) and FPGAs can be used to design implementations of chaotic systems [3-6]. FPGA realizations of chaotic systems are studied in [7, 8] by using Euler algorithm with both integer arithmetic and floating-point number format. Especially, the Lorenz's chaotic system is implemented onto a FPGA with the help of the fourth order Runge-Kutta (RK4) algorithm in [9]. The chaotic behavior can be used in Random Number Generator (RNG) applications in order to generate exactly true random number series which are required for secure communication systems. In literature, the realizations of chaotic systems are proposed by using the Xilinx System Generator technology in order to have the HDL code. In [10], the Lorenz's chaotic system is implemented onto FPGA to obtain chaotic sequence for information security issue and in this design the Xilinx System Generator technology is also used. As Cellular Neural Network (CNN) proposed by Chua [11, 12] has complex dynamics, chaotic behavior can occur in the CNN structures [13]. In this paper, a chaotic attractor is observed with two non-autonomous cell CNN using opposite-sign template. This network is reported in [14] and solved with Runge-Kutta iterative solution method.

In recent years, many chaotic CNN applications are implemented on the FPGA. Main advantage of the usage of the FPGA is that it can be programmed with more flexibility during implementation of design. Furthermore,

development of any design on FPGA is easier and faster than other microprocessors [15]. Implementation of CNN structure on FPGA platform presents all advantages in the meaning of higher operating frequency, minimum resource utilization, reconfigurable systems and more secure applications.

Herewith this introduction, CNN structure and its mathematical model which exhibits chaotic behavior, are presented in Section 2. In Section 3, the implementation of discrete time chaotic CNN model on FPGA platform is also presented. At the end, final section concludes the paper.

2. The Chaotic CNN Structure

The two-cell non-autonomous CNN model given in [14] is described by state equations as follow:

$$\begin{aligned} \dot{x}_1 &= -x_1 + pf(x_1) - sf(x_2) + g(t) \\ \dot{x}_2 &= -x_2 + sf(x_1) + pf(x_2) \end{aligned} \tag{1}$$

$$f(x_i(t)) = \frac{1}{2}\left(|x_i(t)+1| - |x_i(t)-1|\right), \qquad i = 1,2 \tag{2}$$

$$g(t) = A\sin\left(\frac{2\pi t}{T}\right) \tag{3}$$

where $p > 1$, $s > 0$; x_1, x_2 are state variables of each cell and $f(\cdot)$ is an odd piecewise-linear function as defined in Eq. (2). For the CNN parameters $p = 2$, $s = 1.2$, initial condition $x(0) = (0.14, -0.1)$ and a sinusoidal input signal parameters $A = 4.04$ and $T = 4$, the non-autonomous CNN with two-cell given in [14] and chaotic portrait of the system are shown in Fig.1 and Fig.2, respectively.

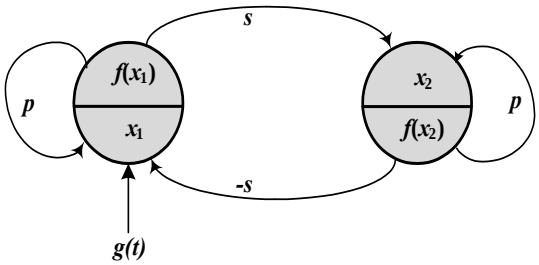

Figure 1. The non-autonomous CNN with two-cell.

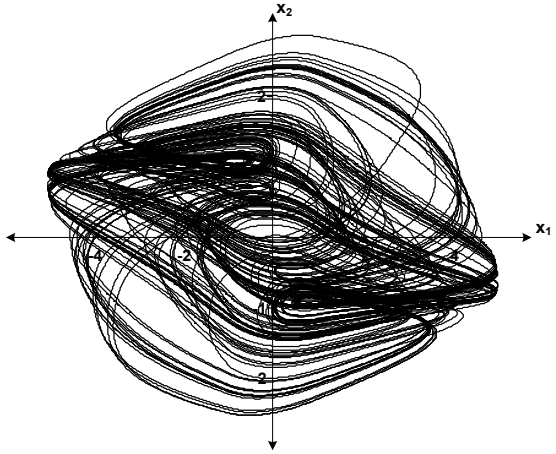

Figure 2. Phase plane portrait of system with initial condition of $x(0) = (0.14, -0.1)$.

In Fig 2, the chaotic behavior is observed. It is clearly appeared that the attractor has a roughly point symmetric property. If the Poincaré map of (x_1, x_2)-plane is obtained, it can be seen that the attractor possesses the horseshoe structure [14].

3. Implementation of Discrete Time Chaotic CNN Model

In order to implement the model on microprocessor, state equations must be discretized. Discrete time model of the CNN based chaotic system is determined as below.

$$\begin{aligned} x_1[n+1] &= \frac{1}{a}\left[\left(\frac{x_1[n]}{T}\right) + pf(x_1[n]) - sf(x_2[n]) + \mathrm{g[n]}\right] \\ x_2[n+1] &= \frac{1}{a}\left[\left(\frac{x_2[n]}{T}\right) + sf(x_1[n]) + pf(x_2[n])\right] \end{aligned} \tag{4}$$

$$f(x_i[n]) = \frac{1}{2}\left(|x_i[n]+1| - |x_i[n]-1|\right), \qquad i = 1,2 \tag{5}$$

$$g[n] = 4.04\sin(\pi n/2) \tag{6}$$

Discrete time response of the model can be acquired by using CNN parameters as $p = 2$, $s = 1.2$, initial conditions $x(0) = (0.14, -0.1)$, $T = 0.005$, $a = 1 + (1/T) = 201$.

In arithmetic operations, fractional numbers are used as well as integers. There are two types of binary number representation format which are floating-point and fixed-point format. Any of

them can be chosen to represent a fractional number as a fractional binary number. A fractional number representation format should be considered that it represents the fractional number as a decimal and then converts it to a fractional binary number by multiplying the decimal number by 0.5 repeatedly. A binary coded number can be defined by equation,

$$...b_2b_1b_0.b_{-1}b_{-2}... = \sum_i b_i x 2^i \quad (7)$$

where there is a binary radix point in the above equation before b_{-1} [16].

In this study, the fixed-point number format is chosen because of advantages on easy coding. In this format, a signed/unsigned number is stored and this number is scaled by a fixed factor which is determined by user with respect to format parameters. This method implies shifting operation the radix point to the left side [16].

Qm.n is the representation of fixed-point number format that has *m* bit for integer part and *n* bit for fractional part as shown in Fig. 3. As we use signed numbers in discrete time model of our network, *Qm.n* format has to be arranged as it contains signed two's complement fixed-point fractional binary number [16].

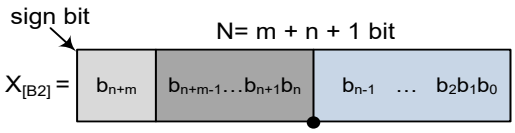

Figure 3. Structure of the *Qm.n* fixed-point signed number format.

Since initial values of state variables and parameters are chosen signed numbers as 16-bits in width, the format is arranged to *Q3.12* and 1 bit is reserved for the sign.

$$x_{[B10]} = \frac{1}{2^n}\left[-2^{N-1}b_{N-1} + \sum_{i=0}^{N-2} 2^i b_i\right] \quad (8)$$

Regarding to equation above (8) where $x_{[B10]}$ represents fractional number in decimal, N = 16, x_{min} = -2^m = -8, x_{max} = 2^m -2^{-n} = 7.99975 and resolution = 2^{-n} = 2.4414.10-3 are determined for *Q3.12* signed two's complement format.

In the implementation stage, three peripheral circuits are used in the CNN core design. These are Clock Generator, CNN Circuit and CNN Cache.

The Clock Generator generates the low frequency clock signal by using the 50 MHz clock signal on FPGA chip. Generated low frequency clock signal is used for iterative solution of model. In FPGA applications, instead of for loop operation, a counter module is arranged in order to calculate the next value of state variables repeatedly. Each increment of counter value is admitted as the beginning of the new loop.

The CNN circuit has two state variables, four parameters for state variables, sampling period register and external sinusoidal input register which are all in 16 bit width. The CNN circuit solves the equations (4), (5) and (6) iteratively by using the clock pulse of the Clock Generator circuit. The model requires previous values of the state variables in order to determine the current values. Therefore, the CNN Cache circuit is used.

The CNN Cache needs 20 KB of total memory in order to emulate the discrete time chaotic CNN model. Since FPGA chip has enough memory in BlockRAM, no external memory is needed. 2 BlockRAM modules are defined for this implementation each one of module has 10 KB capacity. Defined BlockRAM modules have a dual-port interface. Therefore, the values at different addresses can be accessible at the same time.

The CNN core is designed, implemented on Spartan 3e XC3S1600e FPGA development board and programmed by Xilinx ISE Design Suite 13.2. When core is programmed, initial values of state variables and parameters are transferred to the core by using built-in communication interfaces and functions in MATLAB. The whole system is illustrated in Fig.4. Table 1 illustrates resource utilization of the CNN core design.

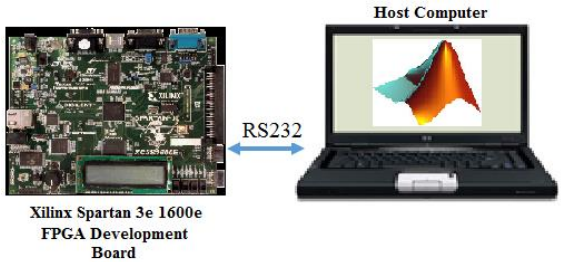

Figure 4. The scheme of the CNN Core Design.

Table 1. Resource utilizations of the design.

Resources	Arithmetic Unit	
	Clock Generator Circuit	CNN Core Design
Used slices	24	4013
Used FFs	33	5214
Used LUTs	47	6589

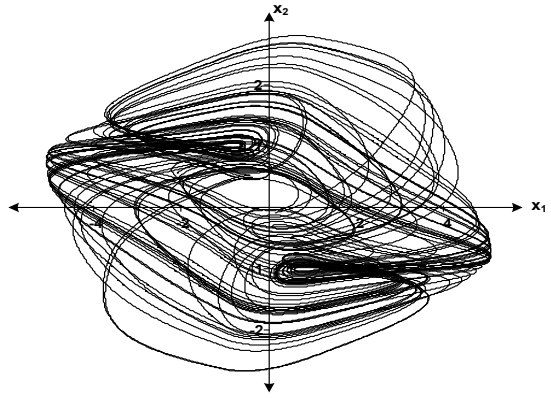

Figure 5. Discrete time response of the model with given parameters.

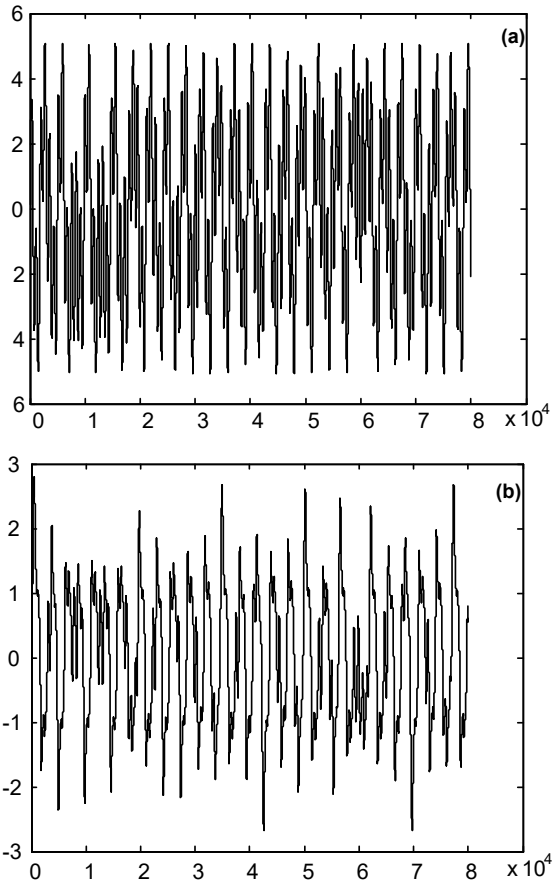

Figure 6. Discrete time response of a) x_1[n], b) x_2[n].

After the whole design is implemented, the programming bit stream of the design is generated and FPGA chip is programmed by Xilinx ISE Design Suite. The maximum operating frequency is obtained as 24.67 MHz from Xilinx ISE Design Suite. Fig. 5 shows the discrete time response of the model, which is very similar to the continuous time response given in Fig. 2. Also, Fig.6 illustrates the discrete time response of each cell. Fig. 5 and Fig. 6 are plotted in MATLAB and the data are obtained from implementation results on Xilinx ISE Design Suite for FPGA of the chaotic discrete time CNN model.

4.Conclusion

In this study, we have presented the implementation of two cell non-autonomous CNN model on FPGA. This study shows that the implemented CNN model have chaotic behaviors and model can be easily adapted to RNG applications using FPGA not only the entropy source of RNG but also the seed of pseudo random number generator application.

5. References

1. Strogatz S.H., Herbert D.F., (1996). Nonlinear dynamics and chaos. Medical Physics-New York-Institute of Physics, **23**(6), 993-995.
2. Natsheh A.N., Al-Habibah E.M.S., (2015). Chaos control DC-DC boost converter by FPGA. *2015 IEEE 42nd Photovoltaic Specialist Conference (PVSC)*, New Orleans, LA, 1-6.
3. Hidalgo R.M., Fernndez J.G., Rivera R.R., Larrondo H.A., (2001). Versatile dsp-based chaotic communication system. Electronic Letters, **37**, 1204–1205.
4. Ali-Pacha A., Said N.H., M'Hamed A., Belgoraf A., (2007). Lorenz's attractor applied to the stream cipher (alipacha generator). Chaos, Solitons and Fractals, **33**(5), 1762-1766.
5. Mazzini G., Setti G., Rovatti R., (1997). Chaotic complex spreading sequences for aynhchronous ds-cdma-part 1: System modeling and results. IEEE Trans. Circuits Sys. 1, **44**(10), 937–947.
6. Setti G., Balestra M., Rovatti R., (2000). Experimental verification of enhanced electromagnetic compatibility in chaotic fm clock signals. in Proceedings of ISCAS'00. IEEE Circuits and Systems Society, III-229–232.
7. Gonz´alez C.M., Larrondo H.A., Gayoso C.A., Arnone L.J., (2003). Generaci´on de secuencias binarias pseudo aleatorias por medio de un mapa ca´otico 3d. in Proceedings del IX Workshop de IBERCHIP.
8. De Micco L., Zabaleta O.G., Gonzlez C.M., Arizmendi C.M., Larrondo H.A., (2010). Estocasticidad de un atractor catico determinista implementado en fpga. Proceedings Iberchip.

9. De Micco L., Larrondo H.A., (2011). FPGA implementation of a chaotic oscillator using RK4 method. *2011 VII Southern Conference on Programmable Logic (SPL)*, Cordoba, 185-190.doi: 10.1109/SPL.2011.5782646.
10. Merah L., Ali-Pacha A., Said N.H., Mamat M., (2013). Design and FPGA implementation of Lorenz chaotic system for information security issues. Applied Mathematical Sciences, **7**(5), 237-246.
11. Chua L.O., Yang L., (1988). Cellular neural networks: Theory. IEEE Trans. Circuits Syst., **35**, 1257-1272.
12. Chua L.O., Yang L., (1988). Cellular neural networks: applications. IEEE Transactions on Circuits and Systems, **35**(10), 1273-1290.
13. Zou F., Nossek J.A., (1993). Bifurcation and chaos in cellular neural networks. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, **40**(3), 166-173.
14. Zou F., Nossek J.A., (1991). A chaotic attractor with cellular neural networks. in *IEEE Transactions on Circuits and Systems*, **38**(7), 811-812.
15. Gerosa A., Bernardini R., Pietri S., (2001). A fully integrated 8-bit, 20MHz, truly random numbers generator, based on a chaotic system. In: Proceedings of the Southwest Symposium on Mixed-Signal Design, SSMSD, 87–92.
16. Karakaya B., Yeniceri R., Yalcin M.E., (2015). Wave computer core using fixed-point arithmetic. 2015 IEEE International Symposium on Circuits and Systems (ISCAS), Lisbon, 1514-1517.