Applied Soft Computing 35 (2015) 237–246

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A new intelligent hardware implementation based on field programmable gate array for chaotic systems

Remzi Tuntas*

Yuzuncu Yil University, Ercis Technical School of Higher Education Electronic and Communication Technologies, Ercis 65080, Van, Turkey

ARTICLE INFO

Article history:
Received 4 February 2015
Received in revised form 2 May 2015
Accepted 2 June 2015
Available online 3 July 2015

Keywords:
Intelligent hardware implementation
The intelligent system model
MVPDOC
FPGA

ABSTRACT

In the present study, a new intelligent hardware implementation was developed for chaotic systems by using field programmable gate array (FPGA). The success and superior properties of this new intelligent hardware implementation was shown by applying the Modified Van der Pol–Duffing Oscillator Circuit (MVPDOC). The validation of intelligent system model was tested with both software and hardware. For this purpose, initially the intelligent system model of MVPDOC was obtained by using the wavelet decompositions and Artificial Neural Network (ANN). Then, the intelligent system model obtained has been written in Very High Speed Integrated Circuit Hardware Description Language (VHDL). In the next step, these configurations have been simulated and tested under ModelSim Xilinx software. And finally the best configuration has been implemented under the Xilinx Virtex-II Pro FPGA (XC2V1000). Furthermore, the High Personal Simulation Program with Integrated Circuit Emphasis (HSPICE) simulation of MVPDOC has been carried out under ModelSim Xilinx software for comparison with proposed intelligent system. The results obtained show that the proposed intelligent system simulation has much higher speed in comparison with HSPICE simulation.

1. Introduction

The FPGA-based approach uses the advantages of both hardware and software models. FPGA have many consistent advantages such as reliable, flexible, fast response rapid prototyping, adaptation, reduced cost, simplicity of design, and programmable architecture [1–3]. Recently, FPGA are becoming more and more popular and are used in many applications such as system modeling, signal processing, fault-tolerant, and computing feature extraction [4,5]. FPGA has been successfully utilized in motor control and robotic systems [6,7]. Wen et al. [8] introduced analysis of two-phase stepper motor driver based on FPGA. Fobel et al. [9] investigated the feasibility of using hardware acceleration, in the form of FPGAs, to improve the performance of placement algorithms. An integrated hardware and software design method is developed to implement an MPC algorithm on FPGA chip by Yang et al. [10]. FPGA model can be easily reprogrammed and is completely customizable. Users can use hardware description language (HDL) such as VHDL or Verilog, to modify the FPGA model code. VHDL is one of the most widely used techniques for programming hardware [11], and it is the programming language of choice for defining mixed analogy/digital systems. The usage of VHDL provides a number of advantages, such as speedy, high capacity, and redesigns [12,13].

Neural networks (NN) have become very important tool for modeling the non-linear input–output characteristics for the devices [14]. The parallel structure of the NN is very important feature for real-time implementations. FPGA-based application is proper for parallel realizations and, is a good solution for the complex problem. In literature, there are various ANN applications on FPGA [15,16]. Cavuslu et al. [17] introduced hardware implementation of ANNs with learning ability on FPGA for dynamic system identification. An approach to implement a general regression neural network (GRNN) based on FPGA is presented by Polat and Yildirim [18]. Wavelet transform is used for the modeling and simulation of Chua's circuit [19]. Switched circuits are modeled based on wavelet decomposition and NN [20]. In [21], Lin and Tsai presented implementation of a wavelet neural network (WNN) with learning ability on FPGA. Fotsin and Woafo [22] designed a controller, which enables both the synchronization of two unidirectional coupled MVPDOC and the estimation of unknown parameters of the drive oscillator by using adaptive control. In another study, chaos synchronization between two chaotic systems consisting of MVPDOC and Chua oscillators has been provided by Fotsin et al. [23].

This paper is aimed to develop a new intelligent hardware implementation for chaotic systems. Wavelet decompositions and ANN approach has been used in the present study to develop an

* Tel.: +90 432 388 2164 5011; fax: +90 432 388 2165.
E-mail address: rtuntas@hotmail.com

http://dx.doi.org/10.1016/j.asoc.2015.06.039
1568-4946/

intelligent system model. Wavelet decomposition was used for extracting feature and multi layer perceptron (MLP) was used as a modeling method. The intelligent system model for MVPDOC was proposed and applied into FPGA Xilinx Virtex-II Pro FPGA (XC2V1000). The effectiveness of proposed intelligent system model is tested on five different data. For this aim, performance analysis of the HSPICE simulation with proposed intelligent system method is performed for simulation speeds by using ModelSim Xilinx software. The simulation results obtained show speed and superiority of the proposed intelligent system in hardware implementation for chaotic systems.

The rest of the paper is organized as follows. Section 2 provides a brief overview on the description of the modified Van der Pol–Duffing oscillator system. Section 3 presents FPGA-based implementation of ANN. Section 4 presents proposed methodology. Section 5 presents results and discussion. Finally, Section 6 also reports the conclusion.

2. Description of the modified Van der Pol–Duffing oscillator system

King and Gaito developed the MVPDOC, which is a model of an autonomous chaotic system introduced in 1992 [24]. The electrical model of the MVPDOC can be represented by the electrical circuit of Fig. 1. The difference in their characteristics of dynamical behavior lies only on the physical realization of the non-linear resistance (N), and on the selection of parameters. The physical realization of the MVPDOC and physical realization of the non-linear resistance that belongs to the MVPDOC are shown in Fig. 2 [23,24]. The differential equations are obtained as follows by using Kirchhoff's laws for the equivalent circuit in Fig. 2

$$\dot{x} = -m\left(x^3 - \propto x - y + \mu\right) \quad (1)$$

$$\dot{y} = x - y - z \quad (2)$$

$$\dot{z} = \beta y - \gamma z \quad (3)$$

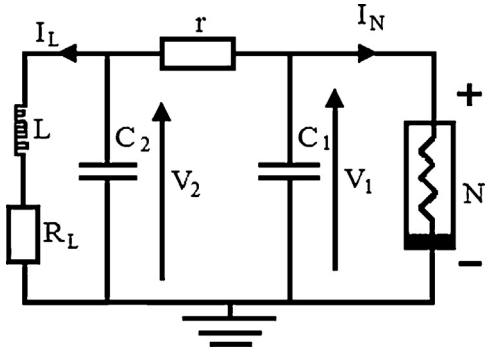

Fig. 1. The electrical model of the MVPDOC.

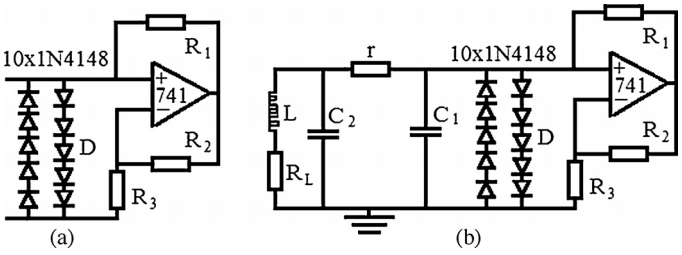

Fig. 2. The physical realization. (a) The non-linear resistance and (b) the modified Van der Pol–Duffing oscillator.

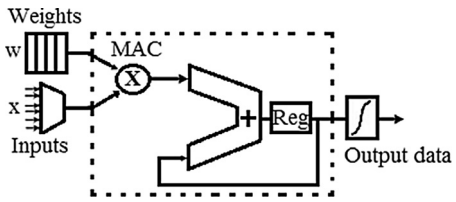

Fig. 3. Structure of the multiplier–accumulator.

3. FPGA-based implementation of ANN

The artificial neuron is one of the most important fundamental components of the NN. The neuron computation methods consist of calculating the pondered sum function and updating the state of the neuron with applying the activation function. Neuron computation methods are carried out in three stages. These stages consist of artificial neuron implementation, data representation and sigmoid activation function approximation. The various steps for the practice of the ANN on FPGA are presented in the following paragraphs.

The processing of the multiplication and addition of parallel inputs and weights is done in the first stage. And the second stage is the non-linear sigmoid function for the output signal. For the first stage, several types of adders can be used to achieve the weighted sum of states: combinatorial, in series, dynamic, carry look ahead, Manchester, Wallace tree, etc. [25]. Likewise, several ways can be used to realize the multiplication, the mostly used are as following: in serial, serial/parallel, and completely parallel. The multiplier–accumulator (MAC) is carried out by a multiplier related to an adder looped on it in order to acquire an accumulation. Structure of MAC is shown in Fig. 3. In case of using a completely parallel NN, we need 20(7 _ 12) MAC's in the hidden layer and 5(12 _ 1) in the output layer in our study.

A signed real numbers are needed for the computation of NN. The fixed-point representation and the floating-point representation are used to define the position of the decimal point. The floating point is more dynamic than the fixed-point. It allows the encoding of a greater number of real values for the same number of bits [26]. The fixed point representation is preferred in various applications. Fixed-point arithmetic is used for encoding the parameters and performing the computations [27]. In this study, a fixed-point representation of 18 bits (9 bits for integer part, and 9 bits for real part) was choosen. This facilitates the MAC (multiplier–accumulator) realization, which is the main element of the NN [28].

The direct application of the sigmoid activation function on an FPGA is very difficult. Because, the non-linear sigmoid function has an infinite exponential series. This non-linear sigmoid function can be applied directly using digital techniques. Digital simulation of the activation function requires a large computation time. Hence, usually computationally simplified version of sigmoid function is used. To approximate the sigmoid function with use of FPGA design, there are three practical approaches consisting of direct approximation, lookup table approximation, and piece-wise linear approximation. The direct approximation approach was chosen in this study. Because it consumes less memory space and has better precision. To explain this situation, a comparison was made between the desired function with Matlab and estimated function with VHDL for three approaches. The obtained simulation results are shown in Fig. 4 by using the ModelSim Software and Matlab.

The above figure indicates that the direct approximation function provides a reasonable approximation for the hardware realization of the sigmoid function. As seen in the figure, the obtained direct approach curve with VHDL shows more fit than the other approaches. Furthermore, in Table 1, synthesis results