Computers and Electrical Engineering 000 (2015) 1–16

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

Digital hardware implementation of a radial basis function neural network☆

Nguyen Phan Thanh [a,b], Ying-Shieh Kung [a,*], Seng-Chi Chen [a], Hsin-Hung Chou [c]

[a] Department of Electrical Engineering, Southern Taiwan University of Science and Technology, Tainan, Taiwan
[b] Faculty of Electronics and Electrical Engineering, Ho Chi Minh City University of Technology and Education, Ho Chi Minh City, Vietnam
[c] Industrial Technology Research Institute, Hsinchu, Taiwan

ARTICLE INFO

Article history:
Received 8 July 2015
Revised 17 November 2015
Accepted 17 November 2015
Available online xxx

Keyword:
Radial basis function neural network (RBF NN)
Stochastic gradient descent (SGD)
Very high-speed IC hardware description language (VHDL)
Simulink and ModelSim co-simulation
Electronic design automation (EDA)
Permanent magnet synchronous motor (PMSM) drive

ABSTRACT

This work studies a digital hardware implementation of a radial basis function neural network (RBF NN) Firstly, the architecture of the RBF NN, which consists of an input layer, a hidden layer of nonlinear processing neurons with Gaussian function, an output layer and a learning mechanism, is presented. The supervising learning mechanism based on the stochastic gradient descent (SGD) method is applied to update the parameters of RBF NN. Secondly, a very high-speed IC hardware description language (VHDL) is adopted to describe the behavior of the RBF NN. The finite state machine (FSM) is applied for reducing the hardware resource usage. Thirdly, based on the electronic design automation (EDA) simulator link, a co-simulation work by Simulink and ModelSim is applied to verify the VHDL code of RBF NN. Finally, some simulation cases are tested to validate the effectiveness of the proposed digital hardware implementation of the RBF NN.

1. Introduction

Differing from other conventional computing, neural networks have the capability of capturing unnecessarily deterministic nonlinear and take the advantages of the fast, highly accurate computation for the non-parametric model. Radial basis function (RBF) becomes a very powerful tool in neural network architecture due to its faster learning and higher nonlinear activated function used in hidden layer, compared with back propagation topology [1]. In particular, RBF neural networks process the information responds to inputs according to a defined learning rule and then the trained network can be used to perform certain tasks depending on the application. Many certain problems such as human face recognition [2], functional approximation, nonlinear system identification, motor control [3] etc. are considered to solve in the practical applications of neural networks. Due to their vigorous approximation properties, RBF neural networks can present a good mapping on the complex models [4].

The RBF NN is specified by a three-layer with a simple topology, where the hidden unit implements a radial basis activated function. These networks have a good ability to identify or approximate the function of unconventional systems with the given set of inputs and their corresponding outputs. To fit the network outputs to the given inputs, the networks need to be trained to optimize their parameters. If the parameters of the RBF are fixed, in other words, the position of neurons and the "bell curve"

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor T-H Meen.

* Corresponding author. Tel.: +886 6 2533131.
E-mail address: kung@mail.stust.edu.tw (Y.-S. Kung).

http://dx.doi.org/10.1016/j.compeleceng.2015.11.017
0045-7906/

of exponential function could not be changed, the RBF effectiveness reduces gradually to the linear problem solving. To cover the highly nonlinear problems, the sufficient learning algorithm that allows modifying the parameters of RBF neural network [5] is introduced, called a stochastic gradient descent (SGD) method. The SGD [6] method is a popular algorithm for training a wide range of models by following the negative gradient of the objective after using only a single or a few training examples and computes the next updated parameters at each iteration. It tends to converge very well to global optima when the objective function is convex or pseudo convex, otherwise converge to local optima [7]. Therefore, its adaptability due to the ability of adjustable network parameters may lead to improving the accuracy and the stability of approximations.

As the preliminary survey on several papers, some studies give the various methods for the implementation of artificial neural networks (ANN) in a digital system. One of the basic problems in the research [8–10] is how to implement the ANN in fixed-point arithmetic as well as floating point behavior due to the nonlinear characteristic of exponential activation functions and perform the good responses undergoing the associated training algorithm for the particular task. Although the floating point number formats are more precise in computation [11–13] but they spend much hardware resources in the FPGA (Field programmable gate array) than the fixed-point number formats. To solve this problem, a 32-bit Q24 fixed-point number is firstly adopted to reduce the hardware resources and still assure high numerical accuracy. The next, this work proposes an efficient algorithm by combining the Taylor series expansion and a look-up table (LUT) to calculate the exponential activation function in RBF NN. For a 32-bit Q24 numerical type, the 24-bit length in the fraction part (Q24) can make the RNF NN or nonlinear Gaussian function have a good numerical precision. Although the range of the integer part (8-bit) is only between +127 to –128, it still can cover the data operation to avoid the numerical overflow condition occurred if the normalized/de-normalized operation in input/output data is applied.

In realization, to speed up the computing performance of RBF NN, FPGA [14–16] gives the best solution for this application due to its characteristics of the programmable hard-wired feature, fast computation power, higher density, etc. However, to describe the behavior of computing the RBF NN in FPGA implementation, VHDL, and Open computing language (OpenCL) [17–19] provide two possible programming tools. VHDL uses in electronic design automation (EDA) to describe digital and mixed-signal systems such as FPGA and integrated circuits [20]. VHDL can also be used as a general purpose parallel programming language. OpenCL is a framework for writing programs that execute across heterogeneous platforms consisting of central processing units (CPUs), graphics processing units (GPUs), digital signal processors (DSPs), FPGAs and other processors [20]. OpenCL provides parallel computing using task-based and data-based parallelism. In this study, VHDL is adopted to describe the behavior of computing the RBF NN, and finite state machine (FSM) is applied. Due to the FSM belongs to the sequential processing method; the FPGA resources usage can be greatly reduced.

Recently, a co-simulation work by electronic design automation (EDA) Simulator Link has been applied to verify the effectiveness of the VHDL code in a digital system [21]. The EDA Simulator Link provides a co-simulation interface between Simulink and ModelSim [22]. This work intends to verify the performance of RBF NN function in a co-simulation work constructed by Simulink and ModelSim based on EDA simulator link. This sufficient correlative tools strongly provide the obvious results for comparing with the floating point results in Matlab. Therefore, in this paper, a co-simulation by EDA Simulator Link is applied to design and verify the proposed computation algorithm for RBF NN. The input stimuli and output results are performed in Simulink and the algorithm to compute the RBF NN is executed in ModelSim.

The remainder of this paper is organized as follows: Section 2 introduces the architecture of RNF NN with SGD-based learning mechanism. Section 3 describes the digital hardware implementation of RBF NN. Section 4 illustrates the simulation results to evaluate the performance of the proposed method. Section 5 gives a summary of the contribution in this work.

2. Radial basis function neural network (RBF NN)

This section aims to give a brief introduction of the structure and specifically orient to the mathematical analysis of RBF NN [23]. However, the RBF NN adopted here is a three-layer architecture that is shown in Fig. 1. It comprises of one input layer, one hidden layer, one output layer and one supervising learning machine using stochastic gradient descent method.

In neural architecture, the input layer is mapped on hidden layer via the nonlinear activated function or radial basis functions (known as neurons), whereas the connection from the hidden layer to output layer performs a linear transformation. The input layer has n_1 inputs by $x_1, x_2, \ldots, x_{n_1}$ and its vector form is represented by

$$X = [x_1\, x_2 \ldots x_{n_1}]^T \tag{1}$$

In hidden layer, the multivariate Gaussian function is chosen as the activated function, and its formulation is shown as follows.

$$\varphi_i = \exp\left(-\frac{\|X - C_i\|^2}{2\sigma_i^2}\right),\ i = 1, 2, \ldots, n_2 \tag{2}$$

where n_2 is the number of neuron in hidden layer, $C_i = [c_{i1}\, c_{i2} \ldots c_{in_1}]^T$,$\sigma_i$ denote the node center of radial basis function and node variance (or width) of ith neuron, and $\|X - C_i\|$ is the norm value (Euclidean distance) which is measured by the inputs and the node center at each neuron. Finally, the network output in Fig. 1 can be written as

$$y_j = \sum_{i=1}^{n_2} w_{ji}\varphi_i,\ j = 1, 2, \ldots, n_3 \tag{3}$$

Please cite this article as: N.P. Thanh et al., Digital hardware implementation of a radial basis function neural network, Computers and Electrical Engineering (2015), http://dx.doi.org/10.1016/j.compeleceng.2015.11.017