International Journal of MC Square Scientific Research Vol.9, No.1 April 2017

REDUCTION OF RESOURCE USAGE IN FPGA BY IMPLEMENTING BACK PROPAGATION ALGORITHM

V.Vijayaragavan,

PG scholar-EEE, Anna University Regional Campus, Coimbatore

Dr.S.N.Deepa M.E.,Ph.D.,

HOD-EEE, Anna University Regional Campus, Coimbatore

Dr.M.Yuvaraju M.E.,Ph.D.,

Asst. Prof-EEE, Anna University Regional Campus, Coimbatore

Abstract - Back propagation learning algorithm is used to implementing the FPGA for reducing the memory usage and it is one type of Supervised Learning network. The overfitting problems occurs in the device is obtained by the three steps of the algorithm. The three steps are training, validation and testing. The FPGA implementation is done by the Ex-OR functions and the VHDL code is written for the Ex-OR function using the BP algorithm. Also, the MATLAB code is written for the BP algorithm. The results show that the reducing of resource usage and increase of computational speed by the comparison of the output of VHDL code, Arduino code and MATLAB code. Both of the implementations of the Back-propagation algorithm is useful for applications to real-world problems.

Keywords—Field Programmable Gate Array (FPGA), Microcontrollers. Supervised Learning, VHSIC Hardware Description Language (VHDL), MATLAB.

1. Introduction

The BP learning algorithm is one of the most important developments in supervised learning in neural networks developed by Brysor and Ho in the year 1969, Werbos in the year 1974, Lecun in the year 1985, parker in the year1985 and Rumelhan in the year 1986. This learning algorithm is used in multilayer feed forward neural networks. Gradient decent learning rule is used to update the weights. For the given set of input pairs the algorithm calculates the net input and error values, based on the error values the weight values will be updated in the network [1]. The numbers of hidden layers used in the network, the training become more complex. The error is nothing but the difference between the actual calculated output and the desired output.

Field Programmable Gate Arrays are the one type of hardware logic device that have the ability to program like a general-purpose computing platform [12]. The FPGA are generally used in the prototype Application Specific Integrated Circuit (ASIC) [4]. FPGA uses the reconfigurable computing logic. FPGAs are quite suitable for neural network applications because they are parallel devices as is the processing information neural network models [7]. These are used as a prototyping tool for the hardware designers. FPGA boards are programmed using the hardware description languages, such as VHDL (VHSIC Hardware Description Language) or Verilog [5]. The programming of FPGA boards is very time consuming process.

In this paper the Back-Propagation algorithm is implemented in VIRTEX-5 XC5VLX110T FPGA, Arduino UNO microcontroller and Standard PC implementation using MATLAB. The aim of this project is 1) to obtain the implementation of both type of FPGA device in real time applications. 2) to compare the efficiency between these devices and to a standard PC based implementation. The organization of this paper follows. Section II includes details about the back-propagation algorithm. The FPGA implementation is described in Section III, which contains four parts: the first three subsections describe each one of the three blocks used for the algorithm implementation, while the fourth subsection deals with specific implementation details. Section IV presents the results of both implementations on a set of FPGA, Microcontroller and MATLAB code for obtaining the resource usage and computational speed. Finally, the discussion and conclusions are drawn in Section V.

2. Back Propagation Algorithm

The Back-propagation algorithm is the mainly used supervised learning network in the real-time applications because it produces the efficient output than the other networks used. The BP algorithm is also different from the other networks because the weight values of the network are updated during the training of the network. The other networks have the difficulty to find the weight values of the hidden neurons. The number of hidden layers used in the network will increase the complexity of the problem and produce the efficient output. The weight values are updated based on the error value calculated in the both the hidden layer and output layer. The error is nothing but the difference between the actual calculated output and the desired target output. The error is measured at the output layer. The information of the error is not directly calculated in the hidden layers. Therefore, other techniques should be used to calculate an error at the hidden layer, which will make the output error will be very low and this is the ideal goal.

The three stages of process involved in the training of an BP algorithm. They are 1) feed forward of the input training pattern, 2) the calculation and back-propagation of the error, and 3) updating of weights. The testing of the BP algorithm involves the computation of feed forward phase only.

2.1 Architecture:

A back propagation neural network is multi-layer, feed forward neural network consisting of an input layer, hidden layer and an output layer. The neurons present in the hidden and output layer have biases, which are the connections from the units whose activation is always 1. The bias terms also act as weights. The architecture of a BPN for only the direction of information flow for feed forward phase. During the back-propagation phase of learning, signals are sent in the reverse direction.

There are two types of inputs are used in the BPN. They are Binary (1, 0) and Bipolar (-1, +1). The activation function could be any function which increases monotonically and is also differentiable.

2.2 Algorithm:

The terminologies used in the Back-Propagation training algorithm are as follows:

X = input training vector $(X_1,\ldots X_i,\ldots..X_n)$

t = target output vector $(t_1,\ldots t_k,\ldots..t_m)$

α = learning rate parameter

X_i = input unit i

V_{oj} = bias on the j^{th} hidden unit

W_{ok} = bias on the k^{th} output unit

Z_j = hidden unit j. The net input to z_j is

$$Zinj = Voj + + \sum_{i=1}^{n} Xi\ Vij$$

And the output is

$$Zj = f\ (Zinj)$$

Yk = output unit k. The net input to Yk is

$$Yink = Wok + + \sum_{j=1}^{p} ZjWjk$$

And the output is

$$Yk = f\ (Yink)$$

3. FPGA Implementation Using BP Algorithm

FPGAs are reprogrammable logic devices which gives the flexibility in the design like software but the performance speeds closer to Application Specific Integrated Circuit. FPGAs has the ability to reconfigured a million of times after it has been already manufactured. These are the prototyping tool for the hardware designers. FPGAs are programmed using a hardware description language (VHDL). For this implementation VIRTEX-5 OpenSPARC Evaluation platform is used that includes a Xilinx VIRTEX-5 XC5VLX110T FPGA. Table 1 shows the main specifications of the VIRTEX-5 XC5VLX110T FPGA that indicates the main logic resources.

The Table gives the information about the number of slice Registers, Slice LUT's (Look Up Tables), Bonded IOB's (Input Output Block), BR (Block RAM) and DSP 48. The FPGA Implementation of BP algorithm was carried out using three blocks and it is shown in the Figure 2. The three blocks 1) Control Block, 2) Pattern Block, and 3) Architecture Block.

TABLE 1

MAIN SPECIFICATIONS OF THE VIRTEX-5 XC5VLX110T FPGA RELATED TO ITS AVAILABLE SLICE LOGIC

Device	Slice Registers	Slice LUT's	Bonded IOB's	BR	DSP 48
VIRTEX-5 XC5VLX110T	69120	69120	34	148	64

3.1 PATTERN BLOCK: The Pattern Block is used for the data exchange between the Personal Computer(PC) and FPGA through Serial Communication. The Serial Communication device RS-232 is used.

3.2 CONTROL BLOCK: The Control Block controls the whole information flow process within the FPGA board by sending and processing the information from the architecture and pattern blocks.

3.3 VALIDATION PROCESS: The validation process is used to prevent the overfitting problems, and it is executed after finishing a training epoch.

3.4 ARCHITECTURE BLOCK: The architecture block gives the physical implementation of the neural network architecture. The number of layers and the number of neurons used in BP algorithm in each layer has to be predefined by the user before the execution of the algorithm.

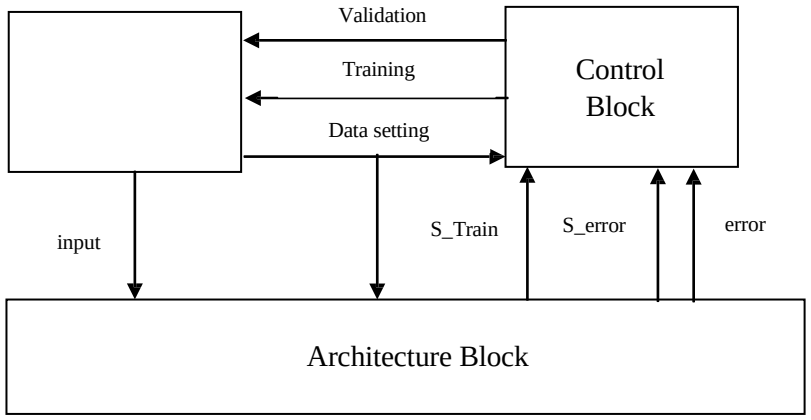

Figure 1. Block diagram for FPGA implementation by BP algorithm

In this paper, we have used the three layer of neural network architecture. They are input layer, output layer and one hidden layer. The input layer contains the two input neurons, hidden layer contains the two hidden neurons and the output layer contains the one output neuron. The

Ex-OR function is used to implement the FPGA by using BP algorithm. The architecture of Ex-OR gate function using BP algorithm is shown in Figure 2.

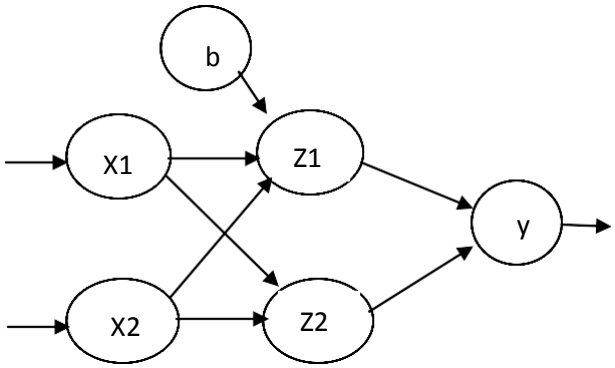

Figure 2. Neural Network Architecture of Ex-OR function

3.5 Implementation Of EX-OR Function in FPGA

The Ex-OR logic function has two inputs and one output. It produces an output only if either one or the other of the inputs is ON, but not if both of the inputs are OFF or both of the inputs are ON. The truth table of Ex-OR function is shown in Table 2.

3.6 IMPLEMENTATION APPROACH: The implementation of the BP algorithm in FPGA device using Ex-OR function has the following procedure.

Step 1: The network is initialized and the random values are given to weights. Step 2: The net inputs for the input neurons and hidden neurons are calculated. Step 3: The obtained value is run through a hard limiter function. Step 4: The same procedure is followed for the output layer using the outputs of the hidden layer as the input. Step 5: The error values (delta) for the output layer and hidden layer is calculated. Step 6: the weight values are updated using the error(delta) values. Step 7: The learning parameter α is taken as 0.5.

TABLE 2

TRUTH TABLE OF EX-OR FUNCTION

INPUT 1	INPUT 2	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

4. RESULTS

In this section, we analyze some features in connection to the implementations of the BP algorithm used in an FPGA board, examining also a third implementation of the algorithm in PC for comparison purpose. The MATLB code is executed for the PC based implementation of the BP algorithm and the code is run in an Intel® core™ i5-3330 CPU at 3GHz with 16GB of RAM memory. All the three implementations of the algorithm follow the same operation steps and the only evident differences between them are the random number generator used for the initialization of the synaptic weights, the type of data representation used in each case, and the computation of the sigmoid function. The FPGA implementation uses an LFSR random number generation routine, while the microcontroller and MATLAB code use the built-in random and randn functions respectively.

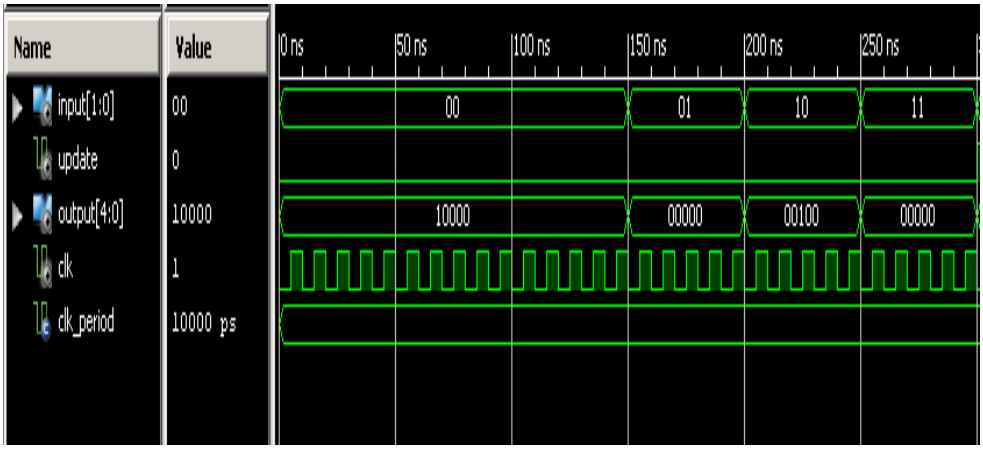

Figure 3. Xilinx FPGA output for Ex-OR function

Figure 3 shows the Xilinx FPGA output result for Ex-OR function. The function of Ex-OR is when the two inputs are same, the output will be low. When the two inputs are different the output will be high. The scale for the following output graph is about 50ns.

The MATLAB output shows the result for the MATLAB coding written for the back-propagation algorithm. The program shows the training of back propagation algorithm for the given input pattern. Figure 4 shows the input values given to the Back-propagation algorithm using MATLAB programming.

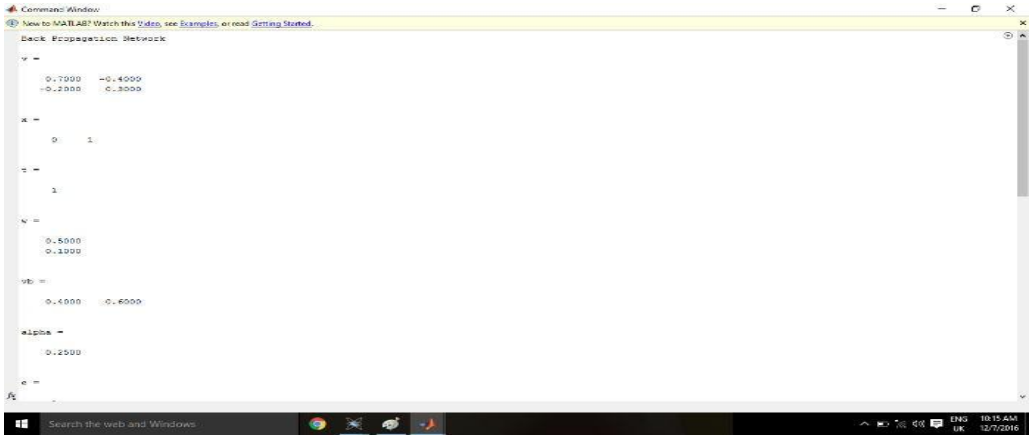

Figure 4. MATLAB input command window

The input values for the Back-propagation algorithm using MATLAB coding defines the given input pattern(x), target value(t), weight values between input and hidden layers(v), weight values between hidden and output layers(w), bias weight values(vb) and learning rate(α).

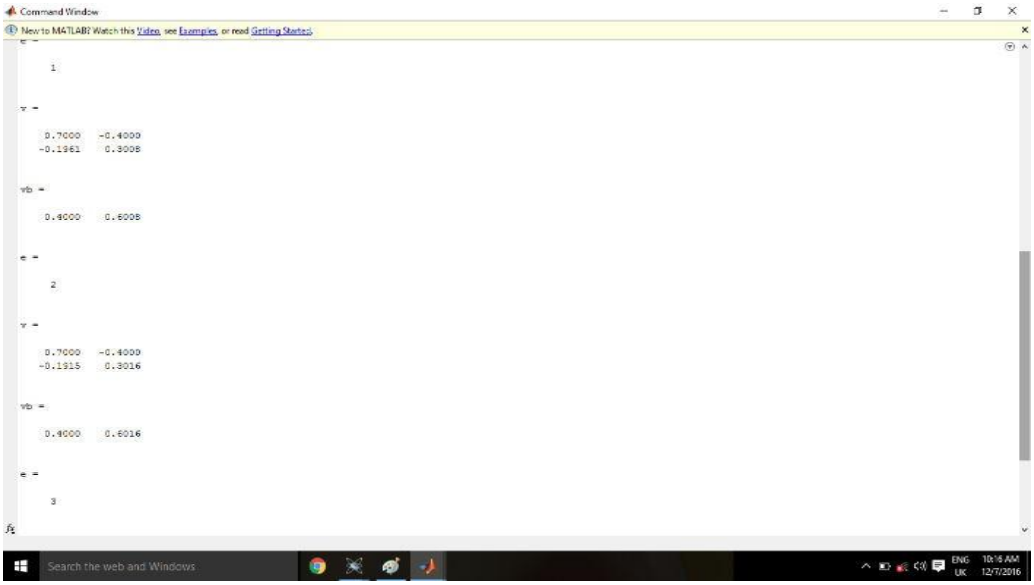

Figure 5. Output command window for 1st Iteration

Figure 6 shows the second iteration of training process, which includes the change in weight values.

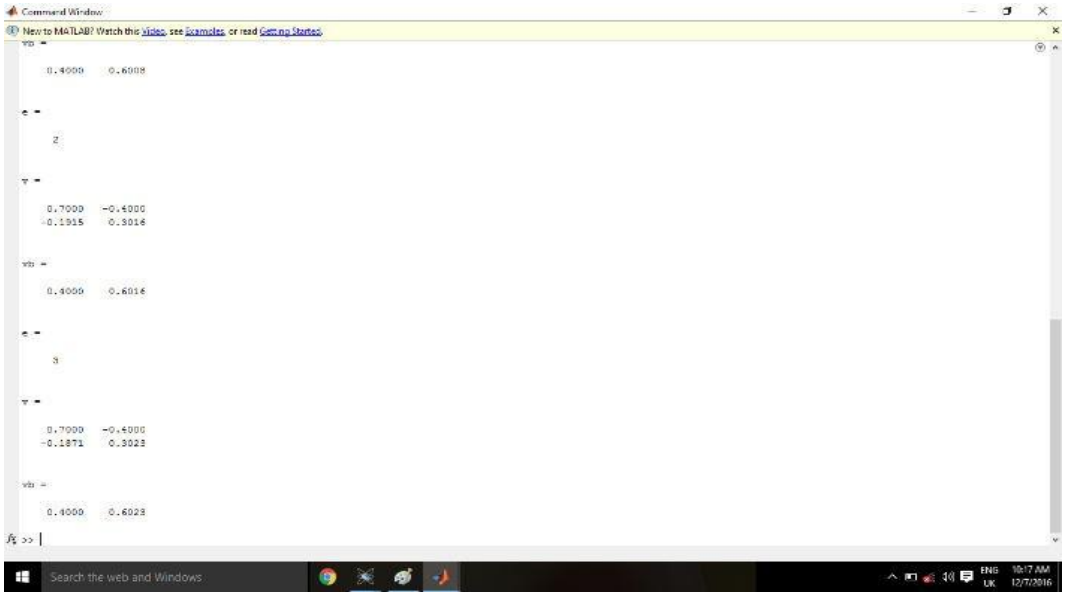

Figure 6. Output command window for 2nd Iteration

5. Discussion and Conclusion

The BP algorithm has been successfully implemented in an FPGA board and Arduino microcontroller and also a third one is standard PC-based implementation under MATLAB code. This learning method is used to prevent the overfitting problems. The implementation of the BP algorithm in FPGA board involves several challenges because the FPGA uses the hardware programming languages and the other two uses the software programming languages. The comparison of these three is the totally different approach. By using the BP algorithm implemented in FPGA we can reduce the memory space up to 25.8% in the total number of memory. For this implementation VIRTEX-5 OpenSPARC Evaluation platform is used that includes a Xilinx VIRTEX-5 XC5VLX110T FPGA. As an overall conclusion, this paper shows the potential advantages of using FPGA boards as hardware accelerator devices for neurocomputing applications giving their intrinsic parallel capabilities, while in relationship to the use of neural networks in microcontrollers, we highlight the on-chip characteristic of the presented implementation that will permit its use in remote sensors using a standalone operation mode.

6. References

[1] An.G, "The effects of adding noise during backpropagation training on a generalization performance," Neural Comput., vol. 8, no. 3, pp. 643-674, Apr. 1996

[2] Canete.E, Chen.J, Luque.R.M, and Rubio.B, "neuralSens: A neural network based frame work to allow dynamic adaption in wireless sensor and actor networks," J. Netw. Comput. Appl., vol. 35, no. 1, pp. 382-393, 2012.

[3] Dinu.A, Cirstea.M.N, and Cirstea.S.E, "Direction neural-network hardware-implemantation algorithm," IEEE Trans. Ind. Electron., vol. 57, no. 5, pp. 1845-1848, may 2010.

[4] Gomperts.A, Ukil.A, and Zurfluh.F, "Development and implementation of parameterized FPGA-based general purpose neural networks for online applications," IEEE Trans. Ind. Informat., vol. 7, no. 1, pp. 78-89, Feb. 2011.

[5] Gomperts.A, Ukil.A, and Zurfluh.F, "Implementation of neural network on parameterized FPGA," in Proc. AAAI Spring Symp., Embedded Reason., 2010, pp. 45-51.

[6] Hawkins.D.M, "The problem of overfitting," J. Chem. Inf. Comput. Sci., vol. 44, no. 1, pp. 1-12, 2004.

[7] Huang.G.B, and Sie.C.K, "Real-time learning capability of neural networks," IEEE Trans. Neural Netw., vol. 17, no. 4, pp. 863-878, Jul. 2006.

[8] Monmasson.E, Idkhajine.L, Cirstea.M, Bahri.I, Tisan.A, and Naouar.M.W, "FPGAs in industrial control applications," IEEE Trans. Ind. Informat., vol. 7, no. 2, pp. 224-243, May 2011.

[9] Ortega-Zamorano.F, Jerez.J.M, Subirats.J.L, Molina.I, and Franco.L, "Smart sensor/actuator node reprogramming in changing environments using a neural network model," Eng. Appl. Artif. Intell., vol. 30, pp. 179-188, Apr. 2014.

[10] Ortega-Zamorano.F, Jerez.J.M, Urda.D, Luque-Baena.R.M and Franco.L, "Efficient Implementations of the algorithm in FPGA and microcontroller," IEEE Trans. Nerual Netw, vol. 27, no. 9, pp. 1840-1850, Sep. 2016.