Performance analysis of Backpropagation Algorithm of Artificial Neural Networks in Verilog

Amrutha J
Department of Electronics & Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri, India
amrutha.ammu.j@gmail.com

Remya Ajai A S
Department of Electronics & Communication Engineering
Amrita Vishwa Vidyapeetham, Amritapuri, India
remya.amrita@gmail.com

***Abstract*— Artificial neural networks learn or get trained to execute definite tasks, instead of programmed computational systems through training algorithms such as Backpropagation algorithm. It is the basic tool for pattern classification application of ANN in field of medical diagnosis and remote sensing. This method involves changing the weights in the network using a training set of input output examples. Digital implementation of these neural networks for classification is suitable as it preserves the parallel architecture of the neurons and can be reconfigured by the user with FPGA. This parallelism in neural networks make it potentially fast for computation of tasks. This work implements backpropagation training algorithm in verilog with Modelsim-Altera 6.5b for a feedforward neural network. Since multiplier algorithms determine the operational speed and power consumption; a performance analysis is made based on different multipliers. The work can be further extended to the implementation of artificial neural networks on FPGA and to implement a classification application in the trained network.**

Keywords—artificial neural networks, backpropagation algorithm, activation function, sigmoid function

I. INTRODUCTION

It is believed that the human brain is the most complex object known in the universe. The fundamental component of brain is the neuron. There are about 100 billion neurons, connected by about 100 trillion linkages. Inspired by principles of computations performed by the biological neural networks of the brain, computational systems called, artificial neural networks are developed. It is adaptive and learns from experience and examples. From the training data, it can react to patterns. Artificial neural networks have applications in many disciplines because of the ability to map, model and classify non linear processes. It includes image processing, system identification and control, pattern recognition etc. For real time applications, the neural computations are to be made low cost and high speed.

Learning can be defined as the ability to perform better at a given task with experience [1]. The learning process of the brain involves altering biological neural structure, changing the strength of connections depending on activity. One of the most attractive features of artificial neural networks is its ability to learn .This learning process can be modelled by modifying the weights of the connections between nodes in the network. Learning algorithms are extremely useful when it comes to complex perceptual problems which are difficult to be written by a programmer. For example, facial recognition is such a problem hard for a human to accurately convert into code. The process of modifying the weights in the connections between network layers in order to achieve the expected output is called training the network. What takes place during training is learning. There are different learning algorithms such as supervised learning, unsupervised learning and reinforcement learning. In supervised learning, network uses a set of input-output examples to get trained .Once the network is trained, it should be able to correctly map a new and unseen input to the output. Backpropagation algorithm is a type of supervised learning. It can be used for training of multilayered neural networks. This work aims to incorporate the advantages of FPGA over artificial neural networks [2]. Here, the backpropagation algorithm is implemented in verilog which can be further implemented in FPGA to implement a classification application in the trained network [3].For example, an image classifier[4] application can be developed using feature extraction and reduction by discrete wavelet transform[5] and further training using backpropagation.

The paper is organized as given: Section II explains the basic Artificial Neural Networks. Section III explains how backpropagation algorithm is implemented in neural networks and its digital implementation. Section IV shows the results obtained for the backpropagation in verilog and performance analysis based on different multipliers. Finally Section V concludes the paper.

II. ARTIFICIAL NEURAL NETWORKS

Artificial neural networks can be defined as a computation method consisting of many simple and linked processor units, processing data by its response to inputs [6]. The biological neuron model, called as spiking neuron model has a mathematical representation based on the biological process [7]. Like the biological neurons of human brain, artificial neural networks are composed of multiple nodes as shown in Fig.1. These nodes represent information processing elements called artificial neurons. Each neuron is connected to one another through linkages. Each link is associated with weight, called synaptic weight. The nodes take input data and process the data. The result from one node in a layer is passed to other nodes in the successive layers. The learning capability of ANN takes place by altering these weights. There are two artificial neural network topologies, FeedForward and Feedback. In FeedForward ANN, the data flow is unidirectional and there are no feedback paths. It has applications in the field of pattern recognition and classification. Feedback loops are allowed in feedback ANNs which are applicable in content addressable memories [8].

The fundamental unit of artificial neural networks is the artificial neurons. It represents the processing unit of the network. The artificial neuron model proposed by McCulloch Pitts is shown in Fig.1[9]. This model is widely used in pattern classification applications of artificial neural networks [10].

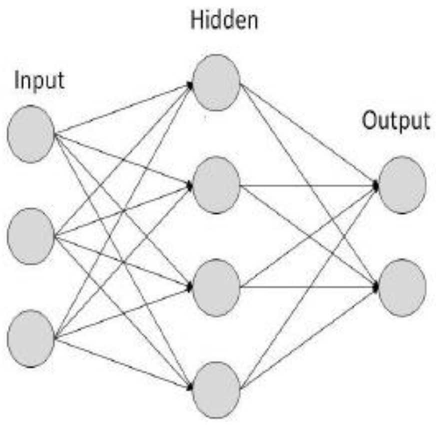


Fig.1.Typical Artificial Neural Network

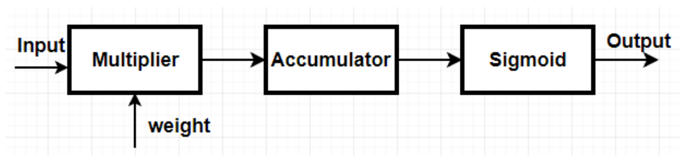


Fig.2.Single neuron architecture

The neuron architecture consists of a summation junction and an activation function. The neuron shown in fig.2 has N inputs,x1, x2, ...xN. Each link connecting these inputs to neuron possess some weights,w1,w2....wN. These inputs are multiplied with the corresponding weights and added together in the summation junction. The output of the summation junction is called net output. So, here a series of multiply and add operations (Multiply-Accumulate, MAC) are carried out. This is one of the major computations in the hardware implementation of artificial neural networks [11].

$$a = \sum_{i=1}^{N} x_i w_i \qquad (1)$$

Then an activation function 'a' is applied. The net value is transformed to output through a non linear activation function [4]. It can a linear, ramp or non linear function. It is also called threshold function which determines whether the neuron is fired or not. Most commonly used activation function is the sigmoid function. So, the neuron output can be as follows.

$$y = f(a) \qquad (2)$$

The sigmoid activation is as follows.

$$f(a) = \frac{1}{1 + e^{-a}} \qquad (3)$$

The computation of activation functions is another major operation in hardware implementation of artificial neural networks [13].It determines whether the neuron is activated or not. Usually, nonlinear functions like sigmoid function is used since it can approximate any function. The direct implementation of sigmoid function is not suitable since it contains infinite exponential series, instead some approximated alternatives is used. The accuracy of the activation functions, and thereby the training of the whole ANN depends on the approximation method selected for sigmoid function. Most commonly used approximation method used is Piece-wise linear approximation of Taylor series method. In this method, sigmoid function is approximated into linear curves of the form y = ax + b. According to this approximation, the sigmoid function can be written as follows[12].

$$f(v) = \begin{cases} 0 & ,for\ v > 4 \\ 0.0625x + 0.75, & for\ 0 < v < 4 \\ 0.5 & ,for\ v = 0 \\ 0.0625x + 0.25, & for - 4 < v < 0 \\ 1 & ,for\ v < -4 \end{cases}$$

III. Backpropagation Algorithm

Backpropagation is a supervised training method of artificial neural networks. It evaluates the error contribution of each neuron after a set of data is processed. The goal of backpropagation is to modify the weights so as to train the neural network to correctly map arbitrary inputs to outputs. Multi-layered perceptrons can be trained using the back-propagation algorithm. The goal is to learn the weights for all linkages in a multi layered network. The minimum of the error function in weight space is calculated using the method of gradient descent. The resultant weight which offers the minimum error function is the solution of the learning problem.

The steps involved in backpropagation algorithm are listed below.

1. Take a training set, inputs, initial weights and outputs.
2. Feed forward training set inputs.
3. Figure out the total net input to each hidden layer neuron.
4. Squash the total net input using an activation function, i.e, sigmoid function.
5. Repeat the process at output layers.
6. Calculate the error at output layer.
7. Backward pass for errors and new weights.
8. Process is continued until the error significantly reduced.

In the backward pass calculate the error as follows.

A. For each output node, calculate the error

$$\partial k = o_k(1 - o_k)(t_k - o_k)$$

where ∂k is the correction factor , o_k is the output value and $t_k$is the target value of the output layer.

B. For each hidden unit, compute the error

C. Update weights of all neurons in the hidden layer and output layer.

$$W_{ij} = W_{ij} + \partial W_{ij}$$

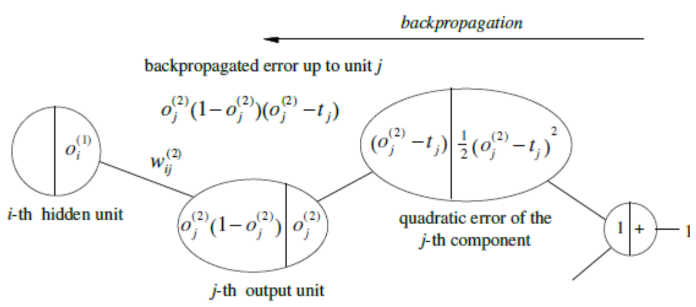

backpropagation
backpropagated error up to unit j
$o_j^{(2)}(1-o_j^{(2)})(o_j^{(2)}-t_j)$
$o_i^{(1)}$
$w_{ij}^{(2)}$
i-th hidden unit
$o_j^{(2)}(1-o_j^{(2)})$ $o_j^{(2)}$
j-th output unit
$(o_j^{(2)}-t_j)$ $\frac{1}{2}(o_j^{(2)}-t_j)^2$
quadratic error of the j-th component
1 +
1

Fig.3.Calculation of Backpropagated error

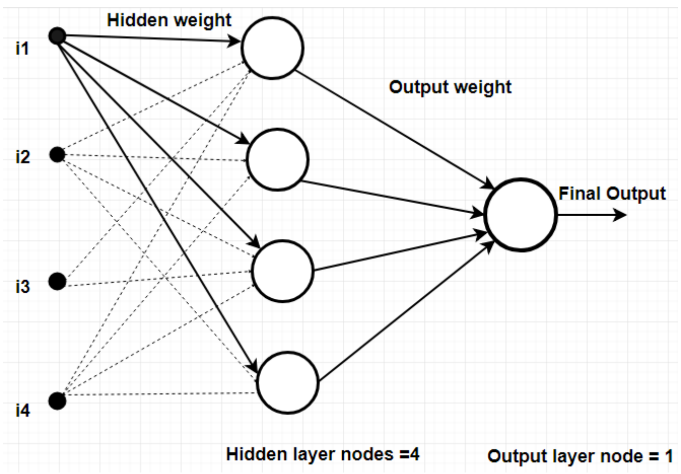


Fig.4.Design of network implemented in Backpropagation

IV. Methodology

A network with one hidden layer and an output layer is selected. The hidden layer consists of 4 neurons and output layer consists of one neuron. It is a 4 input network. Each of the neuron is implemented with a multiplier, summation junction and multiplier as shown in fig.4.These blocks are implemented in verilog in ModelSim 6.5b.A training set of 4 inputs, one output and initial weights are selected. Training set used is as follows, i1=1.5, i2 =1.25, i3=0.875, i4=0.625. Desired output is 0.375 with learn rate 0.25. The weights in the first layer are represented as hidden weights and weights in the output layer are represented as output weights. The hidden and output weights are modified by calculating the error backpropagated.

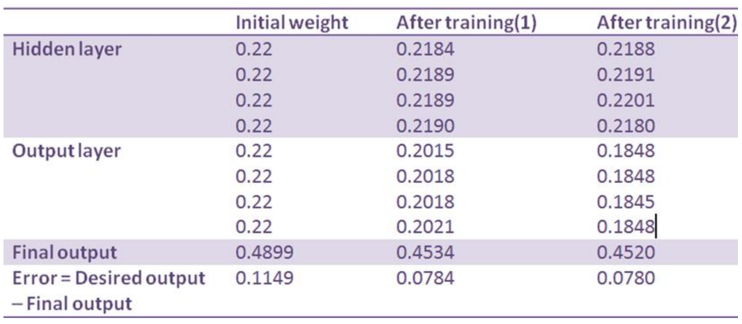
The network in which backpropagation is performed is shown in Fig.2.

V. Stimulation Results

In the first forward pass, the network computes the error or deviation at the output which is the difference between the actual output and the desired output. With respect to this error, correction in each weight is determined which includes the partial derivatives of the error function. The learn rate used is 0.25. From fig.5, it is clear that, the error decreases successively in each iterations as the weights are modified. The error with forward pass was 0.4899 with initial weights,0.22. After the first forward pass, weights are modified and error is decreased to about 0.4534. This process can be continued to a number of iterations depending on the requirement of the application of the network. The trained network with the new value of weights can take an unseen, new input and can correctly map the output. Since multiplier algorithm in backpropagation determines the operational speed, time delay and power consumption, a performance analysis is made based on different multipliers, i.e,Wallace tree multiplier and booth multiplier. The results are as shown in Table II [14].

TABLE I. Analysis Based on Multipliers

Performance	***Wallace Tree Multiplier***	***Booth Multiplier***
Power(µW)	678.43	378.33
Delay(ns)	1.73	3.98
Complexity	Complex	Most complex



	Initial weight	After training(1)	After training(2)
Hidden layer	0.22	0.2184	0.2188
	0.22	0.2189	0.2191
	0.22	0.2189	0.2201
	0.22	0.2190	0.2180
Output layer	0.22	0.2015	0.1848
	0.22	0.2018	0.1848
	0.22	0.2018	0.1845
	0.22	0.2021	0.1848
Final output	0.4899	0.4534	0.4520
Error = Desired output – Final output	0.1149	0.0784	0.0780

Fig.5.Errors obtained on first three iterations.

VI. Conclusion

In this work the architecture of feed forward neural network with 4 inputs, the hidden layer with four sigmoid neurons and the output layer with one sigmoid neuron is implemented. Different applications of neural network require different number of layers and nodes in each layer. As the number of hidden layer increases, modified weights will be more accurate, thus the performance increases. Backpropagation algorithm is implemented in this network as it is the basic tool for pattern classification application of ANN. The different modules of the algorithm are implemented in verilog HDL. The weights are modified according to backpropagated error. Sigmoid function with piece wise linear approximation is used as the activation function. 32bit single precision floating point standard is used. This also includes a performance analysis of the designed network base on different multiplier algorithms.

The motivation of this work stems from the advantages of implementing the neural hardware in VLSI.

ACKNOWLEDGMENT

The authors would like to thank the Department of Electronics and Communication, Amrita School Of Engineering, Kerala for providing us with great infrastructure and conducive atmosphere to work in.

REFERENCES

[1] M.T. Tommiska, "Efficient Digital Implementation of the Sigmoid Function for Reprogrammable Logic", IEEE Proceedings, Computers and Digital Techniques, vol. 150, no. 6, pp. 403- 411, 2003.

[2] R. Omondi, C. Rajapakse,"FPGA Implementation of Neural Networks", Springer U.S., 2006,ISBN 10-0-387-28485-0.

[3] Saravanan K1 and S. Sasithra,"Review on Classification based on artificial neural networks", International Journal of Ambient Systems and Applications (IJASA) Vol.2, No.4, December 2014.

[4] DR.S.M.Mukane,Ms.J.A.Kendule,"Flower Classification based on Neural Neywork based Image processing", IOSR Journal of Electronics and Communication Engineering (IOSR-JECE), Volume 7, Issue 3 (Sep. - Oct. 2013), PP 80-85,2013.

[5] ASR Ajai, N Nagaraj, "A Novel methodology for memory reduction in distributed arithmetic based dicrete wavelet transform,Procedia Engineering 30, 226-233,Dec 2012.

[6] Paul. J.Fox,"Massively parallel neural computation",UCAM-CL-TR-830,ISSN 14/6-2986,University Of Cambridge.

[7] J.Kumar ,Dr.Ramesh Bhakthavatchalu,"Design and implementation of Izhikevich , Hodgkin and Huxley spiking neuron models and their comparison" Proceedings of 2016 International Conference on Advanced Communication Control and computing Technologies, ICACCCT2016, 2016, pp 111-116.

[8] S. Himavathi, D. Anitha, and A. Muthuramalingam,"Feedforward Neural Network Implementation in FPGA using Layer multipliexing for effective resource utilisation", IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 18, NO. 3, MAY 2007.

[9] Simranjeet kaur, 2Baljeet Kaur," for Neural Networks and Their Applications', International Journal of Current Research", Vol. 5, Issue, 01, pp.153-156, January, 2013.

[10] Haitham Kareem Ali, Esraa Zeki Mohammed,"Design Artificial Neural Network using FPGA", IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.8, August 2010.

[11] A. Muthuramalingam, S. Himavathi, " Neural Network Implementaion using FPGA: Issues and Applications" , International Journal of Information Technology Volume 4 Number 2,2007.

[12] Sami El Moukhlis, Abdessamad Elrharras,Abdellatif Hamdoun," FPGA Implementation of Artificial Neural Networks", IJCSI International Journal of Computer Science Issues, Vol. 11, Issue 2, No 1, March 2014.

[13] Aydogan Savran, Serkan Ünsal, "Hardware Implementation of a Feedforward Neural Network Using FPGAs," Ege University,Department of Electrical and Electronics Engineering, 2003.

[14] J.Caroline Bonpapa, Usthulamari Penchalaiah Reddy," Design and Performance Analysis of multipliers using different logic styles, International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 , Vol. 4 Issue 04, April-2015 .