Modeling and Simulation of Backpropogation Algorithm Using VHDL

Jay Kumar[1], Ankit Sinha[2], Guncha Goswami[3], Manisha Kumari[4], Ratan Singh[5]

[1,2,3,4,5] *F.E.T. R.B.S. College, Agra (GBTU, Lucknow)*

[1] jaykumar_1981@yahoo.co.in
[2] maverik.ankit@gmail.com

***Abstract*— This paper deals with artificial neural network (ANN) architecture, the multilayer Feed-forward (MLFF) network with back propagation learning.**

The training of an artificial neural network involves two passes. In the forward pass, the input signals propagate from the network input to the output. In the reverse pass the calculated error signals propagate backwards through the network where they are used to adjust the weights. The calculation of output is carried out layer by layer in the forward direction. The weights of the output neuron layer are adjusted first since the target value of each output neuron is available to guide the adjustment of associated weights.

The system is designed to detect the odd parity. In this particular set, it gives an output of logic '1' when given inputs with odd parity and an output of logic '0' when provided with inputs with even parity.

***Keywords*— Back propagation, Artificial Neural Network (ANN), VHDL, Learning, Pattern Recognition (PR)**

I. INTRODUCTION

A neural network is a powerful data-modelling tool that is able to capture and represent complex input/output relationships. The motivation for the development of neural network technology stemmed from the desire to develop an artificial system that could perform "intelligent" tasks similar to those performed by the human brain. Neural networks resemble the human brain in the following two ways:

- A neural network acquires knowledge through learning.
- A neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights.

Artificial Neural Networks are being counted as the wave of the future in computing. They are indeed self-learning mechanisms which don't require the traditional skills of a programmer.[6]

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons.

A. A Simple Neuron

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.[7]

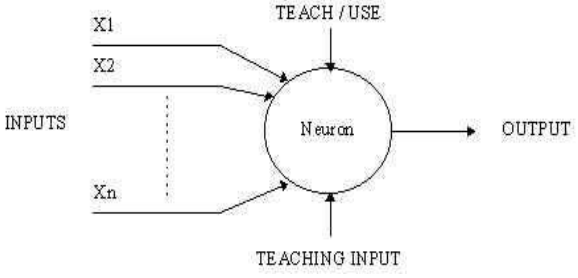

Fig 1: A Simple Neuron [2]

*B. A More **Complicated Neuron***

The previous neuron doesn't do anything that conventional computers don't do already. A more sophisticated neuron is the McCulloch and Pitts model (MCP). The difference from the previous model is that the inputs are 'weighted'; the effect that each input has at decision making is dependent on the weight of the particular input. The weight of an input is a number which when multiplied with the input gives the weighted input. These weighted inputs are then added together and if they exceed a pre-set threshold value, the neuron fires. In any other case the neuron does not fire.

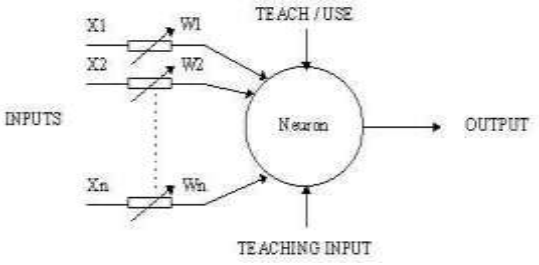

Fig 2: An MCP Neuron [2]

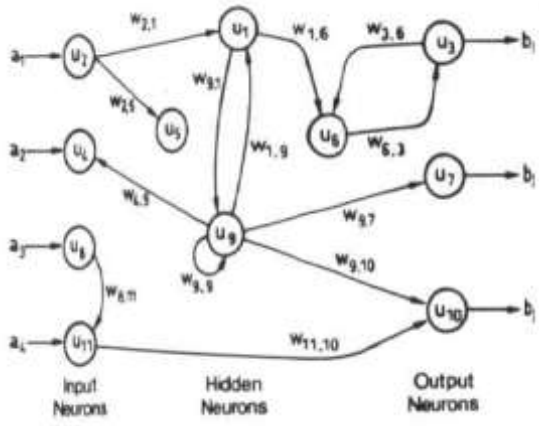

Fig 3: Complicated Network [2]

II. ARCHITECTURE OF NEURAL NETWORKS

A. *Feed Forward Network*

Feed-forward allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

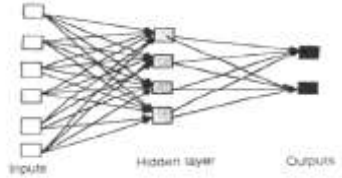

Fig 4: feed-forward Network [7]

B. *Feed Back Network*

Feedback network can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations.[6]

1) Perceptron

The Perceptron turns out to be an MCP model (neuron with weighted inputs) with some additional, fixed, pre--processing. Units labeled A1, A2, Aj, Ap are called association units and their task is to extract specific, localized featured from the input images. Perceptrons mimic the basic idea behind the mammalian visual system. They were mainly used in pattern recognition even though their capabilities extended a lot more.

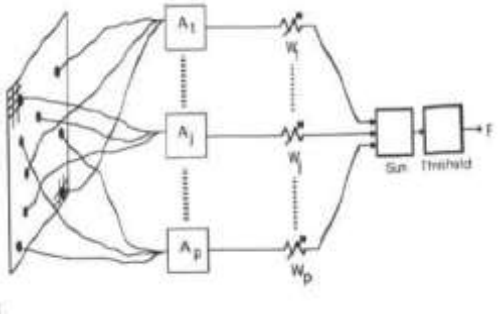

Fig 5: A Perceptron Model[2]

III. THE LEARNING PROCESS

The memorization of patterns and the subsequent response of the network can be categorized into two general paradigms:

A. *Supervised Learning*

It incorporates an external teacher, so that each output unit is told what its desired response to input signals ought to be. During the learning process global information may be required. Paradigms of supervised learning include error-correction learning, reinforcement learning and stochastic learning. Important issue concerning supervised learning is the problem of error convergence, i.e. the minimization of error between the desired and computed unit values. The aim is to determine a set of weights which minimizes the error.

Unsupervised learning uses no external teacher and is based upon only local information. It is also referred to as self-organization, in the sense that it self-organizes

data presented to the network and detects their emergent collective properties.

B. Transfer Function

The behavior of an ANN (Artificial Neural Network) depends on both the weights and the input-output function (transfer function) that is specified for the units.

The function that has been employed here is Sigmoidal Function. The reason being that a artificial neuron trains best for this particular relation between input and output of each subsequent layer.[7]

To make a neural network that performs some specific task, we must choose how the units are connected to one another, and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.

A network can be taught for three-layer network to perform a particular task by using the following procedure:

1. Present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.
2. Determine how closely the actual output of the network matches the desired output.
3. Change the weight of each connection so that the network produces a better approximation of the desired output.

$$\frac{1}{1+e^{-I}}$$

Where I = Input

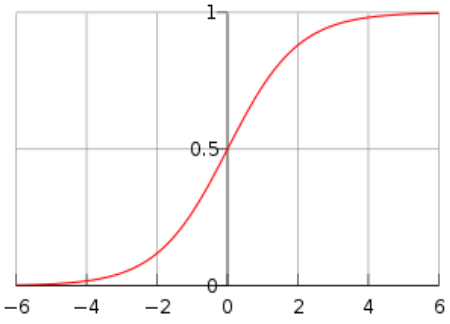

Fig 6: The Sigmoidal Function [2]

IV. THE BACK-PROPAGATION LEARNING

A Back propagation neural network uses a feed-forward topology, supervised learning, and back propagation learning algorithm. This algorithm was responsible in large part for the re-emergence of neural networks in the mid 1980s.

Back propagation is a general purpose learning algorithm. It is powerful but also expensive in terms of computational requirements for training. A back propagation network with a single hidden layer of processing elements can model any continuous function to any degree of accuracy (given enough processing elements in the hidden layer.[6]

The basic back propagation algorithm consists of three steps.

- Calculation of actual output with the help of randomized weights and input data set.
- Determination of error between target output and actual output.
- Updating the weight matrices for tolerable error range.

The input pattern is presented to the input layer of the network. These inputs are propagated through the network until they reach the output units. This forward pass produces the actual or predicted output pattern.
Because back propagation is a supervised learning algorithm, the desired outputs are given as part of the training vector. The actual network outputs are subtracted from the desired outputs and an error signal is produced.

This error signal is then the basis for the back propagation step, whereby the errors are passed back through the neural network by computing the contribution of each hidden processing unit and deriving the corresponding adjustment needed to produce the correct output. The connection weights are then adjusted and the neural network has just "learned" from an experience.

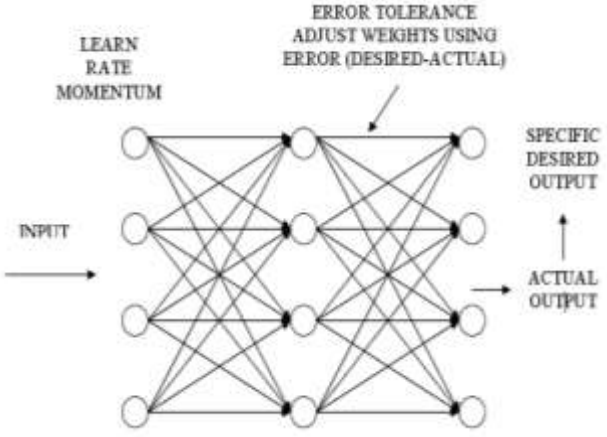

Fig7 : Backpropogation Network[7]

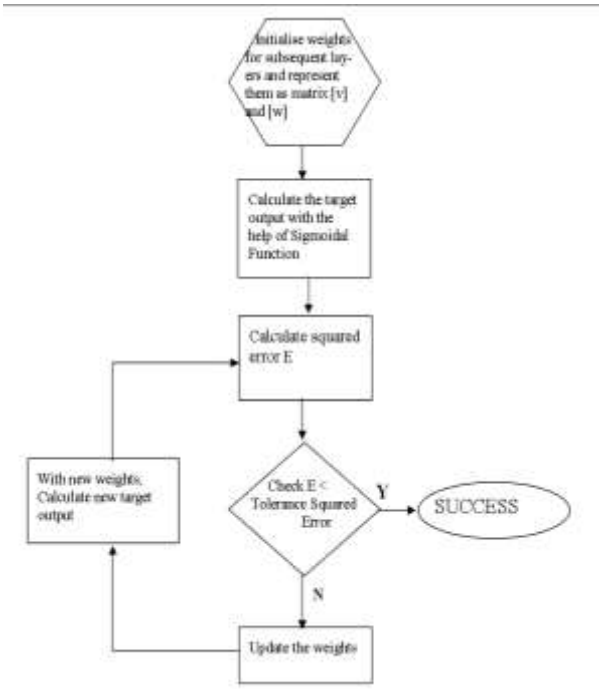

Fig8: Flowchart for Back-Propogation Algorithm

V. VHSIC HARDWARE DESCRIPTION LANGUAGE

VHDL is a hardware description language. It describes the behaviour of an electronic circuit or system, from which the physical circuit or system can then be attained (implemented). VHDL stands for VHSIC Hardware Description Language. VHSIC is itself an abbreviation for Very High Speed Integrated Circuits, an initiative funded by the United States Department of Defence in the 1980s that led to the creation of VHDL.[1]

VHDL has many features appropriate for describing the behaviour of electronic components ranging from simple logic gates to complete microprocessors and custom chips. Features of VHDL allow electrical aspects of circuit behaviour (such as rise and fall times of signals, delays through gates, and functional operation) to be precisely described. The resulting VHDL simulation models can then be used as building blocks in larger circuits (using schematics, block diagrams or system level VHDL descriptions) for the purpose of simulation.[3]

C.

VHDL is also a general-purpose programming language: just as high-level programming languages allow complex design concepts to be expressed as computer programs, VHDL allows the behaviour of complex electronic circuits to be captured into a design system for automatic circuit synthesis or for system simulation. Like Pascal, C and C++, VHDL includes features useful for structured design techniques, and offers a rich set of control and data representation features. Unlike these other programming languages, VHDL provides features allowing concurrent events to be described. This is important because the hardware described using VHDL is inherently concurrent in its operation.

One of the most important applications of VHDL is to capture the performance specification for a circuit, in the form of what is commonly referred to as a test bench. Test benches are VHDL descriptions of circuit stimuli and corresponding expected outputs that verify the behavior of a circuit over time. Test benches should be an integral part of any VHDL project and should be created in tandem with other descriptions of the circuit. Knowledgeable in VHDL is its adoption as a standard in the electronic design community.[4]

A. Data Set for Pattern Recognition Problem

For 3 Bit System

TABLE 1: TRUTH TABLE FOR PR PROBLEM

A(Input)	B(Input)	C(Input)	Y(Output)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Hence, with the help of this data set, the system is designed to detect the parity. In this particular set, it gives an output of logic ‘1’ when given inputs with odd parity and an output of logic ‘0’ when provided with inputs with even parity.

B. Simulation Results

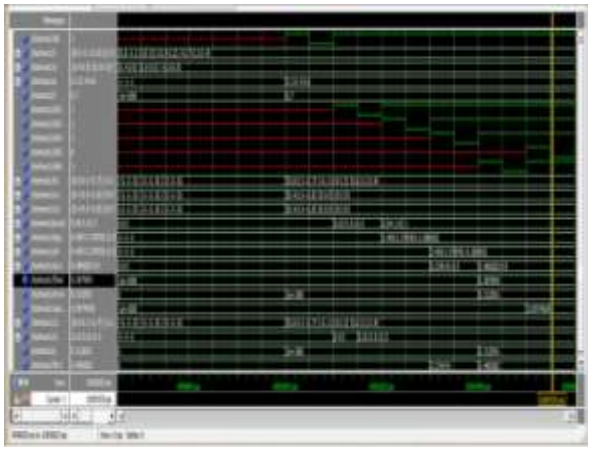

Fig 9: Simulation Window (Simulator: ModelSim SE 6.5)

VI. CONCLUSIONS

This paper describes the VHDL implementation of a supervised learning algorithm for artificial neural networks. The algorithm is the Error Back propagation learning algorithm for a layered feed forward network and this algorithm has many successful applications for training multilayer neural networks. VHDL implementation creates a flexible, fast method and high degree of parallelism for implementing the algorithm.

REFERENCES

[1] "A Brief History of VHDL", Douglas Ltd., 2005.
[2] J. Bhasker, "A VHDL Primer", Third edition, Pearson Education, Inc., 2004.
[3] Karen Parnell & Nick Mehta (2002), "Programmable Logic Design Quick Start Hand Book", Xilinx.
[4] Satish Kumar "Neural Networks "A classroom approach, TMH.
[5] S.Rajasekaran and G.A Vijayalakshmi Pai "Neural Networks, Fuzzy Logic, and Genetic Algorithms" first edition Prentice-hall of, 2006
[6] "Artificial Neural Network", Wikipedia Encyclopedia, Wikimedia Foundation, Inc., 2006.
[7] Douglas Perry "VHDL Programming by example" fourth edition McGraw-Hill Professional 2002.