Optimization of the performance of ANN using VHDL

K.L. Kar1, C.P. Giri2 Balai Ch. Kar3, Versha Dohare4, B.B. Mangaraj5

Abstract— In this paper, we proposed the design method of artificial neural networks using VHDL and implement in FPGA. VHDL is a programming language that has been designed and optimized for describing the behavior of digital systems. Back propagation algorithm for the design of a neuron is described. Back propagation is popular training algorithms for multilayer perceptrons. Over the last years many improvement strategies have been developed to speed up back propagation. It is very difficult to compare these different techniques, because most of them have been tested on very special data sets. Many "optimized" algorithms failed in training the considered task. Still back propagation is superior. The sigmoid nonlinear activation function is also used [1]. The neuron is then used in the design and implementation of a neural network using FPGA [12]. The simulation is done with Xilinx ISE 9.1i software. The neuron is then used in a multilayer neural network.The purpose of this work is to suggest and analyze several neuron implementations, show a way for the integration and control of the neurons within a neural network, and describe a way to implement a simple feed-forward neural network trained by BP algorithm using Xilinx software and implement in FPGA.

Index Terms— FPGA, ANN, BP, Xilinx, Back propagation, ISE, VHDL.

1 INTRODUCTION

A Neural Network is a powerful data-modeling 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, first. A neural network acquires knowledge through learning. Second a neural network's knowledge is stored within inter-neuron connection strengths known as synaptic weights. An ANN is a natural attempt at realizing multi-processor system similar to the biological neural network. However, the human brain is order of magnitude more complex than any ANN implemented so far. The building block of an ANN is the artificial neuron or processing element [3]. The network is composed of highly interconnected artificial neurons by synaptic weights and it performs useful computations through a process of learning.

- *Author name is currently pursuing masters degree program in electric power engineering in University, Country, PH-01123456789. E-mail: author_name@mail.com*
- *Co-Author name is currently pursuing masters degree program in electric power engineering in University, Country, PH-01123456789. E-mail: author_name@mail.com*
  (*This information is optional; change it according to your need.*)

Such an architecture is usually implemented using electronic components or simulated in software on a digital computer. The learning process involves adjusting the magnitude of signals passing through synapses. Such process, also called training, gives ANN an 'experience' by storing and processing information[9]. One of the most common methods of training an ANN is the Back-propagation Algorithm.

The power and usefulness of artificial neural networks have been demonstrated in several applications including speech synthesis, diagnostic problems, medicine, business and finance, robotic control, signal processing, computer vision and many other problems that fall under the category of pattern recognition. For some application areas, neural models show promise in achieving human-like performance over more traditional artificial intelligence techniques.

2 Learning method of ANN

Learning is a process by which the parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. An important property of neural networks is their ability to learn from input data with or without a teacher. Learning has long been a central issue for researchers developing neural networks. The type of learning is determined by the manner in which the parameter changes take place. All learning methods used for neural networks can be classified into two major categories:

2.1 Supervised learning

Supervised learning is a process that incorporates global information and/or a teacher. The teacher regulates the learning, informs the network what it has to learn and checks if it has properly learned or not. Supervised learning has information deciding when to turn off the learning, deciding how long and how often to present each datum for learning, and supplying performance information. Some well-known algorithms that implement supervised learning are error correction learning, reinforcement learning and stochastic learning.
Supervised learning can be subdivided into two subcategories: structural and temporal learning. The first tries to find the best input/output pattern relationship for each pattern pair. It is used to solve problems such as pattern classification and pattern matching. The second one is concerned with finding a sequence of patterns necessary to achieve some final outcome. The current response of the network is dependent on the previous inputs and responses.

2.2 Final Stage

In unsupervised or self-organized learning, the network is not given any external indication as to what the correct responses should be nor whether the generated responses are right or wrong. It is simply exposed to the various input-output pairs and it learns by the environment, that is, by detecting regularities in the structure of input patterns. So, unsupervised learning aims at finding a certain kind of regularity in the data represented by the exemplars. Roughly speaking, regularity means that much less data are actually required to approximately describe or classify the exemplars than the amount of data in exemplars.

3 Neuron Implementation

As A neural network is a powerful data-modeling tool that is able to capture and represent complex input/output relationships. It highly interconnected elementary computational units.They are called neural because the model of the nervous systems of human inspired them. Each computational unit (see Figure 1) has a set of input connections that receive signals from other computational units and a bias adjustment, a set of weights for each input connection and bias adjustment and a transfer function that transforms the sum of the weighted inputs and bias to decide the value of the output from the computational unit. The sum value for the computational unit (node j) is the linear combination of all signals from each connection ($X_{i)}$ times the value of the connection weight between node j and connection i ($W_{j\,i}$).
The term artificial neuron is used in the literature interchangeably with: node, unit, processing element or even computational unit. According to the approach or goals, one of those will be used. Here the term neuron will be kept in order to maintain the analogy to the biological structures when modeling the framework objects. But whenever necessary, neurons will be distinguished by using the terms natural or artificial.

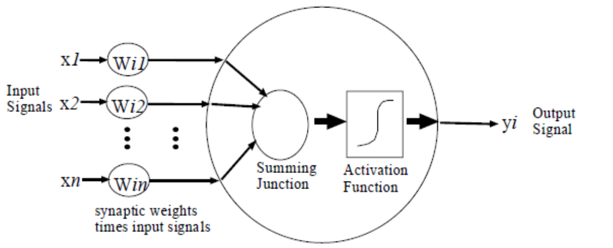

Fig. 1. Basic Neuron Module.

The common mathematical model of a neuron is shown in Figure1. According to the above, the net input to the ith unit of the next layer from the jth node can be written as:

$$\text{net } j = \textstyle\sum_i X_k W_{ik} \text{ ....(1)}$$

Where: X_k: is the input to the node. W_{jk} : is the weight associated to each input to the node from input k to node j.

This sum-of-products calculation plays an important role in the network simulations. Because there is often a very large number of interconnects in a network, the speed at which this calculation can be performed usually determines the performance of any given network simulation. Once the net input is calculated, it is converted to an activation function calculation. The determination of the output of the function is as follows:

$$y_j = f(\text{net } j\,) \text{ ....(2)}$$

4 The Back-propagation learning

The back-propagation algorithm is one of the most useful algorithms of ANN training[3]. We present the neuron implementation using suitable algorithm. A back propagation neural network uses a feed-forward topology, supervised learning and back propagation learning algorithm[7], [8].

It is a general purpose learning algorithm. It is powerful for training. A back propagation network with a single hidden layer of processing elements can model any continuous function to any degree of accuracy shown in figure 2.

There are variety of back propagation in the neural network. Since back propagation is based on a relatively simple form of optimization known as gradient descent, modifications of BP are conjugate gradient and Newton's methods [1]. However, "basic" back propagation is still the most widely used. Its two primary virtues are that it is simple and easy to understand, and it works for a wide range of problems. The basic back propagation algorithm consists of these steps [2][4].

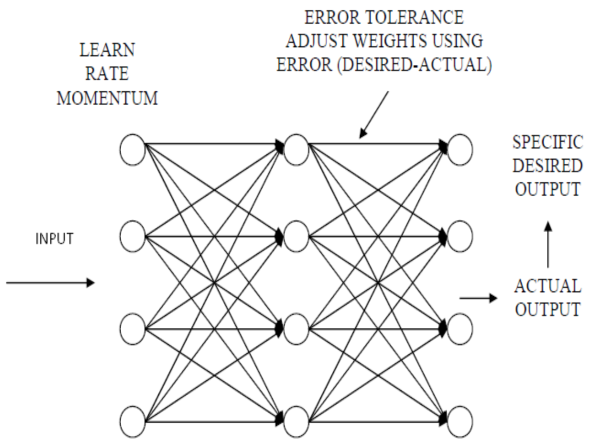

Fig. 2.. Back Propagation Network.

Step 1. Initialize weights and offsets. Set all weights and node offsets to small random values.

Step 2. Present input and desired output. The desired output is 1. The input could be new on each trial or samples from a training set.

Step 3. Calculate actual outputs. Use the sigmoid nonlinearity formulas to calculate outputs.

Step 4. Adapt weights.

Step 5. Repeat by going to step 2.

5 Methodology

For the implementation, VHDL language is used. VHDL is a hardware description language which simplifies the development of complex systems because it is possible to model and simulate a digital system form a high level of abstraction and with important facilities for modular design. VHDL has many features appropriate for describing the behavior of electronic components ranging from simple logic gates to complete microprocessors and custom chips. Features of VHDL allow electrical aspects of circuit behavior (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[5].
Back propagation algorithm logic has been divided into individual modules and these modules have been implemented in VHDL using behavioral modeling .The different modules are: synapse, neuron, error generator at the input, error generator at the output, weight update unit and a weight transfer unit. At the synapse inputs are multiplied by the corresponding weights and the weighted inputs are summed together to get four outputs. After the synapse is the neuron that calculates the output in accordance with the transfer function sigmoid and its derivative has also been calculated[10].
Further the neuron is followed by the synapse and the neuron because in this paper two layer network has been considered. Then the network is followed by the error generator at the output, which compares the output of the neuron with the target signal for which the network has to be trained. Similarly, there is error generator at the input, which updates the weights of the first layer taking into account the error propagated back from the output layer[4].

Finally, a weight transfer unit is present just to pass on the values of the updated weights to the actual weights. Then a final entity having structural modeling has been formulated in which all the entities are port mapped. The results constitute simulation of VHDL codes of different modules in Model-Sim simulator. The simulation of the structural model shows that the neural network is learning and the output of the second layer is approaching the target.

VHDL is used because, this code is valid to programme in FPGA[12]. This language is used to program and design digital circuits, for that, all signals will be digital signals. The way to cope with the problem is using a top-down method, that is, to divide a complex design in easier designs or modules, each module is redefined with greater details or divided in more subsystems.

For the developing of the system we have used a system based in an artificial neuron with only four dendrites as inputs, actually, the numbers of simultaneous spikes, in a short period of time, needed to excite a neuron are about 100 but we think that 4 dendrites are enough to show how a neuron works. To get that these 4 dendrites will be able to fire the output, we will have to increase the value of the weights, associated with the dendrites. They are defined like variables, can let the possibility of change the values to make a dendrite more significant than others. One of the most important characteristics of the neural networks is the ability of learning. For neurons, learning is the modification of the induced behaviour for the interaction with the environment and like result of experiences that drives to establishment of new response models to extern stimulus.

In artificial neuronal networks, the knowledge is represented in the weights of the connections between the neurons. The process of learning implies some numbers of changes in these connections. The neuronal network learns modifying the values of the weights of the network. The use of weights is very important weights are associated to the synapses and can increase or decrease the signals that arrive to a synapse. As mentioned earlier, we have defined variable weights as inputs, every weight is associated with a dendrite to let that the global neuronal network let the ability of learning. All the signals will be digital signals, for that, it is recommended a signal that synchronizes the global system.

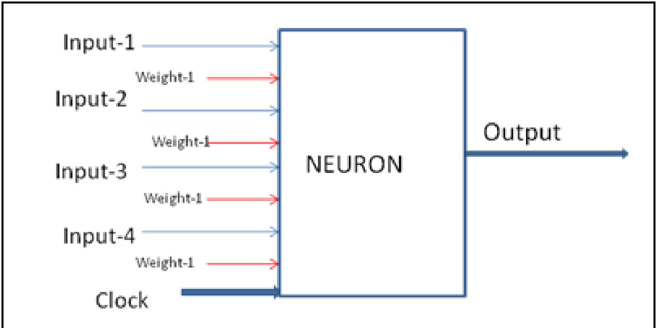

Figure 3. Neuron with weight inputs.

Figure 3 Shows 4 inputs referred to the weights of the connection between other neurons and our neuron. Every weight is in relation with each dendrite. It is necessary a clock signal to make synchronous the system.
When a spike arrives, the soma has basically two function.It will generate the potential action according to the input and compare if the addition of all potential actions in this instance of time is over a threshold.In that case, it will have to generate a pulse through the axon.

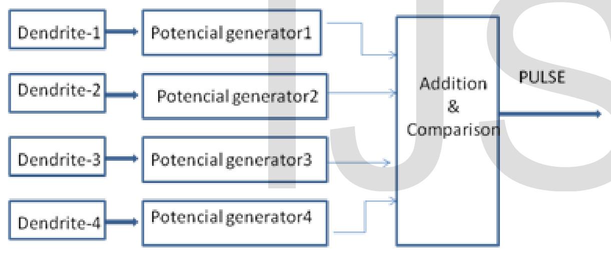

Figure 4. Functions of a neuron.

m43, m44are the intermediate signals obtained by multiplying inputs and the corresponding weights done is the output signals introduced to take care of timing constraints during port mapping.
Output0, Output1, Output2, Output3 and Output4 are the outputs.

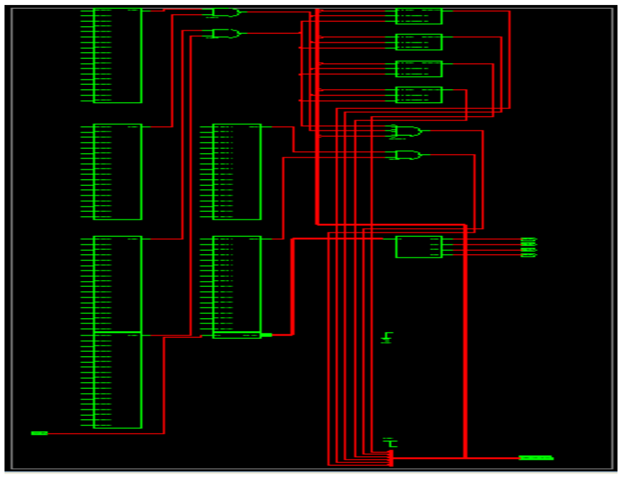

Figure 5. Schematic RTL View of ANN.

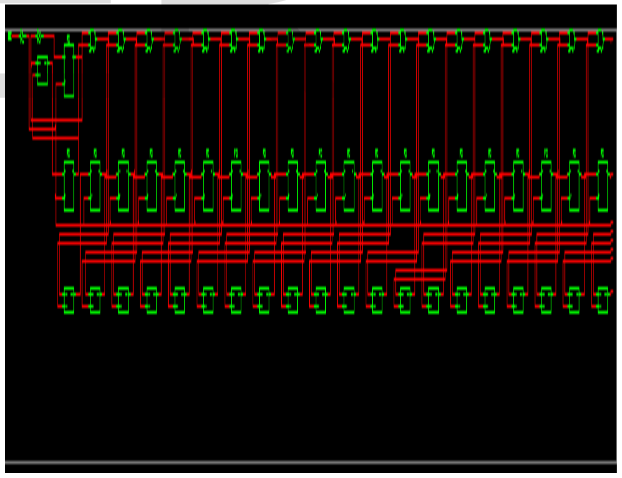

Fig. 6. Back Propagation Network.

6.Result

6.1 RTL View

i1, i2, i3, i4 are the input signals,
w11, w12, w13, w14, w21, w22, w23, w24, w31, w32, w33, w34, w41, w42, w43, w44 arethe input weights,
clock, enter are the input signals introduced to take care of timing constraints during port mapping.

s1, s2, s3, s4 are the sum of weighted outputs,m11, m12, m13, m14, m21, m22, m23, m24, m31, m32, m33, m34, m41, m42,

Fugure 5 and Figure 6 shows the RTL view of schematic and technology of ANN respectively.

6.2 Simulated Output

Figure 7 shows the output waveform of an ANN with corresponding changes of inputs. It is obtained by synthesized and simulated using xilinx 9.1i.

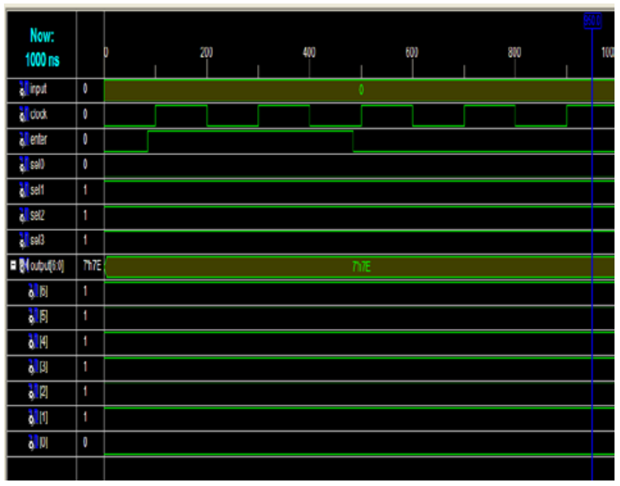

Fig. 7 Simulated result of ANN.

CONCLUSION

Many "optimized" algorithms failed in training the considered task, although most authors promised a algorithm superior to standard back propagation. Many of these algorithms have only been tested by training tiny artificial tasks. These results cannot be transferred to more complicated training sets. Especially for these kind of training sets optimization is necessary, whereas it's of little importance to speed up XOR learning. Nevertheless most of the algorithms are superior to standard back-propagation. 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. The proposed neural network has two layers with four neurons each and sixteen synapses per layer. Different situations may need neural networks of different scales. Such a situation can be overcome by combining a certain number of such unit neural networks.

On the other hand back-propagation updating the connections after every pattern presentation outperforms all global adaptive learning algorithms. Algorithms using local adaptation strategies greatly reduce the training time and also improve the network performance. The results constitute simulation of VHDL codes of different modules in Model-Sim simulator. The simulation of the structural model shows that the neural network is learning and the output of the second layer is approaching the target.

REFERENCES

[1] Simon Haykin, "Neural Networks", Second edition by, Prentice Hall of India, 2005.

[2] Christos Stergiou and Dimitrios Siganos, "Neural Networks", Computer Science Deptt. University of U.K., Journal, Vol. 4, 1996.

[3] Robert J Schalkoff, "Artificial Neural Networks", McGraw-Hill International Editions, 1997.

[4] Uthayakumar Gevaran, "Back Propagation", Brandeis University, Department of Computer Science.

[5] http://www.vhdl-online.de/tutorial/, April 2008.

[6] "Artificial Neural Network", Wikipedia Encyclopedia, Wikimedia Foundation, Inc.,2006.

[7] Pete Mc Collum, "An Introduction to Back Propagation Neural Networks", TheNewsletter of the Seattle Robotics Society.

[8] Pete Mc Collum, "An Introduction to Back Propagation Neural Networks", The Newsletter of the Seattle Robotics Society.

[9] Andrew Blais and David Mertz, "An Introduction to Neural Networks – Pattern Learning with Back Propagation Algorithm", Gnosis Software, Inc., July 2001.

[10] Harry B.Burke, "Evaluating Artificial Neural Networks for Medical Applications", New York Medical College, Deptt. Of Medicine, Valhalla, IEEE, 1997.

[11] Wan Hussain, Wan Ishak, "The Potential of Neural Networks in Medical Applications", Generation 5 Society, 2005.

[12] Syed M. Qasim, Shuja A. Abbasi, Bandar A. Almashary. "Advanced FPGA Architectures for Efficient Implementation of Computation Intensive Algorithms: A State-of-the-Art Review". MASAUM Journal of Computing, Volume 1 Issue 2, September 2009.

Authors Profile

Mr. K. L. Kar born in Bengal, India. He received his B. Tech. degree from N.I.T., Rourkela, India in 2003, M.E.(VLSI Design), from C.S.V.T.U., India in 2009. He join as an Asso. Prof. in E&TC department, CCEM, Raipur. Currently doing PhD from Jadavpur University, Kolkata, India. He is a member of ISTE. His present interests are VLSI Design, Neural Network, Fuzzy Logic and Programming with VHDL and Verilog.

Dr. B. B. Mangaraj received the BE degree from University College of Engineering, Burla, Orissa, India, in 1994 and the ME TeLE degree and Ph. D. (Engg.) degree from Jadavpur University, Kolkata, India, in 2003 and 2012 respectively. In 2006, he joined the Department of Electronics and Telecommunication Engineering, University College of Engineering, Burla, as a Lecturer. In 2011 he joined as a Reader again in the same department. His present interests are in Analysis, design, coding and optimization of wire antenna structures and micro-strip antennas starting from simple to complicated one. He is also taking interest to apply VLSI in RF Engineering. He has been honored with university medal for his ME course by Jadavpur University.