International Journal of Information Technology and Knowledge Management
July-December 2010, Volume 2, No. 2, pp. 529-532

ALGORITHM OF BACK PROPAGATION NETWORK IMPLEMENTATION IN VHDL

Amit Goyal

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. An Artificial neural network (ANN), also called a simulated neural network (SNN) or commonly just neural network (NN) is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing. In most cases ANN is an adaptive system that changes its structure based on internal or external information that flows through the network. Minsky and Papert (1969) showed that there are many simple problems such as the exclusive-or problem which linear neural networks can not solve. Note that term “solve” means learn the desired associative links. Argument is that if such networks can not solve such simple problems how they could solve complex problems in vision, language, and motor control.

1. Introduction

The highlighted area of the paper is the implementation of the back propagation algorithm of neural network in VHDL and formulation of individual modules of the Back Propagation algorithm for efficient implementation in hardware that creates a flexible, fast method and high degree of parallelism for implementing the algorithm. An ANN can create its own organization or representation of the information it receives during learning time. ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability. Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage [1].

Use of back Propagation Neural Network solution:

- A large amount of input/output data is available, but you’re not sure how to relate it to the output.
- The problem appears to have overwhelming complexity, but there is clearly a solution.
- It is easy to create a number of examples of the correct behavior.
- The solution to the problem may change over time, within the bounds of the given input and output parameters (i.e., today 2 + 2 = 4, but in the future we may find that 2 + 2 = 3.8).
- Outputs can be “fuzzy”, or non-numeric.

Swami Devi Dyal Institute of Engineering & Technology, Barwala (India).
Email: amitgoyal1979@yahoo.co.in

Hardware description languages are especially useful to gain more control of parallel processes as well as to circumvent some of the idiosyncrasies of the higher level programming languages.

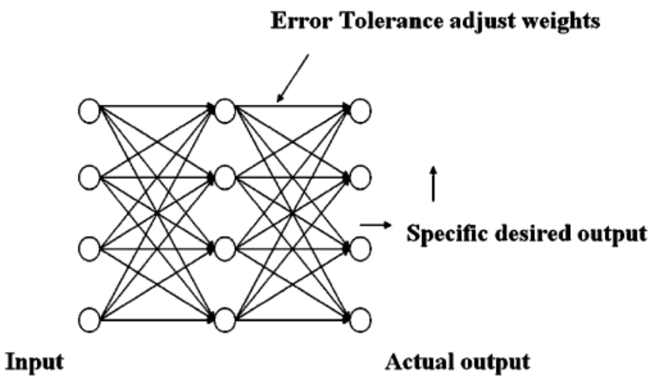


Fig 1: Back Propagation Network

VHDL is a programming language that has been designed and optimized for describing the behavior of digital systems. VHDL has many features appropriate for describing the behavior of electronic components ranging from simple logic gates to complete microprocessors and custom chips. 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.

2. Objective

In this paper I have proposed an expandable on-chip back-propagation (BP) learning neural network. The network has four neurons and 16 synapses. Large-scale neural networks with arbitrary layers and discretional neurons per layer can be constructed by combining a certain number of such unit networks. A novel neuron circuit with programmable parameters, which generates not only the sigmoid function but also its derivative, is proposed. The back-propagation (BP) algorithm often provides a practical approach to a wide range of problems. There have been many examples of implementations of general-purpose neural networks with on-chip BP learning.

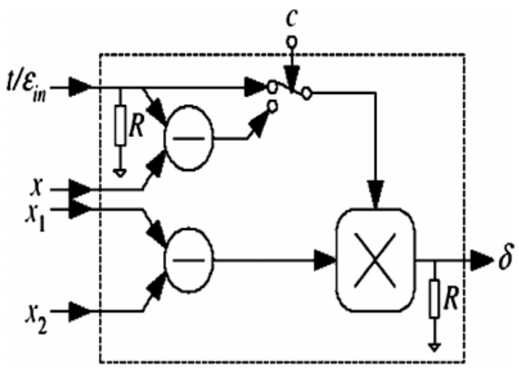


Fig 2: Block Diagram of Error Generator

The block diagram of the error generator unit in shown in the above figure. It provides a weight error signal δ. The control signal c decides whether the corresponding neuron is an output neuron. $t/\ \varepsilon_{in}$ is a twofold port.

If c = 1, a target value t is imported to the t/ε_{in} port and δ is obtained by multiplying

$$(t - x) \text{ by } (x_1 - x_2);$$

If c = 0, a neuron error value ε_{in} is imported to the $t/\ \varepsilon_{in}$ port and d is achieved by multiplying ε_{in} by $(x_1 - x_2)$. So no additional output chips are needed to construct a whole on-chip learning system.

In broad sense the objectives of thesis are:

- Exploration of a supervised learning algorithm for artificial neural networks i.e. the Error Back propagation learning algorithm for a layered feed forward network.
- Formulation of individual modules of the Back Propagation algorithm for efficient implementation in hardware.
- Implementation of the Back Propagation learning algorithm in VHDL. VHDL implementation creates a flexible, fast method and high degree of parallelism for implementing the algorithm.
- Analysis of the simulation results of Back Propagation algorithm.

2.2. Description

The back propagation algorithm is an involved mathematical tool; however, execution of the training equations is based on iterative processes, and thus is easily implement able on a computer [2].

- Weight changes for hidden to output weights just like Widrow-Hoff learning rule.
- Weight changes for input to hidden weights just like Widrow-Hoff learning rule but error signal is obtained by "back-propagating" error from the output units.

2.3. Problem Statement

Minsky and Papert (1969) showed that there are many simple problems such as the exclusive-or problem which linear neural networks can not solve. Note that term "solve" means learn the desired associative links. Argument is that if such networks can not solve such simple problems how they could solve complex problems in vision, language, and motor control. Solutions to this problem were as follows:

- Select appropriate "recoding" scheme which transforms inputs.
- Perceptron Learning Rule — Requires that you correctly "guess" an acceptable input to hidden unit mapping.
- Back-propagation learning rule — Learn both sets of weights simultaneously.

2.4. Background

The study of the human brain is thousands of years old. With the advent of modern electronics, it was only natural to try to harness this thinking process. [1], [3].

One particular structure, the feed forward, back-propagation network, is by far and away the most popular. Most of the other neural network structures represent models for "thinking" that are still being evolved in the laboratories. Yet, all of these networks are simply tools and as such the only real demand they make is that they require the network architect to learn how to use them. [4]

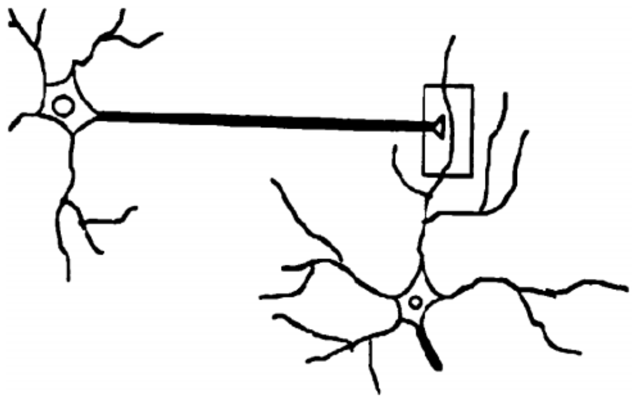


Fig 3: The Synapse

3. Approach

There are many existing theoretical approaches to strategy - designed strategy, strategy as revolution etc and yet few examples of organizations applying these well defined models to secure competitive advantage in an environment of constant change. Proper utility of technology and time will increase the general output as given in the below theory [5].

3.1. Theory

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

- The network is presented 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.
- It is determined how closely the actual output of the network matches the desired output.
- The weight of each connection can be changed so that the network produces a approximation of the desired output.

3.2. Evolutionary Approach

Take a collection of training patterns for a node, some of which cause it to fire (the 1-taught set of patterns) and others, which prevent it from doing so (the 0-taught set). Then the patterns not in the collection cause the node to fire if, on comparison, they have more input elements in common with the 'nearest' pattern in the 1-taught set than with the 'nearest' pattern in the 0-taught set. If there is a tie, then the pattern remains in the undefined state.

For example, a 3-input neuron is taught to output 1 when the input (X1, X2 and X3) is 111 or 101 and to output 0 when the input is 000 or 001. Then, before applying the firing rule, the truth table is:

Truth Table 1
Before Applying the Firing Rule

X1:	0	0	0	0	1	1	1	1
X2:	0	0	1	1	0	0	1	1
X3:	0	1	0	1	0	1	0	1
OUT:	0	0	0/1	0/1	0/1	1	0/1	1

As an example of the way the firing rule is applied, take the pattern 010. It differs from 000 in 1 element, from 001 in 2 elements, from 101 in 3 elements and from 111 in 2 elements. Therefore, the 'nearest' pattern is 000, which belongs, in the 0-taught set. Thus the firing rule requires that the neuron should not fire when the input is 001. On the other hand, 011 is equally distant from two taught patterns that have different outputs and thus the output stays undefined (0/1).

By applying the firing in every column the following truth table is obtained:

Truth Table 2
After Applying the Firing Rule

X_1:	0	0	0	0	1	1	1	1
X_2:	0	0	1	1	0	0	1	1

3.3. Methodology / Planning of Work

Extensive literature survey will be conducted to study the literature available in the field of Back Propagation Algorithm in VHDL Implementation, characterization and simulation. Generally 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. his function typically falls into one of three categories:

- Linear (or ramp): The output activity is proportional to the total weighted output.
- Threshold: The output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.
- Sigmoid: The output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

Step function	Sign funt.	Sigmoid funt.
Step(x) = 1, if	Sign(x) = +1	Sigmoid(x) =
x>=threshold	if x>= 0	1/ (1+ex)
x < threshold	Sign(x) = –1	
	If x < 0	

Fig 4: Transfer Functions

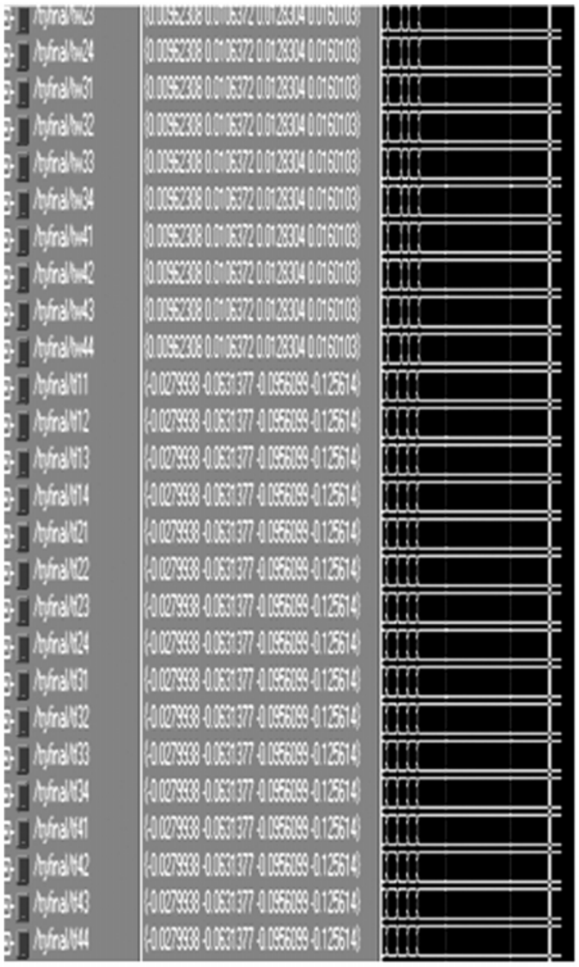
4. Simulation Results

Before starting the back propagation learning process, we need the following:

- The set of training patterns, input, and target;
- A value for the learning rate;
- A criterion that terminates the algorithm;

- A methodology for updating weights;
- The nonlinearity function (usually the sigmoid);
- Initial weight values (typically small random values.

Then the simulation process is implemented. Simulation will allow investigation of models which are highly intractable from a mathematical point of view. Simulation allows checking whether certain approximations made in the mathematical model for the sake of mathematical tractability [6]. Then iterations will be performed and the results of the implementation of the back propagation algorithm may be presented in the form of MODELSIM SE 5.5c.



```
/tyfinal/hw23   {0.00962308 0.0106372 0.0128304 0.0160103}
/tyfinal/hw24   {0.00962308 0.0106372 0.0128304 0.0160103}
/tyfinal/hw31   {0.00962308 0.0106372 0.0128304 0.0160103}
/tyfinal/hw32   {0.00962308 0.0106372 0.0128304 0.0160103}
/tyfinal/hw33   {0.00962308 0.0106372 0.0128304 0.0160103}
/tyfinal/hw34   {0.00962308 0.0106372 0.0128304 0.0160103}
/tyfinal/hw41   {0.00962308 0.0106372 0.0128304 0.0160103}
/tyfinal/hw42   {0.00962308 0.0106372 0.0128304 0.0160103}
/tyfinal/hw43   {0.00962308 0.0106372 0.0128304 0.0160103}
/tyfinal/hw44   {0.00962308 0.0106372 0.0128304 0.0160103}
/tyfinal/ht11   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht12   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht13   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht14   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht21   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht22   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht23   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht24   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht31   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht32   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht33   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht34   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht41   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht42   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht43   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
/tyfinal/ht44   {-0.0279938 -0.0631377 -0.0956099 -0.125614}
```

Fig 5: Simulation Results for the Final Entity

5. CONCLUSION

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. 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 MODELSIM SE 5.5c. 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] Christos Stergiou and Dimitrios Siganos, "Neural Networks", Computer Science Deptt. University of U.K., Journal, 4, 1996.

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

[3] Jordan B.Pollack, "Connectionism: Past, Present and Future", Computer and Information Science Department, The Ohio State University, 1998.

[4] Harry B.Burke, "Evaluating Artificial Neural Networks for Medical applications", New York Medical College, Deptt. of Medicine, Valhalla, IEEE, 1997.

[5] Tom Baker and Dan Hammerstrom, "Characterization of Artificial Neural Network Algorithms", Dept. of Computer Science and Engineering, Oregon Graduate Center, IEEE, 1989.

[6] Christopher Cantrell, Dr. Larry Wurtz, "A Parallel Bus Architecture for Ann's", University of Albana, Deptt. of Electrical Engg, Tuscaloosa, IEEE, 1993.