Performance Evaluation of Error Back Propagation Algorithm for Non-Linear Classification and Function Approximation in VHDL Platform

Soumava Kumar Roy
Department of Electronics and Communication
Manipal Institute of Technology
Manipal, India

Crefeda Faviola Rodrigues
Department of Electronics and Communication
Manipal Institute of Technology
Manipal, India

ABSTRACT
In this paper we present the implementation of Error Back Propagation Training Algorithm (EBPT) in VHSIC Hardware Descriptive Language (VHDL) platform for two standard benchmark problems of Nonlinear Classification of XOR function and Sine wave Generation. The effect of variation of learning parameters on accuracy of the output and speed of convergence of the algorithm are presented. Improved speed of convergence without much change in accuracy was obtained by incorporating Momentum method.

General Terms
Adaptive Signal Processing, Machine Learning, Non Linear Classification.

Keywords
Error Back Propagation Training, VHDL, Momentum Method

1. INTRODUCTION
The concept of Artificial Neural Networks (ANN) has emerged from simulating the versatility of human brain to deal with the ambiguity of digital computers. The ANN consists of layers of basic computing units called neurons. The Computing units include a summing and threshold units and when these computing elements are arranged in layers and trained properly they perform many non-linear functions. One of the popular ways of training of Multi layered perceptron networks is through the use of Error Back Propagation Training Algorithm [1]. Through the principle of gradient descent it minimizes the error of the networks outputs and desired output for a given training cycle and feeds this error to the network for adaptive weight changes. ANNs are implemented in software, trained and simulated on general-purpose sequential computers for emulating a wide range of neural networks models. Software implementations offer flexibility. Whereas Hardware implementation of Neural Networks exploits the inherent parallel processing capabilities in these Neural Networks to achieve faster learning and convergence speed of the desired outputs [2]. FPGA implementation offers a solution for Hardware Neural Network Implementation [3]-[5]. Multilayer Feed Forward Networks implemented have been implemented on FPGA by reducing resource requirements without compromising on speed [4]. Our work aims as evaluating the performance of ANN algorithm like Error Back Propagation using behavioral simulation in VHDL by varying Learning Parameters of EBPT for the successful implementation on Hardware. [6].

2. ERROR BACK PROPAGATION STRUCTURE
The error back propagation topology consists of a layered structure of a hidden layer and output layer. Each layer consist processing units called neurons. The functionality of the neuron is to do a weighted summation of the inputs and depending on the nonlinear function called activation function to generate an output. The input layer feeds the hidden layer neurons and the output of the hidden layer in turn feeds the output layer neurons. Each layer's input set is augmented with a bias of '-1', in accordance with the perceptron learning rule. The EBPT algorithm incorporates two phases for learning, the multilayered feed forward phase and the back propagation phase. The Processing units in the feed forward performs parallel computations of multiplication and addition to produce an output called net and the final output of the neuron is determined by the activation function. The back propagation phase involves computing the error by subtracting the desired output from generated output of the feed forward phase and feeding back the error to the network to adapt the weights until the error lowers itself below a predefined threshold.

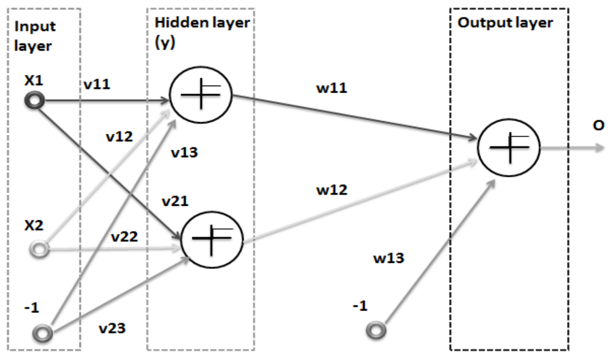
3. NON LINEAR CLASSIFICATION


Figure1. Feed forward structure for XOR classification

Figure 1, depicts the feed forward structure for XOR classification. This structure is implemented using behavioral simulation in VHDL. Each iteration is in accordance with the rising edge of 'clk'. The input layer consist of two input nodes (X1 and X2) is augmented with a bias of '-1'.These inputs feed the hidden layer neurons. The output of the hidden layer (y) is augmented with bias '-1' and in turn is fed to a single neuron at the output layer. The weights of both hidden layer (v11, v12, v13, v21, v22, v23) and output layer (w11, w12, w13) are initialized to random values. The neurons at both hidden layer and output use continuous unipolar neurons. Experimental results are tabulated by varying the learning parameters such as Learning Constant (ɳ), Steepness Coefficient (λ), initial weights, number of hidden layer neurons and positive momentum coefficient (α), Shown in Figure 2, is the simulation window where the counting variable denotes the 4 inputs of the XOR taken in order. And 'o' denotes the approximate outputs in order

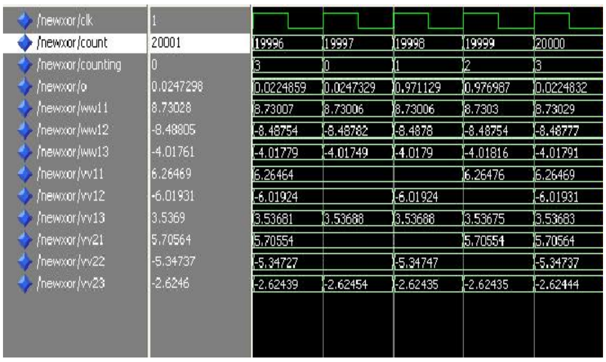


Figure2. Simulation window for XOR function Outputs at iteration 19,997

3.1 Learning Constant (ɳ) variations:

Variation of learning constant has a significant effect on the convergence and effectiveness of the EBPT algorithm. Since optimum learning constant is problem specific, on the basis of trial and error we showed that higher the value of learning constant better was the accuracy of the outputs at a given iteration as given in Table 1. Also observed that higher the value of learning constant lower the number of iterations for a given satisfactory output given in Table 2.

TABLE 1. LEARNING CONSTANT VARIATION AT 10,000 ITERATIONS

Inputs	Theoretical Outputs		Learning Constant		
			ɳ=0.01	*ɳ =0.5*	*= 1.0*
00	0	**Practical Output**	0.47	0.0388	0.0247
01	1		0.4506	0.9542	0.9711
10	1		0.56451	0.9646	0.9769
11	0		0.5024	0.035	0.0224

TABLE 2. NUMBER OF ITERATIONS FOR DIFFERENT VALUES OF LEARNING CONSTANT

Learning Constant	*ɳ = 0.01*	*ɳ = 0.5*	*ɳ = 1.0*
No. of iteration	99,000	20,000	10,000

3.2 Steepness Coefficient (λ) variations:

The neurons activation function is characterized by steepness coefficient (λ), given by Equation 1.

$$f(net) = \frac{1}{1+e^{-\lambda_{net}}} \quad (1)$$

f(net)- continuous unipolar activation function

net- weighted summation of input

Variation in steepness coefficient showed that higher values of steepness coefficient produced approximately similar results as that of learning constant. This implied that varying anyone of the parameters learning constant or steepness coefficient was sufficient to obtain correct classification of XOR outputs as given in Table 3. By increasing steepness Coefficient reduced iterations were observed as given in Table 4.

TABLE 3. STEEPNESS COEFFICIENT VARIATION AT 10,000 ITERATIONS

Steepness Coefficient	*λ= 0.01*	*λ=0.5*	*λ= 1.0*
No. of iterations	109000	10,000	2,500

TABLE 4. NUMBER OF ITERATIONS FOR DIFFERENT VALUES OF STEEPNESS COEFFICIENT

Input	Theoretical outputs		Steepness Coefficient		
			λ= 0.01	*λ= 0.5*	*λ= 1.0*
00	0	**Practical Outputs**	0.49655	0.0742	0.0247
01	1		0.496535	0.9117	0.9711
10	1		0.49656	0.9353	0.9769
11	0		0.49652	0.0649	0.0224

3.3 Initial Weights

Randomness in the initial weights led to better convergence of the output whereas while initialization of same weights (1.0) did not give satisfactory results at 4,90,000 iterations depicted in Table 5.

TABLE 5. OUTPUTS FOR DIFFERENT WEIGHTINITIALIZATIONAT 4,90,000 ITERATIONS

Inputs	Theoretical outputs		Initial Weights	
			Same weights=1.0	***Random weights***
00	0	**Practical Outputs**	0.0042	0.003
01	1		0.6488	0.9964
10	1		0.7016	0.997
11	0		0.7395	0.0028

3.4 Number of Hidden Neurons

By increasing the number of neurons at the hidden layer improvement in speed of learning without affecting the accuracy much was obtained, thereby making the network faster in calculations of the outputs in Table 6 and 7.

TABLE 6. VARIATION IN THE NUMBER OF HIDDEN LAYERS AT 2000 ITERATIONS

Inputs	Theoretical outputs		Number of Hidden Neurons	
			Hidden neurons=2	*Hidden neurons=3*
00	0	Practical Outputs	0.0899	0.0474
01	1		0.8905	0.9172
10	1		0.9263	0.9321
11	0		0.0759	0.0944

TABLE 7. NUMBER OF ITERATIONS FOR DIFFERENTNUMBER OF HIDDEN LAYER

Variation in Hidden Neurons	*Hidden neurons=2*	*Hidden neurons=3*
No. of iterations	4000	2500

3.5 Momentum Method

Momentum method is based on feeding fraction of the previous training cycle weights to the current weight updation cycle as given by Equation 2.

$$\Delta W(t) = \eta \Delta E + \alpha \Delta W(t-1) \quad (2)$$

$\Delta W(t)$- current weight updation

ΔE- Change in error

$\Delta W(t\text{-}1)$- fraction of previous weights

The positive momentum coefficient is given by 'α' and varying this parameter, led to better accuracy of the outputs in Table 8 and drastic improvement in speed in Table 9.

TABLE 8. VARIATION IN MOMENTUM COEFFICIENT AT 1000 ITERATIONS

Inputs	Theoretical outputs		Momentum Coefficient		
			α = 0.01	*α =0.5*	*α = 1.0*
00	0	Practical Outputs	0.1588	0.0739	0.0507
01	1		0.784	0.8834	0.9148
10	1		0.8619	0.9124	0.9335
11	0		0.2027	0.1249	0.09422

TABLE 9. NUMBER OF ITERATION FOR DIFFERENT VALUES OF MOMENTUM COEFFICIENT

Variation in Momentum Coefficient	*α = 0.01*	*α =0.5*	*α = 1.0*
No. of iterations	1800	1,300	1,000

4. SINE WAVE GENERATION

Figure 3, shows the schematic for the feed forward phase for the generation of 50 Hz Sine Wave. The input layer consists of a single input node augmented with bias '-1'. The output of hidden layer is also augmented with a '-1' bias. The weights shown are initialized randomly. The hidden layer consisted of two bipolar continuous neurons as given in Equation 3.

$$f(net) = \frac{2}{1+e^{-\lambda net}} - 1 \qquad (3)$$

The output layer consisted of single linear neuron. The choice of using linear neuron was to ensure the output follows the continuous stream of inputs. Experimental results are tabulated by varying the learning parameters such as Learning Constant (η), Steepness Coefficient (λ) and positive momentum coefficient (α)

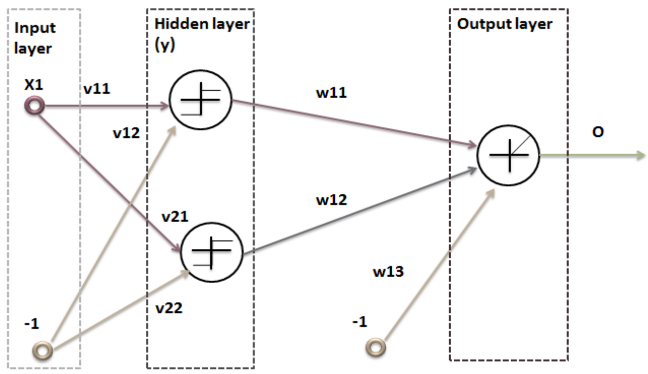


Figure 3 Feed forward phase for Generation of 50 Hz Sine Wave.

The simulation results for 50 Hz Sine wave generation for optimum values of learning factor (η)=1.0, steepness coefficient (λ) =1.0 and positive momentum coefficient (α)=0.5 is shown in Figure 4. The initial weight adaptation is shown encircled.

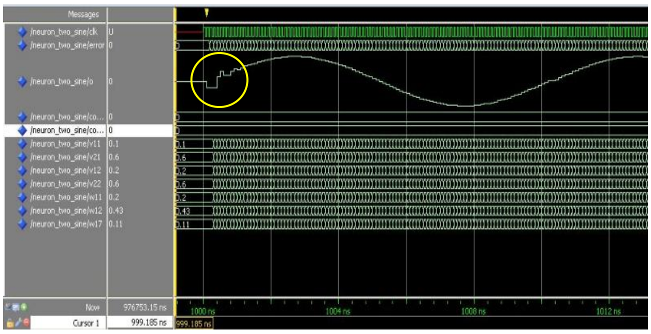


Figure 4 Simulation window for Generation of Sine Wave at 1000ns

As observed from the Figure 5, by changing the momentum coefficient to a higher value (0.8, i.e. by giving higher fraction of weight change of the previous iteration to the current iteration) and keeping other parameters fixed such as learning factor(1.0) and steepness coefficient (1.0),the system was unable to learn and learning stopped at 9017 nanoseconds. The weights go out of the required range and thus it will require a very large number of iterations to track back the correct output.

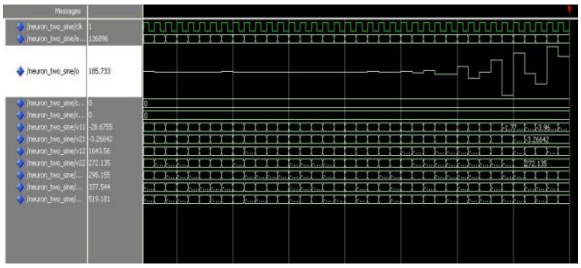


Figure 5. Simulation window for Generation of Sine Wave at 9017ns

In Figure 6, shows, by keeping momentum coefficient at the optimum value of 0.5, learning constant at optimum value 1.0 and varying steepness coefficient to 0.5, a perfect sinusoid output was not tracked.

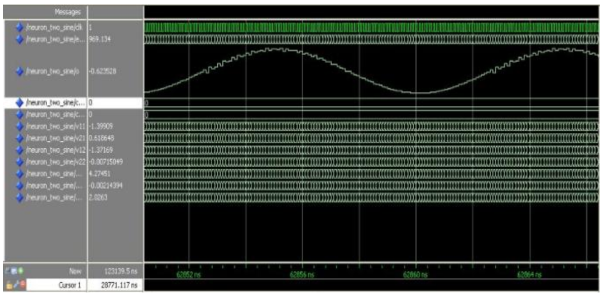


Figure 6. Simulation window for Generation of Sine Wave at 62852ns

By keeping a very low learning factor equal to 0.01 and keep the other parameters at optimum values a perfect sinusoid was not tracked as shown in Figure 7.

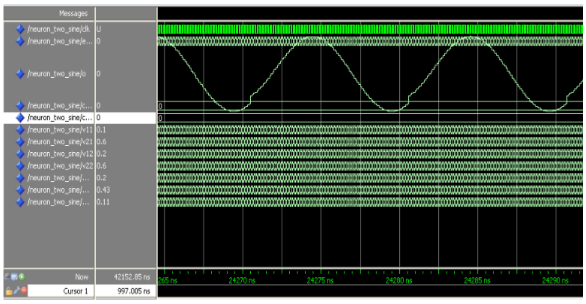


Figure 7. Simulation window for Generation of Sine Wave at 24270ns

5. CONCLUSION

Implementation of Error Back Propagation Algorithm is carried out in VHDL platform for the purpose of in depth analysis of the effects of learning parameters on the accuracy and speed of convergence. Training is performed for a typical Nonlinear XOR classification and Sine Wave Generation problem. Experimental results verify that optimum parameters for satisfactory output are problem specific and are obtained through trial and error.

Figure 8.Shows the effects of varying learning parameters on the speed of learning. For a given problem varying either Learning Constant or Steepness Coefficient is sufficient for satisfactory results. Increase in number of hidden neurons leads to further reduction in the number of iterations. Fastest convergence of the output is obtained by introducing momentum method.

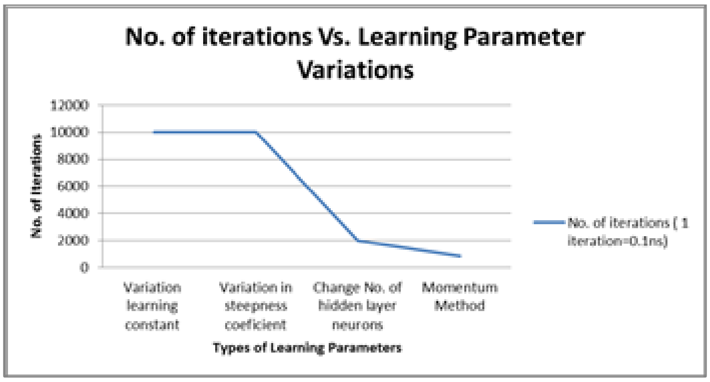


Figure. 8. Graph of effects of Learning Parameters on the Number of iterations for XOR classification

6. ACKNOWLEDGMENT

I would also extend my appreciation towards Dr. K. PrabhakarNayak, HOD, E&C, MIT, Manipal who provided us with ample space and technology access for the smooth completion of the project work and taking a stand for regular presentations to be made so that we could get technical insights from a group of teachers and so that we kept meeting our deadlines.My project would have not seen the light of the day without my guide Mrs. T. K. PadmaShri, Assosiate Professor, Senior Scale, Dept. of E&C, MIT, Manipal whose constant efforts at perfecting even the slightest detail of the project helped render a lot of professional insight into the project, not to mention her technical acumen that finally helped the project come afloat.

7. REFERENCES

[1] Jacek M. Zurada, “ Introduction to Aritificial Neural Networks”, Jaicob Publishing House, India 2002,ISBN 0-3 14-93391-3.

[2] JiPeirong, Wang Peng , Zhao Qin, Zhao Li, “A New Parrallel Back Propagation Algorithm for Neural Networks,” IEEE International Conference on Grey Systems and Intelligent Services (GSIS), September 2011, pp. 807 -810.

[3] YJ Chen, WP du Plessis, “Neural Network Implementation on FPGA”, IEEE Africon Conference 6th, Africa October 2002, vol 1.,pp 337- 342.

[4] Nazeih M. Botros and M. Abdul Aziz, “Hardware Implementation of Artificial Neural Network using Field Programmable Arrays,” IEEE Conference on Industrial Electronics, December 1994, Vol 41. No.6, pp. 665-667.

[5] S. Hariprasath and T.N. Prabakar, “FPGA Implementation of Multilayer Feed forward Neural Network Architecture using VHDL”, IEEE International Conference on Computing, Communication and Applications, Feb 2012, pp. 1-6.

[6] John K. Kruschke and Javier R. Movellan, “Benefits of Gain: Speeded Learning and Minimal Hidden layers in Back Propagation Networks,” IEEE Transaction on Systems, Man, and Cybernetics, January/ February 1991,vol.21, No.1,pp.273- 279.

[7] David E.Rumelhart, Bernard Widrow and Micheal A. Lehr, "The Basic Ideas in Neural Networks," Communication of the ACM, vol.37,No.3, March 1994.

[8] Feldman ,J.A., M.A. Fanty, and N.Goddard.1988. "Computing with structured Neural Networks," IEEE Computer (March):91-103

[9] Hopfield, J.J., and D.W. Tank. 1986 "Computing with Neural Circuits: A Model," Science 233:625-633.

[10] Lippmann, R.P. 1987. "An Introduction to Computing with Neural Nets," IEEE Magazine on Acoustics, Signal and Speech Processing (April):4-22

[11] Jacobs, R.A. 1988. "Increased Rates of Convergence Through Learning Rate Adaption," Neural Networks 1:295-307

[12] Hripcsak, G. 1988 "Problem Solving Using Neural Networks," San Diego, Calif.: SAIC Communications

[13] Mirchandini, G., and W. Cao.1989. "On Hidden Nodes in Neural Nets", IEEE Trans. Circuits and Systems 36(5):661-664

[14] Pao, Y.H. 1989. "Adaptive Pattern Recognition and Neural Networks." Reading Mass; Addision-Wesley Publishing Co.

[15] Cybenko, G. 1990. "Complexity Theory of Neural Networks and Classification Problems," in Neural Networks EURASIP Workshop Proc., ed. L.B. Almeida, C.J. Wellekens. Sesimbra, Portugal, February 1990, pp. 24-44

[16] Funanashi, K.I. 1989. "On the Approximate Realization of Continuous Mappings by Neural Networks," Neural Networks 2:183-192.

[17] Karin, E.D. 1990. "A Simple Procedure for Pruning Back-Propagation Trained Neural Networks," IEEE Trans. On Neural Networks 1(2): 239-242.

[18] White, H. 1989. "Learning in Artificial Neural Networks: A Statistical Perspective," Neural Computation 1(4);425-469

[19] Wieland, A., and R. Leighton. 1988. "Geometric Analysis of Neural Networks Capabilities," MP-88 W 00022. McLean, Va.: Mitre Corporation.

[20] Poggio, T. and F. Girosi. 1990. "Networks for Approximation and Learning" Proc. IEEE 78(9): 1481-1497.