IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 10, Issue 3, Ver. III (May - Jun.2015), PP 45-54
www.iosrjournals.org

FPGA Implementation of a Trained Neural Network

Seema Singh[1], Shreyashree Sanjeevi[2], Suma V[2], Akhil Talashi [2]
[1]*(Associate Professor, Dept. of Electronics and Communication Engineering,*
B.M.S Institute of Technology and Management, India)
[2]*(Dept. of Electronics and Communication Engineering, B.M.S Institute of Technology and Management, India)*

Abstract: *This work presents the implementation of a trained Artificial Neural Network (ANN) for a certain application. A Multi Layer Perceptron (MLP) has been synthesized and implemented on Spartan3 FPGA. The weight matrices and bias has been provided by separate simulation software like MATLAB/Simulink. The implemented network has been verified in Xilinx ISE using Verilog programming language. The device utilization summary illustrates that the implemented perceptron utilizes few slices on FPGA which makes it suitable for large scale implementation. The implementation of FPGA based neural network is verified for a specific application.*
Keywords- *Artificial Neural Network, FPGA implementation, Multilayer Perceptron(MLP), Verilog.*

I. Introduction

1.1. Overview of ANN Structure

An artificial neural network is an interconnected group of nodes which perform functions collectively and in parallel, akin to the vast network of neurons in a human brain [1],[2],[3]. It consists of a number of input vectors, followed by multipliers which are often called weights followed by a summer and a threshold function. The input signals are summed and passed through a threshold function. If the result of the summation operation exceeds the threshold value, the neuron fires i.e. output of the threshold function will be positive else, it will give a negative value.

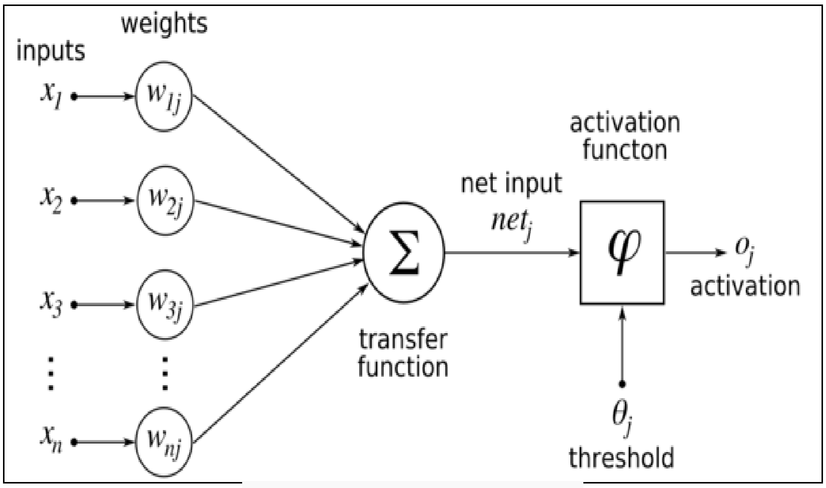

Fig. 1 A single neuron structure

An ANN is typically defined by three types of parameters [4]:

1) The interconnection pattern between different layers of neurons. It deals with number of inputs, outputs and number of hidden layers in the structure. Here, a 3-2-1 Feedforward neural network is considered.
2) The learning process for updating the weights of the interconnections. Here the weight matrices and bias have been provided by separate simulation software like MATLAB/Simulink.
3) The activation function that converts a neuron's weighted input to its output activation. Here, a sigmoid activation function is used between the input and the hidden layers and a linear activation function is used between the hidden and the output layers.

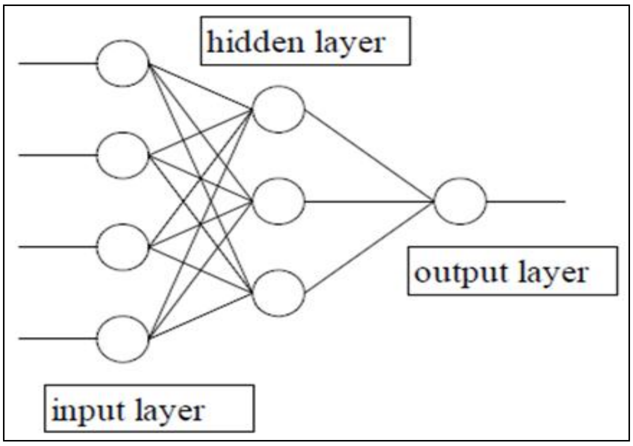

Fig. 2 A MLP of 4-3-1 structure

The Feed forward neural network is the simplest type of ANN devised. In this type of network, the information moves only in the forward direction. The data is delivered to the output nodes from the input nodes after going through the hidden nodes. There are no cycles or loops in this network.

1.2. Hardware requirement of ANN

A large variety of hardware has been designed to exploit the inherent parallelism of the neural network models [5, 6]. With the increasing demands of machine learning and parallel computing, ANN plays a pivotal role in the field of Artificial Intelligence. Today, neural networks are used in various applications like Stock market prediction, process and quality control in industry [9, 10] and medical diagnosis [11]. Most of these applications are used in the simulation mode during the research phase. However, the practical usage of neural networks in the market requires the associated hardware. Hence, the need for hardware implementation of a trained neural network for a given application arises. The hardware chosen is application dependent. The hardware used in this paper is FPGA.

1.3. FPGA vs. DSP

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a designer after manufacturing [3],[5]. FPGA is a device which is used to simulate and test IC designs. They are programmed by using Hardware Description Languages (VHDL/Verilog). The programming language used here is Verilog. A wide range of logic gates- upto a few millions can be applied on a FPGA. It is reconfigurable and has a short design cycle. The biggest advantage that FPGA has over other processors is that it supports parallel computing which is very much required while implementing a neural network.

For Multimedia gadgets that require higher performance and higher algorithm complexity, FPGA has emerged over DSP's. FPGA is a prototype for an IC that has to be manufactured whereas a DSP processor is an integral part of a bigger circuit.

II. Data Representation

Several number representation formats like floating point, fixed point etc. exist. In this paper we use the fixed point format for all the inputs, weights and activation function. Fixed point format implies that the number of decimal places after the point is fixed for all the values used. Although floating point format is more desirable, the hardware complexity increases and hence we are bound to use fixed point format.

As understood from reference [6], precision is dependent on the number of bits used for the representation; as the number of bits increase the resources required increases. The inputs and weights are normalized before being applied to the network. A 10-bit representation is used giving a precision of 1/1024. The format used for the 10-bit representation [7] is as given below: **Table 1** shows various numbers and their 10-bit representation [2].

1 bit SIGN	1 bit INTEGER PART	8 bits FRACTIONAL PART

Table 1: Data representation

Number	Data Representation
-2	1000000000
-1	1100000000
0	0000000000
0.9436	0011110001
0.45	0001110011
1	0100000000
1.99	0111111101

The weight matrices provided by MATLAB/Simulink for the hidden layer is of the format

$$\begin{bmatrix} w11 & w12 & w13 \\ w21 & w22 & w23 \end{bmatrix}$$

Weight matrix format for the output layer

$$[w11 \quad w12]$$

III. Sub Modules Of FPGA Implementation

3.1 Multiply and Accumulate (MAC)

Mac- multiply and accumulate is the module which is used to obtain the weighted sum given by the equation.

$$\sum_1^3 \sum_1^2 x(i)w(ji)$$

The inputs, a, b are of each 10 bits and the output, c is of 20 bits. The data representation as mentioned above is followed. Truncation procedure, which is explained in the next section, is used here to arrive at the answers.

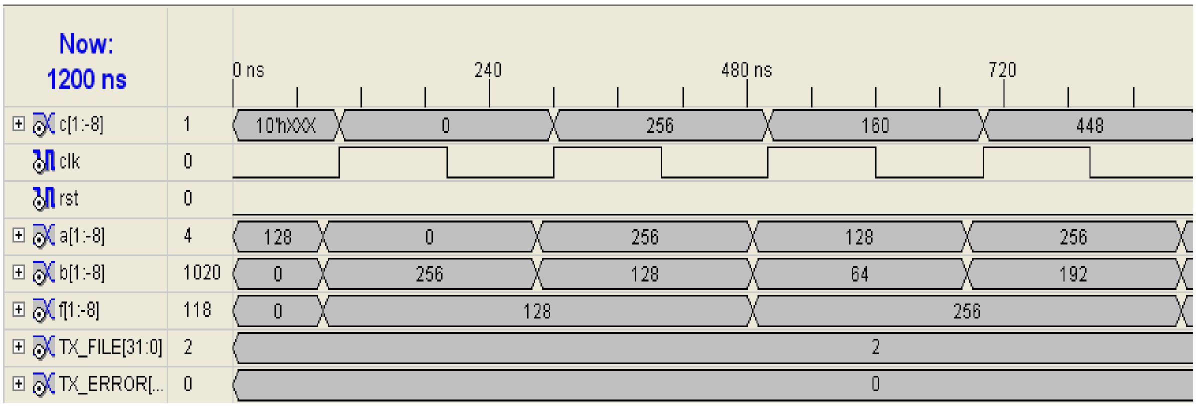

Fig. 3 Simulation for MAC module

3.2 Truncation

The various calculations in the network produce results which are of size greater than what the network has been designed for. This calls for a module that can truncate the results to the desired or suitable size. The truncation is necessary to ensure uniformity throughout the network and to allow for ease of programming. Truncation does not result in loss of accuracy.

The simulation shows the truncation of a 20-bit input into a 10-bit output:

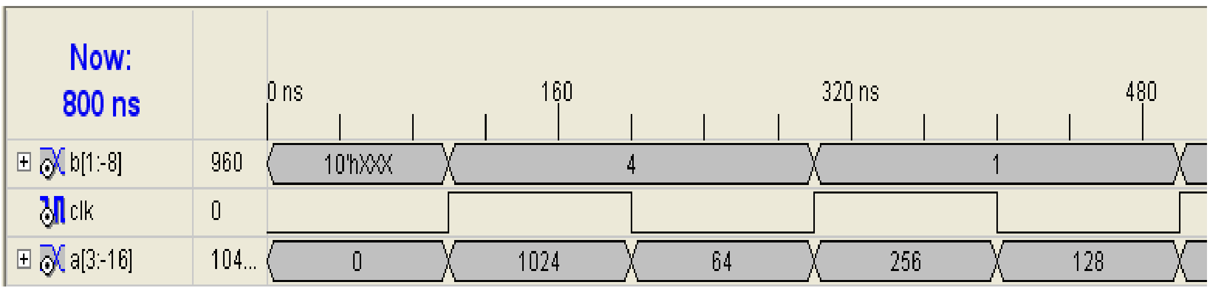

Fig. 4 Simulation for truncation module

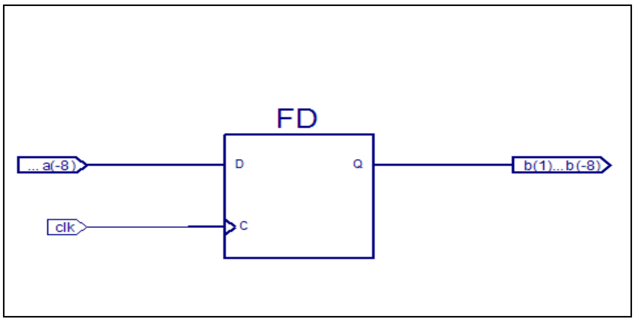

Fig. 5 RTL schematic of truncation module

3.3 Sigmoid transfer function

In order for the neural network to function akin to the human brain, non-linearity needs to be introduced into the network. This is achieved by using the sigmoid activation function. In this paper, implementation of the sigmoid activation function is in the form of a LUT, Reference [2].

An approximation function is widely used to implement the sigmoid curve. However, it is avoided here since this method compromises on accuracy.

To write the LUT, the values for the sigmoid function were first determined using the TANSIG MATLAB function for the range -2 to +2 with a step value of 0.01. These values are in the decimal fixed point format. They are converted to their binary equivalent. These binary equivalents are eventually converted back to decimal integers. This process is carried out for both the input and output and the entire table is constructed, Reference [2]. The representation for the one input sample and its corresponding output is shown below.

INPUT: 0.55 → 0010001100→ 143
OUTPUT: 0.5005 → 0010000000 → 128

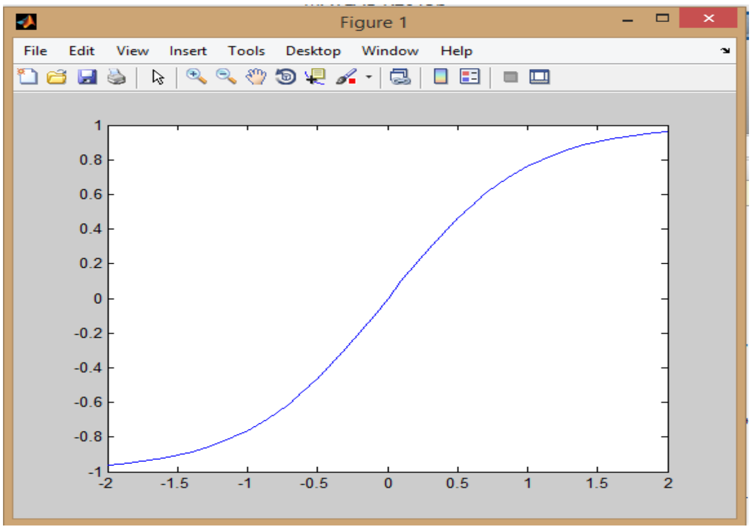

Fig. 6. Matlab generated plot for the chosen range.

3.4 Linear transfer function

Several activation functions may be used at the output layer. The most common being the linear function. This function is implemented by multiplying a "slope" value with the weighted sum from the previous layer. The slope determines how the final output is related to the previous layer output.

In the form of an equation, it is the straight line equation that is used where the value of the intercept is assumed to be zero. The value of slope is determined by trial and error method to suit a given application.

IV. Integrated Network

The network under consideration in this paper is a feedforward 3-2-1 network with the sigmoid activation function at the hidden layer and the linear activation function at the output layer.

All the modules specified in the previous section are integrated in order to form the complete network. The inputs, weights, intermediate values and the output values use the 10-bit data representation.

STEPS:

i) The operation begins with the MAC operation at the hidden layer; the result of which is 21-bit. Thus this result is first truncated and then used for further operations in the network.
ii) The next step is to pass the truncated result as input to the sigmoid function which is implemented as a LUT.
iii) The output from the LUT (10-bit) serves as input to the second layer.
iv) The MAC followed by the linear activation function is applied at the output layer.
v) The output from this layer is the final output of the network which is in-turn used to make a decision about any said application.

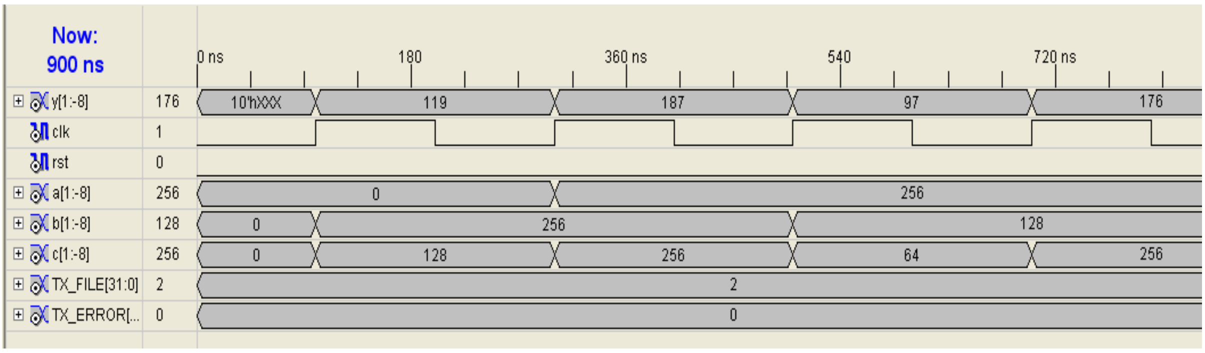

Fig. 7 Full network simulation with only positive inputs

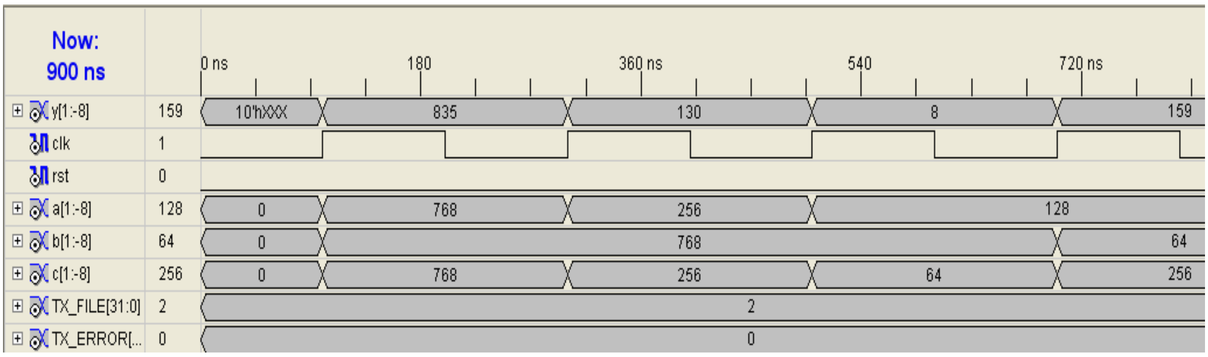

Fig. 8 Integrated network simulation with negative and positive inputs

V. Application

The FPGA based trained neural network is tested with an application of fault identification system for transmission lines. Reference [8] implemented fault identification system on a DSP processor of TMS320C6713. The weight, biases and inputs were simplified to whole numbers for DSP implementation [8]. The same application is tested with FPGA based hardware implementation. The weights and biases are kept at its actual fixed point values without simplification. The fault identification system was successfully synthesized, implemented and tested with the FPGA based neural network developed in this paper. Similarly the FPGA implementation is applied to the neural network based applications in Flight Control system used in references [9] and [10].

VI. Results and Discussion

The experimental set up consisting of Spartan 3 kit, FRC modules, dip switches and LED is shown in Fig. 9. One example is taken as case study to discuss the flow of data from input to output of neural network structure. The intermediate results are shown with the help of each module. Input value, weights at hidden layer and output layer is assumed along with the slope for linear activation function. Manual calculation is used to verify the results of implemented neural network.

Input vector, x= [1 1 1]

Hidden layer weight matrix, w23= $\begin{bmatrix} 0.2 & 0.1 & 0.75 \\ 0.15 & 0.25 & 0.65 \end{bmatrix}$

Output layer weight matrix, v12= [0.25 0.8]

Linear activation function slope =0.7

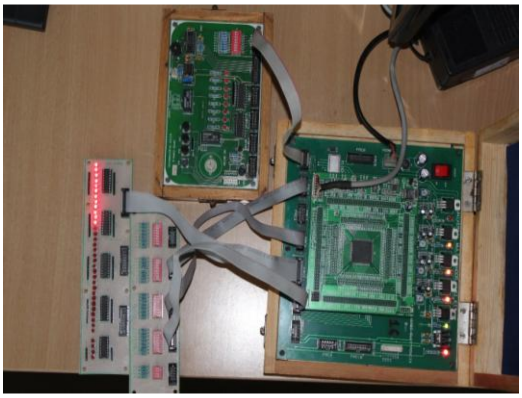

Fig. 9 Experimental setup

Manual calculation:

MAC and truncation at the hidden layer:

$$\begin{bmatrix} 0.2 & 0.1 & 0.75 \\ 0.15 & 0.25 & 0.65 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.05 \\ 1.05 \end{bmatrix}$$

Sigmoid activation function:

$$tansig \begin{bmatrix} 1.05 \\ 1.05 \end{bmatrix} = \begin{bmatrix} 0.7818 \\ 0.7818 \end{bmatrix}$$

MAC and truncation at output layer:

$$\begin{bmatrix} 0.25 & 0.8 \end{bmatrix} \begin{bmatrix} 0.7818 \\ 0.7818 \end{bmatrix} = \begin{bmatrix} 0.7782 \end{bmatrix}$$

Linear activation function:

$$[0.7782][0.7]=[0.5447]$$

Step 1: MAC at hidden layer: Multiply and accumulate: the input matrix is multiplied with weight matrix w23 as per the matrix multiplication rule. Output obtained is $\begin{bmatrix} 68864 \\ 68864 \end{bmatrix}$ which when converted to binary is a 21-bit value as shown in Fig. 10.

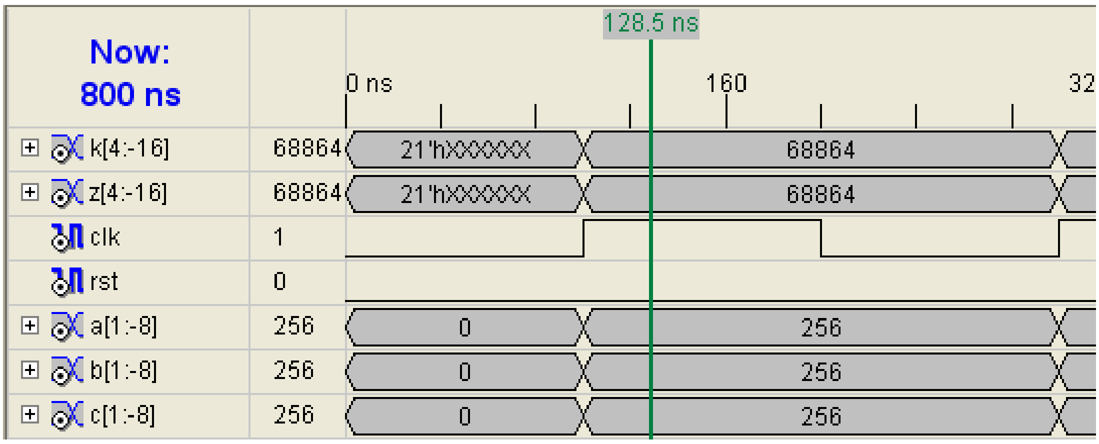

Fig. 10 MAC simulation at the hidden layer

Step 2: Truncation: The 21-bit outputs obtained in Step 1 are truncated to 10-bit for uniformity throughout the network shown in Fig. 11.. The 10-bit outputs are $\begin{bmatrix} 269 \\ 269 \end{bmatrix}$ as seen in figure 11. 269 is approximately equal to the theoretical value which is **1.05**.

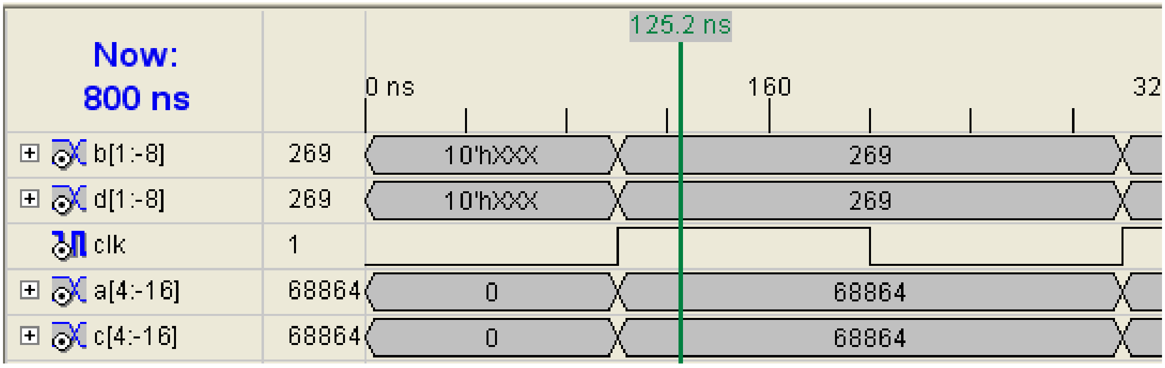

Fig. 11 Truncation simulation

Step 3: Sigmoid Activation Function: The sigmoid function is implemented in the proposed design in the form of a LUT. The input and output of the LUT are 10-bit values. The output, 269 of Step 2 serves as the input to the LUT. The output of the LUT is 200 which is equal to **0.781**. Two such outputs are obtained which are inputs to each hidden layer node $\begin{bmatrix}200\\200\end{bmatrix}$.

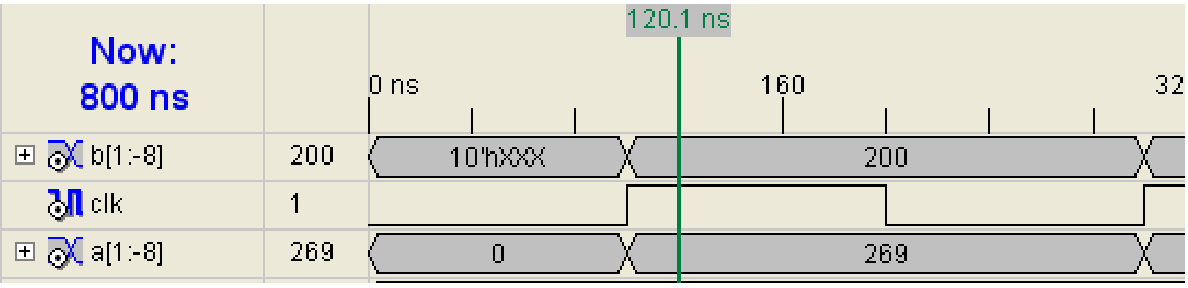

Fig. 12 Sigmoid activation function simulation

Step 4: Mac at The Output Layer: The Step 4 output is the input to a MAC function. This output is truncated as explained before. The result is [179] approximately equal to **0.8198**.

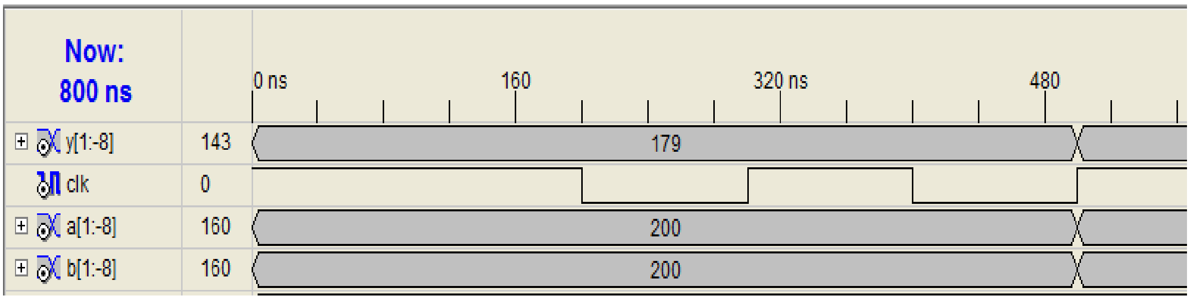

Fig. 13 MAC simulation at the output layer

Step 5: Linear Activation Function: the Step 4 output is multiplied with the slope value giving an output of [146] which is approximately equal to **0.574**. The simulation output shown in Fig. 14 includes truncation operation also.

Step 6: Thus the integrated network's output is [146], approximately equal to **0.574**.

Final theoretical value: 146 and Final practical value: 146.

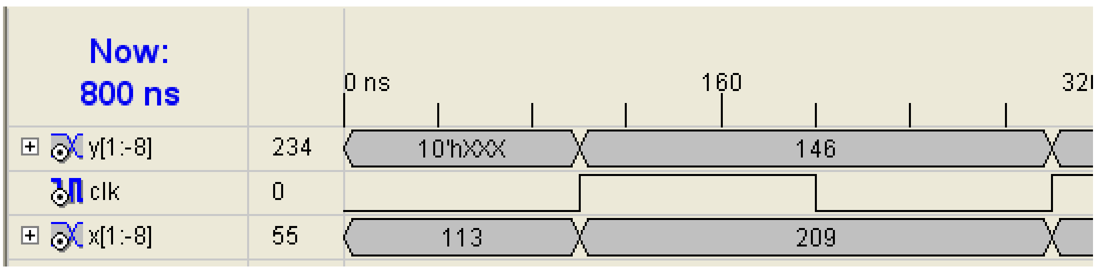

Fig. 14 Linear activation function simulation

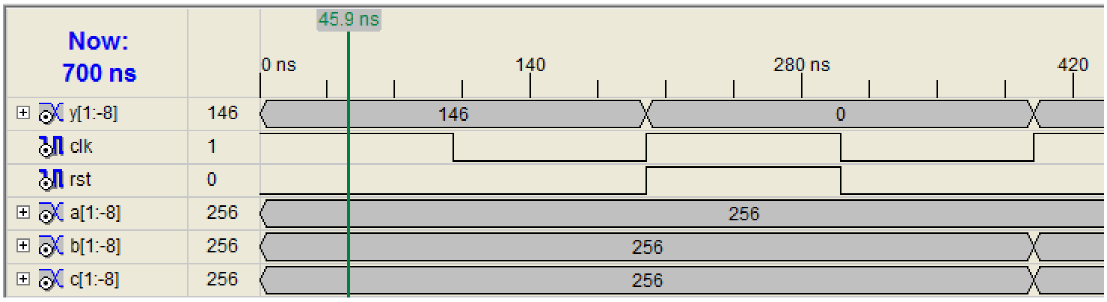

Fig. 15 Integrated network simulation

The resource utilization of Spartan 3, XC3S400-5TQ144 for the sub modules as well as the integrated network is shown in the design summaries.

DESIGN SUMMARY OF MAC MODULE

SUMA Project Status			
Project File:	suma.ise	**Current State:**	Programming File Generated
Module Name:	mac	**• Errors:**	
Target Device:	xc3s400-5tq144	**• Warnings:**	
Product Version:	ISE, 8.1i	**• Updated:**	Sat Jun 6 12:09:33 2015

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	103	7,168	1%	
Logic Distribution				
Number of occupied Slices	60	3,584	1%	
Number of Slices containing only related logic	60	60	100%	
Number of Slices containing unrelated logic	0	60	0%	
Total Number 4 input LUTs	113	7,168	1%	
Number used as logic	103			
Number used as a route-thru	10			
Number of bonded IOBs	42	97	43%	
IOB Flip Flops	10			
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	1,165			
Additional JTAG gate count for IOBs	2,016			

Fig. 16 Design summary of MAC module

Thus the percentage utilization of a single MAC operation is **0.29125%**

DESIGN SUMMARY OF SIGMOID ACTIVATION FUNCTION MODULE

SUMA Project Status			
Project File:	suma.ise	**Current State:**	Placed and Routed
Module Name:	sigmoid	**• Errors:**	No Errors
Target Device:	xc3s400-5tq144	**• Warnings:**	No Warnings
Product Version:	ISE, 8.1i	**• Updated:**	Sat Jun 6 13:00:53 2015

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s**
Number of 4 input LUTs	336	7,168	4%	
Logic Distribution				
Number of occupied Slices	212	3,584	5%	
Number of Slices containing only related logic	212	212	100%	
Number of Slices containing unrelated logic	0	212	0%	
Total Number of 4 input LUTs	336	7,168	4%	
Number of bonded IOBs	21	97	21%	
IOB Flip Flops	10			
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	2,225			

Fig. 17 Design summary of sigmoid activation function module

Thus the percentage utilization of a single sigmoid activation function operation is **0.556%**

DESIGN SUMMARY OF LINEAR ACTIVATION FUNCTION MODULE

SUMA Project Status			
Project File:	suma.ise	**Current State:**	Placed and Route
Module Name:	linear	**• Errors:**	No Errors
Target Device:	xc3s400-5tq144	**• Warnings:**	No Warnings
Product Version:	ISE, 8.1i	**• Updated:**	Sat Jun 6 12:54:2

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of 4 input LUTs	50	7,168	1%
Logic Distribution			
Number of occupied Slices	28	3,584	1%
Number of Slices containing only related logic	28	28	100%
Number of Slices containing unrelated logic	0	28	0%
Total Number 4 input LUTs	53	7,168	1%
Number used as logic	50		
Number used as a route-thru	3		
Number of bonded IOBs	21	97	21%
IOB Flip Flops	10		
Number of GCLKs	1	8	12%
Total equivalent gate count for design	527		

Fig. 18 Design summary of linear activation function module

Thus the percentage utilization of a single linear operation is **0.132%**

DESIGN SUMMARY OF INTEGRATED NETWORK

SUMA Project Status			
Project File:	suma.ise	**Current State:**	Placed and Routed
Module Name:	full_network	**• Errors:**	No Errors
Target Device:	xc3s400-5tq144	**• Warnings:**	77 Warnings
Product Version:	ISE, 8.1i	**• Updated:**	Sat Jun 6 15:01:55 2015

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	1,125	7,168	15%	
Logic Distribution				
Number of occupied Slices	662	3,584	18%	
Number of Slices containing only related logic	662	662	100%	
Number of Slices containing unrelated logic	0	662	0%	
Total Number 4 input LUTs	1,147	7,168	16%	
Number used as logic	1,125			
Number used as a route-thru	22			
Number of bonded IOBs	42	97	43%	
IOB Flip Flops	43			
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	8,863			

Fig. 19 Design summary of integrated network

Thus the percentage utilization of the integrated network is **2.215%**

VII. CONCLUSION

The design of a 3-2-1 neural network structure is proposed. This network is implemented on Xilinx Spartan3 family FPGA using Verilog. The implemented network is tested with an application. The network is broken down into functional modules and the result for each module is presented for a clear understanding of the working of the network. The device utilization for a single MAC operation is 0.29125%, a single sigmoid operation is 0.55%, and single linear activation function operation is 0.132%. The modules are then integrated to form the network under consideration. The integrated neural network consists of 8 multiplier operations, 3 accumulation operation, 3 truncations, 2 sigmoid activation function operation and 2 linear activation function operation. The device utilization of the integrated 3-2-1 feedforward network is 2.215%. A look-up table is used to implement the sigmoid activation function. The Tansig function values are generated using Matlab. This approach is used to maintain accuracy of the values. Although this method of implementing the sigmoid function as LUT consumes more resources and hence more area, the output of the network is accurate. The above network can be expanded further to implement more complex applications involving bigger structure of neural network.

Acknowledgement

The authors would like to thank the B.M.S. Management for providing us with great infrastructure and a conducive atmosphere to work in. A special thanks to Dr. Hariprasad, H.O.D ECE Dept. of BMS Institute of Technology and Management. The authors would also like to extend their heartfelt thanks to Dr. S.L. Pinjare, HOD, ECE Dept. of Reva Institute of Technology and Management for his timely guidance.

References

[1] Makwana, Hardik H, Dharmesh J. Shah, Priyesh P. Gandhi, "FPGA Implementation of Artificial Neural Network.", International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 1, Jan, 2013, pp 672-679 [?], ISSN 2250-2459

[2] Pinjare S. L., Arun Kumar, "Implementation of neural network back propagation training algorithm on FPGA.", International Journal of Computer Applications, Volume 52, Issue 6, August, 2012, pp 1-7, ISSN 0975 – 8887[?]

[3] Lakshmi, K., M. Subadra, "A survey on FPGA based MLP realization for on-chip learning.", International Journal of Scientific & Engineering Research, Volume 4, Issue 1, Jan, 2013 , pp 1-9, ISSN 2229-5518

[4] Gomperts, Alexander, Abhisek Ukil, Franz Zurfluh, "Development and implementation of parameterized FPGA-based general purpose neural networks for online applications.", Industrial Informatics, IEEE Transactions, Volume 7, Issue 1, Feb, 2011, pp 78-89, ISSN 1551-3203

[5] R.Gadea, J.Cerda, F.Ballester, A.Mocholi, “Artificial Neural Network Implementation on a single FPGA of a Pipelined on-line Backpropagation”, IEEE Conference Publication, pp. 225-230, 20-22 Sept, 2000.

[6] Savran, Aydoğan, Serkan Ünsal, "Hardware Implementation of a Feed forward Neural Network Using FPGAs.", The third International Conference on Electrical and Electronics Engineering (ELECO 2003) [?], Dec, 2003, pp 3-7

[7] Panicker, Manish, and C. Babu. "Efficient FPGA Implementation of Sigmoid and Bipolar Sigmoid Activation Functions for Multilayer Perceptrons." IOSR Journal of Engineering (IOSRJEN) (2012): 1352-1356.

[8] Seema Singh, Mamatha K R, Thejaswini S, “Intelligent Fault Identification System for Transmission Lines Using Artificial Neural Network”, IOSR Journal of Computer Engineering (IOSR-JCE), Volume 16, Issue 1, Jan, 2014, pp 23-31, ISSN 2278-0661

[9] Seema Singh and T. V. Rama Murthy, “Neural Network based Sensor Fault Detection for Flight Control Systems”, International Journal of Computer Applications (IJCA), Vol 59, No 13, Dec, 2012, pp 1-8.

[10] Seema Singh and T. V. Rama Murthy, “Neural Network based Sensor Fault Accommodation in Flight Control System, Journal of Intelligent systems (JISYS), De Gruyter, Vol 22, Issue 3, September, 2013, pp 317-333.

[11] Seema Singh, Harini J and Surabhi B R, “A Novel Neural Network Based Automated System for Diagnosis of Breast Cancer from Real Time Biopsy Slides”IEEE Conference proceedings on International Conference on Circuits, Communication, Control and Computing (I4C 2014), November, 2014, pp 50-53