A Survey on FPGA based MLP Realization for On-chip Learning

K. Packia Lakshmi, Dr. M. Subadra

Abstract— The objective of this work is to review the development steps of implementation of MLP neural network in FPGA device, during the recent years. Each development to implement the neural architecture and activation function effectively were studied to take the work one step forward to implement a cost effective, compact and effective on-chip trained neural network. Finally the off-chip trained MLP for XOR problem is implemented on different hardware platform to show the importance of hardware platform selection for on-chip training.

Index Terms— ANN, MLP, FPGA, off-chip training, on-chip training, sigmoid function, XOR problem

1 Introduction

ARITIFICIAL Neural network is a field of artificial intelligence (AI) used to models the human brain activities[1]. The designed analytical model is used to solve the given tasks based on previous experience with reasonable accuracy, at reasonable cost and in a reasonable amount of time [2].

Arificial neural network (ANN) provides efficient parallel processing. Three important computational characteristics typically associated with ANNs are parallelism, modularity and dynamic adaptation [3]. A large variety of hardware has been designed to exploit the inherent parallelism of the neural network models [4].

The important characteristics [5] of the network depend on

a. Structure of the network
b. The activation functions of the processing elements
c. Learning mechanism of the network

A vast majority of neural networks are still implemented on software on sequential machines. It is used for investing the capability of network model and design new algorithms [6]. Although this is not necessarily always a severe limitation, there is much to be gained from directly implementing neural networks in hardware but without excessive costs [7], [8], [9]. Neural network hardware has been found to be useful in some specialized areas, such as image processing, speech synthesis and analysis, pattern recognition, high energy physics and so on[6].

Neural network hardware is usually defined as those devices designed to implement neural architectures and learning algorithm. Especially those devices take advantage of the parallel nature inherent to ANNs [8], [9].

The main focus of this paper is to analyze the processing steps of Field Programmable Gate Array (FPGA) implementation of neural network from the previous work and discuss the future work proposed on FPGA implementation of neural network for on-chip learning.

• K. Packia Lakshmi is currently pursuing post graduation degree program in Applied Electronics in Einstein College of Engineering, Tirunelveli. E-mail: krishbagya@gmail.com
• Dr.M.Subadra is currently working as Professor& Head of Electronics and Communication Department of Einstein College of Engineering,Tirunelveli E-mail: subadra_m@yahoo.com

2 Need for FPGA

ANN may realized by using analog systems or digital systems. Existing systems available may have it own advantages and disadvantages. Some of the existing platforms available for hardware implementation of ANN are summarized below,

2.1 Existing Devices

a. Genereal purpose parallel computer

Fine-grain parallel implementations on massively parallel computers (either SIMD or MIMD) suffer from the connectivity of standard neural models. It may result in costly information exchanges. Furthermore, massively parallel computers are expensive resources and they cannot be used in embedded applications.

b. DSP chips

DSP chips are a kind of processors: they perform given task in a serial fahion. Calculations in ANNs can be performed in parallel, but this parallelism cannot be exploited in DSP. So DSP chips are not suted for ANN implementation.

c. Dedicated parallel computers

Neuro computers are parallel systems dedicated to neural computing. They are based on computing devices such as DSPs (digital signal processors). They are usually suffers from their cost and their development time. So they are rapidly become out-of-date, compared to the most recent sequential processors.

d. Analog ASICs

They are very fast, dense and low power. But analog ASICs introduce specific problems such as precision, data storage and robustness. On-chip learning is difficult. It is an expensive and nonflexible solution.

e. Digital ASICs

Compared to analog chips digital ASICs provide more accuracy. Digital ASICs are more robust. They can handle any standard neural computation. Yet their design requires a strong effort to obtain working chips and it is very expensive when only a few chips are needed.

ASIC implementations have a performance advantage over the other hardware choices.But once an ANN is implemented as an ASIC; it is fixed, and the network configuration cannot be modified. This is a major drawback of ASIC implementation.

2.2 Advantage of FPGA

FPGA based reconfigurable computing architectures are suitable for hardware implementation of neural networks [2], [6]. Some of the solutions provided by FPGA for ANN implementation are listed below,

a. Reprogrammable FPGAs permit prototyping

Moreover a good architecture that has been designed and implemented may be replaced later by a better one without having to design a new chip.

b. On-chip learning

In a reconfigurable FPGA, on-chip learning may be performed prior to a specific optimized implementation of the learned neural network on the same chip.

c. Embedded application

FPGAs may be used for embedded applications when the robustness and the simplicity of neural computations are most needed, even for low-scale productions.

d. Reconfigurability

FPGA based implementations major advantage is that it may be mapped onto new improved FPGAs. FPGA speeds and areas approximately double each year. Even large neural networks may soon be implemented on single FPGAs, provided that the implementation method is scalable enough.

FPGA based reconfigurable computing architectures are well suited to implement ANNs as one can exploit concurrency and rapidly reconfigure to adapt the weights and topologies of an ANN [3], [4].

In existing methods ANNs are realized on FPGA. They are mainly static implementations for specific offline applications without learning capability. In this type of implementations, the purpose of using a FPGA is generally to gain performance advantages through dedicated hardware and parallelism [10]. Interesting FPGA implementations schemes, specially using XILINX FPGAs, have been reported [7].

3 ANN

Neural network is one of the soft computing tools. Artificial neural network are composed of many simple processing unit called neurons [5]. The connection between inputs $[p_i, p_2, p_3, \dots p_n]$ and neurons or between two neurons is called the weight $[w_i, w_2, w_3, \dots w_n]$. The goal of the networks is to learn some association between input andoutput patterns. Basic structure of a neuron with 'n' input is shown in figure.1

Function of the neuron is described by the following equations

$$x = b + \sum_{i=1}^{n} p_i \; w_i \tag{1}$$

and

$$y = f(x) \tag{2}$$

where,

x →weighted sum input
y →neuron output
b →input bias
f(x) → activation function

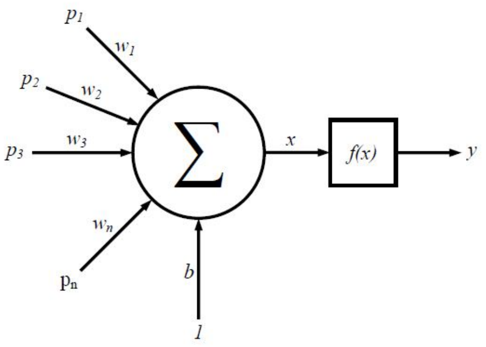

Fig. 1. Structure of a neuron (perceptron)

3.1 MLP-Network Structure

Multi Layer Perceptron (MLP) is an important fully connected feed forward artificial neural network model. It organizes their neurons into three layers. The first layer is called the input layer, the last one the output layer. The intermediate layers are called the hidden layers [11]. A simple MLP architecture is shown in figure 2.

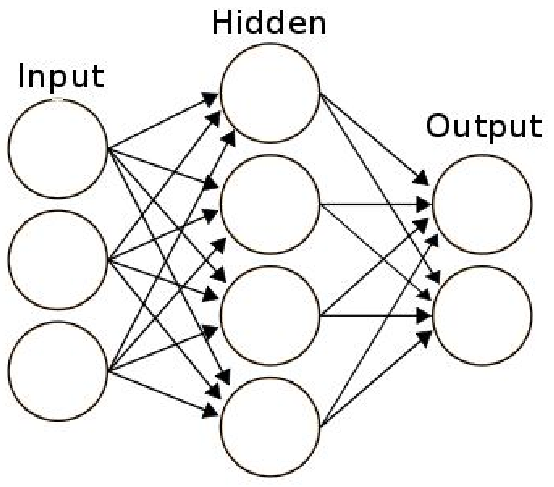

Fig. 2. Multi Layer perceptron architecture

3.2 Learning of MLP

To make decisions, the smart machines are trained on a set of training (learning) examples. Learning is usually accomplished by modification of the connection weights. The only information available to the network is the training data set. The number of neurons, their structure and the corresponding values of the weights are the subjects of learning procedure.

Each new smart machine should be able to learn the problem in its areas of operation. In machine language problem settings there is some unknown nonlinear dependency y= f(x) between some high-dimensional input vector x and scalar y. The only information available to the network is the training data sets.

Most popular algorithm used to train the MLP is back propagation algorithm. Reason for the popularity is its power and simplicity [12].

3.3 Activation function

Most of the artificial neuron models use the same way of action to produce the total input signal, but they are different in terms of how they produce an output response from this input. Artificial neurons use an activation function to compute their activation as a function of total input stimulus. Several different functions may be used as an activation function; the most distinguishing feature between existing neuron model is precisely which transfer function they employ [5]. Some common activation functions used in the neural network literature is listed out in the following Table 1.

TABLE 1
SOME COMMON ACTICATION FUNCTIONS

Name	Function
Binary Threshold	$y=1;x\geq0$ $y=0;x<0$
Bipolar Threshold	$y=1;x\geq0$ $y= -1;x<0$
Sigmoid	$y = 1/(1 + e^{-x})$
Linear	$y=x$
Saturating linear	$y=1;x\geq0$ $y=0;x<0$ $y=x;0\leq x\leq1$
Symmetric saturating linear	$y=1;x>1$ $y= -1;x< -1$ $y=x;-1\leq x\leq1$
Hyperbolic Tangent Sigmoid	$y = (e^{x} - e^{-x})/(e^{x} + e^{-x})$
Positive linear	$y=x;x\geq0$ $y=0;x<0$

Selection of activation function is depending on the application. For back propagation training algorithms the derivative of the activation function is needed [13]; therefore, the activation function selected must be differentiable. The logistic or sigmoid function is satisfies this requirement. This is the most commonly used soft-limiting activation function. Because it squashes the input range into 0 to 1 output range.

So hardware implementation of MLP needs the implementation of sigmoid activation function [14].

4 FPGA

Field Programmable Gate Arrays (FPGAs) is an efficient programmable device to replace the microprocessor yet faster to develop the dedicated chips. FPGAs are completely programmable after the product is shipped. It uses the high circuit densities in modern processes to construct ICs. FPGA chip consists of an array of logic cells surrounded by programmable routing sources.

FPGAs are the best choice for many low to medium volume custom logic applications. Compared to conventional ICs, FPGAs are slower and are less efficient in area and power. But the programming cost is zero. The cost per part is moderate to high. As initial manufacturing costs for custom integrated circuit increases FPGA costs decrease, FPGAs become attractive for more and lower to mid-volume, moderate power applications.

FPGAs are implemented with a flexible, regular and reprogrammable architecture of configurable logic blocks (CLBs), interconnected by versatile programmable routing channels and surrounded by a perimeter of programmable input/output blocks (IOBs). These devices are configured for a specific application by loading configuration data into internal static memory cells. The data stored in the memory cells determines the functionality of CLBs, IOBs and routing channels in the FPGA. The FPGAs can read its configuration data from an external serial PROM in master serial mode or an external device can write data into it in slave serial mode. FPGA consists of CLBs in an array of matrix form provide functional elements for logic implementation. Each CLB has look-up tables, flip flops and group of multiplexers [15].

5 OFF-CHIP LEARNING

Off-chip learning is a learning process, which is take place in PC. The sequentially executable software is used for learning in PCs. The architecture, weights of ANN is fixed after the completion of training. By using that MLP is implemented on hardware. The researchers use FPGA implementation of MLP network model in different application area. Each units of artificial neuron are implemented in hardware part separately. The main requirement of this work is to effectively implement the task with low cost, small module and low timing. The researchers analyzed each algorithms and activation function to improve the model performance. Mostly they implement the model using Xilinx and MATLAB /Simulink tool [16], [17], [18].

5.1 Neuron Implementation

Hardware realization of a Neural Network (NN), to a large extent depends on the efficient implementation of a single neuron [3].Different works has been done by the researchers to effectively implement the neuron on hardware environment.

Hoda and Mahmoud [1999] have implemented the RANN for chemical classification, in an electronic nose. The system consists of 4 neurons and 12 synapses. A neuron has been implemented on a tiny chip using 2.0µm n-well CMOS technology. The system test is under way for recognizing methanol, acetone, benzene and chloroform chemicals. The design may be altered for programmability to recognize other chemicals. They conclude that the approximation used for this circuit is acceptable with a maximum absolute error of 0.045[19].

Jang et al. [2002] tried to develop a new electronic instrument which resembles the human nose activity. The system using the IC chip, reduce system size and cost; this small module is used for many application. They use back propagation algorithm to measure gases and implement in CPLD of two hundred thousand gate level chips by VHDL language.40 pins of the chip were used and 67% of logic cells was used for the neural network implementation on the chip [20].

Tomoo et al. [2002] researched simplified multi-layer neural networks to equip them in one-chip FPGA. They reexamined neuron function, bits number of connection-weights, and learning methods; and proposed a “and/or”-neural network and the network were designed by using HDL. They conclude that the combination of the step/convex and linear functions ie., step/convex function is used for the hidden-layer and linear function on the output-layer is the best choice. [21].

Zheng [2006] et.al developed an intelligent controller based on ANN using FPGA for a four rotor helicopter to be capable of achieving vertical takeoff and to be able to sustain a specified attitude. They implemented the ANN on a Virtex-II Pro XC2VP30 FPGA from Xilinx then analyzed the simulation results to highlight the performance of the hardware. They also analyzed the practical limitation of off-line training. They specially focused on the on line training of neural network design. They concluded that the on-line training improves the system performance, integer data types was unsuitable for FPGA, compared to software implementation of NN in the microcontroller FPGA provided an increase in processing speed [22].

Himavathi et al. [2007] proposed a new FPGA implementation technique of the ANNs. Instead of realizing the whole network on FPGA, they only implemented the largest layer of the network and reused it for the other layers with the help of a controller block. They claimed that their technique is very effective in reducing the hardware cost of the ANNs with a moderate overhead on the speed [23].

Khalil [2008] proposed the hardware implementation using back propagation algorithm. They analyzed the result in terms of operating frequency and chip utilization. They provide the construction solution for implementation of neural network using FPGAs. They design and implement single neurons into a multilayer forward BP neural network. Finally they conclude that FPGAs constitute a very powerful option for implementing ANNs. Since the designer can really exploit the ANNs parallel processing capabilities [24].

Alin et al. [2009] proposed a method to design neural network by means of predefined block systems created in system generator environment and it increased the possibility to create a higher level design tools used to implement neural network in logic circuits [14].

Khodja et al. [2010] realized ANN on a FPGA board to contribute in the hardware integration solo in the areas such as monitoring, diagnosis, maintenance and control of power system as well as industrial process. In this work, they proposed a simple algorithm for the implementation of the ANN. The proposed hardware synthesis algorithm is performed by the system generator. They schedule the ANN on the system generator. The system generator generates VHDL code. They also use MATLAB software for machine learning to obtain a smallest squared error for 202 inputs [25].

5.2 Sigmoid Function Implementation

Networks performance and precision are depend on efficient implementation of activation function on FPGA. The hardware implementation of sigmoid activation is a very important part of hardware implementation of ANN [26]. But direct implementation of sigmoid activation function on FPGA is difficult due to its division and exponential function [10], [24]. Both operations require an inordinate amount of time and hardware resource to compute [10]. So some approximation methods are defined by researchers to find the better method to implement the sigmoid function on hardware. More accurate approximations will result in faster; better convergence, hence more accurate results and those approximation methods are valid for a variety of other sigmoid function. There has been significant amount of research where take place in hardware implementation of sigmoid function [27].

Some approximation methods proposed by researchers to implement sigmoid function on FPGA are Uniform Lookup table methods (LUT), linear approximation methods, Piece wise linear approximation method (PWL), PLAN approximation, A-law approximation, Allippi and Storti-Gajani approximation, Piecewise second-order approximation and Lookup table method with linear interpolation method.

a. LUT Method

This method works fast compared to piece-wise linear approximation method, though the memory requirement is high. It stores output for each input address so the memory requirement is high [20]. If there is not much concern about memory mean this method is preferred. The timing control is determined by the way of look up address generating [26].

Xiaobin et al. [2003] proposed a LUT based sigmoid function implementation and they proved that the suitable for both hidden and output layer. The performance of sigmoid was similar to the ANN which implement sigmoid with floating point data [26].

Savron et al. [2006] implemented the activation function using LUT method. They may need two RAM blocks for implementation [9].

b. Other Approximation Method

Bieu et al. [1994] proposed sum of steps approximation method to compute the sigmoid non-linearity and its derivative in hardware [28].

Basterrextea et al. [2001] presented a recursive algorithm for approximating the sigmoid function [29].

Tomoo et al. [2002] analyzed the sigmoid, linear, sine, quadratic, double / single-bendy and step / convex functions on the points of potential surface among learning points, calculation speed and probability of learning convergence [21].

Jang et al. [2002] used discontinuity linear function to implement the sigmoid function for their work [20]. Taylor series expansion method gave a better performance / price tradeoff over look up tables [25].

Tommiska [2003] proposed a new approach called as combinational approximation. He concluded that the SIG-sigmoid achieved better performance, by comparing it with the PWL and second order approximation in terms of speed, required area resources and accuracy measured by average and maximum error [30].

Khalil et al. [2008] suggested the second order approximation method to implement the activation function. They compared the real and hardware approximated activation function. Finally they concluded that this hardware approximated activation function was implemented directly using digital techniques [24].

Alin et al. [2009] had done the Hardware implementations of the sigmoid approximation in system generator, part of the MATLAB/ Simulink environment. They also analyzed the hardware resource consumption and generated errors for different FPGA implementation of 5 approximations proposed in literature. Then they conclude that the best approximation method is the PLAN function in the case in which the number of the artificial neurons hardware implemented that use sigmoid function as fire function is large than the number of the BRAM blocks available in the FPGA circuit. If the number of artificial neuron is lower than the total BRAM blocks available in the FPGA circuit they suggest look up table method to approximate the sigmoid function [14].

Khodja et al. [2010] proposed the sigmoid function implementation through the use of Taylor series. The sigmoid function was approximated in polynomial form. Then implement it by using Xilinx library [25].

Gomperts et al. [2011] combined the Look-up table method with the interpolation method to address the reduced accuracy of LUT method while maintaining the generalization over variety of functions. They maintained the algorithm flow by using a state machine inside the activation function. When the input was received, the result was registered at output after five clock cycles [10].

Panicker et al. [2012]Discussed the disadvantage of LUT method and showed the approximation values of piece wise approximation method. Piecewise linear (PWL) approximation method was used to obtain low values for both maximum and average error with low computational complexity. The PLAN approximation method used the digital gates to map the input value into output and the obtained output value was approximated one. The proposed method had the maximum mean square error value of 0.00187 for bipolar sigmoid activation [31].

Sahin et al. [2012] calculated the e^x function for RadBas, LogSig, and TanSig activation function using CORDIC-based exponent calculator. So the neurons was used the normalized value [32].

6 On-chip Learning

On-chip learning is an open area of research. Presently, no well defined method exists to define the network architecture (Number of layers, number of neurons, type of activation function, learning algorithm, etc) for a given task. Currently systematic trial and error method is used to find the architecture [33]. Some software tools like MATLAB are used for this purpose [16].

In hardware, there are more network characteristics to consider, many dealing with precision related issues like data and computational precision [37]. Similar simulation or fast prototyping tools for hardware are not well developed.

Backpropagation algorithm is a standard benchmark learning algorithm for hardware based learning [38]. On-chip learning provide good portability to the device and also it is most preferable for higher dimensional problems such as internet, bioinformatics etc [34].

Gomperts et.al [2011] proposed a framework for on-chip learning task. They tried to exploit the reconfigurability property of FPGAs to shift the flexibility of parameterized software based ANNs and ANN simulators to hardware platforms.

A main challenge available in on-chip learning is to find an architecture that minimize hardware costs, while maximizing performance, accuracy and parameterization. They compared the performance of generalized network structure with other hardware-based MLP implementation. Their proposed system can reach 530 million connections per second offline and 140 million online. Their system was capable of producing accurate convergence in training on par close to MATLAB simulation. They preferred the Xilinx Virtex-5 SX50T FPGA platform for their design [10].

7 MLP IMPLEMENTATION

MLP model is suitable to give solution for the non -linearly separable problem. If a hyper plane is used to classify the given input problem classes means that problem is called as linearly separable. In real time applications the problems are non-linearly separable. So the standard non-linear benchmark XOR problem [1] is chosen for MLP implementation.

7.1 XOR Problem

XOR problem have 2 inputs and 1 output. Based on these requirement structured MLP architecture to solve the XOR problem is shown in figure.3 [13]. By using off-chip training the weight and number of neurons in each layer are fixed to construct the MLP architecture.

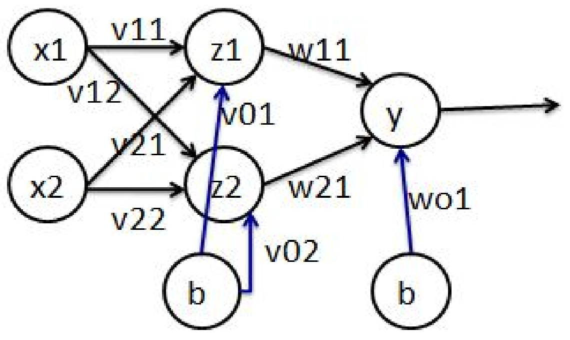

Fig. 3. MLP architecture to solve XOR problem

The detailed description of MLP architecture to solve the XOR problem is given in Table 2.

TABLE 2
MLP ARCHITECTURE DESCRIPTION TO SOLVE XOR PROBLEM

Layer	Activation function	No. of neurons
Input	Linear	2
Hidden	Sigmoid	2
Output	Sigmoid	1

7.2 Off-chip Training

In off-chip training, learning process relying on sequential software execution [34]. In this work training was done by using MATLAB software [16]. After the completion of training the network structure and weights were fixed.

The network was trained by using back propagation algorithm, the obtained final updated weight value to solve the XOR problem is given in Table 3. The performance of the network was tested by given an unseen input vector to the network. This process output is shown in Table 4.

TABLE 3
INITIAL AND UPDATED VALUE OF PARAMETERS

Parameter	Initial Value	Updated Value
v11	0.8147	5.6301
v12	0.1270	3.6205
v21	0.9058	5.6243
v22	0.9134	3.6052
w11	0.6324	6.9970
w12	0.0975	-7.5544
b1-1	0.2785	-2.2974
b1-2	0.5469	-5.5193
b2	0.9575	-3.1571

TABLE 4
TESTING PHASE OF THE NETWORK

Test set	Net output	Final output
[1,1]	-2.5536	0.0722
[0,0]	-4.3211	0.0131
[0,1]	4.2722	0.9862
[1,0]	4.2699	0.9862

TABLE 5
LEARNING RATE AND ERROR VALUE

Parameter	Value
Learning rate	0.9
Error value	0.01

During the feedback pass of back propagation algorithm weight updating process take place by reducing the MSE value till reach the specified error value given in table 5. The plot for epoch Vs MSE (Mean Square Error) is shown in figure 4. The given learning rate determines how fast the network gets converged.

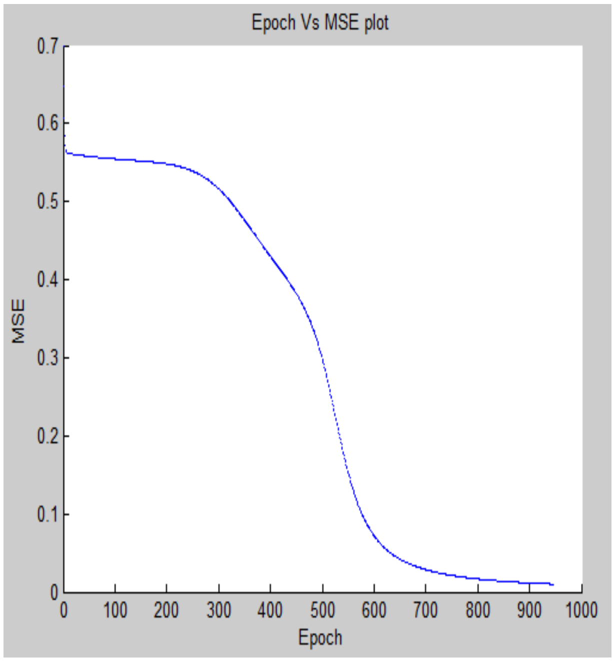

Fig. 4. Epoch Vs MSE plot

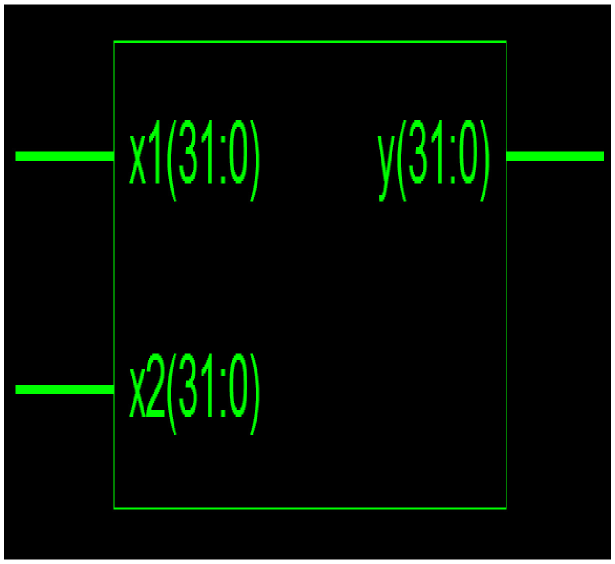

Fig. 5. (a) RTL Schemativ View

From this plot it may clear that for XOR problem the specified MLP structure may take 950 epochs to solve the problem.

7.3 Simulation & Synthesis Results of MLP

After the completion of training, the network structure is fixed. To do the hardware implementation of the structure the final updated weights value obtained during the training process will be used. Design entry to do the hardware implementation was done by using VHDL [17], [18].

Device utilized by the simple classical XOR problem on FPGAdevice is shown in Table 6.

TABLE 6
DEVICE UTILIZED BY XOR PROBLEM

Logic Utilization	**Need for XOR Problem**
No. of Slices	2464
No. of 4 input LUTs	4378
No. of bonded IOBs	96

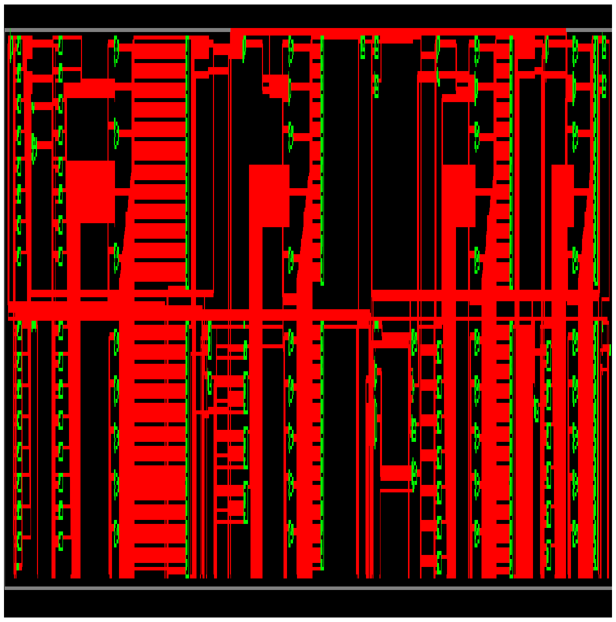

Fig. 5. (b) RTL Schemativ View

Figure.5.a, b shows the RTL schematic view of the synthesized XOR problem. RTL schematic view shows the hardware area required to solve the XOR problem. From this it is clear that the inherent parallelism property of ANN is preserved. Figure.6. shows the simulation result for XOR problem on Modelsim simulator tool. So it is clear that off-chip trained MLP to solve the XOR problem is easily implemented in FPGA.

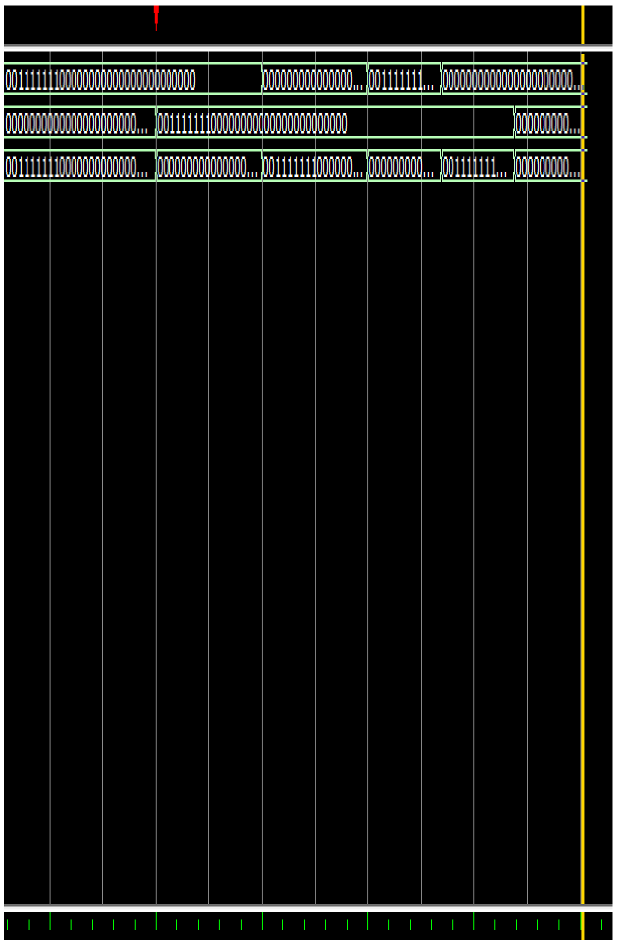

Fig. 6. Simulation Result

During FPGA implementation, to preserve the precision of network single precision floating point representation is used. It may give better accurate result.

Table 7 Shows the Comparison of Device utilization summary of off-chip based learning of XOR problem on different device environment. From the comparison table it is clear that for the same problem xc3s4000I-4fg900 device environment is over fitted and the higher version is needed for the implementation.

TABLE 7
COMPARISON OF DEVICE UTILIZATION PERCENTAGE SUMMARY

Logic Utilization	**xc3s4000I -4fg900**	**xc3s400-4pq 208**	**xc3s1000-fg456**
No. of Slices	Over fitted	61%	8%
No. of 4 input LUTs	89%	63%	7%
No. of bonded IOBs	60%	68%	15%

In off-chip learning this may easy, because the architecture is fixed and every needed for the given task implementation is known in advance. But in on-chip learning this is not a case, because more devices may make use of more resources but some may require less compared to others [35].

Latest Xilinx platform suited for on-chip learning is the Xilinx Virtex-5 SX50T FPGA. This model of the Virtex-5 contains 4080 CLBs and CLBs hold 8 logic function generator, 8 storage elements, a number of multiplexers and carry logic. Now this platform is large enough to test a range of online neural network of varying size [10], [36].

8 Conclusion

In this work various research works done by the researchers for effective implementation of artificial neural networks in FPGAs were analyzed. The complexity available in hardware implementation of sigmoid function and the solution for it also discussed. A simple MLP to solve classical XOR problem is designed and it is implemented on different device environment to shown the importance of hardware platform selection for on-chip learning. Finally the hardware based learning concept was introduced. Then it is planned to implement the neural network for on line classification problem.

9 Future Work

Based on the completed work it is planned to design an On-Chip trained network to do the diabetic retinopathy classification.

ACKNOWLEDGMENT

The authors like to gratefully acknowledge the enthusiastic support of their college management, principal and department of ECE of Einstein College of Engineering, Tirunelveli.

REFERENCES

[1] Simon Haykin, *Neural Networks: A Comprehensive Foundation*, 2ed. Addison Wesley Longman (Singapore) Private Limited, Delhi, 2001.

[2] http://ee.sharif.edu/~eeprojects/Electronics2/114.txt-project thesis.

[3] Himavathu.s, Muthuramalingam.A, Srinivasan.E, "Neural Network Implementation Using FPGA: Issues and Application" *World Academy of Science, Engineering and Technology*, no.48, pp. 625-631, 2008.

[4] Yihua Liao,"Neural Network in Hardware- Survey", http://bit.csc.lsu.edu/~jianhua/shiv2.pdf, 2001.

[5] M.Gopal, *Digital Control and Static Variable Methods.* Tata McGraw Hill,New Delhi, 1997.

[6] 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.*

[7] Jagath C. Rajapakse, *FPGA Implementation of Neural Networks*.Amos R. Omondi,ISBN-10 0-387-28487-7@ *Sprinker, 2006.*

[8] R.Girones, A.Salcedo, "Implementation with FPGAs of a pipelined On-line Backpropagation", *IEEE Conference Publications,* 5-8 Sep, 1999.

[9] Savron and unsal, "Hardware Implementation of a feed forward neural network using FPGAs", ICONIP'06 *Proceedings of the 13th international conference on Neural information processing*, vol.III, pp. 1105-1112, 2006.

[10] Alexander Gomperts, Aghisek Ukil,"Development and Implemenation of Parameterized FPGA-Based General Purpose Neural Networks for Online Applications",*IEEE Transaction on industrial informatics*, vol. 7, no.1, Feb, 2011.

[11] Sathish Kumar, *Neural Networks: A Classroom Approach*. Tata McGraw-Hill Publishing Company Limited, New Delhi, 2004.

[12] David E. Rumelhart, *BackPropagation: Theory, Architectures, and Applications.* Yves Chauvin, google preview, 1995.

[13] Martin T. Hagan, *Neural Network Design*. Howard B. Demuth, and Mark Beale, Thomson Learning, New Delhi, 2009.

[14] Alin Tisan, Stefen Oniga, Daniel MIC, Attila Buchman, "Digital Implementation Of The Sigmoid Function For FPGA Circuits", *ACTA Technica NAPOCENSIS Electronics and Telecommunication*, vol.50, no.2, 2009.

[15] Neil H.E.Weste, *CMOS VLSI DESIGN-A Circuits and Systems Perspective.* Pearson 3rdEdition.(ISBN 978-81-7758-568-1),2003.

[16] R.Hunt, *A Guide to MATLAB for Beginners and Experienced Users*. L.Lipsman, M.Rosenber, Cambridge University Press, ISBN-I3 978-0-511-07792-0, 2001.

[17] J. Bhaskar, *VHDL Primer.* PTR Prentice Hall, 2003.

[18] A.Volnei, *Circuit Design with VHDL.* Pedroni, ISBN 0-262-16224-5, *Library of Congress* Cataloging-in-Publication Data, 2004.

[19] S.A.Hoda and A.Mahmond, "Digital Neural Processing Unit For Electronic Nose", *Proceedings of Ninth Great Lake Symposium on VLSI, IEEE Computer Society,* pp.236-237,1999.

[20] Eu-Ttum Jang, Wan-Young Chung, "Electronics Nose Module with System on Chip", *IEEE proceeding of sensors*, vol.2, pp.1335-1338, 2002.

[21] Tomoo Aoyama, Qianyi Wang, Ryosuke Suematsu, Ryosuke Shimizu, Umpei Nagashima, "Learning Algorithm For a Neural Network in FPGA",*IEEE Transaction, 2002*.

[22] M. Zheng, M. Tarbouchi, D. Bouchard, J. Dunfield, (2006) "FPGA Implementation of a Neural Network Control System for a Helicopter" *Proceedings of the 7th WSEAS International Conference on Neural Networks*, Cavtat, Croatia, pp.7-10, june 12-14, 2006.

[23] Himavathi S., Anitha D., Muthuramalingam A., "Feedforward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization", *IEEE Trans. on Neural Networks*, vol. 18. no. 3, 2007.

[24] Rafid Ahmed Khalil, "Hardware Implementation of Back propagation Neural Networks on Field programmable Gate Array(FPGA)", Al-*Rafidain Engineering*, vol.16, no.3,aug, 2008.

[25] Djalal Eddine Khodja, Aissa Kheldoun and Larbi Refoufi, "Sigmoid Function Approximation For ANN Implementation In FPGA Devices", *Proceedings of the 9th WSEAS international conference on circuits,systems, electronics, control and signal processing*,pp.112-116,2010.(ISBN:978-960-474-262-2).

[26] Xiaobin, Lianwen, Dongsheng, Junxum, "A mixed Parallel Neural Networks Computing Unit Implemented in FPGA", *IEEE Int. Conf. Neural Networks & Signal Processing, China*, pp.324-327, dec 14-17, 2003.

[27] http://bit.csc.lsu.edu/~jianhua/shiv2.pdf

[28] Valeriu Beiu, Jan Peperstraete, Joos Vandewalleand Rudy Lauwereins, "Close Approximations of Sigmoid Functions by Sum of Steps for VLSI Implementation of Neural Networks", *The Scientific Annals, Section: Informatics*, vol. 40 (XXXX), no. 1, 1994.

[29] K.Basterretxea, and J.M.Tarela, "Approximation of sigmoid function and the derivative for artificial neurons", in Mastorakis, N. (Ed.): *Advances in neural networks and applications*, WSES Press, Athens, pp.397–401, 2001.

[30] M.T.Tommiska,"Efficient digital implementation of the sigmoid function for reprogrammable logic", *IEE Proc.-Comput. Digit.Tech*, vol. 150, no. 6, pp. 403-411,nov, 2003.

[31] Manish Panicker, C.Babu, (2012) "Efficient FPGA Implementation of Sigmoid and Bipolar Sigmoid Activation Functions for Multilayer Perceptrons", *IOSR Journal of Engineering (IOSRJEN)* ISSN: 2250-3021, vol. 2, no 6, pp. 1352-1356, june, 2012.

[32] I. Sahin, I. Koyuncu, "Design and Implementation of Neural Networks Neurons with RadBas,LogSig, and TanSig Activation Functions on FPGA", *Electronics And Electrical Engineering*, Issn 1392 – 1215, no. 4(120),2012.

[33] S.N.Sivanandam, *Introduction to Neural Networks Using Matlab 6.0.* Sumathi, Deepa.m, *Tata McGraw-Hill,* ISBN 0-07-059112-1, 2006.

[34] Antonio de Padua Braga, Tiago Mendonca Dasilva, Willian Soares Lacerda,"Pipelined on-line Back-propagation training of an artificial neural network on a parallel multiprocessor system", Learning and Nonlinear Module(L&NLM)-*Journal of the Brazillian Society on Neural Networks*, vo1.8,no.2,pp.120-123, 2010.

[35]http://business.highbeam.com/articles/436704/international-journal-information-technology

[36] http://alexandria.tue.nl/repository/books/644229.pdf

[37] Zhu and P.Sutton, "FPGA implementation of Neural Networks: A Survey of a Decade of Progress", *Lecture Notes in Computer Science*, vol. 2778/2003, pp. 1062-1066, 2003.

[38] Mark Pethick, Michael Liddle, Paul Werstein, and Zhiyi Huang, "Parallelization of a Backpropagation Neural Network on a Cluster Computer", *15th IASTED International Conference on Parallel and Distributed Computing and Systems*, pp. 574-582, 2003.