FPGA BASED SIMULATION OF A FEED FORWARD NEURAL NETWORK USING VHDL

B.M.S.S.S. Aditya[1] , B. Srinivasa Rao[2], P. Premchand[3]
[1]Department of Computer Science Engineering, SRMIST (SRM University), Kattankulathur, Chennai, Tamil Nadu, India- 603203
[2]Department of Computer Science Engineering, Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, Telangana, India-500090
[3]Department of Computer Science Engineering, University College of Engineering, Osmania University, Hyderabad,

Abstract : In this paper a hardware implementation of a neural network using Field Programmable Gate Arrays (FPGA) is presented. Digital system architecture is designed to realize a feed forward neural network. The feed forward network has 7 nodes and 3 layers, and this network achieving node parallelism and layer parallelism the designed architecture is described using Very High Speed Integrated Circuits Hardware Description Language (VHDL). After a brief review the implementation our simulation results have been presented.

Index Terms **- Neural network, FPGA, VHDL, feed forward network**

1. INTRODUCTION

The interest in neural networks comes from the networks' ability to mimic human brain as well as its ability to learn and respond. As a result, neural networks have been used in a large number of applications and have proven to be effective in performing complex functions in a variety of fields like pattern recognition, signal processing, control systems etc. Most of the work done in this field until now consists of software simulations, investigating capabilities of ANN models or new algorithms. But hardware implementations are also essential for applicability and for taking the advantage of neural network's inherent parallelism. There are analog, digital and also mixed system architectures proposed for the implementation of ANNs. The analog ones are more precise but difficult to implement and have problems with weight storage. Digital designs have the advantage of low noise sensitivity, and weight storage is not a problem. With the advance in programmable logic device technologies, FPGAs has gained much interest in digital system design. They are user configurable. ANNs are biologically inspired and require parallel computations in their nature. Microprocessors are not suitable for parallel designs. Designing fully parallel modules can be available by ASICs and VLSIs but it is expensive and time consuming to develop such chips. In addition the design results in an ANN suited only for one target application. FPGAs not only offer parallelism but also flexible designs, savings in cost and design cycle.

Neuron is configurable in many architectures according to application, most common architecture neural network are feed forward neural network, feedback neural network. The Human brain consist of a large number, more than a billion of neurons that process information. Each cell works like a simple processor. The massive interaction between all cells and their parallelism create a network of neuron, is called as neural network. The basic model of a neuron as shown in Fig.1

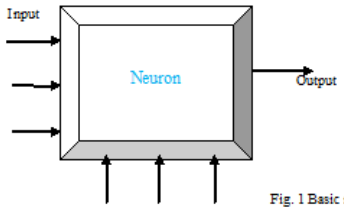


Fig. 1 Basic model of neuron

II. FUNCTIONAL MODEL OF ARTIFICIAL NEURAL NETWORK

Neural networks have multiple neurons and each connected to each other be a connection and each connection has it own strength. This strength called weight of connection. This weight is positive or negative is depending on network connection. In this model we have multiple inputs at one neuron and each connected by a specific connection with is specific weight. When the input is passing through connection then input signal and weight is multiples, and generates a product term. This product term are many, each of every input and weight connection. Now sum of all product term and is give a final value to activate the particular neuron activation function to take the decision of neuron process. So an artificial neuron as shown in Fig1 is the basic element of a neural network. It consists of three basic components that include weights, threshold, and a single activation function. The activation function is most important factor in neural network. This activation function shows the relationship between input and output in linear, nonlinear, discrete or in continuous form. Fig 2 shows the mathematical model for neuron

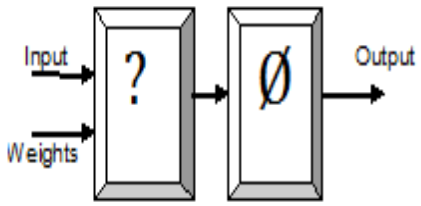


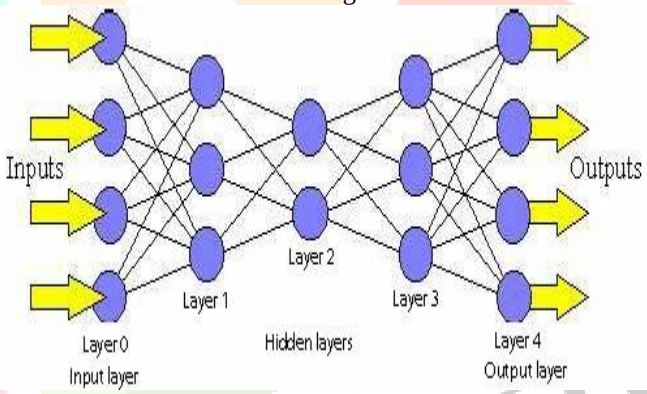
Fig.2 Mathematical model of neuron

where, input in1, in2, in3….inn and is connected by interconnection wire with weight w1, w2, w3,….wn, and activation function is Ø determined that neuron is process or not. A processing unit sums the inputs, and then applies a non linear activation function (i.e. squashing/ transfer/threshold function).An output line transmits the result to other neurons. The sum is intermediate result of summer where all product terms are sum. And finally output get from Ø activation function.

$$\text{sum} = \sum_{i=0}^{n} \text{IN(i)} * \text{W(i)}$$

$$\text{output} = \text{Ø(sum)}$$

Feed Forward Neural Network**:** Feed-forward networks have the following characteristics: Perceptrons are arranged in layers, with the first layer taking in inputs and the last layer producing outputs. The middle layers have no connection with the external world, and hence are called hidden layers. 2. Each perceptron in one layer is connected to every perceptron on the next layer. Hence information is constantly "fed forward" from one layer to the next., and this explains why these networks are called feed-forward networks. 3. There is no connection among perceptrons in the same layer. A general feed-forward neural network is as shown in Fig.3.



In this work we chose multiply and accumulate structure for neurons. In this structure there is one multiplier and one accumulator per neuron. The inputs from previous layer neurons enter the neuron serially and are multiplied with their corresponding weights. Every neuron has its own weight storage. Multiplied values are summed in an accumulator. The precision of the weights and input values are both 8-bits. The multiplier is an 8-bit by 8-bit multiplier, which results in a 16-bit product, and the accumulator is 16-bits wide. The activation function used is threshold value function. In this function sum value compare with a cut off value, when sum value is below then that cut off value then output is 0. And sum value is above then cut off value then output is 1.

III. IMPLEMENTATION APPROACH

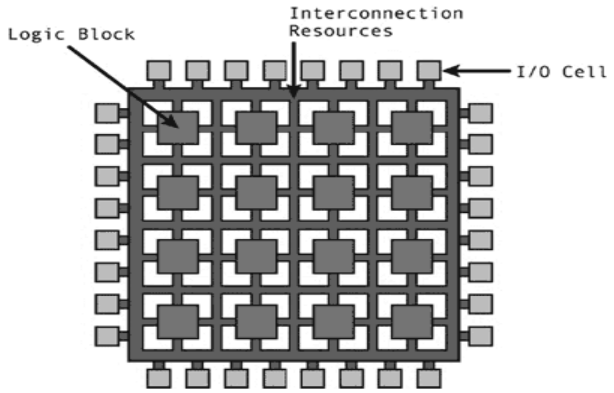
There are many different approaches for neural network implementation like digital, analog, hybrid. In analog implementation approach is adopted for continues response and in digital for discrete value. Digital approach offers easy implementation of any complex application. Digital approach is having many way to implement neural network like custom IC, Digital signal processing, Application specific IC, Programmable logic device (PLD). This entire approaches programmable logic device gives high performances high capacity digital neural network. Implementation PLD is having one more property that is reconfigurable number of time. For PLD device programming is very easily by VHDL, like FPGA.

VHDL: VHDL stands for very high-speed integrated circuit hardware description language. This is one of the programming languages used to model a digital system by dataflow, behavioral and structural style of modeling. This language was first introduced in 1981 for the department of US Defense, in 1983 IBM, Texas instruments started to develop this language. High level modeling capabilities of VHDL enable the use of this language for modeling for neurons. In addition to being a fully concurrent language, VHDL descriptions can be written with one-to-one hardware correspondence that will be useful in the hardware implementation of our neural network. This language has two basic operation synthesis and simulation.

FPGA: FPGA is a silicon chip with unconnected logic gates. It is an integrated circuit that contains many (64 to over 10,000) identical logic cells that can be viewed as standard components. The individual cells are interconnected by a matrix of wires and programmable switches. Field Programmable means that the FPGA's function is defined by a user's program rather than by the manufacturer of the device. Depending

on the particular device, the program is either 'burned' in permanently or semi-permanently as part of a board assembly process, or is loaded from an external memory each time the device is powered up. An FPGA chip model is shown in Fig.5.

FPGA Flow Design: A Typical FPGA design process is shown in Fig.4. The design process and flow design for FPGA is explained taking a typical example of MPEG4 processor. At the outset, the requirements and functional specifications are made. For instance, a customer would like the design team to implement an MPEG4 processor that is able to run at 400 MHz and fit a particular Xilinx Spartan 6 chipset.



The design team would have to break up the MPEG4 processor into various functional blocks, and start to implement the designs. Often, the designer may choose either Verilog or VHDL as the **hardware descriptive language (HDL).** When a functional block is completed, it may be simulated to test its functionality. At this stage, timing delays are ignored, only functionality is checked in behavioral simulation. After the design is completed, the next stage would be to synthesis the design. The tool checks for syntax and design errors, before translating and mapping the codes to its logic gate equivalence. Now the digital logics interfaced to the external world via programmable input/output blocks. Depending on the I/O pins specified by the designer, the logic blocks chosen by the tool would be different. The process of mapping the logic gate equivalence to the actual logic blocks on the FPGA is known as **Place and Route**. The final step would be to generate the machine codes that would be downloaded to configure the FPGA.

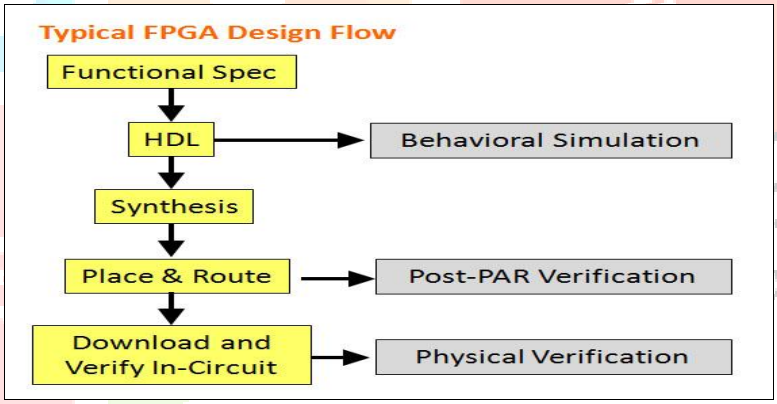


Fig.4 Typical FPGA Design Flow

Application: The application selected in this work feed forward neural network (four neurons in the input layer, two neurons in the hidden layer and one neuron in the output layer) is implemented on a XILINX Vertex 2p chip with 200,000 typical gate counts. Firstly, weights are written to a VHDL Package file. This file, along with other VHDL coding is compiled, synthesized and implemented with Xilinx ISE software tools. Simulations are made with ModelSim 5.4a. Finally the design is realized on a Diligent DIIE demo board having the Xilinx FPGA chip.

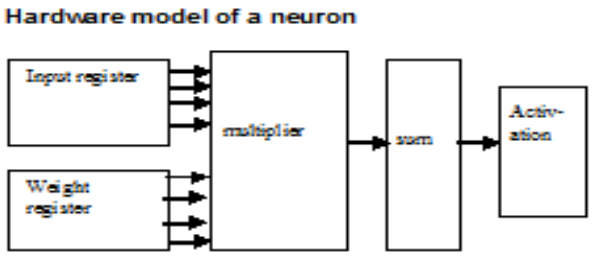


Fig.5: Basic neuron architecture

The neuron is sub divided into 5 components
1. Input register
2. Weight register

3. Multiplier
4. Sum
5. Activation

The input register is 8-bit register to store input value and n is number of input value is 4. Similarly weight register is same to store the weight value and is also 4. Input and weight register output is connected to 8-bit multiplier, where input and weight value product is generate product terms (P). $Prod_n = IN_n * Wh_n$ Here n is number of input value. 0, 1, 2,.n.Prod is product terms, IN is input and Wh is weight. The output of summer is 8-bit, which is connected to comparator (activation function). Comparator is working on LUT condition, and condition is depending on threshold activation function.

IV. RESULTS AND DISCUSSION

In this paper, implementation of neuron is done in VHDL language where synthesis and simulation result shown below.
VHDL code implementation for neuron

- This VHDL code of a neuron is synchronous with respect to positive edge of system clock (clock). And have reset(r) functionality, for any moment system goes back to initial state.
- Input X, which is 4 address arrays with standard logic data type, each address has 8 bit data. {X (0), X (1), X (2), X (3)}.
- Weight w, which is also 4 address array with standard logic data type, each address has 8 bit data. {W (0), W (1), W (2), W (3)}.
- Output is 1 bit data type.

A Device summary
FPGA Device family – virtex2p
Device – xc2vp7
Package – fg456
Speed grade - -5
Optimization goal – speed

HDL synthesis report
8-bit register :
Number of bonded IOBs: 50 out of 248 0%
Number of MULT18X18s:35 out of 44 79%
Number of GCLKs: 1 out of 16 6%
10x9-bit multiplier: 7
8x8-bit multiplier: 28
9-bit comparator less: 7
8-bit adder : 27

Device utilization summary
Selected Device: xc2vp7fg456-5
Number of Slices: 294 out of 4928 5%
Number of Flip Flops: 56 out of 9856 0%
Number of input LUTs: 439 out of 9856 4%
10-bit adder : 7
16-bit adder : 21

Timing Summary:
Minimum period: 14.472ns (Maximum Frequency: 69.099MHz)
Minimum input arrival time before clock: 20.180ns
Maximum output required time after Clock: 4.717ns

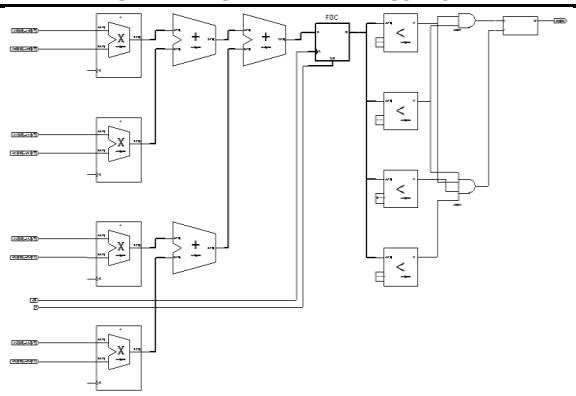


Fig.6 RTL view of neural network

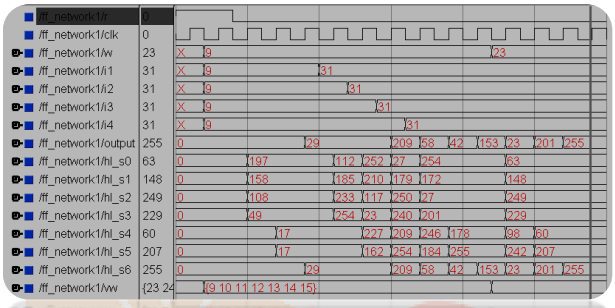


Fig.7 Simulation results

V. FUTURE WORK AND CONCLUSION

The aim of this research is to implement a single neuron in the domain of speed and minimum hardware. This neural is multiple inputs with threshold activation function on FPGA Spartan-II xc2s15 -5 tq144 hardware. This neuron implementation used arithmetic component and LUT to achieve maximum speed of processing. This basic neuron is use to implement any kind neural network and its application like feed forward, feedback, and multi layer parallel network.

VI. ACKNOWLEDGEMENTS

The authors are thankful to the respective affiliated institutions for the facilities provided.

REFERENCES

[1] Lee Yee Ann, Phaklen Ehkan and Mohd Yusoff Mashor Possibility of Hybrid Multilayered Perceptron Neural Network Realization on FPGA and Its Challenges. Available https://www.researchgate.net / publication / 300113167 **2017**

[2] Dondon Philippe, Julien Carvalho, Remi Gardere and Valeri Mladenov" Implementation of a feed-forward Artificial Neural Network in VHDL on FPGAhttps:// www.researchgate.net / publication / 282209687 **2017**

[3] Pankaj kumar Sharma and Mr. Ravishanker Sharma Implementation of Neural Network Model For Cancer Detection Based On Back Propagation With The Help of Python Based Hardware Description Language International Journal of Converging Technologies and Management (IJCTM) Volume 3, Issue 1, **2017** ISSN : 2455-7528

[4] Ismail Koyuncu, Design and Implementation of High Speed Artificial Neural Network Based Sprott 94 S System on FPGA International Journal of Intelligent Systems and Applications in Engineering Vol. 4 No. 2 **2016**.

[5] Amitkumar B. Khondeand, Yogesh Sharma B and Sanjay Badjate Implementation of Multilayer Feed Forward Neural Network using VHDL International Journal of Computer Applications (0975 – 8887) Volume 155 – No 1, http://www.ijcaonline.org/archives / volume155/ number1/khonde-**2016**-ijca-912111.pdf

[6] Sheng - Hsien HSIEH and Ching-Han CHEN https://www.design-reuse.com /articles /13669 / design-and-fpga-implementation-of-an-interpolative - neural-network - for-digital-image **2016**

[7] R.Gadea, J.Cerda, F.Ballester and A.Macholi, Artificial neural network implementation on a single FPGA of a pipelined on-line back propagation **IEEE Xplore** 10.1109/ ISSS .2000.874054 http:// www .ijcaonline.org /archives /volume155 /number1/ khonde-2016-ijca-912111.pdf **2016**

[8] Seema Singh , Shreyashree Sanjeevi , Suma V and Akhil Talashi FPGA Implementation of a Trained Neural Network, 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

[9] Dhirajkumar S. Jinde et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , **2014,** 3431-3433

[10] Alin Tisan and Jeannette Chin "An End User Platform for FPGA-based Design and Rapid Prototyping of FeedForward Artificial Neural Networks with onchip Back Propagation learning" http://arro.anglia.ac.uk/613196/1/IIT-enose-ARRO.pdf **2014**

[11] Chandrashekhar Kalbande and Anil Bavaskar "Implementation of FPGA-Based General Purpose Artificial Neural Network", http://www.irdindia.in/journal_itsi/pdf/vol1_iss3/18.pdfISSN (PRINT) : 2320 – 8945, Vol.1, 3, **2013.**

[12] S. L. Pinjare and Arun Kumar M. Yelahanka,. Implementation of Neural Network Back Propagation Training Algorithm on FPGA International Journal of Computer Applications (0975 – 8887) Volume 52– No.6. **2012.**

[13] Alexander Gomperts, Abhisek Ukil and Franz Zurfluh IEEE Transactions on Industrial Informatics Vol. 7, 1 78 - 89 **2011**

[14] Vinit mahajan , Amit prakash singh and Anuradha Chug FPGA IMPLEMENTATION OF FEED FORWARD NETWORK WITH SIGMOID ACTIVATION FUNCTION SCRIBD Explore: 1-10 **2010**

[15] Janardan Misra and Indranil Saha Artificial neural networks in hardware: A survey of two decades of progress" https://doi.org /10.1016 /j.neucom.2010.03.021 **2010.**

[16]Douglas J. Smith, "HDL Chip design-A practical guide for designing synthesizing and simulating ASIC and FPGA using VHDL or Verilog", Tata McGraw-Hill, **2005**

[17] Jihong Liu and Deqin Liang **"**A Survey of FPGA-Based Hardware Implementation of ANNs" ICNN&B '05 **2005**. **IEEE Xplore DOI:** 10.1109/ICNNB.2005.1614769

[18] Satish Kumar "neural network, Tata McGraw-Hill, 2004

[19] M. Ananda Ro and J. Srinivas, Neural Networks Algorithms and Applications, Alpha Science International Ltd., Part III, pp.157, **2003**.

[20] Some content and figures have been taken from Wikipedia.