*Turkish Journal of Computer and Mathematics Education Vol.12 No.12 (2021), 2192-2199*

*Research Article*

# Implementation Of Perceptron Algorithm For Logic Gates Using ANN

**Chowdhari M I[1], Ganesh B[2] , Harshan K R[3], Dhanush C[4], Prashant.V.Joshi[5]**

[1]School of Electronics and Communication Engineering, REVA University, India.
[2]School of Electronics and Communication Engineering, REVA University, India
[3]School of Electronics and Communication Engineering, REVA University, India.
[4]School of Electronics and Communication Engineering, REVA University, India.
[5]School of Electronics and Communication Engineering, REVA University, India.

**Article History:** Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 23 May 2021

**Abstract**: Biological/organic neurons ar the main components of human brain. Neuron processes and sends information to other neuron by sending electrical signals. Artificial neurons are inspired by biological neurons. These artificial neurons are interconnected to form artificial neural networks. This neural network's structure helps to make computation faster for certain tasks hence making it suitable for implementation in VLSI technology. Then the Logic gates is implemented using feed forward network and the design is realized using Verilog HDL.

**Keywords:** Artificial Neural Network, Verilog ,VHDL, Artificial Neurons, Logic Gates, Perceptron**.**

## 1. Introduction

Biological neurons are the main components of human brain as shown in Fig.1. Neurons consist of body cells, dendrites and axons. It processes and sends information from one neuron to other neuron using electrical signals. Each neuron receives an input signal from its dendrite and produces an output signal on its axon. The outgoing axons are connected to the other neuron dendrites through synapses. Each synapse has the power to control the strength of the influence of one neuron on another neuron. Dendrite carries signals to the body of the target neurons where they are summed up. If the final number is above a certain threshold, the neuron is triggered by sending a surge to the axon.

Artificial neurons are inspired by biological neurons and try to formulate the models described above in mathematical form. Artificial neurons have a limited number of inputs with associated weights and activation functions (also called transfer functions). The output of the neuron is the result of the activation function applied to the number of input weights. Artificial neurons are interconnected to form artificial neural networks.The logic gates are designed using Verilog. And the software used is ModelSim-Altera 6.4a (Quartus II 9.0) Starter Edition

## 2. Literature survey

Design and hardware implementation of a couple of neurons on Field Programmable (FPGA) [1] is completed sequentially. First greater then one Neurons are applied then logic gates and Adder circuit are accomplished the usage of Feed Forward Neural Network. FPGA has been used to lower the neuron hardware with the aid of using planning activation feature in the neuron with the aid of using now no longer the usage of lookup tables.

A hardware implementation of ANN and implementation of logic gates using ANN on Field Programmable Gate Arrays (FPGA) [2] is presented. A digital system architecture for feed forward multilayer neural network is figured it out. The parallel structure of a neural network makes it conceivably fast for the calculation of certain tasks that makes a neural network well suited for implementation in VLSI technology. Then logic gates are executed using Feed Forward Neural Network.

A hardware implementation of a neural network using Field Programmable Gate Arrays (FPGA) [3] is presented. A digital system design is planned to realize a feedforward multilayer neural network. The designed architecture is depicted using Very High-Speed Integrated Circuits Hardware Description Language (VHDL) and implemented in an FPGA chip. The design is confirmed on an FPGA demo board.

ANN[4] is majorly used to learn data from systems for different types of applications. Field Programmable

Gate Array based Multilayer Perceptron. ANN neuron is developed. Experiments were made to show the hardware realization of the AN using FPGA.

How to reconfigure [5] a logic gate for a variety of functions is an interesting topic. A different method of designing logic gates are proposed. Initially, due to the training ability of the multilayer perceptron neural network, it was used to create a new type of logic and full adder gates. In this method, the perceptron network was trained and then tested.

This article [6] will explain how neural network is proficient to make the arithmetic operations like Addition, Subtraction, Multiplication and Division of binary numbers. As we all know that human brain is divided into 2 halves or hemispheres (Right and Left). The Left- brain thinking is verbal (logic, analysis, computation and other functions), whereas Right brain thinking is Non-verbal (creativity, imaginary, intuition and other functions). Here the input layer of neural network is used to represent input, in the same manner hidden units are used to represent the encapsulated operations like conversion from decimal to binary and vice versa, the last layer output is used to produce the output values.

Neuromorphic registering [7] has come to allude to an assortment of mind propelled PCs, gadgets, and models that contrast the inescapable von Neumann PC engineering. This organically roused approach has made profoundly associated manufactured neurons and neurotransmitters can be utilized to make neuroscience hypotheses just as tackle testing AI issues.

Quantum Neural Networks (QNN) [8] Neuron networks with very low accuracy and activation during execution. During quantum weighing and activation, parameter gradients are calculated. During the forwarding process, QNN significantly reduces memory capacity and memory access and replaces most arithmetic operations with budget operations. As a result, a strong reduction in energy consumption is expected. They have trained QNN for MNIST, CIFAR-10, SVHN and Image Net. The resulting QNN achieves predictive accuracy that is comparable to the 32-bit analog.

Description of artificial neural networks (ANN) [9]. Using VHDL allows further application of the FPGA system. In fact, the most important point for using an FPGA is the Flexibility is preferred over other systems, such as ASICS, dedicated to unique architecture, and allows parallel programming of internal ANN calculations and one of its advantages. Usually, FPGAs do not integrate unlimited logic resource packages, and the limitations of this style require design optimization to achieve the best speed of execution and resource consumption.

FPGA implementation of a precision floating point IEEE-754 standard MAC [10] is used in artificial neural networks which are used to feed the weighted inputs. The use of floating- point numbers increases the range of data presentation from very small to very large, which is especially recommended for artificial neural networks.

## 3. Proposed methodology

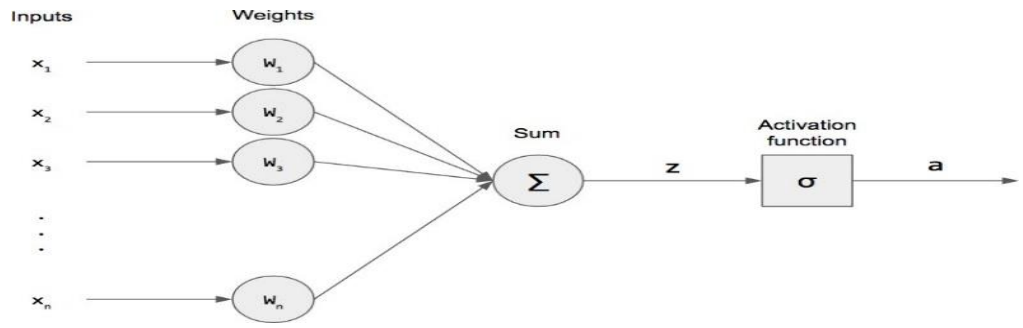



Figure 1. Structural diagram of ANN

The above network has an inputs I.e. X1, X2……Xn. Followed by consists of weights w1, w2,…..wn. Further it has summer where inputs with weights are added and given to the activation function and The Perceptron Model implements the following function:



$$z = \sum_{i=1}^{n} W_i x_i$$

The z value is given to the activation function: y = f(z). The function f(z) is known as activation function.

**UNIT STEP ACTIVATION FUNCTION**

A unit step activation function is a more used in neural networks. The output assumes value 0 for negative argument and 1 for positive argument. The function is as follows

$$f(x) = o \; when \; x < o, \quad 1 \; when \; x >= o$$

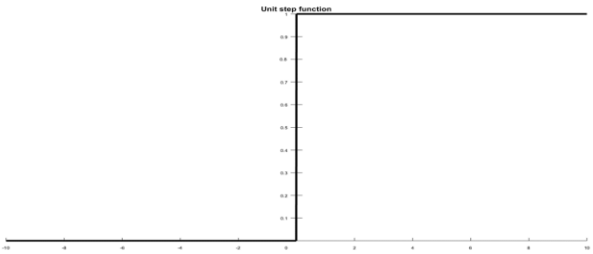
Figure 2. Graph showing the unit step function curve

IMPLEMENTING LOGIC GATES

**3.1.1 NOT gate.**

NOT logical function truth table is of only 1-bit binary input (0 or 1), i.e, the input vector x and the corresponding output y.

| X | ÿ |
|---|---|
| **0** | **1** |
| **1** | **0** |




Now for the corresponding weight vector w of the input vector x, the associated Perceptron Function can be defined as: y = θ(wx+b) ……. (1)

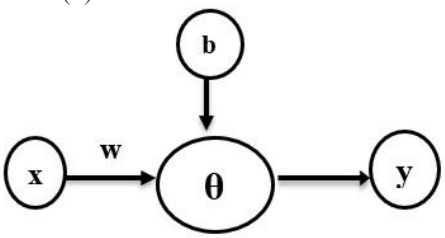



figure 3. Neuron model for NOT gate

For the implementation, considered weight parameter is w = - 1 and the bias parameter is b = 0.5
Substituting w and b in equation 1, y = θ ((-1*x) + (0.5)) ……... (2) Case 1.
Consider x = 0 and substituting in equation 2
y = θ ((-1*0) +(0.5))
y = θ (0.5) …… (3) Applying unit step function on equation 3, Therefore, y=1
Case 2: Consider x = 1 and substituting in equation 2
y = θ ((-1*1) + (0.5))
y = θ (-0.5) …… (4) Applying unit step function on equation 4,
Therefore, y=0

**3.1.2 OR gate.**

OR logical function truth table for 2-bit binary variables, i.e, the input vector x:(x1, x2) and the corresponding output y.

| x1 | x2 | ÿ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Now for the corresponding weight vector w:(w1, w2) of the input vector x:(x1, x2) the associated Perceptron Function can be defined as:

$y = \theta((w1x1) + (w2x2) + b)$ ……. (11)

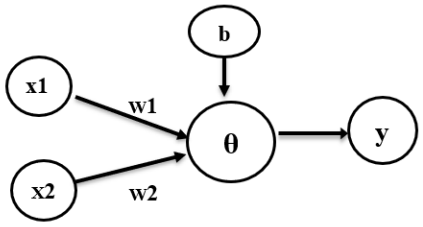



Figure 4. Neuron model for OR gate

For the implementation, considered weight parameters are w1= 1, w2= 1 and the bias parameter is b = -1.5.
Substituting w1, w2 and b in equation 11,

$y = \theta ((1*x1) + (1*x2) + (-1.5))$ ……... (12) Case 1.

Consider x1 = 0, x2 = 0 and substituting in equation 12

$y = \theta ((1*0) + (1*0) + (-1.5))$

$y = \theta (-1.5)$ …… (13) Applying unit step function on equation 13, Therefore, y=0

Case 2.

Consider x1 = 0, x2 = 1 and substituting in equation 12

$y = \theta ((1*0) + (1*1) + (-1.5))$

$y = \theta (-0.5)$ …… (14) Applying unit step function on equation 14, Therefore, y=0

Case 3.

Consider x1 = 1, x2 = 0 and substituting in equation 12

$y = \theta ((1*1) + (1*0) + (-1.5))$

$y = \theta (-0.5)$ …… (15) Applying unit step function on equation 15, Therefore, y=0

Case 4.

Consider x1 = 1, x2 = 1 and substituting in equation 12

$y = \theta ((1*1) + (1*1) + (-1.5))$

$y = \theta (0.5)$ …… (16) Applying unit step function on equation 16, Therefore, y=1

**3.1.3 AND gate.**

AND logical function truth table for 2-bit binary variables, i.e, the input vector x:(x1, x2) and the corresponding output y.

| X1 | X2 | ÿ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Now for the corresponding weight vector w:(w1, w2) of the input vector x:(x1, x2) the associated Perceptron Function can be

defined as:

$y = \theta((w1x1) + (w2x2) + b)$ ……. (11)

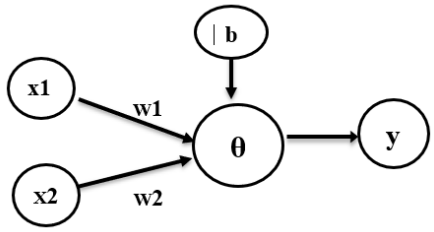



Figure 5. Neuron model for AND gate

For the implementation, considered weight parameters are w1= 1, w2= 1 and the bias parameter is b = -1.5.

Substituting w1, w2 and b in equation 11,

$y = \theta ((1*x1) + (1*x2) + (-1.5))$ ……... (12) Case 1.

Consider x1 = 0, x2 = 0 and substituting in equation 12

$y = \theta ((1*0) + (1*0) + (-1.5))$

$y = \theta (-1.5)$ …… (13) Applying unit step function on equation 13, Therefore, y=0

Case 2.

Consider x1 = 0, x2 = 1 and substituting in equation 12

$y = \theta ((1*0) + (1*1) + (-1.5))$

$y = \theta (-0.5)$ …… (14) Applying unit step function on equation 14, Therefore, y=0

Case 3.

Consider x1 = 1, x2 = 0 and substituting in equation 12

$y = \theta ((1*1) + (1*0) + (-1.5))$

$y = \theta (-0.5)$ …… (15) Applying unit step function on equation 15, Therefore, y=0

Case 4.

Consider x1 = 1, x2 = 1 and substituting in equation 12

$y = \theta ((1*1) + (1*1) + (-1.5))$

$y = \theta (0.5)$ …… (16) Applying unit step function on equation 16, Therefore, y=1

**3.1.4 XOR gate.**

XOR logical function truth table for 2-bit binary variables, i.e, the input vector x:(x1, x2) and the corresponding output y.

| X1 | X2 | ÿ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

We observe that,

XOR(x1, x2) = AND(NOT(AND(x1, x2)), OR(x1, x2)) Designing the Perceptron Network:

1. Step1: Now for the corresponding weight vector w: (w1, w2) of the input vector x: (x1, x2) to the AND and OR node, the associated Perceptron Function can be defined as:

y1 = θ((w1*x1) + (w2*x2) + bAND) …….. (17) y2 = θ((w1*x1) + (w2*x2) + bOR) …… (18)

2. Step2: The output y1 from the AND node will be inputed to the NOT node with weight WNOT and the associated Perceptron Function can be defined as:

y3 = θ (wNOT*y1 + bNOT) …… (19)

3. Step3: The output y2 from the OR node and the output y3 from NOT node as mentioned in Step2 will be inputed to the AND node with weight (wAND1, wAND2). Then the corresponding output y is the final output of the XOR logic function. The associated Perceptron Function can be defined as:

y = θ((wAND1*y3) + (wAND2*y2) +bAND) ……. (20)

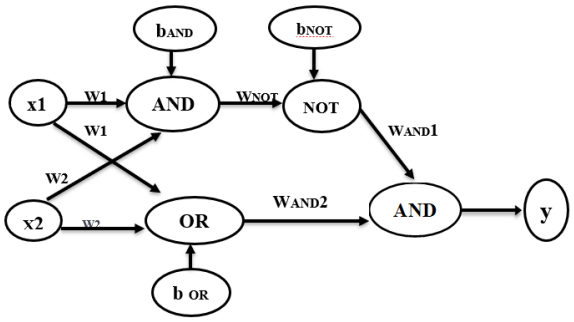



Figure 6. Neuron model for XOR gate

For the implementation, considered weight parameters are w1= 1, w2= 1 , WNOT = -1, wAND1= 1, wAND2= 1 and the bias parameter is bAND = -1.5, bOR = -0.5, bNOT= 0.5.

Substituting w1, w2 and bAND, bOR in equation 17, 18. y1 = θ((1*x1) + (1*x2) + (-1.5)) …….. (21)

y2 = θ((1*x1) + (1*x2) + (-0.5)) …… (22) For equation 21,

Case 1.

Consider x1 = 0, x2 = 0 and substituting in equation 21 y1 = θ ((1*0) + (1*0) + (-1.5))

y1 = θ (-1.5) …… (23) Applying unit step function on equation 23, Therefore, y1=0

Case 2.

Consider x1 = 0, x2 = 1 and substituting in equation 21 y1 = θ ((1*0) + (1*1) + (-1.5))

y1= θ (-0.5) …… (24) Applying unit step function on equation 24, Therefore, y1y=0

Case 3.

Consider x1 = 1, x2 = 0 and substituting in equation 21 y1 = θ ((1*1) + (1*0) + (-1.5))

y1= θ (-0.5) …… (25) Applying unit step function on equation 25, Therefore, y1=0

Case 4.

Consider x1 = 1, x2 = 1 and substituting in equation 21 y1 = θ ((1*1) + (1*1) + (-1.5))

y1 = θ (0.5) …… (26) Applying unit step function on equation 26, Therefore, y1=1

For equation 22, Case 1.

Consider x1 = 0, x2 = 0 and substituting in equation 22 y2 = θ ((1*0) + (1*0) + (-0.5))

y2 = θ (-0.5) …… (27) Applying unit step function on equation 27, Therefore, y2=0

Case 2.

Consider x1 = 0, x2 = 1 and substituting in equation 22 y2 = θ ((1*0) + (1*1) + (-0.5))

y2 = θ (0.5) …… (28) Applying unit step function on equation 28, Therefore, y2=1

Case 3.

Consider x1 = 1, x2 = 0 and substituting in equation 22 y2 = θ ((1*1) + (1*0) + (-0.5))

y2 = θ (0.5) …… (29) Applying unit step function on equation 29, Therefore, y2=1

Case 4.

Consider x1 = 1, x2 = 1 and substituting in equation 22 y2 = θ ((1*1) + (1*1) + (-0.5))

y2 = θ (1.5) …… (30) Applying unit step function on equation 30, Therefore, y2=1

Substituting wNOT, bNOT and y1 result in equation 23, y3= θ ((-1*y1) + (0.5)) ……... (31)

Case 1: Consider y1 = 0 and substituting in equation 31 y3 = θ ((-1*0) +(0.5))

y3 = θ (0.5) …… (32) Applying unit step function on equation 32, Therefore, y3=1

Case 2:Consider y1 = 0 and substituting in equation 31 y3 = θ ((-1*0) +(0.5))

y3 = θ (0.5) …… (33) Applying unit step function on equation 33, Therefore, y3=1

Case 3:Consider y1 = 0 and substituting in equation 31 y3 = θ ((-1*0) +(0.5))

y3 = θ (0.5) …… (34) Applying unit step function on equation 34, Therefore, y3=1

Case 4:Consider x = 1 and substituting in equation 31 y3 = θ ((-1*1) + (0.5))

y3 = θ (-0.5) …… (35) Applying unit step function on equation 35, Therefore, y3=0
Substituting wAND1, wAND2, bAND in equation 24 y = θ((1*y3) +(1*y2) +(-1.5)) …….. (36)
substituting y3 & y2 result in equation 36, Case 1.
Consider y2=0, y3 = 1 and substituting in equation 36 y = θ ((1*1) + (1*0) + (-1.5))
y = θ (-0.5) …… (37) Applying unit step function on equation 37, Therefore, y=0
Case 2: Consider y2=1, y3 = 1 and substituting in equation 36 y = θ ((1*1) + (1*1) + (-1.5))
y = θ (0.5) …… (38) Applying unit step function on equation 38, Therefore, y=1
Case 3: Consider y2=1, y3 = 1 and substituting in equation 36 y = θ ((1*1) + (1*1) + (-1.5))
y = θ (0.5) …… (39) Applying unit step function on equation 39, Therefore, y=1
Case 4: Consider y2=1, y3 = 0 and substituting in equation 36 y = θ ((1*0) + (1*1) + (-1.5))
y = θ (-0.5) …… (40) Applying unit step function on equation 40, Therefore, y=0

## 4. Results and discussion

The perceptron algorithm for logic gates have been written in Verilog HDL The results are furnished below

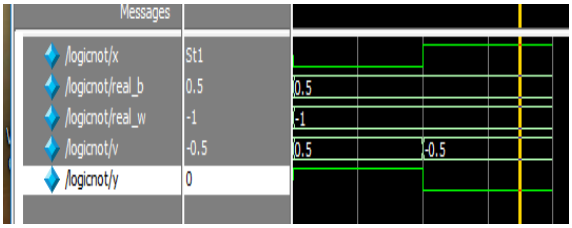



Figure 7. Simulation for NOT Gate

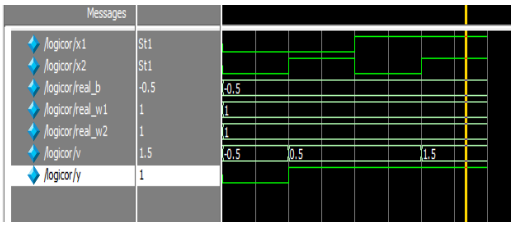



Figure 8. Simulation for OR Gate

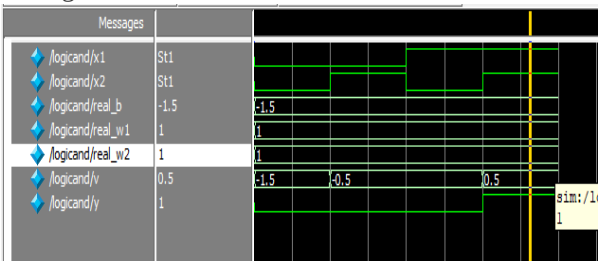



Figure 9. Simulation for AND Gate

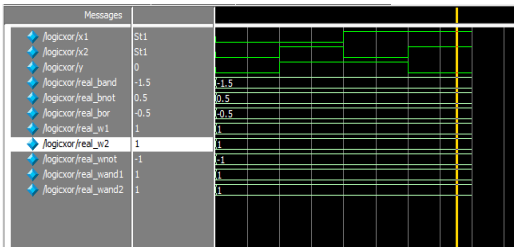



Figure 10. Simulation for XOR Gate

## 5. Conclusion

Here, the model predicted output (ÿ) for each of the test inputs are exactly matched with the logic gates conventional output (y) according to the truth table. The code is written in Verilog and successfully simulated using ModelSim-Altera 6.4a (Quartus II 9.0) Starter Edition. Hence, it is verified that the perceptron algorithm for logic gates is correctly implemented.

Future scope : Full subtractor can be build.

## References

1. [1]. Neelu Farha1 ., Ann Louisa Paul ., Naadiya Kousar L ., Devika ., Prof Ruckmani Divakaran5 .“Design and Implementation of Logic Gates and Adder Circuits on FPGA Using ANN".International Journal for search in Applied Science & Engineering Technology,Volume 4 Issue V, May 2016
2. Bharath Rao Madela, V. Yaswanth Siva Sai “Design and Implementation of Logic Gates using Artificial Neural Networks on FPGA”. International Journal of Scientific Research in Computer Science, Engineering and Information Technology © 2017 IJSRCSEIT | Volume 2 | Issue 5 | ISSN : 2456-3307.
3. Aydoğan Savran, Serkan Ünsal. “Hardware implementation of a feedforward neural network using fpgas”. Ege, Department of Electrical and Electronics Engineering, 2003.
4. Emmanuel adetiba , F.A. Ibikunle , S.A. Daramola , A.T. Olajide. “Implementation of Efficient Multilayer Perceptron ANN Neurons on Field Programmable Gate Array Chip”. International Journal of Engineering & Technology IJET-IJENS Vol:14 No:01

5. Leila Dehbozorgi, Reza Akbari-Hasanjani, Reza Sabbaghi-Nadooshan."New Full Adders Using Multi-Layer Perceptron Network".International Journal of Smart Electrical Engineering, Vol.8, No.3, Summer 2019.
6. C.R.Kavitha "Neural Network In Verbal Brain Arithmetic Operations With Neural Network" 3rd Inernational Conference On Recent Innovations In Science Engineering And Management.
7. Catherine D. Schuman, Member, IEEE, Thomas E. Potok, Member, IEEE, Robert M. Patton, Member, IEEE, J. Douglas Birdwell, Fellow, IEEE, Mark E. Dean, Fellow, IEEE, Garrett S. Rose, Member, IEEE, and James S. Plank, Member, IEEE. "A Survey of Neuromorphic Computing and Neural Networks in Hardware".
8. COMPOSITE VENDOR RATING MANAGEMENT, DN. Amirdhan, B. Dinesh Kumar, International Journal Of Advance Research In Science And Engineering http://www.ijarse.com IJARSE, Volume No. 10, Issue No. 04, April 2021 ISSN-2319-8354(E).
9. Itay Hubara, Matthieu Courbariau "Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations" Journal of Machine Learning Research 18 (2018) 1-30.
10. Philippe, Carvalho, Gardere "Implementation of a Feed-forward Artificial Neural Network in VHDL on FPGA" symposium on Neural Network Applications in Electrical Engineering(NEUREL) 2014.
11. Yadagiri, Prof. Misra "Implementation of 32 Bit Floating Point MAC Unit to Feed Weighted Inputs to Neural Networks" IJRSI Volume II, Issue IV, April 2015 PP 40- 43.