E-ISSN 2581 – 7957
P-ISSN 2277 – 3916

CVR Journal of Science and Technology, Volume 20, June 2021
DOI: 10.32377/cvrjst2009

FPGA Realization of Logic Gates using Neural Networks

R. Ganesh[1] and D. Bhanu Prakash[2]
[1]Assoc. Professor, CVR College of Engineering /ECE Department, Hyderabad, India
Email: rachaganesh@gmail.com
[2]Assoc. Professor, CVR College of Engineering /ECE Department, Hyderabad, India
Email: pbhanududi@gmail.com

Abstract: **The real time systems are designed by using analog and digital sub systems with control logic. In the design of digital sub systems the logic gates are the major building blocks. The Neural Networks found many applications in the areas of pattern recognition, prediction, artificial intelligence and other applications. Most of the Neural Network designs are realized by using software domain and hardware domains. This paper presents the hardware realization of logic gates using Neural Network.**

The logic gates like NAND, AND, NOR, OR, NOT, BUFFER are realized by using single layer neural network. The XNOR and XOR logic gates are realized by using structural representation of neural network based universal NAND gates. All the logic gates are realized by using Verilog HDL with Xilinx Vivado Design suite targeting for Xilinx Zynq-7000 SoC Evaluation Board. The Neural Network based logic gates are simulated for different test cases.

Index Terms: **Neural Network, Logic Gate, Zynq-7000 SoC, Verilog HDL, Xilinx Vivado EDA tool.**

I. INTRODUCTION

The present modern real time systems are designed by using Digital, Analog, Mixed and co-design subsystems, to meet non functional constraints like area, speed, power, cost and other parameters. In all these subsystems the digital design plays an important role in meeting the non functional constraints.

The Neural Networks (NN) found many applications in the areas of pattern recognition, prediction, artificial intelligence, and other applications [1]. Most of the Neural Network designs are realized by using software domain with mainly concentrating on the application input, outputs and not on the internal structure of the design. This application oriented design of software realization has an advantage of simple coding, low cost with a limitation of slower execution time.

The limitation of software realization of NN can be overcome by using hardwar realization using Microprocessor, Digital Signal Processing (DSP) and Very Large Scale Integration (VLSI). The Neural Network design requires lot of parallel computations. The Microprocessors and DSP are not suitable for parallel execution of NN systems. Hence, to design Neural Network systems the VLSI is considered as the best method for parallelism [2].

The design approaches for VLSI systems is shown in Figure 1. The design of VLSI systems can be divided into three sub system domains. i.e. Digital VLSI system domain, Analog VLSI system domain and Mixed VLSI system domains [3].

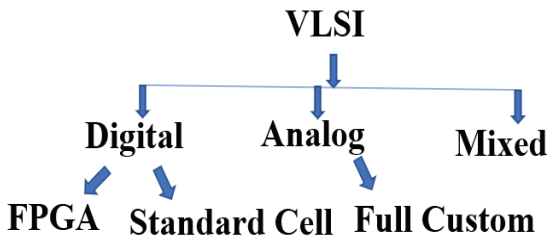


Figure 1. Design approaches in VLSI Design

The digital VLSI systems are realized by using textual hardware languages like Very High Speed Integrated Circuit Hardware Description Language (VHDL), Verilog HDL, System Verilog etc. The Analog VLSI systems are realized by using transistors based schematic representations. The mixed VLSI systems are realized by using the combination of both Analog and Digital design entry representations. The Digital VLSI is divided into FPGA and standard cell-based designs. The Field Programmable Gate Array (FPGA) based design is having the feature of re-programmability, whereas the standard cell-based VLSI require the foundry based standard cell library and it is not having the feature of re-programmability. The Analog VLSI systems using transistors is done by using a full custom based approach, which does not use any HDL coding and re-programmability.

The FPGA based designs are well suited for the realization of digital design systems due to its flexibility of logic gates realization, HDL entry and re-programmability features. The logic gates are the major building blocks in the realization of digital design system. All these logic gates i.e. NAND, AND, NOR, OR, XNOR, XOR, NOT and BUFFER can be realized by using different design methodologies based on the targeted application. The design methodologies are selected to meet the logic functionality equation and truth table of the corresponding logic gate. All these logic gates are used as the basic building blocks for realizing the complex system.

This paper presents realization of neural network based logic gates i.e. NAND, AND, NOR, OR, XNOR, XOR, NOT and BUFFER using FPGA. The chapter II presents the concepts of Neural Networks along with the types of Neural Network architectures. The chapter III presents design methodology of Neural Networks based logic gates. The chapter IV gives the design and realization of NN based logic gates; the chapter V gives the simulation results. The conclusion is presented in chapter VI followed by references.

CVR Journal of Science and Technology, Volume 20, June 2021
DOI: 10.32377/cvrjst2009

II. Neural Networks And Its Architectures

The Artificial Neural Networks are inspired by the biological neural systems. The characteristics of any system sub blocks are modelled by using mathematical representation model. This mathematical system model is realized by using Neural network which contains multiple layers using neurons [4]. The block diagram of single neuron is shown in Figure 2.

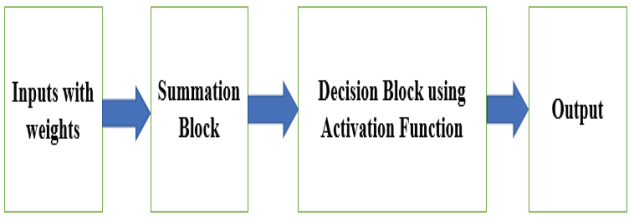


Figure 2. Block diagram of Single Neuron

The single neuron consists of inputs with weights, summation block, activation function with threshold to generate output. The input consists of one or more values to generate the single output. This output is computed by using weighted sum of inputs and the threshold logic calculate the output value for the selected application.

The Artificial Neural Networks are designed by using neuron connection network architectures using neurons. These architectures are classified into five types and the selection of the network architecture is based on the type of the application, design algorithm and other design parameters [5]. The types of ANN are:

3. Single-layer feed forward network
4. Multilayer feed forward network
5. Single node with its own feedback
6. Single-layer recurrent network
7. Multilayer recurrent network

The single layer feed forward network consists of only two layers i.e. input layer and output layer. The Multilayer feed forward network consists of input layer, output layer and one or more hidden layers to enable the network to be computationally stronger [6].

The Single node with its own feedback network outputs are directed back to its inputs to the same layer or preceding layer nodes, which results in feedback networks. The Single-layer recurrent network is having feedback connection in which processing element's output can be directed back to itself or to other processing element or both. In the Multilayer recurrent network, the processing element output can be directed to the processing element in the same layer and in the preceding layer forming a multilayer recurrent network.

The design of logic gates using ANN is done by using the Single-layer feed forward network. This network has input and output layers along with weighted sum of inputs and threshold, this helps in making the decision for generating the output.

III. Neural Networks Based Logic Gates

The general structure of logic gate and its corresponding Neural Network is shown in Figure 3.

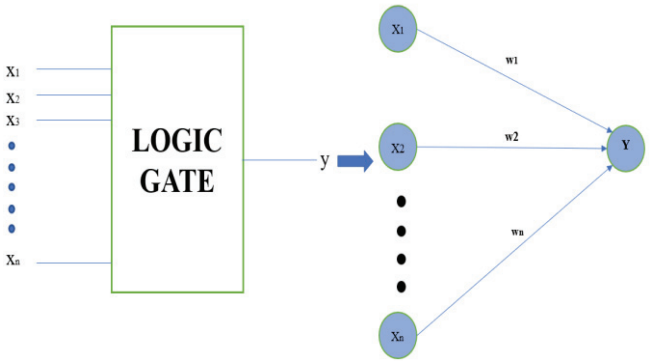


Figure 3. N-input logic gates and its Neural network

The N-input logic gate is realized by using x1,x2,x3, …., xn inputs along with its corresponding weights w1,w2,w3, ….., wn using the threshold is shown in mathematical equation (1).

$$y = \begin{cases} 1, & \text{if } \Sigma\, w_i x_i > \text{threshold} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

In this approach, the inputs are also treated as neurons and the output is chosen as the actual value of the feature. The above equation is rewritten by moving the threshold to other side of the equation and these "-threshold" values are used as the bias values. The mathematical equation is shown in equation (2).

$$y = \begin{cases} 1, & \text{if } \Sigma\, w_i x_i + b > 0 \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

All the basic logic gates are designed by using a single structure with two inputs and different output for different gates as shown in Figure 4. The individual logic gates i.e. NAND, AND, NOR, OR, XNOR, XOR, BUFFER and NOT gates with their corresponding symbols and truth tables are shown in Figure 5.

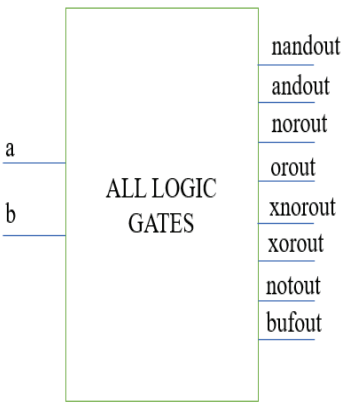
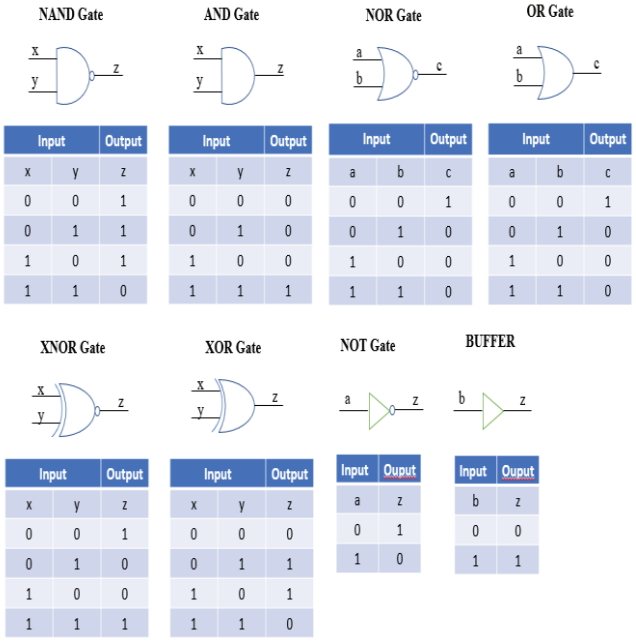


Figure 4. Block diagram of all logic gates

CVR Journal of Science and Technology, Volume 20, June 2021
DOI: 10.32377/cvrjst2009



Input		Output
x	y	z
0	0	1
0	1	1
1	0	1
1	1	0

Input		Output
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

Input		Output
a	b	c
0	0	1
0	1	0
1	0	0
1	1	0

Input		Output
a	b	c
0	0	1
0	1	0
1	0	0
1	1	0

Input		Output
x	y	z
0	0	1
0	1	0
1	0	0
1	1	0

Input		Output
x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

Input	Ouput
a	z
0	1
1	0

Input	Ouput
b	z
0	0
1	1

Figure 5. Symbols and truth tables of all logic gates

The neural network based output computation equations for single term Boolean equation uses the values of a and b as inputs for logic gates. The nandout, andout, norout, orout, notout and bufout are the output values for the NAND, AND, NOR, OR, NOT and buffer logic gates respectively. The weights of the logic gates are updated by using Perceptron algorithm, which is as follows;

Step-1: Initialize weight values and bias with random numbers
Step-2: Forward Propagate
Step-3: Calculate the error
Step-4: Update the weights and bias
Step-5: Repeat for all input values

The error is calculated as the difference between the calculated output from step-2 and the actual desired output. The weights are updated by using the following equations (3) and (4).

$$W_i = W_i + d(W_i) \quad (3)$$
$$d(W_i) = \text{Error} * X_i \quad (4)$$

Where, X is input to the logic gate and 'i' indicates the number of inputs for logic gate i.e. 1,2,3 …….n.

The output values for all logic gates are done by using the [w1*a+w2*b+w3*c+…….+bias] neural network computational equation-2 by using decision making threshold logic [7]. This neural network computation equation can be rearranged as a straight-line equation is used to classify the digital logic gates outputs.

Let us consider an equation w1*a+w2*b+bias=0 which can be rearranged as b=-(w1/w2)*a-bias/w2. This equation looks like y=mx+c. Here m=-(w1/w2) and c=-(bias/w2). Let us consider two input AND gate with a,b as inputs and out as output. The output is logic-1 only when a and b are equals to '1'. In other cases the output is '0'. The Neural network can classify these conditions by using the straight line equation as shown in Figure 6.

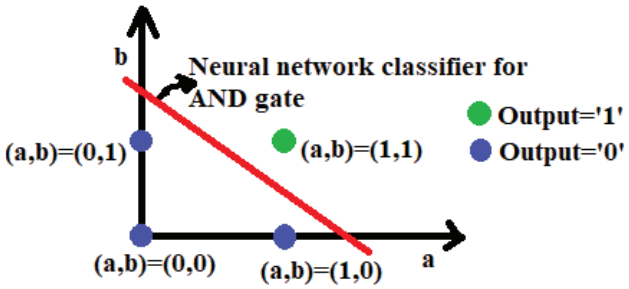


Figure 6. Neural Network Classifier for AND gate

The neural network-based output computations for all logic gates are calculated by using Perceptron algorithm for weights and Neural network classifier for output logic decision are as shown below.

nandout=-2a-2b+3
andout=2a+2b-3
norout=-2a-2b+1
orout=2a+2b-1
notout=-2a+1
bufout=2b-1

The output is logic-1 when the value of computation is large or positive. Similarly, the output is logic-0 when the value of computation is small or negative.

All the logic gates are analysed with the computation equations and neurons are shown in Figure 7. The NAND gate design has weights of -2,-2 and a threshold bias value of 3 to meet the requirements of its truth table.

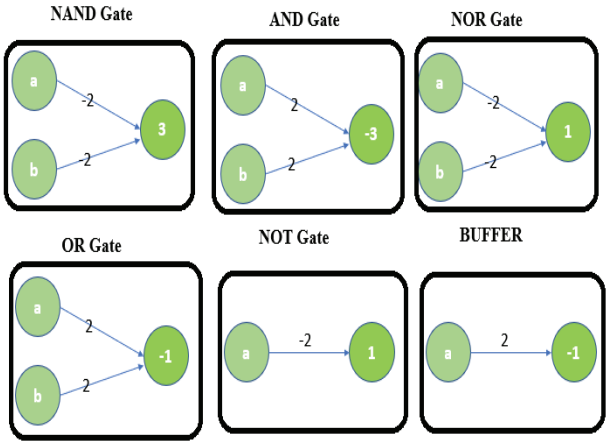


Figure 7. All logic gates using ANN

The AND gate design has weights of 2, 2 and a threshold bias value of -3 to meet the requirements of its truth table. The NOR gate design has weights of -2, -2 and a threshold bias value of 1 to meet the requirements of its truth table. The OR gate design has weights of 2, 2 and a threshold bias value of -1 to meet the requirements of its truth table. The NOT gate design has weight of -2 and a threshold bias value of 1 to meet the requirements of its truth table. The BUFFER gate design has weight of 2 and a threshold bias value of -1 to meet the requirements of its truth table.

The XNOR and XOR logic gates with inputs a and b are having the boolean equations with more than one product and addition terms to generate the xnorout and xorout as shown in equations (5) and (6).

xnorout= ab+a'b' (5)

xorout= ab'+a'b (6)

The above equations 1 and 2 are having two multiplications and one addition terms with direct and inverted input literals. The direct neural network implementation of XNOR and XOR logic gates needs the multilayer feed forward network due to its nature of Boolean equation [8]. Hence, the logic gates XNOR and XOR can be designed by using universal logic gates i.e. NAND and NOR. The NAND logic gate based XNOR and XOR structural designs are shown in Figure 8 and Figure 9 respectively. The neural network based NAND gate is used in realizing the XNOR and XOR gates.

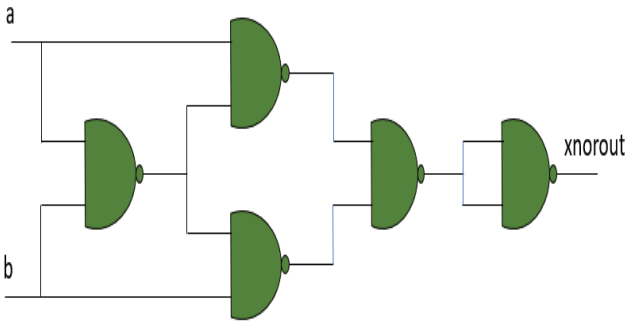


Figure 8. XNOR Gate using structural ANN NAND gates

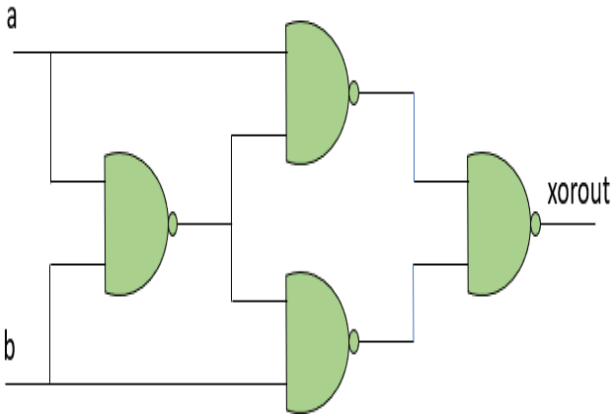


Figure 9. XOR Gate using structural ANN NAND gates

All the logic gates using neural networks and structural gate level circuits are designed by using Verilog HDL and Xilinx FPGA. In this paper, the design and realization of neural network-based logic gates are done by using FPGA.

IV. FPGA Realization Results

All the logic gates using neural networks are designed and realized by using Verilog HDL targeted for Xilinx Zed Board Zynq Evaluation and Development Kit (xc7z020clg484-1). All the logic gates are synthesized by using the Xilinx Vivado EDA tool.The summary of cells utilized and synthesis result for the unified single design with all logic gates is shown in Figure 10.

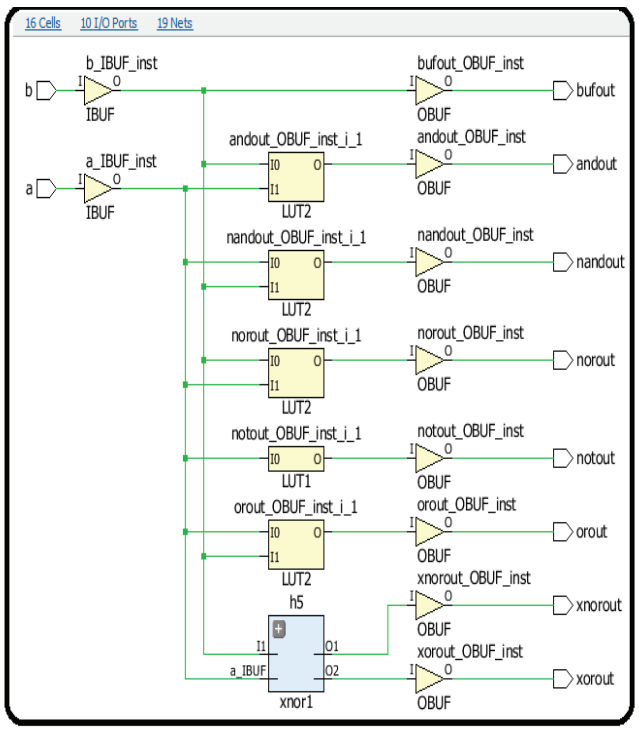


Figure 10. Synthesis Result of Neural Network based all logic gates

The unified all logic gates design is synthesized with 16 cells, 10 I/O ports and 19 nets. This design resulted in input signal as a, b and outputs for all logic gates as bufout, andout, nandout, norout, notout, orout, xnorout, xorout signals as shown in Figure 10.

V. Simulation Results

The simulation results of Neural Network based all logic gates i.e. NAND, AND, NOR, OR, XNOR, XOR, BUFFER and NOT using Verilog HDL with Xilinx Vivado EDA tool are done by targeting Zynq FPGA evaluation board.

The simulation result of two input NAND gate with 'a', 'b' inputs and one output as 'nandout' signal is shown in Figure 11 for all test vectors.

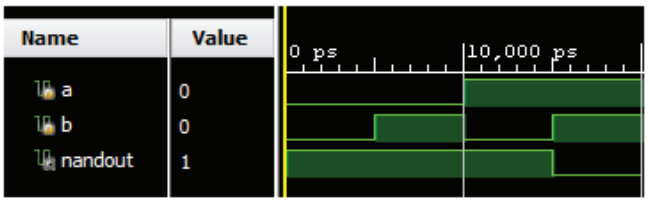


Figure 11. Simulation result of NAND Gate

The simulation result of two input AND gate with 'a', 'b' inputs and one output as 'andout' signal is shown in Figure 12 for all test vectors.

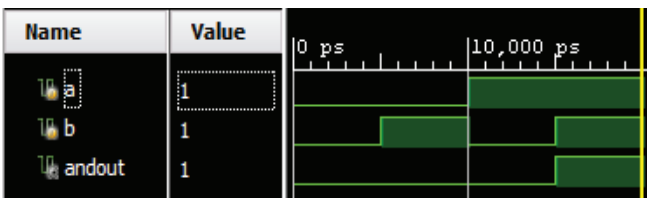


Figure 12. Simulation result of AND Gate

E-ISSN 2581 – 7957
P-ISSN 2277 – 3916

DOI: 10.32377/cvrjst2009

The simulation result of two input NOR gate with ‘a’, ‘b’ inputs and one output as ‘norout’ signal is shown in Figure 13 for all test vectors.

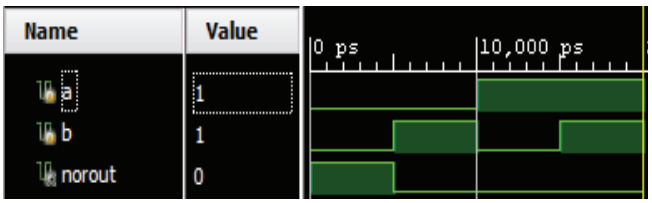


Figure 13. Simulation result of NOR Gate

The simulation result of two input OR gate with ‘a’, ‘b’ inputs and one output as ‘orout’ signal is shown in Figure 14 for all test vectors.

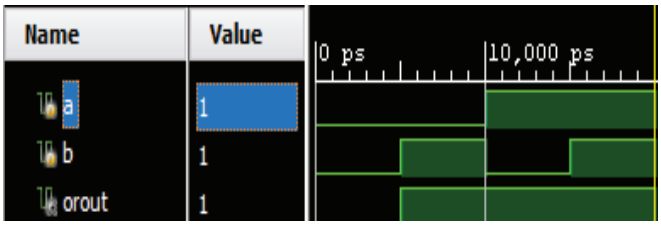


Figure 14. Simulation result of OR Gate

The simulation result of two input XNOR gate with ‘a’, ‘b’ inputs and one output as ‘xnorout’ signal is shown in Figure 15 for all test vectors.

Figure 15. Simulation result of XNOR Gate

The simulation result of two input XOR gate with ‘a’, ‘b’ inputs and one output as ‘xorout’ signal is shown in Figure 16 for all test vectors.

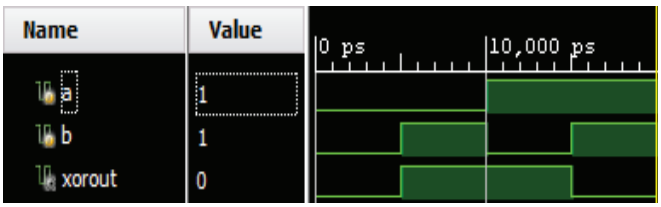


Figure 16. Simulation result of XOR Gate

The simulation result of NOT gate with ‘a’ as input and ‘notout’ as output signal is shown in Figure 17 for all test vectors.

Figure 17. Simulation result of NOT Gate

The simulation result of BUFFER with ‘b’ as input and ‘bufout’ as output signal is shown in Figure 18 for all test vectors.

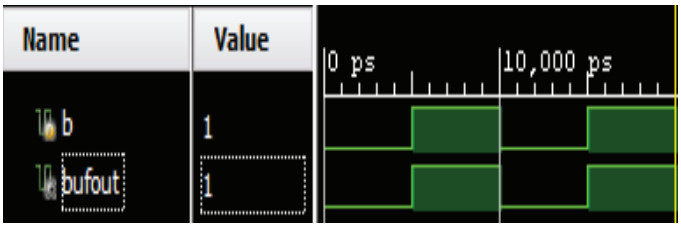


Figure 18. Simulation result of Buffer

The simulation result of unified structure for Neural Network based all logic gate with ‘a’, ‘b’ as inputs and ‘nandout’, ‘andout’, ‘norout’, ‘orout’, ‘xnorout’, ‘xorout’, ‘notout’, ‘bufout’ is shown in Figure 19 for all test vectors.

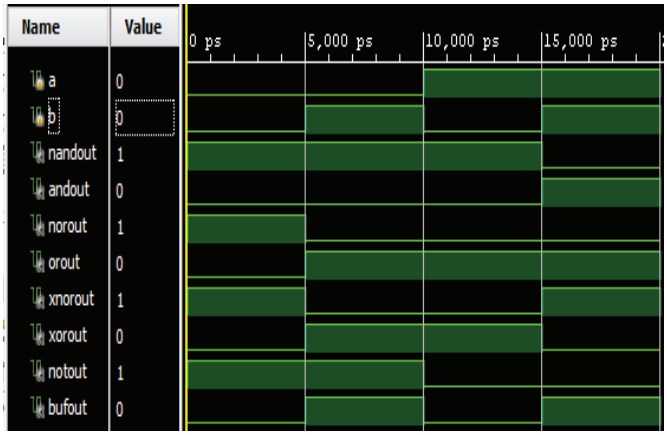


Figure 19. Simulation result of all logic gates

VI. Conclusions

All the logic gates are realized by using Verilog HDL with Xilinx Vivado Design suite targeting for Xilinx Zynq-7000 SoC Evaluation Board.

The logic gates NAND, AND, NOR, OR, NOT, BUFFER are realized by using single layer neural network with computational equations using Verilog HDL. The XNOR and XOR logic gates are realized by using structural representation of neural network based universal NAND gates. All these logic gates are simulated for different test cases.

A unified structure for Neural Network based all logic gates is designed and synthesized using Xilinx Vivado EDA tool. This unified logic gate structure is simulated for different test cases.

REFERENCES

[1]. O. I. Abiodun *et al*., "Comprehensive Review of Artificial Neural Network Applications to Pattern Recognition," in IEEE Access, vol. 7, pp. 158820-158846, 2019.
[2]. Haitham Kareem Ali and Esraa Zeki Mohammed, " Design Artificial Neural Network Using FPGA", IJCSNS International Journal of Computer Science and Network Security, Vol.10, No.8, August 2010.
[3]. Ganesh.R, "Design Procedure for Digital and Analog ICs using Cadence Tools", CVR Journal of Science and Technology, Volume 9, December 2015.
[4]. A. Muthuramalingam, S. Himavathi, E. Srinivasan, " Neural Network Implementation Using FPGA: Issues and Application", International Journal of Information Technology, Vol. 4 Issue 2, p86, 2008.
[5]. Murat H. Sazli, "A Brief Review Of Feed-Forward Neural Networks", Commun. Fac. Sci. Univ. Ank. Series A2-A3, V.50(1), pp 11-17 (2006).
[6]. Crescenzio Gallo, "Artificial Neural Networks Tutorial", January 2015.
[7]. F. Rosenblatt, "The perceptron: a probabilistic model for Information storage and organization in the brain", Psychological Review, Vol. 65, No. 6, 1958.
[8]. Blount, D., Banda, P., Teuscher, C., & Stefanovic, D. (2017), "Feedforward Chemical Neural Network: An In Silico Chemical System That Learns XOR. Artificial Life", Volume 23, Issue 3, p. 295-317.