New Logic Gates Using Neural Network

Reza Akbari-Hasanjani, Leila Dehbozorgi, Reza Sabbaghi-Nadooshan*

Department of Electrical Eng., Islamic Azad University, Central Tehran Branch, Tehran, Iran

Abstract

The present study is to investigate and design the logic gates and half adder circuits by using multilayer neural network. The parallel function of the neural networks allows their application in designing high-speed circuits. DSP and FPGA can be used in implementation of these circuits, which reduces the area of the circuit. This study first considers logic gates, and since half adder circuits are the basic systems in computing, a half adder circuit is designed in this study. To design a full adder circuit, two half adders and an OR gate can be used. The results of this study are consistent with the results of gates designed with other technologies such as CMOS and TTL, except that neural networks use less power.

Keywords: neural network, FPGA and DSP, area
Article history: Received 04-Mar-2020; Revised 21-Mar-2020; Accepted 06-Jun-2020.

1. Introduction

In recent decades, neural networks have gained a wider application in the design and optimization of digital circuits. Artificial neural networks are inspired by biological structures in brain and neurons. Each neural network is composed of a number of neurons [1]. Mcculloch-Pitt first introduced artificial neural neurons in 1943 by combining mathematical logic and biological neurons [2]. Artificial neural networks have simplified and increased the response speed in systems due to parallel data processing.

Each neural network consists of a number of layers, including the input layer, the hidden layer, and the output layer. Each layer consists of a neural network of a number of artificial neurons [2]. Each neuron has a threshold, a weight (W), and a bias (b) [2].

Some of the important applications of the neural network are recognition of pattern, diagnosis and identification of human speech, conversion of black and white images to color images, prediction of earthquakes [3-8].

In this paper, logic gates and half and full adders have been designed using multilayer (ML) neural network with constant weights and zero bias and simulations were made along with coding in MATLAB 2017.

A) Biological neurons and artificial neurons

Human brain is made up of cells called neurons. There are a total of 10^{11} neurons which are interconnected with 10^4 connections per neuron [3]. Each neuron is composed of dendrites, soma (cell body), axons and synapses [3]. Dendrites are made up of nerve filaments that transmit electrical signals to the soma. Soma receives signals and applies a threshold value on them. Ultimately, the axon, which is a long nerve filament, carries these signals from the body to other neurons [4, 3]. Synapse is the junction between the axons of one nerve cell and dendrites of other neurons. Fig. 1 shows a biological neuron.

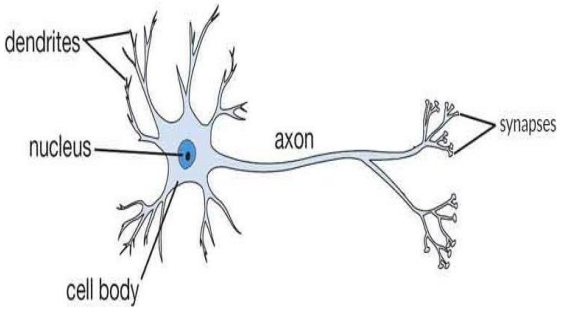

Fig. 1. Biological neurons

B) Artificial neuron structure

The structure of artificial neurons is similar to that of biological neurons, with several inputs and only one output [1]. Each artificial neuron contains inputs, neurons, and bias and threshold values. The equation of the output of each artificial neuron is a follows:

$$F = a_i w_i + b_i \quad (1)$$

Where, B_i is the value of bias, w_i is the weight and x_i is the input value.

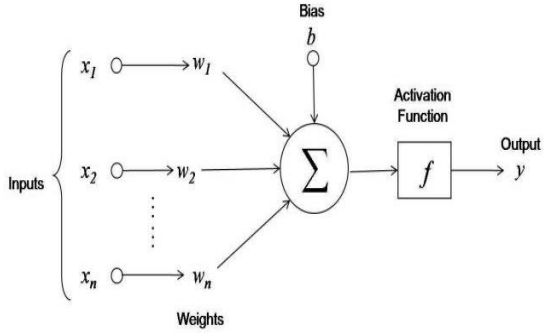

Fig. 2. Artificial neurons

This paper is to design logic gates and digital adder circuits using a multilayer network. Matlab 2017 has been used in this paper for simulation [9-11].

2. Multilayer network

The artificial neural network is a parallel processor that functions as a biological brain [3, 5], and is able to process the data. However, the artificial neural network cannot function at the complexity level of our brain. In designing a multilayer network, two or more artificial neurons can be combined to obtain the multilayer network. Multiple layers of artificial neurons can be designed together in one layer to finally connect the layers together and obtain a multilayer network.

In multilayer networks, each layer has a weight matrix, bias vectors and one or more outputs [6, 7]. In these networks, the output of each layer is the input of the next layer and the last layer is the network output [12-13].

3. Design and simulation of logic gates

In this section, using the multilayer neural network, the common logic gates of AND, OR, NOT and XOR will be discussed. A topology is used for the design of logic gates. The weights of the neural network modified in the design each logic gate. Fig. 3 illustrates the network topology. In the designed networks, the input is in the binary form of zeros and ones. In all arrays, the bias of each neuron is set to zero.

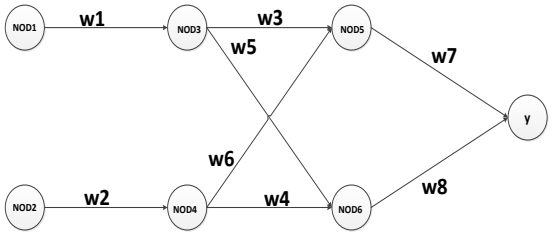

Fig. 3. Topology of the proposed neural network

A) AND gate

In this section, the two-input AND gate is designed, which is a base gate. Eq. 2 represents the mathematical relation of the AND gate. In the AND gate the output equals one when both inputs equal one, otherwise the output is zero. The multilayer network is used in the design of the AND gate.

$$Y=X_1.X_2 \quad (2)$$

Fig. 3 shows the AND gate using a neural network, where $W_1=W_2=W_7=W_8=1$ and $W_3=W_4=W_5=W_6=0.5$. X1 and X2 are inputs to the network. And the threshold value in neurons is $th_1=th_2=th_3=th_4=0.9$ and $th_5=1.9$. From Figure 3, Eq. 3 and 4 will be.

Value of NOD1=$x_1.w_1$ if value of NOD1>th_1 then Output of NOD1 is1 else output of NOD1 is 0 (3)

Value of NOD2=x2.w2 if value of NOD2>th2 then Output of NOD2 is1 else output of NOD2 is 0 (4)

If the values of NOD1 and NOD2 are greater than the threshold values of th_1 and th_2, the neuron will be excited and the output of the neuron will equal one. The outputs in the first and second nodes are shown in Table 1.

Table.1.
Output of the input layer

X_1	X_2	Value Of NOD1	Value Of NOD2	Out put Of NOD1	Out put Of NOD2
0	0	0	0	0	0
0	1	0	1	0	1
1	0	1	0	1	0
1	1	1	1	1	1

The output of the first layer will be the input for the second layer, if the values of NOD3 and NOD4 are greater than th_3 and th_4, the output will equal one. Table 2 shows the output of the second layer, and using Eq. 5 and 6 we have:

Value of NOD3=NOD1.w_3+NOD2.w_6 if value of NOD3>th_3 then Output of NOD3 is1 else output of NOD3 is 0 (5)

Value of NOD4=NOD1.w_5+NOD2.w_4 if value of NOD3>th_4 then Output of NOD4 is1 else output of NOD4 is 0 (6)

Table.2.
Second layer output

Input of layer1		***Values of NOD3 and NOD4***		***If nod3>th_3→out nod3=1 If nod4>th_4→out nod4=1***	
Nod 1	Nod 2	Value Nod3	Value Nod4	Out put Nod3	Out put Nod4
0	0	0	0	0	0
0	1	0.5	0.5	0	0
1	0	0.5	0.5	0	0
1	1	1	1	1	1

According to Table 2, the output of the second layer equals one when the values of NOD3 and NOD4 are greater than th_3 and th_4, and this occurs when the input of the second layer and the first layer both equal one.

In the output layer, the second layer is the input of the final layer and the threshold value of the output layer is 1.9. Eq. 7 shows the relation for the AND gate output:

Value of y= NOD3.w_7+ NOD4.w_8 if value of y >1.9 Then output y is 1 else output is 0 (7)

Table.3.
Output layer

Input for NOD y		***Value of y***	***Out put Y***
NOD3	***NOD4***		
0	0	0	0
0	0	0	0
0	0	0	0
1	1	2	1

According to Table 3, the AND gate has the same response as other technologies such as CMOS and TTL.

B) OR gate

In this section, the design of the two-input OR gate is described which is a base gate and its design relation is derived from Eq. 8. The OR gate output equals one when at least one of the inputs equals one, otherwise the output equals zero.

$y=X_1+X_2$ (8)

In the design of the OR gate, the multilayer neural network topology is used, in which the bias of the neurons is zero, the weights are $W_1=W_2=W_7=W_8=1$ and $W_3=W_4=W_5=W_6=0.5$, and the thresholds are $th_1=th_2=0.9$, $th_3=th_4=0.4$ and $th_5=0.9$.

Considering the OR gate neural network, equations 9-12 are calculated for each layer. In the first layer, Eq. 9 and 10 is follows:

Value of NOD1=x_1.w_1 if value of NOD1> th_1 then Output of NOD1 is1 else output of NOD1 is 0 (9)

Value of NOD2=x_2.w_2 if value of NOD2> th_2 then Output of NOD2 is1 else output of NOD2 is 0 (10)

In Table 4, the output of the first layer is seen at inputs x1 and x2.

Table.4.
Output of the first layer

X_1	***X_2***	***Value Of NOD1***	***Value Of NOD2***	***Out put Of NOD1***	***Out put Of NOD2***
0	0	0	0	0	0
0	1	0	1	0	1
1	0	1	0	1	0
1	1	1	1	1	1

The output of the first layer will be the input to the second layer. If the NOD3 and NOD4 are greater than the thresholds of th_3 and th_4, the output of each NOD will equal one. Using Eq. 11 and 12 in the second layer and Table 5, the values of the output layer of the second layer of the neural network were designed.

Value of NOD3=NOD1.w_3+NOD2.w_6 if value of NOD3>th_3 then Output of NOD3 is1 else output of NOD3 is 0 (11)

Value of NOD4=NOD1.w_5+NOD2.w_4 if value of NOD4>th_4 then Output of NOD4 is1 else output of NOD4 is 0 (12)

Table.5.
Outputs of the second layer

In put		***Value of NOD3***	***Value of NOD4***	***Out put Of NOD3***	***Out put Of NOD4***
NOD1	***NOD2***				
0	0	0	0	0	0
0	1	0.5	0.5	1	1
1	0	0.5	0.5	1	1
1	1	1	1	1	1

In the design network, the output of the second layer is designed as the input to the output layer and the threshold value of the output layer is 0.9. In Table 6, the outputs of the designed gate are observed and Eq. 13 is related to the OR gate output.

According to Table 6, the results obtained here correspond to the results of the OR gate designed with TTL and CMOS technologies.

Value of y= NOD3.w_7+ NOD4.w_8 if value of y>0.9 Then output y is 1 else output is 0 (13)

Table.6.
OR gate output

NOD3	NOD4	Value of y	Out put Y
0	0	0	0
1	1	2	1
1	1	2	1
1	1	2	1

C) NOT gate

This section presents the design of the NOT gate using the proposed multilayer network. Eq. 14 represents the mathematical relation of the NOT gate and its output is the inverse of the gate input.

$$y = \bar{x} \tag{14}$$

The NOT gate was designed using the proposed topology, except that the two inputs were connected. In this scheme W1=W2=W3 = W4 = W5 = W6 = W7 = W8 = 1 for weights and th5 = 0.5, th1 = th2 = th3 = th4 = 0.9 for the thresholds.

Considering the neural form provided for the NOT gate will has the equations 15-19 for layers in the NOT gate. Eq. 15 and 16 are related to the first layer. Table 7 represents the output of the first layer in the NOT gate.

Value of NOD1=x.w_1 if $NOD_1 > th_1$ then output of NOD1 is 1 else output of NOD1 is 0 (15)

Value of NOD2=x.w_2 if $NOD_2 > th_2$ then output of NOD2 is 1 else output of NOD2 is 0 (16)

Table.7.
Output of the first layer

X	Value of NoD1	Value of NoD2	Output of NoD1	Output of NoD2
0	0	0	0	0
1	1	1	1	1

The output of the first layer is the input to the second layer. If the values of NOD3 and NOD4 are greater than th_3 and th_4, the output of each node equals one. Using the Eq. 17 and 18 for the second layer as well as Table 8, the output values of the second layer in the NOT gate are as follows:

The output of the second layer is used as input to the output layer of the neural network and the threshold value of the output layer is 0.5. See Table 9 for the outputs of the designed NOT gate. Eq. 19 represents the output of the final layer in the designed network. Table 9 shows the NOT gate output and according to Table 9, the gate NOT output values correspond to the results of other technologies.

Value of NOD3=NOD1.w_3+NOD2.w_6 if value of NOD3>th_3 then Output of NOD3 is1 else output of NOD3 is 0 (17)

Value of NOD4=NOD1.w_5+NOD2.w_4 if value of NOD4>th_4 then Output of NOD4 is 1 else output of NOD4 is 0 (18)

Value of y= NOD3.w_7+ NOD4.w_8 if value of y<th_5 then output y is1 else out put y is 0 (19)

Table.8.
Output values of the second layer in the NOT gate

NOD1	NOD2	Value of NoD3	Value of NoD4	Output of NoD3	Output of NoD4
0	0	0	0	0	0
1	1	1	1	1	1

Table.9.
NOT gate output results

NOD3	NoD4	Value of y	Output of y
0	0	0	1
1	1	2	0

D) XOR gate

In this section, the two-input XOR gate is design using a multilayer artificial neural network. The mathematical relation of the XOR gate is as Eq. 20. In the XOR gate, the output equals one when the two inputs are opposite each other, otherwise the output is zero.

$$y = x_1 \oplus x_2 \tag{20}$$

In the design of the XOR gate, the proposed multilayer neural network topology is used. The bias values of the neurons equal zero.

In this design, the weights are W_1= W_2=1, W_3= W_4= W_5= W_6=0.5, W_7= W_8=0.6 and th1 = th2 =0.9, th3 =0.9, th4 = 0.4, th5 = 0.6 for the thresholds.

XOR gate is designed with neural network equations 21-25 are calculated for each layer. Eq.21 and 22 are the input layers of the XOR gate. Table 10 shows the output of the first layer.

The first layer output is used as the second layer input. If the NOD3 and NOD4 values are greater than th_5 and th_4, the output of each NOD will equal one. Eq. 23 and 24 belong to the middle layer of the XOR gate, and Table 11 shows the output values of the second layer in the XOR gate.

The output of the second layer is used as the input of the third layer in the XOR gate, and if y is between 0 and 1.1, the output is equal to one, otherwise it will be zero. Eq. 25 shows the value of y and Table 12 represents the output values of the third layer or output of the designed gate.

Value of y= NOD3.w_7+ NOD4.w_8 if 0<value of y<1/1 Then output y is 1 else output is 0 (21)

Value of NOD1=x_1.w_1 if value of NOD1>th_1 then Output of NOD1 is1 else output of NOD1 is 0 (22)

Value of NOD2=x_2.w_2 if value of NOD2>th_2 then Output of NOD2 is1 else output of NOD2 is 0 (23)

Value of NOD3= NOD1.w_3+ NOD2.w_6 if value of NOD3>th_3Then output NOD3 is1 else out put NOD3 is 0 (24)

Value of NOD4= NOD1.w_5+ NOD2.w_4 if value of NOD4>th_4Than output NOD4 is 1 else out put NOD4 is 0 (25)

Table.10.
Results of the first layer of the XOR gate

Output Of NOD1	*Output Of NOD2*	*Value of NOD1*	*Value of NOD2*	*Output Of NOD1*	*Output Of NOD2*
0	0	0	0	0	0
0	1	0	1	0	1
1	0	1	0	1	0
1	1	1	1	1	1

Table.11.
Output values of the second layer in the XOR gate

Output Of NOD1	*Output Of NOD2*	*Value of NOD3*	*Value of NOD4*	*Output Of NOD3*	*Output Of NOD4*
0	0	0	0	0	0
0	1	0.5	0.5	0	1
1	0	0.5	0.5	0	1
1	1	1	1	1	1

Table.12.
Output values of the third layer

Output of NOD3	*Output of NOD4*	*Value of y*	*Output of y*
0	0	0	0
0	1	0.6	1
0	1	0.6	1
1	1	1.2	0

According to Table 12, the output of y corresponds to the output of other technologies used for designing the XOR gate, such as CMOS and TTL.

4. Design of half adder using multilayer neural network

The half adder circuit is one of the combined circuits with two inputs and two outputs of carry and sum. The inputs in the half adder are one-bit inputs. Fig. 4 shows a half adder circuit. In the half adder circuit, if the two inputs are zero, sum and carry are also zero. If one input equals 1 and the other is 0, then sum=1 and carry=0. And if both inputs are one, sum=0 and carry=1.

In this section, a half adder circuit is designed using a multilayer neural network. Fig. 4 shows the half adder circuit designed using a neural network, in which the reference topology [4] has been used.

In this neural network, $W_{1=}$ $W_{2=}$ $W_{9=}$ $W_{10=}1$, $W_{3=}$ $W_{4=}$ $W_{5=}$ $W_{6=}0.5$ and W_7= W_8= 0.6 are the weights, and the threshold values in the sum are th_1= th_2=0.9, th_3= 0.6, th_4=0.4 and 0.1< th_5<1.1. The threshold value in the carry is equal to th_{carry}=1.9. Tables 13 and 14 present the output of the half adder obtained using the multilayer neural network.

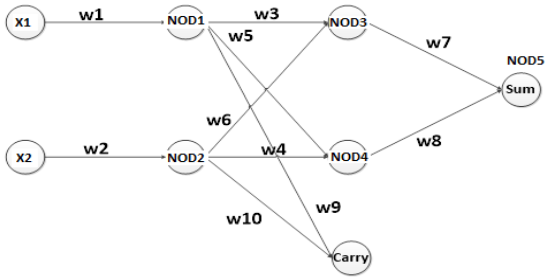

Fig. 4. Half adder obtained using the multilayer neural network

A) Calculation of Sum

Eq. 26 to 30 are related to the layer-by-layer investigation of the topology provided for the half adder.

Value of NOD1=x_1w_1 if Value of NOD1>0.9 then output of NOD1 is 1 else out put NOD1 is 0 (26)

Value of NOD2=x_2w_2 if Value of NOD2>0.9 then output of NOD2 is1 else out put NOD2 is 0 (27)

Value of NOD3=NOD1.w_3+NOD$_2$.w_6 if value of NOD3>0.6 then output of NOD3 is 1 else output of NOD3 is 0 (28)

Value of NOD4=NOD1.w_5+NOD$_2$.w_4 if value of NOD4>0.4 then output of NOD4 is 1 else output of NOD4 is 0 (29)

Value of sum=NOD3.w_7+NOD$_4$.w_8 if 0.1<value of sum<1.1 then sum is 1 else sum is 0 (30)

One can write Eq. 31 for sum with the excited NODs:

$$sum: x_1w_1w_3w_7 + x_2w_2w_3w_7 + x_1w_1w_5w_8 + x_2w_2w_4w_8 \quad (31)$$

Table 13 shows the sum output in the provided network.

B) Calculation of Carry

Eq. 32 to 34 are related to the layer-by-layer investigation of the topology provided for the half adder. Table 14 shows the output values of the carry in the proposed network.

$$\text{Value of } NOD1=x_1w_1 \quad \text{if value of } NOD1>0.9 \quad \text{then output of } NOD1=1 \quad \text{else output of } NOD1=0 \tag{32}$$

$$\text{Value of } NOD2=x_2w_2 \quad \text{if value of } NOD2>0.9 \quad \text{then output of } NOD2=1 \quad \text{else output of } NOD2=0 \tag{33}$$

$$\text{Value of } carry=NOD1.w_9+NOD_2.w_{10} \quad \text{if value of } carry>1.9 \text{ then } carry=1 \text{ else } carry=0 \tag{34}$$

According to Tables 13 and 14 the output of a neural network for a half adder corresponds to the validation table of the half adder designed by CMOS or TTL.

For the design of the full adder circuit, two half adders designed above are used in an OR gate. Fig. 5 shows a block diagram of a full adder circuit designed using two half adders and an OR gate.

Table.13.
Sum output values in the half adder

x_1	x_2	NOD1	NOD2	NOD1	NOD2	NOD3	NOD4	NOD3	NOD4	Value of sum	sum
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0.5	0.5	0	1	0.6	1
1	0	1	0	1	0	0.5	0.5	0	1	0.6	1
1	1	1	1	1	1	1	1	1	1	1.2	0

Table.14.
Output of carry

X_1	X_2	NOD1	NOD2	NOD1	NOD2	Value of carry	Carry
0	0	0	0	0	0	0	0
0	1	0	1	0	1	1	0
1	0	1	0	1	0	1	0
1	1	1	1	1	1	2	1

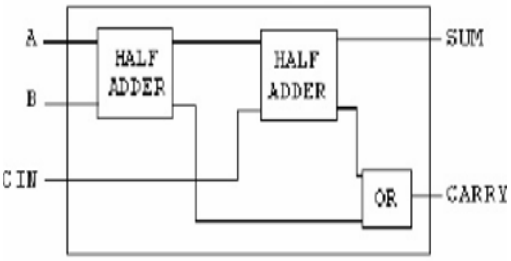

Fig. 5. Block diagram of a full adder circuit

5. Results and discussion

Logic gates can be designed in a variety of ways. Neural network is a new and proposed method of designing logic gates and half adder circuit. In this paper, their designs have been investigated. In [13], another type of design has been studied. In Table 15, the proposed method of the neural network is compared with the method presented in [13]. As you can see, the amount of weights matrix elements used in the proposed gates are less than the paper [13] and the amount of bias is zero. In [13], if we want to use different logical circuits, we need to apply synchronization blocks.

By comparing the logic gate and full adder circuit design with neural network and other designs, we concluded that this method was prioritized in terms of the amount of weights (Fig 6).

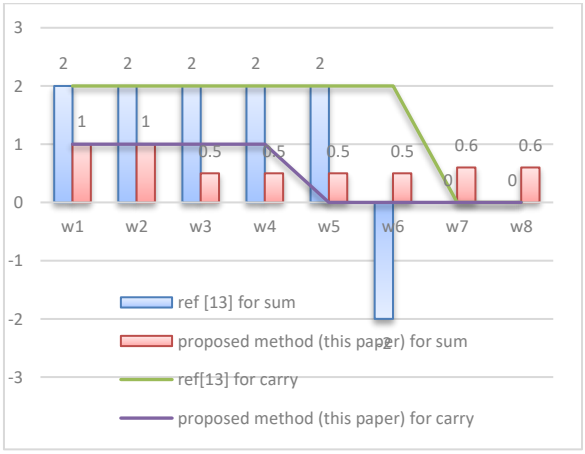

Fig. 6. Comparison of half adder characteristics with ref [13].

6. Conclusion

In this paper, logic gates are designed using a multilayer neural network and at the end a half adder is designed with a multilayer neural network. The results of the simulations are consistent with the results of logic gates and half adder designed with CMOS and TTL technologies. In parallel computing neural networks, the parallelism increases the processing speed and reduces the latency of the system as well as the power consumption. The use of neural network reduces the footprint and area of logic gates produced.

Table.15.
Comparison of proposed Logic Gate Design and full adder characteristics with Paper [13]

	The total number of neurons	***The total number of layers***	***Weights matrix***	***Bias value***
Proposed AND gate - Designed with NN	7	4	$W1=\begin{bmatrix}1\\1\end{bmatrix}$ $W2=\begin{bmatrix}0.5\ 0.5\\0.5\ 0.5\end{bmatrix}$ $W3=\begin{bmatrix}1\\1\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix}$
AND gate [13]	5	3	$W1=\begin{bmatrix}2\ 2\\2\ 2\end{bmatrix}$ $W2=\begin{bmatrix}2\\2\end{bmatrix}$	$\begin{bmatrix}0\\0\\-3\\+1\\-3\end{bmatrix}$
Proposed OR gate - Designed with NN	7	4	$W1=\begin{bmatrix}1\\1\end{bmatrix}$ $W2=\begin{bmatrix}0.5\ 0.5\\0.5\ 0.5\end{bmatrix}$ $W3=\begin{bmatrix}1\\1\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix}$
OR gate ١٣][	5	3	$W1=\begin{bmatrix}4\ 4\\4\ 4\end{bmatrix}$ $W2=\begin{bmatrix}-2\\2\end{bmatrix}$	$\begin{bmatrix}0\\0\\-3\\+1\\+1\end{bmatrix}$
Proposed XOR gate - Designed with NN	7	4	$W1=\begin{bmatrix}1\\1\end{bmatrix}$ $W2=\begin{bmatrix}0.5\ 0.5\\0.5\ 0.5\end{bmatrix}$ $W3=\begin{bmatrix}0.6\\0.6\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix}$
XOR ١٣][	5	3	$W1=\begin{bmatrix}1\ 1\\1\ 1\end{bmatrix}$ $W2=\begin{bmatrix}1\\-1\end{bmatrix}$	$\begin{bmatrix}0\\0\\-1.5\\-0.5\\-0.2\end{bmatrix}$
Proposed HALF-adder Designed with NN	$\begin{cases}SUM=7\\CARRY=5\end{cases}$	$\begin{cases}SUM=4\\CARRY=3\end{cases}$	$\begin{cases}SUM=\begin{cases}W1=\begin{bmatrix}1\\1\end{bmatrix}\\W2=\begin{bmatrix}0.5\ 0.5\\0.5\ 0.5\end{bmatrix}\\W3=\begin{bmatrix}0.6\\0.6\end{bmatrix}\end{cases}\\CARRY=\begin{cases}W1=\begin{bmatrix}1\\1\end{bmatrix} & W2=\begin{bmatrix}1\\1\end{bmatrix}\end{cases}\end{cases}$	$\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix}$ $\begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix}$
HALF-adder [13]	$\begin{cases}SUM=5\\CARRY=5\end{cases}$	$\begin{cases}SUM=3\\CARRY=3\end{cases}$	$\begin{cases}SUM=\begin{cases}W1,2=\begin{cases}\begin{bmatrix}2\ 2\\2\ 2\end{bmatrix}\\\begin{bmatrix}2\\-2\end{bmatrix}\end{cases}\end{cases}\\CARRY=\begin{cases}W3,4=\begin{cases}\begin{bmatrix}2\ 2\\2\ 2\end{bmatrix}\\\begin{bmatrix}2\\2\end{bmatrix}\end{cases}\end{cases}\end{cases}$	$\begin{bmatrix}0\\0\\-1\\-3\\-2\end{bmatrix}$ $\begin{bmatrix}0\\0\\-3\\+1\\-3\end{bmatrix}$

References:

[1] S. MOUKHLIS, A. ELRHARRAS and A. HAMDOUN, "FPGA Implementation of Artificial Neural Networks", International Journal of Computer Science Issues, Vol. 11 Issue 2, No 1, pp.237-239, 2014.

[2] S. Murugan, P.L.K, J.Sundar, M. K, "Design and Implementation of Multilayer Perceptron with On-Chip Learning in Virtex-E", 2nd AASRI Conference on Computational Intelligence and Bioinformatics, AASRI Procedia 6 (2014), pp.82 – 88, 2013.

[3] V.K.Singh, "Proposing Solution to XOR problem using minimum configuration MLP", International Conference on Computational Modeling and Security , pp.263-270, 2016.

[4] D. P. Sharma, "Neural Network Simulation of Digital Circuits", International Journal of Computer Applications (0975 – 8887), Vol79 – No6, pp.7-13, 2013.

[5] R. C, S. AP. "A review of implementation techniques for artificial neural networks", University School of

Information Technology, GGS Indraprastha University, Delhi. 2006.

[6] K.Shirsagar ua, B.Ae, "VLSI IMPLEMENTATION OF BACK PROPAGATED NEURAL NETWORK FOR SIGNAL PROCESSING".

[7] Han J, Li Z, Zheng W, Zhang Y. Hardware implementation of spiking neural networks on FPGA. Tsinghua Science and Technology. 25(4):479-86, 2020.

[8] S. Miyata , "Automatic Recognition of Speed Limits on Speed-Limit Signs by Using Machine Learning", Journal of Imaging, 2017.

[9] L.Debozorgi.'Case Study of Seismic Signals of GHIR sataion before Earthquake", Bullatin of earthquake science and engineering,2017.

[10] A.Chvála , L. Nagy , J. Marek , J. Priesol ,D. Donoval , A. Šatka ," Neural Network for Circuit Models of Monolithic InAlN/GaN NAND and NOR Logic Gates", 14th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS), 2019

[11] A. Fayyazi , S.Shababi , P.Nuzzo , S. Nazarian ; M. Pedram , "Deep Learning-Based Circuit Recognition Using Sparse Mapping and Level-Dependent Decaying Sum Circuit Representations", Design, Automation & Test in Europe Conference & Exhibition (DATE), 2019

[12] M. Natsui , T. Chiba , T. Hanyu , " MTJ-Based Nonvolatile Ternary Logic Gate for Quantized Convolutional Neural Networks", IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), 2018.

[13] N. Farha1, A. Louisa Paul, N. Kousar, R. Divakaran, " Design and Implementation of Logic Gates and Adder Circuits on FPGA Using ANN", International Journal for Research in Applied Science & Engineering Technology (IJRASET), 2016.