Experiment No.: 3

Title: Logic gates using McCulloch-Pits Neuron model

(Autonomous College Affiliated to University of Mumbai)

Batch : B1 **Roll No:** 1824017 **Experiment No:** 03

Aim: To design and implement logic gates using McCulloch-Pits neuron model

Resources needed: Matlab

Theory:

Pre Lab/ Prior Concepts:

The first formal definition of a synthetic neuron model based on the highly simplified considerations of the biological model described in the preceding section was formulated by McCulloch and Pitts (1943). The McCulloch-Pitts model of the neuron is below:

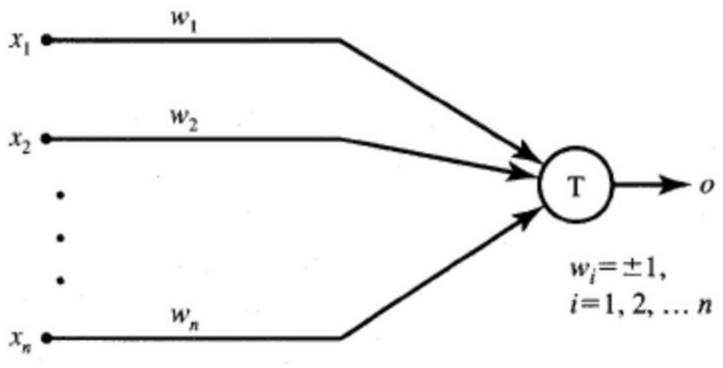


Figure: McCulloch-Pits Neuron Model

The inputs xi, for i = 1, 2, . . . , n, are 0 or 1, depending on the absence or presence of the input impulse at instant k. The neuron's output signal is denoted as o. The firing rule for this model is defined as follows:

$$o^{k+1} = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i^k \geq T \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i^k < T \end{cases}$$

Where,
superscript k = 0, 1, 2, . . . denotes the discrete-time instant,
wi is the multiplicative weight connecting the i'th input with the neuron's membrane, and
T is the neuron's threshold value, which needs to be exceeded by the weighted sum of signals for the neuron to fire.

Although this neuron model is very simplistic, it has substantial computing potential. It can perform the basic logic operations NOT, OR, and AND, provided its weights and thresholds are appropriately selected. Also, any multivariable combinational function can be implemented using either the NOT and OR, or alternatively the NOT and AND, Boolean operations. digital computer hardware of arbitrary complexity can be constructed using an artificial neural network consisting of elementary building blocks.

Features of McCulloch-Pitts model:

- Allows binary 0,1 states only
- Operates under a discrete-time assumption
- Weights and the neurons’ thresholds are fixed in the model and no interaction among network neurons
- Just a primitive model

Design of logic gates:
The design of logic gates AND, OR, NOT, NAND, NOR and XOR is as follows:

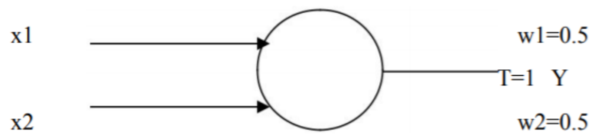
1. AND Gate:

Truth Table:

X1	X2	O	F/I
0	0	0	I
0	1	0	I
1	0	0	I
1	1	1	F

Design equations:
0<T
W2<T
W1<T
W1+W2>=T

Design Of AND Gate:



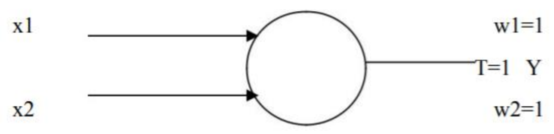
2. OR Gate:

Truth Table:

X1	X2	O	F/I
0	0	0	I
0	1	1	F
1	0	1	F
1	1	1	F

Design Of Equations:
0<T
W2>=T
W1>=T
W1+W2>=T

Design Of OR Gates:



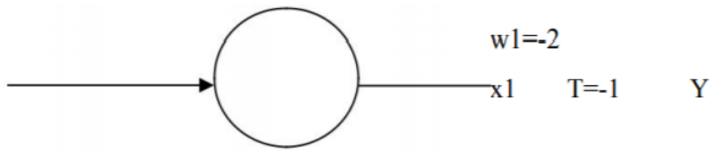
3. NOT Gate:

Truth Table:

X1	O	F/I
0	1	F
0	0	I

Design Equations:
0>=T
W1<T

Design Of NOT Gate:



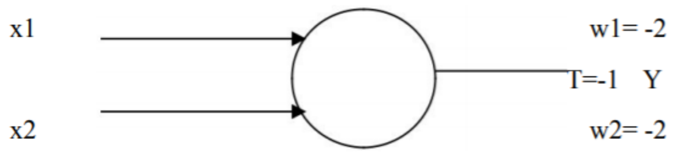
4. NOR Gate:

Truth Table:

X1	X2	O	F/I
0	0	1	F
0	1	0	I
1	0	0	I
1	1	0	I

Design Equations:
0>=T
W2<T
W1<T
W1+W2<T

Design of NOR Gate:



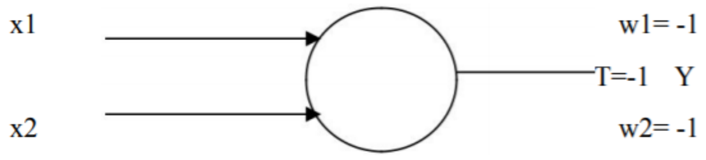
5. <u>NAND Gate</u>:

Truth Table:

X1	X2	O	F/I
0	0	1	F
0	1	1	F
1	0	1	F
1	1	0	I

Design Equations:
0>=T
W2>=T
W1>=T
W1+W2<T

Design of NAND Gate:



6. XOR Gate:

Truth Table:

X1	X2	O	F/I
0	0	0	I
0	1	1	F
1	0	1	F
1	1	0	I

Design Equations:
0<T
W2>=T
W1>=T
W1+W2

Program:
Write a program in Matlab to implement OR, AND, NAND, NOT, NOR gates using McCulloch-Pits Model. The program should accept the inputs related to respective gates and display output.

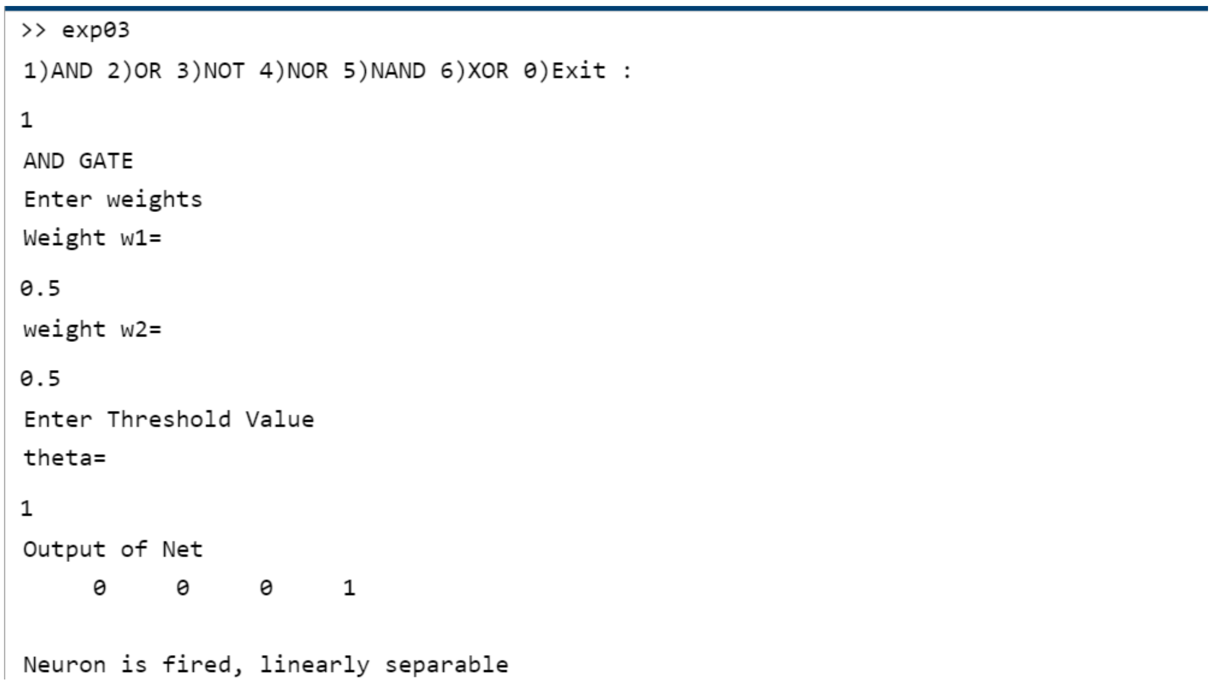
Program:

```
yes = 1 ;

while yes

    choice=input('1)AND 2)OR 3)NOT 4)NOR 5)NAND 6)XOR 0)Exit : ');
switch(choice)
    case 0
    yes=0;

    case 1
        % AND GATE
        disp('AND GATE');
        disp('Enter weights');

        w1=input('Weight w1=');
        w2=input('weight w2=');
        disp('Enter Threshold Value');

        theta=input('theta=');
        y=[0 0 0 0];
        x1=[0 0 1 1];
        x2=[0 1 0 1];
        z=[0 0 0 1];

        con=1;
        while con
        zin=x1*w1+x2*w2;
        for i=1:4
            if zin(i)>=theta
                y(i)=1;
            else
                y(i)=0;
            end
        end
        disp('Output of Net');

        disp(y);
        if y==z
            con=0;
            disp('Neuron is fired, linearly separable');
        else

            disp('Neuron is not fired, non-linearly separable');
            w1=input('weight w1=');
            w2=input('weight w2=');
            theta=input('theta=');

        end
    end
    case 2
        %OR GATE
        disp('OR GATE');
        disp('Enter weights');

        w1=input('Weight w1=');
        w2=input('weight w2=');
        disp('Enter Threshold Value');

        theta=input('theta=');
        y=[0 0 0 0];
        x1=[0 0 1 1];
        x2=[0 1 0 1];
        z=[0 1 1 1];

        con=1;
        while con
            zin=x1*w1+x2*w2;
            for i=1:4
                if zin(i)>=theta
                    y(i)=1;
                else
                    y(i)=0;
                end
            end

        disp('Output of Net');
        disp(y);
        if y==z
            con=0;
            disp('Neuron is fired, linearly separable');
        else
            disp('Neuron is not fired, non-linearly separable');

            w1=input('weight w1=');
            w2=input('weight w2=');
            theta=input('theta=');
        end
    end

    case 3
        % NOT GATE
        disp('NOT GATE');
        disp('Enter weight');

        w1=input('Weight w1=');
        disp('Enter Threshold Value');
        theta=input('theta=');
        y=[0 0];
    x1=[0 1];

    z=[1 0];
    con=1;
    while con
        zin=x1*w1;
        for i=1:2

        if zin(i)>=theta
            y(i)=0;
        else
            y(i)=1;
        end
    end

    disp('Output of Net');
    disp(y);
    if y==z
            con=0;
            disp('Neuron is fired, linearly separable');
        else
            disp('Neuron is not fired, non-linearly separable');
            w1=input('weight w1=');
            theta=input('theta=');
    end
end

case 4
    % NOR GATE
    disp('NOR GATE');

    disp('Enter weights');

    w1=input('Weight w1=');
    w2=input('weight w2=');
    disp('Enter Threshold Value');

    theta=input('theta=');
    y=[0 0 0 0];

    x1=[0 0 1 1];
    x2=[0 1 0 1];
    z=[1 0 0 0];

    con=1;
    while con
    zin=x1*w1+x2*w2;
    for i=1:4
        if zin(i)>=theta
            y(i)=0;
        else
            y(i)=1;
        end
    end
    disp('Output of Net');

    disp(y);
    if y==z
        con=0;
        disp('Neuron is fired, linearly separable');
    else
        disp('Neuron is not fired, non-linearly separable');
        w1=input('weight w1=');
        w2=input('weight w2=');
```

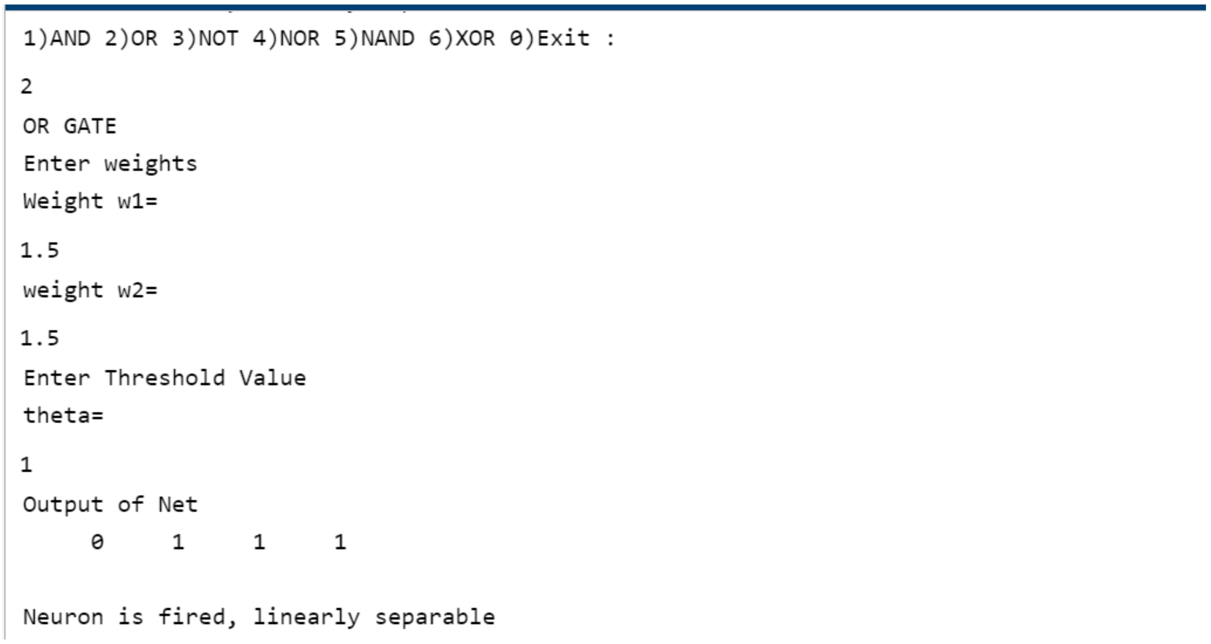
```
            theta=input('theta=');
        end
    end

    case 5
        % NAND GATE
        disp('NAND GATE');
        disp('Enter weights');

        w1=input('Weight w1=');
        w2=input('weight w2=');
        disp('Enter Threshold Value');
        theta=input('theta=');
        y=[0 0 0 0];

        x1=[0 0 1 1];
        x2=[0 1 0 1];
        z=[1 1 1 0];
        con=1;
        while con

        zin=x1*w1+x2*w2;

        for i=1:4
            if zin(i)>=theta
                y(i)=0;
            else

                y(i)=1;
            end
        end
        disp('Output of Net');

        disp(y);
        if y==z
            con=0;
            disp('Neuron is fired, linearly separable');
        else
            disp('Neuron is not fired, non-linearly separable');
            w1=input('weight w1=');
            w2=input('weight w2=');
            theta=input('theta=');
        end
    end

    case 6
        % XOR GATE
        disp('XOR GATE');

        con=0;
        while con < 2
        con=con+1;
        disp('Enter weights');

        w1=input('Weight w1=');
        w2=input('weight w2=');
        disp('Enter Threshold Value');
        theta=input('theta=');
        y=[0 0 0 0];

        x1=[0 0 1 1];
        x2=[0 1 0 1];
        z=[0 1 1 0];
        zin=x1*w1+x2*w2;
```

```
                for i=1:4

                    if zin(i)>=theta
                        y(i)=1;
                    else
                        y(i)=0;
                    end
                end

                disp('Output of Net');
                disp(y);
                    if y==z
                        con=0;
                        disp('Neuron is fired, linearly separable');

                    else
                        disp('This GATE cannot be realized');
                    end
                end
            end

        end
```

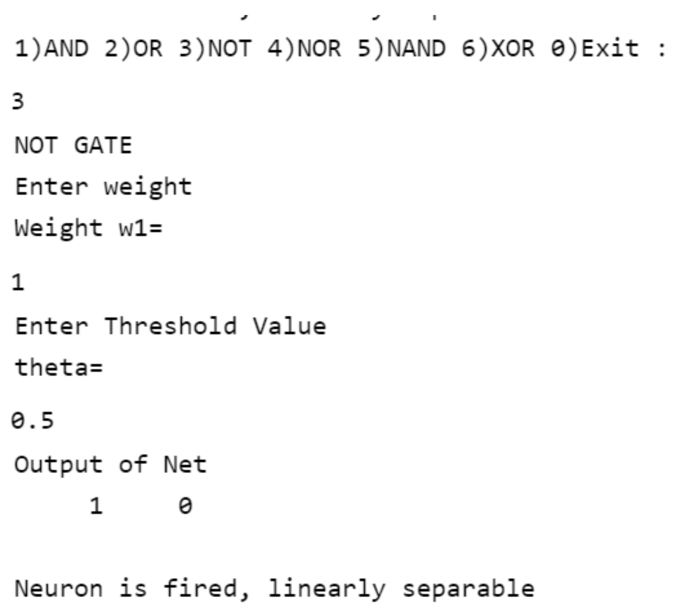
Output:



```
>> exp03
1)AND 2)OR 3)NOT 4)NOR 5)NAND 6)XOR 0)Exit :
1
AND GATE
Enter weights
Weight w1=
0.5
weight w2=
0.5
Enter Threshold Value
theta=
1
Output of Net
     0     0     0     1

Neuron is fired, linearly separable
```

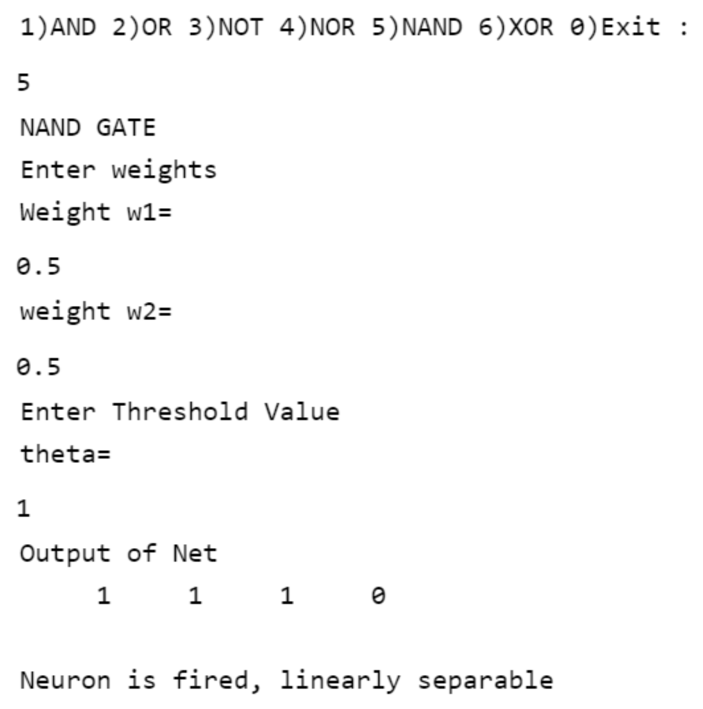


```
1)AND 2)OR 3)NOT 4)NOR 5)NAND 6)XOR 0)Exit :
2
OR GATE
Enter weights
Weight w1=
1.5
weight w2=
1.5
Enter Threshold Value
theta=
1
Output of Net
     0     1     1     1

Neuron is fired, linearly separable
```

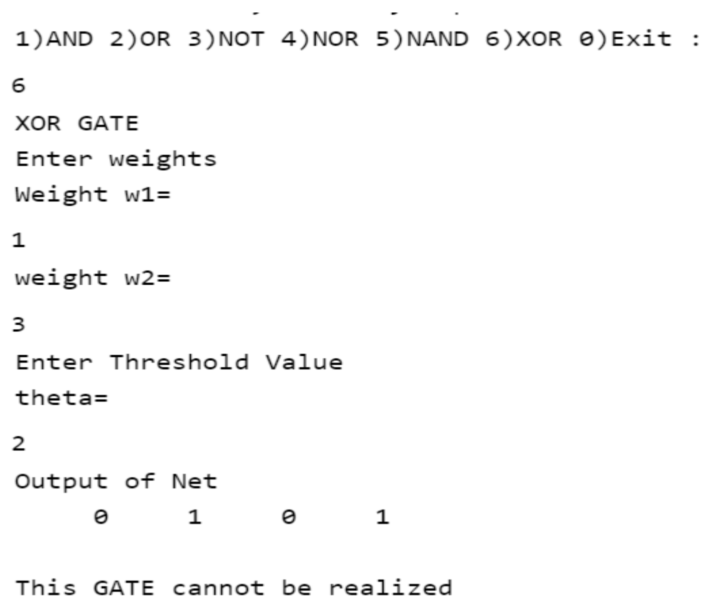


```
1)AND 2)OR 3)NOT 4)NOR 5)NAND 6)XOR 0)Exit :
3
NOT GATE
Enter weight
Weight w1=
1
Enter Threshold Value
theta=
0.5
Output of Net
     1     0

Neuron is fired, linearly separable
```

```
1)AND 2)OR 3)NOT 4)NOR 5)NAND 6)XOR 0)Exit :
4
NOR GATE
Enter weights
Weight w1=
3
weight w2=
2
Enter Threshold Value
theta=
1
Output of Net
     1     0     0     0

Neuron is fired, linearly separable
```



```
1)AND 2)OR 3)NOT 4)NOR 5)NAND 6)XOR 0)Exit :
5
NAND GATE
Enter weights
Weight w1=
0.5
weight w2=
0.5
Enter Threshold Value
theta=
1
Output of Net
     1     1     1     0

Neuron is fired, linearly separable
```



```
1)AND 2)OR 3)NOT 4)NOR 5)NAND 6)XOR 0)Exit :
6
XOR GATE
Enter weights
Weight w1=
1
weight w2=
3
Enter Threshold Value
theta=
2
Output of Net
     0     1     0     1

This GATE cannot be realized
```

Outcomes: Apply various Machine Learning Algorithms to design, analyse and perform experiments on real life problems

Conclusion: (Conclusion to be based on outcomes)
From this experiment, I have successfully created simple neural network models using McCulloch-Pitts neural network model for different logic gates using matlab.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:
Books/ Journals/ Websites:

1. Jacek M. Zurada, “Introduction to artificial neural systems”, West Publishing Company