PROJECT REPORT

MONOPONG GAME

Submitted for CAL in BTech DIGITAL LOGIC(CSE1003)

By

Aastha Sahay-16BCE1296

Meghna Lohani -16BLC1103

Sruthi Shiva- 16BCE1064

Smrithi Vasudevan- 16BCE1140

Slot: F1

Name of faculty: **Maheswari R**

CERTIFICATE

This is to certify that the Project work entitled “MONOPONG” that is being submitted by “Meghna Lohani,Sruthi Shiva, Aastha Sahay,Smrithi Vasudevan ” for CAL in BTech Digital Logic is a record of bonafide work done under my supervision. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

Place: Chennai

Date: 26 April 2017

Signature of Students:

Signature of Faculty:

ACKNOWLEDGEMENTS

We would like to express our thanks and gratitude to our teacher Dr. Maheshwari R as well as SCSE Dean and VIT University who gave us the opportunity to do this wonderful project on the topic "Monopong".

Without her guidance, it would not have been possible to finish the project successfully within the limited time frame.

ABSTRACT

The Game is a 1D-version of the famous PONG game.
In this game, the ball moves from left to right. The player has to push the button at the right moment to strike it back. If he/she misses the ball the other player wins one point.

MonoPong is a simple, addictive game with a strong technology behind. It has CMOS NAND Gate (4011), a 4510 up/down decade counter, 4028 BCD to decimal decoder chip and as an oscillator an NE555 in a stable mode.

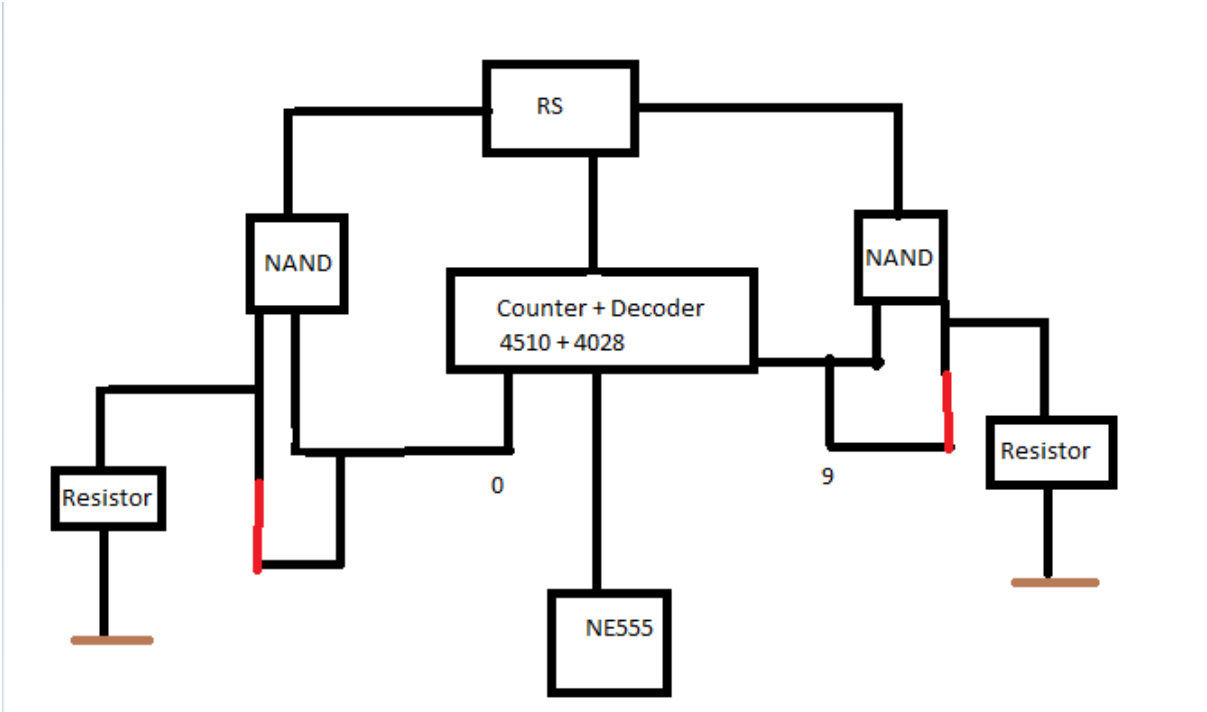
Every research and application implemented is genuine and true to our knowledge and any kind of plagiarism is not done.

Methodology

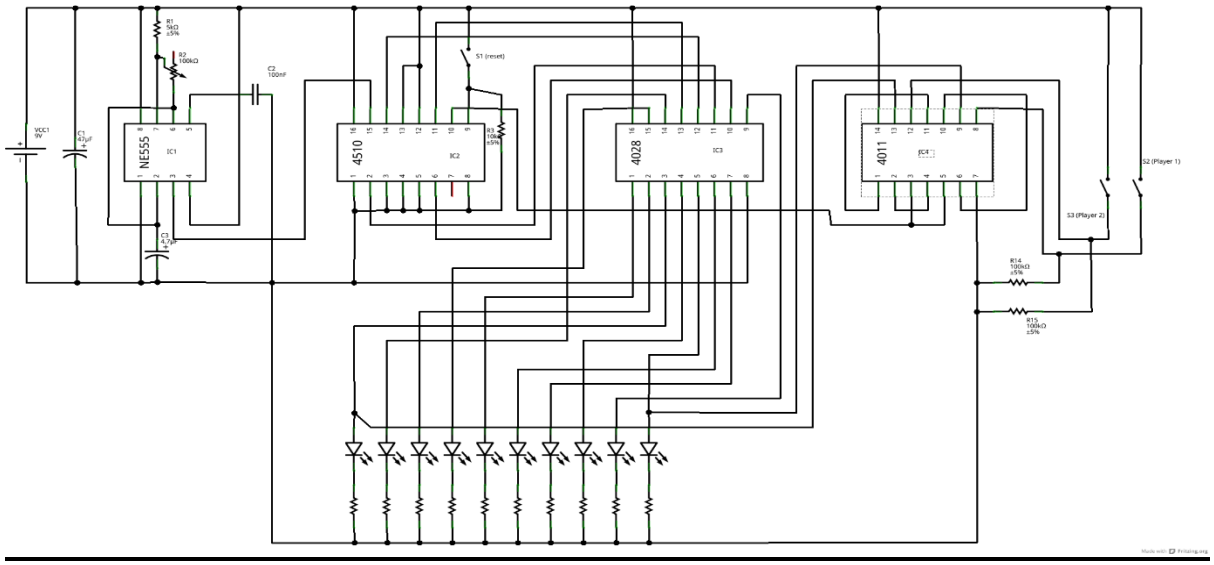
COMPONENTS

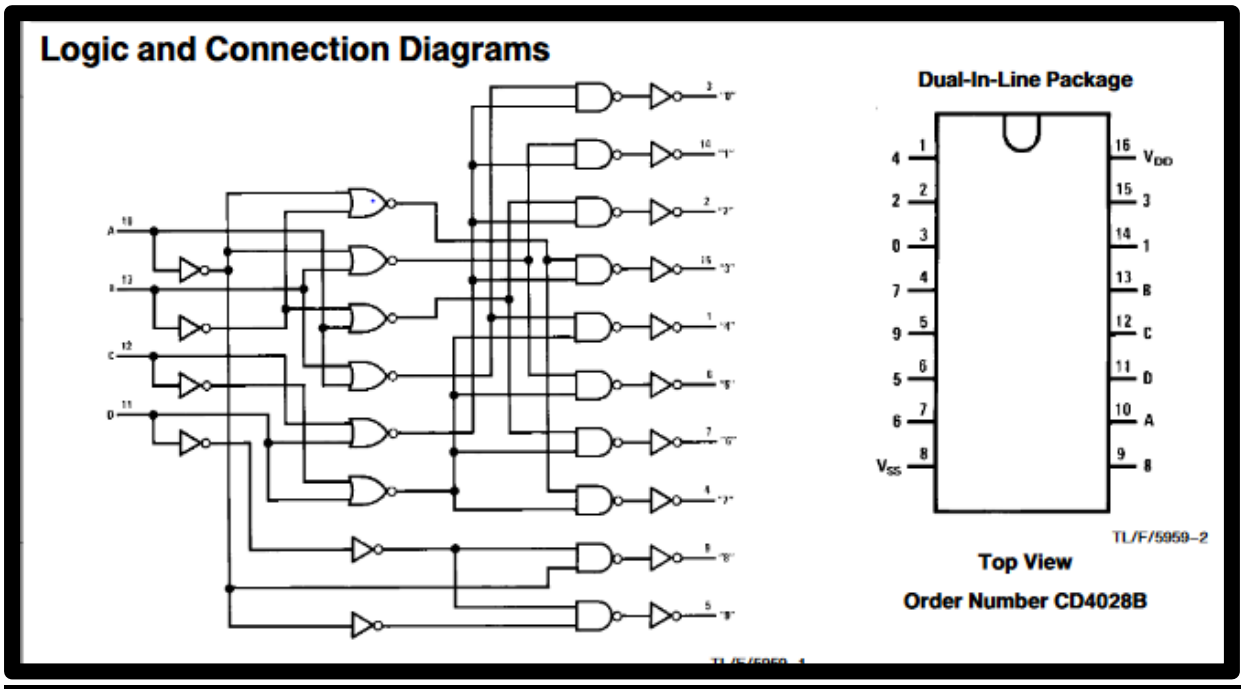
- 10 LEDs: 8 red, 2 blue
- 2 momentary push buttons
- 1 on-off switch
- NE555
- CMOS 4510
- CMOS 4028
- CMOS 4011
- Potentiometer 100k
- Resistors: 1x 5k, 2x 100k, 1x 470 Ohm
- Capacitors: 1x 47µF, 1x 4.7µF, 1x 100nF
- Wires
- Double sided tape
- 9V battery and clip

BLOCK DIAGRAM



CIRCUIT DIAGRAM





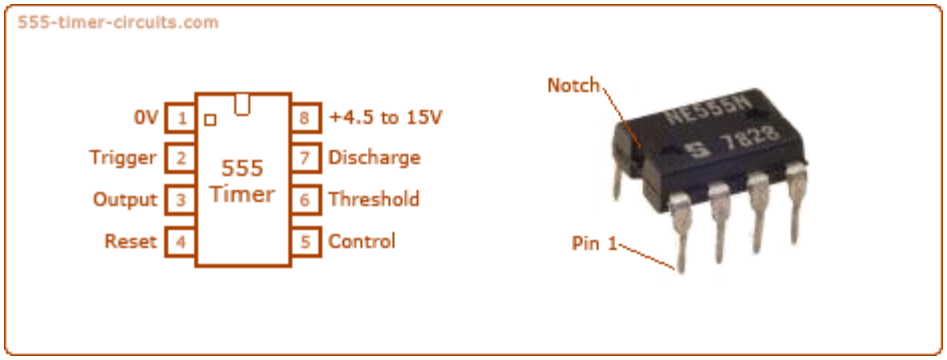
Truth Table

D	C	B	A	0	1	2	3	4	5	6	7	8	9	
0	0	0	0	1	0	0	0	0	0	0	0	0	0	BCD States
0	0	0	1	0	1	0	0	0	0	0	0	0	0	
0	0	1	0	0	0	1	0	0	0	0	0	0	0	
0	0	1	1	0	0	0	1	0	0	0	0	0	0	
0	1	0	0	0	0	0	0	1	0	0	0	0	0	
0	1	0	1	0	0	0	0	0	1	0	0	0	0	
0	1	1	0	0	0	0	0	0	0	1	0	0	0	
0	1	1	1	0	0	0	0	0	0	0	1	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	1	0	
1	0	0	1	0	0	0	0	0	0	0	0	0	1	
1	0	1	0	0	0	0	0	0	0	0	0	1	0	Extraordinary States
1	0	1	1	0	0	0	0	0	0	0	0	0	1	
1	1	0	0	0	0	0	0	0	0	0	0	1	0	
1	1	0	1	0	0	0	0	0	0	0	0	0	1	
1	1	1	0	0	0	0	0	0	0	0	0	1	0	
1	1	1	1	0	0	0	0	0	0	0	0	0	1	

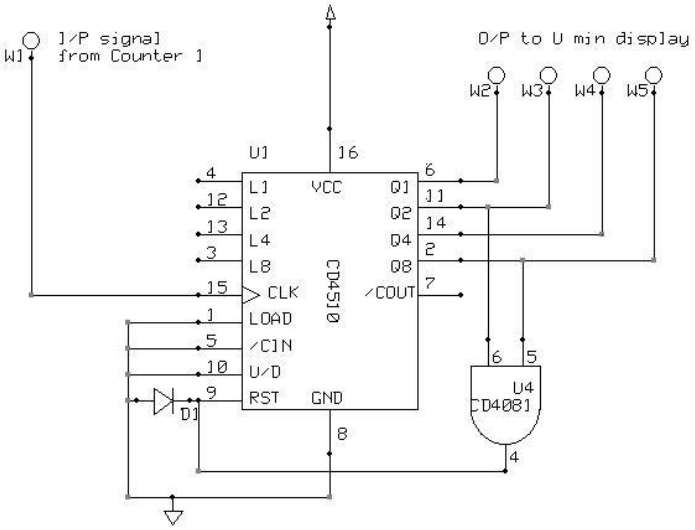
1 = High Level
0 = Low Level

PIN DIAGRAMS

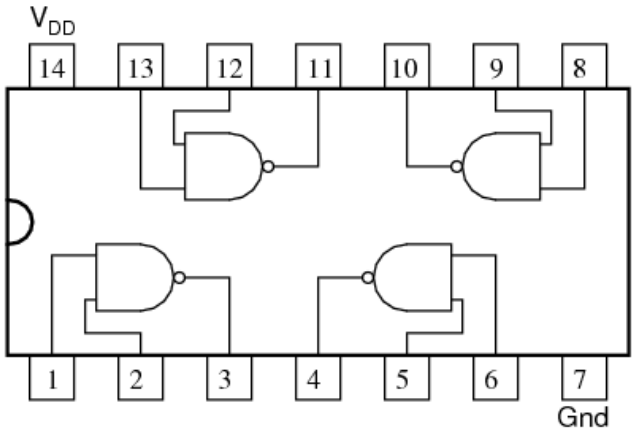
1. NE555



2. CD4510



3. CD4011

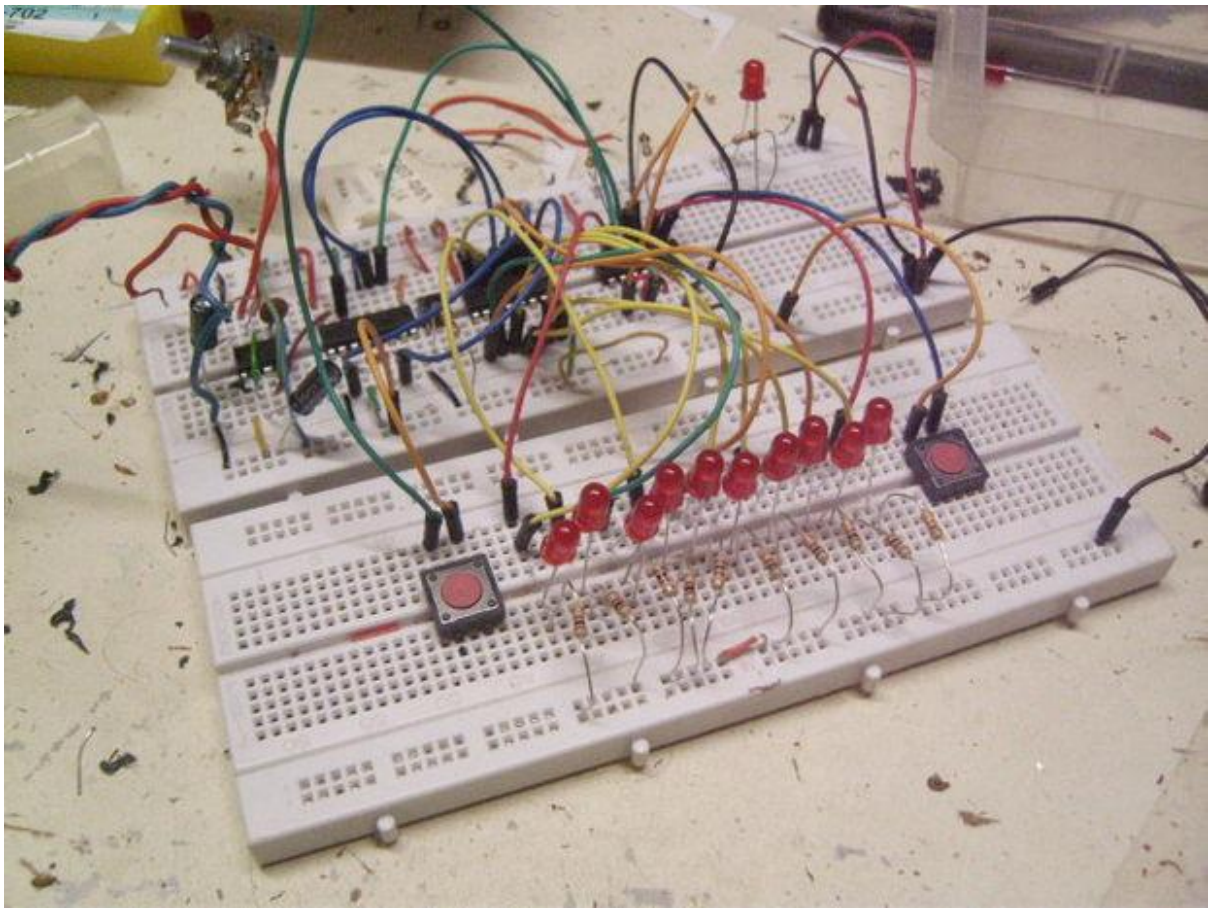


Input		Output
A	B	Y= $\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

WORKING

IC2 in the schematic is a CMOS 4510 up/down counter. Driven by IC1 which is a NE555 in a stable mode it counts from 0 to 9 in BCD code. Normally it counts from 0 to 9, but it can be switched to count in the opposite direction by b setting pin 10 to BCD code is converted into a decimal code by the CMOS 4028 (IC2). This HIGH. The one drives the display and gives out the counted number in form of one of 10 LEDs lighting up.

The last IC is a 4011 NAND gate. Two of the NAND gates are connected to work as a RS flip flop. The inputs of the other two NAND gates are connected to the buttons and LED1 or LED10. The outputs are connected to the RS flip flop. If you push the button at the right moment, which means for example when LED10 lights up, the output of the NAND gate sends a LOW signal to the flip flop which changes its output state (eg. from HIGH to LOW or vice versa). This signal goes to pin 10 of the 4510 to change the direction of the count. The "ball" bounces back.



CODING

1)FOR 9-0 COUNTER

```
//declare the Verilog module- The inputs and output port names.
module decoder4to10(Data_in, Data_out);
//what are the input ports and their sizes.
input [3:0] Data_in;
//what are the output ports and their sizes.
output [9:0] Data_out;
//internal variables
reg [9:0] Data_out;
//Whenever there is a change in the Data_in, execute the always block.
always @(Data_in)
case (Data_in) //case statement. Check all the 8 combinations.
4'b1001 : Data_out=10'b1000000000;
4'b1000 : Data_out=10b'0100000000;
4'b0111 : Data_out=10b'0010000000;
4'b0110 : Data_out=10b'0001000000;
4'b0101 : Data_out=10b'0000100000;
4'b0100 : Data_out=10b'0000010000;
4'b0011 : Data_out=10b'0000001000;
4'b0010 : Data_out=10b'0000000100;
4'b0001 : Data_out=10b'0000000010;
4'b0000 : Data_out=10b'0000000001;
//To make sure that latches are not created create a default value for output.
default : Data_out = 10'b0000000000;
endcase
endmodule

//This is a testbench code used for testing the 4:10 decoder module.
//Since its a testbench code we dont need to define any inputs or outputs for the block.
module tb_decoder;
//Declaring Inputs
reg [3:0] Data-in;
//Declaring Outputs
wire [9:0] Data_out;
//Instantiate the Unit Under Test (UUT)
decoder4to10 uut(.Data_in(Data_in), .Data_out(Data_out));
initial begin
//Apply Input and wait for 100ns
Data_in= 4'b1001; #100;
Data_in= 4'b1000; #100;
Data_in= 4'b0111; #100;
Data_in= 4'b0110; #100;
Data_in= 4'b0101; #100;
Data_in= 4'b0100; #100;
Data_in= 4'b0011; #100;
Data_in= 4'b0010; #100;
```

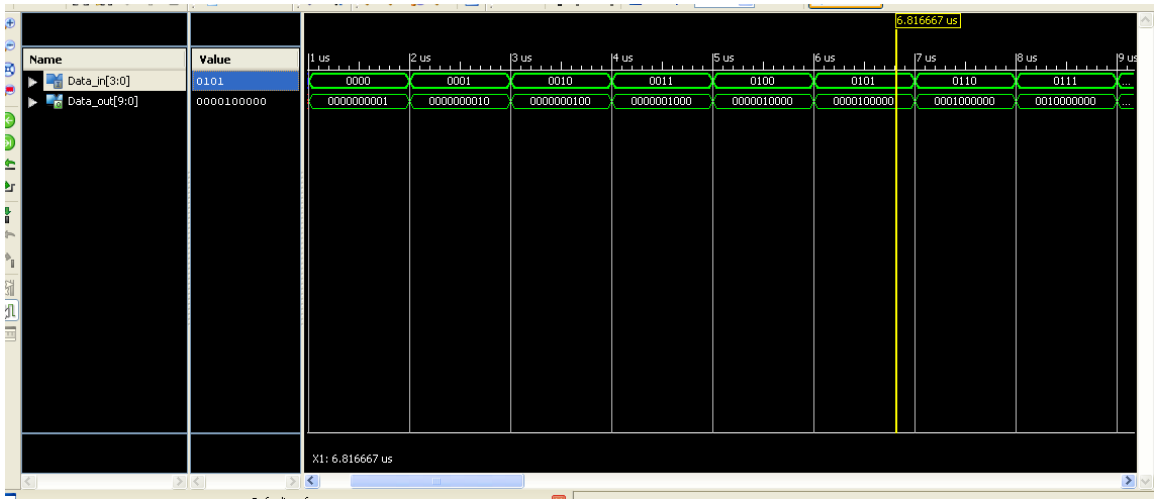
```
Data_in= 4'b0001; #100;
Data_in= 4'b0000; #100;
end
endmodule
```

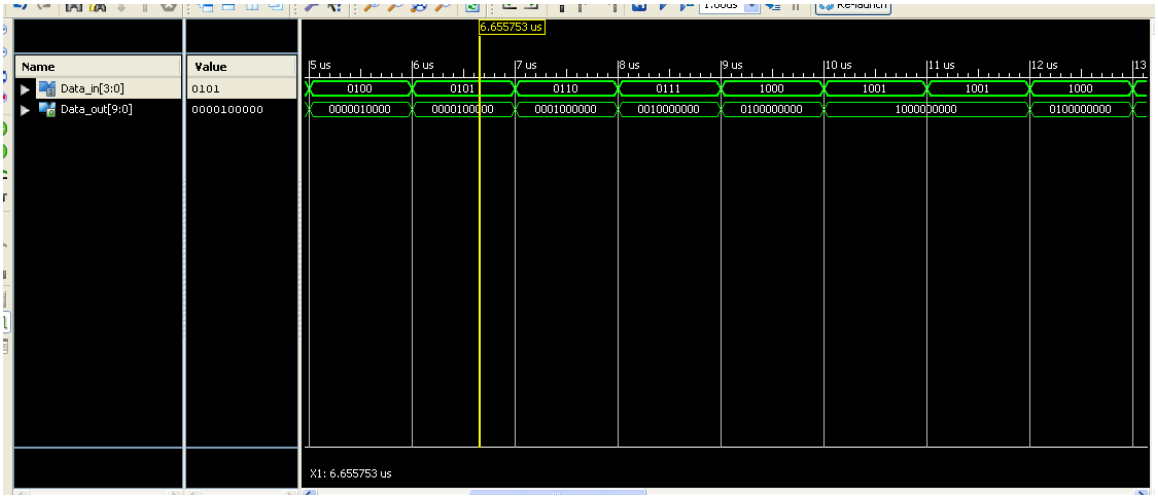
2)**FOR 0-9 COUNTER**

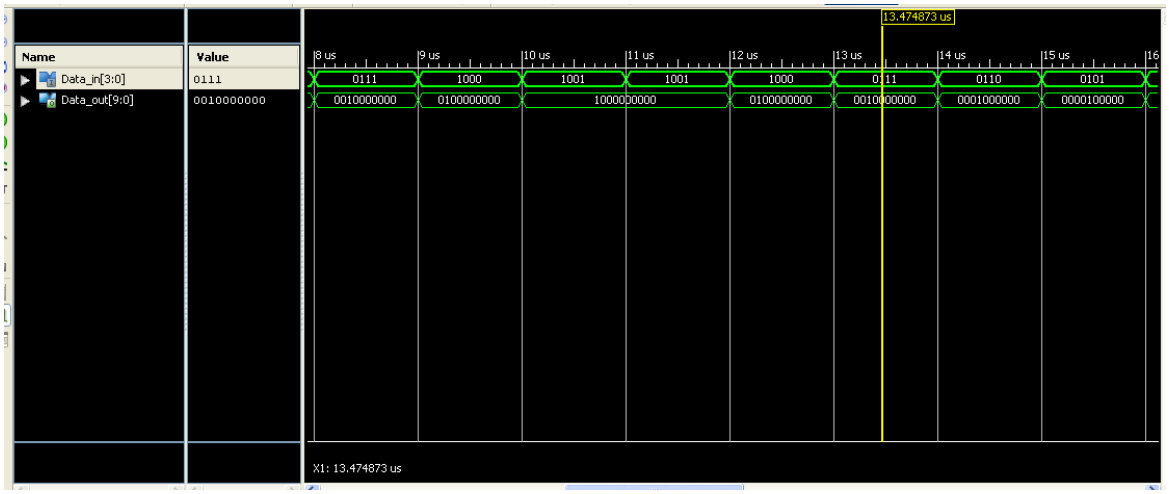
```
//declare the Verilog module- The inputs and output port names.
module decoder4to10(Data_in, Data_out);
//what are the input ports and their sizes.
input [3:0] Data_in;
//what are the output ports and their sizes.
output [9:0] Data_out;
//internal variables
reg [9:0] Data_out;
//Whenever there is a change in the Data_in, execute the always block.
always @(Data_in)
case (Data_in) //case statement. Check all the 8 combinations.
4'b0000 : Data_out=10'b0000000001;
4'b0001 : Data_out=10b'0000000010;
4'b0010 : Data_out=10b'0000000100;
4'b0011 : Data_out=10b'0000001000;
4'b0100 : Data_out=10b'0000010000;
4'b0101 : Data_out=10b'0000100000;
4'b0110 : Data_out=10b'0001000000;
4'b0111 : Data_out=10b'0010000000;
4'b1000 : Data_out=10b'0100000000;
4'b1001 : Data_out=10b'1000000000;
//To make sure that latches are not created create a default value for output.
default : Data_out = 10'b0000000000;
endcase
endmodule

//This is a testbench code used for testing the 4:10 decoder module.
//Since its a testbench code we dont need to define any inputs or outputs for the block.
module tb_decoder;
//Declaring Inputs
reg [3:0] Data-in;
//Declaring Outputs
wire [9:0] Data_out;
//Instantiate the Unit Under Test (UUT)
decoder4to10 uut(.Data_in(Data_in), .Data_out(Data_out));
initial begin
//Apply Input and wait for 100ns
Data_in= 4'b0000; #100;
Data_in= 4'b0001; #100;
Data_in= 4'b0010; #100;
Data_in= 4'b0011; #100;
Data_in= 4'b0100; #100;
```

```
Data_in= 4'b0101; #100;
Data_in= 4'b0110; #100;
Data_in= 4'b0111; #100;
Data_in= 4'b1000; #100;
Data_in= 4'b1001; #100;
end
endmodule
```

SIMULATONS


6.816667 us
Name
Value
Data_in[3:0]
0101
Data_out[9:0]
0000100000
1 us
2 us
3 us
4 us
5 us
6 us
7 us
8 us
9 us
0000
0001
0010
0011
0100
0101
0110
0111
0000000001
0000000010
0000000100
0000001000
0000010000
0000100000
0001000000
0010000000
X1: 6.816667 us


6.655753 us
Name
Value
Data_in[3:0]
0101
Data_out[9:0]
0000100000
5 us
6 us
7 us
8 us
9 us
10 us
11 us
12 us
13
0100
0101
0110
0111
1000
1001
1001
1000
0000010000
0000100000
0001000000
0010000000
0100000000
1000000000
0100000000
X1: 6.655753 us


13.474873 us
Name
Value
Data_in[3:0]
0111
Data_out[9:0]
0010000000
8 us
9 us
10 us
11 us
12 us
13 us
14 us
15 us
16
0111
1000
1001
1001
1000
0111
0110
0101
0010000000
0100000000
1000000000
0100000000
0001000000
0000100000
X1: 13.474873 us


13.508683 us
Name
Value
Data_in[3:0]
0111
Data_out[9:0]
0010000000
12 us
13 us
14 us
15 us
16 us
17 us
18 us
19 us
1000
0111
0110
0101
0100
0011
0010
0001
0100000000
0010000000
0001000000
0000100000
0000010000
0000001000
0000000100
0000000010
X1: 13.508683 us
16.426423 us
Name
Value
Data_in[3:0]
0100
Data_out[9:0]
0000010000
13 us
14 us
15 us
16 us
17 us
18 us
19 us
20 us
0111
0110
0101
0100
0011
0010
0001
0000
0010000000
0001000000
0000100000
0000010000
0000001000
0000000100
0000000010
0000000001
X1: 16.426423 us

CONCLUSION

The Monopong circuit was studied and implemented and the output was thus observed.

REFERNCES

TEXTBOOKS

Morris Mano
Digital Electronics- By Ananda Kumar

WEBSITES

www.instructables.com
www.verilog.com