DIGITAL DESIGN

LAB-2 REPORT

1) Binary to Greycode

Logic Code:

```
module binary_to_greycode(A,G);
input [3:0]A;
output [3:0]G;

assign G[3]=A[3];
assign G[2]=A[3]^A[2];
assign G[1]=A[2]^A[1];
assign G[0]=A[1]^A[0];
endmodule
```

Test Bench:

```
module test_binary_to_greycode();
reg [3:0]A;
wire [3:0]G;
binary_to_greycode BG1(A,G);
initial
begin
    A=4'b0000;
    #10 A=4'b0001;
    #10 A=4'b0010;
    #10 A=4'b0011;
    #10 A=4'b0100;
    #10 A=4'b0101;
    #10 A=4'b0110;
    #10 A=4'b0111;
    #10 A=4'b1000;
    #10 A=4'b1001;
    #10 A=4'b1010;
    #10 A=4'b1011;
    #10 A=4'b1100;
    #10 A=4'b1101;
    #10 A=4'b1110;
    #10 A=4'b1111;
end
initial #160 $finish;
endmodule
```

Name	Valu
A[3:0]	0000
G[3:0]	0000

Name	Valu
A[3:0]	0001
G[3:0]	0001

Name	Valu
A[3:0]	0010
G[3:0]	0011

Name	Valu
A[3:0]	0011
G[3:0]	0010

Name	Value
A[3:0]	0100
G[3:0]	0110

Name	Value
A[3:0]	0101
G[3:0]	0111

Name	Value
A[3:0]	0110
G[3:0]	0101

Name	Value
A[3:0]	0111
G[3:0]	0100

Name	Value
A[3:0]	1000
G[3:0]	1100

Name	Value
A[3:0]	1001
G[3:0]	1101

Name	Value
A[3:0]	1010
G[3:0]	1111

Name	Value
A[3:0]	1011
G[3:0]	1110

Name	Value
A[3:0]	1100
G[3:0]	1010

Name	Value
A[3:0]	1101
G[3:0]	1011

Name	Value
A[3:0]	1110
G[3:0]	1001

Name	Value
A[3:0]	1111
G[3:0]	1000

A
G

A
G

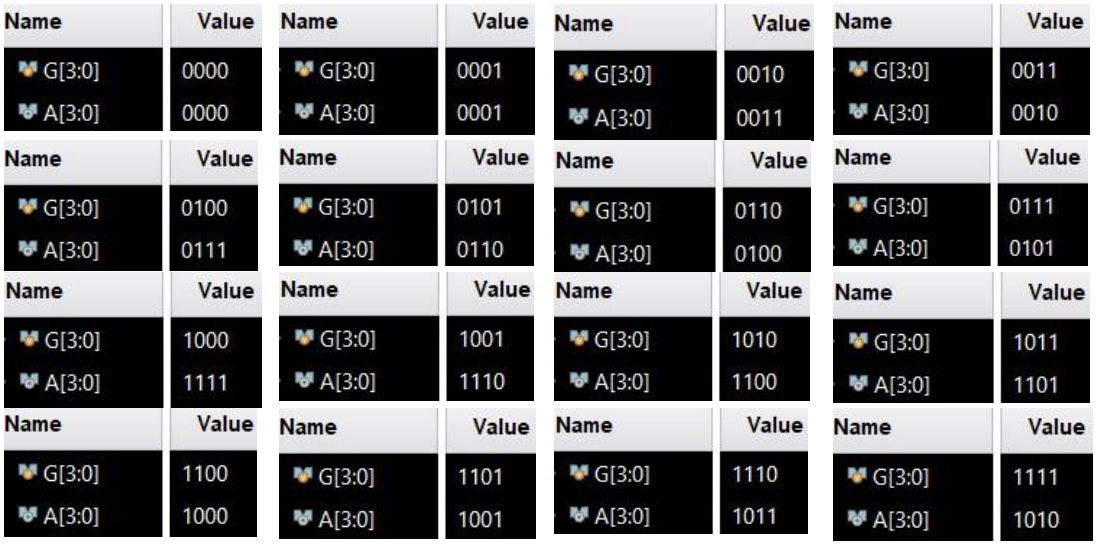
Greycode to Binary

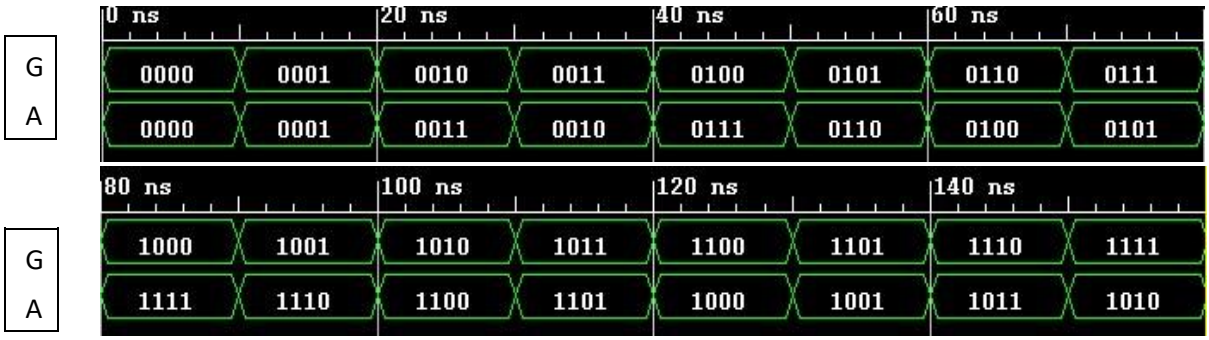
Logic Code:

```
module greycode_to_binary(G,A);
input [3:0]G;
output [3:0]A;

assign A[3]=G[3];
assign A[2]=G[3]^G[2];
assign A[1]=G[3]^G[2]^G[1];
assign A[0]=G[3]^G[2]^G[1]^G[0];
endmodule
```

Test Bench:

```
module test_greycode_to_binary1();
reg [3:0]G;
wire [3:0]A;
greycode_to_binary GB1(G,A);
initial
begin
    G=4'b0000;
    #10 G=4'b0001;
    #10 G=4'b0010;
    #10 G=4'b0011;
    #10 G=4'b0100;
    #10 G=4'b0101;
    #10 G=4'b0110;
    #10 G=4'b0111;
    #10 G=4'b1000;
    #10 G=4'b1001;
    #10 G=4'b1010;
    #10 G=4'b1011;
    #10 G=4'b1100;
    #10 G=4'b1101;
    #10 G=4'b1110;
    #10 G=4'b1111;
end
initial #160 $finish;
endmodule
```

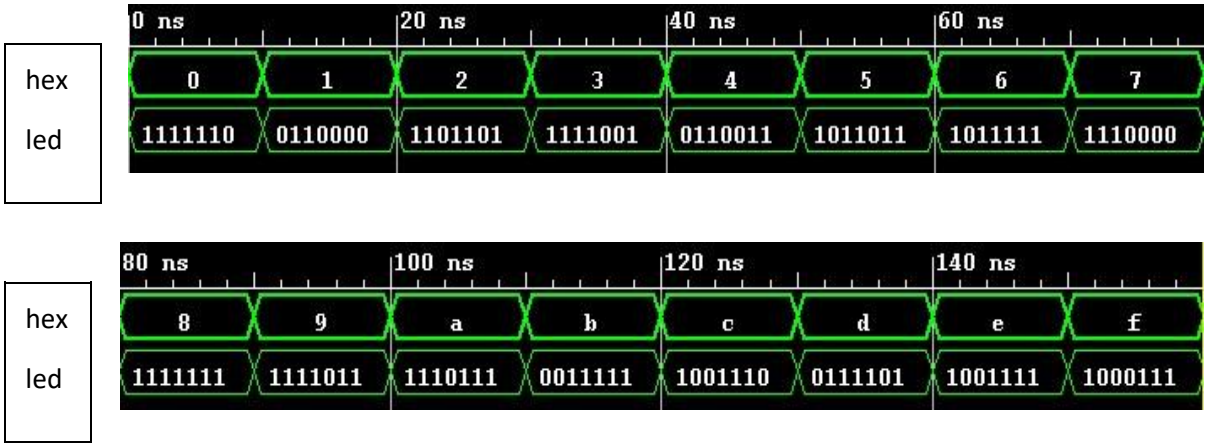




2) Hexadecimal to 7-segment

Logic Code:

```
module hex_to_bcd(hex,led);
input [3:0] hex;
output [0:6]led;
reg [0:6]led;

always @(hex)
    begin
        case(hex)
            0: led=7'b1111110;
            1: led=7'b0110000;
            2: led=7'b1101101;
            3: led=7'b1111001;
            4: led=7'b0110011;
            5: led=7'b1011011;
            6: led=7'b1011111;
            7: led=7'b1110000;
            8: led=7'b1111111;
            9: led=7'b1111011;
            10: led=7'b1110111;
            11: led=7'b0011111;
            12: led=7'b1001110;
            13: led=7'b0111101;
            14: led=7'b1001111;
            15: led=7'b1000111;
            default: led=7'b1111110;
        endcase
    end
endmodule
```



3) Two Player Buzzer

Logic Code:

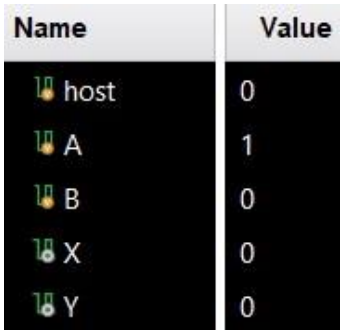
```
module buzzer(host,A,B,X,Y);
input host,A,B;
output X,Y;
reg X='b0,Y='b0;

always @(host or A or B)
begin
assign X=host?(A?(B?((X===0)?'b1:X):'b1):(B?'bz:'b0)):'b0;
assign Y=host?(B?(A?((Y===0)?'b1:Y):'b1):(A?'bz:'b0)):'b0;
end
endmodule
```

Test Bench:

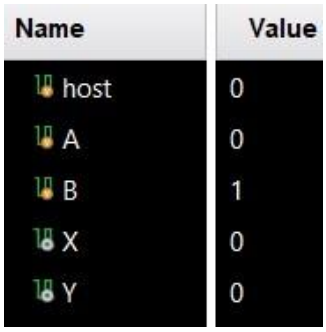
```
module test_buzzer();
reg host,A,B;
wire X,Y;
buzzer B1(host,A,B,X,Y);

initial
begin
    host=1'b0; A=1'b0; B=1'b0;
    #5 host=1'b1; A=1'b0; B=1'b0;
    #5 host=1'b1; A=1'b1; B=1'b0;
    #2 host=1'b1; A=1'b1; B=1'b1;
    #3 host=1'b0; A=1'b0; B=1'b0;
    #5 host=1'b1; A=1'b0; B=1'b0;
    #5 host=1'b1; A=1'b0; B=1'b1;
    #2 host=1'b1; A=1'b1; B=1'b1;
    #3 host=1'b0; A=1'b0; B=1'b0;
    #5 host=1'b0; A=1'b1; B=1'b0;
    #3 host=1'b0; A=1'b0; B=1'b0;
    #4 host=1'b0; A=1'b0; B=1'b1;
    #3 host=1'b0; A=1'b0; B=1'b0;
    #5 host=1'b1; A=1'b0; B=1'b0;
    #5 host=1'b1; A=1'b1; B=1'b1;
end
initial #60 $finish;
endmodule
```

Both A and B press the buzzer without host control ; output X and Y remain 0:

Name	Value
host	0
A	0
B	0
X	0
Y	0



Name	Value
host	0
A	1
B	0
X	0
Y	0



Name	Value
host	0
A	0
B	1
X	0
Y	0

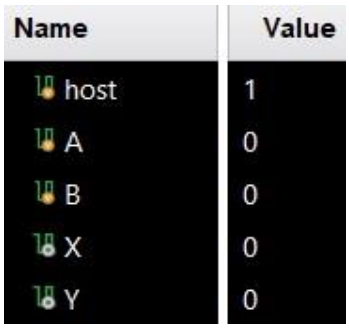
A press the buzzer 1st then B ; X becomes 1 and Y goes to high impedance:

Name	Value
host	1
A	0
B	0
X	0
Y	0

Name	Value
host	1
A	1
B	0
X	1
Y	Z

Name	Value
host	1
A	1
B	1
X	1
Y	Z

B press the buzzer 1st then A ; X goes to high impedance and Y becomes 1:

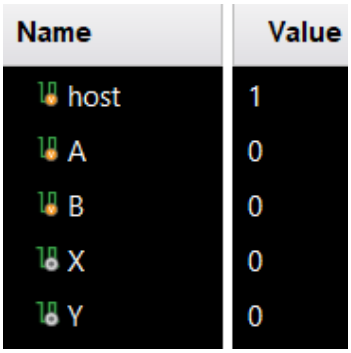


Name	Value
host	1
A	0
B	0
X	0
Y	0

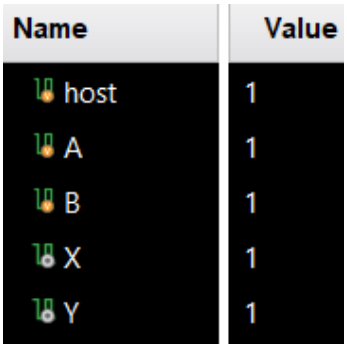
Name	Value
host	1
A	0
B	1
X	Z
Y	1

Name	Value
host	1
A	1
B	1
X	Z
Y	1

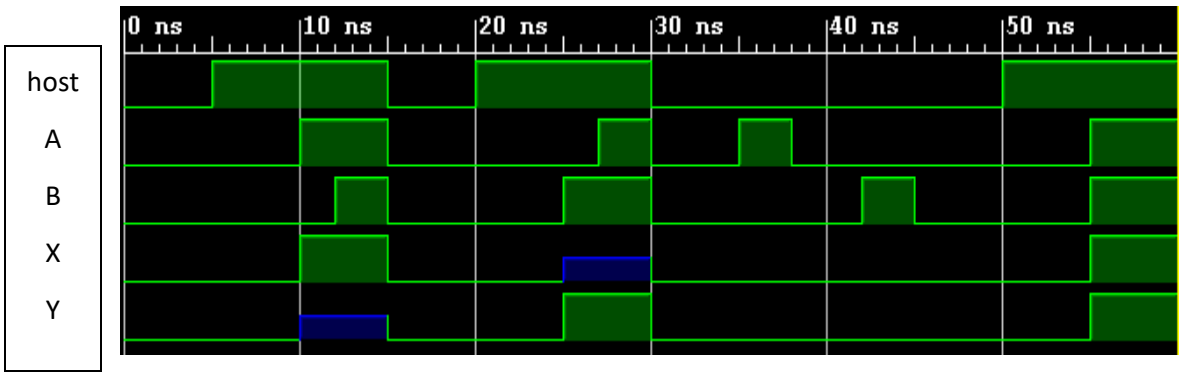
Both A and B press at the same time ; both X and Y become 1:



Name	Value
host	1
A	0
B	0
X	0
Y	0



Name	Value
host	1
A	1
B	1
X	1
Y	1


host
A
B
X
Y
0 ns
10 ns
20 ns
30 ns
40 ns
50 ns