



BIRLA VISHVAKARMA MAHAVIDYALAY
(AN AUTONOMOUS INSTITUTION)
ELECTRONICS ENGINEERING DEPARTMENT
A.Y. 2023-2024

DIGITAL SYSTEM DESIGN

ASSIGNMENT - 1

NAME: SHREYA VERMA

ID NO: 21EL001

SEMESTER: 5TH

BATCH: A – BATCH

BRANCH: ELECTRONICS ENGINEERING

Q1. Write a Verilog code for 2X4 decoder

Code for Test-bench:

```
module testbench;

    reg a,b,en;
    wire [3:0]y;

    decoder2_4 dut(en,a,b,y);

    initial
        begin
            $monitor("en=%b a=%b b=%b y=%b",en,a,b,y);

            en=0; a=1'bx; b=1'bx; #5
            en=1; a=0; b=0; #5
            en=1; a=0; b=1; #5
            en=1; a=1; b=0; #5
            en=1; a=1; b=1; #5

            $dumpfile("decoder2_4.vcd");
            $dumpvars;

            $finish;
        end
endmodule
```

Code for Design:

```
module decoder2_4(en,a,b,y);

    input en,a,b;
    output reg [3:0]y;
    always @(en,a,b)
        begin
            if(en==1)
                begin
                    if(a==1'b0 & b==1'b0) y=4'b0001;
                    else if(a==1'b0 & b==1'b1) y=4'b0010;
                    else if(a==1'b1 & b==1'b0) y=4'b0100;
                    else if(a==1'b1 & b==1'b1) y=4'b1000;
                    else y=4'bxxxx;
                end
            else
                y=4'b0000;
            end
        end
endmodule
```

Output:

```
en=0 a=x b=x y=0000
en=1 a=0 b=0 y=0001
en=1 a=0 b=1 y=0010
en=1 a=1 b=0 y=0100
en=1 a=1 b=1 y=1000
```

Q2. Write a Verilog code for Full subtractor.

Code for Test-bench:

```
module testbench;

reg X,Y,Z;
wire D,B;

Full_Subtractor dut(X,Y,Z,D,B);

initial
begin
    $monitor("X=%b Y=%b Z=%b D=%b B=%b",X,Y,Z,D,B);

    X=0;Y=0;Z=0; #5
    X=0;Y=0;Z=1; #5
    X=0;Y=1;Z=0; #5
    X=0;Y=1;Z=1; #5
    X=1;Y=0;Z=0; #5
    X=1;Y=0;Z=1; #5
    X=1;Y=1;Z=0; #5
    X=1;Y=1;Z=1; #5

    $dumpfile("Full_Subtractor.vcd");
    $dumpvars;
$finish;
end
endmodule
```

Code for Design:

```
module Full_Subtractor(  
    input X, Y, Z,  
    output D, B  
);  
    assign D = X ^ Y ^ Z;  
    assign B = ~X & (Y^Z) | Y & Z;  
  
endmodule
```

Output:

X=0	Y=0	Z=0	D=0	B=0
X=0	Y=0	Z=1	D=1	B=1
X=0	Y=1	Z=0	D=1	B=1
X=0	Y=1	Z=1	D=0	B=1
X=1	Y=0	Z=0	D=1	B=0
X=1	Y=0	Z=1	D=0	B=0
X=1	Y=1	Z=0	D=0	B=0
X=1	Y=1	Z=1	D=1	B=1

Q3. Write a Verilog code for 2-bit comparator.

Code for Test-bench:

```
module testbench;

    reg a0,a1,b0,b1;
    wire p_gr,q_eq,r_less;

    design_2_bit_comparator dut(a0,a1,b0,b1,p_gr,q_eq,r_less);

    initial
        begin
            $monitor("a0=%b a1=%b b0=%b b1=%b p_gr=%b q_eq=%b  
r_less=%b",a0,a1,b0,b1,p_gr,q_eq,r_less);

            a1=0;a0=0; b1=0;b0=0; #5
            a1=0;a0=0; b1=0;b0=1; #5
            a1=0;a0=0; b1=1;b0=0; #5
            a1=0;a0=0; b1=1;b0=1; #5
            a1=0;a0=1; b1=0;b0=0; #5
            a1=0;a0=1; b1=0;b0=1; #5
            a1=0;a0=1; b1=1;b0=0; #5
            a1=0;a0=1; b1=1;b0=1; #5
            a1=1;a0=0; b1=0;b0=0; #5
            a1=1;a0=0; b1=0;b0=1; #5
            a1=1;a0=0; b1=1;b0=0; #5
            a1=1;a0=0; b1=1;b0=1; #5
            a1=1;a0=1; b1=0;b0=0; #5
            a1=1;a0=1; b1=0;b0=1; #5
            a1=1;a0=1; b1=1;b0=0; #5
            a1=1;a0=1; b1=1;b0=1; #5

            $dumpfile("design_2_bit_comparator.vcd");
            $dumpvars;
        $finish;
    end
endmodule
```

Code for Design:

```
module design_2_bit_comparator(  
  
    input  a0,a1,b0,b1,  
  
    output p_gr,q_eq,r_less  
  
);  
    assign p_gr = ((a1 & (~b1)) | ((~b0) & a1 & a0) | (a0 & (~b1) & (~b0)));  
    assign q_eq = ((~(a0^b0)) & ~(a1^b1));  
    assign r_less = (((~a1) & (b1)) | (b0 & (~a0) & (~a1)) | ((~a0) & b1 & b0));  
  
endmodule
```

Output:

```
a0=0 a1=0 b0=0 b1=0 p=0 q=1 r=0  
a0=0 a1=0 b0=1 b1=0 p=0 q=0 r=1  
a0=0 a1=0 b0=0 b1=1 p=0 q=0 r=1  
a0=0 a1=0 b0=1 b1=1 p=0 q=0 r=1  
a0=1 a1=0 b0=0 b1=0 p=1 q=0 r=0  
a0=1 a1=0 b0=1 b1=0 p=0 q=1 r=0  
a0=1 a1=0 b0=0 b1=1 p=0 q=0 r=1  
a0=1 a1=0 b0=1 b1=1 p=0 q=0 r=1  
a0=0 a1=1 b0=0 b1=0 p=1 q=0 r=0  
a0=0 a1=1 b0=1 b1=0 p=1 q=0 r=0  
a0=0 a1=1 b0=0 b1=1 p=0 q=1 r=0  
a0=0 a1=1 b0=1 b1=1 p=0 q=0 r=1  
a0=1 a1=1 b0=0 b1=0 p=1 q=0 r=0  
a0=1 a1=1 b0=1 b1=0 p=1 q=0 r=0  
a0=1 a1=1 b0=0 b1=1 p=1 q=0 r=0  
a0=1 a1=1 b0=1 b1=1 p=0 q=1 r=0
```

Q4. Write a Verilog code for 3-bit binary to gray convertor.

Code for Test-bench:

```
module testbench;

    reg [2:0]binary;
    wire [2:0]gray;

    binary_to_gray dut(binary,gray);

    initial
        begin
            $monitor("binary=%b gray=%b",binary,gray);
            binary[2]=0; binary[1]=0; binary[0]=0; #5
            binary[2]=0; binary[1]=0; binary[0]=1; #5
            binary[2]=0; binary[1]=1; binary[0]=0; #5
            binary[2]=0; binary[1]=1; binary[0]=1; #5
            binary[2]=1; binary[1]=0; binary[0]=0; #5
            binary[2]=1; binary[1]=0; binary[0]=1; #5
            binary[2]=1; binary[1]=1; binary[0]=0; #5
            binary[2]=1; binary[1]=1; binary[0]=1; #5

            $dumpfile("design_2_bit_comparator.vcd");
            $dumpvars;
        $finish;
        end
    endmodule
```


Code for Design:

```
module binary_to_gray (  
    input [2:0]binary,  
    output reg [2:0]gray  
);  
  
    always @(binary)  
    begin  
        gray[2] = binary[2];  
        gray[1] = binary[2] ^ binary[1];  
        gray[0] = binary[1] ^ binary[0];  
    end  
  
endmodule
```

Output:

```
binary=000 gray=000  
binary=001 gray=001  
binary=010 gray=011  
binary=011 gray=010  
binary=100 gray=110  
binary=101 gray=111  
binary=110 gray=101  
binary=111 gray=100
```

Q5. Write a Verilog code for BCD to excess 3 converters.

Code for Test-bench:

```
module testbench;

    reg [3:0]bcd;
    wire [3:0]excess3;

    Bcd_excess3 dut(bcd,excess3);

    initial
        begin
            $monitor("bcd=%b excess=%b",bcd,excess3);

            bcd[3]=0; bcd[2]=0; bcd[1]=0; bcd[0]=0; #5
            bcd[3]=0; bcd[2]=0; bcd[1]=0; bcd[0]=1; #5
            bcd[3]=0; bcd[2]=0; bcd[1]=1; bcd[0]=0; #5
            bcd[3]=0; bcd[2]=0; bcd[1]=1; bcd[0]=1; #5
            bcd[3]=0; bcd[2]=1; bcd[1]=0; bcd[0]=0; #5
            bcd[3]=0; bcd[2]=1; bcd[1]=0; bcd[0]=1; #5
            bcd[3]=0; bcd[2]=1; bcd[1]=1; bcd[0]=0; #5
            bcd[3]=0; bcd[2]=1; bcd[1]=1; bcd[0]=1; #5
            bcd[3]=1; bcd[2]=0; bcd[1]=0; bcd[0]=0; #5
            bcd[3]=1; bcd[2]=0; bcd[1]=0; bcd[0]=1; #5

            $dumpfile("Bcd_excess3.vcd");
            $dumpvars;
            $finish;
        end
endmodule
```

Code for Design:

```
module Bcd_excess3(  
    input [3:0] bcd,  
    output [3:0] excess3  
);  
    assign excess3[3]= bcd[3] | bcd[2] & bcd[1] | bcd[2] & bcd[0];  
    assign excess3[2]= (~bcd[2]) & bcd[1] | (~bcd[2]) & bcd[0] | bcd[2] & (~bcd[1]) & (~bcd[0]);  
    assign excess3[1]= bcd[1] & bcd[0] | (~bcd[1]) & (~bcd[0]);  
    assign excess3[0]= (~bcd[0]);  
endmodule
```

Output:

```
bcd=0000  excess=0011  
bcd=0001  excess=0100  
bcd=0010  excess=0101  
bcd=0011  excess=0110  
bcd=0100  excess=0111  
bcd=0101  excess=1000  
bcd=0110  excess=1001  
bcd=0111  excess=1010  
bcd=1000  excess=1011  
bcd=1001  excess=1100
```