

DIGITAL SYSTEM DESIGN LAB

LAB #05



Spring 2021

CSE308L DSD LAB

Submitted by: **Shah Raza**

Registration No. : **18PWCSE1658**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: _____

Submitted to:

Engr. Madiha Sher

Friday, May 28, 2021

Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar

Objectives:

This lab will enable students to:

- Code using Behavioral level modeling
- Implement multiplexer and de-multiplexer and decoder

Task # 01: Implementation of 8x1 multiplexer (using case)

Problem Analysis:

Truth Table:

S1	S2	S3	O
0	0	0	I1
0	0	1	I2
0	1	0	I3
0	1	1	I4
1	0	0	I5
1	0	1	I6
1	1	0	I7
1	1	1	I8

Code:

Multiplexer:

```
module Mux8x1(Sel,I0,I1,I2,I3,I4,I5,I6,I7,OUT);
```

```
    input [2:0] Sel;
```

```
    input [7:0] I0,I1,I2,I3,I4,I5,I6,I7;
```

```
    output [7:0] OUT;
```

```
reg [7:0] OUT;
```

```
always @(*)
```

```
    case (Sel)
```

```
        3'b000: OUT = I0;
```

```
        3'b001: OUT = I1;
```

```
        3'b010: OUT = I2;
```

```
        3'b011: OUT = I3;
```

```
        3'b100: OUT = I4;
```

```
        3'b101: OUT = I5;
```

```
        3'b110: OUT = I6;
```

```
        3'b111: OUT = I7;
```

```
    endcase
```

```
endmodule
```

TestBench:

```
module testMux;
```

```
    reg [2:0] S;
```

```
    reg [7:0] I0,I1,I2,I3,I4,I5,I6,I7;
```

```
    wire [7:0] O;
```

```
    Mux8x1 m1(S,I0,I1,I2,I3,I4,I5,I6,I7,O);
```

```
    initial
```

```
    begin
```

```
        I0 = 8'h0;I1 = 8'h1;I2 = 8'h2;I3 = 8'h3;I4 = 8'h4;I5 = 8'h5;I6 = 8'h6;I7 = 8'h7;
```

```
        $display("I0 = %b, I1 = %b,I2 = %b,I3 = %b,I4 = %b,I5 = %b,I6 = %b,I7= %b",I0,I1,I2,I3,I4,I5,I6,I7);
```

```
        $display("S                                O");
```

```
        $monitor (" %b %b",S,O);
```

```
        S = 3'b000;
```

```
        #5
```

```
        S = 3'b001;
```

```

#5
S = 3'b010;

#5
S = 3'b011;

#5
S = 3'b100;

#5
S = 3'b101;

#5
S = 3'b110;

#5
S = 3'b111;

end

endmodule

```

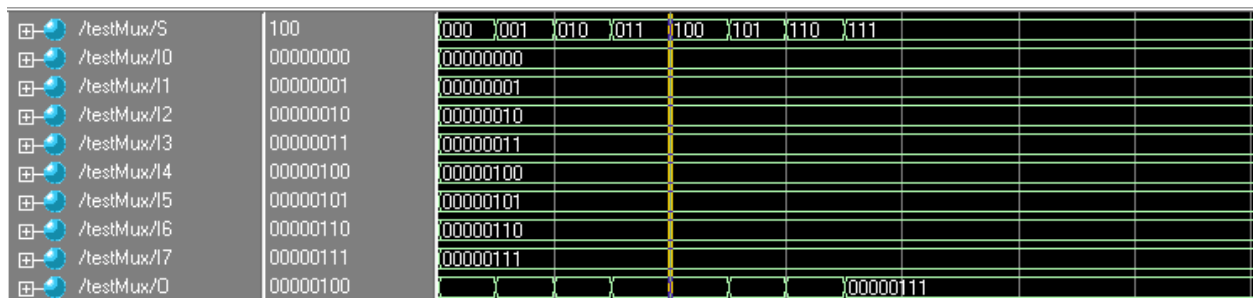
Output:

```

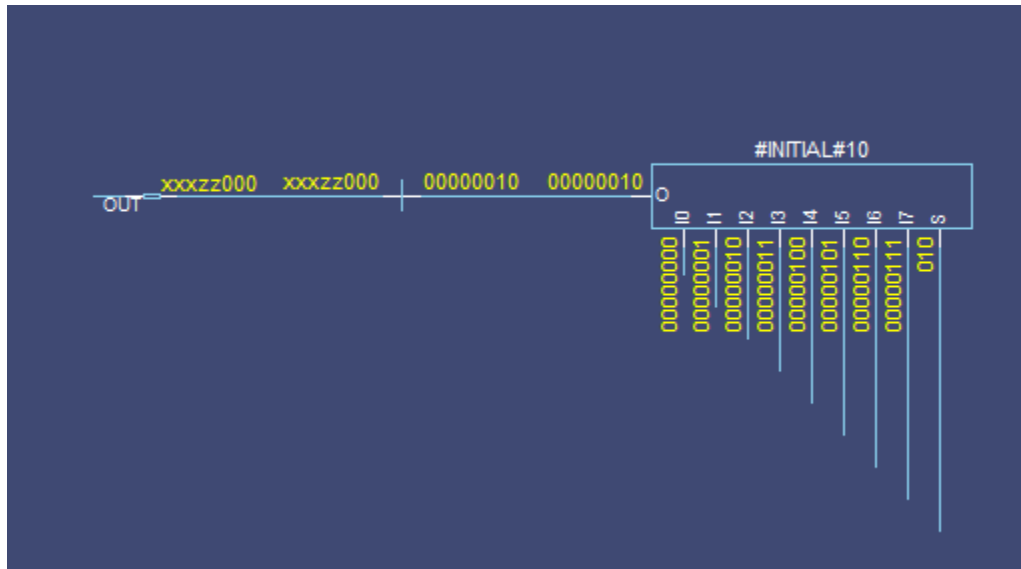
# IO = 00000000, I1 = 00000001, I2 = 00000010, I3 = 00000011, I4 = 00000100, I5 = 00000101, I6 = 00000110, I7 = 00000111
# S   0
# 000 00000000
# 001 00000001
# 010 00000010
# 011 00000011
# 100 00000100
# 101 00000101
# 110 00000110
# 111 00000111

```

Waveform:



Dataflow:



Task # 02: Implementation of 1x8 de-multiplexer (using if/else)

Problem Analysis:

Truth Table:

Data Input	Select Inputs			Outputs							
D	S ₂	S ₁	S ₀	Y ₇	Y ₆	Y ₅	Y ₄	Y ₃	Y ₂	Y ₁	Y ₀
D	0	0	0	0	0	0	0	0	0	0	D
D	0	0	1	0	0	0	0	0	0	D	0
D	0	1	0	0	0	0	0	0	D	0	0
D	0	1	1	0	0	0	0	D	0	0	0
D	1	0	0	0	0	0	D	0	0	0	0
D	1	0	1	0	0	D	0	0	0	0	0
D	1	1	0	0	D	0	0	0	0	0	0
D	1	1	1	D	0	0	0	0	0	0	0

Code:

De-Multiplexer:

```
module DeMux1x8 (D,S,Y);  
    input D;  
    input [2:0] S;
```

```
output [7:0] Y;
reg [7:0] Y;

always @(*)
begin
    if(S==3'b000)
    begin
        Y = 8'b00000000;
        Y[0] = D;
    end
    else if(S==3'b001)
    begin
        Y = 8'b00000000;
        Y[1] = D;
    end
    else if(S==3'b010)
    begin
        Y = 8'b00000000;
        Y[2] = D;
    end
    else if(S==3'b011)
    begin
        Y = 8'b00000000;
        Y[3] = D;
    end
    else if(S==3'b100)
    begin
        Y = 8'b00000000;
        Y[4] = D;
    end
    else if(S==3'b101)
    begin
        Y = 8'b00000000;
        Y[5] = D;
    end
    else if(S==3'b110)
    begin
        Y = 8'b00000000;
        Y[6] = D;
    end
    else if(S==3'b111)
    begin
        Y = 8'b00000000;
        Y[7] = D;
    end
end

end
endmodule
```

TestBench:

```
module testDeMux;
    reg [2:0] S;
    wire [7:0] Y;

    reg D;

    DeMux1x8 dm1(D,S,Y);

    initial
    begin
        D = 1'b1;
        $display("D = %b",D);
        $display("S Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7");
        $monitor (" %b %b %b %b %b %b %b %b %b",S,Y[0],Y[1],Y[2],Y[3],Y[4],Y[5],Y[6],Y[7]);
        S = 3'b000;
        #5
        S = 3'b001;
        #5
        S = 3'b010;
        #5
        S = 3'b011;
        #5
        S = 3'b100;
        #5
        S = 3'b101;
        #5
        S = 3'b110;
        #5
        S = 3'b111;
    end
endmodule
```

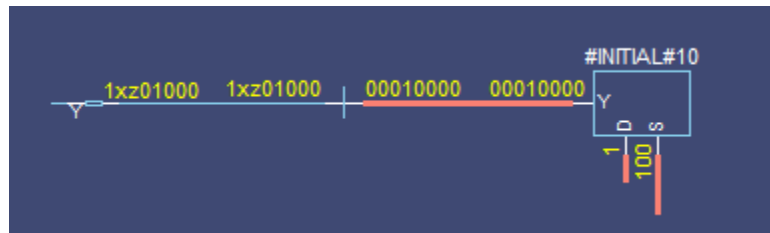
Output:

```
# D = 1
# S Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
# 000 1 0 0 0 0 0 0 0
# 001 0 1 0 0 0 0 0 0
# 010 0 0 1 0 0 0 0 0
# 011 0 0 0 1 0 0 0 0
# 100 0 0 0 0 1 0 0 0
# 101 0 0 0 0 0 1 0 0
# 110 0 0 0 0 0 0 1 0
# 111 0 0 0 0 0 0 0 1
```

Waveform:

/testDeMux/S	100	000	001	010	011	100	101	110	111
/testDeMux/Y	00010000	00000001	00000010	00000100	00001000	00010000	00100000	01000000	10000000
/testDeMux/D	1								

Dataflow:



Task # 03: Implementation of 3x8 decoder

Code:

Decoder:

```
module decoder3x8(sel,out);
    input [2:0] sel;
    output [7:0] out;
    reg [7:0] out;

    always @ (sel)
        case(sel)
            3'b000:
                begin
                    out = 8'b00000000;
                    out[0] = 1'b1;
                end
            3'b001:
                begin
                    out = 8'b00000000;
                    out[1] = 1'b1;
                end
            3'b010:
                begin
                    out = 8'b00000000;
                    out[2] = 1'b1;
                end
            3'b011:
                begin
                    out = 8'b00000000;
                    out[3] = 1'b1;
                end
        end
end
```



```

        3'b100:
        begin
            out = 8'b000000000;
            out[4] = 1'b1;
        end
        3'b101:
        begin
            out = 8'b000000000;
            out[5] = 1'b1;
        end
        3'b110:
        begin
            out = 8'b000000000;
            out[6] = 1'b1;
        end
        3'b111:
        begin
            out = 8'b000000000;
            out[7] = 1'b1;
        end
    end
endcase
endmodule

```

TestBench:

```

module test_decoder3x8;
    reg [2:0]s;
    wire [7:0]o;

    decoder3x8 de(s,o);

    initial
    begin
        $display("Sel out");
        $monitor("%b          %b",s,o);
        s=3'b000;
        #5
        s=3'b001;
        #5
        s=3'b010;
        #5
        s=3'b011;
        #5
        s=3'b100;
        #5
        s=3'b101;
        #5
        s=3'b110;
    end
endmodule

```

```

        #5
        s=3'b111;
    end
endmodule

```





Output:

```

# Sel out
# 000 00000001
# 001 00000010
# 010 00000100
# 011 00001000
# 100 00010000
# 101 00100000
# 110 01000000
# 111 10000000

```

Waveform:

  /test_decoder3x8/s	101	000	001	010	011	100	101	110	111	
  /test_decoder3x8/o	00100000								10000000	

Dataflow:

