DIGITAL DESIGN

LAB9   COMBINATORIAL CIRCUIT3

2022 SUMMER TERM

LAB8

- Combinational circuit(3)
  - Multiplexer
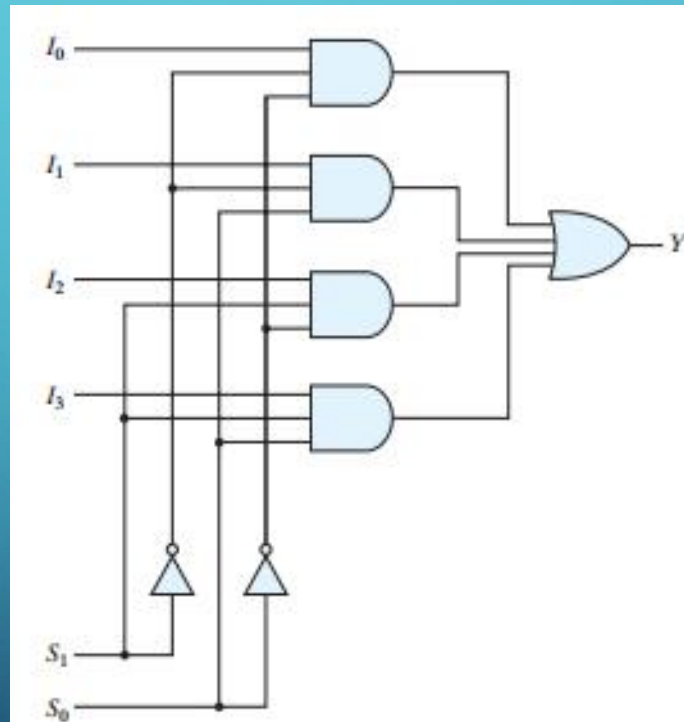 - Demultiplexer

- Practice

MULTIPLEXER

- a **Multiplexer** (or **mux**) is a device that selects one of several input signals and forwards the selected input into a single line.

- A multiplexer of 2^n inputs has n select lines, which are used to select which input line to be sent to the output.

- Multiplexers are mainly used to increase the amount of data that can be sent over the network within a certain amount of time and bandwidth. A multiplexer is also called a **data selector**.

- Multiplexers can also be used to implement Boolean functions of multiple variables.

MULTIPLEXER

$$Y = m_0.D_0 + m_1.D_1 + m_2.D_2 + m_3.D_3$$

selection input		output
s1	s0	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

function table for 4–to-1-line multiplexer

```verilog
module multiplexer(
    input w,
    input x,
    input y,
    input z,
    input [1:0] s, //select
    output reg o
);
always @*
begin
    case (s)
        2'b00: o = w;
        2'b01: o = x;
        2'b10: o = y;
        2'b11: o = z;
    endcase
end
endmodule
```
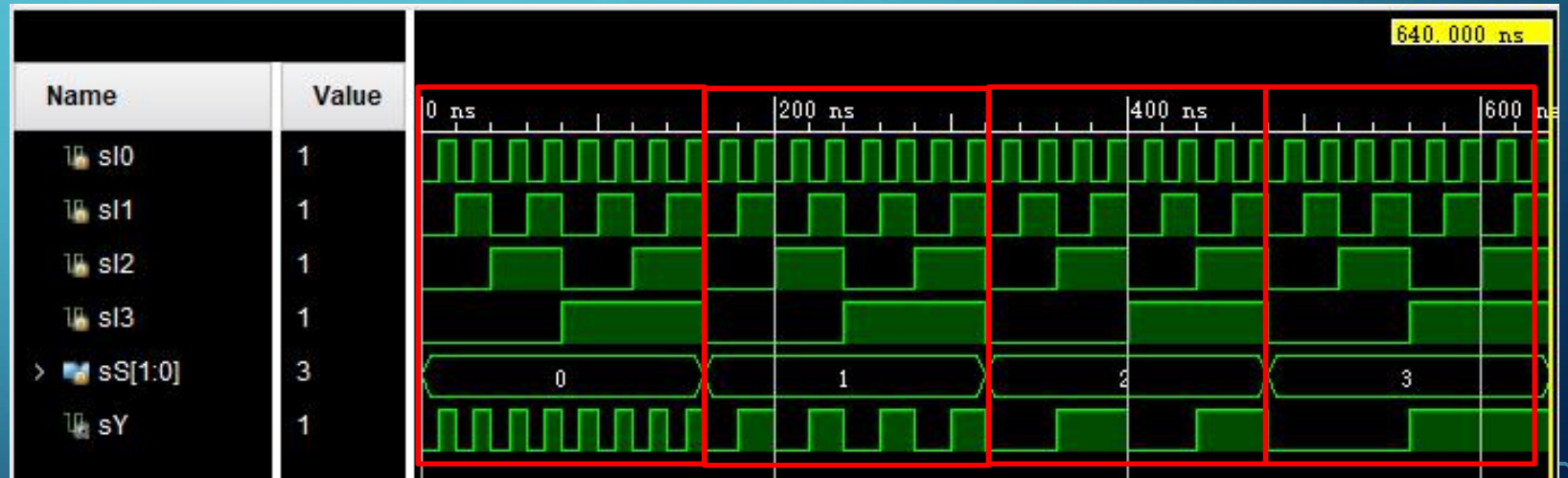
MULTIPLEXER

$$Y = m_0.D_0 + m_1.D_1 + m_2.D_2 + m_3.D_3$$

```
module multiplexer_tb();
  reg sI0, sI1, sI2, sI3;
  reg [1:0]sS;
  wire sY;
  multiplexer u(sI0, sI1, sI2, sI3, sS, sY);
  initial
  begin
    {sS, sI3, sI2, sI1, sI0} = 6'b000000;
    repeat(63) #10 {sS, sI3, sI2, sI1, sI0} = {sS, sI3, sI2, sI1, sI0} + 1;
    #10 $finish;
  end
endmodule
```

selection input		output
s1	s0	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

function table for 4–to–1–line multiplexer

MULTIPLEXER(74151)

MUX

EN'
S0
S1
S2

D0
D1
D2
D3
D4
D5
D6
D7   74151

Y

W

Logic diagram
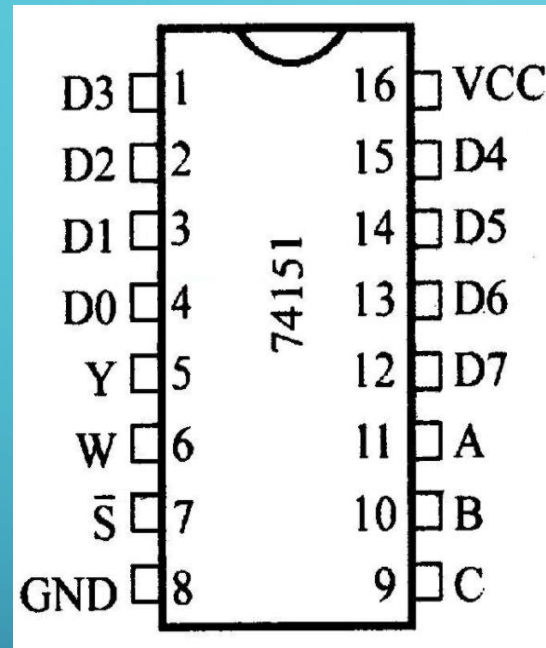
Pin diagram

Physical photo

- **74151**: 8-to-1-line multiplexer

- 1 strobe input (low level effective)

- 3 select lines

- 8 data inputs

- Y is the data output

- W is the reverse code of data output

MULTIPLEXER(74151) COUTINUED

	inputs			output	
EN	S₂	S1	S0	Y	W
1	X	X	X	0	1
0	0	0	0	D0	D0'
0	0	0	1	D1	D1'
0	0	1	0	D2	D2'
0	0	1	1	D3	D3'
0	1	0	0	D4	D4'
0	1	0	1	D5	D5'
0	1	1	0	D6	D6'
0	1	1	1	D7	D7'

function table for 74151

```
module multiplexer74151( EN, S2, S1, S0, D7, D6,
D5, D4, D3, D2, D1, D0, Y, W);
    input EN, S2, S1, S0, D7, D6, D5,
    D4, D3, D2, D1, D0;
    output reg Y;
    output W;
    always @*
    if (~EN)
        case ({S2, S1, S0})
            3'b000: Y = D0;
            3'b001: Y = D1;
            3'b010: Y = D2;
            3'b011: Y = D3;
            3'b100: Y = D4;
            3'b101: Y = D5;
            3'b110: Y = D6;
            3'b111: Y = D7;
        endcase
    else
        Y = 1'b0;
    assign W=~Y;
endmodule
```
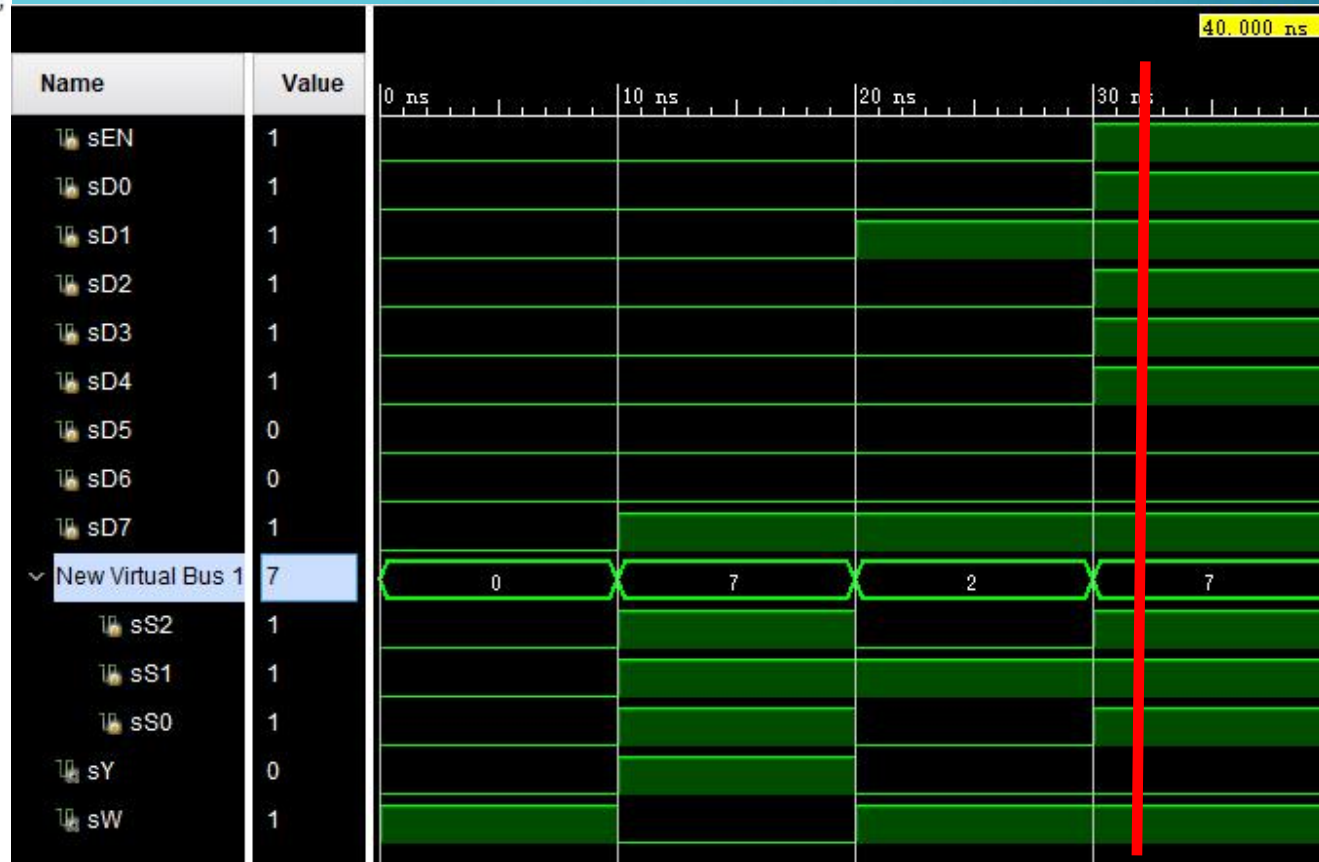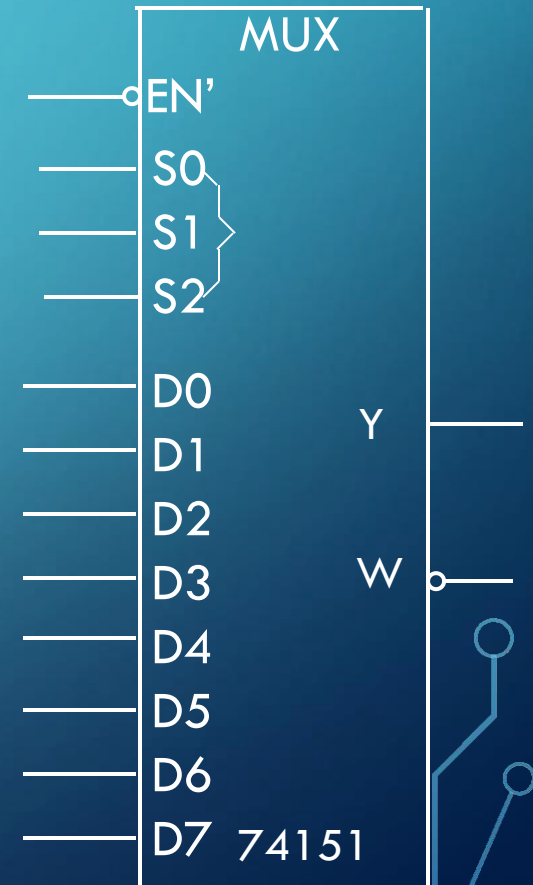
MULTIPLEXER(IMPLEMENT BOOLEAN FUNCTIONS-1)

- Use 74151 implement the following logic function.

$$F(A, B, C) = \overline{A}\overline{C} + \overline{B}\overline{C} + \overline{A}B + BC$$

$$= \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}B\overline{C} + \overline{A}BC + ABC + \overline{A}BC$$

$$= \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + \overline{A}BC + ABC$$

$$= \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + ABC$$

$$= m_0.1 + m_1.0 + m_2.1 + m_3.1 + m_4.1 + m_5.0 + m_6.0 + m_7.1$$

$$Y = m_0.D_0 + m_1.D_1 + m_2.D_2 + m_3.D_3 + m_4.D_4 + m_5.D_5 + m_6.D_6 + m_7.D_7$$

MUX
EN'
S0
S1
S2
D0
D1   Y
D2
D3   W
D4
D5
D6
D7   74151

MULTIPLEXER(IMPLEMENT BOOLEAN FUNCTIONS-2)

$$F(A, B, C) = \overline{A}\,\overline{C} + \overline{B}\,\overline{C} + \overline{A}B + BC$$

```
module multiplexer74151( EN, S2, S1, S0, D7, D6, D5, D4, D3, D2, D1, D0, Y, W);
    input EN, S2, S1, S0, D7, D6, D5,
     D4, D3, D2, D1, D0;
    output reg Y;
    output W;
    always @*
    if (~EN)
        case ({S2, S1, S0})
            3'b000: Y = D0;
            3'b001: Y = D1;
            3'b010: Y = D2;
            3'b011: Y = D3;
            3'b100: Y = D4;
            3'b101: Y = D5;
            3'b110: Y = D6;
            3'b111: Y = D7;
        endcase
    else
        Y = 1'b0;
    assign W=~Y;
endmodule
```

```
module fun_a_b_c(input A, B, C, output F );
    assign F=( (~A)&(~C) ) | ( (~B)&(~C) )  | ((~A)&B) | (B&C) ;
endmodule
```

$$F(A, B, C) = m_0.1 + m_1.0 + m_2.1 + m_3.1 + m_4.1 + m_5.0 + m_6.0 + m_7.1$$

```
module fun_a_b_c_use_mux(input A,B,C, output F);
wire sen,sd7,sd6,sd5,sd4,sd3,sd2,sd1,sd0;
wire snf;

assign {sen,sd7,sd5,sd4,sd3,sd2,sd1,sd0}= 9'b0_1001_1101;

multiplexer74151 u74151(.EN(sen),
.S2(A),.S1(B),.S0(C),
.D7(sd7),.D6(sd6),.D5(sd5),.D4(sd4),.D3(sd3),.D2(sd2),.D1(sd1),.D0(sd0),
.Y(F),.W(snf));

endmodule
```

MULTIPLEXER(IMPLEMENT BOOLEAN FUNCTIONS-3)

```verilog
`timescale 1ns / 1ps
///////////////////////////////////////////////////////////////////////////...
module fun_abc_sim( );
reg sa, sb, sc;
wire sf, sf_mux;
fun_a_b_c uf(.A(sa),.B(sb),.C(sc),.F(sf));
fun_a_b_c_use_mux uf_mux(.A(sa),.B(sb),.C(sc),.F(sf_mux));
/*...*/
initial
begin
    {sa, sb, sc} = 3'b000;
    repeat(7)
    begin
        #100 {sa, sb, sc} = {sa, sb, sc} + 1;
        $display($time,"{sa, sb, sc}=%d_%d_%d sf=%d sf_mux=%d", sa, sb, sc, sf, sf_mux);
    end
    #100 $finish();
end
endmodule
```

$$F(A, B, C) = \overline{A}\,\overline{C} + \overline{B}\,\overline{C} + \overline{A}B + BC$$

```
100{sa, sb, sc}=0_0_1 sf=1 sf_mux=1
200{sa, sb, sc}=0_1_0 sf=0 sf_mux=0
300{sa, sb, sc}=0_1_1 sf=1 sf_mux=1
400{sa, sb, sc}=1_0_0 sf=1 sf_mux=1
500{sa, sb, sc}=1_0_1 sf=1 sf_mux=1
600{sa, sb, sc}=1_1_0 sf=0 sf_mux=0
700{sa, sb, sc}=1_1_1 sf=0 sf_mux=0
$finish called at time : 800 ns : File "D:/xilinx_wor
```

DE-MULTIPLEXER

- a **De-multiplexer** (or **De-mux**) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input.

selection input		output			
S1	S0	Y3	Y2	Y1	Y0
0	0	0	0	0	D
0	1	0	0	D	0
1	0	0	D	0	0
1	1	D	0	0	0

function table of 1-to-4 de-multiplexer
D is the data input

```
module demultiplexer(
    input D,
    input [1:0] S,
    output reg Y0,
    output reg Y1,
    output reg Y2,
    output reg Y3
);
    always@*
    begin
        case (S)
        2'b00: {Y3, Y2, Y1, Y0}={1'b0, 1'b0, 1'b0, D};
        2'b01: {Y3, Y2, Y1, Y0}={1'b0, 1'b0, D, 1'b0};
        2'b10: {Y3, Y2, Y1, Y0}={1'b0, D, 1'b0, 1'b0};
        2'b11: {Y3, Y2, Y1, Y0}={D, 1'b0, 1'b0, 1'b0};
        endcase;
    end
endmodule
```

```
module demultiplexer_tb();
    reg [1:0] sS;
    reg sD;
    wire sY0, sY1, sY2, sY3;
    demultiplexer u(sD, sS, sY0, sY1, sY2, sY3);
    initial
    begin
        {sD, sS} = 3'b000;
        repeat(7) #10 {sD, sS} = {sD, sS}+1;
        #10 $finish;
    end
endmodule
```

Name	Value
sS[1:0]	3
sD	1
sY0	0
sY1	0
sY2	0
sY3	1
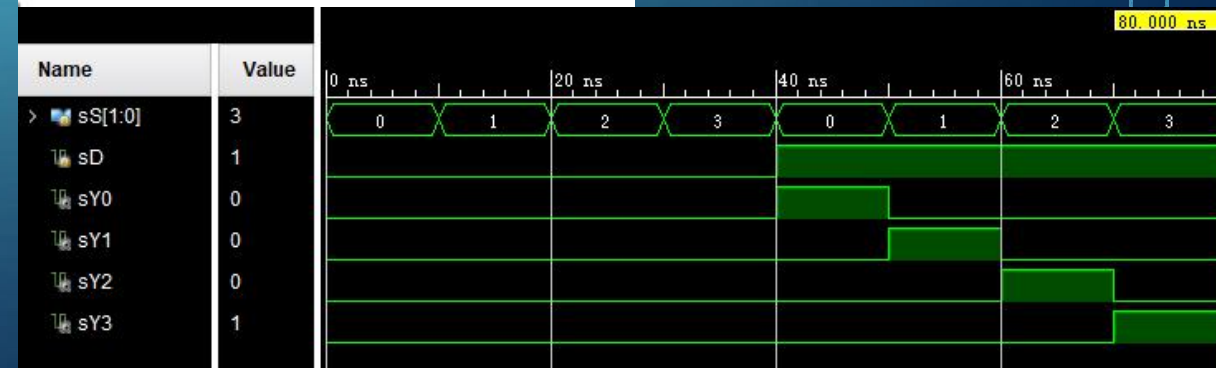
PRACTICE 1

- Is there any relationship between Decoder and De-mux?

- Please try to implement a 1-8 De-mux by using Decoder 74138

- Tips: Use $\overline{G_{2A}}$ pin as the data input of De-mux
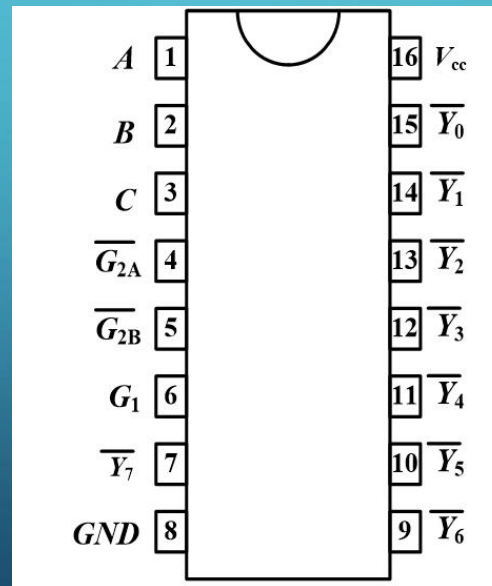
G1	G2A'	G2B'	C	B	A	Y0'	Y1'	Y2'	Y3'	Y4'	Y5'	Y6'	Y7'
0	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	1	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	1	1	0	1	1	1	1	1	1
1	0	0	0	1	0	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	0	0	1	1	1	1	0	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	0	1	1	0	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0

truth table for 74138 decoder

PRACTICE 2

- Use 74151(8-to-1-line multiplexer) realize the following logic function

$$Y = A'B'C'D' + BC'D + A'C'D + A'BCD + ACD$$

- Do the design

- Verify the function of your design

- Create the constraint file, do the synthetic and implementation, generate the bitstream file and program the device, then test on EGO1.