

A decorative graphic on the left side of the slide, consisting of a network of white lines and circles on a teal background, resembling a circuit board or a tree structure.

DIGITAL DESIGN

LAB7 COMBINATORIAL CIRCUIT - MUX - DMUX

2022 FALL TERM @ CSE . SUSETCH

LAB7

- Combinational circuit(3)
 - Multiplexer
 - The logical expression of Multiplexer
 - Using Multiplexer to implement a combinational circuit
 - Demultiplexer
- Practices

MULTIPLEXER

- a **Multiplexer** (or **mux**) is a device that selects one of several input signals and forwards the selected input to the output.
- A multiplexer of 2^n inputs has n select lines. Select lines are used to select one of the input line to be sent to the output. *use n select lines to determine which of the 2^n inputs to be used*
- 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(4-TO-1-LINE MULTIPLEXER1)

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

$$Y = (s_1' \cdot s_0') \cdot D_0 + (s_1' \cdot s_0) \cdot D_1 + (s_1 \cdot s_0') \cdot D_2 + (s_1 \cdot s_0) \cdot D_3$$

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

1. There are **4** input data ports(D0,D1,D2,D3), **2** select lines(S1,S0), **1** output port(Y).

The value of output Y is determined by the value of select lines and the related input data port.

2. "s1" is the **MSB** of the select lines, "s0" is the **LSB** of the select lines.

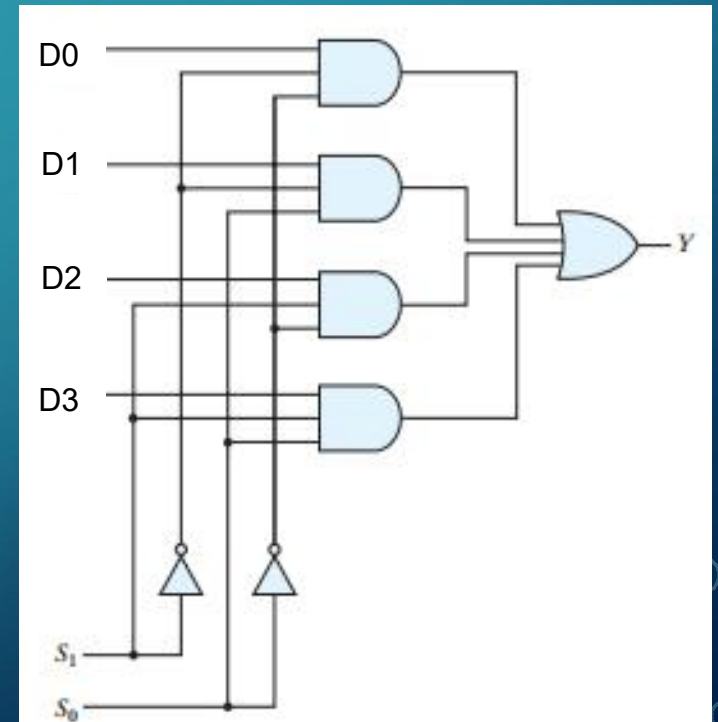


表 2.3 3 个二进制变量的最小项和最大项

x	y	z	最 小 项		最 大 项	
			项	标 识	项	标 识
0	0	0	$x'y'z'$	m_0	$x+y+z$	M_0
0	0	1	$x'y'z$	m_1	$x+y+z'$	M_1
0	1	0	$x'yz'$	m_2	$x+y'+z$	M_2
0	1	1	$x'yz$	m_3	$x+y'+z'$	M_3
1	0	0	$xy'z'$	m_4	$x'+y+z$	M_4
1	0	1	$xy'z$	m_5	$x'+y+z'$	M_5
1	1	0	xyz'	m_6	$x'+y'+z$	M_6
1	1	1	xyz	m_7	$x'+y'+z'$	M_7

MULTIPLEXER(4-TO-1-LINE MULTIPLEXER2)

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

$$Y = (s_1' \cdot s_0') \cdot D_0 + (s_1' \cdot s_0) \cdot D_1 + (s_1 \cdot s_0') \cdot D_2 + (s_1 \cdot s_0) \cdot D_3$$

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

1. There are **4** input data ports(D0,D1,D2,D3), **2** select lines(S1,S0), **1** output port(Y).

The value of output Y is determined by the value of select lines and the related input data port.

2. “s1” is the **MSB** of the select lines, “s0” is the **LSB** of the select lines.

```
module multiplexer(  
    input D0,D1,D2,D3, //data-input  
    input [1:0] s, //select  
    output reg o  
);  
    always @ * begin  
        case(s)  
            2'b00: o = D0;  
            2'b01: o = D1;  
            2'b10: o = D2;  
            2'b11: o = D3;  
        endcase  
    end  
endmodule
```

MULTIPLEXER(4-TO-1-LINE MULTIPLEXER3)

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

$$Y = (s_1'.s_0').D_0 + (s_1'.s_0).D_1 + (s_1.s_0').D_2 + (s_1.s_0).D_3$$

```
module multiplexer(  
    input D0,D1,D2,D3, //data-input  
    input [1:0] s, //select  
    output reg o  
);  
    always @ * begin  
        case(s)  
            2'b00: o = D0;  
            2'b01: o = D1;  
            2'b10: o = D2;  
            2'b11: o = D3;  
        endcase  
    end  
endmodule
```

```
module mux_tb();  
    reg sD0, sD1, sD2, sD3;  
    reg [1:0] sS;  
    wire sY;  
    //module multiplexer( input D0,D1,D2,D3, input [1:0] s, output reg o);  
    multiplexer u(sD0, sD1, sD2, sD3, sS, sY);  
    initial begin  
        {sS, sD0, sD1, sD2, sD3} = 6'b0;  
        repeat(63) #10 {sS, sD0, sD1, sD2, sD3} = {sS, sD0, sD1, sD2, sD3} + 1;  
        #10 $finish;  
    end  
endmodule
```

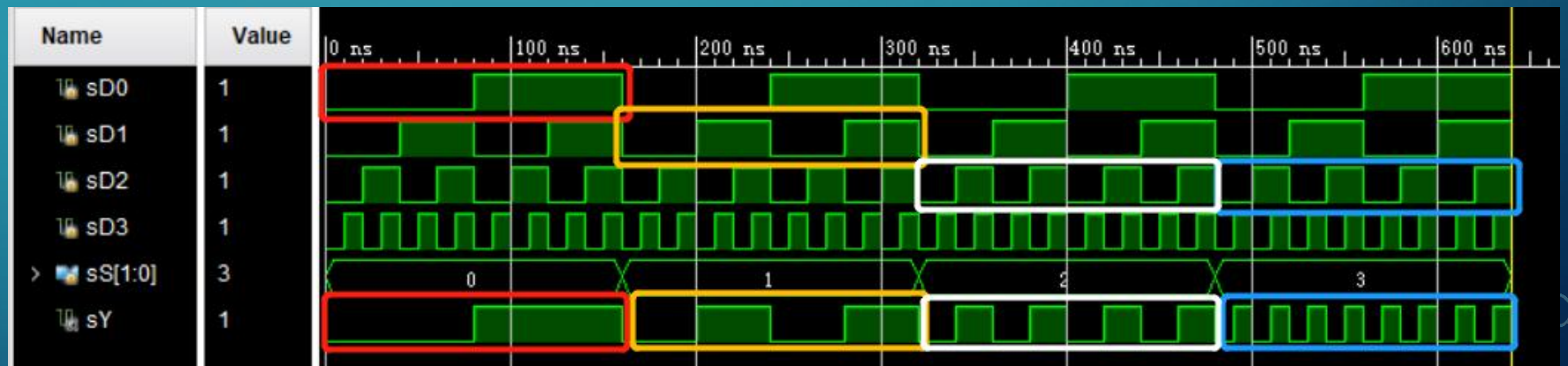

MULTIPLEXER(4-TO-1-LINE MULTIPLEXER4)

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

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

$$Y = (s1'.s0').D0 + (s1'.s0).D1 + (s1.s0').D2 + (s1.s0).D3$$



MULTIPLEXER(74151:8-TO-1-LINE MULTIPLEXER1)

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

1. EN is low level effective.

While EN is effective, the circuit work as a 8-to-1-line multiplexer.

2. There are 8 input data ports, 3 select lines.

The value of output Y is determined by the value of select lines and the related input data port, W is the invert of Y.

3. "s2" is the MSB of the select lines, "s0" is the LSB of the select lines.

```
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(74151:8-TO-1-LINE MULTIPLEXER2)

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

While EN is effective, the logical express of 74151 (about output Y and data inputs and the select lines) is :

S2 is MSB, S0 is LSB

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

The logical express of 74151 is :

$$Y = EN'$$

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

$$W = Y'$$

MULTIPLEXER(IMPLEMENT BOOLEAN FUNCTIONS1)

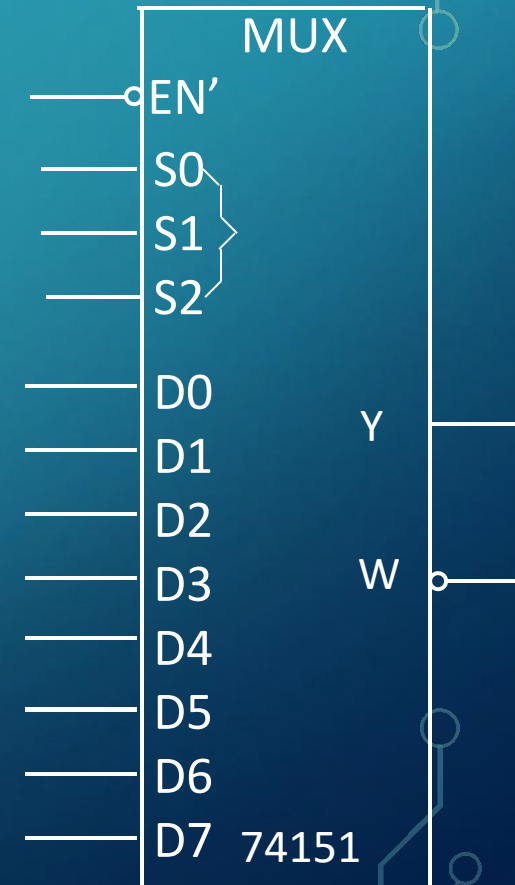
- Use 74151 implement the following logic function.

$$\begin{aligned} F(A, B, C) &= \overline{A}\overline{C} + \overline{B}\overline{C} + \overline{A}B + BC \\ &= \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}BC + \overline{A}BC + ABC + \overline{A}BC \\ &= \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + \overline{A}BC + ABC \\ &= \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}\overline{C} + ABC \\ &= \end{aligned}$$

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

While EN is effective, the logical express of 74151 (about output Y and data inputs and the select lines) is :

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



MULTIPLEXER(IMPLEMENT BOOLEAN FUNCTIONS2)

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

```
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)

```
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
```

PRACTICE1

Use 74151(8-to-1-line multiplexer) realize the following logic function
$$F(A, B, C) = \overline{A}\overline{C} + \overline{B}\overline{C} + \overline{A}B + BC$$

- It is asked that A is the LSB of select lines , C is the MSB of select lines.
- Do the design and verify the function of your design.

PRACTICE2

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 and 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 the minisys develop board.

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					

DE-MULTIPLEXER1

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
```

DE-MULTIPLEXER2

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_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
```



PRACTICE3

1. Is there any relationship between Decoder and De-mux?
2. Please try to implement a 1-4 De-mux by using a 2-4 Decoder