Course Code: CS223

Course Name: Digital Design

Section: 1

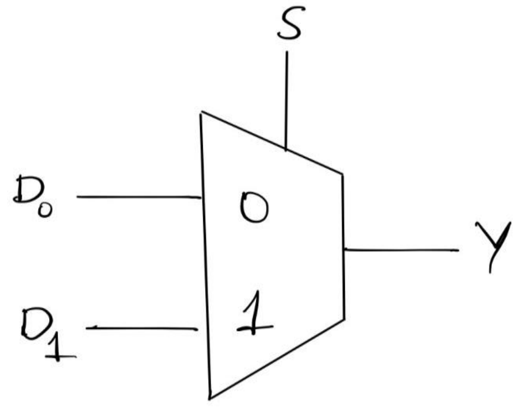
Lab 3

Mustafa Mert Gülhan

22201895

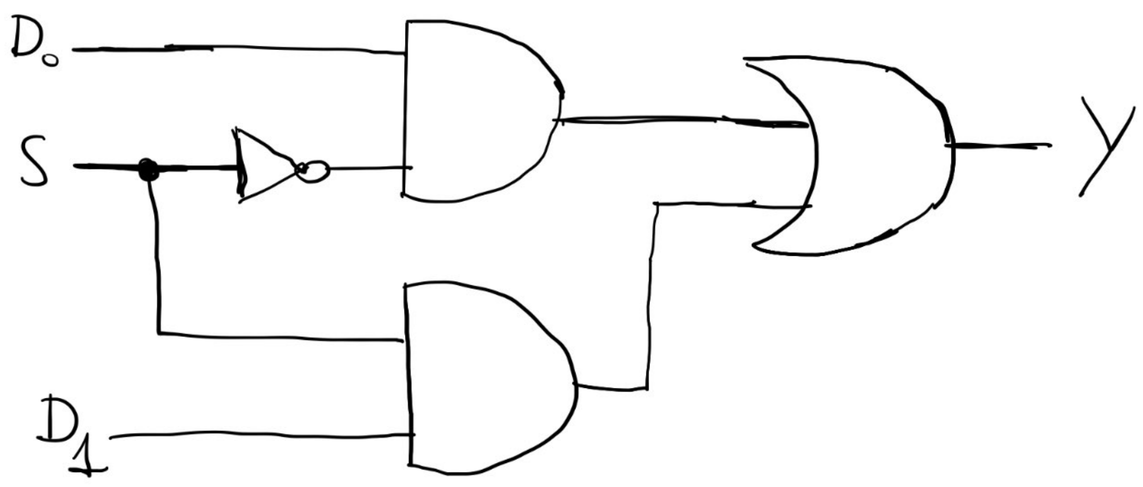
Date: 17.03.2024

Graphical Symbol and Truth Table for 2:1 Mux



S	D_0	D_1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Logic Diagram for 2:1 Mux



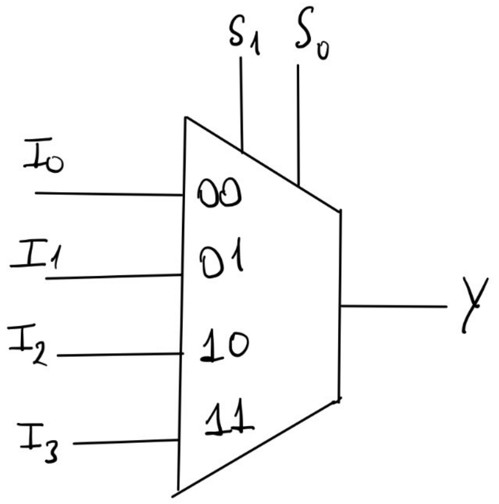
Behavioral 2:1 Multiplexer Module

```
module twomux_behavioral(input logic s,d0,d1,
output logic y);
    assign y = (d0 & (~s)) | (d1 & s);
endmodule
```

<u>Behavioral 2:1 Multiplexer Testbench</u>

```
module twomux_behavioral_tb();
    logic d0,d1,s,y;
    twomux_behavioral dut(s,d0,d1,y);
    initial begin
        s=0; d0=0; d1=0; #20;
        s=0; d0=0; d1=1; #20;
        s=0; d0=1; d1=0; #20;
        s=0; d0=1; d1=1; #20;
        s=1; d0=0; d1=0; #20;
        s=1; d0=0; d1=1; #20;
        s=1; d0=1; d1=0; #20;
        s=1; d0=1; d1=1; #20;
    end
endmodule
```

<u>Graphical Symbol and Truth Table for 4:1 Mux</u>



S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Behavioral 4:1 Multiplexer Module

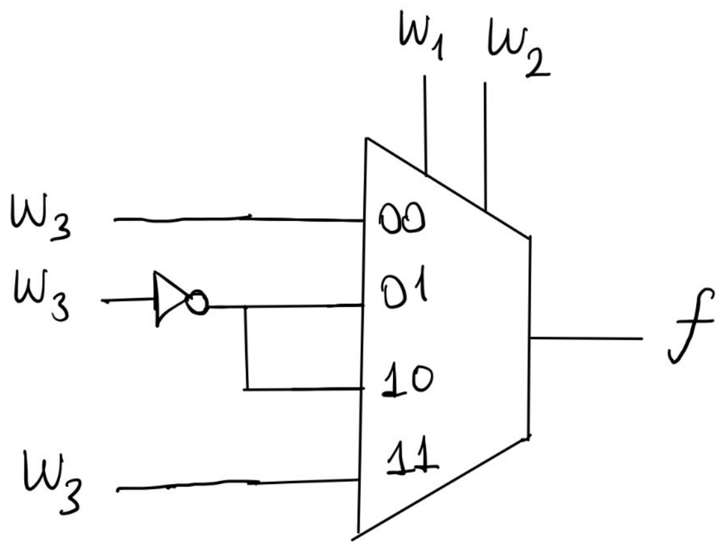
```
module fourmux_behavioral(input logic
s[1:0],i[3:0], output logic y);
    logic w1,w2;
    twomux_behavioral mux1(s[0],i[0],i[1],w1);
    twomux_behavioral mux2(s[0],i[2],i[3],w2);
    twomux_behavioral mux3(s[1],w1,w2,y);
endmodule
```

Behavioral 4:1 Multiplexer Testbench

```
module fourmux_behavioral_tb();
    logic s[1:0],i[3:0],y;
    fourmux_behavioral dut(s,i,y);
    initial begin
```

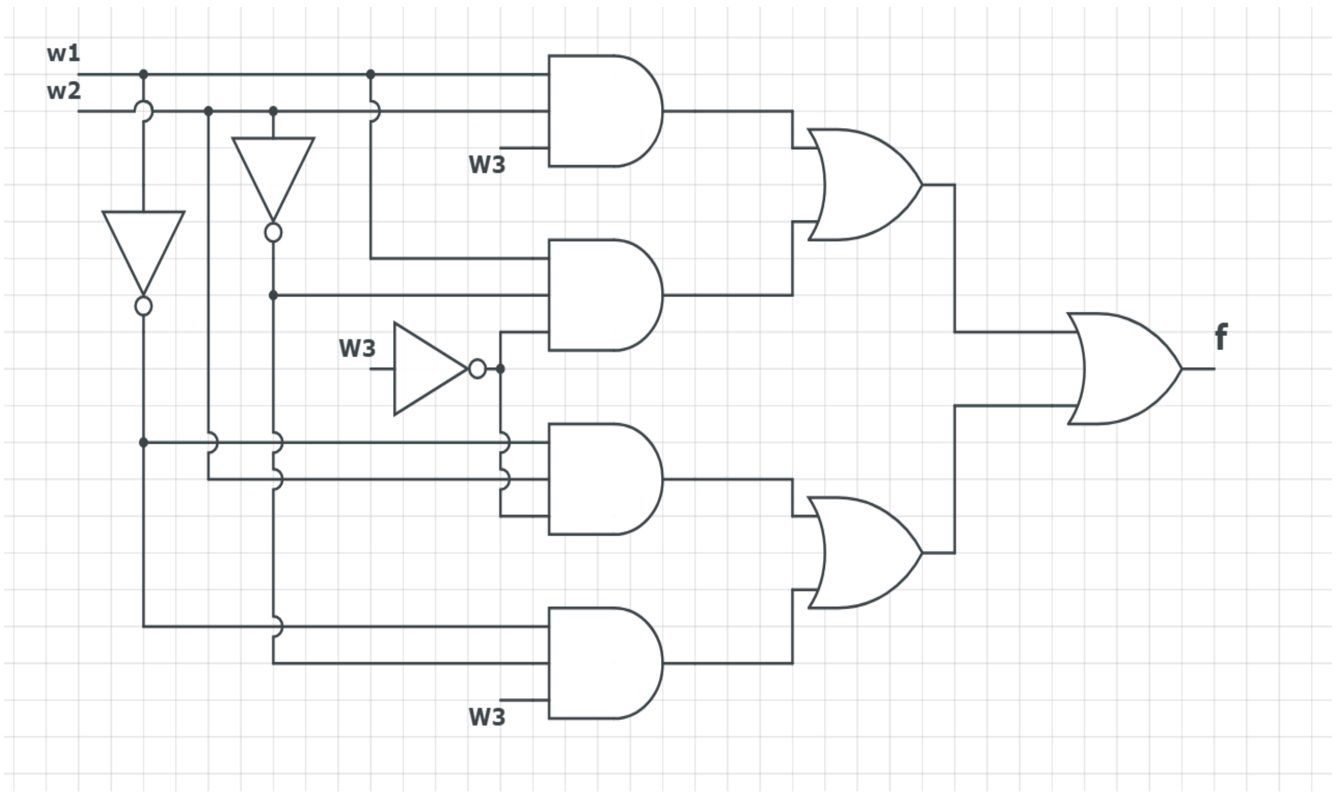
```
        s={0,0}; i={0,0,0,0}; #20;
s={0,0}; i={1,0,0,0}; #20;
        s={0,0}; i={0,1,0,0}; #20;
        s={0,0}; i={0,0,1,0}; #20;
        s={0,0}; i={1,1,1,0}; #20;
        s={0,0}; i={0,0,0,1}; #20;
        s={0,1}; i={0,0,0,0}; #20;
        s={0,1}; i={1,1,0,1}; #20;
        s={0,1}; i={0,0,1,0}; #20;
        s={1,0}; i={0,0,0,0}; #20;
        s={1,0}; i={1,0,1,1}; #20;
        s={1,0}; i={0,1,0,0}; #20;
        s={1,1}; i={0,0,0,0}; #20;
        s={1,1}; i={0,1,1,1}; #20;
        s={1,1}; i={1,0,0,0}; #20;
    end
endmodule
```

Implementation of Function f using 4:1 mux

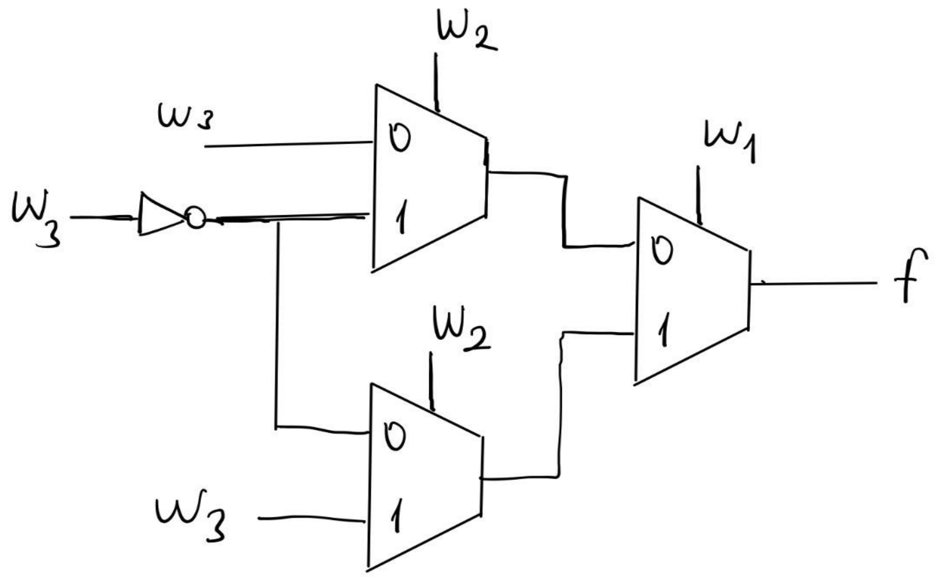


- Description: Basically, acts as a 4 to 1 multiplexer with selects w1 as s1, w2 as s0 and w3 as input0, input3 and w3’ for input1, input2

<u>Logic Diagram of Function f using 4:1 mux</u>

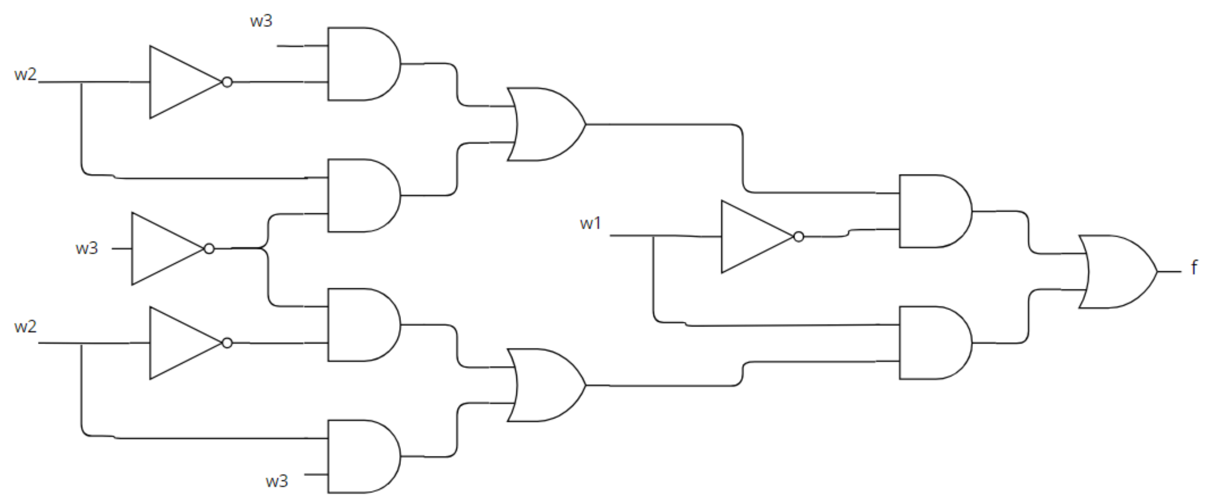


Implementation of Function f using 2:1 mux



- Description: Similar to 4:1 version because 4:1 mux can be implemented by using three 2:1 multiplexers.

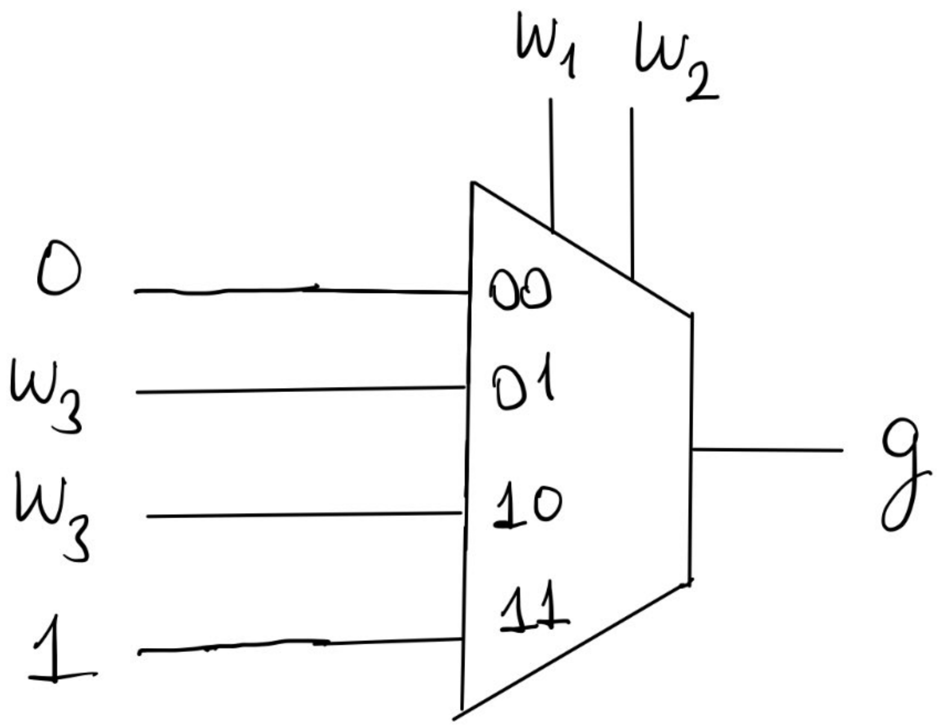
Logic Diagram of Function f using 2:1 mux



Operation of Function f

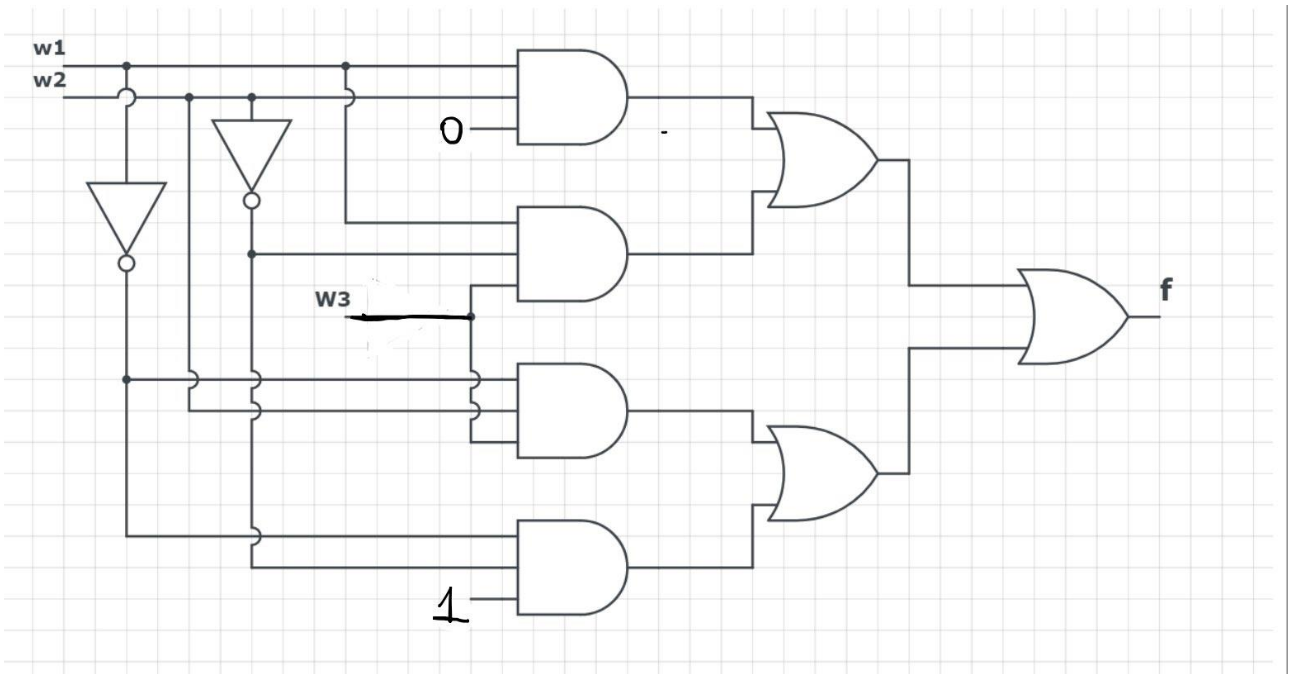
The operation realized is 3 input XOR gate

Implementation of Function g



- Description: input0 and input3 are always constant. However, input1 and input2 depend on w3 while w1 and w2 act as select. Basically, if more than 1 input is true output is true.

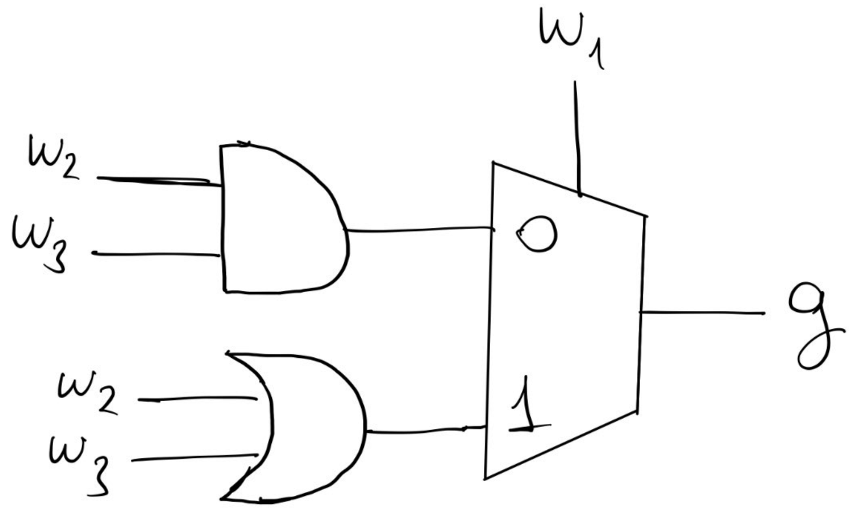
<u>Logic Diagram of Function g using 4:1 Mux</u>



Modified Truth Table for Function g

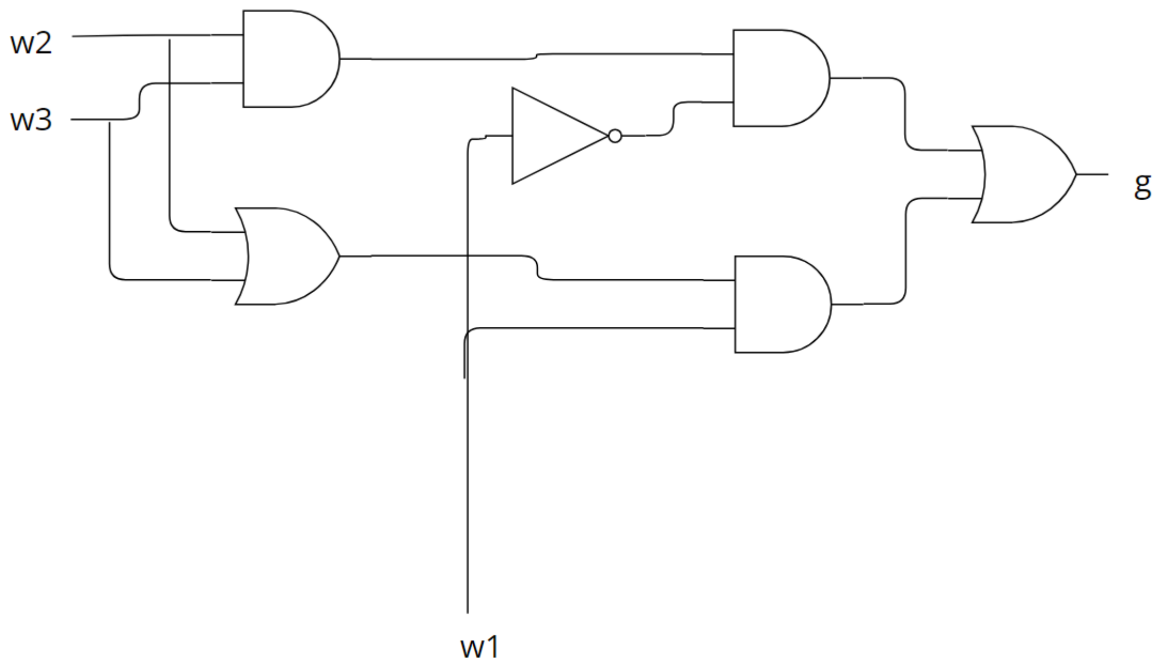
w_1	w_2	w_3	g
0	0	X	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	X	1

Implementation of Function g using 2:1 Mux



- Description: Similar to 4 to 1 Mux version however only w1 acts as a select in this version.

Logic Diagram of Function g using 2:1 Mux



Modified Truth Table of Function g

w_1	w_2	w_3	g
0	0	X	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	X	1

Behavioral Module of Function g using 2:1 Mux

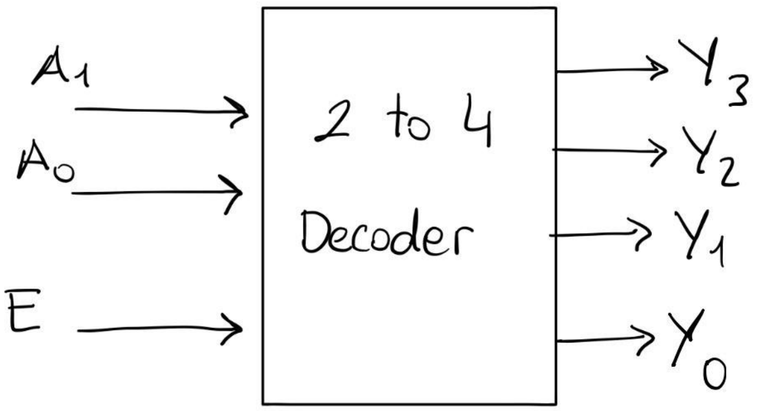
```
module functiong_behavioral(input logic
w1,w2,w3, output logic g);
    logic mid1,mid2;
    assign mid1 = (w2&w3)&(~w1);
    assign mid2 = (w2|w3)&w1;
    assign g = mid1 | mid2;
endmodule
```

Behavioral Module Testbench of Function g

```
module functiong_behavioral_tb();
    logic w1,w2,w3,g;
    functiong_behavioral dut(w1,w2,w3,g);
    initial begin
        w1=0;w2=0;w3=0; #20;
        w1=0;w2=0;w3=1; #20;
        w1=0;w2=1;w3=0; #20;
        w1=0;w2=1;w3=1; #20;
        w1=1;w2=0;w3=0; #20;
        w1=1;w2=0;w3=1; #20;
        w1=1;w2=1;w3=0; #20;
```

```
            w1=1;w2=1;w3=1; #20;
    end
endmodule
```

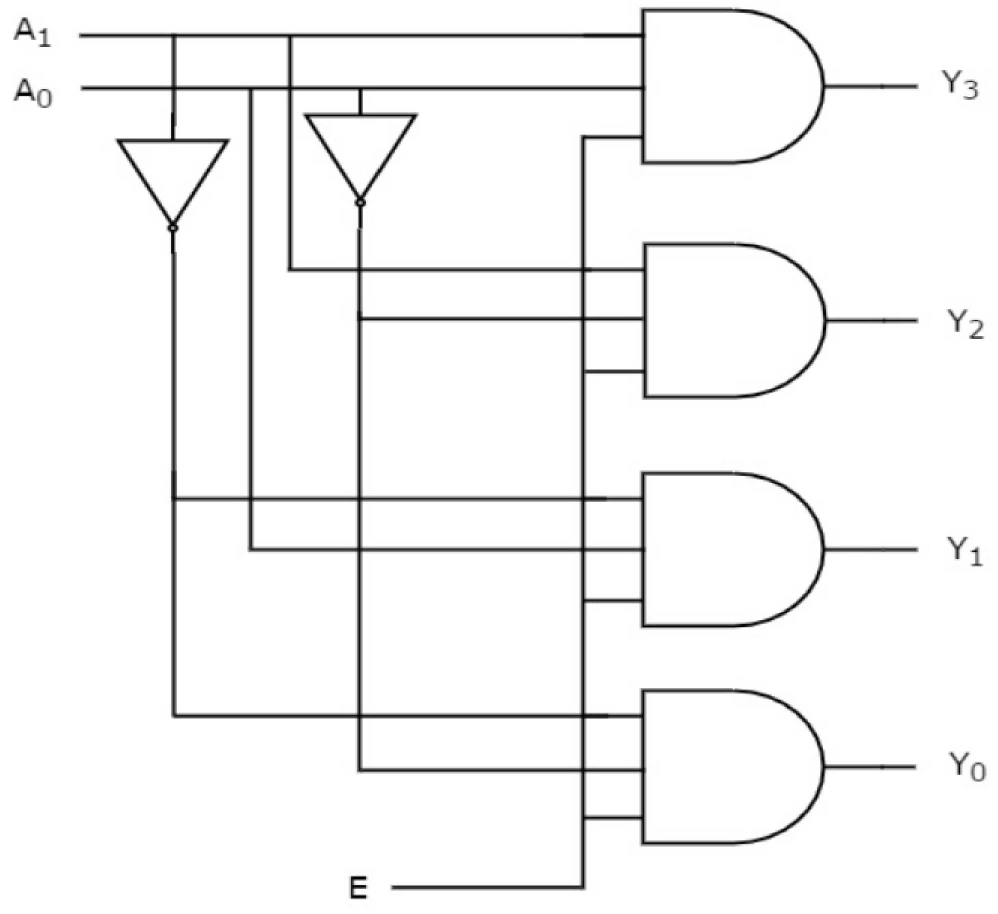
Graphical Symbol of 2 to 4 Decoder



Truth Table of 2 to 4 Decoder

Enable	Inputs		Outputs			
E	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Logic Diagram of 2 to 4 Decoder



Behavioral Module of 2 to 4 Decoder

```
module fourdecoder_behavioral(input logic
a[1:0],e, output logic y[3:0]);
    assign y[3] = a[1] & a[0] & e;
    assign y[2] = a[1] & (~a[0]) & e;
    assign y[1] = (~a[1]) & a[0] & e;
    assign y[0] = (~a[1]) & (~a[0]) & e;
endmodule
```

Behavioral Testbench of 2 to 4 Decoder

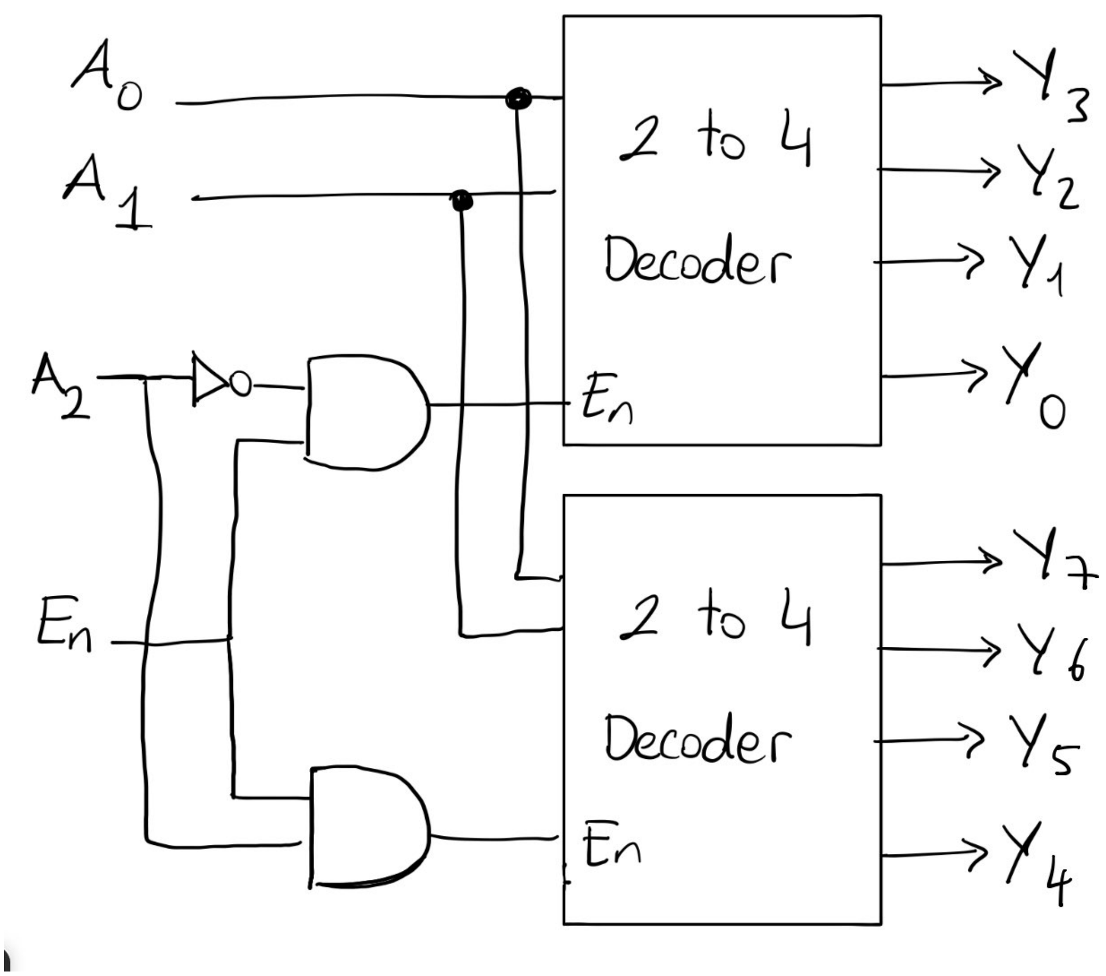
```
module fourdecoder_behavioral_tb();
    logic a[1:0],e,y[3:0];
    fourdecoder_behavioral
dut(a[1:0],e,y[3:0]);
    initial begin
        a={0,0}; e=1; #20;
        a={0,1}; e=1; #20;
        a={1,0}; e=1; #20;
        a={1,1}; e=1; #20;
        a={0,0}; e=0; #20;
        a={0,1}; e=0; #20;
        a={1,0}; e=0; #20;
```

```
            a={1,1}; e=0; #20;

    end

endmodule
```

Logic Diagram of 3 to 8 Decoder



Structural 3 to 8 Decoder Module

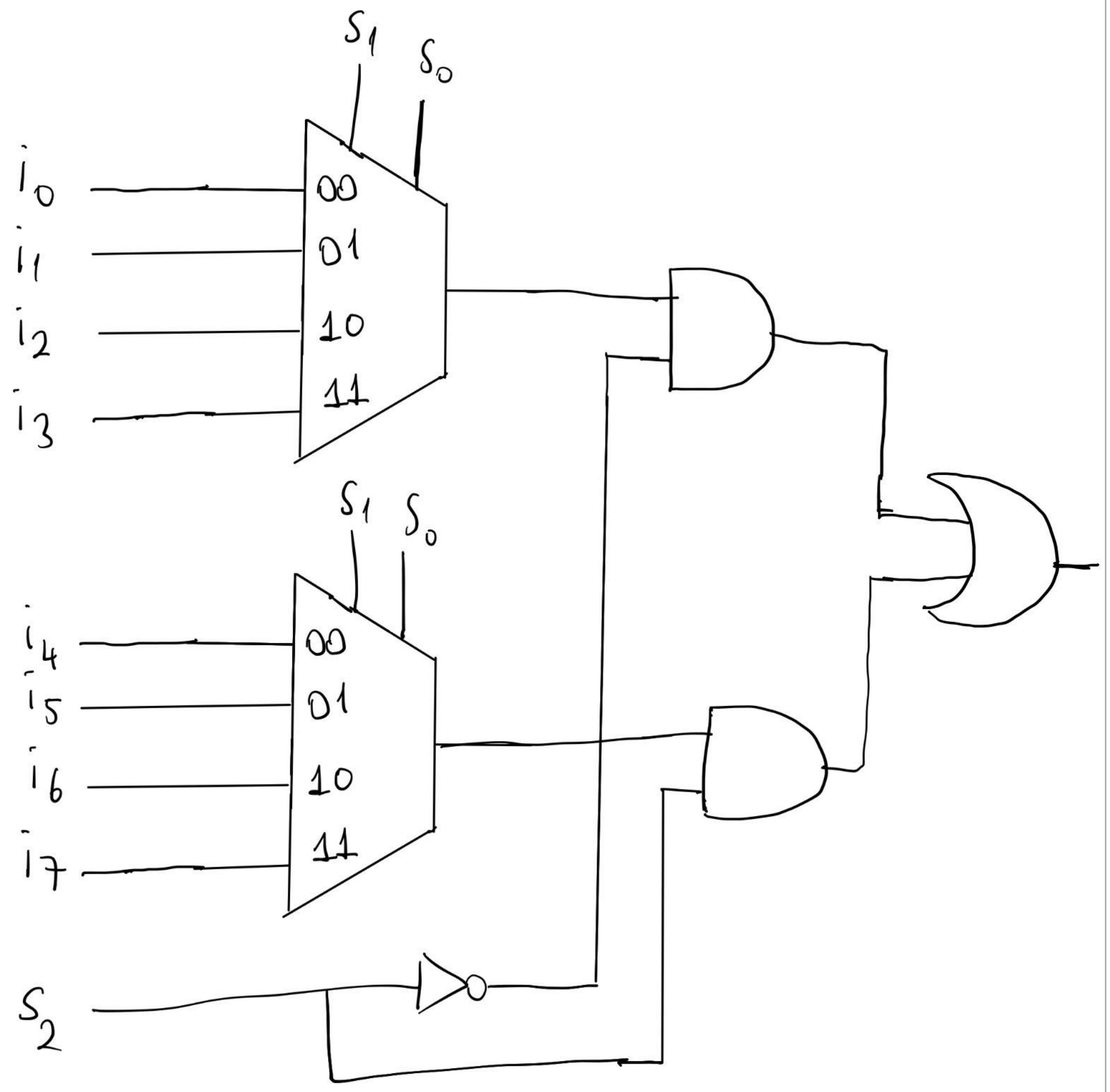
```
module eightdecoder_structural(input logic a[2:0],e,
output logic y[7:0]);
```

```
    logic en1,en2,af2;
    not(af2,a[2]);
    and(en1,af2,e);
    and(en2,a[2],e);
    fourdecoder_behavioral first(a[1:0],en1,y[3:0]);
    fourdecoder_behavioral second(a[1:0],en2,y[7:4]);
endmodule
```

Structural 3 to 8 Decoder Testbench Module

```
module eightdecoder_structural_tb();
    logic a[2:0],e,y[7:0];
    eightdecoder_structural
dut(a[2:0],e,y[7:0]);
    initial begin
        a={0,0,0}; e=1; #20;
        a={0,0,1}; e=1; #20;
        a={0,1,0}; e=1; #20;
        a={0,1,1}; e=1; #20;
        a={1,0,0}; e=1; #20;
        a={1,0,1}; e=1; #20;
        a={1,1,0}; e=1; #20;
        a={1,1,1}; e=1; #20;
        a={0,0,0}; e=0; #20;
        a={0,0,1}; e=0; #20;
        a={0,1,0}; e=0; #20;
        a={0,1,1}; e=0; #20;
        a={1,0,0}; e=0; #20;
        a={1,0,1}; e=0; #20;
        a={1,1,0}; e=0; #20;
        a={1,1,1}; e=0; #20;
    end
endmodule
```

Block Diagram of 8 to 1 Mux



Module of 8 to 1 Mux

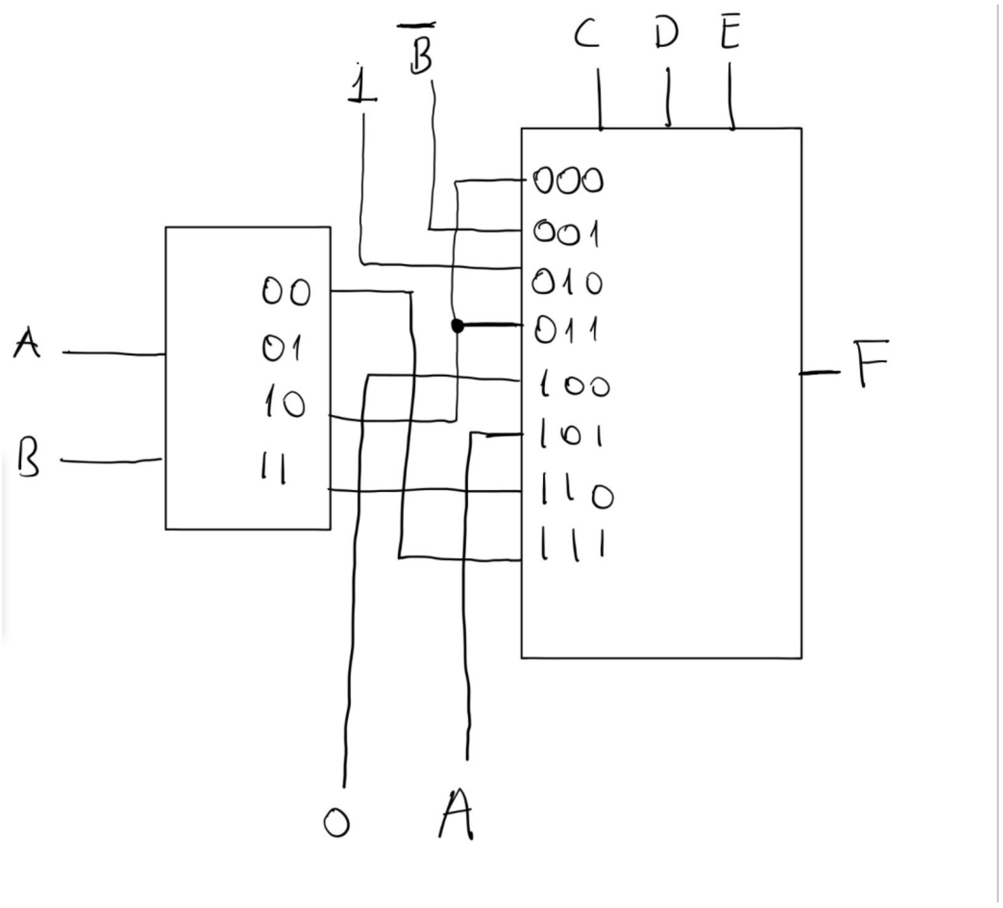
```
module eightmux(input logic i[7:0],s[2:0],
output logic y);

    logic w1,w2;
    fourmux_behavioral m1(s[1:0],i[3:0],w1);
    fourmux_behavioral m2(s[1:0],i[7:4],w2);
    assign y = (~(s[2]) & w1) | (s[2] & w2);
endmodule
```

Testbench Module of 8 to 1 Mux

```
module eightmux_tb();
    logic i[7:0],s[2:0],y;
    eightmux dut(i[7:0],s[2:0],y);
    initial begin
    i = {0,0,0,0,0,0,0,0}; s = {0,0,0}; #20;
    i = {0,0,0,0,0,0,0,1}; s = {0,0,0}; #20;
    i = {0,0,0,0,0,0,0,0}; s = {0,0,1}; #20;
    i = {0,0,0,0,0,0,1,0}; s = {0,0,1}; #20;
    i = {0,0,0,0,0,0,0,0}; s = {0,1,0}; #20;
    i = {0,0,0,0,0,1,0,0}; s = {0,1,0}; #20;
    i = {0,0,0,0,0,0,0,0}; s = {0,1,1}; #20;
    i = {0,0,0,0,1,0,0,0}; s = {0,1,1}; #20;
    i = {0,0,0,0,0,0,0,0}; s = {1,0,0}; #20;
    i = {0,0,0,1,0,0,0,0}; s = {1,0,0}; #20;
    i = {0,0,0,0,0,0,0,0}; s = {1,0,1}; #20;
    i = {0,0,1,0,0,0,0,0}; s = {1,0,1}; #20;
    i = {0,0,0,0,0,0,0,0}; s = {1,1,0}; #20;
    i = {0,1,0,0,0,0,0,0}; s = {1,1,0}; #20;
    i = {0,0,0,0,0,0,0,0}; s = {1,1,1}; #20;
    i = {1,0,0,0,0,0,0,0}; s = {1,1,1}; #20;
    end
endmodule
```

Block Diagram of Function F(A,B,C,D,E)



SystemVerilog Module of Function F(A,B,C,D,E)

```
module functionF(input logic a,b,c,d,e, output
logic f);
    logic
decin[1:0],decout[3:0],muxin[7:0],muxsel[2:0];
    assign decin = {b,a};
    fourdecoder_behavioral
decoder(decin,1,decout);
    assign muxin =
{decout[2],~b,1,decout[2],0,a,decout[3],decout[
0]};
    assign muxsel = {e,d,c};
    eightmux mux(muxin,muxsel,f);
endmodule
```