

# Lecture 10. Overview of basic combinatorial circuits.

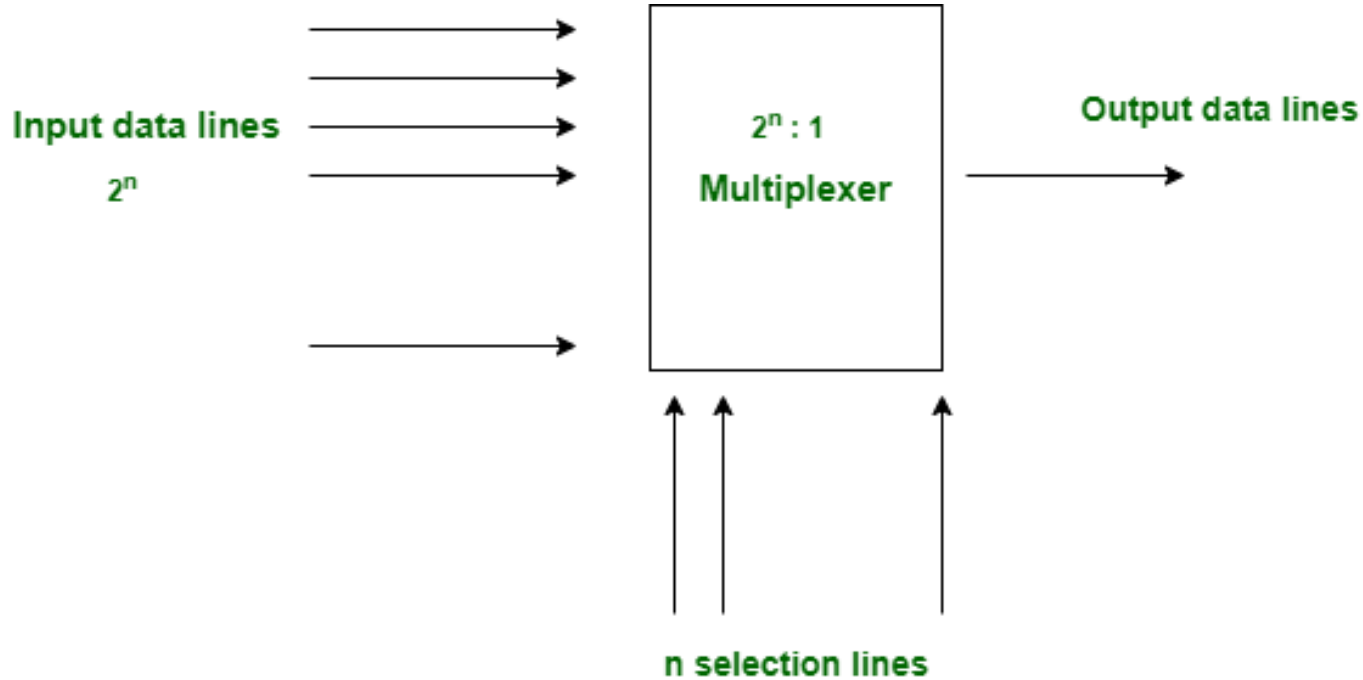
Dmitry Zaitsev

<http://daze.ho.ua>

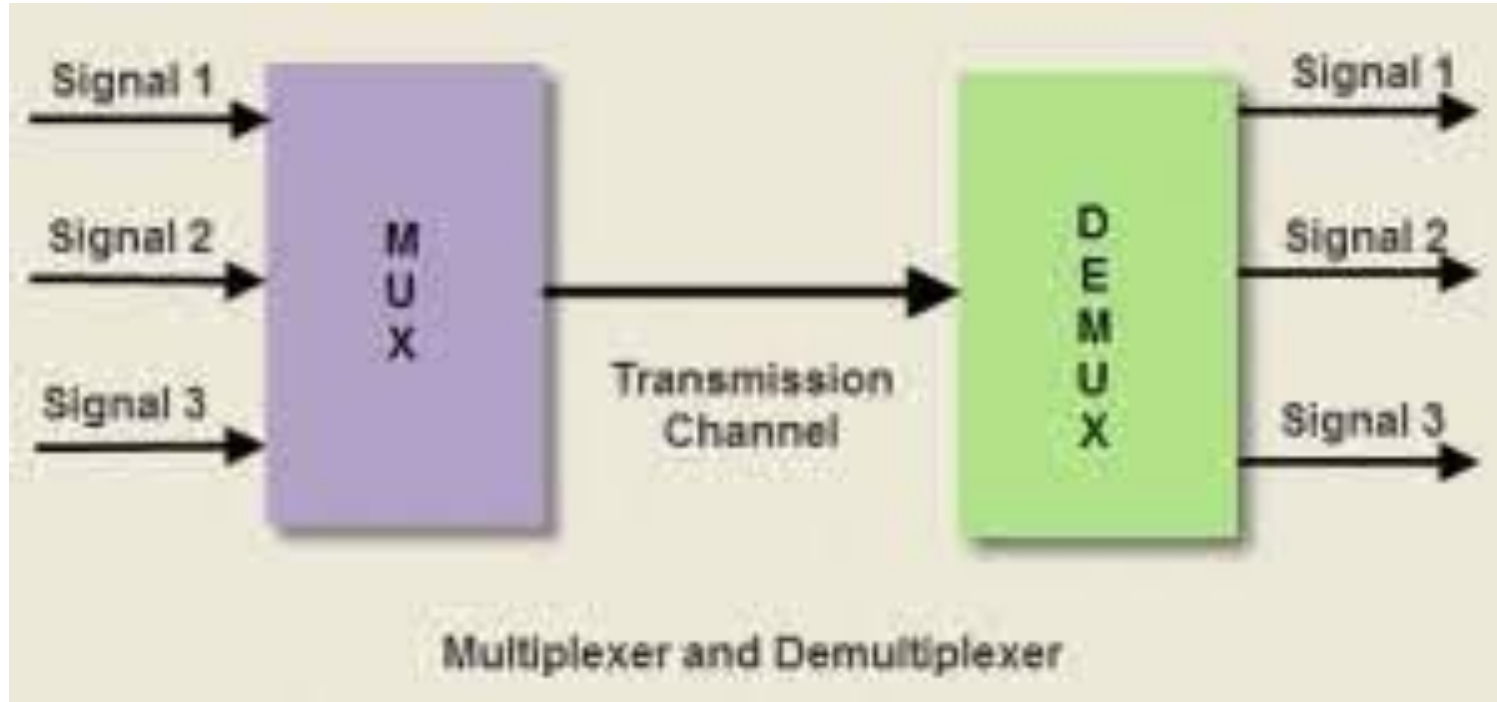
# List of basic combinatorial circuits

- Adder:  $x+y$
- Subtractor:  $x-y$  (addition with 2's complement)
- Comparator:  $x < y$  (sign of subtraction)
- Multiplexer: selects an input from several inputs
- Demultiplexer: switch single input between several outputs
- Encoder: unary code to binary code ( $2^n$  to  $n$ )
- Decoder: binary code to unary code ( $n$  to  $2^n$ )

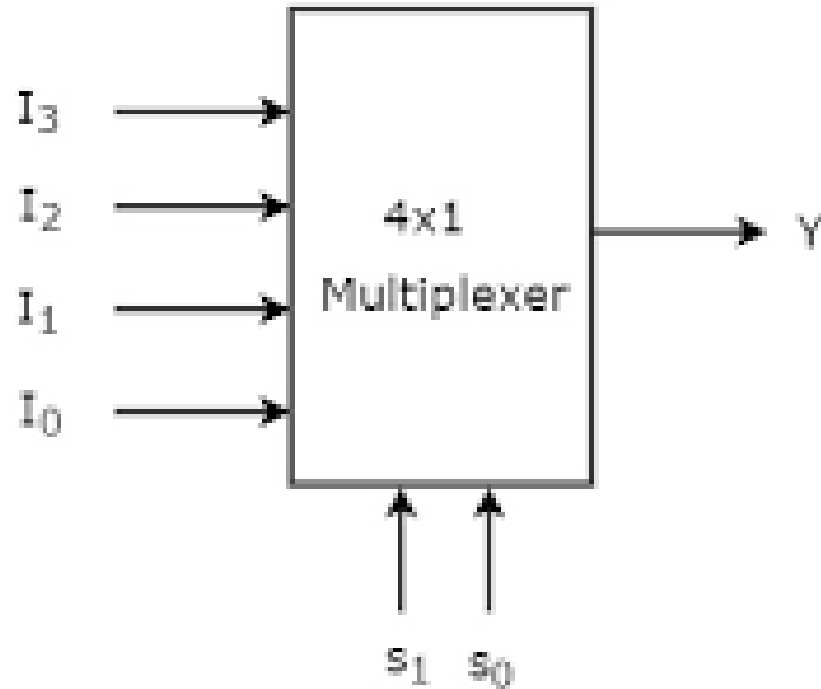
# Multiplexer



# Multiplexer and demultiplexer



# 4x1 Multiplexer

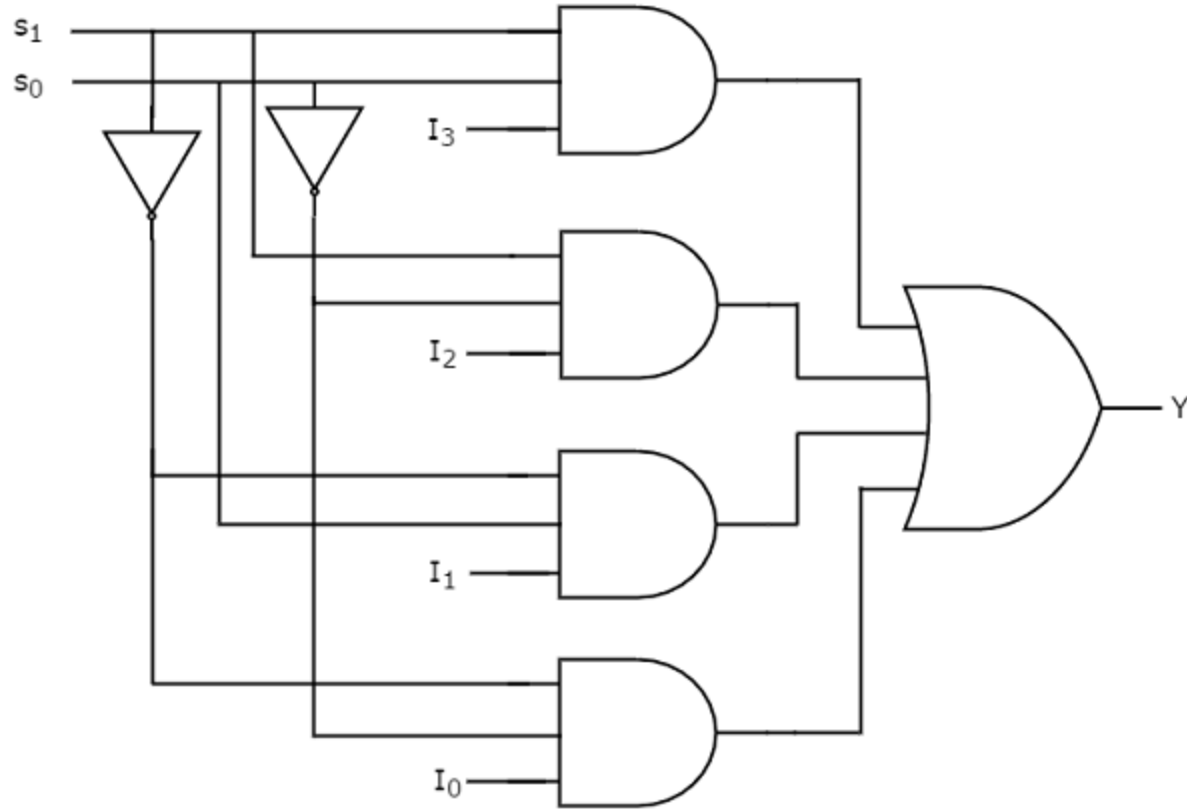


# 4x1 multiplexer design

s1	s0	y
0	0	i0
0	1	i1
1	0	i2
1	1	i3

$$y = i_0 \bar{s}_1 \bar{s}_0 \vee i_1 \bar{s}_1 s_0 \vee i_2 s_1 \bar{s}_0 \vee i_3 s_1 s_0$$

# 4x1 Multiplexer layout



# 4x1 Multiplexor Verilog module

```
module MUX4x1(  
    input [3:0] i,  
    input [1:0] s,  
    output y  
);  
    assign  
    y = i[0]&~s[1]&~s[0] |  
        i[1]&~s[1]&s[0] |  
        i[2]&s[1]&~s[0] |  
        i[3]&s[1]&s[0];  
  
endmodule
```



# 4x1 Multiplexor design in Verilog

DigitalJS Online

Not secure | digitaljs.tilk.eu

12055

Setup I/O

unnamed.sv

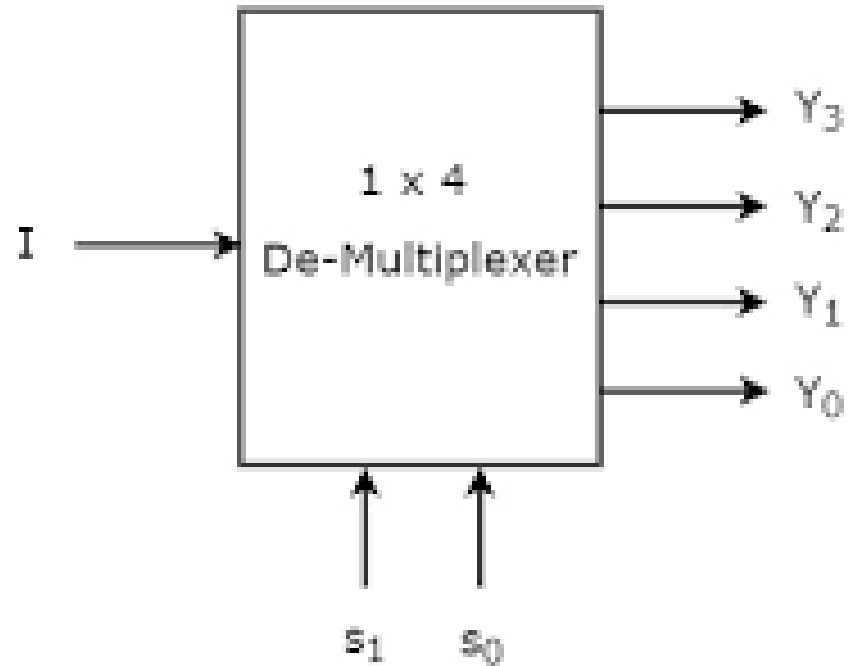
```
1 module MUX4x1(  
2     input [3:0] i,  
3     input [1:0] s,  
4     output y  
5 );  
6  
7 assign  
8 y = i[0]&~s[1]&~s[0] |  
9     i[1]&~s[1]&s[0] |  
10    i[2]&s[1]&~s[0] |  
11    i[3]&s[1]&s[0];  
12  
13 endmodule  
14
```

Synthesize and simulate!

00000000

Construction on Aven... 9:00 AM 5/10/2023

# 1x4 Demultiplexer



# 1x4 Demultiplexer design

Input			Output			
I	$S_1$	$S_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
I	0	0	I	0	0	0
I	0	1	0	I	0	0
I	1	0	0	0	I	0
I	1	1	0	0	0	I

# 1x4 Demultiplexer truth table

Input			Output			
I	S <sub>1</sub>	S <sub>0</sub>	Y <sub>0</sub>	Y <sub>1</sub>	Y <sub>2</sub>	Y <sub>3</sub>
0	*	*	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

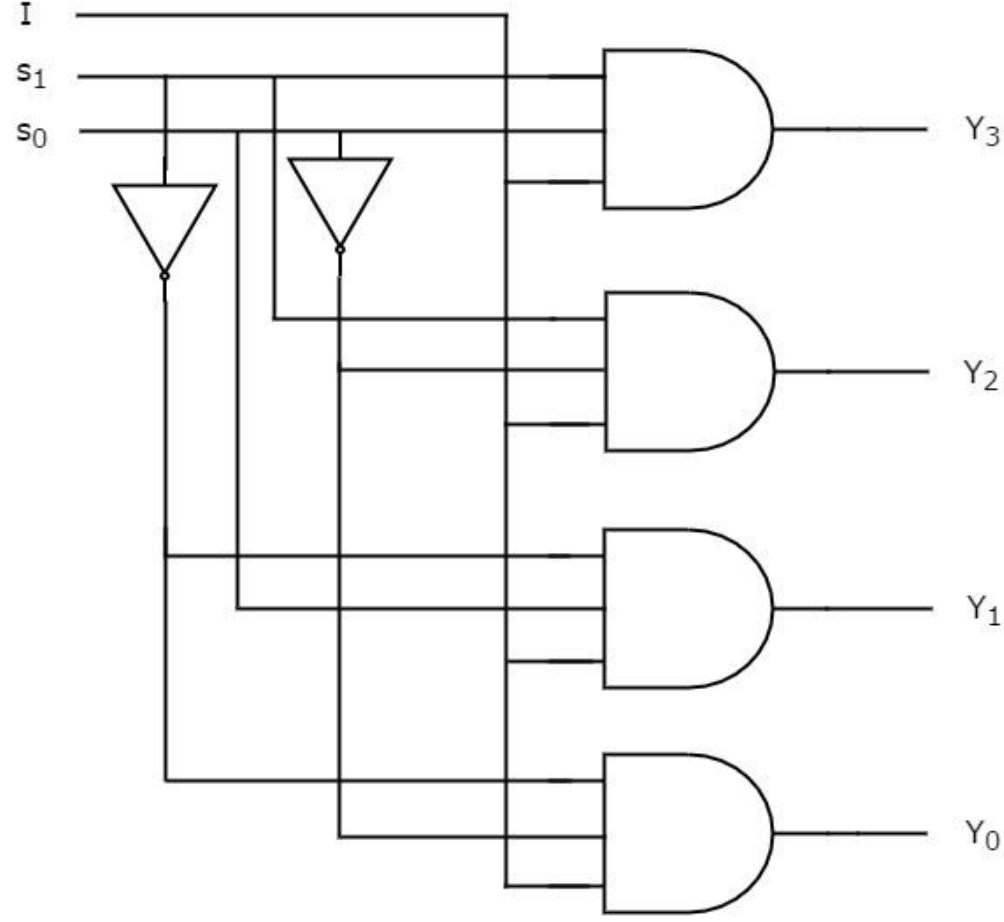
$$Y_0 = I\bar{S}_1\bar{S}_0$$

$$Y_1 = I\bar{S}_1S_0$$

$$Y_2 = IS_1\bar{S}_0$$

$$Y_3 = IS_1S_0$$

# 1x4 Demultiplexer layout



# 1x4 Demultiplexer Verilog module

```
module DEMUX1x4(  
    input i,  
    input [1:0] s,  
    output [3:0] y  
);  
  
    assign  
        y[0] = i&~s[1]&~s[0],  
        y[1] = i&~s[1]&s[0],  
        y[2] = i&s[1]&~s[0],  
        y[3] = i&s[1]&s[0];  
  
endmodule
```

# 1x4 Demultiplexer design in Verilog

DigitalJS Online

Not secure | digitaljs.tilk.eu

26253

Setup I/O unnamed.v

```
1 module DEMUX1x4(  
2     input i,  
3     input [1:0] s,  
4     output [3:0] y  
5 );  
6  
7 assign  
8     y[0] = i&~s[1]&~s[0],  
9     y[1] = i&~s[1]&s[0],  
10    y[2] = i&s[1]&~s[0],  
11    y[3] = i&s[1]&s[0];  
12  
13 endmodule
```

Synthesize and simulate!

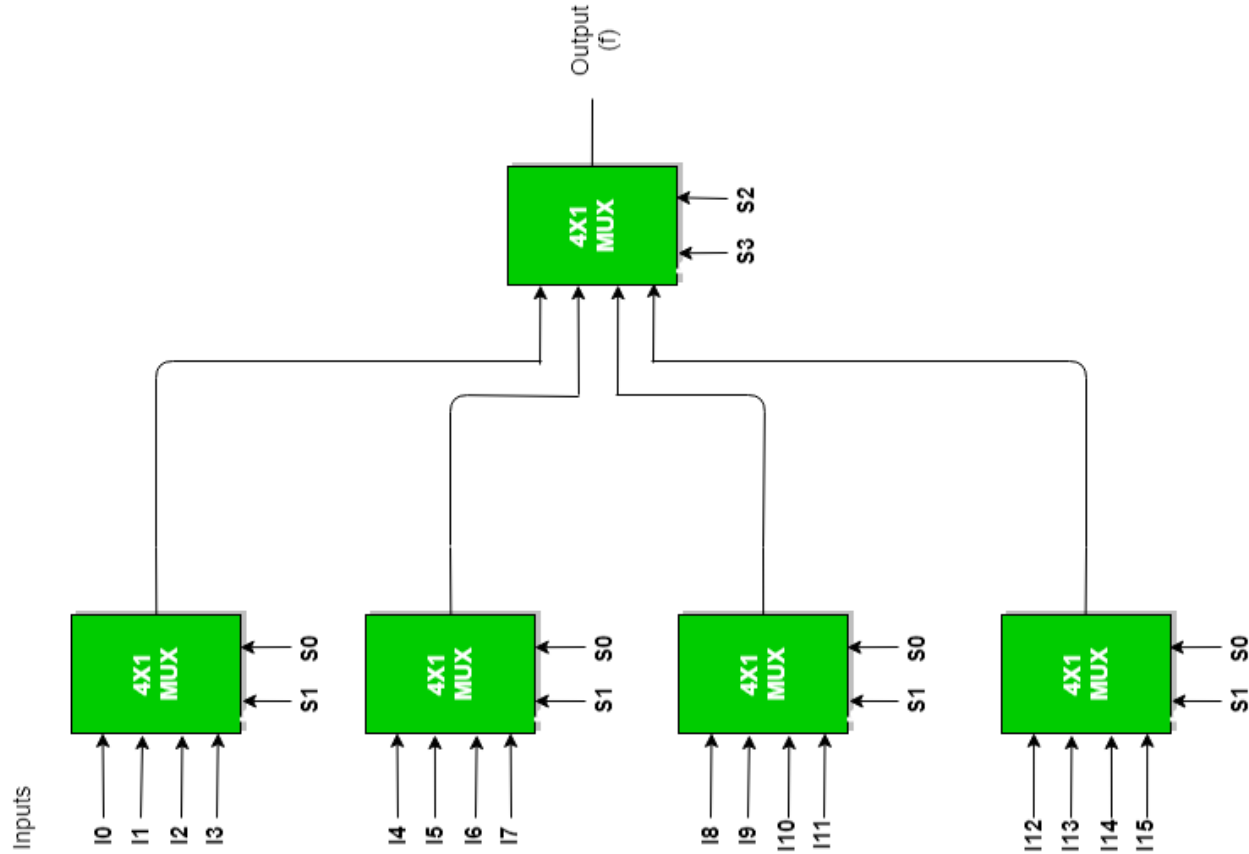
The circuit diagram illustrates the implementation of the 1x4 Demultiplexer. The input 'i' is connected to the left input of four 2-input AND gates. The select inputs 's[1]' and 's[0]' are connected to the right inputs of the AND gates through inverters and buffers. The outputs of the AND gates are connected to a 4-bit bus 'y'. The bus 'y' is connected to a 4-bit display showing '0100'.

# Digital circuits design using multiplexers

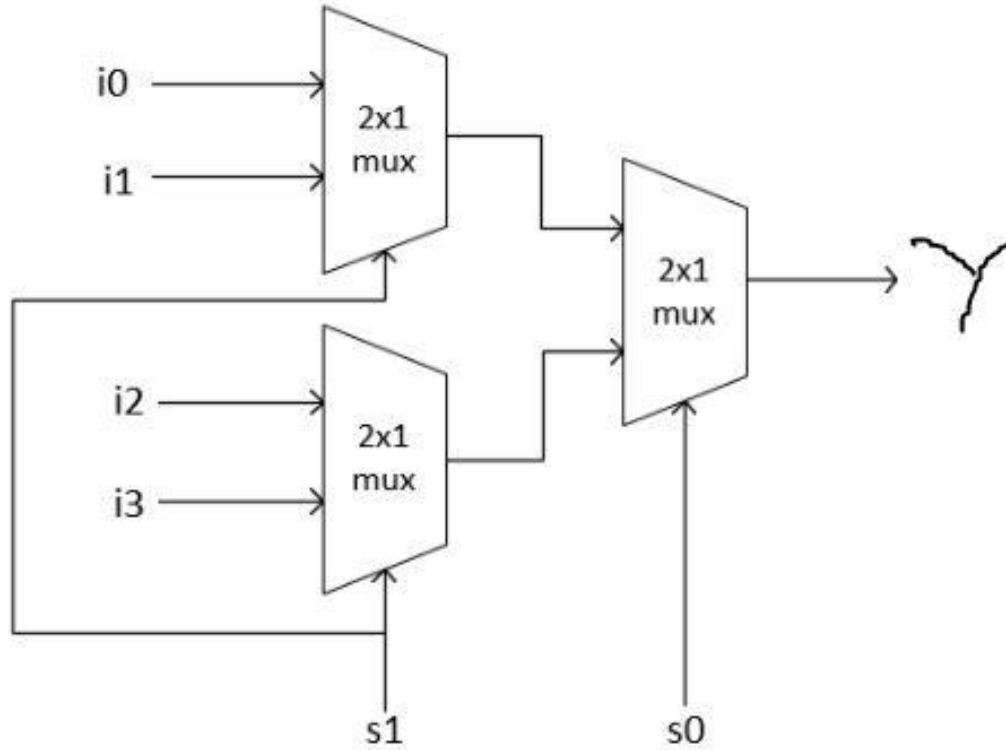
- Basic gates in MUX 2x1:
  - NOT:  $01(x)$
  - AND:  $y1(x)$
  - OR:  $1y(x)$
- MUX 4x1 as composition of 2 MUX 2x1 etc
- Adders and other arithmetics



# MUX 16x1 as 5 MUX 4x1



# MUX4x1 design of 3 MUX2x1



# Verilog design of MUX2x1

DigitalJS Online x 16-to-1 multiplexer (16X1 MUX) x +

Not secure | digitaljs.tilk.eu

2377

Setup I/O unnamed.sv

```
1 module MUX_2X1(input I0, I1, S, output Y);
2   wire sbar, w1, w2;
3   not G1 (sbar, S);
4   and G2 (w1, I0, sbar);
5   and G3 (w2, I1, S);
6   or G4 (Y, w1, w2);
7 endmodule
8
9
```

Synthesize and simulate!

00000000

Construction on Aven... 9:13 AM 5/10/2023

# Verilog MUX4x1 design of 3 MUX2x1

DigitalJS Online x 16-to-1 multiplexer (16X1 MUX) x +

Not secure | digitaljs.tilk.eu

2963

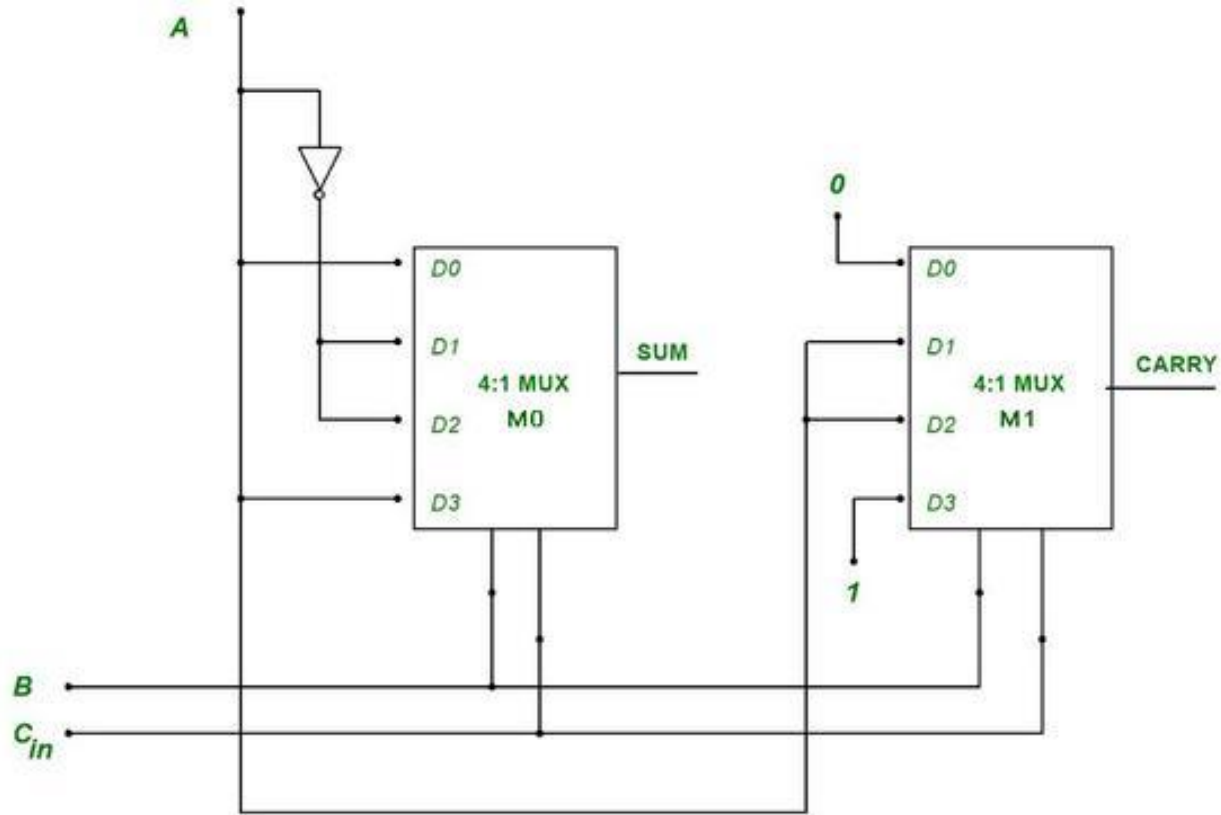
Setup I/O unnamed.sv

```
1 module MUX_2X1(input I0, I1, S, output Y);
2   wire sbar, w1, w2;
3   not G1 (sbar, S);
4   and G2 (w1, I0, sbar);
5   and G3 (w2, I1, S);
6   or G4 (Y, w1, w2);
7 endmodule
8
9 module MUX_4X1(input i0, i1, i2, i3, s0, s1, output Y);
10  wire Y1, Y2;
11  MUX_2X1 M1 (i0, i1, s0, Y1);
12  MUX_2X1 M2 (i2, i3, s0, Y2);
13  MUX_2X1 M3 (Y1, Y2, s1, Y);
14 endmodule
```

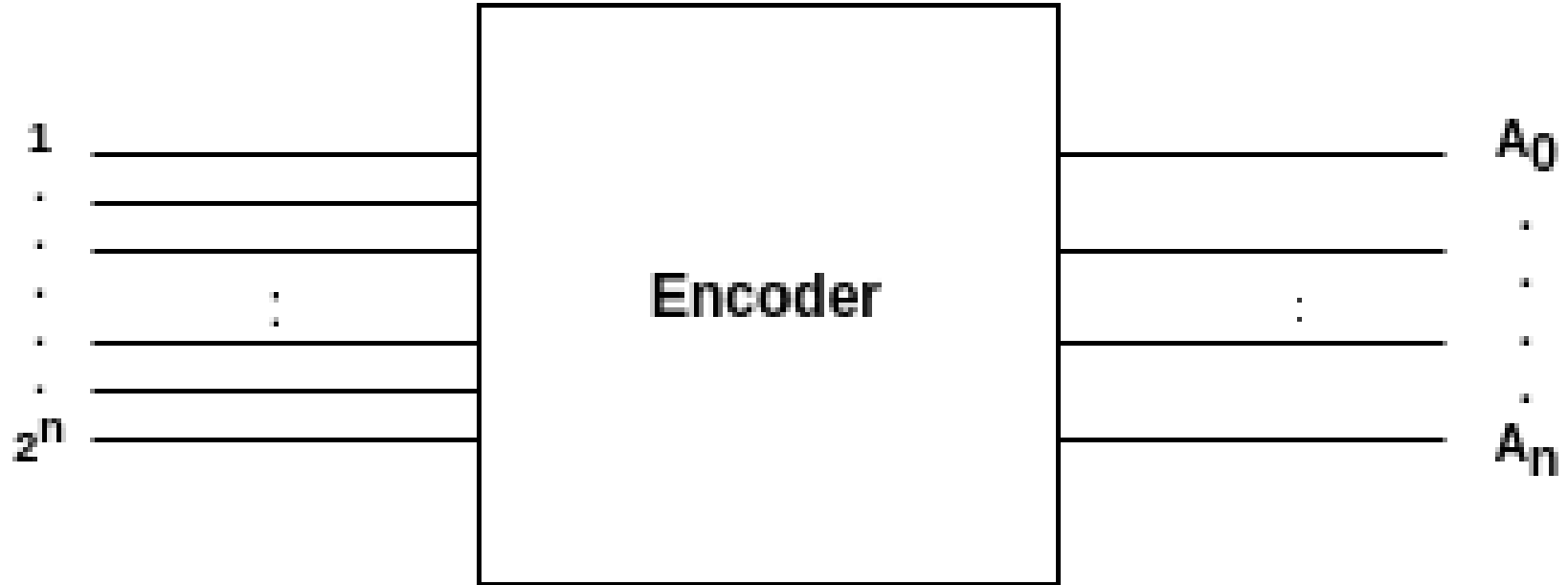
Synthesize and simulate!

65°F Mostly cloudy 9:17 AM 5/10/2023

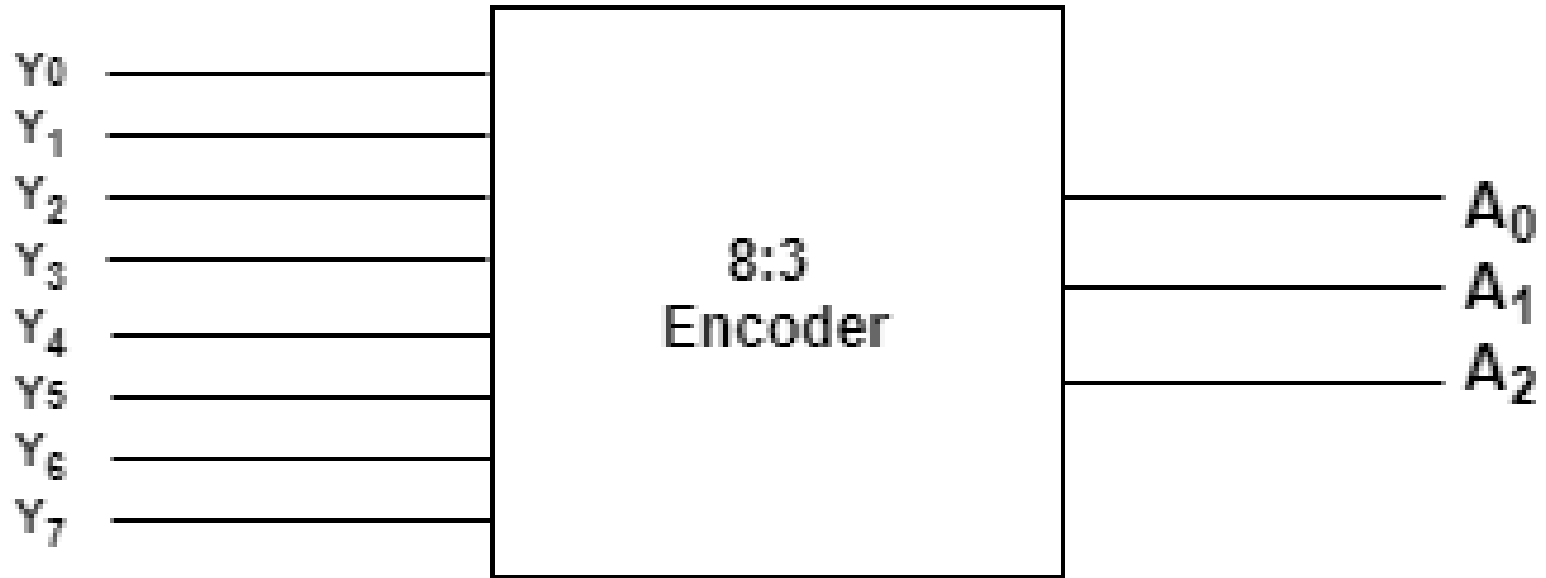
# 1 bit full adder on MUX4x1



# Encoder



# 8x3 Encoder

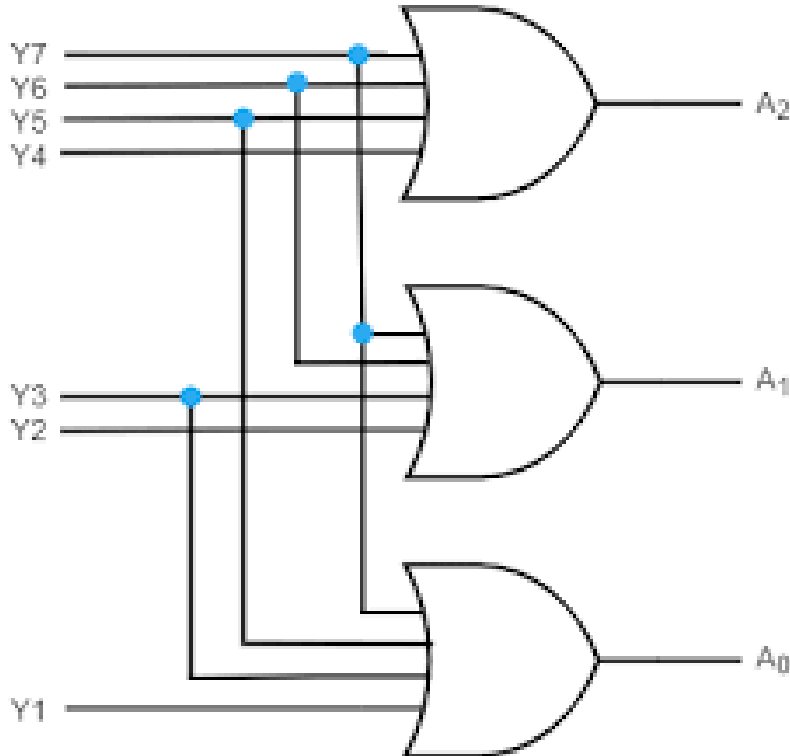


## 8x3 Encoder truth table

[illegible]



# 8x3 Encoder layout



$$A2 = Y4 \vee Y5 \vee Y6 \vee Y7$$

$$A1 = Y2 \vee Y3 \vee Y6 \vee Y7$$

$$A0 = Y1 \vee Y3 \vee Y5 \vee Y7$$

# Ambiguity of encoder

- only one input can be active at any given time;  
Priority Encoder – assigned priority of input lines
- all inputs are 0 – output is 000 the same as for Y0; extra output – the valid bit: 0 when all inputs 0 and 1 otherwise

# 4x2 Priority Encoder with Valid Bit

X3	X2	X1	X0	A1	A0	V
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

# Decoder



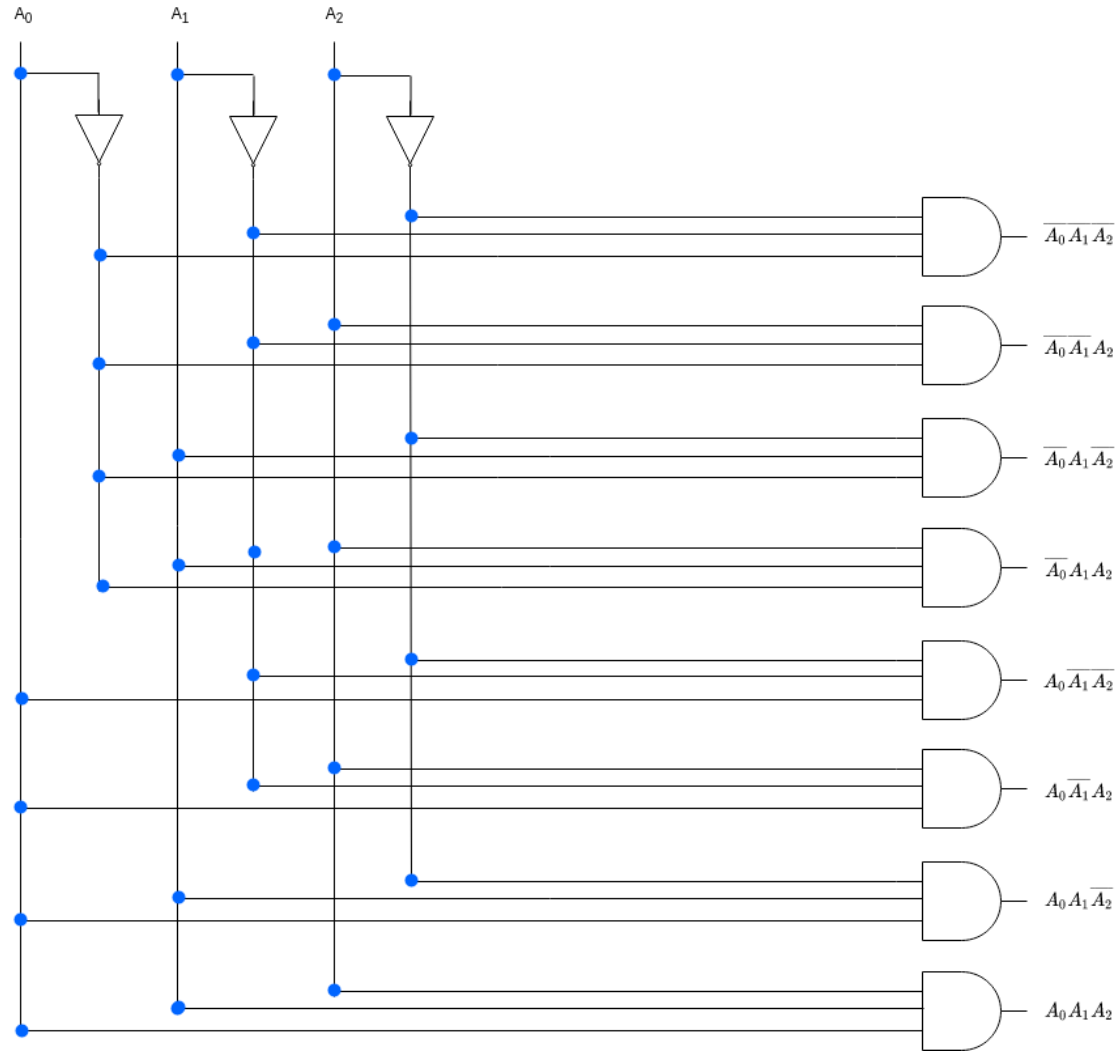
# 3x8 Decoder



# 3x8 Decoder truth table

Decimal Digit	Binary			Output							
	A0	A1	A2	0	1	2	3	4	5	6	7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

# 3x8 Decoder layout



# Full adder using 3x8 decoder

