ELCT201: DIGITAL LOGIC DESIGN

Prof. Dr. Eng. Tallal El-Shabrawy,
tallal.el-shabrawy@guc.edu.eg

Dr. Eng. Wassim Alexan,
wassim.joseph@guc.edu.eg

Lecture 6

محرم 1441 هـ
Spring 2020

German University in Cairo

Following the slides of Dr. Ahmed H. Madian

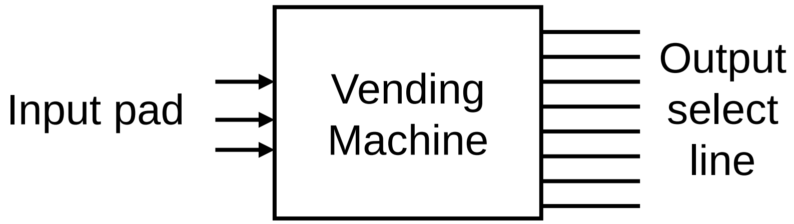
COURSE OUTLINE

LECTURE OUTLINE

- Combinational Logic Circuits
  - Decoders
  - Encoders
  - Multiplexers
  - Tri-state Buffers

DECODERS

- A Decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines
- If the n −bit coded information has unused combinations, the decoder may have fewer than 2^n outputs
- Consider a vending machine that takes 3 bits as input and releases a single product, out of the available 8 product sorts



DECODERS

- It is required to design a combinational circuit with two inputs (a, b) and four outputs (D_0, D_1, D_2, D_3), such that:

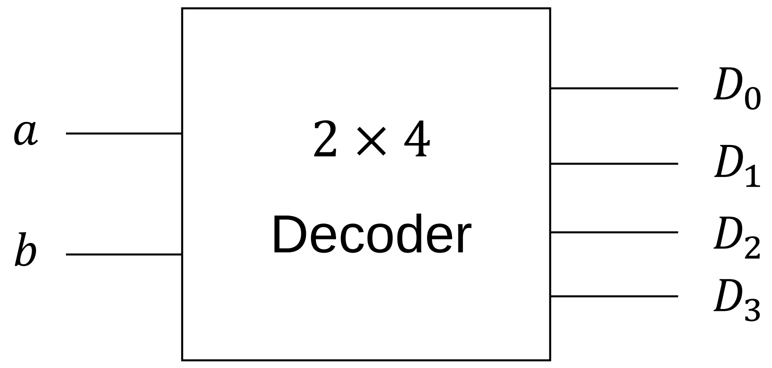
  - $D_0 = 1$ when $a = 0$ and $b = 0$
  - $D_1 = 1$ when $a = 0$ and $b = 1$
  - $D_2 = 1$ when $a = 1$ and $b = 0$
  - $D_3 = 1$ when $a = 1$ and $b = 1$

GUC German University in Cairo

DECODERS

Solution

1. From the specifications of the circuit, determine the required number of inputs and outputs and assign a letter (symbol) to each



DECODERS

2. Derive the truth table that defines the required relationship between the inputs and outputs

Inputs		Outputs			
a	b	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

DECODERS

3. Obtain the simplified Boolean functions for each output as a function of the input variables

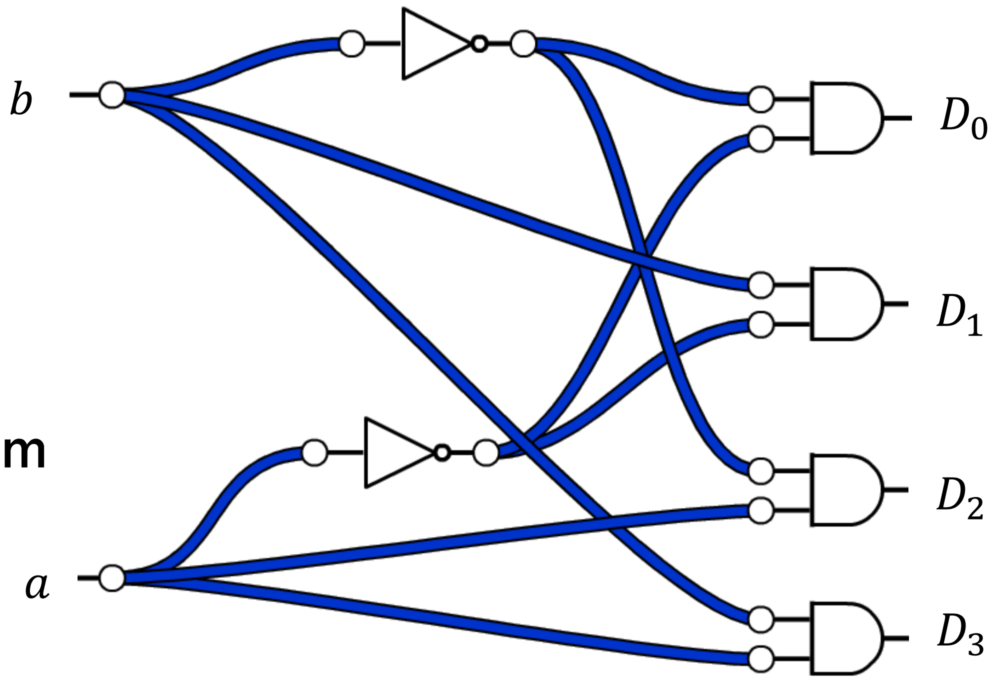
$D_0 = a'b'$

$D_1 = a'b$

$D_2 = ab'$

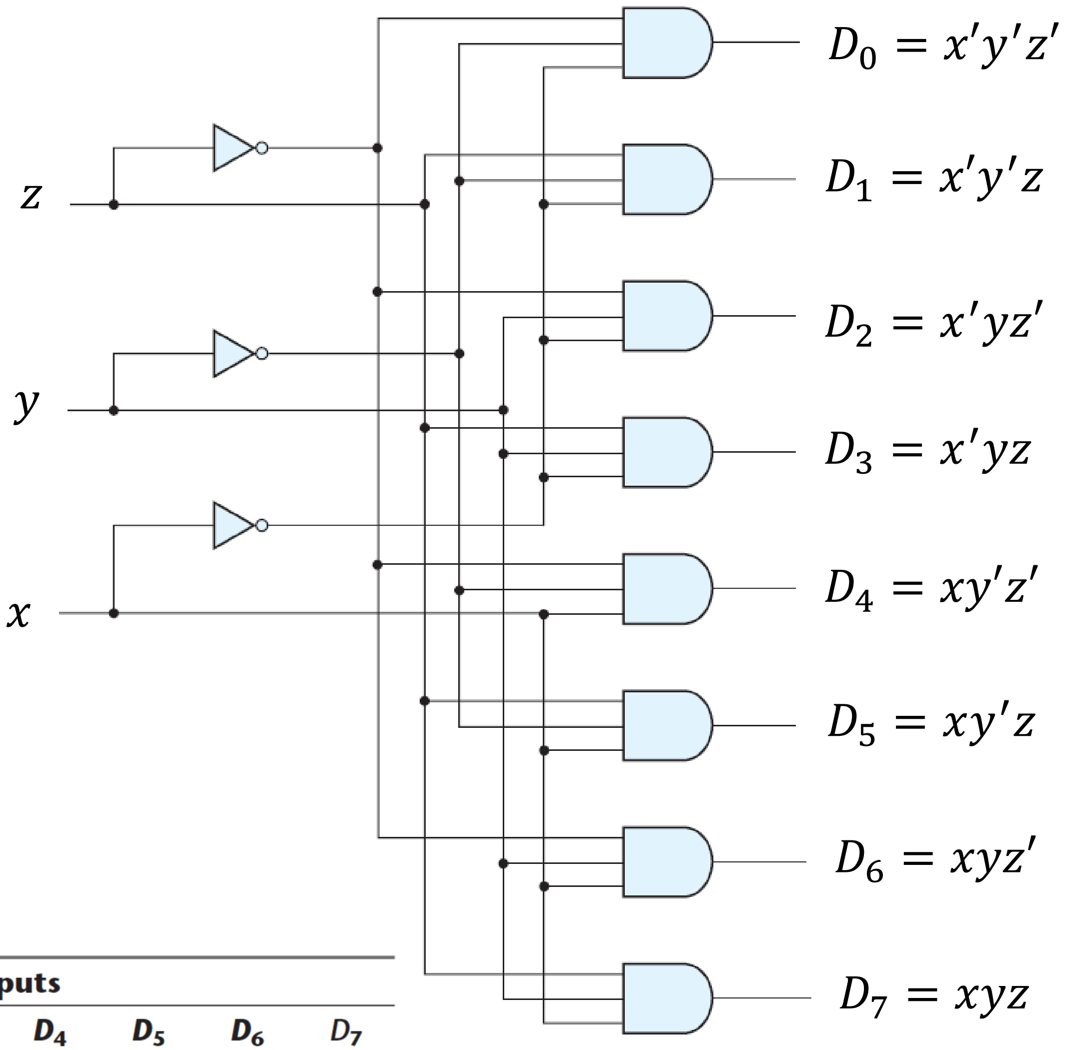
$D_3 = ab$

4. Sketch the logic diagram



3×8 DECODER

- A 3 × 8 line decoder decodes 3 input bits into one of 8 possible outputs
- Each output represents one of the minterms of the 3 input variables



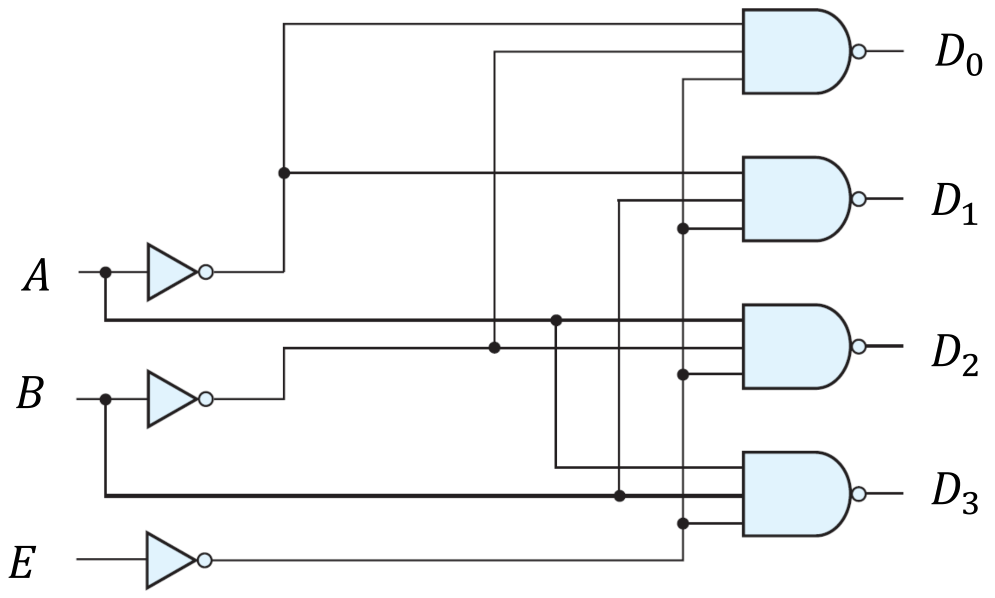
Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

2×4 DECODER

- A decoder could include an *Enable* input to control the circuit operation

- A decoder could be implemented with NAND gates and thus produces the minterms in their complemented form

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



IMPLEMENTING FUNCTIONS USING DECODERS

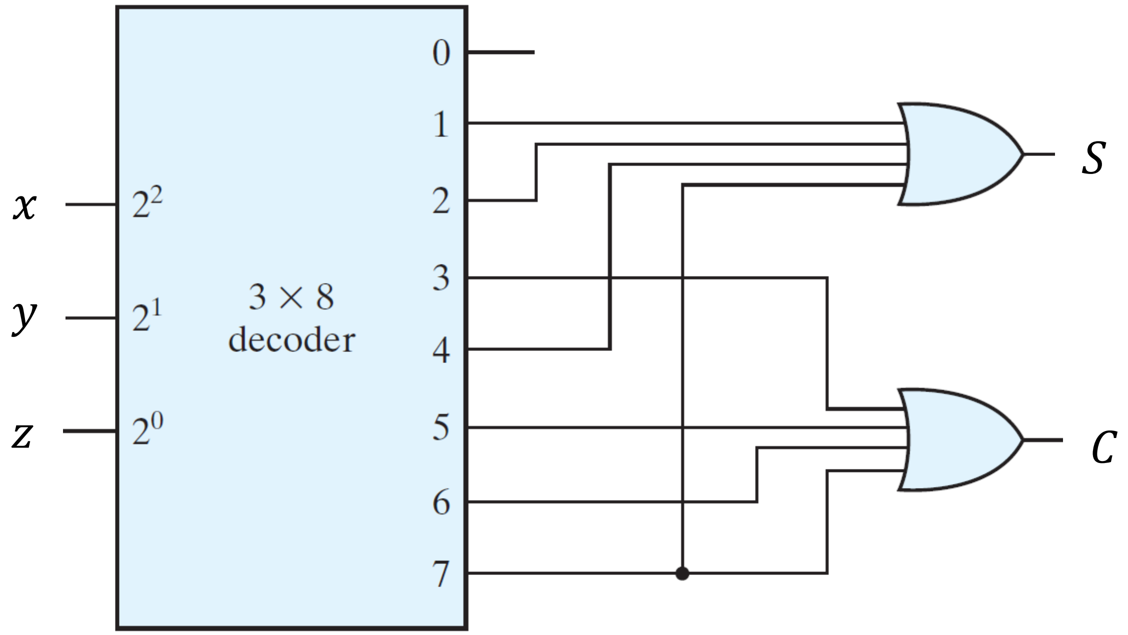
- Any combinational circuit can be constructed using decoders and OR gates (the decoder generates the minterms and the OR gate performs the summation)
- **Example:** Implement a full adder circuit with a decoder and two OR gates
- Full adder equations:

$$S(x, y, z) = \Sigma m(1,2,4,7) \text{ and}$$
$$C(x, y, z) = \Sigma m(3,5,6,7)$$

- Since there are 3 inputs, we need a 3×8 decoder

IMPLEMENTING FUNCTIONS USING DECODERS

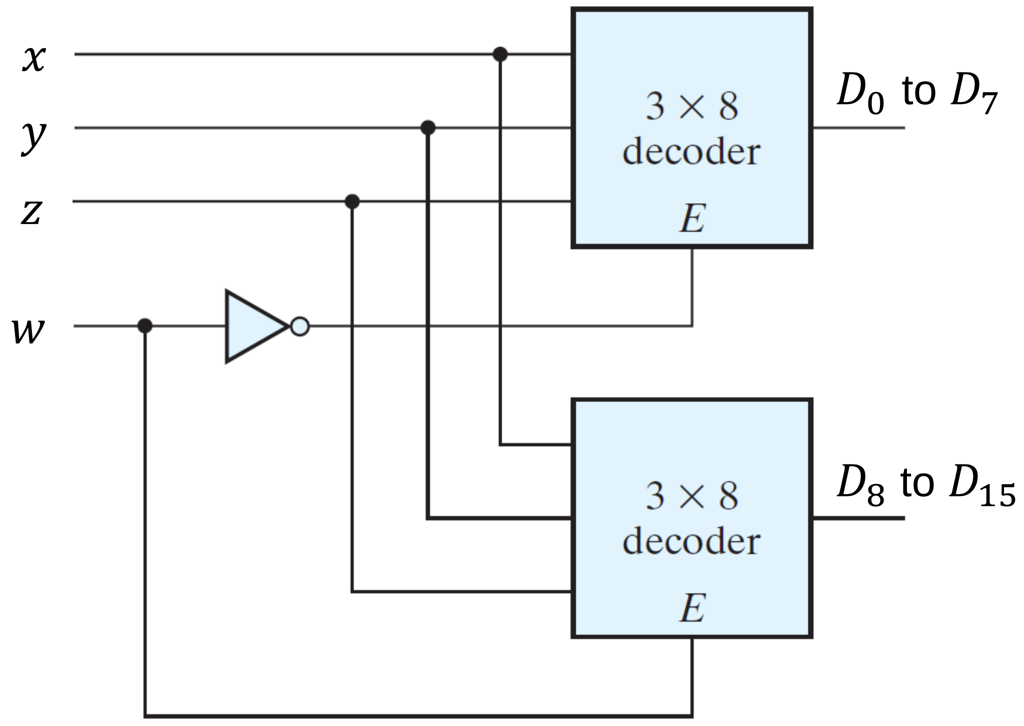
- $S(x, y, z) = \Sigma m(1,2,4,7)$ and
- $C(x, y, z) = \Sigma m(3,5,6,7)$



Inputs			Outputs	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

DECODER EXPANSIONS

- Larger decoders can be constructed using a number of smaller ones
- For example, a 3×8 decoder can be built using a couple of 2×4 decoders and a 4×16 decoder can be built using a couple of 3×8 decoders

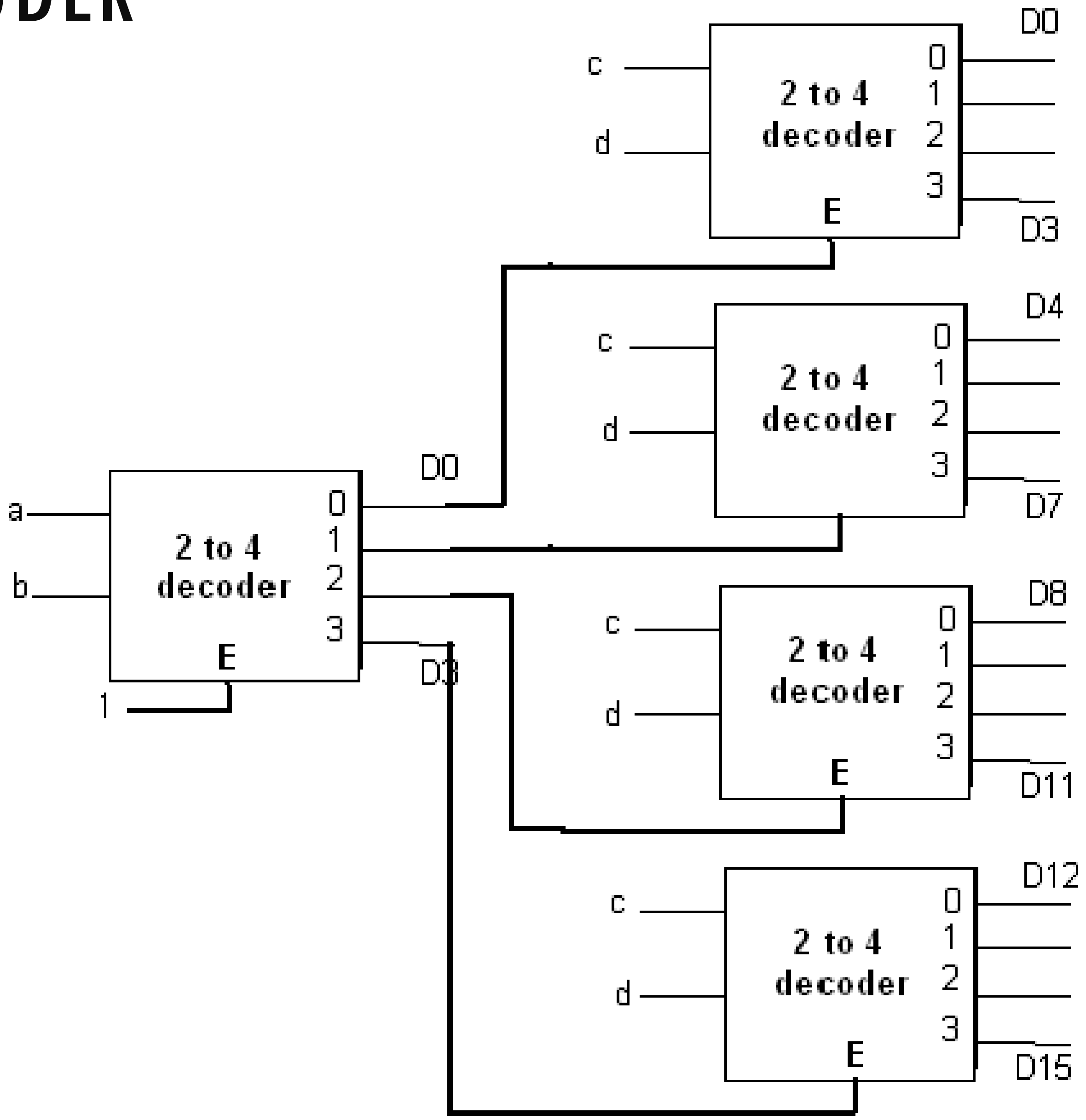


4×16 Decoder

EXERCISE

- Can you sketch a 4×16 decoder using a number of 2×4 decoders?
- I am now giving you 5 minutes to attempt it
- After these 5 minutes, the lecture will continue

4 × 16 DECODER USING 2 × 4 DECODERS



ENCODERS

- An encoder is a digital circuit that performs the inverse operation of a decoder
- An encoder has 2^n input lines and n output lines
- The output lines generate the binary equivalent of the input line whose value is 1

8×3 OCTAL-TO-BINARY ENCODER

$$x = D_4 + D_5 + D_6 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$z = D_1 + D_3 + D_5 + D_7$$

What happens if more than one input is active (set to HIGH) at the same time? For example, D_3 and D_6?

What happens if all inputs are equal to 0?

Inputs								**Outputs**		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

8×3 OCTAL-TO-BINARY ENCODER

What happens if more than one input is active (set to HIGH) at the same time? For example, D_3 and D_6?

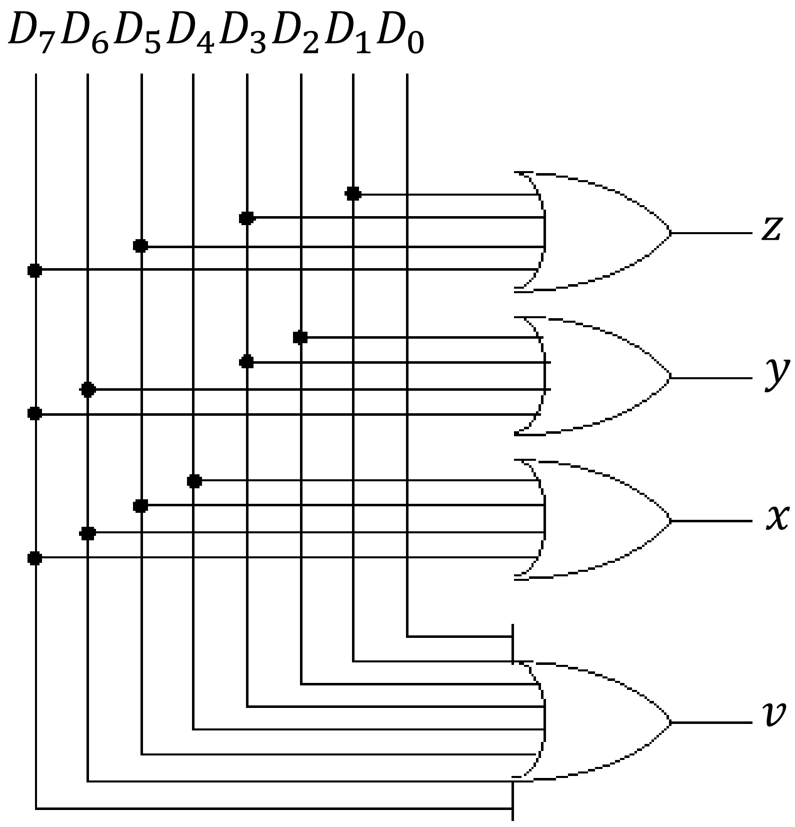
- If D_3 and D_6 are active simultaneously, the output would be $(111)_2 = (7)_{10}$, because all three outputs would be equal to 1
- But this does not reflect the actual input which should have resulted in an output of $(011)_2 = (3)_{10}$ for D_3 or $(110)_2 = (6)_{10}$ for D_6
- To overcome this problem, we use *priority encoders*
- If we establish a higher priority for inputs with higher subscript numbers, and if both D_3 and D_6 are active at the same time, the output would be $(110)_2$ because D_6 has higher priority than D_3

$$x = D_4 + D_5 + D_6 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$z = D_1 + D_3 + D_5 + D_7$$

8×3 OCTAL-TO-BINARY ENCODER

What happens if all inputs are equal to 0?

- The encoder output would be $(000)_2$, but in fact this is the output when D_0 is equal to 1
- This problem can be solved by providing an extra output to indicate whether at least one input is equal to 1
- v is the *valid* output



4×2 PRIORITY ENCODER

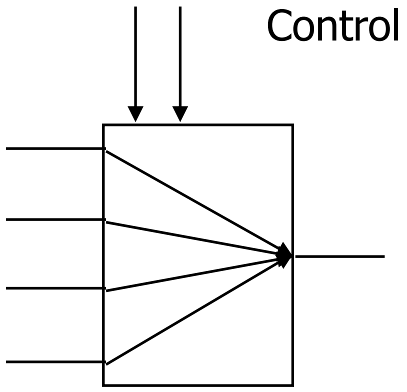
- The input D_3 has the highest priority, regardless of the values of the other inputs
- Thus, if D_3 is 1, the output will indicate that $A_1A_0 = 11$, i.e. the code $A_1A_0 = 11$ means that any data appearing on line D_3 will have the highest priority and will pass through the system irrespective of other inputs
- If $D_2 = 1$ and $D_3 = 0$, the output code will be $A_1A_0 = 10$ and this means that D_2 has the highest priority in this case

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	0	0	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

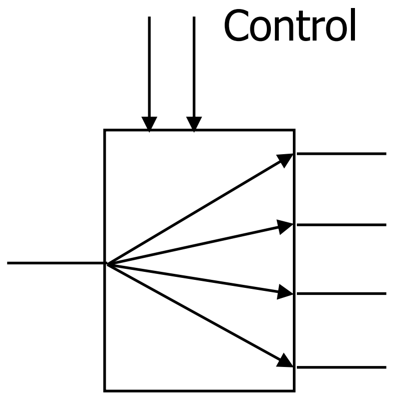
GUC
German University in Cairo

MAKING CONNECTIONS

- Direct point-to-point connections between gates are made up of *wires*
- Routing one of many inputs to a single output is carried out using a *multiplexer*
- Routing a single input to one of many outputs is carried out using a *demultiplexer*



Multiplexer



Demultiplexer

MULTIPLEXERS

- A multiplexer is used to connect 2^n points to a single point
- The control signal pattern forms the binary index of the input to be connected to the output

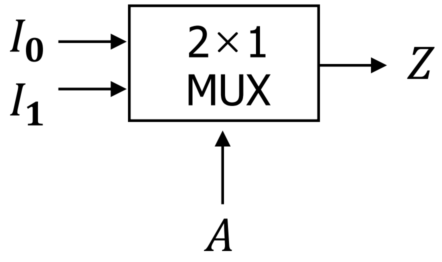
I_1	I_0	A	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Logical form

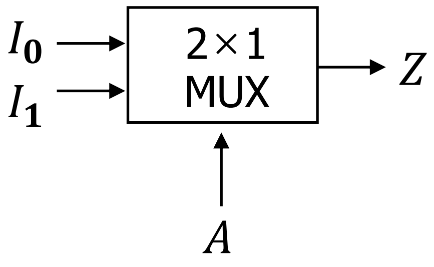
$$Z = A'I_0 + AI_1$$

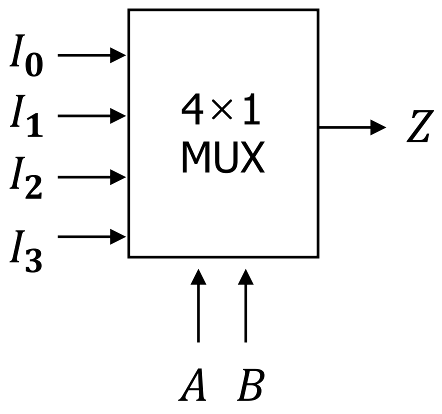
A	Z
0	I_0
1	I_1

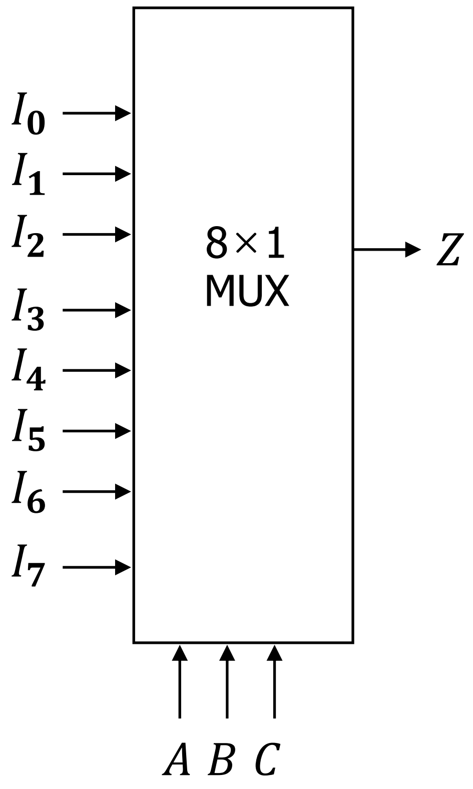
Functional form



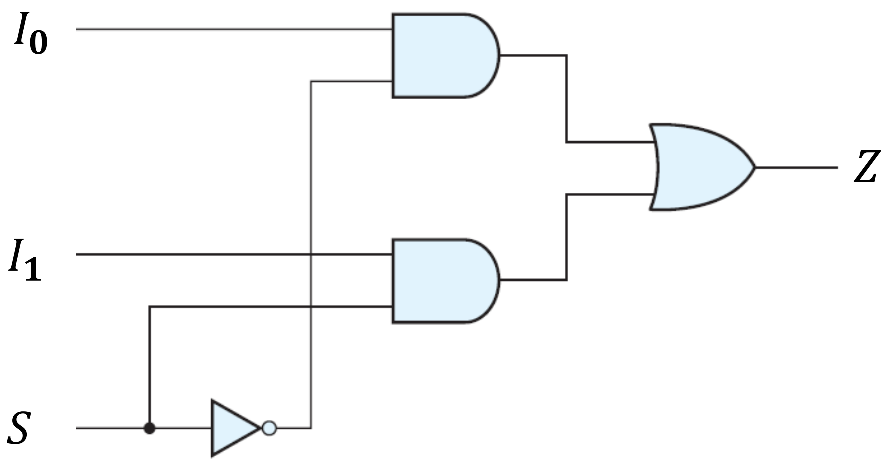
MULTIPLEXERS

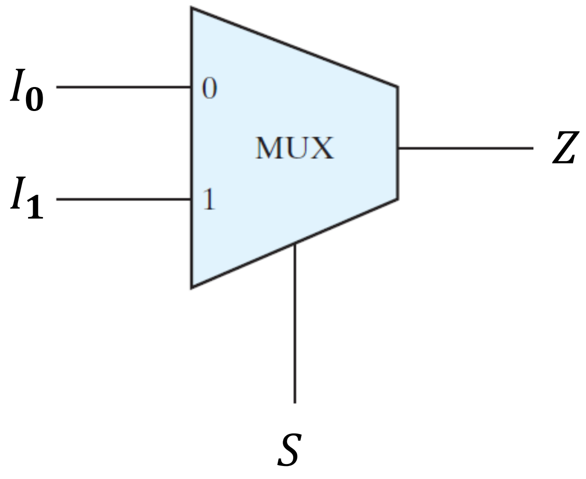






2×1 LINE MULTIPLEXER

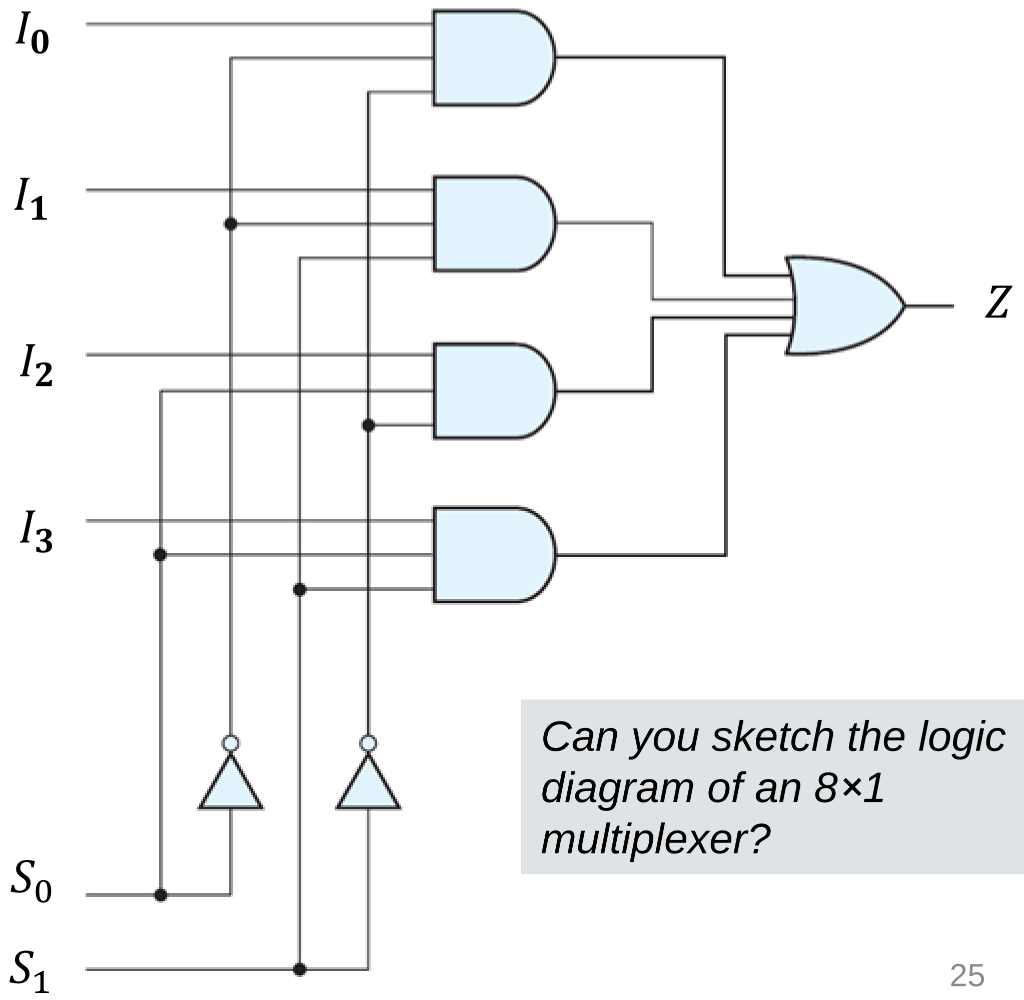




4×1 LINE MULTIPLEXER

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

Functional form

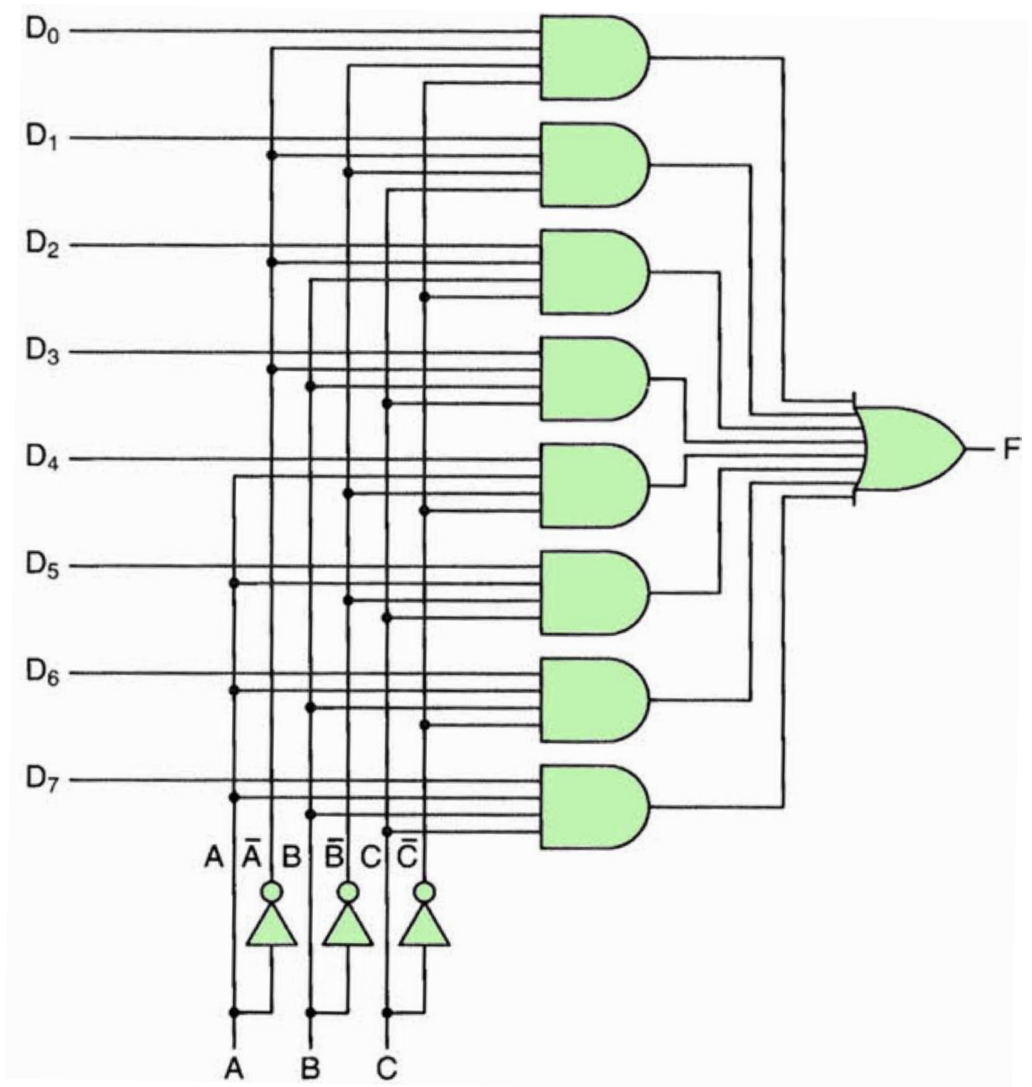


Can you sketch the logic diagram of an 8×1 multiplexer?

8×1 LINE MULTIPLEXER

- I am now giving you 5 minutes to attempt sketching the logic diagram of an 8×1 line multiplexer
- After these 5 minutes, the lecture will continue

8×1 LINE MULTIPLEXER



MULTIPLEXERS AS GENERAL-PURPOSE LOGIC

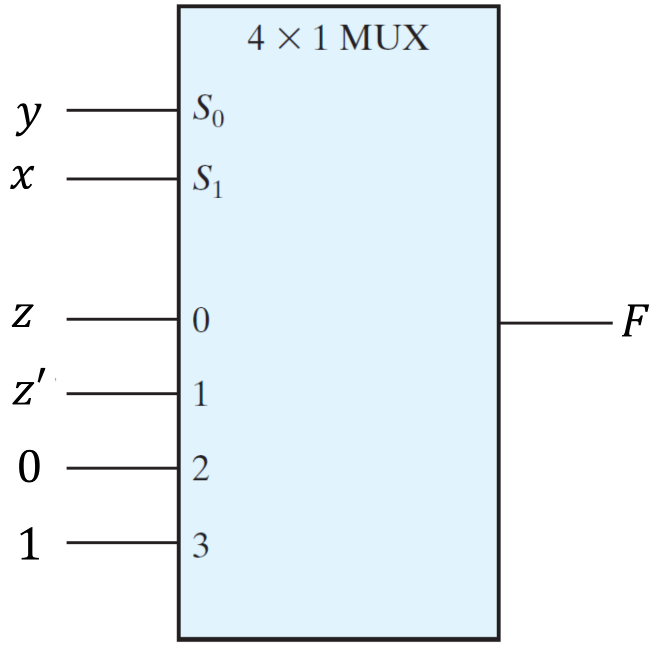
- A $2^{n-1}:1$ multiplexer can implement any function of n variables
- Steps:
  1. The Boolean function is listed in a truth table
  2. The first $n-1$ variables in the table are applied to the selection inputs of the MUX
  3. For each combination of the selection variables, evaluate the output as a function of the last variable
  4. The values are then applied to the data inputs in the proper order

German University in Cairo

MULTIPLEXERS AS GENERAL-PURPOSE LOGIC: EXAMPLE I

$$F(x,y,z) = \Sigma(1,2,6,7)$$

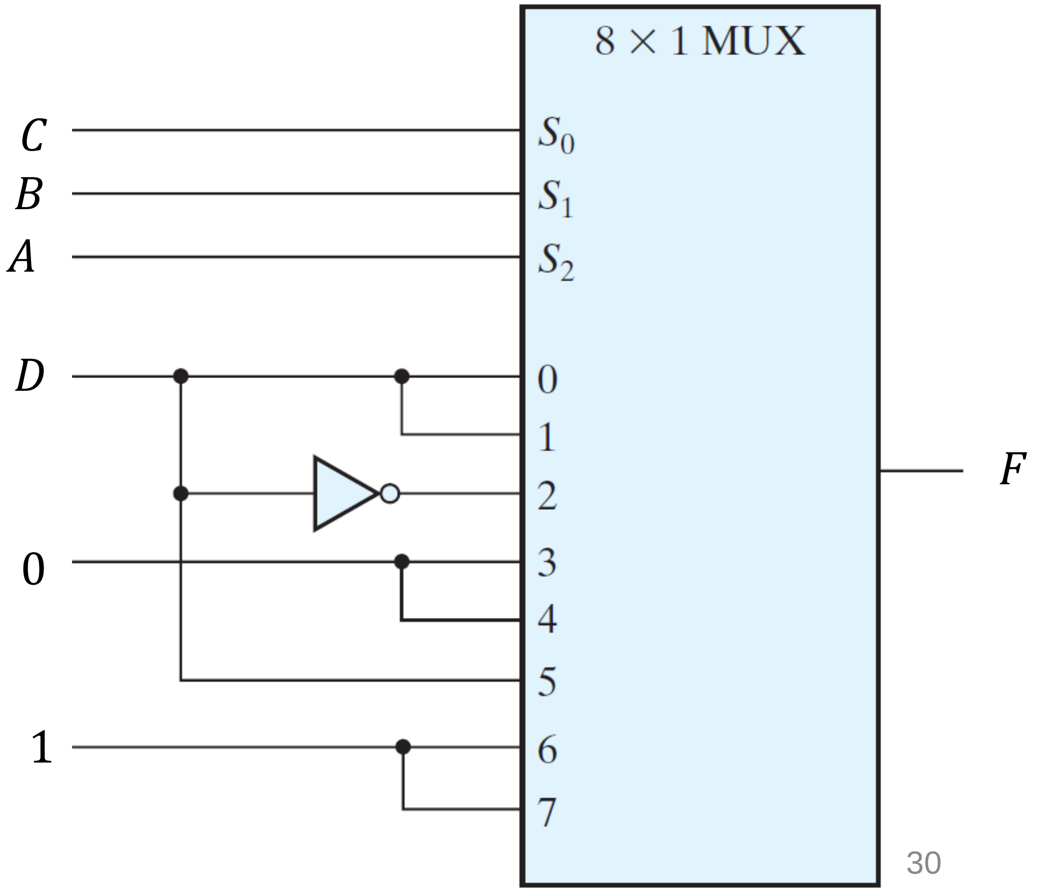
x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	



MULTIPLEXERS AS GENERAL-PURPOSE LOGIC: EXAMPLE II

$$F(A,B,C,D) = \Sigma(1,3,4,11,12,13,14,15)$$

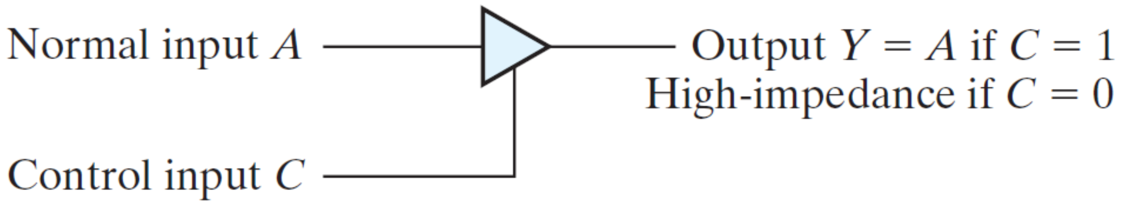
A	B	C	D	F
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	



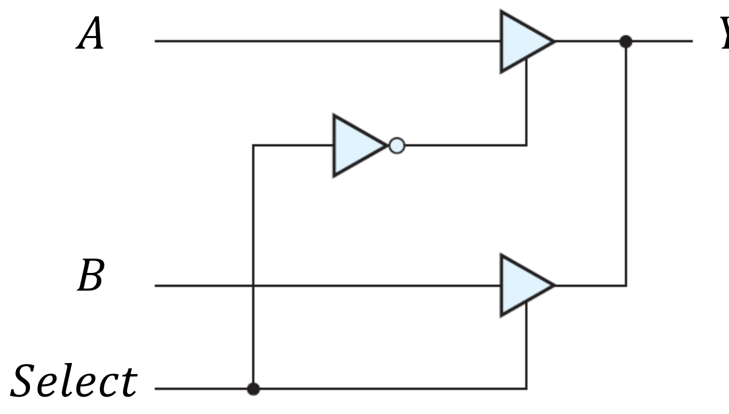
THREE-STATE BUFFERS (TRI-STATE BUFFERS)

- These are digital circuits that exhibit three states
- Two of these states are logic 0 and logic 1
- The third state is a *high-impedance* state in which:

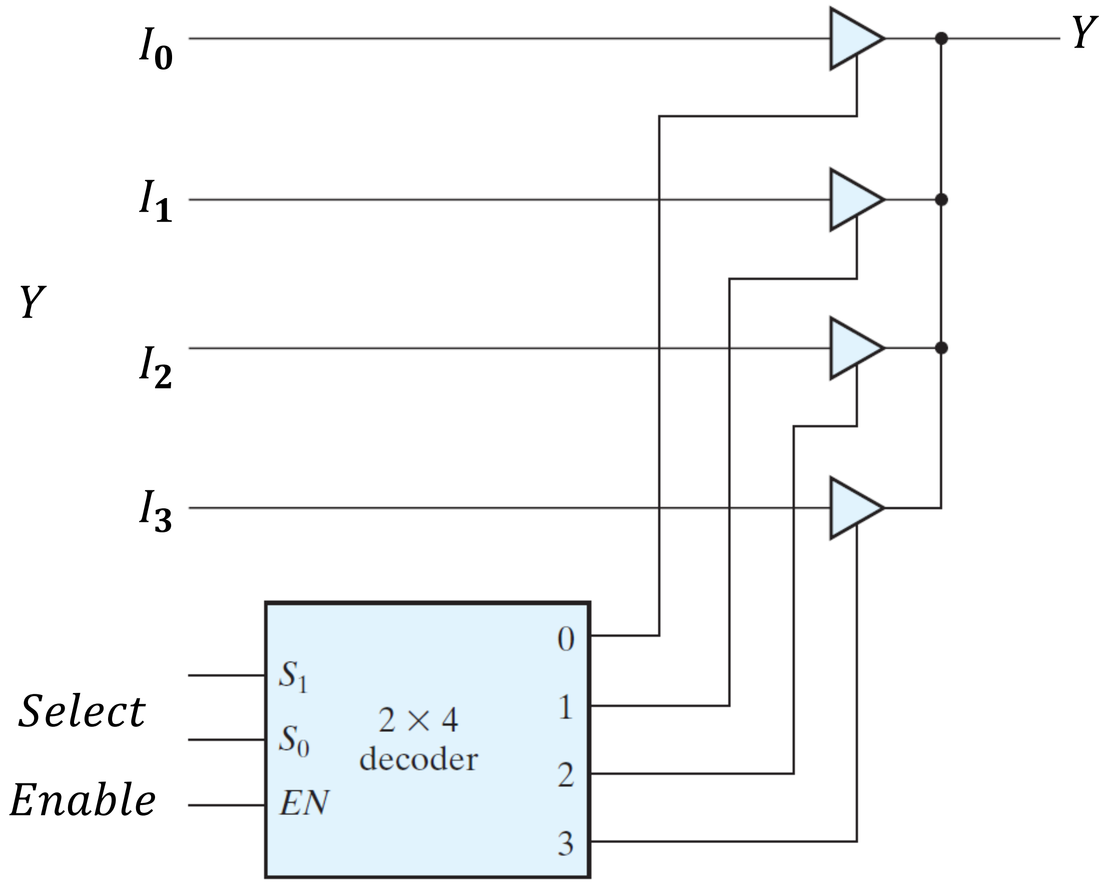
1. *The logic behaves like an open circuit*
2. *The circuit has no logic significance*
3. *The circuit connected to the output of the three-state gate is not affected by the inputs to the gate*



MULTIPLEXERS USING THREE-STATE BUFFERS



2 × 1 line MUX



4 × 1 line MUX