

Digital Logic Design Chapter V Combinational logic

Addis Ababa Science and
Technology College of Electrical and
Mechanical Engineering

Lecturer: Yonas Haregot

Introducti on

- Logic circuit for digital systems maybe combinational or sequential.
- A **combinational circuit** consists of **logic gates** whose outputs at any time are determined from only the present combination of inputs.
- A combinational circuit performs an operation that can be specified logically by a set of Boolean functions.
- In contrast, **sequential circuits** employ **storage elements** in addition to **logic gates**.
- The outputs of sequential circuits are a function of the input and the state of the storage elements.

Basic combinational logic circuits

- AND-OR logic circuit
 - Directly implements an SOP expression.
- AND-OR-Inverter logic circuit
 - This kind of implementation leads to POS expression.
- Exclusive-OR logic circuit
 - A combination of two AND gates, one OR gate and two inverters.
- Exclusive-NOR logic circuit
 - Implemented by simply inverting the output of an exclusive-OR.

Functions of combinational logic circuit

- A combinational circuit consists of an interconnection of logic gates.
- For n inputs variables, there are 2^n possible combination of the binary inputs. For each possible input combination, there is one possible value for each output variable.
- A combinational circuit also can be described by m Boolean functions, one for each output variable. Each output function is expressed in terms of the n output variables

DESIGN

PROCEDURE

The design of combinational circuits starts from the specification of the **design objectives** and is converted to a set of Boolean functions from which the logic diagram can be obtained.

- The procedure involves the following steps;
 1. Understand the full operation
 2. Determine the required number of inputs and outputs and assign a symbol to each.
 3. Derive the truth table that defines the required relationship between inputs and outputs.
 4. Obtain the simplified Boolean functions for each output as a function of the input variables.
 5. Draw the logic diagram and verify the correctness of the design.

BASIC ADDERS (Arithmetic Binary Adder)

- Adders are important in computers and also in other types of digital systems in which numerical data are processed.
- An understanding of the basic adder operation is fundamental to the study of digital systems.
- **Th** Recall the basic rules for binary addition as stated in Chapter 2.

□

$$0 + 0 = 0$$

$$0 + 1 = 1$$

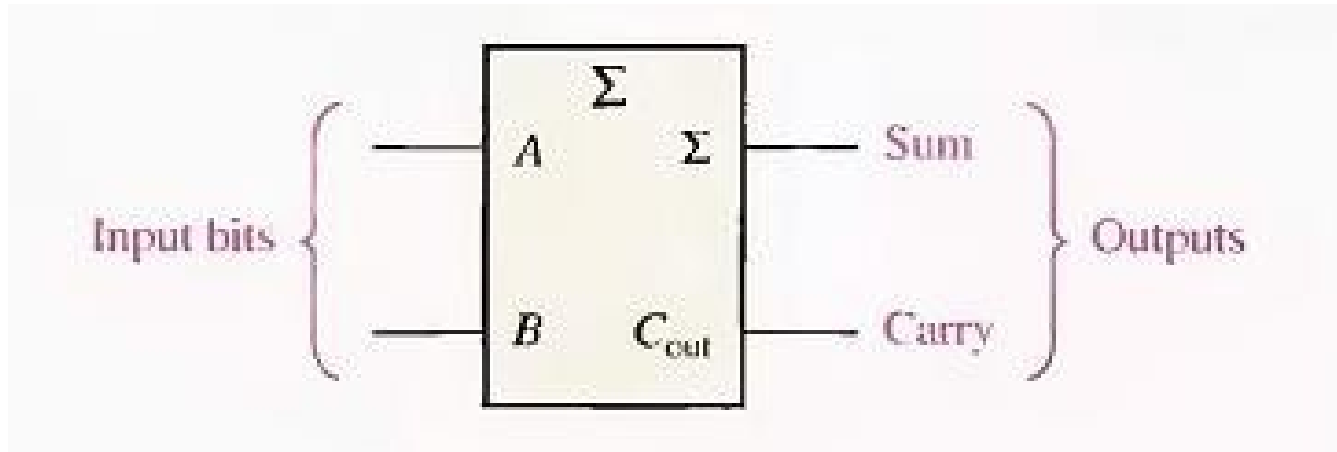
$$1 + 0 = 1$$

$$1 + 1 = 10$$

- The operations are performed by a logic circuit called a **half-adder**

BASIC ADDERS (Arithmetic Binary Adder)

- The half-adder accepts two binary digits on its inputs and produces two binary digits on its output, a **sum bit** and



- Expressions can be derived for the sum and the output carry as functions of the inputs. Notice that the output carry is a 1 only when both A and B are 1. so **$C_{out} = AB$**

BASIC ADDERS (Arithmetic Binary Adder)

- Observe that the sum output is a 1 only if the input variables A and B are not equal. The sum can therefore be expressed as the XOR of the input variables.

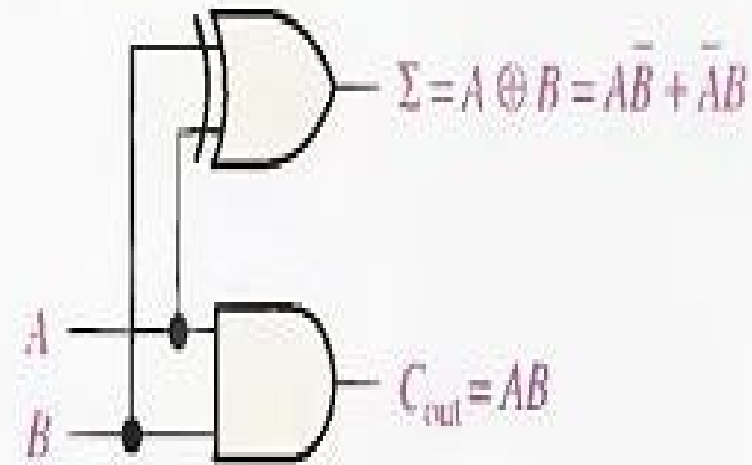
$$\Sigma = A \oplus B$$

A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = sum

C_{out} = output carry

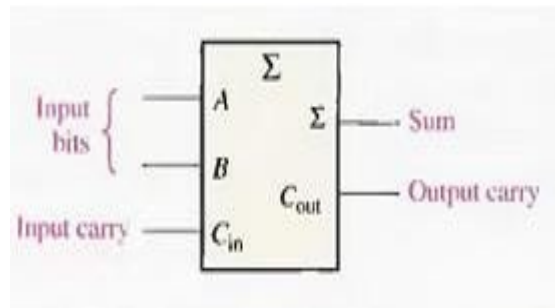
A and B = input variables (operands)



BASIC ADDERS (Arithmetic Binary Adder)

• The Full-Adder

- The second category of adder is the full-adder
- The full-adder accepts two inputs bits and an input carry and generates a sum output and an output carry. The basic difference between a full-adder and a half-adder is that the full-adder accepts an input carry



A	B	C _{in}	C _{out}	Σ
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

C_{in} = input carry, sometimes designated as CI

C_{out} = output carry, sometimes designated as CO

Σ = sum

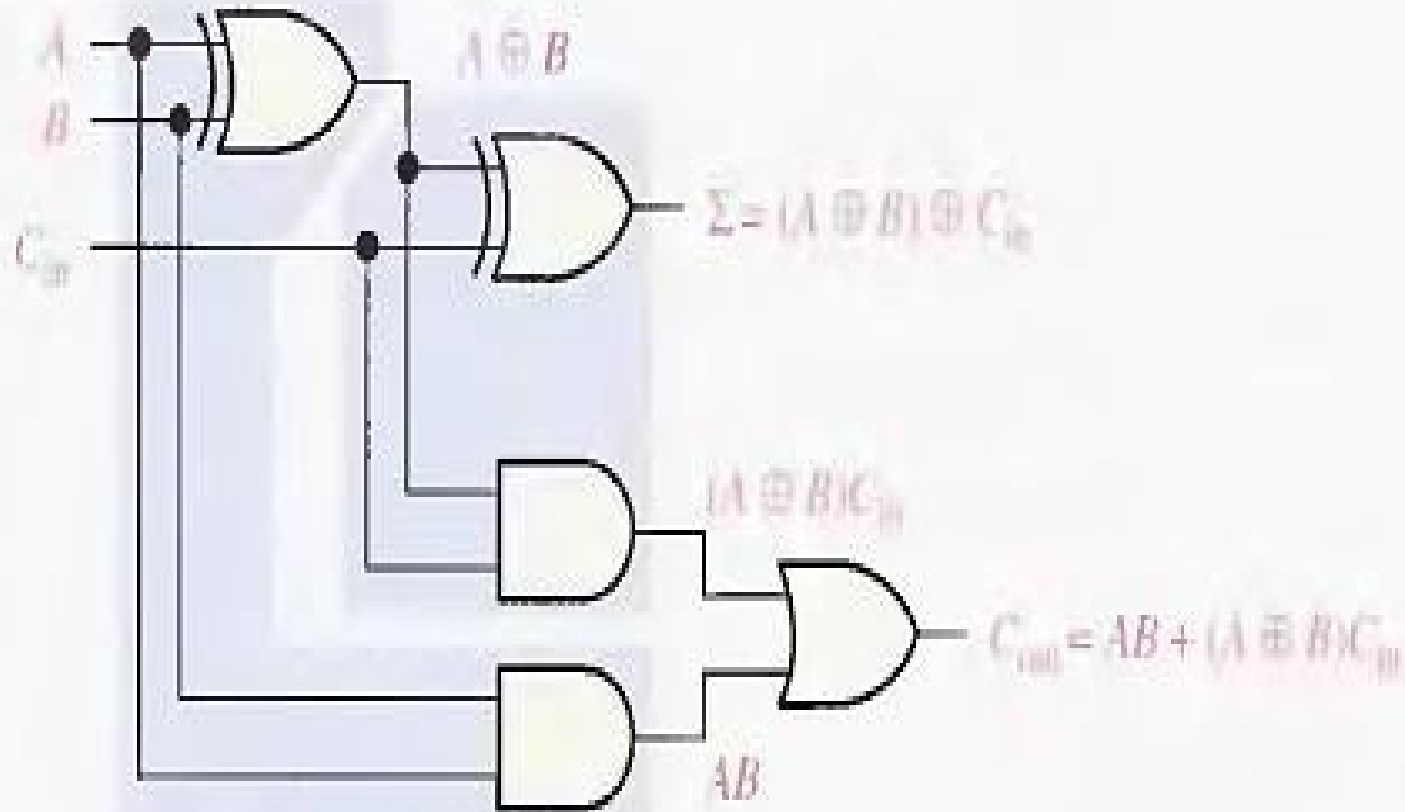
A and B = input variables (operands)

BASIC ADDERS (Arithmetic Binary Adder)

- The full adder must add the two input bits and the input carry.
- From the half-adder you know that the sum of the input bits A and B is the XOR of those two variables.
- For the input carry to be added to the input bits, it must be XORed with
- For the input carry (C_{in}) to be added to the input bits, it must be XORed with . This means that to implement the full-adder sum function, the 2-input XOR gate can be used.

$$\Sigma = (A \oplus B) \oplus C_{in}$$

BASIC ADDERS (Arithmetic Bina

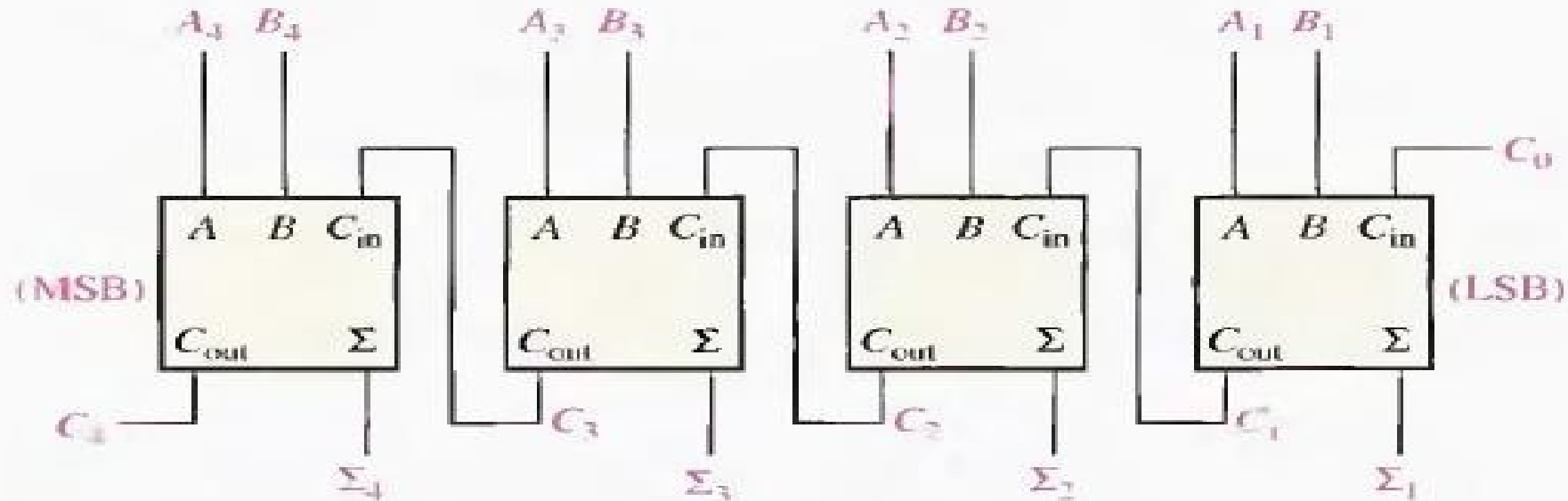


Parallel Binary Adders

- Two or more full-adders are connected to form parallel binary adders.
- To add two binary numbers, a full-adder is required for each bit in the numbers. So,
 - For 2-bit numbers, two adders are needed
 - For 4-bit numbers, four adders are used and so on
- The carry output of each adder is connected to the carry input of the next higher-order adder.

4-bit Parallel Adders

- A group of four bits is called

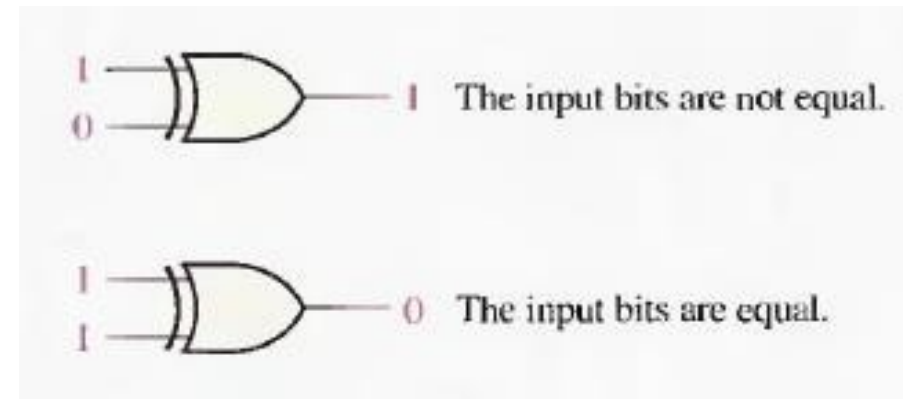
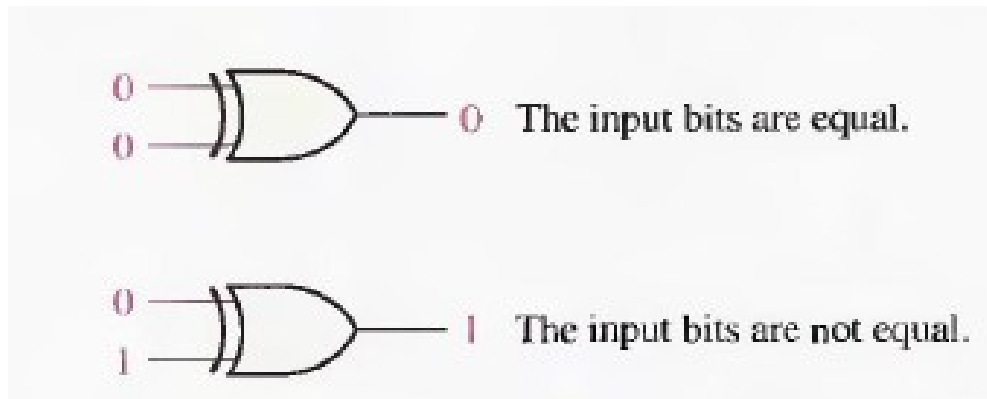


(a) Block diagram

Magnitude

Comparators

- The basic function of a comparator is to compare the magnitude of two binary quantities to determine the relationship of those quantities.
- Example 2-bit comparator;



- In order to compare binary numbers containing two bits each, an additional XOR gate is necessary

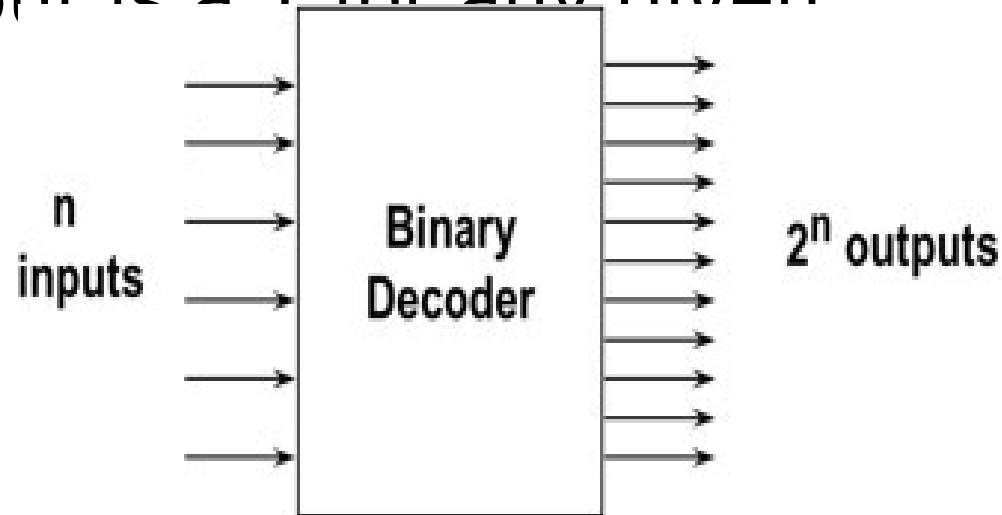
Decode

rs

- A **decoder** is digital circuit that **detects** the presence of a specified combination of bits (**code**) on its inputs and indicates the presence of that code by a specified output level.
- In its general form, a decoder has n inputs lines to handle n bits and from 1 to 2^n output lines to indicate the presence of one or more n - bit combinations.
- Decoder can be developed using AND logic gates.

Decoder

- Black box with n inputs lines and output lines.
- Only one output is a 1 for any given input.

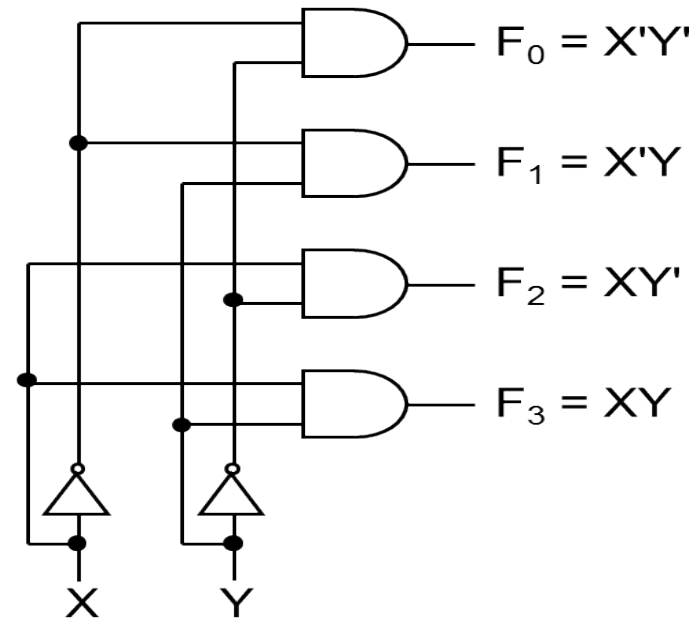
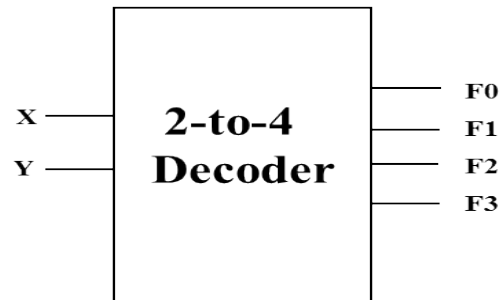


2-to-4 Binary Decoder

Truth Table

X	Y	F ₀	F ₁	F ₂	F 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

- From truth table, circuit for 2x4 decoder is:
- Note: Each output is a 2-variable minterm ($X'Y'$, $X'Y$, XY' or XY)



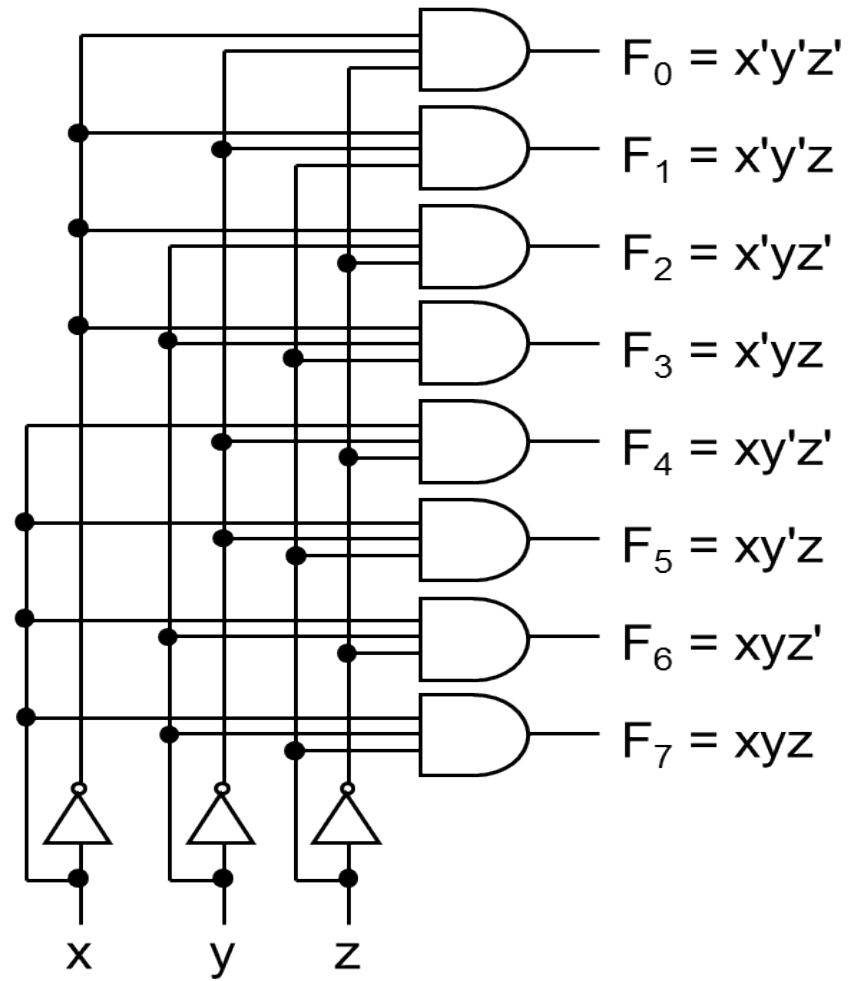
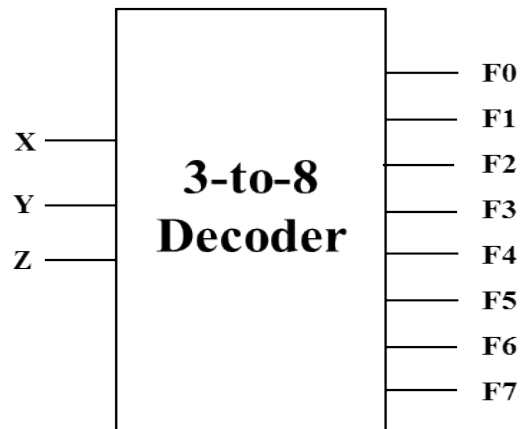
Individual Assignments 1

Design a decoder for 3 binary
input

3-to-8 Binary Decoder

Truth Table:

x	y	z	F ₀	F ₁	F ₂	F ₃	F ₄	F ₅	F ₆	F 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



Decoder with an enable input

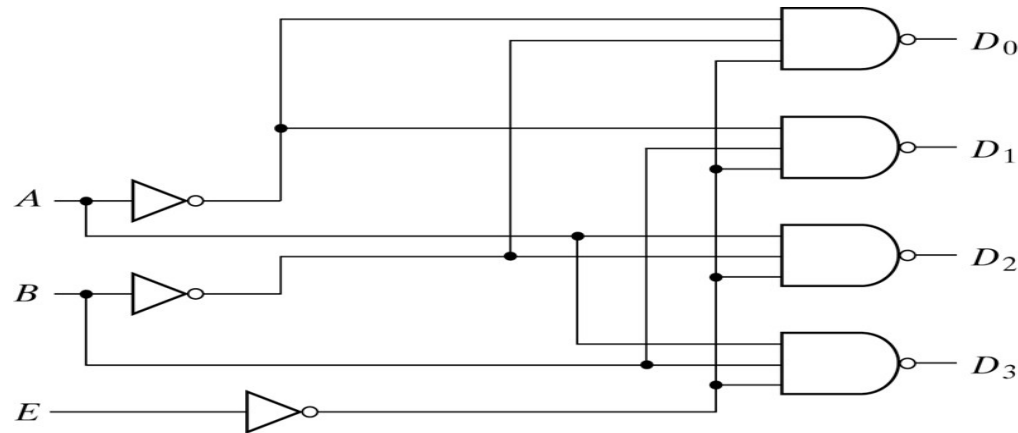
- Decoders include one or more enable inputs to control the circuit operation.

Building a Binary Decoder with NAND Gates

- Start with a 2-bit decoder
 - Add an enable signal (E)

Note: use of NANDs

only one 0 active!



if E = 0

<i>E</i>	<i>A</i>	<i>B</i>	<i>D</i> ₀	<i>D</i> ₁	<i>D</i> ₂	<i>D</i> ₃
1	<i>X</i>	<i>X</i>	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

(a) Logic diagram

(b) Truth table

Fig. 4-19 2-to-4-Line Decoder with Enable Input

4-to-16 line decoder (using two 3-to-8 decoder)

- In order to decoder all possible combinations are required. $2^4 = 16^{\text{th}}$ combinations are required.
- Decoders with enable inputs can be connected together to form a larger decoder circuit.

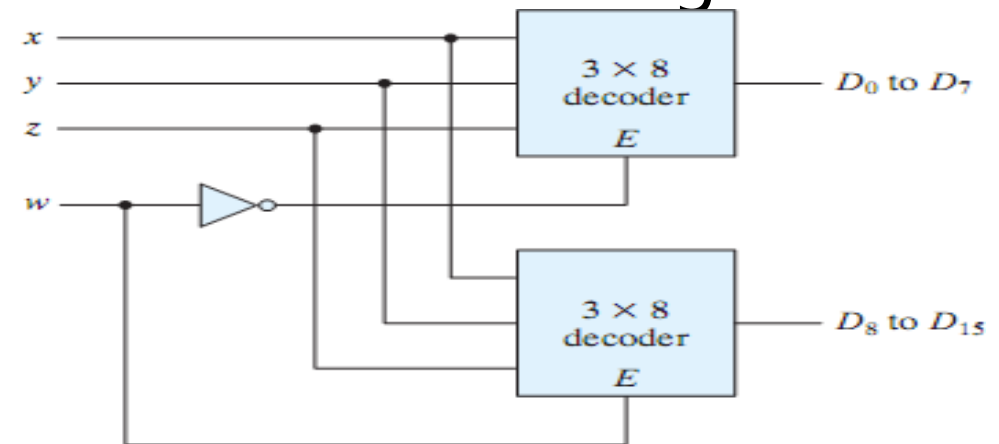
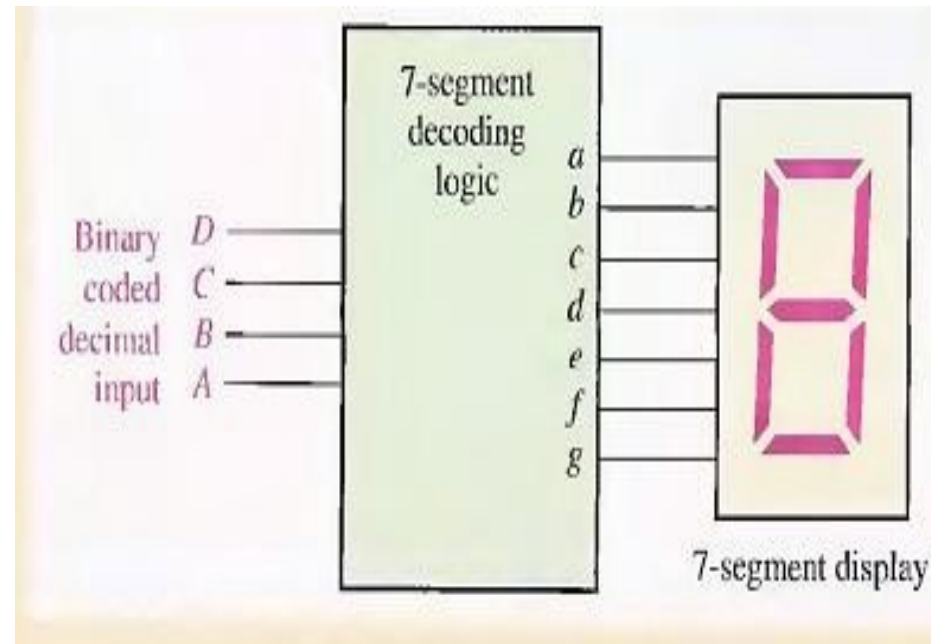


FIGURE 4.20

4 $\times$ 16 decoder constructed with two 3 $\times$ 8 decoders

Decoder - Applications

- The BCD-to-7-Segment Decoder



DECIMAL DIGIT	INPUTS				SEGMENT OUTPUTS						
	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	X	X	X	X	X	X	X
11	1	0	1	1	X	X	X	X	X	X	X
12	1	1	0	0	X	X	X	X	X	X	X
13	1	1	0	1	X	X	X	X	X	X	X
14	1	1	1	0	X	X	X	X	X	X	X
15	1	1	1	1	X	X	X	X	X	X	X

Output = 1 means segment is activated (on)

Output = 0 means segment is not activated (off)

Output = X means "don't care"

Decoder - Applications

- The BCD-to-Decimal Decoder
 - 4-line-to-10-line decoder

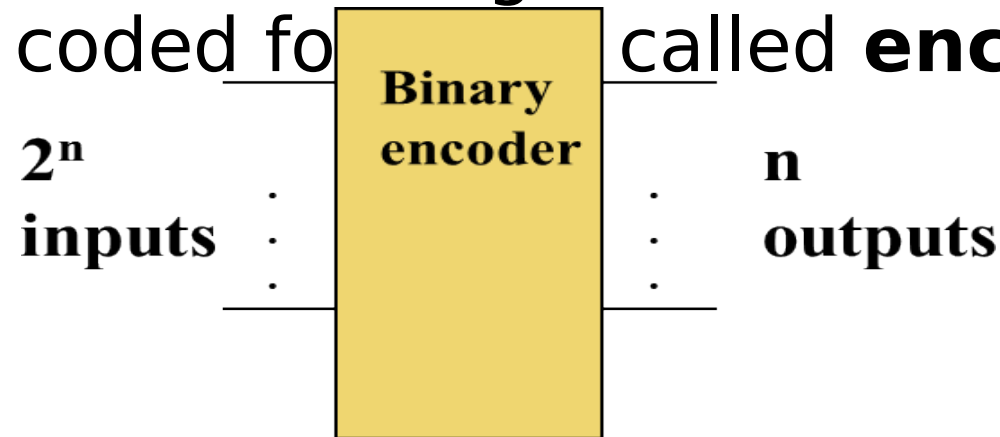
DECIMAL DIGIT	BCD CODE				DECODING FUNCTION
	A_3	A_2	A_1	A_0	
0	0	0	0	0	$\bar{A}_3\bar{A}_2\bar{A}_1\bar{A}_0$
1	0	0	0	1	$\bar{A}_3\bar{A}_2\bar{A}_1A_0$
2	0	0	1	0	$\bar{A}_3\bar{A}_2A_1\bar{A}_0$
3	0	0	1	1	$\bar{A}_3\bar{A}_2A_1A_0$
4	0	1	0	0	$\bar{A}_3A_2\bar{A}_1\bar{A}_0$
5	0	1	0	1	$\bar{A}_3A_2\bar{A}_1A_0$
6	0	1	1	0	$\bar{A}_3A_2A_1\bar{A}_0$
7	0	1	1	1	$\bar{A}_3A_2A_1A_0$
8	1	0	0	0	$A_3\bar{A}_2\bar{A}_1\bar{A}_0$
9	1	0	0	1	$A_3\bar{A}_2\bar{A}_1A_0$

Individual Assignments X

Design a binary(4) to HEX
Decoder

Encoder

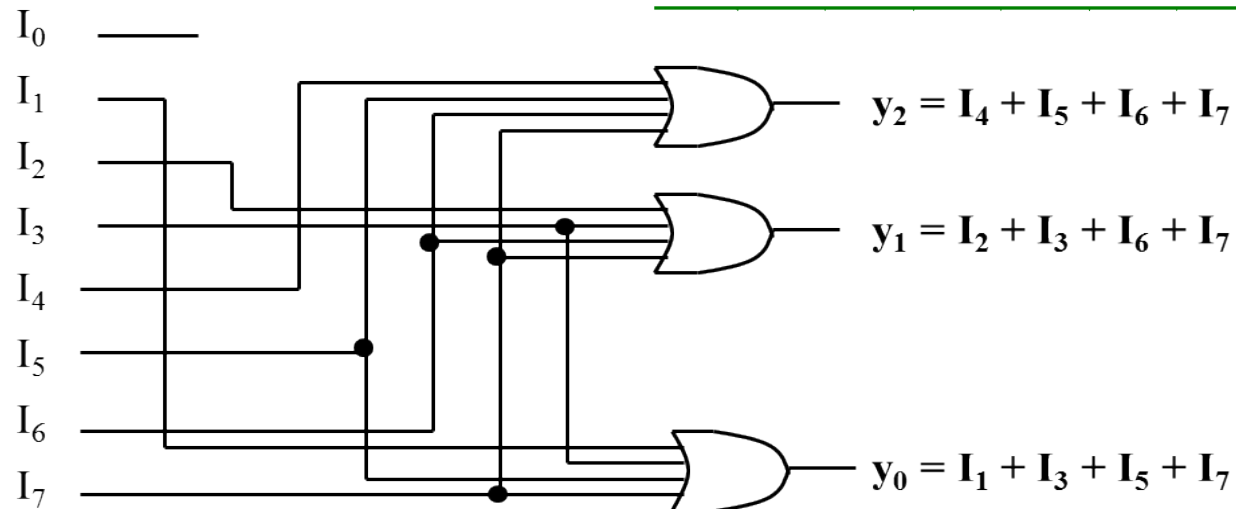
- An **encoder** accepts an active level on one of its inputs representing a digit, such as a **decimal** or **octal** digit, and converts it to a coded **output**, such as **BCD** or **binary**.
- The process of converting from familiar symbols or numbers to a coded form is called **encoding**.



8-to-3 Binary Encoder

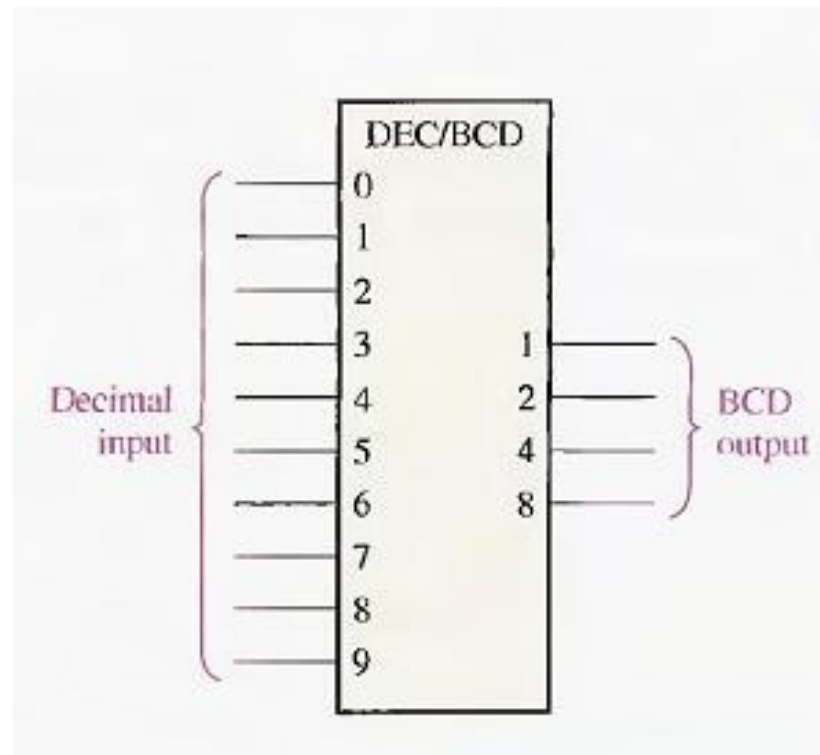
At any one time, only
one input line has a value of 1.

Inputs								Outputs		
I ₀	I ₁	I ₂	I ₃	I ₄	I ₅	I ₆	I ₇	y ₂	y ₁	y ₀
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



Encoder - Application

- **The Decimal-to-BCD Encoder**



e-to

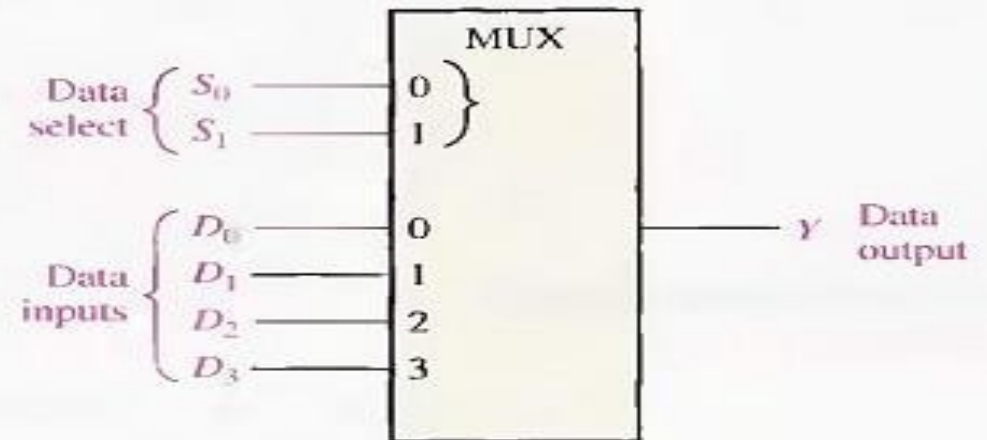
DECIMAL DIGIT	BCD CODE			
	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Individual Assignments 2

Design a HEX to BCD
Encoder

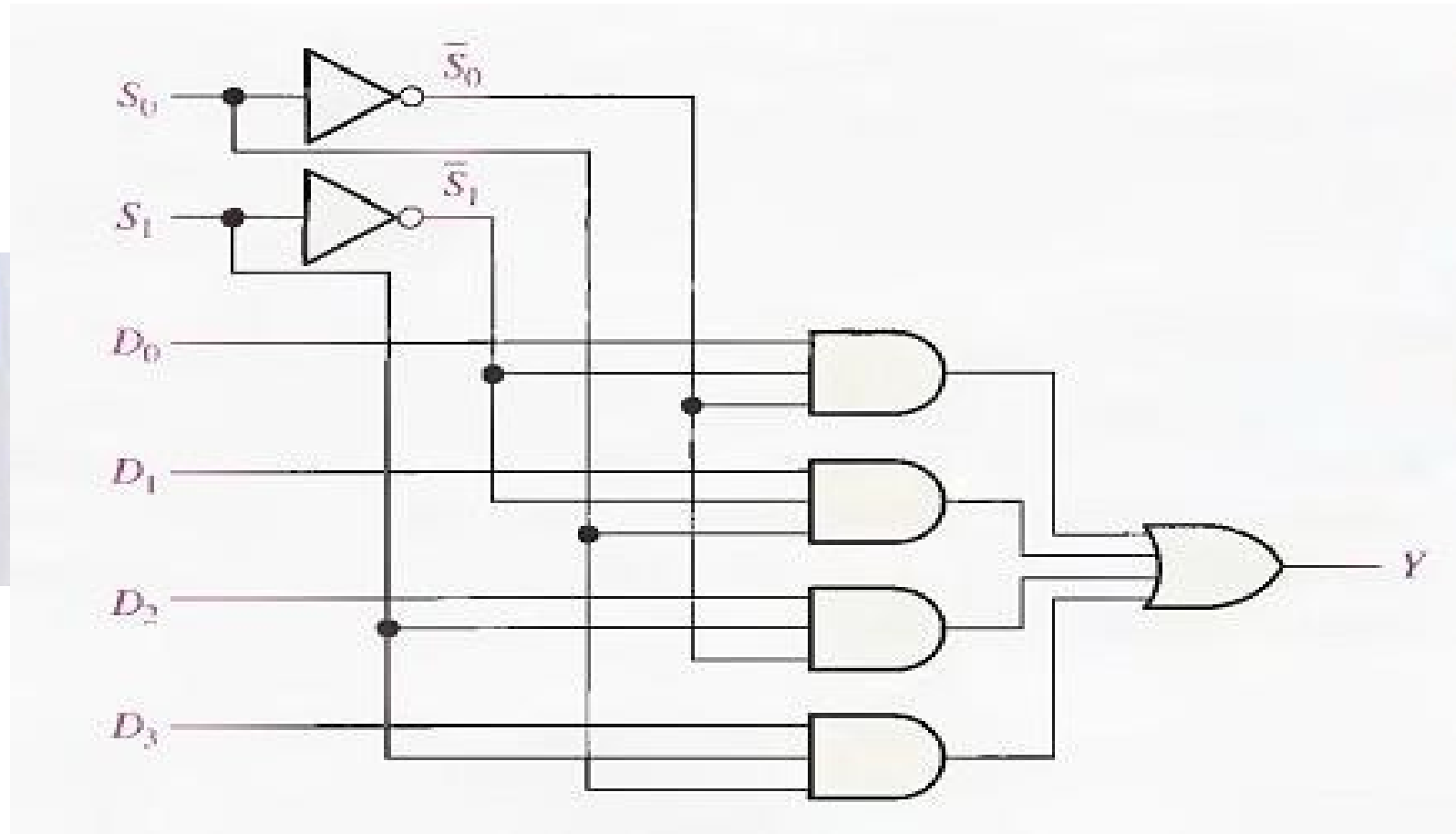
Multiplexers (Data Selectors)

- A multiplexer (MUX) is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination.
- The basic multiplexer has several data-input lines and a single output line.
- It has data-select inputs, which permit any one of the inputs to be switched to the output line.



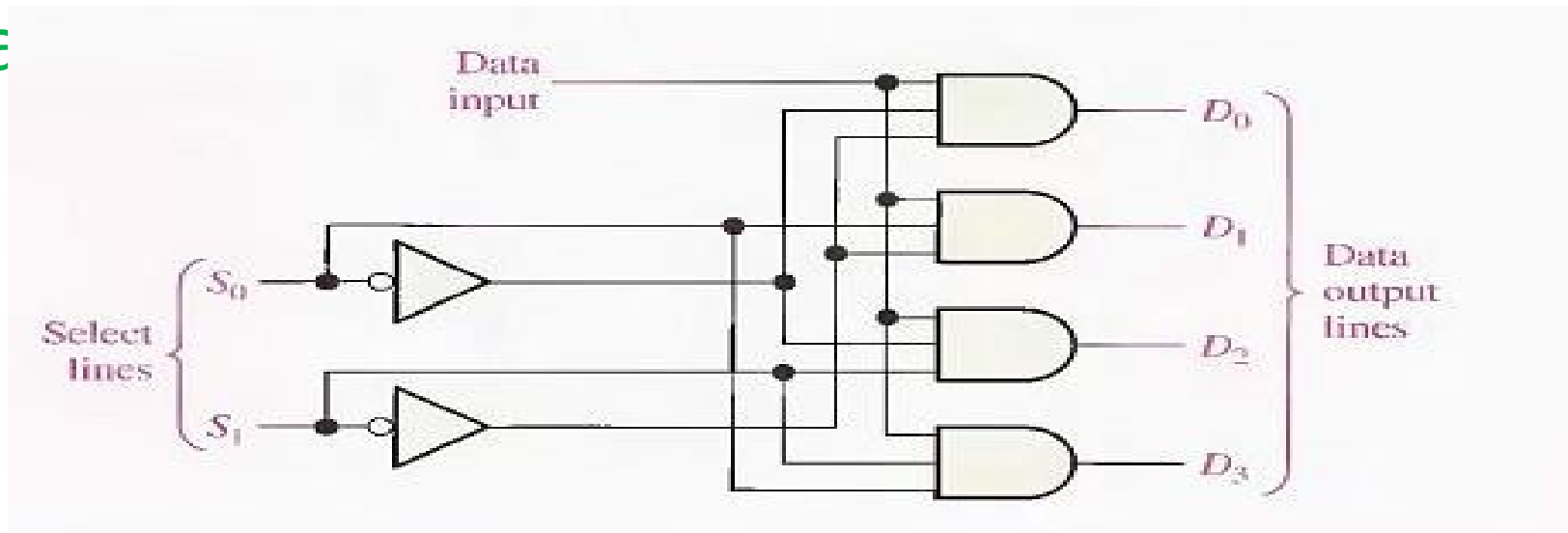
Multiplexers (Data Selectors)

DATA-SELECT INPUTS		INPUT SELECTED
S_1	S_0	
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3



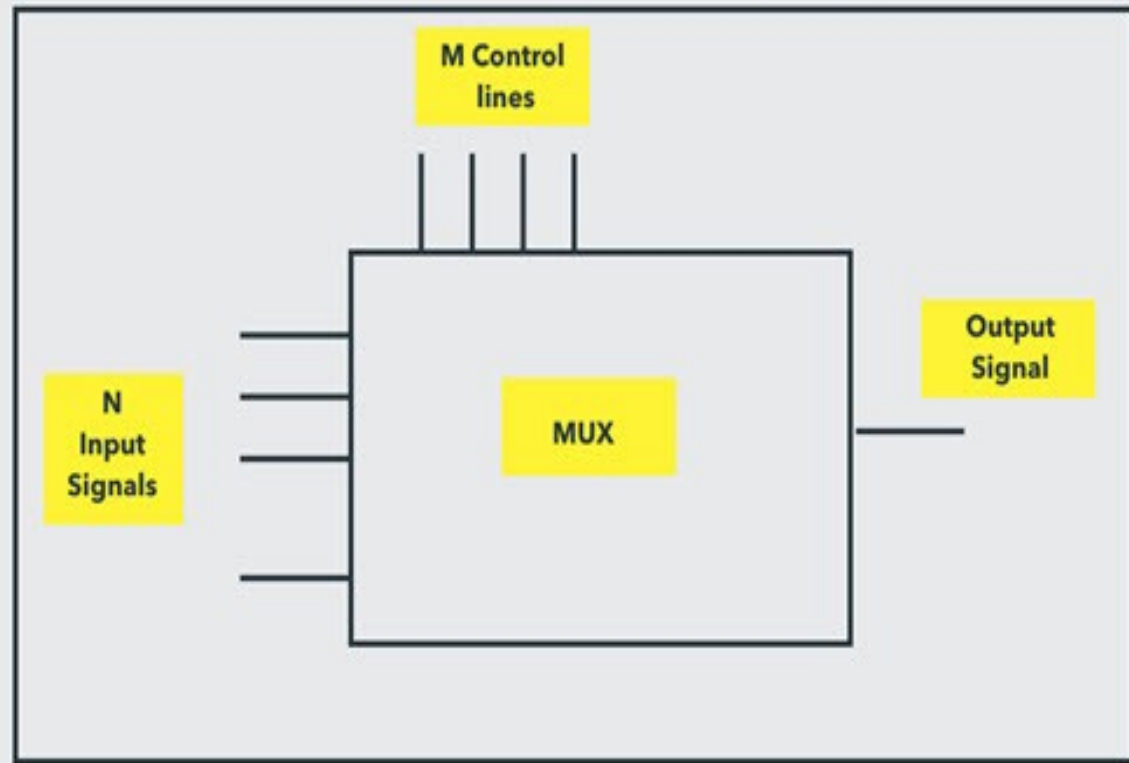
Demultiplexers (Data Distributor)

- A demultiplexer (DEMUX) basically **reverse** the **multiplexing** function.
- It takes **digital information form one line** and **distributes** it to a given number of **output lines**.
- Data

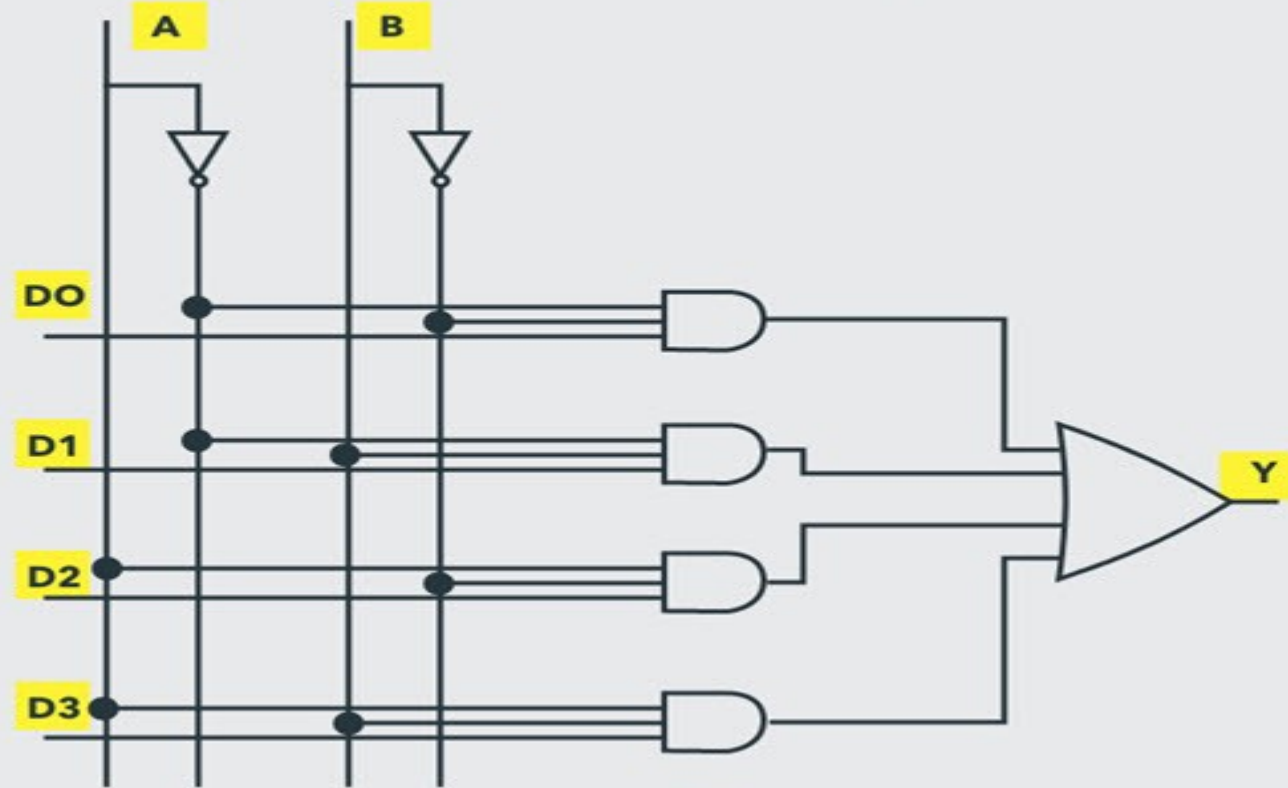


Multiplexer

Block Diagram Of A Multiplexer

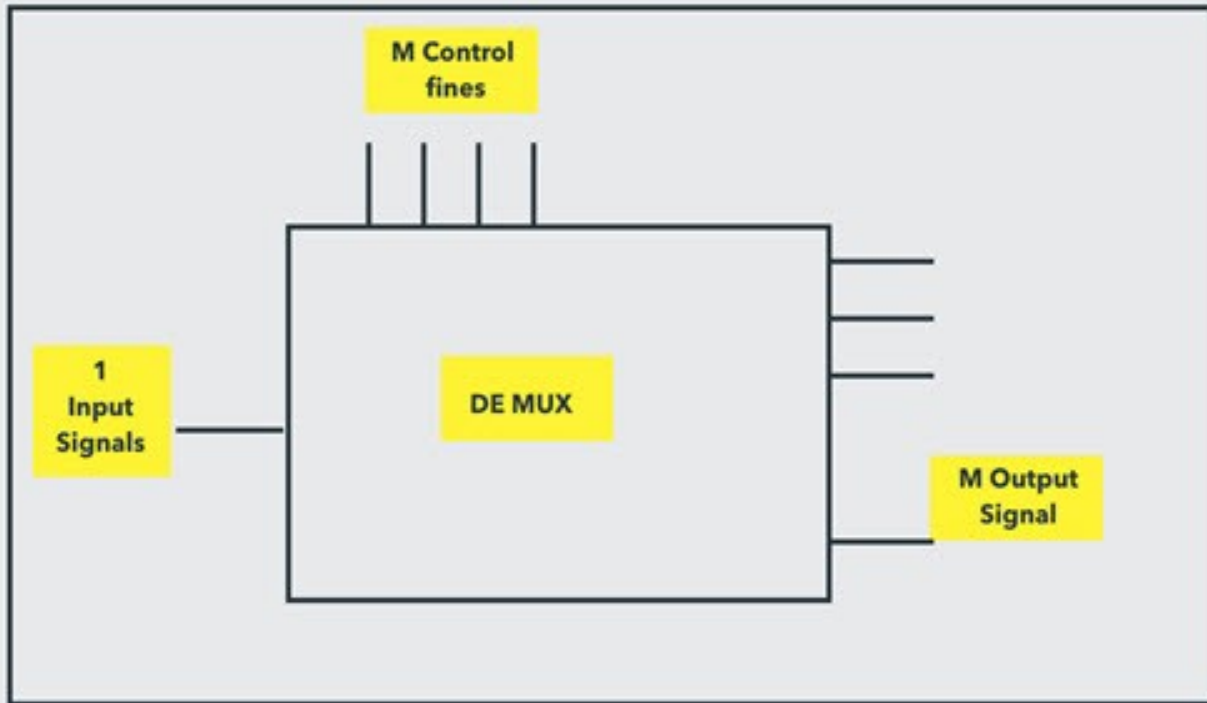


Logic Circuit Of 4-To-1 Multiplexer

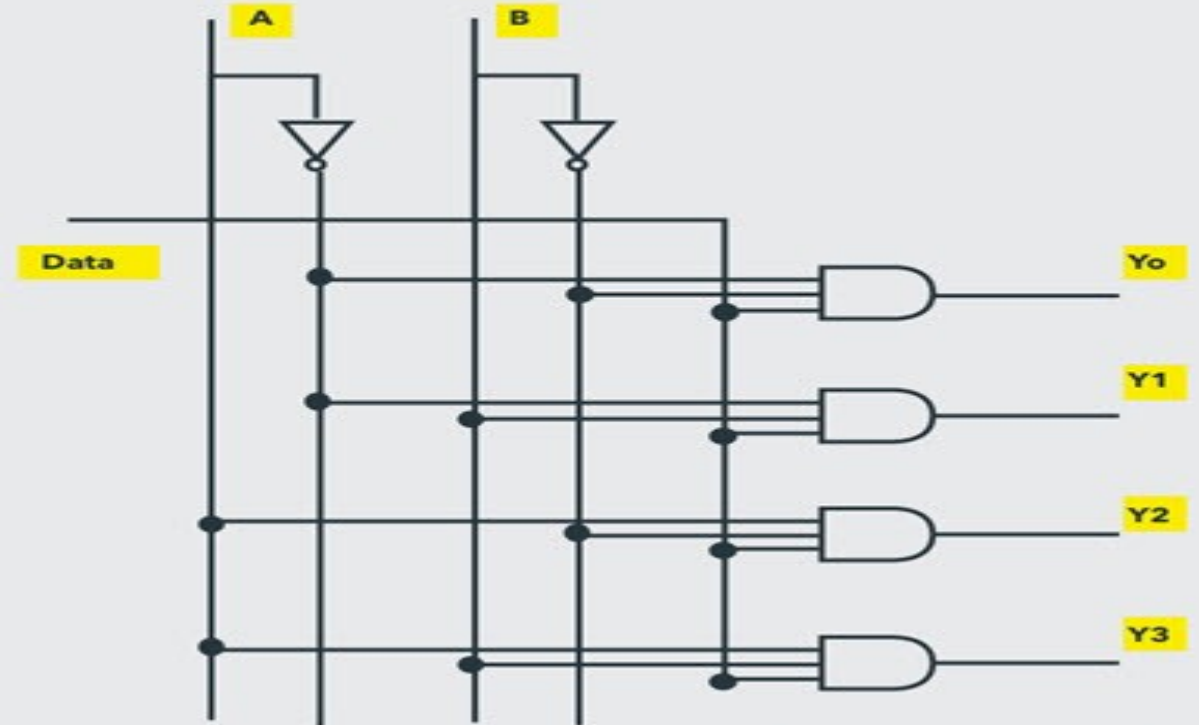


Demultiplexer

Block Diagram Of A Demultiplexer



Logic Circuit Of 1-To-4 Demultiplexer



End of
Chapter V