الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

كـلـيـــة هندسة الحاسوب والمعلوماتية والاتصالات
Faculty of Computer & Informatics
and Communications Engineering

# Logic Circuits

*Dr. Eng.*

*Hassan M. Ahmad*

Hassan.Ahmad@spu.edu.sy, istamo48@mail.ru

# *Chapter_ 6*

# Functions of Combinational Logic

## *Lecture _9*

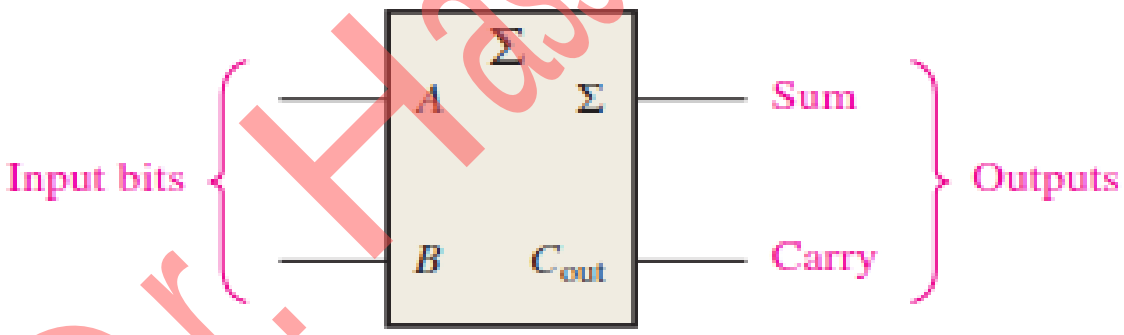
## *Adders , Comparators, Decoders, Encoders & Multiplexers*

# 9-1. Half and Full Adders

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$

**Half-Adder** (الجامع النصفي = نصف الجامع)

- Recall the basic rules for binary addition are:
- The operations are performed by a logic circuit called a **half-adder**.
- The half-adder accepts two binary digits on its inputs and produces two binary digits on its outputs – a sum bit and a carry bit.
- A half-adder is represented by the logic symbol in Fig.



## Half-adder logic

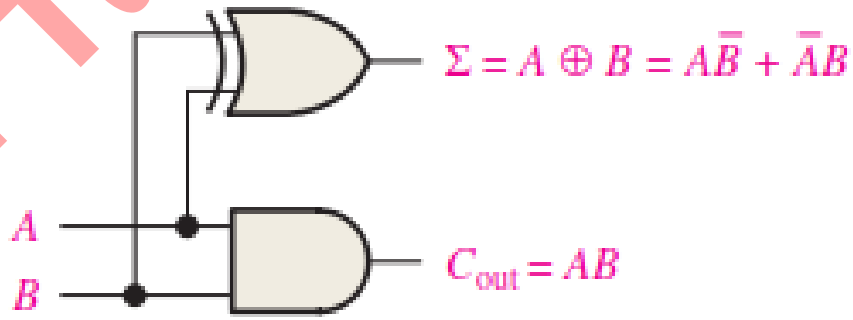
- The inputs and outputs can be summarized on a truth table.
  - The **output carry** ($C_{out}$) can be expressed as the **AND** of the input variables: $C_{out} = AB$
  - The **sum output** ($\Sigma$) can be expressed as the **exclusive-OR** of the input variables: $\sum = A \oplus B$

Half-adder truth table.

| $A$ | $B$ | $C_{out}$ | $\Sigma$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

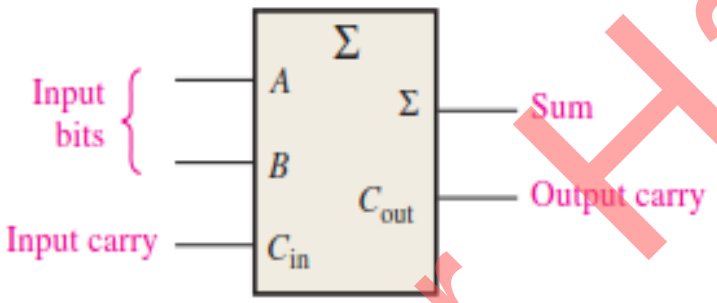
$\Sigma$ = sum
$C_{out}$ = output carry
$A$ and $B$ = input variables (operands)

- The logic implementation required for the half-adder function can be developed, as shown in Fig.



# Full-Adder (الجامع الكلي = الجامع التام)

- The second category of adder is the **full-adder**.
- The full-adder accepts two input 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 logic symbol for a full-adder is shown in Fig., and the truth table in Table the operation of a full-adder.



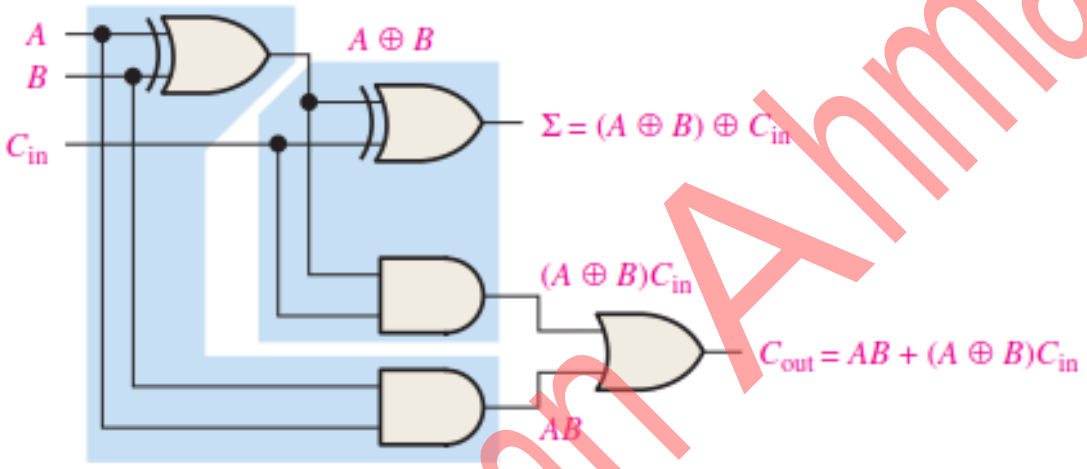

Full-adder truth table.

| $A$ | $B$ | $C_{in}$ | $C_{out}$ | $\Sigma$ |
|---|---|---|---|---|
| 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 |

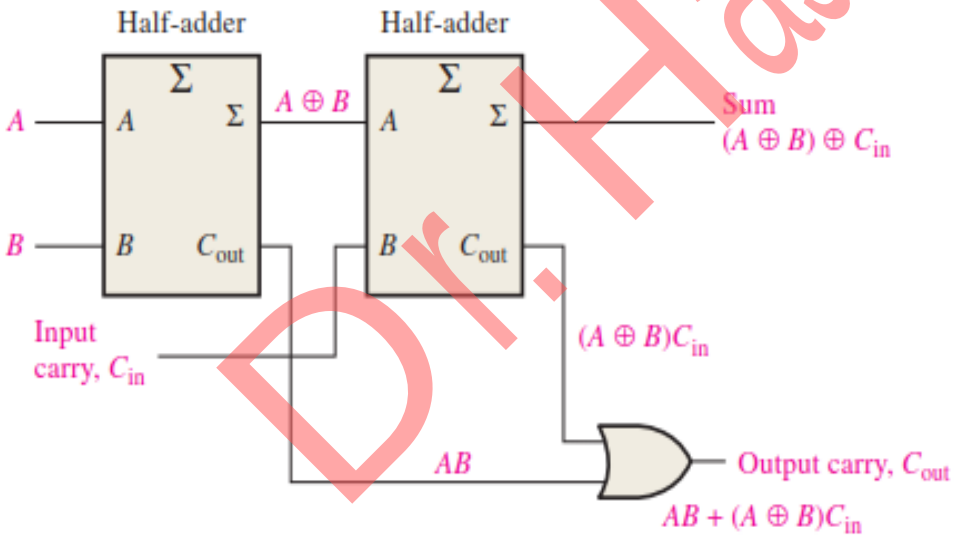
$C_{in}$ = input carry, sometimes designated as $CI$
$C_{out}$ = output carry, sometimes designated as $CO$
$\Sigma$ = sum
$A$ and $B$ = input variables (operands)

# Full-Adder Logic

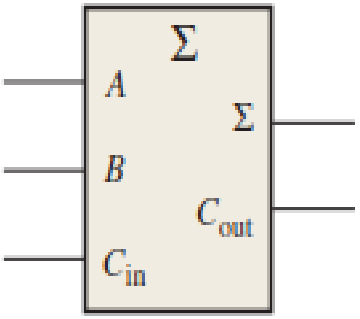
- The complete logic circuit for a full-adder can be constructed from **two half adders** (each half-adder is enclosed by a shaded area) as shown in Fig.




**Full-adder Logic block diagram.**

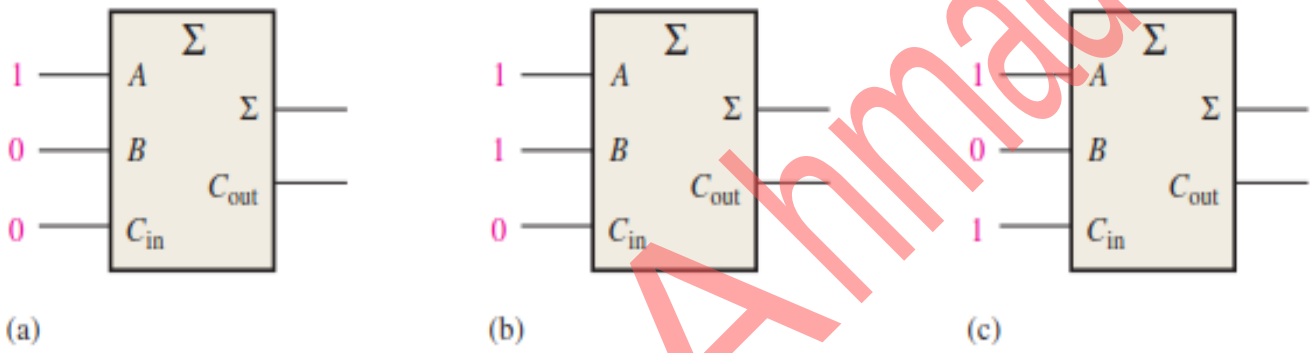



**Full-adder logic symbol**



# Example 9-1

For each of the three full-adders in Fig., determine the outputs for the inputs shown.




# Solution

a) The input bits are $A = 1$, $B = 0$, and $C_{in} = 0$. ➔ $1 + 0 + 0 = 1$ with no carry.

Therefore, **Σ = 1** and $\boldsymbol{C_{out} = 0}$.

b) The input bits are $A = 1$, $B = 1$, and $C_{in} = 0$. ➔ $1 + 1 + 0 = 0$ with a carry of 1.

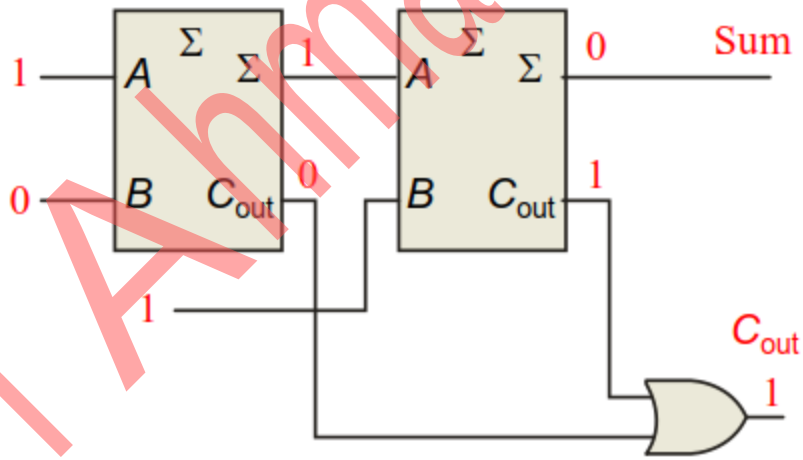
Therefore, **Σ = 0** and $\boldsymbol{C_{out} = 1}$.

c) The input bits are $A = 1$, $B = 0$, and $C_{in} = 1$. ➔ $1 + 0 + 1 = 0$ with a carry of 1.

Therefore, **Σ = 0** and $\boldsymbol{C_{out} = 1}$.

# Example 9-2

For the given inputs, determine the intermediate and final outputs of the full adder.




# Solution

- The first half-adder has inputs of 1 and 0; therefore the $\Sigma = 1$ and the $C_{out} = 0$.
- The second half-adder has inputs of 1 and 1; therefore the $\Sigma = 0$ and the $C_{out} = 1$.
- The OR gate has inputs of 1 and 0, therefore the final $C_{out} = 1$.

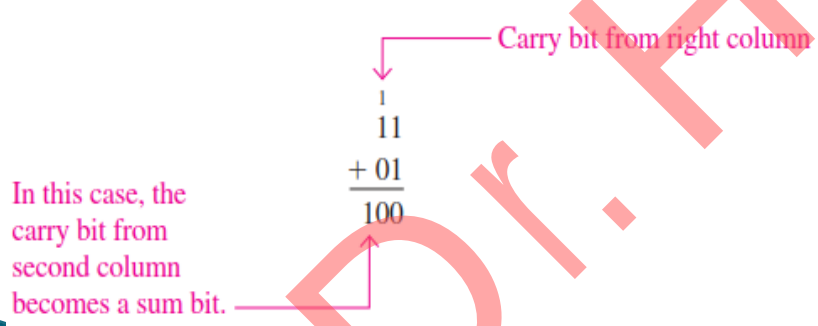
Notice that the result can be read directly on the truth table for a full adder.

| Inputs | | | Outputs | |
|---|---|---|---|---|
| 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 |

# Parallel Binary Adders

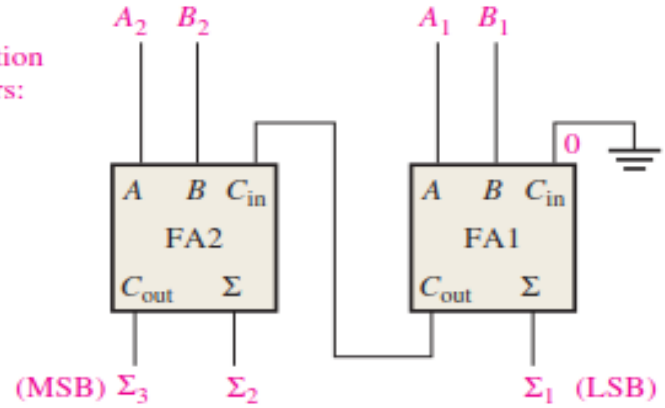
## Two binary numbers

- To add two binary numbers, a full-adder (FA) 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, as shown in Figure for a 2-bit adder.



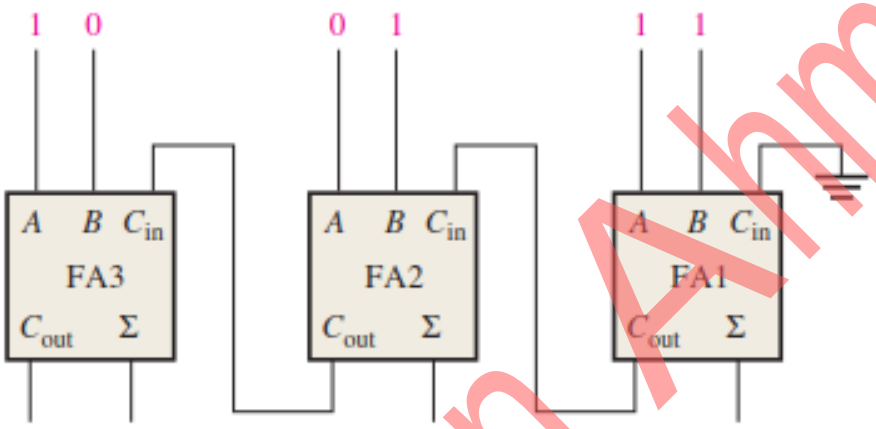

General format, addition of two 2-bit numbers:

$$\begin{array}{r} A_2A_1 \\ +\ B_2B_1 \\ \hline \Sigma_3\Sigma_2\Sigma_1 \end{array}$$



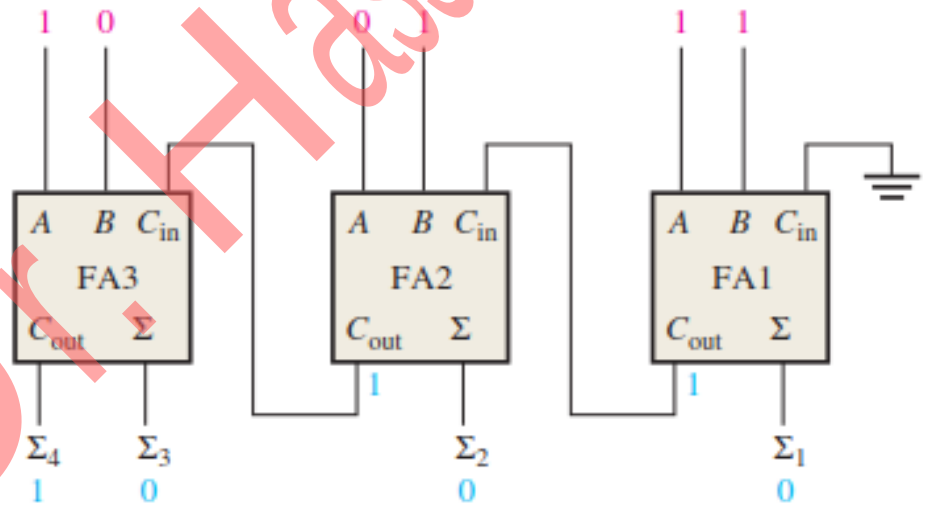
# Example 9-3

Determine the sum generated by the **3-bit** parallel adder in Figure and show the intermediate carries when the binary numbers 101 and 011 are being added.



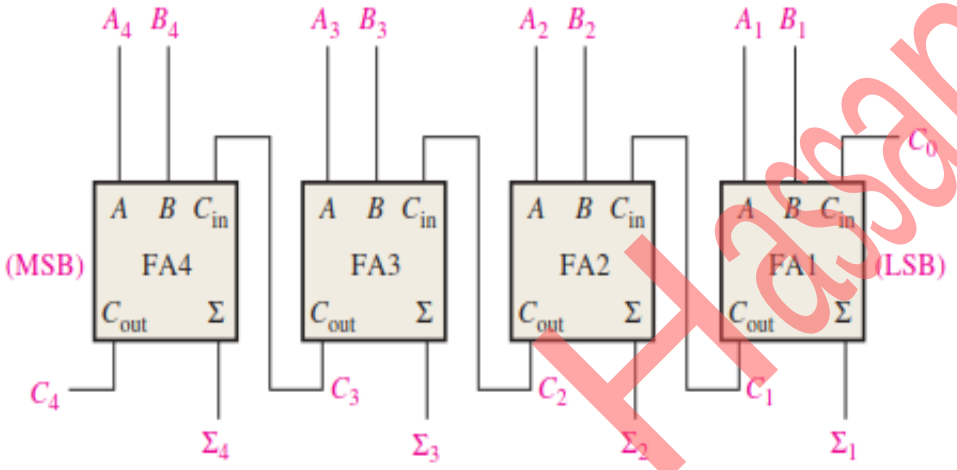

# Solution

The sum bits and the intermediate carries are indicated in blue in Figure.

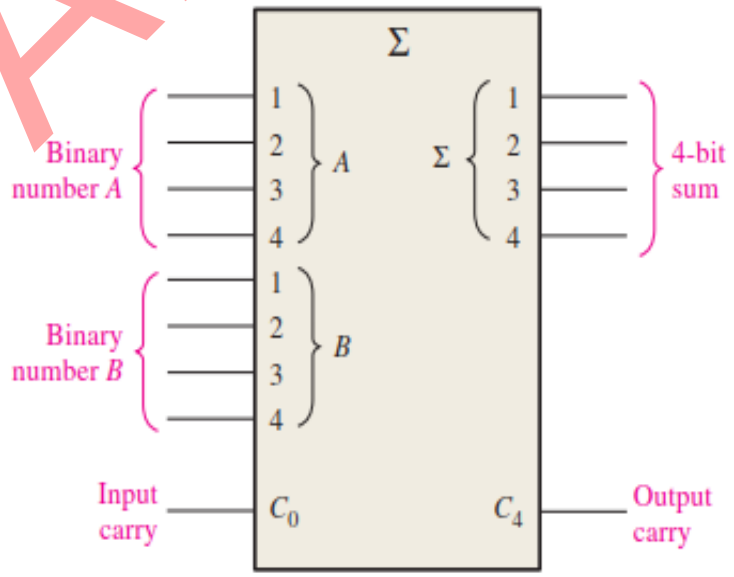


## Four-Bit Parallel Adders

- A group of four bits is called a **nibble**.
- A basic 4-bit parallel adder is implemented with four full-adder stages as shown in Figure.




(a) Block diagram



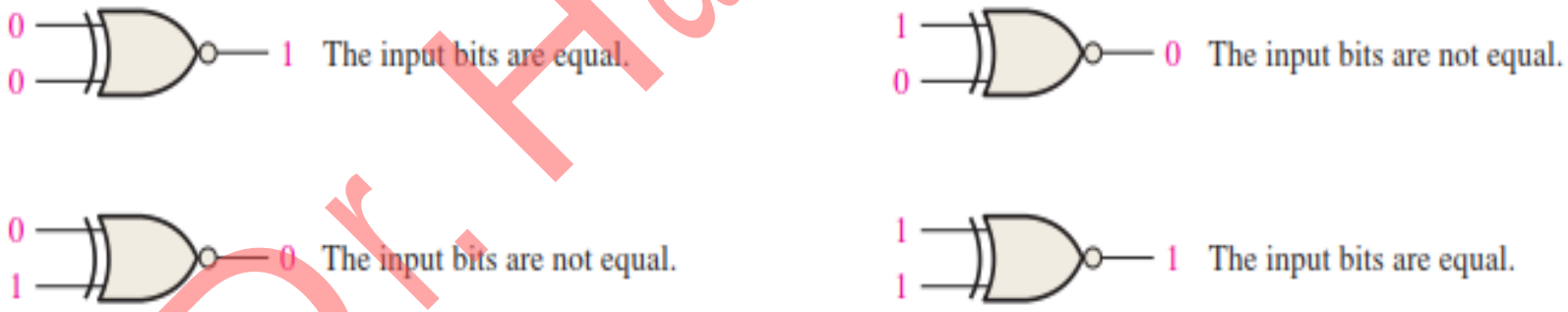

(b) Logic symbol

# 9-2. Comparators (المقارنات)

## Equality (المساواة)

- The basic function of a **comparator** is to compare the magnitudes of two binary quantities to determine the relationship of those quantities.
- In its simplest form, a comparator circuit determines whether two numbers are **equal**, i.e. a comparator can test for equality using **exclusive-NOR** gate because its output is a **0** if the two input bits are not equal and a **1** if the input bits are equal.

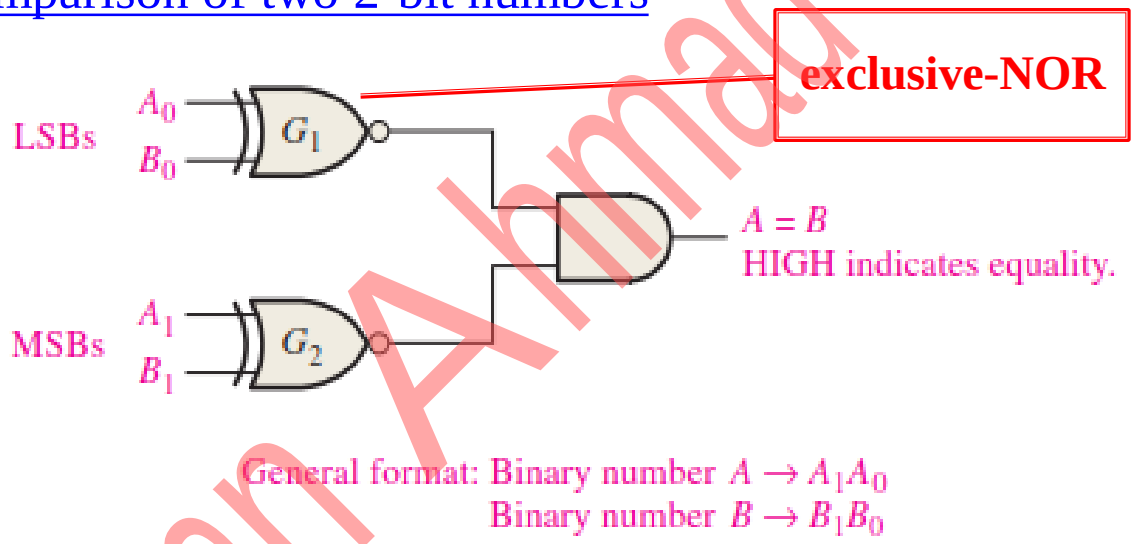
Basic 2-bit comparator operation



# Logic diagram for equality comparison of two 2-bit numbers

Least Significant Bit
(البت ذات القيمة الصغرى، وهوالبت الموجود في أقصى يمين الرقم )

Most Significant Bit
(البت ذات القيمة العظمى، وهوالبت الموجود في أقصى يسار الرقم )



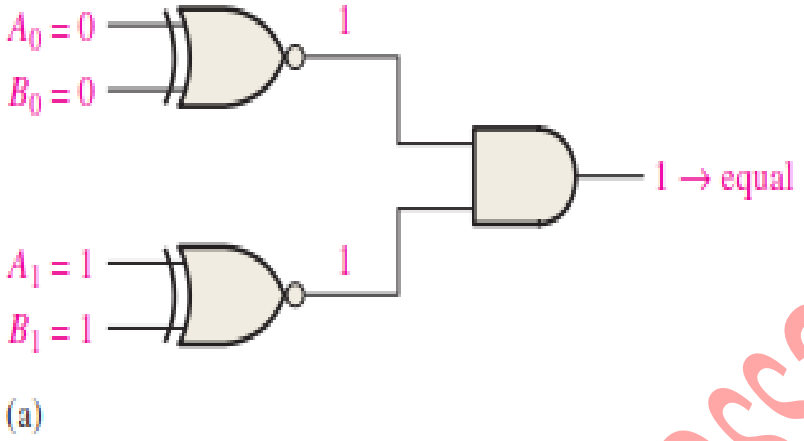

The two least significant bits (LSBs) of the two numbers are compared by gate $G_1$, and the two most significant bits (MSBs) are compared by gate $G_2$, as shown in Fig.

- If the two numbers are equal, their corresponding bits are the same, and the output of each exclusive-NOR gate is a **1**.
- If the corresponding sets of bits are not equal, a **0** occurs on that exclusive-NOR gate output.

# Example 9-4

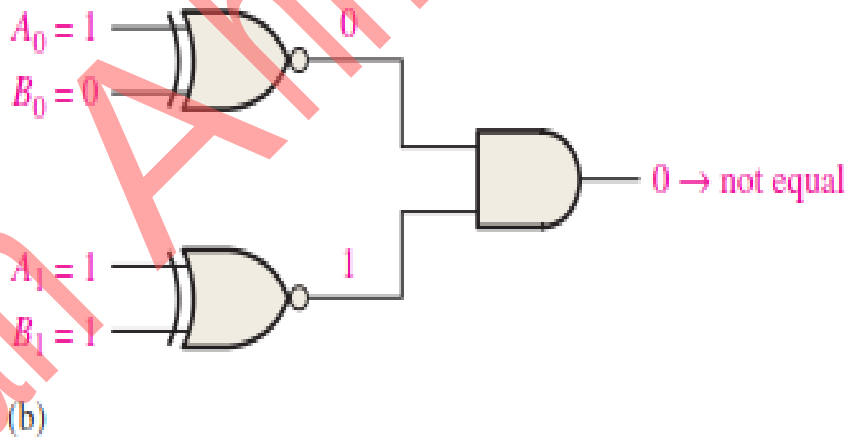
Apply each of the following sets of binary numbers to the comparator inputs in Fig., and determine the output by following the logic levels through the circuit.

(a) 10 and 10




(a)

(b) 11 and 10



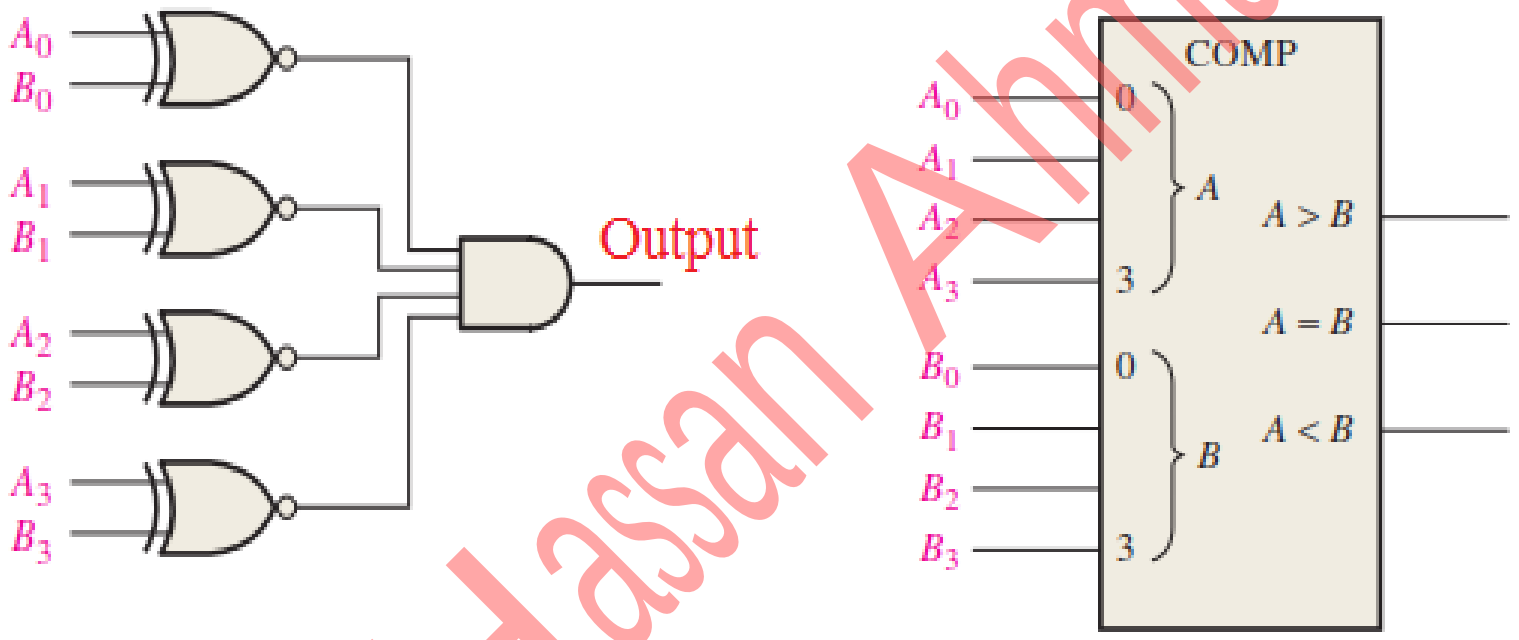

(b)

# Solution

a) The output is 1 for inputs 10 and 10, as shown in Fig.(a).

b) The output is 0 for inputs 11 and 10, as shown in Fig.(b).

- The **logic circuit (and symbol) for a 4-bit comparator** in Fig.



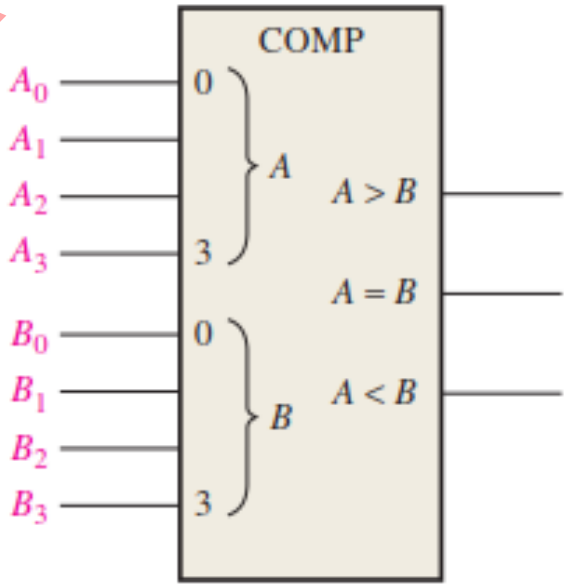
## Inequality (اللامساوة/التباين)

- When number $A$ is greater than number $B$ ($A > B$)
- When number $A$ is less than number $B$ ($A < B$)
- **For example**, the **for a 4-bit comparator**.

To determine an inequality of binary numbers $A$ and $B$, you first examine the highest order bit in each number.

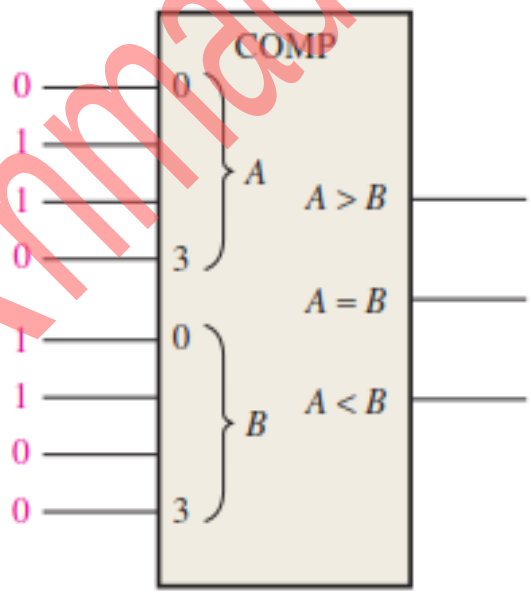
**The following conditions are possible**:

1. If $A_3 = 1$ and $B_3 = 0$, number $A$ is greater than number $B$.
2. If $A_3 = 0$ and $B_3 = 1$, number $A$ is less than number $B$.
3. If $A_3 = B_3$, then you must examine the next **lower** bit position for an inequality.



# Example 9-5

Determine the $A = B$, $A > B$, and $A < B$ outputs for the input numbers shown on the comparator in Fig.




# Solution

From Fig., we found that $A_3 = B_3 = 0$,

then we must examine the next lower bit position for an inequality, i.e. $A_2$ with $B_2$ .

We find that $A_2 = 1$ and $B_2 = 0$ ➔ $A > B$,

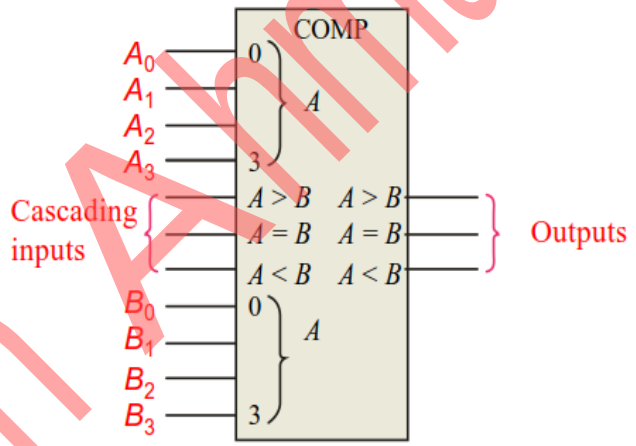
That is, the number on the $A$ inputs is 0110 and the number on the $B$ inputs is 0011.

The $A > B$ ➔ output is HIGH and the other outputs are LOW.

# Cascaded Arrangement of Comparators (الترتيب المتعاقب للمقارنات)

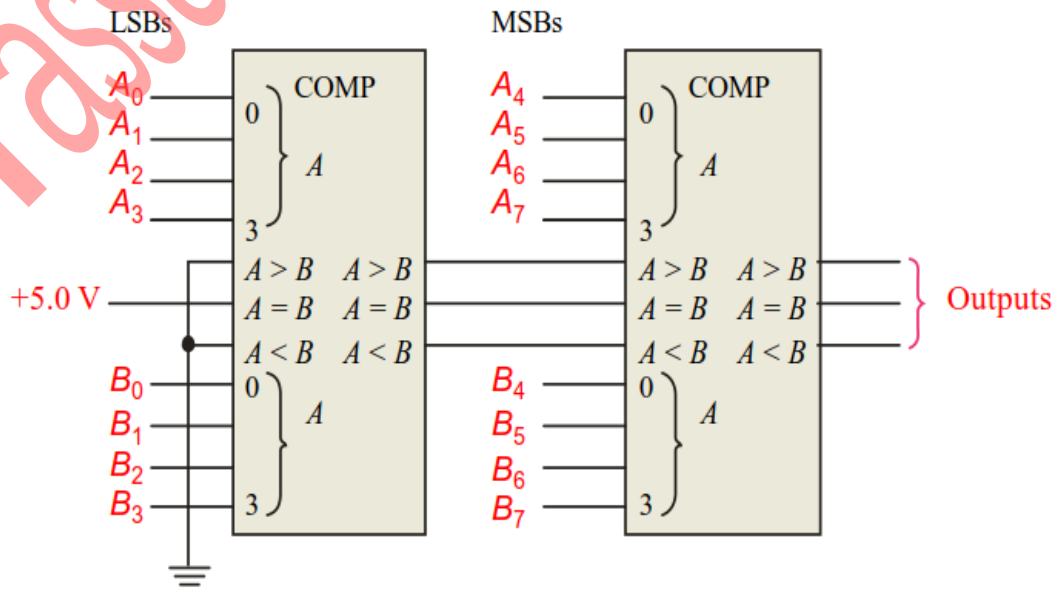
- Cascading inputs are provided to expand the comparator to larger numbers.

4-bit magnitude comparator



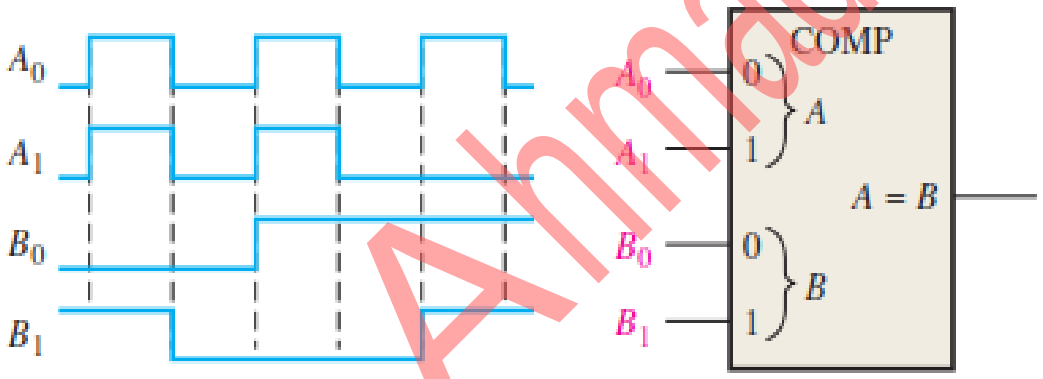

8-bit magnitude comparator using two 4-bit magnitude comparators

The lowest order comparator has a HIGH (+5.0 V) on the $A = B$ input.



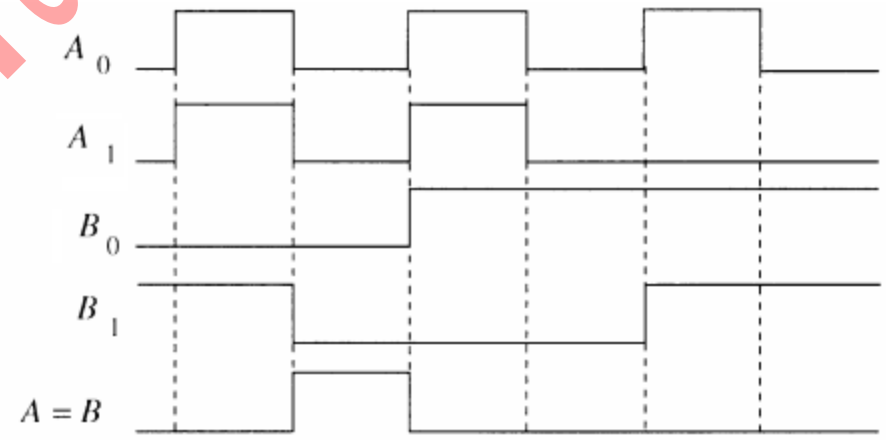
# Example 9-6

The waveforms in Fig. are applied to the comparator as shown. Determine the output ($A = B$) waveform.



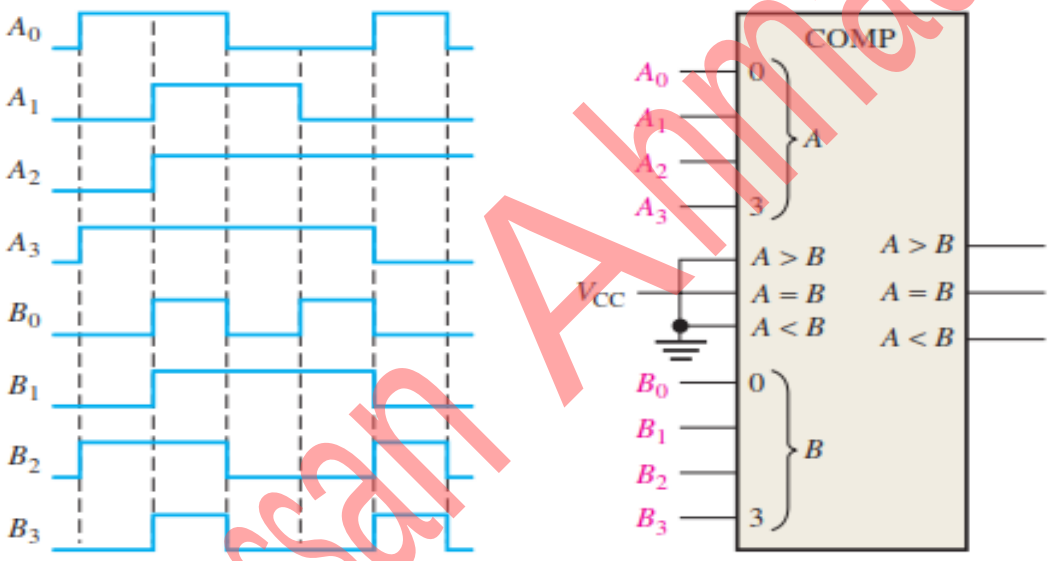

# Solution

- If $A_1 = B_1$, then you must examine (in the same time) the next lower bit position for an inequality.
- After comparing we find that: the $A = B$ output is HIGH when $A_0 = B_0$ and $A_1 = B_1$ in the same time.



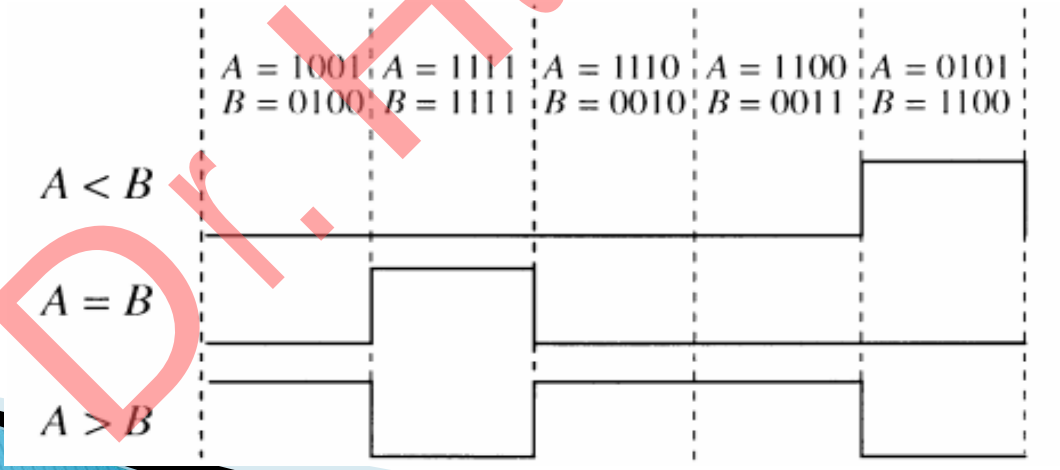
# Example 9-7

For the 4-bit comparator in Fig., plot each output waveform for the inputs shown. The outputs are active-HIGH. ($V_{CC}$ = +5 V)



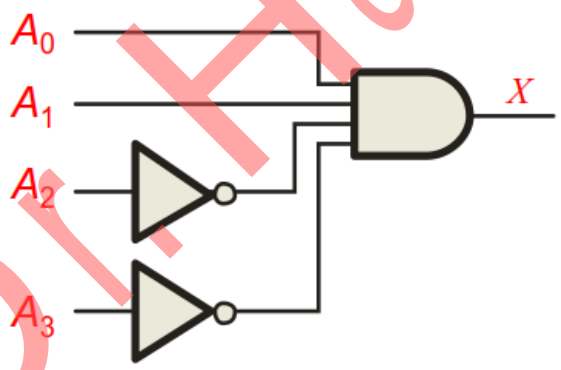

# Solution

Following the conditions for inequality, the result is

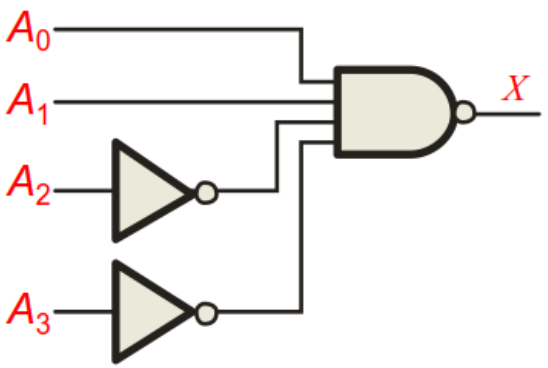


# 9-3. Decoders (مفككات الترميز)

- A **decoder** is a 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* input** lines to handle *n* bits and from **one** to **$2^n$ output** lines to indicate the presence of one or more *n*-bit combinations.
- **For example**, two simple decoders that detect the presence of the binary code 0011 are shown. The first has an active HIGH output; the second has an active LOW output.



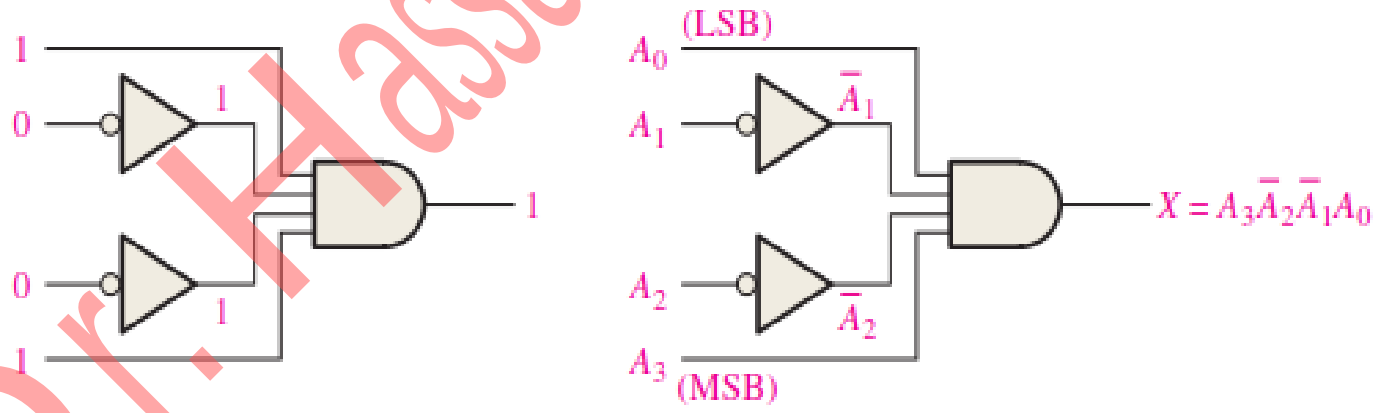
Active HIGH decoder for 0011



Active LOW decoder for 0011

# 1. The Basic Binary Decoder

- Suppose you need to determine when a binary **1001** occurs on the inputs of a digital circuit.
- An **AND gate** can be used as the basic decoding element because it produces a **HIGH output** only when **all of its inputs are HIGH**.
  - Therefore, you must make sure that all of the inputs to the AND gate are HIGH when the binary number 1001 occurs; this can be done by inverting the two middle bits (the 0s).
  - The logic equation for the resulting decoder is developed as illustrated in Figure.



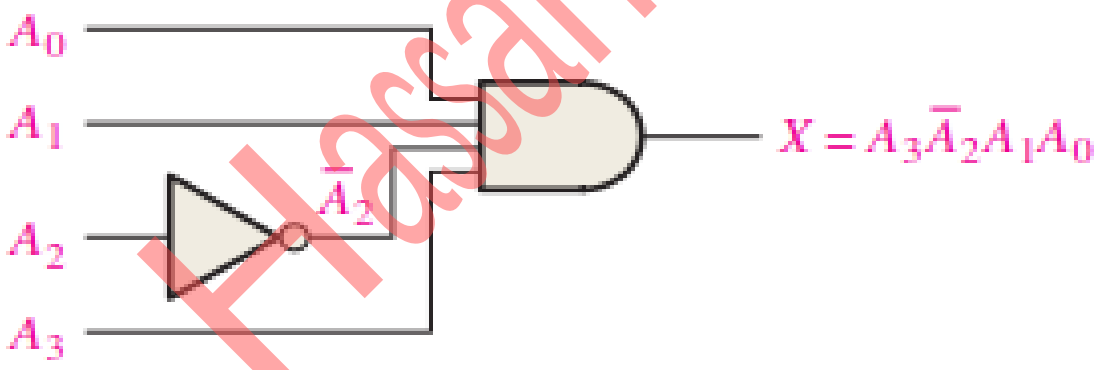
# Example 9-8

Determine the logic required to decode the binary number **1011** by producing a HIGH level on the output.

# Solution

The decoding function can be formed by complementing only the variables that appear as 0 in the desired binary number, as follows:

$$X = A_3\overline{A}_2A_1A_0 \quad (1\,0\,1\,1)$$



## 2. The 4-Bit Decoder

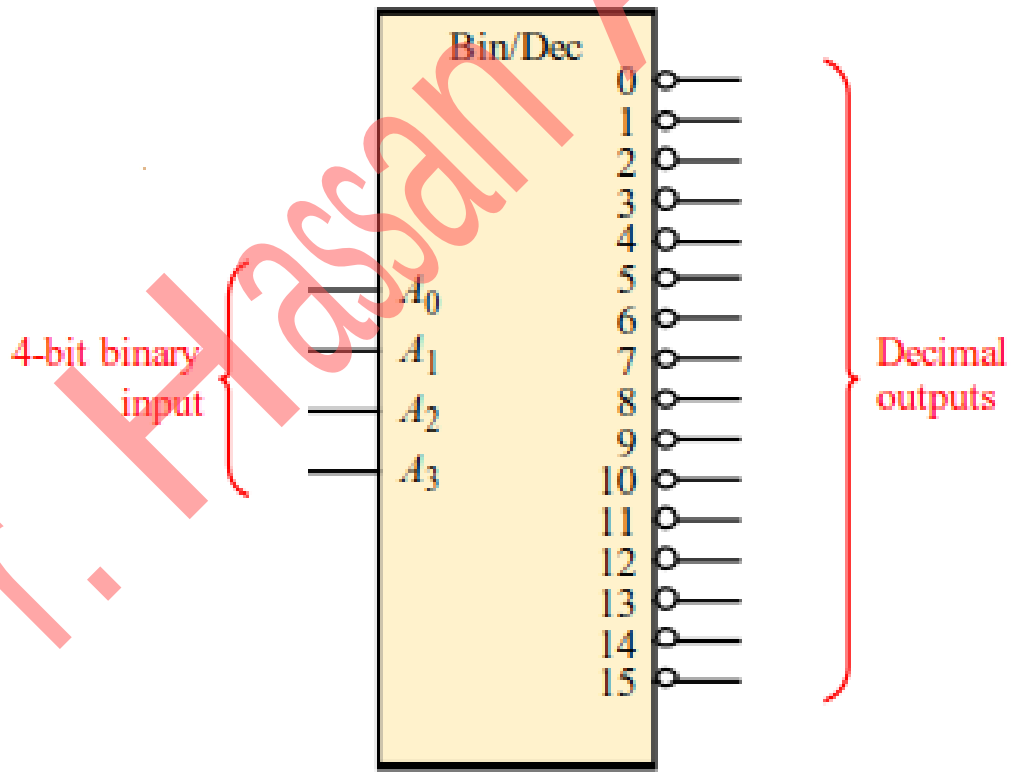
- In order to decode all possible combinations of four bits, sixteen decoding gates are required ($2^4 = 16$).
- This type of decoder is commonly called either a **4-line-to-16-line** decoder because there are four inputs and sixteen outputs or a **1-of-16 decoder** because for any given code on the inputs, **one** of the sixteen outputs is activated (**active-LOW** output) (منخفض الفعالية), and all others are **active-HIGH** (مرتفع الفعالية).
- For example, the combination 0000 on the inputs, the output terminal **0** is activated; for combination 0001, the output terminal **1** is activated;....etc.

Decoding functions and truth table for a 4-line-to-16-line (1-of-16) decoder with **active-LOW outputs**.

| Decimal Digit | Binary Inputs $A_3$ | $A_2$ | $A_1$ | $A_0$ | Decoding Function | Outputs (Active LOW) 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | $\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$ | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | $\overline{A}_3\overline{A}_2\overline{A}_1A_0$ | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | $\overline{A}_3\overline{A}_2A_1\overline{A}_0$ | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 3 | 0 | 0 | 1 | 1 | $\overline{A}_3\overline{A}_2A_1A_0$ | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | $\overline{A}_3A_2\overline{A}_1\overline{A}_0$ | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | $\overline{A}_3A_2\overline{A}_1A_0$ | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | $\overline{A}_3A_2A_1\overline{A}_0$ | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | $\overline{A}_3A_2A_1A_0$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | $A_3\overline{A}_2\overline{A}_1\overline{A}_0$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | $A_3\overline{A}_2\overline{A}_1A_0$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | $A_3\overline{A}_2A_1\overline{A}_0$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 11 | 1 | 0 | 1 | 1 | $A_3\overline{A}_2A_1A_0$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | $A_3A_2\overline{A}_1\overline{A}_0$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 13 | 1 | 1 | 0 | 1 | $A_3A_2\overline{A}_1A_0$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | $A_3A_2A_1\overline{A}_0$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 15 | 1 | 1 | 1 | 1 | $A_3A_2A_1A_0$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

# Logic Symbol for a 4-line-to-16-line (1-of-16) Decoder

- A **logic symbol** for a 4-line-to-16-line (1-of-16) decoder with **active-LOW outputs** is shown in Fig.
  - The **BIN/DEC** label indicates that a **binary** input makes the corresponding **decimal** output active.



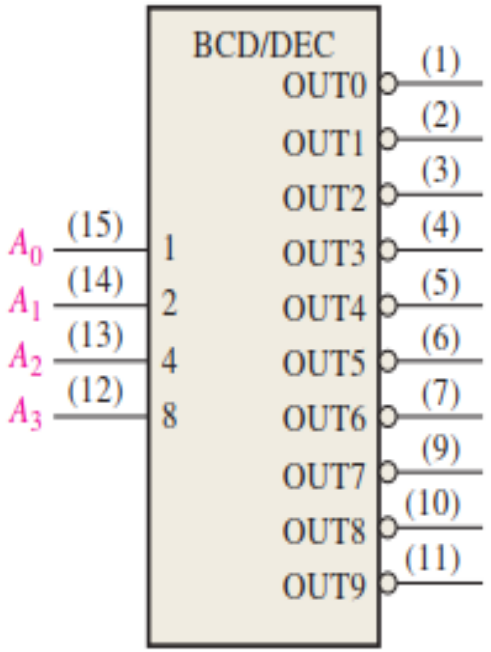
## 3. The BCD-to-Decimal Decoder

- The BCD-to-decimal decoder converts each BCD code (8421 code) into one of ten possible decimal digit indications.
- It is frequently referred as a **4-line-to-10-line** decoder or a **1-of-10 decoder**.
- The method of implementation is the same as for the 1-of-16 decoder previously discussed, except that **only ten decoding gates** are required because the BCD code represents only the ten decimal digits 0 through 9.

# BCD decoding functions

| Decimal Digit | BCD Code $A_3$ | $A_2$ | $A_1$ | $A_0$ | Decoding Function |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | $\overline{A}_3\overline{A}_2\overline{A}_1\overline{A}_0$ |
| 1 | 0 | 0 | 0 | 1 | $\overline{A}_3\overline{A}_2\overline{A}_1A_0$ |
| 2 | 0 | 0 | 1 | 0 | $\overline{A}_3\overline{A}_2A_1\overline{A}_0$ |
| 3 | 0 | 0 | 1 | 1 | $\overline{A}_3\overline{A}_2A_1A_0$ |
| 4 | 0 | 1 | 0 | 0 | $\overline{A}_3A_2\overline{A}_1\overline{A}_0$ |
| 5 | 0 | 1 | 0 | 1 | $\overline{A}_3A_2\overline{A}_1A_0$ |
| 6 | 0 | 1 | 1 | 0 | $\overline{A}_3A_2A_1\overline{A}_0$ |
| 7 | 0 | 1 | 1 | 1 | $\overline{A}_3A_2A_1A_0$ |
| 8 | 1 | 0 | 0 | 0 | $A_3\overline{A}_2\overline{A}_1\overline{A}_0$ |
| 9 | 1 | 0 | 0 | 1 | $A_3\overline{A}_2\overline{A}_1A_0$ |

A **logic symbol** for a 4-line-to-10-line (1-of-10) decoder with **active-LOW outputs** is shown in Fig.



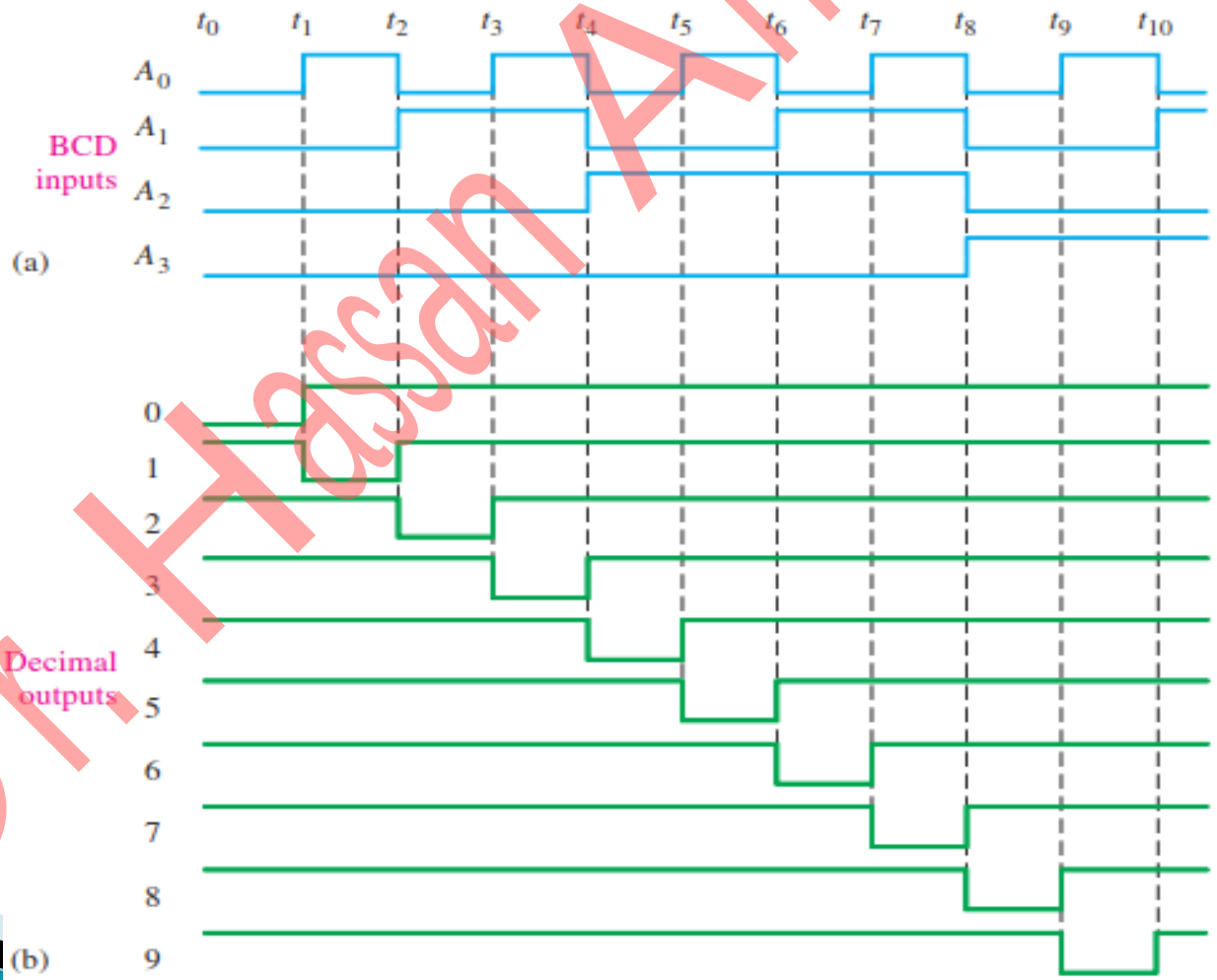
# Example 9-9

If the input waveforms in Fig.(a) are applied to the inputs of the BCD-to-decimal decoder , show the output waveforms.

# Solution

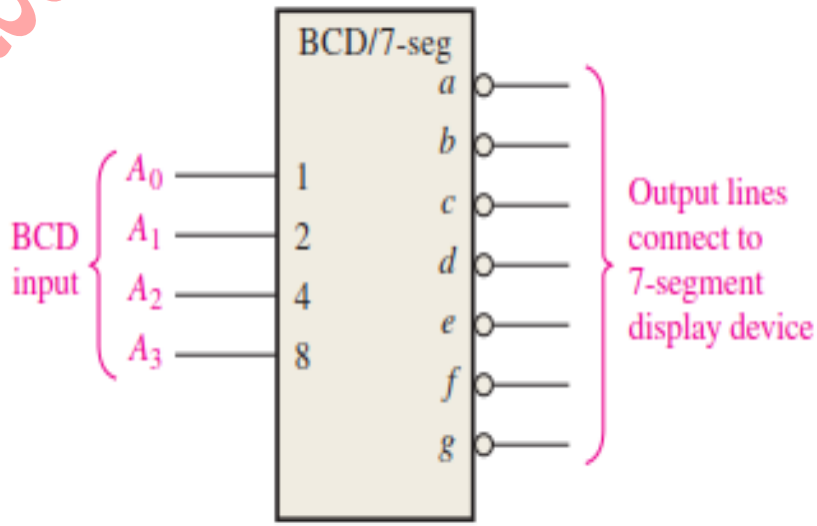
The output waveforms are shown in Fig.(b).

From **Table on slide 24** we note that $t_0 \rightarrow t_1$ =OUT 0; $t_1 \rightarrow t_2$ =OUT 1; …; $t_9 \rightarrow t_{10}$ =OUT 9.



# 4. The BCD-to-7-Segment Decoder

- The BCD-to-7-segment decoder accepts the BCD code on its inputs and provides outputs to **drive 7-segment display** devices to produce a decimal readout.
- This type of decoders is called a **BCD-to-7-segment decoder/driver with active-LOW** outputs.
- The logic diagram for a basic BCD-to-7-segment decoder/driver with active-LOW outputs. is shown in Fig.



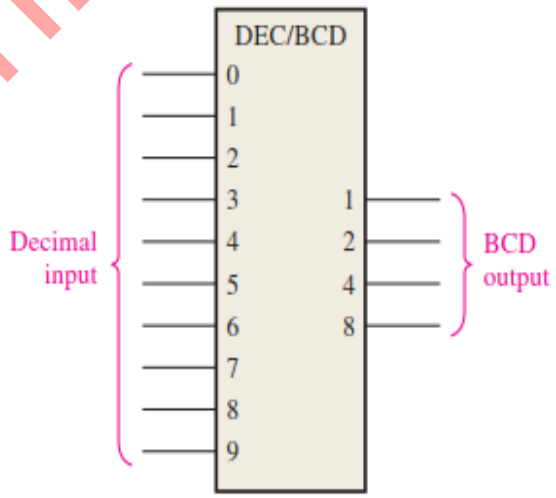
# 9-4. Encoders (المشفرات)

- An encoder is a combinational logic circuit that essentially performs a "reverse" decoder function.
- 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 format is called **encoding.**

# The Decimal-to-BCD Encoder

- This type of encoder has ten inputs—one for each decimal digit—and four outputs corresponding to the BCD code, as shown in Fig.
- This is a basic **10-line-to-4-line encoder** (Logic symbol).
- The BCD (8421) code is listed in Table.

| 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 |




- From this table we can determine the relationship between each BCD bit and the decimal digits in order to analyze the logic.
  - Bit of the BCD code, $A_3$, is always a 1 for decimal digit 8 or 9. An **OR** expression for bit $A_3$ in terms of the decimal digits can therefore be written as: $A_3 = 8+9$;
  - Also for others:

$A_2 = 4+5+6+7$; $A_1 = 2+3+6+7$; $A_0 = 1+3+5+7+9$

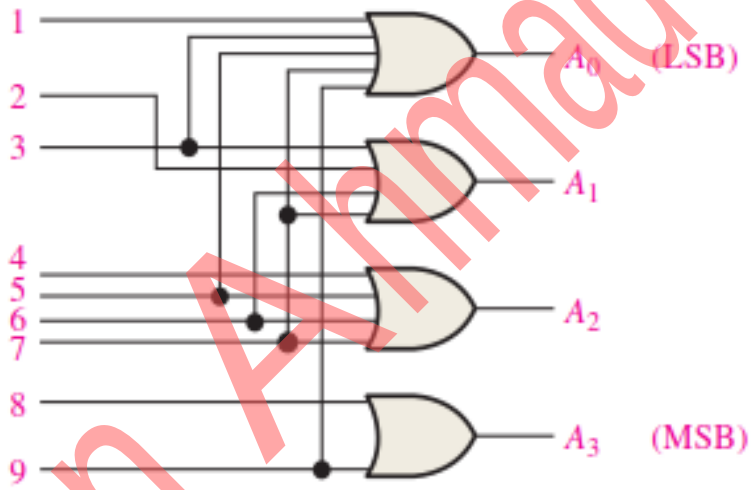
- From these expressions we have the **basic logic diagram of a decimal-to-BCD encoder** as shown in Fig.

$$A_3 = 8 + 9$$

$$A_2 = 4 + 5 + 6 + 7$$

$$A_1 = 2 + 3 + 6 + 7$$

$$A_0 = 1 + 3 + 5 + 7 + 9$$

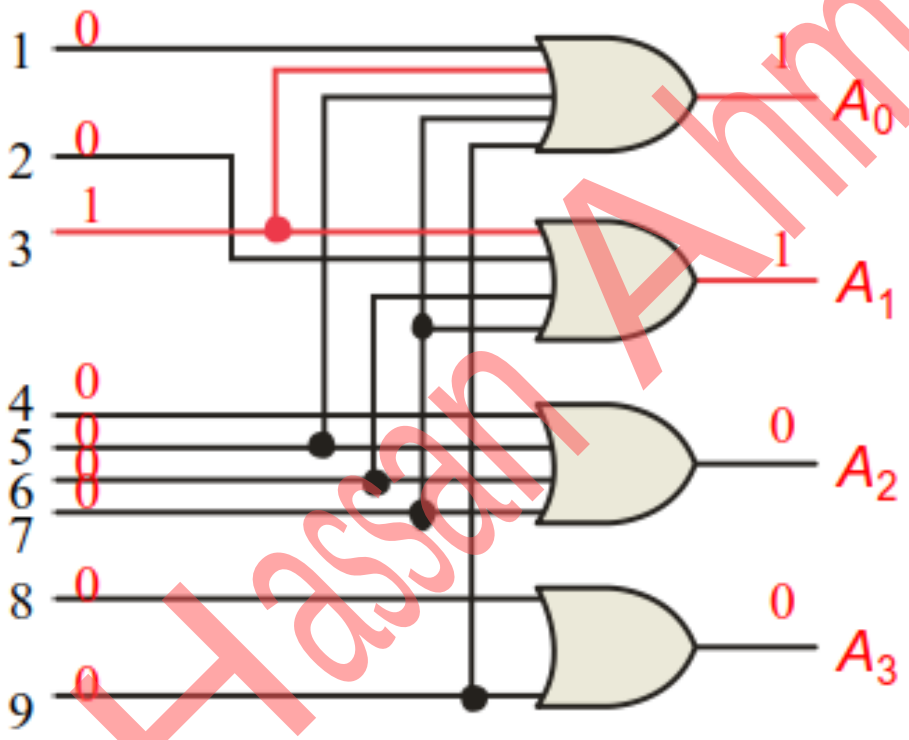



- A **0-digit** input is not needed because the BCD outputs are all LOW when there are no HIGH inputs.
- The **basic operation** of the circuit in Fig. is as follows:
  - When a HIGH appears on one of the decimal digit input lines, the appropriate levels (المستويات المناسبة) occur on the four BCD output lines.
    - **For example**, if input line 9 is HIGH (assuming all other input lines are LOW), this condition will produce a HIGH on outputs $A_0$ and $A_3$ and LOWs on outputs $A_1$ and $A_2$, which is the BCD code (1001) for decimal 9.

# Example 9-10

Show how the decimal-to-BCD encoder converts the decimal number 3 into a BCD 0011.

# Solution




The top two OR gates have ones as indicated with the red lines. Thus the output is 0011.

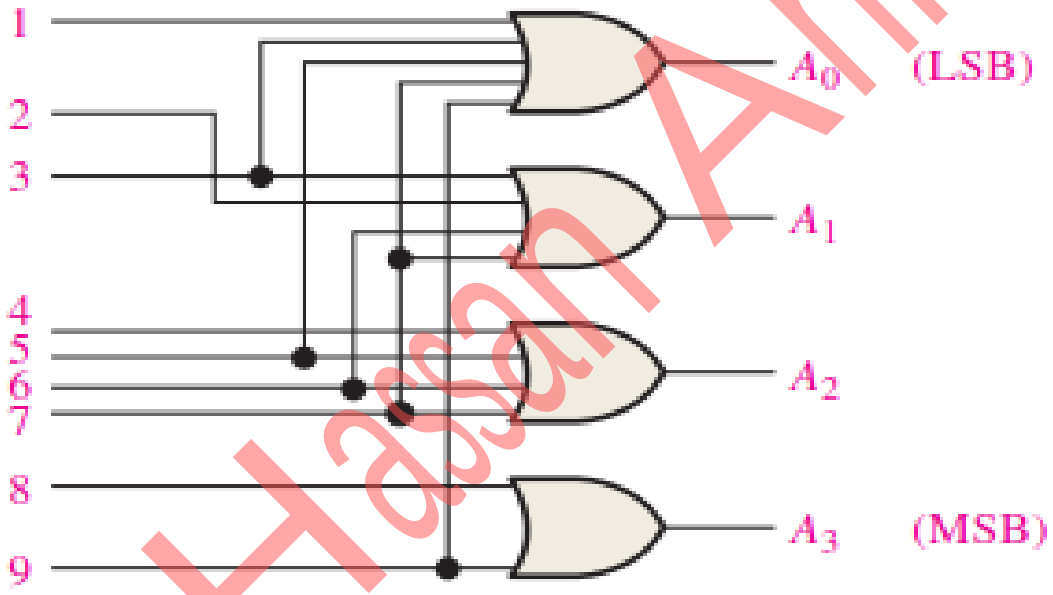
# Example 9-11

For the decimal-to-BCD encoder logic of Fig., assume that the 9 input and the 3 input are both HIGH.

a) What is the output code?

b) Is it a valid BCD (8421) code?

# Solution

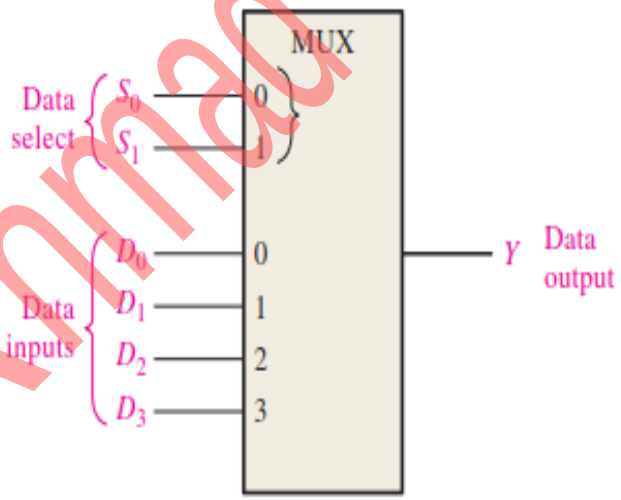



$A_0$, $A_1$ , and $A_3$ are HIGH. ➔ $A_3\ A_2\ A_1\ A_0$ = 1011, which is an invalid BCD code because the BCD 1011 is represents the decimal number 11.

# 9-5. Multiplexers (Data Selectors) (ناخب البيانات)

- 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 (وجهة=اتجاه).
  - The basic multiplexer has several data-input lines and a single output line.
  - It also has data-select inputs, which permit (يسمح) digital data on any one of the inputs to be switched to the output line.
- Multiplexers are also known as **data selectors**.

- A **logic symbol for a 4-input multiplexer (MUX)** is shown in Fig.
  - Notice that there are two data-select lines because with two select bits, any one of the four data-input lines can be selected.
  - In Fig., a 2-bit code on the data-select (S) inputs will allow the data on the selected data input to pass through to the data output.
  - The **operation of data selection** for a **1-of-4-multiplexer** is given in Table.




Logic symbol for a 1-of-4 data selector/multiplexer

| Data-Select Inputs | | |
|---|---|---|
| $S_1$ | $S_0$ | Input Selected |
| 0 | 0 | $D_0$ |
| 0 | 1 | $D_1$ |
| 1 | 0 | $D_2$ |
| 1 | 1 | $D_3$ |

## Multiplexing operation

- The data output is equal to the state of the selected data input.
- We can therefore, derive a logic expression for the output in terms of the data input and the select inputs, as shown in Table.

The data output is equal to $D_0$ only if $S_1 = 0$ and $S_0 = 0 : Y = D_0 \overline{S_1}\,\overline{S_0}$.

The data output is equal to $D_1$ only if $S_1 = 0$ and $S_0 = 1 : Y = D_1 \overline{S_1} S_0$.

The data output is equal to $D_2$ only if $S_1 = 1$ and $S_0 = 0 : Y = D_2 S_1 \overline{S_0}$.

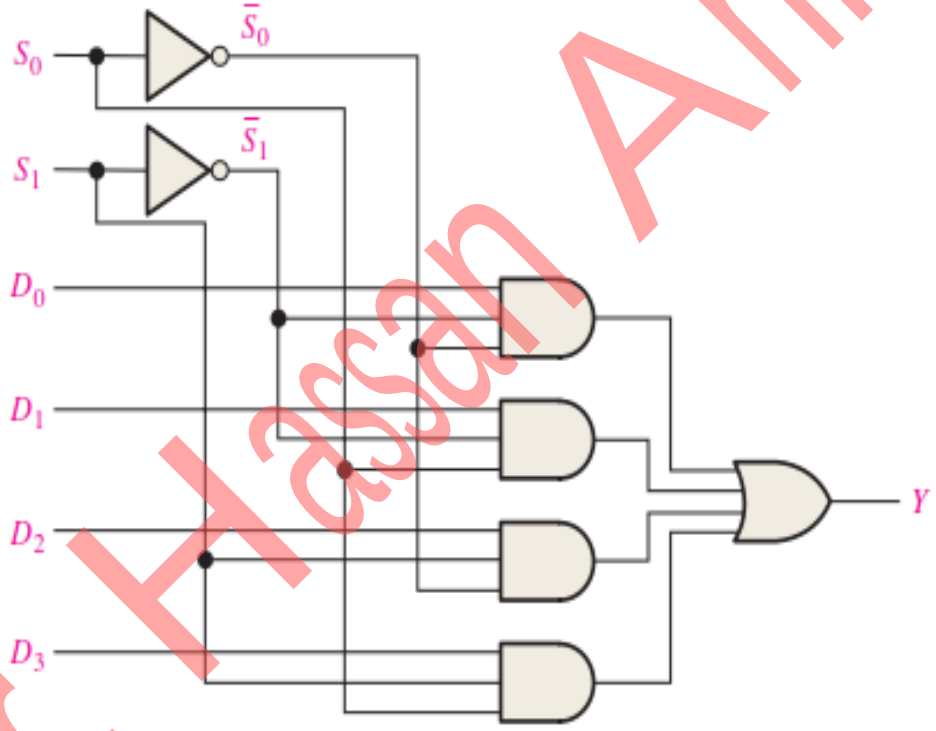
The data output is equal to $D_3$ only if $S_1 = 1$ and $S_0 = 1 : Y = D_3 S_1 S_0$.

- When these terms are **OR**ed, the total expression for the data output is

$$Y = D_0 \overline{S_1}\,\overline{S_0} + D_1 \overline{S_1} S_0 + D_2 S_1 \overline{S_0} + D_3 S_1 S_0$$

$$Y = D_0\overline{S_1}\,\overline{S_0} + D_1\overline{S_1}S_0 + D_2S_1\overline{S_0} + D_3S_1S_0$$

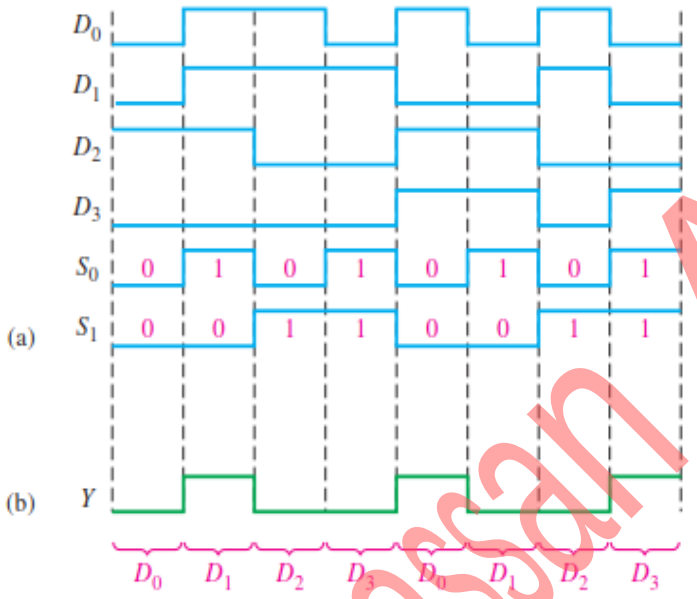
- The implementation of this equation requires **four** 3-input AND gates, a 4-input OR gate, and **two** inverters to generate the complements of $S_1$ and $S_0$, as shown in following Fig. (Logic diagram for a 4-input multiplexer)

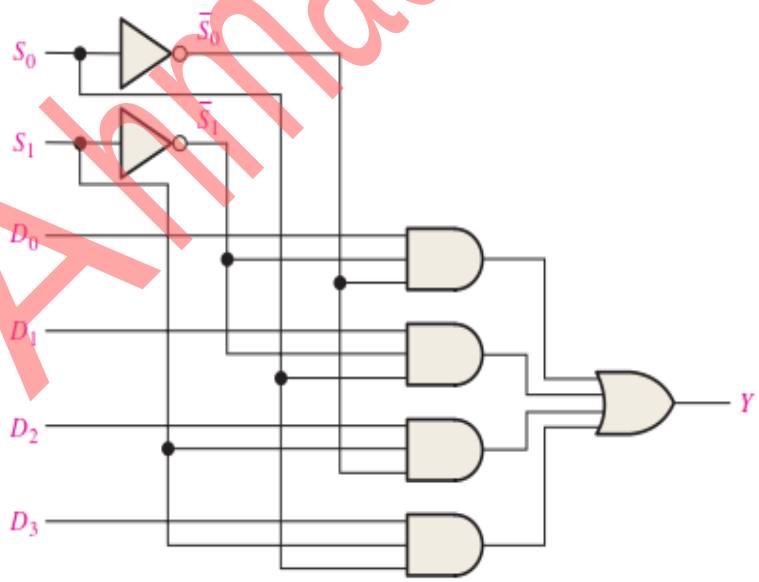



- Because data can be selected from any one of the input lines, this circuit is also referred to as a **data selector**.

# Example 9-12

The data-input and data-select waveforms in Fig.(a) are applied to the 4-input multiplexer. Determine the output waveform in relation to the inputs.




# Solution

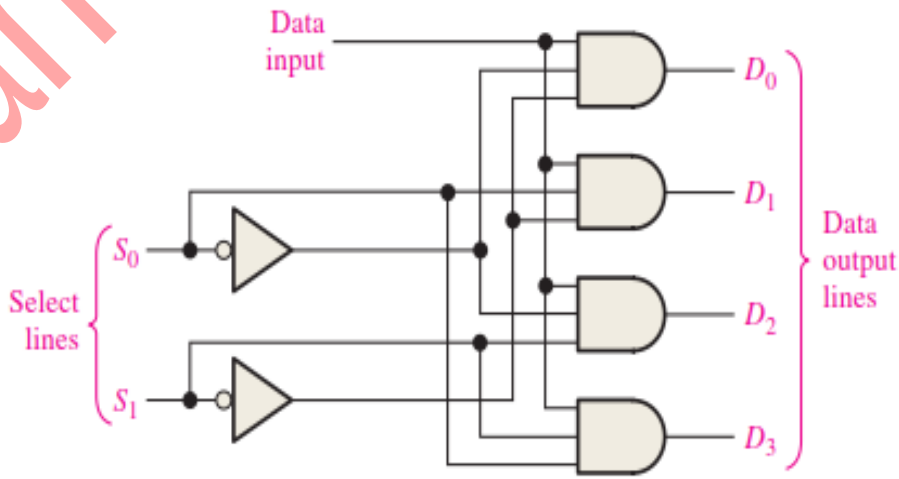
$$Y = D_0\overline{S_1}\,\overline{S_0} + D_1\overline{S_1}S_0 + D_2S_1\overline{S_0} + D_3S_1S_0$$

- The binary state of the data-select inputs during each interval determines which data input is selected.
- Notice that the data-select inputs go through a repetitive binary sequence 00, 01, 10, 11, 00, 01, 10, 11, and so on.
- Using the table of Multiplexing operation, the resulting output waveform is shown in Fig.(b).

# 9-6. Demultiplexers (الموزع)

- A **demultiplexer (DEMUX)** basically reverses the multiplexing function.
  - It takes digital information from one line and distributes it to a given number of output lines.
  - In other words, it switches data from one input line to two or more data lines depending on the select inputs.
  - For this reason, the demultiplexer is also known as a **data distributor**.

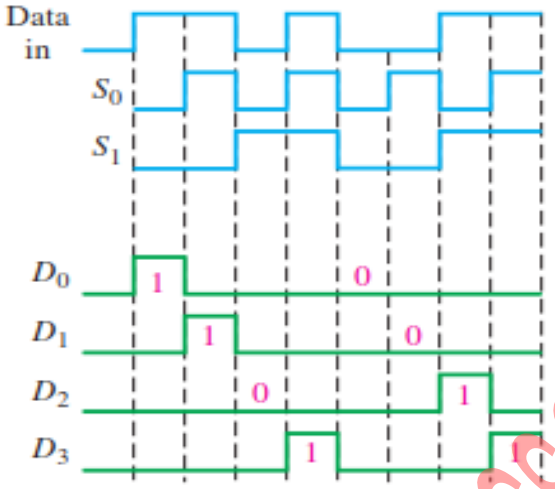
- Fig. shows a **1-line-to-4-line** demultiplexer (DEMUX) circuit.
- The data-input line goes to all of the AND gates.
- The two data-select lines enable only **one** gate at a time, and the data appearing on the data-input line will pass through the selected gate to the associated data-output line.

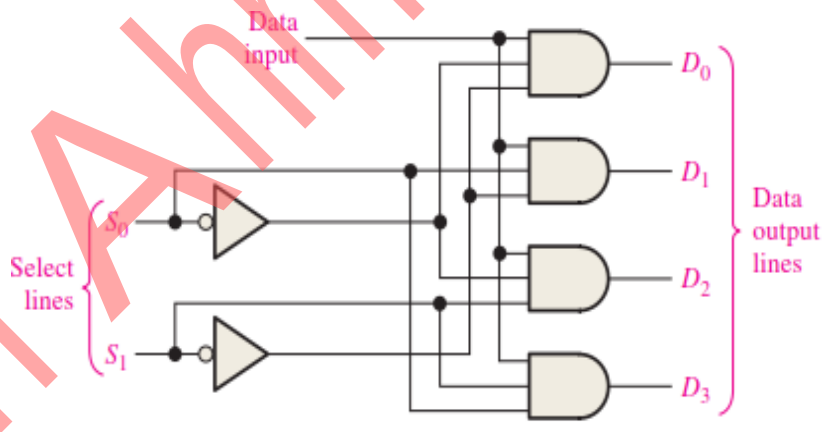


# Example 9-13

The serial data-input waveform (Data in) and data-select inputs ($S_0$ and $S_1$ ) are shown in Fig.

Determine the data-output waveforms on $D_0$ through $D_3$ for the demultiplexer in Fig.




# Solution

Notice that the select lines go through a binary sequence so that each successive input bit is routed to $D_0$, $D_1$, $D_2$, and $D_3$ in sequence, as shown by the output waveforms in Fig.

| Data-Select Lines $S_1$ | $S_0$ | Output Selected |
|---|---|---|
| 0 | 0 | $D_0$ |
| 0 | 1 | $D_1$ |
| 1 | 0 | $D_2$ |
| 1 | 1 | $D_3$ |

# Selected Key Terms

| | |
|---|---|
| ***Cascading*** | Connecting two or more similar devices in a manner that expands the capability of one device. |
| ***Comparator*** | A digital circuit that compares the magnitudes of two quantities and produces an output indicating the relationship of the quantities. |
| ***Decoder*** | A digital circuit that converts coded information into a familiar or noncoded form. |
| ***Demultiplexer (DEMUX)*** | A circuit that switches digital data from one input line to several output lines in a specified time sequence. |
| ***Encoder*** | A digital circuit that converts information to a coded form. |
| ***Full-adder*** | A digital circuit that adds two bits and an input carry to produce a sum and an output carry. |
| ***Half-adder*** | A digital circuit that adds two bits and produces a sum and an output carry. It cannot handle input carries. |
| ***Multiplexer (MUX)*** | A circuit that switches digital data from several input lines onto a single output line in a specified time sequence. |

# True/False Quiz

1. A half-adder adds two binary bits.
2. A half-adder has a carry output only.
3. A full adder adds two bits and produces two outputs.
4. A full-adder can be realized only by using 2-input XOR gates.
5. When the input bits are both 1 and the input carry bit is 1, the sum output of a full adder is 1.
6. The output of a comparator is 0 when the two binary inputs given are equal.
7. A decoder detects the presence of a specified combination of input bits.
8. The 4-line-to-10-line decoder and the 1-of-10 decoder are two different types.
9. An encoder essentially performs a reverse decoder function.
10. A multiplexer is a logic circuit that allows digital information from a single source to be routed onto several lines.

**1.** T **2.** F **3.** F **4.** F **5.** T
**6.** F **7.** T **8.** F **9.** T **10.** F

# Quiz

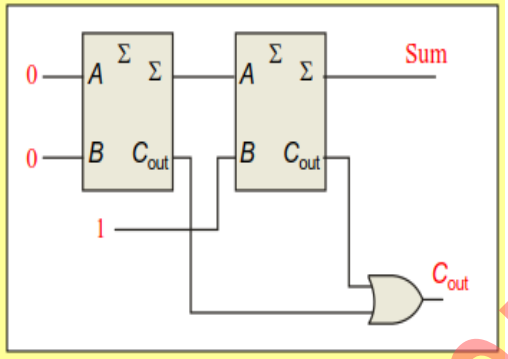
1. For the full-adder shown, assume the input bits are as shown with $A = 0$, $B = 0$, $C_{in} = 1$. The Sum and $C_{out}$ will be

a. Sum = 0 $C_{out}$ = 0

b. Sum = 0 $C_{out}$ = 1

c. Sum = 1 $C_{out}$ = 0

d. Sum = 1 $C_{out}$ = 1



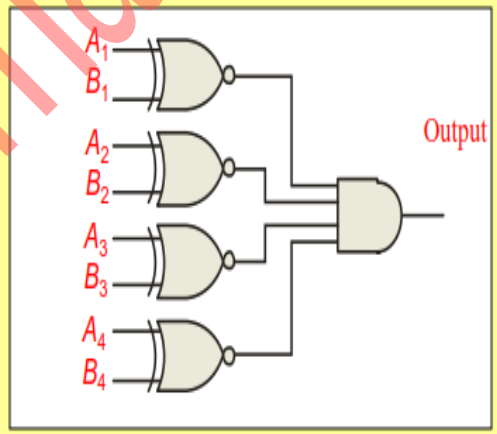

2. The output will be LOW if

a. $A < B$

b. $A > B$

c. both a and b are correct

d. $A = B$




3. If you expand two 4-bit comparators to accept two 8-bit numbers, the output of the least significant comparator is

a. equal to the final output

b. connected to the cascading inputs of the most significant comparator

c. connected to the output of the most significant comparator

d. not used

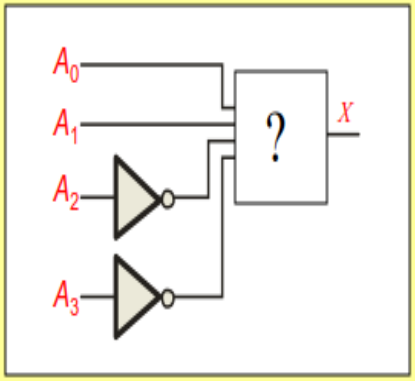
4. Assume you want to decode the binary number 0011 with an active-LOW decoder. The missing gate should be

a. an AND gate

b. an OR gate

c. a NAND gate

d. a NOR gate



# Quiz

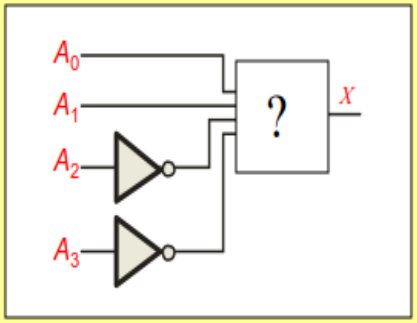
5. Assume you want to decode the binary number 0011 with an active-HIGH decoder. The missing gate should be

a. an AND gate

b. an OR gate

c. a NAND gate

d. a NOR gate




6. The 74138 is a 3-to-8 decoder. Together, two of these ICs can be used to form one 4-to-16 decoder. To do this, connect

a. one decoder to the LSBs of the input; the other decoder to the MSBs of the input

b. all chip select lines to ground

c. all chip select lines to their active levels

d. one chip select line on each decoder to the input MSB

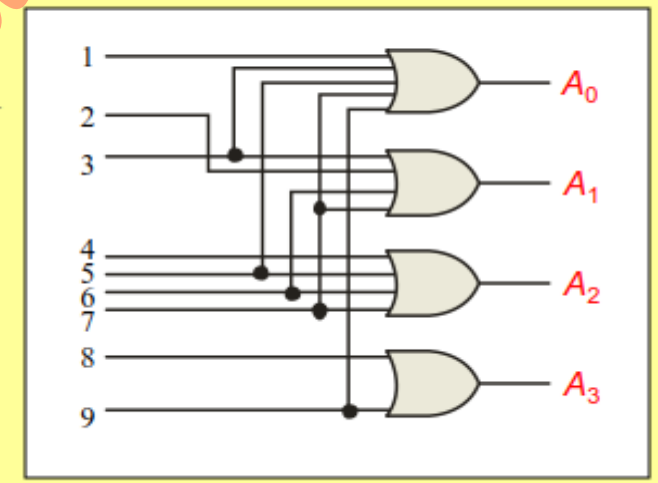
7. The decimal-to-binary encoder shown does not have a zero input. This is because

a. when zero is the input, all lines should be LOW

b. zero is not important

c. zero will produce illegal logic levels

d. another encoder is used for zero



# Quiz

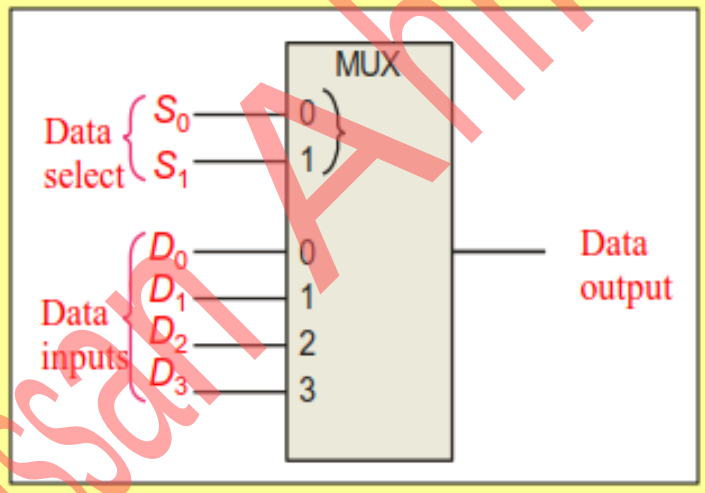
8. If the data select lines of the MUX are $S_1S_0 = 11$, the output will be

a. LOW

b. HIGH

c. equal to $D_0$

d. equal to $D_3$




9. The 74138 decoder can also be used as

a. an encoder

b. a DEMUX

c. a MUX

d. none of the above

Answers:

| | |
|---|---|
| 1. c | 6. d |
| 2. c | 7. a |
| 3. b | 8. d |
| 4. c | 9. b |
| 5. a | 10. b |

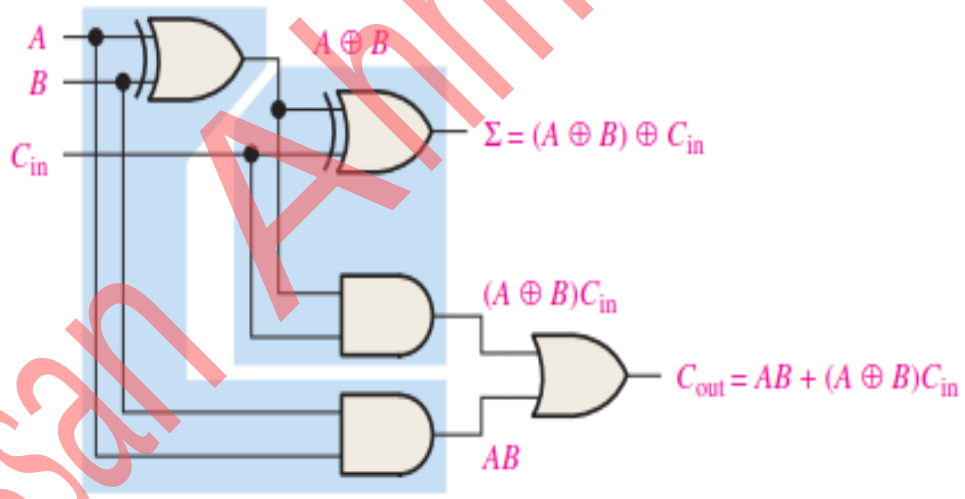
# Problems & Solutions

**Prob. 6-1** For the full-adder in Figure, determine the logic state (1 or 0) at each gate output for the following inputs

*a*) $A = 1,\ B = 1,\ C_{in} = 1;$

*b*) $A = 0,\ B = 1,\ C_{in} = 1;$

*c*) $A = 0,\ B = 1,\ C_{in} = 0;$




**Sol.**

(a) XOR (upper) output = 0, Sum output = 1, AND (upper) output = 0, AND (lower) output = 1, Carry output = 1

(b) XOR (upper) output = 1, Sum output = 0, AND (upper) output = 1, AND (lower) output = 0, Carry output = 1

(c) XOR (upper) output = 1, Sum output = 1, AND (upper) output = 0, AND (lower) output = 0, Carry output = 0

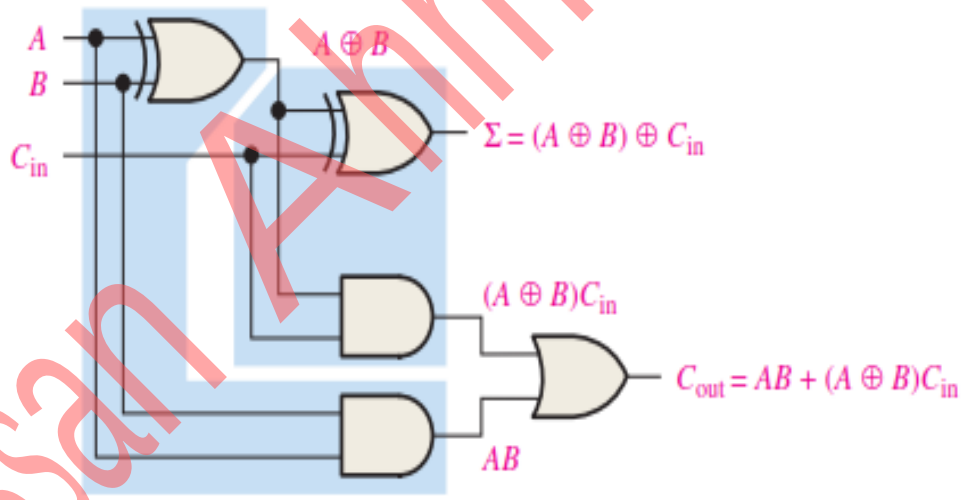
**Prob. 6-2** Determine the outputs of a full-adder for each of the following inputs:

*a*) $A=1,\ B=0,\ C_{in}=0;$

*b*) $A=0,\ B=0,\ C_{in}=1;$

*c*) $A=0,\ B=1,\ C_{in}=1;$

*d*) $A=1,\ B=1,\ C_{in}=1;$




**Sol.**

(a) $\Sigma = 1, C_{out} = 0$

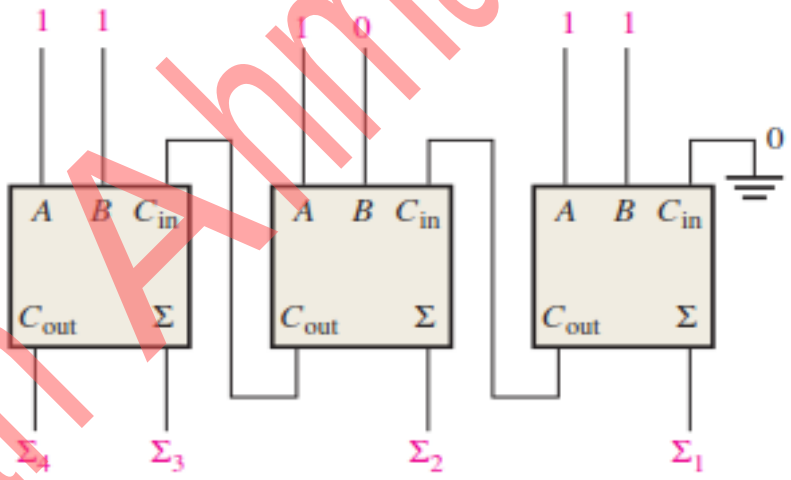
(b) $\Sigma = 1, C_{out} = 0$

(c) $\Sigma = 0, C_{out} = 1$

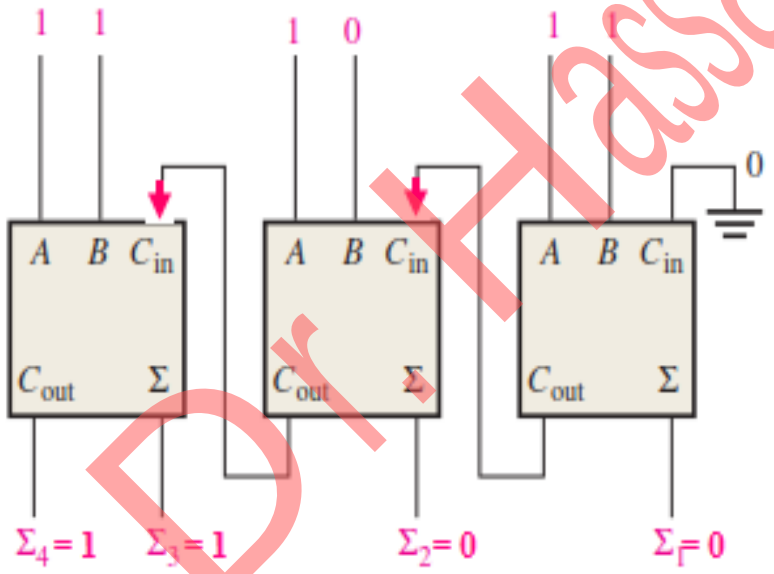
(d) $\Sigma = 1, C_{out} = 1$

# Prob. 6-3

For the parallel adder in Figure, determine the complete sum by analysis of the logical operation of the circuit. Verify your result by longhand addition of the two input numbers.




## Sol.

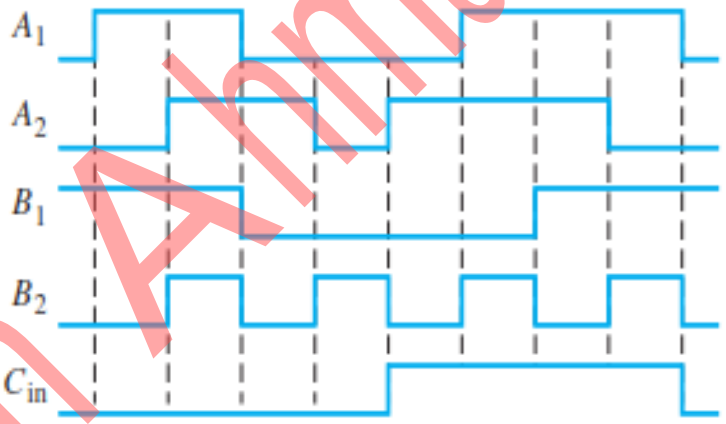



longhand addition

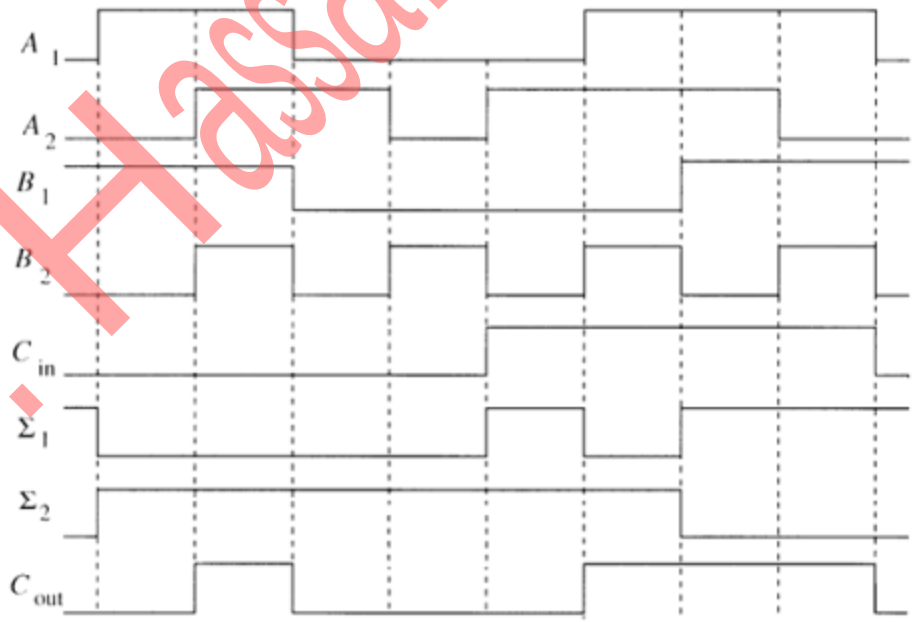
```
1 1 1
  111
  101  +
-----
 1100
```

# Prob. 6-4

The input waveforms in Figure are applied to a 2-bit adder. Determine the waveforms for the sum and the output carry in relation to the inputs by constructing a timing diagram.



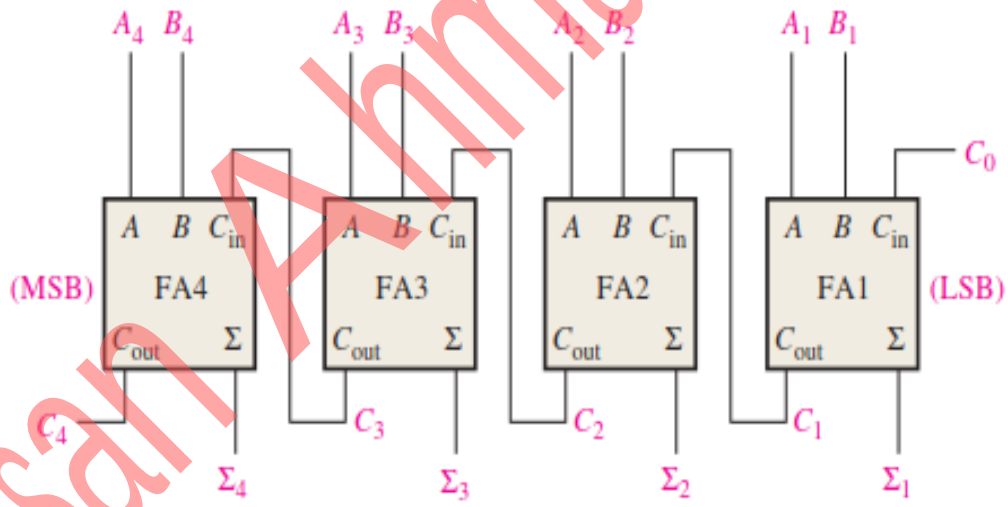

**Sol.**



# Prob. 6-5

The following sequences of bits (right-most bit first) appear on the inputs to a 4-bit parallel adder shown in Fig. Determine the resulting sequence of bits on each sum output.

| | |
|---|---|
| $A_1$ | 1010 |
| $A_2$ | 1100 |
| $A_3$ | 0101 |
| $A_4$ | 1101 |
| $B_1$ | 1001 |
| $B_2$ | 1011 |
| $B_3$ | 0000 |
| $B_4$ | 0001 |

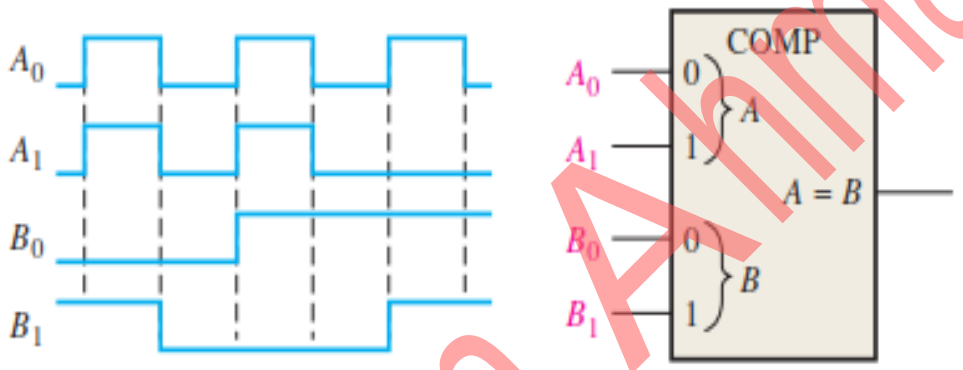



## Sol.

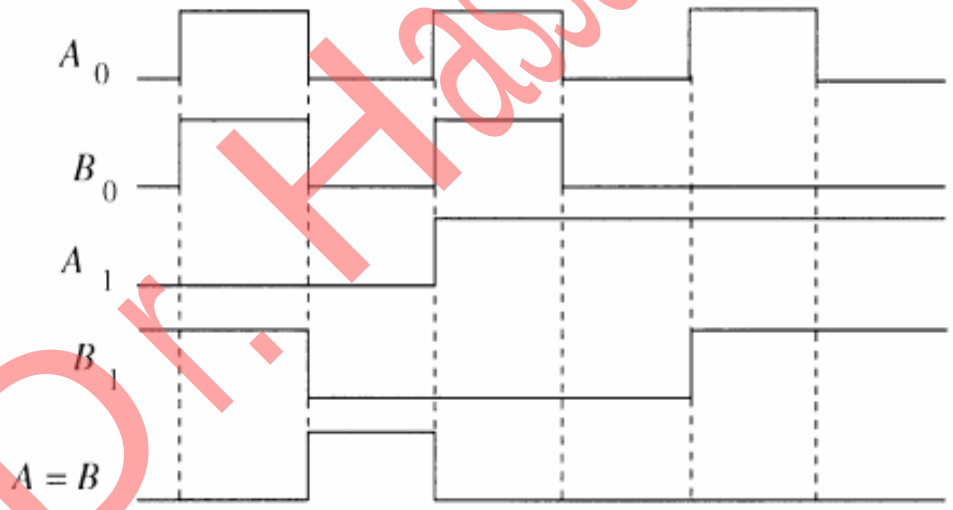
| $A_4$ | $A_3$ | $A_2$ | $A_1$ | $B_4$ | $B_3$ | $B_2$ | $B_1$ | $\Sigma_5$ | $\Sigma_4$ | $\Sigma_3$ | $\Sigma_2$ | $\Sigma_1$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

$\Sigma_1 = 0110$
$\Sigma_2 = 1011$
$\Sigma_3 = 0110$
$\Sigma_4 = 0001$
$\Sigma_5 = 1000$

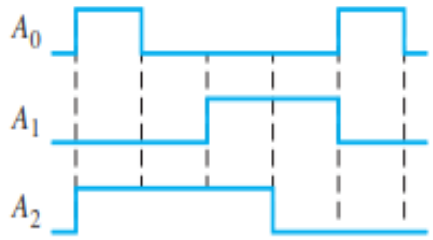
**Prob. 6-6** The waveforms in Figure are applied to the comparator as shown. Determine the output ($A = B$) waveform.

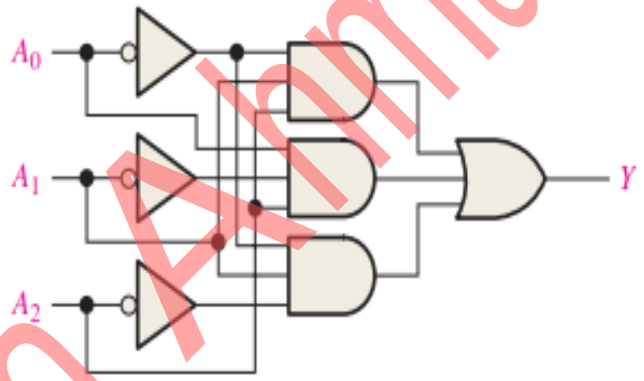



**Sol.** The $A = B$ output is HIGH when $A_0 = B_0$ and $A_1 = B_1$ .



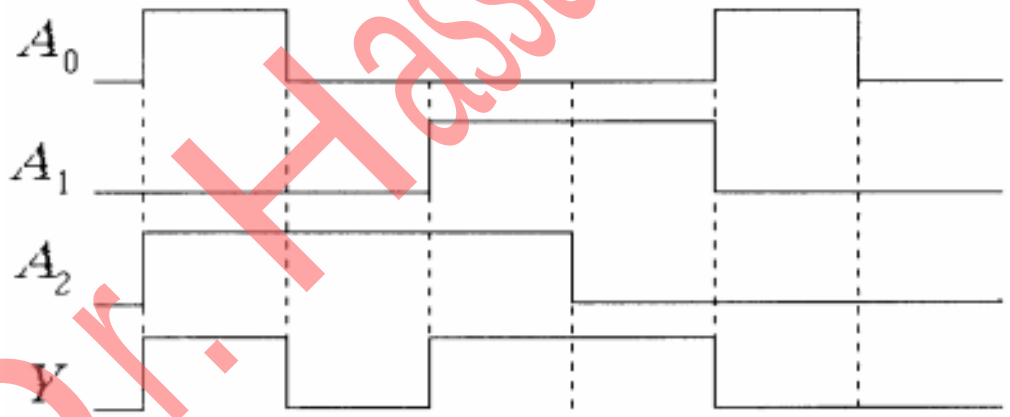
**Prob. 6-7** If the input waveforms are applied to the decoding logic as indicated in Figure, sketch the output waveform in proper relation to the inputs.



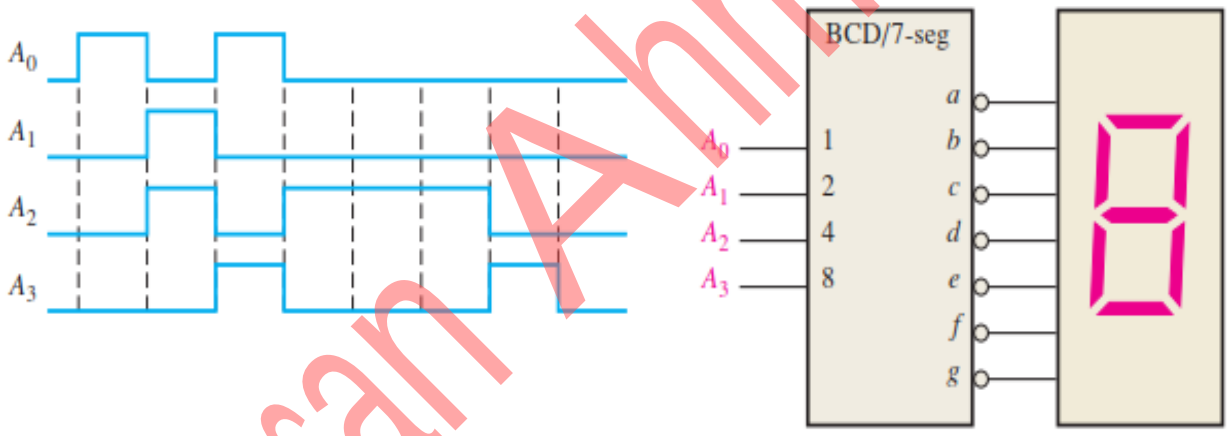

**Sol.**

$$Y = A_2 A_1 \overline{A_0} + A_2 \overline{A_1} A_0 + \overline{A_2} A_1 A_0$$



# Prob. 6-8

A 7-segment decoder/driver drives the display in Figure. If the waveforms are applied as indicated, determine the sequence of digits that appears on the display.

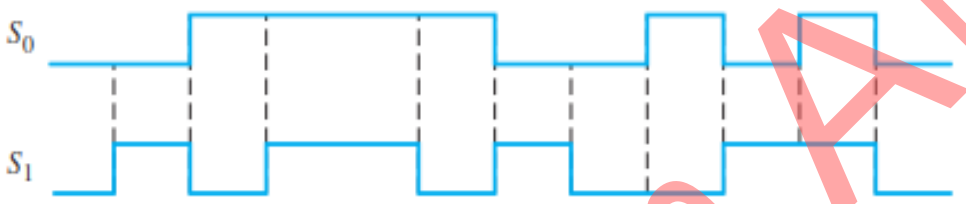


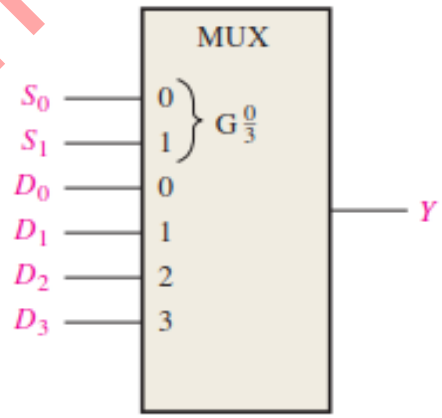

# Sol.

0 1 6 9 4 4 4 8 0

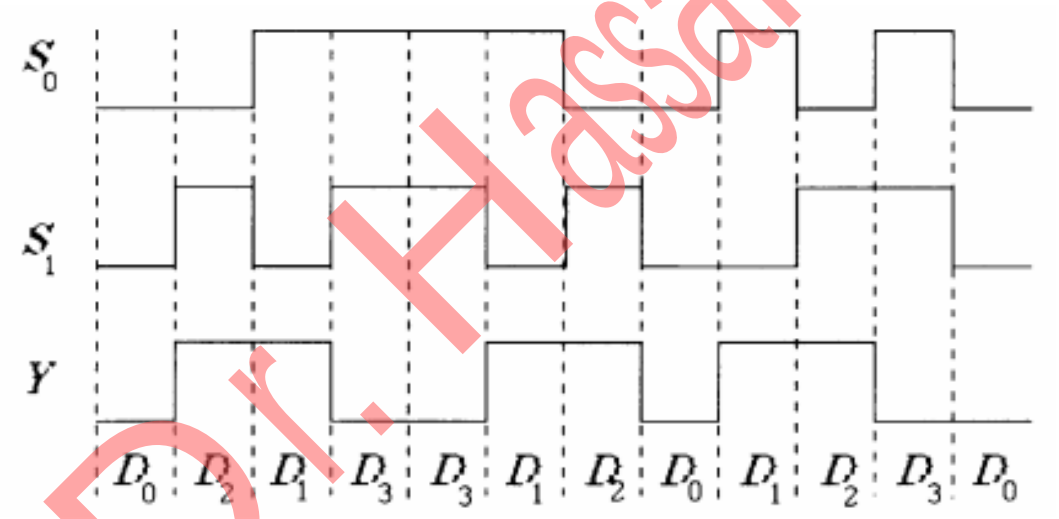
# Prob. 6-9

If the data-select inputs to the multiplexer in Figure are sequenced as shown by the waveforms in Figure, determine the output waveform with the data inputs: $D_0 = 1,\ D_1 = 0, D_2 = 0, D_3 = 1, S_0 = 0, S_1 = 1.$




## Sol.




# The end of Lecture_09, chapter 6