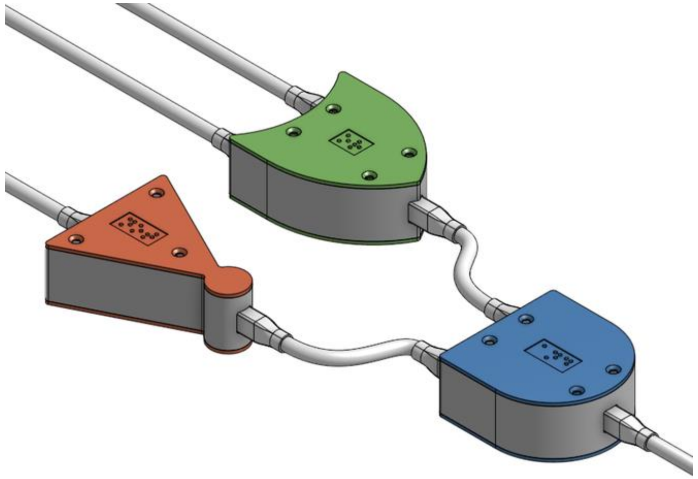
جامعة شقراء
Shaqra University



# CSC 211 - Digital Logic Design
# 211 عال - تصميم المنطق الرقمي

First Term - 1439/1440

## Lecture #9

Dr. Hazem Ibrahim Shehata
Assistant Professor
College of Computing and Information Technology

# Administrivia

- Assignment #2:
  - To be released on Sunday.

Website: http://hshehata.github.io/courses/su/cs211

# Chapter 5: Combinational Logic Analysis

# Types of Digital Logic Circuits

- There are two types of Digital Logic Circuits:

1. Combinational Logic Circuits:
   - Output is a pure function in the present inputs only!
   - Examples: adders, subtractors, encoder, decoders, multiplexers, … etc.
2. Sequential Logic Circuits:
   - Output depends not only on the present input but also on the history of the input!!
   - Examples: flip-flops, latches, registers, counters, memory, … etc.

# Implementing Combinational Logic

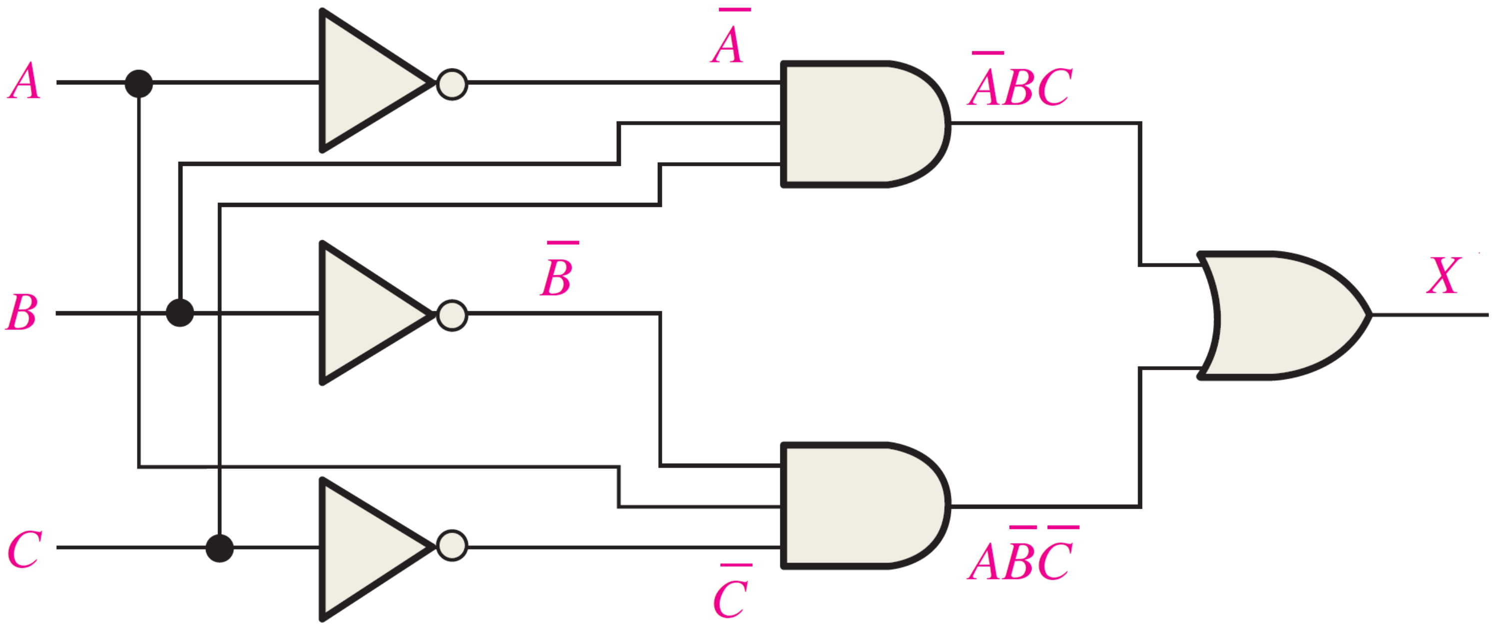
- Methods for implementing Combinational Logic:
  1. AND-OR Logic
     - Usage: implementing SOP and POS expressions
  2. NAND Logic
     - Usage: implementing SOP expressions
  3. NOR Logic
     - Usage: implementing POS expressions
  4. …

# Implementing SOP expressions using AND-OR Logic

- Each product term is implemented by an AND gate whose inputs are the literals of the product term.
- All the outputs of the AND gates are fed to an OR gate.

➢Example: SOP ➔ AND-OR logic

$$X = \bar{A}BC + A\bar{B}\bar{C}$$

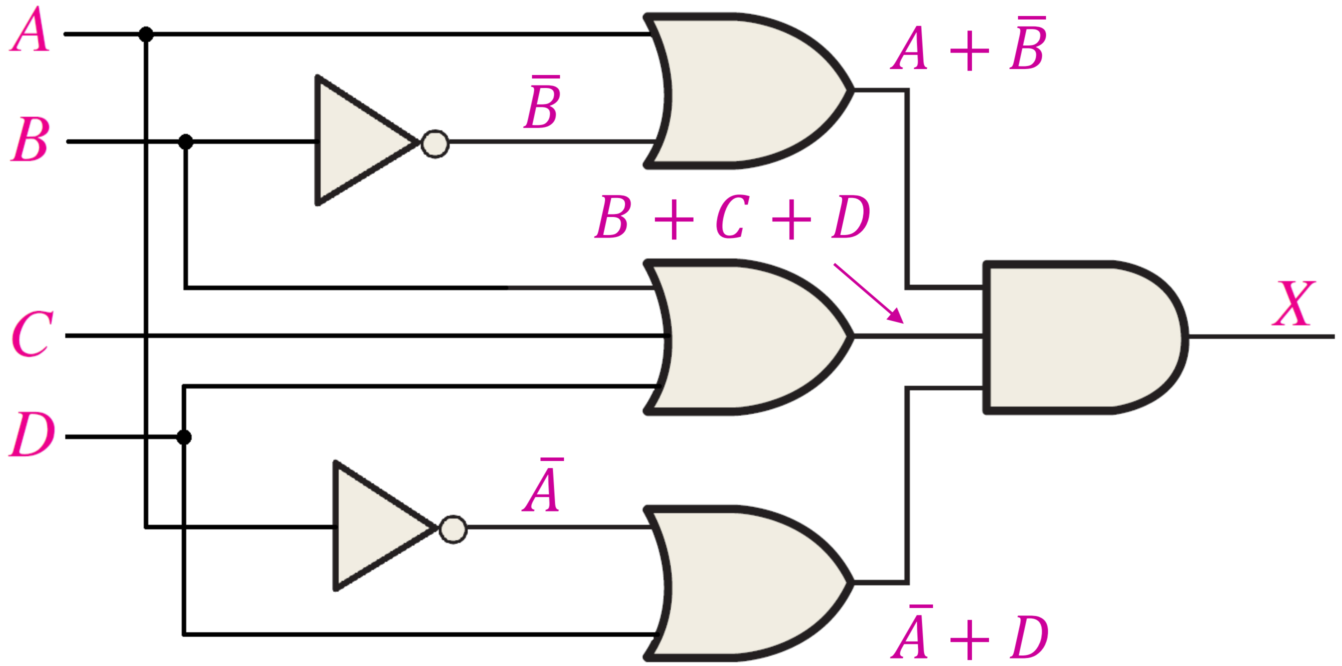


# Implementing POS expressions using AND-OR Logic

- Each sum term is implemented by an OR gate whose inputs are the literals of the product term.
- All the outputs of the OR gates are fed to an AND gate.

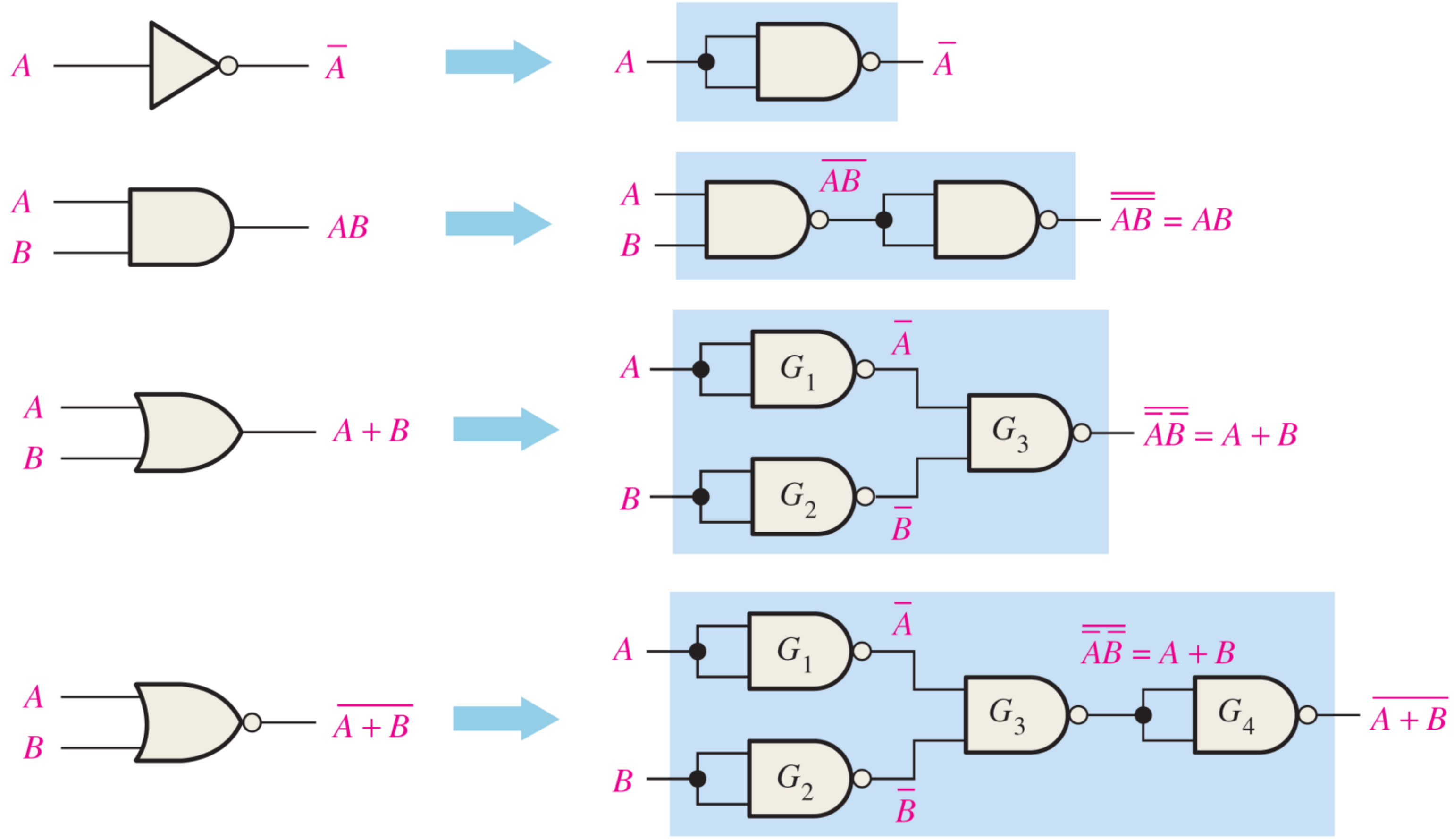
➢ Example: POS ➔ AND-OR logic

$$X = (A + \bar{B})(B + C + D)(\bar{A} + D)$$



# Universality of NAND gate

- NAND gates can be used to implement all other gates: NOT, AND, OR, NOR!!
- Hence, NAND gates can be used to implement any combinational circuit!!!

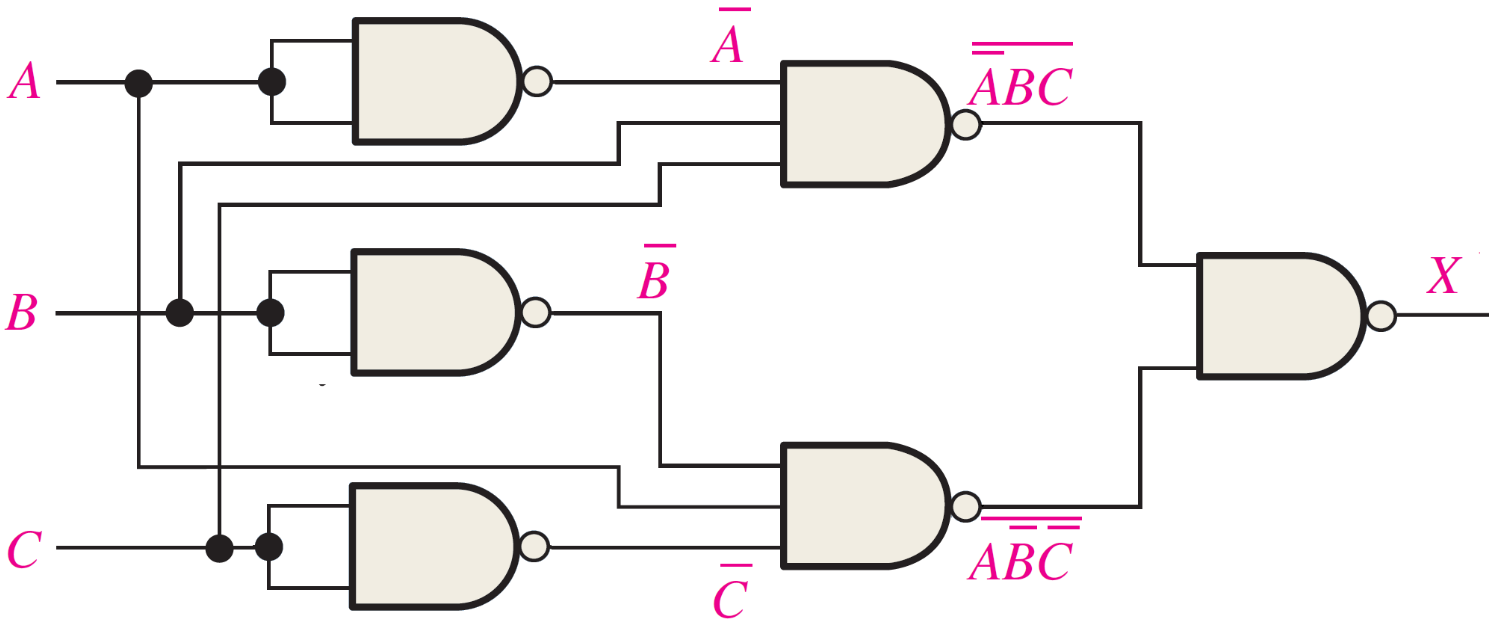


# Implementing SOP expressions using NAND Logic

- Start from the AND-OR implementation (like slide #7) and then replace each:
  - Inverter ➔ NAND gate (Connected inputs).
  - AND ➔ NAND
  - OR ➔ NAND

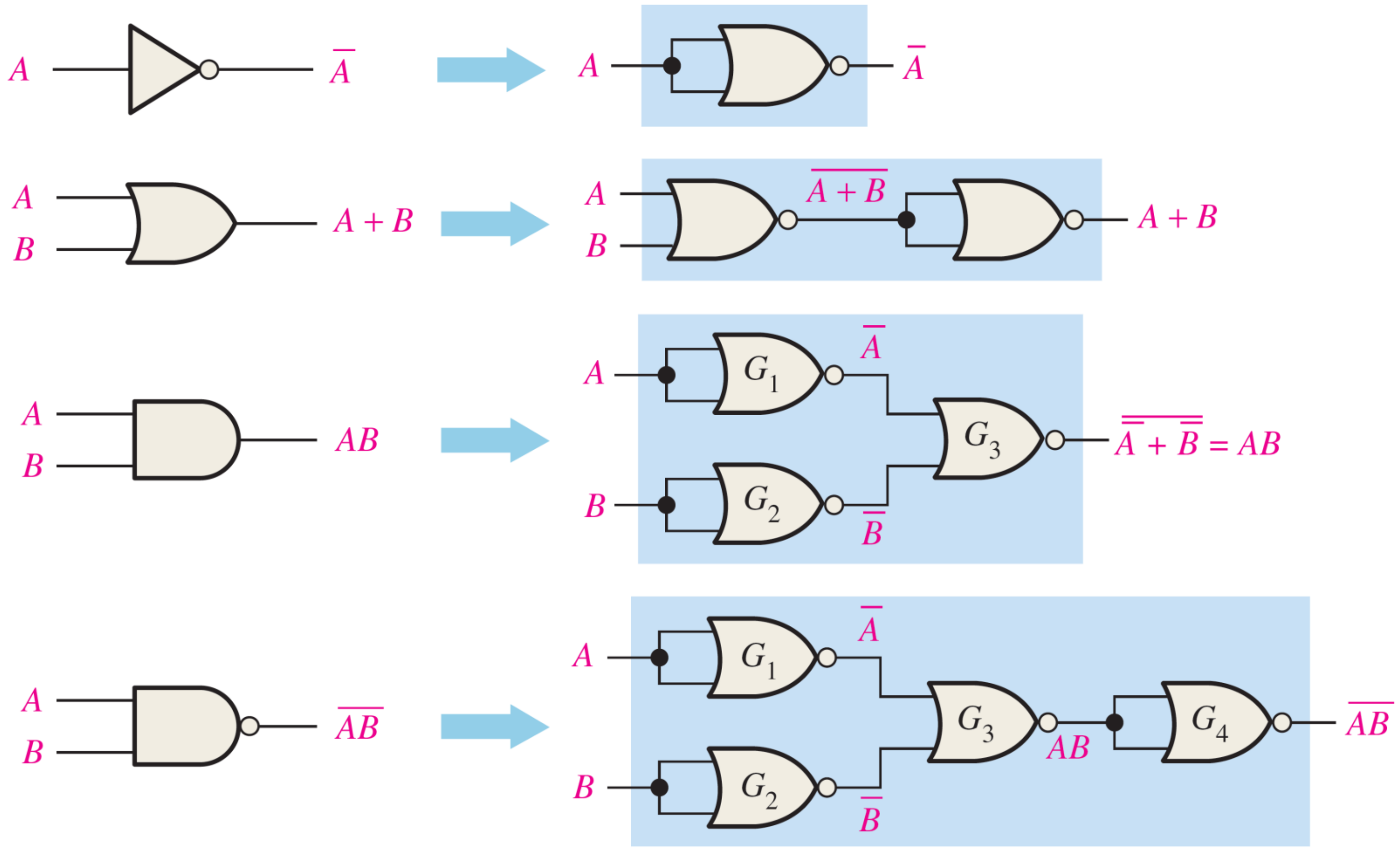
➢ Example: SOP ➔ NAND logic

$$X = \bar{A}BC + A\bar{B}\bar{C}$$



# Universality of NOR gate

- NOR gates can be used to implement all other gates: NOT, AND, OR, NAND!!
- Hence, NOR gates can be used to implement any combinational circuit!!!

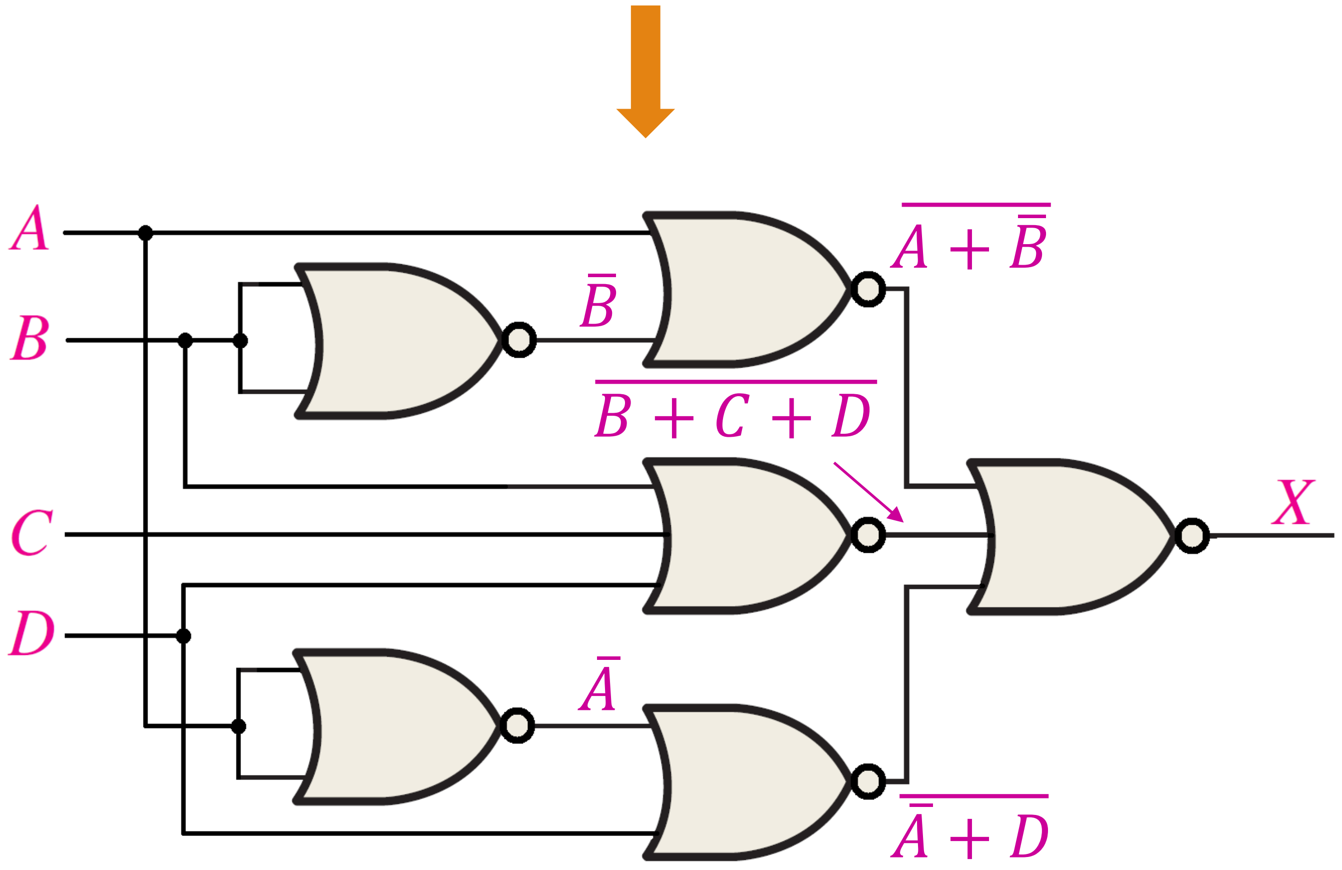


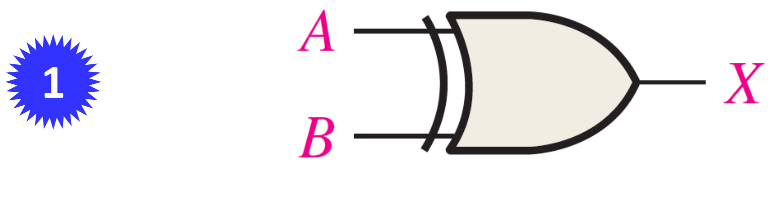
# Implementing POS expressions using NOR Logic

- Start from the AND-OR implementation (like slide #8) and then replace each:
  - Inverter ➔ NOR gate (Connected inputs).
  - OR ➔ NOR
  - AND ➔ NOR

➢ Example: POS ➔ AND-OR logic

$$X = (A + \bar{B})(B + C + D)(\bar{A} + D)$$



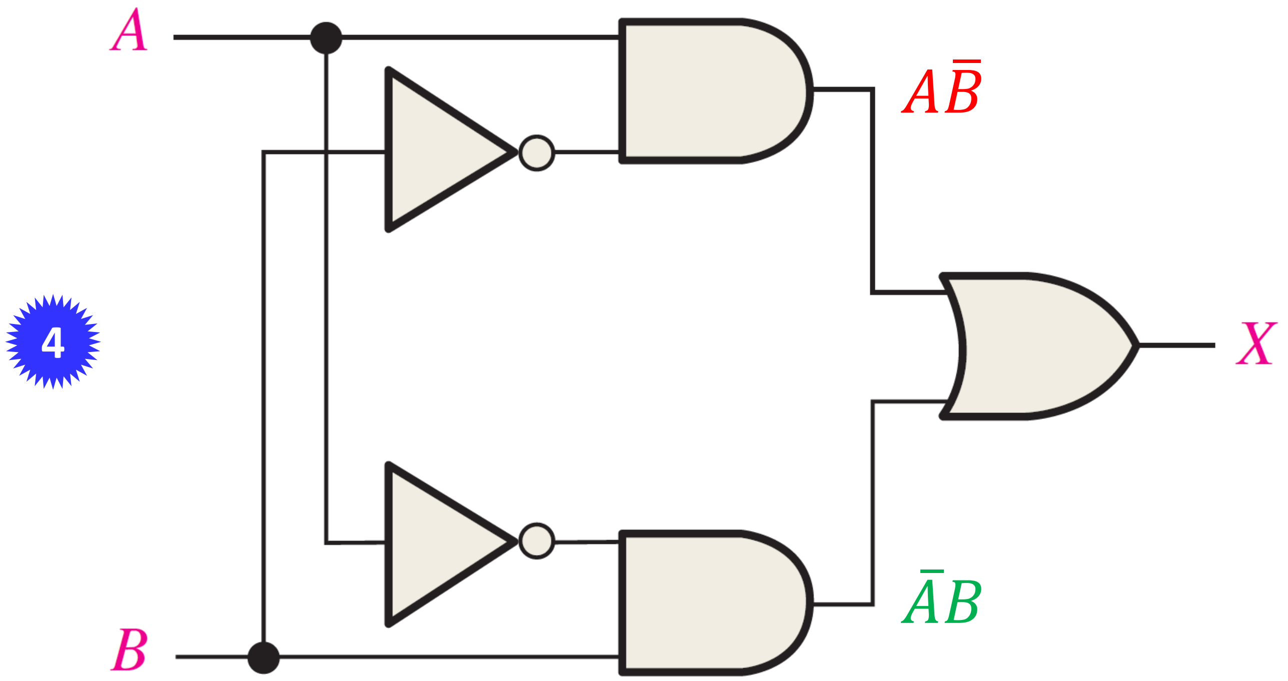



2

| $A$ | $B$ | $X = A \oplus B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 → $\bar{A}B$ |
| 1 | 0 | 1 → $A\bar{B}$ |
| 1 | 1 | 0 |

3

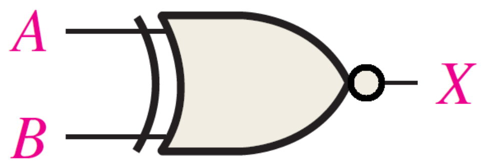
$$X = \bar{A}B + A\bar{B}$$




# Implementing XOR Gate (⊕) using AND-OR Logic

# Implementing XNOR Gate ($\odot$) using AND-OR Logic

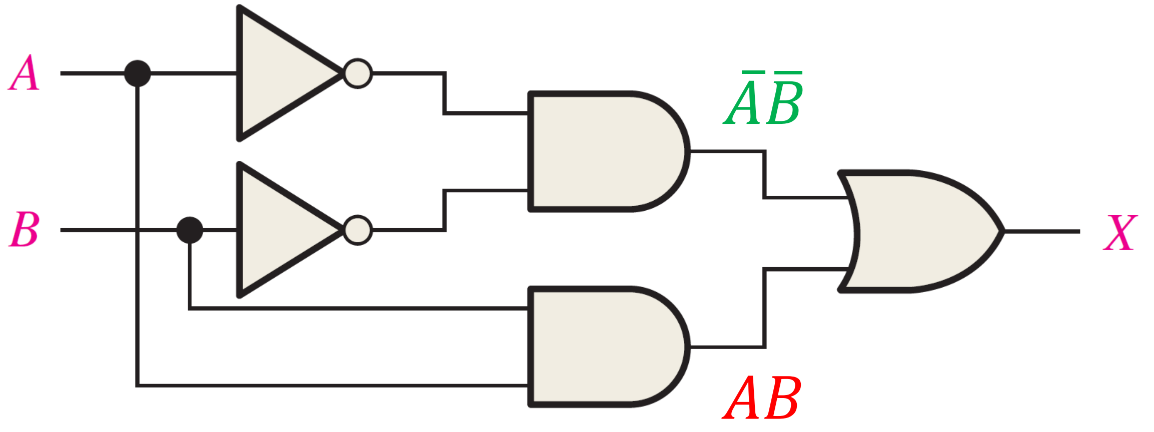
1




2

$$
\begin{aligned}
X &= A \odot B \\
&= \overline{A \oplus B} \\
&= \overline{\bar{A}B + A\bar{B}} \\
&= (A + \bar{B})(\bar{A} + B) \\
&= A\bar{A} + \bar{A}\bar{B} + AB + B\bar{B} \\
&= \bar{A}\bar{B} + AB
\end{aligned}
$$

3



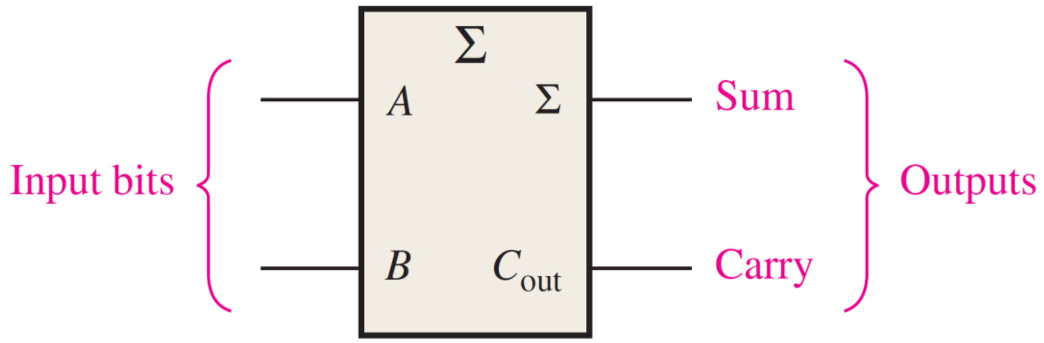
# Chapter 6: Functions of Combinational Logic

# The Half-Adder

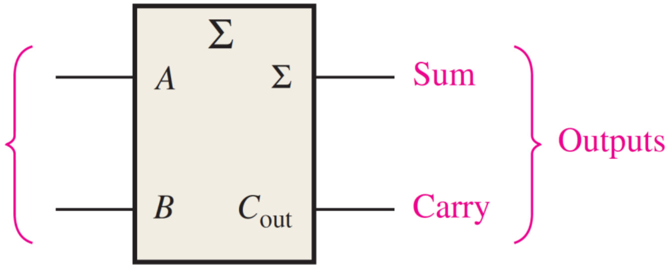
- A combinational logic circuit.
- Purpose: adds two bits, by rules of binary addition:

$$\begin{array}{r} 0 \\ +\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +\ 1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\ 1 \\ \hline 10 \end{array}$$

- Input/Output:
  - Accepts two binary digits
  - Produces a sum bit and a carry bit.



1



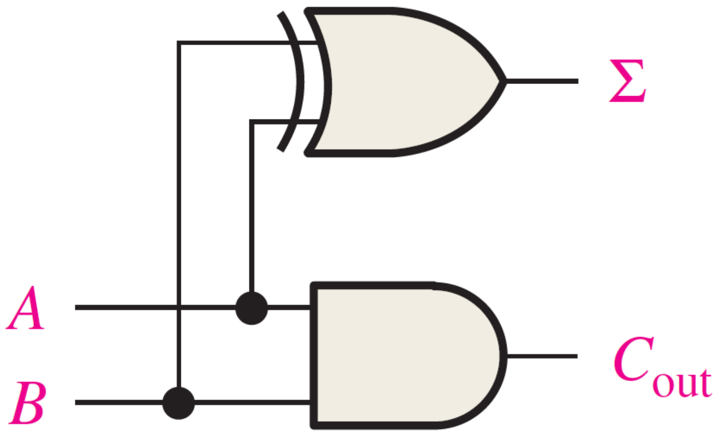

2

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

3

$$C_{out} = AB$$
$$\Sigma = \bar{A}B + A\bar{B} = A \oplus B$$

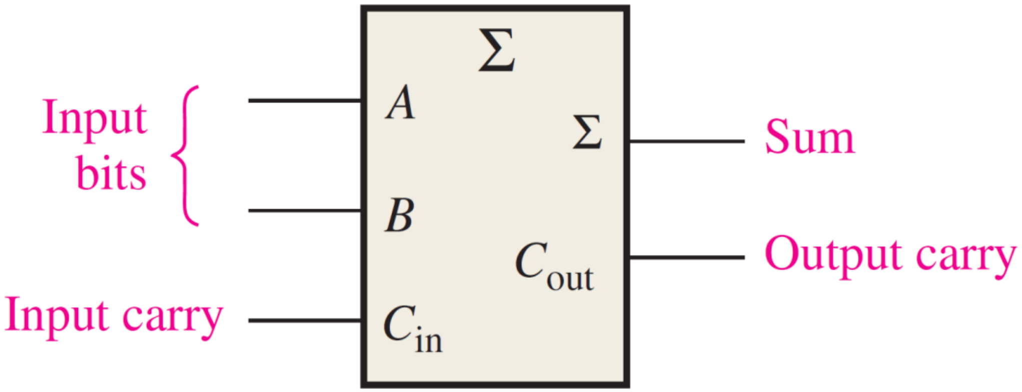
4



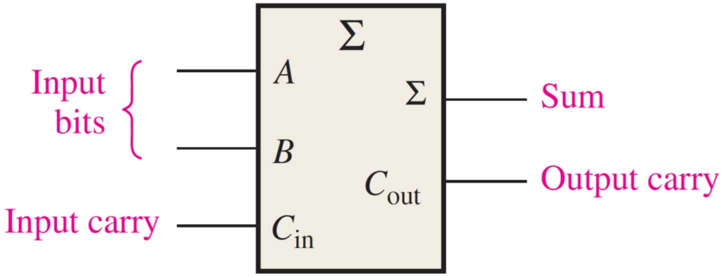

# Implementing the Half Adder

# The Full-Adder

- A combinational logic circuit.
- Purpose: adds three bits
- Input/Output:
  - Accepts two binary digits & input carry
  - Produces a sum bit and a carry bit.



1




2

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

3

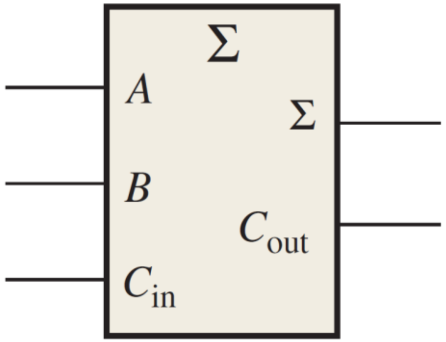
$$\begin{aligned} C_{out} &= \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in} \\ &= (\bar{A}B + A\bar{B})C_{in} + AB(\bar{C}_{in} + C_{in}) \\ &= (A \oplus B)\, C_{in} + AB \end{aligned}$$

$$\begin{aligned} \Sigma &= \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in} \\ &= (\bar{A}\bar{B} + AB)C_{in} + (\bar{A}B + A\bar{B})\bar{C}_{in} \\ &= (A \odot B)C_{in} + (A \oplus B)\bar{C}_{in} \\ &= (\overline{A \oplus B})C_{in} + (A \oplus B)\bar{C}_{in} \\ &= (A \oplus B) \oplus C_{in} \end{aligned}$$

# Implementing the Full Adder

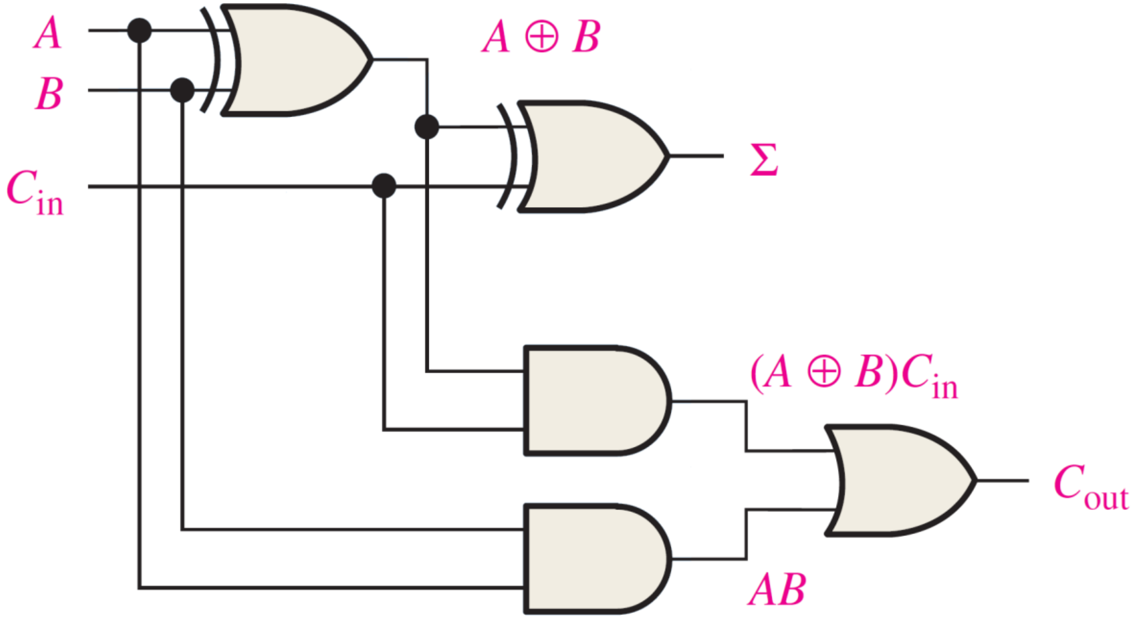
$$\Sigma = (A \oplus B) \oplus C_{in}$$

3

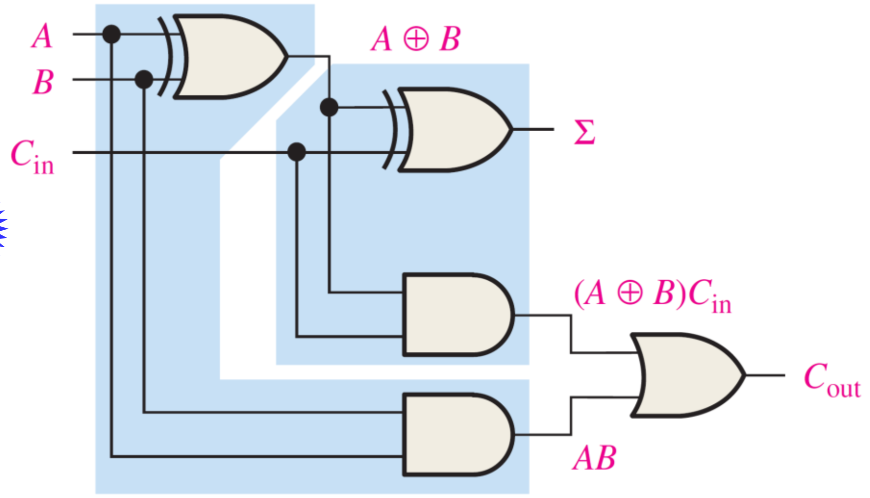


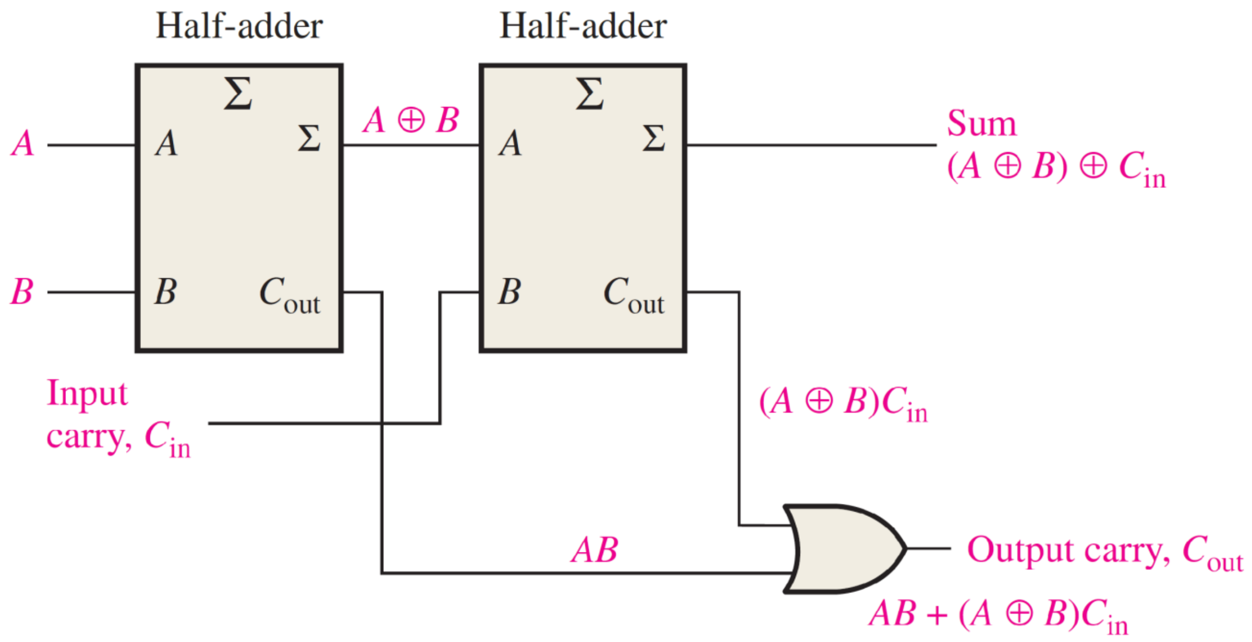

$$C_{out} = (A \oplus B)\, C_{in} + AB$$

4




# Implementing the Full Adder




# Implementing the Full Adder

# Reading Material

- Floyd, Chapter 5:
  - Pages 234 – 235, 237, 239 – 248, 250, 252 – 255
- Floyd, Chapter 6:
  - Pages 285 - 289