

Digital Circuits

ECS 371

Dr. Prapun Suksompong

prapun@siit.tu.ac.th

Lecture 9

Office Hours:

BKD 3601-7

Monday 9:00-10:30, 1:30-3:30

Tuesday 10:30-11:30

ECS371.PRAPUN.COM

Announcement

- HW3 posted on the course web site
 - Chapter 4: 5(b,d), 26b, 30b, 32a, 34a, 44
 - **Write down all the steps** that you have done to obtain your answers.
 - Due date: July 9, 2009 (Thursday)
- Today
 - Use handout from lecture 8 first.
 - The new handout is for Chapter 5 (except the first few slides).

Caution

When you see \overline{ABC} or \overline{ABC} on quiz/HW/exam, please always **double-check** whether the bars on the top are disconnected.

This is the K-map for
 $X = \overline{ABC}$ which is the
same as $X = \overline{A} \cdot \overline{B} \cdot \overline{C}$

			C	
	AB			C
			1	0
			0	0
			0	0
			0	0
A				B

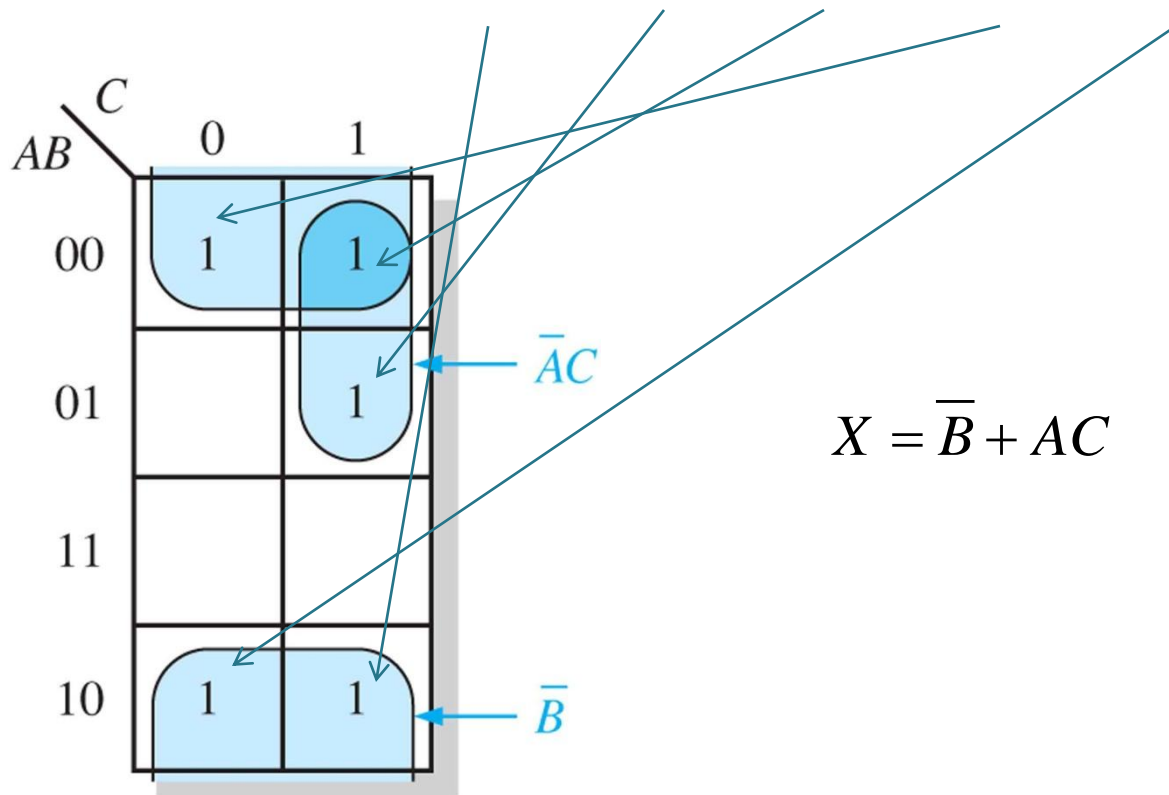
This is the K-map for $X = \overline{ABC}$
which is equivalent to
 $X = \overline{A} + \overline{B} + \overline{C}$

			C	
	AB			C
			1	1
			1	1
			1	0
			1	1
A				B

Example

Use a K-map to minimize the following expression

$$X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + \overline{A}B\overline{C} + \overline{A}B\overline{C} + \overline{A}B\overline{C}$$

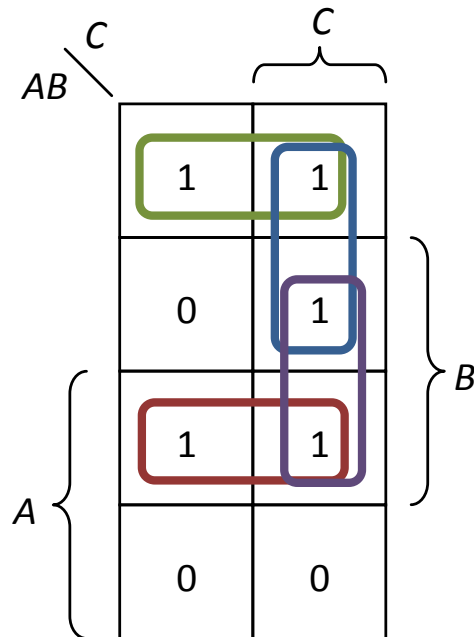


$$X = \overline{B} + AC$$

Non-uniqueness

Use a K-map to minimize the following expression

$$AB + \overline{A}\overline{B} + \overline{A}BC$$



Solution 1: $AB + \overline{A}\overline{B} + \overline{A}C$

Solution 2: $AB + \overline{A}\overline{B} + BC$

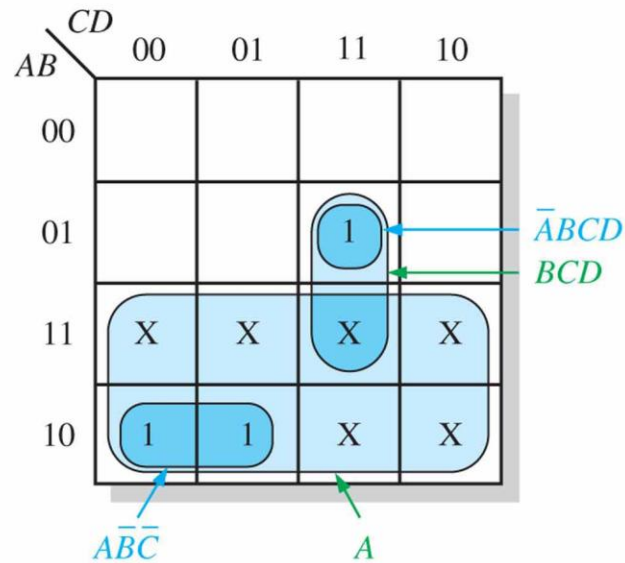
“Don’t Care” Input Combinations

- Sometimes the output doesn’t matter for certain input combinations.
 - For example, the combinations are not allowed in the first place.
- These combinations are called “don’t care”.
- The “don’t care” term can be used to advantage on K-map.
- For each “don’t care” term, place an X in the corresponding cell.
- When grouping the 1s,
 - the Xs can be treated as 1s to make a larger grouping
 - or as 0s if they cannot be used to advantage.

Example

INPUTS				OUTPUT
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

(a) Truth table



(b) Without "don't cares" $Y = \bar{A}\bar{B}\bar{C} + \bar{A}BCD$
 With "don't cares" $Y = A + BCD$

Alternative Methods

- Disadvantages of using K-maps
 - Not applicable for more than five variables
 - Practical only for up to four variables
 - Difficult to automated in a computer program
- There are other ways to minimize Boolean functions.
 - More practical for more than four variables
 - Easily implemented with a computer
 1. Quine-McClusky method
 - Inefficient in terms of processing time and memory usage
 2. Espresso Algorithm
 - de facto standard

New Perspective: 0

- So far, all of our techniques focus on the 1s in the truth tables/K-maps.
- We can look at the 0s as well.

Caution: From this perspective, you are in a different world. In fact, it is a dual world. Techniques used here will be the dual of what we used before.

Canonical Product

- Product-of-Sums (POS) Form

Example: $(A + \bar{B}) \cdot (A + B + C)$

- Standard POS Form (Canonical Product)

Example: $(A + \bar{B} + C) \cdot (A + \bar{B} + \bar{C}) \cdot (A + B + C)$

- Convert expression in POS form into canonical product:

Hint:

$$\begin{aligned} X &= X + 0 \\ &= X + Y \cdot \bar{Y} \\ &= (X + Y) \cdot (X + \bar{Y}) \end{aligned}$$

Truth Table for Canonical Product

Find the value of X for all possible values of the variables when

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

Old way: Convert to SOP form

$$\begin{aligned} X &= (A + \bar{B} + C) \cdot (\bar{A} + B + C) \cdot (\bar{A} + \bar{B} + C) \\ &= \left((A + \bar{B}) \cdot (\bar{A} + B) \cdot (\bar{A} + \bar{B}) \right) + C \\ &= \left((A + \bar{B}) \cdot (\bar{A} + (B \cdot \bar{B})) \right) + C \\ &= \left((A + \bar{B}) \cdot \bar{A} \right) + C \\ &= (\bar{A} \cdot \bar{B}) + C \end{aligned}$$



A	B	C	X	$\bar{A} \cdot \bar{B}$	C
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	0	1

Then, construct the truth table.

We can use the property of sum terms to construct the truth table directly.

Maxterm

- A sum term in a canonical product is called a **maxterm**.
- A maxterm is equal to 0 for only one combination of variable values.

$$A + \bar{B} + C = 0 \text{ iff } (A, B, C) = (0, 1, 0)$$

$$A + \bar{B} + \bar{C} = 0 \text{ iff } (A, B, C) = (0, 1, 1)$$

$$A + B + C = 0 \text{ iff } (A, B, C) = (0, 0, 0)$$

- We say that the maxterm $A + \bar{B} + C$ has a binary value of 010 (decimal 2)
- Maxterm list: $(A + \bar{B}) \cdot (A + B + C) = \prod_{A,B,C} (0, 2, 3)$

because

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

Truth Table for Canonical Product

Find the value of X for all possible values of the variables when

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

New way:

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$\leftarrow A + \bar{B} + C$

$\leftarrow \bar{A} + B + C$

$\leftarrow \bar{A} + \bar{B} + C$

Minterm/Maxterm & Truth Table

Row #	A	B	C	Minterm	Maxterm
0	0	0	0	$\bar{A} \cdot \bar{B} \cdot \bar{C}$	$A + B + C$
1	0	0	1	$\bar{A} \cdot \bar{B} \cdot C$	$A + B + \bar{C}$
2	0	1	0	$\bar{A} \cdot B \cdot \bar{C}$	$A + \bar{B} + C$
3	0	1	1	$\bar{A} \cdot B \cdot C$	$A + \bar{B} + \bar{C}$
4	1	0	0	$A \cdot \bar{B} \cdot \bar{C}$	$\bar{A} + B + C$
5	1	0	1	$A \cdot \bar{B} \cdot C$	$\bar{A} + B + \bar{C}$
6	1	1	0	$A \cdot B \cdot \bar{C}$	$\bar{A} + \bar{B} + C$
7	1	1	1	$A \cdot B \cdot C$	$\bar{A} + \bar{B} + \bar{C}$

In the same way that each minterm corresponds to a unique row of the truth table,

each maxterm corresponds to a unique row of the truth table (in a dual way).

“1”

“0”

Conversion

Row #	A	B	C	Minterm	Maxterm
0	0	0	0	$\bar{A} \cdot \bar{B} \cdot \bar{C}$	$A + B + C$
1	0	0	1	$\bar{A} \cdot \bar{B} \cdot C$	$A + B + \bar{C}$
2	0	1	0	$\bar{A} \cdot B \cdot \bar{C}$	$A + \bar{B} + C$
3	0	1	1	$\bar{A} \cdot B \cdot C$	$A + \bar{B} + \bar{C}$
4	1	0	0	$A \cdot \bar{B} \cdot \bar{C}$	$\bar{A} + B + C$
5	1	0	1	$A \cdot \bar{B} \cdot C$	$\bar{A} + B + \bar{C}$
6	1	1	0	$A \cdot B \cdot \bar{C}$	$\bar{A} + \bar{B} + C$
7	1	1	1	$A \cdot B \cdot C$	$\bar{A} + \bar{B} + \bar{C}$

This tells that the output column of the truth table is 1 on row # 0, 1, 2, 3.

This tells that the output column of the truth table is 0 on row # 4, 5, 6, 7.

$$\Sigma_{A,B,C}(0,1,2,3) = \Pi_{A,B,C}(4,5,6,7)$$

$$\Sigma_{X,Y}(1) = \Pi_{X,Y}(0,2,3)$$

$$\Sigma_{W,X,Y,Z}(0,1,2,3,5,7,11,13) = \Pi_{W,X,Y,Z}(4,6,8,9,10,12,14,15)$$

K-Map POS Minimization

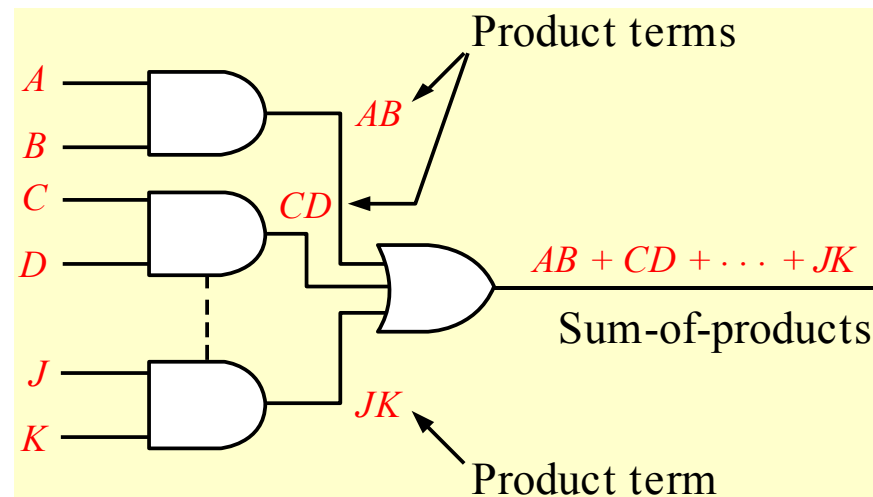
- Goal: Find the “Minimal Product”
- Appendix B in the textbook.
- For a POS expression in standard form, a 0 is placed on the K-map for each sumterm in the expression.
- The cells that do not have a 0 are the cells for which the expression is 1.
- Group 0s to produce instead of grouping 1s.

Combinational Logic

- Chapter 5 and 6
- Reading Assignment:
 - Read Section 5-1 to 5-5.
- Definition: A **combinational logic** is a combination of logic gates interconnected to produce a specified Boolean function with no storage or memory capability.
- Sometimes called **combinatorial logic**.

SOP Implementation: AND-OR Circuit

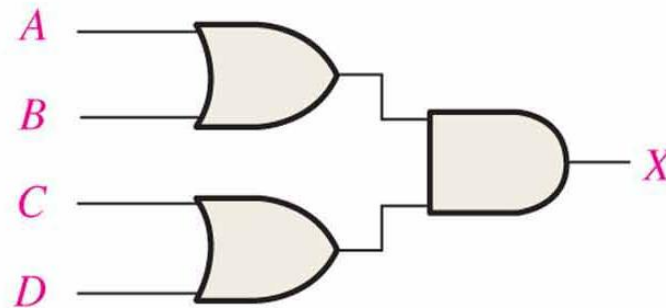
In Sum-of-Products (SOP) form, basic combinational circuits can be directly implemented with AND-OR combinations: first forming the AND terms; then the terms are ORed together.



This is called the **AND-OR configuration**.

Example

Write the output expression of the following circuit as it appears in the figure and then change it to an equivalent AND-OR configuration.

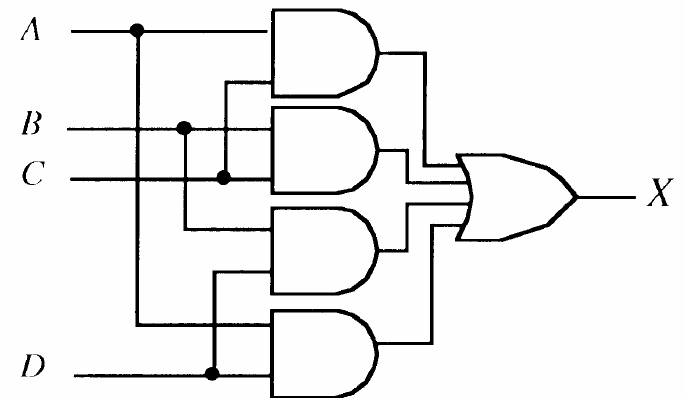


Solution:

$$X = (A + B) \cdot (C + D)$$

$$= (A + B) \cdot C + (A + B) \cdot D$$

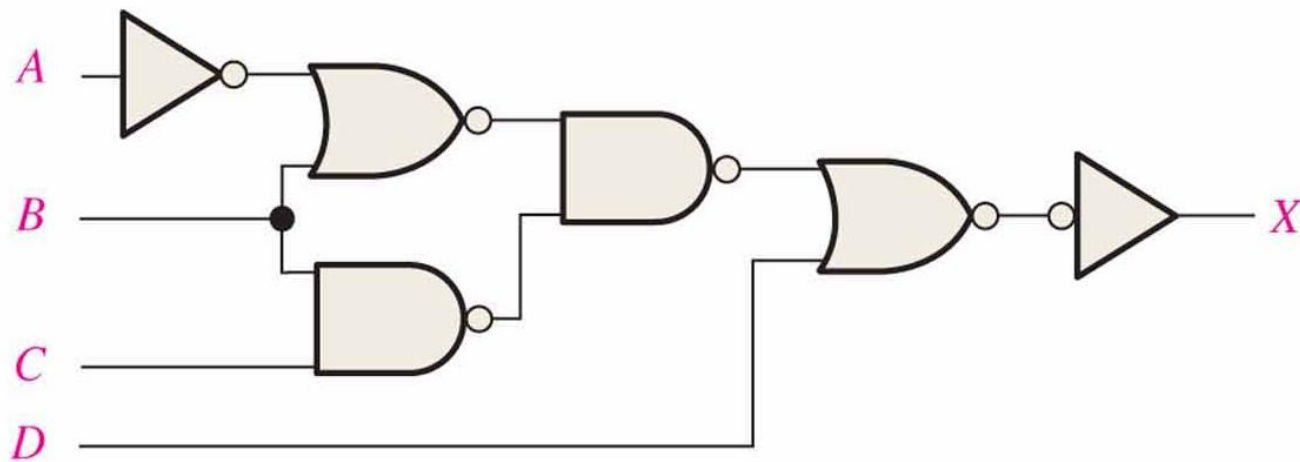
$$= AC + BC + AD + BD$$



(a)

Example

Write the output expression of the following circuit as it appears in the figure and then change it to an equivalent AND-OR configuration.



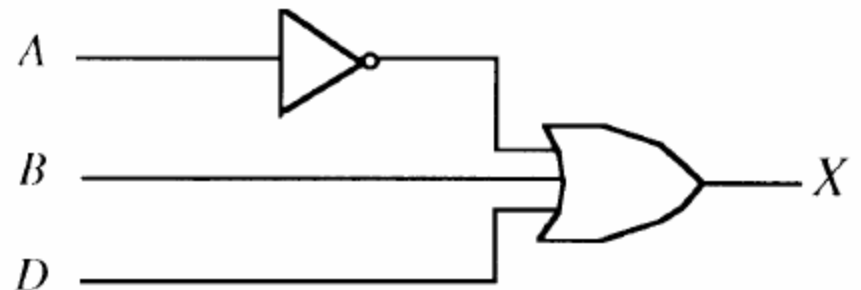
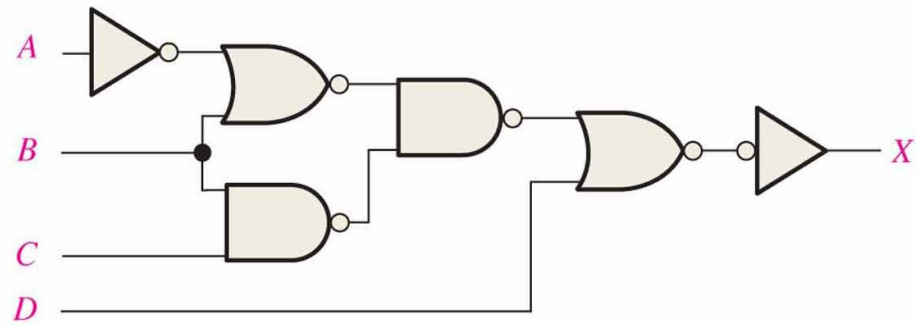
Solution

$$X = \overline{\overline{\overline{A+B}} \cdot \overline{B \cdot C}} + D$$

$$= \overline{\overline{A+B}} \cdot \overline{B \cdot C} + D$$

$$= \overline{A} + B + B \cdot C + D$$

$$= \overline{A} + B + D$$



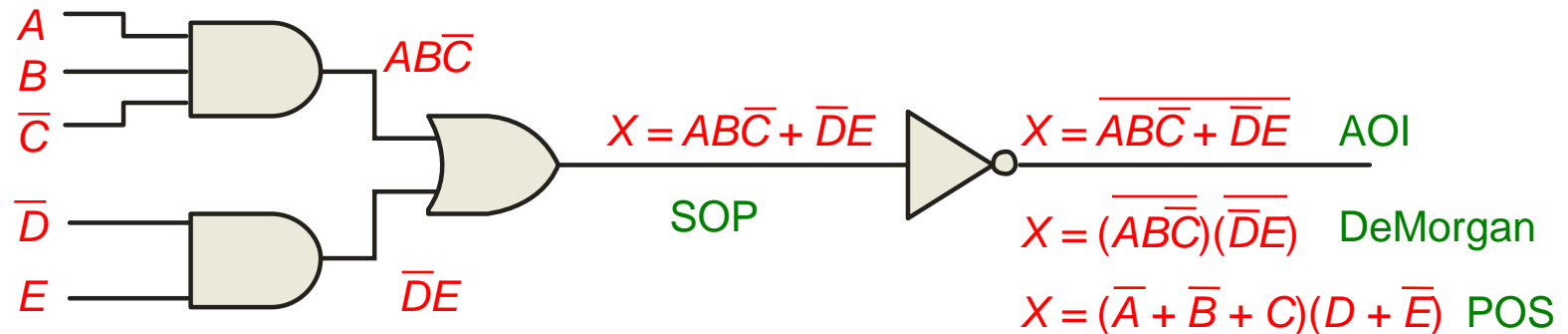
Remark

1. From any logic expression, you can construct a truth table.
2. From the truth table you can get a canonical sum or a minterm list. (This can be simplified to a minimal sum. In any case, you get a SOP expression)
3. Any SOP expression can be implemented using AND gates, OR gates, and inverters.

AND-OR-Invert (AOI) circuit

When the output of a SOP form is inverted, the circuit is called an **AND-OR-Invert circuit**.

The AOI configuration lends itself to product-of-sums (POS) implementation.

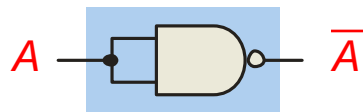


Universal gate

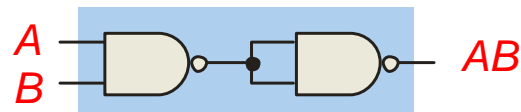
- The term **universal** refers to a property of a gate that permits any logic function to be implemented by that gate or by a combination of gates of that kind.
- Example: NAND gates, NOR gates

NAND Gate as a Universal Gate

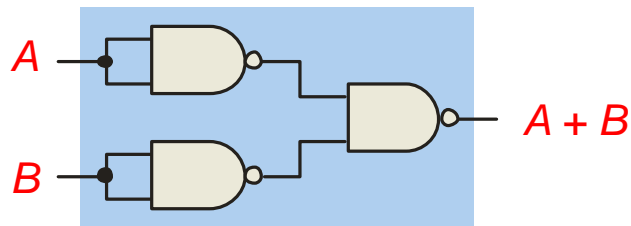
NAND gates are sometimes called **universal** gates because they can be used to produce the other basic Boolean functions.



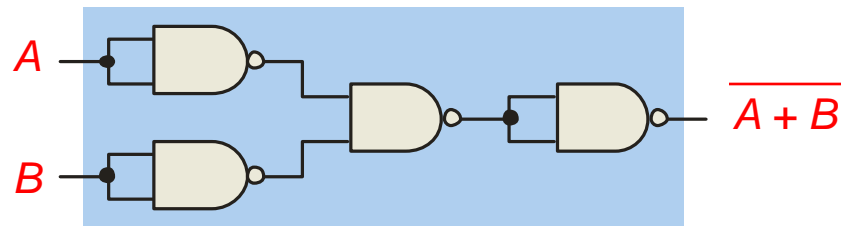
Inverter



AND gate



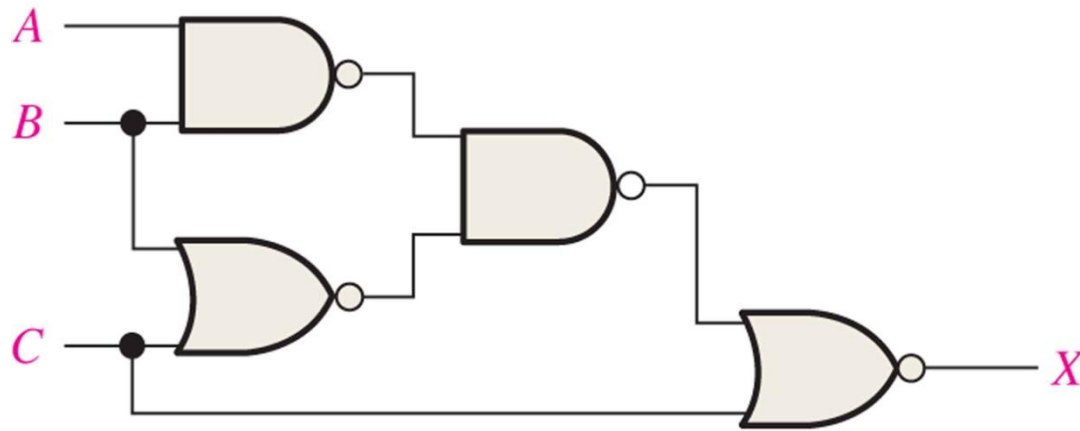
OR gate



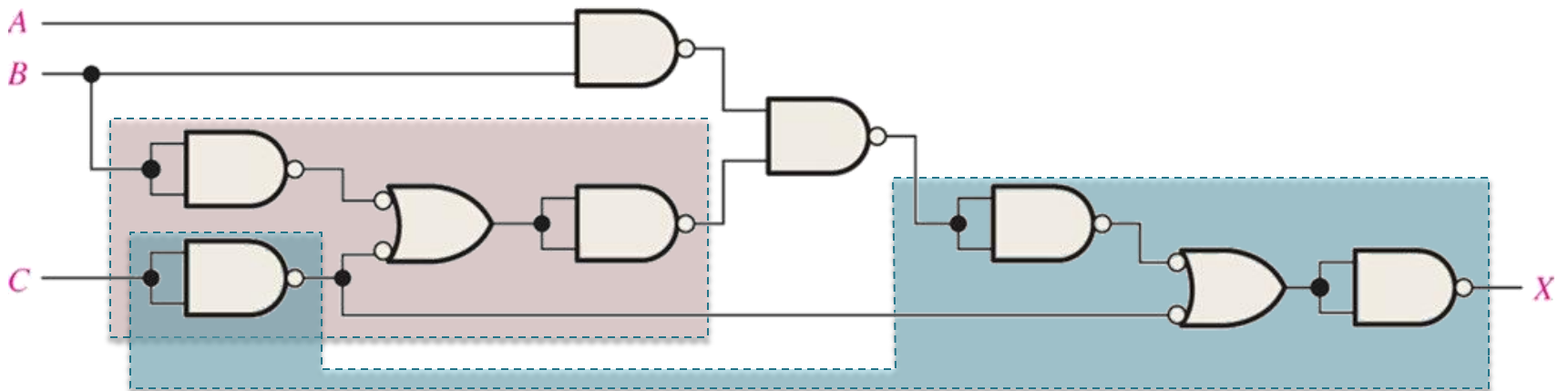
NOR gate

Example

Implement the following logic circuit using only NAND gates:

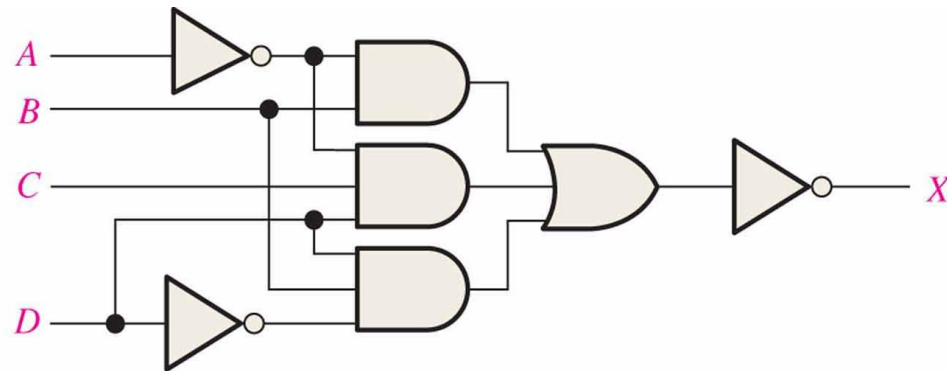


Solution:

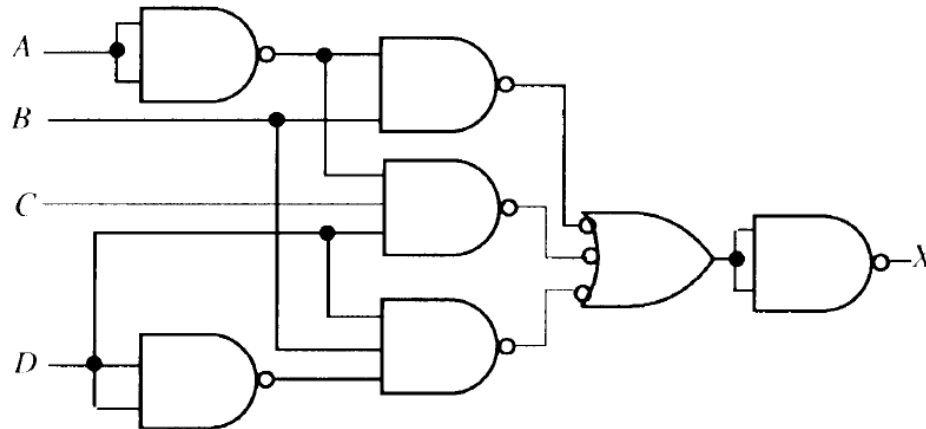


Example

Implement the following logic circuit using only NAND gates:

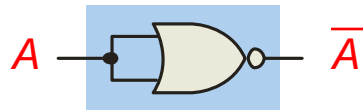


Solution:

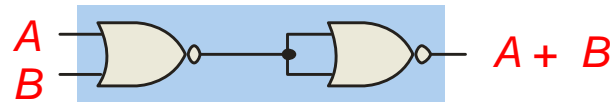


NOR Gate as a Universal Gate

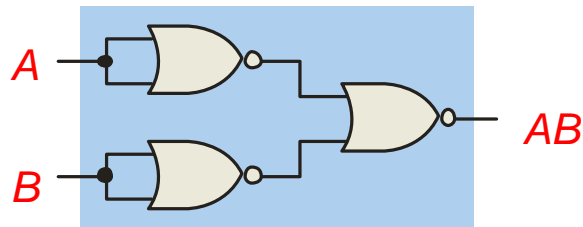
NOR gates are also **universal** gates and can form all of the basic gates.



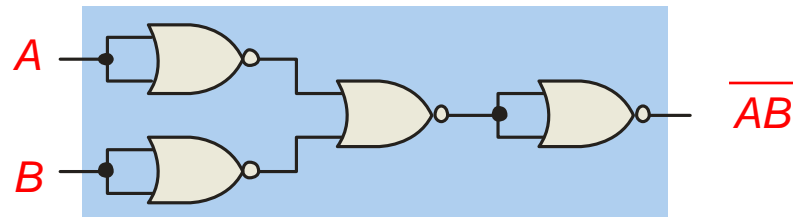
Inverter



OR gate



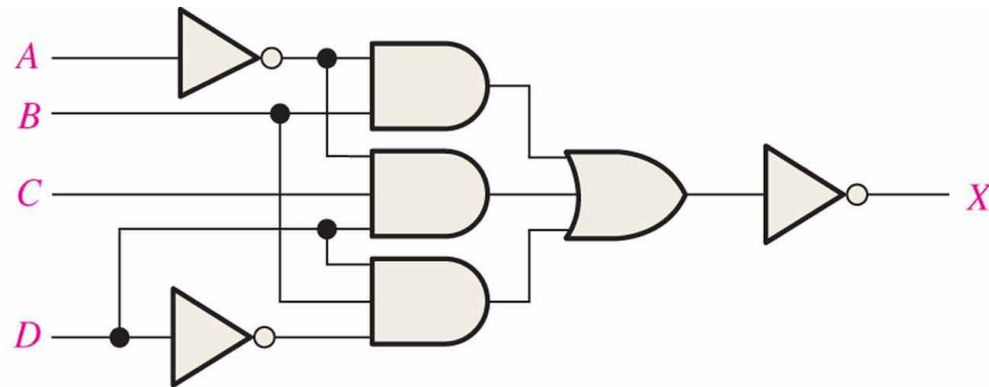
AND gate



NAND gate

Example

Implement the following logic circuit using only NOR gates:



Solution:

