Chapter 4:
Combinational Logic Circuits

Dr. Aloke Kumar Saha

Professor

Department of CSE, UAP

Chapter 4 Objectives

- *Selected areas covered in this chapter*:
  - Converting logic expressions to sum-of-products expressions.
  - Boolean algebra and the Karnaugh map as tools to simplify and design logic circuits.
  - Operation of exclusive-OR & exclusive-NOR circuits.
  - Designing simple logic circuits without a truth table.

4-1 Sum-of-Products Form

- A **Sum-of-products (SOP)** expression will appear as two or more **AND** terms **OR**ed together.

1. $ABC + \overline{A}B\overline{C}$
2. $AB + \overline{A}B\overline{C} + \overline{C}\,\overline{D} + D$
3. $\overline{A}B + C\overline{D} + EF + GK + H\overline{L}$

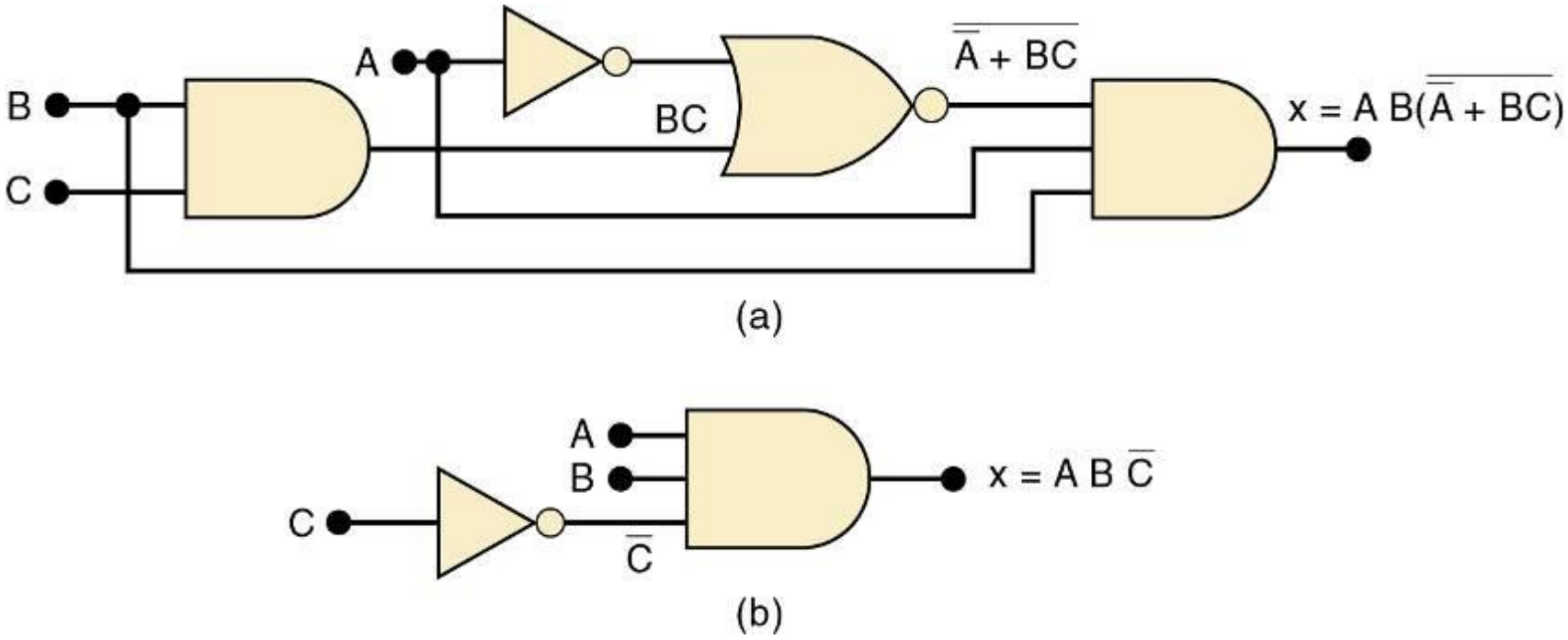
4-1 Sum-of-Products Form

- The **product-of-sums (POS)** form consists of two or more **OR** terms (sums) **AND**ed together.

1. $(A + \overline{B} + C)(A + C)$
2. $(A + \overline{B})(\overline{C} + D)F$
3. $(A + C)(B + \overline{D})(\overline{B} + C)(A + \overline{D} + \overline{E})$

4-2 Simplifying Logic Circuits

- The circuits shown provide the same output
  - Circuit (b) is clearly less complex.

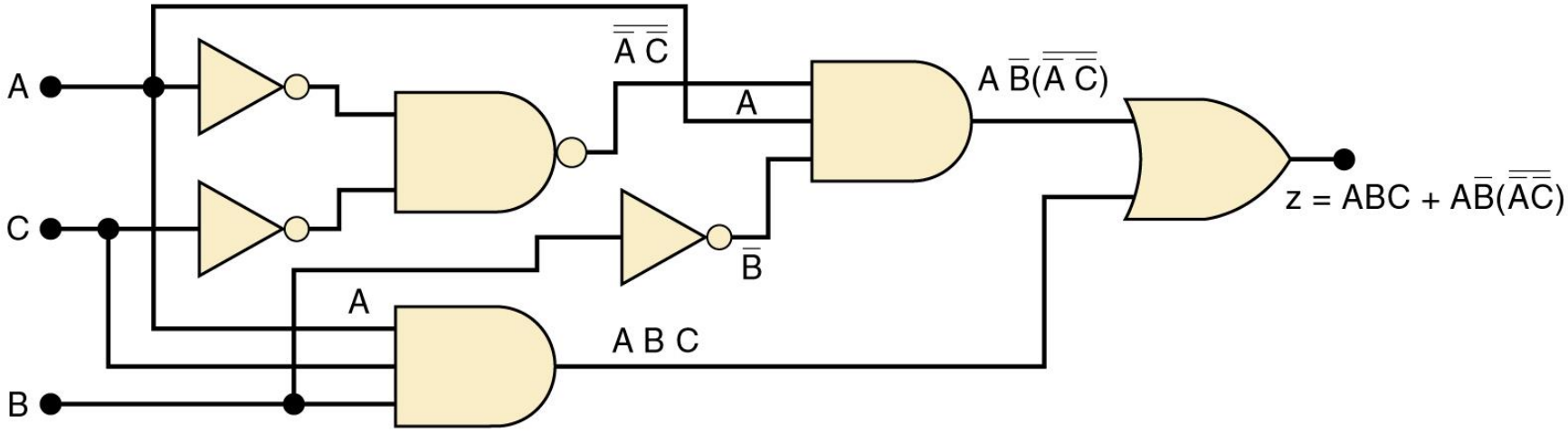


Logic circuits can be simplified using Boolean algebra and Karnaugh mapping.

4-3 Algebraic Simplification

- Place the expression in SOP form by applying DeMorgan's theorems and multiplying terms.
- Check the SOP form for common factors.
  - Factoring where possible should eliminate one or more terms.

Simplify the logic circuit shown.

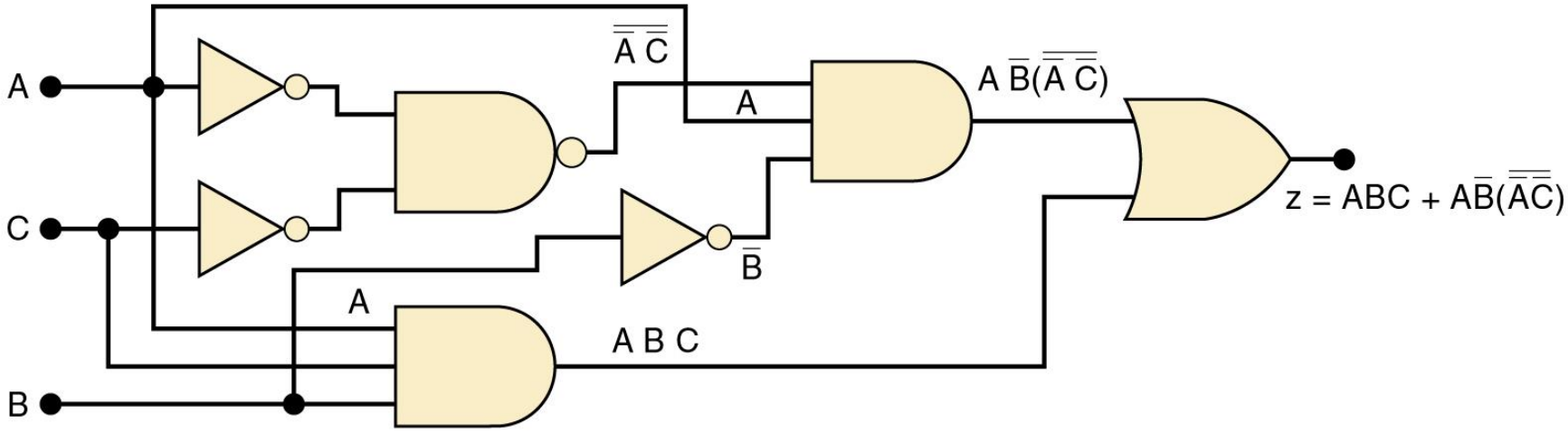


The first step is to determine the expression for the output: $z = ABC + A\overline{B} \cdot (\overline{\overline{A}\,\overline{C}})$

Once the expression is determined, break down large inverter signs by DeMorgan's theorems & multiply out all terms.

$$
\begin{aligned}
z &= ABC + A\overline{B}(\overline{\overline{A}} + \overline{\overline{C}}) && \text{[theorem (17)]}\\
&= ABC + A\overline{B}(A + C) && \text{[cancel double inversions]}\\
&= ABC + A\overline{B}A + A\overline{B}C && \text{[multiply out]}\\
&= ABC + A\overline{B} + A\overline{B}C && [A \cdot A = A]
\end{aligned}
$$

Simplify the logic circuit shown.



Factoring—the first & third terms above have ***AC*** in common, which can be factored out:

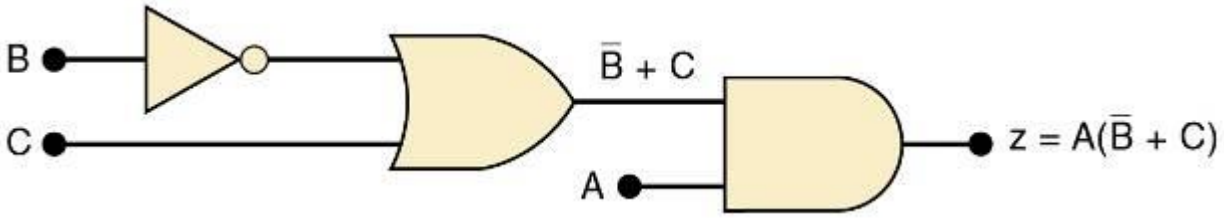
$$z = AC(B + \overline{B}) + A\overline{B}$$

Since ***B*** + ***B*** = 1, then…

$$\begin{aligned} z &= AC(1) + A\overline{B} \\ &= AC + A\overline{B} \end{aligned}$$

Factor out ***A***, which results in…

Simplifed logic circuit.

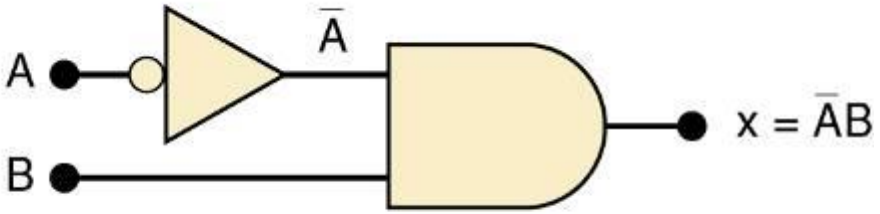


$$z = A(C + \overline{B})$$

- To solve any logic design problem:
  - Interpret the problem and set up its truth table.
  - Write the **AND** (product) term for each case where output = 1.
  - Combine the terms in SOP form.
  - Simplify the output expression if possible.
  - Implement the circuit for the final, simplified expression.

Circuit that produces a 1 output only for the A = 0, B = 1 condition.

A	B	x
0	0	0
0	1	1
1	0	0
1	1	0

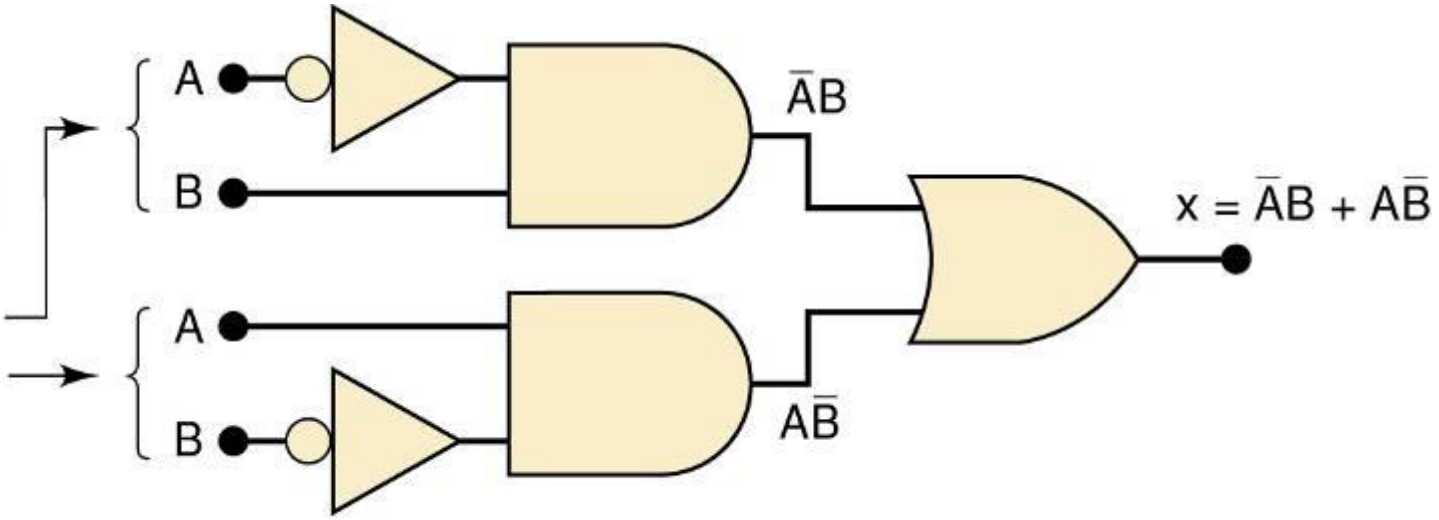


4-4 Designing Combinational Logic Circuits

Each set of input conditions that is to produce a 1 output is implemented by a separate **AND** gate.

The **AND** outputs are **OR**ed to produce the final output.

A	B	x
0	0	0
0	1	1
1	0	1
1	1	0



Truth table for a 3-input circuit.

A	B	C	x	
0	0	0	0	
0	0	1	0	
0	1	0	1	$\rightarrow \bar{A}B\bar{C}$
0	1	1	1	$\rightarrow \bar{A}BC$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	$\rightarrow ABC$

AND terms for each case where output is 1.

Design a logic circuit with three inputs, A, B, and C. Output to be HIGH only when a majority inputs are HIGH.

Truth table.

A	B	C	x
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

AND terms for each case where output is 1.

$\rightarrow \bar{A}BC$

$\rightarrow A\bar{B}C$

$\rightarrow AB\bar{C}$

$\rightarrow ABC$

SOP expression for the output:

$$x = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

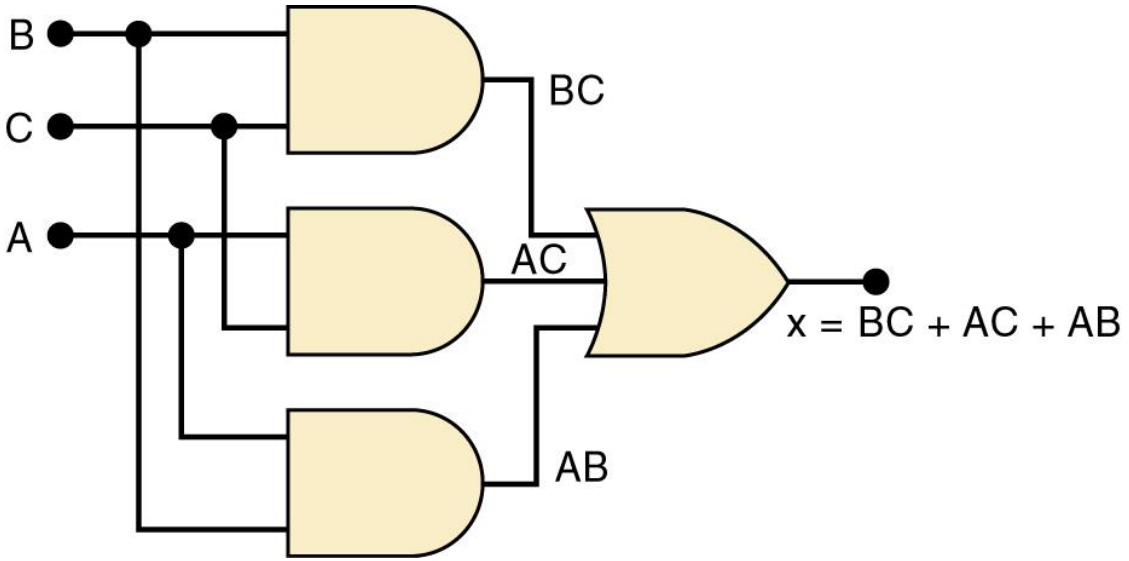
Design a logic circuit with three inputs, A, B, and C. Output to be HIGH only when a majority inputs are HIGH.

Simplified output expression:

$$x = ABC + ABC + ABC + ABC + ABC + ABC$$

Implementing the circuit after factoring:

$$x = BC + AC + AB$$



Since the expression is in SOP form, the circuit is a group of **AND** gates, working into a single **OR** gate,

4-5 Karnaugh Map Method

- A graphical method of simplifying logic equations or truth tables—also called a K map.
- Theoretically can be used for any number of input variables—practically limited to 5 or 6 variables.

The truth table values are placed in the K map. Shown here is a two-variable map.

A	B	X
0	0	$1 \rightarrow \overline{A}\,\overline{B}$
0	1	0
1	0	0
1	1	$1 \rightarrow AB$

$$x = \overline{A}\,\overline{B} + AB$$

	$\overline{B}$	B
$\overline{A}$	1	0
A	0	1

Four-variable K-Map.

A	B	C	D	X	
0	0	0	0	0	
0	0	0	1	1	$\rightarrow \bar{A}\bar{B}\bar{C}D$
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	$\rightarrow \bar{A}B\bar{C}D$
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	
1	1	0	1	1	$\rightarrow AB\bar{C}D$
1	1	1	0	0	
1	1	1	1	1	$\rightarrow ABCD$

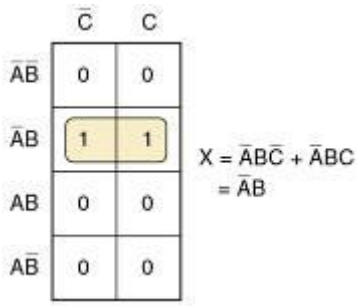
$$X = \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + AB\bar{C}D + ABCD$$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	1	0	0
AB	0	1	1	0
$A\bar{B}$	0	0	0	0

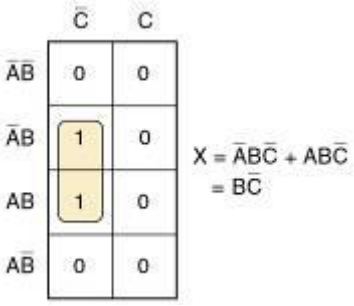
Adjacent K map square differ in only one variable both horizontally and vertically.

A SOP expression can be obtained by **OR**ing all squares that contain a 1.

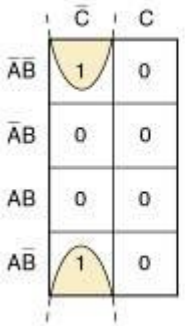
Looping 1s in adjacent groups of 2, 4, or 8 will result in further simplification.



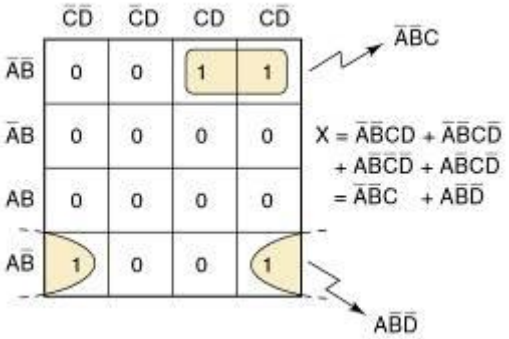
$X = A\bar{B}\bar{C} + \bar{A}BC$
$= \bar{A}B$



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



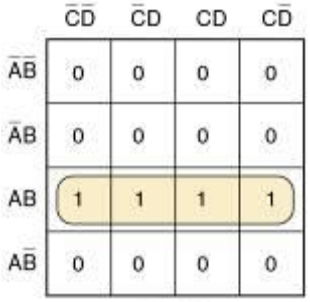
$X = A\bar{B}\bar{C} + AB\bar{C} = B\bar{C}$



$X = A\bar{B}CD + A\bar{B}C\bar{D}$
$+ AB\bar{C}D + AB\bar{C}\bar{D}$
$= A\bar{B}C + AB\bar{D}$

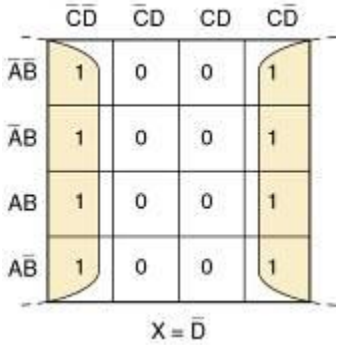
Looping groups of 2 (Pairs)

Groups of 4 (Quads)



$X = AB$

Groups of 8 (Octets)



$X = \bar{D}$

4-5 Karnaugh Map Method

- When the largest possible groups have been looped, only the common terms are placed in the final expression.
  - Looping may also be wrapped between top, bottom, and sides.

4-5 Karnaugh Map Method

- Complete K map simplification process:
  - Construct the K map, place 1s as indicated in the truth table.
  - Loop 1s that are not adjacent to any other 1s.
  - Loop 1s that are in pairs.
  - Loop 1s in octets even if they have already been looped.
  - Loop quads that have one or more 1s not already looped.
  - Loop any pairs necessary to include 1st not already looped.
  - Form the **OR** sum of terms generated by each loop.

When a variable appears in both complemented and uncomplemented form within a loop, that variable is eliminated from the expression.

Variables that are the same for all squares of the loop must appear in the final expression.

4-5 Algorithm for K-Map

- **Step 1** Construct the K map and place 1s in those squares corresponding to the 1s in the truth table. Place 0s in the other squares.
- **Step 2** Examine the map for adjacent 1s and loop those 1s that are not adjacent to any other 1s. These are called isolated 1s.
- **Step 3** Next, look for those 1s that are adjacent to only one other 1. Loop any pair containing such a 1.
- **Step 4** Loop any octet even if it contains some 1s that have already been looped.

4-5 Algorithm for K-Map

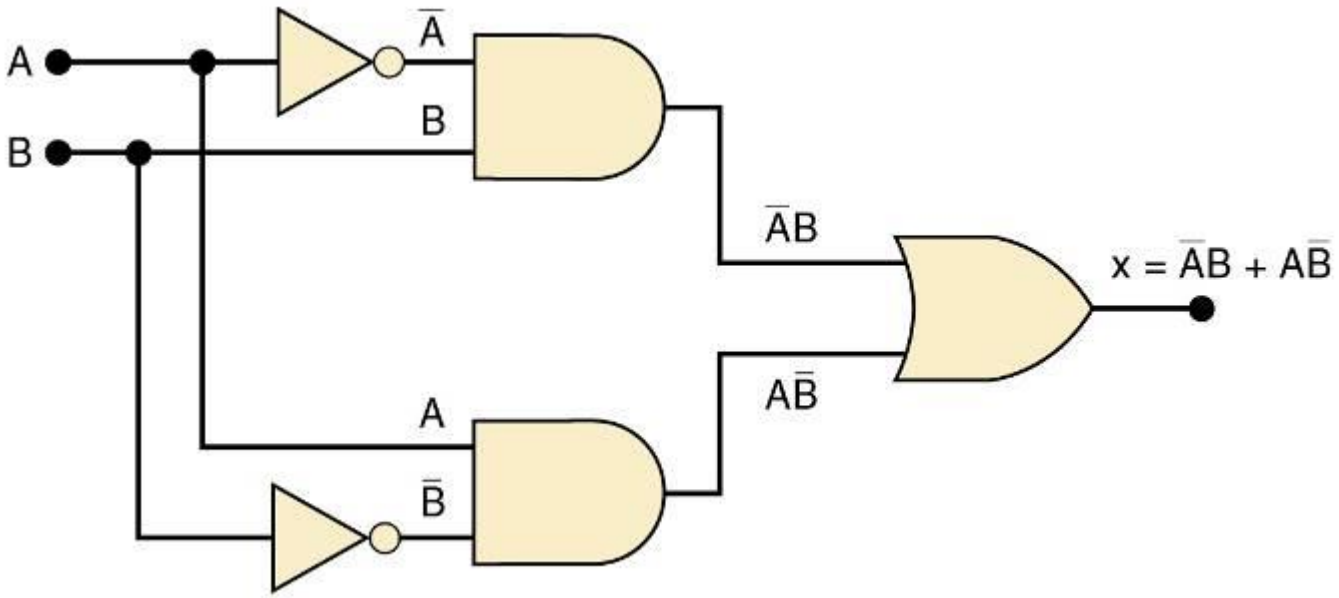
- Step 5 Loop any quad that contains one or more 1s that have not already been looped, making sure to use the minimum number of loops.
- Step 6 Loop any pairs necessary to include any 1s that have not yet been looped, making sure to use the minimum number of loops.
- Step 7 Form the OR sum of all the terms generated by each loop.

4-6 Exclusive OR and Exclusive NOR Circuits

- The exclusive **OR** (**XOR**) produces a HIGH output whenever the two inputs are at *opposite* levels.

4-6 Exclusive OR and Exclusive NOR Circuits

Exclusive **OR** circuit and truth table.

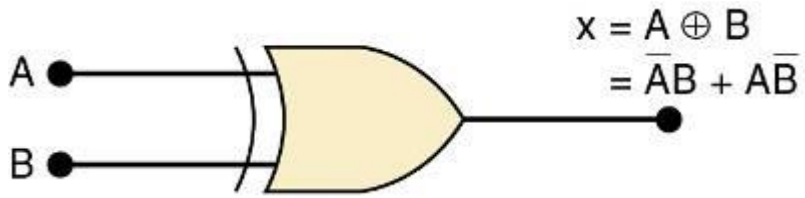


A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

Output expression: $x = \overline{A}B + A\overline{B}$

This circuit produces a HIGH output whenever the two inputs are at opposite levels.

Traditional **XOR** gate symbol.



An **XOR** gate has only *two* inputs, combined so that $x = \overline{A}B + A\overline{B}$.

A shorthand way indicate the **XOR** output expression is: $x = A \oplus B$.

…where the symbol ⊕ represents the **XOR** gate operation.

Output is HIGH only when the two inputs are at different levels.

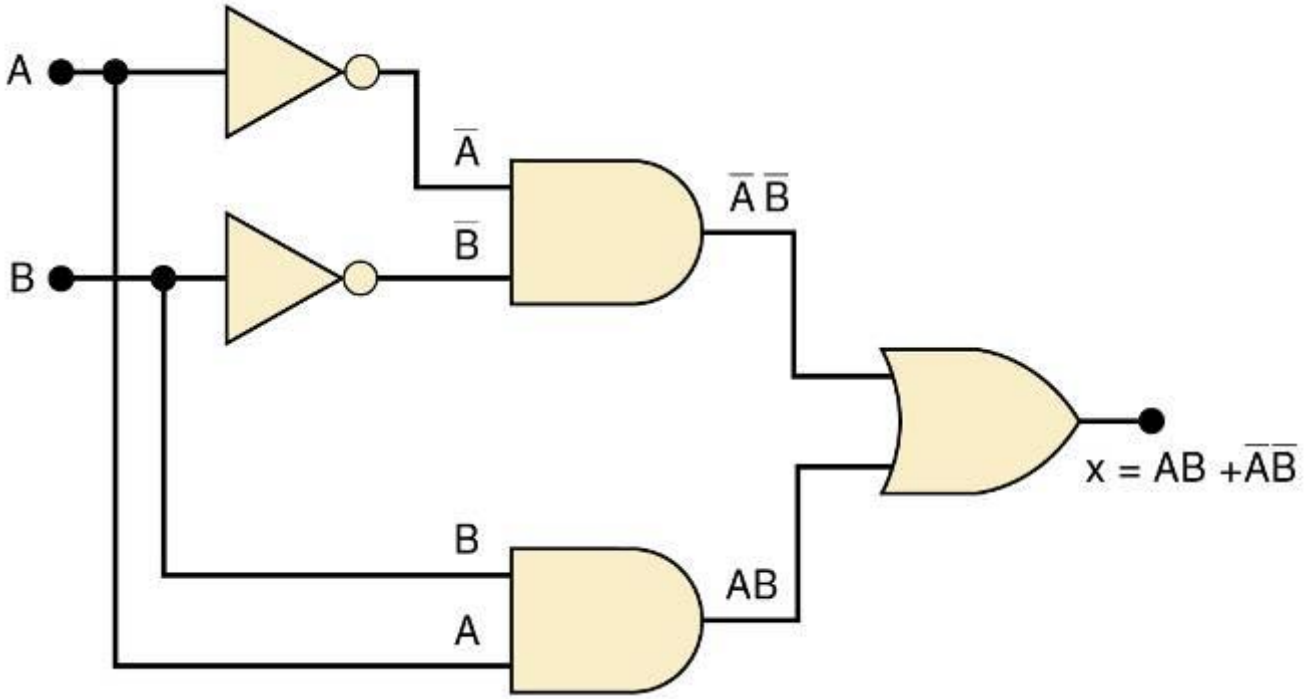
Quad XOR chips containing four XOR gates.

74LS86 Quad **XOR** (TTL family)
74C86 Quad **XOR** (CMOS family)
74HC86 Quad **XOR** (high-speed CMOS)

4-6 Exclusive OR and Exclusive NOR Circuits

- The exclusive **NOR** (**XOR**) produces a HIGH output whenever the two inputs are at the *same* level.
  - **XOR** and **XNOR** outputs are opposite.

Exclusive **NOR** circuit and truth table.

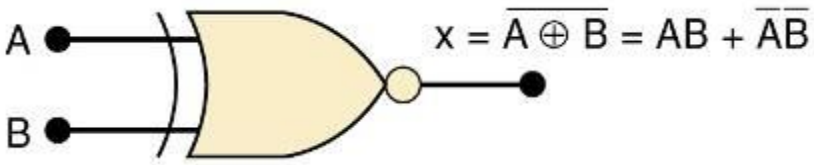


A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

Output expression: $x = AB + \overline{A}\,\overline{B}$

XNOR produces a HIGH output whenever the two inputs are at the same levels.

Traditional **XNOR** gate symbol.



An **XNOR** gate has only *two* inputs, combined so that $x = AB + \overline{A}\,\overline{B}$.

A shorthand way indicate the **XOR** output expression is: $x = \overline{A \oplus B}$.

XNOR represents inverse of the **XOR** operation.

Output is HIGH only when the two inputs are at the same level.

Quad XNOR chips with four XNOR gates.

74LS266 Quad **XNOR** (TTL family)
74C266 Quad **XOR** (CMOS)
74HC266 Quad **XOR** (high-speed CMOS)

4-7 More Examples about this chapter

Example 1 Design a logic circuit whose output is HIGH only when a majority of inputs are HIGH.

Example 2 Design a logic circuit whose output is HIGH only when a majority of inputs are LOW.

Example 3 Design a logic circuit that follows the following requirements:

a) Output X will equal A when B and C are the same.

b) X will remain HIGH when B and C are different.

4-7 More Examples about this chapter

Example 4 A four-bit binary number is represented as DCBA, where D, C, B, and A represent the individual bits and A is equal to the LSB. Design a logic circuit that will produce a HIGH output whenever the binary number is greater than 0010 and less than 1010.

Example 5 A 4-bit binary number is represented as D C B A, where D, C, B and A represent the individual bits with A equal to the LSB. Design a logic circuit that will produce a HIGH output whenever (the binary number is greater than 0011 and less than 1011) or (all inputs are LOW) or (all inputs are HIGH).

4-7 More Examples about this chapter

Example 6 Design a logic circuit which has four inputs A, B ,C , D and an output Y that is to be HIGH only when input A is HIGH at the same time that at least two other inputs are HIGH. Design the circuit.

Example 7 Implement the function F(A, B, C, D) = $\sum$(0, 1, 2, 3 , 6, 7, 9, 11, 14, 15) using K-map.

Example 8 Implement the function F(A, B, C, D) = $\sum$(0, 2, 3, 4, 8, 9, 12, 13, 14, 15) using K-map.

Example 9 Simplify the following expression using K map Y=ABCD+A`B`CD+ABC`D`+ABC+ AB +CD+A

	A`B`	AB`	AB	A`B
C`D`		1	1	
CD`		1	1	
CD	1	1	1	1
C`D		1	1	

Final Result:

Y=A+CD

Example 10 Simplify the following expression using K map Y=A`B`C`D`+C`D+AB`C+D`

	A`B`	AB`	AB	A`B
C`D`	1	1	1	1
CD`	1	1	1	1
CD		1		
C`D	1	1	1	1

Final Result:

Y=D`+C`+AB`

4-8 Indexing

	A`B`	AB`	AB	A`B
C`D`	0	1	3	2
CD`	4	5	7	6
CD	12	13	15	14
C`D	8	9	11	10

4-8 Indexing

	A`B`	AB`	AB	A`B
C`	0	1	3	2
C	4	5	7	6