Logic Operation Fundamentals

EIC 0844091

Digital Circuit and Logic Design

Associate Prof. Luo Jie

Huazhong University of Science & Technology

Presentation Outline

- Basic logic operation
- Boolean algebra
- Logical function expression
- Logic Simplification by Karnaugh Map

Basic logic operation

In digital circuits, the most basic elements of circuits are logic gates that can perform logic operations.

The three basic operations are AND, OR and NOT. We can use these basic operations to carry out very complex operations.

Basic logic operation

AND logic operation

The result is true none but the all of conditions are true.

For the case of two conditions

If using *A* and *B* to express two conditions, *L* to express the result.

Logic expression:

$$L = A \cdot B = AB$$

What is a truth table?

truth table

A	B	L
false	false	false
false	true	false
true	false	false
true	true	true

true = 1

false = 0

truth table

A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

The truth table is such a table that lists all possible combinations of all conditions and corresponding results

Basic logic operation

Any 0 then 0, all 1s then 1

□ AND logic operation

Logic expression:

$$L = A \cdot B = AB$$

truth table

A	B	L
false	false	false
false	true	false
true	false	false
true	true	true

true = 1

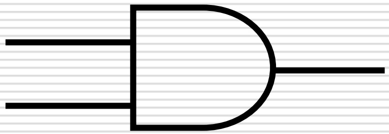
false = 0

truth table

A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

Logic symbol: (AND gate)

(inputs) A B



L (output)

(AND standard distinctive shape symbol)

true = high
false = low

high = 1
low = 0

truth table (voltage level)

A	B	L
low	low	low
low	high	low
high	low	low
high	high	high

The output is 0 as long as any input is 0

The output is 1 none but all of inputs are 1s

Basic logic operation

Any 0 then 0, all 1s then 1

AND logic operation

Logic expression:

$$L = A \cdot B = AB$$

truth table (voltage level)

A	B	L
low	low	low
low	high	low
high	low	low
high	high	high

high = 1

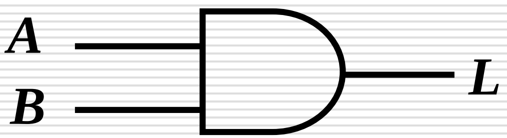
low = 0

truth table

A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

Logic symbol: (AND gate)

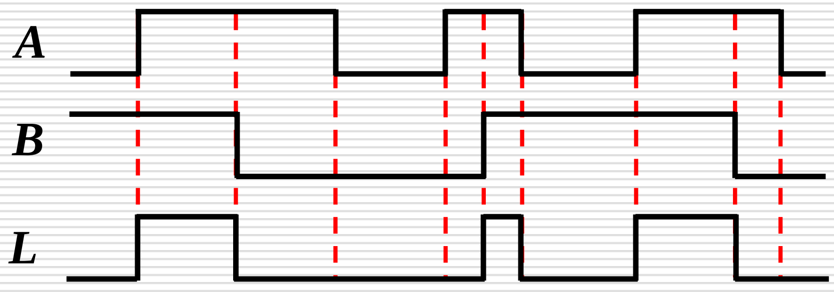
(inputs)



(output)

(AND standard distinctive shape symbol)

Pulsed operation

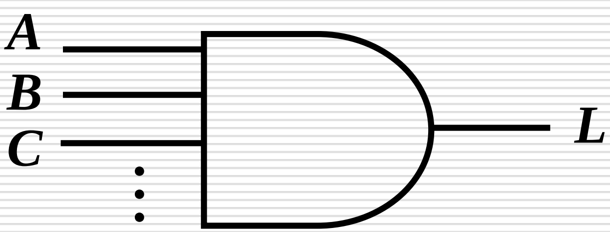


Basic logic operation

AND logic operation

For more then two inputs

Logic symbol:



Logic expression:

$$L = ABC\cdots$$

Basic logic operation

OR logic operation

The result is true as long as anyone of all conditions is true.

For the case of two conditions

Logic expression:

$$L = A + B$$

The plus sign indicates logic OR operation

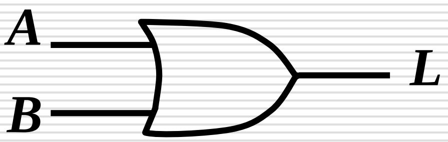
truth table

A	B	L
0	0	0
0	1	1
1	0	1
1	1	1

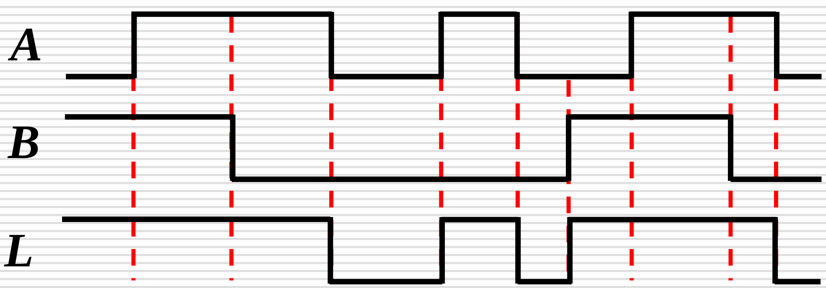
(true = 1, false = 0)

Any 1 then 1, all 0s then 0

Logic symbol: (OR gate)



Pulsed operation:

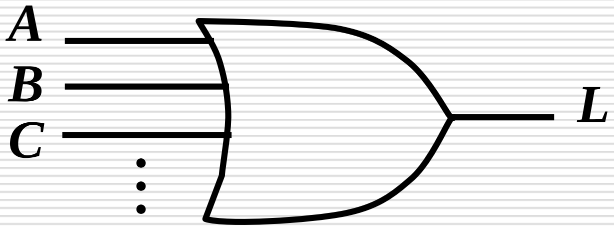


Basic logic operation

OR logic operation

For more then two inputs

Logic symbol:



Logic expression:

$$L = A + B + C + \cdots$$

Basic logic operation

NOT logic operation

The result and the condition are opposite.

Logic expression:

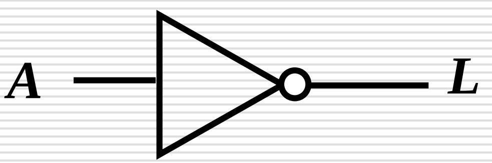
$$L = \overline{A}$$

truth table

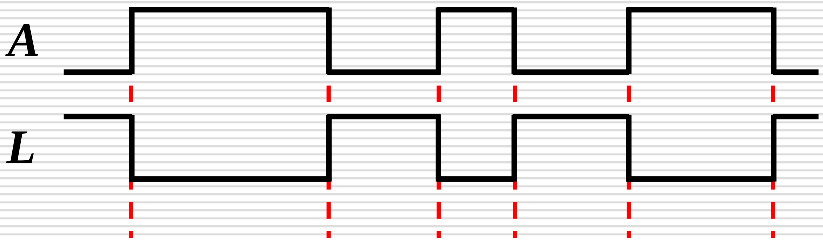
A	L
0	1
1	0

(true = 1, false = 0)

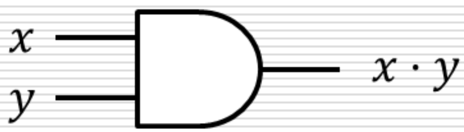
Logic symbol: (NOT gate)



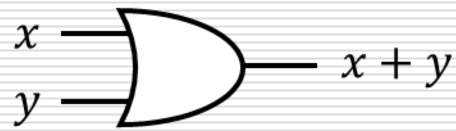
Pulsed operation:



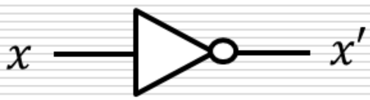
Additional Logic Gates and Symbols



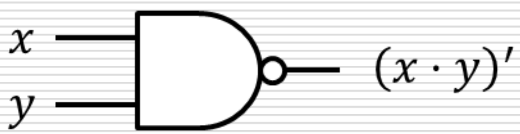
AND gate



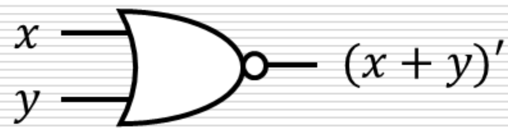
OR gate



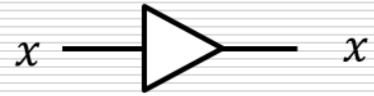
NOT gate (inverter)



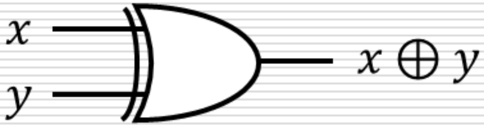
NAND gate



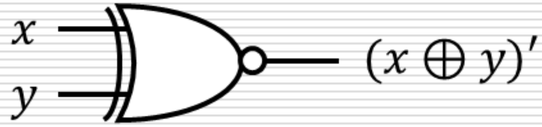
NOR gate



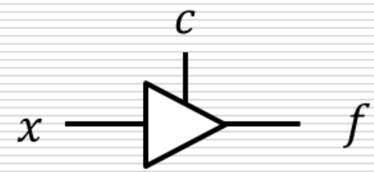
Buffer



XOR gate



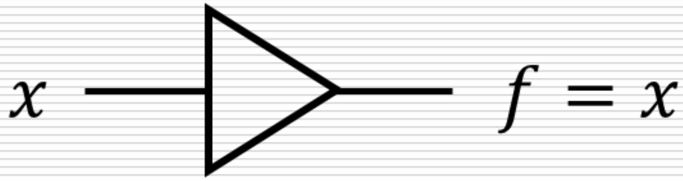
XNOR gate



3-state gate

Buffer

- A buffer is a gate with the function $f = x$



x	f
0	0
1	1

- ***As a Boolean function, a buffer is like a connection!***
- ***So why use it?***
  - A buffer is used to amplify an input signal
  - Permits more gates to be attached to output
  - Also, increases the speed of circuit operation

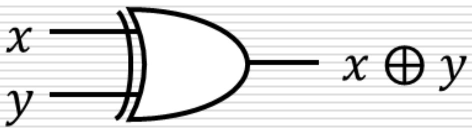
Exclusive OR / Exclusive NOR

- Exclusive OR (XOR) is an important Boolean operation used extensively in logic circuits
- Exclusive NOR (XNOR) is the complement of XOR

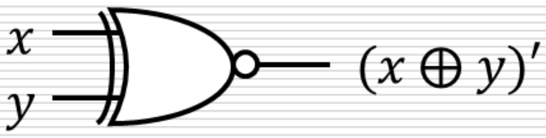
x y	XOR
0 0	0
0 1	1
1 0	1
1 1	0

x y	XNOR
0 0	1
0 1	0
1 0	0
1 1	1

XNOR is also known as **equivalence**



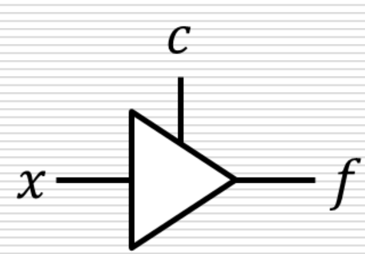
XOR gate



XNOR gate

3-State Gate

- Logic gates studied so far have two outputs: 0 and 1
- Three-State gate has three possible outputs: 0, 1, Z
  - **Z** is the **Hi-Impedance** output
  - **Z** means that the output is **disconnected** from the input
  - Gate behaves as an **open switch** between input and output
- Input *c* connects input to output
  - ***c*** is the control (enable) input
  - If ***c*** is **0** then f = Z
  - If ***c*** is **1** then f = input x

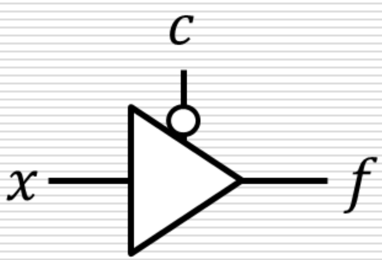


3-state gate

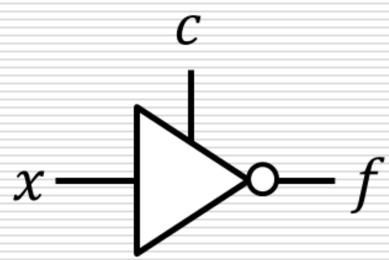
c	x	f
0	0	Z
0	1	Z
1	0	0
1	1	1

3-State Gate

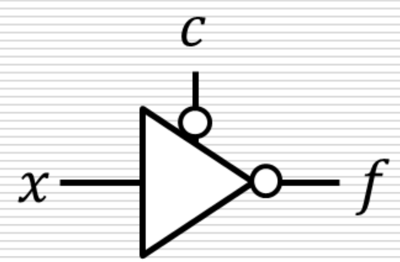
- Control input c and output f can be inverted
- A bubble is inserted at the input c or output f



inverted *c*



inverted *f*



inverted *c*, *f*

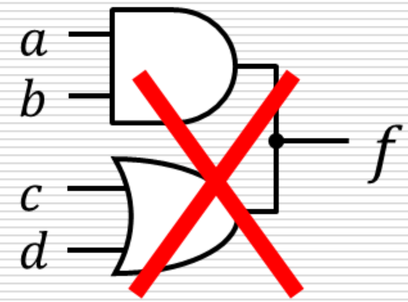
c	x	f
0	0	0
0	1	1
1	0	Z
1	1	Z

c	x	f
0	0	Z
0	1	Z
1	0	1
1	1	0

c	x	f
0	0	1
0	1	0
1	0	Z
1	1	Z

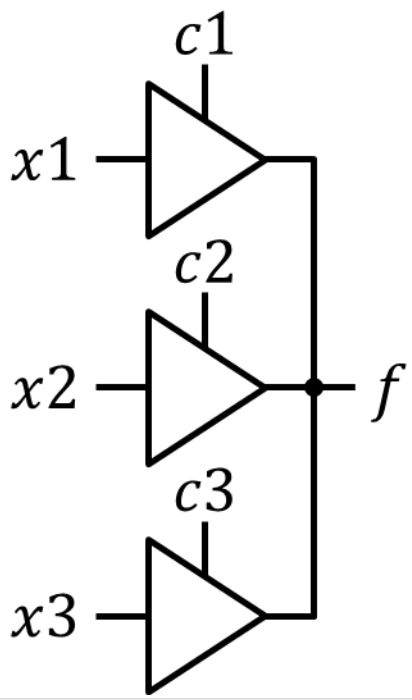
Wired Output

Logic gates with 0 and 1 outputs **cannot** have their outputs wired together



This will result in a **short circuit** that will burn the gates

3-state gates **can wire** their outputs together



c1	c2	c3	f
0	0	0	Z
1	0	0	x1
0	1	0	x2
0	0	1	x3
0	1	1	Burn
1	0	1	Burn
1	1	0	Burn
1	1	1	Burn

At most one 3-state gate can be enabled at a time

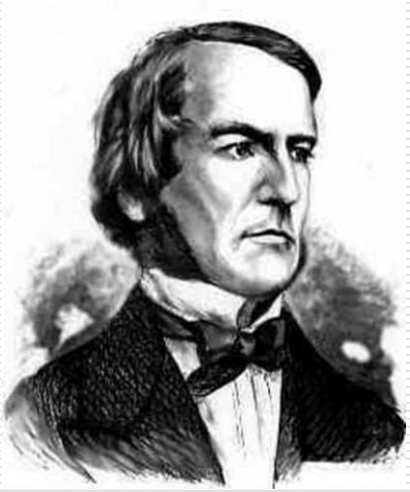
Otherwise, conflicting outputs will burn the circuit

The topic we'll discuss

- Basic logic operation
- Boolean algebra
- Logical function expression
- Logic Simplification by Karnaugh Map

Boolean algebra

In 1854, the logical algebra known today as Boolean algebra was founded and formulated in publication written by George Boole.



George Boole

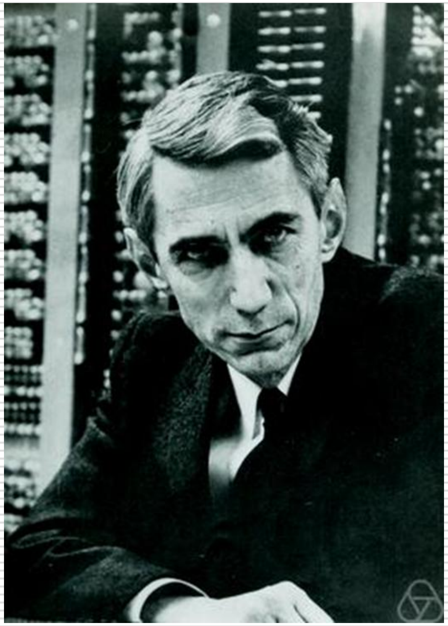
(1815.11~1864.12)

位于爱尔兰科克

的布尔之墓

Boolean algebra

In 1938, Claude Shannon was the first to apply Boole's work to the analysis and design of logic circuits.



Claude E. Shannon
(1916.4 ~ 2001.2)

在**1937**年，他完成硕士学位论文：*A Symbolic Analysis of Relay and Switching Circuits*，
1938年发表了该论文。

Boolean algebra

Boolean algebra is the mathematics of digital systems.

A basic knowledge of Boolean algebra is indispensable （必需） to the study and analysis of logic circuits.

It is a convenient and systematic way of expressing and analyzing the operation of logic circuits.

The basic knowledge of Boolean algebra includes 3 laws, 12 basic rules and 2 theorems (DeMorgan’s theorems)

Laws of Boolean algebra

- Commutative laws（交换律）

$$A + B = B + A \qquad A \cdot B = B \cdot A$$

- Associative laws（结合 律）

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

- Distributive law（分配律）

$$A(B + C) = AB + AC$$

Here, each of the laws is illustrated with two or three variables, but the number of variables is not limited to this.

Rules of Boolean algebra

- 12 basic rules

1. $A+0=A$	7. $A\cdot A=A$
2. $A+1=1$	8. $A\cdot\overline{A}=0$
3. $A\cdot 0=0$	9. $\overline{\overline{A}}=A$
4. $A\cdot 1=A$	10. $A+AB=A$
5. $A+A=A$	11. $A+\overline{A}B=A+B$
6. $A+\overline{A}=1$	12. $(A+B)(A+C)=A+BC$

***A*, *B*, or *C* can represent a single variable or a combination of variables.**

Rules of Boolean algebra

- 12 basic rules

Prove the rule 11 $A+\overline{A}B=A+B$

Solution

$$
\begin{aligned}
A+\overline{A}B &= A\cdot 1+\overline{A}B && A\cdot 1=A \\
&= A(1+B)+\overline{A}B && A+1=1 \\
&= A+AB+\overline{A}B && \text{Distributive law} \\
&= A+(A+\overline{A})B && \text{Distributive law} \\
&= A+1\cdot B && A+\overline{A}=1 \\
&= A+B && A\cdot 1=A
\end{aligned}
$$

Theorems of Boolean algebra

(DeMorgan's theorems)

- The complement of a product of variables is equal to the sum of the complements of the variables.

$$\overline{ABC\cdots} = \overline{A} + \overline{B} + \overline{C} + \cdots$$

- The complement of a sum of variables is equal to the product of the complements of the variables.

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

Each variable can also represent a combination of other variables.

Example

$$\overline{AB(C+D)\cdots} = \overline{A} + \overline{B} + \overline{C+D} + \cdots = \overline{A} + \overline{B} + \overline{C} \cdot \overline{D} + \cdots$$

$$\overline{A+B+CD+\cdots} = \overline{A} \cdot \overline{B} \cdot \overline{CD} \cdots = \overline{A} \cdot \overline{B} \cdot (\overline{C} + \overline{D}) \cdots$$

Theorems of Boolean algebra

- (DeMorgan's theorems)

Prove them by the truth table (two variables)

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

Prove:

A	B
0	0
0	1
1	0
1	1

DeMorgan's theorems are often applied to logical function transformation and simplification

Two columns are equivalent that mines: $\overline{AB} = \overline{A} + \overline{B}$

The topic we'll discuss

- Basic logic operation
- Boolean algebra
- Logical function expression
- Logic Simplification by Karnaugh Map

Logical function expression

- Standard forms of Boolean expressions (include SOP, standard SOP, POS, and standard POS)
- Conversion of Boolean expressions
- Expressing of logical functions

Logical function expression

Standard forms of Boolean expressions

All expressions can be converted into the Sum-of-Products form or the Product-of-Sums form. It makes the evaluation, simplification, and implementation of Boolean expressions much more systematic and easier.

The Sum-of-Products form abbreviated SOP

The Product-of-Sums form abbreviated POS

Standard forms of Boolean expressions

The SOP form (Sum-of-Products)

Two or more product terms are summed by Boolean addition.

Which belong to SOP form?

$AB + ACD$ √ $\quad A\overline{B} + \overline{A}\,\overline{C}\,\overline{D}$ √ $\quad \overline{A}(B+C)$ ×

$\overline{AB} + \overline{A}CD$ × $\quad \overline{A} + B + C$ √

! A single overbar cannot extend over than one variable.

! A product term can be only one variable.

Are these product terms?

$A\overline{BC}$ × $\quad \overline{AB+C}$ × $\quad \overline{\overline{A}BC}$ × $\quad \overline{A}\,\overline{B}\,\overline{C}$ √

Standard forms of Boolean expressions

The standard SOP form (sum of minterms)

A standard SOP expression is one in which **all the variables** in the domain **appear in each product term** in the expression.

The domain is the set of variables contained in the expression in either original or complemented form.

Such product term is called standard product term or minterm

Which are standard SOP form?

$AB+AC$ ✗ $A\overline{B}C+\overline{A}B\overline{C}+ABC$ √

$A\overline{B}CD+\overline{A}\,\overline{B}CD+A\overline{BCD}$ ✗ $A\overline{B}CD+\overline{A}\,\overline{B}CD+\overline{A}BC\overline{D}$ √

! **Each variable must appear once in each product term with the original variable or the complementary variable (inverse variable).**

Standard forms of Boolean expressions

- The speciality of the standard product terms (minterms)
  - If the number of variables in the domain is n, the number of standard product terms is 2^n.
  - A standard product term equal to 1 for only one combination of variables values. （变量取值只有一种组合，使标准乘积项等于1）

Example For a standard product term $\bar{A}BC\bar{D}$

Here, n = 4, the number of standard product terms is 16.

And only when $A = 0,\ B = 1,\ C = 1,\ D = 0$

Then $\bar{A}BC\bar{D} = \bar{0}\cdot 1\cdot 1\cdot \bar{0} = 1\cdot 1\cdot 1\cdot 1 = 1$

Standard forms of Boolean expressions

- The speciality（特点） of the standard product terms (minterms)

Truth table of all standard product terms for three variables

A	B	C	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}\,\overline{B}C$	$\overline{A}B\overline{C}$	$\overline{A}BC$	$A\overline{B}\,\overline{C}$	$A\overline{B}C$	$AB\overline{C}$	ABC
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

Standard forms of Boolean expressions

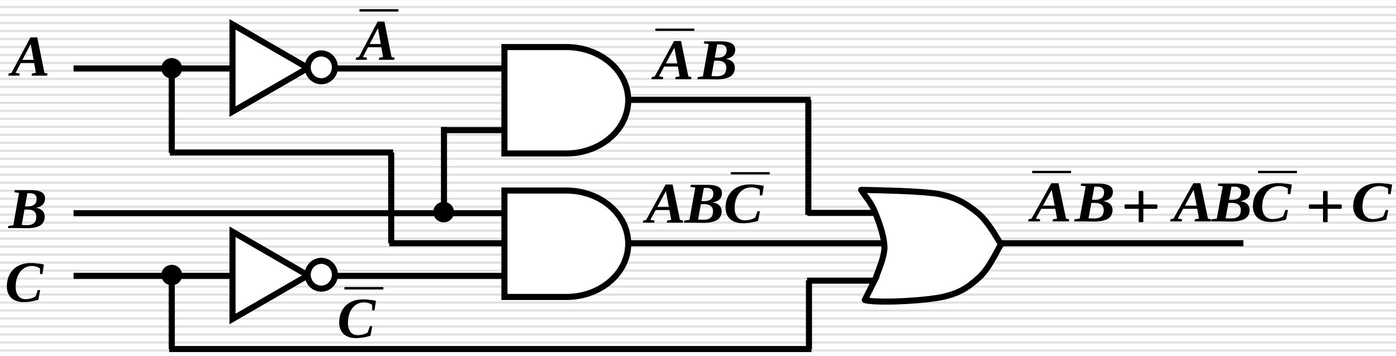
Implementation of an SOP expression

For product operation using AND gate

For sum operation using OR gate

For complement (overbar) operation using NOT gate

Example Implementing an SOP expression $\overline{A}B + AB\overline{C} + C$



Logical function expression

- Standard forms of Boolean expressions (include SOP, standard SOP, POS, and standard POS)
- Conversion of Boolean expressions
- Expressing of logical functions

Conversion of Boolean expression

- Convert each of the following expression to SOP form:

$$(a)\quad (A+B)(\overline{B}+C+D) \qquad\qquad (b)\quad \overline{\overline{A+B}+C}$$

Solution

$$(a)\quad (A+B)(\overline{B}+C+D)$$

$$= A\overline{B}+AC+AD+BC+BD$$

(Distributive law)

$$(b)\quad \overline{\overline{A+B}+C}$$

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

(DeMorgan's theorems)

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

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

> The same logical function can be expressed many different forms of Boolean expressions

- Continuously convert them to standard SOP form:

Conversion of Boolean expression

- Continuously convert them to standard SOP form:

Solution

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

$$= A\overline{C}\cdot 1 + B\overline{C}\cdot 1 \qquad (A\cdot 1 = A)$$

$$= A\overline{C}\cdot(B+\overline{B}) + B\overline{C}(A+\overline{A}) \qquad (A+\overline{A}=1)$$

$$= AB\overline{C} + A\overline{B}\,\overline{C} + AB\overline{C} + \overline{A}B\overline{C} \qquad \text{(Distributive low)}$$

Standard forms of Boolean expressions

- The POS form (Product-of-Sums) and standard POS form (product of maxterms)

Learn it by yourself

Logical function expression

- Standard forms of Boolean expressions (include SOP, standard SOP, POS, and standard POS)
- Conversion of Boolean expressions
- Expressing of logical functions

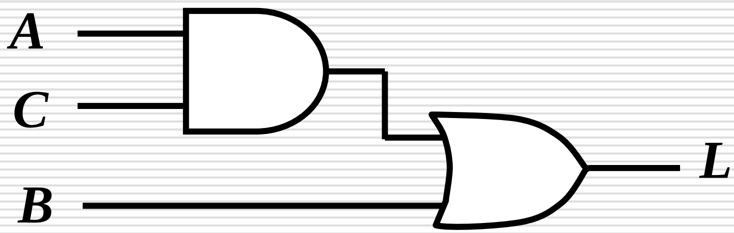
Expressing of logical functions

Three ways to express a logical function

1. Logical function expressions

$$L = AC + B$$

2. Logic circuit (logic diagram)



Note:
In truth table, all possible combinations of values for the input variables should be listed.

3. Truth table

inputs			output
A	*B*	*C*	*L*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

It is often needed to convert between these three formats

Expressing of logical functions

Converting SOP expressions to truth table format

Example develop a truth table for the standard SOP expression

$$L = \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + ABC$$

Solution 3 input variables

8 possible combinations

$\overline{A}\,\overline{B}C = 1 \qquad L = 1$

$A\overline{B}\,\overline{C} = 1 \qquad L = 1$

$ABC = 1 \qquad L = 1$

Truth table

inputs			output
A	*B*	*C*	*L*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Expressing of logical functions

Converting SOP expressions to truth table format

Example develop a truth table for the SOP expression

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

$$\text{SOP} \longrightarrow \text{Standard SOP} \longrightarrow \text{Truth table}$$

$$\text{SOP} \xrightarrow{?} \text{Truth table}$$

Solution

As long as $A\overline{C} = 1$

Namely $A = 1$ and $C = 0$

Then $L = 1$

As long as $B = 1$

Then $L = 1$

Truth table

inputs			output
A	B	C	L
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Expressing of logical functions

Converting truth table format to logic circuit

Step: Truth table → Standard SOP → SOP → Logic circuit

Example: draw a logic circuit from the truth table

Truth table

inputs			output
A	B	C	L
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Solution Determine the standard products corresponding to $L=1$

$$\bar{A}B\bar{C} \quad \bar{A}BC \quad A\bar{B}C \quad AB\bar{C} \quad ABC$$

Standard SOP

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

Get SOP by simplifying

Expressing of logical functions

Converting truth table format to logic circuit

Solution continuously

(Distributive law)

Get SOP by simplifying $L = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

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

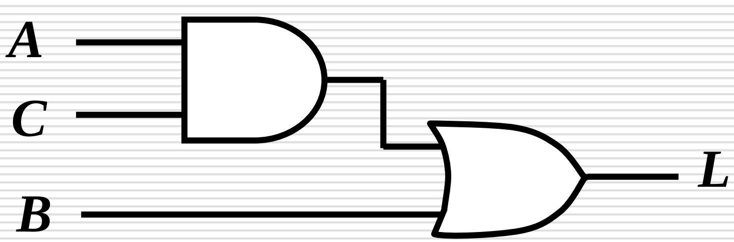
$= \bar{A}B + A\bar{B}C + AB$ $\quad (\bar{A} + A = 1,\ A \cdot 1 = A)$

$= (\bar{A} + A)B + A\bar{B}C$ $\quad$ (Distributive law)

$= B + A\bar{B}C$ $\quad (\bar{A} + A = 1,\ A \cdot 1 = A)$

$= B + AC$ $\quad (A + \bar{A}B = A + B)$

Logic circuit



The simplified expression can make logic circuit simpler

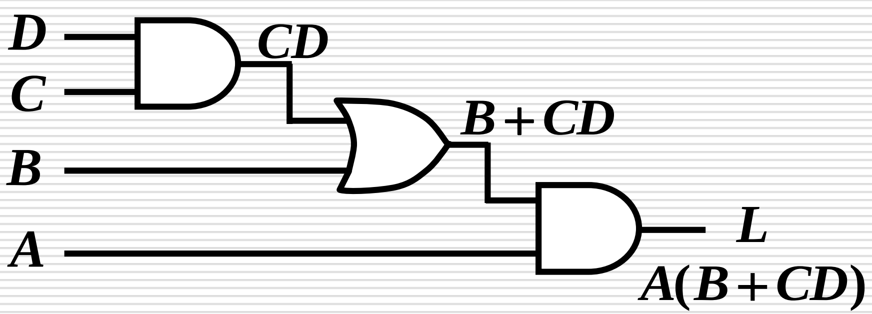
The simplification looks like difficult to you, because it is difficult to determine if the result is simplest. It relies on your skill of Boolean algebra.

Expressing of logical functions

- Converting logic circuit to truth table format

Step: Logic circuit → SOP → Truth table

Example: Determine a truth table for the logic circuit



Solution

$$L = A(B+CD) = \underline{AB} + \underline{ACD}$$

Truth table

$ABCD$	L
0000	0
0001	0
0010	0
0011	0
0100	0
0101	0
0110	0
0111	0
1000	0
1001	0
1010	0
1011	1
1100	1
1101	1
1110	1
1111	1

Logical function expression

- Standard forms of Boolean expressions (include SOP, standard SOP, POS, and standard POS)
- Conversion of Boolean expressions
- Expressing of logical functions

(Review this section)

The topic we'll discuss

- Basic logic operation
- Boolean algebra
- Logical function expression
- Logic Simplification by Karnaugh Map

Logic Simplification by Karnaugh Map

- Karnaugh Map
- Mapping a logic function on K-Map
- Karnaugh Map Simplification

Karnaugh Map

- Karnaugh Map

A Karnaugh Map is a graphical representation of a logic function's truth table. Abbreviation K-Map.

- It is a graphical style.
- It can represent a logic function.
- It is similar to a truth table.

Usually, K-Map is used for expressions with three and four variables.

Karnaugh Map

The 3-variables K-Map

The number of cells in a K-Map is equal to the total number of possible input variable combinations as is the number of rows in a truth table. Here, the number of cells is $2^3=8$.

$AB \backslash C$	0	1
00		
01		
11		
10		

(Notice the sequence)

$AB \backslash C$	0	1
00	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$
01	$\bar{A}B\bar{C}$	$\bar{A}BC$
11	$AB\bar{C}$	ABC
10	$A\bar{B}\bar{C}$	$A\bar{B}C$

Standard product terms

Truth table

$A\ B\ C$	Z
0 0 0	
0 0 1	
0 1 0	
0 1 1	
1 0 0	
1 0 1	
1 1 0	
1 1 1	

Karnaugh Map

- The 4-variables K-Map

The number of cells is $2^4=16$.

AB \ CD	00	01	11	10
00				
01				
11				
10				

AB \ CD	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}\bar{B}C\bar{D}$
01	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
11	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
10	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

Truth table

ABCD	Z
0 0 0 0	
0 0 0 1	
0 0 1 0	
0 0 1 1	
0 1 0 0	
0 1 0 1	
0 1 1 0	
0 1 1 1	
1 0 0 0	
1 0 0 1	
1 0 1 0	
1 0 1 1	
1 1 0 0	
1 1 0 1	
1 1 1 0	
1 1 1 1	

Karnaugh Map

Cell adjacency（单元格相邻性）

Why are the cells arranged like this sequence?

To ensure that there is only a single-variable change between adjacent cells.

Adjacency is defined by a single-variable change.

$AB \backslash CD$	00	01	11	10
00	$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$	$\overline{A}\,\overline{B}\,\overline{C}D$	$\overline{A}\,\overline{B}CD$	$\overline{A}\,\overline{B}C\overline{D}$
01	$\overline{A}B\overline{C}\,\overline{D}$	$\overline{A}B\overline{C}D$	$\overline{A}BCD$	$\overline{A}BC\overline{D}$
11	$AB\overline{C}\,\overline{D}$	$AB\overline{C}D$	$ABCD$	$ABC\overline{D}$
10	$A\overline{B}\,\overline{C}\,\overline{D}$	$A\overline{B}\,\overline{C}D$	$A\overline{B}CD$	$A\overline{B}C\overline{D}$

“wrap-around” adjacency

（循环相邻）

Next item we are going to discuss

- Karnaugh Map
- Mapping a logic function on K-Map
- logic Simplification by Karnaugh Map

Mapping a logic function on K-Map

- Mapping directly from a truth table

Z

AB \ C	0	1
00	1	
01		
11	1	1
10	1	

Truth table

A	B	C	Z
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Mapping a logic function on K-Map

Mapping a standard SOP expression

For a standard SOP expression, the 1 is placed on K-Map for each product term in the expression.

Each 1 is placed in a corresponding to the value of a product term.（每个1被放置在对应于乘积项的值的位置。）

Example :

$$L = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C}$$

001　　010　　110

L: AB \ C	0	1
00		1
01	1	
11	1	
10		

Mapping a logic function on K-Map

Mapping a nonstandard SOP expression

We can map a nonstandard SOP expression on a K-Map like filling it in a truth table.

Example :

$$L = \overline{A} + A\overline{B} + AB\overline{C}$$

Truth table

$A\ B\ C$	L
0 0 0	1
0 0 1	1
0 1 0	1
0 1 1	1
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	0

L $AB \backslash C$	0	1
00	1	1
01	1	1
11	1	
10	1	1

Mapping a logic function on K-Map

Mapping a nonstandard SOP expression

Example :

$$L = A + \overline{C}D + AC\overline{D} + \overline{A}BC\overline{D}$$

L

AB \ CD	00	01	11	10
00		1		
01		1		1
11	1	1	1	1
10	1	1	1	1

Next item we are going to discuss

- Karnaugh Map
- Mapping a logic function on K-Map
- Karnaugh Map Simplification

Karnaugh Map Simplification

The goal

The goal of using K-Map is to simplify logic functions.

The process that results in an expression containing the fewest possible terms with the fewest possible variables is called ***minimization*（最小化）**.

When we obtain a minimum SOP expression, we can use fewest possible logic gates to perform this logic function.
（当我们得到最小**SOP**表达式时，我们可以使用尽可能少的逻辑门来实现这个逻辑功能）

Karnaugh Map Simplification

Simplifying principle with K-Map

Grouping any adjacent 2^i （$i = 1, 2, 3, \cdots ; n$） cells in the K-Map which contain 1s will eliminate i variables.

Group 1 creates the product term is

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

$$= \bar{A}BD$$

Group 2 creates the product term is

$$ABCD + ABC\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D}$$

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

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

$$= AC(B + \bar{B})$$

$$= AC$$

AB \ CD	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}\bar{B}C\bar{D}$
01	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$ 1	$\bar{A}BCD$ 1	$\bar{A}BC\bar{D}$
11	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$ 1	$ABC\bar{D}$ 1
10	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$ 1	$A\bar{B}C\bar{D}$ 1

Group 1

Group 2

Karnaugh Map Simplification

- The steps of K-Map simplification
  - Grouping the 1s
  - Determining the product term for each group
  - Summing the resulting product terms

Grouping the 1s（对1分组）

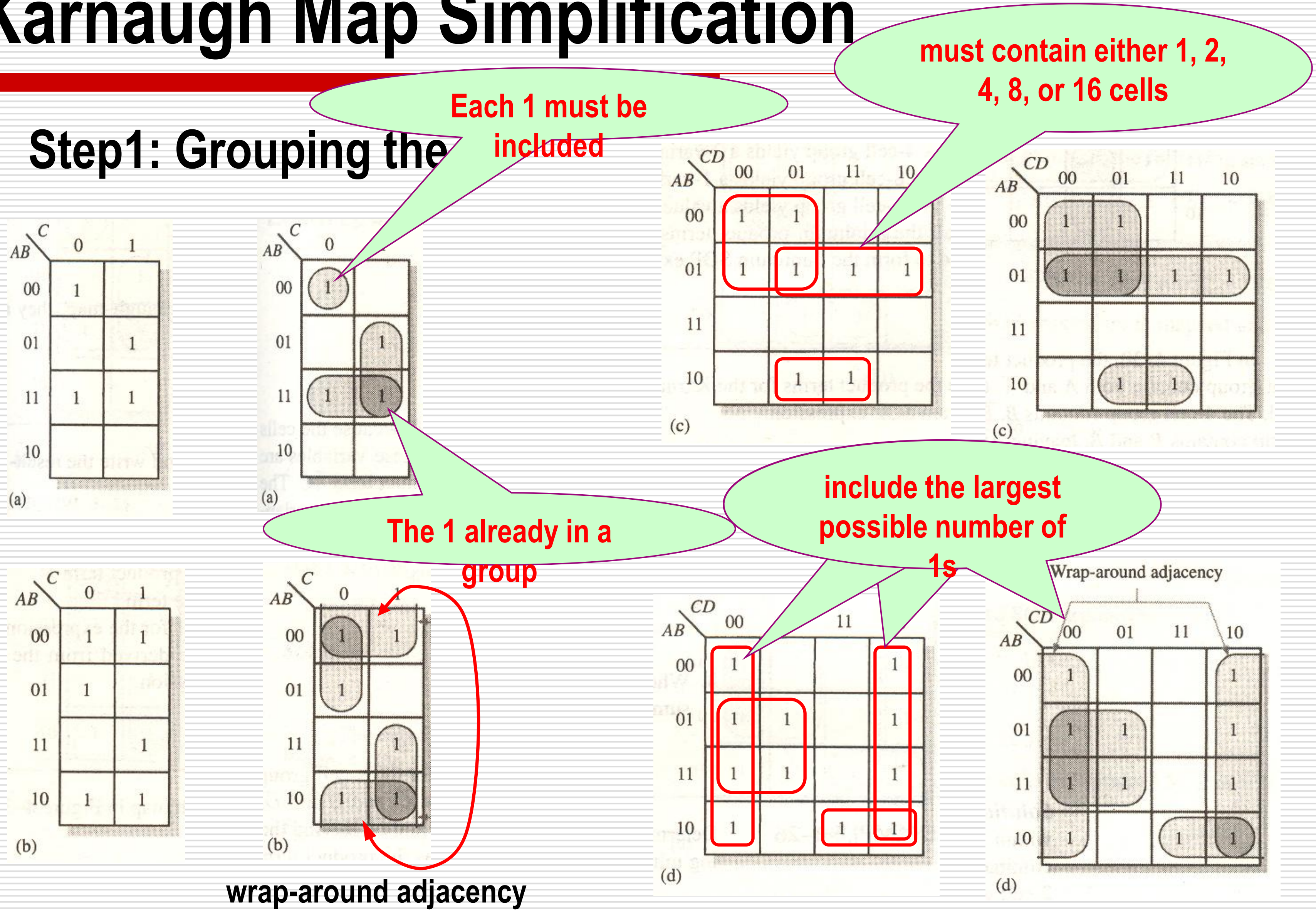
The goal is to maximize the size of the groups and minimize the number of groups.

Grouping the 1s rules

1. A group must contain either 1, 2, 4, 8, or 16 cells.
2. Each cell in a group must be adjacent to one or more cells in that same group.
3. Always include the largest possible number of 1s in a group.
4. Each 1 on the map must be included in at least one group.
5. The 1s already in a group can be included in another group as long as the overlapping groups include non-common 1s.（只要重叠组中有不属于其他组的1）

Karnaugh Map Simplification

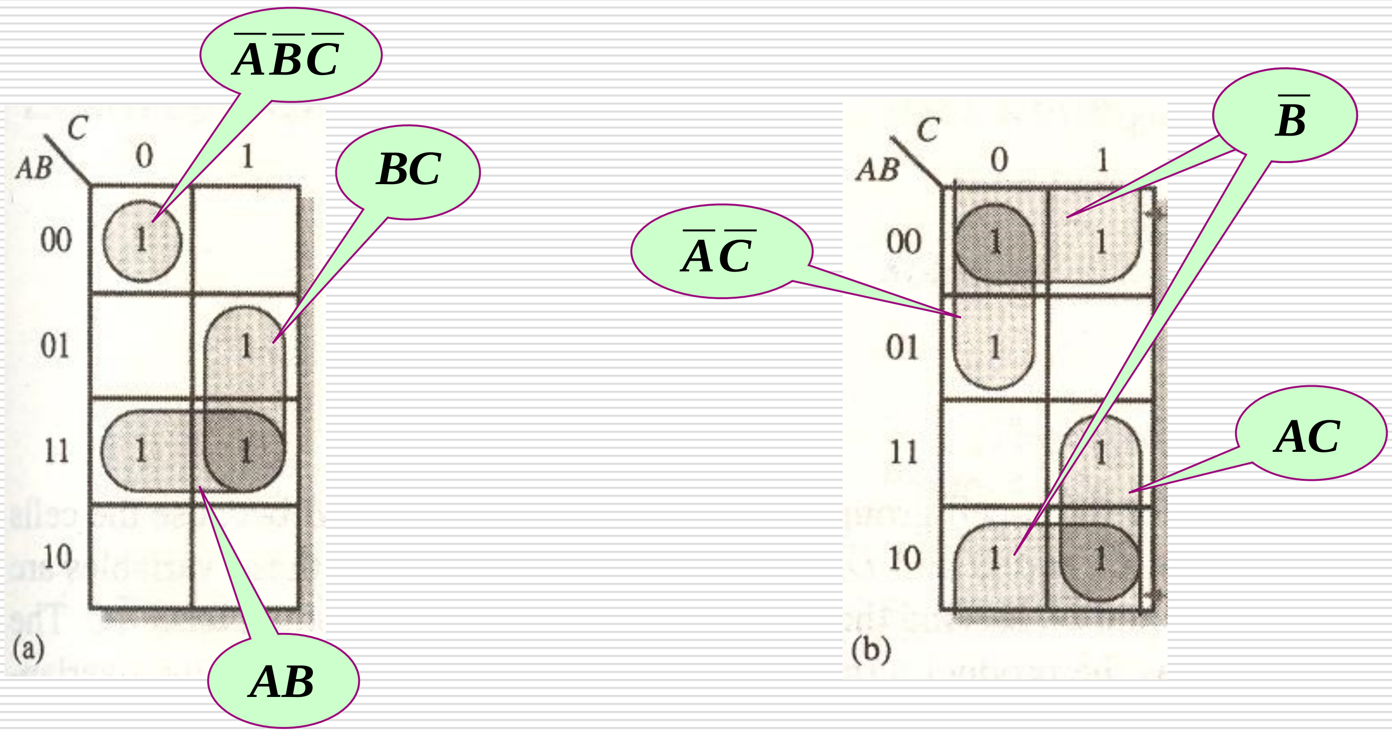
- Step1: Grouping the



Karnaugh Map Simplification

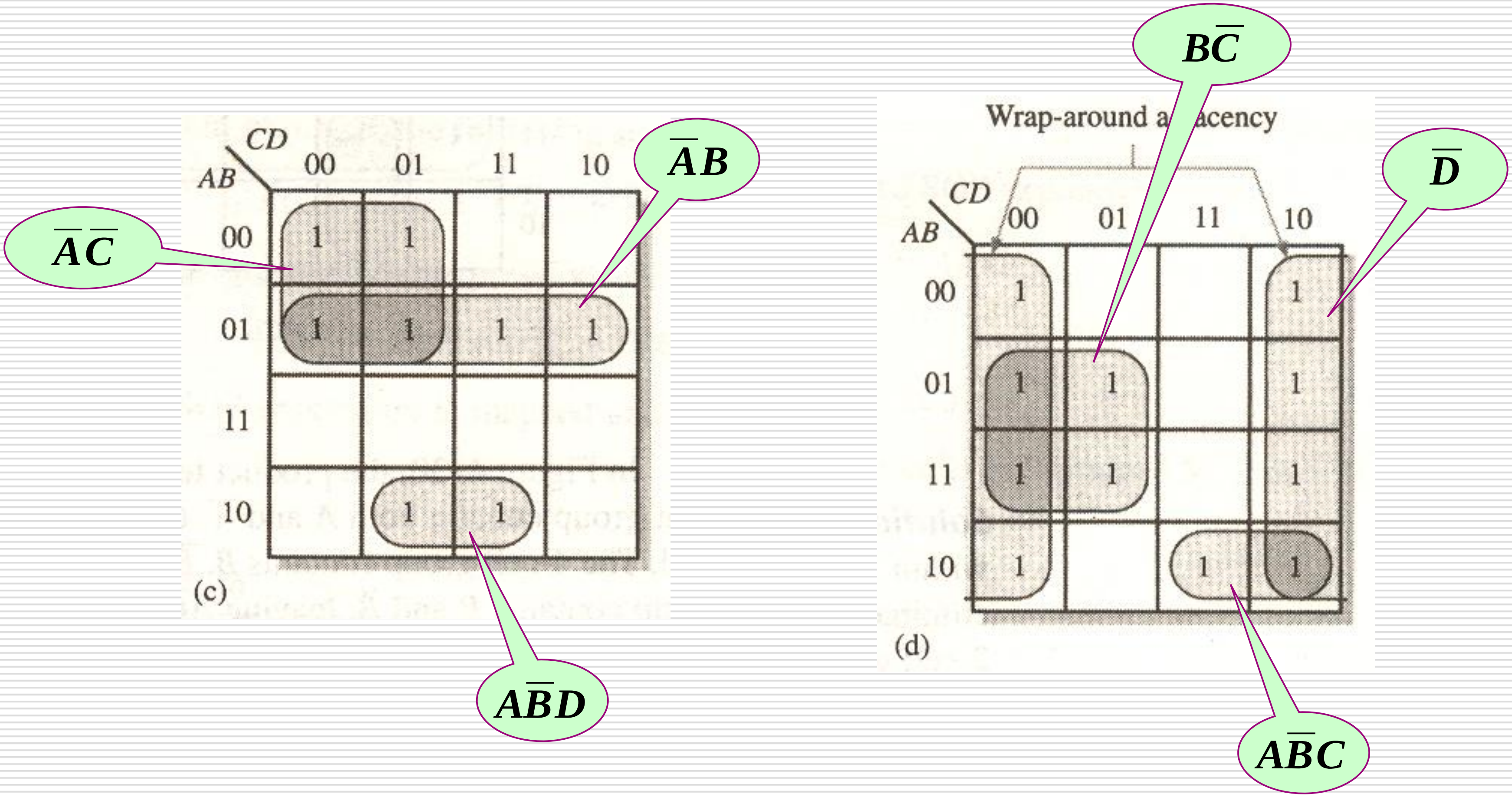
- **Step2: Determining the product term for each group**

Variables that occur both uncomplemented and complemented within the group are eliminated. These are called *contradictory variables* (矛盾变量).



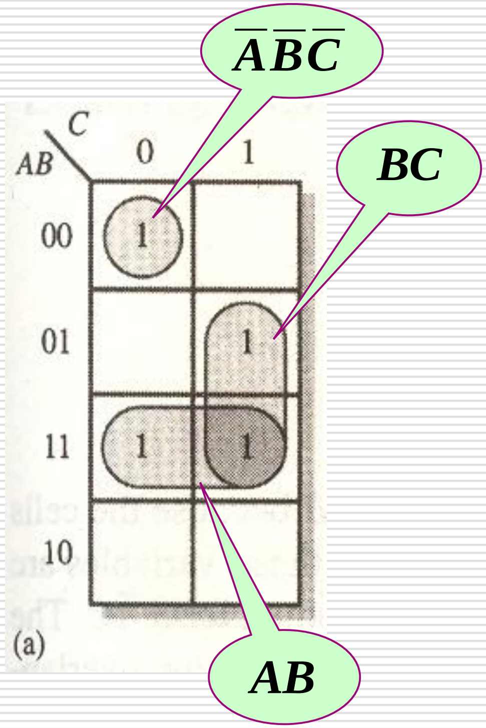
Karnaugh Map Simplification

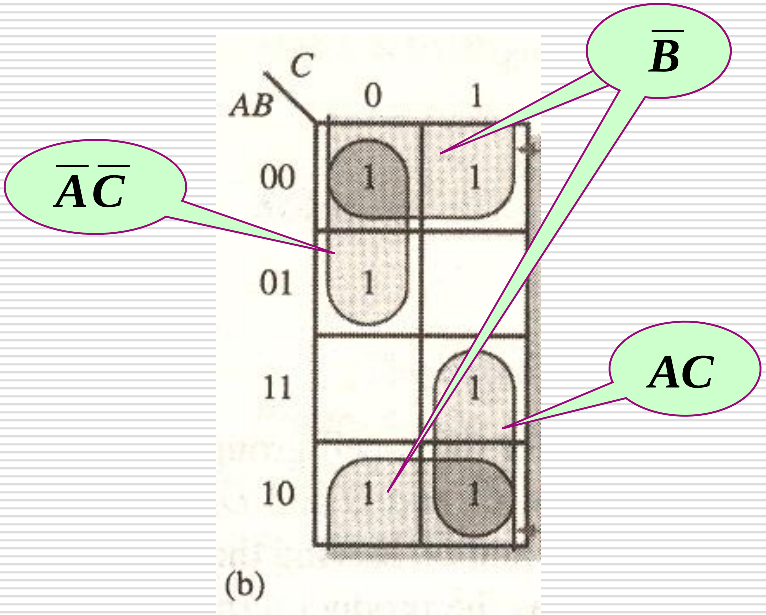
- Step2: Determining the product term for each group



Karnaugh Map Simplification

- Step3: Summing the resulting product terms





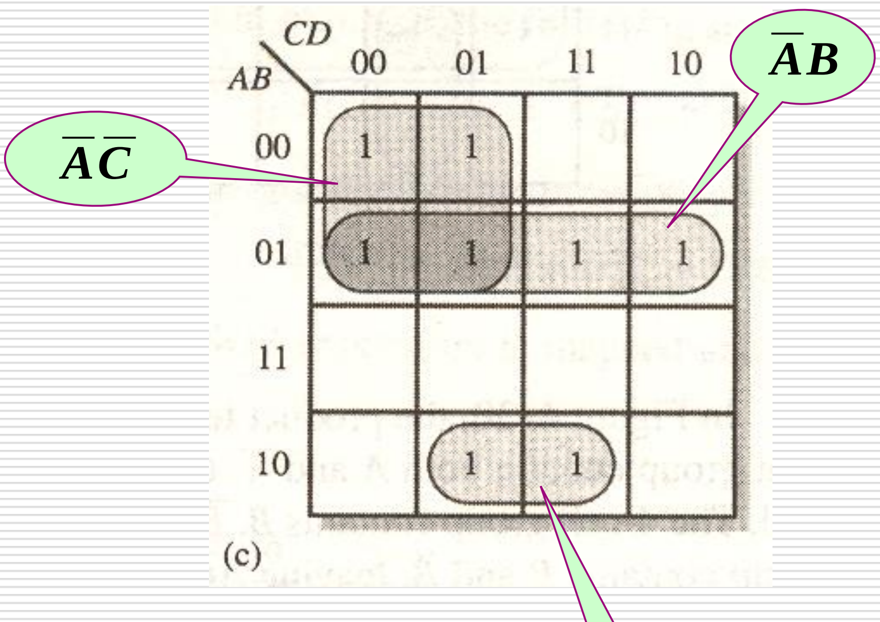
Minimum SOP expression

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

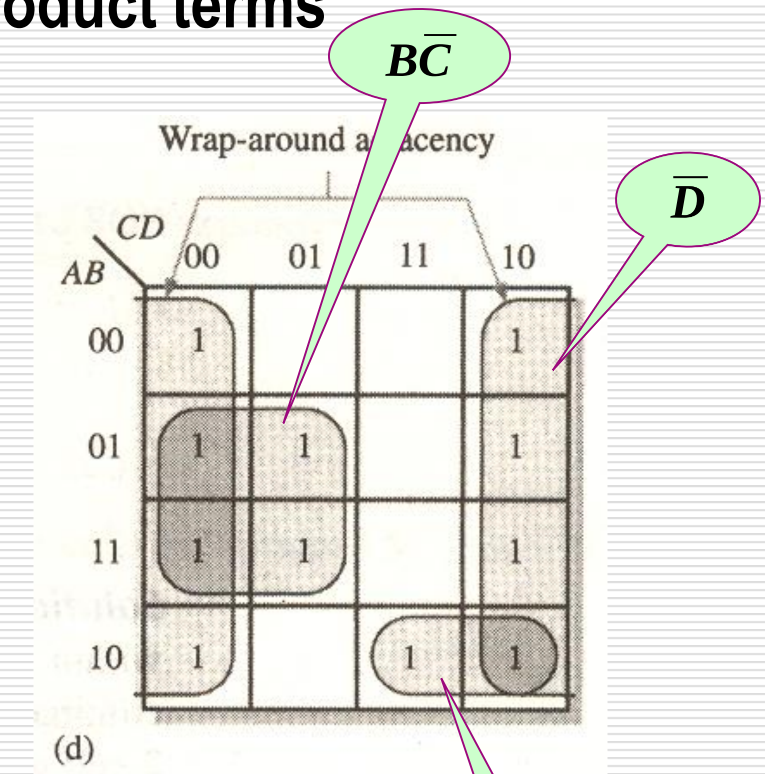
$$\overline{A}\overline{C}+AC+\overline{B}$$

Karnaugh Map Simplification

- Step3: Summing the resulting product terms



$$A\overline{B}D+\overline{A}\,\overline{C}+\overline{A}B$$



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

Karnaugh Map Simplification

Example

Use a K-Map to simplify the following SOP expression:

$$\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + W\overline{X}YZ + W\overline{X}\,\overline{Y}Z + \overline{W}YZ + W\overline{X}\,\overline{Y}\,\overline{Z}$$

Solution:

Mapping the expression on a K-Map

Grouping the 1s

Determining the product terms and summing them

$$W\overline{X}Z + \overline{W}YZ + \overline{X}\,\overline{Y}\,\overline{Z}$$

WX \ YZ	00	01	11	10
00	1		1	
01			1	
11				
10	1	1	1	

Karnaugh Map Simplification

Example

Use a K-Map to simplify the following SOP expression:

$$\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + W\overline{X}YZ + W\overline{X}\,\overline{Y}Z + \overline{W}YZ + W\overline{X}\,\overline{Y}\,\overline{Z}$$

Solution:

Mapping the expression on a K-Map

Grouping the 1s

Determining the product terms and summing them

$$W\overline{X}Z + \overline{W}YZ + \overline{X}\,\overline{Y}\,\overline{Z}$$

$$W\overline{X}\,\overline{Y} + \overline{W}YZ + \overline{X}\,\overline{Y}\,\overline{Z} + W\overline{X}Y$$

Non-minimization

WX \ YZ	00	01	11	10
00	1		1	
01			1	
11				
10	1	1	1	

About simplification

- In practice there are many software that can help us to do logic simplification. So it now becomes easier than before.
- The goal that we learn it is to understand the aims, the principles, the method, the meanings of logic simplification.