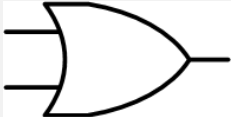
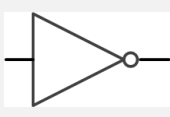


Revision on Boolean Algebra & K-map

- 1.1 Basic logic gates
- 1.2 Boolean algebra
- 1.3 Logic Circuit and Boolean Expression
- 1.4 Logic function - SOP, POS, minterm, maxterm, canonical form
- 1.5 Minimization using Boolean algebra
- 1.6 Karnaugh map
- 1.7 Minimization using Karnaugh map
- 1.8 Logic functions with don't care conditions

1.1 Basic Logic Gates

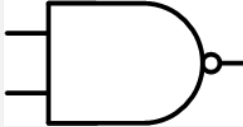
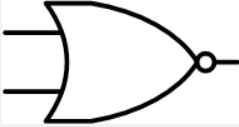
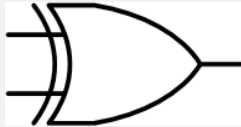
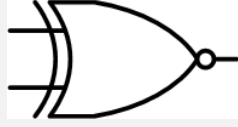
	OR gate	AND gate	NOT gate
Binary/Unary operator ?	Binary	Binary	Unary
Symbols	1: $+$ 2: $\vee$	1: $\cdot$ 2: $\wedge$ 3: absence of an operator	1: $'$ 2: $\sim$ 3: $-$
Examples	1: $a + b$ 2: $a \vee b$	1: $a \cdot b$ 2: $a \wedge b$ 3: ab	1: a' 2: $\sim a$ 3: $\bar{a}$
Logic Gate Symbol			

Truth table of OR		
a	b	$a + b$
0	0	0
0	1	1
1	0	1
1	1	1

Truth table of AND	
a b	a b
0 0	0
0 1	0
1 0	0
1 1	1

Truth table of NOT	
a	$\bar{a}$
0	1
1	0

Negated Gates

Operation	NAND	NOR	XOR	XNOR
$a \quad b$	$(ab)'$	$(a + b)'$	$a \oplus b$	$\overline{a \oplus b}$
0 0	1	1	0	1
0 1	1	0	1	0
1 0	1	0	1	0
1 1	0	0	0	1
Logic Gate Symbol				

1.2 Boolean Algebra

- A set of element S with at least two different elements (x , y) satisfying binary operations $(+)$ and $(\cdot)$.
- For Boolean algebra in which $S = \{0,1\}$, the formulation is referred as switching function.

Basic Postulates

If $x, y \in S$,

$$x + y = y + x$$

$$x \cdot y = y \cdot x$$

commutative

If $x, y, z \in S$,

$$x + (y + z) = (x + y) + z$$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

associative

If $x, y, z \in S$,

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

distributive

Distributive Law

- Proof

$$x + y \cdot z = (x + y) \cdot (x + z)$$

$x \ y \ z$	$y \cdot z$	$x + y \cdot z$	$x + y$	$x + z$	$(x + y) \cdot (x + z)$
0 0 0	0	0	0	0	0
0 0 1	0	0	0	1	0
0 1 0	0	0	1	0	0
0 1 1	1	1	1	1	1
1 0 0	0	1	1	1	1
1 0 1	0	1	1	1	1
1 1 0	0	1	1	1	1
1 1 1	1	1	1	1	1

Duality

- If an expression is valid in Boolean algebra, the dual of the expression is also valid.
- Principle of duality:

$$0 \cdot x = 0 \qquad 1 + x = 1$$

$$1 \cdot x = x \qquad 0 + x = x$$

$$x \cdot x = x \qquad x + x = x$$

$$x \cdot x' = 0 \qquad x + x' = 1$$

The expressions are interchangeable by replacing “0” by “1” and “+” by “ $\cdot$ ”.

Theorem

Idempotent

$$x + x = x$$

$$x \cdot x = x$$

Involution

$$(x')' = x$$

Absorption

$$x + x y = x$$

$$x (x + y) = x$$

Logical adjacency

$$x y + x y' = x$$

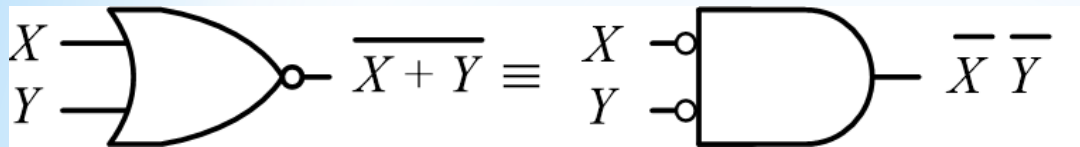
DeMorgan

$$\overline{(x + y)} = \bar{x} \bar{y}$$

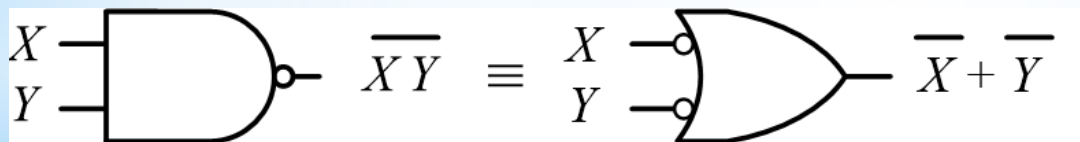
$$\overline{xy} = \bar{x} + \bar{y}$$

- The complement of sum is equal to the product of the complement
- The complement of product is equal to the sum of the complement

DeMorgan



X	Y	$\overline{X + Y}$	$\overline{X} \overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

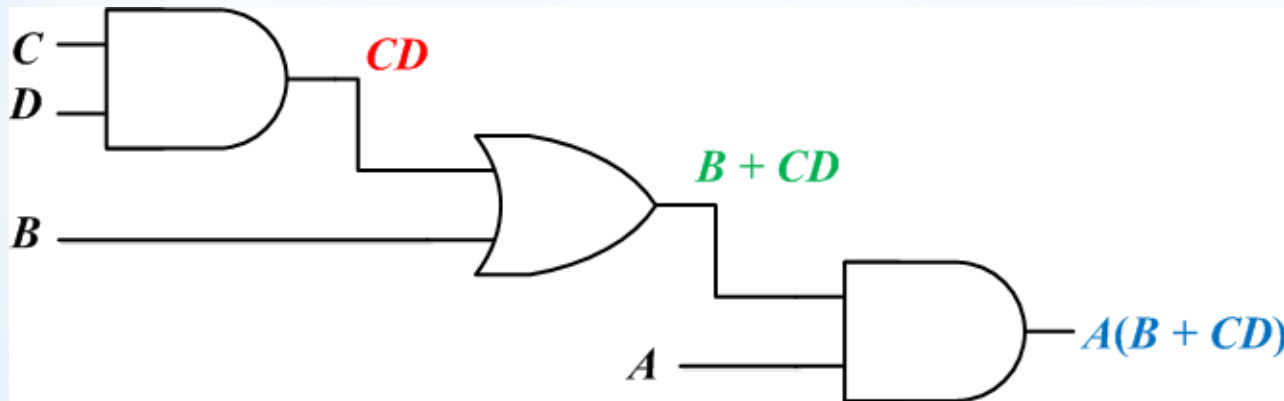


X	Y	$\overline{X Y}$	$\overline{X} + \overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Note that $\overline{X + Y} \neq \overline{X} + \overline{Y}$ and $\overline{X Y} \neq \overline{X} \overline{Y}$, a very common mistake !!

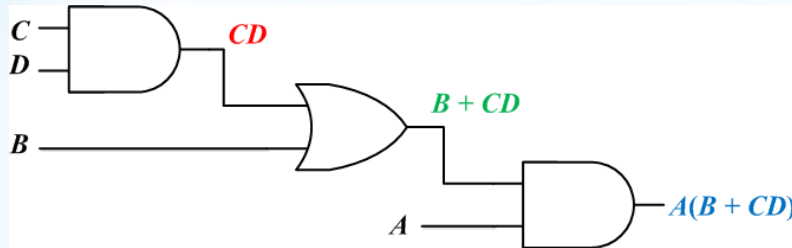
1.3 Logic Circuit and Boolean Expression

Boolean expression from a logic circuit



- Write down the output expression from all logic operators
- The Boolean function of this circuit is $A(B + CD)$
- Construct a truth table for above logic circuit

Truth table for a logic circuit



Completed solution for a logic circuit design must include:

1. Boolean Algebra
2. Circuit schematic
3. Truth table

Examples of numbering systems		Inputs				Output
Decimal	Hexadecimal	A	B	C	D	$A(B+CD)$
0	0	0	0	0	0	0
1	1	0	0	0	1	0
2	2	0	0	1	0	0
3	3	0	0	1	1	0
4	4	0	1	0	0	0
5	5	0	1	0	1	0
6	6	0	1	1	0	0
7	7	0	1	1	1	0
8	8	1	0	0	0	0
9	9	1	0	0	1	0
10	A	1	0	1	0	0
11	B	1	0	1	1	1
12	C	1	1	0	0	1
13	D	1	1	0	1	1
14	E	1	1	1	0	1
15	F	1	1	1	1	1

For the truth table, find the output as a following:

1. Write down all possible input combinations
2. Write down the final stage output (i.e. $A(B + CD)$)

Boolean function $\rightarrow$ Truth Table

Example: $f(x,y,z)=xy'+x'z'$

Input(s)		Output
x y z	xy' $x'z'$	$xy' + x'z'$
0 0 0	0 1	1
0 0 1	0 0	0
0 1 0	0 1	1
0 1 1	0 0	0
1 0 0	1 0	1
1 0 1	1 0	1
1 1 0	0 0	0
1 1 1	0 0	0

Truth Table → Boolean function

Inputs	Output
<i>a b c</i>	<i>f</i>
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	0



f is 1 if $\{(a = 0) \text{ AND } (b = 0) \text{ AND } (c = 1)\}$ OR
 $\{(a = 1) \text{ AND } (b = 0) \text{ AND } (c = 0)\}$ OR
 $\{(a = 1) \text{ AND } (b = 0) \text{ AND } (c = 1)\}$



$$f(a,b,c) = a'b'c + ab'c' + ab'c$$



Is it the simplest form?

1.4 Logic Function - SOP and POS

Logic functions are generally expressed with least number of literals (variables).

- *Sum of products (SOP)*

$$f(a,b,c,d) = ab'c + b'd' + a'cd$$

- *Product of sums (POS)*

$$f(a,b,c,d) = (a' + b + c)(b' + c + d')(a + c')$$

1.4.1 Minterm and Maxterm

Minterm :- For a function of n variables, if a product term contains all n variables **exactly one time** in its complemented or uncomplemented form, the product term is called *minterm*. Complement = 0 and Uncomplement = 1.

Function	Minterm	Not minterm	Not minterm
$f(A, B, C)$	$A' B' C$	$(A B)' C$	$A' B'$ or $AABC$

Maxterm :- If a sum term of a function of n variables contains all n variables **exactly one time** in its complemented or uncomplemented form, the sum term is called a *maxterm*. Complement = 1 and Uncomplement = 0.

Function	Maxterm	Not maxterm	Not maxterm
$f(A, B, C)$	$A' + B' + C$	$(A + B)' C$	$A' + B'$ or $A' + B + B + C$

Noted that the minterm and maxterm cannot be simplified.

Minterms and maxterms for 3 variables logic function

			Minterms		Maxterms	
x	y	z	Term	designation	term	designation
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

The number of minterms and maxterm of a logic function of n variables equals to 2^n , e.g. $f(a,b,c,d) \rightarrow 16$ minterms (maxterms);

$f(a,b,c,d,e) \rightarrow 32$ minterms (maxterms).

1.4.2 Canonical form

The canonical form of a logic function is a presentation in either minterms or maxterms.

In minterm form (with logic output “1”):

$$f(A, B, C) = \overline{A}B\overline{C} + A\overline{B}\overline{C} + \overline{A}BC + ABC$$

Minterm	Code	Number
$A'BC'$	010	m_2
ABC'	110	m_6
$A'BC$	011	m_3
ABC	111	m_7

$$f(A, B, C) = m_2 + m_3 + m_6 + m_7$$

Canonical form $f(A, B, C) = \sum m(2, 3, 6, 7)$

Noted for minterm: Complement = 0 and Uncomplement = 1

The canonical form of a logic function is a presentation in either minterms or maxterms.

In maxterm form (with logic output “0”):

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

Maxterm	Code	Number
$A + B + C$	000	M_0
$A + B + C'$	001	M_1
$A' + B + C$	100	M_4
$A' + B + C'$	101	M_5

$$f(A, B, C) = M_0 M_1 M_4 M_5$$

Canonical form $f(A, B, C) = \prod M(0, 1, 4, 5)$

Noted for maxterm: Complement = 1 and Uncomplement = 0

Canonical SOP representation:

$$f(a, b, c) = a + \bar{b}c$$

$$= a(b + \bar{b}) + \bar{b}c(a + \bar{a})$$

$$= ab(c + \bar{c}) + a\bar{b}(c + \bar{c}) + \bar{b}c(a + \bar{a})$$

$$= abc + ab\bar{c} + a\bar{b}c + a\bar{b}\bar{c} + \bar{a}bc + \bar{a}\bar{b}c$$

$$= m_1 + m_4 + m_5 + m_6 + m_7$$

$$f(a, b, c) = \sum m(1, 4, 5, 6, 7)$$

Canonical POS representation:

$$\begin{aligned}
 f(x, y, z) &= xy + \bar{x}z \\
 &= (xy + \bar{x})(xy + z) = (x + \bar{x})(y + \bar{x})(x + z)(y + z) \\
 &= (\bar{x} + y)(x + z)(y + z) = (\bar{x} + y + z\bar{z})(x + z + y\bar{y})(y + z + x\bar{x}) \\
 &= (\bar{x} + y + z)(\bar{x} + y + \bar{z})(x + z + y)(x + \bar{y} + z)(x + y + z)(\bar{x} + y + z) \\
 &= (x + y + z)(x + \bar{y} + z)(\bar{x} + y + z)(\bar{x} + y + \bar{z})
 \end{aligned}$$

$$f(x, y, z) = M_0 M_2 M_4 M_5 = \Pi M(0, 2, 4, 5)$$

Conversion between canonical forms

Relationship

$$\overline{\text{maxterm}_i} = \text{minterm}_i \text{ (i.e. } \overline{M_i} = m_i \text{)}$$

$$\overline{\text{minterm}_i} = \text{maxterm}_i \text{ (i.e. } \overline{m_i} = M_i \text{)}$$

$$\begin{aligned} \text{e.g. } M_3' &= (a + b' + c') \\ &= a'bc \text{ (De Morgan's Theorem)} \\ &= m_3 \end{aligned}$$

Canonical form conversion: SOP $\leftrightarrow$ POS

For minterm representation (**choose all the terms with output 1**):

$$f(x, y, z) = \bar{x}\bar{y}z + x\bar{y}\bar{z} + x\bar{y}z + xyz$$

$$f(x, y, z) = m_1 + m_4 + m_5 + m_7 = \sum m(1, 4, 5, 7)$$

For maxterm representation (**choose all the terms with output 0**):

$$\overline{f(x, y, z)} = \bar{x}\bar{y}\bar{z} + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$$

$$f(x, y, z) = \overline{\bar{x}\bar{y}\bar{z} + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz}$$

$$= (x + y + z)(x + \bar{y} + z)(x + \bar{y} + \bar{z})(\bar{x} + \bar{y} + z)$$

$$f(x, y, z) = M_0 M_2 M_3 M_6 = \prod M(0, 2, 3, 6)$$

$x \ y \ z$	f
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	1

Noted that for maxterm the uncomplemented variable = 0,
complemented variable = 1

1.5 Minimization using Boolean Algebra

- Obtain a simple (or simplest) logic circuit
- Reduce the cost of production
 - Less logic gate ICs required
 - PCB (Printed Circuit Board)
 - ➔ size reduced
- Reduce the power consumption
 - ➔ longer battery usage time

Example:

Given 4 equivalent Boolean functions f_1 to f_4 expressed in SOP form already (to be proved in later session).

$f_1(a,b,c) = a'bc' + a'bc + ab'c' + ab'c + abc$ (5 product terms, 15 literals)

$f_2(a,b,c) = a'b + ab' + abc$ (3 product terms, 7 literals)

$f_3(a,b,c) = a'b + ab' + ac$ (3 product terms, 6 literals)

$f_4(a,b,c) = a'b + ab' + bc$ (3 product terms, 6 literals)

Both f_3 & f_4 are the minima

How can you simplify f_1 to f_3 ?

Simplification

$$\begin{aligned}
 f_1(a,b,c) &= a'bc' + a'bc + ab'c' + ab'c + abc \\
 &= (a'bc' + a'bc) + (ab'c' + ab'c) + abc \\
 &= a'b + ab' + abc = f_2 \\
 &= a'b + a(b' + bc) \\
 &= a'b + a(b' + c) \\
 &= a'b + ab' + ac \\
 &= f_3
 \end{aligned}$$

How to obtain f_4 ?

$$f_4(a,b,c) = a'b + ab' + bc$$

Example:

$$\begin{aligned}
 f(A, B, C, D) &= \overline{AB + AC} + \bar{A} \bar{B} \bar{C} \\
 &= (\overline{AB}) (\overline{AC}) + \bar{A} \bar{B} \bar{C} \\
 &= (\bar{A} + \bar{B}) (\bar{A} + \bar{C}) + \bar{A} \bar{B} \bar{C} \\
 &= \bar{A} \bar{A} + \bar{A} \bar{C} + \bar{A} \bar{B} + \bar{B} \bar{C} + \bar{A} \bar{B} \bar{C} \\
 &= \bar{A} + \bar{A} \bar{C} + \bar{A} \bar{B} + \bar{B} \bar{C} \\
 &= \bar{A} + \bar{A} \bar{B} + \bar{B} \bar{C} \\
 &= \bar{A} + \bar{B} \bar{C}
 \end{aligned}$$

DeMorgan

Example:

$$\begin{aligned} f(A, B, C, D) &= AB + B\overline{C} + CD + B\overline{D} \\ &= AB + CD + B(\overline{C} + \overline{D}) \\ &= AB + CD + \overline{BCD} \\ &= AB + (CD + B)(CD + \overline{CD}) \\ &= AB + CD + B \\ &= B(A + 1) + CD \\ &= B + CD \end{aligned}$$

A very useful rule for simplifying Boolean function

Basic: $a + \bar{a}b = (a + \bar{a})(a + b) = a + b$

We can have the following general form:

$$a + \bar{a}[\dots] = a + [\dots]$$

Examples: $x + \bar{x}(w + yz) = x + (w + yz)$

$$a + \bar{a}(bd + \overline{bcd} + \bar{c}d) = a + bd + \overline{bcd} + \bar{c}d$$

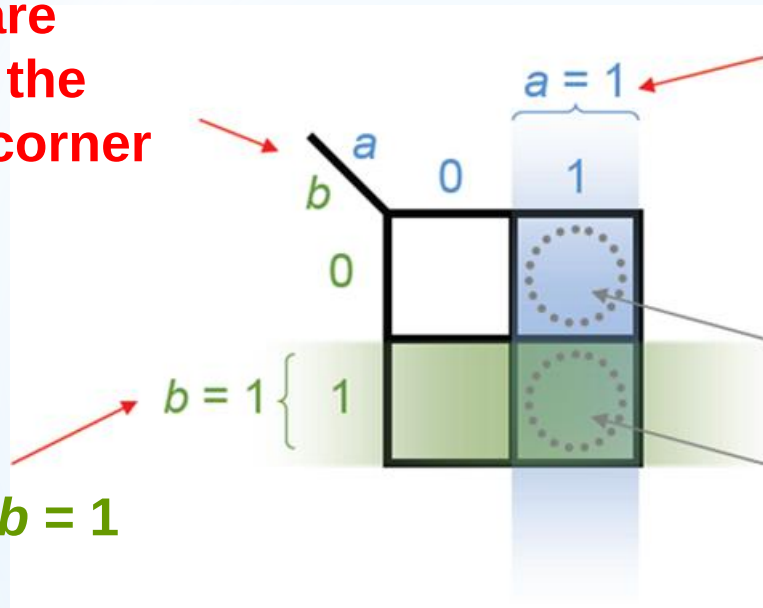
1.6 Karnaugh Map

- In 1953, Maurice Karnaugh introduced a map method known as **Karnaugh map (K-map)**
 - A straightforward procedure for minimizing Boolean functions in a table form
 - Graphical representation of a truth table
 - Minterm is used in the cell of the K-map
 - It is n -variable function (defined by 2^n):
 - Two-variable K-map has 4 cells
 - Three-variable K-map has 8 cells
 - Four-variable K-map has 16 cells

Two-variable K-map

Variables are labeled on the upper left corner of the map

This row represents $b = 1$



This column represents $a = 1$

This cell means $(a = 1)$ AND $(b = 0)$

This cell means $(a = 1)$ AND $(b = 1)$

a	0	1
b		
0	$a'b'$	ab'
1	$a'b$	ab

a	0	1
b		
0	m_0	m_2
1	m_1	m_3

Minterm representations

Plotting functions in K-map

$f(a,b) = \sum m(0,3)$ Canonical form (contain Minterm)

$a \backslash b$	0	1
0		
1		

or

$a \backslash b$	0	1
0	1	
1		1

Put a 0 or leave blank for those minterms not included in the function

Put a 1 in the corresponding cells

$f(a,b) = a'b + ab'$ Function must be formed by Minterm

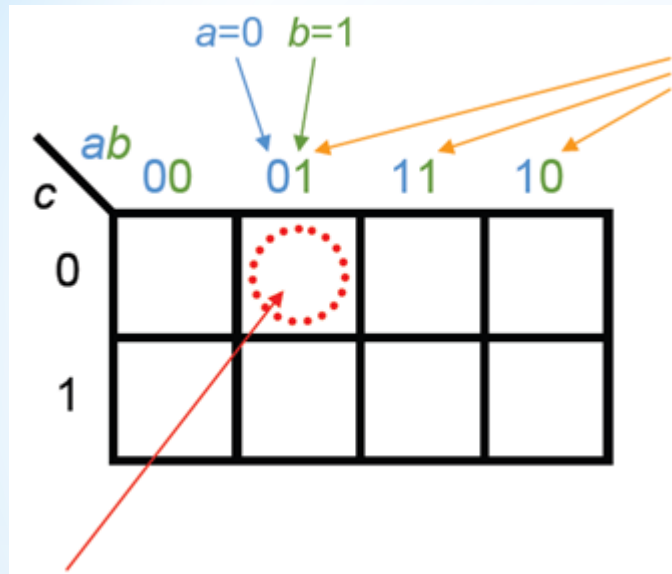
$a \backslash b$	0	1
0	1	1
1	0	0

or

$a \backslash b$	0	1
0	1	1
1		

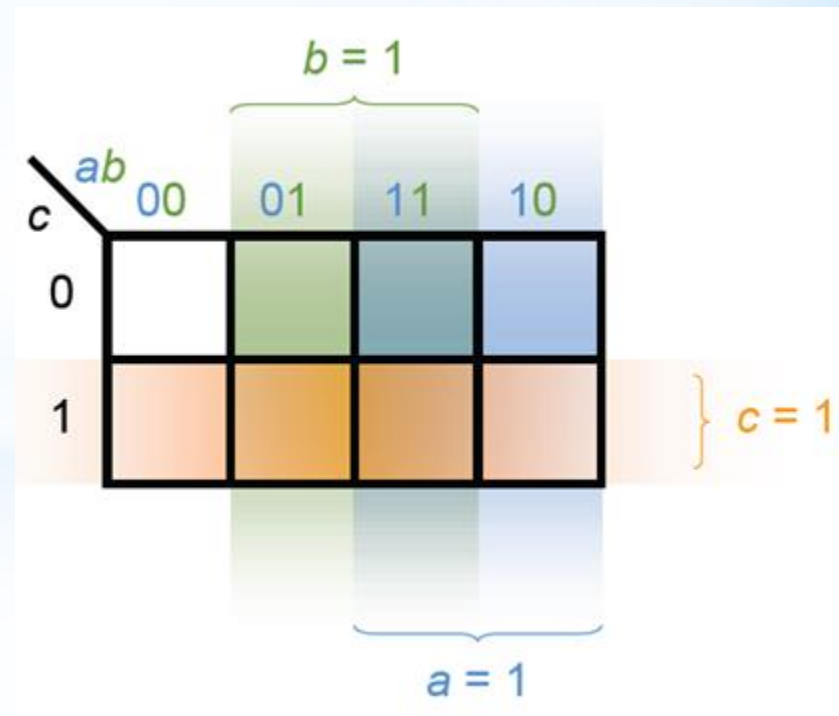
Functions represented graphically with corresponding minterm cells labeled to value 1

Three-variable K-map



Note: the columns are not in numerical order, but Gray code order (why)?

This cell means:
 $(a = 0) \text{ AND } (b = 1) \text{ AND } (c = 0)$



Minterm representations

c \ ab	00	01	11	10
0	$a'b'c'$	$a'bc'$	abc'	$ab'c'$
1	$a'b'c$	$a'bc$	abc	$ab'c$

c \ ab	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

Simplification of Production Terms

Example: Simplify $f(a,b,c) = \sum m(6,7)$

	<i>ab</i>	00	01	11	10
<i>c</i>	0			1	
	1			1	

This group contains both 0 and 1 for c (i.e. no longer depends on c , depends on a and b only)

Using Boolean Algebra:

$$f(a,b,c) = abc' + abc$$

$$= ab \text{ (adjacency)}$$

Only one-variable difference

Rule: whenever we group two adjacent cells, they can form a product term with one variable less

Wrap-around Adjacency

ab		00	01	11	10
		0	1	1	0
c	0	1*			1
	1				

Adjacent cells (1-bit change only)

ab		00	01	11	10	00	01
		0	1	1	0	0	1
c	0	1*			1	1*	
	1						

ab		00	01	11	10
		0	1	1	0
c	0	1*			1
	1				

ab		00	10
		0	1
c	0	*1	1
	1	01	11

Form a cylinder!

More examples

		<i>ab</i>			
		00	01	11	10
<i>c</i>	0			1	1
	1				

$$f(a,b,c) = abc' + ab'c'$$

$$= ac' \text{ (adjacency)}$$

We can even group adjacent 1's across the edges:

		<i>ab</i>			
		00	01	11	10
<i>c</i>	0	1			1
	1				

Also one-variable difference!

$$f(a,b,c) = a'b'c' + ab'c'$$

$$= b'c' \text{ (adjacency)}$$

Format of three-variable

Label rows with first variable,
Columns with the others

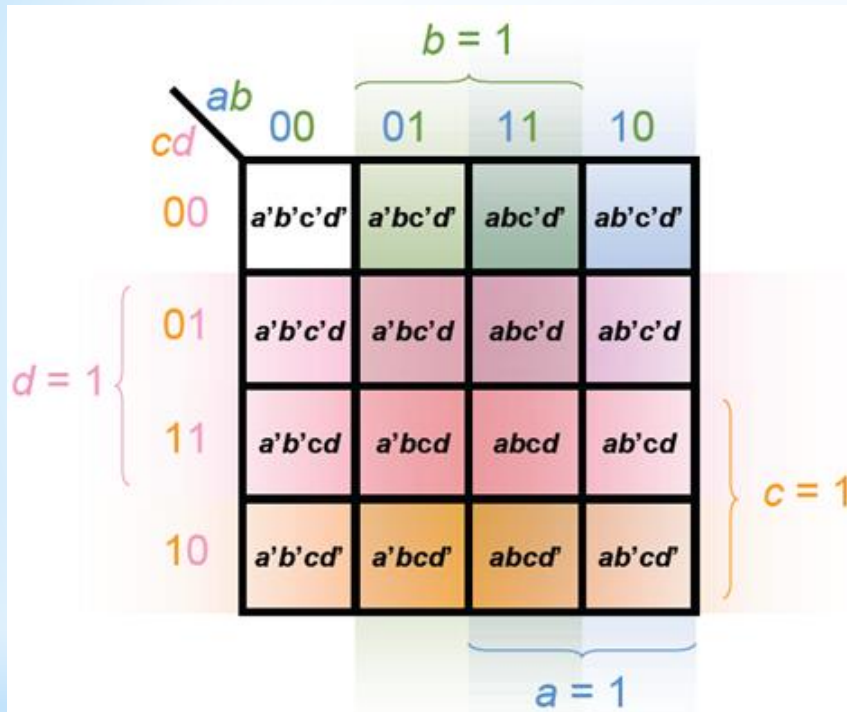
$a \backslash bc$	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

Vertical orientation of
three-variable K-map

$bc \backslash a$	0	1
00	m_0	m_4
01	m_1	m_5
11	m_3	m_7
10	m_2	m_6

Although there are different ways
drawing the K-map, we use the same
method to group the adjacent 1's!

Four-variable K-map



		ab			
		00	01	11	10
cd	00	m_0	m_4	m_{12}	m_8
	01	m_1	m_5	m_{13}	m_9
	11	m_3	m_7	m_{15}	m_{11}
	10	m_2	m_6	m_{14}	m_{10}

Not: Only 1 bit difference between adjacent cells

Wrap-around Adjacency for 4 Variables

ab		00	01	11	10
cd	00	m_0	m_4	m_{12}	m_8
	01	m_1	m_5	m_{13}	m_9
	11	m_3	m_7	m_{15}	m_{11}
	10	m_2	m_6	m_{14}	m_{10}

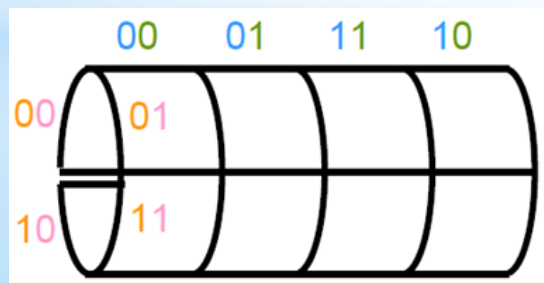
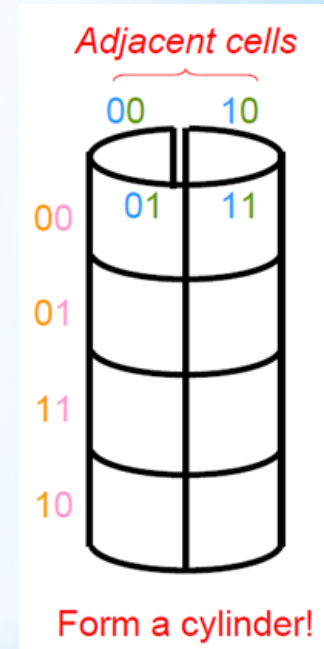
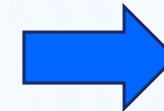
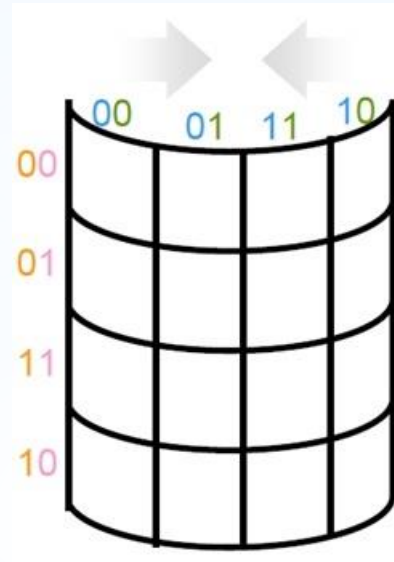
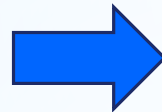


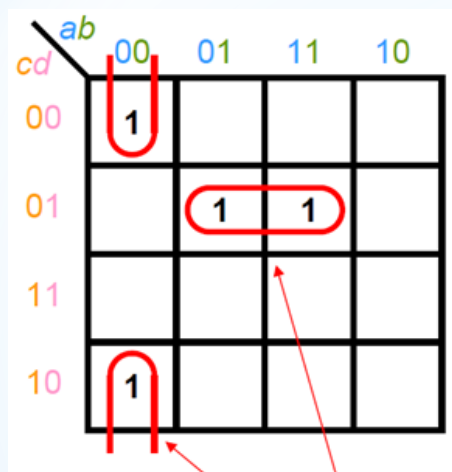
Image the map as

1-bit change only for
every adjacent cells!

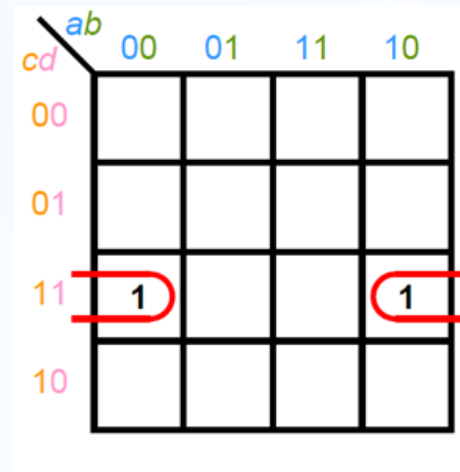
1-bit change only for
every adjacent cells!

$cd \backslash ab$	00	01	11	10	00	01	11	10
00	m_0	m_4	m_{12}	m_8	m_0	m_4	...	
01	m_1	m_5	m_{13}	m_9	m_1	m_5		
11	m_3	m_7	m_{15}	m_{11}	m_3	m_7		
10	m_2	m_6	m_{14}	m_{10}	m_2	m_6		
00	m_0	m_4	m_{12}	m_8	m_0	m_4		
01	m_1	m_5	m_{13}	m_9	m_1	m_5		
11	...							
10								

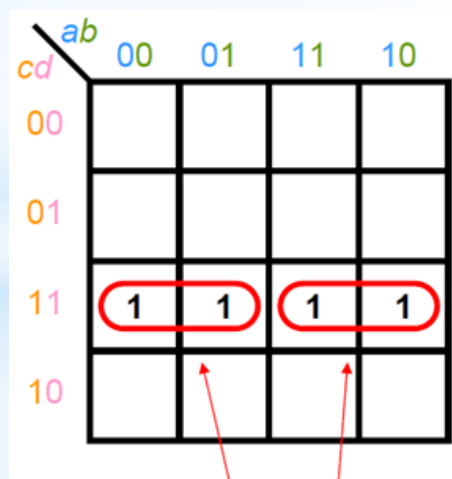
Examples of 4-variable K-map



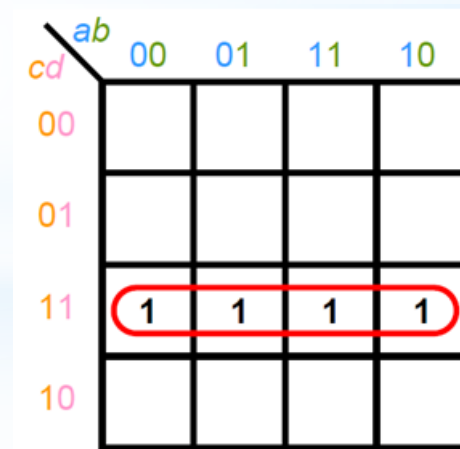
$$f(a,b,c,d) = a'b'd' + bc'd$$



$$f(a,b,c,d) =$$

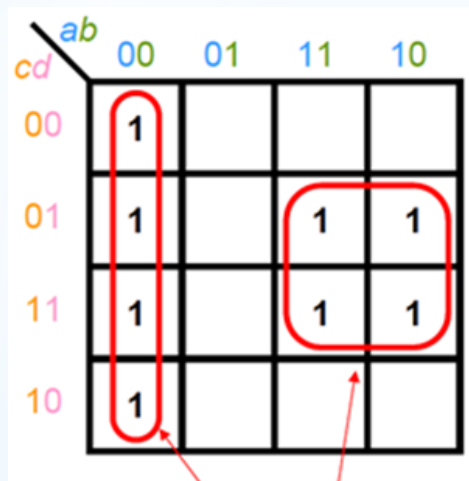


$$f(a,b,c,d) = a'cd + acd = cd$$

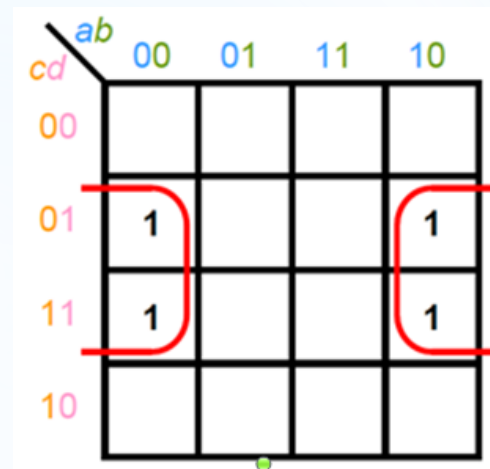


$$f(a,b,c,d) = cd$$

Examples of 4-variable K-map

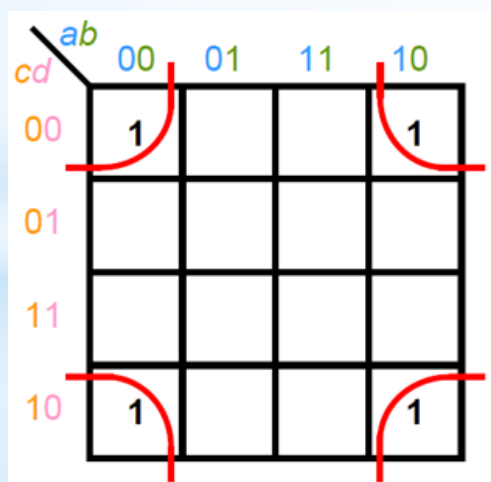


$$f(a,b,c,d) = a'b' + ad$$



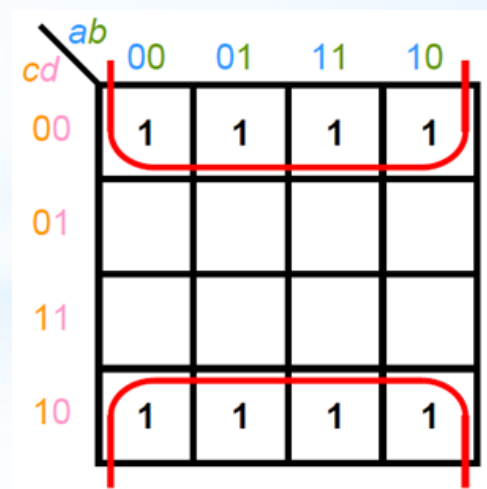
$$f(a,b,c,d) =$$

Across 4 corners:



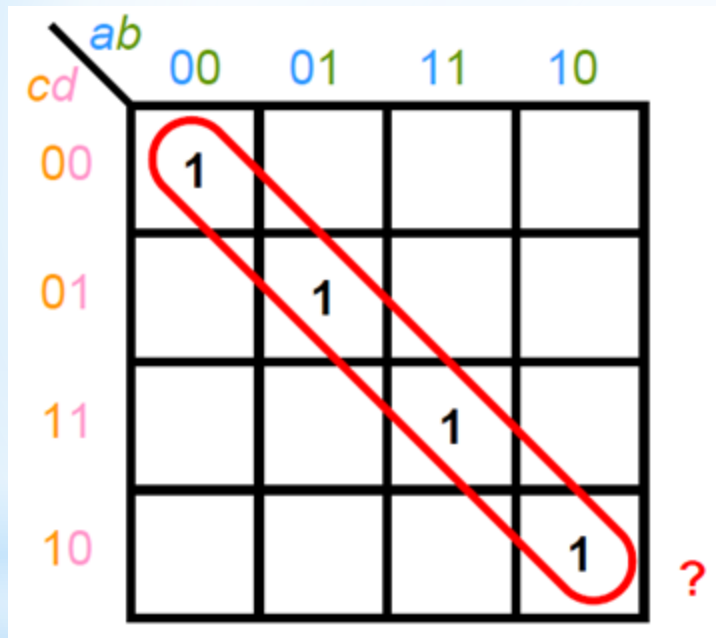
$$f(a,b,c,d) =$$

Group of 8 cells:

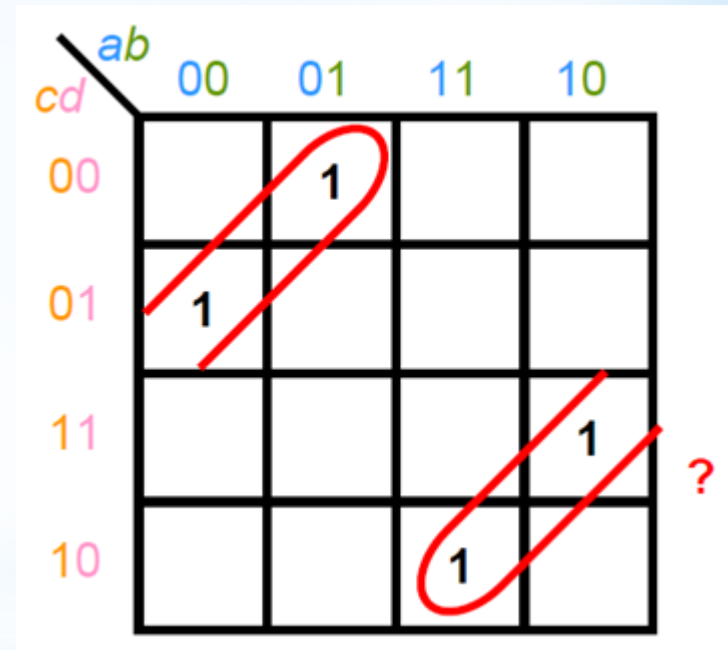


$$f(a,b,c,d) =$$

Are They Adjacent Cells?

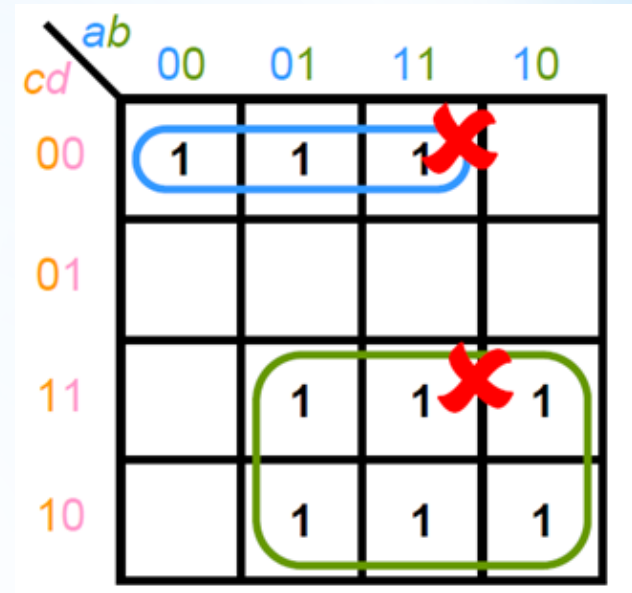
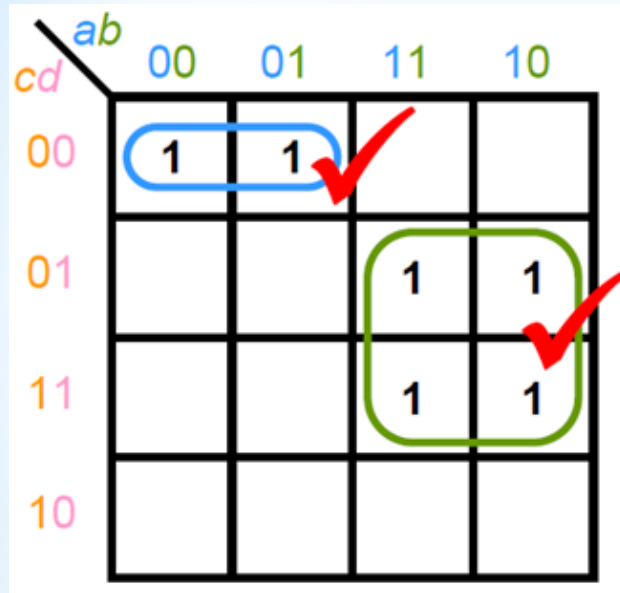


Diagonal **X**



Magic square **X**

Summary for K-map method



Group size is power of 2 (e.g. 2, 4, 8)

Other group size is illegal!

Limitations

- The Boolean functions minimized by K-map are always in SOP or POS form
- Can handle minimization for two-level circuits, but not three or more levels

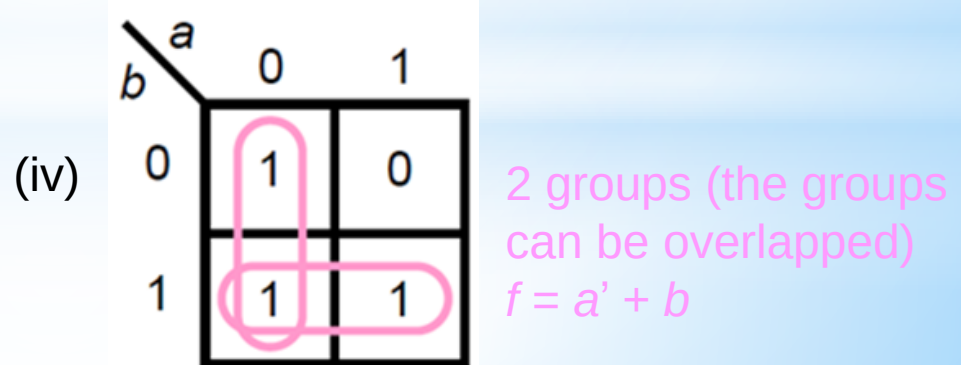
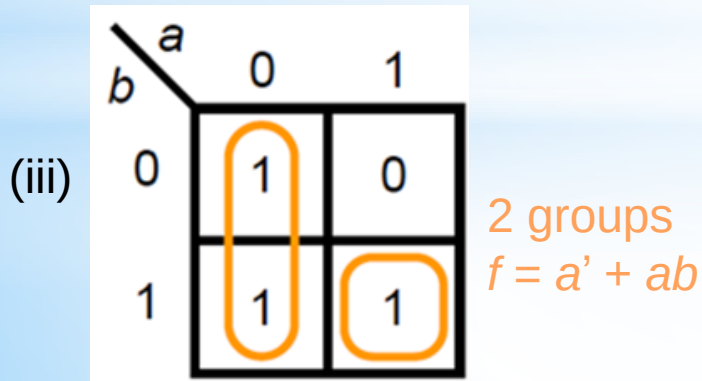
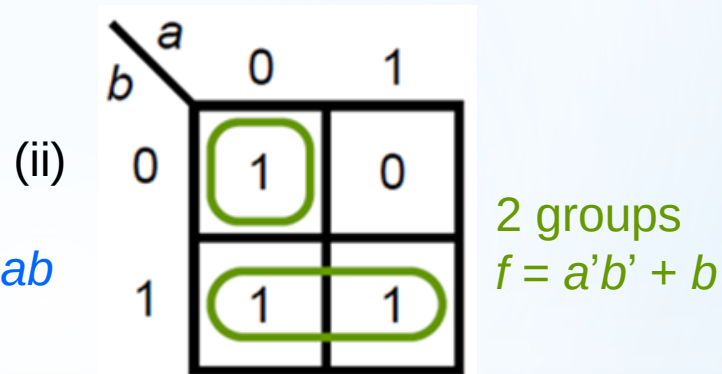
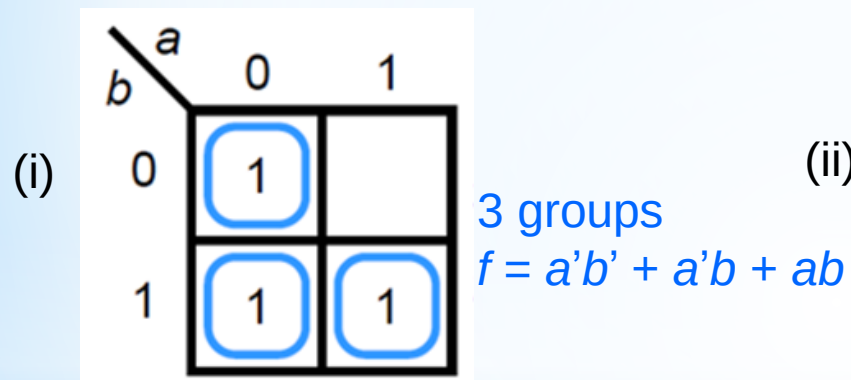
1.7 Minimization using Karnaugh map

- Group the adjacent cells (the number of cells must be a power of 2)
- Rules
 - (1) To find the fewest group that covers all cells with marked of 1s
 - (2) The groups should be as large as possible
- Goal
 - Reduce the number of products (terms) to minimum
 - Save the cost

Example: Two-variable K-map

Simplify $f(a,b) = \sum m(0,1,3)$

Many ways to group them. Which is best solution?

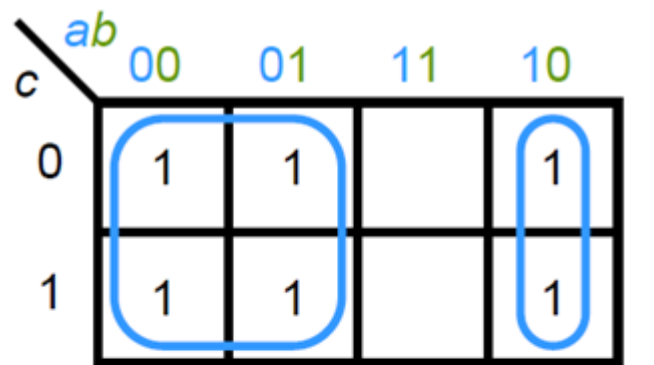


Example: Three-variable K-map

Simplify $f(a,b,c) = \sum m(0,1,2,3,4,5)$

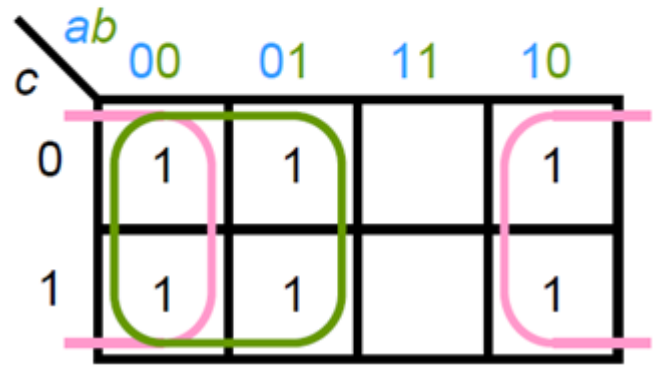
Which solution is better?

(i)



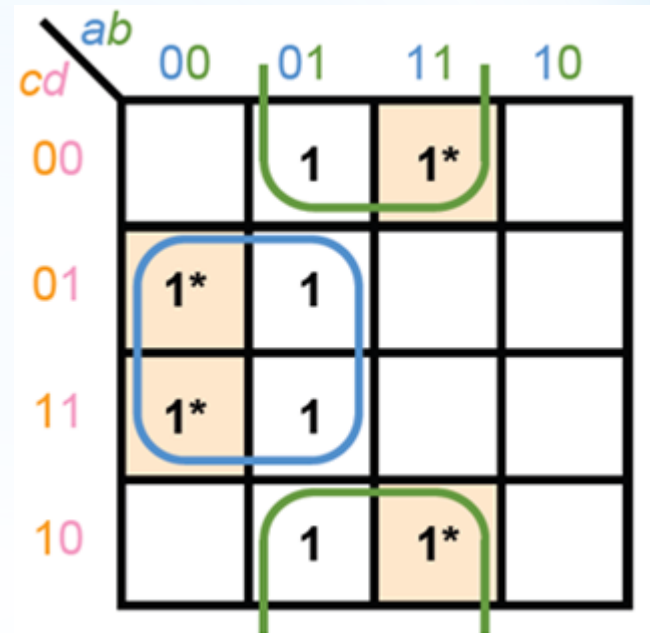
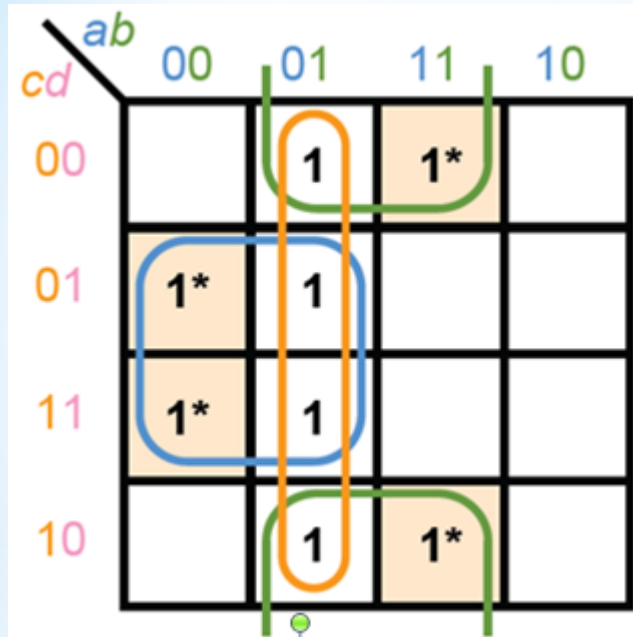
groups
 $f(a,b,c) = a' + ab'$

(ii)



groups
 $f(a,b,c) = a' + b'$

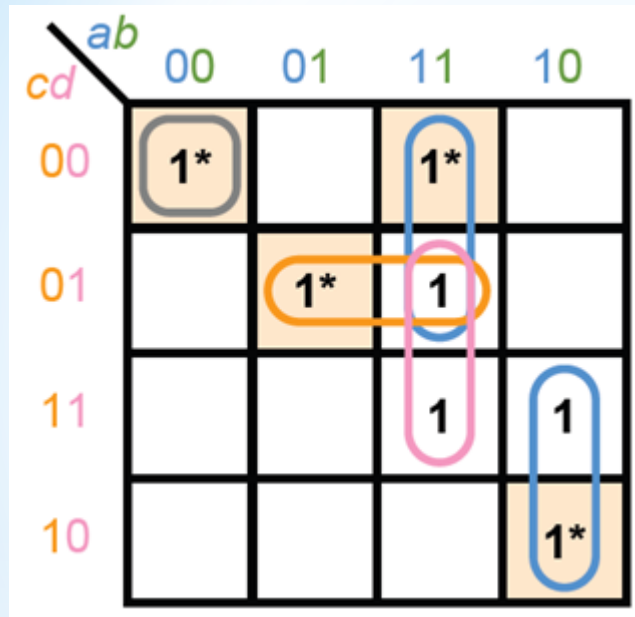
Example: Four-variable K-map



Select the least number of groups to cover all minterms:

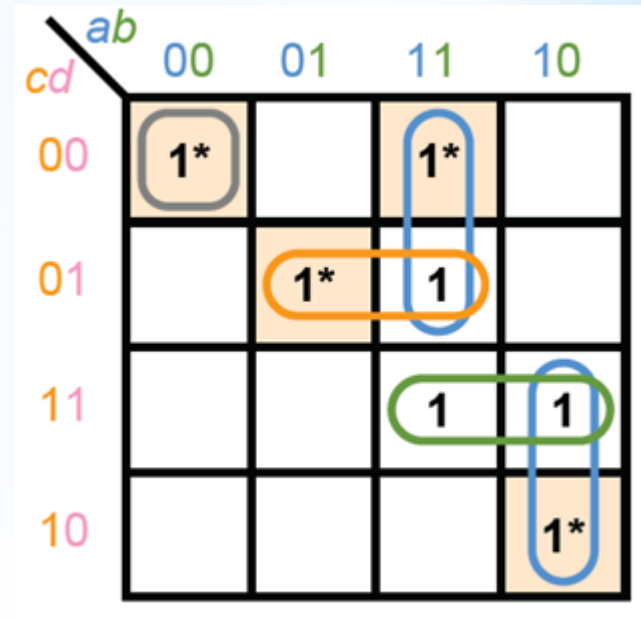
$$f(a, b, c, d)' = a'd + bd'$$

Two Solutions



$$f(a,b,c,d) = a'b'c'd' + abc' + ab'c + bc'd + abd$$

or



$$f(a,b,c,d) = a'b'c'd' + abc' + ab'c + bc'd + acd$$

We can choose either  or 

Logic function in POS with K-map

- Previous examples only show the simplified Boolean functions in **SOP** form
 - How to obtain functions in **POS** form
 - Step 1:- Group the **0**s to obtain f' in **SOP** form
 - Step2:- Apply DeMorgan's Theorem to find f in **POS** form
- or
- Group the **0**s and express them directly in POS with Maxterms (require to complement the variables with **+**)

Example: Find POS

Simplify $f(a,b,c,d) = \sum m(0,1,2,5,8,9,10)$ in POS form

	ab 00 01 11 10			
cd 00 01 11 10		0	0	
			0	
	0	0	0	0
		0	0	

Fill the 1s and 0s into the map

	ab 00 01 11 10			
cd 00 01 11 10		0	0	
			0	
	0	0	0	0
		0	0	

Group the 0s using the same procedure as grouping the 1s

$$f'(a,b,c,d) = ab + cd + bd'$$

$$f(a,b,c,d) = (a'+b')(c'+d')(b'+d)$$

1.8 Logic functions with Don't-care conditions

The output of Boolean functions are **incompletely specified functions**,

- For some input conditions, the outputs are unspecified
- Input condition has no effects to the function
i.e., the output for those input are of no concern
- Output values are defined as **don't-care**
- Don't-care term can be minterm / maxterms
- Don't-care term indicates by an $\times$, d , ϕ or φ

Truth Table with Don't Care

a	b	f
0	0	0
0	1	1
1	0	1
1	1	X

What the table says is:

f is 0 if $(a = 0 \text{ AND } b = 0)$
 f is 1 if $(a = 0 \text{ AND } b = 1), \text{ or } (a = 1 \text{ AND } b = 0)$

f can be 0 or 1 if $(a = 1 \text{ AND } b = 1)$

a	b	f_1	f_2
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1

$$f(a, b) = \sum m(1, 2) + \sum d(3)$$

Both f_1 or f_2 of the table on the left are acceptable

K-map with Don't-care

Which solution is better?

		<i>a</i>		0	1
		<i>b</i>		0	1
0					1
1		1		X	

		<i>a</i>		0	1
		<i>b</i>		0	1
0					1
1		1			

f_1 implementation

2 groups

$$f = a'b + ab'$$

		<i>a</i>		0	1
		<i>b</i>		0	1
0					1
1		1		1	

f_2 implementation

2 groups

$$f = a + b$$

Procedure for K-map in don't-care cases

$$\text{Simplify } f(a,b,c,d) = \sum m(1,3,7,11,15) + \sum d(0,2,5)$$

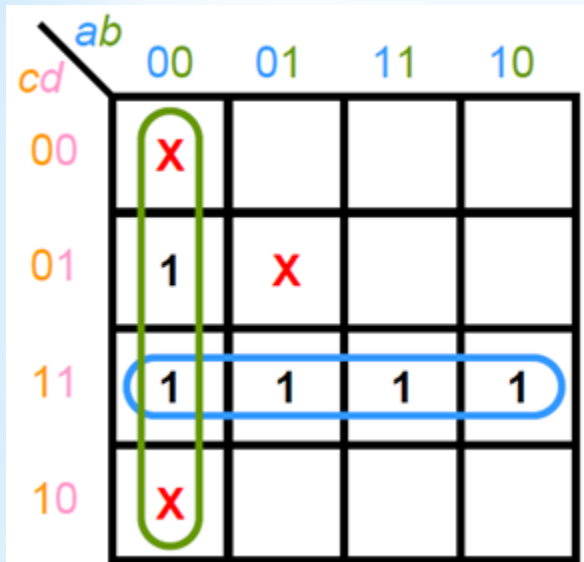
	<i>ab</i>	00	01	11	10
<i>cd</i>	00	X			
	01	1	X		
	11	1	1	1	1
	10	X			

	<i>ab</i>	00	01	11	10
<i>cd</i>	00	X			
	01	1	X		
	11	1	1	1	1
	10	X			

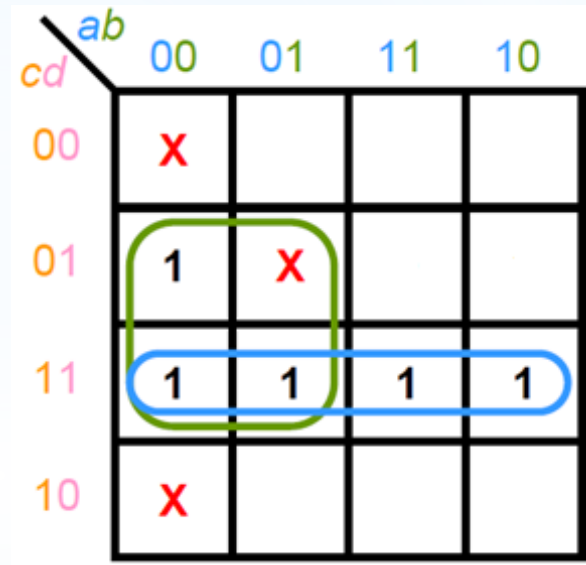
$$f(a,b,c,d) = a'b'd + cd$$

Is it a good solution?

Other solutions



$$f(a,b,c,d) = a'b' + cd$$



$$f(a,b,c,d) = a'd + cd$$



Choose to include those Xs that eliminates more literals