

EE2000 Logic Circuit Design

Minimization of Logic Functions Part 1

Outline

- 3.1 Introduction of minimization (Gate Input V.S Gate Cost)
- 3.2 Minimization using Boolean algebra
- 3.3 Karnaugh map
- 3.4 Minimization using Karnaugh map

3.1 Introduction of minimization

- Boolean algebra is a tool for simplifying / minimizing logic circuits

Example: $f = x'yz + x'yz' + xz$

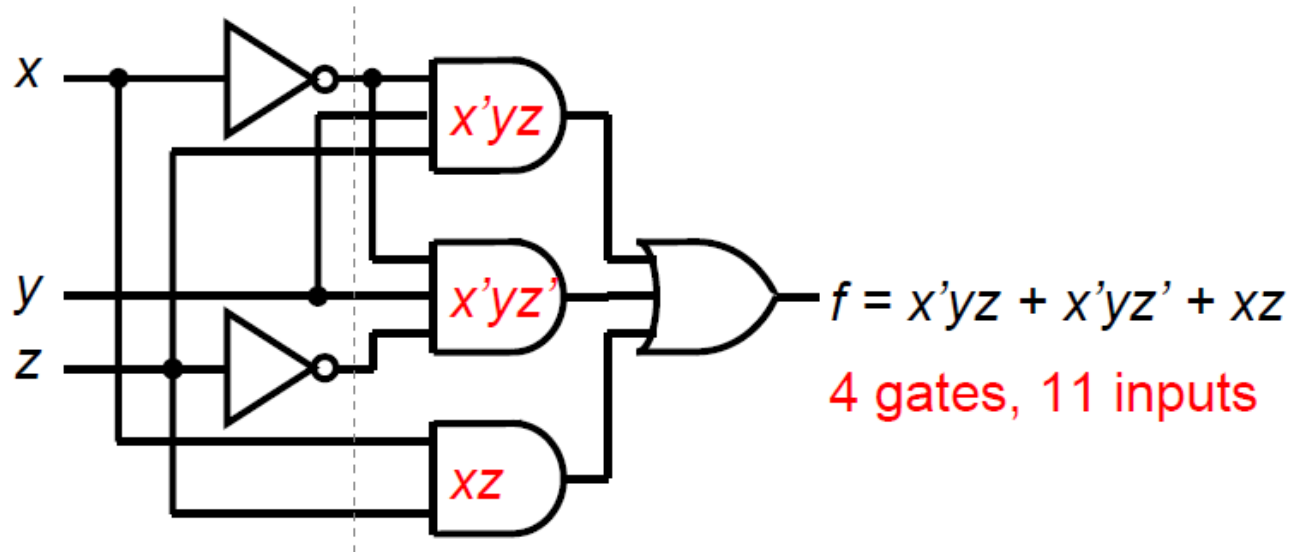
can be simplified as

$$f = x'y + xz$$

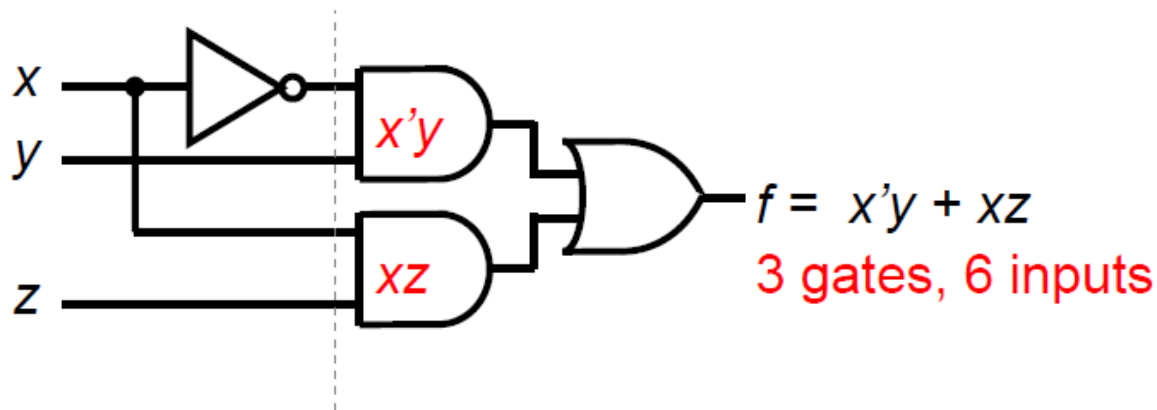
Result:

Reduced f from sum of 3 products to sum of 2 product terms.

Logic circuit with minimization



|| equivalent



Advantages of minimization

- Obtain a simple (or simplest) logic circuit
- Reduce the cost of circuit

Cost in logic circuit

- **Gate** cost (number of gates in the implementation)
- **Gate-input** cost (number of inputs to the gates)
- Total cost = **Gate** cost + **Gate-input** cost

Gate-input cost

- The number of gate-input is proportional to the number of transistors in the logic circuit
- The cost can be determined by checking logic diagram / schematic and the Boolean function

$$F_1 = abcd + a'b'c'd'$$

$$F_2 = (a' + b)(b' + c)(c' + d)(d' + a)$$

F_1 has 3 no. of gate and 10 no. of gate-input

Total cost = 13

F_2 has 5 no. of gate and 12 no. of gate-input

Total cost = 17

Cost Reduction

The key of simplifying logic functions is to reduce the **no. of terms** and **no. of literals**

↓no. of literals = ↓ no. of gate inputs
↓ no. of terms = ↓ no. of gates

In addition, costs in space and power consumption can be reduced.

3.2 Minimization using Boolean algebra

- e.g. Given 4 equivalent Boolean functions f_1 to f_4 expressed in SOP form already (to be proved in next chapter).
 - $f_1(a,b,c) = a'bc' + a'bc + ab'c' + ab'c + abc$ (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 (the canonical sum form) to f_3 (the MSP form)?

Simplification

$$\blacksquare f_1(a,b,c) = a'bc' + a'bc + ab'c' + ab'c + abc$$

$$\blacksquare = (a'bc' + a'bc) + (ab'c' + ab'c) + abc$$

$$\blacksquare = a'b + ab' + abc = f_2$$

$$\blacksquare = a'b + a(b' + bc)$$

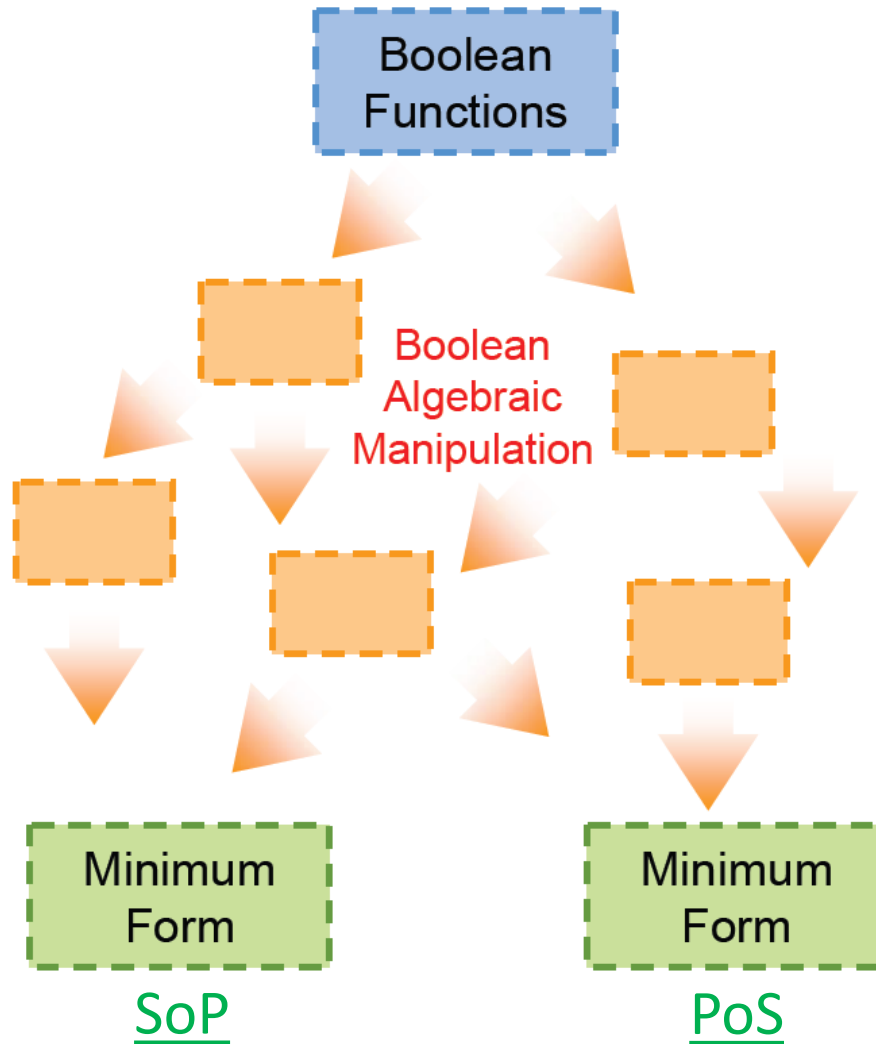
$$\blacksquare = a'b + a(b' + c)$$

$$\blacksquare = a'b + ab' + ac$$

$$\blacksquare = f_3$$

How to obtain f_4 ?

Simplification



Properties of Boolean algebra:

Absorption
Redundancy
DeMorgan
Consensus
etc...

3.3 Karnaugh Map

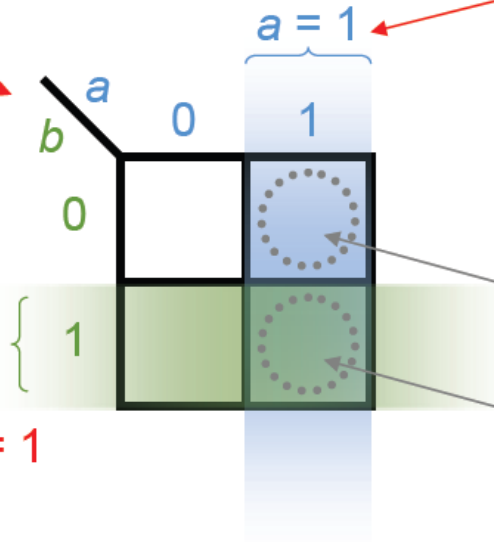
- In 1953, Maurice Karnaugh introduced a map method known as **Karnaugh map (K-map)**
 - A straightforward procedure for minimizing Boolean functions in a table form
 - Graphical presentation of a truth table
 - Minterm is used in the cell of the K-map
 - It is n -variable function (defined by 2^n):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

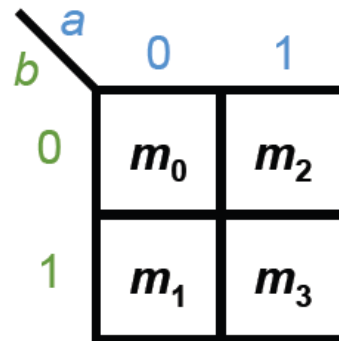
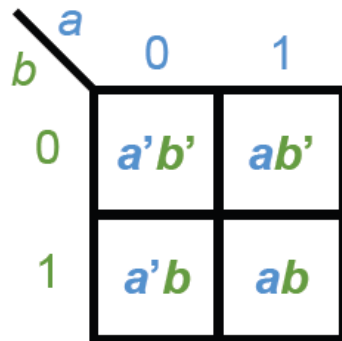
This column represents $a = 1$

This row represents $b = 1$



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

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



Minterm representations

Plotting functions in K-map

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

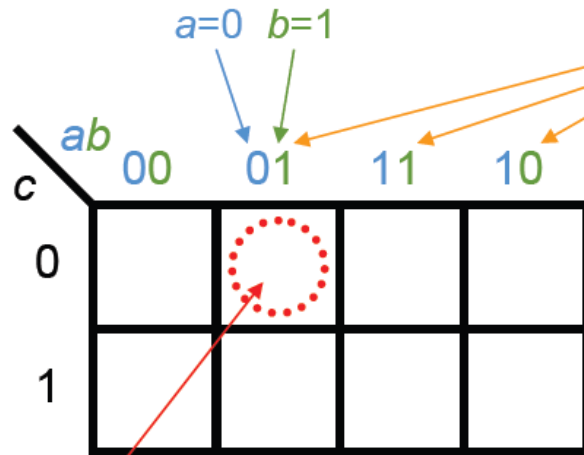
$a \backslash b$	0	1
0	1	1
1	0	0

or

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

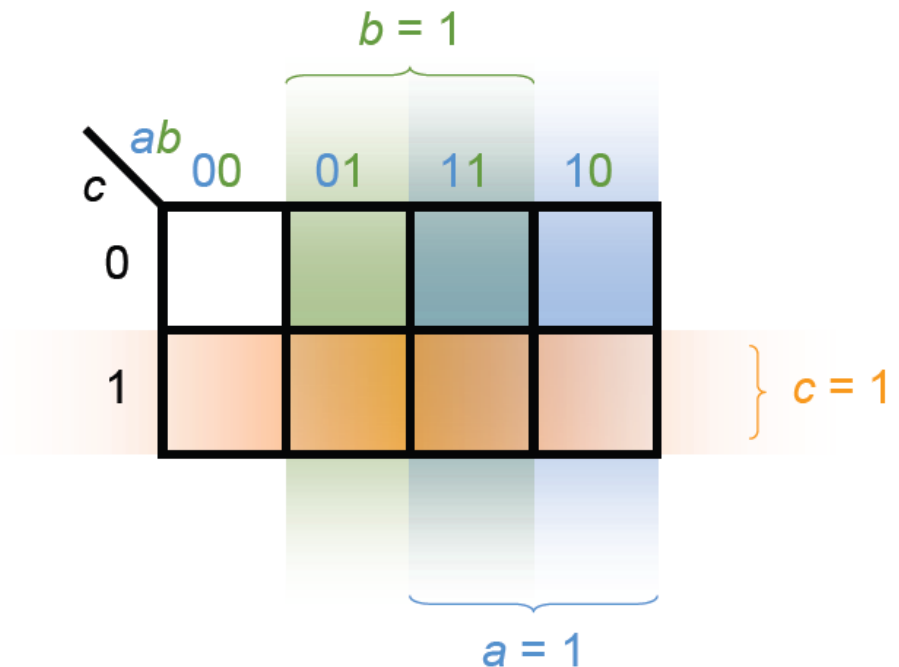
Functions represented graphically with corresponding minterm cells labeled to value 1

Three-variable K-map



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

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



Minterm representations

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

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

Gray code in K-map

- Adjacent cells have 1-bit (1-variable) difference only
 - e.g. 00, 01, 10, 11 (two-variable)
 - ab & ab' have 1-variable difference only!
- Any two adjacent cells sharing a common edge can form a pair of adjacent binary combinations
- This property can be used to simplify the product terms
- By this rule, we can group adjacent 1's on the map to form simplified product terms

Simplification of Product Terms

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

$c \backslash ab$	00	01	11	10
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:

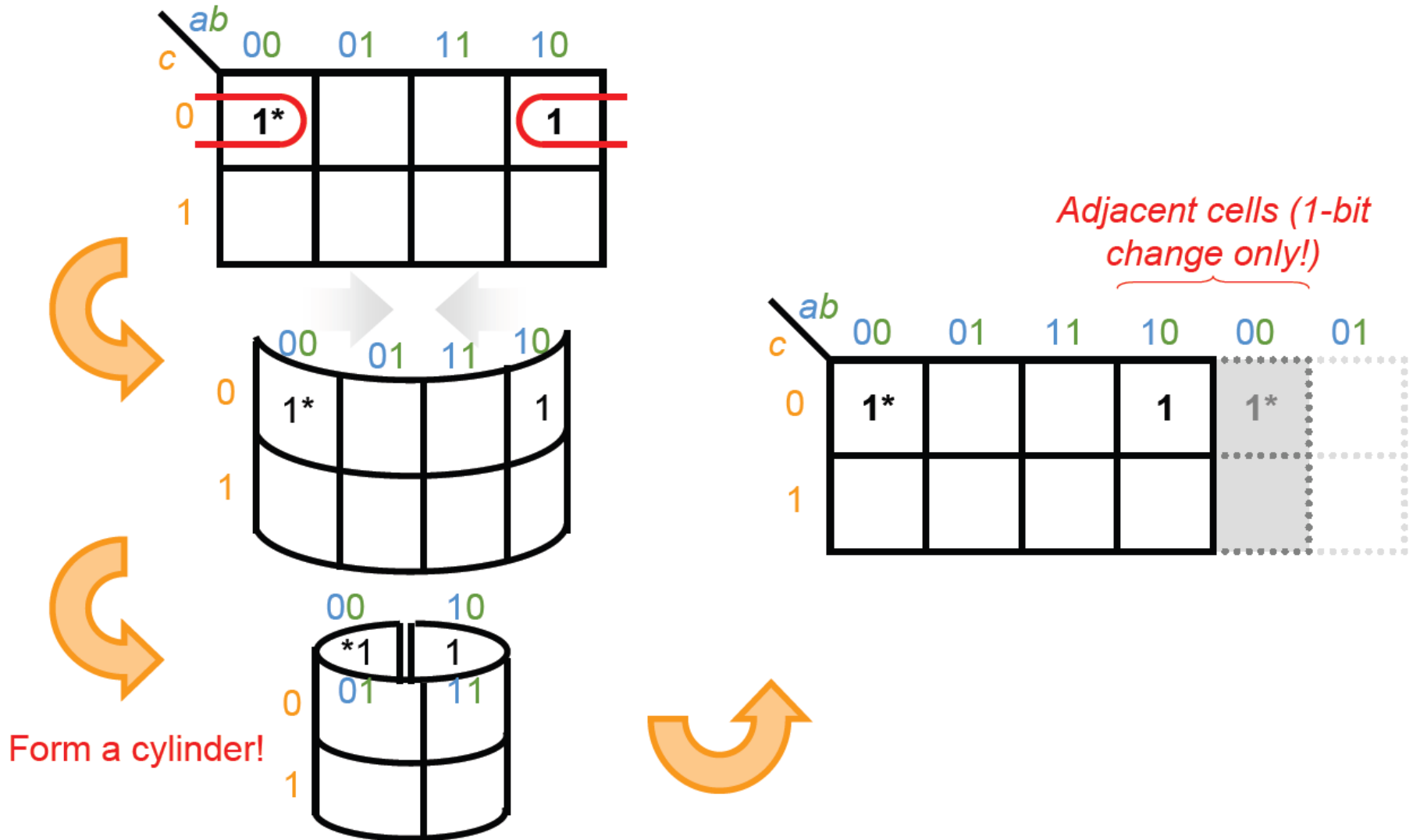
$$f(a, b, c) = abc' + abc$$

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

Only one-variable difference

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

Wrap-around Adjacency



More examples

c \ ab	00	01	11	10
0			1	1
1				

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

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

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

c \ ab	00	01	11	10
0	1			1
1				

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

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

Also one-variable difference!

If We Don't Use Gray Code...

We can group these two adjacent 1's:

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

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

$$= bc'$$

But not these two: *Incorrect arrangement*

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

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

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

two-variable difference!

Cannot be simplified into a single term!

Format of three-variable

Label rows with first variable,
columns with the others

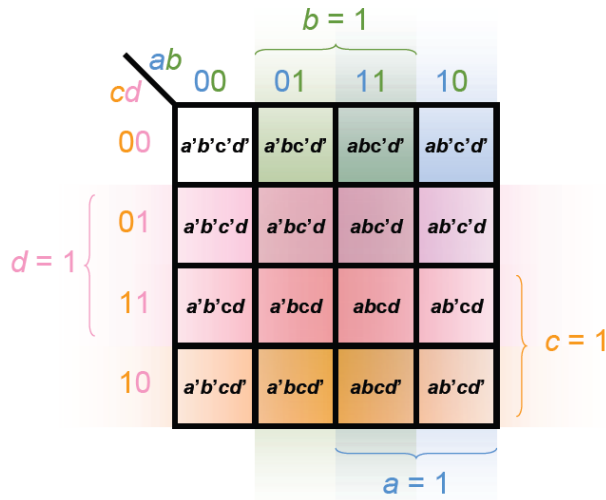
a	bc			
	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

Vertical orientation of
three-variable K-map

bc	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-Maps, we use the same
method to group the adjacent 1's!

Four-variable K-map

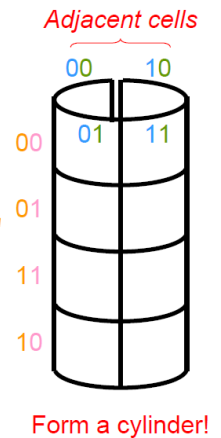
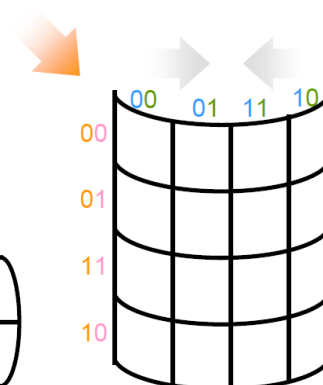
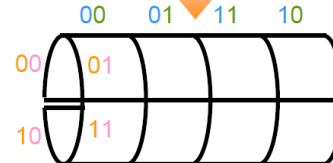


cd \ ab	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

Wrap-around Adjacency

Note the Gray code order of the rows and columns

cd \ ab	00	01	11	10
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}

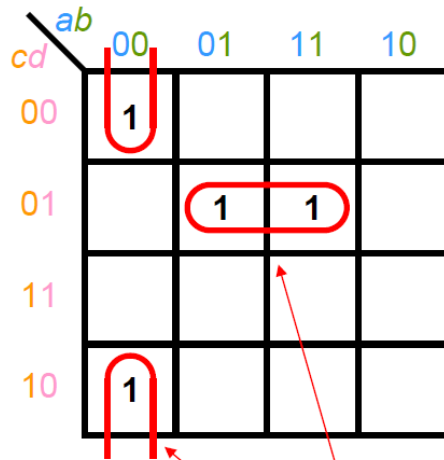


Imagine the Map as...

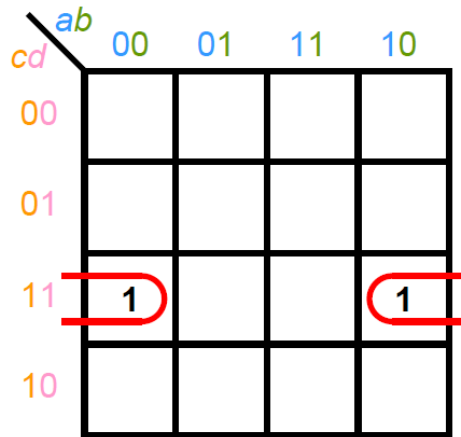
1-bit (1-variable) change only for every adjacent cells!

cd		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									

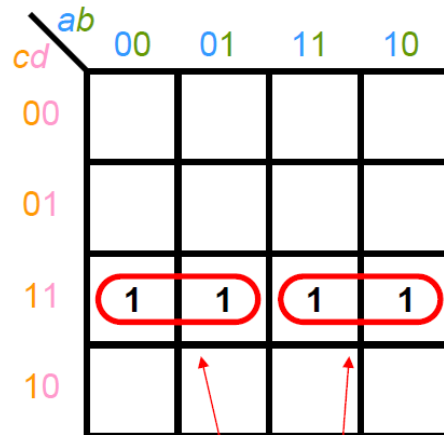
Examples of 4-variable K-map



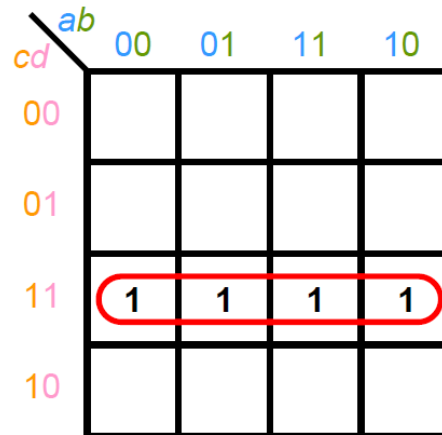
$$f(a, b, c, d) = a'b'd' + bc'd$$



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

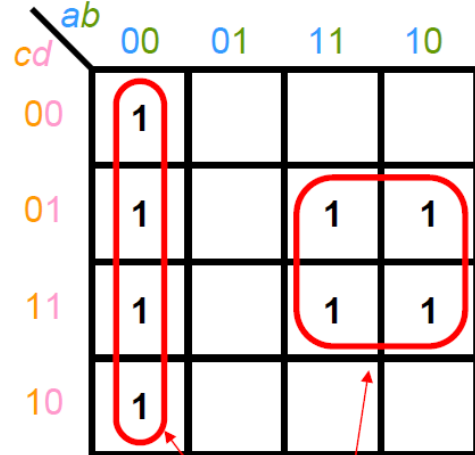


$$f(a, b, c, d) = a'cd + acd \\ = cd$$

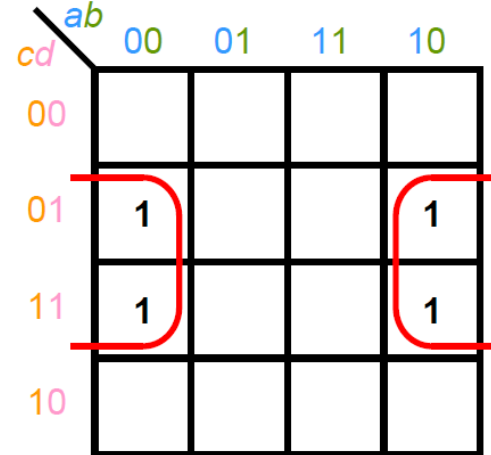


$$f(a, b, c, d) = cd$$

Examples of 4-variable K-map

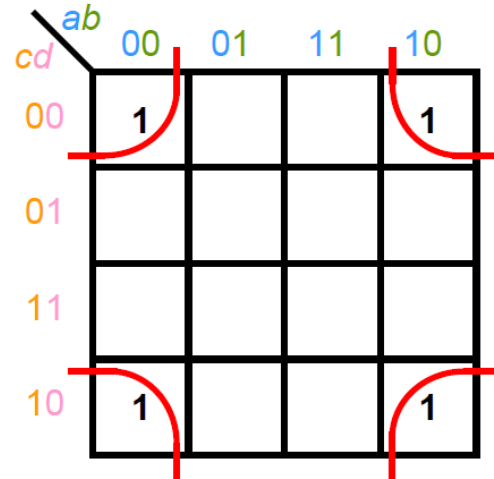


$$f(a, b, c, d) = a'b' + ad$$



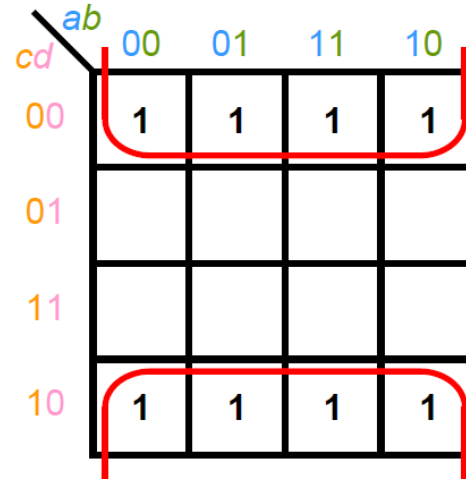
$$f(a, b, c, d) =$$

Across 4 corners:



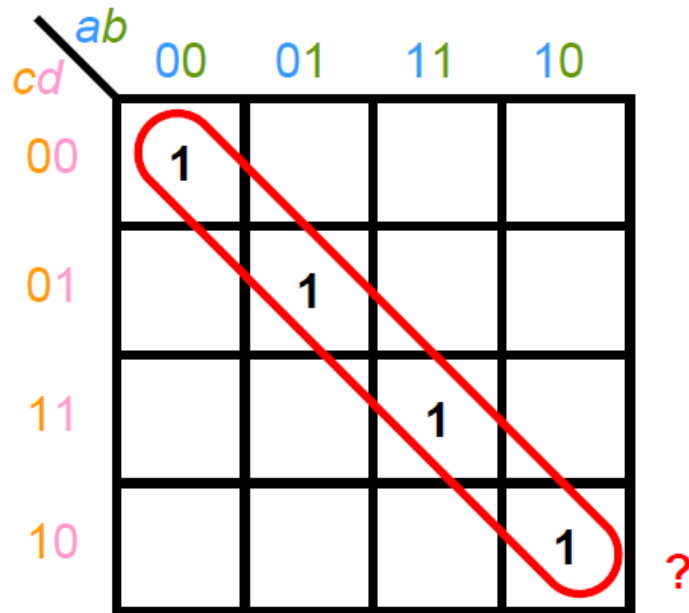
$$f(a, b, c, d) =$$

Group of eight:

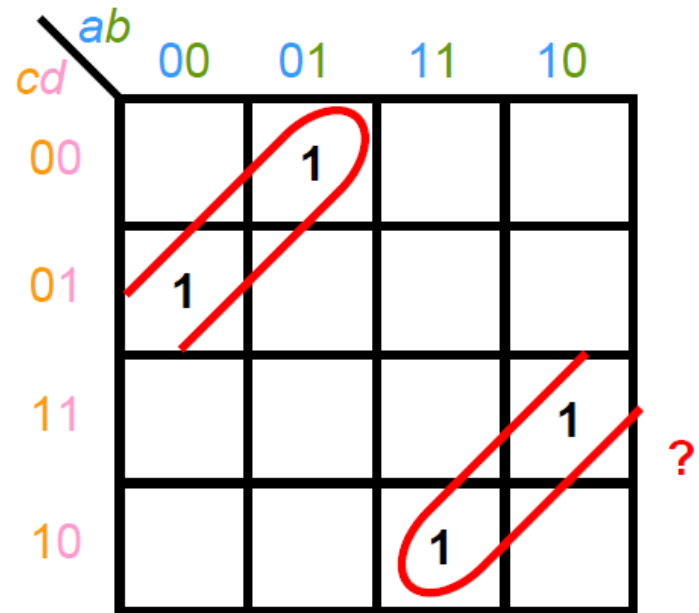


$$f(a, b, c, d) =$$

Are They Adjacent Cells?

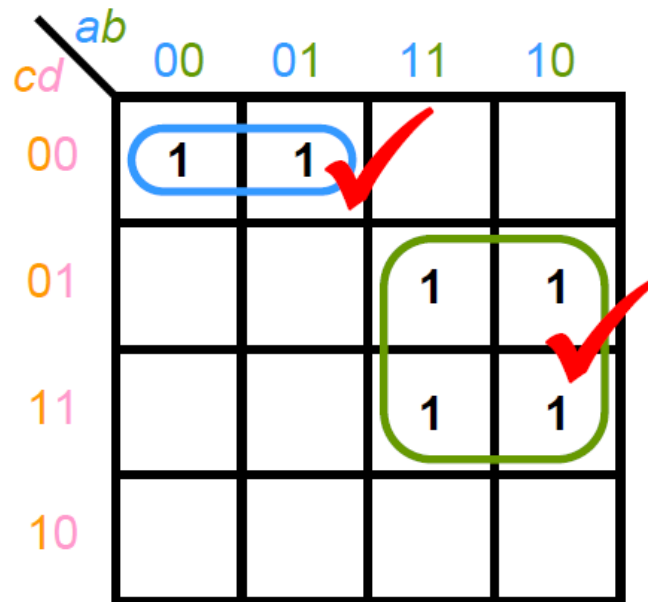


Diagonal?

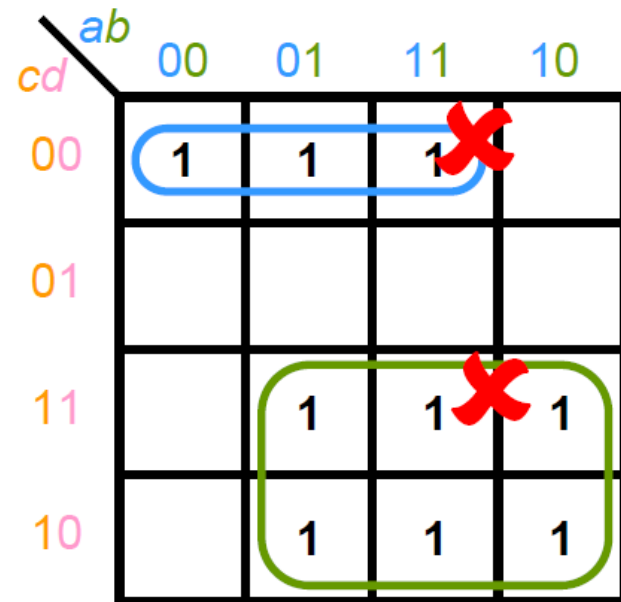


Magic square?

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

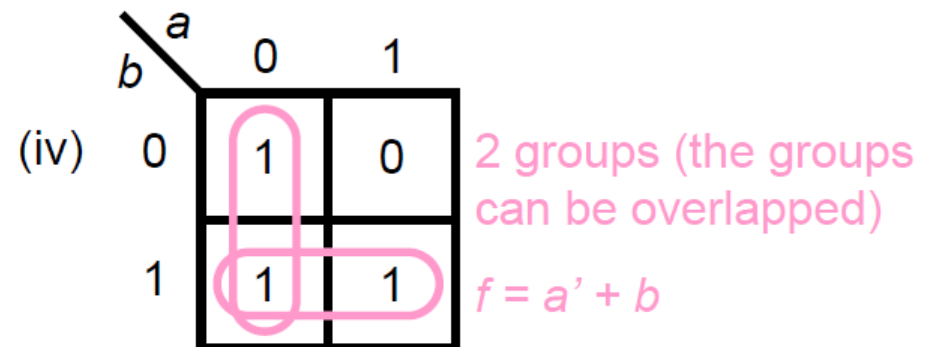
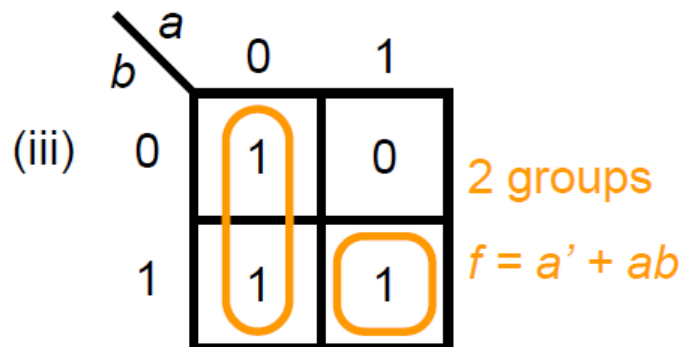
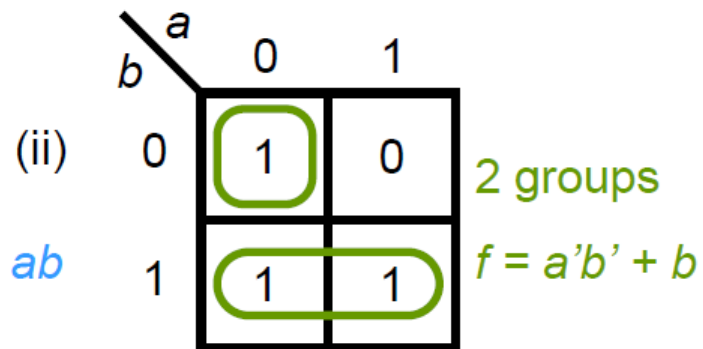
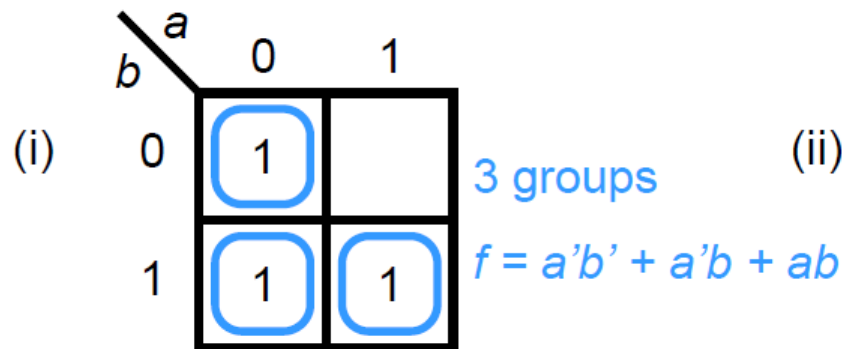
3.4 Minimization using Karnaugh map

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

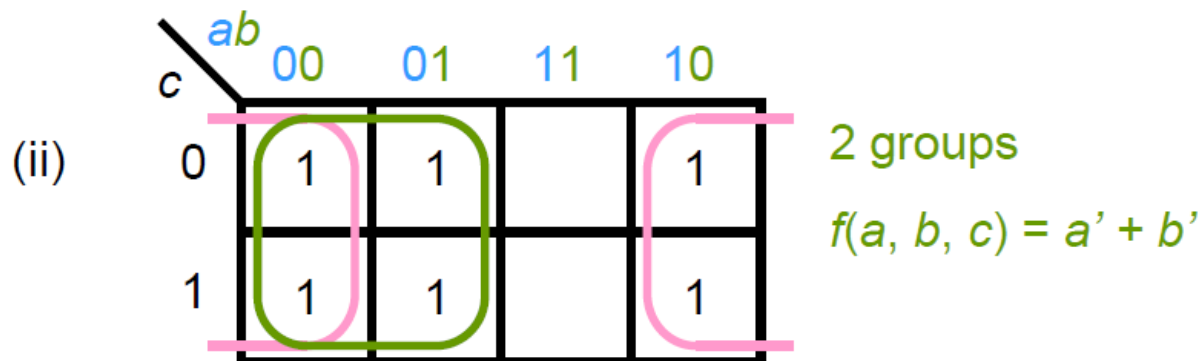
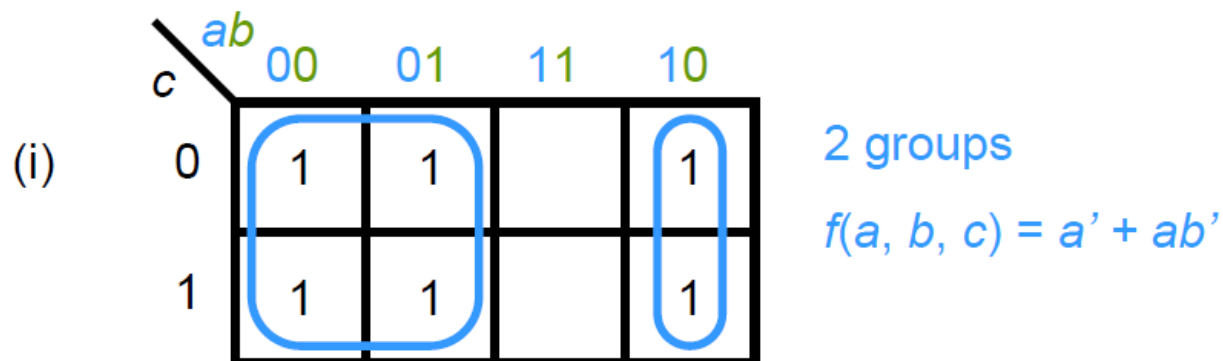
Many ways to group them. Which is the best solution?



Example: Three-variable K-map

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

Which solution is better?

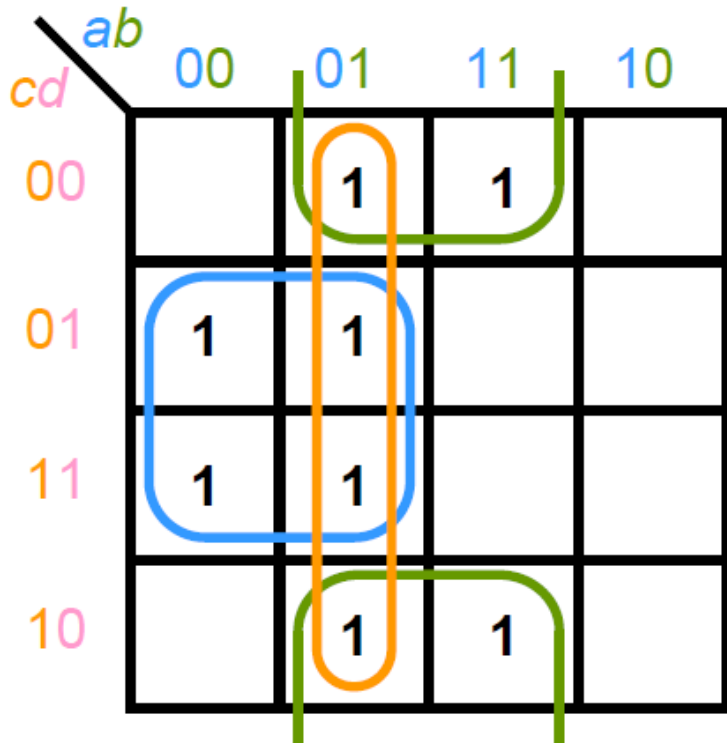


Example: Four-variable K-map

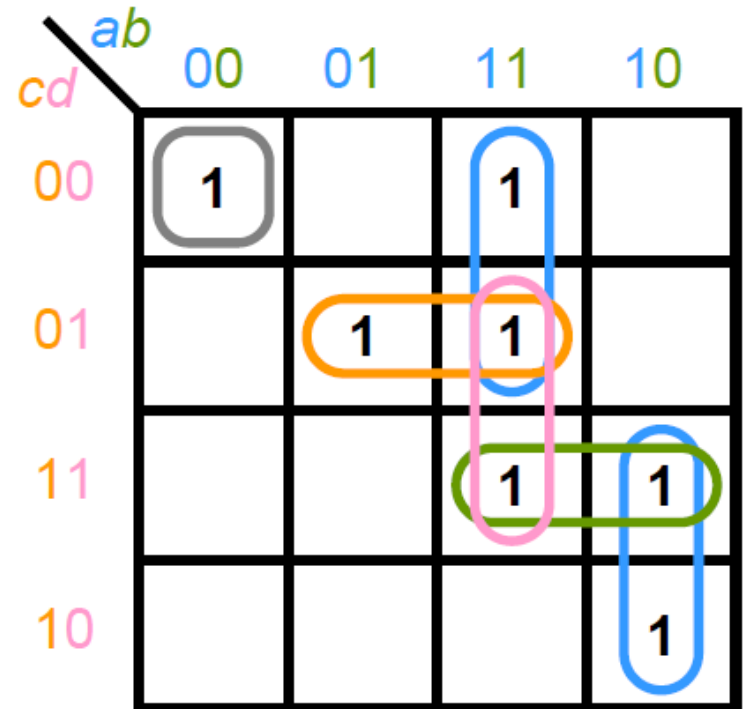
Simplify $f(a, b, c, d) = \sum m(0, 1, 2, 3, 4, 5, 7, 8, 10, 11, 15)$

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

Grouping of K-map



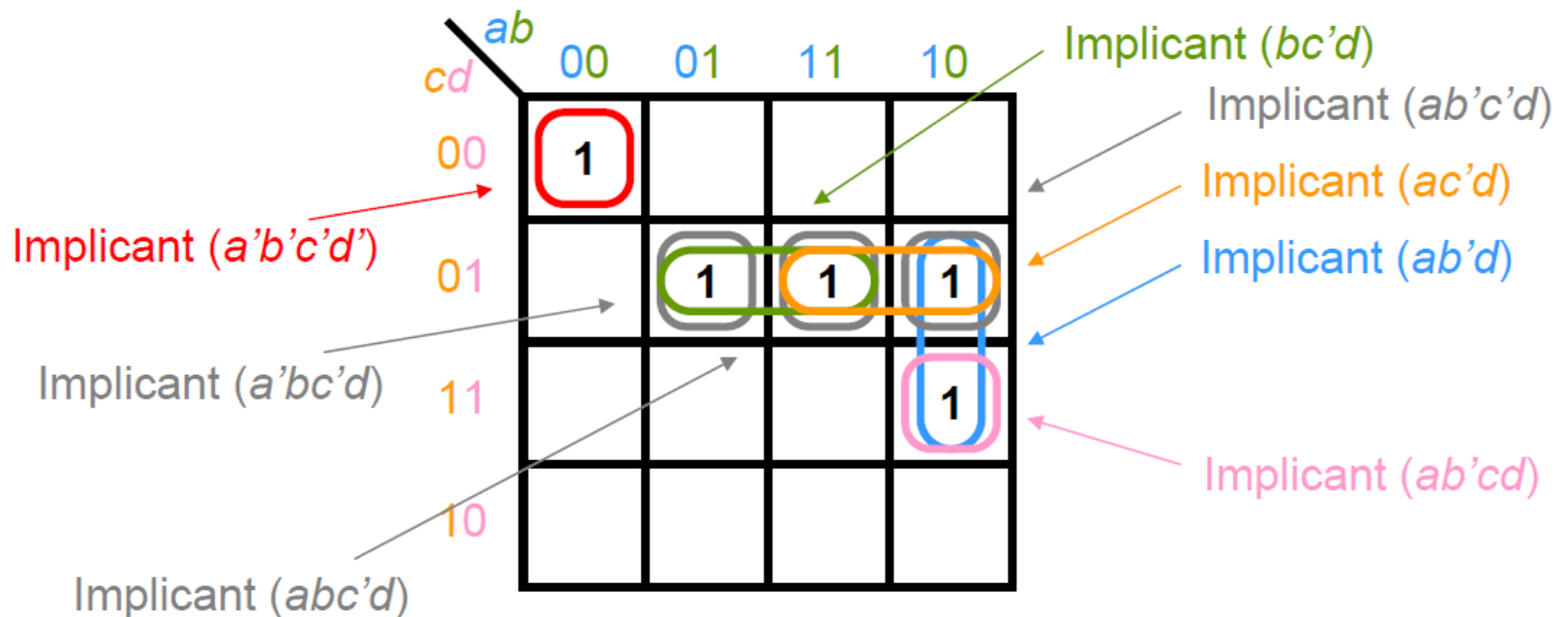
Three groups overlapped!



Too many overlaps!

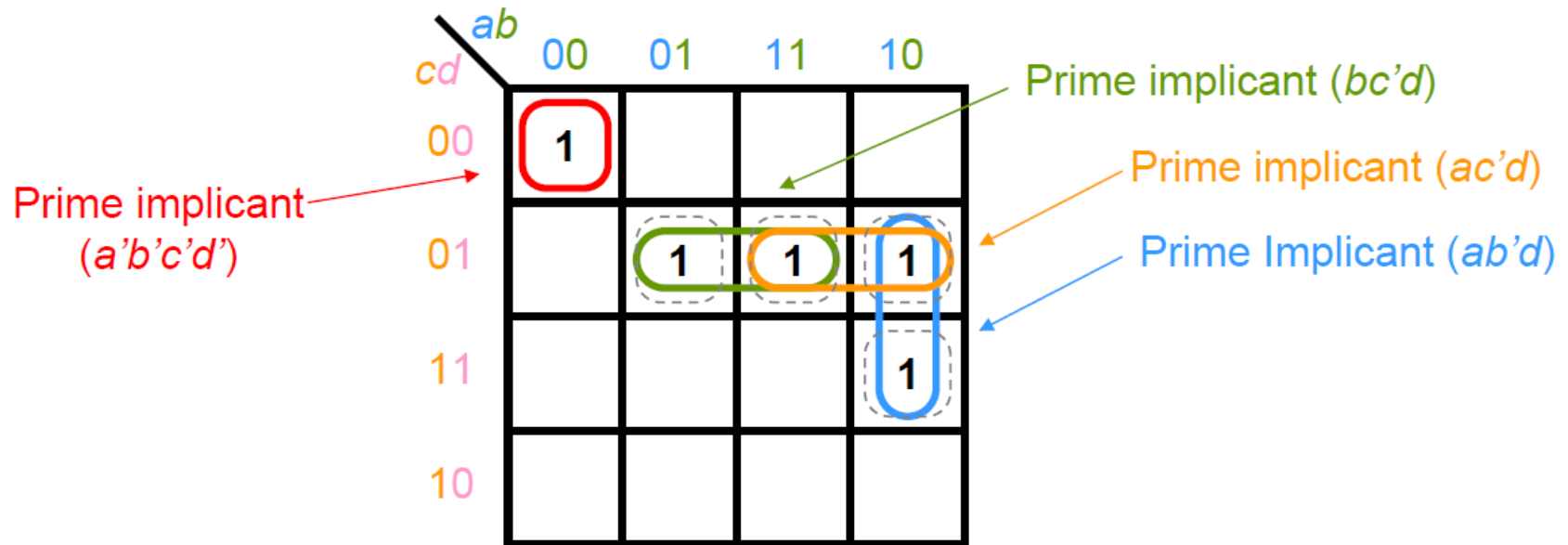
Terminology: Implicant

- The product term is an **implicant** of a function if the function has the value 1 for all minterms of the product term



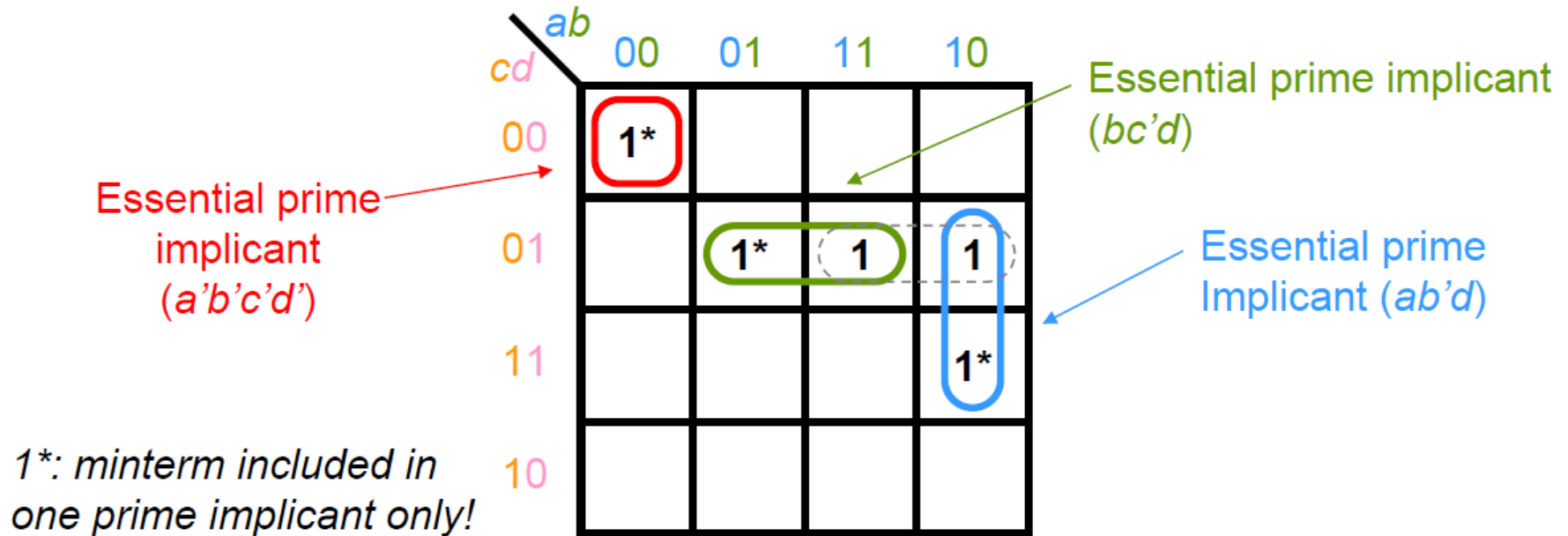
Prime Implicant (PI)

- If the removal of any literal from an implicant P results in a product term that is not an implicant of the function, P is a **prime implicant**



Essential Prime Implicant

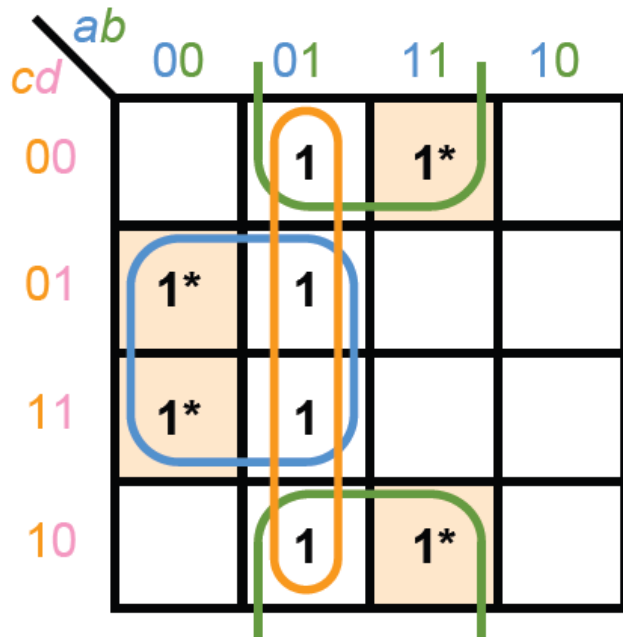
- If a minterm of a function is included in only one prime implicant, that prime implicant is **essential prime implicant**



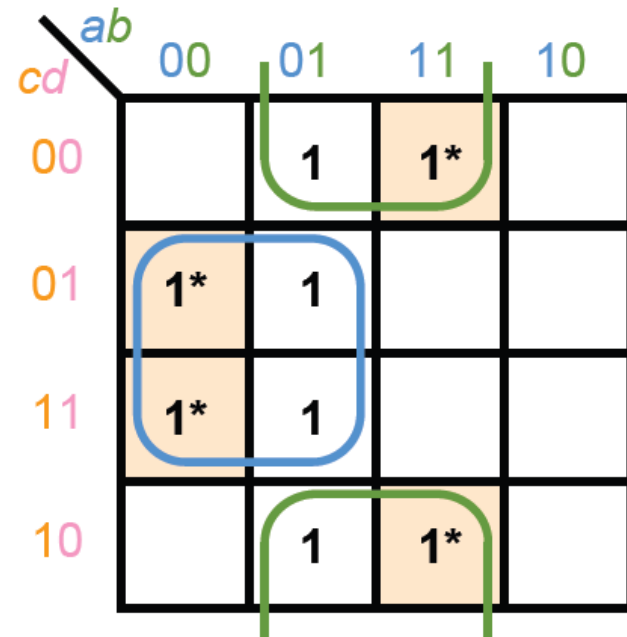
Systematic Procedure

- How to satisfy rule (1) and (2)?
- To find the minimized expression from the K-map
 - Step 1) First determine all Pls & EPls
 - Step 2) Select the EPls
 - Step 3) If there are remaining minterms, select the Pls that including them

Back to the Previous Example



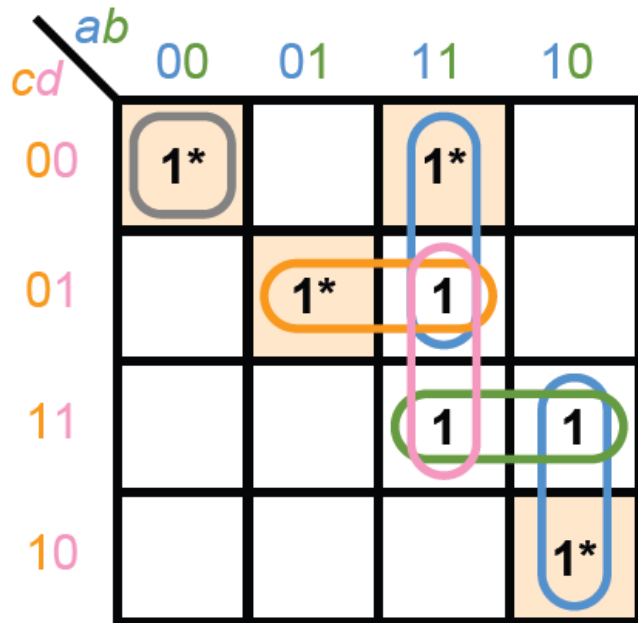
PIs: 
 EPI: 



Select the essential prime implicants
and no remaining minterms left!

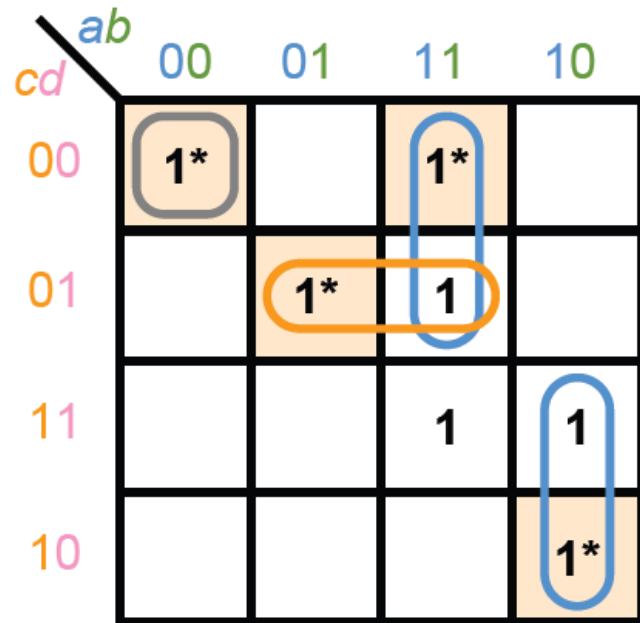
$$f(a, b, c, d) = a'd + bd'$$

Back to the Previous Example



PIs:

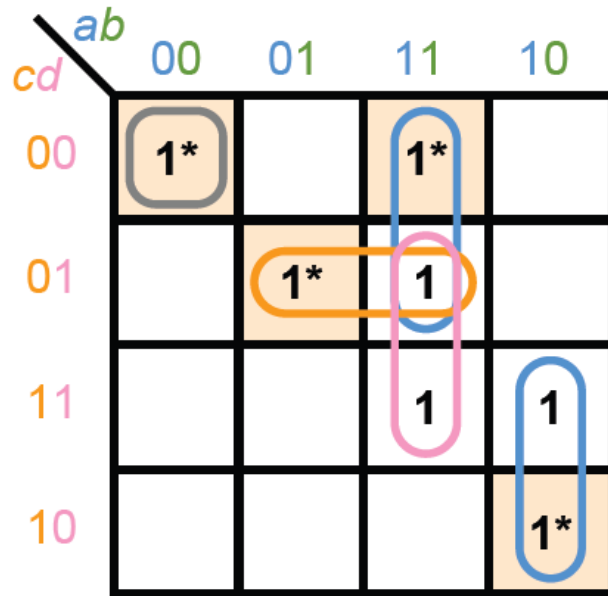
EPIs:



Select the essential prime implicants first

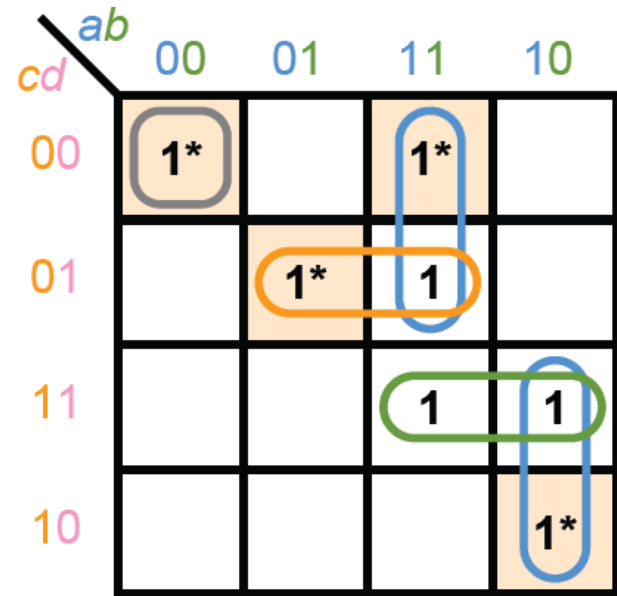
Still have a remaining minterm

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 