EE2000 Logic Circuit Design

Chapter 2 – Minimization of Logic Functions

Outline

- 2.1 Minimization using Boolean algebra
- 2.2 Karnaugh map
- 2.3 Minimization using Karnaugh map
- 2.4 Boolean functions with Don’t’-care cases
- 2.5 Minimization using Quine-McCluskey method

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 Reduce

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

$\downarrow$no. of literals = $\downarrow$ no. of gate inputs
$\downarrow$ no. of terms = $\downarrow$ no. of gates

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

2.1 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

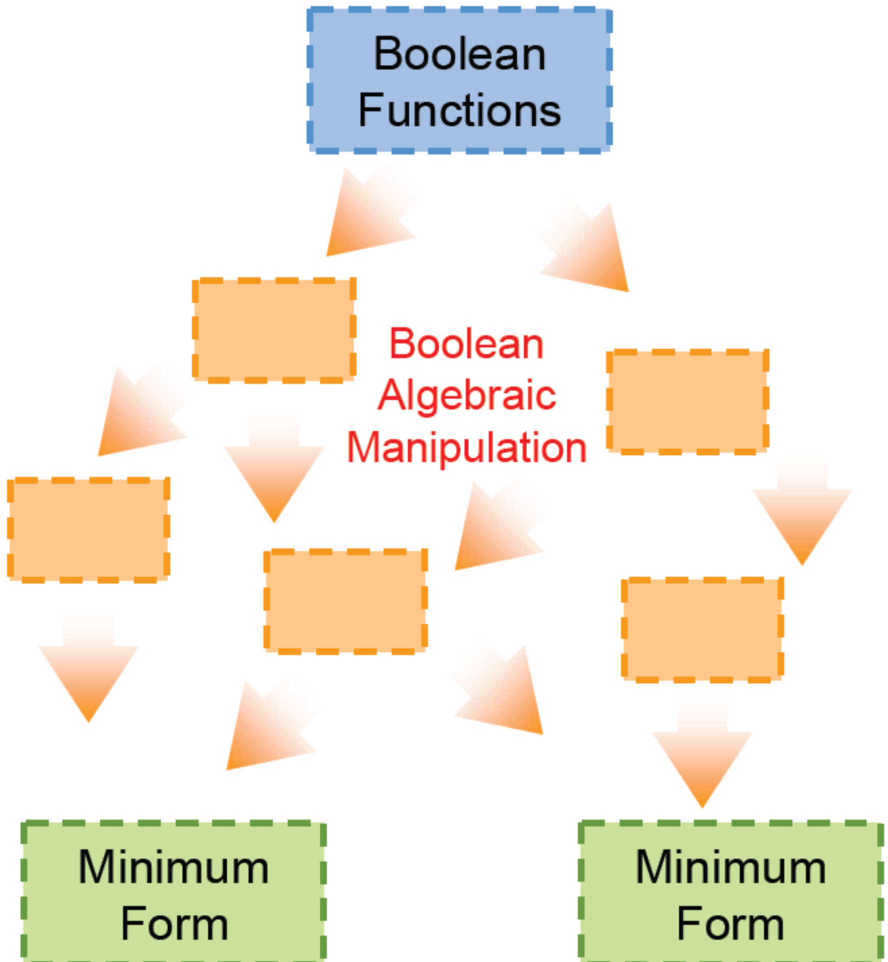
- How can you simplify f_1 (the canonical sum form) to f_3 (the MSP form)?

Simplification

- $f_1(a,b,c) = a'bc' + a'bc + ab'c' + ab'c + abc$
- $= (a'bc' + a'bc) + (ab'c' + ab'c) + abc$
- $= a'b + ab' + abc = f_2$
- $= a'b + a(b' + bc)$
- $= a'b + a(b' + c)$
- $= a'b + ab' + ac$
- $= f_3$

How to obtain f_4?

Simplification



MSP MPS

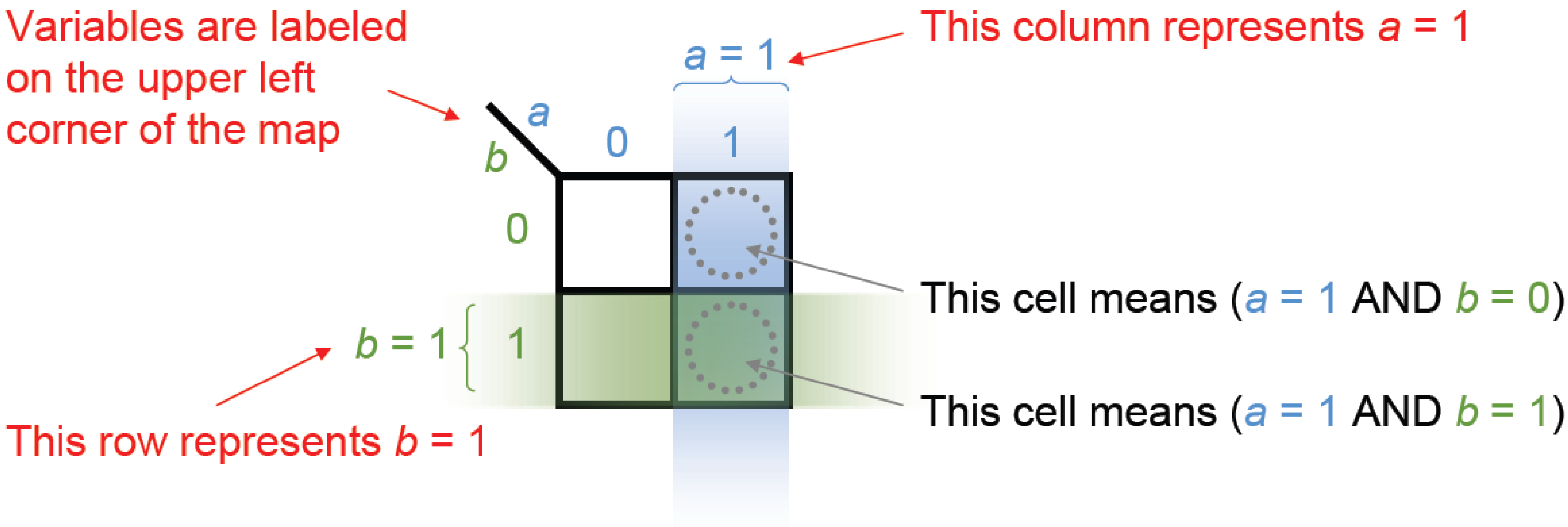
Properties of Boolean algebra:

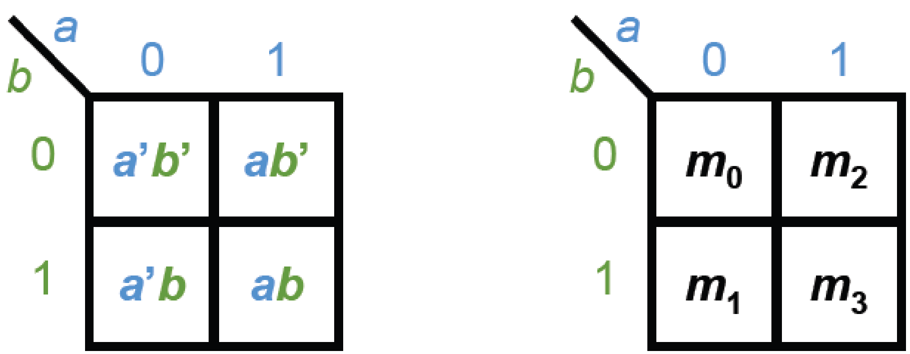
Absorption
Redundancy
DeMorgan
Consensus
etc...

2.2 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





Minterm representations

Plotting functions in K-map

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

b \ a	0	1
0		
1		

or

b \ a	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)

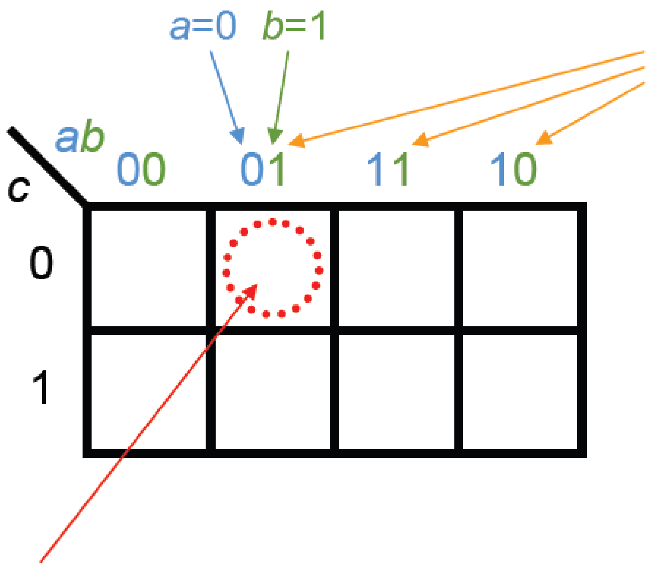
b \ a	0	1
0	1	1
1	0	0

or

b \ a	0	1
0	1	1
1		

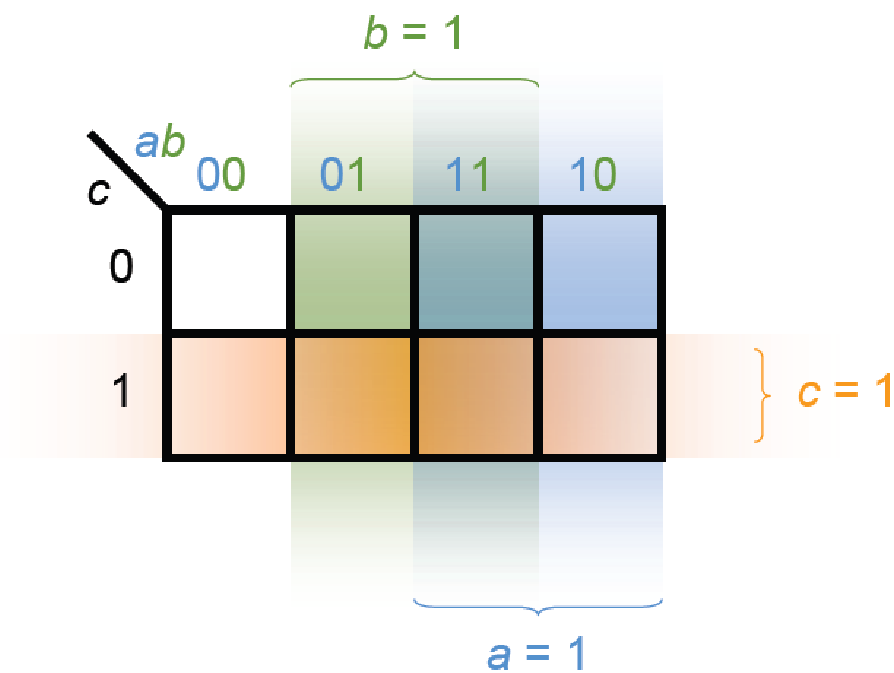
Functions represented graphically with corresponding minterm cells labeled to value 1

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 $f(a, b, c) = \Sigma\, m(6, 7)$

c \ 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 Albegra:

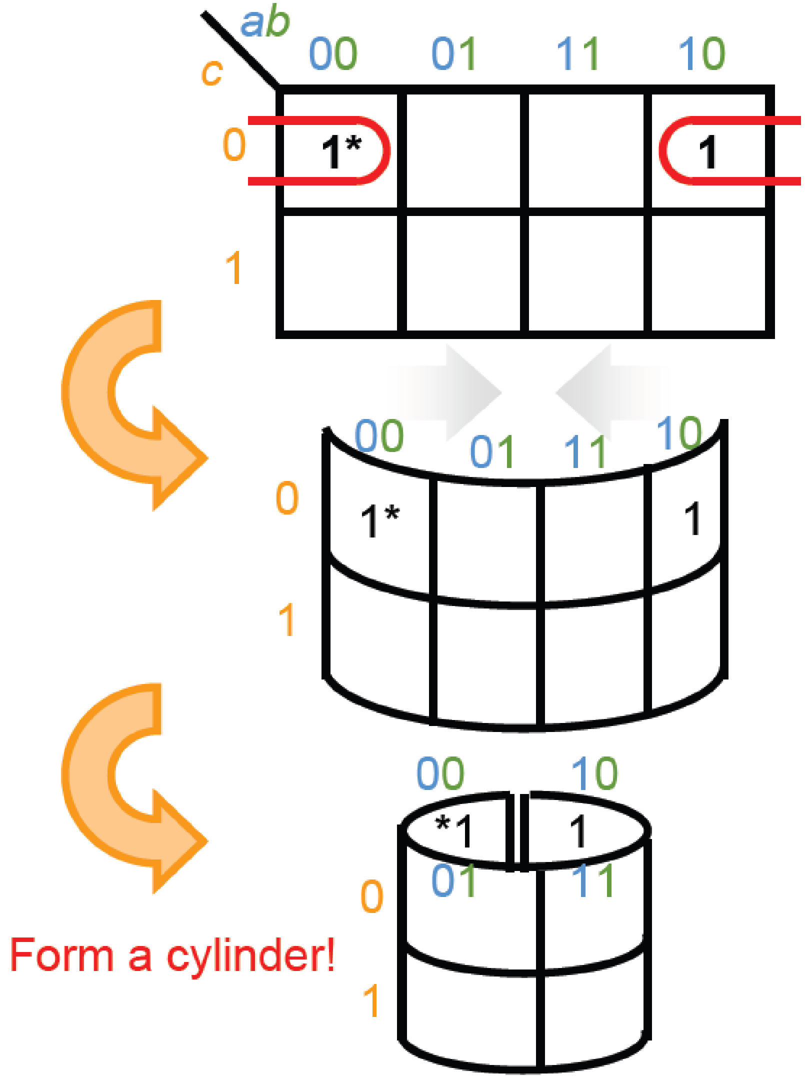
Only one-variable difference

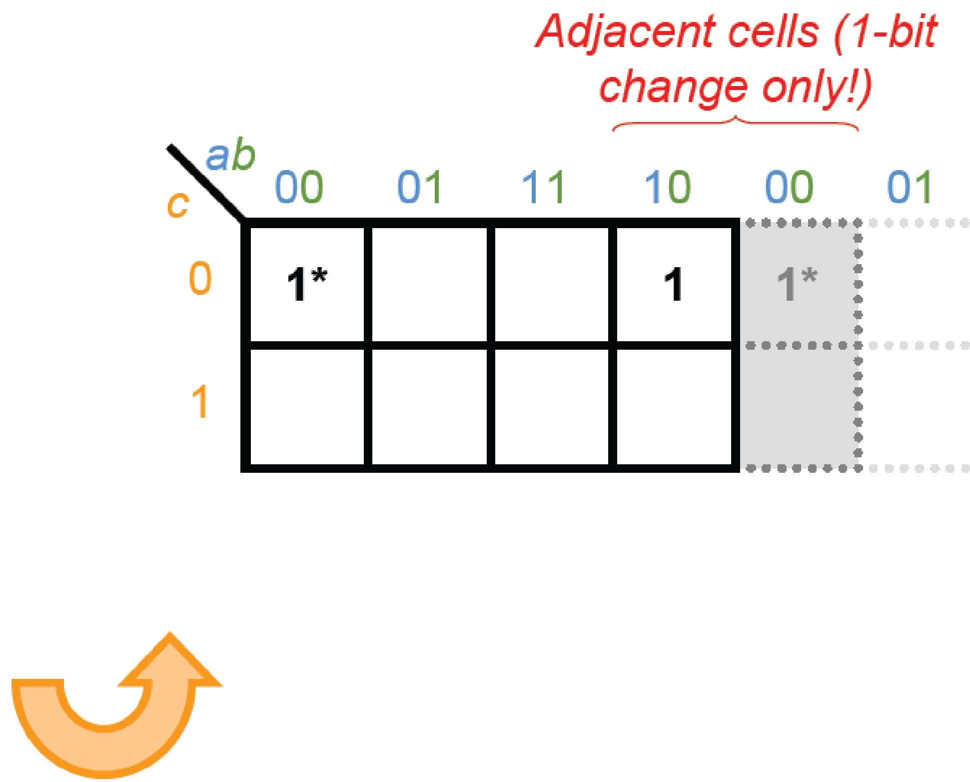
$f(a, b, c) = \overbrace{abc' + abc}$

$= ab$ *(adjacency)*

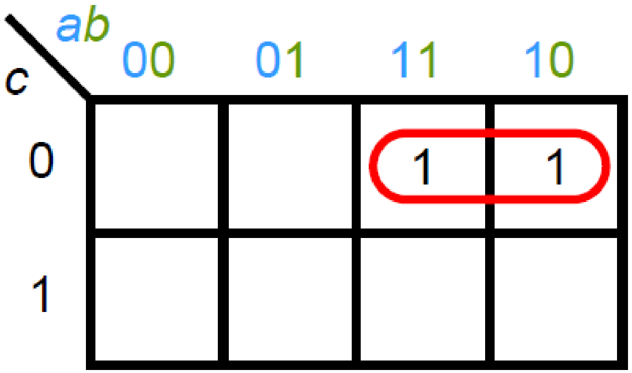
- Rule: whenever we group two adjacent cells, they can form a product term with one less variable

Wrap-around Adjacency





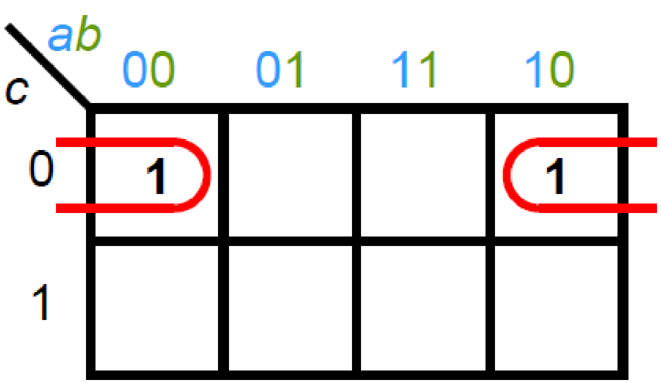
More examples



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

$= ac'$ *(adjacency)*

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



$f(a, b, c) = \overbrace{a'b'c' + ab'c'}^{\text{Also one-variable difference!}}$

$= b'c'$ *(adjacency)*

If We Don't Use Gray Code…

We can group these two adjacent 1's:

c \ ab	00	01	11	10
0		1	1	
1				

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

$= bc'$

But not these two:

Incorrect arrangement

c \ ab	00	01	10	11
0		1	1	
1				

two-variable difference!

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

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

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

b = 1

cd \ ab	00	01	11	10
00	a'b'c'd'	a'bc'd'	abc'd'	ab'c'd'
01	a'b'c'd	a'bc'd	abc'd	ab'c'd
11	a'b'cd	a'bcd	abcd	ab'cd
10	a'b'cd'	a'bcd'	abcd'	ab'cd'

d = 1

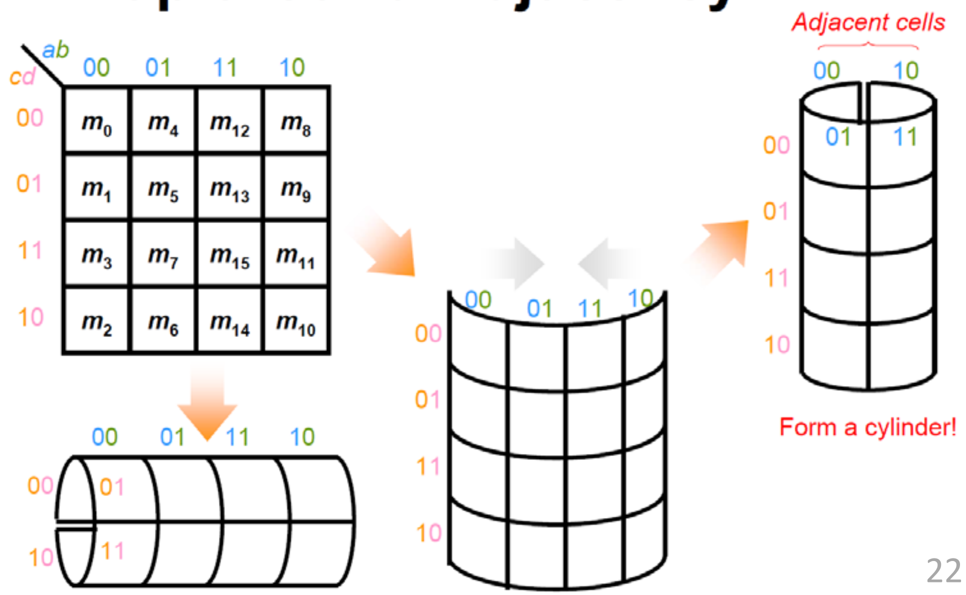
c = 1

a = 1

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}

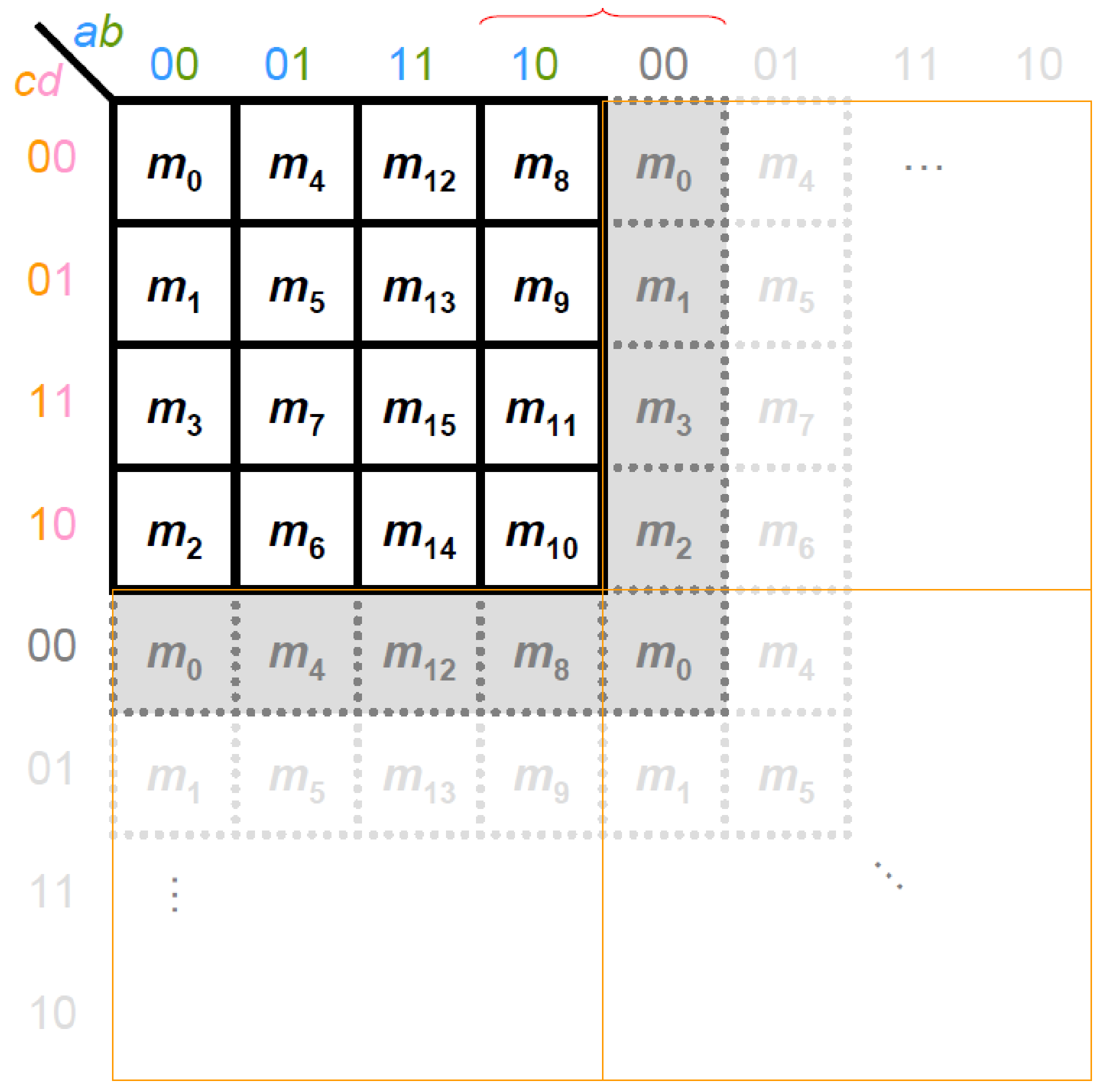
Note the Gray code order of the rows and columns

Wrap-around Adjacency

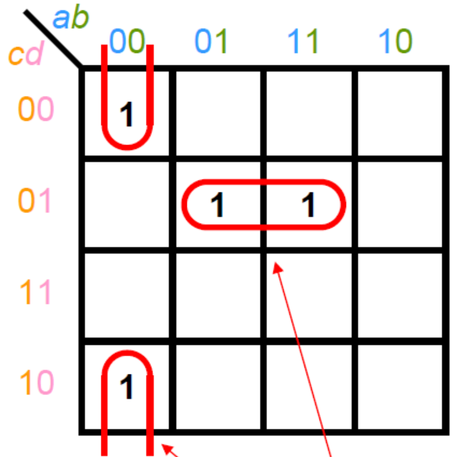


Imagine the Map as…

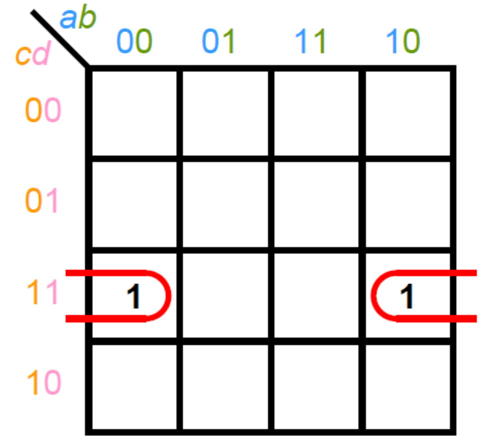
1-bit (1-variable) change only for every adjacent cells!



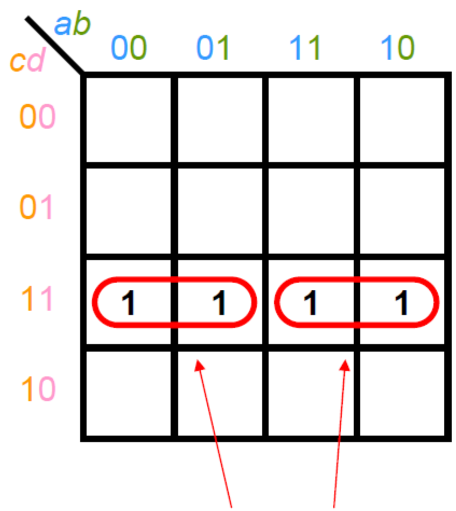
Examples of 4-variable K-map



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

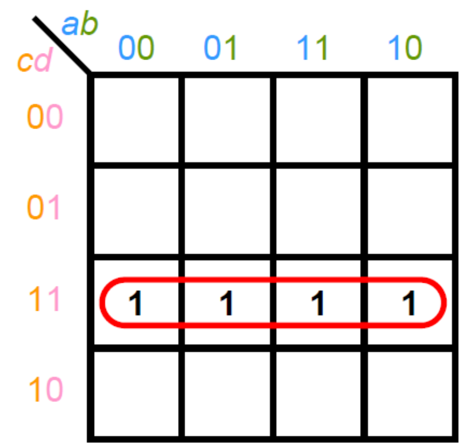


$f(a, b, c, d) =$



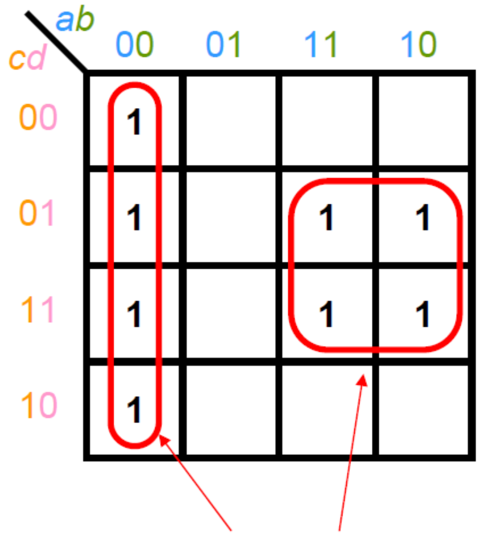
$f(a, b, c, d) = a'cd + acd$

$= cd$

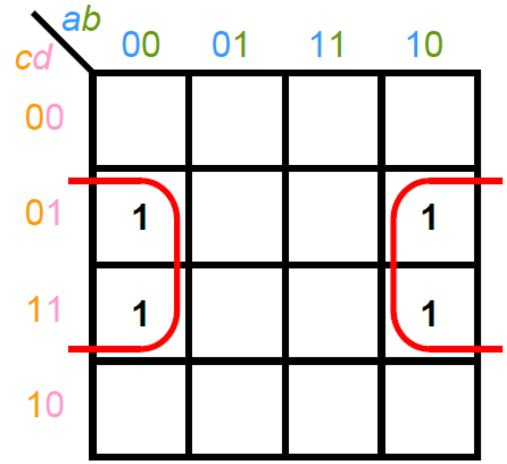


$f(a, b, c, d) = cd$

Examples of 4-variable K-map

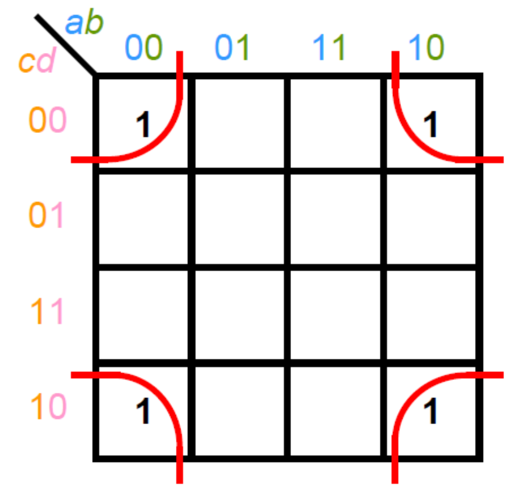


f(a, b, c, d) = a'b' + ad



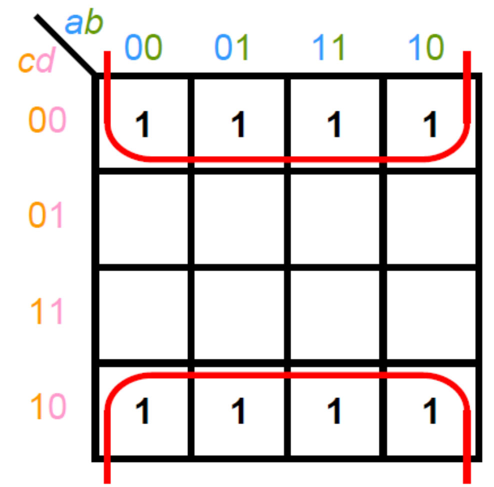
f(a, b, c, d) =

Across 4 corners:



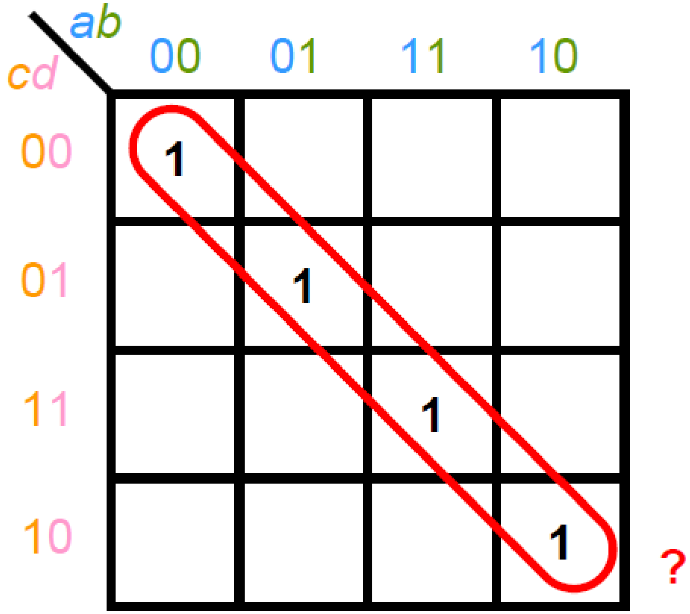
f(a, b, c, d) =

Group of eight:

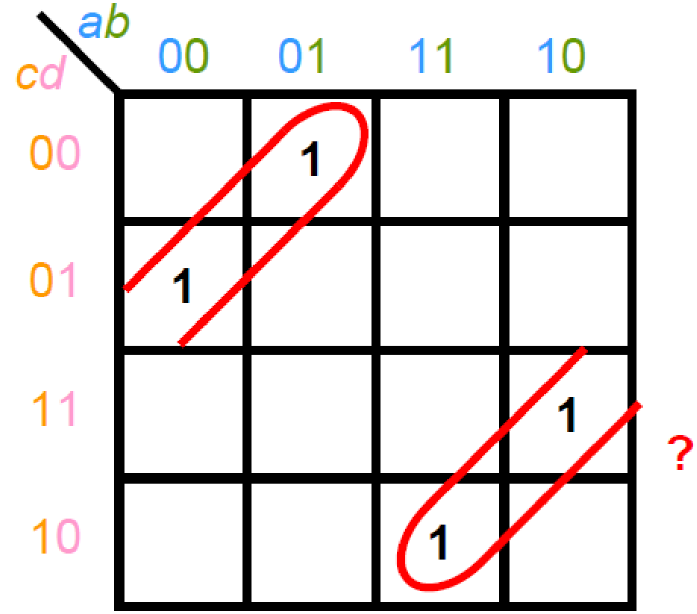


f(a, b, c, d) =

Are They Adjacent Cells?

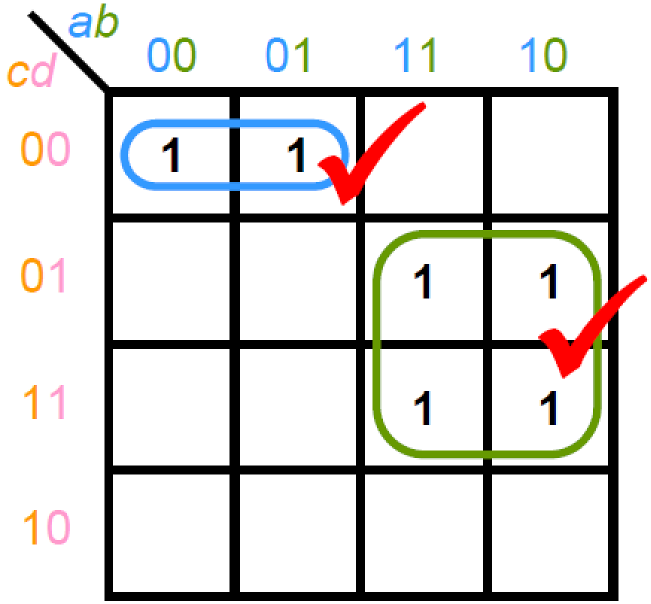


Diagonal?

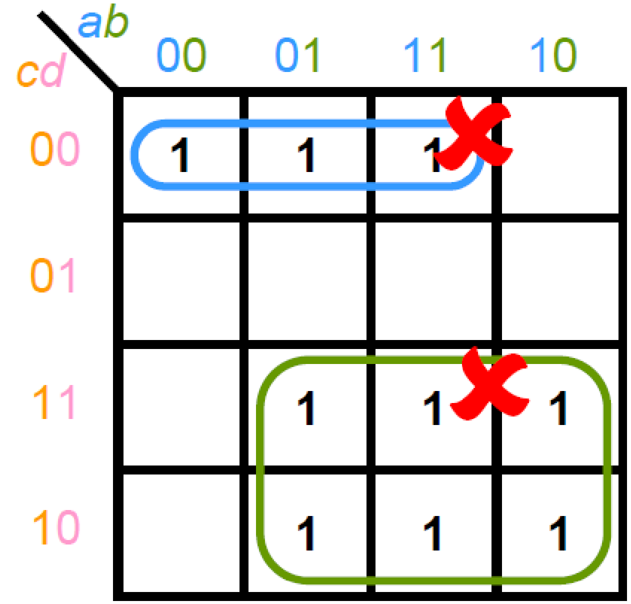


Magic square?

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

2.3 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 $f(a, b) = \Sigma m(0, 1, 3)$

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

(i)

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

3 groups

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

(ii)

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

2 groups

$f = a'b' + b$

(iii)

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

2 groups

$f = a' + ab$

(iv)

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

2 groups (the groups can be overlapped)

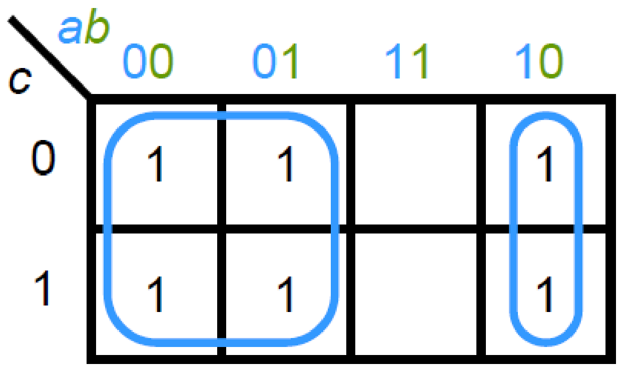
$f = a' + b$

Example: Three-variable K-map

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

Which solution is better?

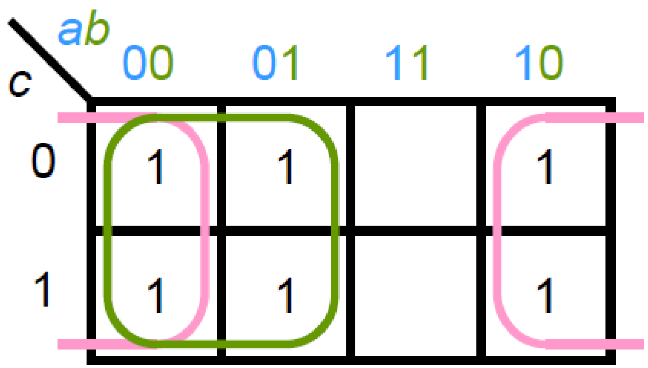
(i)



2 groups

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

(ii)



2 groups

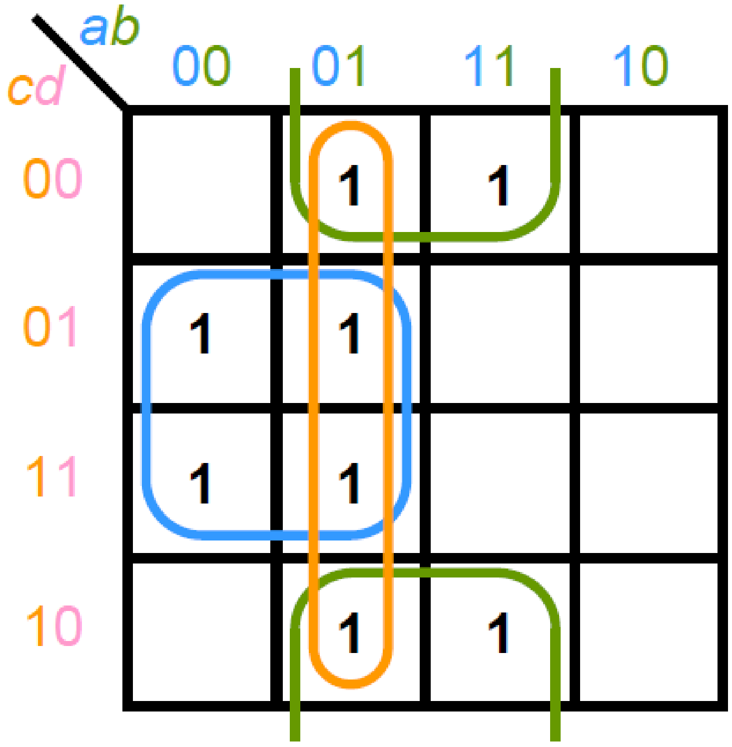
$f(a, b, c) = a' + b'$

Example: Four-variable K-map

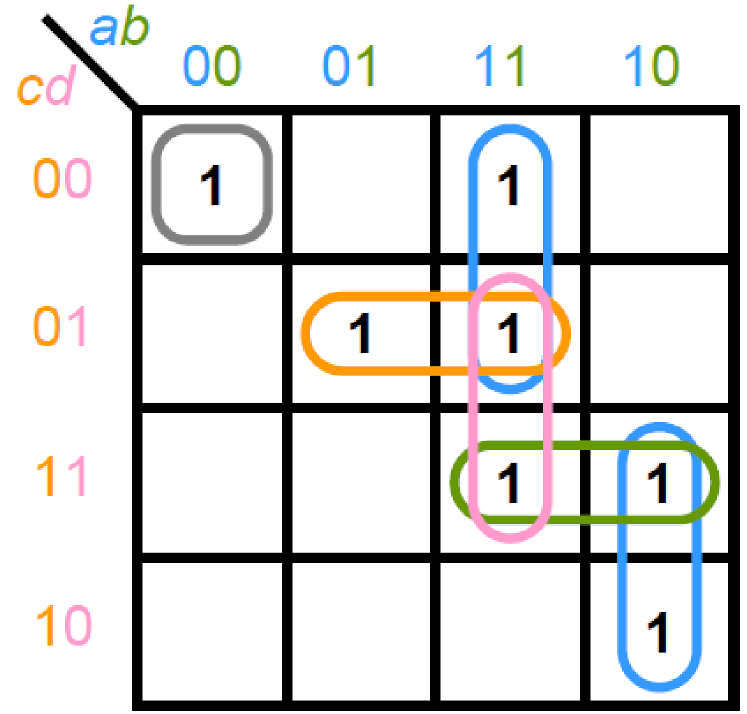
Simplify $f(a, b, c, d) = \Sigma m(0, 1, 2, 3, 4, 5, 7, 8, 10, 11, 15)$

$cd \backslash ab$	00	01	11	10
00	1	1		1
01	1	1		
11	1	1	1	1
10	1			1

Grouping of K-map



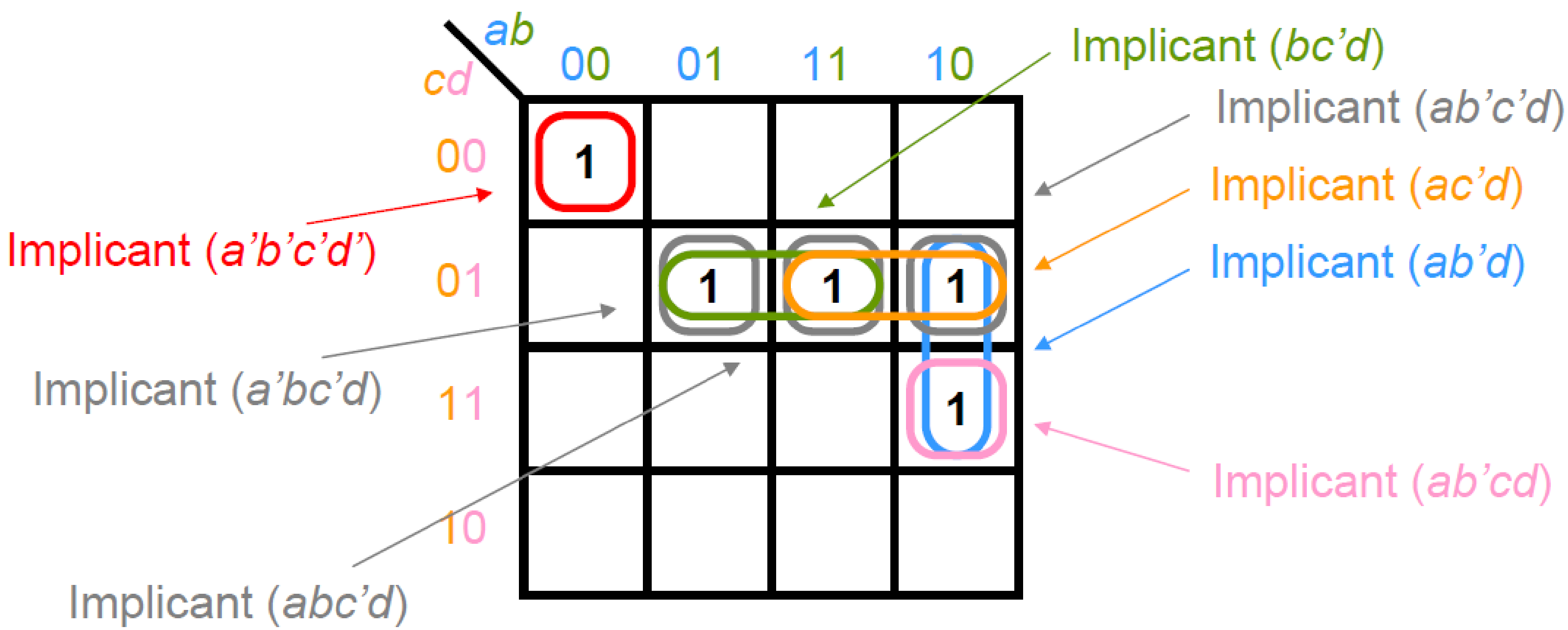
Three groups overlapped!



Too many overlaps!

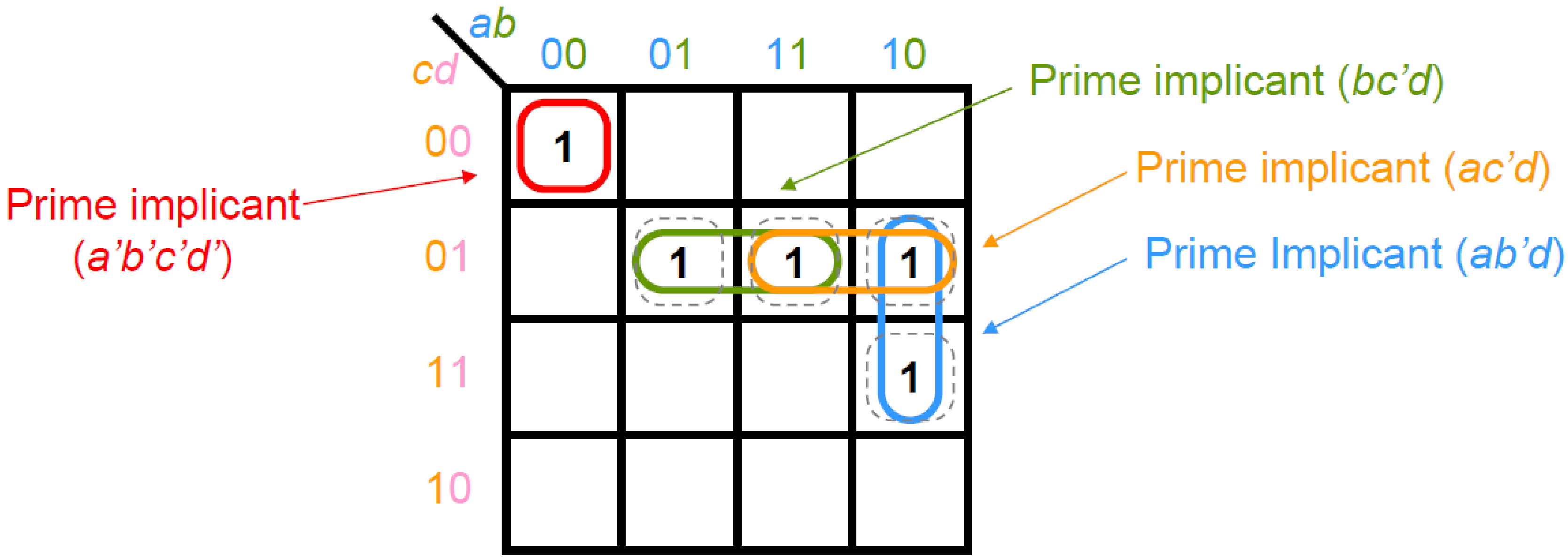
Terminology: Implicant

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



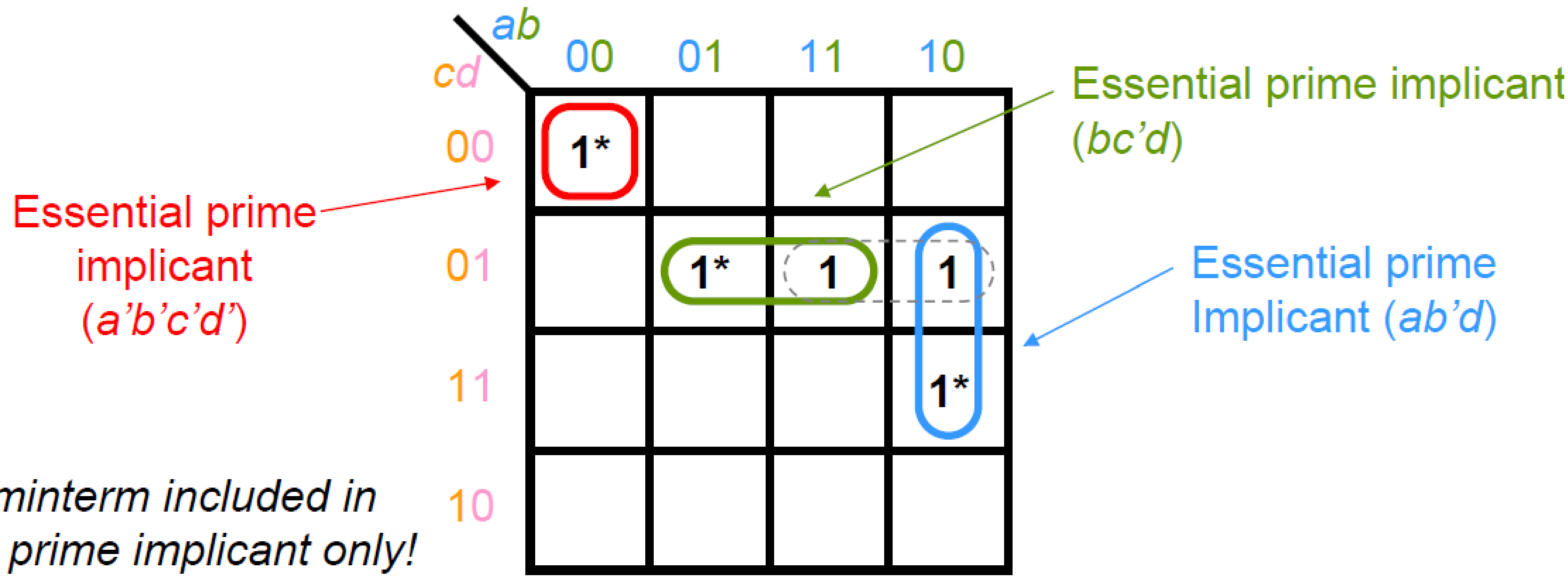
Prime Implicant (PI)

- 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**

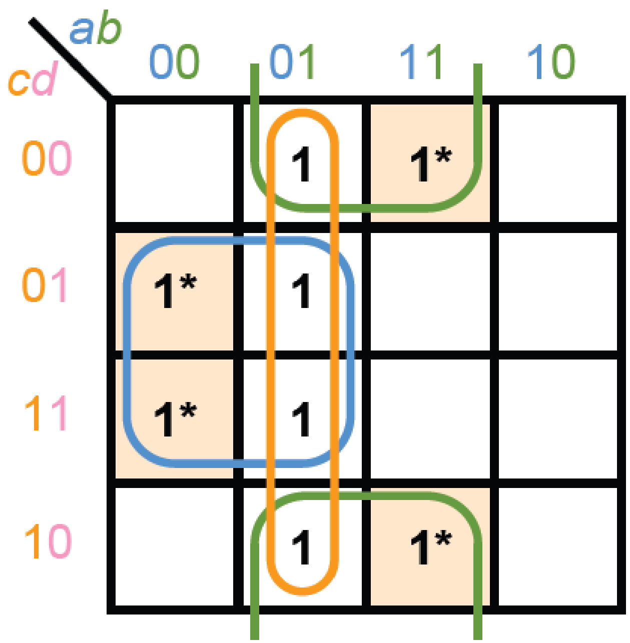


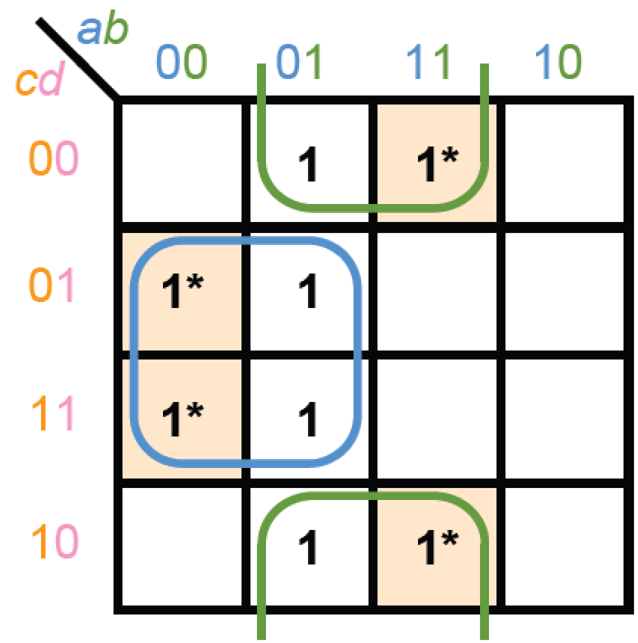
1: minterm included in one prime implicant only!*

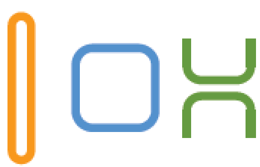
Systematic Procedure

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

Back to the Previous Example





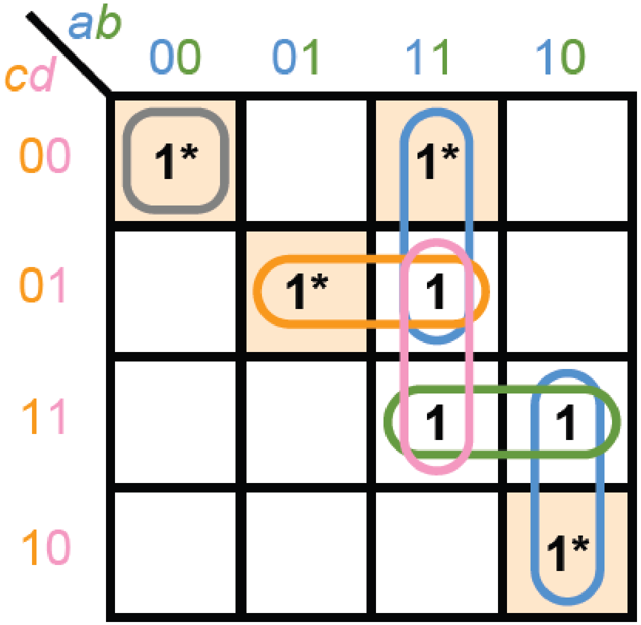
PIs: 

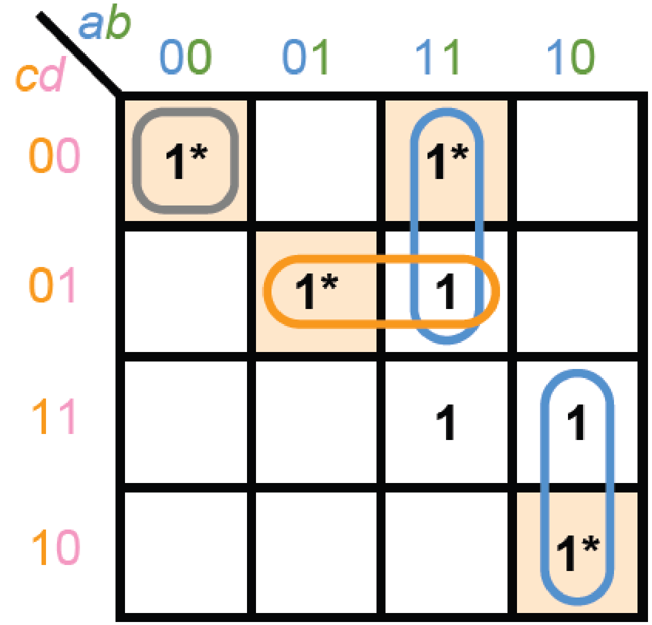
EPI: 

Select the essential prime implicants and no remaining minterms left!

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

Back to the Previous Example





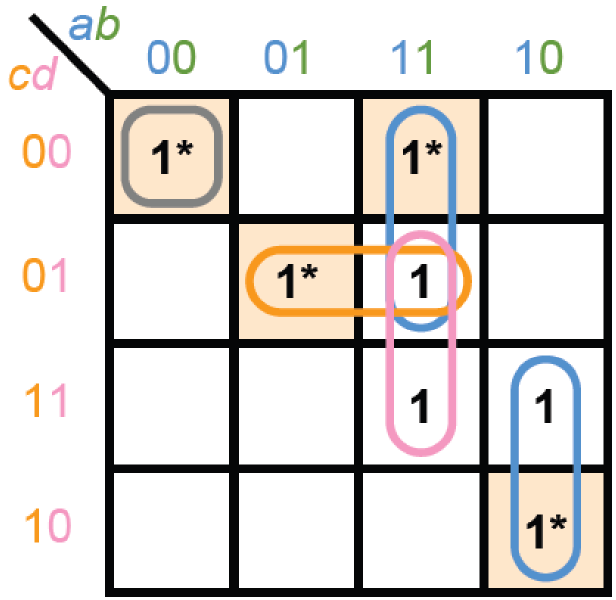
PIs: 

EPIs: 

Select the essential prime implicants first

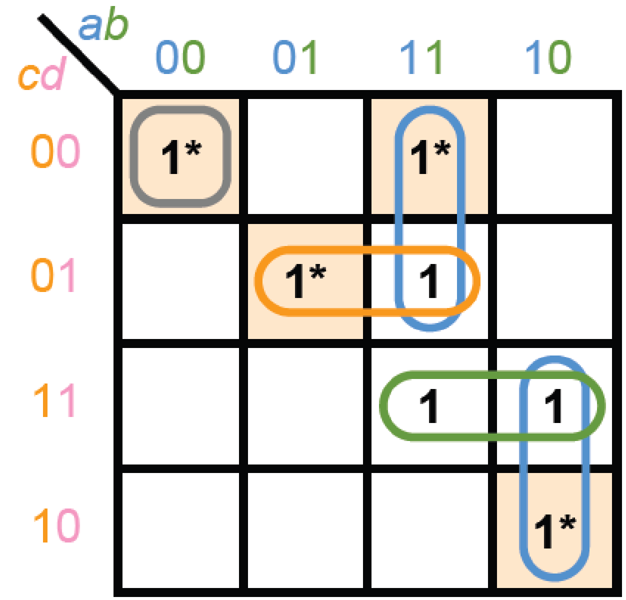
Still have a remaining minterm

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

Minimization using Karnaugh Map

- Previous examples only show the simplified Boolean functions in SOP form
- How to obtain functions in POS form
  - [Step 1] Group the 0s to obtain the complement of the f in SOP form
  - [Step2] Apply DeMorgan's Theorem to find f' in POS form

Example: Find POS

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

cd \ ab	00	01	11	10
00	1	0	0	1
01	1	1	0	1
11	0	0	0	0
10	1	0	0	1

Fill the 1s and 0s into the map

cd \ ab	00	01	11	10
00	1	0	0	1
01	1	1	0	1
11	0	0	0	0
10	1	0	0	1

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

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

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

2.4 Boolean functions with Don’t’-care cases

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
- Output values are defined as **don’t-care**
- Don’t-care term can be minterm / maxterms
- Don’t-care term indicates by an ×, *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 AND b = 0)
f is 1 if	(a = 0 AND b = 1), or
	(a = 1 AND b = 0)
f can be 0 or 1 if	(a = 1 AND b = 1)

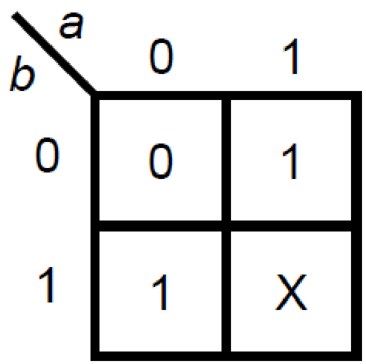
$f(a, b) = \Sigma m(1, 2) + \Sigma d(3)$

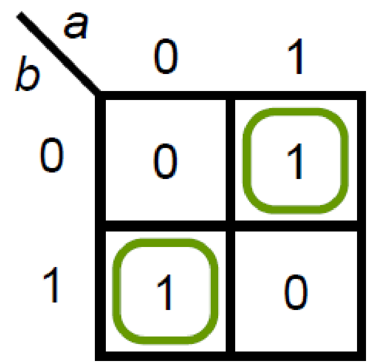
a	b	f_1	f_2
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1

Both f_1 or f_2 of table on the left are acceptable

Don’t-care in K-map

Which solution is better?

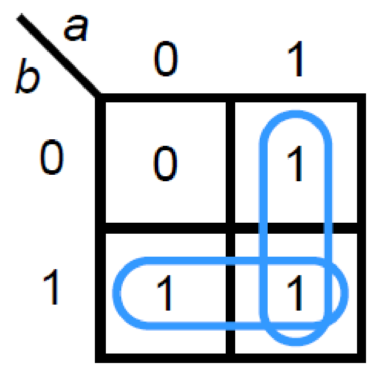




f_1 implementation

2 groups

$f = a'b + ab'$



f_2 implementation

2 groups

$f = a + b$

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

1. Must include all 1s in the map (but × is optional)
2. Select the EPIs first , then remaining Pis
3. Choose the largest PI terms that may contains don't'-care terms ×

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

cd \ ab	00	01	11	10
00	X	0	0	0
01	1	X	0	0
11	1	1	1	1
10	X	0	0	0

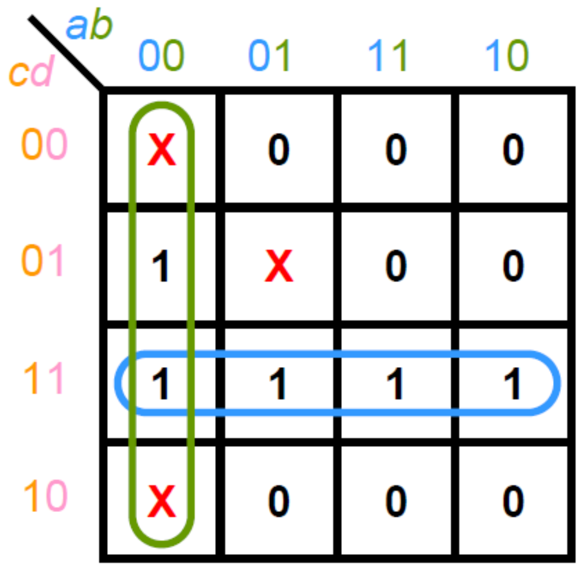
cd \ ab	00	01	11	10
00	X	0	0	0
01	1	X	0	0
11	1	1	1	1
10	X	0	0	0

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

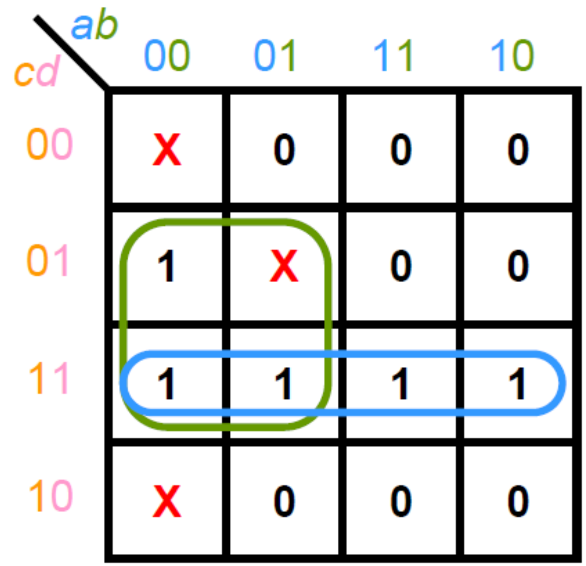
Is it a good solution?

Other solutions

1. Must include all 1s in the map (but × is optional)
2. Select the EPIs first , then remaining PIs
3. Choose the largest PI terms that may contains don't'-care terms ×



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



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

Choose to include those Xs that give largest PIs

2.5 Minimization using Quine-McCluskey (QM) Method

- Developed by W. V. Quine and E. J. McCluskey in 1956
- Functionally identical to Karnaugh map
- More efficient in computer algorithms
- Ease to handle large number of variables

For number of variables is less then 4, we use K-map; otherwise, QM method will be more efficient.

Procedure of QM-method

- Partitioning
- Combining
- Identifying Prime Implicants (PI)
- Generating PI chart
- Reducing chart
- Reporting result

QM-method

- Partitioning

Simplify $f(a, b, c, d) = \Sigma m(1, 4, 5, 6, 8, 9, 10, 12, 14)$

List all minterms of the function first

Minterms	*abcd*
m_1	0001
m_4	0100
m_5	0101
m_6	0110
m_8	1000
m_9	1001
m_{10}	1010
m_{12}	1100
m_{14}	1110

Partition them into groups

Minterms	*abcd*	
m_1	0001	
m_4	0100	1 1s
m_8	1000	
m_5	0101	
m_6	0110	
m_9	1001	2 1s
m_{10}	1010	
m_{12}	1100	
m_{14}	1110	3 1s

QM-method

- Combing

Compare the partitioned terms that follows Gary code property

Combine adjacent group implicants into (n-1)-variable implicants

Minterms	***abcd***
m_1	0001 ✔
m_4	0100 ✔
m_8	1000 ✔
m_5	0101 ✔
m_6	0110 ✔
m_9	1001 ✔
m_{10}	1010 ✔
m_{12}	1100 ✔
m_{14}	1110 ✔

implicants

Mark the changed bit as "-" and give a ✔ to the combined implicants

Minterms	***abcd***
m_1, m_5	0-01
m_1, m_9	-001
m_4, m_5	010-
m_4, m_6	01-0
m_4, m_{12}	-100
m_8, m_9	100-
m_8, m_{10}	10-0
m_8, m_{12}	1-00
m_6, m_{14}	-110
m_{10}, m_{14}	1-10
m_{12}, m_{14}	11-0

implicants

0001
0101

QM-method

- Combing

Further compare the new partitioned terms that follows Gary

Further combine adjacent group implicants into (n-2)-variable implicants

Minterms	*abcd*
m_1, m_5	0-01
m_1, m_9	-001
m_4, m_5	010-
m_4, m_6	01-0 ✔
m_4, m_{12}	-100 ✔
m_8, m_9	100-
m_8, m_{10}	10-0 ✔
m_8, m_{12}	1-00 ✔
m_6, m_{14}	-110 ✔
m_{10}, m_{14}	1-10 ✔
m_{12}, m_{14}	11-0 ✔

Again, mark the changed bit as "-" and give a ✔ to the combined implicants

Minterms	*abcd*
m_4, m_6, m_{12}, m_{14}	-1-0
$m_8, m_{10}, m_{12}, m_{14}$	1--0

No more combination

m_4, m_6: 01-0
m_{12}, m_{14}: 11-0

m_4, m_{12}: -100
m_6, m_{14}: -110

QM-method

- Identifying Prime Implicants (PI)

Minterms	*abcd*
m_1	0001 ✓
m_4	0100 ✓
m_8	1000 ✓
m_5	0101 ✓
m_6	0110 ✓
m_9	1001 ✓
m_{10}	1010 ✓
m_{12}	1100 ✓
m_{14}	1110 ✓

Minterms	*abcd*
m_1, m_5	0-01 PI_3
m_1, m_9	-001 PI_4
m_4, m_5	010- PI_5
m_4, m_6	01-0 ✓
m_4, m_{12}	-100 ✓
m_8, m_9	100- PI_6
m_8, m_{10}	10-0 ✓
m_8, m_{12}	1-00 ✓
m_6, m_{14}	-110 ✓
m_{10}, m_{14}	1-10 ✓
m_{12}, m_{14}	11-0 ✓

Minterms	*abcd*
m_4, m_6, m_{12}, m_{14}	-1-0 PI_1
$m_8, m_{10}, m_{12}, m_{14}$	1--0 PI_2

The remaining unmarked implicants are prime implicants. Label them as PI_i (label the rightmost column first)

QM-method

- Generating PI chart

Mark an X in the corresponding columns if this PI covers that minterms

e.g. PI_1 involves m_4, m_6, m_{12}, m_{14}

Essential prime implicants

PI	Numeric	1	4	5	6	8	9	10	12	14
PI_1	-1-0		x		x				x	x
PI_2	1--0					x		x	x	x
PI_3	0-01	x		x						
PI_4	-001	x					x			
PI_5	010-		x	x						
PI_6	100-					x	x			

These two columns have one X only!
Minterms 6 & 10 are covered by only one prime implicant
i.e. PI_1 and PI_2 are essential prime implicants

QM-method

- Reducing chart

Reduce the chart by removing those rows of essential prime implicants and columns that covered by them

PI	Numeric	1	4	5	6	8	9	10	12	14
PI_1	-1-0		x		⊗				x	x
PI_2	1--0					x		⊗	x	x
PI_3	0-01	x		x						
PI_4	-001	x					x			
PI_5	010-		x	x						
PI_6	100-					x	x			

QM-method

- Reducing chart

The reduced PI chart may not have any essential prime implicants

PI	Numeric	1	5	9
PI_3	0-01	x	x	
PI_4	-001	x		x
PI_5	010-		x	
PI_6	100-			x

How to further reduce the chart?

QM-method

- Reducing chart (covering rule)

Now go back to the reduced PI chart

PI	Numeric	1	5	9
PI_3	0-01	x	x	
PI_4	-001	x		x
PI_5	010-		x	
PI_6	100-			x

PI	Numeric	1 ✓	5 ✓	9 ✓
✓ PI_3	0-01	x	x	
✓ PI_4	-001	x		x

PI_5 is covered by PI_3 (i.e. PI_5 can be removed)

PI_6 is covered by PI_4 (i.e. PI_6 can be removed)

By choosing PI_3 & PI_4, all minterms (m_1, m_5, m_9) have been selected

So combing with previous result (all rows with ✓), the minimized function is

$$f(a, b, c, d) = PI_1 + PI_2 + PI_3 + PI_4$$

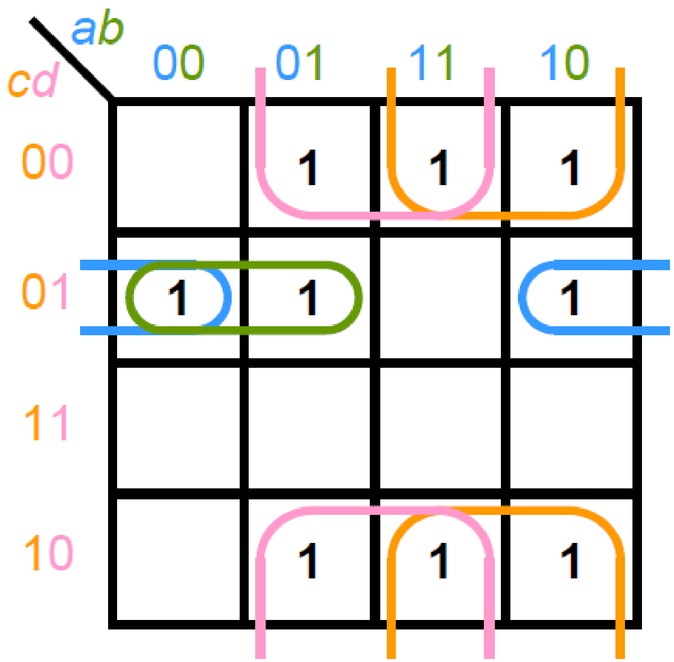
$$= \text{-1-0} + \text{1--0} + \text{0-01} + \text{-001}$$

$$= bd' + ad' + a'c'd + b'c'd$$

Result!!!

Verify the Result by K-map

- Simplify $f(a, b, c, d) = \Sigma m(1, 4, 5, 6, 8, 9, 10, 12, 14)$



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

Don’t Care Conditions

- How to minimize incompletely specified functions using Q-M method?
  - The first three steps are the same (list, partition and combine)
  - But do NOT list the don’t care minterms in the PI chart in step 4
- The reason is
  - We set don’t care terms to minterms so as to find PIs
  - But omit them as the don’t care terms are not essential to be covered

Example: Don't Care

Simplify $f(a, b, c, d) = \Sigma m(4, 8, 9, 10, 12, 15) + \Sigma d(2, 6, 13)$

The don't care terms are listed together

Minterms	*abcd*
m_2	0010
m_4	0100
m_6	0110
m_8	1000
m_9	1001
m_{10}	1010
m_{12}	1100
m_{13}	1101
m_{15}	1111

Partition the terms as usual

Minterms	*abcd*
m_2	0010
m_4	0100
m_8	1000
m_6	0110
m_9	1001
m_{10}	1010
m_{12}	1100
m_{13}	1101
m_{15}	1111

Example: Don't Care

Combine them to form prime implicants

Minterms	*abcd*
m_2	0010 ✓
m_4	0100 ✓
m_8	1000 ✓
m_6	0110 ✓
m_9	1001 ✓
m_{10}	1010 ✓
m_{12}	1100 ✓
m_{13}	1101 ✓
m_{15}	1111 ✓

Minterms	*abcd*
m_2, m_6	0-10 PI_2
m_2, m_{10}	-010 PI_3
m_4, m_6	01-0 PI_4
m_4, m_{12}	-100 PI_5
m_8, m_9	100- ✓
m_8, m_{10}	10-0 PI_6
m_8, m_{12}	1-00 ✓
m_9, m_{13}	1-01 ✓
m_{12}, m_{13}	110- ✓
m_{13}, m_{15}	11-1 PI_7

Minterms	*abcd*
m_8, m_9, m_{12}, m_{13}	1-0- PI_1

Example: Don't Care

Note that the don't care terms m_2, m_6, m_{13} are not listed in this chart!

Find the essential prime implicants and reduce the chart as usual

PI	Numeric	4	8	9	10	12	15
PI_1	1-0-		x	x		x	
PI_2	0-10						
PI_3	-010				x		
PI_4	01-0	x					
PI_5	-100	x				x	
PI_6	10-0		x		x		
PI_7	11-1						x

PI	Numeric	4	10
PI_3	-010		x
PI_4	01-0	x	
PI_5	-100	x	
PI_6	10-0		x

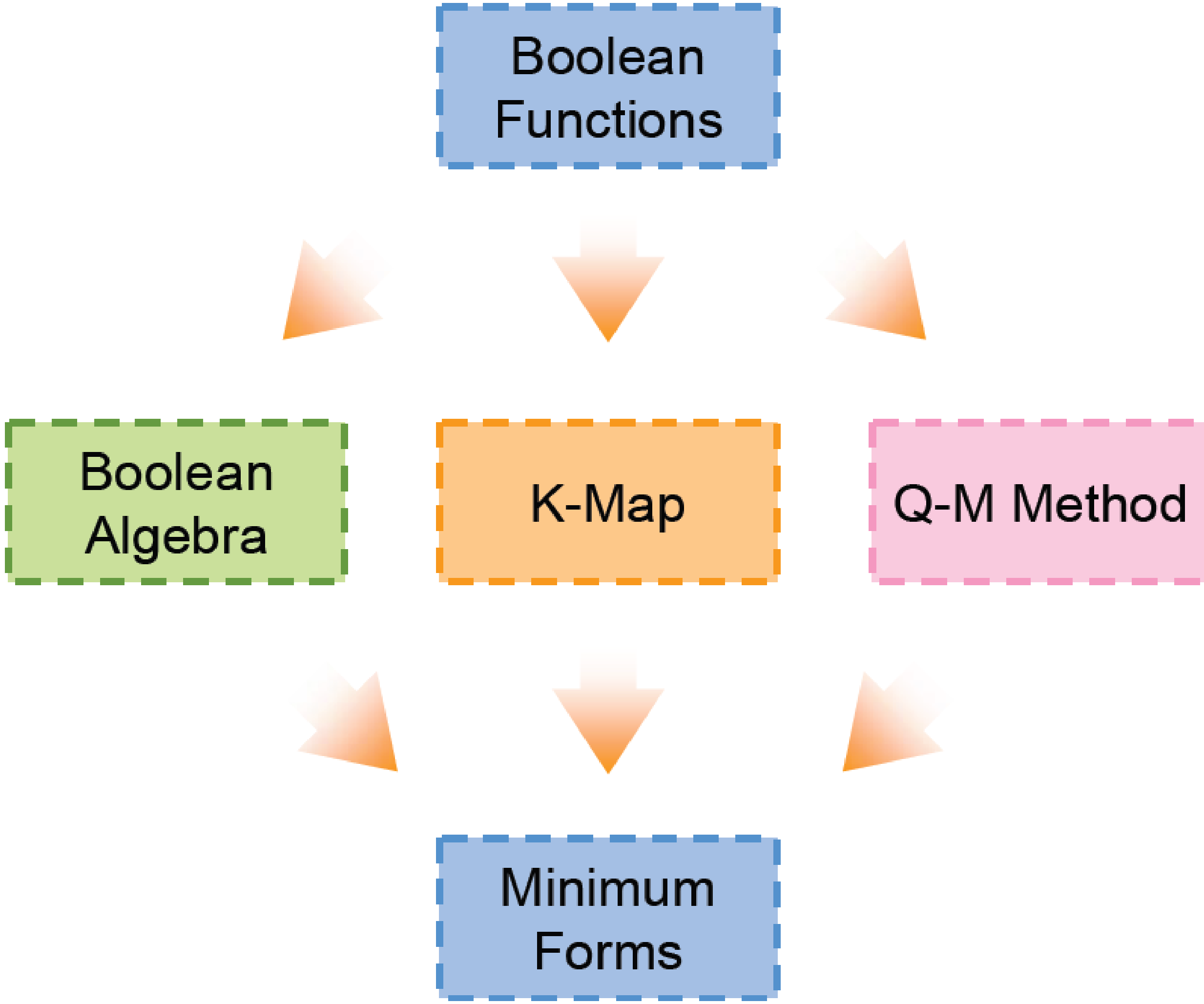
PI	Numeric	4	10
PI_3	-010		x
PI_4	01-0	x	

$$f(a, b, c, d) = PI_1 + PI_3 + PI_4 + PI_7$$

$$= 1\text{-}0\text{-} + \text{-}010 + 01\text{-}0 + 11\text{-}1$$

$$= ac' + b'cd' + a'bd' + abd$$

Summary


Boolean Functions
Boolean Algebra
K-Map
Q-M Method
Minimum Forms