Digital Systems
18B11EC213

Module 1: Boolean Function Minimization Techniques and Combinational Circuits-9

Dr. Saurabh Chaturvedi

Function Simplification

- Why simplify?
  - Simpler expression uses less logic gates.
  - Therefore, less expensive, less power, faster (sometimes).
- Simplification techniques:
  - Algebraic Simplification
    - simplify symbolically using Boolean theorems/postulates.
  - Karnaugh Maps (K-Maps)
    - diagrammatic technique using 'Venn-like diagram'.
    - easy for humans (pattern-matching skills).
    - simplified standard forms.
    - limited to not more than 6 variables.

Algebraic Simplification

- Algebraic simplification aims to minimise
  - (i) number of literals, and
  - (ii) number of terms
- Let's aim at reducing the number of literals.

Absorption

(a) $x + x \cdot y = x$ (b) $x \cdot (x + y) = x$

Absorption (variant)

(a) $x + x' \cdot y = x+y$ (b) $x \cdot (x' + y) = x \cdot y$

Algebraic Simplification

- Example:

$(x+y).(x+y').(x'+z)$	(6 literals)
$= (x.x+x.y'+x.y+y.y').(x'+z)$	(assoc.)
$= (x+x.(y'+y)+0).(x'+z)$	(idemp.,assoc., compl.)
$= (x+x.(1)+0).(x'+z)$	(complement)
$= (x+x+0).(x'+z)$	(identity 1)
$= (x).(x'+z)$	(idemp, identity 0)
$= (x.x'+x.z)$	(assoc.)
$= (0+x.z)$	(complement)
$= x.z$	(identity 0)

Number of literals reduced from 6 to 2.

Algebraic Simplification

- Find minimal SOP and POS expressions of

$f(x,y,z) = x'.y.(z + y'.x) + y'.z$

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

$= x'.y.z + x'.y.y'.x + y'.z$ (distributivity)

$= x'.y.z + 0 + y'.z$ (complement, null element 0)

$= x'.y.z + y'.z$ (identity 0)

$= x'.z + y'.z$ (absorption)

$= (x' + y').z$ (distributivity)

Minimal SOP of f = **$x'.z + y'.z$** (Two 2-input AND gates and one 2-input OR gate)

Minimal POS of f = **$(x' + y').z$** (One 2-input OR gate and one 2-input AND gate)

Algebraic Simplification

- Find minimal SOP expression of

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

$= a.b.c + a.b.d + a'.b.c' + c.d + b.d'$

$= a.b.c + a.b + a'.b.c' + c.d + b.d'$ (absorption)

$= a.b.c + a.b + b.c' + c.d + b.d'$ (absorption)

$= a.b + b.c' + c.d + b.d'$ (absorption)

$= a.b + c.d + b.(c' + d')$ (distributivity)

$= a.b + c.d + b.(c.d)'$ (DeMorgan)

$= a.b + c.d + b$ (absorption)

$= b + c.d$ (absorption)

Number of literals reduced from 13 to 3.

Introduction to K-Maps

- Systematic method to obtain simplified (minimized) sum-of-products (SOP) Boolean expressions.
- Objective: Fewest possible terms/literals.
- Diagrammatic technique based on a special form of Venn diagram.
- Advantage: Easy with visual aid.
- Disadvantage: Limited to 5 or 6 variables.

2-variable K-maps

- **Karnaugh-map** (K-map) is an abstract form of Venn diagram, organised as a matrix of squares, where
  - each square represents a minterm
  - adjacent squares always differ by just one literal (so that the unifying theorem may apply: $a + a' = 1$)

2-variable K-maps

- Alternative layouts of a 2-variable (a, b) K-map

Alternative 1:

		b
	a'b'	a'b
a	ab'	ab

OR

		b
	m0	*m1*
a	*m2*	*m3*

Alternative 2:

		a
	a'b'	ab'
b	a'b	ab

OR

		a
	m0	*m2*
b	*m1*	*m3*

Alternative 3:

	a	
b	ab	a'b
	ab'	a'b'

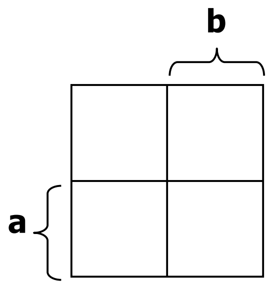
OR

	a	
b	*m3*	*m1*
	m2	*m0*

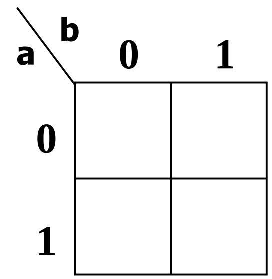
and others…

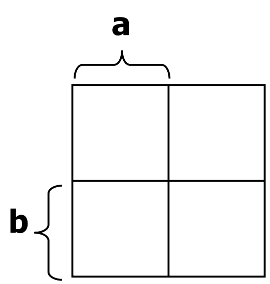
2-variable K-maps

- Equivalent labeling:

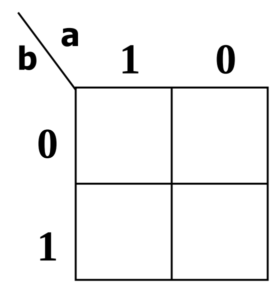


equivalent to:



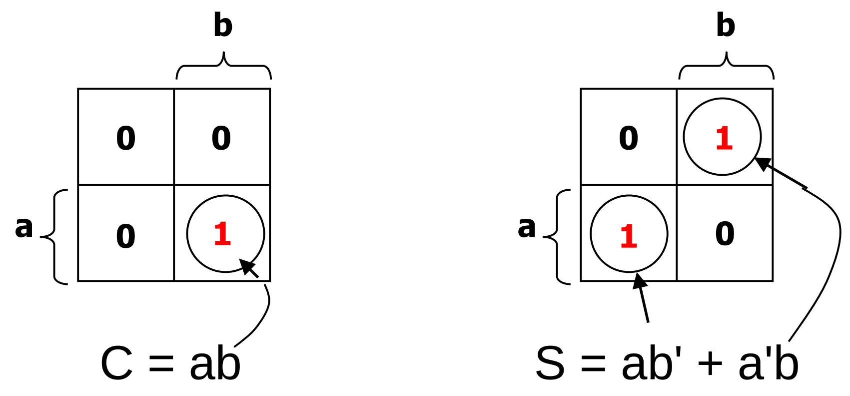


equivalent to:



2-variable K-maps

- The K-map for a function is specified by putting
  - a ‘1’ in the square corresponding to a minterm
  - a ‘0’ otherwise
- For example: Carry and Sum of a half adder:



3-variable K-maps

- There are 8 minterms for 3 variables (a, b, c). Therefore, there are 8 cells in a 3-variable K-map.

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

OR

a \ bc	00	01	11	10
0	*m0*	*m1*	*m3*	*m2*
1	*m4*	*m5*	*m7*	*m6*

Note *Gray code* sequence

Above arrangement ensures that minterms of adjacent cells differ by only *ONE literal*. (Other arrangements which satisfy this criterion may also be used.)

Example

Entries in K-map: The K-map of a 3-variable function F is shown below.

a \ bc	00	01	11	10
0	**1**	**0**	**0**	**1**
1	**0**	**1**	**0**	**0**

4-variable K-maps

- There are 16 cells in a 4-variable (w, x, y, z) K-map.

wx \ yz	00	01	11	10
00	*m0*	*m1*	*m3*	*m2*
01	*m4*	*m5*	*m7*	*m6*
11	*m12*	*m13*	*m15*	*m14*
10	*m8*	*m9*	*m11*	*m10*

5-variable K-maps

- Maps of more than 4 variables are more difficult to use because the geometry (hyper-cube configurations) for combining adjacent squares becomes more involved.
- For 5 variables, e.g., vwxyz, need $2^5 = 32$ squares.

5-variable K-maps

- Organised as two 4-variable K-maps:

v '

wx \ yz	00	01	11	10
00	***m0***	***m1***	***m3***	***m2***
01	***m4***	***m5***	***m7***	***m6***
11	***m12***	***m13***	***m15***	***m14***
10	***m8***	***m9***	***m11***	***m10***

v

wx \ yz	00	01	11	10
00	***m16***	***m17***	***m19***	***m18***
01	***m20***	***m21***	***m23***	***m22***
11	***m28***	***m29***	***m31***	***m30***
10	***m24***	***m25***	***m27***	***m26***

Corresponding squares of each map are adjacent.
Can visualise this as being *one 4-variable map* on TOP *of the other 4-variable map*.

Larger K-maps

- 6-variable K-map is pushing the limit of human "pattern-recognition" capability.
- K-maps larger than 6 variables are practically unheard of!
- Normally, a 6-variable K-map is organised as four 4-variable K-maps, which are mirrored along two axes.

Larger K-maps

b

cd \ ef	a'.b' 00	01	11	10	a'.b 10	11	01	00	ef / cd
00	*m0*	*m1*	*m3*	*m2*	*m18*	*m19*	*m17*	*m16*	00
01	*m4*	*m5*	*m7*	*m6*	*m22*	*m23*	*m21*	*m20*	01
11	*m12*	*m13*	*m15*	*m14*	*m30*	*m31*	*m29*	*m28*	11
10	*m8*	*m9*	*m11*	*m10*	*m26*	*m27*	*m25*	*m24*	10
a 10	*m40*	*m41*	*m43*	*m42*	*m58*	*m59*	*m57*	*m56*	10
11	*m44*	*m45*	*m47*	*m46*	*m62*	*m63*	*m61*	*m60*	11
01	*m36*	*m37*	*m39*	*m38*	*m54*	*m55*	*m53*	*m52*	01
00	*m32*	*m33*	*m35*	*m34*	*m50*	*m51*	*m49*	*m48*	00
cd / ef	00	01	11	10	10	11	01	00	cd / ef
	a.b'				a.b				

Try stretch your recognition capability by finding simplest sum-of-products expression for Σ m(6,8,14,18,23,25,27,29,41,45,57,61).

Simplification Using K-maps

- Based on the Unifying Theorem:

$$A + A' = 1$$

- In a K-map, each cell containing a '1' corresponds to a minterm of a given function F.
- Each group of adjacent cells containing '1' (group must have size in powers of twos: 1, 2, 4, 8, …) then corresponds to a simpler product term of F.
  - Grouping 2 adjacent squares eliminates 1 variable, grouping 4 squares eliminates 2 variables, grouping 8 squares eliminates 3 variables, and so on. In general, grouping 2^n squares eliminates *n* variables.

Simplification Using K-maps

- Group as many squares as possible.
  - The larger the group is, the fewer the number of literals in the resulting product term.
- Select as few groups as possible to cover all the squares (minterms) of the function.
  - The fewer the groups, the fewer the number of product terms in the minimized function.

Simplification Using K-maps

- Example:

$$F(w,x,y,z) = w'.x.y'.z' + w'.x.y'.z + w.x'.y.z' + w.x'.y.z + w.x.y.z' + w.x.y.z$$

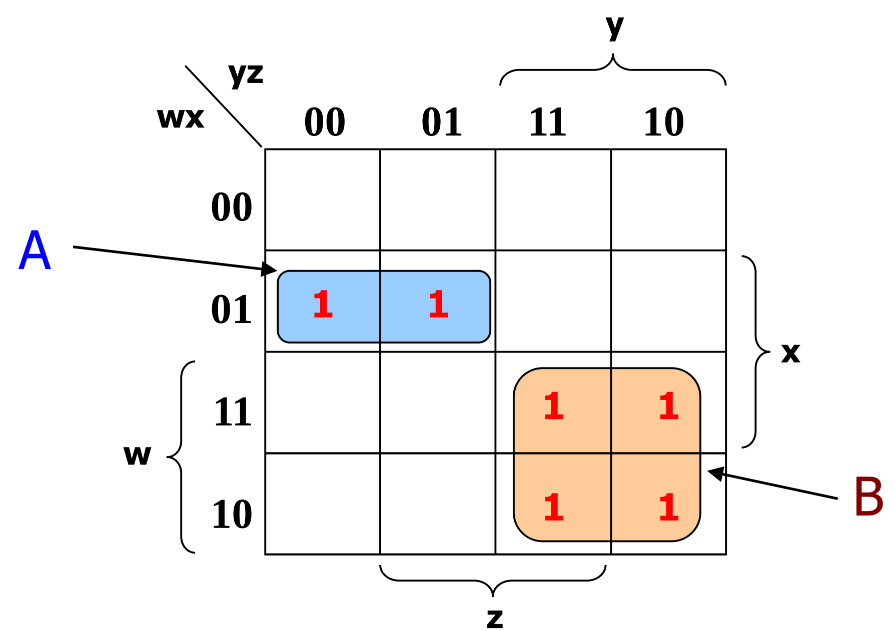
$$= \Sigma\, m(4, 5, 10, 11, 14, 15)$$

wx \ yz	00	01	11	10
00				
01	1	1		
11			1	1
10			1	1

(cells with ‘0’ are not shown for clarity)

Simplification Using K-maps

- Each group of adjacent minterms (group size in powers of twos) corresponds to a possible product term of the given function.

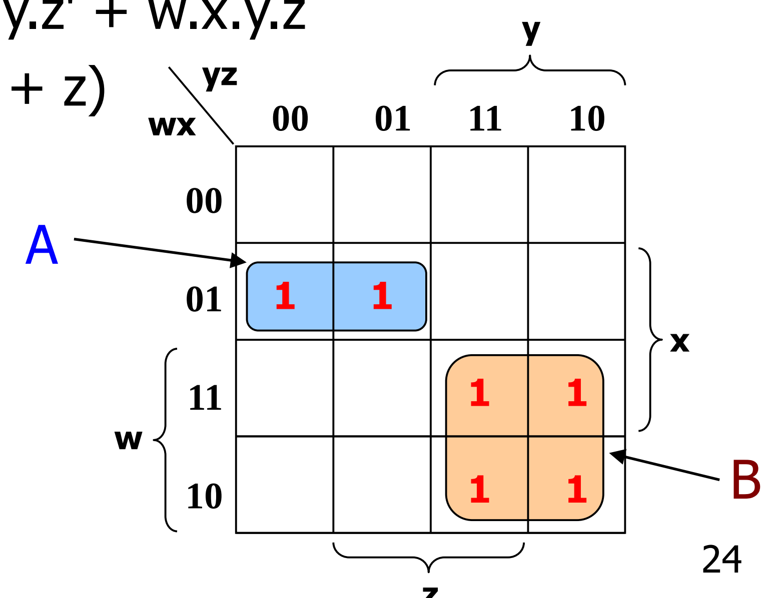


Simplification Using K-maps

- There are 2 groups of minterms: A and B, where:

A = w'.x.y'.z' + w`.x.y'.z
  = w'.x.y'.(z' + z)
  = w'.x.y'

B = w.x'.y.z' + w.x'.y.z + w.x.y.z' + w.x.y.z
  = w.x'.y.(z' + z) + w.x.y.(z' + z)
  = w.x'.y + w.x.y
  = w.(x'+x).y
  = w.y



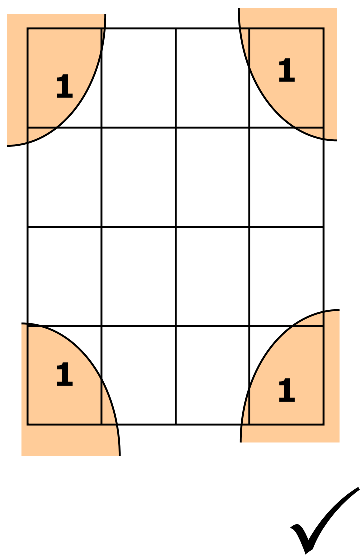
Simplification Using K-maps

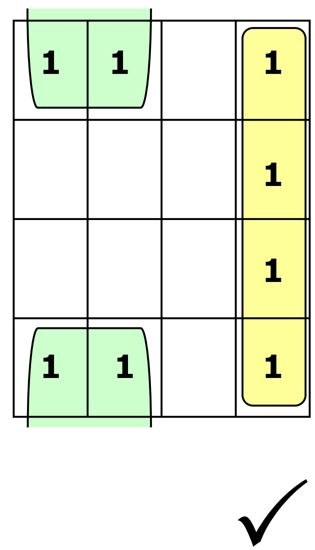
- Each product term of a group, $w'.x.y'$ and $w.y$, represents the sum of minterms in that group.
- Boolean function is therefore the sum of product terms (SOP) which represent all groups of the minterms of the function.

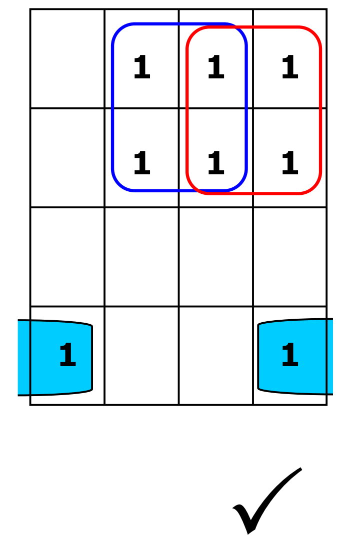
$$F(w,x,y,z) = A + B = w'.x.y' + w.y$$

Simplification Using K-maps

- Other possible valid groupings of a 4-variable K-map include:



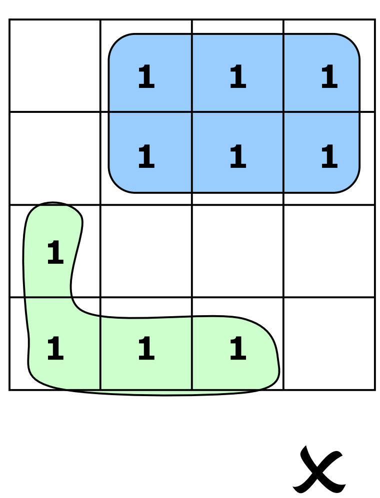


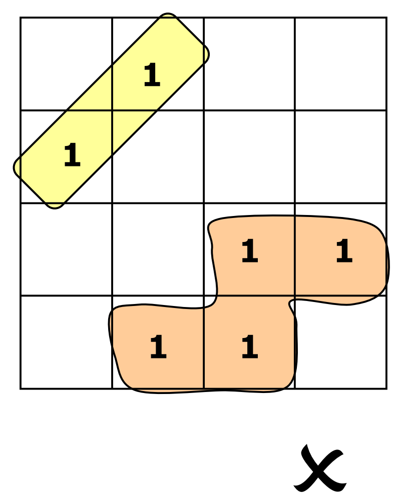


Simplification Using K-maps

- Groups of minterms must be
  - (1) rectangular, and
  - (2) have size in powers of two.

  Otherwise they are invalid groups. Some examples of invalid groups:





Converting to Minterms Form

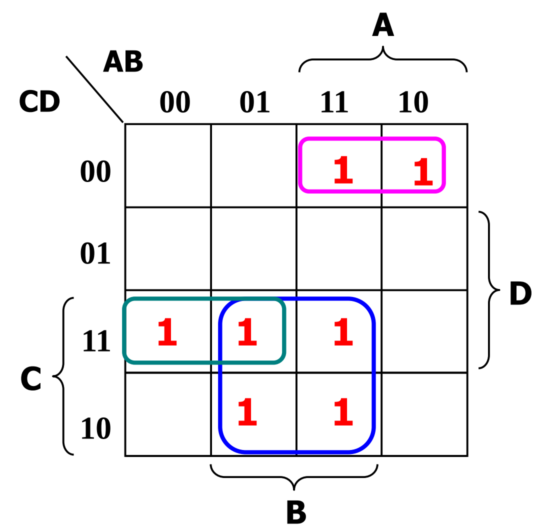
- The K-map of a function is easily drawn when the function is given in **canonical** sum-of-products (SOP) or sum-of-minterms form.
- What if the function is not in sum-of-minterms?
  - Convert it to sum-of-products (SOP) form.
  - Expand the SOP expression into sum-of-minterms expression, or fill in the K-map directly based on the SOP expression.

Converting to Minterms Form

- Example:

$$F(A,B,C,D) = A(C+D)'(B'+D') + C(B+C'+A'D)$$
$$= A(C'D')(B'+D') + BC + CC' + A'CD$$
$$= AB'C'D' + AC'D' + BC + A'CD$$

F = AB'C'D' + AC'D' + BC + A'CD
= AB'C'D' + AC'D'(B+B') + BC + A'CD
= AB'C'D' + ABC'D' + AB'C'D' + BC(A+A') + A'CD
= AB'C'D' + ABC'D' + ABC + A'BC + A'CD
= AB'C'D' + ABC'D' + ABC(D+D') + A'BC(D+D') + A'CD(B+B')
F = AB'C'D' + ABC'D' + ABCD + ABCD' + A'BCD + A'BCD' + A'B'CD
Canonical SOP form of the function F.



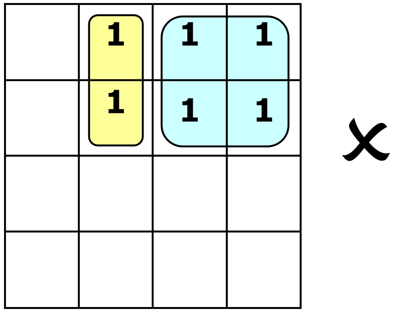
Simplest SOP Expressions

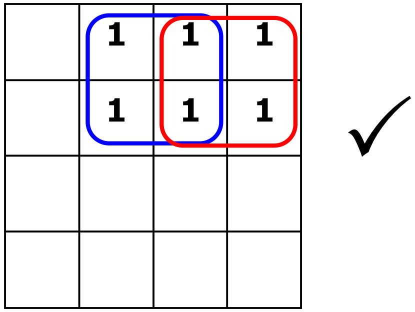
- To find the simplest possible sum of products (SOP) expression from a K-map, we need to obtain:
  - minimum number of literals per product term; and
  - minimum number of product terms
- This is achieved in K-map using
  - bigger groupings of minterms (prime implicants) where possible; and
  - no redundant groupings (look for essential prime implicants)

Implicant: a product term that could be used to cover minterms of the function.

Simplest SOP Expressions

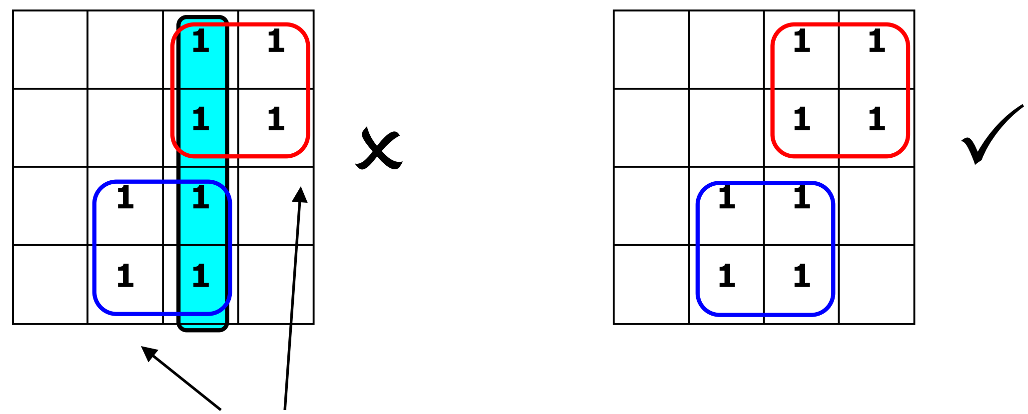
- A prime implicant (PI) is a product term obtained by combining the maximum possible number of minterms from adjacent squares in the map.
- Use bigger groupings (prime implicants) where possible.





Simplest SOP Expressions

No redundant groups:



Essential prime implicants

- An essential prime implicant (EPI) is a prime implicant (PI) that includes at least one minterm that is not covered by any other prime implicant.

Simplest SOP Expressions

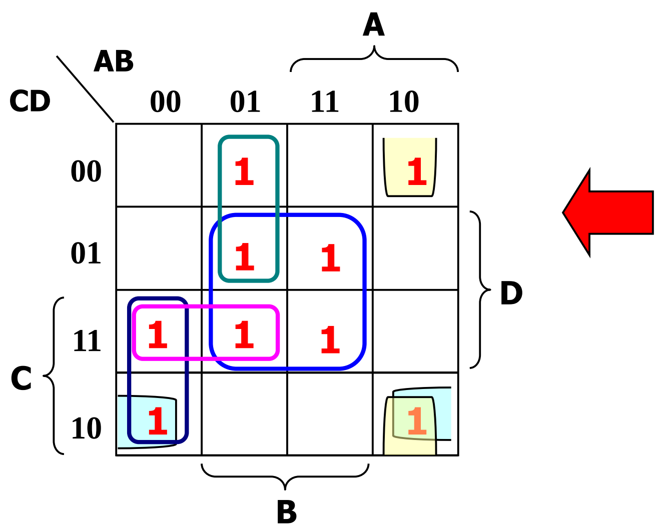
- Steps:

  1. Circle all prime implicants (PIs) on the K-map.
  2. Identify and select all essential prime implicants (EPIs) for the cover.
  3. Select a minimum subset of the remaining prime implicants to complete the cover, that is, to cover those minterms not covered by the essential prime implicants.

Simplest SOP Expressions

- Example:

$$f(A,B,C,D) = \sum m(2,3,4,5,7,8,10,13,15)$$

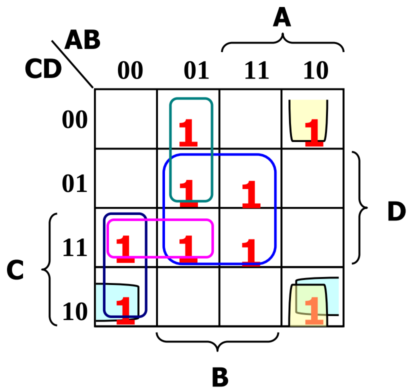


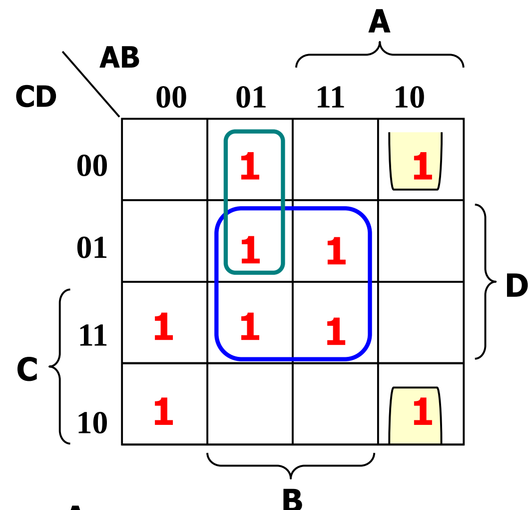
All prime implicants

PI terms: AB'D', A'BC', A'CD, A'B'C, B'CD', BD

Total 6 PI terms

Simplest SOP Expressions

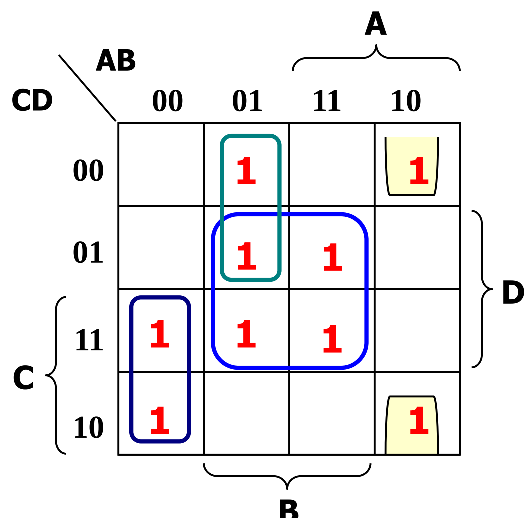




Essential prime implicants

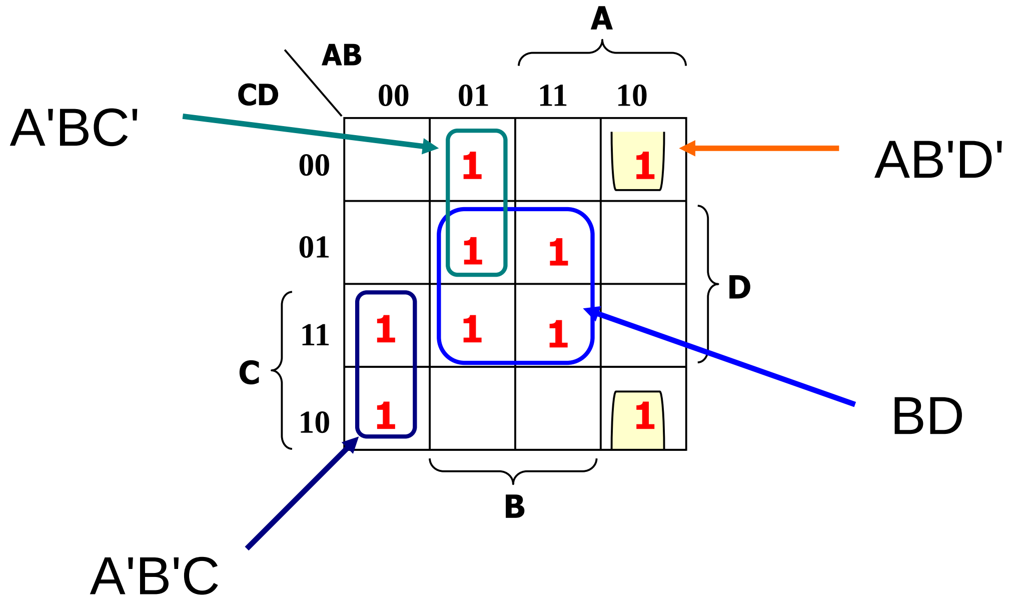
EPI terms: AB'D', A'BC', BD

Total 3 EPI terms



Minimum cover

Simplest SOP Expressions



Minimized expression:

f(A,B,C,D) = B.D + A'.B'.C + A.B'.D' + A'.B.C'

Therefore, 4 terms are required in the minimized expression of the function f.

Quick Review Question

Find the simplified expression for G(A,B,C,D).

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

References

- M. M. Mano, *Digital Logic and Computer Design*, 5th ed., Pearson Prentice Hall, 2013.
- R. P. Jain, *Modern Digital Electronics*, 4th ed., Tata McGraw-Hill Education, 2009.