K-maps

Minimization of Boolean expressions

- *The minimization will result in reduction of the number of gates (resulting from less number of terms) and the number of inputs per gate (resulting from less number of variables per term)*
- *The minimization will reduce cost, efficiency and power consumption.*
- **$y(x+x`)=y.1=y$**
- **$y+xx`=y+0=y$**
- **$(x`y+xy`)=x\oplus y$**
- **$(x`y`+xy)=(x\oplus y)`$**

THE ARC.

Minimum SOP and POS

- The *minimum sum of products (MSOP)* of a function, f, is a SOP representation of f that contains the fewest number of product terms and fewest number of literals of any SOP representation of f.

Minimum SOP and POS

- *f= (xyz +x`yz+ xy`z+ …..)*

Is called sum of products.

The + is sum operator which is an OR gate.

The product such as xy is an AND gate for the two inputs x and y.

THE ARC.

Example

- ***Minimize the following Boolean function using sum of products (SOP):***

- ***f(a,b,c,d) = ∑m(3,7,11,12,13,14,15)***

	abcd	
3	0011	**a`b`cd**
7	0111	**a`bcd**
11	1011	**ab`cd**
12	1100	**abc`d`**
13	1101	**abc`d**
14	1110	**abcd`**
15	1111	**abcd**

THE ARC

Example

$f(a,b,c,d) = \sum m(3,7,11,12,13,14,15)$

$= a`b`cd + a`bcd + ab`cd + abc`d` + abc`d + abcd` + abcd$

$= cd(a`b` + a`b + ab`) + ab(c`d` + c`d + cd` + cd)$

$= cd(a`[b` + b] + ab`) + ab(c`[d` + d] + c[d` + d])$

$= cd(a`[1] + ab`) + ab(c`[1] + c[1])$

$= ab + ab`cd + a`cd$

$= ab + cd(ab` + a`)$

$= ab + cd(a + a`)(a` + b`)$

$= ab + a`cd + b`cd$

$= ab + cd(a` + b`)$

Minimum product of sums (MPOS)

- The *minimum product of sums (MPOS)* of a function, f, is a POS representation of f that contains the fewest number of sum terms and the fewest number of literals of any POS representation of f.
- The zeros are considered exactly the same as ones in the case of sum of product (SOP)

Example

$f(a,b,c,d) = \prod M(0,1,2,4,5,6,8,9,10)$

$=\sum m(3,7,11,12,13,14,15)$

$=[(a+b+c+d)(a+b+c+d`)(a+b`+c`+d`)$

$(a`+b+c`+d`)(a`+b`+c+ d)(a`+b`+c+ d`)\ (a`+b`+c`+d)(a`+b`+c`+d`)]$

Karnaugh Maps (K-maps)

- Karnaugh maps -- A tool for representing Boolean functions of up to six variables.
- K-maps are tables of rows and columns with entries represent 1`s or 0`s of SOP and POS representations.

THE ARC.

Karnaugh Maps (K-maps)

- An *n*-variable K-map has 2^n cells with each cell corresponding to an *n*-variable truth table value.
- K-map cells are labeled with the corresponding truth-table row.
- K-map cells are arranged such that adjacent cells correspond to truth rows that differ in only one bit position (*logical adjacency*).

THE ARC

Karnaugh Maps (K-maps)

- If m_i is a minterm of f, then place a 1 in cell i of the K-map.
- If M_i is a maxterm of f, then place a 0 in cell i.
- If d_i is a don’t care of f, then place a d *or* x in cell i.

Examples

- *Two variable K-map f(A,B)=*$\sum$*m*(0,1,3)=A`B`+A`B+AB

B \ A	0	1
0	1	0
1	1	1

Three variable map

- *f(A,B,C)* = $\sum m(0,3,5)$= A`B`C`+A`BC+AB`C

	A`B` 0 0	A`B 0 1	A B 1 1	A B` 1 0
C` 0	1		A`B`C`	
C 1		1 A`BC		1 AB`C

THE ARC.

Maxterm example

		(A+B)	(A+B\`)	(A\`+B\`)	(A\`+B)
		A\`B\`	A\`B	AB	AB\`
C	C\`		0	0	0
C\`	C	0		0	

$f(A,B,C) = \prod M(1,2,4,6,7)$

=(A+B+C\`)(A+B\`+C)(A\`+B+C) )(A\`+B\`+C) (A\`+B\`+C\`)

Note that the complements are (0,3,5) which are the minterms of the previous example

Four variable example

(a) Minterm form. (b) Maxterm form.

$$f(a,b,Q,G) = \sum m(0,3,5,7,10,11,12,13,14,15) = \prod M(1,2,4,6,8,9)$$

QG \ ab	00	01	11	10
00	0: 1	4:	12: 1	8:
01	1:	5: 1	13: 1	9:
11	3: 1	7: 1	15: 1	11: 1
10	2:	6:	14: 1	10: 1

(a)

QG \ ab	00	01	11	10
00	0:	4: 0	12:	8: 0
01	1: 0	5:	13:	9: 0
11	3:	7:	15:	11:
10	2: 0	6: 0	14:	10:

(b)

Simplification of Boolean Functions Using K-maps

- K-map cells that are physically adjacent are also logically adjacent. Also, cells on an edge of a K-map are logically adjacent to cells on the opposite edge of the map.

- If two logically adjacent cells both contain logical 1s, the two cells can be combined to eliminate the variable that has value 1 in one cell's label and value 0 in the other.

Simplification of Boolean Functions Using K-maps

- This is equivalent to the algebraic operation, $aP + a'P = P$ where P is a product term not containing a or a'.

- A group of cells can be combined only if all cells in the group have the same value for some set of variables.

THE ARC.

Simplification Guidelines for K-maps

- Always combine as many cells in a group as possible. This will result in the fewest number of literals in the term that represents the group.
- Make as few groupings as possible to cover all minterms. This will result in the fewest product terms.
- Always begin with the largest group, which means if you can find eight members group is better than two four groups and one four group is better than pair of two-group.

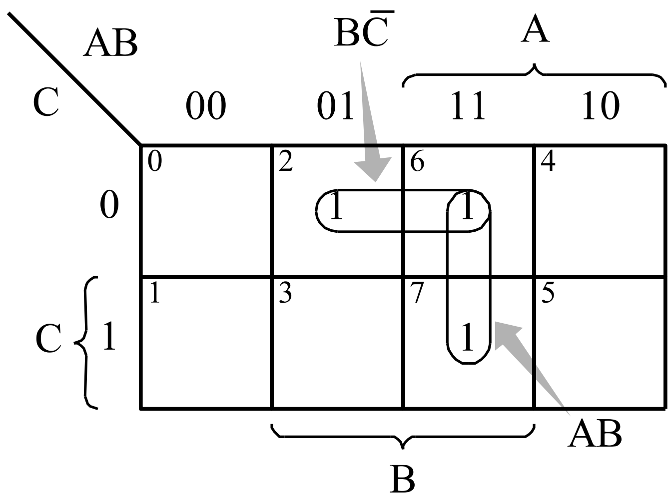
THE ARC

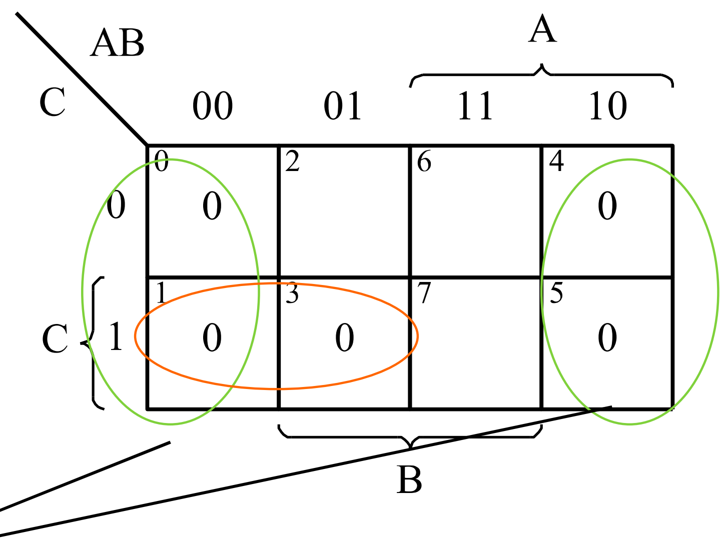
Example

Simplify f= A`BC`+ A B C`+ A B C **using;**

(a) Sum of minterms. (b) Maxterms.

- Each cell of an *n*-variable K-map has *n* logically adjacent cells.





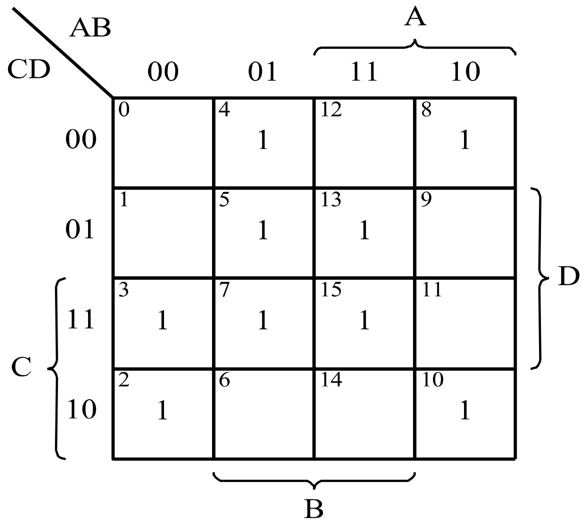
$F` = B` + A`C$ $\quad$ $F = B(A+C`)$

a- $f(A,B,C) = AB + BC'$

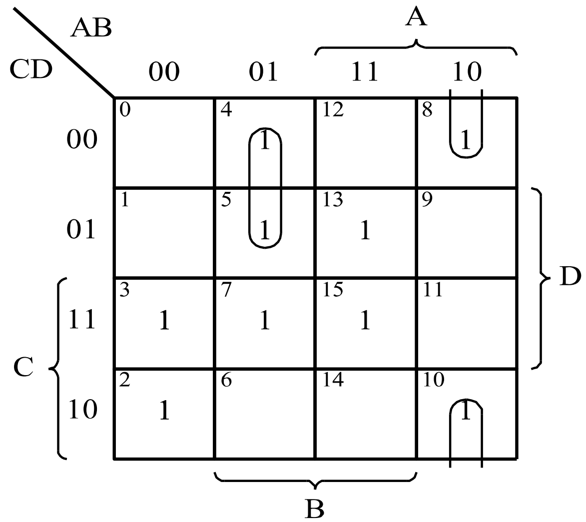
b- $f(A,B,C) = B(A + C')$

Example Simplify

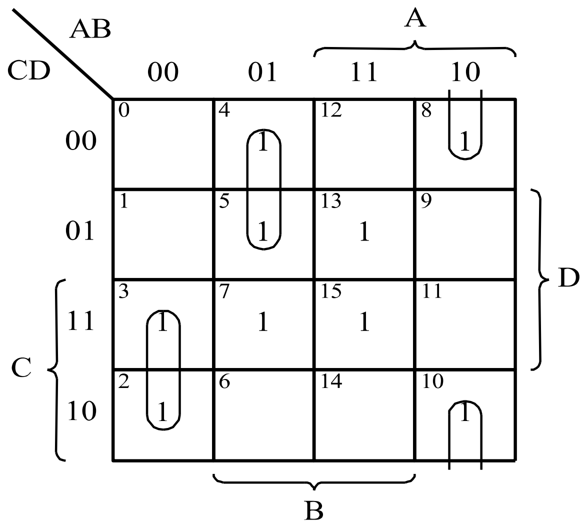
$$f(A,B,C,D) = \sum m(2,3,4,5,7,8,10,13,15)$$



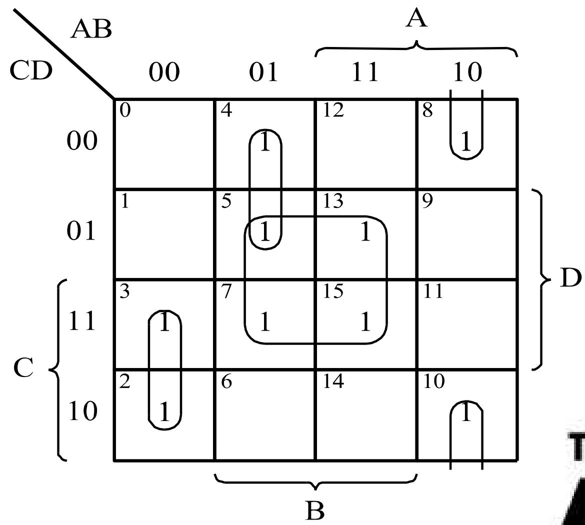
(a)



(b)



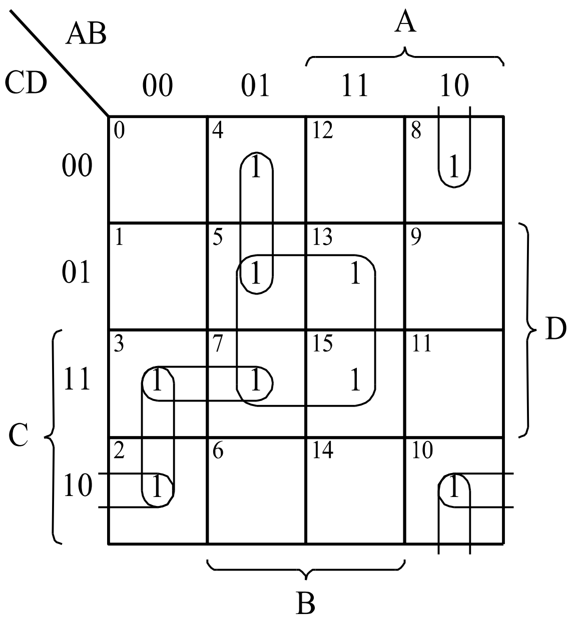
(c)



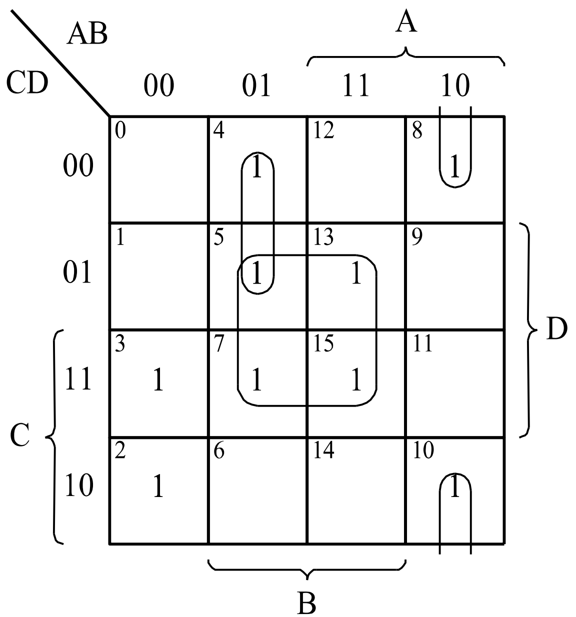
(d)

Example Multiple selections

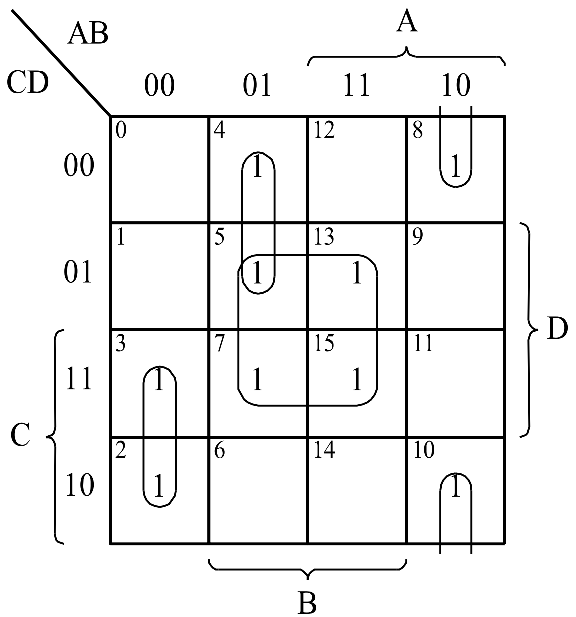
$$f(A,B,C,D) = \sum m(2,3,4,5,7,8,10,13,15)$$



(a)



(b)

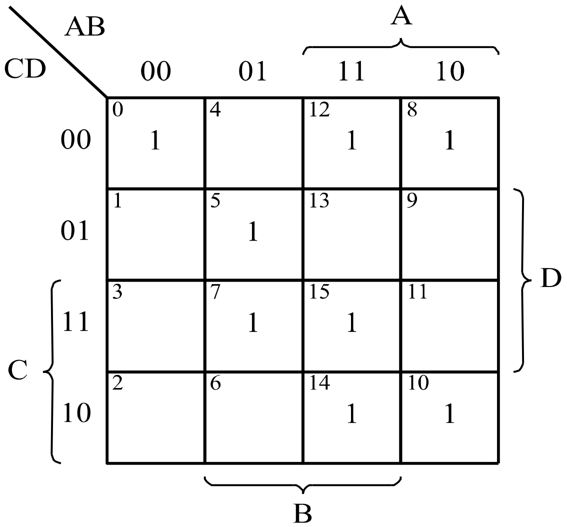


(c)

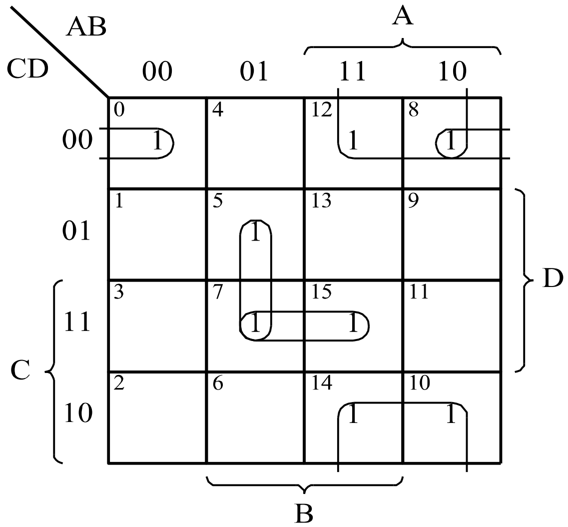
c produces less terms than a

Example Redundant selections

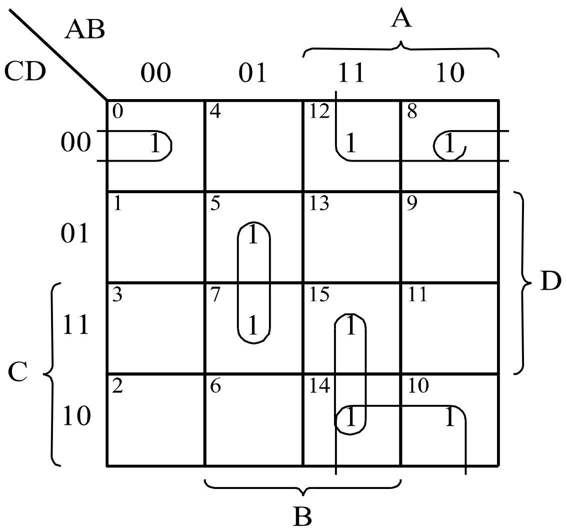
$f(A,B,C,D) = \sum m(0,5,7,8,10,12,14,15)$



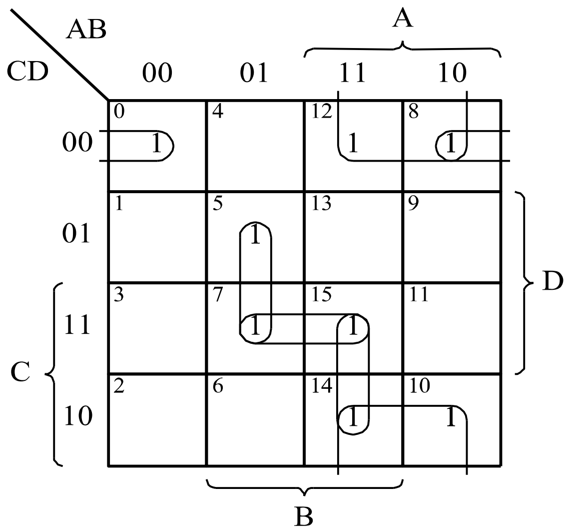
(a)



(b)

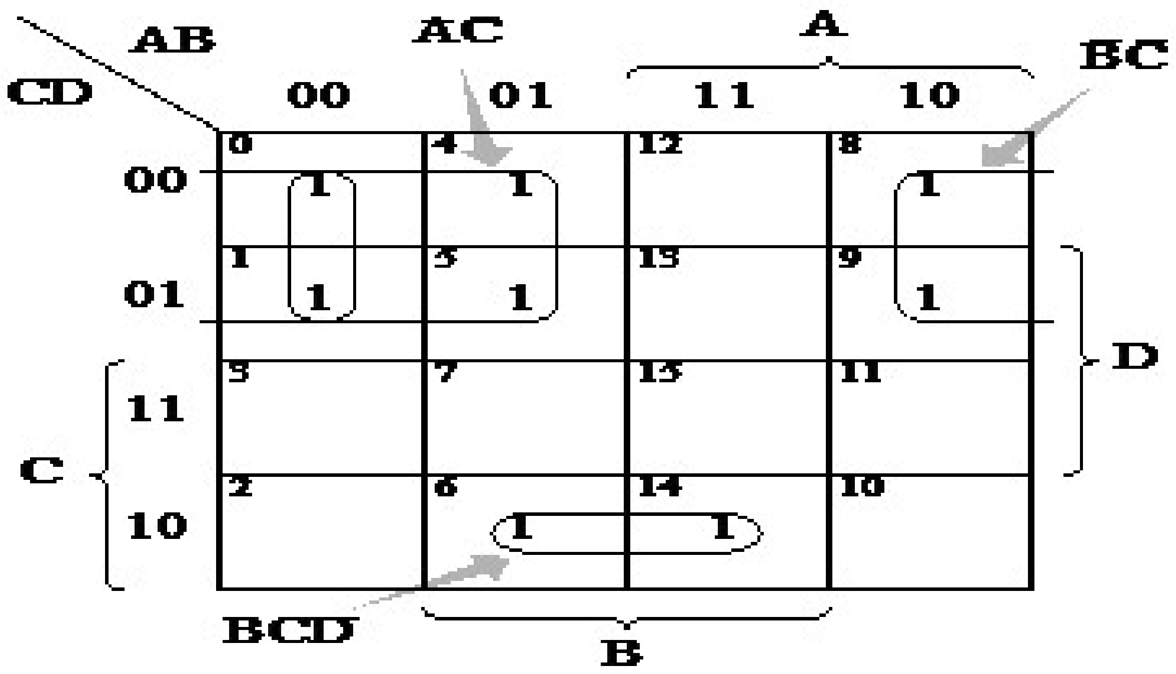


(c)



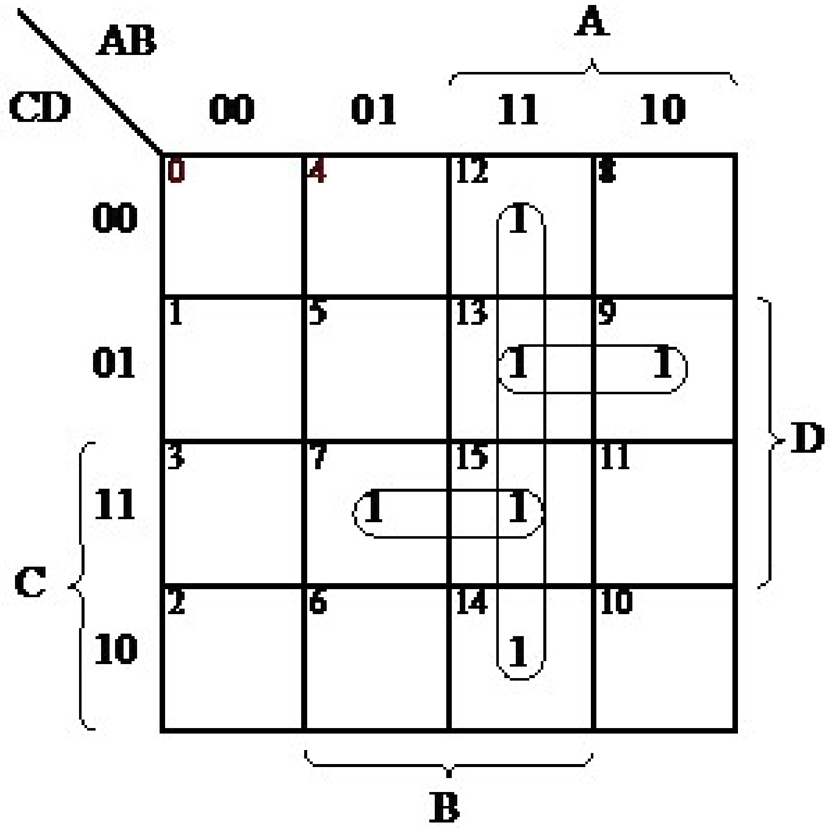
(d)

Example



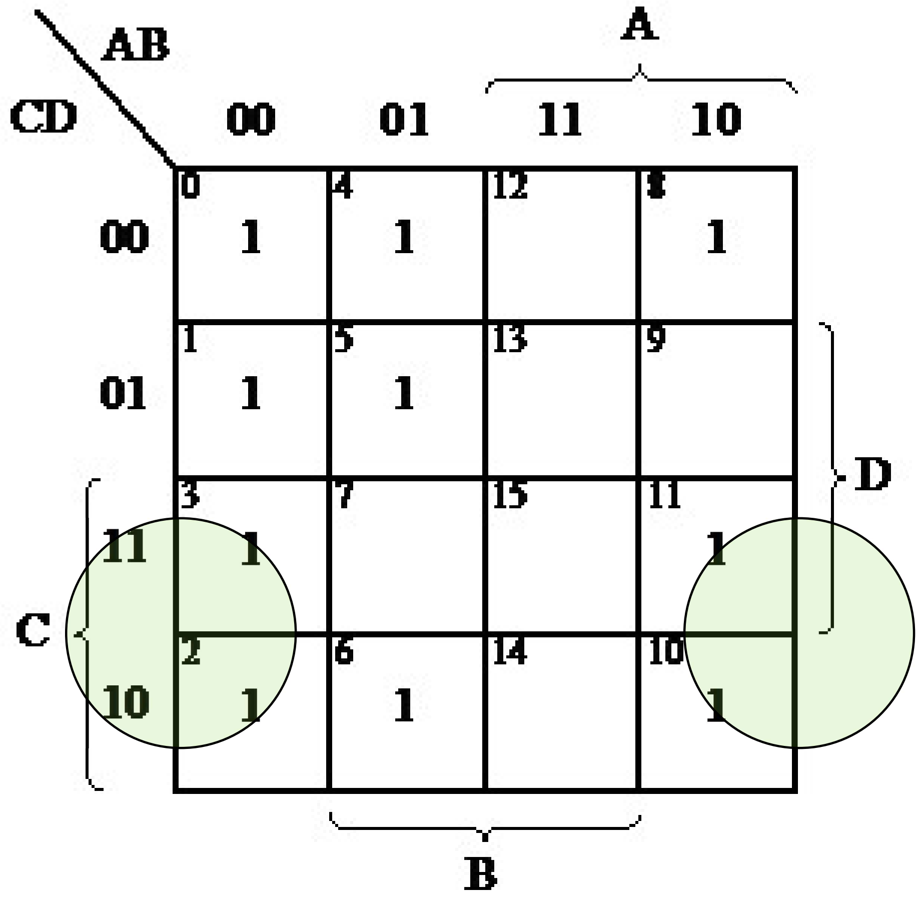
AB
CD
AC
A
BC
00
01
11
10
0
4
12
8
1
5
13
9
3
7
15
11
2
6
14
10
1
1
1
1
1
1
1
1
D
C
BCD
B

Example



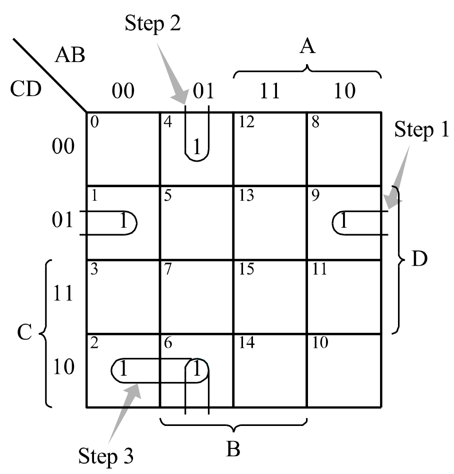
(a)

Example



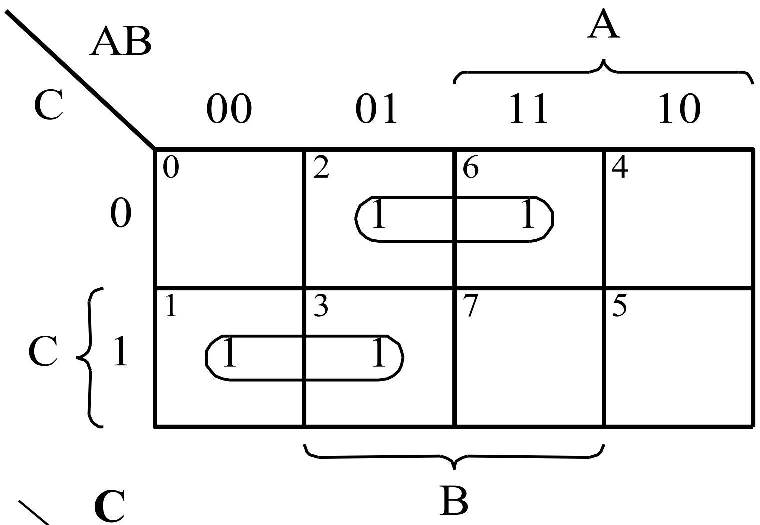
(b)

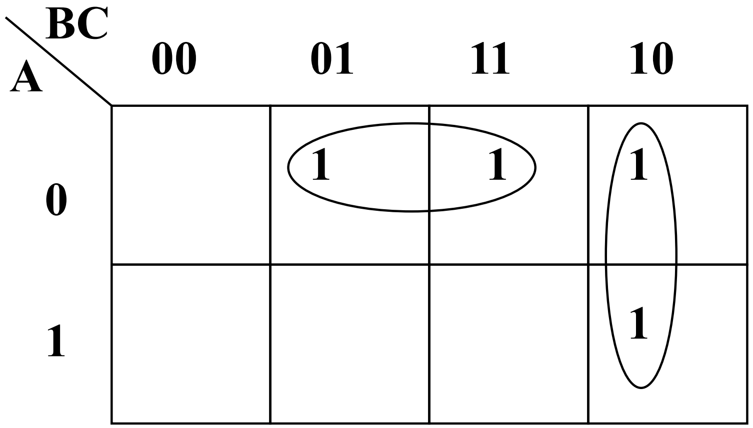
$$f(A,B,C,D) = \sum m(1,2,4,6,9)$$

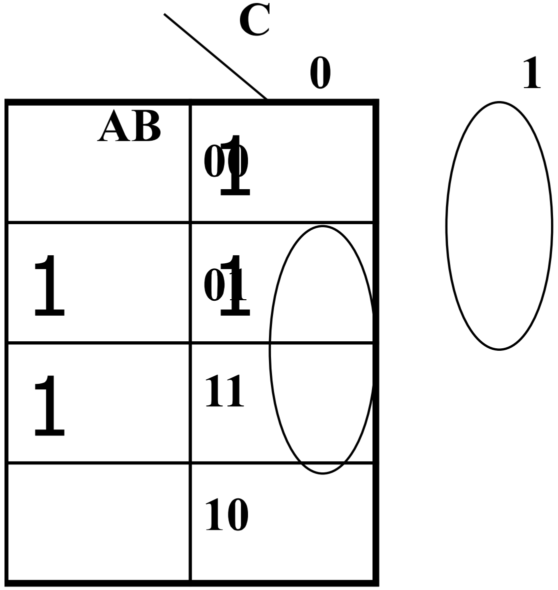


Different styles of drawing maps

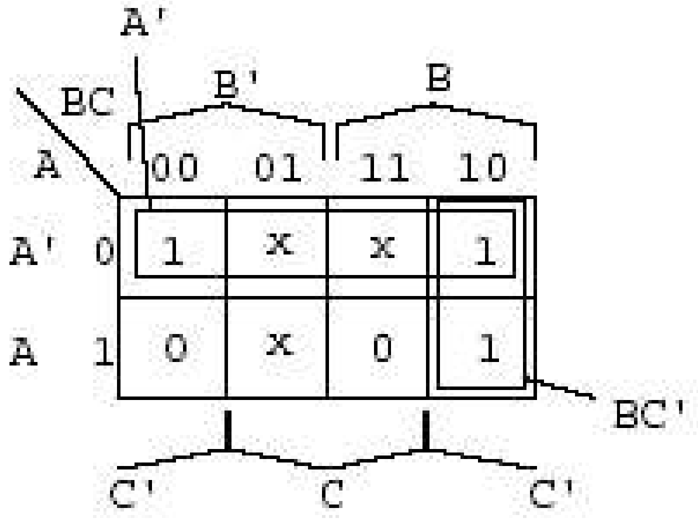
$f(A,B,C) = \sum m(1,2,3,6) = A'C + BC'$







Don’t-care condition



- Minterms that may produce either 0 or 1 for the function.
- They are marked with an ´ in the K-map.
- This happens, for example, when we don’t input certain minterms to the Boolean function.
- These don’t-care conditions can be used to provide further simplification of the algebraic expression.

(Example) F = A`B`C`+A`BC` + ABC`

d=A`B`C +A`BC + AB`C

F = A` + BC`