CSE 260

BRAC University

Boolean Functions (Solve ?)

- Examples:

SOP, minte -rm

F1= xyz'

F2= x + y'z

F4=xy'+x'z

F3=(x'y'z)+(x'yz)+(xy')

x	y	z	F1	F2	F3	F4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

From the truth table, F3=F4.
Can you also prove by algebraic manipulation that F3=F4?

Try it yourself: simplify the following equations

1. $f(a,b,c,d) = abc + abd + a'bc' + cd + bd'$

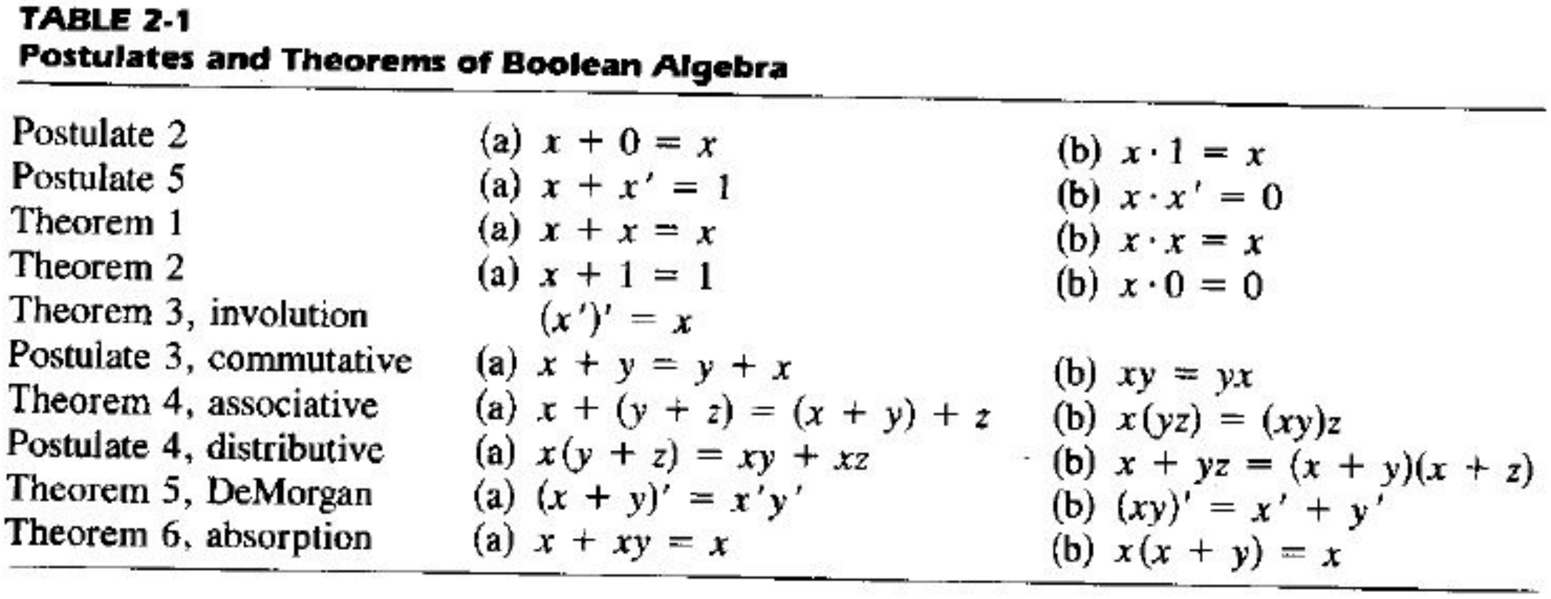
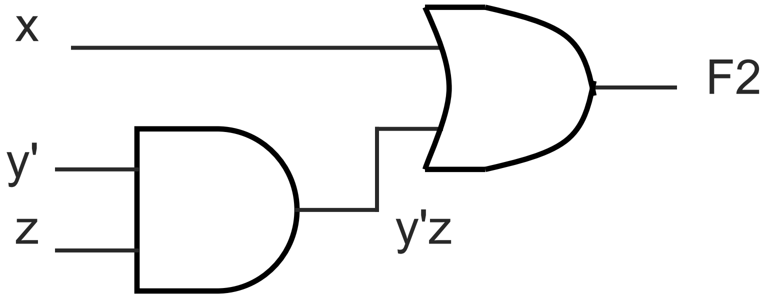


TABLE 2-1
Postulates and Theorems of Boolean Algebra

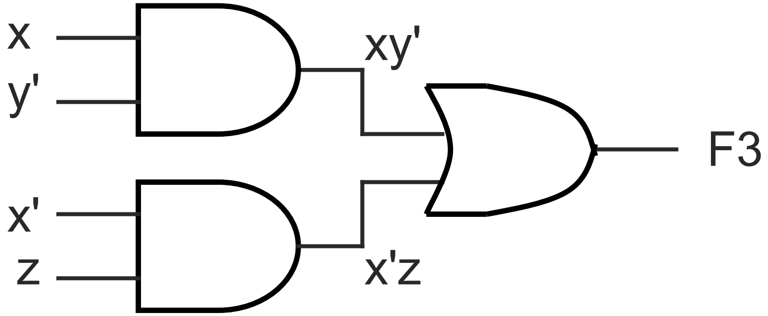
Postulate 2	(a) $x + 0 = x$	(b) $x \cdot 1 = x$
Postulate 5	(a) $x + x' = 1$	(b) $x \cdot x' = 0$
Theorem 1	(a) $x + x = x$	(b) $x \cdot x = x$
Theorem 2	(a) $x + 1 = 1$	(b) $x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Postulate 3, commutative	(a) $x + y = y + x$	(b) $xy = yx$
Theorem 4, associative	(a) $x + (y + z) = (x + y) + z$	(b) $x(yz) = (xy)z$
Postulate 4, distributive	(a) $x(y + z) = xy + xz$	(b) $x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a) $(x + y)' = x'y'$	(b) $(xy)' = x' + y'$
Theorem 6, absorption	(a) $x + xy = x$	(b) $x(x + y) = x$

Drawing Logic Circuit

(ii) **F2 = x + y'z** **(can assume that variables and their complements are available)**

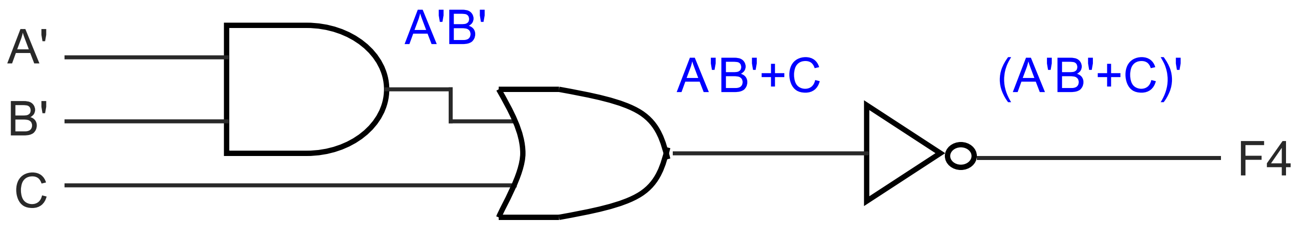


(iii) **F3 = xy' + x'z**



Analysing Logic Circuit

- When a logic circuit is provided, we can analyse the circuit to obtain the logic expression.
- Example: What is the Boolean expression of F4?



F4 = (A'B'+C)'

Review

- If expression F2= x + y'z is given, below is its truth table (obtained by “AND”ing y’ with z and then “OR”ing the product with x.)
- So from truth tabe, considering the 1’s we can say,
  F2=x’y’z+xy’z’ + xy’z + xyz’ + xyz
  which when simplified using postulates, will result to,
  F2=x+y’z

$$\text{so } \underbrace{x+y'z}_{\text{Simplified expression}} = \underbrace{x'y'z+xy'z' + xy'z + xyz' + xyz}_{\text{SOP Canonical Format of the expression}}$$

X	Y	Z	F2
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	1
1	1	1	1

Review: Solve it yourself

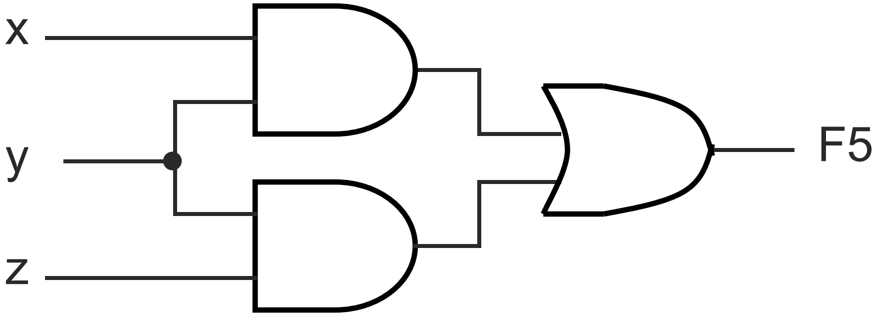
Q1. Draw a logic circuit for BD + BE + D'F

Q2. Draw a logic circuit for

A'BC + B'CD + BC'D + ABD'

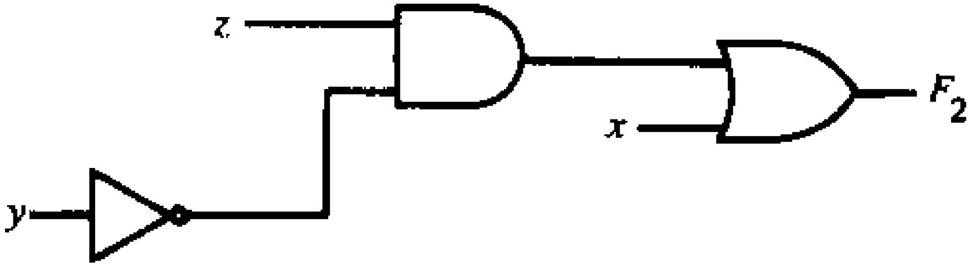
Review: Solve it yourself

- What is Boolean expression of F5?



Review: Solve it yourself

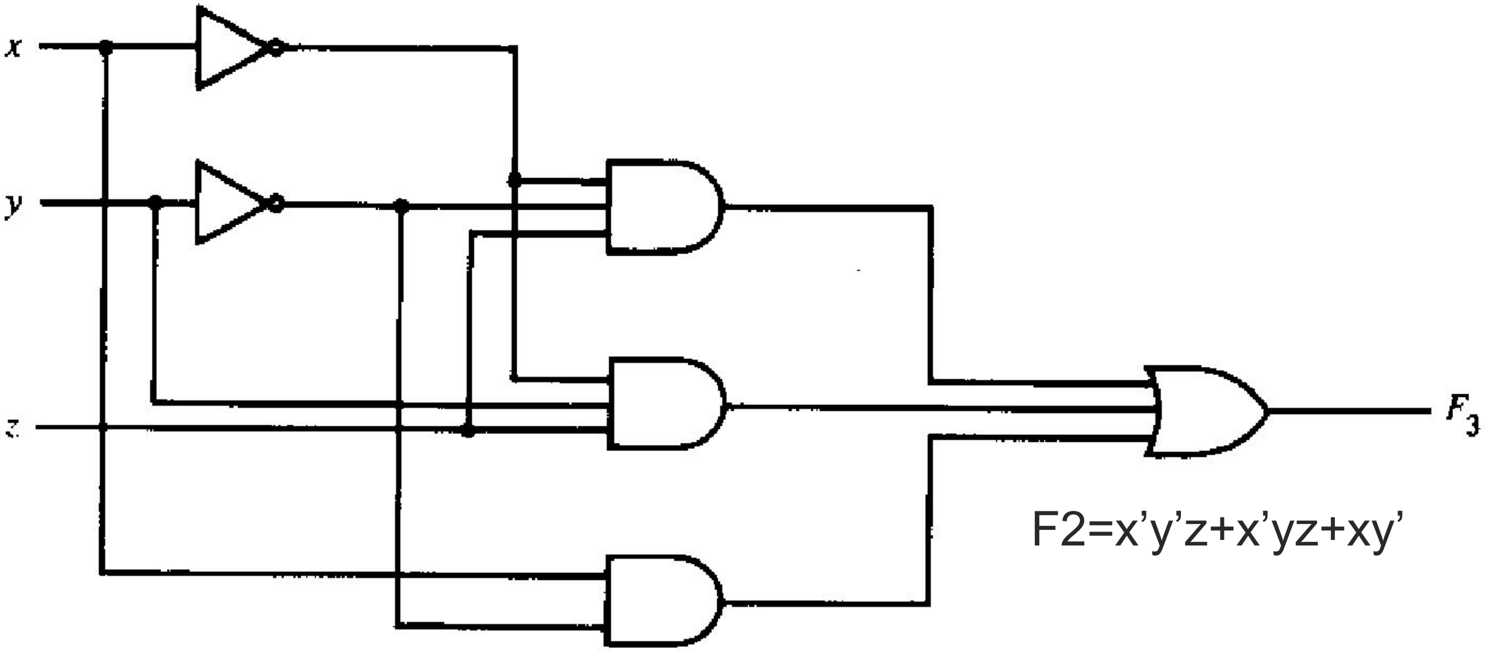
- What is Boolean expression of F2?



F2=zy'+x

Review: Solve it yourself

- What is Boolean expression of F3?



F2=x'y'z+x'yz+xy'

Standard Forms

- Two Standard Forms:
  Sum-of-Products and ***Product-of-Sums***
- **Literals**: a variable on its own or in its complemented form. Examples: x, x' , y, y'
- **Product Term**: a single literal or a logical product (AND) of several literals.

  Examples: x, xyz', A'B, AB

Standard Forms

- **Sum Term**: a single literal or a logical sum (OR) of several literals.
  Examples: x, $x+y+z'$, $A'+B$, $A+B$
- **Sum-of-Products (SOP) Expression**: a product term or a logical sum (OR) of several product terms.
  Examples: x, $x+yz'$, $xy'+x'yz$, $AB+A'B'$
- **Product-of-Sums (POS) Expression**: a sum term or a logical product (AND) of several sum terms.
  Examples: x, $x(y+z')$, $(x+y')(x'+y+z)$, $(A+B)(A'+B')$
- Every boolean expression can either be expressed as sum-of-products or product-of-sums expression.

Examples:

SOP: $x'y + xy' + xyz$

POS: $(x + y')(x' + y)(x' + z')$

Note: SOP and POS are complements

Minterm

- Consider two binary variables x, y.
- Each variable may appear as itself or in complemented form as literals (i.e. x, x' & y, y')
- For two variables, there are four possible combinations with the AND operator, namely:

  x'y', x'y, xy', xy
- These product terms are called the *minterms*.
- A **minterm** of *n* variables is the product of *n* literals from the different variables.
- In general, *n* variables can give 2^n minterms.

Maxterm

- In a similar fashion, a **maxterm** of *n* variables is the sum of *n* literals from the different variables.

  Examples: x'+y', x'+y, x+y',x+y

- In general, *n* variables can give 2^n maxterms.

Minterm & Maxterm

- The minterms and maxterms of 2 variables are denoted by m0 to m3 and M0 to M3 respectively:

Note:
$m_0=x'y'z'$
$m_0'=xyz$
$M_0=(x+y+z)$
$m_0'= M_0$

Minterms				Maxterms	
x	y	term	notation	term	notation
0	0	x'y'	m0	x+y	M0
0	1	x'y	m1	x+y'	M1
1	0	xy'	m2	x'+y	M2
1	1	xy	m3	x'+y'	M3

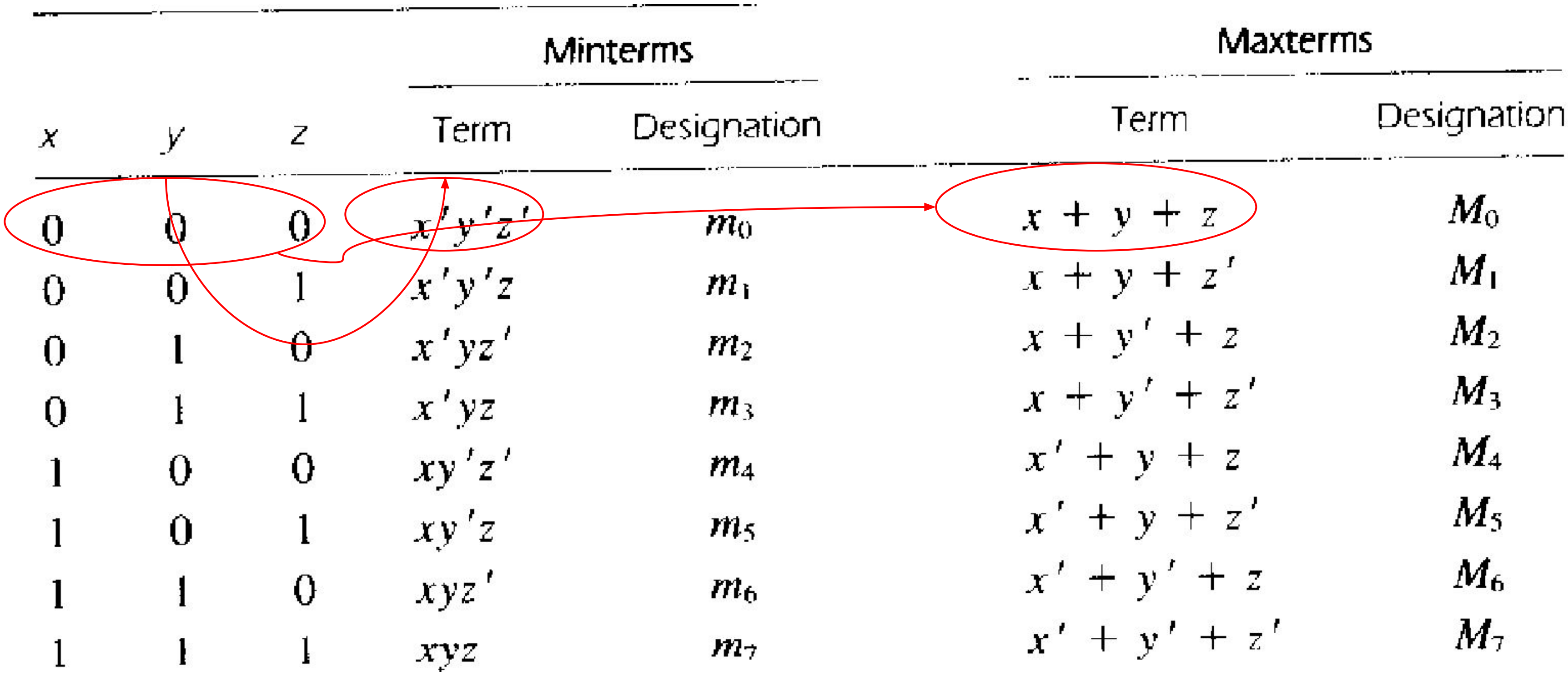
Minterm number is the decimal representation

Each minterm is the complement of the corresponding maxterm:

Example: m2 = xy'

m2' = (xy')' = x' + (y')' = x'+y = M2

Minterms and Maxterms



			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

Canonical Form: Sum of Minterms

Obtain Sum-of-Minterms by gathering/summing the minterms of the function (where result is a 1)

F1 = xyz' = Σ(m6)

F2 = x'y'z+xy'z'+xy'z+xyz'+xyz = Σ (m1,m4,m5,m6,m7)

F3 = x'y'z+x'yz+xy'z'
+xy'z
= Σ(m1,m3,m4,m5)

x	y	z	F1	F2	F3
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	0	1	0

Canonical Form: Product of Maxterms

- Maxterms are sum terms.
- **For Boolean functions, the maxterms of a function are the terms for which the result is 0.**
- Boolean functions can be expressed as Products-of-Maxterms.

Canonical Form: Product of Maxterms

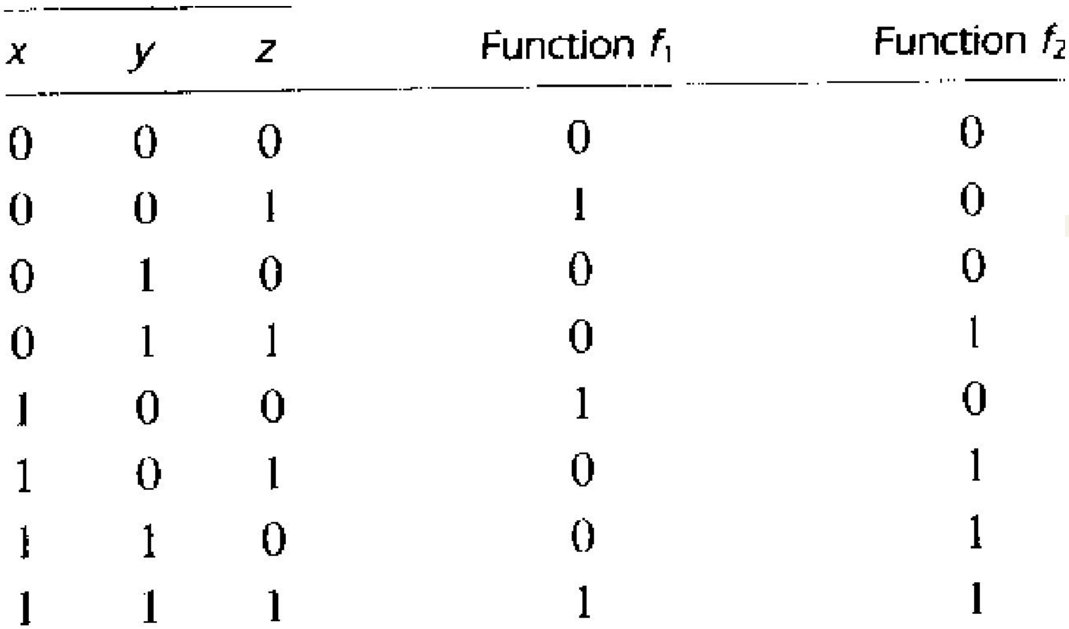
E.g.: F2 = Π(M0,M2,M3) = (x+y+z)(x+y'+z)(x+y'+z')

F3 = Π(M0,M2,M6,M7)

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

x	y	z	F1	F2	F3
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	0	1	0

SOP
->POS



x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- Considering the 1's-> $f_1=x'y'z+xy'z'+xyz'=m_1+m_4+m_7$
  $=\sum (m_{1,} m_4, m_7 \ )$
- Considering the 0's -> $f_1'=x'y'z'+x'yz'+x'yz+xy'z+xyz'=m_0+m_2+m_3+m_5+m_6$
- $(f_1')'=f_1=(x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z)$
- $f_1=M_0M_1M_2M_5M_6 \ =\prod(M_{0,}M_{2,}M_{3,}M_{5,}M_{6)}$

Try writing f_2 in (i) SOP and (ii) POS

Solution

- SOP: $F2 = x'yz+xy'z+xyz'+xyz$
- POS: $F2=(x+y+z)(x+y+z')\ (x+y'+z)(x'+y+z)$

Canonical Form: Product of Maxterms

- Why is this so? Take F2 as an example.
  F2 = Σ(m1,m4,m5,m6,m7)
- The complement function of F2 is:
  F2' = Σ(m0,m2,m3)
  = m0 + m2 + m3

Therefore:

F2 = (m0 + m2 + m3)'
= **m0' . m2' . m3'** DeMorgan
= **M0 . M2 . M3** m*x*' = M*x*
= Π(M0,M2,M3)

- Every Boolean function can be expressed as either Sum-of-Minterms or Product-of-Maxterms.

x	y	z	F2	F2'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Comparison between SOP and POS

- **Complement of a function**

  = original function-minterms

  =missing miniterms
- $m_j' = M_j$
- If $f(x,y,z)=\sum(1,3,6,7)=\prod(0,2,4,5)$

Function can be expressed in both SOP and POS.
Step 1: interchange $\sum$ with $\prod$
Step 2: list missing numbers from the original

Expressing Boolean Function in terms of Canonical SOP

Check if each term contains all variable, if not AND (x+x’) if x is the missing term

$F=A+B'C$

$=A(B+B')(C+C')+B'C(A+A')$

$=(AB+AB')(C+C')+B'C(A+A')$

$=AB(C+C')+AB'(C+C')+B'C(A+A')$

$=ABC+ABC'+\boldsymbol{AB'C}+AB'C'+\boldsymbol{AB'C}+A'B'C$

$=ABC+ABC'+AB'C+AB'C'+A'B'C$

$=\sum(1,4,5,6,7)$

Try using Truth Table

Expressing Boolean function in terms of Canonical POS

1. Often distributive law $(x+yz)=(x+y)(x+z))$ is used
2. If then terms, like x, are missing, OR xx'

$F=xy+x'z=(xy+x')(xy+z)=(x+x')(y+x')(x+z)(y+z)$

$=(1)(y+x')(x+z)(y+z)=)(y+x')(x+z)(y+z)$

Each POS is missing a term so OR missing terms

$=(y+x'+zz')(x+z+yy')(y+z+xx')$

Again applying distributive law

$=(x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')=\prod(0,2,4,5)$

Try using Truth Table

Solving it yourself

1. A Boolean function of 5 variables can have up to ____ minterms.

2. Given a Boolean function $F\ (A,B,C) = \Sigma\ m(0,\ 5,\ 7)$. Which of the following is correct?
   a) $F'\ (A,B,C) = \Sigma m(0,5,7)$ b) $F\ (A,B,C) = \Pi M(1,2,3,4,6)$
   c) $F\ (A,B,C) = \Pi M(0,5,7)$ d) $F'\ (A,B,C) = \Sigma m(1,2,3)$
   e) None above

3. Given a Boolean function $F\ (x,y,z) = y'.(x + z') + x'.z$. Which of the following is correct?
   a) $F\ (x,y,z) = \Sigma m(0,1)$ b) $F\ (x,y,z) = \Sigma m(0,1,4,5)$
   c) $F\ (x,y,z) = \Sigma m(0,1,2,3,4)$ d) $F\ (x,y,z) = \Sigma m(0,1,3,4,5)$
   e) $F\ (x,y,z) = \Sigma m(1,2,3,4,5)$

Solution

- 1. 2^5=32
- 2. (b)
- 3.(d)

Working:

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

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

=x'y'z'+x'y'z+x'yz+xy'z'+xy'z

=m0+m1+m3+m4+m5

$= F(x,y,z) = \Sigma m(0,1,3,4,5)$

CSE260 Digital Logic Design

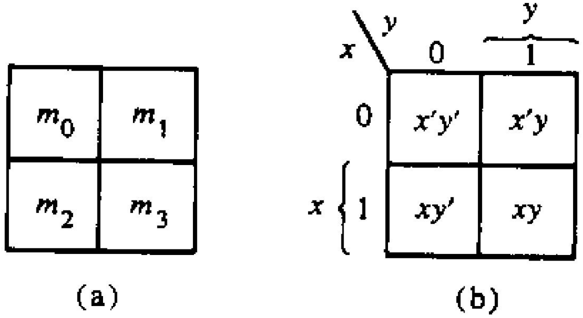
Karnaugh Maps

Khadija Rasul

Aniqua Zereen

The map method

- The map method provides a simple procedure for minimizing Boolean functions.
- The map is made up of squares where each square represent a minterm.
- We represent each minterm of an equation by ‘1’s
- We groups the adjacent ‘1’s in groups of 2, groups of 4 or groups of 8 and so on.
- In those grouping try to find the common literal.
- The map method is often known as Karnaugh Map or Veitech Diagram. In short we will call it K-map



m_0	m_1
m_2	m_3

(a)

$x \backslash y$	0	1 (y)
0	$x'y'$	$x'y$
(x) 1	xy'	xy

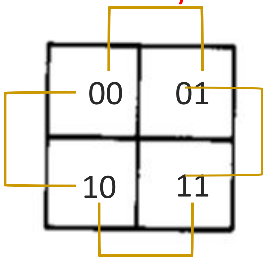
(b)

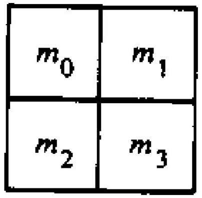
Two variable map

- There are 4 minterm of 2 variable; hence the map is four squares, one for each minterm.

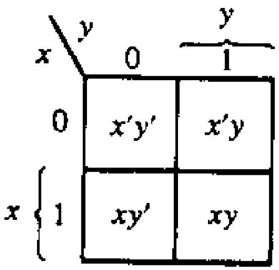
Only 1 bit changes

Only 1 bit changes





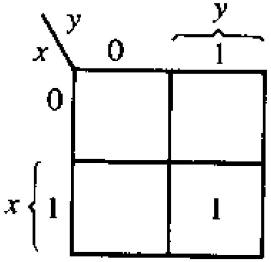
(a)



(b)

Representing function in K-map

- (a) F1=xy;



(a) xy

- (b) F2=xy'+xy+x'y

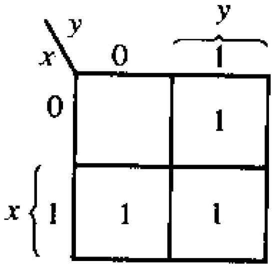
  =x(y'+y)+x'y

  =x+x'y

  =(x+x')(x+y)

  =(1)(x+y)

  =x+y



(b) $x + y$

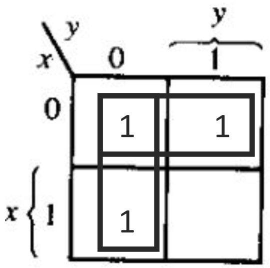
Try it yourself

F=xy’+x’y’+x’y,

Simplify the equation using k-map

Solution

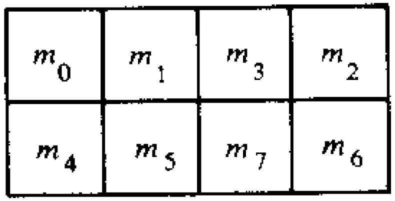
- $F = xy' + x'y' + x'y = x' + y'$



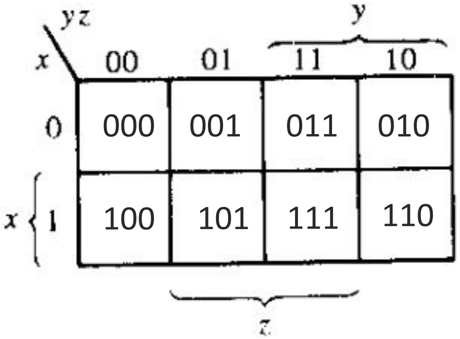
Three variable map

- Sequence in 3-variable map is not in binary sequence but in reflected code. In reflected code, at a time only 1 bit changes from 0 to 1 and 1 to 0.
- Numbering of minterms follow binary numbers. Example m5 is at 101 i.e. row 1 and column 01.

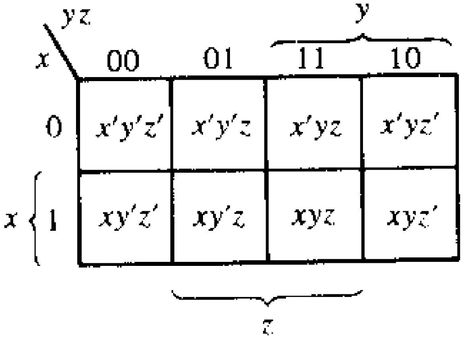
Note: Any 2 adjacent square differ by 1 variable (which is primed in one and unprimed in the other)



m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6



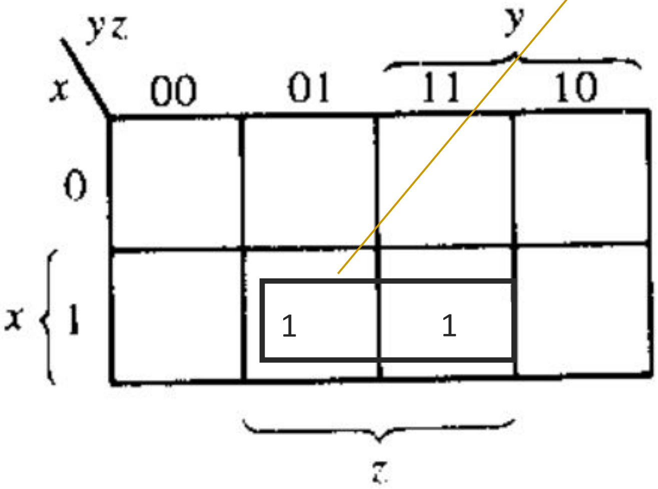
	00	01	11	10
0	000	001	011	010
1	100	101	111	110

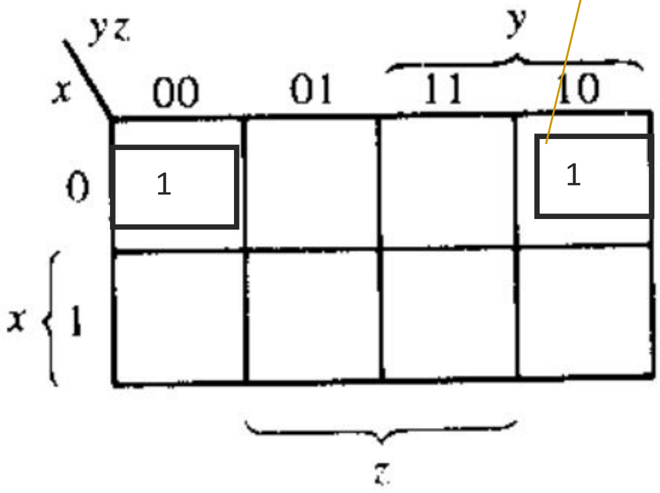


	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

minterms in the adjacent square are ORed, will remove the different term. As a result , we get a single ANDed term of only 2 common literals

- F1=∑(5,7)=xy’z+xyz=xz(y’+y)=xz
  - F2=∑(0,2)=x’y’z’+x’yz'=x’z’(y’+y)=x’z’

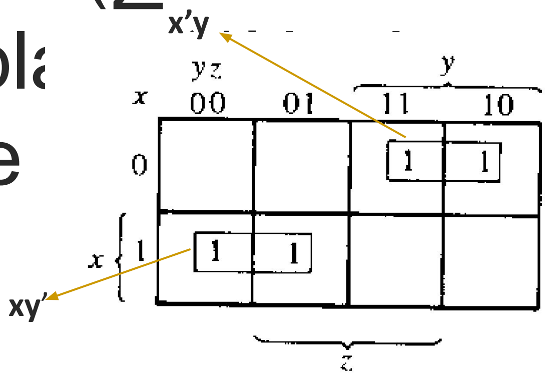




Simplifying functions using k-map

- F=∑(2,3,4,5)=x'yz+x'yz'+xy'z'+xy'z (note: in other words, F= (∑ (010,011,100,101) so pl positions and group the

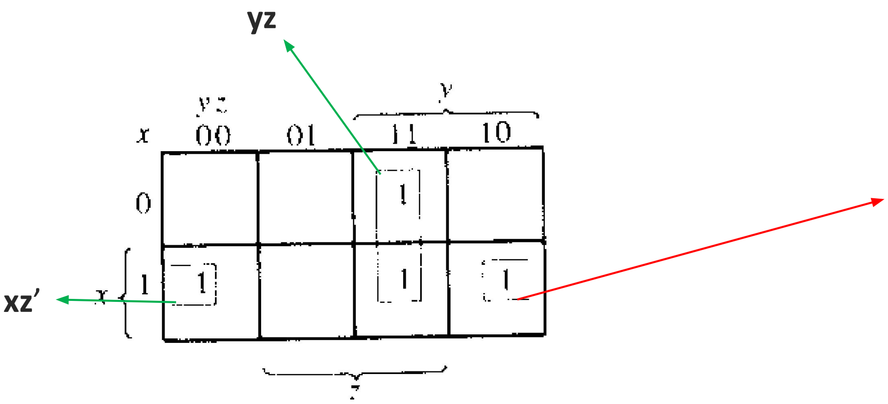
F=x'y+xy'



- Grouping of 1 can be done in groups of two '1',four '1',eight '1'….2^n '1'.

- $F1=x'yz+xy'z'+xyz+xyz'=xz'+yz$

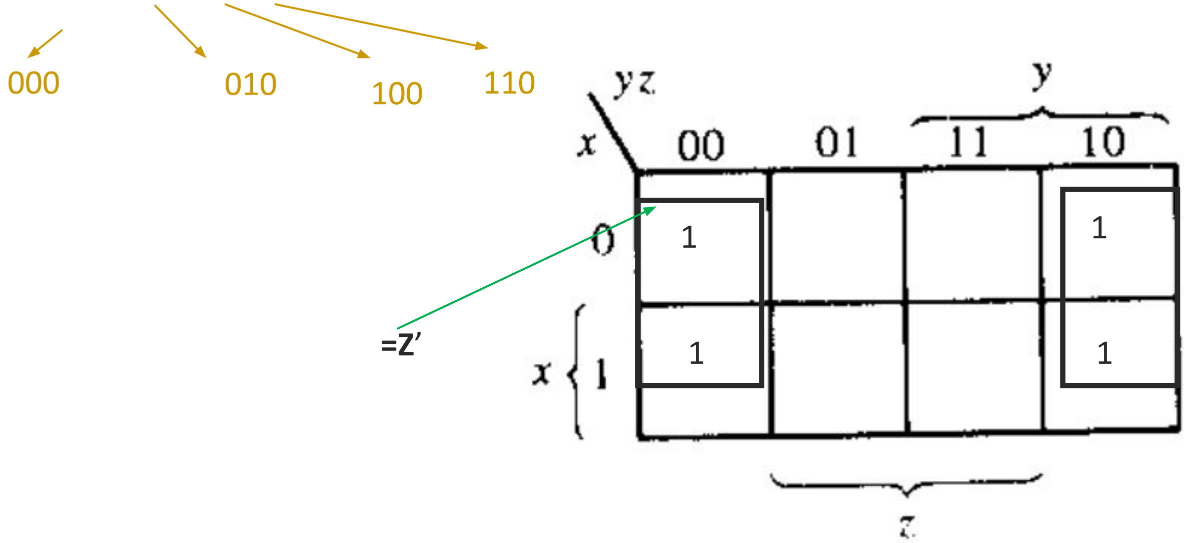
011 100 111 110



x \ yz	00	01	11	10
0			1	
1	1		1	1

Notice even 2 opposite edges of the map are adjacent, thus they can be grouped

- If F=∑(0,2,4,6), then simplify it



x \ yz	00	01	11	10
0	1			1
1	1			1

$$m_0 + m_2 + m_4 + m_6 = x'y'z' + x'yz' + xy'z' + xyz'$$

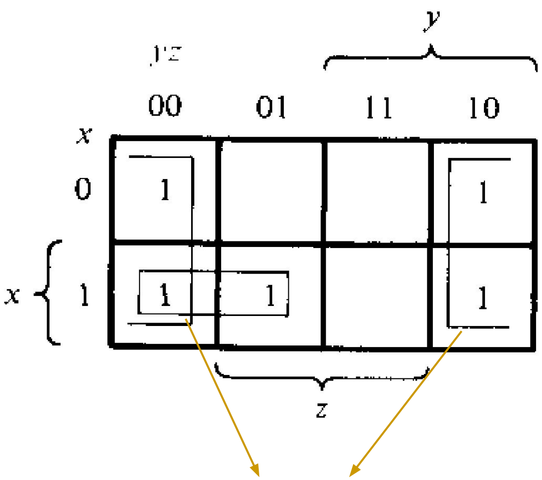
$$= x'z'(y' + y) + xz'(y' + y)$$

$$= x'z' + xz' = z'(x' + x) = z'$$

Try it yourself

- $F=\sum(0,2,4,5,6)$, Simplify it.

Solution



x \ yz	00	01	11	10
0	1			1
1	1	1		1

F= xy' + z'

Try it yourself:

- F=A’C+A’B+AB’C+BC

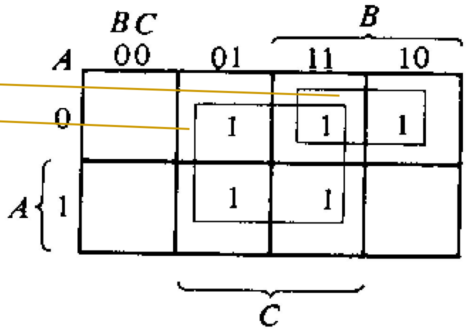
a) Express it as SOP (i.e. ∑)

b) Simplify it using Kmap

Solution

- (a) F= ∑(1,2,3,5,7)

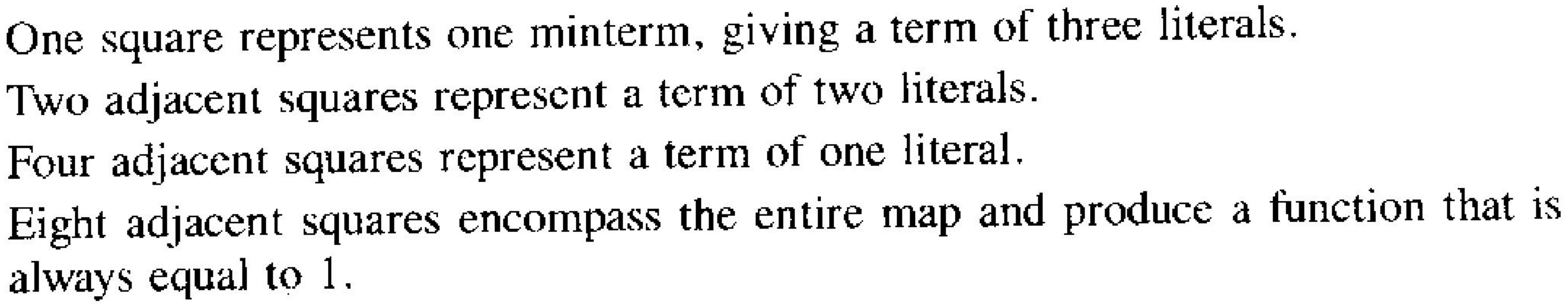
(b)F=C + A’B



A	00	01	11	10
0		1	1	1
1		1	1	

To find A’C, coincide A’ (first row) with C(middle 2 column)
To find A’B, coincide A’ (first row) with B(last 2 column)
To find BC, coincide B (last 2 column) with C(middle 2 column)
To find AB’C, is m5(2nd row 2nd column)

Summary of grouping in 3-variable map

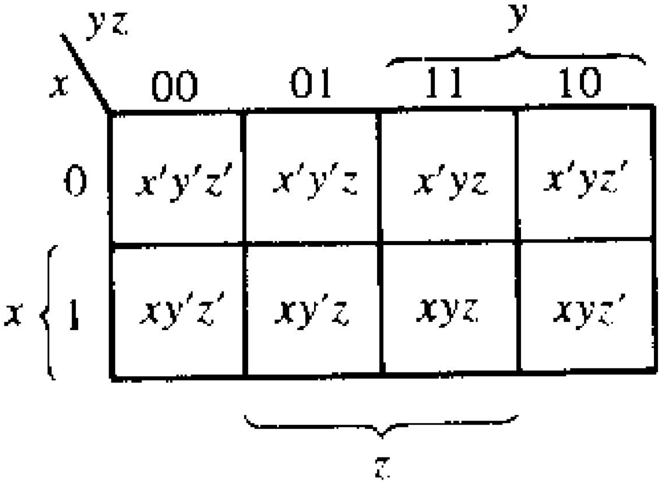


One square represents one minterm, giving a term of three literals.

Two adjacent squares represent a term of two literals.

Four adjacent squares represent a term of one literal.

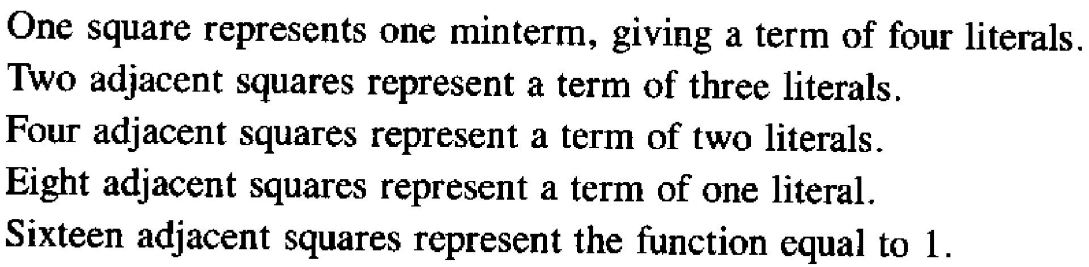
Eight adjacent squares encompass the entire map and produce a function that is always equal to 1.



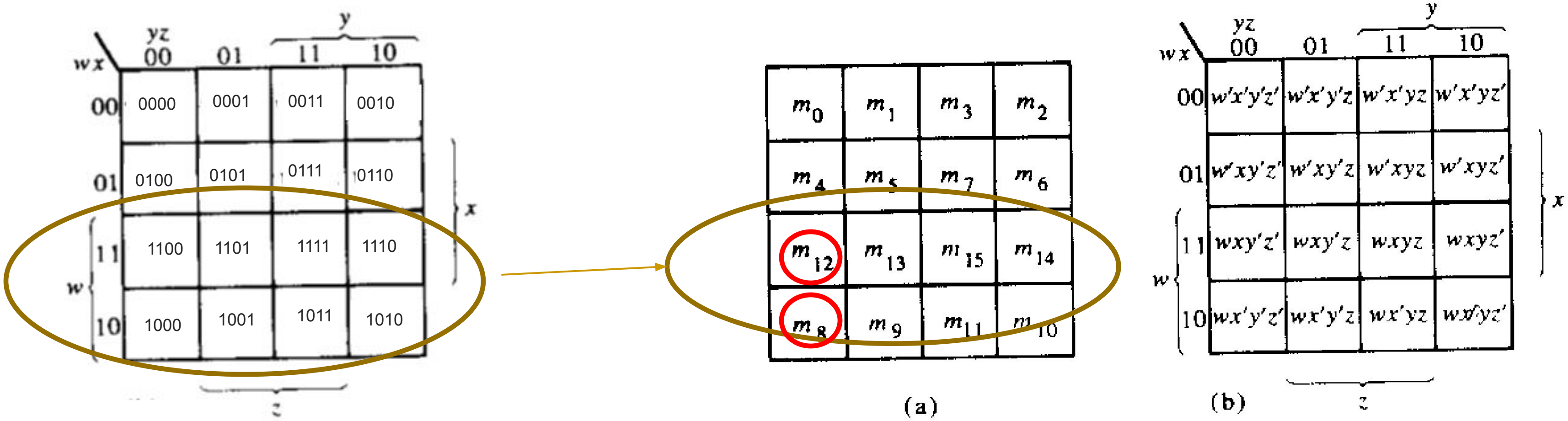
x \ yz	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

Four variable map

- Adjacent squares can be side by side or lie in 4 corner or top-bottom or left-right edge condition. Example, m0 -m2 are adjacent and similarly m3-m11 are adjacent



One square represents one minterm, giving a term of four literals.
Two adjacent squares represent a term of three literals.
Four adjacent squares represent a term of two literals.
Eight adjacent squares represent a term of one literal.
Sixteen adjacent squares represent the function equal to 1.



wx \ yz	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

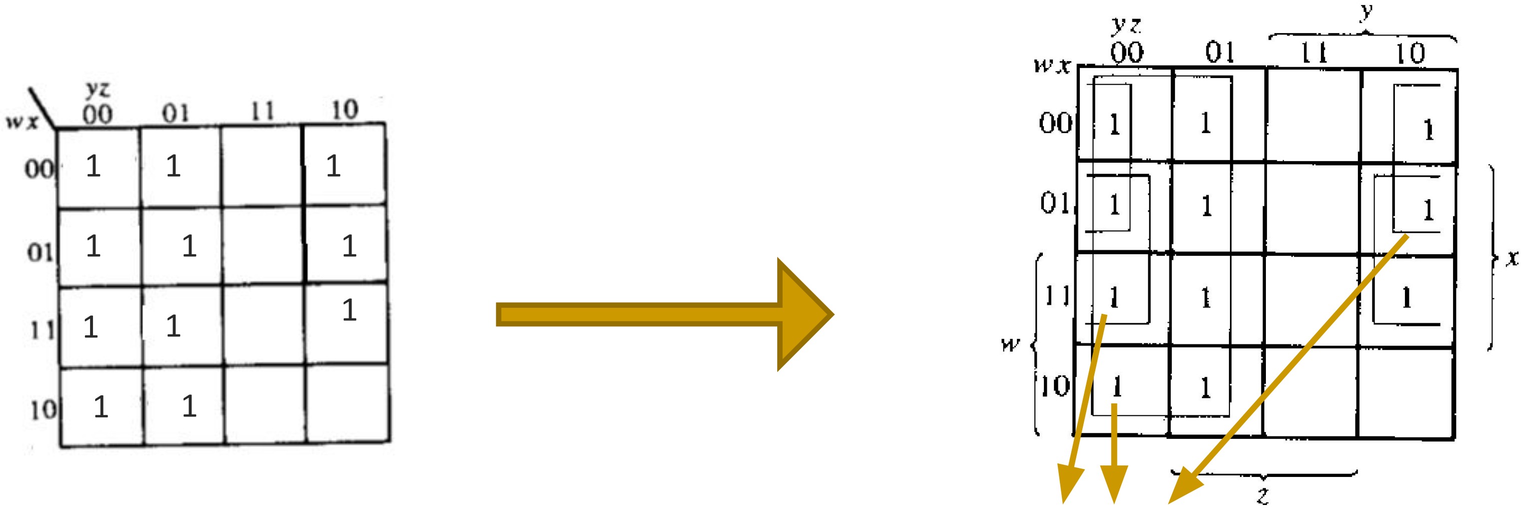
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

(a)

wx \ yz	00	01	11	10
00	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
01	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
11	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

(b)

- Simplify F=∑(0,1,2,4,5,6,8,9,12,13,14)
- Note: it is allowed to use the same ‘1’ more than once.



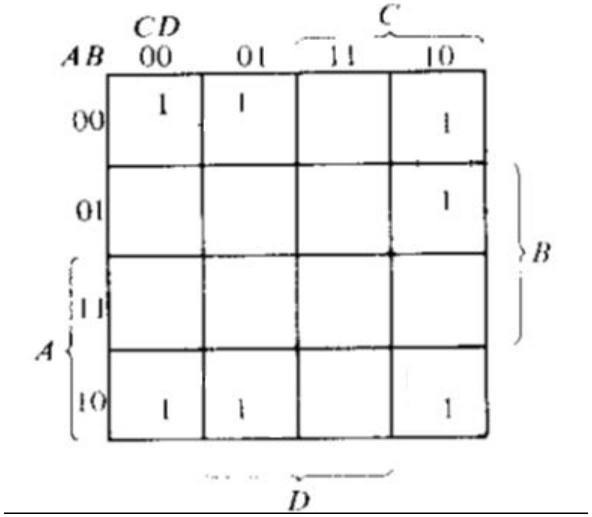
wx \ yz	00	01	11	10
00	1	1		1
01	1	1		1
11	1	1		1
10	1	1		

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

F= xz’+ y’+ w’z’

Try it yourself

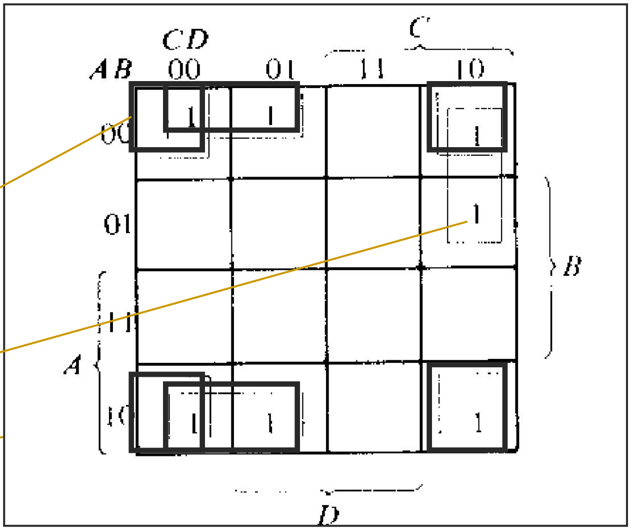
- F=A’B’C’+B’CD’+A’BCD’+AB’C’



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

Solution

- F=A’B’C’+B’CD’+A’BCD’+AB’C’



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

F=B’D’+B’C’+A’CD’

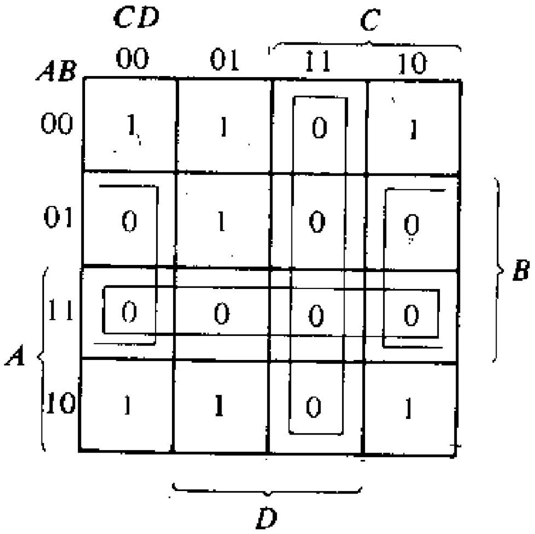
Product of Sum (POS) simplification

- F is represented with the ‘1’s
  -> F’ is represented with the ‘0’s

  -> (F’)’ =F so find F’ and then find it’s complement using DeMorgan’s law.
- In other word, ‘1’s represent the minterms whereas ‘0’s represent the maxterms.
- If Equation is provided in SOP format, mark the ‘1’s and then mark the remaining with ‘0’s. On the other hand, if Equation is provided in POS format, mark the ‘0’s and then mark the remaining with ‘1’s. Once marking is done, function can be simplified in SOP or POS format.

SOP to POS

- SOP-> F=∑(0,1,2,5,8,9,1
- Combining ‘1’s we get F=B’D’+B’C’+A’C’D
- Combining ‘0’s we get F’=AB+CD+BD’
- Applying De-morgan F=(A’+B’)(C’+D’)(B’+D) <-POS



AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

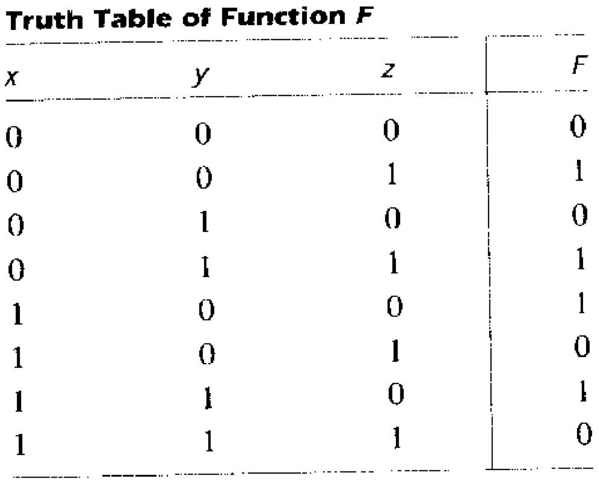
- If SOP: $\sum(1,3,4,6)$

then POS: $\prod(0,2,5,7)$

So grouping ‘1’s, F=x’z+xz’

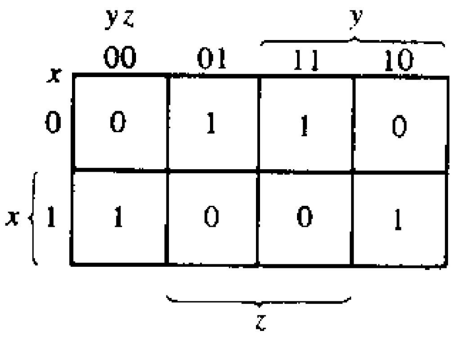
Grouping ‘0’s, F’=xz+x’z’

therefore, F= (x’+z’)(x+z)



Truth Table of Function *F*

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



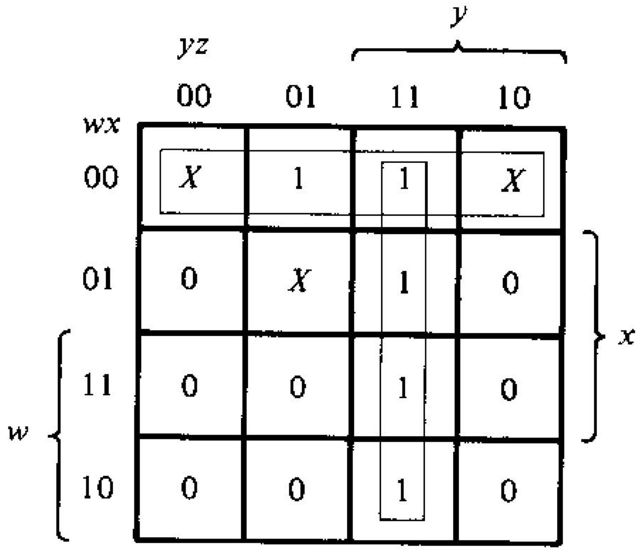
x \ yz	00	01	11	10
0	0	1	1	0
1	1	0	0	1

Don’t Care condition

- Don't cares in a Karnaugh map, or truth table, may be either **1**s or **0**s, as long as we don't care what the output is for an input condition we never expect to see.
- We plot these cells with a special sign, ‘X’, among the normal **1**s and **0**s.
- When forming groups of cells, treat the don't care cell as either a **1** or a **0**, or ignore the don't cares. This is helpful if it allows us to form a larger group than would otherwise be possible without the don't cares.
- There is no requirement to group all or any of the don't cares. (i.e. do not make any group only consisting the don’t cares). Only use them in a group if it simplifies the logic.

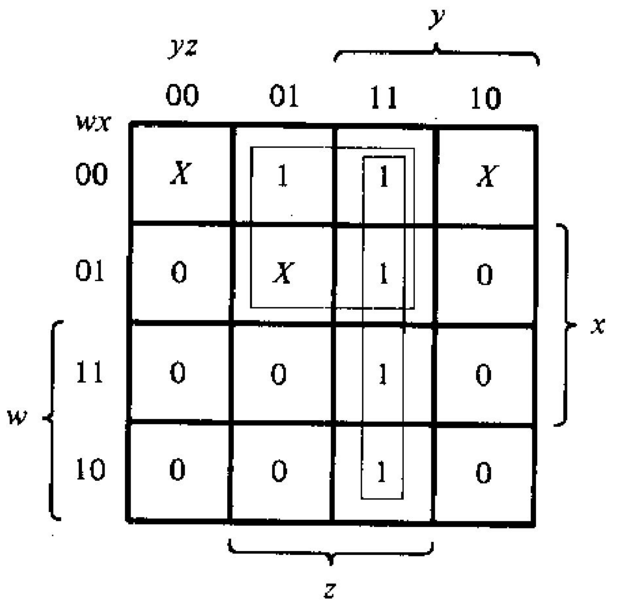
Simplify F(w,x,y,z)= ∑(1,3,7,11,15) and don’t care d(w,x,y,z)= ∑(0,2,5)

Solution: Even though 2 solutions are not same, but either 1 is acceptable. Such scenario only applicable for Don’t care scenarios!



wx \ yz	00	01	11	10
00	X	1	1	X
01	0	X	1	0
11	0	0	1	0
10	0	0	1	0

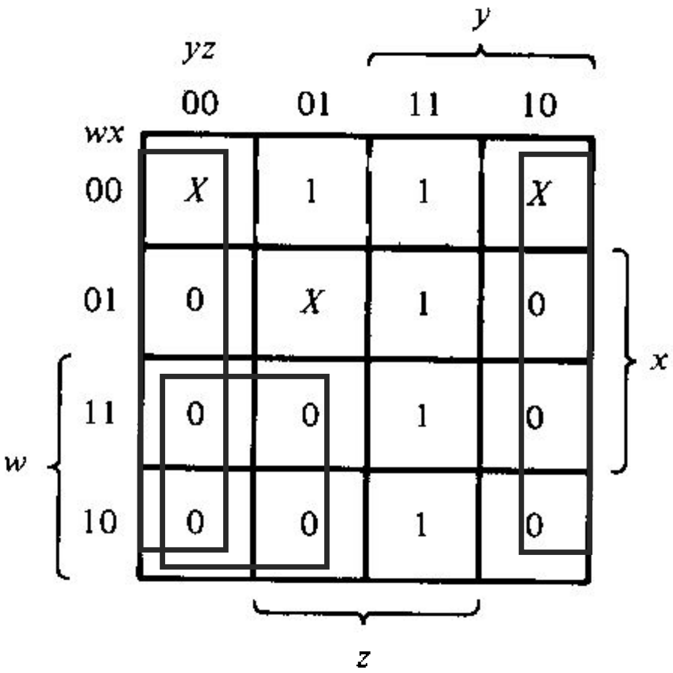
(a) $F = yz + w'x'$



wx \ yz	00	01	11	10
00	X	1	1	X
01	0	X	1	0
11	0	0	1	0
10	0	0	1	0

(b) $F = yz + w'z$

Complement of the function and POS considering don’t cares



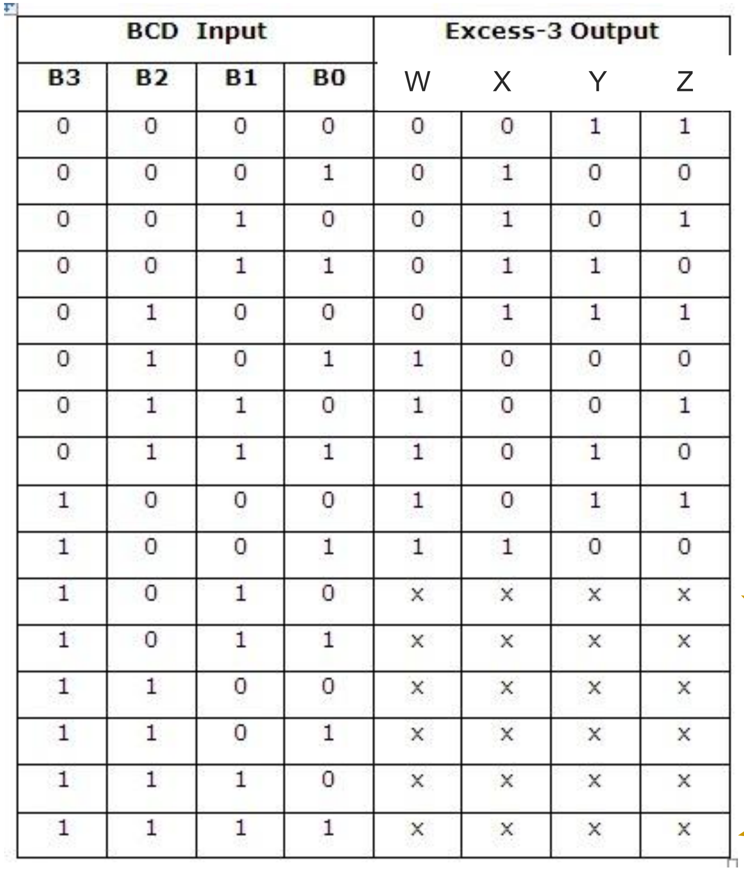
wx \ yz	00	01	11	10
00	X	1	1	X
01	0	X	1	0
11	0	0	1	0
10	0	0	1	0

F’=z’+wy’

F=(z)(w’+y)

APPLICATIONS OF K-MAP

Example 1: Design and draw the circuit diagram of BCD to Excess-3 code converter



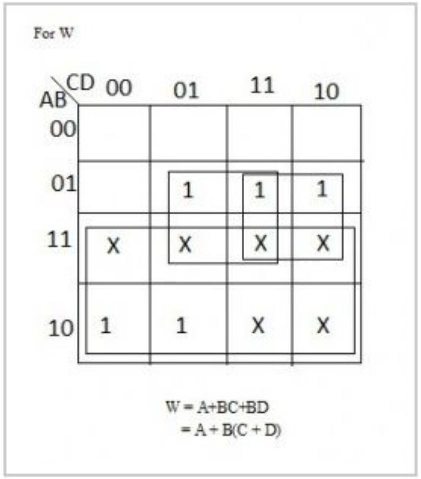
BCD Input				Excess-3 Output			
B3	B2	B1	B0	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

Q1.Treat each W, X, Y and Z as a function and write them as SOP

Q2. Based on the SOP, draw the circuit diagram of BCD to Excess-3 code converter

W X
Y Z

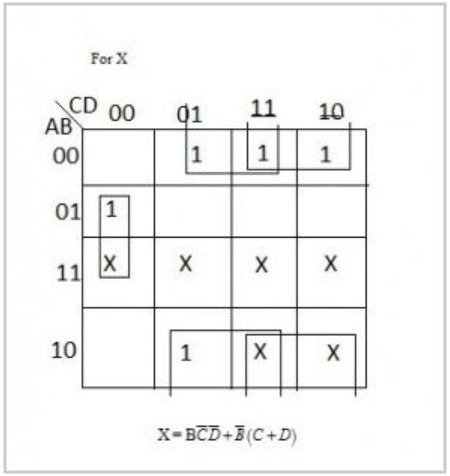
Any value higher than 9 will result into don’t care output



For W

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

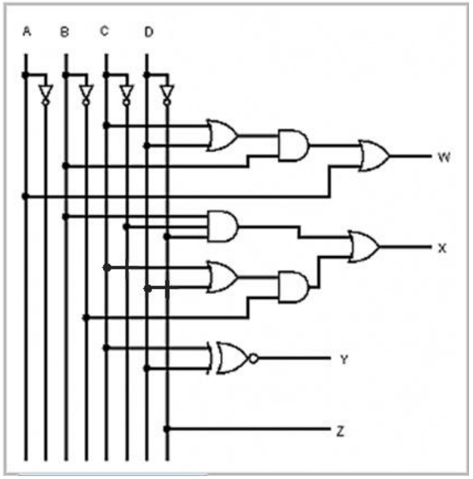
$W = A+BC+BD$
$= A + B(C + D)$

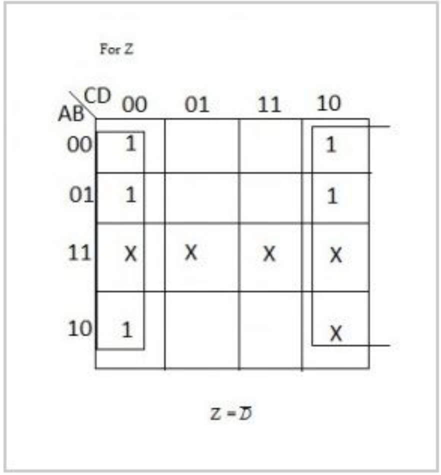


For X

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

$X = B\overline{C}\overline{D} + \overline{B}(C+D)$

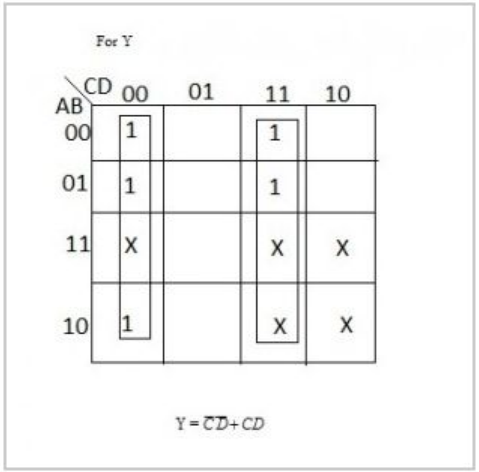




For Z

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

$Z = \overline{D}$

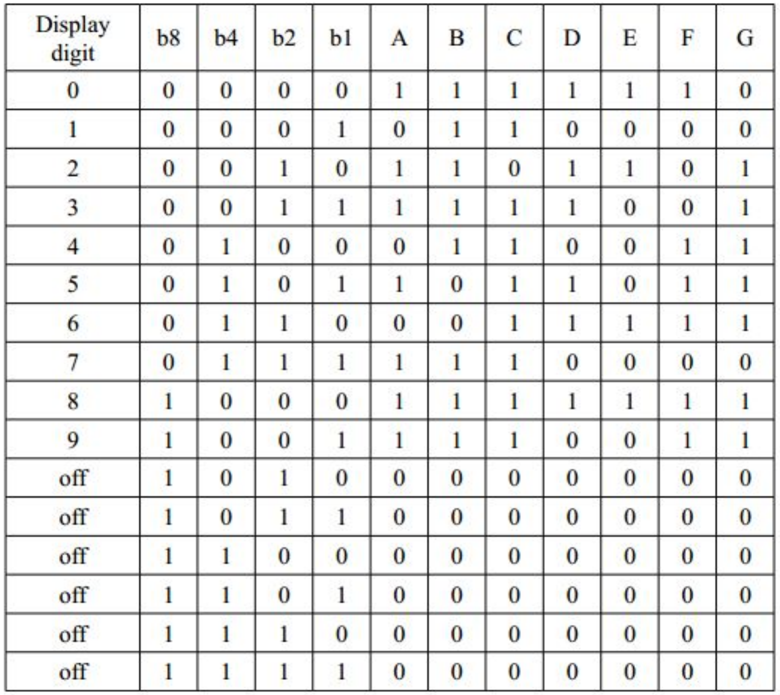


For Y

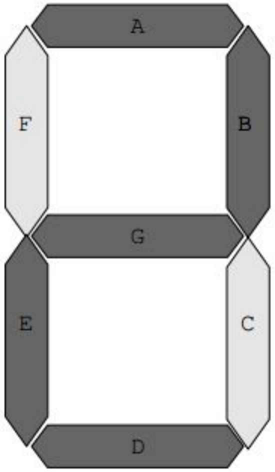
AB \ CD	00	01	11	10
00	1		1	
01	1		1	
11	X		X	X
10	1		X	X

$Y = \overline{C}\overline{D}+CD$

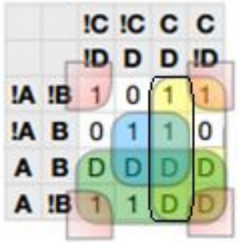
Example 2: 7 segment display board



Display digit	b8	b4	b2	b1	A	B	C	D	E	F	G
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
off	1	0	1	0	0	0	0	0	0	0	0
off	1	0	1	1	0	0	0	0	0	0	0
off	1	1	0	0	0	0	0	0	0	0	0
off	1	1	0	1	0	0	0	0	0	0	0
off	1	1	1	0	0	0	0	0	0	0	0
off	1	1	1	1	0	0	0	0	0	0	0



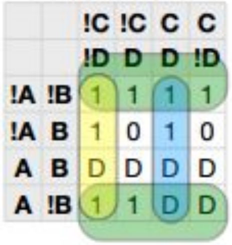
Segment a



		!C	!C	C	C
		!D	D	D	!D
!A	!B	1	0	1	1
!A	B	0	1	1	0
A	B	D	D	D	D
A	!B	1	1	D	D

A + CD+ B D + !B !D

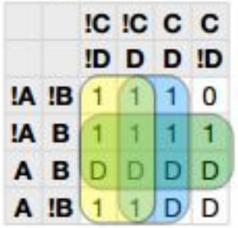
Segment b



		!C	!C	C	C
		!D	D	D	!D
!A	!B	1	1	1	1
!A	B	1	0	1	0
A	B	D	D	D	D
A	!B	1	1	D	D

!B + !C !D + C D

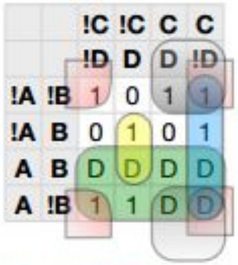
Segment c



		!C	!C	C	C
		!D	D	D	!D
!A	!B	1	1	1	0
!A	B	1	1	1	1
A	B	D	D	D	D
A	!B	1	1	D	D

B + !C + D

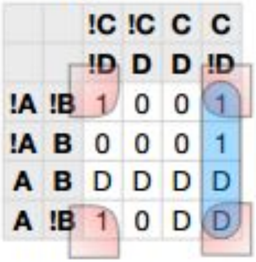
Segment d



		!C	!C	C	C
		!D	D	D	!D
!A	!B	1	0	1	1
!A	B	0	1	0	1
A	B	D	D	D	D
A	!B	1	1	D	D

A + C !D + !B !D + !B C + B !C D

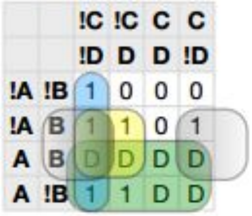
Segment e



		!C	!C	C	C
		!D	D	D	!D
!A	!B	1	0	0	1
!A	B	0	0	0	1
A	B	D	D	D	D
A	!B	1	0	D	D

C !D + !B !D

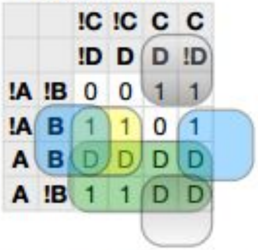
Segment f



		!C	!C	C	C
		!D	D	D	!D
!A	!B	1	0	0	0
!A	B	1	1	0	1
A	B	D	D	D	D
A	!B	1	1	D	D

A + B !C + !C !D + B !D

Segment g



		!C	!C	C	C
		!D	D	D	!D
!A	!B	0	0	1	1
!A	B	1	1	0	1
A	B	D	D	D	D
A	!B	1	1	D	D

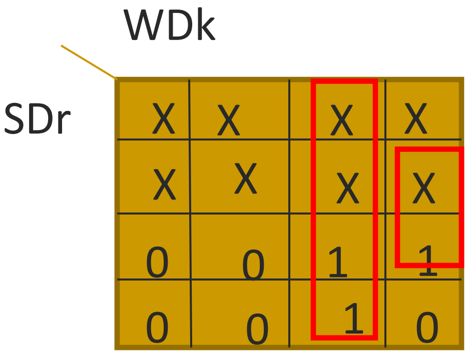
A + B !C + B !D + !B C

Try it by your self

- A **car garage** has a front door and one window, each of which has a sensor to detect whether it is open. A third sensor detects whether it is dark outside. A security system for the garage follows this rule: the alarm rings if the **alarm switch is turned on** and **either the front door is not closed or it is dark** and **the side window is not closed**.

Solution

Switch On	Door	Window	Dark	AlarmRing
0	0	0	0	x
0	0	0	1	x
0	0	1	0	x
0	0	1	1	x
0	1	0	0	x
0	1	0	1	x
0	1	1	0	x
0	1	1	1	x
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



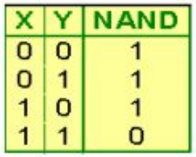
X	X	X	X
X	X	X	X
0	0	1	1
0	0	1	0

AlarmRing=WDk+WDr

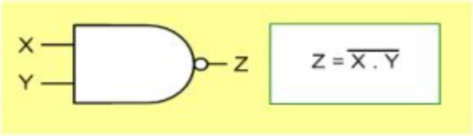
Assuming
AlarmRing = 1 if Alarm rings, 0 otherwise.
Switch On = 1 if alarm switch is on, 0 otherwise
Door = 1 if door is not closed, 0 otherwise.
Window = 1 if window is not closed.
Dark = 1 if it is dark outside, 0 otherwise.

UNIVERSAL LOGIC GATES

NAND

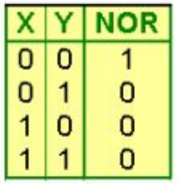


X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

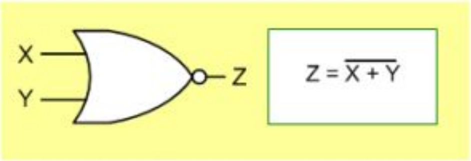


$Z = \overline{X . Y}$

NOR



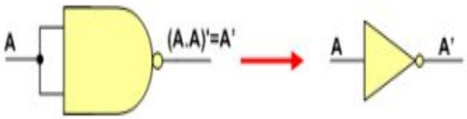
X	Y	NOR
0	0	1
0	1	0
1	0	0
1	1	0



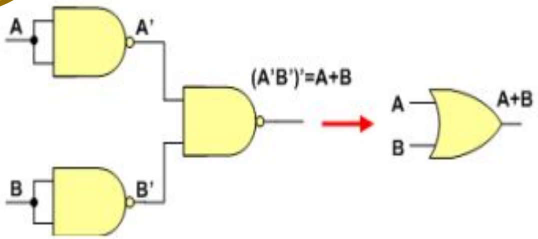
$Z = \overline{X + Y}$

Using NAND

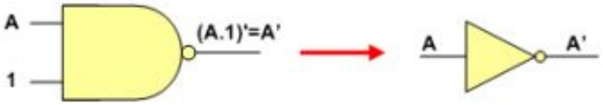
NOT Gate



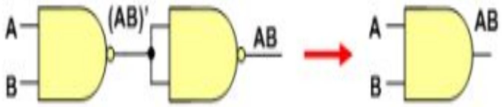
OR Gate



NOT Gate

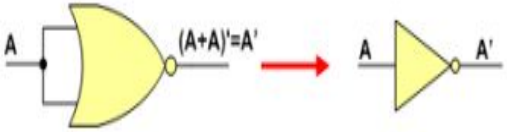


AND Gate

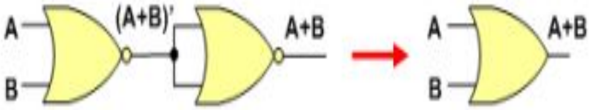


Using NOR

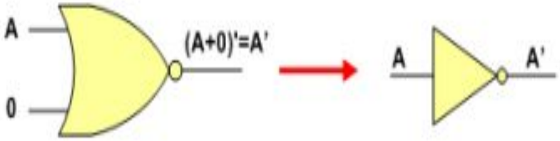
NOT Gate



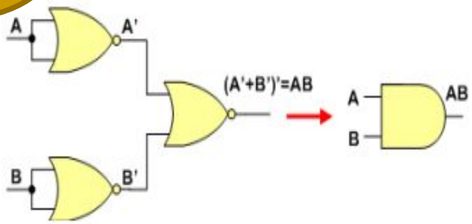
OR Gate



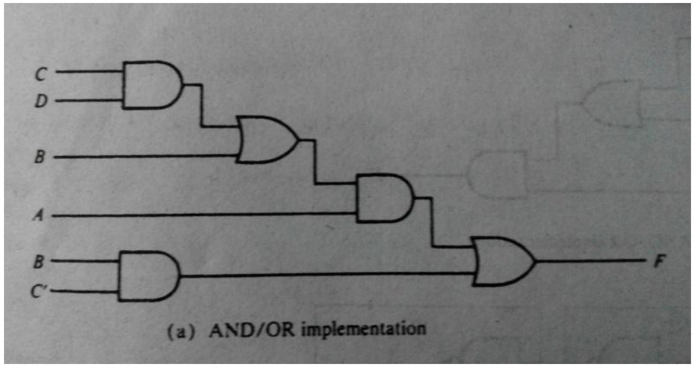
NOT Gate



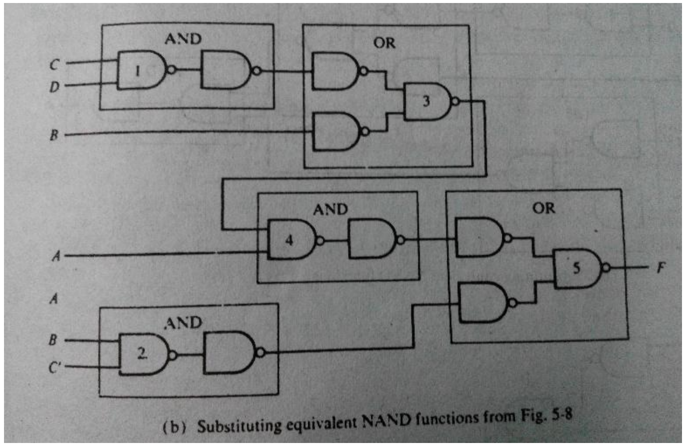
AND Gate



Use NAND to represent F=A(B+CD)+BC’

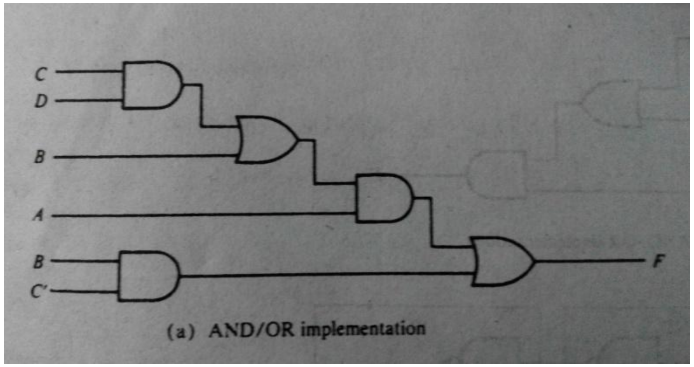


(a) AND/OR implementation

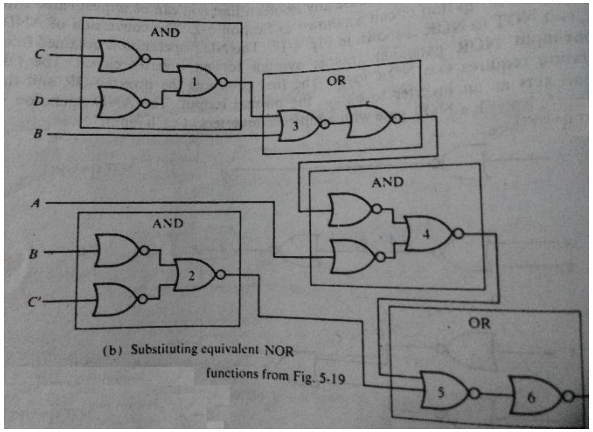


(b) Substituting equivalent NAND functions from Fig. 5-8

Use NOR to represent $F=A(B+CD)+BC'$



(a) AND/OR implementation



(b) Substituting equivalent NOR functions from Fig. 5-19

A Must Self study from the book

- Morris Mano: Chapter 3.6 (NAND and NOR implementation)to Chapter 3.7 (Other two level implementation)