Digital Electronics

Prof. Mitul Patel, Assistant Professor
Electronics & Communication Engineering

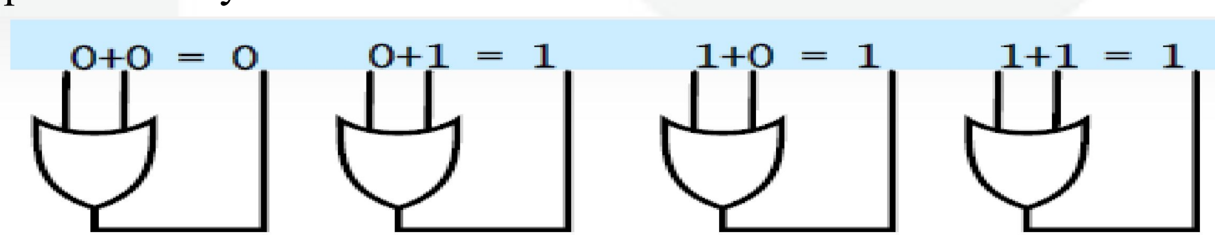
CHAPTER-2

Minimization Techniques

2.1 Boolean Algebra

- **Boolean Algebra:** In 1854 Logical algebra was published by **George Boole** known today as "Boolean Algebra". It's a convenient way and systematic way of expressing and analyzing the operation of logic circuits. 1938: **Claude Shannon** was the first to apply Boole's work to the analysis and design of logic circuits.
- **Boolean Operations & Expressions:**
    - Variable: A symbol used to represent a logical quantity.
    - Complement: The inverse of a variable and is indicated by a bar over the variable.
    - Literal: A variable or the complement of a variable.
- **Boolean Addition:**

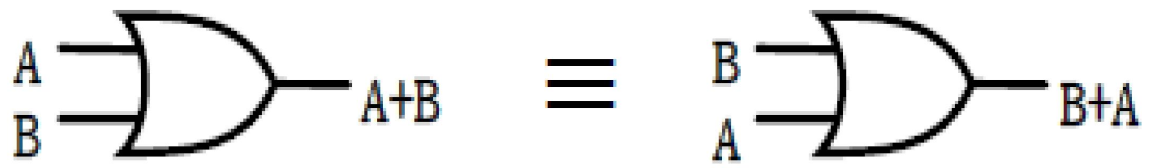
Boolean addition is equivalent to the OR operation.
A sum term is equal to 1 when one or more of the literals in the term are 1.
A sum term is equal to 0 only if each of the literals is 0.



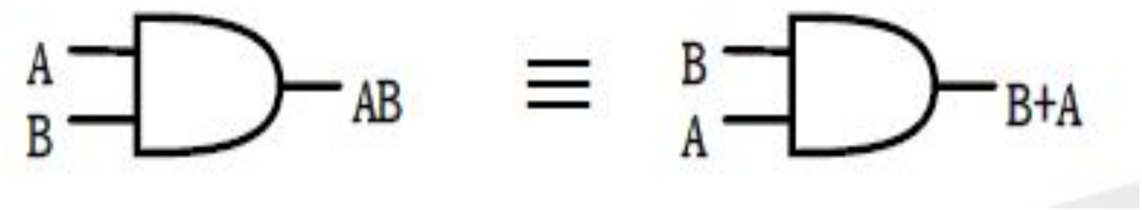
2.2 Laws & Rules of Boolean Algebra

Commutative Laws: The commutative law of addition for two variables is written as:

A+B = B+A.



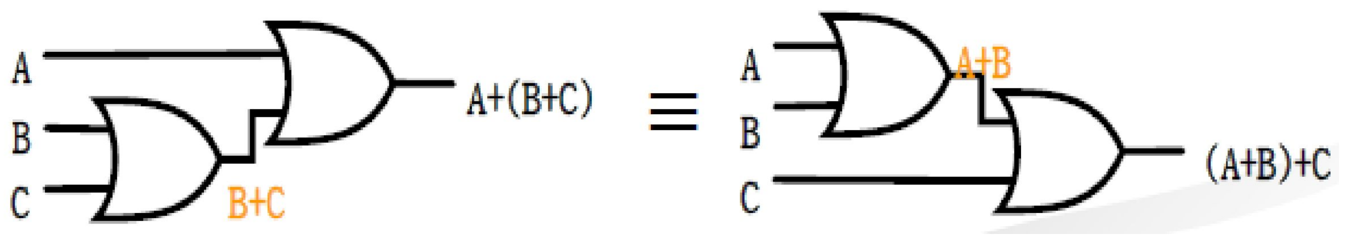
▪The commutative law of multiplication for two variables is written as: AB = BA.



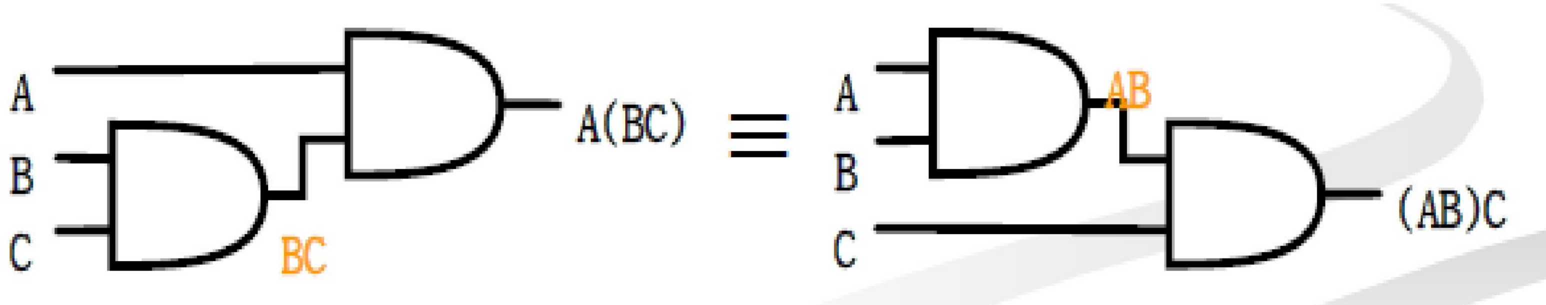
.

Associative Laws: The associative law of addition for 3 variables is written as:

A+(B+C) = (A+B)+C

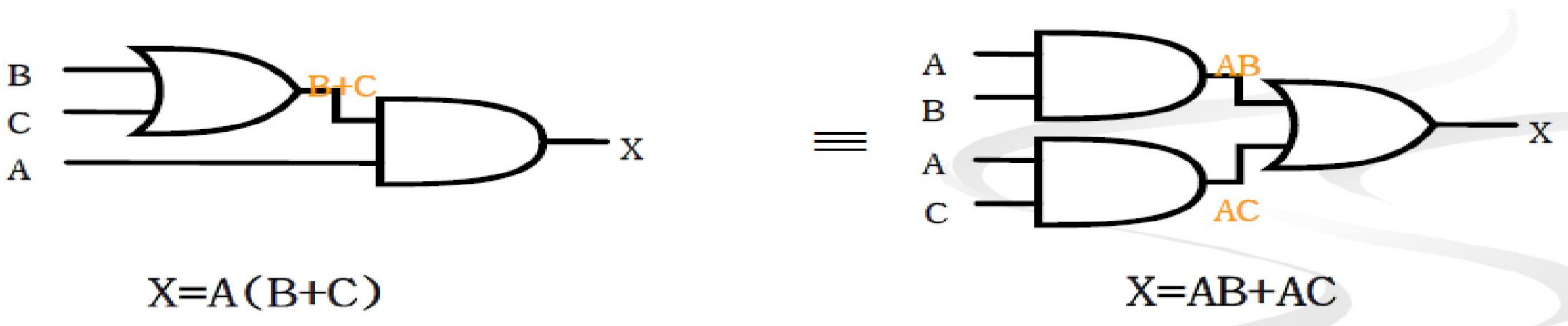


The associative law of multiplication for 3 variables is written as: A(BC) = (AB)C



Distributive Laws:

The distributive law is written for 3 variables as follows: A(B+C) = AB + AC



2.3 Basic Boolean Identities

- As with algebra, there will be Boolean operations that we will want to simplify We apply the following Boolean identities to help

Identity Name	AND Form	OR Form
Identity Law	$1x = x$	$0+x = x$
Null (or Dominance) Law	$0x = 0$	$1+x = 1$
Idempotent Law	$xx = x$	$x+x = x$
Inverse Law	$x\overline{x} = 0$	$x+\overline{x} = 1$
Commutative Law	$xy = yx$	$x+y = y+x$
Associative Law	$(xy)z = x(yz)$	$(x+y)+z = x+(y+z)$
Distributive Law	$x+yz = (x+y)(x+z)$	$x(y+z) = xy+xz$
Absorption Law	$x(x+y) = x$	$x+xy = x$
DeMorgan's Law	$(\overline{xy}) = \overline{x}+\overline{y}$	$(\overline{x+y}) = \overline{x}\overline{y}$
Double Complement Law	$\overline{\overline{x}} = x$	

2.4 Rules of Boolean Algebra

- As with algebra, there will be Boolean operations that we will want to simplify We apply the following Boolean identities to help

$$1. A + 0 = A$$

$$2. A + 1 = 1$$

$$3. A \bullet 0 = 0$$

$$4. A \bullet 1 = A$$

$$5. A + A = A$$

$$6. A + \overline{A} = 1$$

$$7. A \bullet A = A$$

$$8. A \bullet \overline{A} = 0$$

$$9. \overline{\overline{A}} = A$$

$$10. A + AB = A$$

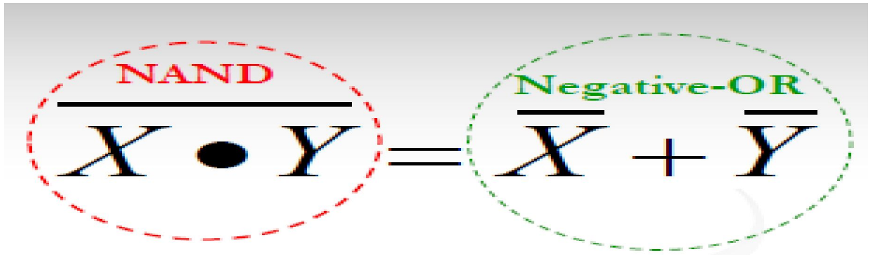
$$11. A + \overline{A}B = A + B$$

$$12. (A + B)(A + C) = A + BC$$

2.5 DeMorgan's Theorems

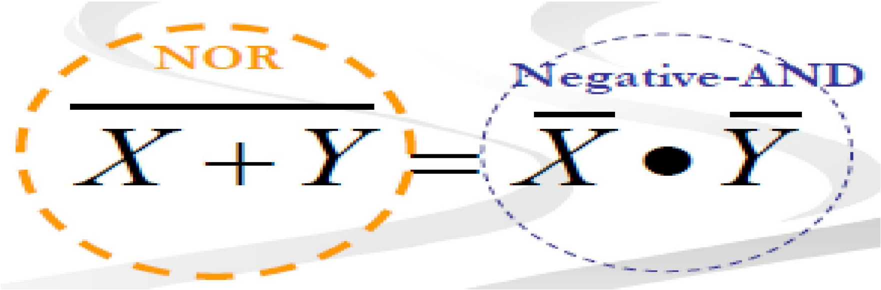
- DeMorgan's theorems provide mathematical verification of:
    1. The equivalency of the NAND and negative-OR gates.

The complement of two or more ANDed variables is equivalent to the OR of the complements of the individual variables.



$$\overline{X \bullet Y} = \overline{X} + \overline{Y}$$

2. The equivalency of the NOR and negative-AND gates.



$$\overline{X + Y} = \overline{X} \bullet \overline{Y}$$

The complement of two or more ORed variables is equivalent to the AND of the complements of the individual variables.

2.6 Duality theorem

- It states that if you have a true **Boolean** equation then the dual of this statement is true.
- The dual of a boolean statement is found by replacing the statement's symbols with their counterparts. This means a "0" becomes a "1", "1" becomes a "0", "+" becomes a "." and "." becomes a "+".

Given Expression	Dual
0 = 1	1 = 0
0.1 = 0	1 + 0 = 1
A.0 = 0	A + 1 = 1
A.A = 0	A + A = 1
A.(B.C) = (A.B).C	A+(B+C) = (A+B) + C

2.7 Minterm

- If a boolean function contains n variables than a product term which contains all the variables once in either complemented or uncomplemented form is called minterm.
- In the minterm main property of each minterm is that it will have value one only.
- For n variable expression the no. of minterms are 2n like- if n=2 than minterms are 4, n=3 than minterms are 8.
- All the minterms are can be represented by m0, m1, m2………..

Example:

X	Y	X . Y	Minterms	
0	0	0	X'Y'	mo
0	1	0	X'Y	m1
1	0	0	XY'	m2
1	1	1	XY	m3

2.8 Maxterm

- If a Boolean function contains n variables and the sum term contains all the possible combinations of n variables than the sum term is known as **maxterm**.
- This term is opposite of minterm because the value of each maxterm is 0. The maxterm can be represented as M0, M1,………..
- The main property of each maxterm is that it has value 0 only for one combination of n input variables.

Example:

X	Y	X + Y	Maxterms	
0	0	0	X+Y	M0
0	1	1	X+Y’	M1
1	0	1	X’+Y	M2
1	1	1	X’+Y’	M3

2.9 Sum of Products (SOP)

- When we perform the sum of logically multiplied inputs than the resultant expression is called sum of product. The canonical or standard SOP is the sum of minterms.

Example: Y(A,B,C)= m1 + m4 + m5 + m7

A	B	C	Minterm	
0	0	0	A'B'C'	m0
0	0	1	A'B'C	m1
0	1	0	A'BC'	m2
0	1	1	A'BC	m3
1	0	0	AB'C'	m4
1	0	1	AB'C	m5
1	1	0	ABC'	m6
1	1	1	ABC	m7

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

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

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

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

2.10 Product of Sums (POS)

- The product of sum can be difined as the logical product of the maxterm for which it has the value 0.
- **Example:**

$$f(A,B,C) = (A'+B'+C) \,.\, (A+C) \,.\, (A'+B')$$

$$= (A'+B'+C) \,.\, (A+C+BB') \,.\, (A'+B'+CC')$$

$$= (A'+B'+C) \,.\, ((A+C)+BB') \,.\, ((A'+B')+CC')$$

By appling: $x+y \,.\, z = (x+y) \,.\, (x+z)$

$$= (A'+B'+C) \,.\, (A+C+B) \,.\, (A+C+B') \,.\, (A'+B'+C) \,.\, (A'+B'+C')$$

$$= M6, M0, M2, M6, M7$$

$$= M0, M2. M6, M7$$

2.11 Product of Sums (POS)

- The product of sum can be difined as the logical product of the maxterm for which it has the value 0.
- **Example:**

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

$$= (A'+B'+C) . (A+C+BB') . (A'+B'+CC')$$

$$= (A'+B'+C) . ((A+C)+BB') . ((A'+B')+CC')$$

By appling: $x+y . z= (x+y) . (x+z)$

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

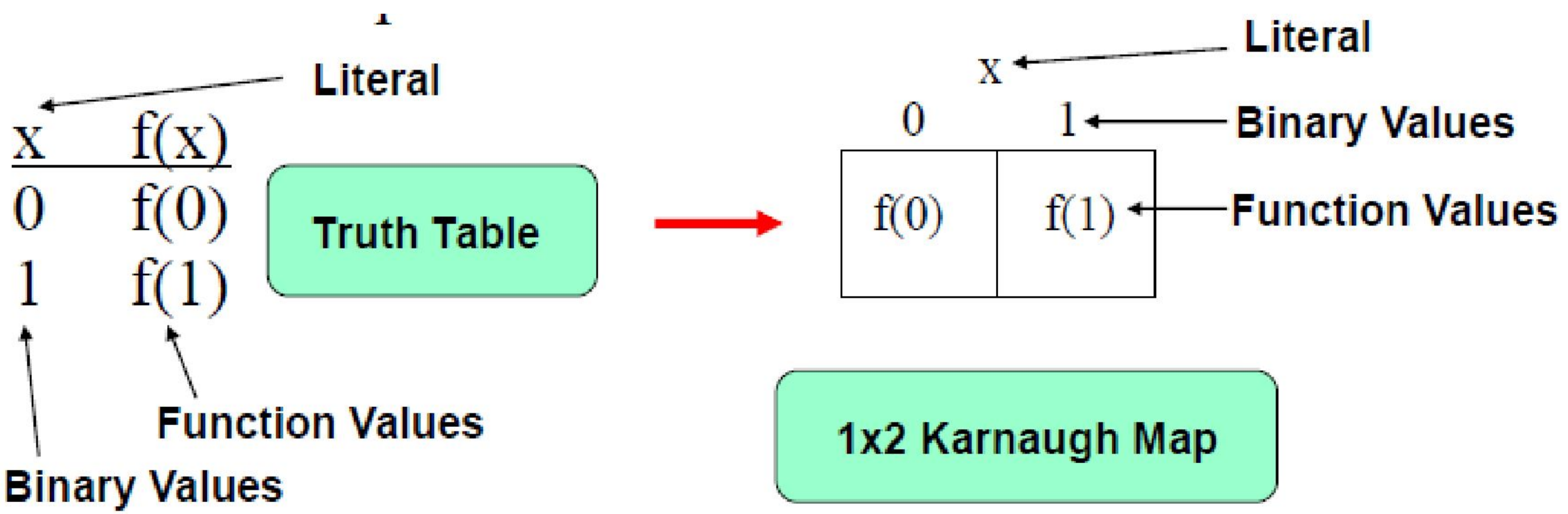
$$= M6, M0, M2, M6, M7$$

$$= M0, M2. M6, M7$$

2.12 Karnough Map (K- map)

- Some times the Boolean theorems and laws makes the simplification logics of Boolean functions more complex. Than we have to use the k-map techniques.
- It is a graphical method which is used to simplify a Boolean function or to convert a truth table into its equivalent logic circuit.
- The k-map is designed by squares where each square represents a minterm or maxterm.
- We will determine these techniques by studying examples in order to establish the rules for map manipulation.

1 variable map:



2 variable map:

x	y	f(x,y)
0	0	f(0,0)
0	1	f(0,1)
1	0	f(1,0)
1	1	f(1,1)

Truth Table

x \ y	0	1
0	f(0,0)	f(0,1)
1	f(1,0)	f(1,1)

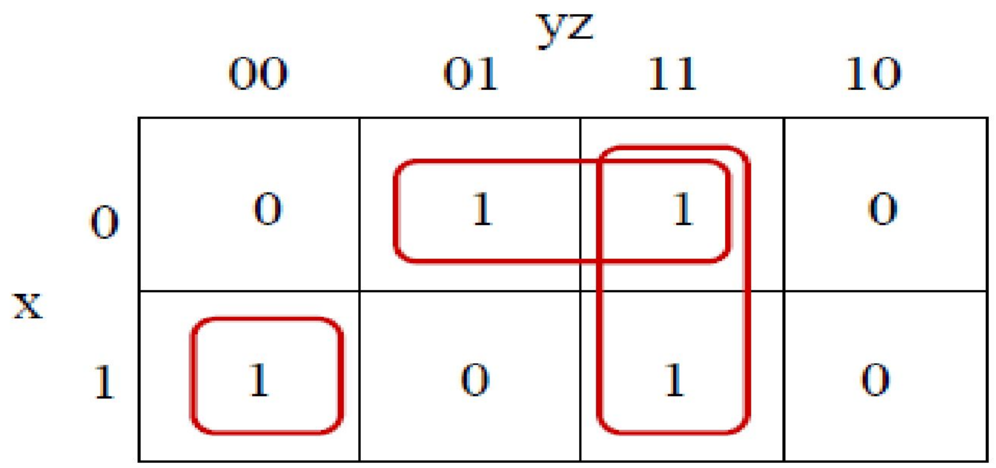
2x2 Karnaugh Map

3 variable map:

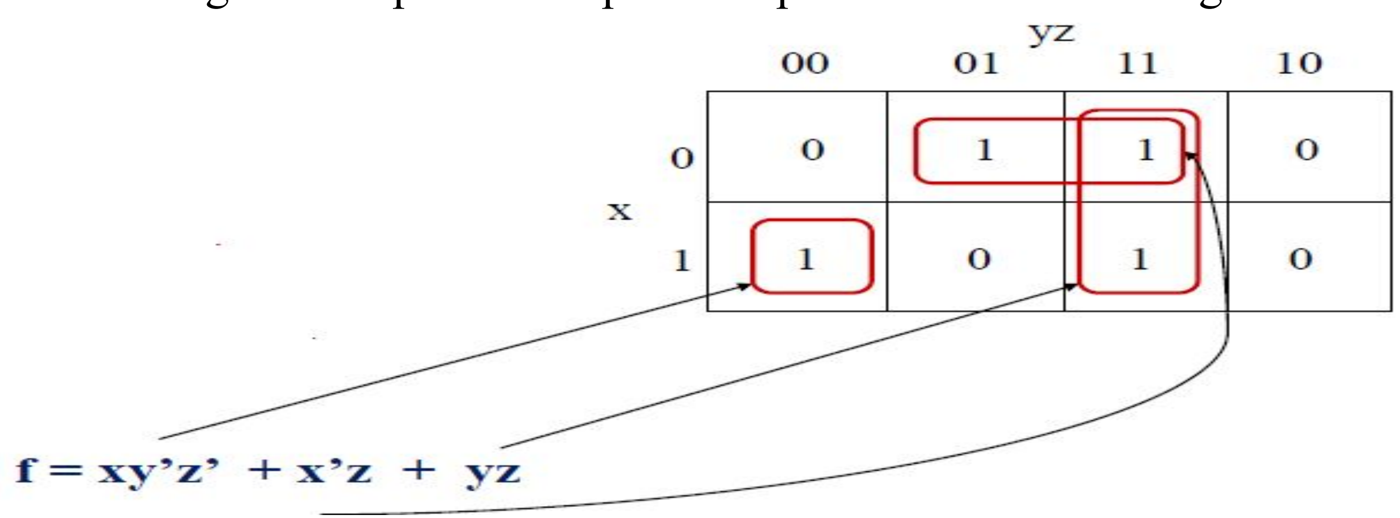
x	y	z	f(x)
0	0	0	f(000)
0	0	1	f(001)
0	1	0	f(010)
0	1	1	f(011)
1	0	0	f(100)
1	0	1	f(101)
1	1	0	f(110)
1	1	1	f(111)

x \ yz	00	01	11	10
0	f(000)	f(001)	f(011)	f(010)
1	f(100)	f(101)	f(111)	f(110)

2x4 Karnaugh Map



- Circle all 1 entries that, taken together, form a rectangle.
- Start with the largest rectangle, then proceed to smaller rectangle.
- Generally speaking, there will be more than one independent rectangle. each reflecting a different prime implicant.
- Each rectangle corresponds to a prime implicant term. Gathering all terms in SOP form,



4 variable map:

w	x	y	z	f(w,x,y,z)
0	0	0	0	f(0000)
0	0	0	1	f(0001)
0	0	1	0	f(0010)
0	0	1	1	f(0011)
0	1	0	0	f(0100)
0	1	0	1	f(0101)
0	1	1	0	f(0110)
0	1	1	1	f(0111)
1	0	0	0	f(1000)
1	0	0	1	f(1001)
1	0	1	0	f(1010)
1	0	1	1	f(1011)
1	1	0	0	f(1100)
1	1	0	1	f(1101)
1	1	1	0	f(1110)
1	1	1	1	f(1111)

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

4 variable map:

w	x	y	z	f(w,x,y,z)
0	0	0	0	f(0000)
0	0	0	1	f(0001)
0	0	1	0	f(0010)
0	0	1	1	f(0011)
0	1	0	0	f(0100)
0				
0				
0				
1				
1				
1				
1	0	1	1	f(1011)
1	1	0	0	f(1100)
1	1	0	1	f(1101)
1	1	1	0	f(1110)
1	1	1	1	f(1111)

Note the way that both the row and column indices change by <u>only</u> 1 bit at a time.

wx \ yz	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

WRAP-AROUND!

This implies that two rows, or columns, whose indices differ by only 1 bit value, are adjacent.

2.13 Minimization of logic functions using K-map

Example: Minimize the below boolean equation using K-map

$$F = \overline{A}.\overline{B}.\overline{C}.\overline{D} + \overline{A}.B.\overline{C}.D + \overline{A}.B.\overline{C}.\overline{D} + A.\overline{B}.\overline{C}.\overline{D} + \overline{A}.\overline{B}.C.\overline{D} + A.\overline{B}.C.\overline{D} + \overline{A}.\overline{B}.\overline{C}.D$$

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

$$F = \overline{B}.\overline{D} + \overline{A}.\overline{C}$$

2.14 Don’t care conditions using K-map

- When constructing the terms in the simplification procedure, we can choose to either cover or not cover the don’t care conditions.
- To distinguish the don’t care conditions from 1’s and 0’s , an **X** will be used.
- Don’t care conditions are part of function specification. They can be used for both sum-of-product and product-of-sum forms of functions.

$$f = \sum m(.....) + \sum d(.....) \qquad f = \prod M(.....) \prod D(.....)$$

Example: Minimize the below boolean equation using K-map:

$$F = \Sigma m(1, 3, 7) + \Sigma d(0, 5)$$

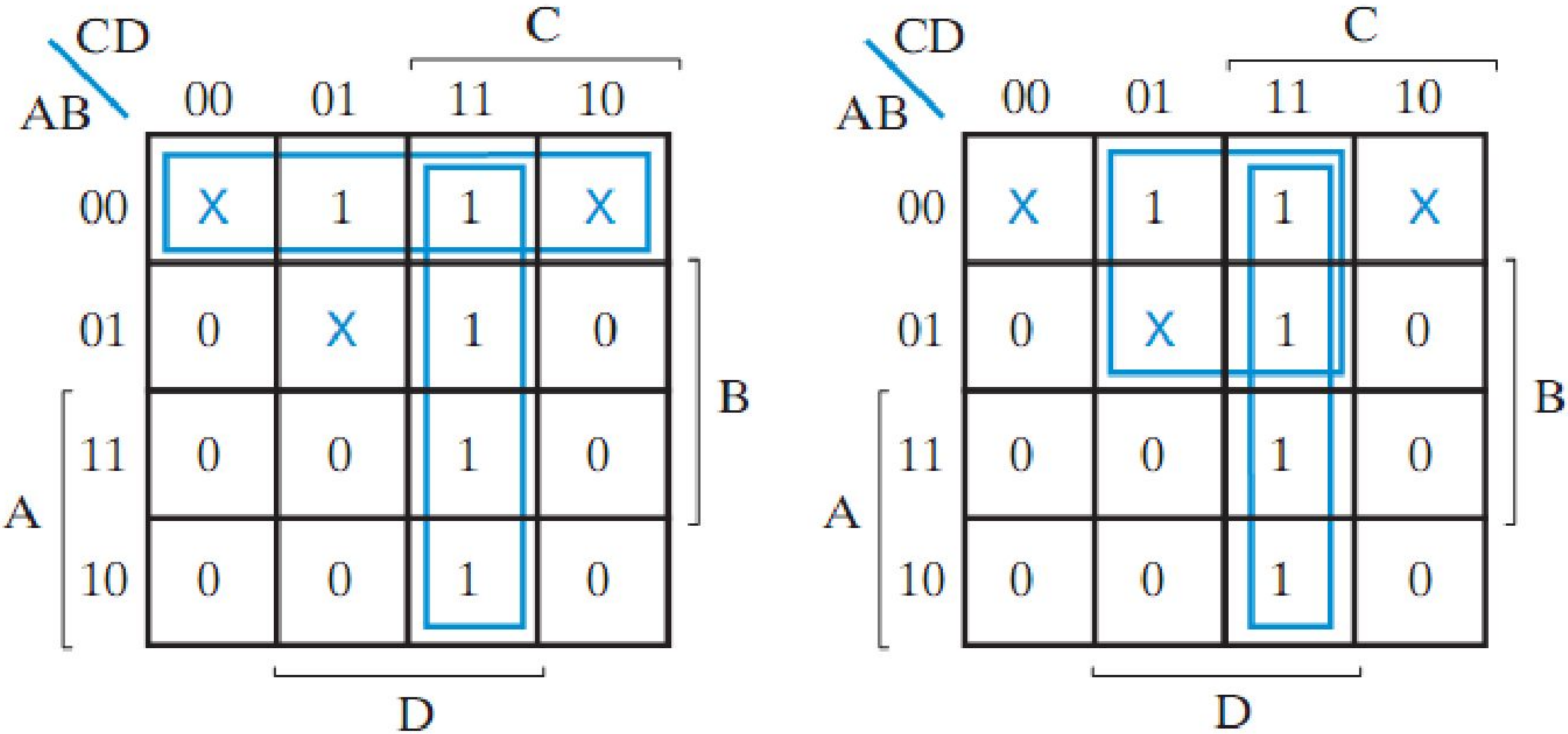
- Circle the x’s that help get bigger groups of 1’s (or 0’s if POS).

Reduced form : F = C

A \ BC	00	01	11	10
0	0 X	1 1	3 1	2
1	4	5 X	7 X 1	6

Example: Minimize the below boolean equation using K-map:

$$F(A, B, C, D) = \Sigma\, m(1, 3, 7, 11, 15) + \Sigma\, d(0, 2, 5)$$



(a) $F = CD + \overline{A}\,\overline{B}$　　(b) $F = CD + \overline{A}D$

- Here two solution, both are right but second solution is right for circuit design.

2.15 Quine-McCluskey method of minimization (Tabular Method)

- Compute all prime implicants.
- Find a minimum expression for Boolean functions.
- No visualization of prime implicants.
- Can be programmed and implemented in a computer.

Example: Minimize the below boolean equation:

$$F(W,X,Y,Z) = \sum m(0,3,5,6,7,10,12,13) + \sum d(2,9,15)$$

Step 1 : Divide all the minterms(and don't cares) of a function into groups

For Minterms:

Minterm ID	W	X	Y	Z
0	0	0	0	0
3	0	0	1	1
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
10	1	0	1	0
12	1	1	0	0
13	1	1	0	1

For don't cares:

Minterm ID	W	X	Y	Z
2	0	0	1	0
9	1	0	0	1
15	1	1	1	1

Groups	Minterm ID	W	X	Y	Z	Merge Mark
G0	0	0	0	0	0	
G1	2	0	0	1	0	
G2	3	0	0	1	1	
	5	0	1	0	1	
	6	0	1	1	0	
	9	1	0	0	1	
	10	1	0	1	0	
	12	1	1	0	0	
G3	7	0	1	1	1	
	13	1	1	0	1	
G4	15	1	1	1	1	

Step 2: Merge minterms from adjacent groups to form a new implicant table

Groups	Minterm ID	W	X	Y	Z	Merge Mark
G0	0	0	0	0	0	✓
G1	2	0	0	1	0	✓
G2	3	0	0	1	1	✓
	5	0	1	0	1	✓
	6	0	1	1	0	✓
	9	1	0	0	1	✓
	10	1	0	1	0	✓
	12	1	1	0	0	✓
G3	7	0	1	1	1	✓
	13	1	1	0	1	✓
G4	15	1	1	1	1	✓

Groups	Minterm ID	W	X	Y	Z
G0'	0, 2	0	0	d	0
G1'	2, 3	0	0	1	d
	2, 6	0	d	1	0
	2, 10	d	0	1	0
G2'	3, 7	0	d	1	1
	5, 7	0	1	d	1
	6, 7	0	1	1	d
	5, 13	d	1	0	1
	9, 13	1	d	0	1
	12, 13	1	1	0	d
G3'	7, 15	d	1	1	1
	13, 15	1	1	d	1

Step 3: Repeat step 2 until no more merging is possible

Groups	Minterm ID	W	X	Y	Z	Merge Mark
G0'	0, 2	0	0	d	0	
G1'	2, 3	0	0	1	d	✓
	2, 6	0	d	1	0	✓
	2, 10	d	0	1	0	
G2'	3, 7	0	d	1	1	✓
	5, 7	0	1	d	1	✓
	6, 7	0	1	1	d	✓
	5, 13	d	1	0	1	✓
	9, 13	1	d	0	1	
	12, 13	1	1	0	d	
G3'	7, 15	d	1	1	1	✓
	13, 15	1	1	d	1	✓

Groups	Minterm ID	W	X	Y	Z
G1''	2, 3, 6, 7	0	d	1	d
	2, 6, 3, 7	0	d	1	d
G2''	5, 7, 13, 15	d	1	d	1
	5, 7, 13, 15	d	1	d	1

Step 3: Repeat step 2 until no more merging is possible

Groups	Minterm ID	W	X	Y	Z	Merge Mark
G0"	0, 2	0	0	d	0	
G1"	2, 3, 6, 7	0	d	1	d	
	2, 10	d	0	1	0	
G2"	5, 7, 13, 15	d	1	d	1	
	9, 13	1	d	0	1	
	12, 13	1	1	0	d	

No more merging possible!

Step 4: Put all prime implicants in a cover table (don't cares excluded)

Minterm ID	$\overline{W}\,\overline{X}\,\overline{Z}$	$\overline{W}Y$	$\overline{X}Y\overline{Z}$	XZ	$WX\overline{Y}$	$W\overline{Y}Z$
0	1					
3		1				
5				1		
6		1				
7		1		1		
10			1			
12					1	
13				1	1	1

Step 5: Identify essential minterms, and hence essential prime implicants

Minterm ID	$\overline{W}\ \overline{X}\ \overline{Z}$	$\overline{W}Y$	$\overline{X}Y\overline{Z}$	XZ	$WX\overline{Y}$	$W\overline{Y}Z$
0	$\boxed{1}$					
3		$\boxed{1}$				
5				$\boxed{1}$		
6		$\boxed{1}$				
7		1		1		
10			$\boxed{1}$			
12					$\boxed{1}$	
13				1	1	1

E.M.T E.P.I

Step 6: Add prime implicants to the minimum expression of F until all minterms of Fare covered (Already covered)

Minterm ID	$\overline{W}\,\overline{X}\,\overline{Z}$	$\overline{W}Y$	$\overline{X}Y\overline{Z}$	XZ	$WX\overline{Y}$	$W\overline{Y}Z$
0	$\boxed{1}$					
3		$\boxed{1}$				
5				$\boxed{1}$		
6		$\boxed{1}$				
7		1		1		
10			$\boxed{1}$			
12					$\boxed{1}$	
13				1	1	1

$\square$ E.M.T $\square$ E.P.I

So after simplification through QM method, a minimum expression for F(W, X, Y, Z) is:

$$F(W,X,Y,Z) = \overline{W}\,\overline{X}\,\overline{Z} + \overline{W}Y + \overline{X}Y\overline{Z} + XZ + WX\overline{Y}$$

2.16 Variable Entered Maps:

- Due to the difficulty in managing K map exceeding 4 variables like 5 or 6 variables we have the technique called map entered variables[VEM].
- As we all know for n variables a K map has 2n variables.To allow a smaller map to handle a larger number of variables that is to reduce the number of squares the concept of variable entered mapping was introduced.
- This is done by choosing one variable as map entered variable and thus including it in the k map along with the zeros, ones and the don't care conditions.

Example: Minimize the below boolean equation using K-map:

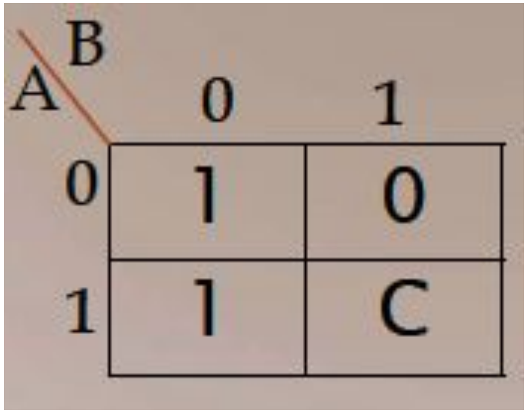
$$f(A, B, C) = \sum m_i(0,1,4,5,7)$$

A	B	C	Y	
0	0	0	1	1
0	0	1	1	
0	1	0	0	0
0	1	1	0	
1	0	0	1	1
1	0	1	1	
1	1	0	0	C
1	1	1	1	

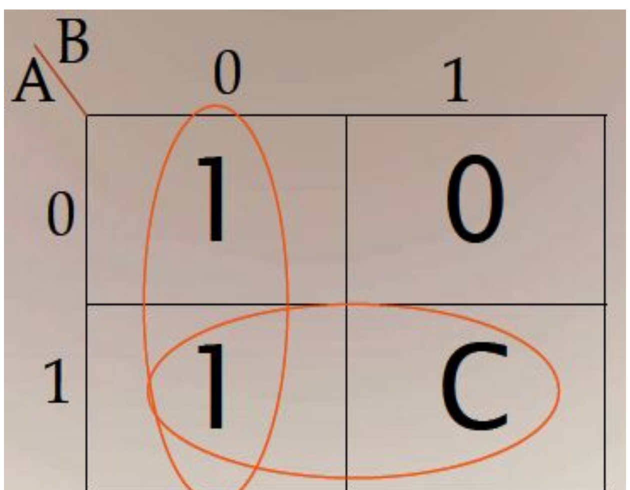
A	B	Y
0	0	1
0	1	0
1	0	1
1	1	C

Truth table considering C as MEV is

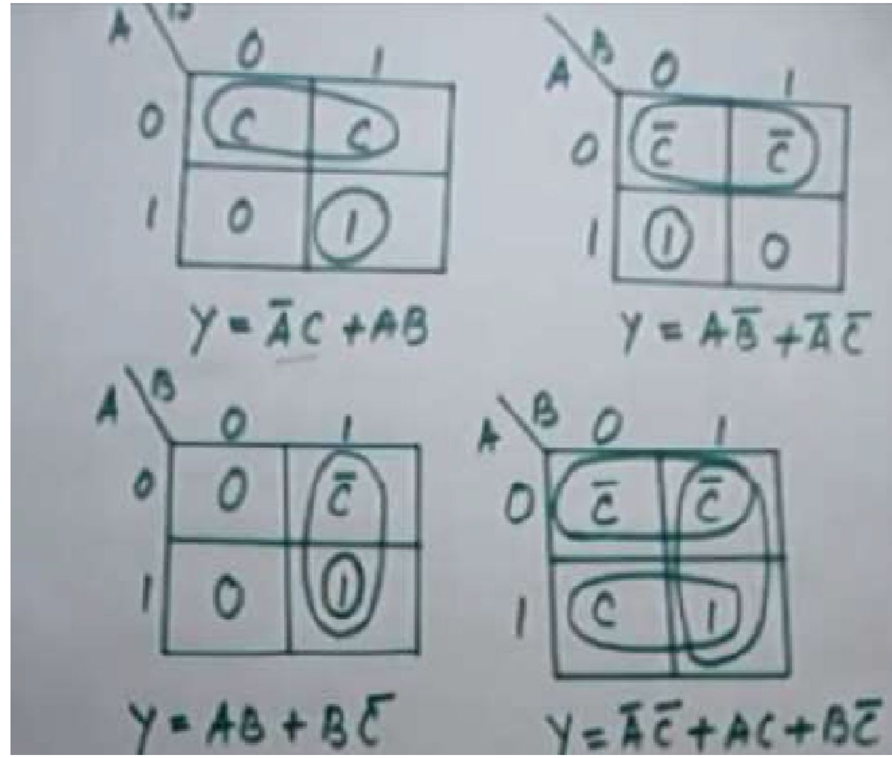
MEV K-map is



So grouping in our MEV K-map is as shown and equation is



Grouping in vem



you can group C and C’s, 1 and C’s, 1 and C’s but your cannot group C and C′s.

$$Y = \bar{B} + \bar{A}\bar{C}$$

Verification using K-map

A \ BC	00	01	11	10
0	1	1	0	1
1	1	1	0	0

$Y = \bar{B} + \bar{A}\bar{C}$, verified!

2.17 Realizing Logic Function with Gates:

Implement the below Boolean equation using logic gates:

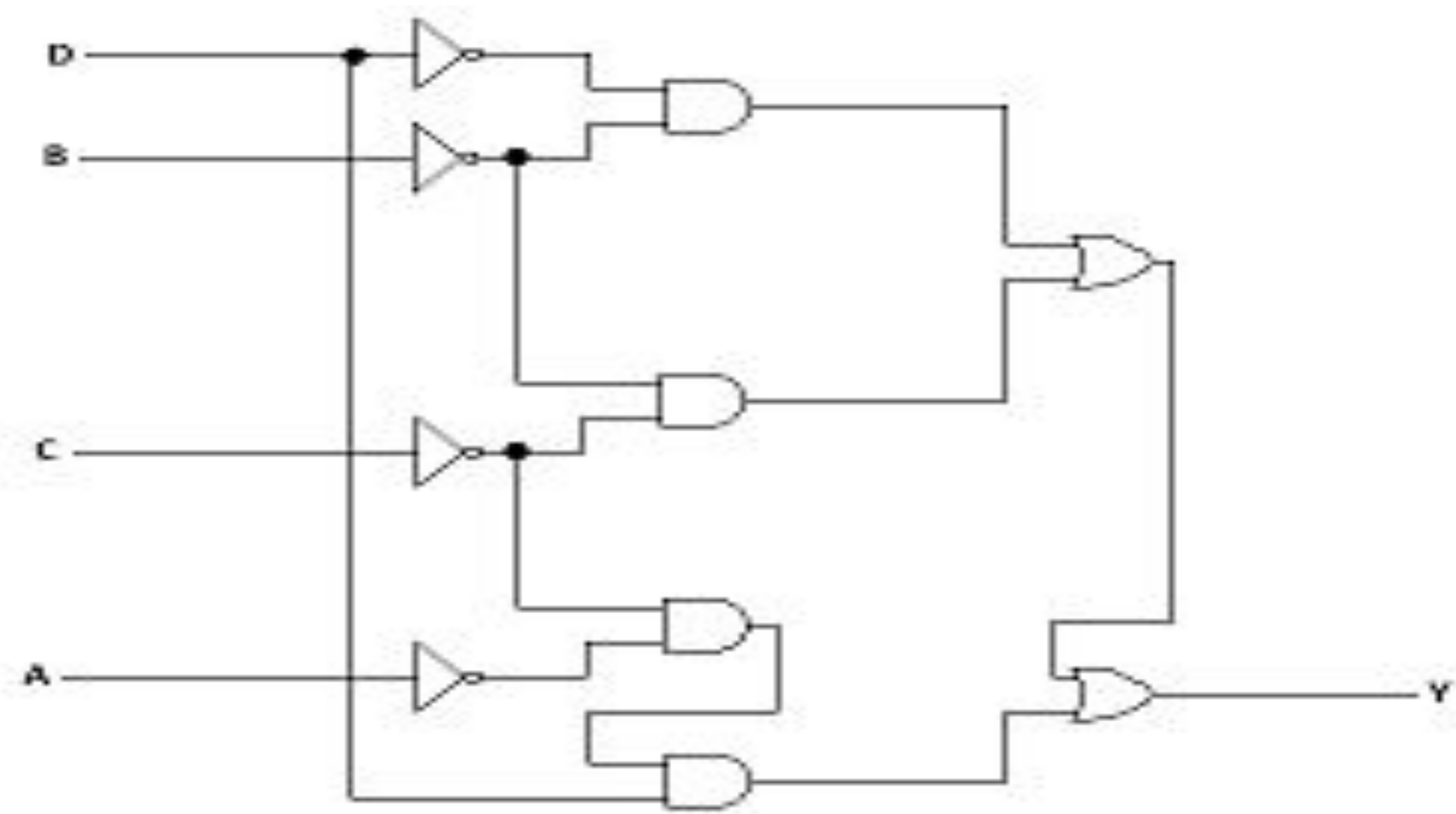
F(A,B,C,D)=A’B’C’D’+A’B’C’D+A’BC’D+AB’C’D’+AB’C’D+A’B’CD’+AB’CD’

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

F(A,B,C)=B’C’+B’D’+A’C’D

So, we can see that function reduces to simple form. Now ,we implement this derived function using gates which gives o/p exactly same as real function.

K-map Simplification Circuit:




Thank
You