# Wachemo University

## Department of Electrical and Computer Engineering

## Digital Logic Design(DLD)

**Chapter 3** Boolean algebra and logic simplification

**Pre. Getahun.T**

Dec.2023

# *3.Boolean algebra and logic simplification*

- One of the primary requirements when dealing with digital circuits is to find ways to make them as simple as possible. This constantly requires that complex logical expressions be reduced to simpler expressions that nevertheless produce the same results under all possible conditions.
- The simpler expression can then be implemented with a smaller, simpler circuit, which in turn saves the price of the unnecessary gates, reduces the number of gates needed, and reduces the power and the amount of space required by those gates.
- One tool to reduce logical expressions is the mathematics of logical expressions, introduced by George Boole in 1854 and known today as *Boolean Algebra. The rules of Boolean Algebra are simple and straightforward and can be applied to any logical expression. The resulting reduced expression can then be readily tested with a Truth Table, to verify that the reduction was valid.*
- Boolean algebra is an algebraic structure defined on a set of elements B, together with two binary operators(+, **.) provided the following postulates are satisfied.**

# Cont…

1. Closure concerning operator + and Closure concerning operator**.**
2. An identity element with respect to + designated by 0: X+0= 0+X=X

An identity element concerning. designated by 1: X.1= 1.X=X

3. Commutative concerning +: X=Y=Y+X

Commutative concerning**.: X.Y=Y.X**

4. . distributive over +: X.(Y+Z)=X.Y+X.Z

+ distributive over .: X+(Y.Z)=(X+Y).(X+Z)

5. For every element x belonging to B, there exists an element x′ or called the complement of x such that x. x′=0 and x+ x′=1 *x*
6. There exists at least two elements x,y belonging to B such that x≠y

The two-valued Boolean algebra is defined on a set B={0,1} with two binary operators + and**.**

| X | y | x.y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| X | Y | x+y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| x | x′ |
|---|---|
| 0 | 1 |
| 1 | 0 |

# *Cont…*

- Closure. From the tables, the result of each operation is either 0 or 1, and 1,0 belongs to B
- Identity. From the truth table, we see that 0 is the identity element for + and 1 is the identity element for.
- Commutative law is obvious from the symmetry of binary operators table.

Distributive Law. x.(y+z)=x.y+x.z

| x | y | z | y+z | x.(y+z) | x.y | x.z | x.y+x.z |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Distributive of + over . can be shown as in the truth table above

From the complement table we can see that x+ x′=1 i.e 1+0=1 and x. x′=0 i.e 1.0=0

Principle of duality of Boolean algebra

The principle of duality state that every algebraic expression which can be deduced from the postulates of Boolean algebra remains valid if the operators and the identity elements are interchanged. This mean that the dual of an expression is obtained changing every AND(.) to OR(+), every OR(+) to AND(.) and all 1's to 0's and vice-versa

# Cont…

## Laws of Boolean Algebra

**Postulate 2 :**

(a) $0 + A = A$ (b) $1.A = A$

**Postulate 5 :**

(a) $A + A' = 1$ (b) $A.\ A' = 0$

**Theorem1 : Identity Law**

(a) $A + A = A$ (b) $A\ A = A$

**Theorem2**

(a) $1 + A = 1$ (b) $0.\ A = 0$

**Theorem3: involution**

$A'' = A$

# *Cont…*

**Postulate 3 : Commutative Law**

(a) $A + B = B + A$ (b) $A\,B = B\,A$

**Theorem4: Associate Law**

(a) $(A + B) + C = A + (B + C)$ (b) $(A\,B)\,C = A\,(B\,C)$

**Postulate4: Distributive Law**

(a) $A\,(B + C) = A\,B + A\,C$ (b) $A + (B\,C) = (A + B)\,(A + C)$

**Theorem5 : De Morgan's Theorem**

(a) $(A+B)' = A'B'$ (b) $(AB)' = A' + B'$

**Theorem6 : Absorption**

(a) $A + A\,B = A$ (b) $A\,(A + B) = A$

# Basic Theorems

- Table 3.1 lists six theorems of Boolean algebra and four of its postulates. The notation
- is simplified by omitting the binary operator whenever doing so does not lead to
- confusion. The theorems and postulates listed are the most basic relationships in Boolean algebra.

*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) | $x(x + y) = x$ |

# *Cont…*

- The postulates are basic axioms of the algebraic structure and need no proof. The theorems must be proven from the postulates. Proofs of the theorems with one variable are presented next. At the right is listed the number of the postulate which justifies that particular step of the proof.

**THEOREM 1(a):** $x + x = x$.

| Statement | Justification |
|---|---|
| $x + x = (x + x) \cdot 1$ | postulate 2(b) |
| $= (x + x)(x + x')$ | 5(a) |
| $= x + xx'$ | 4(b) |
| $= x + 0$ | 5(b) |
| $= x$ | 2(a) |

**THEOREM 1(b):** $x \cdot x = x$.

| Statement | Justification |
|---|---|
| $x \cdot x = xx + 0$ | postulate 2(a) |
| $= xx + xx'$ | 5(b) |
| $= x(x + x')$ | 4(a) |
| $= x \cdot 1$ | 5(a) |
| $= x$ | 2(b) |

# *Cont…*

**THEOREM 2(a):** $x + 1 = 1$.

| Statement | Justification |
|---|---|
| $x + 1 = 1 \cdot (x + 1)$ | postulate 2(b) |
| $= (x + x')(x + 1)$ | 5(a) |
| $= x + x' \cdot 1$ | 4(b) |
| $= x + x'$ | 2(b) |
| $= 1$ | 5(a) |

**THEOREM 2(b):** $x \cdot 0 = 0$ by duality.

**THEOREM 3:** $(x')' = x$. From postulate 5, we have $x + x' = 1$ and $x \cdot x' = 0$, which together define the complement of $x$. The complement of $x'$ is $x$ and is also $(x')'$.

# *Cont…*

Therefore, since the complement is unique, we have $(x')' = x$. The theorems involving two or three variables may be proven algebraically from the postulates and the theorems that have already been proven. Take, for example, the absorption theorem:

**THEOREM 6(a):** $x + xy = x$.

| Statement | Justification |
|---|---|
| $x + xy = x \cdot 1 + xy$ | postulate 2(b) |
| $= x(1 + y)$ | 4(a) |
| $= x(y + 1)$ | 3(a) |
| $= x \cdot 1$ | 2(a) |
| $= x$ | 2(b) |

**THEOREM 6(b):** $x(x + y) = x$ by duality.

# BOOLEAN FUNCTIONS

- Boolean algebra is an algebra that deals with binary variables and logic operations. A Boolean function described by an algebraic expression consists of binary variables, the constants 0 and 1, and the logic operation symbols. For a given value of the binary variables, the function can be equal to either 1 or 0.
- As an example, consider the Boolean function

$$F1 = x + yz$$

- The function *F1 is equal to 1 if x is equal to 1 or if both y and z are equal to 1. F1 is equal to* 0 otherwise. The complement operation dictates that when *y = 1, y = 0. Therefore,F1 = 1 if x = 1 or if y = 0 and z = 1. A Boolean function expresses the logical relationship* between binary variables and is evaluated by determining the binary value of the expression for all possible values of the variables.
- A Boolean function can be represented in a truth table. The number of rows in the truth table is 2*n, where n is the number of variables in the function. The binary combinations for* the truth table are obtained from the binary numbers by counting from 0 through 2*n - 1*.
- *Table 3.2 shows the truth table for the function F1. There are eight possible* binary combinations for assigning bits to the three variables *x, y, and z. The column* labeled *F1 contains either 0 or 1 for each of these combinations. The table shows that* the function is equal to 1 when *x = 1 or when yz = 01 and is equal to 0 otherwise.*
- A Boolean function can be transformed from an algebraic expression into a circuit diagram composed of logic gates connected to a particular structure. The logic-circuit diagram (also called a schematic) for *F1 is shown in Fig. 3.1 . There is an inverter for input y to generate its complement. There is an AND gate for the term yz and an OR gate*

# Cont…

Truth Tables for $F_1$ and $F_2$

| x | y | z | $F_1$ | $F_2$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Table 3.2

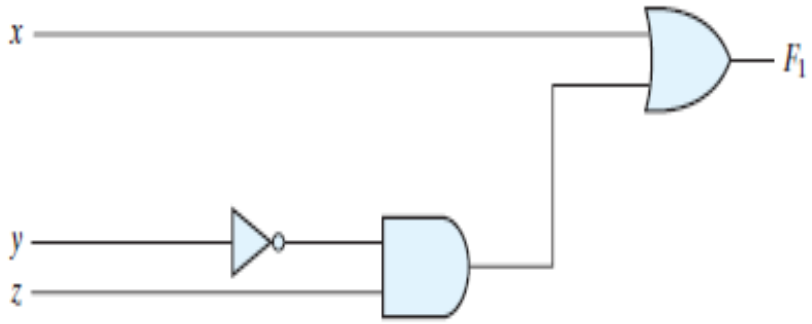



figure 3.1

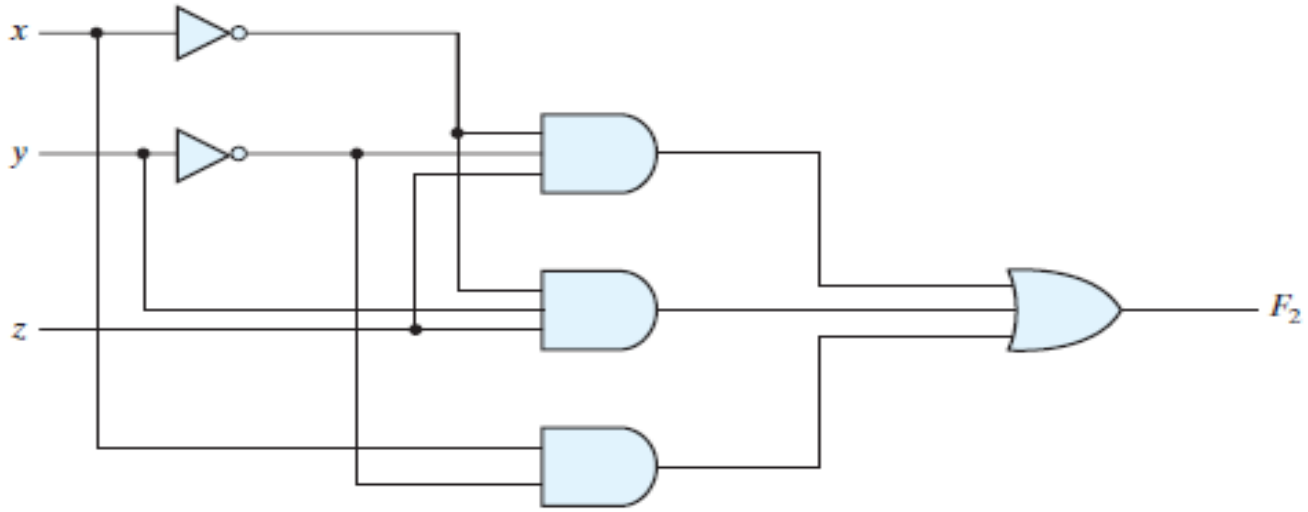
Example -1 Implement the following Boolean function ?

$$F_2 = x'y'z + x'yz + xy'$$

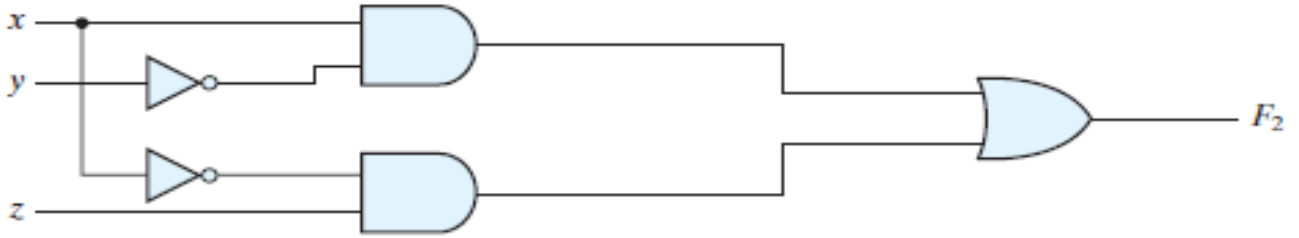
- Now consider the possible simplification of the function by applying some of the identities of Boolean algebra:

$$F_2 = x'y'z + x'yz + xy' = x'z(y' + y) + xy' = x'z + xy'$$

# *Cont…*




(a) $F_2 = x'y'z + x'yz + xy'$




(b) $F_2 = xy' + x'z$

figure-3.2 implementation Boolean function a) without simplification b) with simplification

# Algebraic Manipulation

- When a Boolean expression is implemented with logic gates, each term requires a gate and each variable within the term designates an input to the gate. We define a *literal to* be a single variable within a term, in complemented or uncomplemented form.
- The function of Fig. 3.2 (a) has three terms and eight literals, and the one in Fig. 3.2 (b) has two terms and four literals. By reducing the number of terms, the number of literals, or both in a Boolean expression, it is often possible to obtain a simpler circuit. The manipulation of Boolean algebra consists mostly of reducing an expression to obtain a simpler circuit.

Example -3 Simplify the following Boolean functions to a minimum number of literals.

1. $x(x' + y) = xx' + xy = 0 + xy = xy.$
2. $x + x'y = (x + x')(x + y) = 1(x + y) = x + y.$
3. $(x + y)(x + y') = x + xy + xy' + yy' = x(1 + y + y') = x.$
4. $$\begin{aligned} xy + x'z + yz &= xy + x'z + yz(x + x') \\ &= xy + x'z + xyz + x'yz \\ &= xy(1 + z) + x'z(1 + y) \\ &= xy + x'z. \end{aligned}$$
5. $(x + y)(x' + z)(y + z) = (x + y)(x' + z)$, by duality from function 4.

# Complement of a Function

- The complement of a function *F is F' and is obtained from an interchange of 0's for 1's* and 1's for 0's in the value of *F. The complement of a function may be derived algebraically* through DE Morgan's theorems, listed in Table 3.1 for two variables.
- DE Morgan's theorems can be extended to three or more variables. The three-variable form of the first DE Morgan's theorem is derived as follows, from postulates and theorems listed in Table 3.1 :

$$
\begin{aligned}
(A + B + C)' &= (A + x)' && \text{let } B + C = x \\
&= A'x' && \text{by theorem 5(a) (DeMorgan)} \\
&= A'(B + C)' && \text{substitute } B + C = x \\
&= A'(B'C') && \text{by theorem 5(a) (DeMorgan)} \\
&= A'B'C' && \text{by theorem 4(b) (associative)}
\end{aligned}
$$

DeMorgan's theorems for any number of variables resemble the two-variable case in form and can be derived by successive substitutions similar to the method used in the preceding derivation. These theorems can be generalized as follows:

$$(A + B + C + D + \cdots + F)' = A'B'C'D' \ldots F'$$

$$(ABCD \ldots F)' = A' + B' + C' + D' + \cdots + F'$$

The generalized form of DeMorgan's theorems states that the complement of a function is obtained by interchanging AND and OR operators and complementing each literal.

# *Cont…*

Example 3.3

Find the complement of the functions $F_1 = x'yz' + x'y'z$ and $F_2 = x(y'z' + yz)$. By applying DeMorgan's theorems as many times as necessary, the complements are obtained as follows:

$$F_1' = (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' = (x + y' + z)(x + y + z')$$

$$\begin{aligned} F_2' = [x(y'z' + yz)]' &= x' + (y'z' + yz)' = x' + (y'z')'(yz)' \\ &= x' + (y + z)(y' + z') \\ &= x' + yz' + y'z \end{aligned}$$

# CANONICAL AND STANDARD FORMS

## Minterms and Maxterms

A binary variable may appear either in its normal form (x) or in its complement form (x').

- Now consider two binary variables *x and y combined with an AND operation. Since each* variable may appear in either form, there are four possible combinations: *x'y,' x'y, xy',* and *xy. Each of these four AND terms is called a minterm, or a standard product.* In a similar manner, *n variables can be combined to form* $2^n$ *minterms.*
- Each minterm is obtained from an AND term of the *n variables, with each variable being* primed if the corresponding bit of the binary number is a 0 and unprimed if a 1. A symbol for each mintmaerm is also shown in the table and is of the form $m_j$, *where the subscript j* denotes the decimal equivalent of the binary number of the minterm designated.
- The $2^n$ *different minterms* may be determined by a method similar to the one shown in Table 3.3 for three variables.
- Similarly, *n variables forming an OR term, with each variable being primed* or unprimed, provide 2*n possible combinations, called max terms, or standard sums.*
- The eight maxterms for three variables, together with their symbolic designations, are listed in Table 3.3 .It is important to note that (1) each maxterm is obtained from an OR term of the *n variables, with* each variable being unprimed if the corresponding bit is a 0 and primed if a 1, and (2) each maxterm is the complement of its corresponding minterm and vice versa.

# Cont…

*Minterms and Maxterms for Three Binary Variables*

| | | | 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 | $xyz$ | $m_7$ | $x' + y' + z'$ | $M_7$ |

# *Cont…*

- **Boolean functions expressed as a sum of minterms or product of maxterms are said to be in *canonical form* .**

Example –Sum of minterms

Express the Boolean function $F = A + B'C$ as a sum of minterms. The function has three variables: $A$, $B$, and $C$. The first term $A$ is missing two variables; therefore,

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

This function is still missing one variable, so

$$\begin{aligned} A &= AB(C + C') + AB'(C + C') \\ &= ABC + ABC' + AB'C + AB'C' \end{aligned}$$

The second term $B'C$ is missing one variable; hence,

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

Combining all terms, we have

$$\begin{aligned} F &= A + B'C \\ &= ABC + ABC' + AB'C + AB'C' + A'B'C \end{aligned}$$

But $AB'C$ appears twice, and according to theorem 1 ($x + x = x$), it is possible to remove one of those occurrences. Rearranging the minterms in ascending order, we finally obtain

$$\begin{aligned} F &= A'B'C + AB'C + AB'C + ABC' + ABC \\ &= m_1 + m_4 + m_5 + m_6 + m_7 \end{aligned}$$

■

When a Boolean function is in its sum-of-minterms form, it is sometimes convenient to express the function in the following brief notation:

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

- **Product of Maxterms**
- *Each of the $2^{2n}$ functions of n binary variables can be also expressed as a product of max terms. To express a Boolean function as a product of max terms, it must first be* brought into the form of OR terms. This may be done by using the distributive law, $x + yz = (x + y)(x + z)$. *Then any missing variable x in each OR term is with xx'. The procedure is clarified in the following example.*

Express the Boolean function $F = xy + x'z$ as a product of maxterms. First, convert the function into OR terms by using the distributive law:

$$\begin{aligned} F &= xy + x'z = (xy + x')(xy + z) \\ &= (x + x')(y + x')(x + z)(y + z) \\ &= (x' + y)(x + z)(y + z) \end{aligned}$$

The function has three variables: $x$, $y$, and $z$. Each OR term is missing one variable; therefore,

$$\begin{aligned} x' + y &= x' + y + zz' = (x' + y + z)(x' + y + z') \\ x + z &= x + z + yy' = (x + y + z)(x + y' + z) \\ y + z &= y + z + xx' = (x + y + z)(x' + y + z) \end{aligned}$$

Combining all the terms and removing those which appear more than once, we finally obtain

$$\begin{aligned} F &= (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z') \\ &= M_0M_2M_4M_5 \end{aligned}$$

A convenient way to express this function is as follows:

$$F(x, y, z) = \Pi(0, 2, 4, 5)$$

The product symbol, Π, denotes the ANDing of maxterms; the numbers are the indices of the maxterms of the function.

# *Cont…*

**Conversion between Canonical Forms**

- The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function. This is because the original function is expressed by those minterms which make the function equal to 1, whereas its complement is a 1 for those minterms for which the function is a 0. As an example, consider the function

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

This function has a complement that can be expressed as

$$F'(A, B, C) = \Sigma(0, 2, 3) = m_0 + m_2 + m_3$$

Now, if we take the complement of $F'$ by DeMorgan's theorem, we obtain $F$ in a different form:

$$F = (m_0 + m_2 + m_3)' = m_0' \cdot m_2' \cdot m_3' = M_0 M_2 M_3 = \Pi(0, 2, 3)$$

The last conversion follows from the definition of minterms and maxterms as shown in Table 2.3. From the table, it is clear that the following relation holds:

$$m_j' = M_j$$

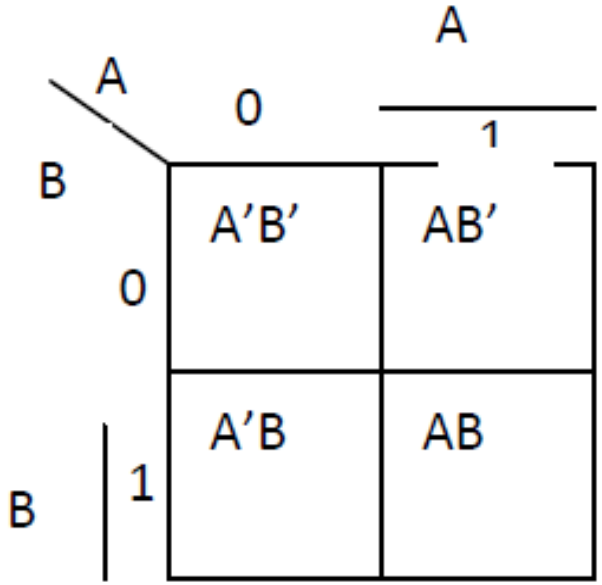
**That is, the maxterm with subscript $j$ is a complement of the minterm with the same subscript $j$ and vice versa.**

# K-map simplification

- The Karnaugh map also known as Veitch diagram or simply as K map is a two dimensional form of the truth table, drawn in such a way that the simplification of Boolean expression can be immediately be seen from the location of 1's in the map.
- The map is a diagram made up of squares , each square represent one minterm. Since any Boolean function can be expressed as a sum of min terms, it follows that a Boolean function is recognized graphically in the map from the area enclosed by those squares whose minterms are included in the function.

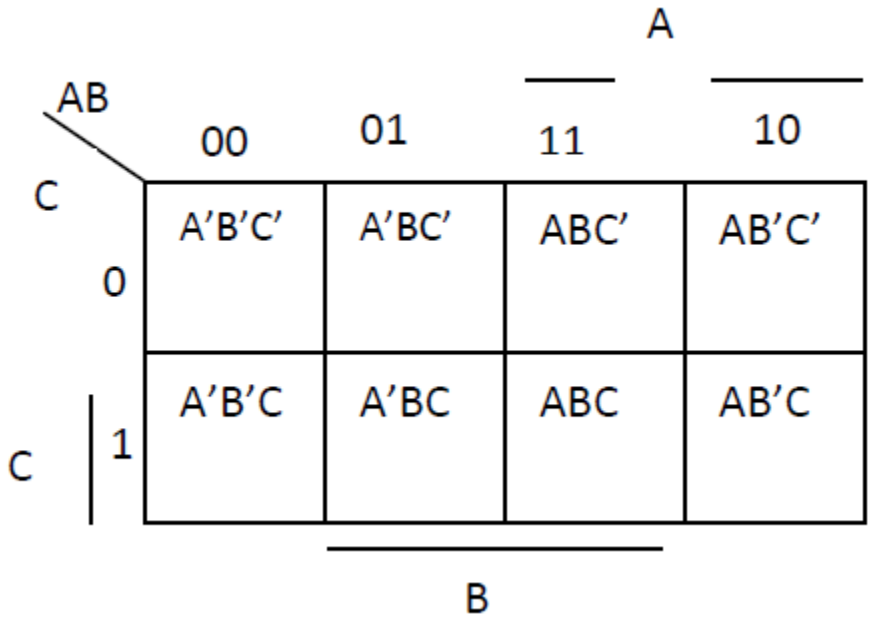
# Cont..

- A two variable Boolean function can be represented as follow



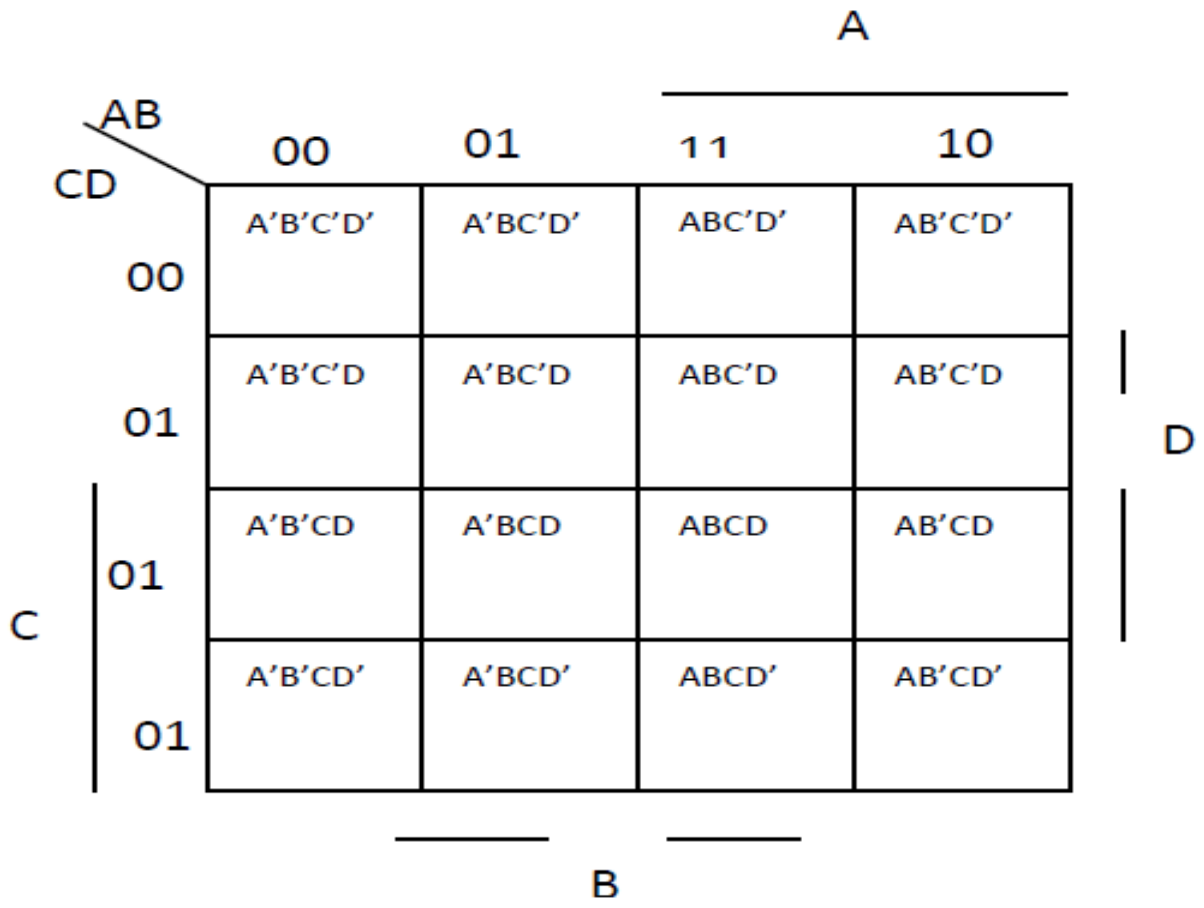
# Cont…

- A three variable function can be represented as follow



# Cont…

- A four variable Boolean function can be represented in the map bellow



# Cont…

- To simplify a Boolean function using karnaugh map, the first step is to plot all ones in the function truth table on the map.
- The next step is to combine adjacent 1’s into a group of one, two, four, eight, sixteen. The group of minterm should be as large as possible.
- A single group of four minterm yields a simpler expression than two groups of two minterms.

# Cont..

- In a four variable karnaugh map, 1 variable product term is obtained if 8 adjacent squares are covered
- 2 variable product term is obtained if 4 adjacent squares are covered
- 3 variable product term is obtained if 2 adjacent squares are covered
- 1 variable product term is obtained if 1 square is covered
- A square having a 1 may belong to more than one term in the sum of product expression

# Cont…

- The final stage is reached when each of the group of minterms are orderd together to form the simplified sum of product expression The karnaugh map is not a square or rectangle as it may appear in the diagram.
- The top edge is adjacent to the bottom edge and the left-hand edge adjacent to the right-hand edge. Consequently, two squares in karnaugh map are said to be adjacent if they differ by only one variable

# Minimization of Boolean expressions using Karnaugh maps.

Given the following truth table for the majority function.

| a | b | C | M(output) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

# Cont…

- The Boolean algebraic expression is

$$m = a'bc + ab'c + abc' + abc.$$

- the minimization using algebraic manipulation can be done as follows.

$$m = a'bc + abc + ab'c + abc + abc' + abc$$

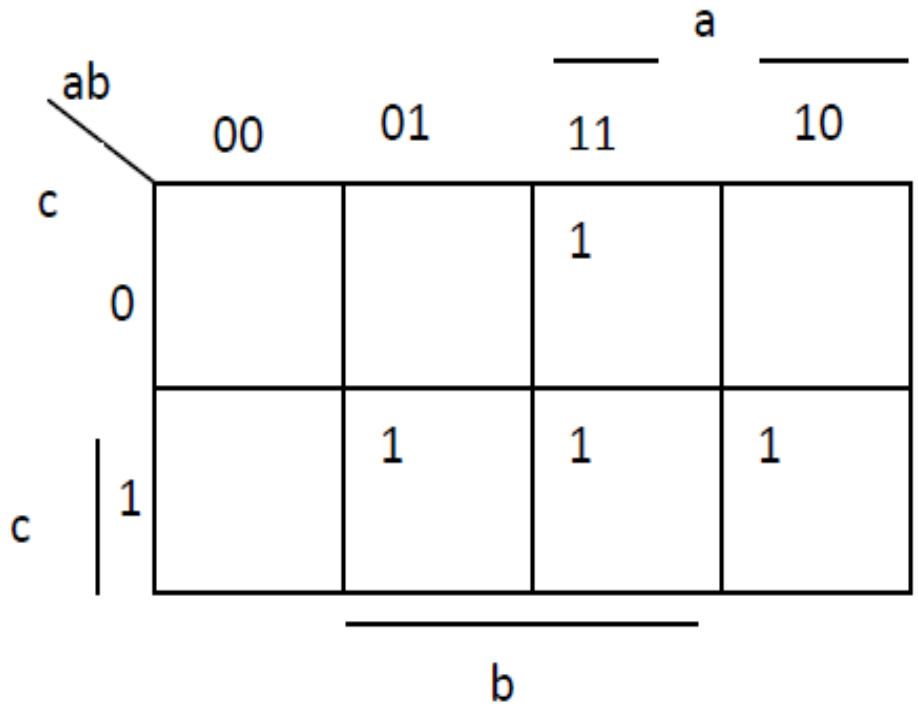
$$= (a' + a)bc + a(b' + b)c + ab(c' + c)$$

$$= bc + ac + ab$$

# Cont…

- The abc term was replicated and combined with the other terms.
- To use a Karnaugh map we draw the following map which has a position (square) corresponding to each of the 8 possible combinations of the 3 Boolean variables. The upper left position corresponds to the 000 row of the truth table, the lower right position corresponds to 101.

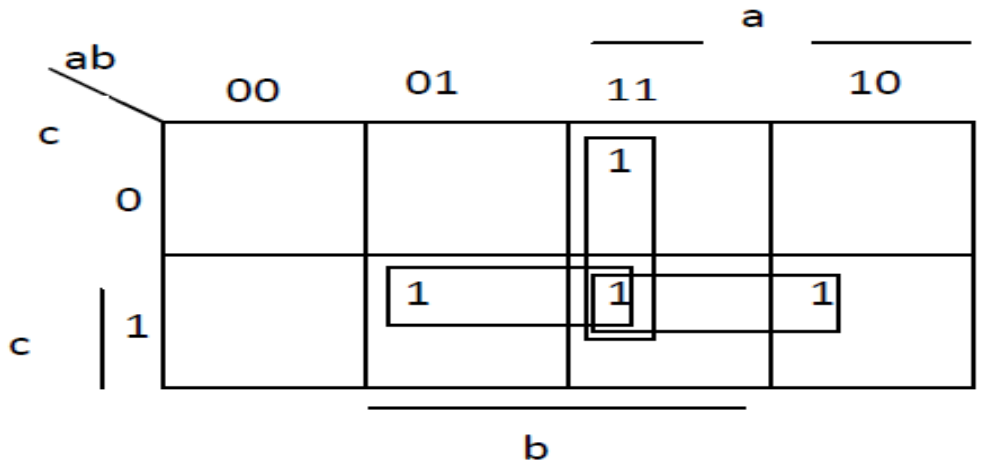
# Cont…



# Cont…

- The 1s are in the same places as they were in the original truth table. The 1 in the first row is at position 110 (**a = 1, b = 1, c = 0).**
- The minimization is done by drawing circles around sets of adjacent 1s. Adjacency is horizontal, vertical, or both. The circles must always contain $2^n$ 1s where n is an integer.

# Cont…



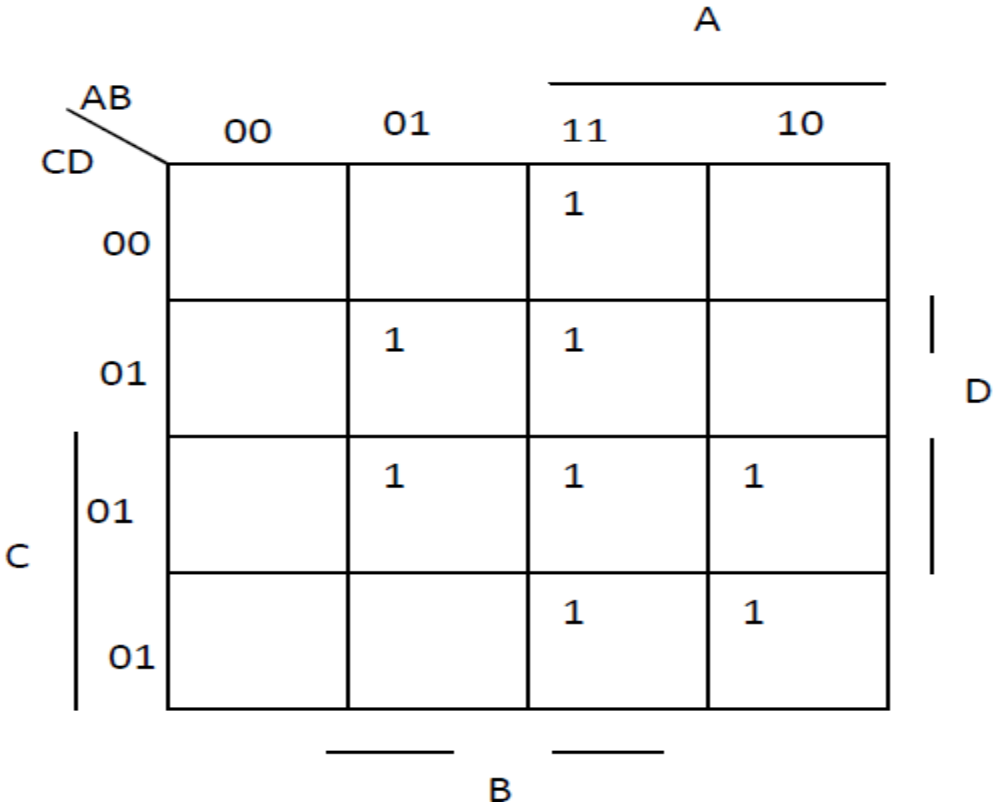

We have circled two 1s. The fact that the circle spans the two possible values of a (0 and 1) means that the a term is eliminated from the Boolean expression corresponding to this circle. Now we have drawn circles around all the 1s.

# cont…

- Thus the expression reduces to bc + ac + ab
- as we saw before. What is happening? What does adjacency and grouping the 1s together have to do with minimization? Notice that the 1 at position 111 was used by all 3 circles.
- This 1 corresponds to the abc term that was replicated in the original algebraic minimization. Adjacency of 2 1s means that the terms corresponding to those 1s differ in one variable only. In one case that variable is negated and in the other, it is not.
- The map is easier than algebraic minimization because we just have to recognize patterns of 1s in the map instead of using the algebraic manipulations. Adjacency also applies to the edges of the map

# Cont…

- Now for 4 Boolean variables. The Karnaugh map is drawn as shown below.



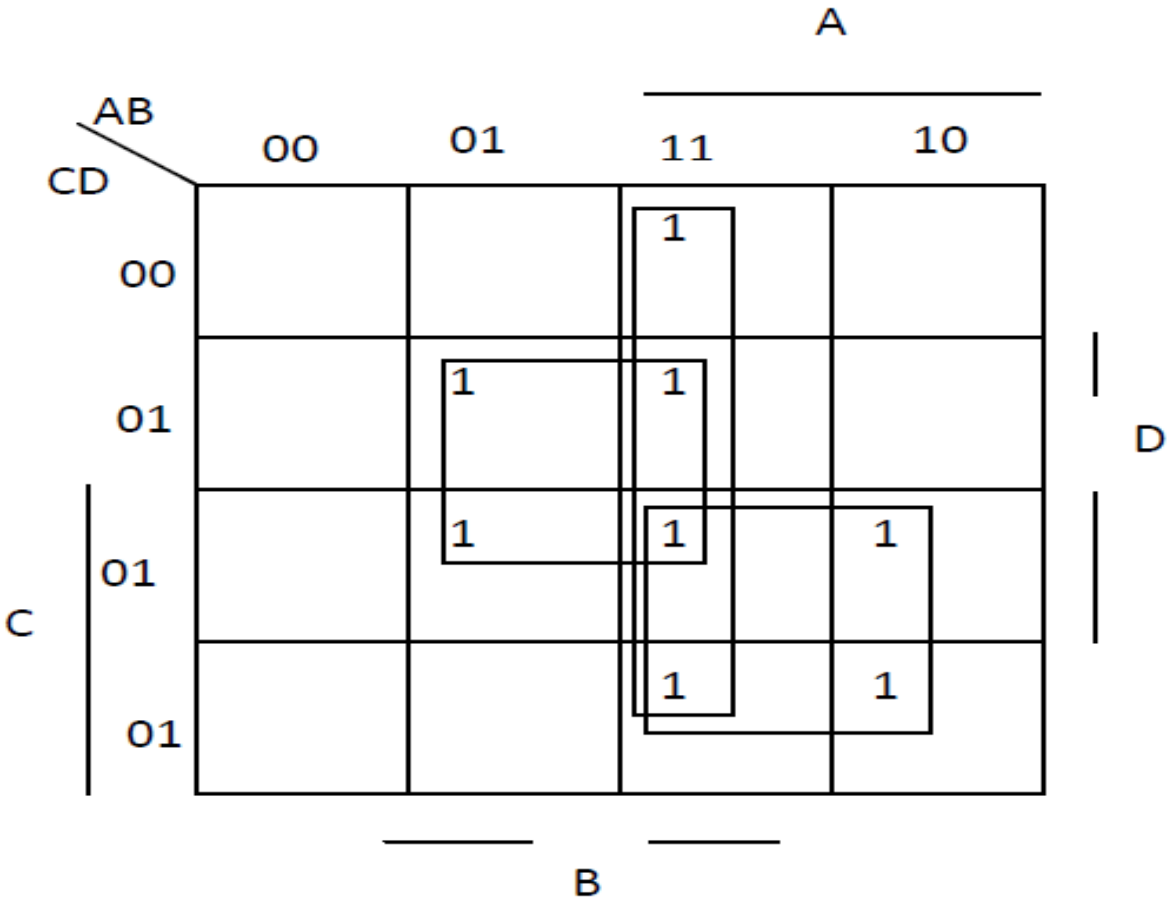
# Cont…

The following corresponds to the Boolean expression

$$Q = A'BC'D + A'BCD + ABC'D' + ABC'D + ABCD + ABCD' + AB'CD + AB'CD'$$

- RULE: Minimization is achieved by drawing the smallest possible number of circles, each containing the largest possible number of 1s. Grouping the 1s together results in the following.

# Cont..



# Cont…

- The expression for the groupings above is Q = BD + AC + AB This expression requires 3 2-input **AND gates and 1 3-input OR gate.**

**Exercise**

**1. F=A’B+AB**

2. F=A′B′C′+A′B′C+A′BC′+ABC′+ABC

3. **F=AB+A′BC′D+A′BCD+AB′C′D′**

4. F=AC′D′+A′B′C+A′C′D+AB′D

5. F=A′B′C′D′+AB′C′D′+A′BC′D+ABC′D+A′BCD+ABCD

Thank you!