MODULE - 2

Syllabus

I INTRODUCTION

Boolean Algebra

Boolean algebra is an algebra that may be defined with a set of elements, a set of operators and a number of unproved axioms or postulates. A set of elements is any collection of objects having common property The set of operators AND (Boolean product) operation (·), the OR (Boolean Sum) operation (+) and the NOT (complement operation) (') are defined in the set.

II Postulates

The postulates are the basic assumptions from which it is possible to deduce the rules, theorems and properties of the system.

1. Closure.

A set S is closed with respect to a binary operator if for every pair of elements of S, the binary operator specifies a rule for obtaining a unique element of S. Set of natural numbers $N = \{1, 2, 3, 4, \ldots\}$ is closed under binary operator (+) since any $a, b \in N$ we obtain a unique $c \in N$ by the operation $a + b = c$. Set of natural numbers is not closed with respect to binary operator minus (−) because $2 - 3 = (-1)$ where $-1 \notin N$

2. Associative Law

A binary operator * on a set S is said to associative whenever

$(x * y) * z = x * (y * z)$ for all $x, y, z \in S$

3. Commutative Law

A binary operator * on a set S is said to be commutative whenever

$x * y = y * x$ for all $x, y \in S$

4. Identity element

A set S is said to have an identity element respect to a binary operation * on S if there exists an element $e \in S$ with the property

$e * x = x * e = x$ for every $x \in S$

5. Inverse

A set S having the identity element e with respect to a binary operator * is said to have an inverse whenever, for every $x \in S$, there exists an element $y \in S$ such that

$x * y = e$

eg. $a + (-a) = 0$

6. Distributive Law

If * and • are two binary operators on a set S, * is said to be distributive over •, whenever

$x * (y \bullet z) = (x * y) \bullet (x * z)$

III POSTULATES OF BOOLEAN ALGEBRA

Boolean algebra is an algebric structure defined on a set of elements B together with two binary operators + and •

1 a) Closure with respect to the operator +.

b) Closure with respect to the operator •

2 a) An identity element with respect to +, designated by 0:

$$x + 0 = 0 + x = x$$

b) An identity element with respect to ·, designated by 1:

$$x \cdot 1 = 1 \cdot x = x$$

3 a) Commutative with respect to +:

$$x + y = y + x$$

b) Commutative with respect to ·:

$$x \cdot y = y \cdot x$$

4 a) · is distributive over +:

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$$

b) + is distributive over ·:

$$x + (y \cdot z) = (x + y) \cdot (x + z)$$

5. For every element $x \in B$, there exists an element $x' \in B$ (called the complement of x) such that:

a) $x + x' = 1$ &

b) $x \cdot x' = 0$

6. There exists at least two elements $x, y \in B$, such that $x \neq y$

Duality

Duality principle is an important property of Boolean algebra.

Duality states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged.

If the dual of an algebraic expression is desired simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

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) $x \cdot y = y \cdot x$
Theorem 4, associative	a) $x+(y+z)=(x+y)+z$	b) $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Postulate 4, distributive	a) $x(y+z)=xy+yz$	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) $x(x+y)=x$

Operator Precedence

The operator precedence for evaluating Boolean expression is
1) parentheses
2) NOT
3) AND
4) OR

Boolean Functions

A binary variable can take the value 0 or 1.
A boolean function is an expression formed with binary variables, the two binary operators OR and AND, the unary operator NOT, parentheses and equal sign

$$F_1 = xyz'$$

The function F_1 is equal to 1, if $x=1$ and $y=1$ and $z'=1$; otherwise $F_1=0$

literal

A **literal** is a primed or unprimed variable. When a boolean function is implemented with logic gates, each literal in the function designates an input to a gate and each term is implemented with a gate.

The minimization of the number of literals and the number of terms results in a circuit with less equipment.

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 theorem.

$$(A+B+C)' = (A+X)' \qquad \text{let } B+C = X$$

$$= A'X' \qquad \text{by De Morgan's theorem}$$

$$= A' \cdot (B+C)' \qquad \text{substitute } B+C=X$$

$$= A' \cdot (B' \cdot C') \qquad \text{by De Morgan theorem}$$

$$= A' \cdot B' \cdot C' \qquad \text{by associative theorem.}$$

IV CANONICAL AND STANDARD FORMS

i) Minterms and Maxterms

A binary variable may appear either in its normal form (x) or in its complement form (x'). Consider two binary variable x and y combined with an AND operation.

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. n variables can be combined to form 2^n minterms. Each minterm is obtained from an AND term of the n variables.

In a similar fashion, n variables forming an OR term with each variable provide 2^n possible combinations. These are called maxterms or standard sums.

			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

Table: Minterms & Maxterms for three binary variables.

(ii) Canonical Form

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

(iii) Sum of Minterms (Sum of Products - SOP)

Two or more AND functions ORed together

$AB+CD$

$AB+BCD$

$AB\bar{C} + \bar{D}E\bar{F} + FGH + A\bar{F}G$

Sum of products form can also contain a term with a single variable

$A + BCD + EFG$

In an SOP expression, a single overbar cannot extend over more than one variable, more than one variable in a term can have an overbar.

$\bar{A}\bar{B}\bar{C}$ ✓

$\overline{ABC}$ ✗

It is an useful form of Boolean expression because of the straight forward manner in which it can be implemented with logic gates

Canonical SOP contains all the term with all the variables either in complemented or uncomplemented form

eg $A\bar{B}C + A\bar{B}\bar{C} + \bar{A}BC$

Representation of SOP

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

$= m_1 + m_4 + m_5 + m_6 + m_7$

or the short notation

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

Σ → ORing of terms

(1, 4, 5, 6, 7) → minterms

(A, B, C) → variables

Qn | Express the Boolean function $F = A + B'C$ in a sum of minterms.

Soln:

$A + B'C$ (1st term, 2nd term)

Function has three variables A, B, C

First term: A is missing two variables

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

missing one more variable C

$$AB(C+C') + AB'(C+C')$$

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

Second term: B'C is missing one variable

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

Combining first & second term

$$F = A + B'C$$

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

(AB'C repeats)

AB'C appears twice, and according to theorem

$x + x = x$

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

$$= m_1 + m_4 + m_5 + m_6 + m_7$$

(iv) Product of Maxterms (Products of Sum) - POS

It is the dual of sum of products form. It is the AND of two or more OR functions

$(A+B)(B+C+D)$

$(A+B+\bar{C})(\bar{D}+\bar{E}+F)(F+G+H)$

Product of Sums expression can also contain a single variable term, such as $A(B+C+D)(E+F+G)$

In canonical POS, each sumterm contains all the variables either in complement or uncomplemented form

eg. $(A+B)(\bar{A}+B)\cdot(A+\bar{B})$

Representation of POS

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

$= M_0 M_2 M_4 M_5$

or the short notation

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

IV 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.

Consider the function

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

Complement is

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

Take the complement of F' by DeMorgan's theorem

$F = (m_0 + m_2 + m_3)'$

$= m_0' \cdot m_2' \cdot m_3'$

$= M_0 M_2 M_3$

$= \Pi(0,2,3)$

$F(x,y,z) = \Pi(0,2,4,5)$ is expressed in the product of maxterm form. Its conversion to Sum of Minterms is

$$F(x,y,z) = \Sigma(1,3,6,7)$$

Standard Forms

Standard form is used to express Boolean functio-ns. There are two types of standard forms:

i) The Sum of Products

ii) The Product of Sums

Sum of Products is a Boolean expression containing AND terms called Product terms of one or more literals each

eg. $F_1 = y' + xy + x'yz'$ → 3 product terms, 3 literals

Products of sums is a boolean expression containing OR terms, called sum terms. Each term may have any number of literals

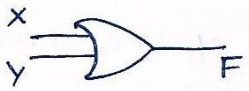
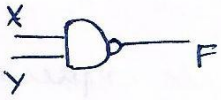
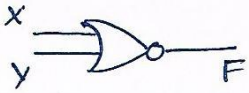
eg $F_2 = x(y'+z)(x'+y+z'+w)$ → 3 sum terms, 4 literals.

A boolean function may be expressed in a non standard form

$$F_3 = (AB+CD)(A'B'+C'D')$$

V LOGIC GATES

The manipulation of binary information is done by logic circuits called gates. Gates are blocks of hard-ware that produce signals of binary 1 or 0 when input logic requirements are satisfied. A variety of logic gates are commonly used in digital computer systems.

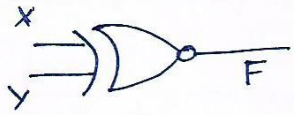
Name	Graphic Symbol	Algebraic function	Truth table
AND		$F = XY$	X Y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = X + Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter NOT		$F = X'$	X | F 0 | 1 1 | 0
Buffer		$F = X$	X | F 0 | 0 1 | 1
NAND		$F = (XY)'$	X Y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = (X+Y)'$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0

Exclusive-OR (XOR)

$F = xy' + x'y$
$= x \oplus y$

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-NOR or equivalence



$F = xy + x'y'$
$= x \odot y$

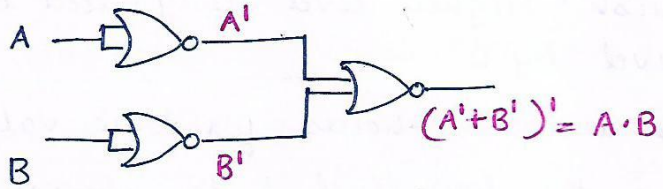
X	Y	F
0	0	1
0	1	0
1	0	0
1	1	1

Qu Construct AND gate using NOR gate

Soln:

$A \cdot B = \overline{\overline{(A \cdot B)}}$

$= \overline{(A' + B')}'$ Using De Morgon's theorem



IC Digital Logic Families

TTL - Transistor - transistor logic

ECL - Emitter - coupled logic

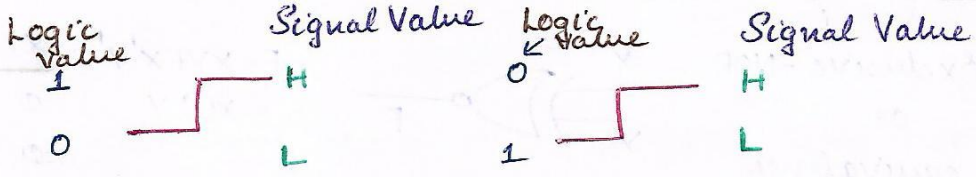
MOS - Metal - oxide semiconductor

CMOS - Complementary metal - oxide semiconductor

I^2L - Integrated - injection logic

Positive and Negative Logic

Binary signal at the input or output of any gate can have one of two values, except during transition. One signal value represents logic 1 and other logic 0



a) Positive Logic b) Negative Logic

fig. Signal - amplitude assignment and type of logic

Consider the two values of a binary signal. One value must be higher than the other since the two values must be different in order to distinguish. Higher level designated by H & lower level by L

There are two choices for logic value assignment choosing the low-level L to represent logic-1 defines a negative-logic system. Choosing the high-level H to represent logic-1 defines a positive-logic system.

Special Characteristics

i) Fan-out : specifies the number of standard loads that the output of a gate can drive without impairing its normal operation. A standard load is usually defined as the amount of current needed by an input of another gate in the same IC family.

ii) Power dissipation : is the supplied power required to operate the gate. This parameter is expressed in milliwatts (mW) and represents the actual power dissipated in the gate.

iii) Propagation delay : is the average transition delay time for a signal to propagate from input to output when the binary signals change in value.

The signals through a gate take a certain amount of time to propagate from the input to the output. This interval of time is defined as the propagation delay of the gate. It is expressed in nano seconds (ns)

iv) Noise Margin : is the maximum noise voltage added to the input signal of a digital circuit that does not cause an undesirable change in the circuit output.

VI SIMPLIFICATION / MINIMIZATION OF BOOLEAN FUNCTIONS

i) The Map Method / Veitch Diagram / Karnaugh Map

The complexity of the digital logic gates that implements a boolean function is directly related to the complexity of the algebraic expression from which the function is implemented. Boolean functions may be simplified by algebraic means but it lacks specific rules to predict each succeeding step. The map method provides a simple straight forward procedure for minimizing boolean functions. The map method first proposed by Veitch & slightly modified by Karnaugh, and is known as the "Veitch diagram" or the "Karnaugh map"

The map is a diagram made up of squares. Each square represents one minterm. Map presents a visual diagram of all possible ways a function may be expressed in a standard form.

(ii) Two- and Three-variable Map

Two-Variable Map

m_0	m_1
m_2	m_3

x \ y	0	1
0	$x'y'$	$x'y$
1	xy'	xy

Two-variable map

There are four minterms for two variables, hence map consist of four squares, one for each minterm.

eg $F = xy$

Since xy is equal to m_3, a 1 is placed inside the square that belongs to m_3

x \ y	0	1
0		
1		1

Three-Variable map

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

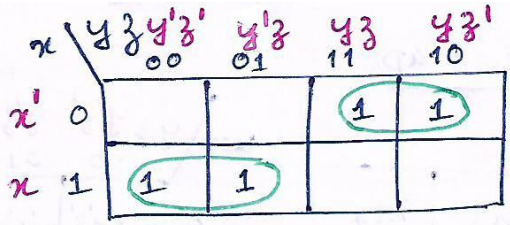
x \ yz	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

There are 8 minterms for three binary variables, hence map consist of eight squares.

Simplify the boolean function:

$$F = x'yz + x'yz' + xy'z' + xy'z$$

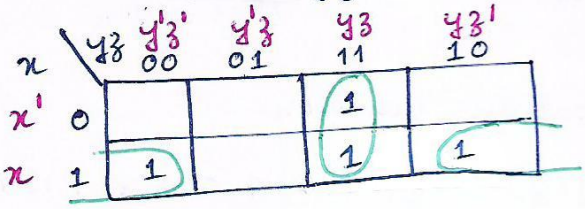
1. First mark 1 in each square to represent the function.
2. Group the adjacent 1's
3. Take the area of the group and represent as the sum of two terms



$F = xy' + x'y$

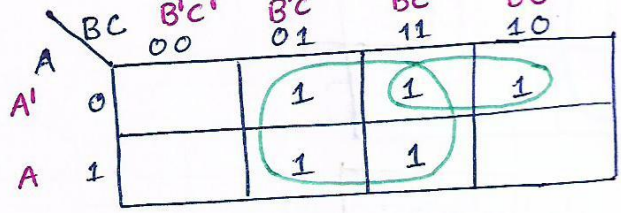
Qu 1. Simplify the boolean function:

$F = x'yz + xy'z' + xyz + xyz'$



$F = yz + xz'$

2. $F = A'C + A'B + AB'C + BC$



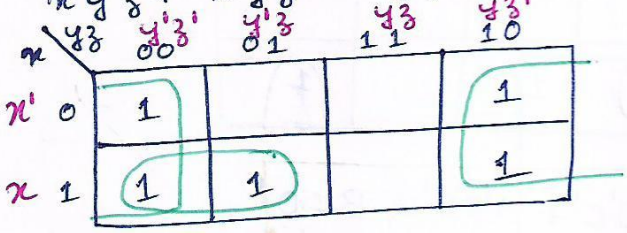
$F = C + A'B$

3. $F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$

$m_0 + m_2 + m_4 + m_5 + m_6$

$000 + 010 + 100 + 101 + 110$

$x'y'z' + x'yz' + xy'z' + xy'z + xyz'$



$F = z' + xy'$

Four - Variable Map

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}

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

Qn 1. Simplify the Boolean function

$$F = (w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$

wx \ yz	$y'z'$ 00	$y'z$ 01	yz 11	yz' 10
$w'x'$ 00	1	1		1
$w'x$ 01	1	1		1
wx 11	1	1		1
wx' 10	1	1		

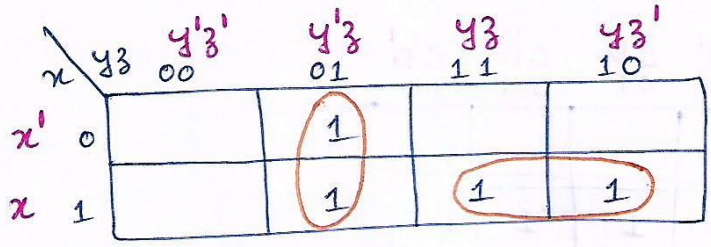
$$F = y' + w'z' + xz'$$

2. Simplify the Boolean function $F = A'B'C' + B'CD' + A'BCD' + AB'C'$

AB \ CD	$C'D'$ 00	$C'D$ 01	CD 11	CD' 10
$A'B'$ 00	1	1		1
$A'B$ 01				1
AB 11				
AB' 10	1	1		1

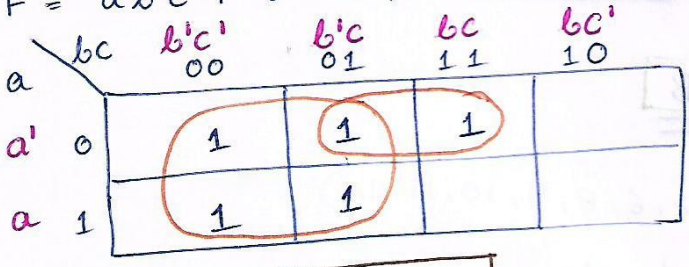
$$F = B'C' + A'CD' + B'D'$$

3. $F = xyz + xy'z + xyz' + x'y'z$



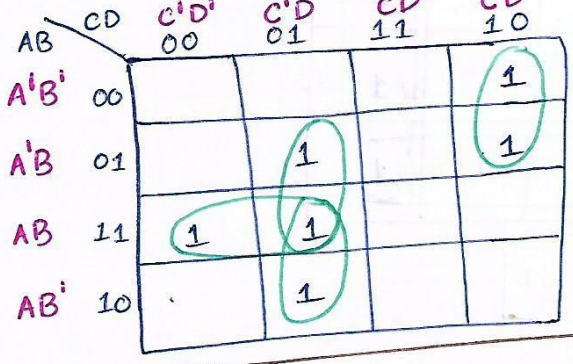
$F = y'z + xy$

4. $F = ab'c + a'bc + a'b'c + a'b'c' + ab'c'$



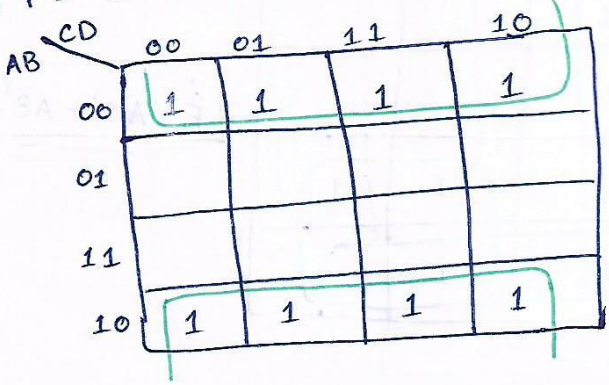
$F = b' + a'c$

5. $F = \Sigma(2, 5, 6, 9, 12, 13)$



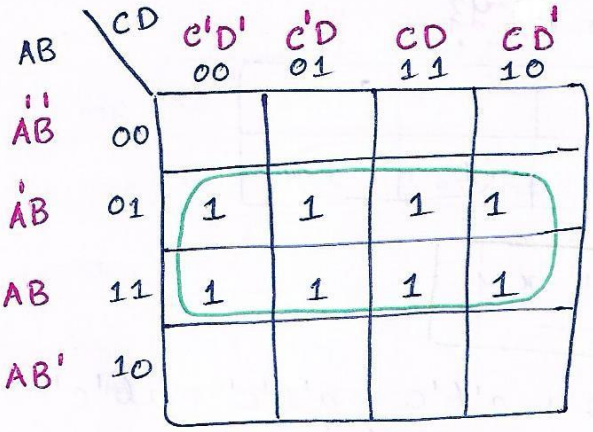
$F = A'CD' + AC'D + BC'D + ABC'$

6. $F = \Sigma(0, 1, 2, 3, 8, 9, 10, 11)$



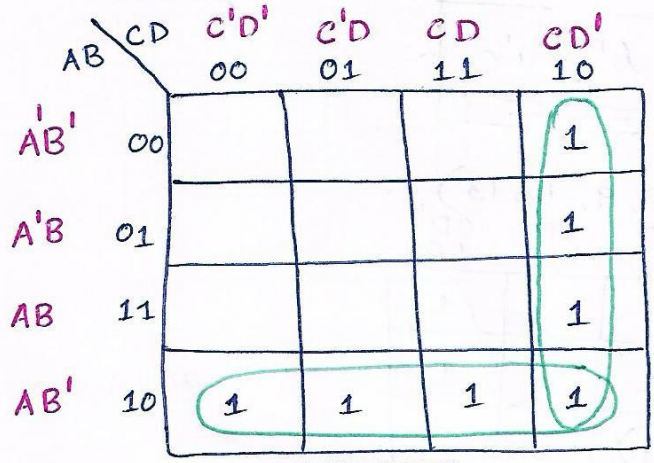
$F = \overline{B}$

7. $F = \Sigma(4, 5, 6, 7, 12, 13, 14, 15)$



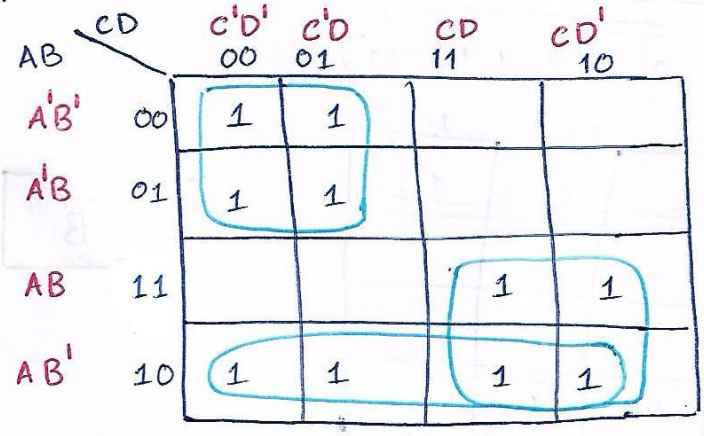
$F = B$

8. $F = \Sigma(2, 6, 8, 9, 10, 11, 14)$



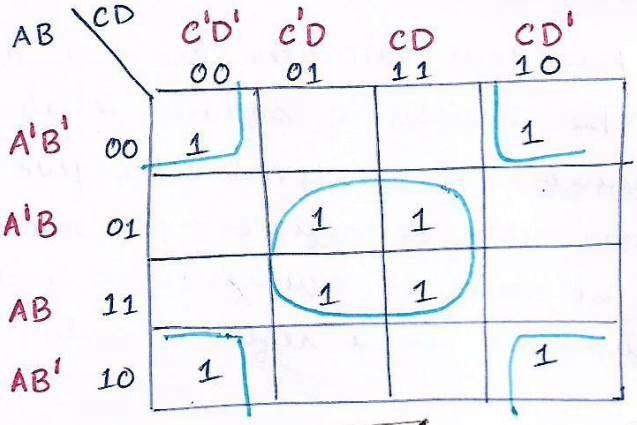
$F = AB' + CD'$

9. $F = \Sigma(0, 1, 4, 5, 8, 9, 10, 11, 14, 15)$



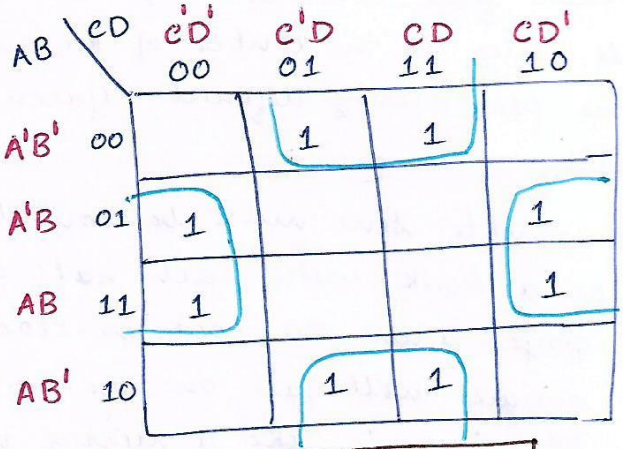
$F = A'C' + AB' + AC$

10. $F = \Sigma(0,2,5,7,8,10,13,15)$



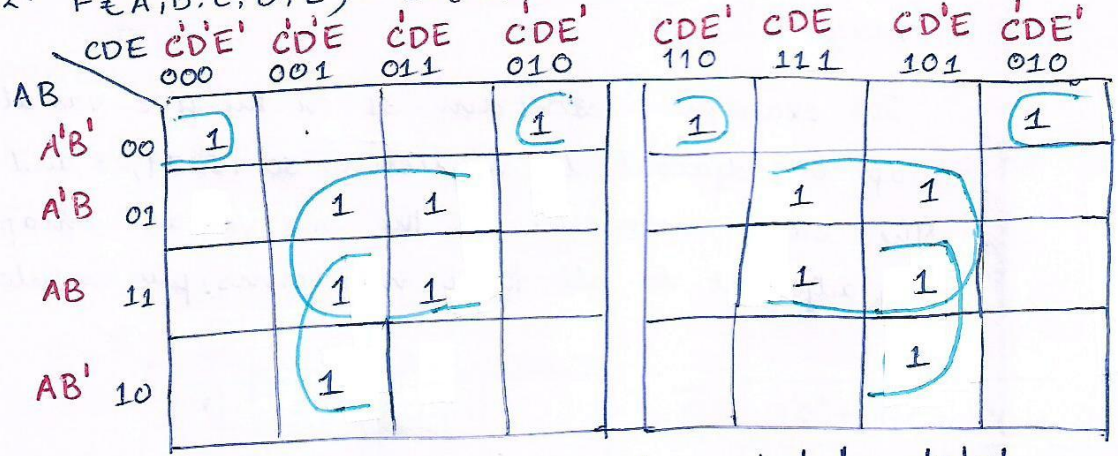
$F = BD + B'D'$

11. $F = \Sigma(1,3,4,6,9,11,12,14)$



$F = BD' + B'D$

12. $F(A,B,C,D,E) = \Sigma(0,2,4,6,9,11,13,15,17,21,25,27,29,31)$



$F = BE + AD'E + A'B'E' + A'B'E'$

$F = BE + AD'E + A'B'E'$

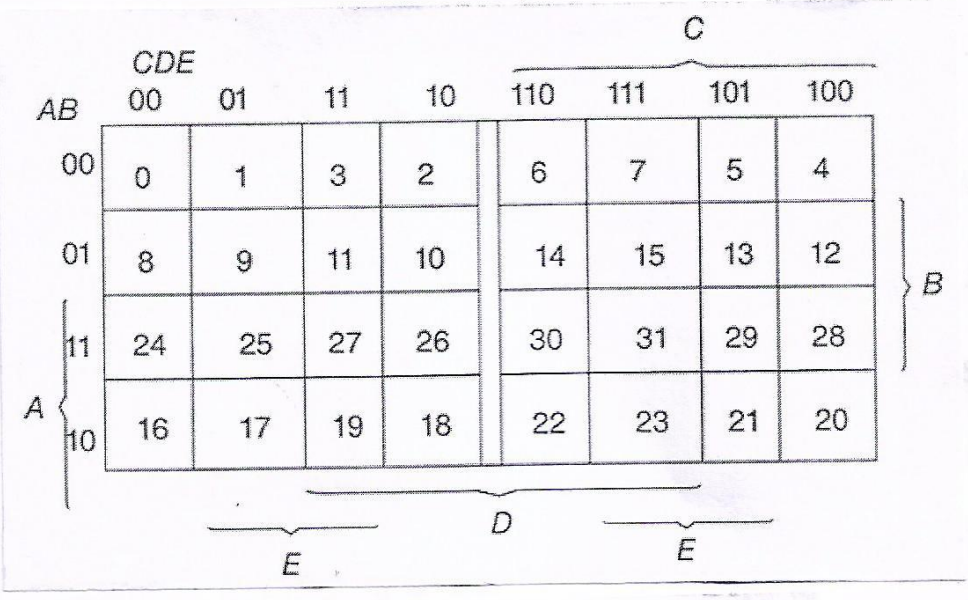
Five and Six Variable Maps

Maps of more than four variables are not as simple to use. The number of squares is always equal to the number of minterms. For five variable maps, we need 32 squares; for six variable maps we need 64 squares. Row and columns are numbered in a reflected-code sequence.

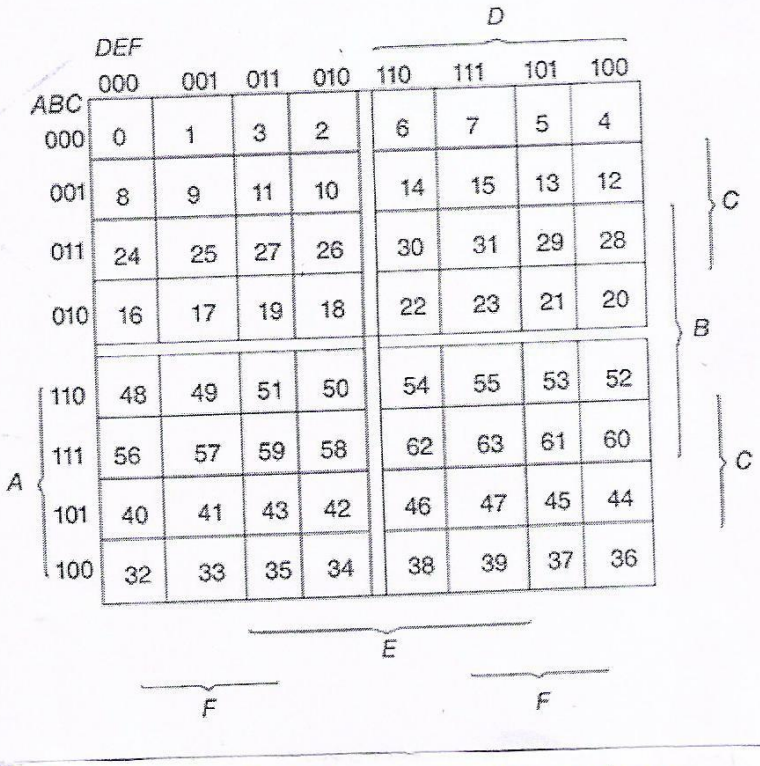
The five variable map must be thought to consist of two four variable maps, and the six variable map to consist of four four-variable map. Each of these four-variable maps is recognized from the double lines in the center of the map, each retains the previously defined adjacency when taken individually.

The center double line must be considered as the center of a book, with each half of the map being a page. When the book is closed two adjacent squares will fall one on the other. The center double line is like a mirror with each square being adjacent, not only to its four neighbouring squares, but also to its mirror image.

For example minterm 31 in the five variable map is adjacent to minterm 30, 15, 29, 23 and 27. The same minterm in the six variable map is adjacent to all these minterms plus minterm 63.



⇒ Five Variable map

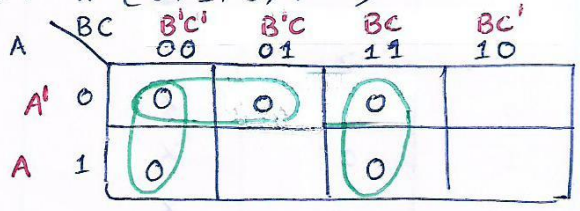


⇒ Six Variable Map

VII PRODUCT OF SUMS SIMPLIFICATION

Minimization of POS is similar to SOP except plotting of 0's instead of 1's

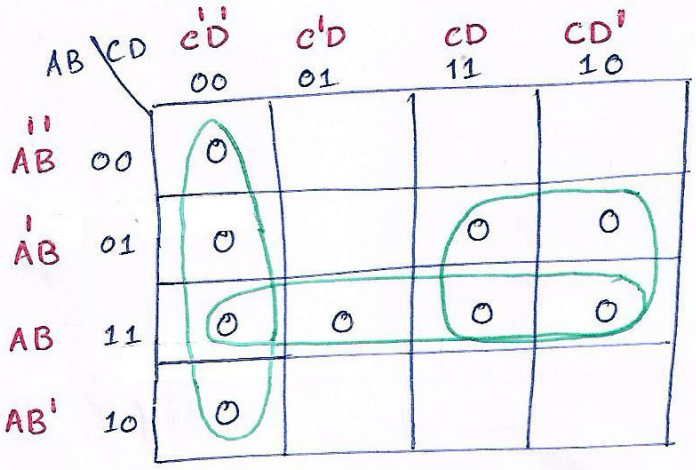
1. $F = \Pi(0, 1, 3, 4, 7)$



$\overline{F} = B'C' + BC + A'B' +$

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

2. $F = \Pi\ (0, 4, 6, 7, 8, 12, 13, 14, 15)$



$\overline{F} = c'D' + AB + BC$

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

VII DON'T - CARE CONDITIONS / CANNOT HAPPEN STATES

Some logic circuits can be designed so that there are certain input conditions for which there are no specified output levels, or we "don't care". In such cases, the output level is not defined, it can be either HIGH or LOW. These output levels are indicated by 'X' or 'd' or 'ϕ' in the truth tables and are called "dont care outputs" and such combinations are called don't care conditions.

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	X
1	1	0	X
1	1	1	X

When choosing adjacent squares to simplify the function in the map, the X's may be assumed to be either 0 or 1, whichever gives the simplest expression. An X need not be used at all if it does not contribute to covering a larger area.

A \ BC	$B'C'$ (00)	$B'C$ (01)	BC (11)	BC' (10)
A' (0)	0	1	0	0
A (1)	1	X	X	X

$F = A + B'C$

1. $F(A, B, C) = \Sigma m(0, 1, 5) + \Sigma d(4, 7)$

A \ BC	$B'C'$ (00)	$B'C$ (01)	BC (11)	BC' (10)
A' (0)	1	1		
A (1)	X	1	X	

$F = B'$

2. $F(A, B, C, D) = \Sigma m(0, 1, 3, 7, 15) + \Sigma d(2, 11, 12)$

AB \ CD	$C'D'$ (00)	$C'D$ (01)	CD (11)	CD' (10)
$A'B'$ (00)	1	1	1	X
$A'B$ (01)			1	
AB (11)	X		1	
AB' (10)			X	

$F = CD + A'B'$

3. $F(A, B, C, D) = \Sigma m(4, 6, 7, 13, 14) + \Sigma d(5, 10, 12, 15)$

AB \ CD	$C'D'$ 00	$C'D$ 01	CD 11	CD' 10
$A'B'$ 00				
$A'B$ 01	1	X	1	1
AB 11	X	1	X	1
AB' 10				X

$F = B$

4. $F(A, B, C, D) = \Pi M(0, 2, 6, 8, 12) + \Pi d(3, 4, 7, 10, 14)$

AB \ CD	$C'D'$ 00	$C'D$ 01	CD 11	CD' 10
$A'B'$ 00	0		X	0
$A'B$ 01	X		X	0
AB 11	0			X
AB' 10	0			X

$F = D'$

5. $F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15) + d(0, 2, 5)$

wx \ yz	$y'z'$ 00	$y'z$ 01	yz 11	yz' 10
$w'x'$ 00	X	1	1	X
$w'x$ 01		X	1	
wx 11			1	
wx' 10			1	

$F = yz + w'z$

VIII

TABULATION METHOD (QUINMCCLUSKY METHOD)

The map method of simplification is convenient as long as the number of variables does not exceed five or 6. Disadvantage of map is that it is essentially a trial and error procedure which relies on the ability of the human user to recognize certain patterns. For function of six or more variables, it is difficult to be sure that the best selection has been made.

Tabulation method is a specific step by step procedure that is guaranteed to produce a simplified standard form expression for a function with any number of variables.

It consists of two parts

(1) Find by an exhaustive search all the terms that are candidates for inclusion in the simplified function called prime implicants.

(2) Choose among the prime implicants those that give an expression with the least number of literals.

(1) Determination of Prime Implicants

↳ Compare each min term with every other minterm

↳ If two minterms differ in only one variable that variable is removed and term with one less literal is found

↳ Repeat the process for every minterm until the exhaustive search is completed. Matching process cycle is repeated for those new terms just found.

↳ Process continued until the exhaustive search is completed.

↳ The remaining terms and all the terms that did not match during the process comprise the prime implicants.

② Selection of Essential Prime Implicants

↳ Must select minimum number of PIs which cover all minterms

↳ Prepare a prime implicant chart with the given minterms as columns and the PIs as rows.

↳ Put 'X' in each column corresponding to the minterms which covers the PIs

↳ If any column contains just a single 'X', the PI corresponding to the row is essential PI and is included in the simplified expression

↳ Once the EPI are obtained, check whether they are covering all the minterms or not. If they are covering the resultant expression is unique

Qn 1. $F(A,B,C,D) = \Sigma m(0,2,3,6,7,8,10,12,13)$

Minterm	Binary representation	No. of 1's
m_0	0 0 0 0	0
m_2	0 0 1 0	1
m_3	0 0 1 1	2
m_6	0 1 1 0	2
m_7	0 1 1 1	3
m_8	1 0 0 0	1
m_{10}	1 0 1 0	2
m_{12}	1 1 0 0	2
m_{13}	1 1 0 1	3

Minterm	Index	Binary representation A B C D
m_0	0	0 0 0 0
m_2	1	0 0 1 0
m_8	1	1 0 0 0
m_3	2	0 0 1 1
m_6	2	0 1 1 0
m_{10}	2	1 0 1 0
m_{12}	2	1 1 0 0
m_7	3	0 1 1 1
m_{13}	3	1 1 0 1

Minterms Group	Binary representation
0, 2	0 0 – 0 ✓
0, 8	– 0 0 0 ✓
2, 3	0 0 1 – ✓
2, 6	0 – 1 0 ✓
2, 10	– 0 1 0 ✓
8, 10	1 0 – 0 ✓
8, 12	1 – 0 0
3, 7	0 – 1 1 ✓
6, 7	0 1 1 – ✓
12, 13	1 1 0 –

Minterms	Binary representation
0, 2, 8, 10	– 0 – 0
0, 8, 2, 10	– 0 – 0
2, 6, 3, 7	0 – 1 –
2, 3, 6, 7	0 – 1 –

PI table

Prime Implicants	Binary Representation	Literal representation
8, 12	1 – 0 0	$AC'D'$
12, 13	1 1 0 –	ABC'
0, 2, 8, 10	– 0 – 0	$B'D'$
2, 3, 6, 7	0 – 1 –	$A'C$

PI Chart

Prime Implicants	m_0	m_2	m_3	m_6	m_7	m_8	m_{10}	m_{12}	m_{13}
$AC'D'$						X		X	
ABC'								X	X
$B'D'$	X	X				X	X		
$A'C$		X	X	X	X				

$$F = \bar{B}D' + A'C + ABC'$$

Qn 2. $F = \Sigma(0, 1, 2, 8, 10, 11, 14, 15)$

Minterm	Binary representation	No of 1's
m_0	0 0 0 0	0
m_1	0 0 0 1	1
m_2	0 0 1 0	1
m_8	1 0 0 0	1
m_{10}	1 0 1 0	2
m_{11}	1 0 1 1	3
m_{14}	1 1 1 0	3
m_{15}	1 1 1 1	4

Minterm	Index	Binary representation
m_0	0	0 0 0 0
m_1	1	0 0 0 1
m_2	1	0 0 1 0
m_8	1	1 0 0 0
m_{10}	2	1 0 1 0
m_{11}	3	1 0 1 1
m_{14}	3	1 1 1 0
m_{15}	4	1 1 1 1

Minterms Group	Binary Representation
0,1	0 0 0 –
0,2	0 0 – 0 ✓
0,8	– 0 0 0 ✓
2,10	– 0 1 0 ✓
8,10	1 0 – 0 ✓
10,11	1 0 1 – ✓
10,14	1 – 1 0 ✓
11,15	1 – 1 1 ✓
14,15	1 1 1 – ✓

Minterms	Binary representation
0,2,8,10	– 0 – 0
0,8,2,10	– 0 – 0
10,11,14,15	1 – 1 –
10,14,11,15	1 – 1 –

PI Table

Prime Implicants	Binary Representation	Literal representation
0,1	0 0 0 –	$w'x'y'$
0,2,8,10	– 0 – 0	$x'z'$
10,11,14,15	1 – 1 –	wy

PI Chart

Prime Implicants	m_0	m_1	m_2	m_8	m_{10}	m_{11}	m_{14}	m_{15}
$w'x'y'$	X	X						
$x'z'$	X		X	X	X			
wy					X	X	X	X
		✓	✓	✓		✓	✓	✓

Reduced function is

$$F = w'x'y' + x'z' + wy$$