MODULE – 2

THE COMBINATIONAL LOGIC CIRCUITS

THE BASIC GATES

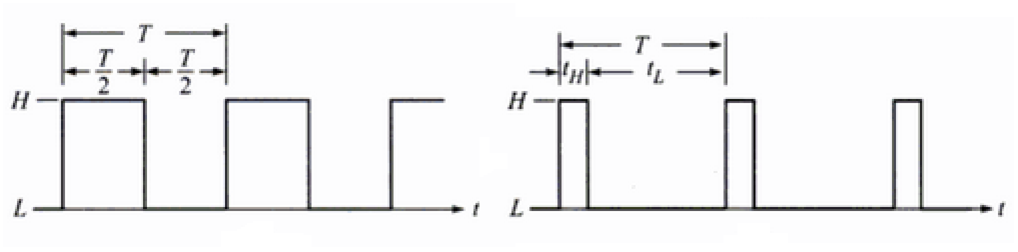
PREREQUISITES:

Electronic circuits and systems can be divided into two broad categories – *analog* and *digital*. Analog circuits are designed for use with small signals and are used in a linear fashion. Digital circuits are generally used with large signals and are considered nonlinear. Any quantity that changes with time can be represented as an analog signal or it can be treated as digital signal.

Digital electronics involves circuits that have exactly two possible states. A system having only two states is said to be *binary*. The binary number system is widely used in digital electronics.

Hexa-Decimal	Decimal	Binary		Hexa-Decimal	Decimal	Binary
0	0	0 0 0 0		8	8	1 0 0 0
1	1	0 0 0 1		9	9	1 0 0 1
2	2	0 0 1 0		A	10	1 0 1 0
3	3	0 0 1 1		B	11	1 0 1 1
4	4	0 1 0 0		C	12	1 1 0 0
5	5	0 1 0 1		D	13	1 1 0 1
6	6	0 1 1 0		E	14	1 1 1 0
7	7	0 1 1 1		F	15	1 1 1 1

The operation of electronic circuits can be described in terms of its voltage levels – *high* (H) level and *low* (L) level. This could be related to the binary number system by assigning L = 0 = F (false) and H = 1 = T (true).



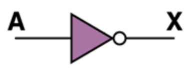
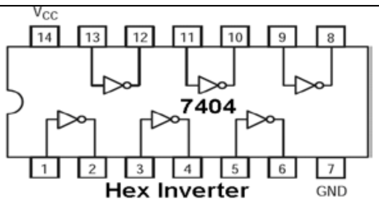
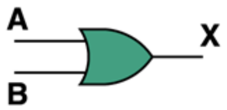
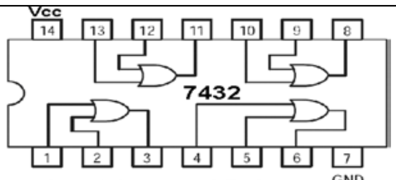
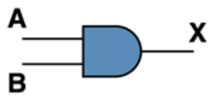
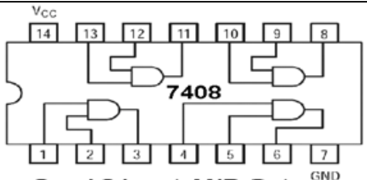
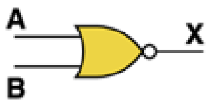
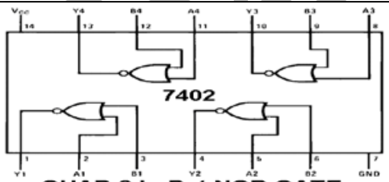
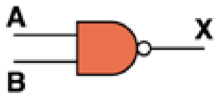
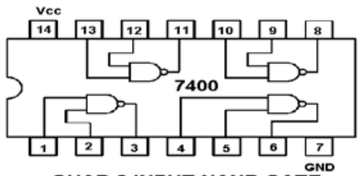
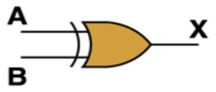
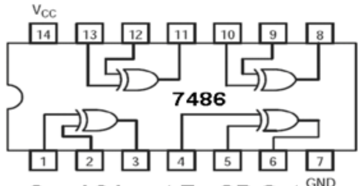
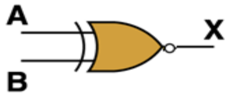
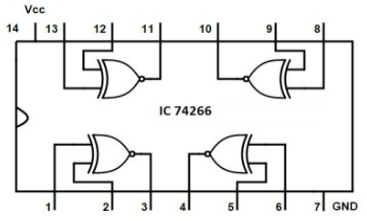
Symmetrical Signal & Asymmetrical Signal

The frequency is defined as, f = 1 / T where, T is the period of the signal.

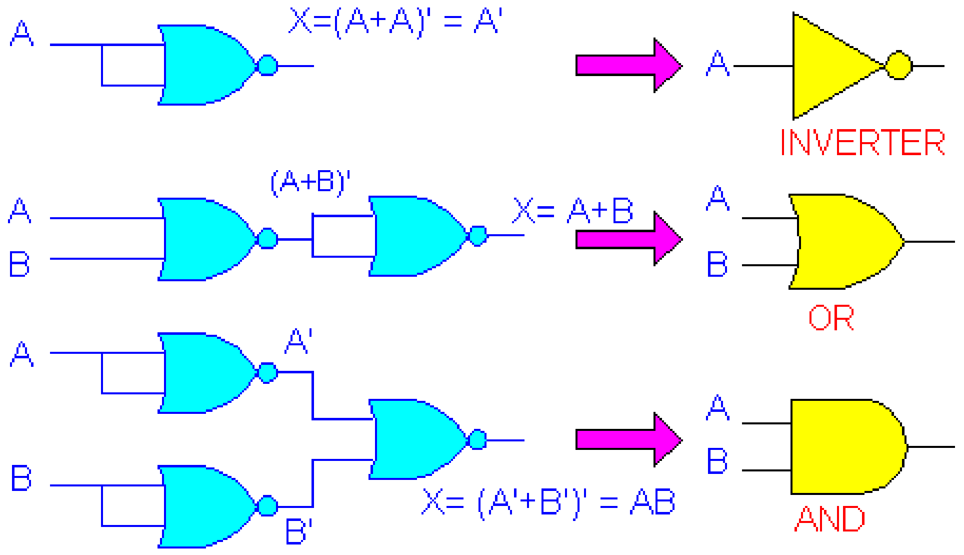
Duty Cycle is a convenient measure of how symmetrical or how unsymmetrical a waveform is.

$$Duty\ Cycle = \frac{T_{on}}{T_{on}+T_{off}} \qquad Duty\ Cycle, H = \frac{T_{on}}{T_{on}+T_{off}} \qquad Duty\ Cycle, L = \frac{T_{off}}{T_{on}+T_{off}}$$

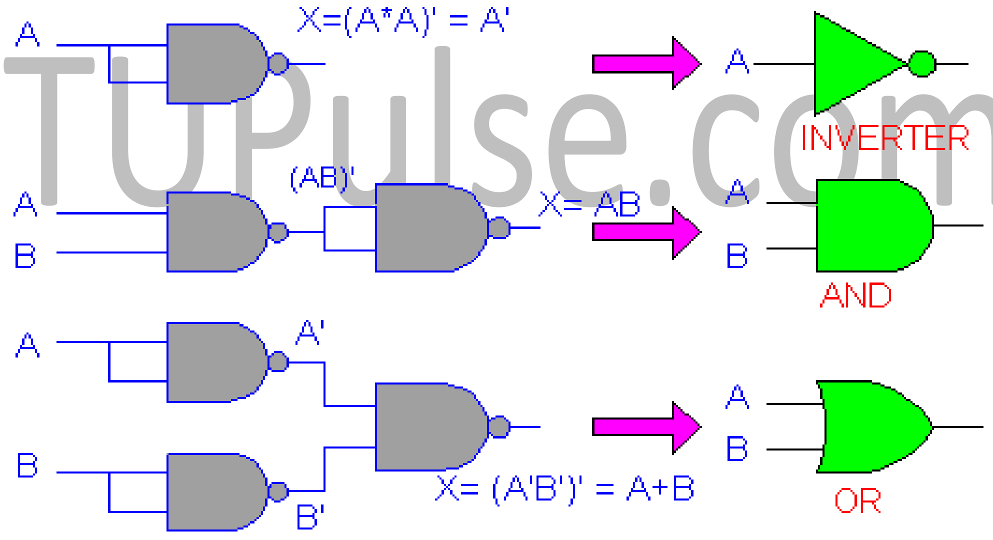
REVIEW OF LOGIC GATES:

Circuit Symbol	Truth Table			VHDL	IC Details
	A	B	X		
NOT Gate:  $X = \bar{A}$	0	-	1	X = ~A X <= **not** A;	
	1	-	0		
OR Gate:  $X = A + B$	0	0	0	X = A | B X <= A **or** B;	
	0	1	1		
	1	0	1		
	1	1	1		
AND Gate:  $X = A.B$	0	0	0	X = A & B X <= A **and** B;	
	0	1	0		
	1	0	0		
	1	1	1		
NOR Gate:  $X = \bar{A}.\bar{B}$	0	0	1	X = ~(A | B) X <= A **nor** B;	
	0	1	0		
	1	0	0		
	1	1	0		
NAND Gate:  $X = \bar{A} + \bar{B}$	0	0	1	X = ~(A & B) X <= A **nand** B;	
	0	1	1		
	1	0	1		
	1	1	0		
XOR Gate:  $X = A \oplus B$ $= \bar{A}B + A\bar{B}$	0	0	0	X = A ^ B X <= A **xor** B;	
	0	1	1		
	1	0	1		
	1	1	0		
XNOR Gate:  $X = A \odot B$ $= \bar{A}\bar{B} + AB$	0	0	1	X = ~(A ^ B) X <= A **xnor** B;	
	0	1	0		
	1	0	0		
	1	1	1		

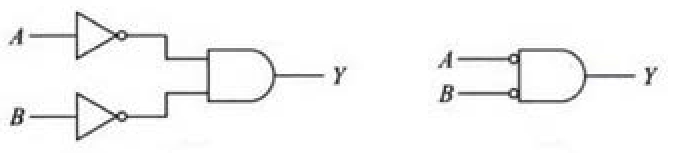
Universality of NOR Gate:



Universality of NAND Gate:



Bubbled AND Gate:

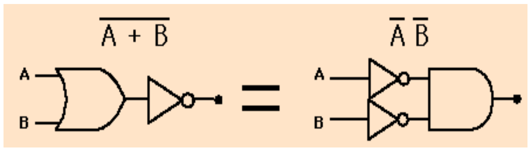


Bubbled AND gate and NOR gate are equivalent

De Morgan's First Theorem:

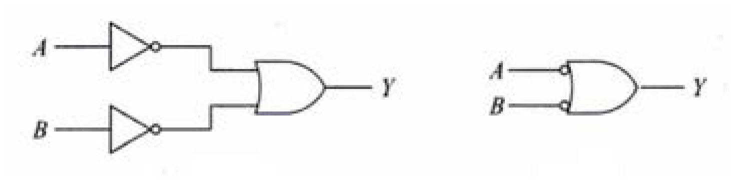
The complement of a sum equals the product of the complements. $\overline{A+B} = \bar{A}.\bar{B}$

Proof:

A	B	A+B	$\overline{A+B}$	$\bar{A}$	$\bar{B}$	$\bar{A}.\bar{B}$	
0	0	0	1	1	1	1	
0	1	1	0	1	0	0	
1	0	1	0	0	1	0	
1	1	1	0	0	0	0	

NOR Gate Bubbled AND Gate

Bubbled OR Gate:

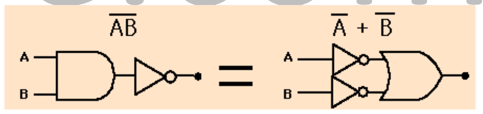


Bubbled OR gate and NAND gate are equivalent

De Morgan's Second Theorem:

The complement of a sum equals the product of the complements. $\overline{AB} = \bar{A} + \bar{B}$

Proof:

A	B	AB	$\overline{AB}$	$\bar{A}$	$\bar{B}$	$\bar{A}+\bar{B}$	
0	0	0	1	1	1	1	
0	1	0	1	1	0	1	
1	0	0	1	0	1	1	
1	1	1	0	0	0	0	

NAND Gate Bubbled OR Gate

Duality Theorem: *Starting with a Boolean relation, you can derive another Boolean relation by –*

1. *Changing each OR sign to an AND sign*
2. *Changing each AND sign to an OR sign*
3. *Complementing any 0 or 1appearing in the expression.*

Example: 1. We say that, A+0 = A; the dual is, A.1 = A

2. Consider, A(B+C) = AB + AC

By changing the OR and AND operation, we get the dual relation:

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

Laws of Boolean Algebra:

- The following laws are of immense use in the simplification of Boolean expressions.

- Note that, if A is a variable, then either A = 0 or A = 1. Also, when A = 0, A ≠ 1; and when A = 1, A ≠ 0.

De Morgan's First Theorem:-

The complement of sum is equal to the product of the complements.

(A + B)' = A' . B'   i.e., a bubbled AND gate & a NOR gate are equivalent.

De Morgan's Second Theorem:-

The complement of a product is equal to the sum of the compliments.

(A . B)' = A' + B'   i.e., a bubbled OR gate & a NAND gate are equivalent.

1) *Commutative Law:-*

   A + B = B + A   and   A . B = B . A

2) *Associative Law:-*

   A + (B + C) = (A + B) + C   and   A . (BC) = (AB) . C

3) *Distributive Law:-*

   A(B + C) = AB + AC

4) *In relation to OR operation, the following laws hold good:-*

   A + 0 = A

   A + A = A

   A + 1 = 1 and

   A + A' = 1

5) *In relation to AND operation, the following laws hold good:-*

   A . 1 = A

   A . A = A

   A . 0 = 0

   A . A' = 0

   A'' = A

6) *Some more useful Boolean relations:-*

   A + AB = A

   A + A'B = A + B

   A (A + B) = A

   A (A' + B) = AB

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

Simplification of Boolean Expressions:-

❖ The following **hints** are found to be of use, in reducing complex Boolean expressions –

1. If there are parentheses present in the given expression, they are removed first; since, multiplication should precede addition.

   E.g.:- AB + C (A + B) = AB + AC + BC

2. If there are several identical terms, all except one can be removed.

   E.g.:- A + B + C + A . 1 = A + B + C + A = A + B + C

3. If a variable repeats in a term, only one variable may be retained.

   E.g.:- A . A = A

   B .B . C = BC

4. If in any term, both a variable & its complement are present, that term may be removed; since, AA’ = 0.

   E.g.:- XX’Y = 0 . Y = 0

5. Identify pairs of terms which contains same variables. If in a pair, a variable is absent in one term, it can be removed.

   E.g.:- ABCD + ABC = ABC (D + 1)

   = ABC . 1 since, 1 + D =1

   = ABC

6. If, in a pair of terms, several variables are common, and another variable is present in one term & its complement is present in another term, this variable & its complement can be removed.

   E.g.:- ABC + A’BC = BC (A’ + A)

   = BC . 1 since, A’ + A = 1

   = BC

KARNAUGH MAPS

MINIMUM FORMS OF SWITCHING FUNCTIONS:

When a function is realized using AND and OR gates, *the cost of realizing the function is directly related to the number of gates and gate inputs used*. The Karnaugh map techniques developed, lead directly to *minimum cost* two-level circuits composed of AND and OR gates. An expression consisting of a *sum-of-product* terms corresponds directly to a two-level circuit composed of a group of AND gates feeding a single OR gate (see the following Figure). Similarly, a *product-of-sums* expression corresponds to a two-level circuit composed of OR gates feeding a single AND gate.

Therefore, to find minimum cost two-level AND-OR gate circuits, we must find minimum expressions in sum-of-products or product-of-sums form.

A *minimum sum-of-products* expression for a function is defined as a sum of product terms which

a) has a minimum number of terms and
b) of all those expressions which have the same minimum number of terms, has a minimum number of literals.

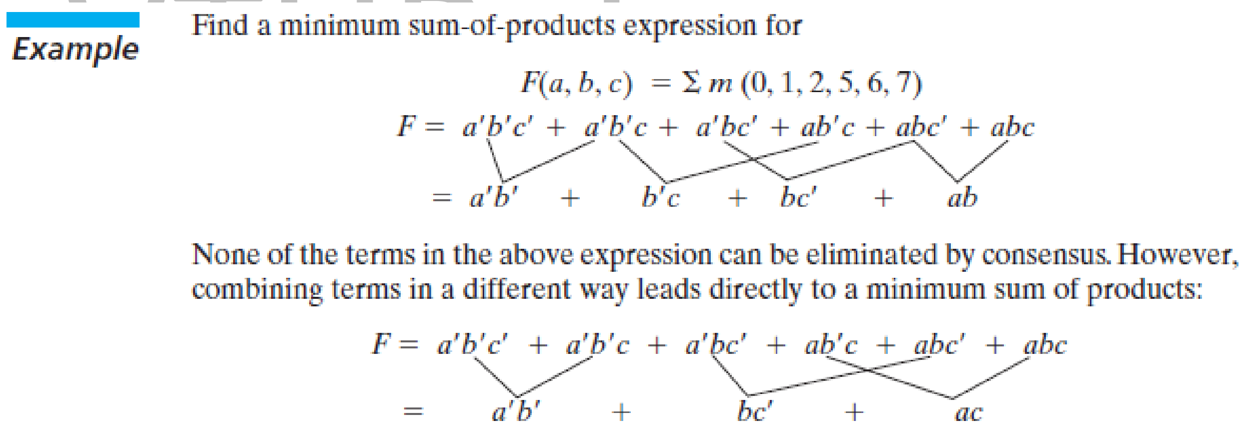
The minimum sum of products corresponds directly to a minimum two-level gate circuit which has

a) a minimum number of gates and
b) a minimum number of gate inputs.

Unlike the minterm expansion for a function, the minimum sum of products is not necessarily unique; that is, a given function may have two different minimum sum-of-products forms, each with the same number of terms and the same number of literals.

Given a minterm expansion, the minimum sum-of products form can often be obtained by the following procedure:

1. Combine terms by using $XY' + XY = X(Y' + Y) = X$. Do this repeatedly to eliminate as many literals as possible. A given term may be used more than once because X+X=X.
2. Eliminate redundant terms by using the theorems of Boolean Algebra.



Example Find a minimum sum-of-products expression for

$$F(a, b, c) = \Sigma\, m\,(0, 1, 2, 5, 6, 7)$$

$$F = a'b'c' + a'b'c + a'bc' + ab'c + abc' + abc$$

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

None of the terms in the above expression can be eliminated by consensus. However, combining terms in a different way leads directly to a minimum sum of products:

$$F = a'b'c' + a'b'c + a'bc' + ab'c + abc' + abc$$

$$= a'b' + bc' + ac$$

A *minimum product-of-sums* expression for a function is defined as a product of sum terms which

a) has a minimum number of factors, and
b) of all those expressions which have the same number of factors, has a minimum number of literals.

Unlike the maxterm expansion, the minimum product-of-sums form of a function is not necessarily unique. Given a maxterm expansion, the minimum product of sums can often be obtained by a procedure similar to that used in the minimum sum-of-products case, except that the theorem $(X + Y')(X + Y) = X$ is used to combine terms.

Example

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

$$= (A+B'+D') \quad (A+B'+C') \quad (B'+C'+D) \quad (B+C'+D)$$

$$= (A+B'+D') \quad \underbrace{(A+B'+C')} \quad (C'+D)$$ ← eliminate by consensus

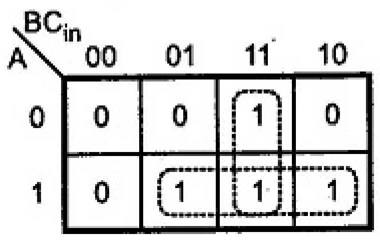
$$= (A+B'+D')(C'+D)$$

A	B	C	Y – Fundamental Product	Min-term
0	0	0	0	m0
0	0	1	0	m1
0	1	0	0	m2
0	1	1	1 – $\bar{A}BC$	m3
1	0	0	0	m4
1	0	1	1 – $A\bar{B}C$	m5
1	1	0	1 – $AB\bar{C}$	m6
1	1	1	1 – ABC	m7

A	B	C	Y – Fundamental Sum	Max-term
0	0	0	0 – A + B + C	M0
0	0	1	0 – A + B + $\bar{C}$	M1
0	1	0	0 – A + $\bar{B}$ + C	M2
0	1	1	1	M3
1	0	0	0 – $\bar{A}$ + B + C	M4
1	0	1	1	M5
1	1	0	1	M6
1	1	1	1	M7

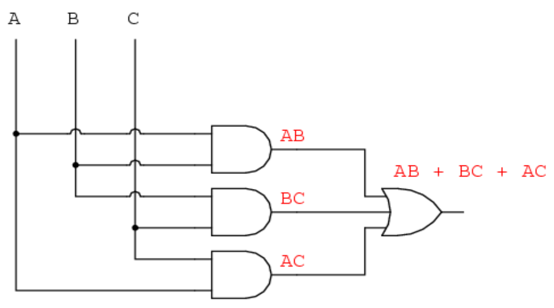
$Y_{SOP} = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

$= \sum m(1, 2, 4, 7).$



$$Y_{SOP} = AB + BC + AC$$

SOP Circuit Diagram:

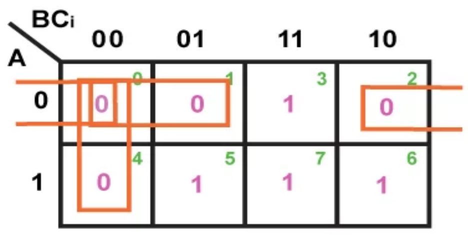


No. of Gates = 4

No. of Gate Inputs = 9

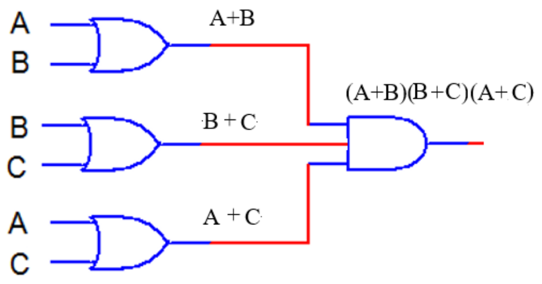
$Y_{POS} = (A + B + C)\,(A + B + \bar{C})\,(A + \bar{B} + C)\,(\bar{A} + B + C)$

$= \prod M(0, 1, 2, 4).$



$$Y_{POS} = (A + B)(B + C)(A + C)$$

POS Circuit Diagram:



No. of Gates = 4

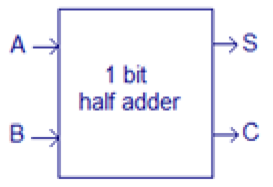
No. of Gate Inputs = 9

Example: Adders & Subtractors

Adder *circuit is a combinational digital circuit that is used for adding two numbers. A typical adder circuit produces a sum bit (denoted by S) and a carry bit (denoted by C) as the output. Adder circuits are of two types: Half adder ad Full adder.*

Subtractor *is the one which used to subtract two binary numbers (digits) and provides Difference and Borrow as an output.*

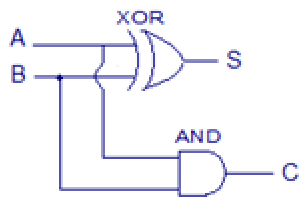
Half Adder & Half Subtractor:



Schematic

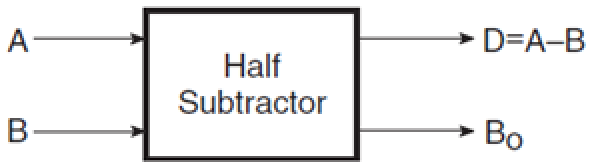
Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth table

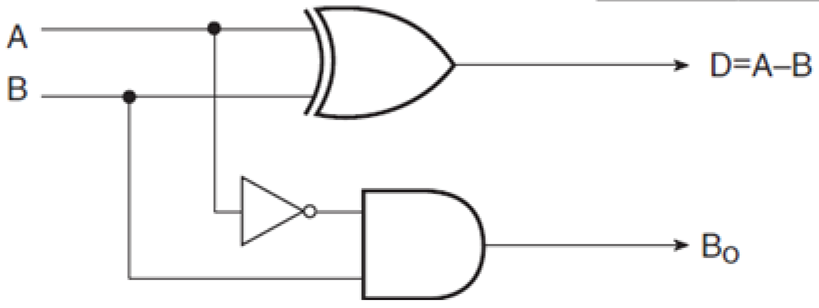


Realization

$Sum, S = \bar{A}B + A\bar{B}$ $\qquad$ $Carry, C = AB$



A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



$Difference, D = \bar{A}B + A\bar{B}$ $\qquad$ $Borrow, B_o = \bar{A}B$

Full Adder & Full Subtractor:

A	*B*	*Ci*	*S*	*Co*
0	*0*	*0*	*0*	*0*
0	*0*	*1*	*1*	*0*
0	*1*	*0*	*1*	*0*
0	*1*	*1*	*0*	*1*
1	*0*	*0*	*1*	*0*

A	*B*	*Ci*	*D*	*Bo*
0	*0*	*0*	*0*	*0*
0	*0*	*1*	*1*	*1*
0	*1*	*0*	*1*	*1*
0	*1*	*1*	*0*	*1*
1	*0*	*0*	*1*	*0*

1	0	1	0	1		1	0	1	0	0
1	1	0	0	1		1	1	0	0	0
1	1	1	1	1		1	1	1	1	1

Sum, $S = \sum m\,(1, 2, 4, 7) = \prod M(0, 3, 5, 6)$.

$$Sum, S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$Or, S = \bar{C}(\bar{A}B + A\bar{B}) + C(\bar{A}\bar{B} + AB)$$

$$Or, S = \bar{C}(A \oplus B) + C(A \odot B)$$

$$Or, S = \bar{C}(A \oplus B) + C(\overline{\overline{A \oplus B}})$$

$$Therefore, \boldsymbol{S = A \oplus B \oplus C}$$

Difference, $D = \sum m\,(1, 2, 4, 7) = \prod M(0, 3, 5, 6)$.

$$Diff., D = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$Or, D = \bar{C}(\bar{A}B + A\bar{B}) + C(\bar{A}\bar{B} + AB)$$

$$Or, D = \bar{C}(A \oplus B) + C(A \odot B)$$

$$Or, D = \bar{C}(A \oplus B) + C(\overline{\overline{A \oplus B}})$$

$$Therefore, \boldsymbol{D = A \oplus B \oplus C}$$

Carry Out, $Co = \sum m\,(3, 5, 6, 7) = \prod M(0, 1, 2, 4)$.

$$Carry\ Out, Co = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$Or, Co = \bar{C}(AB) + C(\bar{A}B + A\bar{B} + AB)$$

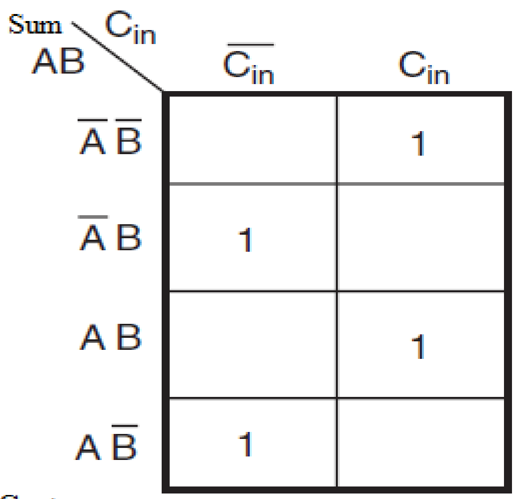
$$Therefore, \boldsymbol{Co = AB + BC + AC}$$

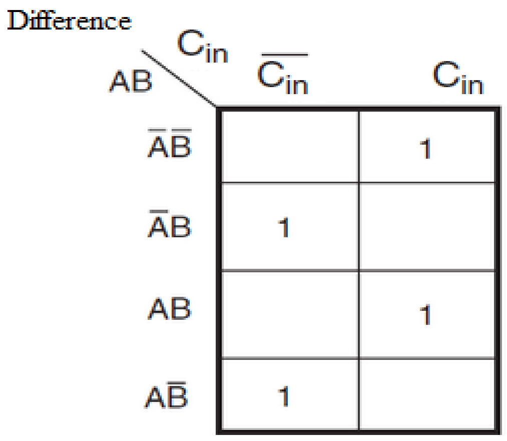
Borrow, $Bo = \sum m\,(1, 2, 3, 7) = \prod M(0, 4, 5, 6)$.

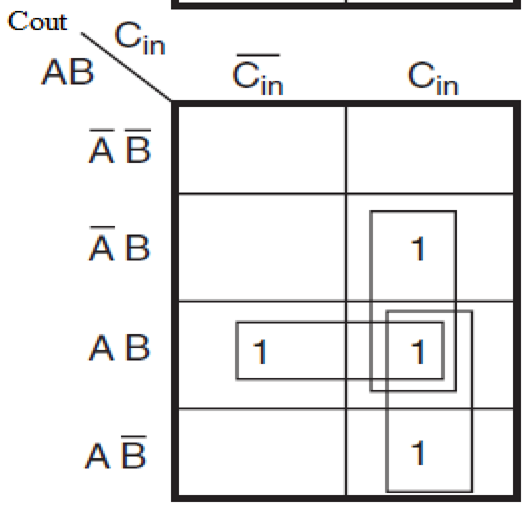
$$Borrow, Bo = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + ABC$$

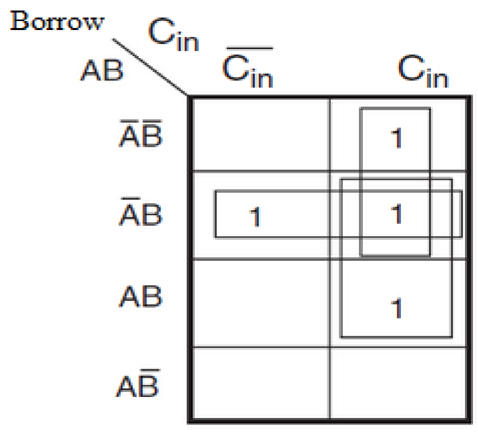
$$Or, Bo = \bar{C}(\bar{A}B) + C(\bar{A}\bar{B} + \bar{A}B + AB)$$

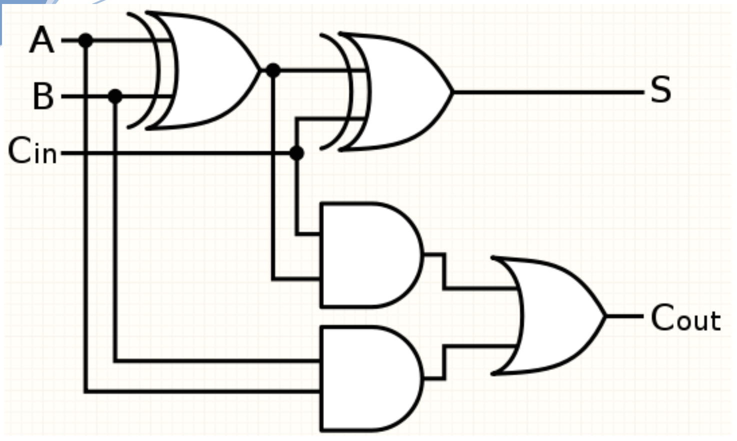
$$Therefore, \boldsymbol{Bo = \bar{A}B + BC + \bar{A}C}$$

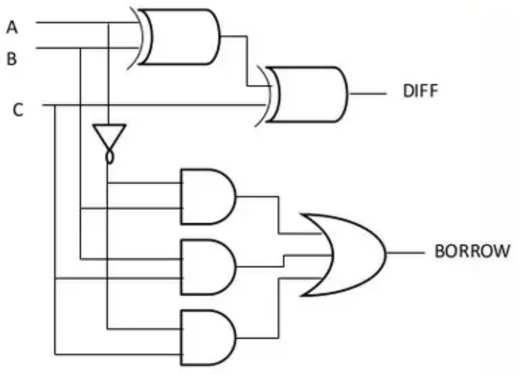






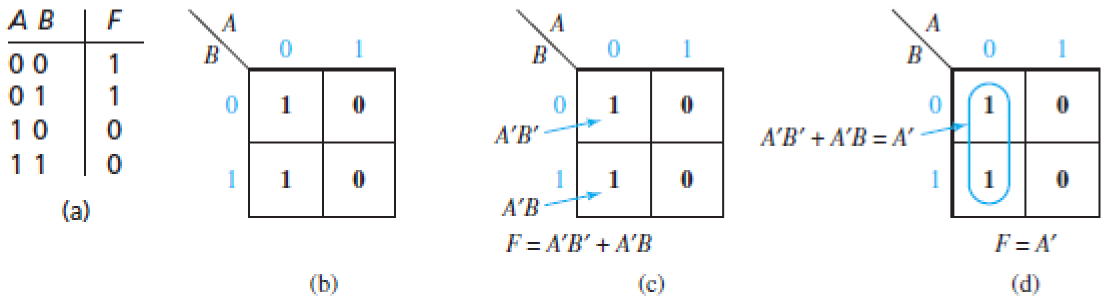




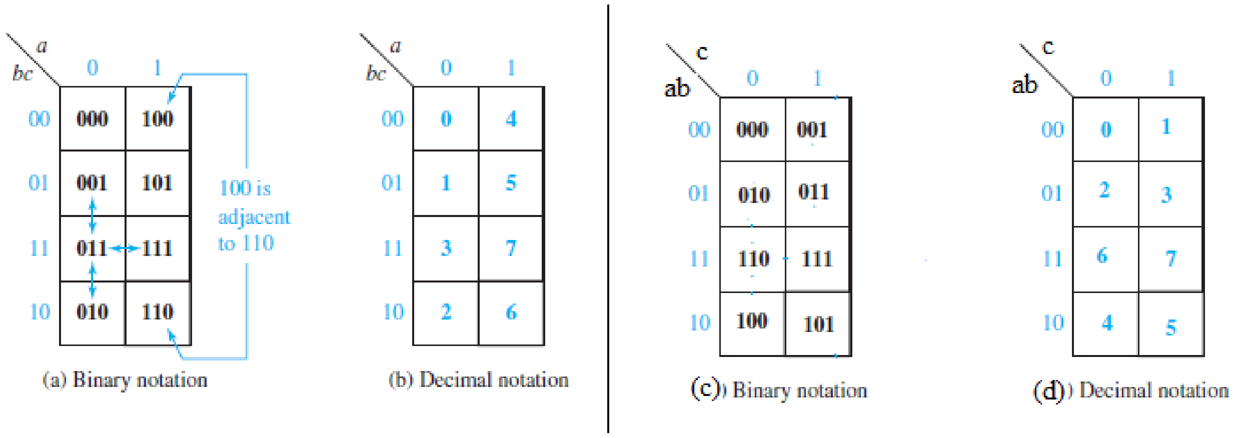


TWO AND THREE VARIABLE KARNAUGH MAPS:

Just like a truth table, the Karnaugh map of a function specifies the value of the function for every combination of values of the independent variables. The following Figure shows the truth table for a function F and the corresponding Karnaugh map:



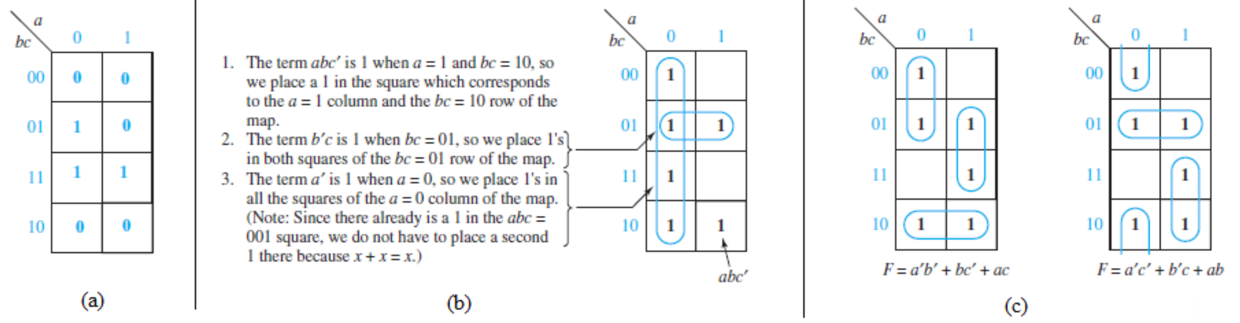
The following Figure shows a three-variable truth table and the corresponding Karnaugh map:



Example: *Write the Karnaugh Map for – (a)* $f = \sum m\,(1, 3, 5)$ *(b)* $f(a,b,c) = abc' + b'c + a'$

(c) $F = \sum m\,(0, 1, 2, 5, 6, 7)$

Solution: (a) $f = \sum m\,(1, 3, 5)$ *(b)* $f(a,b,c) = abc' + b'c + a'$ *(c)* $F = \sum m\,(0, 1, 2, 5, 6, 7)$



1. The term abc' is 1 when $a = 1$ and $bc = 10$, so we place a 1 in the square which corresponds to the $a = 1$ column and the $bc = 10$ row of the map.
2. The term $b'c$ is 1 when $bc = 01$, so we place 1's in both squares of the $bc = 01$ row of the map.
3. The term a' is 1 when $a = 0$, so we place 1's in all the squares of the $a = 0$ column of the map. (Note: Since there already is a 1 in the abc = 001 square, we do not have to place a second 1 there because $x + x = x$.)

(a) (b) (c)

Example: *Find two different minimum sum-of-products expressions for the function G = ∑m (0, 2, 3, 4, 5, 7).*

Solution: Given, G = ∑m (0, 2, 3, 4, 5, 7);

bc \ a	0	1
00	1	1
01		1
11	1	1
10	1	

G

bc \ a	0	1
00	1	1
01		1
11	1	1
10	1	

G

G = ________

G = ________

FOUR-VARIABLE KARNAUGH MAPS:

The following Figure shows the location of minterms on a four-variable map & plot of four-variable expression $f(a,b,c,d) = acd + a'b + d'$ on a Karnaugh map:

f	$\bar{a}\bar{b}$	$\bar{a}b$	ab	$a\bar{b}$
$\bar{c}\bar{d}$	0	4	12	8
$\bar{c}$d	1	5	13	9
cd	3	7	15	11
$c\bar{d}$	2	6	14	10

f	00	01	11	10
00				
01				
11				
10				

f (a, b, c, d) =

Example: *Write the Karnaugh map for* (a) $f = x'z' + wxy + x'y$

(b) $Y = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + AB\bar{C}$

Solution:

f	$\bar{w}\bar{x}$	$\bar{w}x$	wx	$w\bar{x}$
$\bar{y}\bar{z}$				
$\bar{y}z$				
yz				
$y\bar{z}$				

Y	0	1
00		
01		
11		
10		

f (w, x, y, z) =

Y (A, B, C) =

Example: *Write the Karnaugh map for (a) f1 = = ∑m (3, 4, 5, 6, 7, 9, 12, 13); (b) f2 = = ∑m (2, 3, 5, 7, 8, 10, 11, 15).*

Solution: (a) Given, f1 = = ∑m (3, 4, 5, 6, 7, 9, 12, 13) & f2 = = ∑m (2, 3, 5, 7, 8, 10, 11, 15);

cd \ ab	00	01	11	10
00		1	1	
01		1	1	1
11	1	1		
10		1		

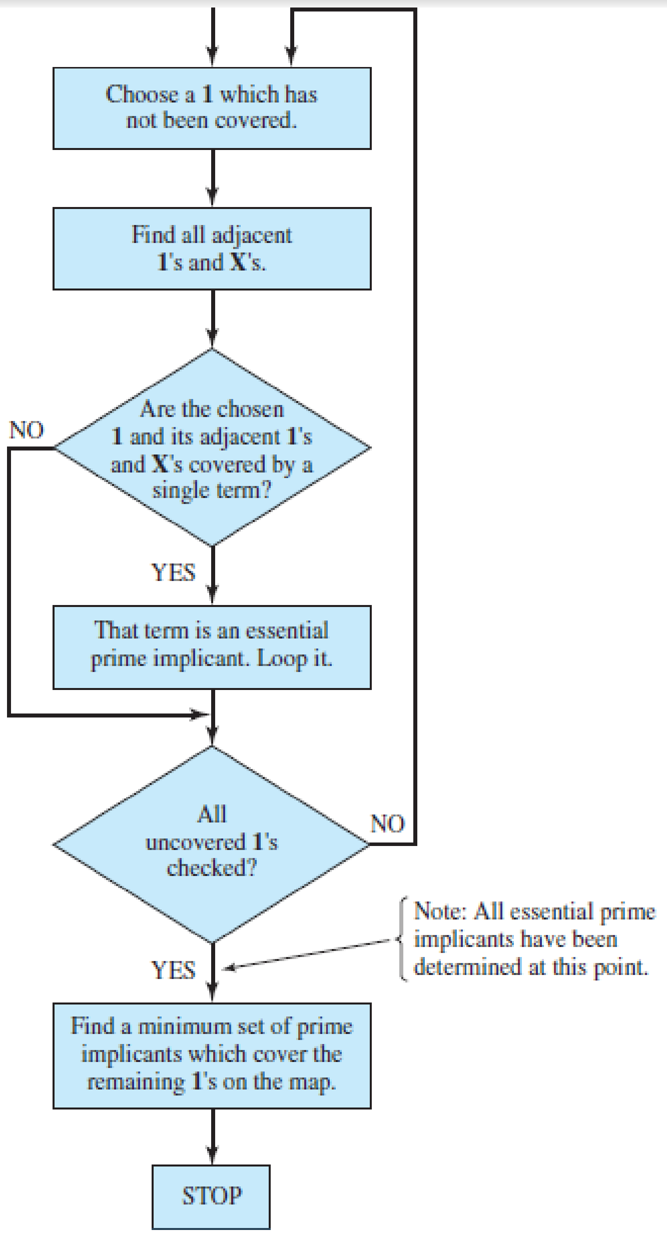
cd \ ab	00	01	11	10
00				1
01		1		
11	1	1	1	1
10	1			1

$f_1 =$ ______________________________ $f_2 =$ ______________________________

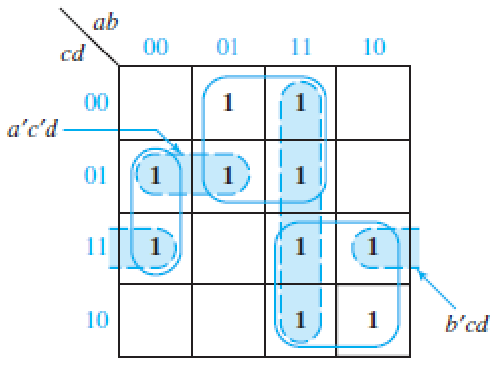
DETERMINATION OF MINIMUM EXPRESSIONS USING ESSENTIAL PRIME IMPLICANTS:

Any single 1 or any group of 1's which can be combined together on a map of the function F represents a product term which is called an implicant of F. Several implicants of F MAY BE POSSIBLE. A product term implicant is called a *prime implicant* if it cannot be combined with another term to eliminate a variable.

The following Figure shows the flowchart for determining a Minimum Sum of Products using a Karnaugh Map with an Example.


Choose a 1 which has not been covered.
Find all adjacent 1's and X's.
Are the chosen 1 and its adjacent 1's and X's covered by a single term?
NO
YES
That term is an essential prime implicant. Loop it.
All uncovered 1's checked?
NO
YES
Note: All essential prime implicants have been determined at this point.
Find a minimum set of prime implicants which cover the remaining 1's on the map.
STOP

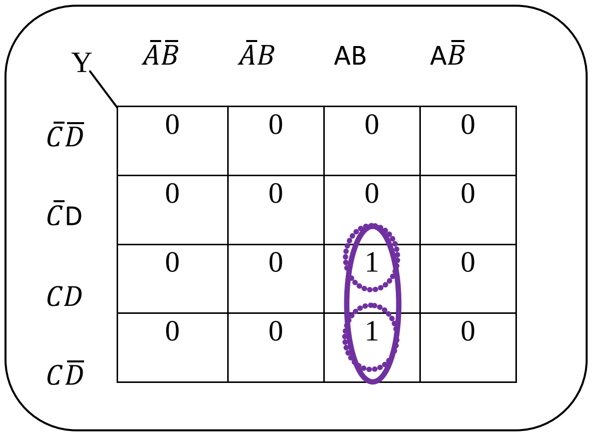
Minimum solution: $F = a'b'd + bc' + ac$
All prime implicants: $a'b'd$, bc', ac, $a'c'd$, ab, $b'cd$



1. Choose a minterm (a 1) which has not yet been covered.
2. Find all 1’s and X’s adjacent to that minterm (Check the n adjacent squares on an n-variable map.
3. If a single term covers the minterm and all of the adjacent 1’s and X’s, then that term is an essential prime implicant, so select that term. (Note that don’t-care terms are treated like 1’s in steps 2 and 3 but not in step 1.)
4. Repeat steps 1, 2, and 3 until all essential prime implicants have been chosen.
5. Find a minimum set of prime implicants which cover the remaining 1’s on the map. (If there is more than one such set, choose a set with a minimum number of literals.)

PAIRS, QUADS, AND OCTETS:

Pairs: The following K-map contains a pair of 1s that are horizontally adjacent. Two adjacent 1s, such as these are called a *pair*. *A pair eliminates one variable and its complement*.

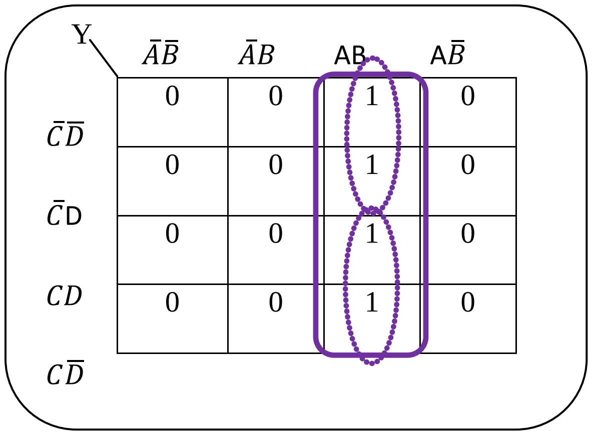


Y	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	0	0	0	0
CD	0	0	1	0
$C\bar{D}$	0	0	1	0

The sum-of-product equation is:

$Y = ABCD + ABCD' = ABC(D + D') = ABC$

Quad: A *quad* is a group of four 1s that are horizontally or vertically adjacent. *A quad eliminates two variables and their complements*.

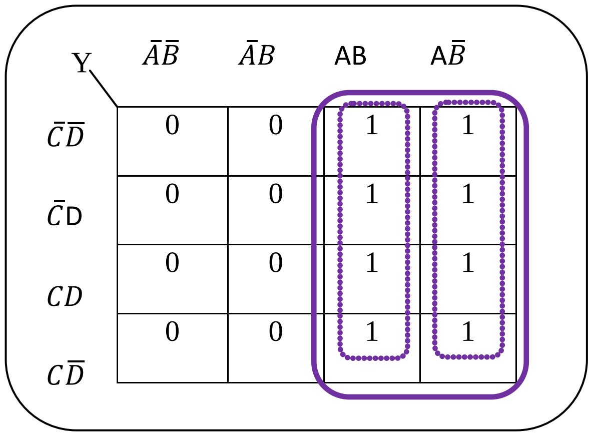


Y	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	1	0
$\bar{C}D$	0	0	1	0
CD	0	0	1	0
$C\bar{D}$	0	0	1	0

The sum-of-product equation is:

$Y = ABC' + ABC = AB(C + C') = AB$

The Octet: The *octet* is a group of eight 1s, as shown in the following Fig. *An octet eliminates three variables and their complements*.



Y	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	1	1
$\bar{C}D$	0	0	1	1
CD	0	0	1	1
$C\bar{D}$	0	0	1	1

The sum-of-product equation is:

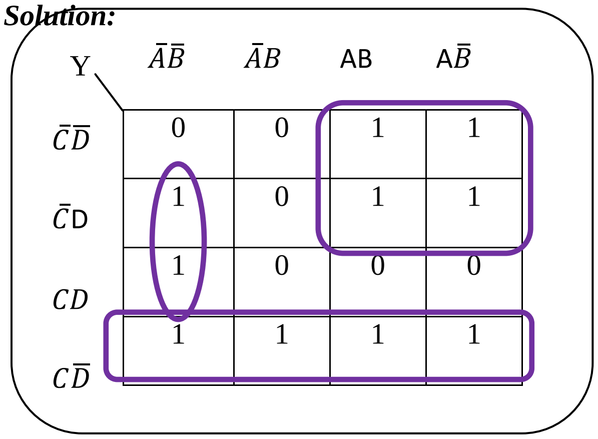
$Y = AB + AB' = A(B + B') = A$

KARNAUGH SIMPLIFICATIONS:

A pair eliminates one variable and its complement. A quad eliminates two variables and their complements. An octet eliminates three variables and their complements. Because of this, after drawing the K-map, first encircle the octets, then the quads, and finally the pairs, to get highest simplification.

Example: *Using K-map, simplify; Y = ∑m (1, 2, 3, 6, 8, 9, 10, 12, 13, 14).*

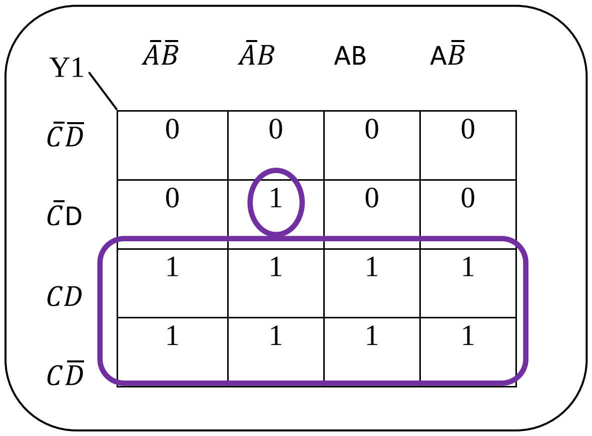
Solution:



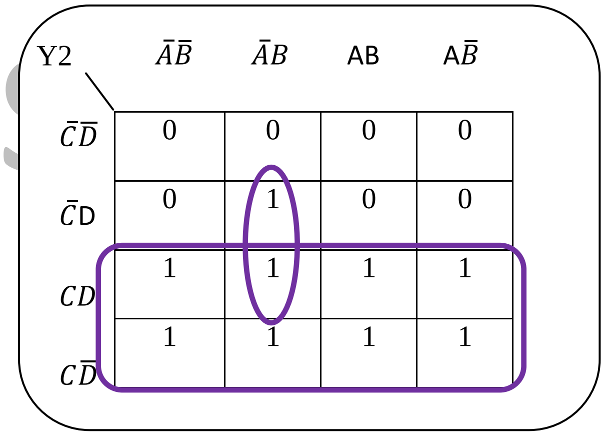
Y	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	1	1
$\bar{C}D$	1	0	1	1
CD	1	0	0	0
$C\bar{D}$	1	1	1	1

$$Y = AC' + CD' + A'B'D$$

Overlapping Groups: Always overlap groups.



Y1	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	0	1	0	0
CD	1	1	1	1
$C\bar{D}$	1	1	1	1

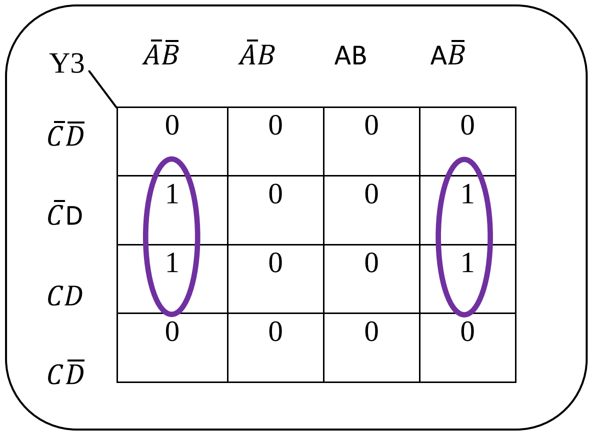


Y2	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	0	1	0	0
CD	1	1	1	1
$C\bar{D}$	1	1	1	1

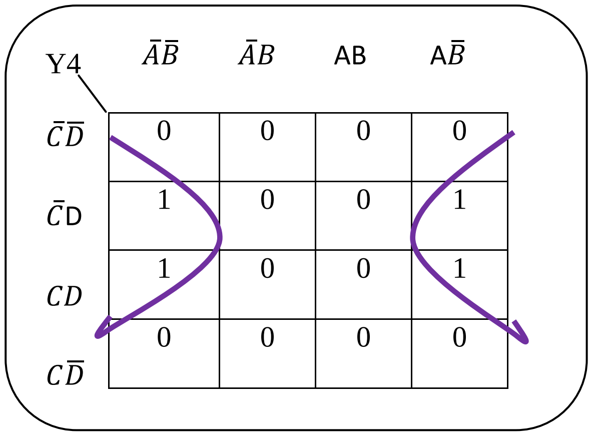
Y1 =

Y2 =

Rolling the Map:



Y3	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	1	0	0	1
CD	1	0	0	1
$C\bar{D}$	0	0	0	0

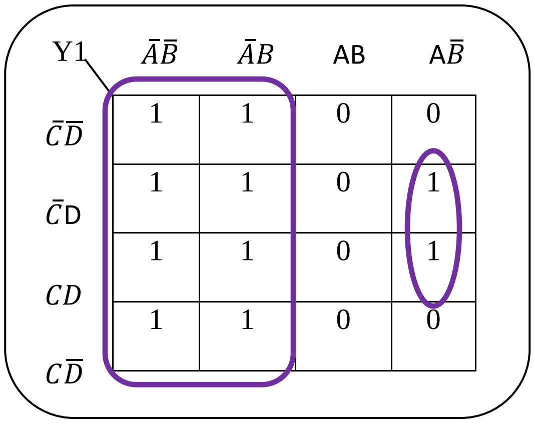


Y4	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	1	0	0	1
CD	1	0	0	1
$C\bar{D}$	0	0	0	0

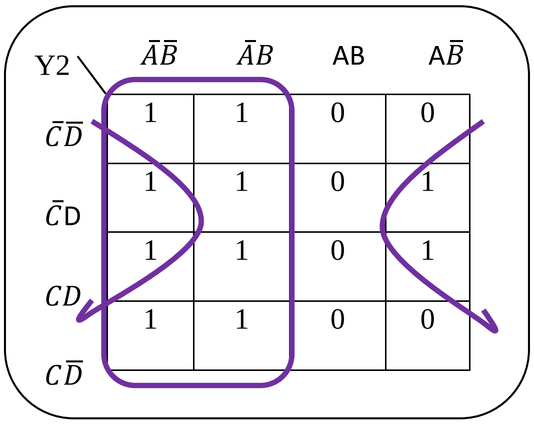
Y3 =

Y4 =

Rolling and Overlapping:



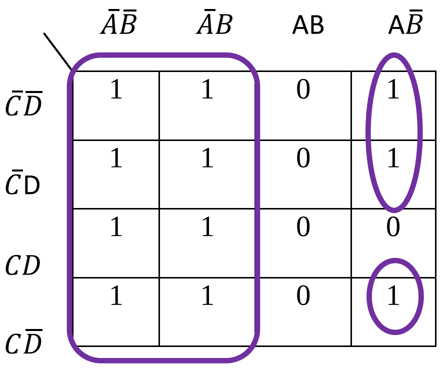
Y1	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	0	0
$\bar{C}D$	1	1	0	1
CD	1	1	0	1
$C\bar{D}$	1	1	0	0



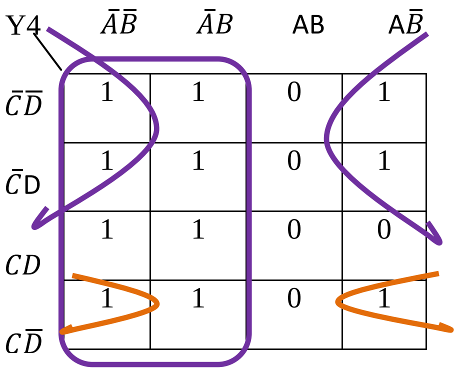
Y2	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	0	0
$\bar{C}D$	1	1	0	1
CD	1	1	0	1
$C\bar{D}$	1	1	0	0

Y1 =

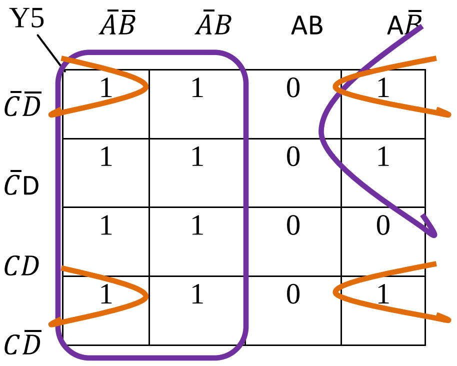
Y2 =



	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	0	1
$\bar{C}D$	1	1	0	1
CD	1	1	0	0
$C\bar{D}$	1	1	0	1



Y4	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	0	1
$\bar{C}D$	1	1	0	1
CD	1	1	0	0
$C\bar{D}$	1	1	0	1



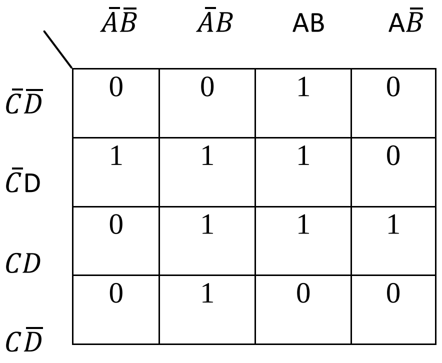
Y5	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	0	1
$\bar{C}D$	1	1	0	1
CD	1	1	0	0
$C\bar{D}$	1	1	0	1

Y3 =

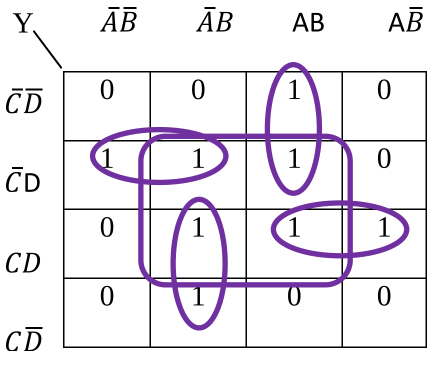
Y4 =

Y5 =

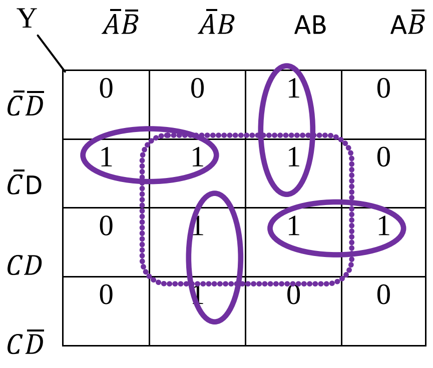
Eliminating Redundant Groups: After encircling groups, eliminate any *redundant groups*. This is a group whose 1s are already used by other groups.



	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	1	0
$\bar{C}D$	1	1	1	0
CD	0	1	1	1
$C\bar{D}$	0	1	0	0



Y	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	1	0
$\bar{C}D$	1	1	1	0
CD	0	1	1	1
$C\bar{D}$	0	1	0	0



Y	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	1	0
$\bar{C}D$	1	1	1	0
CD	0	1	1	1
$C\bar{D}$	0	1	0	0

Homework:

1] Determine the minimum sum-of-products for –

a) f1 (a, b, c) = $\sum$(1, 3, 4, 5, 6, 7)

b) f2 (a, b, c) = Π (2, 4, 7)

c) f3 (a, b, c, d) = $b'c'd' + bcd + acd' + a'b'c + a'bc'd$

2] Determine the minimum product-of-sums for –

a) f1 (a, b, c) = $\sum$ (0, 1, 2,3, 4, 6, 7)

b) f2 (a, b, c) = Π (1, 4, 5)

c) f3 (a, b, c, d) = $b'c'd' + bcd + acd' + a'b'c + a'bc'd$

3] Solve for the simplified Boolean expression using K-Map:

a) $f1\,(a,b,c,d) = \bar{a}\bar{c}d + \bar{a}cd + \bar{b}\bar{c}\bar{d} + a\bar{b}c + \bar{a}\bar{b}c\bar{d}$

b) $f2\,(a,b,c,d) = (a + b + \bar{d})(\bar{a} + b + \bar{d})(a + \bar{b} + \bar{c} + d)(\bar{a} + \bar{b} + \bar{c} + \bar{d})(\bar{a} + \bar{b} + \bar{c} + d)$

4] Find the minimum sum-of-products for –

(a) f1 (a, b, c) = m0 + m2 + m5 + m6

(b) f2 (d, e, f) = $\sum$m (0, 1, 2, 4)

(c) f3 (r, s, t) = $rt' + r's' + r's$)

(d) f4 (x, y, z) = M0 . M5

5] Design a 3-input, 1-output, minimal two-level gate combinational circuit; which has an output equal to 1 when majority of its inputs are at logic 1, and has output 0 when majority of inputs are at logic 0.

6] Design a minimal sum and minimal product combinational gate circuit to generate the odd parity bit for an 8421 BCD code.

QUINE McCLUSKEY (QM) METHOD

The Karnaugh map method is an effective way to simplify switching functions which have a small number of variables. When the number of variables is large or if several functions must be simplified, the use of a digital computer is desirable.

The *Quine-McCluskey* method provides a systematic simplification procedure which can be readily programmed for a digital computer. The Quine-McCluskey method reduces the minterm expansion (standard sum-of-products form) of a function to obtain a minimum sum of products.

DETERMINATION OF PRIME IMPLICANTS:

- ✓ In order to apply the Quine-McCluskey method to determine a minimum sum-of-products expression for a function, the function must be given as a sum of minterms.
- ✓ In the first part of the Quine-McCluskey method, all of the prime implicants of a function are systematically formed by combining minterms.

- ✓ Two minterms will combine if they differ in exactly one variable. The examples given below show both the binary notation and its algebraic equivalent.

$$AB'CD' + AB'CD = AB'C$$

$$\underbrace{1\ 0\ 1}_{X}\ \underbrace{0}_{Y} + \underbrace{1\ 0\ 1}_{X}\ \underbrace{1}_{Y'} = \underbrace{1\ 0\ 1}_{X}\ - \text{ (the dash indicates a missing variable)}$$

$$A'BC'D + A'BCD' \text{ (will not combine)}$$

$$0\ 1\ 0\ 1 + 0\ 1\ 1\ 0 \text{ (will not combine)}$$

- ✓ In order to find all of the prime implicants, all possible pairs of minterms should be compared and combined whenever possible. To reduce the required number of comparisons, the binary minterms are sorted into groups according to the number of 1's in each term.

Now, function; *f(a, b, c, d)* = ∑*m (0, 1, 2, 5, 6, 7, 8, 9, 10, 14)* can be represented by following list of minterms:

group 0	0	0000
group 1	1	0001
	2	0010
	8	1000
group 2	5	0101
	6	0110
	9	1001
	10	1010
group 3	7	0111
	14	1110

- ✓ In this list, the term in group 0 has zero 1's, the terms in group 1 have one 1, those in group 2 have two 1's, and those in group 3 have three 1's.
- ✓ Two terms can be combined if they differ in exactly one variable. Only terms in adjacent groups must be compared.
- ✓ First, we will compare the term in group 0 with all of the terms in group 1.Terms 0000 and 0001 can be combined to eliminate the fourth variable, which yields 000– ($a'b'c'$).
- ✓ Similarly, 0 and 2 combine to form 00–0 ($a'b'd'$), and 0 and 8 combine to form –000 ($b'c'd'$). The resulting terms are listed in Column II of the following Table.
- ✓ Whenever two terms combine, the corresponding decimal numbers differ by a power of 2 (1, 2, 4, 8, etc.).
- ✓ Since the comparison of group 0 with groups 2 and 3 is unnecessary, we proceed to compare terms in groups 1 and 2. Comparing term 1 with all terms in group 2, we find that it combines with 5 and 9 but not with 6 or 10. Similarly, term 2 combines only with 6 and 10, and term 8 only with 9 and 10. The resulting terms are listed in Column 2.
- ✓ Each time a term is combined with another term, it is checked off. Also note that, a term may be used more than once. Even though two terms have already been combined with other terms, they still must be compared and combined if possible.
- ✓ At this stage, we may generate redundant terms, but these redundant terms will be eliminated later.

✓ We finish with Column 1 by comparing terms in groups 2 and 3. New terms are formed by combining terms 5 and 7, 6 and 7, 6 and 14, and 10 and 14.

a	b	c	d	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

<u>Column 1</u>	<u>Column 2</u>	<u>Column 3</u>
abcd	abcd	abcd

✓ Note that the terms in Column 2 have been divided into groups. In order to combine two terms, the terms must have the same variables, and the terms must differ in exactly one of these variables. Thus, it is necessary only to compare terms which have dashes (missing variables) in corresponding places and which differ by exactly one in the number of 1's.

✓ Terms in the first group in Column 2 need only be compared with terms in the second group which have dashes in the same places. Term 000– (0, 1) combines only with term 100– (8, 9) to yield –00– ($b'c'$).

✓ The resulting term is listed in Column 3 along with the designation 0, 1, 8, 9 to indicate that it was formed by combining minterms 0, 1, 8, and 9.

✓ Term (0, 2) combines only with (8, 10), and term (0, 8) combines with both (1, 9) and (2, 10).

✓ Again, the terms which have been combined are checked off. Comparing terms from the second and third groups in Column 2, we find that (2,6) combines with (10, 14), and (2, 10) combines with (6,14).

✓ Note that there are three pairs of duplicate terms in Column 3. These duplicate terms were formed in each case by combining the same set of four minterms in a different order.

- ✓ After deleting the duplicate terms, we compare terms from the two groups in Column 3. Because no further combination is possible, the process terminates.

- ✓ In general, we would keep comparing terms and forming new groups of terms and new columns until no more terms could be combined. The terms which have not been checked off because they cannot be combined with other terms are called *prime implicants*. Because every minterm has been included in at least one of the prime implicants, the function is equal to the sum of its prime implicants. In this example we have;

$$f = \underset{(1,5)}{a'c'd} + \underset{(5,7)}{a'bd} + \underset{(6,7)}{a'bc} + \underset{(0,1,8,9)}{b'c'} + \underset{(0,2,8,10)}{b'd'} + \underset{(2,6,10,14)}{cd'}$$

Definition:

- ✓ Given a function F of n variables, a product term P is an ***implicant*** of F iff for every combination of values of the n variables for which P = 1, F is also equal to 1.
- ✓ A ***prime implicant*** of a function F is a product term implicant which is no longer an implicant if any literal is deleted from it.

Consider an Example:

$$F(a,b,c) = a'b'c' + ab'c' + ab'c + abc = b'c' + ac$$

- o In the above function, the implicant $a'b'c'$ is not a prime implicant because a can be eliminated, and the resulting term $b'c'$ is still an implicant of F. The implicants $b'c'$ and ac are prime implicants because if we delete a literal from either term, the term will no longer be an implicant of F.

The Quine-McCluskey method, as previously illustrated, finds all of the product term implicants of a function. The implicants which are nonprime are checked off in the process of combining terms, so that the remaining terms are prime implicants. Any nonprime term in a sum-of-products expression can thus be replaced with a prime implicant, which reduces the number of literals and simplifies the expression.

THE PRIME IMPLICANT CHART:

- ✓ The second part of the Quine-McCluskey method employs a prime implicant chart to select a minimum set of prime implicants. The minterms of the function are listed across the top of the chart, and the prime implicants are listed down the side. A prime implicant is equal to a sum of minterms, and the prime implicant is said to cover these minterms. If a prime implicant covers a given minterm, an ***X*** is placed at the intersection of the corresponding row and column. The following Table shows the prime implicant. All of the prime implicants (terms which have not been checked off in the above Table) are listed on the left.

-	0	1	2	5	6	7	8	9	10	14
(0, 1, 8, 9) $(b'c')$										
(0, 2, 8, 10) $(b'd')$										
(2, 6, 10, 14) (cd')										
(1, 5) $(a'c'd)$										
(5, 7) $(a'bd)$										
(6, 7) $(a'bc)$										

- ✓ In the first row, *X*'s are placed in columns 0, 1, 8, and 9, because prime implicant $b'c'$ was formed from the sum of minterms 0, 1, 8, and 9. Similarly, the all other *X*'s are placed.
- ✓ If a minterm is covered by only one prime implicant, then that prime implicant is called an *essential prime implicant* and must be included in the minimum sum of products. Essential prime implicants are easy to find using the prime implicant chart. If a given column contains only one *X*, then the corresponding row is an essential prime implicant. In the above Table, columns 9 and 14 each contain one X, so prime implicants $b'c'$ and cd' are essential.
- ✓ Each time a prime implicant is selected for inclusion in the minimum sum, the corresponding row should be crossed out. After doing this, the columns which correspond to all minterms covered by that prime implicant should also be crossed out.
- ✓ A minimum set of prime implicants must now be chosen to cover the remaining columns. In this example, the resulting minimum sum of products is –

$$f = b'c' + cd' + a'bd$$

Example: *Solve using QM method: F= ∑m (0, 1, 2, 5, 6, 7).*

Solution:

a	b	c	F
1	0	0	
1	0	1	
1	1	0	
1	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Column 1 abc	Column 2 abc

The following Table shows the resulting prime implicants chart:

-	0	1	2	5	6	7

Therefore, ***F*** *=*

NOTE: A prime implicant chart which has two or more *X*'s in every column is called a *cyclic prime implicant chart*.

Example: *Solve, using Quine Mc-Cluskey method & K-Map method:* $F = \sum m\ (0, 1, 2, 8, 10, 11, 14, 15)$.

Solution:

Quine Mc-Clusky method:

A	B	C	D	F
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Stage 1 ABCD	Stage 2 ABCD	Stage 3 ABCD

-	0	1	2	8	10	11	14	15
(0, 1) $(A'B'C')$								
(0, 2, 8, 10) $(B'D')$								
(10, 11, 14, 15) (AC)								

Therefore, ***F*** = $AC + B'D' + A'B'C'$

K-Map Method:

F	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$				
$\bar{C}D$				
CD				
$C\bar{D}$				

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

Homework: *Using Quine-McClusky method, simplify;*

a) $Y = \sum m\ (0, 1, 2, 3, 10, 11, 12, 13, 14, 15)$

b) $Y = \sum m\ (2, 6, 7)$.

PATRICK'S METHOD:

- ✓ Petrick's method is a technique for determining all minimum sum-of-products solutions from a prime implicant chart. The example discussed above has two minimum solutions. As the number of variables increases, the number of prime implicants and the complexity of the prime implicant chart may increase significantly. In such cases, a large amount of trial and error may be required to find the minimum solution(s).
- ✓ Petrick's method is a more systematic way of finding all minimum solutions from a prime implicant chart than the method used previously. Before applying Petrick's method, all essential prime implicants and the minterms they cover should be removed from the chart.

Consider the following Table:

- ✓ First, we will label the rows of the table P1, P2, P3, etc. We will form a logic function, P, which is true when all of the minterms in the chart have been covered. Let P1 be a logic variable which is true when the prime implicant in row P1 is included in the solution, P2 be a logic variable which is true when the prime implicant in row P2 is included in the solution, etc.
- ✓ Since, column 0 has X's in rows P1 and P2, we must choose row P1 or P2 in order to cover minterm 0. Therefore, the expression (P1+P2) must be true.
- ✓ In order to cover minterm 1, we must choose row P1 or P3; therefore, (P2+P3) must be true. In order to cover minterm 2, (P2+P4) must be true.
- ✓ Similarly, in order to cover minterms 5, 6, and 7, the expressions (P3+P5), (P4+P6) and (P5+P6) must be true.
- ✓ Since we must cover all of the minterms, the following function must be true:
- ✓ The next step is to reduce P to a minimum sum-of-products. This is easy because there are no complements. First, we multiply out, using $(X+Y)(X+Z)=X+YZ$ and the ordinary distributive law:
- ✓ Next we use $X+XY=X$ to eliminate redundant terms from P, which gives;
- ✓ Because P must be true (P = 1) in order to cover all of the minterms, we can translate the equation back into words as follows. In order to cover all of the minterms, we must choose rows P1 and P4 and P5, or rows P1 and P2 and P5 and P6, or . . . or rows P2 and P3 and P6.
- ✓ Although there are five possible solutions, only two of these have the minimum number of rows. Thus, the two solutions with the minimum number of prime implicants are obtained by choosing rows P1, P4, and P5 or rows P2, P3, and P6.
- ✓ Thus; $F=a'b'+bc'+ac$ or F = a'c' + b'c + ab are two minimum solutions.

SIMPLIFICATION OF INCOMPLETELY SPECIFIED FUNCTIONS:

In some digital systems, certain input conditions never occur during normal operation; therefore, the corresponding output never appears. Since the output never appears, it is indicated by an *X* in the truth table. The *X* is called a *don't-care condition*.

Remember these points about don't-care conditions:

1. Given the truth table, draw the K-map and transfer 0s, 1s, and don't-care terms.
2. Encircle the actual 1s on the K-map in the largest groups you can find treating don't cares as 1s.
3. After the actual1s have been included in the groups, disregard the remaining don't cares by visualizing them as 0s.

Example: Consider the following truth table with don't care conditions for all the inputs from 1010 to 1111.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$				
$\bar{C}$D				
CD				
$C\bar{D}$				

Y =

Problem: *What is the simplest logic circuit for –*

a) *Y1 = F (A, B, C, D) = ∑m (0) + ∑d (8, 9, 10, 11, 14, 15)*

b) *Y2 = F (A, B, C, D) = ∑m (0) + ∑d (12, 13, 14, 15)*

c) *Y3 = F (A, B, C, D) = ∑m (7) + ∑d (10, 11, 12, 13, 14, 15).*

Solution:

(a)

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$				
$\bar{C}$D				
CD				
$C\bar{D}$				

Therefore, Y1 = $B'C'D'$

(b)

Y2	$\bar{A}\bar{B}$	$\bar{A}B$	AB	A$\bar{B}$
$\bar{C}\bar{D}$				
$\bar{C}$D				
CD				
$C\bar{D}$				

Therefore, $Y2 = A'B'C'D'$

(c)

Y3	$\bar{A}\bar{B}$	$\bar{A}B$	AB	A$\bar{B}$
$\bar{C}\bar{D}$				
$\bar{C}$D				
CD				
$C\bar{D}$				

Therefore, $Y3 = BCD$

Don't-Care Conditions in Quine McCluskey Method:

- In the process of finding the prime implicants, we will treat the don't-care terms as if they were required minterms. In this way, they can be combined with other minterms to eliminate as many literals as possible. When forming the prime implicant chart, the don't-cares are not listed at the top.
- This way, when the prime implicant chart is solved, all of the required minterms will be covered by one of the selected prime implicants. However, the don't-care terms are not included in the final solution unless they have been used in the process of forming one of the selected prime implicants.

Homework: *Using Quine-McCluskey method (same questions can be asked to solve by using Patrick's method also), simplify;*

a) *f (a, b, c, d) = ∑m (3, 4, 5, 7, 10, 12, 14, 15) + ∑d (2)*
b) *f (a, b, c, d) = ∑m (1, 5, 7, 9, 11, 12, 14, 15)*
c) *f (a, b, c, d) = ∑m (0, 1,0 3, 5, 6, 7, 8, 10, 14, 15)*
d) *f (a, b, c, d) = ∑m (1, 3, 4, 5, 6, 7, 10, 12, 13) + ∑d (2, 9, 5)*
e) *f(a, b, c, d) = ∑m (9, 12, 13, 15) + ∑d (1, 4, 5, 7, 81 11, 14).*

Example: *Solve using QM method: F (A, B, C, D) = ∑m (2, 3, 7, 9, 11, 13) + ∑d (1, 10, 15).*

Solution: The don't-care terms are treated like required minterms when finding the prime implicants:

A	B	C	D	F
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Column 1	Column 2	Column 3
ABCD	ABCD	ABCD

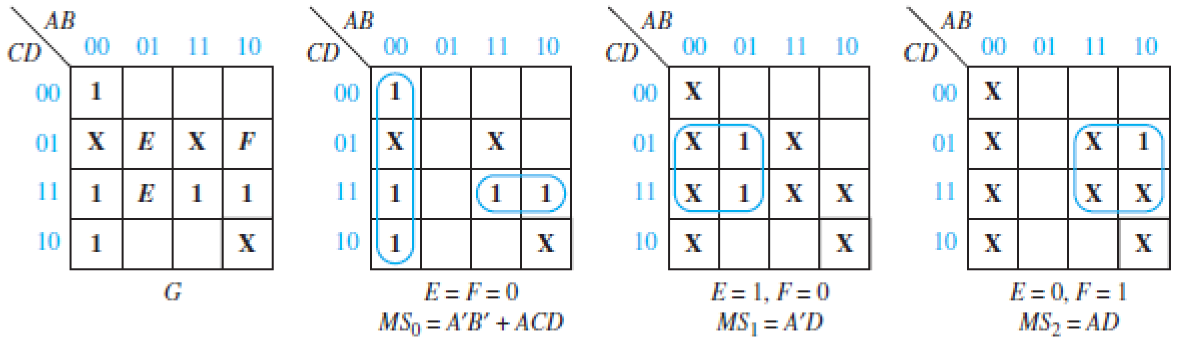
The don't-care columns are omitted when forming the prime implicant chart:

-	2	3	7	9	11	13

Therefore, F =

SIMPLIFICATION USING MAP-ENTERED VARIABLES:

Although the Quine-McCluskey method can be used with functions with a fairly large number of variables, it is not very efficient for functions that have many variables and relatively few terms. Some of these functions can be simplified by using a modification of the Karnaugh map method. By using map-entered variables, Karnaugh map techniques can be extended to simplify functions with more than four or five variables. The following Figure shows a four-variable map with two additional variables entered in the squares in the map.



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

G

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

$E = F = 0$
$MS_0 = A'B' + ACD$

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

$E = 1, F = 0$
$MS_1 = A'D$

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

$E = 0, F = 1$
$MS_2 = AD$

When *E* appears in a square, this means that if *E* = *1*, the corresponding minterm is present in the function G, and if *E* = *0*, the minterm is absent. Thus, the map represents the six-variable function;

$$G(A, B, C, D, E, F) = m_0 + m_2 + m_3 + Em_5 + Em_7 + Fm_9 + m_{11} + m_{15}$$
$$(+ \text{ don't-care terms})$$

Example: *Simplify Y (A, B, C) = ∑m (2, 6, 7) by using entered variable map method by taking –*

a) "C" as map entered variable

b) "A as map entered variables.

Solution: Let Y = ∑m (2, 6, 7)

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

A	B	Y_C
0	0	
0	1	
1	0	
1	1	

B	C	Y_A
0	0	
0	1	
1	0	
1	1	

(a)	A'	A
B'	0	0
B	C'	1

Y_C = BC' + AB

(b)	B'	B
C'	0	1
C	0	A

Y_A = AB + BC'

(c)	A'	A
B'	0	C
B	C'	1

Y = AB + BC'

Simplification is similar to K-map method. In Fig (a), C' is grouped with 1 to get a larger group as 1 can be written ac 1 = 1 + C'. Similarly, A is grouped with 1 in Fig (b).

Now, the product term representing each group is obtained by including map entered variable (MEV) in the group as an additional ANDed term.

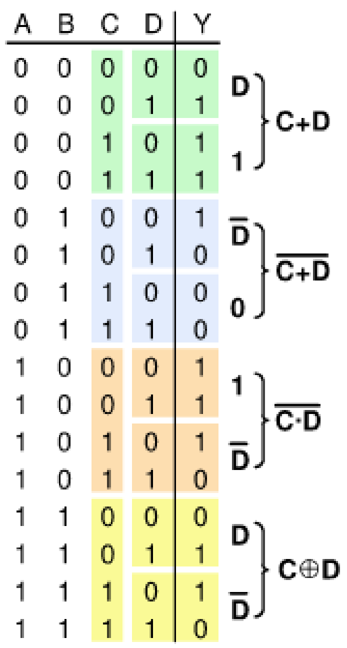
Hence, for Fig (a): $Y = B\bar{C} + AB$*. For Fig (b):* $Y = B\bar{C} + AB$*.*

Consider the EBM shown in Fig (c). This has only two product terms; and doesn't need a separate coverage for 1. This is because, one can write 1 = C + C', and C is included in one group and C' is included in other group.

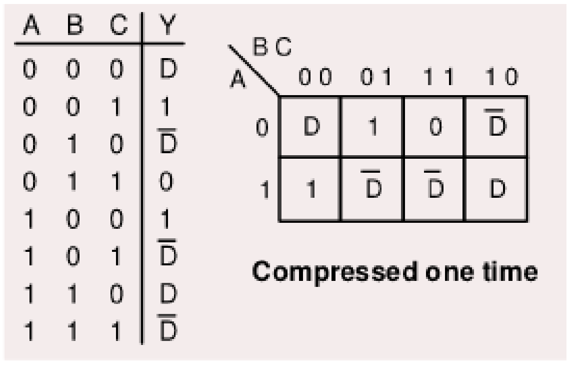
Example: *Simplify Y (A, B, C) = ∑m (1, 2, 3, 4, 8, 9, 10, 13, 14) by using entered variable map method by taking –* *a) "D" as map entered variable*

b) "C and D" as map entered variables.

Solution:



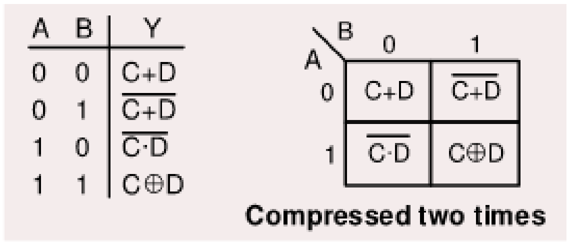
A	B	C	D	Y		
0	0	0	0	0	D	$C+D$
0	0	0	1	1		
0	0	1	0	1	1	
0	0	1	1	1		
0	1	0	0	1	$\bar{D}$	$\overline{C+D}$
0	1	0	1	0		
0	1	1	0	0	0	
0	1	1	1	0		
1	0	0	0	1	1	$\overline{C \cdot D}$
1	0	0	1	1		
1	0	1	0	1	$\bar{D}$	
1	0	1	1	0		
1	1	0	0	0	D	$C \oplus D$
1	1	0	1	1		
1	1	1	0	1	$\bar{D}$	
1	1	1	1	0		



A	B	C	Y
0	0	0	D
0	0	1	1
0	1	0	$\bar{D}$
0	1	1	0
1	0	0	1
1	0	1	$\bar{D}$
1	1	0	D
1	1	1	$\bar{D}$

A \ BC	00	01	11	10
0	D	1	0	$\bar{D}$
1	1	$\bar{D}$	$\bar{D}$	D

Compressed one time



A	B	Y
0	0	$C+D$
0	1	$\overline{C+D}$
1	0	$\overline{C \cdot D}$
1	1	$C \oplus D$

A \ B	0	1
0	$C+D$	$\overline{C+D}$
1	$\overline{C \cdot D}$	$C \oplus D$

Compressed two times

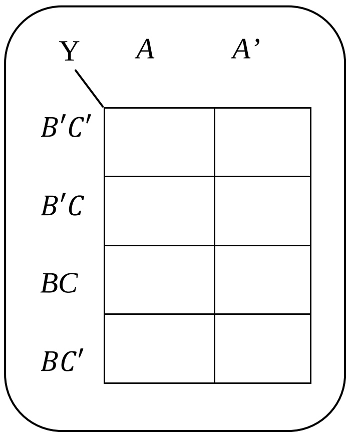
Example: *Solve by using (a) K-Map method & (b) MEV method taking "D" as map entered variable:*
$F(A,B,C,D) = A'B'C + A'BC + A'BC'D + ABCD + (AB'C)$; *where $AB'C$ is a don't-care term.*
Solution: Given $F(A,B,C,D) = A'B'C + A'BC + A'BC'D + ABCD + (AB'C)$
i.e., $F = A'B'C(D + D') + A'BC(D + D') + A'BC'D + ABCD + [AB'C(D + D')]$
or $F = A'B'C'D' + A'B'C'D + A'BCD' + A'BCD + ABCD + (AB'CD' + AB'CD)$
i.e., $F = \sum m\,(0, 1, 6, 7, 15) + \sum d\,(10, 11)$.

Y	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$				
$\bar{C}$D				
CD				
$C\bar{D}$				

Y(A, B, C, D) =

MEV method taking "D" as map entered variable:

A	B	C	D	Y	Y_D
0	0	0	0	1	1
0	0	0	1	1	
0	0	1	0	0	0
0	0	1	1	0	
0	1	0	0	0	0
0	1	0	1	0	
0	1	1	0	1	1
0	1	1	1	1	
1	0	0	0	0	0
1	0	0	1	0	
1	0	1	0	X	X
1	0	1	1	X	
1	1	0	0	0	0
1	1	0	1	0	
1	1	1	0	0	D
1	1	1	1	1	

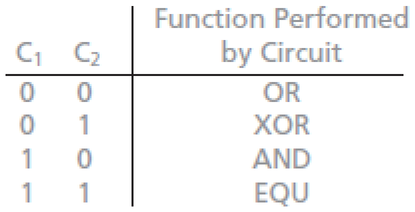


Y (A, B, C) =

Exercise:

a) *Design (a) Binary-to-Gray Code Converter, and (b) Gray-to-Binary Code Converter*

b) *A switching circuit has two control inputs (C1 and C2), two data inputs (X1 and X2), and one output (Z). The circuit performs one of the logic operations AND, OR, EQU (equivalence), or XOR (exclusive OR) on the two data inputs. The function performed depends on the control inputs:*



C_1	C_2	Function Performed by Circuit
0	0	OR
0	1	XOR
1	0	AND
1	1	EQU

(i) *Derive a truth table for Z*

(ii) *Use a Karnaugh Map to find minimum AN-OR Gate Circuit to realize Z.*

c) *A logic circuit realizing the function f has four inputs a, b, c, d. The three inputs a, b, and c are the binary representation of the digits 0 through 7 with a being the most significant bit. The input* *d is an odd-parity bit; that is, the value of d is such that a, b, c, and d always contains an odd number of 1's. (For example, the digit 1 is represented by abc = 001 and d = 0, and the digit 3 is represented by abcd = 0111.) The function f has value 1 if the input digit is a prime number. (A number is prime if it is divisible only by itself and 1; 1 is considered to be prime, and 0 is not.)*

a. *Draw a Karnaugh map for f*
b. *Find all prime implicants of f*
c. *Find all minimum sum of products for f*
d. *Find all prime implicants of f'*
e. *Find all minimum product of sums for f.*

Try these: *Design a minimal sum combinational circuit to –*

a) *Find the 9s complement of BCD numbers*
b) *Convert BCD to Excess-3*
c) *Multiply two 2-bit numbers*
d) *Output a 1 when an illegal BCD code occurs*
e) *Output the 2s complement of a 4-bit binary number.*