Logic design

First year/ computer science department/ education collage

Prepared by
Assist. Prof. :safana Hyder Abbas

Chapter one

-Number systems and codes-

Number systems :

In general any number N can be represented in the base (radix) R as show below :

$N_R = d_nR^n + d_{n-1} R^{n-1} + \ldots\ldots\ldots d_1R^1 + d_0R^0$

1-Decimal system (R= 10) :

This system uses (10) symbols (0 – 9)

Ex: $N_{10} = 2^3 3^2 6^1 8^0 = 2x10^3 + 3 \times 10^2 + 6x10^1 + 8x10^0 = 2000 + 300 + 60 + 8 = 2368$

2-Binary system (R = 2) :

It uses only two basic symbol (0,1) . It is the most suitable number system for digital circuits.

Ex: $N_2 = 1^4 1^3 0^2 0^1 1^0 = 1(2)^4 + 1(2)^3 + 0(2)^2 + 0(2)^1 + 1(2)^0 = 16 + 8+0+0+1 = (25)_D$

Ex: $1^3 0^2 0^1 1^0 = 1(2)^3 + 0(2)^2 + 0(2)^1 + 1(2)^0 = 8+0+0+1 = (9)_D$

3-Octal system (R= 8) :

It uses 8 symbols (0- 7)

Ex : $1^1 7^0 = 1(8)^1 + 7(8)^0 = 8+7 = (15)_D$

Ex: $7^3 0^2 4^1 5^0 = 7(8)^3 + o(8)^2 + 4(8)^1 + 5(8)^0 = 3584 + 0 + 32 + 5 = (3621)_D$

4- Hexadecimal system (R =16)

It uses 16 symbol :

(0,1,2,……..,9,A,B,C,D,E,F)

Ex: $1^1 A^0 = 1(16)^1 + A(16)^0 = 16+10 = (26)_D$

Ex: $3^2 9^1 5^0 = 3(16)^2 + 9(16)^1 + 5(16)^0 = 768 + 144 + 5 = (773)_D$

The following table gives the correspondence between the four number system :

Decimal	Binary	Octal	Hexadecimal
0	0	0	0
1	01	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	.	C
.	.	.	D
.	.	.	E
.	.	.	F
.	.	.	10
.	.	.	11
.	.	.	12

Continue the above table.

(Binary , octal and Hexadecimal) to Decimal :

For binary number :

Ex : $(11010 . 101)_{2} = 1(2)^{4}+1(2)^{3}+ 0(2)^{2}+ 1(2)^{1}+ 0(2)^{0}+ 1(2)^{-1}+ 0(2)^{-2}+1(2)^{-3} =$
$16+8+2+0.5+0.125 = (26.625)_{10}$

For octal number :

Ex: $(613.24)_{8} = 6(8)^{2} +1(8)^{1}+3(8)^{0}+2(8)^{-1} + 4(8)^{-2} =$
$384 + 8 + 3 + 0.25 + 0.0625 = (395.3125)_{10}$

For hexadecimal :

Ex: $(5A.E)_{16} =5(16)^{1}+ A(16)^{0}+ E(16)^{-1} =$
$5(16)+10(16)+ 14(16)^{-1} = 80+10+0.875 = (90.875)_{10}$

Decimal to (Binary , octal and Hexadecimal) :

1- To Binary :

Ex: $(353)_{10} = (?)_{2}$

		الباقي	
353	2	1	← **LSB**
176	2	0	
88	2	0	
44	2	0	
22	2	0	
11	2	1	
5	2	1	
2	2	0	
1	2	1	← **MSB**
0			

$(353)_{10} = (101100001)_{2}$

H.W.: $(49302)_{10} = (?)_{2}$
$(10010)_{10} = (?)_{2}$

EX: $(0.65625)_{10} = (?)_2$

0.65625
2 x
1 ← 1.31250

اول مرتبه بعد الفارزه

0.31250
2 x
0 ← 0.62500
2 x
1 ← 1.250
2 x
0 ← 0.50
2 x
1 ← 1.0

$(0.65625)_{10} = (0.10101)_2$

***H.W.*:** $(0.8134)_{10} = (?)_2$
$(741.528)_{10} = (?)_2$

2- to Octal :

EX : $(254.75)_{10} = (?)_8$

		الباقي
254	8	6 → **LSB**
31	8	7
3	8	3
0		

0.75
8 x
6 ← 6.00

$(254.75)_{10} = (376.6)_8$

H.W. : $(3070.446)_{10} = (?)_8$
$(7706.77)_{10} = (?)_8$

3- to Hexadecimal :

EX: $(567.1875)_{10} = (?)_{16}$

567	16	7	← **LSB**
35	16	3	
2	16	2	
0			

0.1875
16 x
3 ← 3.0000

$(567.1875)_{10} = (237.3)_{16}$

H.W. $(1005.36)_{10} = (?)_{16}$
$(3987.055)_{10} = (?)_{16}$

Binary , Octal and Hexadecimal conversion(fast method) :

EX:

1 2 7 5 4 3 Octal
1 0 1 0 1 1 1 1 0 1 1 0 0 0 1 1 Binary
A F 6 3 Hexadecimal

$=(44899)_{decimal}$ how !

EX:

2 1 4 4 3 5 Octal
1 0 0 0 1 1 0 0 1 0 0 0 1 1 1 0 1 Binary
1 1 9 1 D Hexadecimal

H.w:

$(1011100011)_B = (?)_0 = (?)_H$
$(AF31C)_H = (?)_O = (?)_B$
$(37012)_{0=} (?)_B = (?)_H$

Arithmetic operation :

1-Binary addition :

	Carry	sum
0 + 0 =		0
0 + 1 =		1
1 + 0 =		1
1 + 1 =	1	0

EX:

1		1 1	
11	100	111	110
+11	+ 10	+ 11	+100
110	110	1010	1010

2-Binary subtraction :

0 – 0 = 0
1 – 1 = 0
1 – 0 = 1
$^{10}\cancel{0}$ – 1 = 1 with borrow 1

EX:

		10	10
11	11	1$\cancel{01}$	11$\cancel{0}$
-01	-10	-011	-101
10	01	010	001

3-Binary multiplication :

0 x 0 = 0
0 x 1 = 0 the same manner as in Decimal
1 x 0 = 0
1 x 1 = 1

EX:

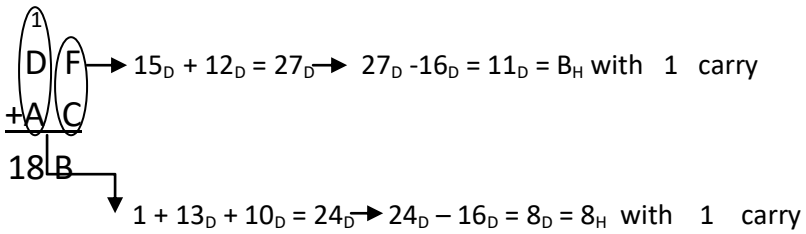
11	111
X 11	x 101
11	111
+ 11	+ 000
1001	111
	100011

Hexadecimal addition :

EX:

23	58	2B
+16	+22	+84
39_H	$7A_H$	AF_H

EX:

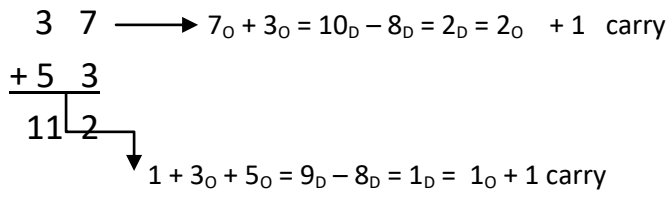


Octal addition :

EX:

14
+23
37_o

EX:



Complements :

Complements are used in digital computer for simplifying the subtraction operation and for logical manipulation . There are two types of complement for each base (R) system :

1-The R´s complement
2-the (R-1)´s complement

For binary number ⟶ 1´s and 2´s complement
For decimal number ⟶ 9´s and 10´s complement
For octal number ⟶ 7's and 8's complement
For hexadecimal number ⟶ 15's and 16's complement

The 1´s and 2´s complement :

The 1´s complement of a binary number is the no. we get when we change each (0) to (1) and each (1) to (0) (or subtracting each binary no. from 1)

EX: 1´s comp. of 1001 ⟶ 0110
1´s comp. of 110010 ⟶ 001101
2´s comp. = 1´s comp. + 1
2´s comp. of 1011 is 0100 + 1 = 0101
2´s comp. of 1110 is 0001 + 1= 0010

Using 2´s complement in subtraction :

Instead of subtraction a number , we can add it's 2's comp, and disregard the last carry.

EX: decimal

7 111 ⟶ 111
-5 -101 1´s ⟶ 010 2´s ⟶ 011
2 1+ 1 010 + ve. No.
011 X carry

EX: 13 1101 ⟶ 1101
-10 1010 1´s ⟶ 0101 2´s ⟶ 0110
3 1+ 1 0011 +ve. No.
0110 X carry

EX: 4 100 ⟶ 100
-7 -111 1´s ⟶ 000 2´s ⟶ 001+
-3 1+ 101
001 No carry ⟶ -ve. No.

So 101 ⟶ 100 ⟶ 011

Using 1´s complement in subtraction :

Instead of subtracting a number we add the 1´s complement of the number , the last carry is then added to the number to get the final answer .

EX: 7 111 → 111

-5 - 101 1´s → 010 +

2 carry → 1 001

+ ve. No. → 1 +

010

EX: 3 011 → 011

-5 101 1' s → 010 +

101

No carry → - ve. No.

101 → 010

4-Binary division :

The standard division format is:

$$\frac{\text{Dividend}}{\text{Divisor}} = \text{quotient}$$

The divisor can be subtracted from the dividend a number of times equal to the quotient.
For example in decimal:

21 ⟹ 21-7=14 1st

7

14-7=7 2nd

7-7=0 3rd so quotient= 3

In binary:

EX: 1100

100

100 ⟶ 0100 1's comp. 1011 2's comp. 1100

1100

1100 +

X 1 1000 q=1

1100+

X 1 0100 q=2

1100 +

X 1 0000 q=3 the result

EX: Divide 1010 by 101:

101 ⟶ 0101 1's comp. 1010 2's comp. 1011

1010

1011 +

X 1 0101 q=1

1011 +

X 1 0000 q=2 the result

Signed numbers:

There are three ways in which signed binary numbers may be expressed:

- Signed magnitude (SM)
- One's complement and
- Two's complement.

In an 8-bit word, signed magnitude representation places the absolute value of the number in the 7 bits to the right of the sign bit.

Ex: in **8-bit signed magnitude(SM),** positive 3 is: 00000011
Negative 3 is: 10000011

Ex: in 8-bit **one's complement**, positive 3 is: 00000011
Negative 3 is: 11111100

Ex: Adding 1 gives us -3 in **two's complement** form: 11111101

Ex: convert using SM method $\mathbf{(01011001)_2} = +(1 * 2^6 + 0 * 2^5 + 1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0)$

$= + (64 + 0 + 16 + 8 + 0 + 0 + 1)$

$= \mathbf{(+89)_{10}}$

Ex: convert using SM method $\mathbf{(10011100)_2} = - (0 * 2^6 + 0 * 2^5 + 1 * 2^4 + 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 0 * 2^0)$

$= - (0 + 0 + 16 + 8 + 4 + 0 + 0)$

$= \mathbf{(-28)_{10}}$

7´s and 8´s complements in octal :

7´s = 7 – each digit

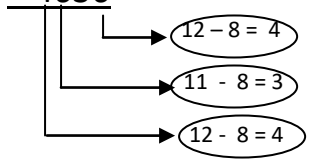
8´s = 7´s + 1

EX: 7777
- 2415
5362 is 7´s comp.
+1
5363 is 8´s comp.

EX: Perform 7526_8 - 3142_8 using 8's comp.:

7777
-3142
4635 + 1 = 4636

1 1
7526
+ 4636



1 4364 the result

H.W. Perform the following using 8's complement:

$545_8 – 14_8$ =

$6776_8 – 4337_8$ =

15´s and 16´s complements in hexadecimal :

EX: Find 15´s and 16´s comp. of (1 F A D)$_{16}$

15	15	15	15	
- 1	F	A	D	
E	0	5	2	← 15´s comp.
			1 +	
E	0	5	3	← 16´s comp.

EX: Perform A B E D – 1 F A D using 16's comp. :

```
  1 1
  A B E D
+ E 0 5 3
```

16 – 16 = 0

20 – 16 = 4

24 – 16 = 8

1 8 C 4 0 the result

H.W. :Perform the following using 16's complement :

F E E D$_{16}$ – D A F 3$_{16}$ = ANS: 23FA$_{16}$

9 8 A E$_{16}$ – 1 F E E$_{16}$ =

Codes :

The binary number is the most natural system , but people are familiar to the decimal system . one way to solve this conflict is to convert all input decimal numbers into binary numbers and then convert the binary results back to decimal for the human user to understand . However , it is also possible for the computer to perform arithmetic operations directly with decimal numbers provided they are placed in registers in a coded form . When decimal numbers are used for internal arithmetic computations , they are converted to a binary code with four bits per digit .

It is very important to understand difference between the conversion of decimal numbers into binary and the binary coding for decimal numbers .

BCD (binary – coded – decimal) :

These are codes that combine some of the features of both decimal and binary numbers. There are different types of BCD codes :

1- Excess-3 code :

It is an important BCD code . To encode decimal number to it's excess-3 , we add (3) to each decimal digit before converting to binary :

Decimal	ex-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100
10	0100 0011
11	0100 0100

Note : It is un weighted code .

← أكثر رقم نحتاجه هو 12

2- BCD 8421 code :

It is weighted code .

Decimal	BCD 8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001 0000
11	0001 0001

أكثر رقم نحتاجه هو 9 ←

3- Other 4-bit BCD codes :

Many other 4-bit codes exists , such as 7421 , 6311 , 5421 ,5311 , 5211 , 2421 …..
All are weighted codes .

EX: $(16)_D$ to 2421 code

1 6

0001 1100

$(75)_D$ to 5421 code

1010 1000

$(693)_D$ to 6311 code

1000 1100 0100

D	2421	6311	7421
0	0000	0000	0000
1	0001	0001	0001
2	0010	0011	0010
3	0011	0100	0011
4	0100	0101	0100
5	1011	0111	0101
6	1100	1000	0110
7	1101	1001	1000
8	1110	1011	1001
9	1111	1100	1010

Decimal	5421	5311
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0100
4	0100	0101
5	1000	1000
6	1001	1001
7	1010	1011
8	1011	1100
9	1100	1101

Gray code :

It is un weighted code the main characteristic of this code is that each gray number differs from the preceding number by single bit .

Decimal	Gray	Binary
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

EX:

Alpha numeric codes :

It is an assignment of bit combinations to the letters of the alpha bet , the decimal digit (0-9) , punctuation marks , and several special character such as # .

The most widely used of alpha numeric codes are :

1- EBCDIC (Extended Binary Coded Decimal Interchange Code).

2- ASCII (American Standard Code for Information Interchange).

The EBCDK code uses 8-bit to represent each symbols while the ASCII code use 7-bit code.

Parity method for error detection :

Even parity(ep) : makes the total no. of 1's even

Odd parity (op): makes the total no. of 1's odd

number	even P	odd P
0000	0	1
0001	1	0
0010	1	0
0011	0	1
0100	1	0
0101	0	1
0110	0	1
0111	1	0
1000	1	0
1001	0	1
1010	0	1
1011	1	0
1100	0	1
1101	1	0
1110	1	0
1111	0	1

EX: Check an even parity(ep) and odd parity(op) for the following numbers:

0101, 0001

For 0101 ep=0, op=1

For 0001 ep=1, op=0

Chapter two

-Boolean algebra and logic gates-

Boolean algebra :

Boolean variable is a quantity that is either (1) or (0) .

Boolean algebra is a mathematical method to manipulate Boolean variable in order to simplify the Boolean equation .

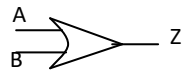
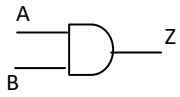
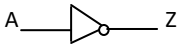
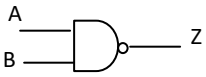
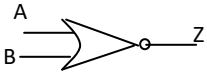
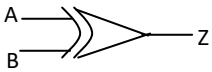
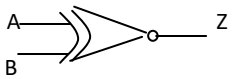
There are three basic operation in Boolean algebra :

1- OR (logical addition) (+)
2- AND (logical multiplication) (.)
3- NOT (logical inversion) ($\bar{}$)

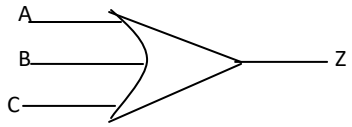
Gate :

It is a device with one output and two or more inputs . output is related to the input in Boolean (logic) equation .

In the following , the basic logic gates with their symbols , truth table and equation :

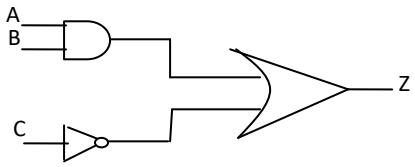
Gate	Symbol	Truth table	Equation
OR		AB | Z 00 | 0 01 | 1 10 | 1 11 | 1	Z=A+B
AND		AB | Z 00 | 0 01 | 0 10 | 0 11 | 1	Z=A.B
NOT		A | Z 0 | 1 1 | 0	$Z=\overline{A}$
NAND		AB | Z 00 | 1 01 | 1 10 | 1 11 | 0	$Z=\overline{A.B}$
NOR		AB | Z 00 | 1 01 | 0 10 | 0 11 | 0	$Z=\overline{A+B}$
EX_OR		AB | Z 00 | 0 01 | 1 10 | 1 11 | 0	$Z=A \oplus B$
EX_NOR		AB | Z 00 | 1 01 | 0 10 | 0 11 | 1	$Z=\overline{A \oplus B}$

EX: Describe the T.T. for three input OR- gate :



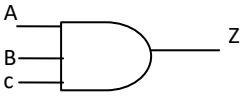
ABC	Z
000	0
001	1
010	1
011	1
100	1
101	1
110	1
111	1

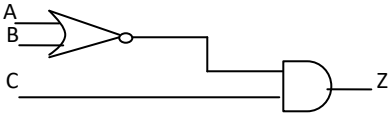
EX: Describe the T.T for the following logic circuit :

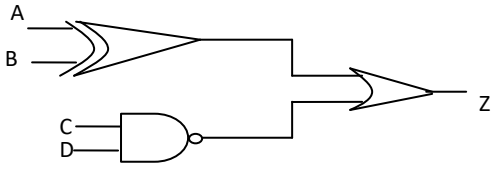


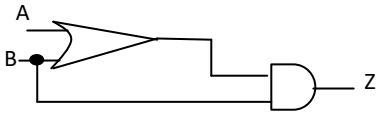
ABC	Z
000	1
001	0
010	1
011	0
100	1
101	0
110	1
111	1

<u>H.w.</u>: Describe the T.T for the following :









The basic rules and identities of Boolean algebra :

1- Identities :

a- $x.0=0$ **b-** $x+0=x$ **c-** $x.1=x$ **d-** $x+1=1$

e- $x.x=x$ **f-** $x+x=x$ **g-** $x.\overline{x}=0$ **h-** $x+\overline{x}=1$

i- $\overline{\overline{x}}=x$

2-Commutative laws :

a- $x.y=y.x$

b- $x+y=y+x$

3-Associative laws :

a- $(x+y)+z = x+(y+z) = x+y+z$

b- $(x.y).z = x.(y.z) = x.y.z$

4-Distributive laws :

a- $x.(y+z) = (x.y)+(x.z)$

b- $x+(y.z) = (x+y).(x+Z)$

5-Absorption laws :

a- $x+x.y = x$

b- $x+\overline{x}.y = x+y$

c- $x.(x+y) = x$

6-De Morgan´s theorem :

a- $\overline{x.y} = \overline{x}+\overline{y}$

b- $\overline{x+y} = \overline{x}.\overline{y}$

All of these Boolean theorems useful in simplifying a logic expression that is in reducing the no. of terms in the expression .

EX: Prove that $\overline{\overline{A}.B} = A+\overline{B}$

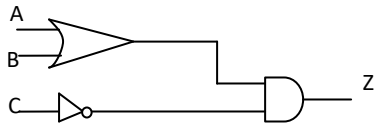
Sol: $\overline{\overline{A}.B} = \overline{\overline{A}}+\overline{B} = A+\overline{B}$

EX: Simplify the following :

$Z=A(A+B)$

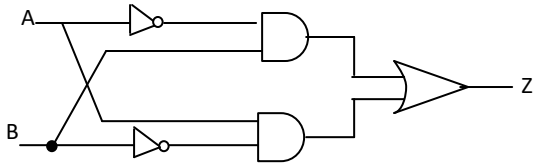
Sol: $Z = AA+AB = A+AB = A$

EX: write the Boolean exp. For the following circuit



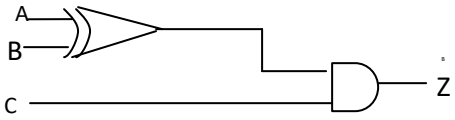
Sol: $Z = (A+B).\overline{C}$

EX: write the Boolean exp. For the following logic cct. :



Sol : $Z = \overline{A}B+A\overline{B} = A \oplus B$

EX: write Boolean exp. For the following logic cct. :



Sol: $Z = (\overline{A}B+A\overline{B}).C = (A \oplus B).C$

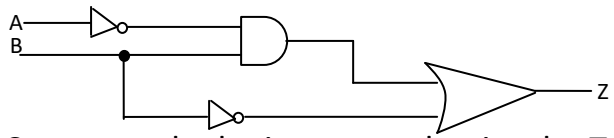
NOTE : The priorities of logic operation are :

( )
AND
OR

EX: Construct the logic circuit for the following Boolean exp. :

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

Sol:

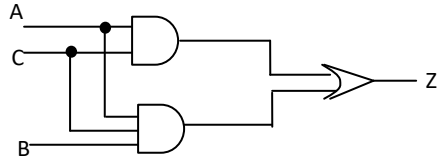


We need 2-input AND gate
2-input OR gate
2-inverters

EX: Construct the logic cct. , and write the T.T. for the following logic eq.

$Z = AC+ABC$

Sol:



ABC	Z
000	0
001	0
010	0
011	0
100	0
101	1
110	0
111	1

***Ex*:** simplify the following :

$A= \overline{x}\,\overline{y}\,\overline{z} + \overline{x}\,y\,\overline{z} + x\,\overline{y}\,\overline{z} + x\,y\,\overline{z} = \overline{x}\,\overline{z}\,(y+\overline{y}) + x\,\overline{z}\,(\overline{y}+y) = \overline{x}\,\overline{z} + x\,\overline{z} = \overline{z}\,(\overline{x}+x) = \overline{z}$

Ex: simplify the following :

$Z= AB + A(B+C) + B(B+C)$
$= AB + AB + AC + BB + BC$
$= AB + AC + B + BC$
$= AB + AC + B$
$= B + AC$

Ex: Simplify the following :

$Z = A\overline{B}C + \overline{A}BC + \overline{A}\overline{B}C$
$= A\overline{B}C + \overline{A}C(B+\overline{B})$
$= A\overline{B}C + \overline{A}C$
$=C\,(A\overline{B}+\overline{A})$
$=C\,(\overline{A}+\overline{B})$
$=C\overline{A}+C\overline{B}$

Ex: Simplify the following using Boolean Algebra

$F= A\{\, BC(A+B+C+D)\}$
$= ABC(A+B+C+D)$
$=ABCA +ABCB +ABCC +ABCD$
$=ABC+ABC+ABC+ABCD$
$=ABC+ABCD$
$=ABC(1+D)$
$= ABC$

EX: Simplify:

$F = A\overline{C} + ABC + AC\overline{D} + CD$
$= A(\overline{C}+BC) + C(A\overline{D}+D)$
$= A(\overline{C}+B)+C(A+D)$
$= A\overline{C} + AB + CA +CD$
$= A(\overline{C}+C) + AB + CD$
$= A + AB +CD$
$= A(1+B) + CD$
$= A + CD$

EX: Prove that :

$F = ABC + A\bar{B}C + AB\bar{C} = A(B+C)$

$= AC(B+\bar{B}) + AB\bar{C}$

$= AC + AB\bar{C}$

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

$= A(C+B)$

EX: Simplify :

$X = AB + ABC + \bar{A}B + A\bar{B}C$

$= AB(1+C) + \bar{A}B + A\bar{B}C$

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

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

$= B + \bar{B}AC$

$= B + AC$

EX: In a 3-input cct. The output is (1) if the majority of input is (1) , and otherwise , it is zero. White the T.T. for this cct. :

ABC	Z
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

Sum -of- product representation of logic function :

A SP expression is a product term or several product terms, logically added together e.g:

$F = A.B + A\bar{B}\bar{C} + BD + \ldots\ldots$

product (AND)

Derivation of sp :

1-construct the T.T.

2-construct a multiplication column of product of all inputs.

3-the desired expression is the sum of the product of all terms in which the output is 1 .

EX: For the following T.T. , write the logic function using sp method :

AB	Z
00	1
01	0
10	0
11	1

P terms	
$\bar{A}\bar{B}$	←
$\bar{A}B$	
$A\bar{B}$	
AB	←

$Z = \bar{A}\bar{B} + AB$

EX: For the following T.T. , write the logic function using sp method , then simplify it :

ABC	Z	P terms	min terms	
000	0	$\bar{A}\bar{B}\bar{C}$	m_0	
001	0	$\bar{A}\bar{B}C$	m_1	
010	0	$\bar{A}B\bar{C}$	m_2	
011	1	$\bar{A}BC$	m_3	←
100	0	$A\bar{B}\bar{C}$	m_4	
101	1	$A\bar{B}C$	m_5	←
110	1	$AB\bar{C}$	m_6	←
111	1	ABC	m_7	←

$Z = \bar{m}_3 + m_5 + m_6 + m_7$

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

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

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

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

$= CB + CA + AB\bar{C} = CB + A(C+B\bar{C})$

$= CB + A(C+B)$

$= CB + AC + AB$

Product -of- sum representation of logic function :

A PS is a sum term or several sum terms logically multiplied together e.g. :

$F = (A+B)(A+B+\bar{C})(\bar{A}+D)\ldots..$

Derivation of PS :

1-construct the T.T.

2-construct a sum column of sum of all inputs (0=uncomplement , 1=complement)

3-The desired output exp. Is the product of the sum of all terms in which the output is zero.

EX: For the following T.T. , write the logic function using PS method :

AB	Z	⟹	S. treams	Max terms
00	1		$(A+\bar{B})$	M_0
01	0		$(A+\bar{B})$	M_1 ←
10	0		$(\bar{A}+B)$	M_2 ←
11	0		$(\bar{A}+\bar{B})$	M_3 ←

$Z = M_1 . M_2 . M_3$

$= (A+\bar{B})(A+B)(\bar{A}+\bar{B})$

EX: Simplify the following function using SP and PS methods :

$F(A,B,C) = \pi(M_2 , M_3 , M_6)$

Sol:

ABC	Z	
000	1	m_0
001	1	m_1
010	0	M_2
011	0	M_3
100	1	m_4
101	1	m_5
110	0	M_6
111	1	m_7

1-By SP method :

$Z = m_0 + m_1 + m_4 + m_5 + m_7$

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

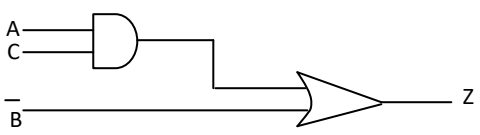
$= \bar{A}\bar{B}(\bar{C}+C) + \bar{A}B(\bar{C}+C) + ABC$

$= \bar{A}\bar{B} + \bar{A}B + ABC$

$= \bar{B}(\bar{A}+A) + ABC$

$= \bar{B} + BAC$

$Z = \bar{B} + AC$



2- By PS method :

$Z = M_2.M_3.M_6$

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

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

$=(A\bar{A}+\bar{B}A+CA+A\bar{B}+\bar{B}\bar{B}+C\bar{B}+A\bar{C}+\bar{B}\bar{C}+C\bar{C})$

$=(A+\bar{B}A+CA+A\bar{B}+\bar{B}+C\bar{B}+A\bar{C}+\bar{B}\bar{C}).($

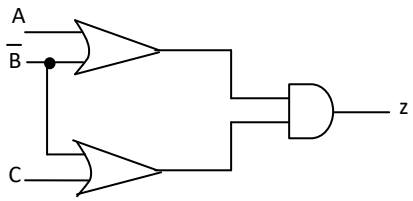
$=(A(1+\bar{B}+C+\bar{B}+\bar{C})+\bar{B}(1+C+\bar{C})).($

$=(A+\bar{B})(A\bar{A}+\bar{B}A+CA+\bar{A}\bar{B}+\bar{B}\bar{B}+C\bar{B}+\bar{A}C+\bar{B}C+C\bar{C})$

$=(A+\bar{B})(\bar{B}A+CA+\bar{A}\bar{B}+\bar{B}+C\bar{B}+\bar{A}C+\bar{B}C+C)$

$=(A+\bar{B})(\bar{B}(A+\bar{A}+1+C+C)+C(A+\bar{A}+1))$

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



PS method require one more gate than SP .

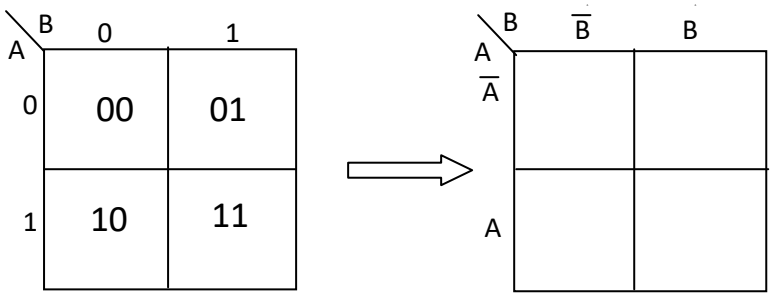
Chapter three

-Karnaugh map-

Karnaugh map simplification (k-map) :

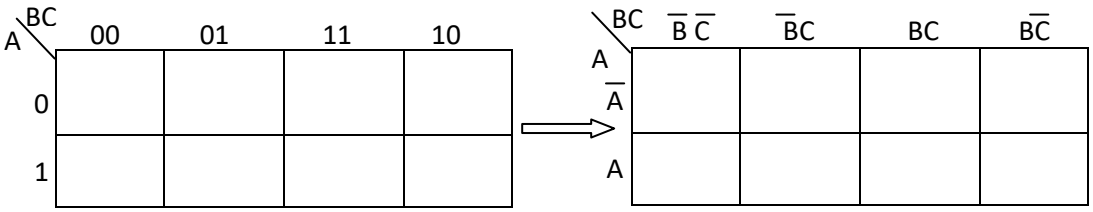
It is an important method to simplify or minimize a Boolean expression . It is composed of number of adjacent "cells" . Each cell corresponds to a T.T. row , therefore there must be 2^n Cells in the k-map (where n=no. of input variables) .

For two input variables (A&B) :



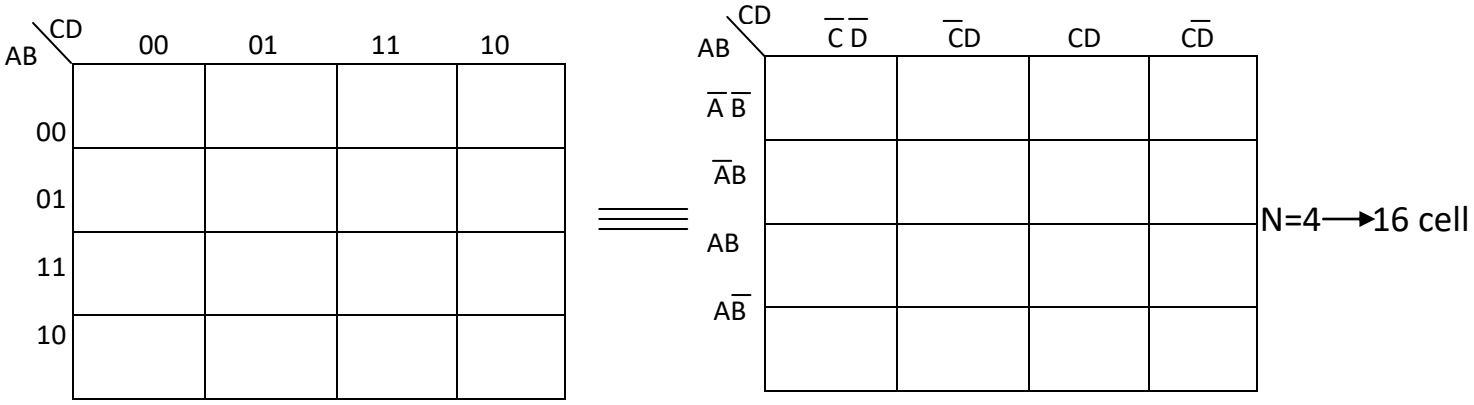
n=2 → 2^2 =4 cell

For three input variables (A,B,C) :



N=3 8 cell

For four input variables (A,B,C,D) :

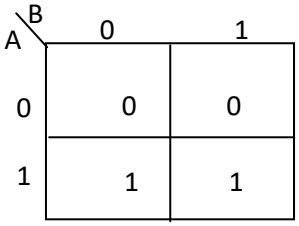


N=4 → 16 cell

The first step in the minimization method is to implement the T.T. to the K-map.
1's and 0's in the output of the T.T. is placed in the cells corresponding to the input variables of the T.T.

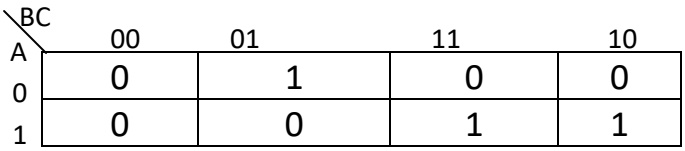
EX:

AB	Z
00	0
01	0
10	1
11	1



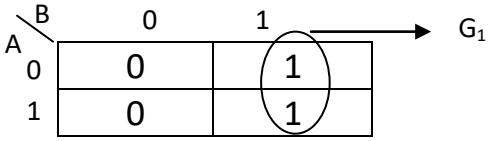
Ex:

ABC	Z
000	0
001	1
010	0
011	0
100	0
101	0
110	1
111	1

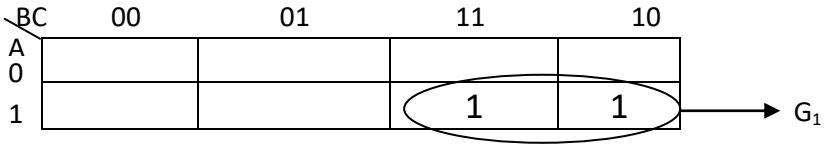


Adjacent cells :

The adjacent cells on k- map are those that differ by only one variable (only one variable changes from 0 to 1 or 1 to 0)

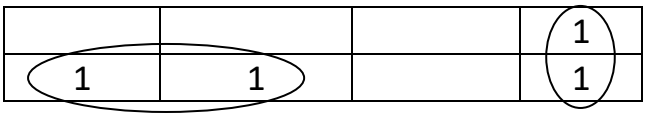


$G_1 = B$

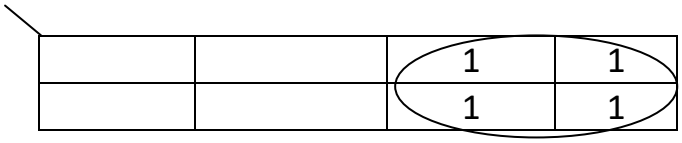


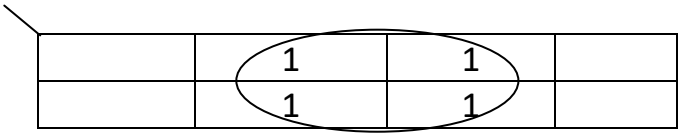
$G_1 = AB$

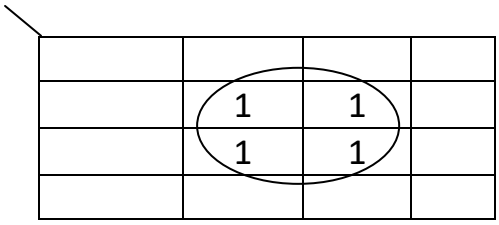
If more than one pair exist on k-map , we can OR the simplified products to get the final Boolean exp.

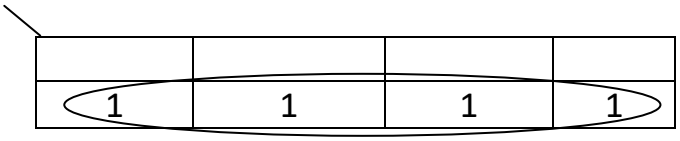


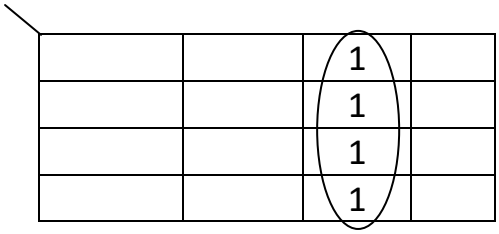
A quad is a group of four 1's that are horizontally or vertically adjacent . two variables are eliminated in the quad group.

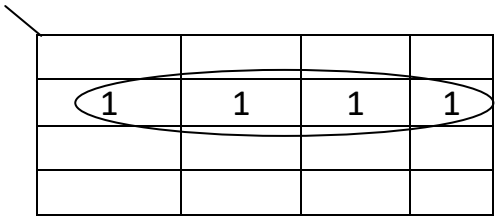




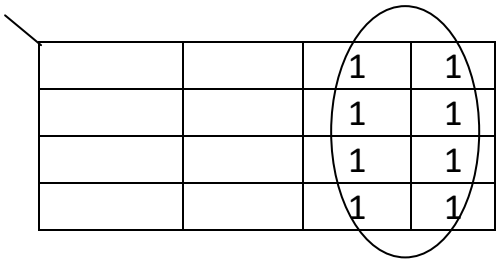


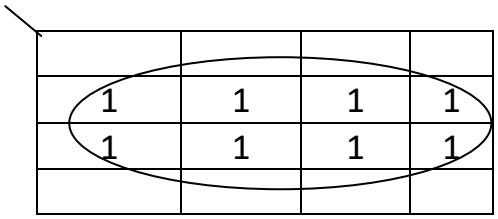






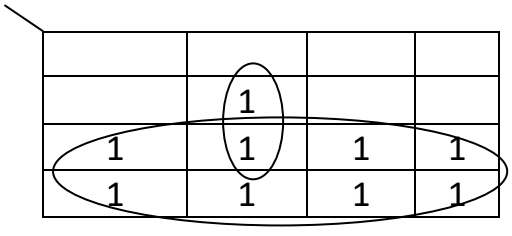
Octet is group of eight 1's that are horizontally or vertically adjacent , so three variables can be eliminated .

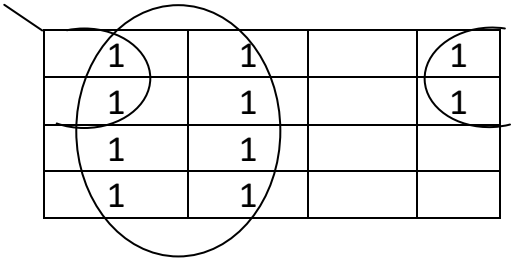




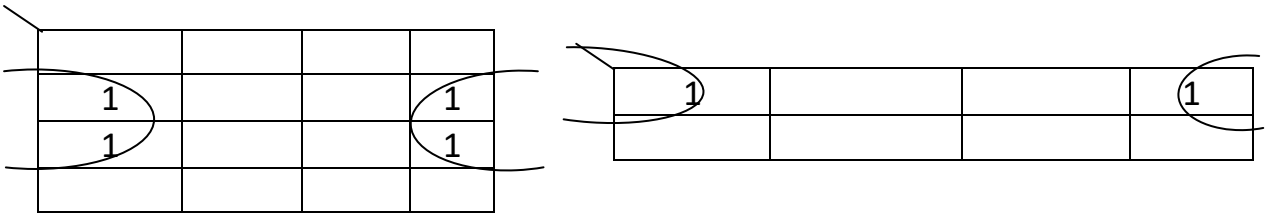
Over lapping :

The same (1) can be used for more than one group





Rolling :



Summary of k-map method :

1-Implement the T.T. to k-map

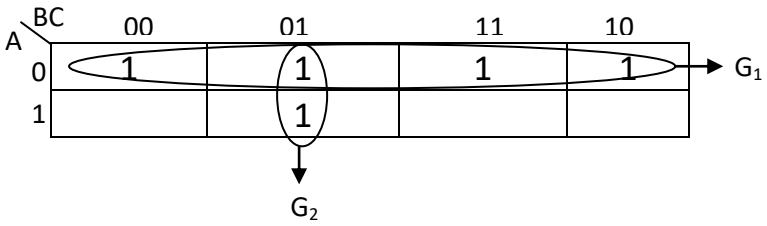
2-Encircle the octet , quads and pairs . Remember to roll and overlap to get the largest possible group.

3- If any isolated 1's , encircle each .

4-Write the Boolean exp. By ORing the products corresponding to the encircle groups.

EX: simplify the following function using k-map

$F(A,B,C) = \sum (0,1,2,3,5)$

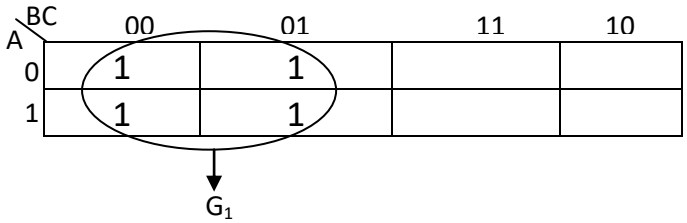


$F = G_1 + G_2$

$= \overline{A} + \overline{B}C$

EX: Simplify the following function using k-map

$F(ABC) = \sum m_0, m_1, m_4, m_5$

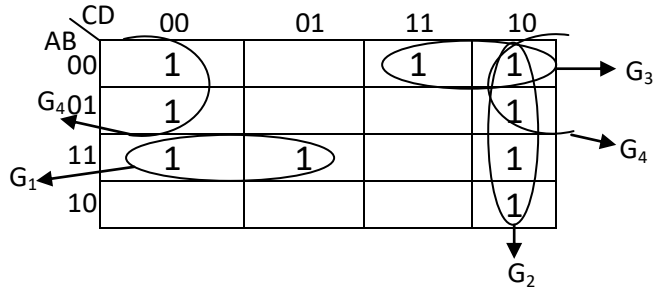


$F = G_1$

$= \overline{B}$

EX: Simplify the following function using k-map :

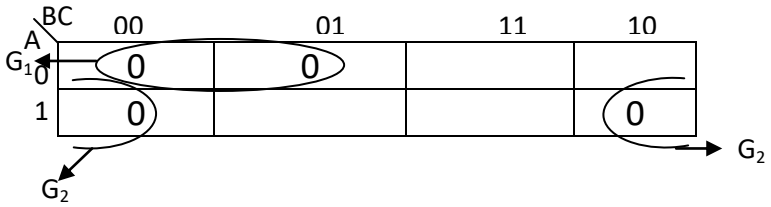
F (ABCD) = ∑ (0,2,3,4,6,10,12,13,14)



$F = G_1 + G_2 + G_3 + G_4$

EX: Find the simplified output in PS method using k-map for the following function :

F = (ABC) = π (0,1,4,6)

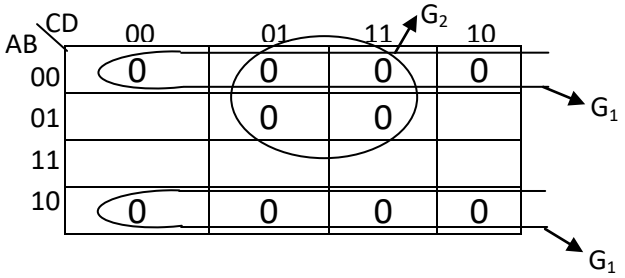


$F = G_1.G_2$

$= (A+B)(\overline{A}+C)$

EX: Simplify the following using k-map :

F (ABCD) = π(0,1,2,3,5,7,8,9,10,11)



$F = G_1 . \overline{G_2}$

$= B(A+\overline{D})$

Don't care condition :

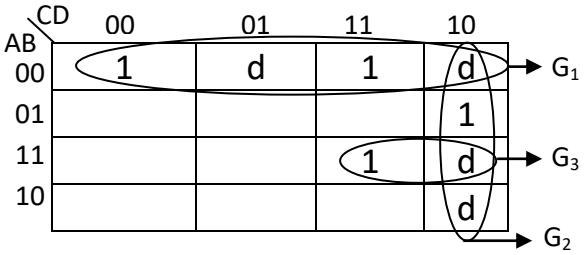
Some logic ccts. can be designed so that there are certain input conditions for which there are no specified output levels , because these input conditions will never occur.

It is necessary to specify the output for these conditions by either (0) or (1) in order to produce the simplest output exp.

EX: Simplify the following using k-map.

F(ABCD) = ∑(0,3,6,15)

dcc = 1,2,10,14



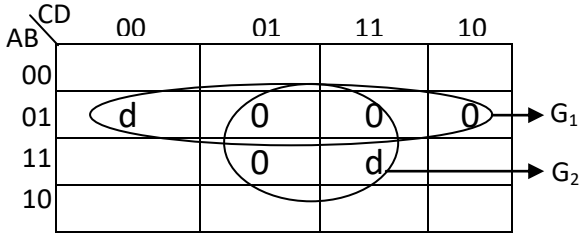
$F = G_1 + G_2 + G_3$

$= \overline{A}\,\overline{B} + C\overline{D} + ABC$

EX: Simplify the following function using k-map :

F(ABCD) = π(5,6,7,13)

dcc = (4,15)



$F = G_1 \cdot G_2$

$= (A+\overline{B})\,(\overline{B}+\overline{D})$

H.W.: Simplify the following using PS and SP method by k-map :

F(ABCD) = π(2,3,6,7,13)

dcc = (0,4,8,9,10,12,14)