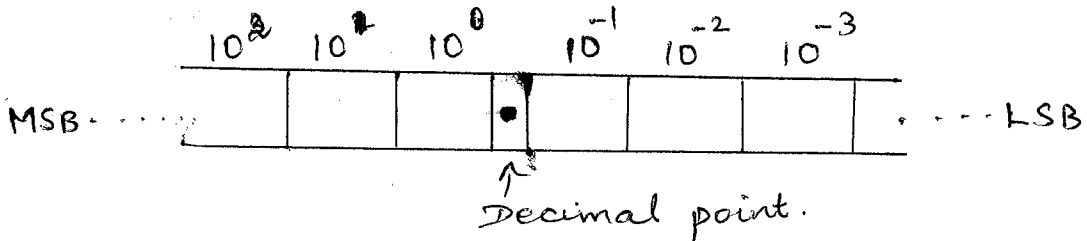
UNIT-1 DIGITAL FUNDAMENTALS

Review of binary Number system

→ decimal, binary, octal, hexadecimal

(i) Decimal number system ( )$_{10}$

In decimal number system, a number can be expressed in units, tens, hundreds, thousands and so on.



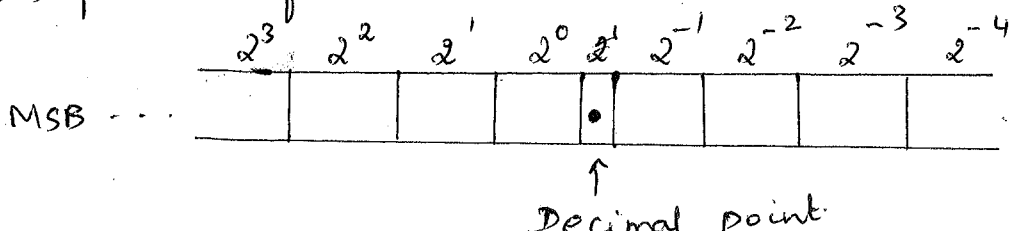
The leftmost digit, which has the greatest weight is called most significant digit. The rightmost digit which has the least weight is called least significant digit. Its weight are expressed as powers of 10. (eg:) $(50.2)_{10}$ ← base or radix

(ii) Binary Number system ( )$_2$

→ It is base-two system.

→ The two binary digits are '1' and '0'.

→ binary system weight is expressed as power of 2.



eg: $(1101.101)_2$ ← base or radix.

(iii) Octal Number system. ( )$_8$

→ uses 8 digits, 0 to 7

→ its base is 8.

(iv) Hexadecimal Number system

→ base 16., has 16 digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F.

→ each hexadecimal digit represents a group of four binary digits called nibbles.

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

NUMBER SYSTEM CONVERSION.

CONVERSION BETWEEN BINARY, OCTAL AND HEXADECIMAL NUMBERS.

Decimal	Binary	octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

1) Binary to octal conversion:

2 to 2^3 ← number of digit in integer portion of number (n).

→ group by three digits of binary number then convert each group digit to octal equivalent.

$(111\ 101\ 100)_2$

7 5 4

Ans $(754)_8$.

(ii) <u>octal to binary conversion</u>.

→ Convert each octal digit to three digit binary number.

$(6\ 3\ 4)_8$

6 3 4

110 011 100

Ans: $(110011100)_2$.

$(725.63)_8$ to binary

7 2 5 . 6 3

111 010 101 . 110 011

Ans: $(111010101.110011)_2$.

(iii) <u>Binary to hexadecimal conversion</u>

→ group by four digits of binary number then convert each group digit to hexadecimal equivalent.

$(1101100010011011)_2$

D 8 9 B

Ans: $(D89B)_H$

CONVERTING ANY RADIX TO DECIMAL EQUIVALENT.

$$N = A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \ldots + A_1 r^1 + A_0 r^0 + A_{-1}r^{-1} + A_{-2}r^{-2} + \ldots + A_{-m}r^{-m}.$$

N = Number in decimal.

A = Digit.

r = Radix or base of a Number system.

n = Number of digits in the integer portion of number.

m = Number of digits in fractional portion of number.

(i) Binary to decimal.

$(11010.11)_2$ to $(?)_{10}$

					.		
1	1	0	1	0	.	1	$1 \times 2^{-2} = 0.25$
							$1 \times 2^{-1} = 0.5$
							$0 \times 2^{0} = 0$
							$1 \times 2^{1} = 2$
							$0 \times 2^{2} = 0$
							$1 \times 2^{3} = 8$
							$1 \times 2^{4} = 16$
							26.75

$(11010.11)_2 = (26.75)_{10}$.

(ii) Hexadecimal to decimal:

$(16.5)_{16}$ to $(?)_{10}$.

1	6	.	5	$5 \times 16^{-1} = 0.3125$
				$6 \times 16^{0} = 6$
				$1 \times 16^{1} = 16$
				22.3125

$(16.5)_{16} = (22.3125)_{10}$

(iv) Hexadecimal to binary conversion.

→ Convert each hexadecimal digit to four digit binary numbers.

$(3\ F\ D)_H$

Ans: $(0011111111101)_2$

3 F D

0011 1111 1101

(v) Octal to hexadecimal conversion.

→ Convert octal number to its binary equivalent.

→ Convert binary number to its hexadecimal equivalent.

$(6\ 1\ 5)_8$

1, → Octal to binary

6 1 5

110 001 101 ⇒ $(110\ 001\ 101)_2$

2, → Binary to hexadecimal.

000 1100 0101

1 8 D ⇒ $(18D)_H$: Ans

(vi) Hexadecimal to Octal conversion.

→ Convert hexadecimal number to its binary

→ Convert binary number to its octal.

$(25B)_H$

1, → Hexadecimal to binary.

2 5 B

0010 0101 1011

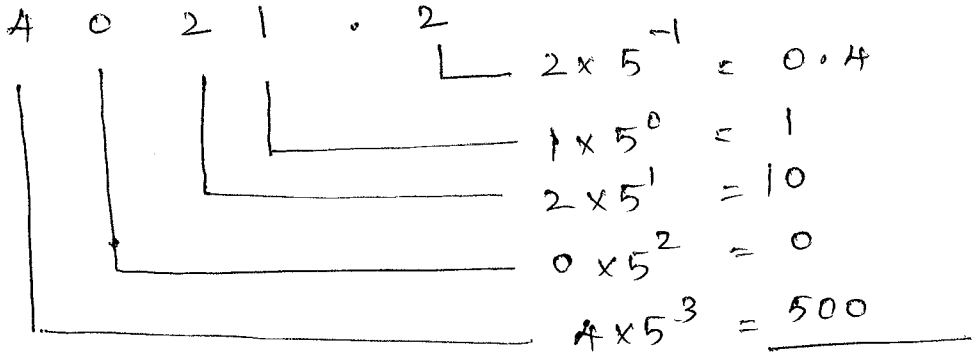
2, → binary to octal.

Ans: $(1133)_8$

001 001 011 011

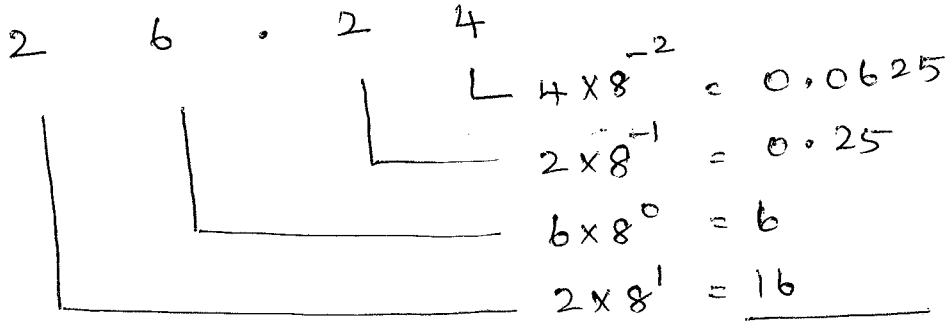
1 1 3 3

(iii) $(4021.2)_5$ to $(\ ?\ )_{10}$



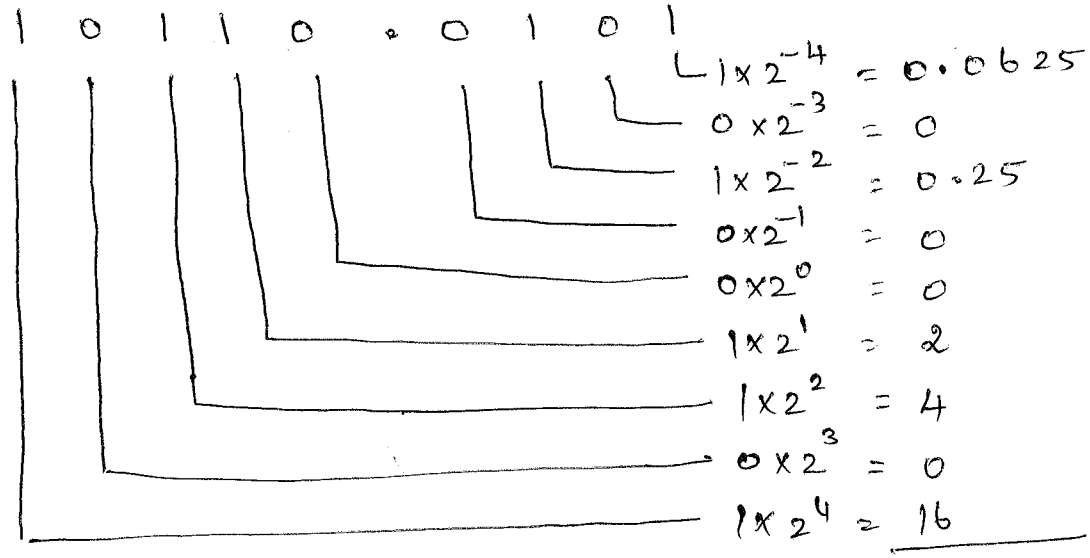
$(4021.2)_5 = (511.4)_{10}$

(iv) $(26.24)_8$ to $(\ ?\ )_{10}$



$(26.24)_8 = (22.3125)_{10}$

(v) $(10110.0101)_2$ to $(\ ?\ )_{10}$



$(10110.0101)_2 = (22.3125)_{10}$

CONVERSION OF DECIMAL NUMBERS TO ANY RADIX NUMBER.

step 1: Convert integer part by successive division method.

step 2: Convert fractional part by successive multiplication method.

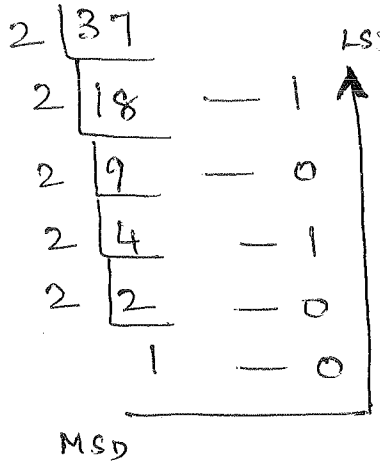
Successive division for integer part conversion

→ repeatedly divide the integer part of decimal number by radix 'r', until it cannot be divided further.

→ The remainders are taken in reverse order to form a new radix number.

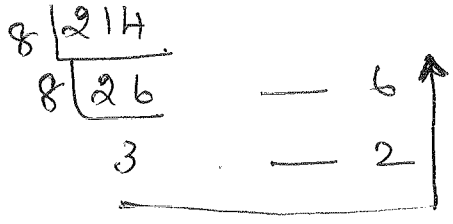
→ The first remainder is LSD.
The last " " MSD.

$(37)_{10}$ to binary number.



$(37)_{10} = (100101)_2$

214 to octal.



$(214)_{10} = (326)_8$

$(54)_{10}$ to radix 4.

$$4 \mid 54$$
$$4 \mid 13 \quad -2$$
$$3 \quad -1$$

$(54)_{10} = (312)_4$.

$(3509)_{10}$ to hexadecimal.

			Quotient
16	3509		
16	219	-5	13
	13	$-$ B	11
			5

$(D\ B\ 5)_H$.

SUCCESSIVE MULTIPLICATION FOR FRACTIONAL PART.

step 1: The number to be converted is multiplied by radix of new number, producing a product that has an integer part and a fractional part.

step 2: The integer part (carry) of the product becomes a numeral in the new radix number.

step 3: The fractional part is again multiplied by the radix and this process is repeated until fractional part reaches '0' with carry downwards.

(i) $(0.8125)_{10}$ to $(?)_2$.

MSD

$0.8125 \times 2 = 1.625$	carry = 1	
$0.625 \times 2 = 1.25$	carry = 1	$(0.1101)_2$
$0.25 \times 2 = 0.5$	" = 0	
$0.5 \times 2 = 1.0$	" = 1	

LSD.

↳ fraction = 0 stop.

ii) $(0.640625)_{10}$ into octal.

$0.640625 \times 8 = 5.125$ Carry = 5

$0.125 \times 8 = 1.0$ Carry = 1

$(0.640625)_{10} = (0.51)_8$

(iii) $(0.1289062)_{10}$ into hexadecimal

$0.1289062 \times 16 = 2.0624992$ carry = 2

$0.0624992 \times 16 = 0.9999872$ carry = 0

$(0.1289062)_{10} = (0.21)_{16}$

1. Convert $(AF3.15)_{16}$ to base 10.

A F 3 . 1 5

$5 \times 16^{-2} = 0.01953125$

$1 \times 16^{-1} = 0.0625$

$3 \times 16^{0} = 3$

$15 \times 16^{1} = 240$

$10 \times 16^{2} = 2560$

$(2803.0820031)_{10}$

2. Find octal equivalent of decimal 200.

8 | 200

8 | 25 — 0 ↑

3 — 1

$(310)_8$

$(310)_8$

3 1 0

$0 \times 8^{0} = 0$

$1 \times 8^{1} = 8$

$3 \times 8^{2} = 192$ / $(200)_{10}$

COMPLEMENTS.

→ Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation.

→ Two types of complements for base-r systems:

r's complement
- → referred as r's complement.
- → for binary number → 2's complement.
- → for decimal number – 10's complement.

diminished r's complement.
- → referred as (r–1)'s complement.
- → for binary number 1's complement.
- → for decimal number 9's complement.

DIMINISHED RADIX COMPLEMENT (r–1)'s complement.

9's complement

9's complement is obtained by subtracting each digit from 9.

ex: find 9's complement of 546700 & 012398

(i) 546700

Subtract each digit by '9'.

```
  999999
  546700
  ------
  453299
  ------
```

(ii) 012398

subtract each digit by '9'.

```
  999999
  012398
  ------
  987601
  ------
```

1's complement

→ binary number $r = 2$ & $(r-1) = 2-1 = 1$'s.

$2^4 - 1 = 16 - 1 = 15$ = binary equivalent $(1111)_2$

→ Change bit from '0' to '1' (or) '1' to '0'.

ex: 1011000_2 → 1's complement 0100111

0101101 → 1's complement = 1010010

$(r-1)$'s complement of octal or hexadecimal number is obtained by subtracting each digit from 7 or 'F'.

RADIX COMPLEMENT

→ r's complement is obtained by adding '1' to $(r-1)$'s Complement.

10's complement

→ take 9's complement for the given decimal number.
→ add 1 to 9's complemented value.

(i) 012398

⇒ 999999
012398

987601 ⇒ 9's complement of 012398
+ 1 ⇒ add 1 to 9's complemented value.

987602

(ii) 246700

step1: 9's complement. 999999 − 246700 = 753299

step2: add 1 → 753299 + 1 = 753300 Ans.

2's complement.

→ take 1's complement for the given binary number.

→ add '1' to the complemented value.

(i) 1101100

step 1: take 1's complement.

change 1's to '0' & '0's to '1'.

```
  1101100
-------------
  0010011  → 1's complement.
+       1
-------------
  0010100
-------------
```

step 2: add 1 to complemented value

2's complement of 1101100 is 0010100.

(ii) 01101

step 1: take 1's complement for 01101

10010

step 2: add '1' ⇒ +1

10011

2's complement of 01101 is 10011.

SUBTARCTION WITH COMPLEMENTS.

(r-1)'s complement

1-s
1. take (r-1)'s complement
2. Add
3. end around carry.

s-1
1. take (r-1)'s complement
2. Add.
3. Once again take complement and add -ve sign.

r's complement

1-s
1. take r's complement
2. Add
3. discard carry

s-1
1. take r's complement
2. Add.
3. take once again r's complement and add -ve sign.

1. Subtraction of smaller number from larger number.
2. Subtraction of larger number from smaller number.

Subtraction of smaller number from larger number

1. Determine the 1's complement of smaller number.
2. Add 1's complement to the larger number
3. Remove the carry and add it to result. This is called end-around carry.

1. Subtract 1010111_2 from 1110001_2 using 1's Complement method.

$1010111 = 43$ and $1110001 = 97$

smaller from larger.

step 1: take 1's complement for smaller number.

$43 = 101011$

1's complement = 010100.

step 2: add 1's complement to larger no.

$97 = 1110001 \Rightarrow$

$$\begin{array}{r} 1110001 \\ +\ 0101000 \\ \hline \text{carry} \rightarrow (1)\ 0011001 \\ \qquad\qquad \rightarrow +1 \\ \hline 0011010 \end{array}$$

step 3: Remove carry and add To the LSB.

Ans: 0011010 $\quad 97 - 43 = 14$

$1 \times 2^0 = 0$
$1 \times 2^1 = 2$
$0 \times 2^2 = 0$
$1 \times 2^3 = 8$
14

$X - Y = 1010100 - 1000011$

smaller from larger.

Rough
84 − 67 = (17)
↓ → 1000011.
1010100

1000011
→ 1's complement.
0111100

X = 1010100
1's complement of Y } 0111100
0010000 0
1 ← add carry ⇒ + 1
(17) ⇐ 0010001 Ans.

Subtraction of larger number from smaller number.

1. Determine 1's complement of larger number.
2. Add 1's complement to smaller number.
3. Sum is taken, 1's complement and assign a negative sign to the answer.

1. subtract 111001_2 (57) from 101011_2 (43). = 43 − 57

Step 1: take One's Complement for larger number.

111001 → 000110.

Step 2: add 1's complement to the smaller number.

43 ⇒ 101011
1's complement of 111001 } ⇒ 000110
Sum = 110001

Step 3: take 1's complement of the answer and assign negative sign.

110001 (Sum) —1's complement→ −(001110)$_2$ Ans

2. $y - x = 1000011 - 1010100$

Step 1: take 1's complement for larger number.

$84 = 1010100 \xrightarrow{\text{1's complement}} 0101011$

Step 2: add 1's complement to smaller number.

$$\begin{array}{r} 1000011 \\ +\ 0101011 \\ \hline \text{Sum} = 1101110 \end{array}$$

Step 3: take 1's complement for sum and add a negative sign.

$1101110 \xrightarrow{\text{1's complement}} (-0010001)_2$ Ans.

2's COMPLEMENT SUBTRACTION.

1. SUBTRACTION OF SMALLER NUMBER FROM LARGER NO.

1. Determine 2's complement of smaller number.
2. Add 2's complement to larger number.
3. Discard carry.

Subtract $101\overset{43}{0}11_2$ from $\overset{57}{1}11001_2$ using 2's Complement.

Step 1: 2's complement of smaller number.

$101011 \xrightarrow[\text{complement}]{\text{1's}} 010100$

$+1 \leftarrow$ 1's Complement $+1$

$010101 \leftarrow$ 2's Complement

larger Number 57 → 111001

010101 ← 2's complement of $(101011)_2$

1 001110

discard carry.

Ans: $(001110)_2$.

(ii) 1010100 − 1000011.

Step 1: take 2's complement of smaller number.

1000011 —1's complement→ 0111100

+ 1

2's complement → 0111101

Step 2: Add 2's complement to the larger number.

larger number → 1010100

+ 0111101 ← 2's complement of $(1000011)_2$

discard Carry → 1 0010001

Ans $(0010001)_2$

SUBTRACTION OF LARGER NUMBER FROM SMALLER NUMBER.

1. Determine 2's complement of larger number.
2. Add 2's complement to smaller number.
3. take once again 2's complement and add a negative sign to the sum.

Subtract 111001_2 from 101011_2.

Step 1: find 2's complement of larger number.

111001 —1's complement→ 000110

+ 1

2's complement → 000111

Step 2: add the complemented value to the smaller number.

$$\begin{array}{r} 1\,0\,1\,0\,1\,1 \\ +\ 0\,0\,0\,1\,1\,1 \leftarrow \text{2's complement of } (111001)_2 \\ \hline \text{Sum}\quad 1\,1\,0\,0\,1\,0 \end{array}$$

Step 3: take 2's complement of sum.

$$110010 \xrightarrow[\text{complement}]{1\text{'s}} 001101 \qquad \begin{array}{r} 001101 \\ +\ 1 \\ \hline -\,001110 \leftarrow \text{2's Complement.} \end{array}$$

Ans $(-001110)_2$

2. $Y - X$

$1000011 - 1010100.$

Step 1: take 2's complement for the larger value.

$$1\,0\,1\,0\,1\,0\,0 \xrightarrow[\text{Complement}]{1\text{'s}} 0\,1\,0\,1\,0\,1\,1 \qquad \begin{array}{r} 0101011 \\ +\ 1 \\ \hline \end{array}$$

2's complement → $0\,1\,0\,1\,1\,0\,0$

Step 2: add the 2's complement value to the smaller number.

$$\begin{array}{r} \text{Smaller no:} \rightarrow\ 1\,0\,0\,0\,0\,1\,1 \\ +\ 0\,1\,0\,1\,1\,0\,0 \leftarrow \text{2's complement of } (1010100)_2 \\ \hline \text{Sum:}\quad 1\,1\,0\,1\,1\,1\,1 \end{array}$$

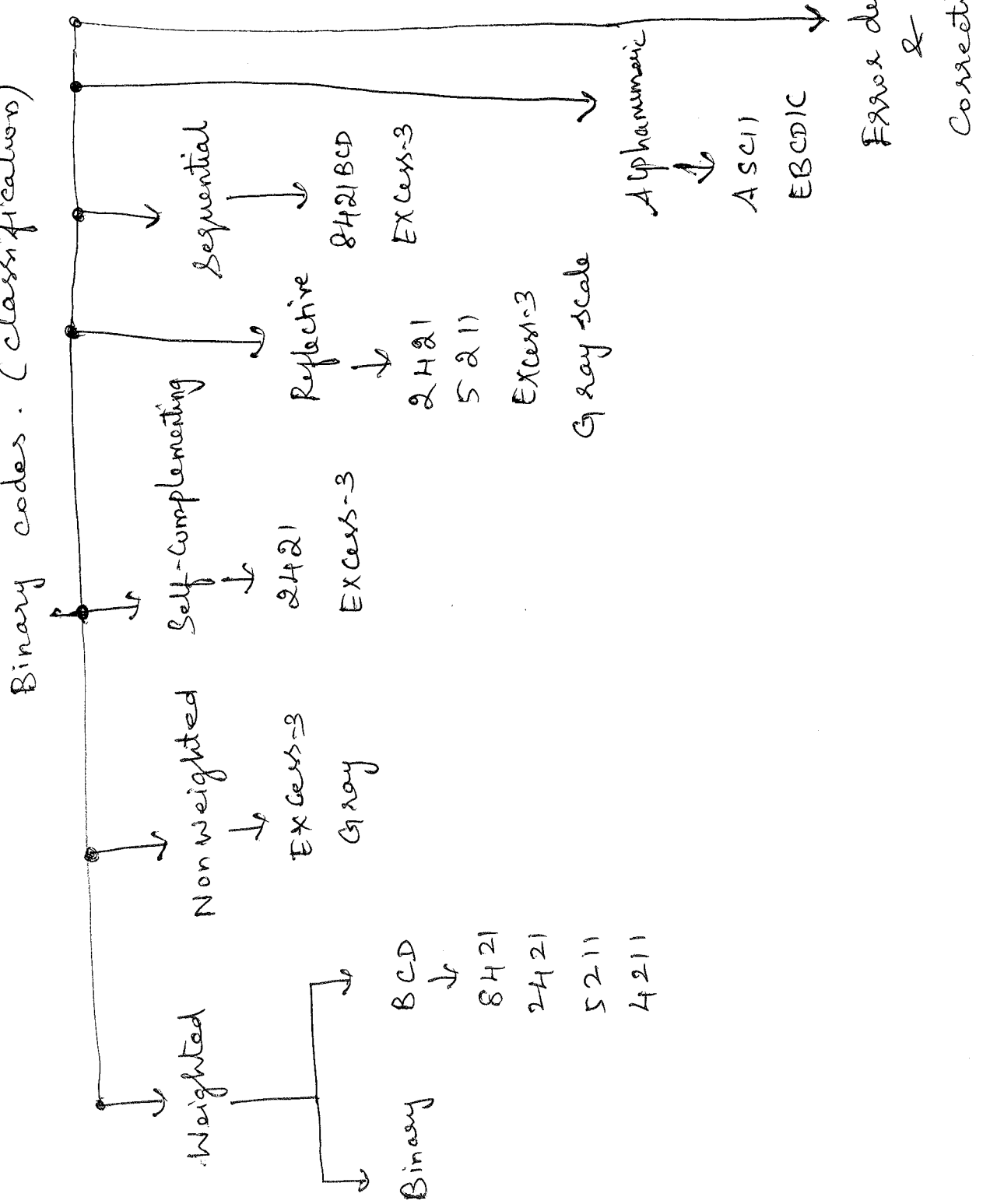
Step 3: take 2's complement of sum and add negative sign

$$110111 \xrightarrow[\text{complement}]{1\text{'s}} 001000 \qquad \begin{array}{r} 001000 \\ +\ 1 \\ \hline 0010001 \text{ 2's Compl} \end{array}$$

Ans $(-0010001)_2$.

CODES

A code is a symbol or group of symbols that stands for something. Binary bits are often used in groups to stand for things such as decimal or alphanumeric characters. Since binary code is represented only with 0's & 1's the implementation process becomes easy.



1. Weighted Binary codes

This obey their positional weighting Principles. Each position of a number represents a specific weight. In a weighted binary code, bits are multiplied by the weights indicated, the sum of these weighted bits give the decimal digit. In (BCD) Binary coded Decimal form in which each integer of a decimal number is represented by a four bit binary number.

Decimal	binary	BCD code 8421	2421	4221	5211
0	0000	0000	0000	0000	0000
1	0001	0001	0001	0001	0001
2	0010	0010	0010	0010	0011
3	0011	0011	0011	0011	0101
4	0100	0100	0100	1000	0111
5	0101	0101	1011	0111	1000
6	0110	0110	1100	1100	1010
7	0111	0111	1101	1101	1100
8	1000	1000	1110	1110	1110
9	1001	1001	1111	1111	1111
10	1010	✗	✗	✗	✗
11	1011	✗	✗	✗	✗
12	1100	✗	✗	✗	✗
13	1101	✗	✗	✗	✗
14	1110	✗	✗	✗	✗
15	1111	✗	✗	✗	✗

2. Non-Weighted Code

Non-weighted codes are codes that are non-positional weight. This means that each position within a binary number is not assigned a fixed value. The Excess-3 is derived from 8421 BCD code by adding binary value $(0011)_2$ to each code in 8421. Gray code is also called as unit distance code & reflected code. This has 1 bit change between each successive value.

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

3. Self Complementary codes

The 2421 and excess-3 code is an example of self-complementary code. In this code, the 1's complement of excess-3 code is excess-3 code for 9's complement of corresponding decimal number.

Example:

The excess-3 code for 2 is 0101, the 1's complement of 0101 is 1010. It is excess-3 code of 7 and 7 is the 9's complement of 2.

4. Unit Distance code

There will be a unit distance between two consecutive codes. The bit pattern for two consecutive numbers differ in only one bit position. ex: Gray code.

~~5. Alphanumeric~~

5 ~~Reflective~~ code

A code is said to be sequential when each succeeding code is one binary number is greater than its preceding code. This generally helps mathematical manipulation of data while 8 4 2 1.

6. Reflective code.

A code is said to ~~the~~ be reflective

when the code for 9 is the 9's complement of the code of 0, 8 for 1, 7 for 2, etc., The 2421, 5211 are reflective code.

6. Alphanumerical code.

The binary codes of alphabets, numbers and special symbols are known as alphanumerical codes. Example: ASCII code, EBCDIC code

ASCII code:

American standard code for information interchange (ASCII) is the most widely used alphanumeric code, used in most microcomputer. It is a 7 bit code, has $2^7 = 128$ possible code groups. All the keys of keyboard are represented by 7 bit binary code. This code is used to interchange information between input/output device and computer and stored into memory.

EBCDIC code:

Extended binary coded decimal interchange code is an alphanumeric code. This is an 8 bit code and it has $2^8 = 256$ possible code group.

5. ERROR DETECTING AND CORRECTING CODES.

When information in digital form is transmitted to a long distance, errors may get introduced and '1' becomes '0' and '0' becomes '1'. Special codes are used to detect and correct such errors. Parity and Hamming codes are commonly used for error detection and correction.

BINARY CODED DECIMAL CODE (BCD CODE)

It is a 4-bit binary coded decimal number. Each digit of decimal number is represented by four bits, the digit 0 in BCD is represented as 0000 and the digit 9 is represented as 1001.

Decimal digit	BCD Codes. B_3 (8)	B_2 (4)	B_1 (2)	B_0 (1)
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

The codes 1010, 1011, 1100, 1101, 1110, 1111 are invalid. BCD codes are also called

as 8-4-2-1 code because the weights of B_3, B_2, B_1, B_0 are 8, 4, 2 and 1.

→ An N digit decimal number can be represented by 4 x N bits in BCD code.

decimal number	binary code	BCD code
4	100	0100
15	1111	0001 0101 (1 5)

for example: BCD code of 15_{10} is eight bits and binary code of 15_{10} is four bits.

∴ BCD code is not efficient as binary code. Because BCD code requires more space and time to transmit information.

BCD ADDITION

In BCD addition, each digit of a decimal number is converted into its 4-bit binary equivalent and the addition of two BCD codes is carried out using the rules of binary addition.

After the addition of two BCD codes, the result may be valid or an invalid BCD.

→ If the result is invalid, it is converted into an valid BCD by adding $(6)_{10}$ or $(0110)_2$ for the 4 bit addition.

→ If carry is generated after addition of $(0110)_2$, it is added to the next bit.

STEPS:

1. Convert the decimal numbers into their equivalent BCD codes.
2. Add the BCD numbers using the rules of binary addition.
3. Check the result. If it is valid (less than or equal to 9), no correction is needed and if the result is invalid (greater than 9), go to next step, otherwise stop.
4. Add 0110 to 4 bit sum to get the correct result.

SUM EQUAL TO 9 (OR) < 9 WITH CARRY '0'. [Valid BCD].

1. 6 + 3 = 9.

6	0 1 1 0	
3	0 0 1 1	
9	1 0 0 1	No carry & Sum = 9.

2. 15 + 14 = 29

15	0 0 0 1	0 1 0 1	
14	0 0 0 1	0 1 0 0	No carry, {each sum ≤ 9
	0 0 1 0	1 0 0 1	Valid BCD
	2	9	← Ans.

SUM > 9 WITH CARRY = 0. [INVALID BCD]

1.

6	0 1 1 0	
+ 8	1 0 0 0	
14	1 1 1 0	> 9 invalid.
		∴ add 0110

Sum = 1 1 1 1 0 > 9 Invalid.

+6 = 0 1 1 0

0 0 0 1 | 0 1 0 0

1 4 ← Ans.

2. 19 0 0 0 1 | 1 0 0 1

22 0 0 1 0 | 0 0 1 0

41 0 0 1 1 | 1 0 1 1

Valid ✓ ↘ Invalid. ∴ add +6 or 0110.

↘ carry = 0.

(ii) 1 1 1 1 1 1

0 0 1 1 | 1 0 1 1

+ 0 1 1 0

0 1 0 0 | 0 0 0 1

4 1 → Ans.

SUM WITH CARRY.

8 1 0 0 0

+ 9 1 0 0 1

17 1 | 0 0 0 1 ← carry = 1 ∴ add 0110.

Carry ↑ 0 1 1 0

0 0 0 1 | 0 1 1 1

1 7 → Ans.

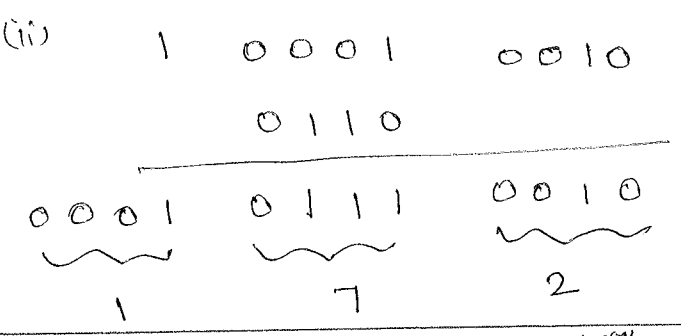
2. 91 1 0 0 1 | 0 0 0 1

81 1 0 0 0 | 0 0 0 1

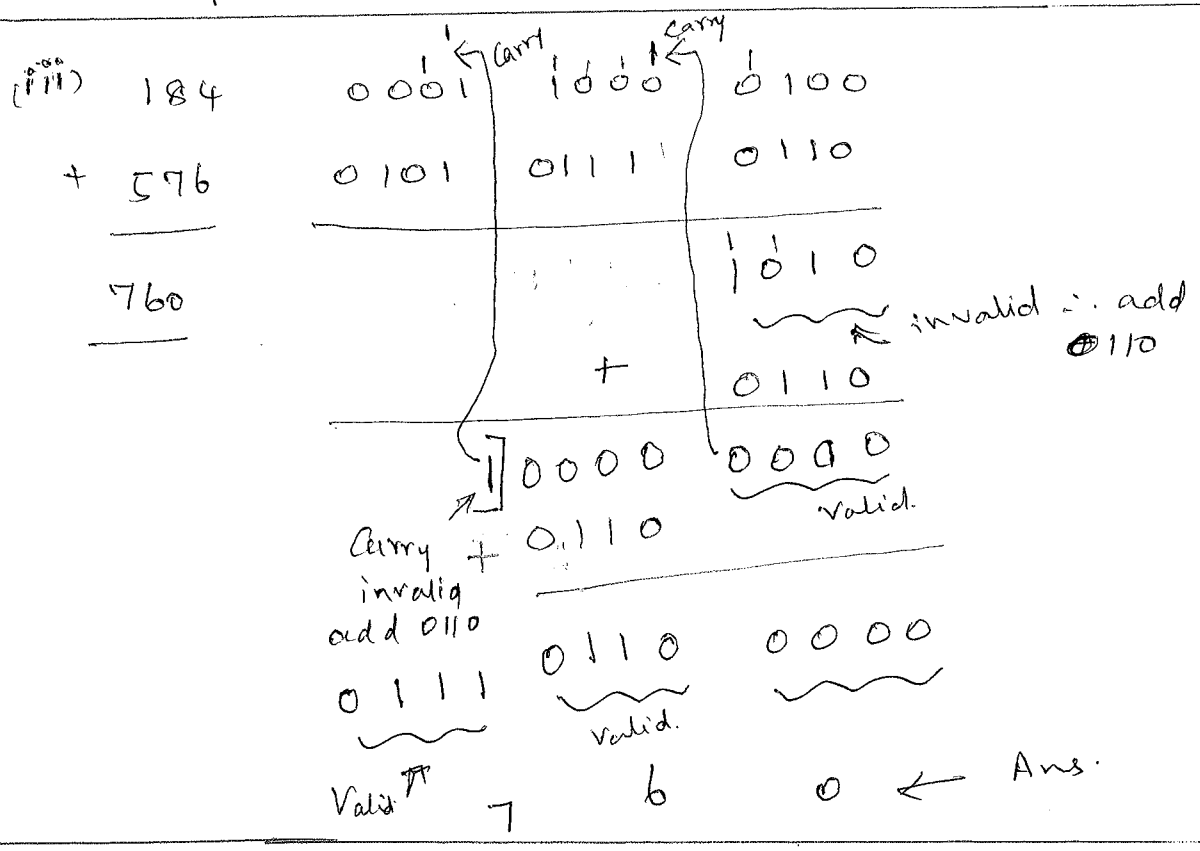
172 1 | 0 0 0 1 | 0 0 1 0

Carry = 1 ∴ Invalid, add 0110 ↖ Valid.

(ii)



(iii)



BCD SUBTRACTION.

→ The subtraction of numbers is nothing but an addition of second negative number with the first number $A - B = A + (-B)$.

→ The subtraction of numbers is addition of signed BCD number.

→ The negative BCD numbers can be expressed by 9's or 10's complement.

$$A - B = A + C$$

where $-B = +C$ (complemented value)

→ BCD subtraction is performed by 9's or 10's complement.

BCD SUBTRACTION USING 9'S COMPLEMENT

1. Find 9's complement of the subtractor.
2. Perform BCD addition of the first number with the 9's complement of the second number.
3. If carry is generated, then the result is positive. Add the carry to the result to get the correct result.
4. If carry is not generated, then the result is negative and it is in 9's complement form. once again take 9's complement.

1. SUBRACTION OF SMALLER NUMBER FROM LARGER NUMBER.

$8 - 3 = 8 + (9\text{'s complement of } 3)$

$= 8 + 6$

$$\begin{array}{r} 9 \\ -3 \\ \hline 6 \end{array}$$

8 = 1000

9's complement of 3 is 6 = 0110

	1110	BCD is invalid > 9
∴ add 6 to Sum	+ 0110	
Carry exist →	1 0100	
	→ + 1	add carry to result.
	0101	Ans: BCD code of 5.

---

68 − 24

68 + (9's complement of 24)

68 + 75.

$$\begin{array}{r} 99 \\ 24 \\ \hline 75 \end{array}$$

68	0110	1000	
75	0111	0101	
	1101	1101	Both BCD is invalid > 9
add 6 to Sum	0110	0110	

Carry → 1 0 0 1 1 + 1 0 0 1 1

→ +1 carry → +1 → Add carry.

0 1 0 0 0 1 0 0

BCD code of 44.

2. SUBTRACTION OF LARGER NUMBER FROM SMALLER NUMBER

$3 - 8 = 3 + (9\text{'s complement of } 8)$

$= 3 + 1$

3 = 0 0 1 1

9's complement of 8 is '1' = 0 0 0 1

0 1 0 0 → 4

Carry not generated ∴ result is negative. Sum in 9's complement

Ans = −5

9 − 4 = −5 Ans

$24 - 68 = 24 + (9\text{'s complement of } 68)$

$= 24 + 31.$

24 = 0010 0100

9's Complement of 68 is 31 = 0011 0001

0101 0101 → 5 5

Carry not generated ∴ result is negative and is in 9's complement form.

99 − 55 = −44

Ans: −44.

99 − 68 = 31

BCD SUBTRACTION USING 10'S COMPLEMENT

1. Find 9's complement subtractor.
2. Find 10's complement.
3. ~~Perform~~ Add First number with the 10's complemented BCD number.
4. If carry is not generated then the result is negative. and is in 10's complement form. once again take 10's complement.
5. If carry generated, then the result is positive

discard the carry to get the correct result.

1. SUBTRACTION OF SMALLER NUMBER FROM LARGER NUMBER.

$8 - 4 = 8 + (10\text{'s Complement of } 4)$

$= 8 + 6.$

$8 = 1000$

$6 = 0110$

1110 > 9 invalid BCD

0110 Add +6 to the sum.

$1\,0100$

Carry exist ↗ 1 ∴ discard carry. Ans = 0100 (4)

$9 - 4 = 5$ ← 9's comp.

$5 + 1 = 6$ ← 10's

$68 - 24 = 68 + (10\text{'s Complement of } 24)$

$= 68 + 76.$

$68 = 0110\ \ 1000$

$76 = 0111\ \ 0110$

$1101\ \ 1110$

$0110\ \ 0110$

$1\,0100\ \ 0100$

discard Carry ↗ 1

0100 (4) 0100 (4)

Ans : 44

$99 - 24 = 75$ ← 9's

$75 + 1 = 76$ ← 10's

---

2. SUBTRACTION OF LARGER NUMBER FROM SMALLER NUMBER.

$4 - 8 = 4 + (10\text{'s complement of } 8)$

$= 4 + 2.$

$4 = 0100$

$2 = 0010$

0110

Once again take 10's comp. ↙ (6)

Ans: −4

$6 = 0110$

9's comp. of $6 = 9 - 6 = 3$

$3 + 1 = 4$ → 10's comp.

Carry not generated,

∴ result is negative.

$9 - 8 = 1$

$1 + 1 = 2$ (10's)

$24 - 68 = 24 + (10's\ complement\ of\ 68)$

$= 24 + 32.$

24	=	0010	0100
32	=	0011	0010
		0101	0110
		5	6

In 10's complement form.

$$\begin{array}{r} 99 \\ 68 \\ \hline 31 \\ +1 \\ \hline 32 \end{array}$$ (10's complement)

$$\begin{array}{r} 99 \\ 56 \\ \hline 43 \\ +1 \\ \hline -44 \end{array}$$

← 9's complement of 56

← 10's complement. Ans: −44.

2-4-2-1 CODE.

It is a weighted code, the weight of binary symbol depends on its position. The weights of b_3, b_2, b_1 and b_0 are 2, 4, 2 and 1, respectively. Thus this code is referred to as 2-4-2-1 code.

Decimal digit	2-4-2-1 code			
	b_3 (2)	b_2 (4)	b_1 (2)	b_0 (1)
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1

8	1	1	1	0
9	1	1	1	1

EXCESS – 3 CODE:

obtained by adding 3 to each digit of the decimal number and then, it is represented by a 4-bit binary code.

Decimal digit	BCD code				Excess-3 code.			
	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Find excess-3 code for 17

17 → BCD code + 3 = excess-3.

17 = 0001 0111

+3 = 0011

20 = 0010 1010 → > 9 ∴ add 6

0110

0010 0000 → 20 : Ans.

excess-3 of 17 = 0101 0000.

EXCESS – 3 ADDITION:

1. If carry = 1, add 3 to sum of 2 digits.
2. If carry = 0, subtract 3.

CARRY:

8 + 6 = 14.

excess 3 of 8	=	1011
excess 3 of 6	=	1001
Carry. →		10100
add 3 to each digit		0011 0011
		0100 0111
		1 4

NO CARRY:

1 + 3 = 4.

excess 3 of 1	=	0100
excess 3 of 3	=	0110
No carry →		1010
Subtract 3		− 0011
		0111 = excess-3 code 4.

10 − 1 = 1

0 − 1 = 1 borrow

EXCESS – 3 SUBTRACTION.

1. take complement for subtrahend.
2. Add complement subtrahend to minuend.
3. If carry = 1, then result is positive. Add end 3 and end around carry.
4. If carry = 0, then result is negative and subtract 3. and take once again complement.

WITH CARRY:

$8 - 5 = 8 + (9\text{'s complement of } 5) = 8 + 4$

$$\begin{array}{r} 9 \\ -\ 5 \\ \hline 4 \end{array}$$

excess 3 of 8 = 1011

excess 3 of 4 = 0111

Carry exist → 1 0010

∴ add 3 and end around carry → 0010 + 0011 + 1

0110 excess 3 of 3.

WITHOUT CARRY:

$5 - 8 = 5 + (9\text{'s complement of } 8)$

$= 5 + 1$

$$\begin{array}{r} 9 \\ -\ 8 \\ \hline 1 \end{array}$$

excess of 5 = 1000

excess 3 of 1 = 0100

Carry = 0 → 1100 − 0011

∴ subtract 3

1001 → excess 3 of 6 (9's complement)

$$\begin{array}{r} 9 \\ -\ 6 \\ \hline 3 \end{array}$$

Take 9's complement of 6 = −3.

Ans −3 = 0110.

GRAY CODE

It is a 4 bit numeric code. Decimal numbers, 0 to 15, are represented by 4-bit binary codes. This code is also known as unit distance code because two consecutive codes differs in only one bit position

→ It is an unweighted code, the bit position in the code group do not have any specific weight assigned to them.

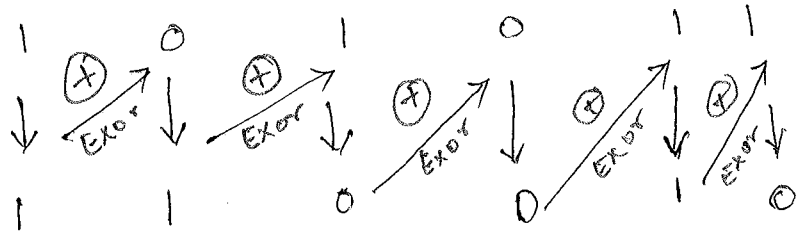
→ Due to this gray code is not suitable for arithmetic operations, but it finds application in analog-to-digital Converters.

Decimal digit	Gray code				Decimal digit	Gray code			
	G_3	G_2	G_1	G_0		G_3	G_2	G_1	G_0
0	0	0	0	0	8	1	1	0	0
1	0	0	0	1	9	1	1	0	1
2	0	0	1	1	10	1	1	1	1
3	0	0	1	0	11	1	1	1	0
4	0	1	1	0	12	1	0	1	0
5	0	1	1	1	13	1	0	1	1
6	0	1	0	1	14	1	0	0	1
7	0	1	0	0	15	1	0	0	0

Convert Gray code into binary.

EXOR table

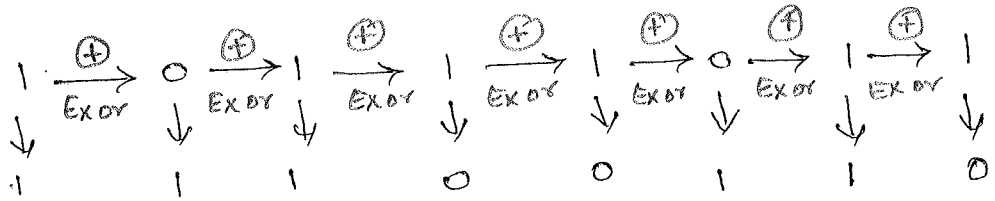
$1 \oplus 1 = 0$
$0 \oplus 0 = 0$
$1 \oplus 0 = 1$
$0 \oplus 1 = 1$



Ans: $(1100100)_2$

Convert binary to Gray code.

1 0 1 1 1 0 1 1



Ans: $(11100110)_{Gr}$.

LOGIC GATES.

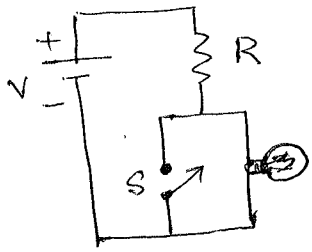
→ basic elements to make up digital system.
→ to operate on a number of binary input in order to perform particular logical function.
→ NOT, AND, OR, NAND, NOR, EX-OR (XOR), EX-NOR.
→ except EX-NOR all are available as monolithic I.C. form.
→ C.E. Shannon father of Information technology gave this two valued 0, 1 logic gates ON & OFF.

NOT / INVERTER:

SYMBOL.

→ Inversion or complementation
→ changes logic 1 to logic 0 and logic 0 to logic 1.
→ The bubble appearing on the output is the negation (inversion) indicator.

Switching circuit analogy:

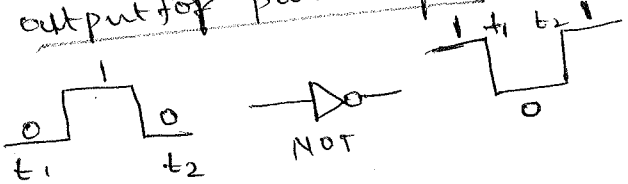


input	output
Switch open (low)	Lamp ON (high)
Switch closed (high)	Lamp OFF (low)

Truth table

input	output
0	1
1	0

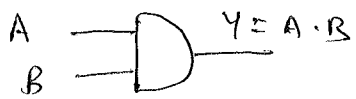
output for pulse input.



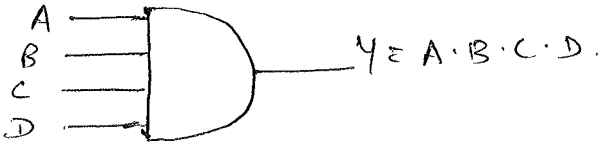
(IC: 7404)

AND GATE:

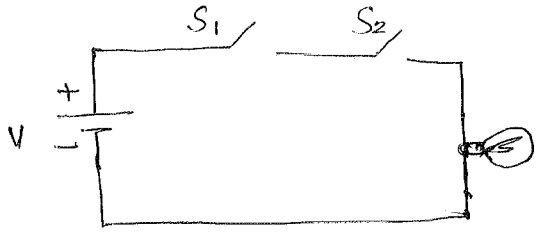
→ To perform logical multiplication.



2 input



4 input.

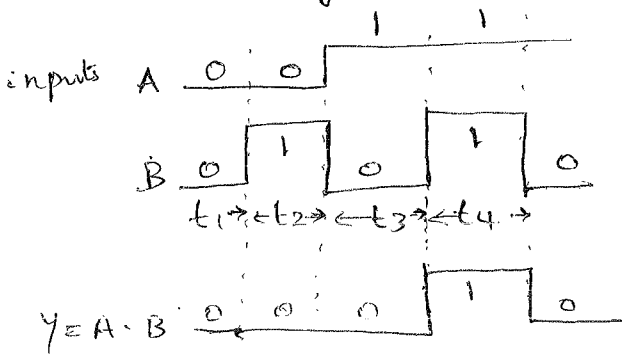


S_1 & S_2 closed then only lamp is ON; for all other cases circuit is open.

Truth table.

input		output
A	B	Y = A·B
0	0	0
0	1	0
1	0	0
1	1	1

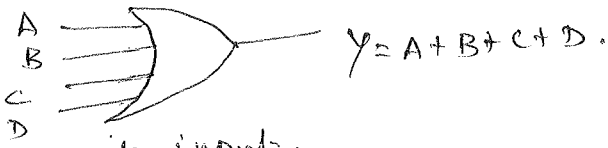
Timing diagram:



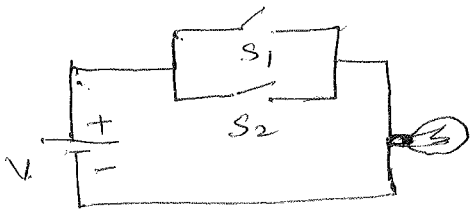
OR GATE:

→ OR gate is for logical addition.
→ produces high output when any of the input is high.
→ output is low only when all inputs are low.

2 input



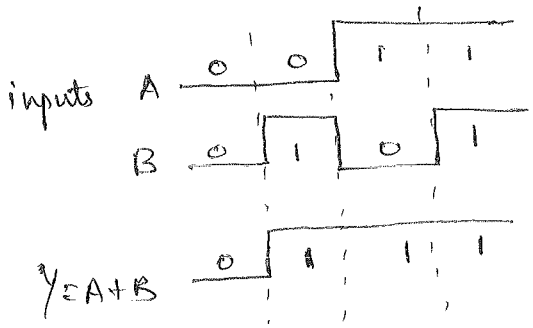
4 input.



→ S_1 & S_2 closed, lamp ON.
→ S_1 closed, lamp ON.
→ S_2 closed, lamp ON.
→ S_1 & S_2 open → lamp OFF.

Truth table

input		output
A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1



Timing diagram

BOOLEAN ALGEBRA.

1854 – George boole → logic – Boolean algebra.

1938 – C.E. Shannon – 2-valued boolean algebra called switching diagram.

→ Boolean algebra is nothing but expressions and formulas.

RULES IN BOOLEAN ALGEBRA.

1. Binary 1 will represent HIGH level and binary 0 will represent LOW level.
2. Complement is represented as bar over the letter eg $\bar{A}$ or A'.
3. AND function of two variables ~~either~~ are represented either by writing a 'dot' between the two variables, such as A.B or just as AB.
4. OR function of two variables is represented by '+' sign between two variables A+B.
5. Addition in boolean algebra 0+0 = 0, 1+0 = 1, 0+1 = 1, 1+1 = 1 (logical OR)
6. Multiplication in boolean algebra 0.0 = 0, 0.1 = 0, 1.0 = 0, 1.1 = 1.

Rules

1. A + 0 = A
2. A + 1 = 1
3. A . 0 = 0
4. A . 1 = A
5. A + A = A (Idempotent).

→ A can also be represented as 'x'.

→ B as 'y'

6. $A + \bar{A} = 1$ — Complementary.

7. $A \cdot A = A$ — Idempotent.

8. $A \cdot \bar{A} = 0$ — Complementary.

9. $\bar{\bar{A}} = A$ — Involution.

10. $A + AB = A$. — absorbtion law.

11. $A + \bar{A}B = A + B$ — "

12. $(A+B)(A+C) = A + BC$. — "

Rule 10: $A + AB = A$

Let $A + AB = A(1+B)$ $\quad \because 1 + B = 1$

$= A \cdot 1$ $\quad 1 \cdot A = A$

$= A$.

Rule 11:

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

Proof: by rule 10 $A + AB = A$.

∴ we can write

$\underbrace{A + AB}_{A} + \bar{A}B = A + B(A + \bar{A})$ → rule 6 $A + \bar{A} = 1$

$= A + B \cdot 1$ → rule 4 $A \cdot 1 = A$

$= A + B$

Rule 12:

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

Proof: $(A+B)(A+C) = AA + AC + BA + BC$.

$A \cdot A = A$ ∴ $A + AC + BA + BC = A(1 + C + B) + BC$

→ $1 + A = 1$

∴ $A + BC$

LAWS OF BOOLEAN ALGEBRA.

1. COMMUTATIVE LAW.

(i) $A+B = B+A$

The order in which the Variables are ORed makes no difference in output.

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

≡

B	A	B + A
0	0	0
0	1	1
1	0	1
1	1	1

(ii) $AB = BA$.

The order in which the Variables are ANDed makes no difference in output.

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

≡

B	A	BA
0	0	0
0	1	0
1	0	0
1	1	1

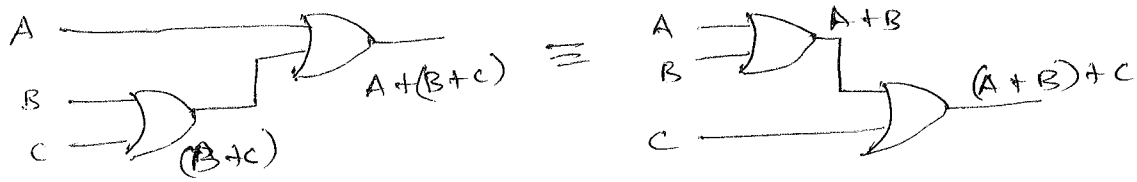
2. ASSOCIATIVE LAW:

i) $A + (B+C) = (A+B) + C$

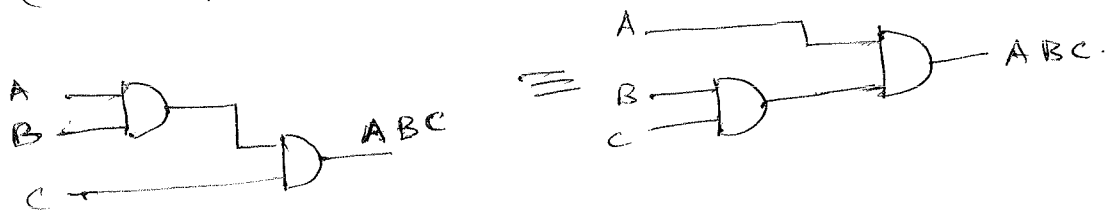
A	B	C	A+B	(A+B)+C
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

≡

A	B	C	B+C	A+(B+C)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

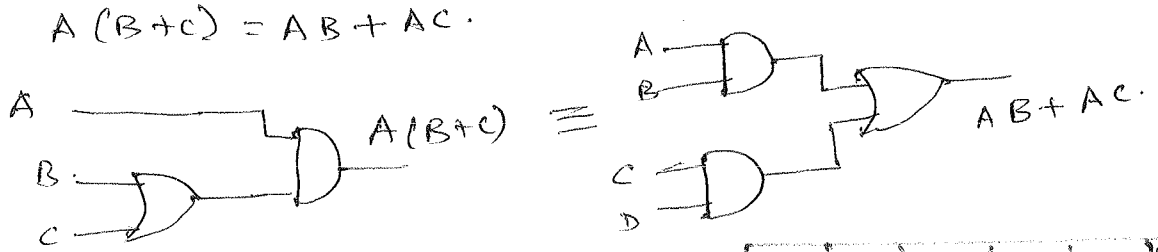


(ii) Ans (AB)C = A(BC)



3. DISTRIBUTIVE LAW.

$A(B+C) = AB + AC$.



A	B	C	B+C	A(B+C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

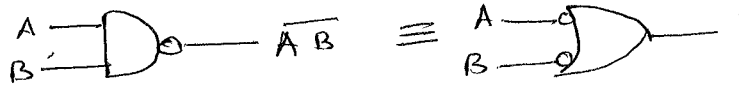
A	B	C	AB	AC	AB+AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Distributive property is often used in reverse, (i.e) given AB + AC, we replace it by its equivalent, A(B+C). As in ordinary algebra, this process is called factoring. We factored A out of expression AB + AC.

DEMORGAN'S THEOREM.

1. $\overline{AB} = \overline{A} + \overline{B}$.

Complement of product is equal to sum of complements.

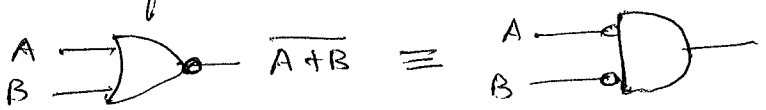


Truth table.

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

2. $\overline{A+B} = \overline{A}\,\overline{B}$.

Complement of a sum is equal to the product of complement.



Truth table:

A	B	$\overline{A+B}$	$\overline{A}\,\overline{B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

DUALITY THEOREM.

The duality theorem states that, starting with a boolean relation, we can derive another boolean relation by

1. Changing each OR sign to an AND sign.
2. " " AND " " " OR "
3. Complementing any 0 or 1 appearing in expression $A+0 = A$, $A \cdot 1 = A$.

ORed to ANDed.

$A(B+C) = AB + AC.$

Change OR to AND.

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

} both are same.

CONSENSUS THEOREM:

→ first find pair of terms, one of which contain Variable and the other contains its complement.

→ find the third term which should contain the remaining Variables from pair of terms eliminating selected Variables and its Complement

$AB + \bar{A}C + BC$ → Third term eliminate

Variable & its complement.

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

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

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

↳ (1) ↳ (1)

$= AB + \bar{A}C.$ (Hence proved).

BOOLEAN FUNCTIONS

→ Boolean algebra is an algebra that deals with binary Variables and logic operations.

→ Boolean function described by an algebraic expression consist of binary Variables, the constants 0 & 1 and logic operation symbols. For a given binary variables, the function can be equal to either '1' or '0'.

eg: $F_1 = x + y'z.$

The function $F_1 = 1$, if $x = 1$ or if both $y' + z = 1$, $F_1 = 0$ otherwise.

→ A boolean function can be represented in a truth table, number of binary combinations in the truth table is 2^n, where n is the number of variables in the function.

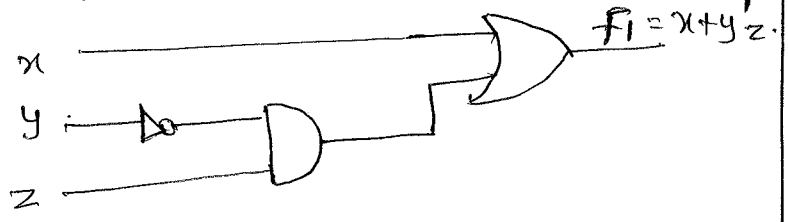
Truth table of f_1

x	y	z	f_1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Boolean expression

$$f_1 = x + y'z$$

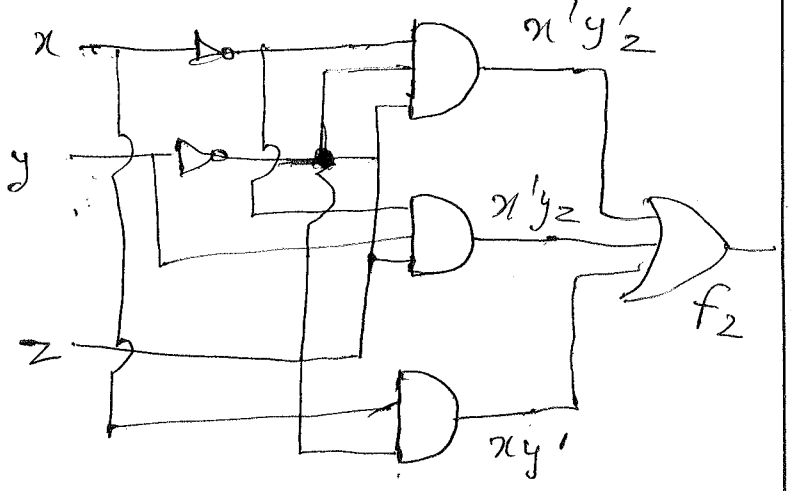
A boolean function can be transformed from an algebraic expression into a circuit diagram composed as logic gates.



By manipulating a boolean expression according to boolean algebra rules, it is sometimes possible to obtain a simplex expression for the same function and thus reduce the number of gates in the circuit and number of inputs to gate.

$$f_2 = x'y'z + x'yz + xy'$$

x	y	z	f_2
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

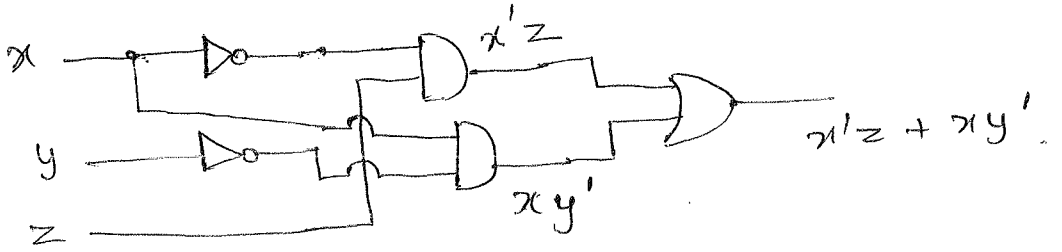


$f_2 = x'y'z + x'yz + xy'$ (simplified function)

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

(1)

$= x'z + xy'$



→ Number of gates is reduced.

→ Number of inputs to the gates is reduced.

→ Number of literal reduced.

ALGEBRIC MANIPULATION.

→ A literal is a primed or unprimed Variable.

→ When 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. It is not always possible to minimize both simultaneously.

→ There are no specific rules to follow to guarantee the final answer. Use of postulates, the basic theorems and any other manipulation method which becomes familiar with use.

Simplification of Boolean function using Boolean algebra.

1. Simplify the following boolean function to a minimum number of literals.

(1) $x + x'y$

$= (x + x')(x + y)$ → by distributive postulate.

↳ (1)

$= 1 \cdot (x + y)$

$= x + y$.

---

(2) $x(x' + y) = xx' + xy$.

$\because xx' = 0$.

$= xy$.

3) $x'y'z + x'yz + xy'$

$= x'z(y + y') + xy'$ $\because y + y' = 1$

$= x'z(1) + xy'$

$= x'z + xy'$.

4) $xy + x'z + yz$.

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

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

$= xy(1 + z) + x'z(1 + y)$

↳ (1) ↳ (1)

$= xy + x'z$.

5) $(x + y)(x + y')$

$= xx + xy' + xy + yy'$.

$= x + x(y + y') + yy'$.

↳ (1) ↳ (6)

$x + x = x$.

$= x + x + 0$

$= x$ //.

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.

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

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

→ The demorgan's theorem states that the complement of a function is obtained by interchanging AND and OR operators and complementing each literal.

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

$= (x'yz')'(x'y'z)'$ [change OR to AND]

$= (x+y'+z)(x+y+z')$ [change AND to OR and complement the literals.]

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

$F' = [x(y'z'+yz)]'$

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

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

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

Simplify the expression $AB + \bar{A}C + A\bar{B}C(AB+C)$

$AB+\bar{A}C + A\bar{B}C(AB+C) = AB + \bar{A}C + A\bar{B}C \cdot AB + A\bar{B}C \cdot C$ → (0)

$= AB + \bar{A}C + A\bar{B}C$ $\quad [B\bar{B}=0,\ C\cdot C = C]$

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

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

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

$= A + \bar{A} + \bar{C}$ → (1)

$= 1 + \bar{C} = 1$

2. $\overline{\overline{A\bar{B} + ABC} + A(B + A\bar{B})}$

$\overline{\overline{A\bar{B} + ABC} + AB + A\bar{B}}$ $\quad \because B + \bar{B} = 1$

$= \overline{\overline{A\bar{B} + ABC} + A(B+\bar{B})}$ → (1)

$= \overline{A(\bar{B} + BC)} + A$ $\quad$ by demorgans law $\overline{AB} = \bar{A} + \bar{B}$

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

$= \overline{(A+\bar{A}) + (\bar{B} + \bar{B}C)}$ $\quad$ by demorgans law $\overline{A+B} = \bar{A}\cdot\bar{B}$

→ (1)

$= \overline{1 + B \cdot \bar{B}C}$

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

$= \overline{1 + B\bar{B} + B\bar{C}}$ $\quad \because 1 + X = 1$

→ (0)

$= \overline{1 + B\bar{C}}$

$= \bar{1}$

$= 0$

3. Prove that $(A+B)(\bar{A}\bar{C}+C)(\overline{\bar{B}+AC}) = \bar{A}B$.

$(A+B)(\bar{A}\bar{C}+C)(\overline{\bar{B}+AC})$

$= (A\bar{A}C + AC + \bar{A}B\bar{C} + BC)(\bar{\bar{B}} \cdot \overline{AC})$ $\quad [\because A\bar{A} = 0,\ \overline{A+B} = \bar{A}\cdot\bar{B}]$

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

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

$= AC \cdot \bar{A}B + AC \cdot B\bar{C} + \bar{A}B\bar{C} \cdot \bar{A}B + \bar{A}B\bar{C} \cdot B\bar{C} + BC\bar{A}B +$

$BC \cdot B\bar{C}$

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

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

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

$= \bar{A}B$.

4. find the complement of the expression $Y = ABC + AB\bar{C} + \bar{A}\bar{B}C + \bar{A}BC$.

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

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

use demorgan's

$\overline{A+B} = \bar{A}\cdot\bar{B}$

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

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

$(A+B+C)$

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

We know $(A+B)(A+C) = A + BC$ [distributive law]

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

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

CANONICAL AND STANDARD FORMS.

PRODUCT TERM:

The AND function is referred to as product. The variable in a product term can appear as either in complemented or uncomplemented form eg: $x\bar{y}$

SUM TERM:

An OR function is referred as sum. Variable in a sum term can appear either in complemented or uncomplemented form. eg: $x + y + \bar{z}$.

SUM OF PRODUCT (SOP):

- logical sum of 2 or more logical product terms.
- An OR operation of AND operated variables.
- eg: $F = xy + \bar{x}z + yz$

PRODUCT OF SUM (POS):

- logical product of 2 or more logical sum terms.
- AND operation of OR operated variables.
- eg: $F = (x+y)(z+\bar{x})$

MINTERM:

A product term containing all the k variables of function in either complemented or uncomplemented form is called minterm.

Inputs			Minterm	
x	y	z	terms	Designation
0	0	0	$\bar{x}\bar{y}\bar{z}$	m_0

0	0	1	$x'y'z$	m_1
0	1	0	$x'yz'$	m_2
0	1	1	$x'yz$	m_3
1	0	0	$xy'z'$	m_4
1	0	1	$xy'z$	m_5
1	1	0	xyz'	m_6
1	1	1	xyz	m_7

CANONICAL SUM OF PRODUCT EXPRESSION

- If each term in SOP form contain all the literals then the SOP form is known as "standard or canonical SOP form". It is also called as sum of minterm.
- Can be given in a compact form by listing the decimal codes in correspondence with the minterm containing a function value of 1.

$$F(A,B,C) = \overset{\text{OR}}{\Sigma}_{\downarrow \text{AND}} m(0,5,6)$$

$$= m_0 + m_5 + m_6$$

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

(Note:
$x = 1$
$\bar{x} = 0$
Where x any variable)

Procedure:

1. Check each term in the given logic function. retain it if it is a minterm, Otherwise go to step 2. ~~the next term in the same manner~~
2. Find the variables that are missing in each product which is not a minterms.
3. Multiply the product by $(x + \bar{x})$, for each variable 'x' that is missing.

4. Multiply all the products that are missing and omit the redundant terms.

5. Expand the terms by applying boolean laws and reorder the literal in product term.

1. Obtain the canonical sum of product form. for the function $Y(A,B) = A + B$

. The given function contains two Variable A and B

. The Variable B is missing in first term and the Variable A is " " Second term.

$\therefore \quad A + B = A \cdot 1 + B \cdot 1$

$B + \bar{B} = 1$

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

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

A	B	m	
0	0	m_0	$\bar{A}\bar{B}$
0	1	m_1	$\bar{A}B$
1	0	m_2	$A\bar{B}$
1	1	m_3	AB

$Y(A,B) = AB + A\bar{B} + \bar{A}B.$

$= m_3 + m_2 + m_1$

$= \Sigma_m(1, 2, 3)$

---

2. Convert the boolean expression to canonical form.

$F = A + B'C$

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

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

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

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

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

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

---

3. $F = y' + xy + x'yz'$

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

$= (xz + xz' + x'z + x'z')y' + xyz + xyz' + x'yz'$

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

$= m_5 + m_4 + m_1 + m_0 + m_7 + m_6 + m_2$.

$= \Sigma m(0, 1, 2, 4, 5, 6, 7)$.

MAXTERM.

- sumterm containing all the k-variables of the function in either complemented or uncomplement form is called Maxterm.
- Each maxiterm can be obtained by OR operation of all variables of function.
- Uncomplemented / unprimed form — Value 0
  complemented / primed form — Value 1.

x	y	z	Maxterm	
			term	Designation
0	0	0	$x+y+z$	M_0
0	0	1	$x+y+z'$	M_1
0	1	0	$x+y'+z$	M_2
0	1	1	$x+y'+z'$	M_3
1	0	0	$x'+y+z$	M_4
1	0	1	$x'+y+z'$	M_5
1	1	0	$x'+y'+z$	M_6
1	1	1	$x'+y'+z'$	M_7.

Conversion of POS to canonical POS form:

Procedure:

1. Find missing literal in each sum terms.
2. add each sum term having missing literal with term formed by ANDing literal and its complements $+(x\bar{x})$
3. Expand the terms by applying distributive law.
4. Reduce the expression by omitting repeated sum terms if any.

CANONICAL PRODUCT OF SUM EXPRESSION.

If each term in the POS form contains all the literals then the POS form is known as Canonical POS (or) Sum of Maxterm.

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

$$= \overline{\Pi}_M(2,3).$$

AND → $\overline{\Pi}_M$ → OR

Express the boolean expression in canonical POS form:

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

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

$A + BC = (A+B)(A+C)$ distributive law.

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

$$= (A+\bar{B}+C)(A+\bar{B}+\bar{C})(A+B+C)(A+\bar{B}+C)$$
$$= (A+\bar{B}+C)(A+\bar{B}+\bar{C})(A+B+C)$$
$$= M_2 M_3 M_0$$

$$Y(A,B,C) = \overline{\Pi}_M(0,2,3)$$

---

2. $F = xy + x'z$.

Distributive law

$xy + x'z = (A+B)(A+C)$

$A = xy$, $B = x'$, $C = z$

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

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

(1)

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

$$= (x'+y+zz')(xx'+y+z)(x+yy'+z)$$

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

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

$$F(x,y,z) = M_0 M_2 M_4 M_5 = \overline{\Pi}_M(0,2,4,5).$$

CONVERSION BETWEEN CANONICAL FORM:

- The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function.
- The original function is expressed by minterms that make the function equal to 1, whereas its complement is 1 for those minterms that the function is a '0'.

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

Take complement (ie) write down the missing minterm

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

If we take complement of F' by demorgan's theorem we get a function (ie) Maxterm.

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

demorgan's theorem

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

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

$$F = M_0 M_2 M_3 = \Pi_M(0,2,3).$$

$$\therefore \boxed{m_j' = M_j}$$

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

Note:

Standard SOP = Standard POS.

subscript 'j' is complement of minterm with same subscript 'j' & vice versa.

∴ To convert one canonical form to other

1. Interchange Symbols Σ and Π

$$\Sigma \longleftrightarrow \Pi$$

2. List those numbers missing from the original form.
3. In order to find the missing term, see the total

Number of minterms or maxterms is 2^n, where 'n' is the number of binary variables in the function.

Express the following function in sum of minterms and product of maxterms. $F(A,B,C,D) = B'D + A'D + BD$. — (A)

Sum of minterms.

1. $B'D$

A & B literals are missing

$\therefore (A+A')(C+C')B'D$.

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

$AB'CD + AB'C'D + A'B'CD + A'B'C'D$ — (1)

2. $A'D$.

B and C literals are missing.

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

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

$A'BCD + A'BC'D + A'B'CD + A'B'C'D$ — (2)

3. BD

A and C are missing

$BD(A+A')(C+C')$

$BD(AC + AC' + A'C + A'C')$

$ABCD + ABC'D + A'BCD + A'BC'D$ — (3)

Sub (1), (2), (3) in (A).

$F(A,B,C,D) = AB'CD + AB'C'D + A'B'CD + A'B'C'D + A'BCD + A'BC'D + A'B'CD + A'B'C'D + ABCD + ABC'D + A'BCD + A'BC'D$

$= AB'CD + AB'C'D + A'B'CD + A'B'C'D + ABCD + ABC'D + A'BCD + A'BC'D$.

$= m_{11} + m_9 + m_3 + m_1 + m_7 + m_5 + m_{15} + m_{13}$

$= \Sigma m (1,3,5,7,9,11,13,15)$

Take complement for sum of misterm using demorgan's theorem; Product of maxterm.

$F' = \Sigma(0,2,4,6,8,9,10,12,14)'$

$F(A,B,C,D) = (m_0 + m_2 + m_4 + m_6 + m_8 + m_9 + m_{10} + m_{12} + m_{14})'$

$= m_0' \cdot m_2' \cdot m_4' \cdot m_6' \cdot m_8' \cdot m_9' \cdot m_{10}' \cdot m_{12}' \cdot m_{14}'$

$= M_0 M_2 M_4 M_6 M_8 M_9 M_{10} M_{12} M_{14}$

Product of Maxterm

$F(A,B,C,D) = \Pi_M (0,2,4,6,8,9,10,12,14).$

A Boolean function can be converted from an algebric expression to a product of maxterms by using a truth table and canonical conversion procedure. $F = xy + x'z$

First write truth table

x	y	z	$F = xy + x'z$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$\therefore F = xy + x'z$

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

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

$= m_7 + m_6 + m_3 + m_1$

$= \Sigma (1, 3, 6, 7)$

The missing terms in sum of misterm is product of maxterm(s) in truth table the value = 0.

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

Find sum of minterm and product of maxterm for

$f(x,y,z) = (xy+z)(y+xz)$

$= xyy + xxyz + zy + xzz$

$= xy + xyz + zy + xz$

$= xy(z+z') + xyz + (x+x')yz + (y+y')xz$

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

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

$= m_7 + m_6 + m_3 + m_5$

$= \Sigma m(3,5,6,7)$. Sum of minterm.

Product of maxterm $F'(x,y,z) = \Sigma(0,1,2,4) = (m_0 + m_1 + m_2 + m_4)$

$f(x,y,z) = m_0' \cdot m_1' \cdot m_2' \cdot m_4'$

$= M_0 \cdot M_1 \cdot M_2 \cdot M_4$

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

$(x'+y+z)$.

$f(A,B,C) = (A'+B)(B'+C)$, find standard SOP and standard POS forms.

$= A'B' + A'C + B'B + BC$.

$= A'B'(C+C') + A'C(B+B') + 0 + (A+A')BC$

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

$= A'B'C + A'B'C' + A'BC + ABC$.

$= m_1 + m_0 + m_3 + m_7$.

$= \Sigma m(0,1,3,7)$ Sum of minterms

Find Missing term $\Rightarrow F' = \Sigma m(2,4,5,6)$

$F'(A,B,C) = m_2 + m_4 + m_5 + m_6)' = m_2' \cdot m_4' \cdot m_5' \cdot m_6'$

$F(A,B,C) = M_2 \cdot M_4 \cdot M_5 \cdot M_6 = \Pi M(2,4,5,6)$

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

$(x'+y'+z)$.

MAP METHOD.

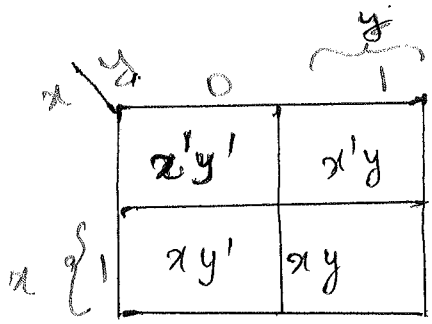
→ Provide a simple straight forward procedure for minimizing boolean function.
→ Pictorial form of truth table.
→ It is also called as Karnaugh map or "K-map".
→ map is a diagram made up of squares, with each square representing one minterm of function.
→ boolean functions can be expressed as sum of misterms.
→ Map represents a visual diagram of all possible ways a function may be expressed in standard form.

The simplified expressions produced by the map are always in one of the two standard forms: Sum of product or product of sum. It is assumed that the simplest algebric expression is one with a minimum number of terms and with the fewest possible number of literals in each term.
→ This produces a circuit diagram with a minimum number of gates and minimum number of inputs to the gate.

TWO VARIABLE MAP.

→ 2 Variable – 4 minterms, the map consist of 4 squares, one for each minterm.
→ Variable 'x' appears primed in row 0 and unprimed in row 1.
→ Variable 'y' appears primed in column '0' and unprimed in column '1'.

m_0	m_1
m_2	m_3



Two Variable map.

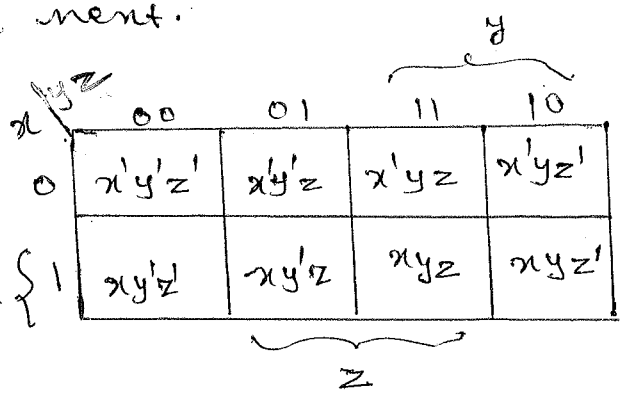
THREE VARIABLE MAP:

→ There are eight minterms for three binary variables.

→ map consist of 8 squares, the minterms are not arranged in binary sequence, but in a sequence similar to Gray code.

→ Only one bit changes in value from one adjacent column to the next.

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6



Any adjacent squares in the map differ by only one variable, which is primed in one square and unprimed in the other.

example:

m_5 and m_7 lie in two adjacent squares. Variable y is primed in m_5 and unprimed in m_7, whereas the other two variables are the same in both squares. From the postulates of Boolean algebra, it follows that the sum of two minterms in adjacent squares can be simplified to a single AND term consisting of only two literals.

Consider the sum of two adjacent squares such as m_5 and m_7.

$$m_5 + m_7 = xy'z + xyz = xz(y'+y) = xz.$$

The two squares differ by the variable y, which can be removed when the sum of two minterm is formed. Thus, any two minterms in adjacent squares that are ORed together will cause a removal of the different variable.

→ One square represents one minterm, giving a term of three literals.
→ Two adjacent squares represent a term of two literals.
→ Four adjacent squares represent a term of one literal.

FOUR VARIABLE MAP:

→ 16 minterms and the squares are assigned to each.
→ rows and columns are numbered in Gray code sequence with only one digit changing value between 2 adjacent rows or columns.

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

(y: 00, 01, 11, 10; x: 11, 10; z: 00, 01, 11, 10)

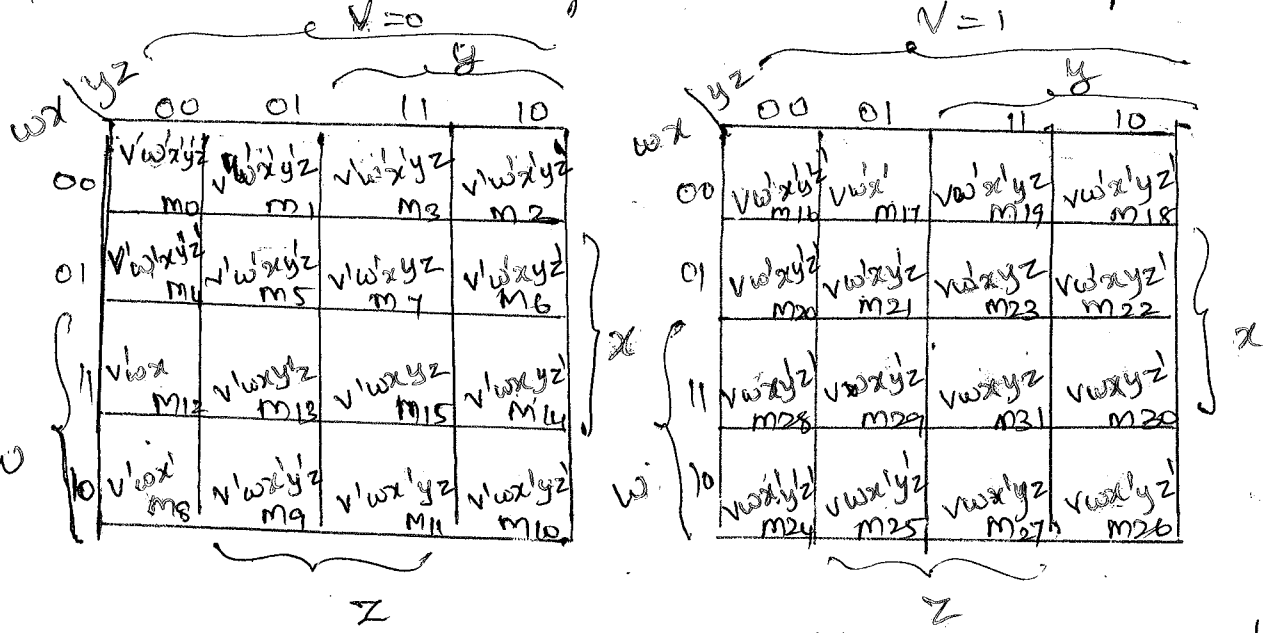
→ One Square represents one minterm, giving a term of four literals.
→ Two adjacent squares represent a term of 3 literals.
→ 4 adjacent squares represent a term of 2 literals.
→ 8 adjacent squares represent a term of one literals.
→ 16 adjacent squares represent the function equal to 1.

FIVE VARIABLE MAP.

→ 5 Variable map needs 32 squares and a six variable map needs 64 squares.

→ It consist of 2 four-Variable maps with variables V, W, X, Y and Z.

→ Variable V distinguishes between 2 maps.



→ The left-hand four variable map represents the 16 squares where $V=0$, and the other four-Variable map represents the squares where $V=1$.

→ Minterms 0 through 15 belong with $V=0$ and minterm 16 through 31 with $V=1$.

→ each square in $V=0$ map is adjacent to the corresponding square in $V=1$ map.

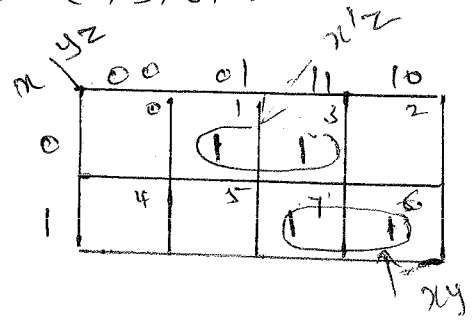
→ eg: Minterm 2 is adjacent to minterm 18.

→ Consider 2 half maps as being one on top of the other.

→ Any 2 squares that fall one over the other are Considered adjacent.

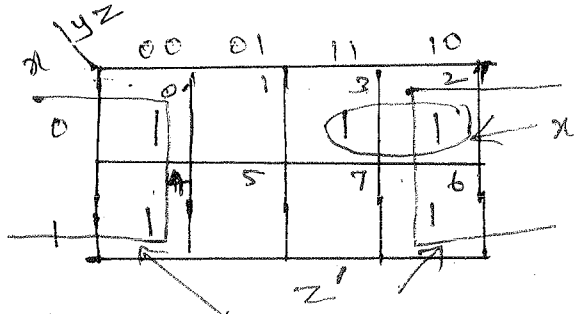
Karnaugh map method (K-map).

1. $f = \Sigma(1, 3, 6, 7)$



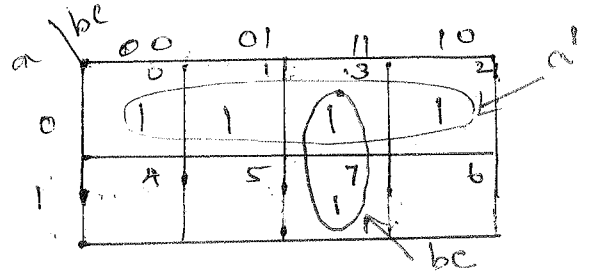
$f = x'z + xy.$

2. $f = \Sigma(0, 2, 3, 4, 6)$



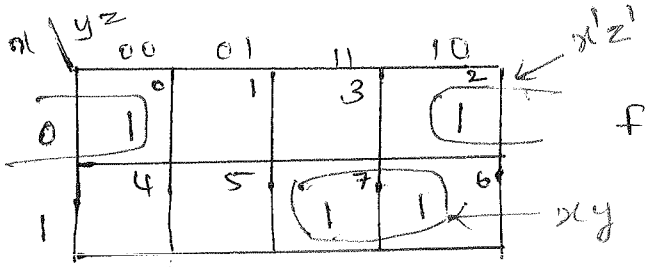
$f = x'y + z'.$

3. $F(a, b, c) = \Sigma(0, 1, 2, 3, 7)$



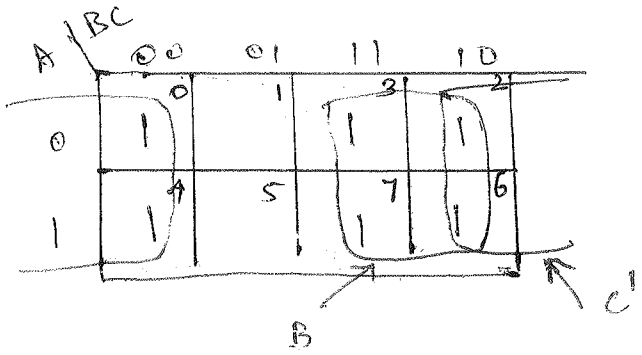
$f = a' + bc.$

4. $f(x, y, z) = \Sigma(0, 2, 6, 7)$



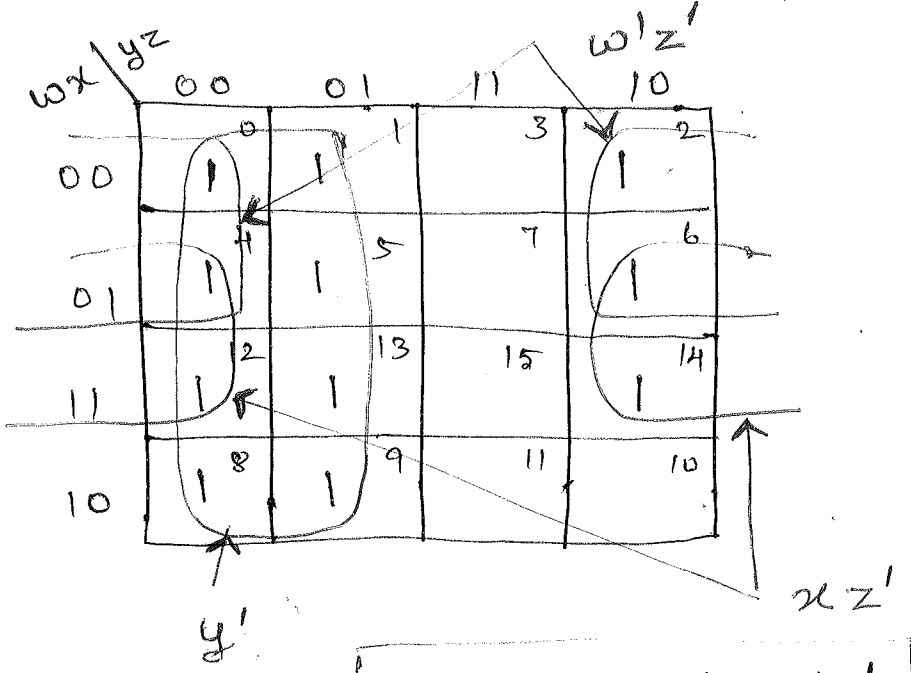
$f = x'z' + xy.$

5. $f(A, B, C) = \Sigma(0, 2, 3, 4, 6, 7)$



$f = B + c'$

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



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

7. $F = A'B'C' + B'CD' + A'BCD' + AB'C'$

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

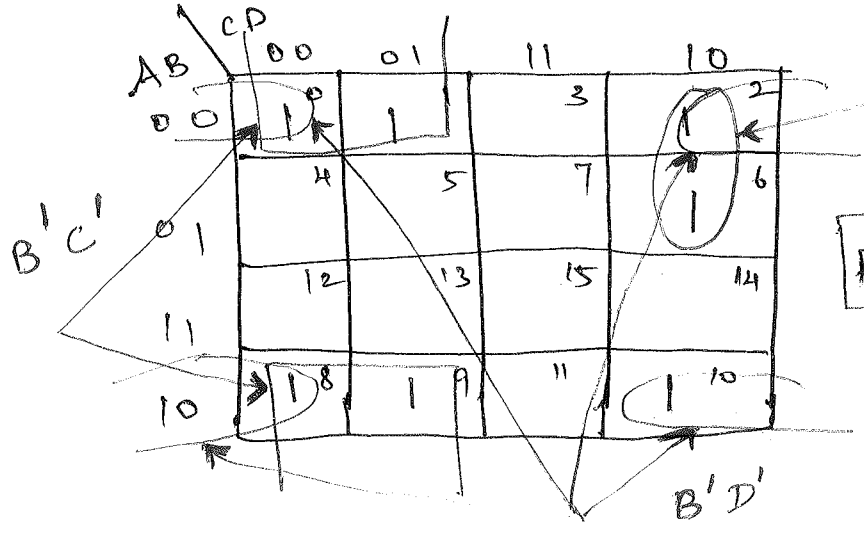
$+ AB'C'(D + D')$

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

$A'BCD' + AB'C'D + AB'C'D'$

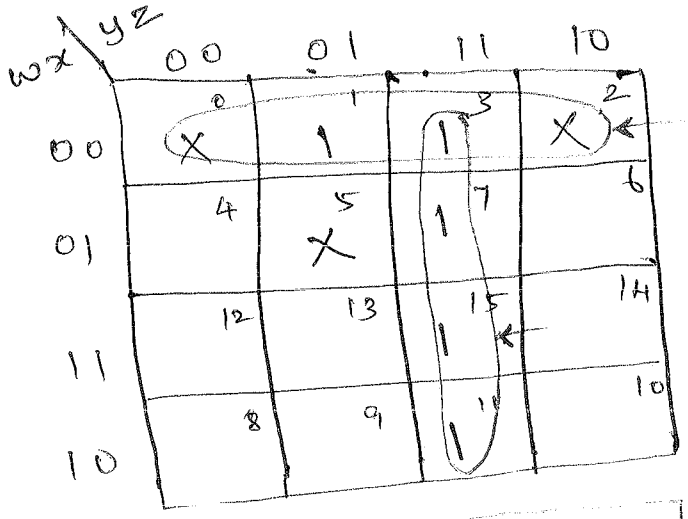
$= m_1 + m_0 + m_{10} + m_2 + m_6 + m_9 + m_8$

$F = \Sigma m(0, 1, 2, 6, 8, 9, 10)$



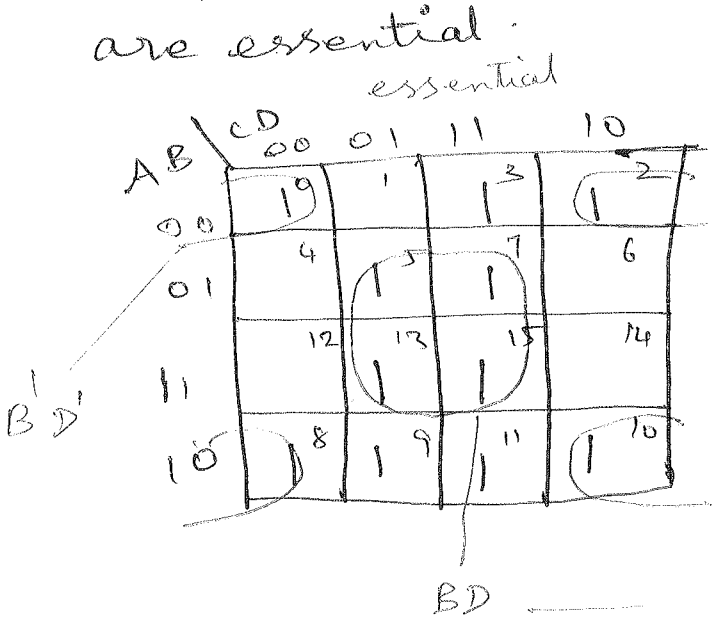
$$F = B'C' + B'D' + A'CD'$$

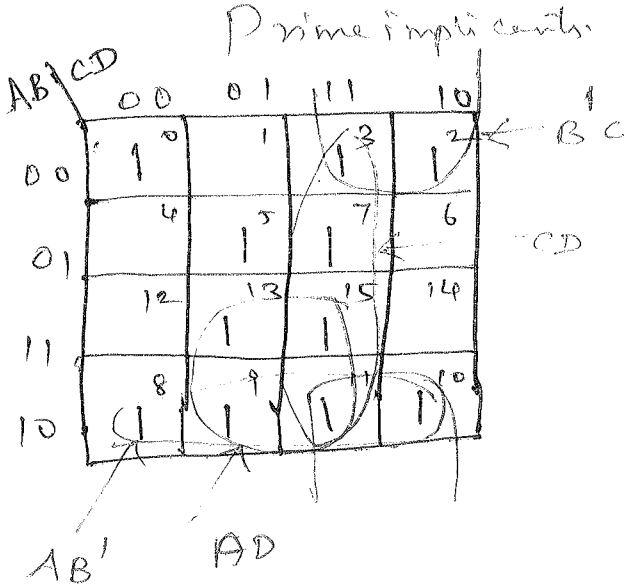
8. $F(w,x,y,z) = \Sigma m(1,3,7,11,15) + \Sigma d(0,2,5)$



$$F = \bar{w}\bar{x} + yz$$

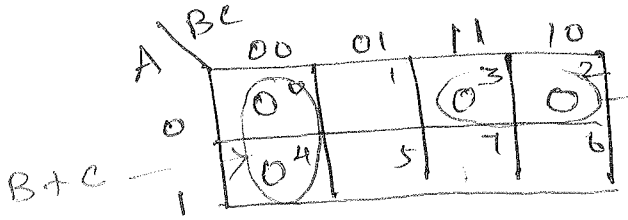
9. Find all Prime implicant for $\Sigma m(0,2,3,5,7,8,9,10,11,13,15)$ and determine which are essential.





$$F = BD + B'D' + AD \text{ or } AB' + CD \text{ or } B'C$$

10. $F = \prod_M(0,2,3,4)$



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

TABULATION METHOD.

① Simplify the following boolean function by using Quine - Mc cluskey Method $F(A,B,C,D) = \Sigma m(0,2,3,6,7,8,10,12,13)$

Minterm	Binary representation
m_0	0000
m_2	0010
m_3	0011
m_6	0110
m_7	0111
m_8	1000
m_{10}	1010
m_{12}	1100
m_{13}	1101

Minterm	Binary representation	
m_0	0000	← Zero ones
m_2	0010	One ones
m_8	1000	
m_3	0011	Two ones.
m_6	0110	
m_{10}	1010	
m_{12}	1100	
m_7	0111	three ones
m_{13}	1101	

Compare each binary number with every term in the adjacent next higher category and if they differ only by one position, put a check mark and copy the term in next column with '–' in the position that they differed.

Minterms	binary representation
0, 2	00–0 ✓
0, 8	–000 ✓
2, 3	001– ✓
2, 6	0–10 ✓
2, 10	–010 ✓
8, 10	10–0 ✓
8, 12	1–00
3, 7	0–11 ✓
6, 7	011– ✓
12, 13	110–

Minterm	binary representation
0, 2, 8, 10	–0–0
2, 3, 6, 7	0–1–

List prime implicants

Prime implicants	Binary representation
8, 12	1 _ 00
12, 13	110 _
0, 2, 8, 10	_ 0 _ 0
2, 3, 6, 7	0 _ 1 _

Prime implicants	m_0	m_2	m_3	m_6	m_7	m_8	m_{10}	m_{12}	m_{13}
8, 12						X		X	
12, 13 ✓								X	✓
0, 2, 8, 10 ✓	✓	X				X	X		
2, 3, 6, 7 ✓		X	✓	X	X				

$f = 110\_ + \_0\_0 + 0\_1\_ = ABc' + B'D' + A'c.$

② Simplify the following using tabulation method

$F(w, x, y, z) = \Sigma(1, 2, 3, 5, 9, 12, 14, 15) + \Sigma d(4, 8, 11)$

Minterm	binary representation
m_1	0001
m_2	0010
m_3	0011
m_5	0101
m_9	1001
m_{12}	1100
m_{14}	1110
m_{15}	1111
dm_4	0100
dm_8	1000

dm_{11} : 1011

Minterms	Binary repn.	✓minterm	binary rep	minterm	binary rep.
m_1	0001	1, 3 ✓	00–1	1, 3, 9, 11	–0–1
m_2	0010	1, 5	0–01		
m_4	0100	1, 9 ✓	–001		
m_8	1000	2, 3.	001–		
		4, 5	010–		
m_3	0011	4, 12	–100		
m_5	0101	8, 9	100–		
m_9	1001	8, 12	1–00		
m_{12}	1100				
m_{11}	1011	3, 11 ✓	–011		
		9, 11 ✓	10–1		
m_{14}	1110	12, 14	11–0		
m_{15}	1111	11, 15	1–11		
		14, 15	111–		

Prime implicants	binary representation
1, 5	0–01
2, 3	001–
4, 5	010–
4, 12	–100
8, 9	100–
8, 12	1–00
12, 14	11–0
11, 15	1–11
14, 15	111–
1, 3, 9, 11	–0–1

Only column 2 has single dot and hence the prime implicant corresponding to it $m_{(2,3)}$ is included in final expression, then search for multi dot columns. column 1 has multi-dot. $m_{1,5}$ or $m_{1,3,9,11}$ → include maximum number of minterms. leave column with don't care find the rest such that all minterms are covered.

Prime implicants	m_1	m_2	m_3	m_5	m_9	m_{12}	m_{14}	m_{15}	dm_4	dm_8	dm_{11}
1, 5 ✓	• X			(•)							
2, 3 ✓		(•)	• X								
4, 5				• X					• X		
4, 12						• X			• X		
8, 9					• X					• X	
8, 12						•				• X	
12, 14 ✓						(•)	• X				
11, 15								• X			• X
14, 15 ✓							(•)	• X			
1, 3, 9, 11 ✓	(•)		• X		• X						•

$F = 0\_01 + 001\_ + 11\_0 + 111\_ + \_01\_$

$= A'C'D + A'B'C + ABD' + ABC + B'D.$

3. Simplify the following five Variable Boolean expression using Quine Mccluskey Method.

$F = \Sigma m(0, 1, 9, 15, 24, 29, 30) + \Sigma d(8, 11, 31)$

Minterm	Binary rep.
m_0	00000
m_1	00001
m_9	01001
m_{15}	01111
m_{24}	11000
m_{29}	11101
m_{30}	11110
dm_8	01000
dm_{11}	01011
dm_{31}	11111

Minterm	binary rep.
m_0 ✓	00000
m_1 ✓	00001
dm_8 ✓	01000
m_9 ✓	01001
m_{24} ✓	11000
dm_{11} ✓	01011
m_{15} ✓	01111
m_{29} ✓	11101
m_{30} ✓	11110
dm_{31} ✓	11111

Minterm	Binary rep.
0, 1	0000 – ✓
0, 8	0 – 000
9	0 – 001
8, 9	0 1 00 – ✓
8, 24	– 1000
9, 11	0 10 – 1
11, 15	01 – 11
15, 31	– 1111
29, 31	111 – 1
30, 31	1111 –

Minterm	Binary rep.
0, 1, 8, 9	0 – 00 –

Ans: $F = -1000 + -111$
$+ 111 - 1 + 1111 - +$
$0 - 00 -$
$= BC'D'E + BCDE + ABCE +$
$ABCD + A'C'D'.$

Prime Implicant	Binary rep.
8, 24	– 1000
9, 11	0 10 – 1
11, 15	01 – 11
15, 31	– 1111
29, 31	111 – 1
30, 31	1111 –
0, 1, 8, 9	0 – 00 –

Prime implicants	m_0	m_1	m_9	m_{15}	m_{24}	m_{29}	m_{30}	dm_8	dm_{11}	dm_{31}
8, 24 ✓			⊗		◎			X		
9, 11			X						X	
11, 15				X					X	
15, 31 ✓				◎						X
29, 31 ✓						◎				X
30, 31 ✓							◎			X
0, 1, 8, 9 ✓	◎	X	X					X		