philosophy of Number Systems

— Number system is a basis for counting various items.

we are familiar with decimal number system with its 10 digits: 0, 1, 2, 3, 4, . . . , 9.

- But modern computers communicate and operate with binary numbers which are only 0 and 1

Ex: 18 → 10010.

In decimal it is represented by two digits where as in binary 5 digits.

∴ If decimal quantities are represented in binary form they take more digits.

— for large decimal number people have to deal with very large binary strings and therefore, they do not like working with binary numbers: This fact give rise to there new number systems.

octal, Hexadecimal and Binary coded decimal.

— These number systems represent binary number in a compressed form.

∴ these number systems are now widely used to compress long strings of binary.

Decimal number systems.

- In this we can express any number in units, hundreds, & thousand and so on.

Ex: 5678.9

$: 5000 + 600 + 70 + 8 + 0.9 = 5678.9$

can also be written as $(5678.9)_{10}$ where 10 subscript indicates the radix or base.

- In power of 10 we can write

$5\times10^3 + 6\times10^2 + 7\times10^1 + 8\times10^0 + 9\times10^{-1}$

- The left most bit which has the greatest weight is called most significant bit (MSB)
- The right most bit which has the least weight is called least significant bit (LSB).

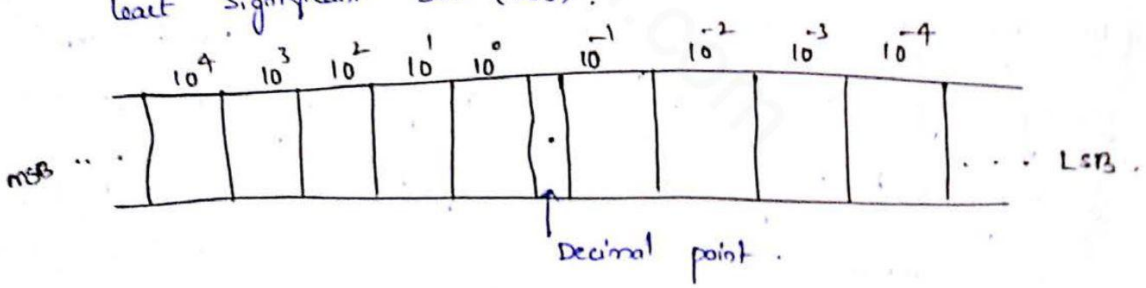


fig: Decimal position values as powers of 10.

Ex: Represent decimal number 98.72 in power of 10

$9\times10^1 + 8\times10^0 + 7\times10^{-1} + 2\times10^{-2}$

for 9 power of 10 is 1 for 2 power of 1/100 is -2

for 8 power of 10 is 0 for 7 power of 1/10 is -1

Binary Number System :-

Binary system with its two digits is a base - two system. The two binary digits are 1 and 0. Each binary digits are known as bit.

- In binary system weight is expressed as power of 2.

Ex: Represent binary number 1101.101 in power of 2 and find its decimal Equivalent

$$N = 1\times 2^3 + 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 1\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3}$$

$$= 8+4+0+1+\tfrac{1}{2}+0+\tfrac{1}{8}$$

$$= (13.625)_{10}$$

1.3 octal Number System :-

It uses first eight digits of decimal number System 0, 1, 2, 3, 4, 5, 6, and 7. As it uses 8 digits, its base is 8

Ex: Represent octal number 567 in power of 8 and find its decimal Equivalent

$$5\times 8^2 + 6\times 8^1 + 7\times 8^0 = 5\times 64 + 6\times 8 + 7\times 1 = (375)_{10}$$

1.4 Hexadecimal Number System :-

- Its base is 16, it is having 16 digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Since its base is a power of 2, (2^4) it is easy to convert hexadecimal to binary numbers and vice versa.

- It is useful for human communications with a computer
- Each hexadecimal digit represents a group of four binary digits called nibbles.

www.jntuworldupdates.org

Relation b/n decimal, binary, Hexadecimal, Octal

Decimal	Binary	Hexadecimal	Octal
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	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

Represent hexadecimal number 3FD in power of 16 and find its decimal equivalent

$$3\times16^2 + F\times16^1 + D\times16^0$$

$$\downarrow \quad \downarrow \quad \downarrow$$

$$3 \quad F \quad D$$

$$3\times256 + 15\times16 + 13\times1$$

$$= 768 + 240 + 13 = (1021)_{10}.$$

5. **Counting in Radix (Base):—**

— In general, a number represented in radix r, has r characters in its set and r can be any value. They are 0, 1, ... r-1.

Radix	characters in set
2	0, 1
3	0, 1, 2
4	0, 1, 2, 3
⋮	
7	0, 1, 2, 3, 4, 5, 6,
8 (octal)	0, 1, 2, 3, 4, 5, 6, 7
⋮	⋮
(10) (Decimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
⋮	
(16) (Hexadecimal)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Ex: find decimal Equivalent of $(231.23)_4$

$= 2\times 4^2 + 3\times 4^1 + 1\times 4^0 + 2\times 4^{-1} + 3\times 4^{-2}$

$= 32 + 12 + 1 + 0.5 + 0.1875$

$= (45.6875)_{10}$

Ex: Count from 0 to 9 in radix 5

Sol Radix 5 has 5 characters.

0, 1, 2, 3, 4, 5 10, 11, 12, 13, 14

5 in radix $(10)_5$ —— $(5)_{10}$

$1\times 5^1 + 0\times 5^0 = 5+0 = 5_{10}$

$(11)_5$ —— $(6)_{10}$

$(11)_5 = 1\times 5^1 + 1\times 5^0 = 5+1 = 6_{10}$

$(7)_{10}$ — $(12)_5 \Rightarrow 1\times 5^1 + 2\times 5^0 = 7_{10}$

$(8)_{10}$ —— $(13)_5 \Rightarrow 1\times 5^1 + 3\times 5^0 = 8_{10}$

$(9)_{10}$ —— $(14)_5 \Rightarrow 1\times 5^1 + 4\times 5^0 = 9_{10}$

Ex: Represent $(13)_{10}$ in octal

$(13)_{10}$ —— $(?)_8$

$8 \,\big|\, 13$

$1 - 5$

$= (15)_8$

check $(15)_8$ —— $(?)_{10}$

$1\times 8^1 + 5\times 8^0$

$= 8+5 = (13)_{10}$

Number System Conversion

Binary to octal :-

— for octal Numbers base is 8 and the base for binary is 2. i.e the base for octal number is the third power of base for binary numbers

∴ By grouping of 3 digits of binary numbers and then converting Each group digits to its octal Equivalent

Ex: octal to Binary Convert $(111\ 101\ 100)_2$ to octal

— $(7\ 5\ 4)_8$.

2.2 Octal to Binary

— Each digit of the octal number is individually converted to its binary Equivalent to get octal to binary conversion of the number

Ex: $(634)_8$ — $(?)_2$

$= (110\ 011\ 100)_2$

Ex: $(725.63)_8$ — $(?)_2$

$(111010101.110011)_2$

2.3 Binary to Hexadecimal

— Base for hexadecimal is 16 and the base for binary is 2.

The base for hexadecimal number is the fourth power of the base for binary numbers

www.jntuworldupdates.org

∴ By grouping 4 digits of binary number and ...

-ng Each group digit to its hexadecimal Equivalent.

We can Convert binary number to its hexadecimal Equivalent

Ex: Convert $(1101\ 1000\ 1001\ 1011)_2$ to hexadecimal Equivalent

$(D\ B\ 9\ B)_H$

2.4 Hexadecimal to Binary Conversion:

— Each digit of the hexadecimal number is individually converted to its binary Equivalent to get hexadecimal to binary conversion of the number

Ex: Convert $(5A9.B4)_H$ to binary

$(0101\ 1010\ 1001 . 1011\ 0100)_2$

Ex: Convert $(3FD)_H$ to binary

$(0011\ 1111\ 1101)_2$.

2.5 octal to hexadecimal:

1. Convert octal to binary number
2. Convert binary number to its hexadecimal Equivalent

Ex: $(615)_8$ —— $(\ )_{16}$

(i) $(\underline{110}\ \underline{001}\ \underline{101})_2 \rightarrow (ii) = (18D)_H$

1.6 Hexadecimal to octal Conversion

(i) Convert hexadecimal number to its binary Equivalent

(ii) Convert binary number to its octal Equivalent

Ex: $(25B)_H$ —— $(?)_8$

$(0010\ 0101\ 1011)_2$.

001 001 011 011

1 1 3 3 = $(1133)_8$

Converting any Radix to Decimal

- In general numbers Can be represented as

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

N = Number in Decimal

A = Digit

r = Radix

n = The no. of digits in the integer portion of the number

m = The no. of digits in the fractional portion of number

Ex: $(1101.1)_2$ —— $(?)$

$= 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1}$

$= 8+4+0+1+0.5$

$= (13.5)_{10}$

Ex: $(475.25)_8 \longrightarrow (?)_{10}$

$N = 4\times8^2 + 7\times8^1 + 5\times8^0 + 2\times8^{-1} + 5\times8^{-2}$

$= 256 + 56 + 5 + 0.25 + 0.078125$

$= (317.328125)_{10}$

Ex: $(3102.123)_4 \longrightarrow (?)_{10}$

$N = 3\times4^3 + 1\times4^2 + 0\times4^1 + 2\times4^0 + 1\times4^{-1} + 2\times4^{-2}$

$= 192 + 16 + 0 + 2 + 0.25 + 0.125$

$= (210.375)_{10}$

Ex: $(614.15)_7 \longrightarrow (?)_{10}$

$= 6\times7^2 + 1\times7^1 + 4\times7^0 + 1\times7^{-1} + 5\times7^{-2}$

$= 294 + 7 + 4 + 0.142857 + 0.102$

$= (305.24468)_{10}$

Conversion of Decimal Numbers to any Radix Number

Two Steps:

Step 1: Convert integer part. — This is done by Successive division method.

Step 2: Convert fractional part — This is done by Successive multiplication method.

(i) Successive Division for integer part:—

Ex: $(37)_{10}$ ——— $(?)_2$

$$
\begin{array}{r|l}
2 & 37 \\
2 & 18 - 1 \\
2 & 9 - 0 \\
2 & 4 - 1 \\
2 & 2 - 0 \\
 & 1 - 0 \uparrow
\end{array}
= (\overset{2^5}{1}\overset{2^4}{0}\overset{2^3}{0}\overset{2^2}{1}\overset{2^1}{0}\overset{2^0}{1})_2 \qquad 32 + 4 + 1 = 37
$$

Ex: $(3509)_{10}$ ——— $(\ )_{16} - (\ )_H$

$$
\begin{array}{r|l}
16 & 3509 \\
16 & 219 - 5 \\
 & 13 - 11 \uparrow
\end{array}
= (DB5)_{16}
$$

Ex: $(54)_{10}$ — $(?)_4$

$$
\begin{array}{r|l}
4 & 54 \\
4 & 13 - 2 \\
 & 3 - 1 \uparrow
\end{array}
= (312)_4
$$

(ii) Successive multiplication for fractional parts

- The number to be converted is multiplied by radix of new number, producing a product that an integer part and a fractional part.
- The integer part of the product becomes a numberal in the new radix

www.jntuworldupdates.org

— The fractional part is again multiplied by the radix. The process is repeated until fractional part reaches 0 or the new radix number is carried out to sufficient digits. The integer part of each product is read downward to represent the new radix number.

Ex: $(0.8125)_{10}$ —— $(?)_2$

Fraction	Radix	Result	Recorded carry.	
0.8125×2	$= 1.625$	$= 0.625$	1	MSB
0.625×2	$= 1.25$	$= 0.25$	1	
0.25×2	$= 0.5$	$= 0.5$	0	
0.5×2	$= 1.0$	$= 0.0$	1	LSB

$(0.1101)_2$

$(0.1101)_2$ — $(0.8125)_{10}$

Ex: $(0.95)_{10}$ —— $(?)_2$

0.95×2	$= 1.9$	$= 0.9$	1
0.9×2	$= 1.8$	$= 0.8$	1
0.8×2	$= 1.6$	$= 0.6$	1
0.6×2	$= 1.2$	$= 0.2$	1
0.2×2	$= 0.4$	$= 0.4$	0
0.4×2	$= 0.8$	$= 0.8$	0
0.8×2	$= 1.6$	$= 0.6$	1

— 0.8 is repeated and if we multiply further, we will get repeated sequence. If we stop here, we get 7 binary digits (0.1111001)

Ex: $(0.1287062)_{10}$ —— $(?)_{16}$

1. 0.1287062×16 — 2.0625 2

0.0625×16 — 1.0 1

Ans:

$(0.21)_{16}$

Ex: Convert $(24.6)_{10}$ — $(?)_2$

Step1: Separate out integer and fraction part.

Integer part : 24 , fraction part : 0.6

Step2: find Equivalent binary number for integer part.

2 | 24
2 | 12 — 0
2 | 6 — 0
2 | 3 — 0
1 — 1

$(11000)_2$

Step3: find Equivalent binary number for fractional

0.6×2	$= 1.2$	0.2	1
0.2×2	$= 0.4$	0.4	0
0.4×2	$= 0.8$	0.8	0
0.8×2	$= 1.6$	0.6	1
0.6×2	$= 1.2$	0.2	1

$(0.10011)_2$.

$\therefore (11000.10011)_2$

Questions

1) Convert $(725.25)_8$ to dec, bin, hex dec.

Sol (i) $= 7\times8^2 + 2\times8^1 + 5\times8^0 + 2\times8^{-1} + 5\times8^{-2}$

$= 448 + 16\times5 + \frac{1}{4} + \frac{5}{64}$

$= 469 + 0.25 + 0.078125$

$= (469.328125)_{10}$

(ii) $(725.25)_{10}$ — $(\ )_2$

$(111010101.010101)_2$

(iii) 001 1101 0101 . 0101 0100

$(1\ D\ 5\ .\ 54)_{16}$

Weighted Codes:

In this Each digit position of the number represents a Specific weight.

Ex: If number is 567 then wt of 5 is 100
wt of 6 is 10
wt of 7 is 1

Ex: Binary, BCD

- BCD is an abbreviation for binary coded Decimal. BCD is a numeric code in which Each digit of a decimal number is represented by a separate group of bits.

Complement Representation of negative numbers.

In digital Computers, to simply the Subtraction operation and for logical manipulation Complements are used.

There are two types of Complements for each radix System:

The radix complement and diminished radix complement.

The first is referred to as the r's complement and Second as the (r-1)'s complement.

for Example, in binary System we substitute base value 2 in place of r to refer Complements as 2's complement and 1's complement. In decimal number system, we substitute base Value 10 in place of r to refer complements as 10's Complement and 9's complement.

1's complement Representation:

The In 1's Complement Representation, the number changes all 1's to zero's and all zero's to 1's.

find 1's complement of $(1101)_2$

1101 ← number

0010 ← 1's complement.

find 1's Complement of 10111001

10111001 ← number

01000110 ← 1's complement.

2's complement Representation.

The 2's complement is the binary number that results when we add 1 to the 1's complement. It is given as

2's complement = 1's complement + 1

The 2's complement form is used to represent negative numbers.

find 2's complement of $(1001)_2$

```
  1 0 0 1  ← number
  0 1 1 0  ← 1's complement
    +   1
  -------
  0 1 1 1  - 2's complement
```

find 2's complement of $(1010001)_2$

```
  1 0 1 0 0 0 1  ← number
  0 1 0 1 1 1 0  — 1's complement
            +1
  -----------
  0 1 0 1 1 1 1
```

1's complement Subtraction

Subtraction of binary numbers can be accomplished by the direct method by using the 1's complement method, which allows to perform subtraction using only addition.

For subtraction of two numbers we have two cases.

- Subtraction of smaller number from larger number and
- Subtraction of larger number from smaller number

Subtraction of smaller number from larger number.

method:

1. Determine the 1's complement of the smaller number.
2. Add the 1's complement to the large number.
3. Remove the carry and add 1 to the result.

1. Subtract $(101011)_2$ from $(111001)_2$ using the 1's complement.

```
  1 1 1 0 0 1
  0 1 0 1 0 0   == 1's Complement
  ---------------
(1) 0 0 1 1 0 1
    ---------->+1
  ---------------
  0 0 1 1 1 0   => final answer.
```

$$\begin{array}{r} 57 \\ 43 \\ \hline 14 \end{array}$$

Subtraction of large number from smaller number.

method:

1. Determine 1's complement of larger number.
2. Add the 1's complement to the smaller number.
3. Answer is in 1's complement form. To get the answer in true form take the 1's complement and assign negative sign to the answer.

Subtract $(111001)_2$ from $(101011)_2$ using the 1's complement method.

```
  1 0 1 0 1 1
- 0 0 0 1 1 0   = 1's complement of 111001
  -----------
  1 1 0 0 0 1   = Answer in 1's complement form.
  -
  - 0 0 1 1 1 0   = Answer in true form.
```

2's Complement Subtraction

In 2's complement subtraction, the subtraction is accomplished by only addition. Let us see the methods for 2's complement subtraction.

→ Subtraction of smaller number from larger number.

method:

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

Subtract $(101011)_2$ from $(111001)_2$ using the 2's complement method.

```
      1 1 1 0 0 1                    0 1 0 1 0 0
2's comp → 0 1 0 1 0 1 .                     +1
      ---------------                ---------
      (1) 0 0 1 1 1 0                0 1 0 1 0 1
      ↓
      Discard
```

0 0 1 1 1 0 — final answer.

Subtraction of larger number from smaller number:

method:

1. Determine the 2's complement of larger number.
2. Add the 2's complement to the smaller number.
3. Answer is in the 2's complement form. To get the answer in the true form take the 2's complement and assign negative sign to the answer.

www.jntuworldupdates.org

Subtract $(111001)_2$ from $(101011)_2$ using 2's complement method

```
  1 0 1 0 1 1
  0 0 0 1 1 1
  -----------
  1 1 0 0 1 0
```

```
  000110
     +1
  ------
  000111  - 2's complement.
```

↳ Answer in 2's complement form

```
  00 11 01
     +1
  --------
- 001110
```

Decimal	Signed magnitude	1's complement	2's complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1010	1001	1010
-7	1111	1000	1001

www.jntuworldupdates.org

Subtract $(111001)_2$ from $(101011)_2$ using 2's complement method.

```
  1 0 1 0 1 1
  0 0 0 1 1 1
  -----------
  1 1 0 0 1 0
```

```
  0 0 0 1 1 0
        + 1
  -----------
  0 0 0 1 1 1   - 2's complement.
```

↳ Answer in 2's complement form

```
  0 0 1 1 0 1
        + 1
  -----------
- 0 0 1 1 1 0
```

Decimal	Signed magnitude	1's complement	2's complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1010	1001	1010
-7	1111	1000	1001

230 − 120

2 | 230	2 | 120
2 | 115 - 0	2 | 60 - 0
2 | 57 - 1	2 | 30 - 0
2 | 28 - 1	2 | 15 - 0
2 | 14 - 0	2 | 7 - 1
2 | 7 - 0	2 | 3 - 1
2 | 3 - 1	1 - 1
1 - 1	

11100110 −

01111000

10000111

+1

10001000

11100110

10001000

(1) 01101110

$2^6+2^5+2^3+2^2+2^1$

$64+32+8+4+2$

110

Signed Binary Numbers

Signed Binary Numbers mainly used for represent sign of the number.

However, because of hardware limitations, in computers both positive and negative numbers are represented with only binary digits.

The left most bit (sign bit) in the number represents sign of the number.

The sign bit is 0 positive numbers

The sign bit is 1 negative numbers.

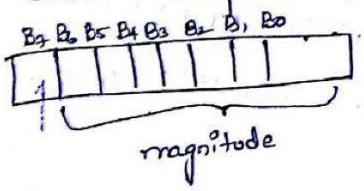
These numbers are represented by the signed magnitude format.

Below fig shows the sign magnitude format for 8-bit signed number.

Here, the most significant bit (MSB) represents sign of the number.

if MSB is 1, number is negative and if MSB is 0, number is +ve, The remaining bits represent magnitude of the number.

Here some examples for sign-magnitude numbers.



+6

2 | 6
2 | 3 - 0
1 - 1

00000110

-14

2 | 14
2 | 7 - 0
2 | 3 - 1
1 - 1

10001110

+24 00011000

-64 1100 0000

In case of unsigned 8-bit binary numbers the decimal range is 0 to 255..

For Signed magnitude 8-bit binary numbers the largest magnitude is reduced from 255 to 127.

because we need to represent both positive and negative numbers.

maximum +ve number 0111 1111 = +127

maximum -ve number 1111 1111 = -128

```
 0 1 0
 1 0 1
 -----
 0 0 0
```

We have seen +ve and -ve number representation in the signed magnitude format.

This is the only way to represent +ve numbers!

however there are two more ways represent negative numbers:

Signed -1's Complement representation.

Signed 2's complement representation

```
 1001
 0011   9
 ----   3
 0110   6
```

Let us see how -6 represents in three formats:

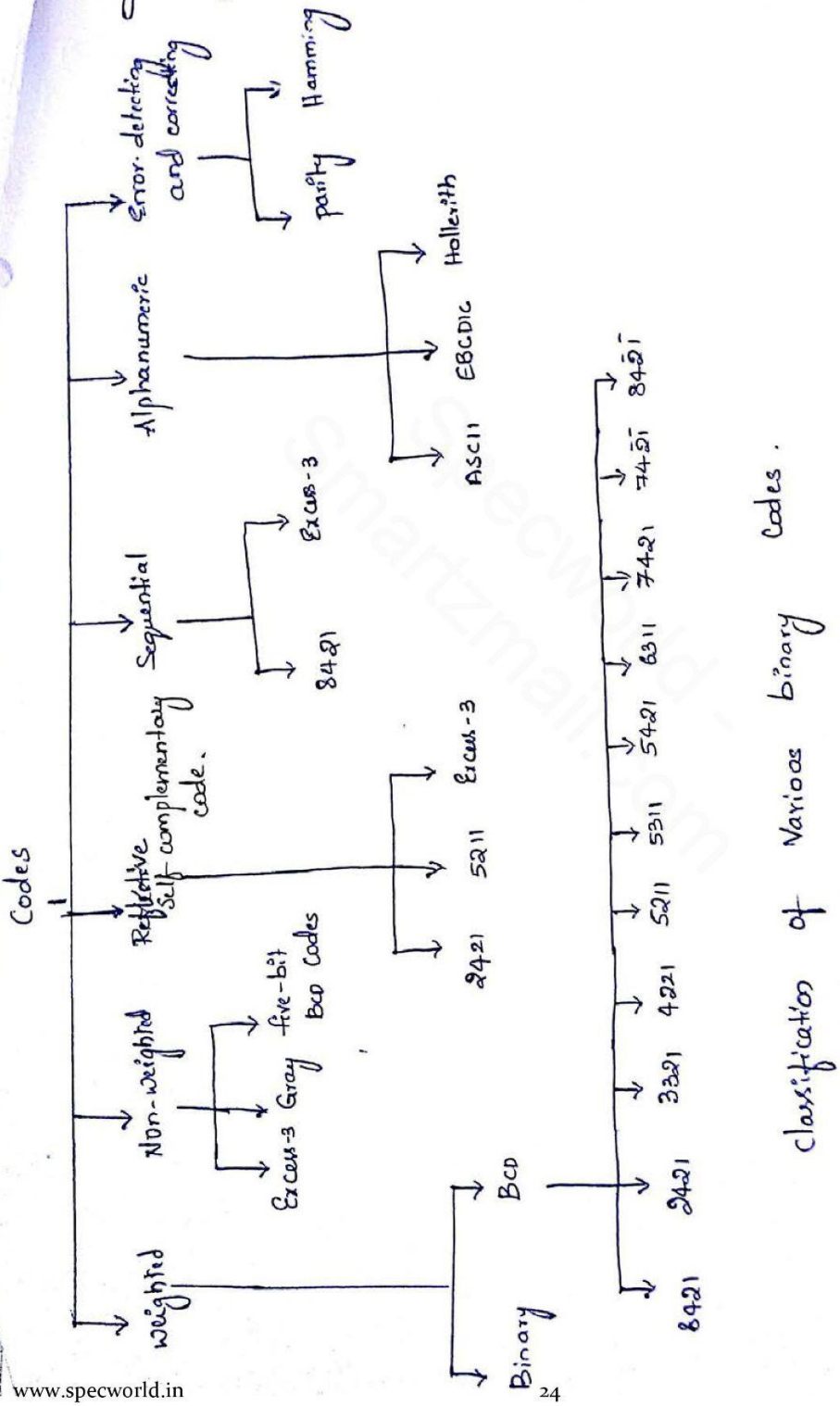
Signed - magnitude representation 10000110

Signed 1's Complement representation 11111001

Signed 2's complement representation 11111010

Binary codes :-

The digital data is transmitted, stored and represented as groups of binary digits (bits). The group of bits, also known as binary code,



Classification of Various binary codes.

Alphanumeric codes : The code which consists of both alphabets and numbers.

ASCII code:- It is a 7-bit code which includes 0-9 - bit code upper case alphabets, lower case alphabets, and other symbols. It is preceded by 011 for decimal no.

0 → 011 0000
9 → 011 1001

A — 65 → 100 0001
Z — 90 → 10 11010
a — 97 → 110 0001
z — 122 — 111 1010.

$$\begin{array}{r|l} 2 & 90 \\ 2 & 45 - 0 \\ 2 & 22 - 1 \\ 2 & 11 - 0 \\ 2 & 5 - 1 \\ 2 & 2 - 1 \\ & 1 - 0 \end{array}$$

EBCDIC:- It is a 8-bit code. It is preceded by 1111 for decimal digits.

a — 129
z — 154.

Decimal Digit	BCD Code 8	4	2	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

— In multidigit coding, each decimal digit is individual coded with 8421 BCD code

Ex: $(58)_{10} \xrightarrow{8421}$ 0101 1000.

— If we want to represent same 58 in binary the equivalent is $(111010)_2$ i.e we require only 6 digits that means 8421 BCD is less efficient than pure binary number.

— Advantage: It is Easy to convert between it and decimal.
— Disadvantage: low efficiency, the arithmetic operations are more complex than they are in pure binary.

2421 code:

Decimal Digit	2421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1110
9	1111

2: Non weighted Codes: These are not assigned with any weight to Each digit position. Ex:

Excess -3 Code:

It is a modified form of a BCD number. It can be derived from the natural BCD Code by adding 3 to Each Coded number

Ex: 12 in BCD as 0001 0010

Now adding 3 to Each digit we get Excess-3 code as

$$\begin{array}{cc} 0001 & 0010 \\ 11 & 11 \\ \hline 0100 & 0101 \end{array}$$ — Excess-3 code for (12)10

BCD Addition:

The addition of two BCD numbers can be best understood by considering the three cases that occur when two BCD digits are added.

Sum Equal to 9 or less with carry 0.

let us consider additions of 3 and 6 in BCD.

```
6     0110 ← BCD for 6.
+3    0011 ← BCD for 3
-----------
9     1001 ← BCD for 9.
```

```
  9
+ 3
----
 12
```

Sum greater than 9 with carry 0.

```
6     0110
8     1000
----------
14    1110 ← invalid BCD (1110) > 9.
```

Whenever invalid BCD no. occurs, the sum has to be corrected by the addition of six (0110) in the invalid BCD no. as shown below.

```
6     0110
8     1000
----------
14    1110 — invalid BCD no.
      0110
----------
0001 0100 ← BCD for 14.
  1    4
```

Sum Equals 9 or less with carry 1.

let us consider addition of 8 and 9 in BCD.

```
 8        1000
 9        1001
---      --------
17      00011001   — incorrect BCD result.

        00010001   — add 6 for correction
            0110
        --------
        00010110
        --------
          1   7    — BCD for 17.
```

```
24      0010     0100
18      0001     1000
        ----     ----
        0011     1100   invalid BCD no.
                        1100>9.
                 0110   add 6 for correction.
        ----     ----
        0100     0010   — propagate carry to
                          next higher digit.
```

```
589     0101     1000     1001
199     0001     1001     1001
---     ----     ----     ----
788     0111     0010     0010
                 0110     0110
                 ----     ----
        0111     1000     1000   — corrected sum.
```

```
 79        0111  1001          99
-26        0111  0011 73-9's   26
 53        ----  ----  complement 73
           1110  1100  for BCD 26.
           ----------
                 0110        1100 > 9 add 6 to result.
           ----------
           1111  0010        1111 > 9.
           0110
           ----------
           10101 0010
           ----------
           01 01 0011        BCD for 53.
```

```
 89        99        1000  1001
-54        54        0100  0101
           45        ----------
                     1100  1110        1110 > 9 So add 6.
                           0110
                     ----------
                     1101  0100        1101 > 9 add 6.
                     0110
                     ----------
                     10011 0100
                     ----------
                     0011 0101         BCD for 35.
```

175	0001	0111	0101	
326	0011	0010	0110	
501				
	0100	1001	1011	1011 > 9 .
			0110	add 6 to 1011 for correction .
		1 1		
	0100	1010	0001	1010 > 9 . add 6.
	1	0110		
	0101	0000	0001	propagate carry to next higher digit .

BCD Subtraction :

Addition of Signed BCD numbers can be performed by using 9's or 10's Complement methods.

A negative BCD number can be Expressed by taking the 9's or 10's complement.

8	9	8	8045	456
-2.	2	+7	-9161	-436
	7.	15		
		1		
		6		

79 − 26. perform the BCD subtraction using 9's complement method.

BCD Subtraction using 10's complement.

Regular Subtraction

```
  8
 -2
  6
```

10's complement Subtraction.

```
  9
  8
  7
 +1
  8
```

```
  8
  8
 16.
```

```
  8        1000
 +8        1000
 16       10000
           0110
           0110 — 6.
```

- find the 10's complement of negative numbers.
- Add two numbers using BCD addition
- if carry is not generated find the 10's complement of the result otherwise find the 9's complement of the result.

```
→ 79        99        79        0111 1001
  -26       26        +77       0111 0100
   53       73                  1110 1101      1101 > 9.
            +1                        0110
                                 1111 0011      1111 > 9
                                 0110
                                 1101 0011
                                   5    3
```

discard 1

www.jntuworldupdates.org

Excess-3 :- It is a modified form of BCD. & add 3 in this code add 3 to BCD of each digit.

	BCD		Excess-3	
12 —	0001	0010	0100	0101

It is a Self Complementing code.

Ex: BCD

98 1001 1000

Excess 1100 1011

0011 0100

99

98

01

Excess-3 addition:

Ex:- $(592)_{10}$ — 1000 1100 0101

9's comp(592) 0111 0011 1010

Excess-3. addition

To perform Excess-3 addition we have to

- Add two Excess-3 numbers.
- if carry = 1 → add 3 to the sum of two digits
  = 0 → subtract 3.

```
  8      1011
+ 6      1001
 14
     00010100
     00110011
     01000111
       1. 4
```

```
1   0100 — Excess-3
+2  0101 — Excess-3
 3  1001
Sub3 -0011
     0110 — Excess-3 for 3.
```

Decimal Digit	Excess-3 code
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

— In Excess-3 code, we get 9's complement of a number by just complementing each bit. Due to this Excess-3 code is called self-complementing code.

Ex: 592 in Excess-3 1000 1100 0101 9's complement of 592

is 999
592
407

407 in Excess-3 0111 0011 1010

i.e nothing but just complementing the 592 in Excess-3 you will get 9's complement of 592.

For 403 in Excess-3 0111 0011 0110

9's complement of 403 is 1000 1100 1001

$$\begin{array}{r} 999 \\ -403 \\ \hline 596 \\ \hline \end{array}$$

596 in Excess-3 is 1000 1100 1001

Gray Code :

It is a special case of unit - distance code. In this bit patterns for two consecutive numbers differ in only one bit position. It is also called as <u>cyclic code</u>.

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

For gray code any two adjacent code groups differ only in one bit position. It is also called as reflected code, because two least significant bits for 4 through 7 are the mirror images of those for 0 to 3.

— Similarly, the three LSB for 8 to 15 are the mirror images of are from 0 to 7.

Binary to Gray conversion.

Let us represent binary numbers as $B_1, B_2, B_3, B_4, \ldots B_n$ and its Equivalent gray code as $G_1, G_2, G_3 \ldots G_n$

$$G_1 = B_1$$
$$G_2 = B_1 \oplus B_2$$
$$G_3 = B_2 \oplus B_3$$
$$G_4 = B_3 \oplus B_4$$
$$\vdots$$
$$G_n = B_{n-1} \oplus B_n$$

Ex: 1) $(10110110)_2 \xrightarrow{Gray} (11101101)$

2) $(10101010110)_2 \xrightarrow{Gray} (11111111101)$

3) $(110010)_2 \xrightarrow{Gray} (101011)$

1.20 Floating Point Numbers :

➤➤➤ [University Exam : 2003-2004 III]

- We need to use many bits in order to represent very large **integer** (whole) numbers.
- The same problem is faced when numbers with both integer and fractional parts such as 64.8629 needs to be represented.
- The **floating point number system** is the remedy to this problem. It is based on scientific notation and is capable of representing very large and very small numbers without increasing the number of bits.
- This system can be used for representing numbers which have both integer and fractional parts. It uses power of ten.

1.20.1 Parts of Floating Point Numbers :

- The floating point numbers are also called as the real numbers. They consist of two parts and a sign. The two parts are :

  1. Mantissa 2. Exponent

- **Mantissa** is the part of a floating point number which represents the magnitude of the number.
- **Exponent** is the part of the floating point number that represents the number of places that the decimal point (or binary point) is to be moved.

Ex. 1.20.1 : Convert the following decimal number into a floating point number.

N = 541, 369, 841

Soln. : The given number can be represented in the floating point number as follows :

$$\underbrace{0.541369841}_{\text{Mantissa}} \times 10^{\overset{\text{Exponent}}{9}}$$

Thus mantissa is the magnitude and exponent (9) represents the number of places that the decimal point is moved.

1.20.2 Binary Floating Point Numbers :

For binary floating point numbers, the format is defined by ANSI / IEEE standard 754 – 1985 in the following three forms :

1. Single precision floating point binary numbers.
2. Double precision floating point binary numbers.
3. Extended precision floating point binary numbers.

1.20.3 Single Precision Floating Point Binary Numbers :

- Fig. 1.20.1 shows the 32 bit standard format for a single precision floating point binary number.
- The MSB is assigned to indicate the sign (S). The next eight bits indicate the exponent (E) and the last 23 bits are used for indicating the mantissa or the fractional part (F).
- In the mantissa (F) part, the binary point is assumed to placed to the left of the 23 bits as shown in Fig. 1.20.1.
- The eight bit exponent (E) actually represents a biased exponent. That means this exponent is obtained by adding 127 to the actual exponent.
- The bias is used to allow very large or very small numbers without requiring a separate sign bit for the exponent. The biased exponent will allow the actual exponent to occupy the range of – 126 to + 128.

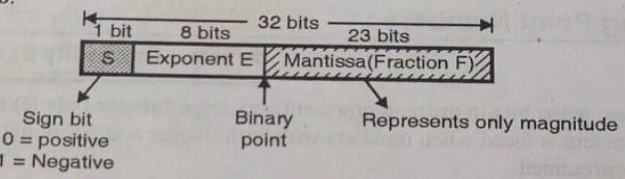


Fig. 1.20.1 : Format of single precision floating point binary numbers

Ex. 1.20.2 : Represent the following binary number using a single precision floating point format. $N = (1001010011101)_2$

Soln. :

Step 1 : Represent the given number in fractional form

The given binary number contains 13 bits. Let us express it as 1 plus a fractional binary number by moving the binary point 12 places to the left and then multiply by appropriate power of two as follows :

$$N = 1001010011101 \quad \text{...Given 13 bit number}$$

Binary point

$$= 1.001010011101 \times 2^{12} \quad \text{...Exponent} \qquad \text{...(1.20.1)}$$

12 bits mantissa

Step 2 : Decide the values of S, E and F :

- Assuming the number to be the positive one the sign bit will have a zero value.

$$\therefore S = 0$$

- The exponent is actually 12 because we have shifted the binary point by 12 places. In order to get the biased exponent we have to add 127 to the actual exponent.

$\therefore$ Biased exponent E = Actual exponent + 127

$\therefore$ E = $12 + 127 = (139)_{10}$

Convert the biased exponent to binary

2	139	
2	69	1
2	34	1
2	17	0
2	8	1
2	4	0
2	2	0
2	1	0
	0	1

LSB

MSB

$\therefore$ E = $(10001011)_2$

The mantissa is the fractional part (F) of the binary number. Because there is always a 1 to the left of the binary point as shown in Equation (1.20.1), it is not included in the mantissa. The 12 bits in the mantissa are same as those in Equation (1.20.1) but eleven zeros are appended to these 12 bits to form the 23 bit mantissa as follows :

F = 001010011101 00000000000 23 bit mantissa

eleven zeros

12 bits from Equation (1.20.1)

Step 3 : Write the complete floating point number :

The complete single precision floating point number is as follows :

S	E	F
0	10001011	00101001110100000000000
	8 bits	23 bits

Conversion from floating point to binary :

- Now let us see the conversion from floating point number to binary number.
- Use the formulae given below for the same.

$$\text{Binary number} = (-1)^S (1 + F) (2^{E-127}) \quad \ldots(1.20.2)$$

- The values of S, F and E are to be obtained from the given floating point number.

Ex. 1.20.3 : Convert the following floating point number into equivalent binary number.

S	E	F
1	10001011	00101001110100000000000

Soln. :

From the given floating point number we get,

$S = 1, E = 10001011$

$F = 001010011101$

$$\therefore \text{ Binary number } = (-1)^S (1 + F)(2^{E-127})$$

$$\text{But } E = \underset{128}{1} + \underset{8}{0} + \underset{2}{0} + \underset{1}{1} = 139$$

$$\therefore \text{ Binary number } = (-1)^1 (1 + 0.001010011101)(2^{139-127})$$

$$= -(1.001010011101) \times 2^{12}$$

$$= -1001010011101$$

Note :

- Since the biased exponent (E) can take any value between – 126 to + 128, it is possible to express extremely large and small numbers using the floating point number systems.
- A 32 bit floating point number can replace a binary integer number having 129 numbers.
- The number 0.0 is represented by all 0s and infinity is represented by all 1s in the exponent and all zeros in the mantissa.

Review of Boolean algebra

→ system of mathematical logic

Boolean Algebra Terminology

Variable → symbol, represents an arbitrary elements of an boolean algebra.

→ any single or a fn of several variables can have either a 1 or 0 value.

Constant → fixed value.

Complement → represented by a bar or (') (eg) $\bar{A}$, A'

$A=0$, $\bar{A}=1$

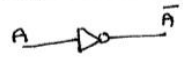
Literal → each occurrence of a variable in Boolean function either in a complemented or an uncomplemented

Boolean function → constructed by connecting the boolean constants and variables with the boolean operation.

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

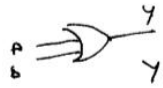
Boolean Algebra

Inversion



OR

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



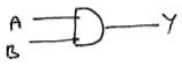
$Y = A+B$

Output of an OR gate is 1 when either input is 1

AND operation

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

$Y = A \cdot B$



Output will be 1 when both the inputs are high

Boolean laws

Laws of union

1. $A+0 = A$
2. $A+1 = 1$

Laws of intersection

1. $A \cdot 0 = 0$
2. $A \cdot 1 = A$

Laws of Tautology

1. $A+A = A$
2. $A \cdot A = A$

Laws of complements

1. $A+\bar{A} = 1$
2. $A \cdot A = 0$

Law of double complement

1. $\bar{\bar{A}} = A$

Laws of commutation

1. $A+B = B+A$
2. $AB = BA$

Laws of association

1. $A+(B+C) = (A+B)+C$
2. $A(BC) = (AB)C$

Laws of Distribution

1. $A(B+C) = AB+AC$
2. $(A+B)(C+D) = AC+AD+BC+BD$

Laws of Absorption

1. $A(A+B) = A$
2. $A+AB = A$
3. $A(\bar{A}+B) = AB$
4. $AB+\bar{B} = A+\bar{B}$
5. $A\bar{B}+B = A+B$

De morgan's Theorem

1. $\overline{A+B} = \bar{A} \cdot \bar{B}$
2. $\overline{A \cdot B} = \bar{A}+\bar{B}$

(1) Simplify : $\underline{\underline{x}} + x'y$

$\because A = A+AB$

$\Rightarrow$

$x + xy + x'y$

$x + y(x+x') = \underline{\underline{x+y}}$

ENGINEERING
Apprise Education, Reprise Innovations

(5) Simplify the following Boolean expression

$a\bar{b}\bar{c} + a\bar{b}c + abc$

$= a[\bar{b}\bar{c} + \bar{b}c + bc]$

$= a[\bar{b}(\bar{c}+c) + bc]$

$= a[\bar{b} + bc]$ $\quad \because A + \bar{A}B = A + B$

$= a[\bar{b} + c]$

$= a\bar{b} + ac$

(6) Simplify the following Boolean expressions to a minimum number of literals.

$(X+Y)(X+\bar{Y})$

$= XX + X\bar{Y} + YX + Y\bar{Y}$

$= X + X\bar{Y} + YX + 0$

$= X[1 + \bar{Y} + Y] = X$

EASY ENGINEERING
Apprise Education, Revrise Innovations

Consensus Theorem

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

$AC + B\bar{C} + AB$ [C is one term, $\bar{C}$ is one term Extra product term of the remaining 2 variables] $AB \rightarrow$ Consensus term

$F = AC + B\bar{C} + AB$

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

$= AC + B\bar{C} + ABC + AB\bar{C}$

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

(2) prove that $ABC + D\bar{C} + ABD = ABC + D\bar{C}$ using consensus theorem.

$ABC + D\bar{C} + ABD$

$= ABC + D\bar{C} + ABD(C+\bar{C})$

$= ABC + D\bar{C} + ABCD + AB\bar{C}D$

$= ABC(1+D) + \bar{C}D(1+AB)$

$= ABC + D\bar{C}$ proved.

(3) simplify $XY + \bar{X}Z + YZ$

$XY + \bar{X}Z + YZ = XY + \bar{X}Z + YZ(X+\bar{X})$

$= XY + \bar{X}Z + XYZ + \bar{X}YZ$

$= XY(1+Z) + \bar{X}Z(1+Y)$

$= XY + \bar{X}Z$

EASY ENGINEERING
Apprise Education, Reprise Innovations

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

By applying duality

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

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

$= AC + B\bar{C} + ABC + AB\bar{C}$

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

$= AC + B\bar{C}$

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

(5) prove that $ABC + D\bar{C} + ABD = ABC + D\bar{C}$ using consensus theorem

$$F = ABC + D\bar{C} + ABD$$
$$= ABC + D\bar{C} + ABD(C + \bar{C})$$
$$= ABC + D\bar{C} + ABCD + AB\bar{C}D$$
$$= ABC(1 + D) + D\bar{C}(1 + AB)$$
$$= ABC + D\bar{C}$$ hence proved.

Representation of Logic functions. SOP & POS Forms

→ Boolean fn → variables in complemented or an uncomplemented form

<u>Literal</u>

→ each occurence of a variable in either a complemented or an uncomplemented form

→ product term → either a literal or a product.

<u>Sum of Product form</u> (SOP)

→ symbolic representations of the OR and AND fns.

→ product → group of literals that are ANDed together

→ sum → group of literals that are ORed together.

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

→ SOP → disjunctive normal form or disjunctive normal formula.

<u>Product of Sum Form</u>

→ sum terms ANDed together.

$$f(A,B,C) = (A+B)\cdot(\bar{B}+C)$$
$$f(P,Q,R,S) = (P+Q)\cdot(R+\bar{S})\cdot(P+S)$$

Standard SOP and standard POS forms

→ If each term in SOP/POS form contains all the literals then the SOP form is known as standard or canonical SOP form.

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

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

Converting Expressions in standard SOP or POS form

Steps to convert SOP to standard SOP

1. Find the missing literal in each product term if any
2. AND each product term having missing literals with terms form by ORing the literal and its complement
3. Expand the terms by applying distributive law & reorder the literals in the product terms.
4. Reduce the expression by omitting repeated product terms if any.

① Convert the given expression in standard SOP form.

$$f(A,B,C) = AC + AB + BC$$

$$AC + AB + BC$$

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

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

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

③ Define canonical form. Express $F = BC' + AC$ in a canonical SOP form.

$$F = BC' + AC$$
$$= BC'(A+A') + AC(B+B')$$
$$= ABC' + A'BC' + ABC + AB'C$$

④ Determine the canonical SOP form of $f(x,y,z) = (xy+\bar{z})(y+x\bar{z})$

$$f(x,y,z) = (xy+\bar{z})(y+x\bar{z})$$
$$= \left[xy(z+\bar{z}) + \bar{z}(x+\bar{x})(y+\bar{y})\right] \; y + x\bar{z}$$
$$= \left[xyz + xy\bar{z} + (x\bar{z} + \bar{z}\bar{x})(y+\bar{y})\right] \cdot \left[y(x+\bar{x})(z+\bar{z}) + x\bar{z}(y+\bar{y})\right]$$
$$= \left[xyz + \underline{xy\bar{z}} + x\underline{y\bar{z}} + \bar{x}y\bar{z} + \right.$$
$$\left[(yx + y\bar{x})(z+\bar{z}) + x\bar{z}(y+\bar{y})\right]$$
$$= (xyz + xy\bar{z} + \bar{x}y\bar{z} + x\bar{y}\bar{z} + \bar{x}\bar{y}\bar{z})$$
$$(xyz + \bar{x}y\bar{z} + \underline{xy\bar{z}} + \bar{x}\cdot yz + \underline{xy\bar{z}} + x\bar{y}\bar{z})$$
$$= \left[xyz + xy\bar{z} + \bar{x}y\bar{z} + x\bar{y}\bar{z} + \bar{x}\bar{y}\bar{z}\right]$$
$$\left[xyz + \bar{x}y\bar{z} + xy\bar{z} + \bar{x}yz + x\bar{y}\bar{z}\right]$$
$$= xyz + xy\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}\bar{z} + x\bar{y}\bar{z}$$

ENGINEERING ASY
Apprise Education, Reprise Life

Steps to convert POS to standard POS

(5)

1. Find the missing literals in each sum term if any
2. OR each sum term having missing literals with terms form by ANDing the literal and its complement
3. Expand the terms by applying distributive law and reorder the literals in the sum terms.
4. Reduce the expression by omitting repeated sum terms if any.

(1) convert the given expression in standard POS form.

$$f(A,B,C) = (A+B)(B+C)(A+C)$$

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

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

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

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

(2) convert the given expression in standard POS form

$$f(A,B,C) = (A+B)\cdot(B+C)$$

$$f(A,B,C) = (A+B)\cdot(B+C)$$

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

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

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

91 / 263

ENGINESPIRE
© Apprise Education, Aspire Innovations ©

M. notations: Minterms and Manterms

→ minterm → each individual term in SOP

Manterm → each individual term in POS

Variables				minterms	Manterms
decimal	A	B	C	m_i	M_i
0	0	0	0	$\bar{A}\bar{B}\bar{C}$ – m_0	$A+B+C$ – M_0
1	0	0	1	$\bar{A}\bar{B}C$ – m_1	$A+B+\bar{C}$ – M_1
2	0	1	0	$\bar{A}B\bar{C}$ – m_2	$A+\bar{B}+C$ – M_2
3	0	1	1	$\bar{A}BC$ – m_3	$A+\bar{B}+\bar{C}$ – M_3
4	1	0	0	$A\bar{B}\bar{C}$ – m_4	$\bar{A}+B+C$ – M_4
5	1	0	1	$A\bar{B}C$ – m_5	$\bar{A}+B+\bar{C}$ – M_5
6	1	1	0	$AB\bar{C}$ – m_6	$\bar{A}+\bar{B}+C$ – M_6
7	1	1	1	ABC – m_7	$\bar{A}+\bar{B}+\bar{C}$ – M_7

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

$= m_0 + m_1 + m_3 + m_6$

$= \Sigma m(0,1,3,6)$ [sum of product]

② $f(A,B,C) = (A+B+\bar{C})(A+\bar{B}+\bar{C})(\bar{A}+\bar{B}+C)$

$= M_1 \cdot M_3 \; M_6$

$= \Pi M(1,3,6)$ [Product of sum]

Complements of standard forms

$f(A,B,C) = m_0 + m_1 + m_3 + m_4 + m_6 + m_7$

$= M_2 + M_5$

$f(A,B,C) = \Sigma m(0,1,3,4,6,7) = \Pi M(2,5)$

$f(A,B,C,D) = \Sigma m(0,2,4,6,8,10,12,14)$

$= \Pi M(1,3,5,7,9,11,13,15)$

www.jntuworldupdates.org

Map Method:

for simplification of boolean Expressions by boolean algebra, so we need better understanding of boolean laws, Rules and theorems. During the process of simplification we have to predict Each Successive step.

for these Reasons we can more simplified Expressions by boolean algebra alone.

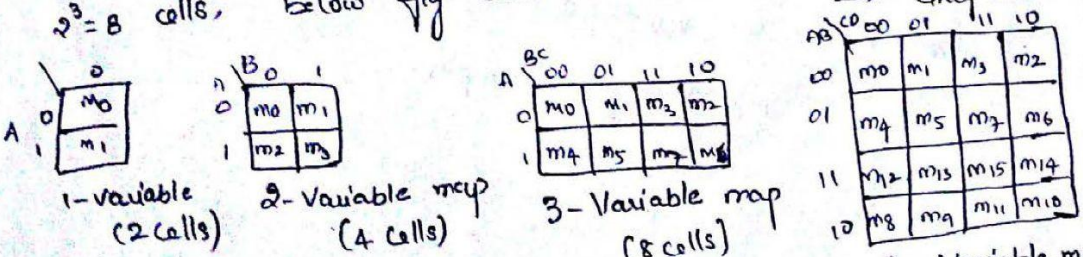
Map method gives us systematic approach for simplifying a boolean Expression.

The map method, first proposed by Veitch and modified by Karnaugh. hence it is known as the Veitch diagram or the Karnaugh map.

One Variable, two - Variable, Three - Variable, and four - Variable maps.

The basis of this method is a graphical chart known as Karnaugh map (K-map). it Contains boxes called cells. Each of the cell represents one of the 2^n possible products that can be formed from n Variables.

A 2-Variable map Contains $2^2=4$ Cells, a 3-Variable map Contains $2^3=8$ cells, below fig shows the 1, 2, 3, and 4 Variable maps.



1-Variable (2 Cells) 2-Variable map (4 Cells) 3-Variable map (8 cells) 4-Variable map (16 cells)

plotting a K-map.

logic function can be represented in various forms such as truth table, SOP boolean Expression and POS boolean Expression.

Representation of truth table on Karnaugh map.

K-map plotted from truth tables with 2, 3 and 4 variables. the terms which are having output 1, have the corresponding cells marked with 1's. The other cells are marked with zeros.

Ex:

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

A \ B	0	1
0	0	1
1	1	0

OR

A \ B	$\bar{B}$	B
$\bar{A}$	0	1
A	1	0

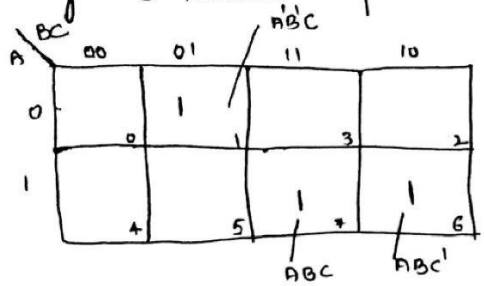
Representation of 2-variable truth table on K-map.

Sum terms Representation on Karnaugh map.

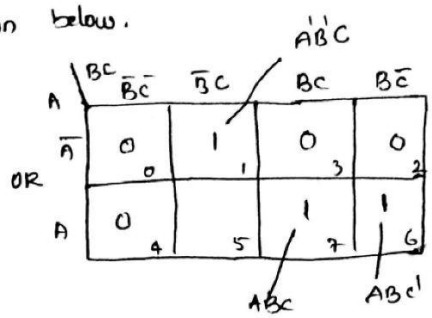
A Boolean Expression with Sum terms (Sum of product form) can be plotted with the K-map by placing a 1 in each cell corresponding to a term (minterm) in the SOP Expression. Remaing cells are filled with zeros. This is illustrated in the following Examples.

Example: plot Boolean Expression $Y = AB\bar{C} + ABC + \bar{A}\bar{B}C$ on the K-map.

Solution: The Expression has 3 Variables and hence it can be plotted using 3-Variable map as shown below.



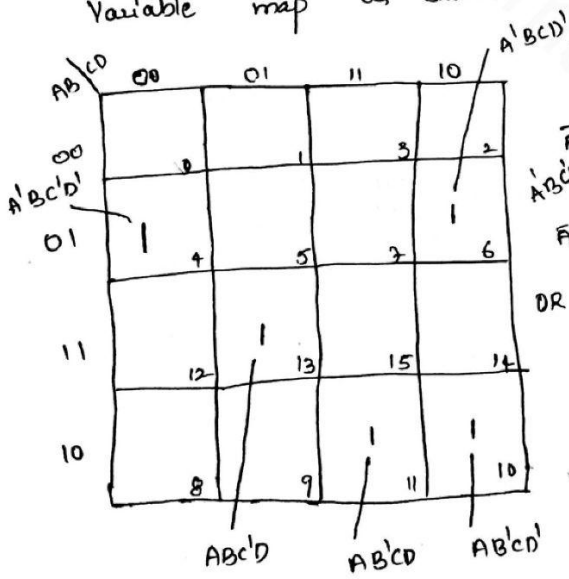
OR



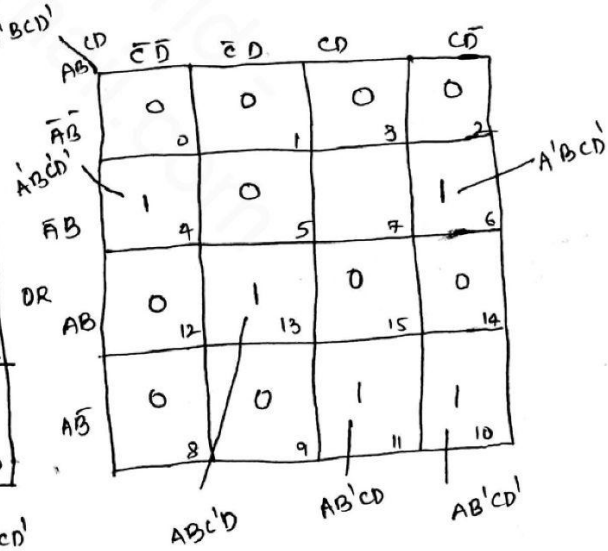
Example: plot Boolean Expression.

$$Y = A'B'C'D' + AB'CD' + A'BCD' + AB'CD + ABCD'$$ on the Karnaugh map.

The Expression has 4-Variables and hence it can be plotted using Variable map as shown below.



OR

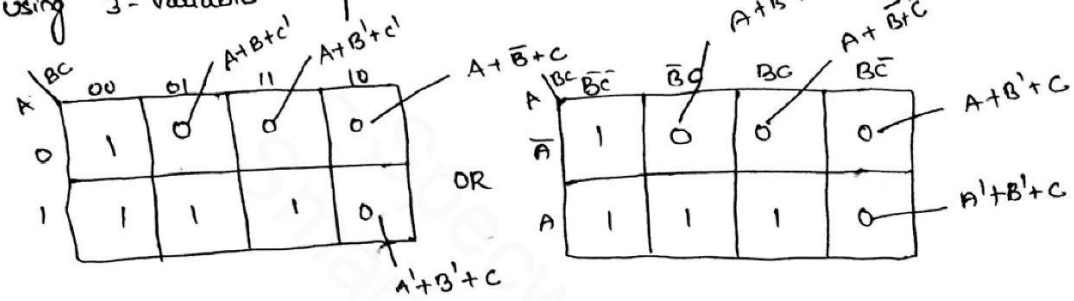


product terms Representation on Karnaugh maps.

A Boolean Expression with product terms (the product of sum) can be plotted on the Karnaugh map by placing a 0 in Each corresponding to a term (maxterm) in Expression. Remaining cells are filled with ones. This is illustrated in the following Examples.

Example: plot Boolean Expression $Y = (A+B+\bar{C})(A+\bar{B}+\bar{C})(\bar{A}+\bar{B}+C)(A+B+\bar{C})$ on the Karnaugh map.

Solution: The Expression has 3 Variables and hence it can be plotted using 3-Variable map as shown below.



$m_2 = A+\bar{B}+C$ $\quad m_3 = A+\bar{B}+\bar{C}$, $\quad m_6 = \bar{A}+\bar{B}+C$ $\quad m_1 = A+B+\bar{C}$

Using K-maps to obtain minimal Expressions for complete Boolean function

Once the boolean function is plotted on the Karnaugh map we have to use grouping technique to simplify the boolean function.

Grouping nothing but a combining terms in adjacent cells. Two cells are said to be adjacent if they conform single change Rule.

When adjacent 1's are grouped then we get result in the SOP form otherwise we get result in the POS form.

Let us See Various grouping rules.

Grouping two Adjacent ones (pair).

Ex: $Y = \bar{A}\bar{B}C + \bar{A}BC$

A \ BC	00	01	11	10
0		1	1	
1				

The K-map contains a pair of 1's that are horizontally adjacent to Each other, the first Represents $\bar{A}\bar{B}C$ and Second $\bar{A}BC$.

In this two terms only the B Variable appears in both normal and complemented form. So, by Combining these two we can Eliminate the B term and the result is $\bar{A}C$.

Ex: $A'BC + ABC = BC$

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

- In K-map we can Combine two Vertically adjacent 1's
- In a K-map leftmost column and right most column are considered to be adjacent.

Thus, two 1's in these columns with a Common row Can be Combined to Eliminate one Variable.

- Ex: $Y = AB'C' + AB'C'$

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

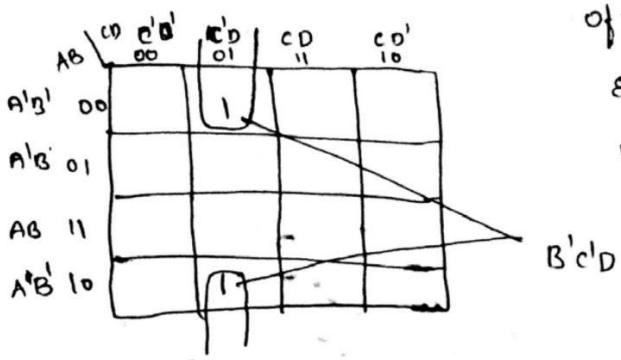
$= AC'$

In a K-map the leftmost column and rightmost column are Considered to be adjacent. Thus, the two 1's in these columns with a Common row can be Combined to Eliminate one variable

Ex: $Y = A'B'C'D + AB'C'D$.

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

$= B'C'D$.



Let us see this Example. 1's from top row and bottom row of Some column are combined. Eliminate Variable A, Since in a map the top row and bottom row are considered to be adjacent.

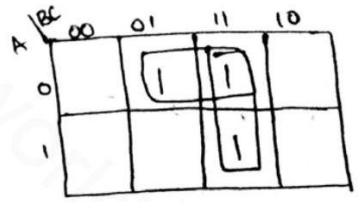
Ex: $A'B'C + A'BC + ABC$

$= A'B'C + A'BC + A'BC + ABC$ Rule $[A+A=A]$

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

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

$= A'C + BC$.



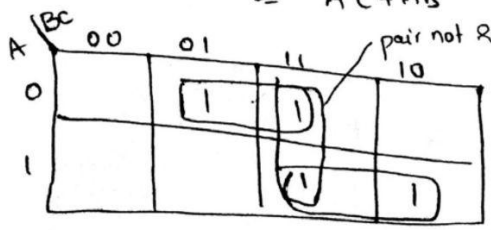
This shows a K-map that has two overlapping pair of that 1's. This shows that we can share one term between two pairs

→ Another Example

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

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

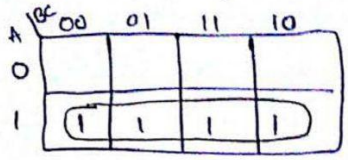
$= A'C + AB$ $[A'+A=1]$.



This fig shows a K-map where three group of pairs can be formed. But only two pairs are Enough to include all 1s present in the K-map.

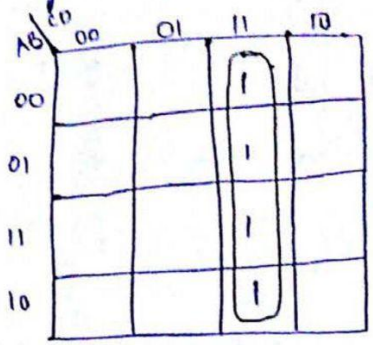
Grouping of 4 adjacent ones (Quad)

- In a K-map we can group four adjacent 1's. The resultant group is called Quad.
- four 1's may be horizontally adjacent or vertically adjacent.



Horizontally adjacent

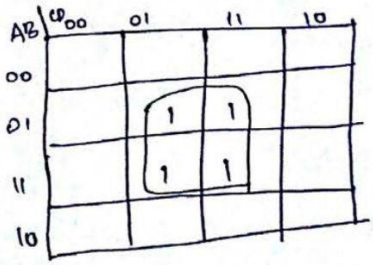
$Y = A$



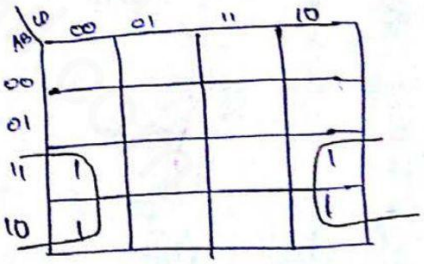
Vertically adjacent.

$Y = CD$

If a K-map contains four 1's in a square, and they are considered as adjacent to each other. [fig①]



fig① $Y = BD$



fig②

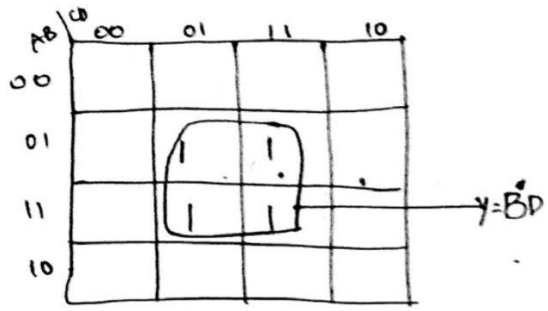
$Y = AD$.

- The four 1's in fig ② are also adjacent, because the top and bottom rows are considered to adjacent to each other and the leftmost and rightmost columns are also adjacent to each other.

www.jntuworldupdates.org

Ex: $Y = \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}D + ABCD$.

Solution: As there are 4-Variable total no. of cells $= 2^4 = 16$.



3. Grouping Eight adjacent ones (octet):

- Grouping of eight adjacent 1's is called as octet.
- either you can group horizontally, vertically.
- When an octet is combined in a four variable map, three of four variables are eliminated because only one variable remains unchanged.

Simplification of SOP Expression:

- The combination of pairs, quads, and octets on a karnaugh map can be used to obtain a simplified expression.
- A pair of 1's eliminates one variable.
- A quad of 1's eliminates two variables
- An octet of 1's eliminates three variables.
- In general, when a variable appears in both complemented and uncomplemented form within a group, that variable eliminated from a resultant expression.

Generalised procedure to Simplify boolean Expression

1. plot the K-map and place 1's in those cells corresponding to the 1s in the truth table or Sum of product Expression. place as in other cells.
2. check the K-map for adjacent 1s and then circle those 1's which are not adjacent to any other 1s. These are called isolated 1s.
3. check for those 1s which are adjacent to only one other 1 and Encircle such pairs.
4. Check for quards and octets of adjacent 1s even if it contains some 1s that have already been Encircled. while doing this make sure that there are minimum number of groups.
5. Combine any pairs necessary to include any 1's that have not yet been grouped.
6. from the Simplified Expression by Summing product terms of all the groups.

Example: minimize the Expression $Y = A\bar{B}C + \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$

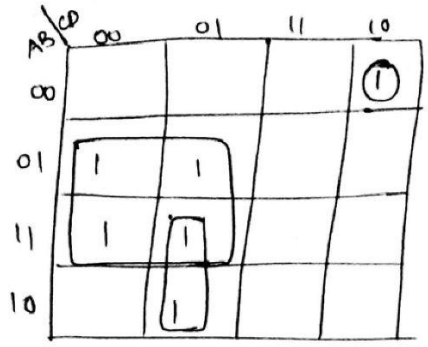
A \ BC	00	01	11	10
0	1	1	1	
1	1	1		

1. first see for isolated ones. In this there are no. isolated ones.
2. See for adjacent pair of 1's — 1
3. See for Quad — 1

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

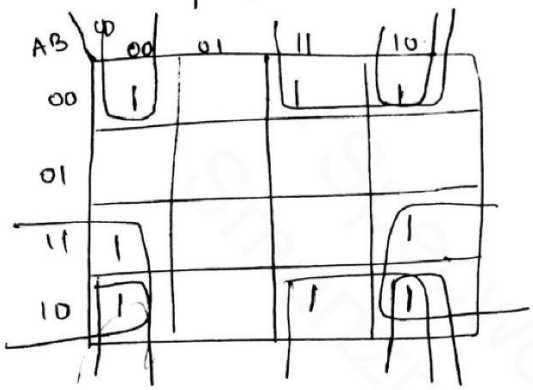
2) minimize the Exp $y = \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + AB\bar{C}\bar{D} + AB\bar{C}D + A\bar{B}\bar{C}D + A\bar{B}CD$ represent using A, B and

www.smartzworldupdates.com



$y = BC + AB'C + A'B'CD'$

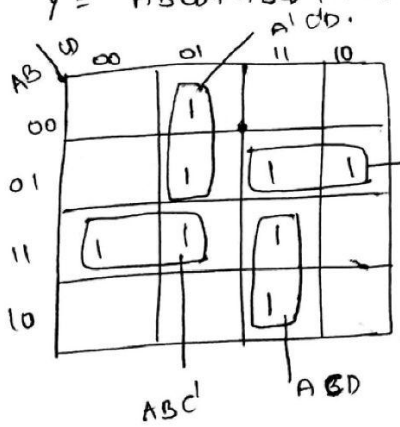
3) minimize the Exp $y = A'B'CD' + ABCD' + AB'CD' + AB'C'D' + ABC'D' + A'B'CD + A'B'C'D'$



$y = B'D' + B'C + AD'$

4) Reduce the following fn to its minimum SOP form.

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



— There are no isolated 1's

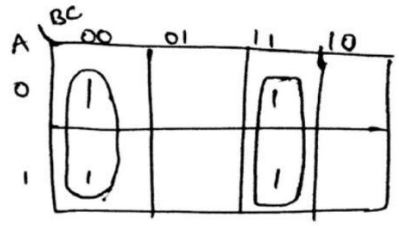
— There is no octet, but there is a quad. However, all 1's in the quad have already been grouped. Therefore this quad is ignored.

$y = A'C'D + A'BC + ABC' + ACD.$

www.jntuworldupdates.org

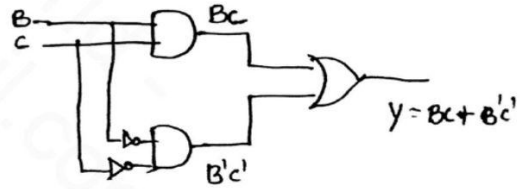
mplement the logic function specified by the truth table (Below) using the Karnaugh map method. y is the output variable and A, B and C are the input variables.

A	B	C	y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



$B'C' + BC$.

The boolean Expression can be implemented as follows



Simplification of POS Expression:-

Once the Expression is plotted on the K-map instead of making the group of ones, we have to make groups of zero.

Steps

1. plot the K-map and place 0's in those cells Corresponding to the 0's in the truth table or maxterms in the products of sum Expression.
2. check the K-map for adjacent 0's and Encircle those 0's which are not adjacent to any other 0's. These are called isolated 0's
3. check for those 0's which are adjacent to only one other 0 and Encircle Such pairs.

www.jntuworldupdates.org

4. Check for quads and octets of adjacent 0's even if it contains that have already been encircled. While doing this make sure there are minimum number of groups.

5. Combine any pairs necessary to include any 0's that have not been grouped.

6. From the simplified SOP expression for $\bar{F}$ by summing product terms of all the groups.

Note: The simplified SOP necessary to include any 0s expression is in the complemented form because we have grouped 0s to simplify the expression.

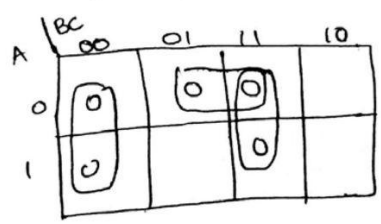
7. Use Demorgan's theorem on $\bar{F}$ to produce the simplified expression in POS form.

Ex: minimize the expression

$Y = (A+B+\bar{C})\ (A+\bar{B}+\bar{C})\ (\bar{A}+\bar{B}+\bar{C})\ (\bar{A}+B+C)\ (A+B+C)$ · (POS) (maxterms)

Solution: $(A+B+\bar{C}) = m_1$, $(A+\bar{B}+\bar{C}) = m_3$, $(\bar{A}+\bar{B}+\bar{C}) = m_7$

$(\bar{A}+B+C) = m_4$, $(A+B+C) = m_0$.



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

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

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

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

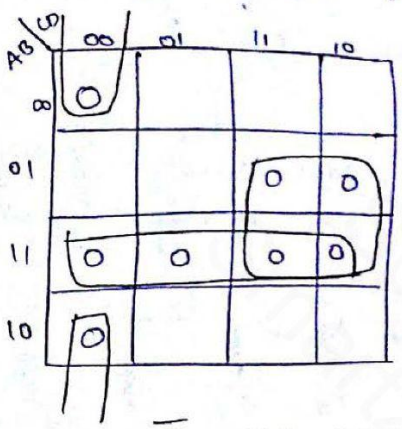
Sol: Minimize the following Exp in POS form.

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

Solution: $(\bar{A}+\bar{B}+C+D) = M_{12}$ $\quad (\bar{A}+\bar{B}+\bar{C}+D) = M_{14}$ $\quad (\bar{A}+\bar{B}+\bar{C}+\bar{D}) = M_{15}$

$(\bar{A}+B+C+D) = M_8$, $(A+\bar{B}+\bar{C}+D) = M_6$, $(A+\bar{B}+\bar{C}+\bar{D}) = M_7$

$(A+B+C+D) = M_0$ and $(\bar{A}+\bar{B}+C+\bar{D}) = M_{13}$



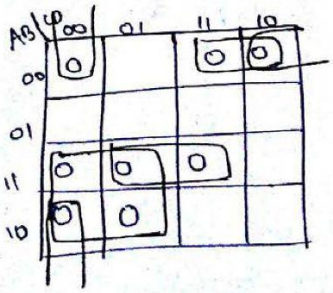
There is no isolated 0's.

$$\bar{Y} = BC + AB + B'C'D'$$

$$Y = \bar{\bar{Y}} = \overline{BC + AB + B'C'D'}$$
$$= (\bar{B}+\bar{C})(\bar{A}+\bar{B})(\bar{B'}\bar{C'}+\bar{D'})$$
$$= (\bar{B}+\bar{C})(\bar{A}+\bar{B})(B+C+D)$$

Ex: Reduce the following function using K-map technique.

$f(A,B,C,D) = \Sigma m(0,2,3,8,9,12,13,15)$



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

$$\bar{F} = \overline{AC' + ABD + A'B'C + A'B'D'}$$
$$= (A'+C)(A'+B'+D')(A+B+C')(A+B+D)$$

www.jntuworldupdates.org

Don't Care Conditions

In some logic ckts, certain input conditions never ∴ the corresponding o/p never appears. In such cases the o/p is not designed, defined, it can be either High or low. These output levels are indicated by 'x' or 'd' in the truth table are called don't care o/p's.

- A ckt designer is free to make the o/p for any "don't care" condition either a '0' or '1' in order to produce the simplest o/p Expression.
- It is important to decide which don't cares to change to and which to 1 to produce the best K-map grouping.

Ex: find the reduced SOP form of the following function

$f(A,B,C,D) = \Sigma m(1,3,7,11,15) + \Sigma d(0,2,4)$

AB \ CD	00	01	11	10
00	X	1	1	X
01	X	0	1	0
11	0	0	1	0
10	0	0	1	0

To form a quad of cells 0, 1, 2, and 4 the don't care conditions 0 and 2 are replaced by 0 since it is not required to form any group.

AB \ CD	00	01	11	10
00	1	1	1	1
01	0	0	1	0
11	0	0	1	0
10	0	0	1	0

$= AB' + CD.$

Ex: Reduce the following fn using K-map.

$f(A,B,C,D) = \Sigma m(5,6,7,12,13) + \Sigma d(4,9,14,15)$

AB \ CD	00	01	11	10
00	0	0	0	0
01	X	1	X	X
11	1	1	X	X
10	0	X	0	0

To form a octet of cells 4, 5, 6, 7, 12, 13, 14 and 15 the don't care conditions 4, 14, and 15 are replaced by 1s. The remaining don't care condition 9 is replaced by 0 to get simplified fn.

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

$f(A,B,C,D) = B$

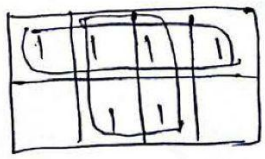
www.jntuworldupdates.org

Ex:- Implement the following using logic gates.

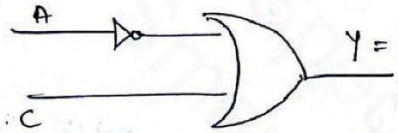
$f(A,B,C) = \sum(0,1,3,7) + \sum d(2,5)$

A \ BC	00	01	11	10
0	1	1	1	X
1	0	X	1	0

To form two quads both don't care conditions are replaced by 1s and we get.



$= \bar{A} + C$



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

Five Variable K-map:-

- A 5-Variable K-map require $2^5 = 32$ cells, but adjacent are difficult to identify on a single 32-cell map.

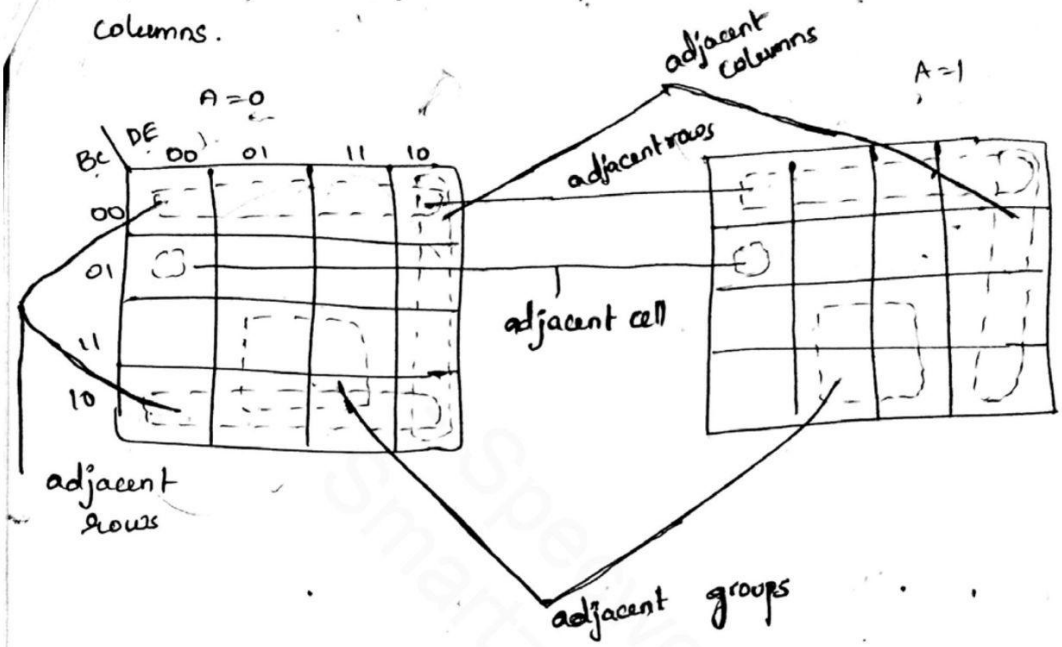
  ∴ Two 16-bit K-maps are used. If the variables A, B, C, D and E two identical 16-cell maps containing B, C, D, E can be constructed. One map is used for A and other for $\bar{A}$.

- In order to identify the adjacent grouping in the five variable we must imagine the two maps superimposed on one another.
- Every cell in one map is adjacent to the corresponding cell in the other map, because only one variable changes between such corresponding cells.
- Every row on one map is adjacent to the corresponding row
- The rightmost and leftmost columns within each 16-cell map are

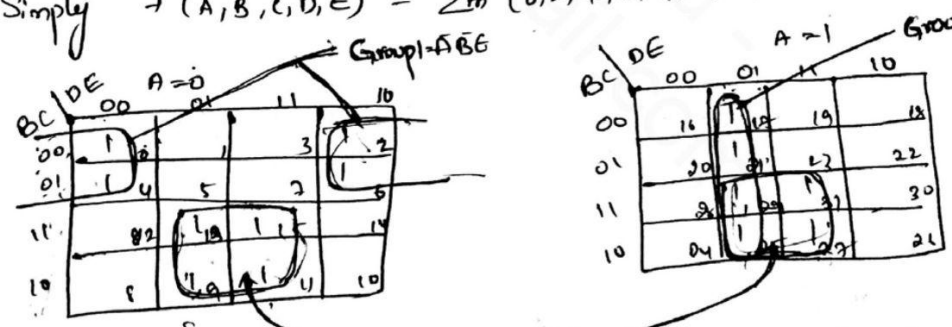
www.jntuworldupdates.org

jacent, just as they are in any 16-cell map, as are the top and bottom rows.

However, the rightmost column of one map is not adjacent to the leftmost column of other map. Since those are not corresponding columns.



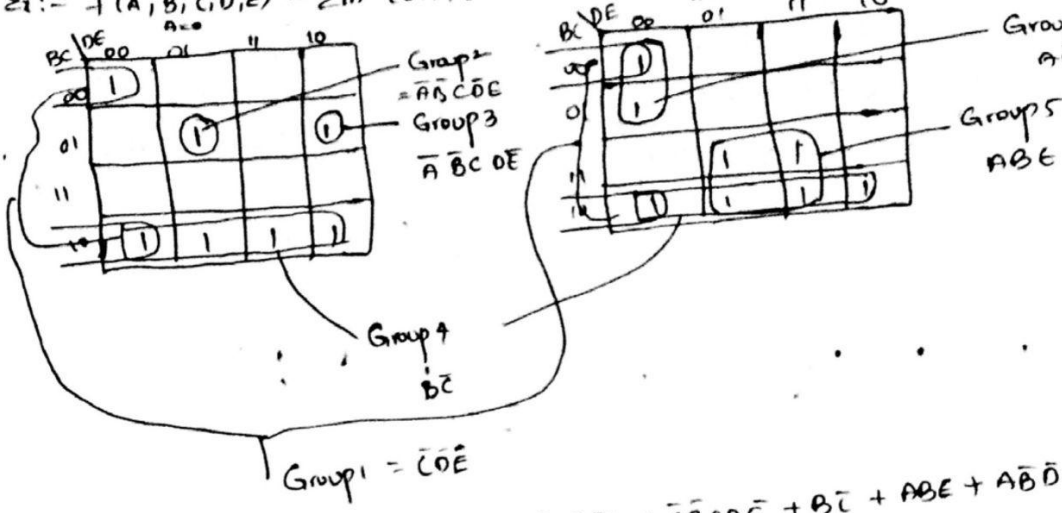
Simply $f(A,B,C,D,E) = \Sigma m(0,2,4,6,9,11,13,15,17,21,25,27,29,31)$



$f(A,B,C,D,E) = \bar{A}\bar{B}\bar{E} + BE + A\bar{D}\bar{E}$

www.jntuworldupdates.org

Ex:- $f(A,B,C,D,E) = \sum m\,(0,5,6,8,9,10,11,16,20,24,25,26,\ldots)$



$$f(A,B,C,D,E) = \bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}\bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}CD\bar{E} + B\bar{C} + ABE + A\bar{B}\bar{D}\bar{E}$$

Ex: $f(A,B,C,D,E) = \sum m\,(1,4,8,10,11,20,22,24,25,26) + d\,(0,12,14,17)$

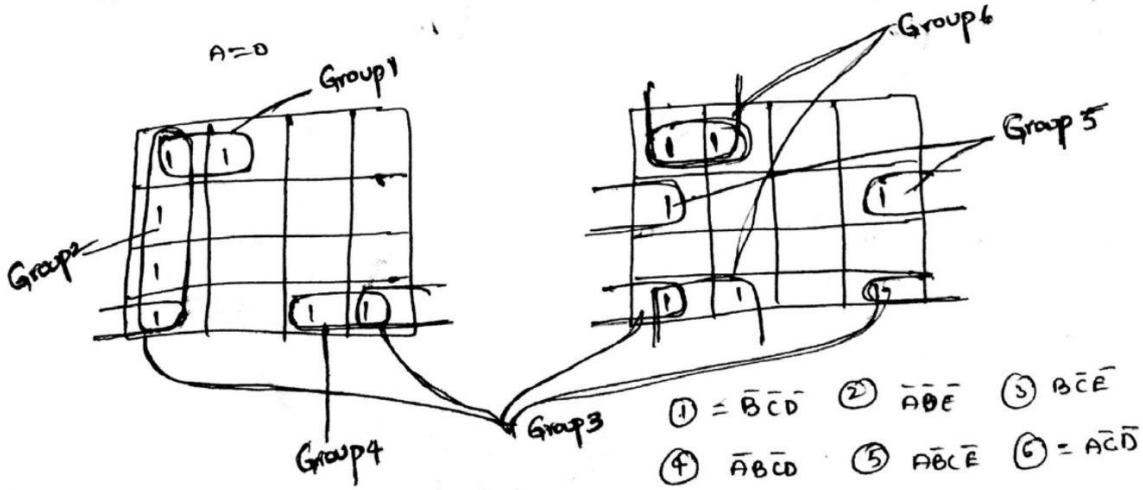
A=0

BC \ DE	00	01	11	10
00	X	1		
01	1			
11	X			
10	1		1	1

A=1

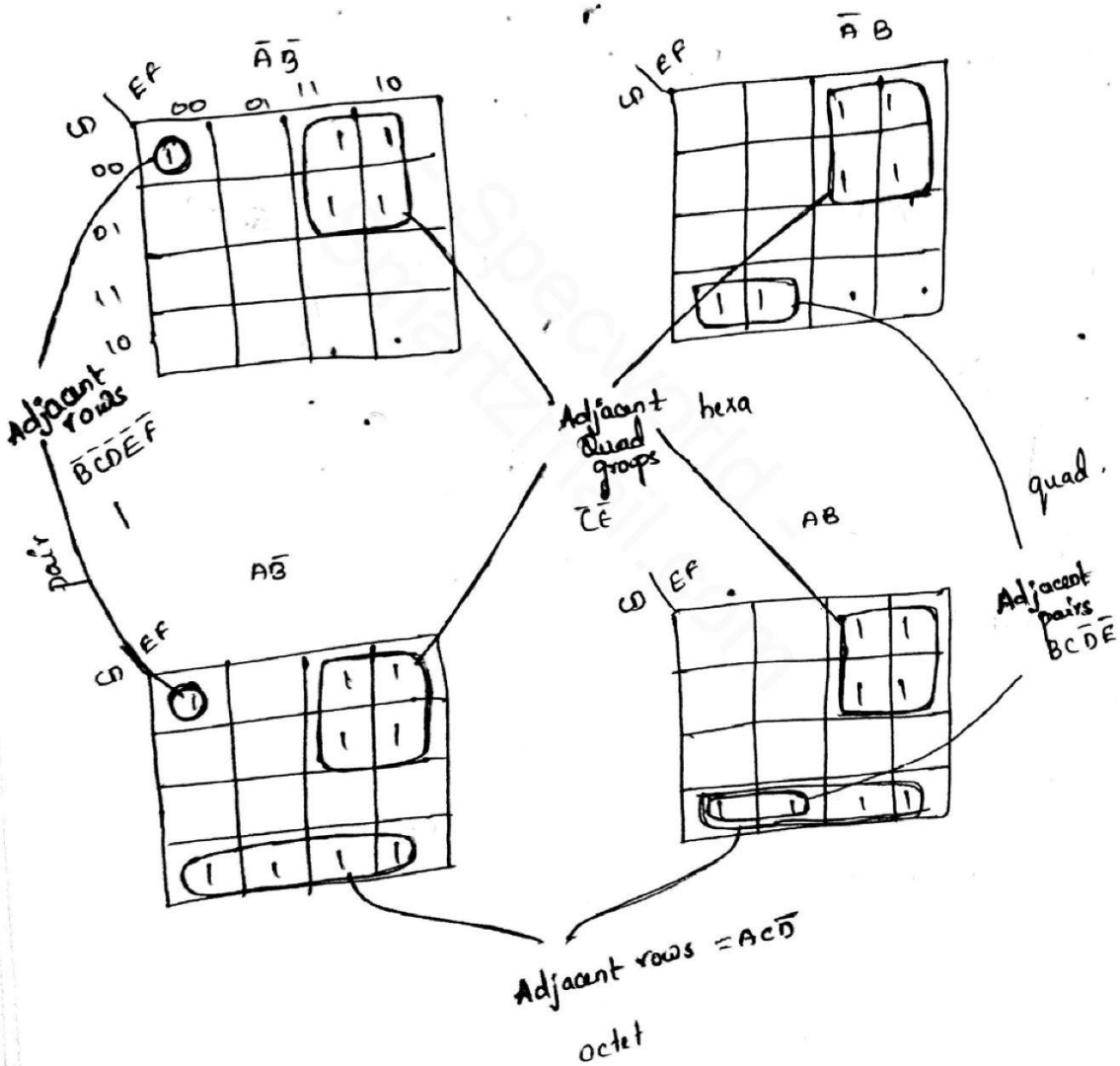
BC \ DE	00	01	11	10
00	X	X		
01	1			1
11				
10	1	1		1

In order to minimize the given Exp all don't Care. has to be taken as 1's.



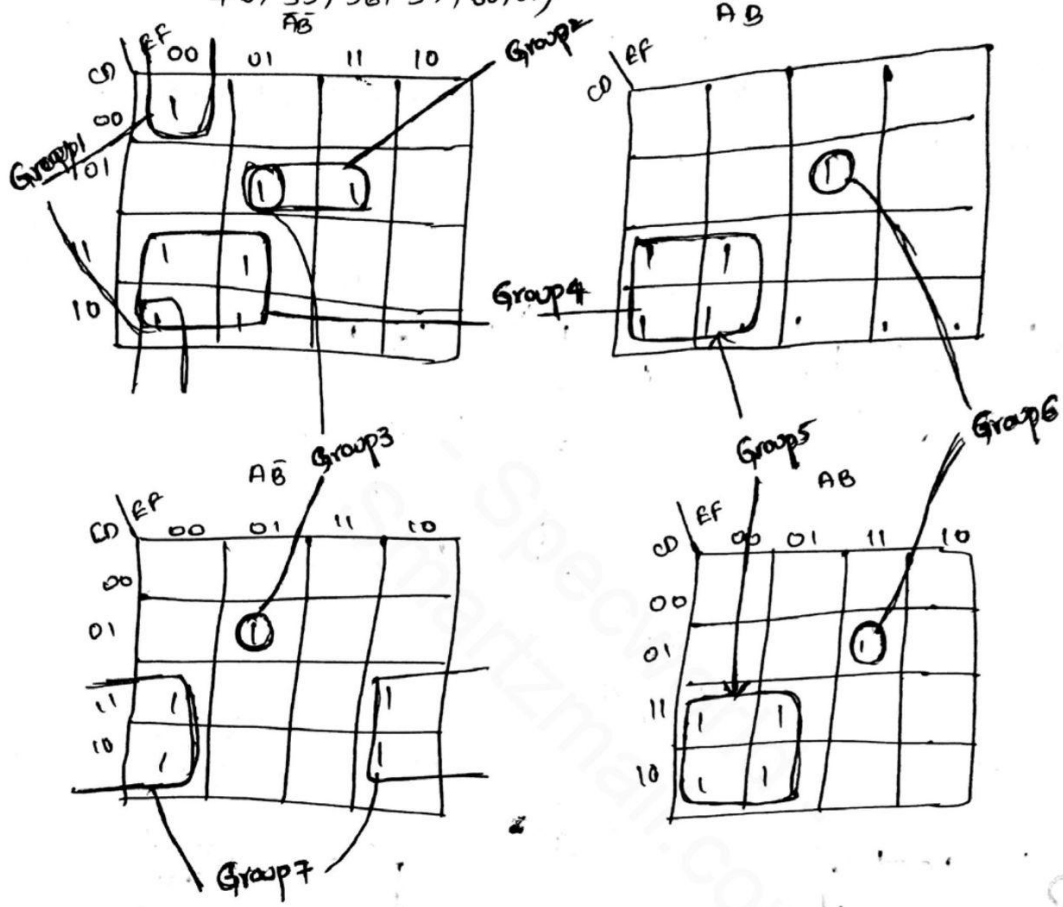
Variable K-map :-

A 6-Variable K-map requires $2^6 = 64$ cells. These cells are divided into four identical 16-cell maps. If the variables are A, B, C, D, E and F 16-cell maps contain C, D, E and F and each 16-cell map represents four combinations of A and B.



Ex: Simplify the Boolean fn

$f(A,B,C,D,E,F) = \sum m(0,5,7,8,9,12,13,23,24,25,28,29,37,40,46,55,56,57,60,61)$



① $= \bar{A}\bar{B}\bar{D}\bar{E}\bar{F}$ ② $\bar{A}\bar{B}\bar{C}DF$ ③ $\bar{B}\bar{C}D\bar{E}F$

④ $\bar{A}C\bar{E}$ ⑤ $BC\bar{E}$ ⑥ $B\bar{C}DEF$ ⑦ $A\bar{B}C\bar{F}$

$f(A,B,C,D,E,F) = \bar{A}\bar{B}\bar{D}\bar{E}\bar{F} + \bar{A}\bar{B}\bar{C}DF + BCDEF$

Quine - Mc Cluskey Method :-

The minimization of Boolean Expressions using K-maps is usually to a maximum of Six variables.

The Quine - mc cluskey method, also known as the tabular method. It is a more Systematic method of minimizing expressions of Even larger number of variables.

principle :- The minterms whose binary Equivalents differ only in one place can be combined to reduce the minterms.

Example: obtain the set of prime implicants for the boolean Expression $f = \sum m(0, 1, 6, 7, 8, 9, 13, 14, 15)$ using the tabular method.

Step1: list all minterms in binary form

minterm	Binary representation
m_0	0000
m_1	0001
m_6	0110
m_7	0111
m_8	1000
m_9	1001
m_{13}	1101
m_{14}	1110
m_{15}	1111

www.jntuworldupdates.org

Step2: Arrange minterms according to no. of 1's

minterm	Binary Rep.
m0	0000
m1	0001
m8	1000
m6	0110
m9	1001
m7	0111
m13	1101
m14	1110
m15	1111

Step3: Compare Each term in Each group with next higher group terms. If they differ in single variable, mark that position with '–'

minterm	Binary Rep.
0, 1	000 –
0, 8	– 000
1, 9	– 001
8, 9	1 00 –
6, 7	011 –
6, 14	– 110
9, 13	1 – 01
7, 15	– 111
13, 15	11 – 1
	111 –

Step4: Repeat step 3

minterm	Binary Rep.
0, 1, 8, 9	00 –
0, 8, 1, 9	– 00 –
6, 7, 14, 15	– 11 –
6, 14, 7, 15	– 11 –

Step 5: List P.I

P.I	Binary Rep.
9, 13 (1-01)	1-01
13, 15 (11-1)	11-1
0, 1, 8, 9 (-00-)	-00-
6, 7, 14, 15 (-11-)	-11-

Step 6: EPI Chart (essential prime implicant)

EPI / mintarms	m_0	m_1	m_6	m_7	m_8	m_9	m_{13}	m_{14}	m_{15}
✓ 9, 13 either or						⊙	⊙		
13, 15							⊙		⊙
✓ 0, 1, 8, 9	⊙	⊙			⊙	⊙			
✓ 6, 7, 14, 15			⊙	⊙				⊙	⊙

Check for single dot in a column:- 0, 1, 8, 14, 6, which are not cover all mintarms, so check for double dot

double dot : 9, 13 (1-01)

0, 1, 8, 6, 7, 14.

9, 13, 15 9, 13

9, 13

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

9, 13 1-01

0, 1, 8, 9 -00-

6, 7, 14, 15 -11-

$$AC'D + B'C' + BC$$

② $f(A, B, C, D, E) = \Sigma m(0, 2, 4, 5, 6, 7, 8, 10, 14, 17, 18, 21, 29, 31) + \Sigma d(11, 20, 22)$

* When a Switching function has don't cares, we set them Equal and find P.I.

* In determining E.P.I and minimal cover of the function we set all don't cares Equal to 0. that is the don't care decimal number donot appear in P.I table.

Sol: Step1: List all minterms with binary Equivalents.

Step2: Arrange the minterms according to no. of 1's.

minterm	Binary Rep.	
0	00000	✓
2	00010	✓
4	00100	✓
8	01000	✓
5	00101	✓
6	00110	✓
10	01010	✓
17	10001	
18	10010	✓
20	10100	✓
7	00111	✓
11	01011	✓
14	01110	✓
21	10101	✓
22	10110	✓
29	11101	✓

www.intuworldupdates.org

Compare each term in each group with next higher order term.

minterm	Binary representation
0, 2	000 – 0 ✓
0, 4	00 – 00 ✓
0, 8	0 – 000 ✓
2, 6	00 – 10 ✓
2, 10	0 – 010 ✓
2, 18	– 0010 ✓
4, 5	0010 – ✓
4, 6	001 – 0 ✓
4, 20	– 0100 ✓
8, 10	010 – 0 ✓
5, 7	001 – 1 ✓
5, 21	– 0101 ✓
6, 7	0011 ✓
6, 14	0 – 110 ✓
6, 22	– 0110 ✓
10, 11	0101 – ✓
10, 14	01 – 10 ✓
17, 21	10 – 01
18, 22	10 – 10 ✓
20, 21	1010 – ✓
20, 22	101 – 0 ✓
21, 29	1 – 101
29, 31	111 – 1

www.jntuworldupdates.org

step4: Repeat step3

minterms	Binary rep.
0, 2, 4, 6	00 – – 0
0, 2, 8, 10	0 – 0 – 0
2, 6, 10, 14	0 – – 10
2, 6, 18, 22	– 0 – 10
4, 5, 6, 7	001 – –
4, 5, 20, 21	– 010 –
4, 6, 20, 22	– 01 – 0

Step5: List all P.I

P.I	Binary rep.
10, 11	0101 –
17, 21	10 – 01
21, 29	1 – 101
29, 31	1111 – 1
0, 2, 4, 6	00 – – 0
0, 2, 8, 10	0 – 0 – 0
2, 6, 10, 14	0 – – 10
2, 6, 18, 22	– 0 – 10
4, 5, 6, 7	001 – –
4, 5, 20, 21	– 010 –
4, 6, 20, 22	– 01 – 0

Step6: EPI chart

PI / minterms	m_0	m_2	m_4	m_5	m_6	m_7	m_8	m_{10}	m_{14}	m_{17}	m_{18}	m_{21}	m_{29}	m_{31}	d_{11}	d_{20}	d_{22}
10, 11								⊙							⊙		
✓ 17, 21										⊙		⊙					
21, 29												⊙	⊙				
✓ 29, 31													⊙	⊙			
0, 2, 4, 6	⊙	⊙	⊙		⊙												
✓ 0, 2, 8, 10	⊙	⊙					⊙	⊙									
✓ 2, 6, 10, 14		⊙			⊙			⊙	⊙								
✓ 2, 6, 18, 22		⊙			⊙						⊙						⊙
✓ 4, 5, 6, 7			⊙	⊙	⊙	⊙											
4, 5, 20, 21			⊙	⊙								⊙			⊙		
4, 6, 20, 22			⊙		⊙									✓	⊙		⊙
						✓	✓		✓	✓							

$y = A + B + C$

$Y = 17,21 + 29,31 + 0,2,8,10 + 2,6,10,14 +$
$2,6,18,22 + 4,5,6,7$

$= A\bar{B}\bar{D}E + ABCE + A\bar{C}\bar{E} + \bar{A}D\bar{E} + \bar{B}D\bar{E} +$
$\bar{A}\bar{B}C$

Problems on K-map

3-var:-

1. $F(x,y,z) = \Sigma m(2,3,4,5) = x \oplus y$
2. " $= \Sigma m(3,4,6,7) = yz + xz'$
3. " $= \Sigma m(0,2,4,5,6) = z' + xy'$
4. " $= \Sigma m(1,2,3,5,7) = C + A'B$
5. " $= \Sigma m(0,2,6,7) = xy + \bar{x}\bar{z}$
6. " $= \Sigma m(0,2,3,4,6)$
7. " $= \Sigma m(0,1,2,3,7) = \bar{a} + bc$
8. " $= \Sigma m(3,5,6,7) = xy + xz + yz$
9. $xy + x'y'z' + x'yz' = \Sigma m(0,2,6,7) = xy + \bar{x}\bar{z}$
10. $x'y' + yz + x'yz' = \Sigma m(0,1,2,3,7) = \bar{x} + yz$

4-var:-

1. $F(w,x,y,z) = \Sigma m(0,1,2,4,5,6,8,9,12,13,14)$
$= y' + w'z' + xz$
2. " $= \Sigma m(0,1,2,6,8,9,10) = \bar{B}\bar{D} + \bar{B}C + \bar{A}C\bar{D}$

www.jntuworldupdates.org

→ Wired Logic :—

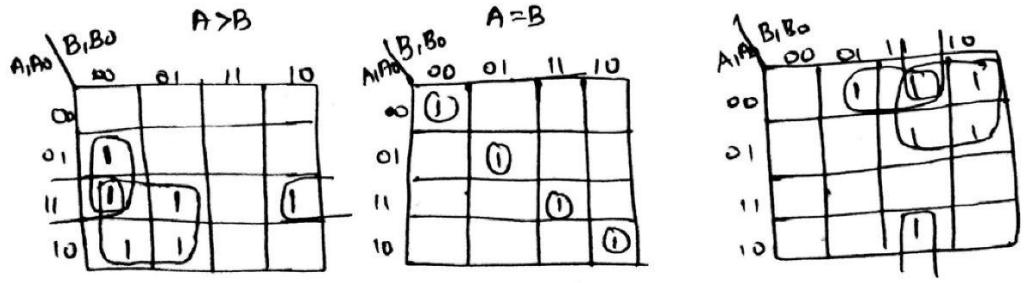
The types of gates most often found in ICs are NAND and NOR. for this reason, NAND and NOR logic implementations are most important from a practical point of view. Some NAND or NOR gates allow the possibility of wire connection between the o/p's of two gates to provide a specific logic function. This logic is called wired logic.

Design a 2-bit comparator using gates.

T.T

Inputs				outputs		
A_1	A_0	B_1	B_0	$A>B$	$A=B$	$A<B$
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

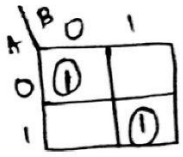
K-map simplification



Design a 1-bit comparator using basic gates.

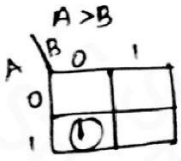
Inputs		outputs		
A	B	$Y_{A=B}$	$Y_{A>B}$	$Y_{A<B}$
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

A=B



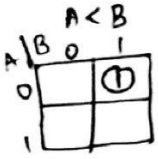
$A'B' + AB$

A>B

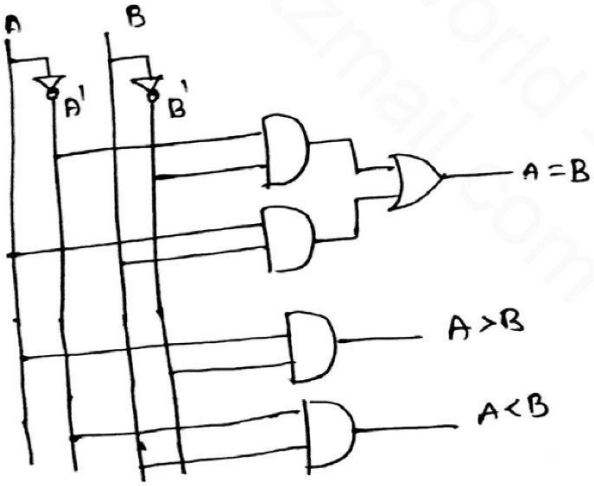


AB'

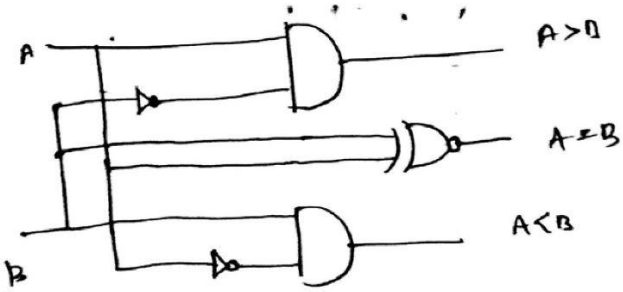
A<B



$A'B$



(or)



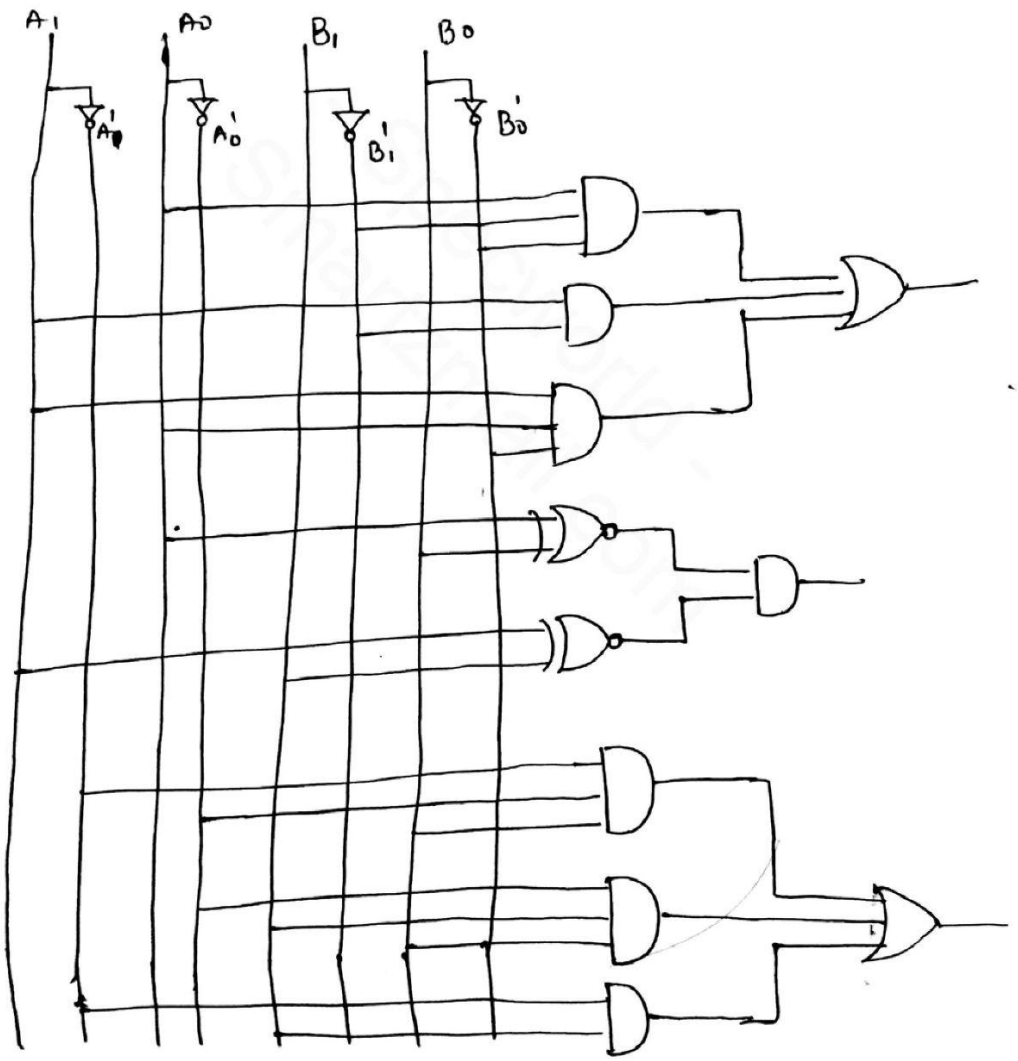
$A > B \Rightarrow A_0 B_1' B_0' + A_1 B_1' + A_1 A_0 B_0'$

$A = B$ $\quad A_1' A_0' B_1' B_0' + A_1' A_0 B_1' B_0 + A_1 A_0 B_1 B_0 + A_1 B_0' B_1 B_0'$

$A_1' B_1' (A_0' B_0' + A_0 B_0) + A_1 B_1 (A_0 B_0 + A_0' B_0')$

$(A_0 \odot B_0) \quad (A_1 \odot B_1)$

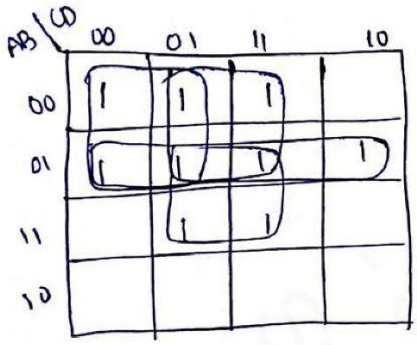
$A < B$ $\quad A_1' B_0' B_0 + A_0' B_1 B_0 + A_1' B_1$



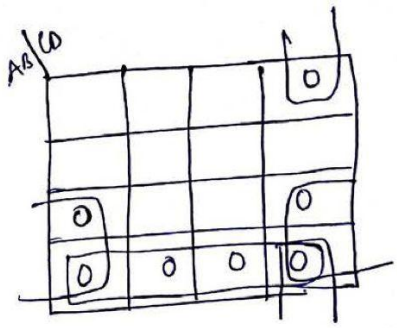
duce using mapping the Expression $f = \pi m(2, 8, 9, 10, 11, 12, 14)$

the given POS is $\pi m(2, 8, 9, 10, 11, 12, 14)$

the given Expression in the SOP is $\Sigma m(1, 3, 4, 5, 6, 7, 13, 15)$



$f_{min} = \bar{A}\bar{C} + \bar{A}D + \bar{A}B + BD$



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

Examples on K-map.

1) Reduce the following Expression using SOP & POS forms using mapping.

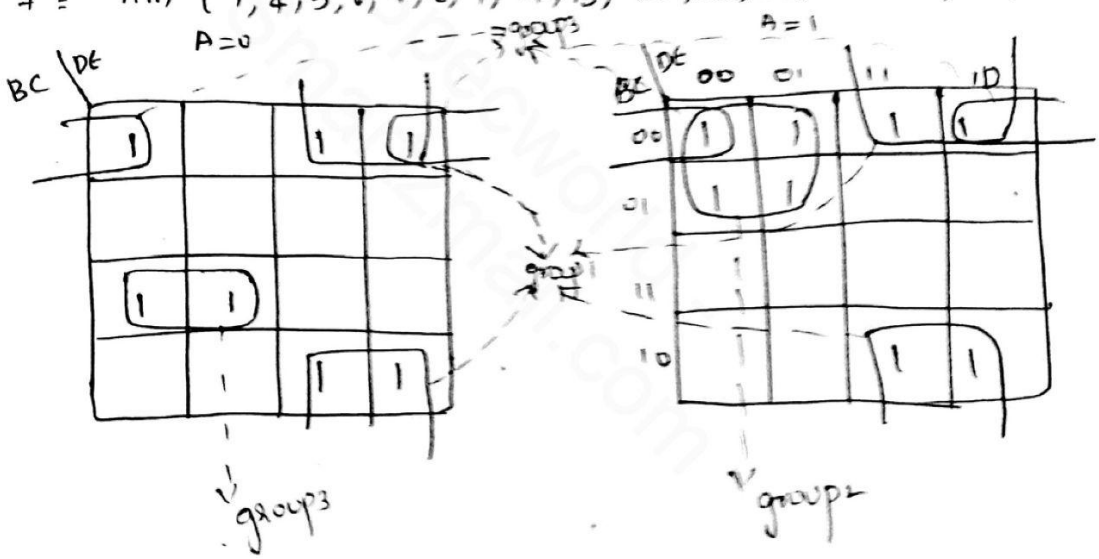
$$f = \sum m(0, 2, 3, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21, 26, 27)$$

Sol: The given Exp is in SOP form

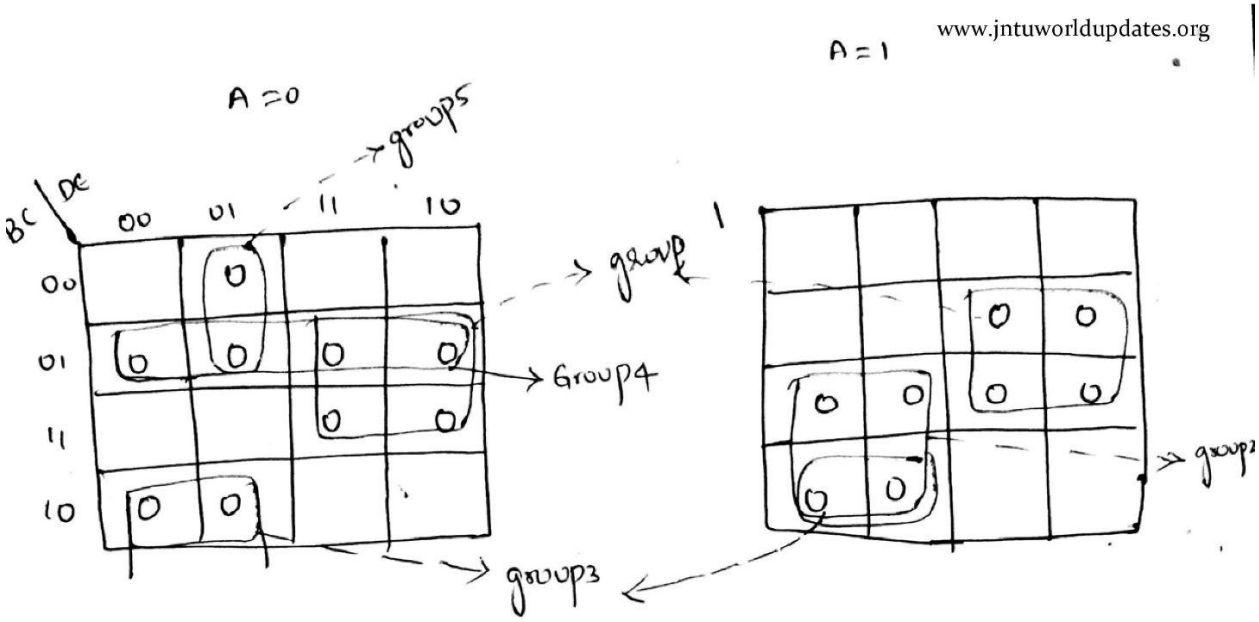
$$f = \sum m(0, 2, 3, 10, 11, 12, 13, 16, 17, 18, 19, 20, 21, 26, 27)$$

∴ This can be written in POS form as

$$f = \prod m(1, 4, 5, 6, 7, 8, 9, 14, 15, 22, 23, 24, 25, 28, 29, 30, 31)$$



$$f_{min} = \bar{A}BC\bar{D} + \bar{B}\bar{C}\bar{E} + A\bar{B}\bar{D} + \bar{C}D$$



The minimized POS Expression is

$$f_{min} = (\bar{C}+\bar{D})(\bar{A}+\bar{B}+D)(\bar{B}+C+D)(A+B+\bar{C})(A+B+D+\bar{E})$$

x	y	z	$\bar{x}$	$y\bar{z}$	$\bar{x}+y\bar{z}$
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	1	1
1	1	1	0	0	0

$x \backslash yz$	00	01	11	10
0				
1	0	0	0	

$= (\bar{x}+y)(\bar{x}+\bar{z})$

(2) Reduce mapping the Expression $f = \sum m(0,1,2,3,5,7,8,9,10,12,13)$

given SOP is $\sum m(0,1,2,3,5,7,8,9,10,12,13)$

The given Expression in the POS $(4,6,11,14,15)$

SOP

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

AB \ CD	00	01	11	10
00				
01	0			0
11			0	0
10			0	

$f_{min} = \bar{B}\bar{D} + A\bar{C} + \bar{A}D$

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

www.jntuworldupdates.org

(4) Show the truth table for each of the following functions and find its simplest POS form

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

(b) $f(x, y, z) = \bar{x} + y\bar{z}$

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

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

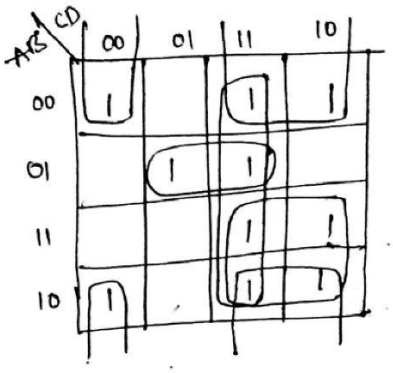
(b) $f(x, y, z) = \bar{x} + y\bar{z} = (\bar{x} + y)(\bar{x} + \bar{z})$

x	y	z	x (y+z)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

x \ yz	00	01	11	10
0	0	0	0	0
1	0			

$(y+z)(x)$

2] $f(A,B,C,D) = \Sigma m(0,2,3,5,7,8,10,11,14,15)$

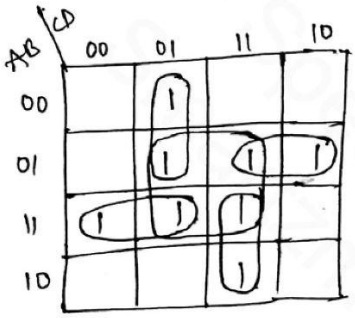


$= CD + \bar{B}\bar{D} + AC + \bar{A}BD + \bar{B}C$

↳ P.I.

EPI $= \bar{B}\bar{D} + \bar{A}BD + AC$

3] $f(A,B,C,D) = \Sigma m(1,5,6,7,11,12,13,15)$



$BD + \bar{A}\bar{C}D + AB\bar{C} + \bar{A}BC + ACD$

No. of PI = 5

$(1,5) \rightarrow \bar{A}\bar{C}D$

$(11,15) \rightarrow ACD$

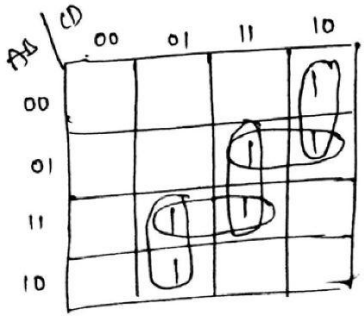
$(6,7) \rightarrow \bar{A}BC$

$(12,13) \rightarrow AB\bar{C}$

$(5,7,13,15) \rightarrow BD$

EPI $= \bar{A}\bar{C}D + ACD + \bar{A}BC + AB\bar{C}$ $\{=4\}$.

4]. $f(A,B,C,D) = \Sigma m(2,6,7,9,13,15)$



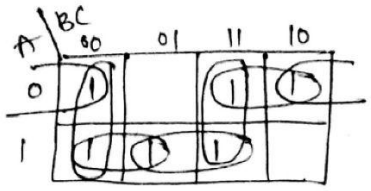
$= \bar{A}C\bar{D} + A\bar{C}D + BCD$

no. of PI = 5 ⇒ $\bar{A}C\bar{D}$ (2,6), BCD (7,15), $A\bar{C}D$ (9,13), ABD (13,15), $\bar{A}BC$ (6,7)

no. of EPI = 2 ⇒ (2,6), (9,13)

$\bar{A}C\bar{D} + A\bar{C}D$.

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



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

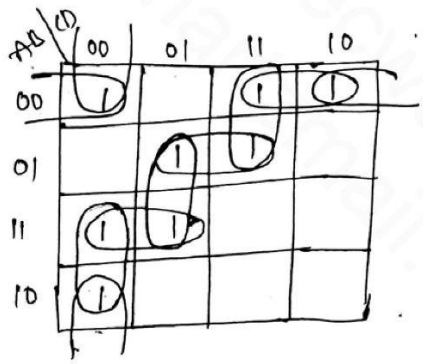
↳ P.I

no. of PI = 6 ⇒ (0,2), (2,3), (3,7), (5,7), (4,5), (0,4)

$f_3 = \bar{A}\bar{C} + A\bar{B} + BC$ (or) $\bar{B}\bar{C} + AC + \bar{A}B$

no. of EPI = 0

6]. $f(A,B,C,D) = \Sigma m(0,2,3,5,7,8,12,13)$

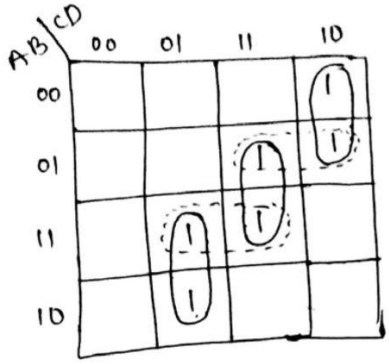


$f_4 = \bar{A}\bar{B}\bar{D} + \bar{A}CD + B\bar{C}D + A\bar{C}\bar{D}$

no. of PI = 8

No. of EPI = 0.

Ex: $f(A,B,C,D) = \Sigma m(2,6,7,9,13,15)$



$= \bar{A}C\bar{D} + BCD + A\bar{C}D$

No. of PI = 5 → $\bar{A}C\bar{D}$ (2,6), BCD (7,15), $A\bar{C}D$ (9,13), $\bar{A}BD$ (13,15), $\bar{A}BC$ (6,7)

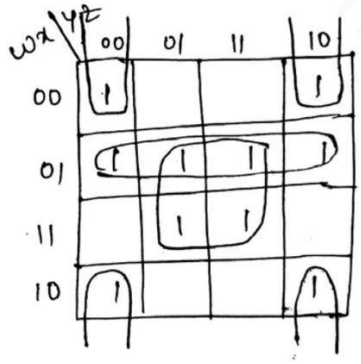
No. of EPI = 2 ⇒ (2,6), (9,13)

ABD
BD

0101 – 5
0111 – 7
1101 – 13
1111 – 15

Find the prime Implicant for the following and determine which are essential.

1) $f(w,x,y,z) = \Sigma m(0,2,4,5,6,7,8,10,13,15)$



$= \bar{w}x + xz + \bar{x}\bar{z} + \bar{w}\bar{z}$

↳ prime Implicants

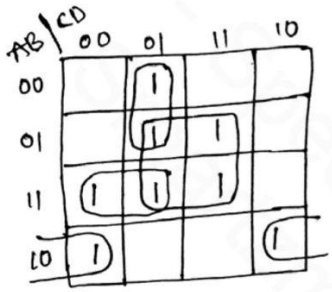
$\bar{x}\bar{z}$ and xz are not covered by any other cells, so those are called as the EPI.

Prime Implicant:

It is a smallest possible product term removing of any literal from which is not possible. The product must be an implicant of a given fn for it to be a prime implicant.

→ The bunch of is on the K-map which form a 2-square, 4-square, etc. is called PI or Subcube.

Ex: $f(A,B,C) = \Sigma m(1,5,7,8,10,12,13,15)$



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

$h_1 = A\bar{B}\bar{D}$

if A is removed $\bar{B}\bar{D} \rightarrow (0,1,8,10)$ is not an implicant

$\bar{B}$ $\quad A\bar{D} \rightarrow (8,10,12,14)$ " " " .

if $\bar{D}$ is removed $A\bar{B} \rightarrow (8,9,10,13)$ " " " .

$\therefore h_1$ is a prime implicant

Essential prime Implicant:

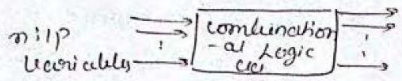
It is the prime Implicant, It must cover at least one minterm which is not covered by any other P.I.

$EPI = \{\bar{A}\bar{C}D, BD, A\bar{B}\bar{D}, AB\bar{C}\}$

combinational logic design

when logic gates are connected together to produce a specified o/p for certain specified combinations of i/p variables with no storage involved, the resulting circuit is called combinational logic. (here o/p variables are at all times dependent on i/p variables).

Block dia: of combinational ckt.

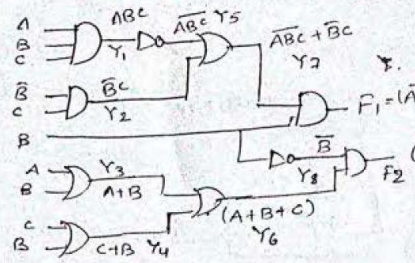


procedure to determine Boolean fun for o/p's of ckts from given ckt:

1. Make sure that given ckt is comb ckt & not sequen ckt.
   ⇓
   has logic gates with no feedback path (or) memory element.

2. label all gate o/p's that are a fun of i/p variables with arbitrary sym & determine Boolean fun for each gate o/p.
3. label the gates that are a fun of i/p variables & previously labelled gate & determine the B.fun for them.
4. Repeat step 3 until B.fun for o/p of the ckt obtained.
5. finally sub previously defined B.fun, obtain the o/p of B.fun in term of i/p variables.

Q. Obtain B.fun for o/p's of.

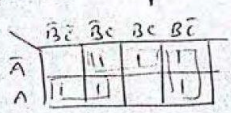


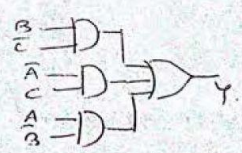
T.T:

A	B	C	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	F_1	F_2
0	0	0	0	0	0	0						
0	0	1	0	1	0	1						
0	1	0	0	0	1	1						
0	1	1	0	0	1	1						
1	0	0	0	0	1	0						
1	0	1	0	1	1	1						
1	1	0	0	0	1	1						
1	1	1	1	0	1	1						

simplified Boolean fun: for $f(A,B,C) = \Sigma m(1,2,3,4,5,6)$

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



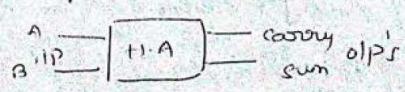


Adders:

A	B	S	C
0 + 0 :	0		
0 + 1 :	1		
1 + 0 :	1		
1 + 1 :	10_2		

T.T

Half adder:



Block schematic

K-map:

sum:

	$\bar{A}$	$\bar{B}$
$\bar{A}$	0	1
A	1	0

carry

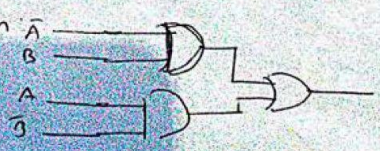
	0	0
	0	1

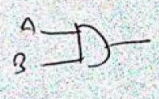
Sum: $\bar{A}B + A\bar{B}$

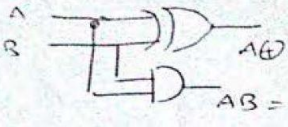
$\bar{A}B + A\bar{B}$

$C = AB$

logic diagram:



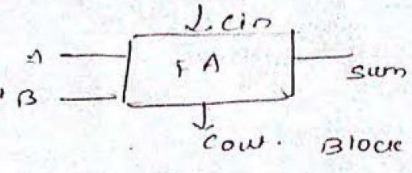




<u>Limitations</u>: 3 bit addition is not possible.

<u>Full adder</u>:

i/p's			o/p's	
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Block schematic

K-map:

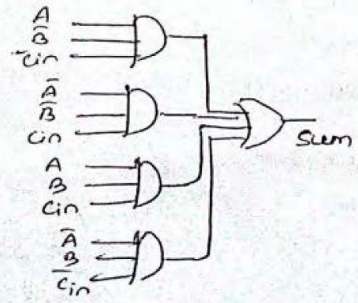
A \ BCin	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	1	0	1
A	1	0	1	0

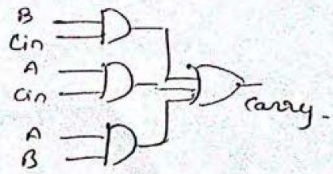
0	0	1	0
0	1	1	1

sum = $A\bar{B}\bar{C}in + \bar{A}\bar{B}Cin + ABCin + \bar{A}B\bar{C}in$

carry = $BCin + A\bar{B}Cin + AB$.

<u>logic dia</u>:





Simplification of sum:

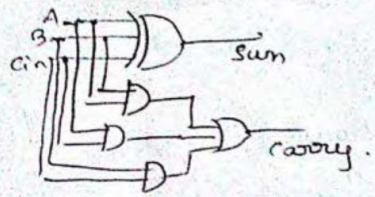
$$A\bar{B}\bar{C}in + \bar{A}\bar{B}Cin + ABCin + \bar{A}B\bar{C}in$$

$$= \bar{B}Cin[A+\bar{A}] + ABCin + \bar{A}B\bar{C}in \Rightarrow Cin[\bar{A}\bar{B} + \bar{A}B] + \bar{C}in[\bar{A}B + A\bar{B}]$$

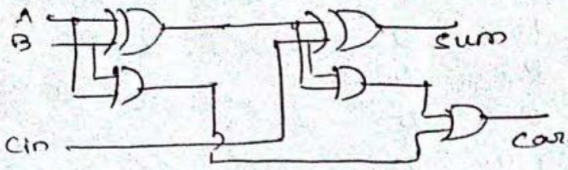
$$= Cin[\bar{B} + AB] + \bar{A}B\bar{C}in \Rightarrow Cin[A \oplus B] + \bar{C}in[A \oplus B]$$

$$= Cin[\bar{B} + A] + \bar{A}B\bar{C}in \Rightarrow Cin[\overline{A \oplus B}] + \bar{C}in[A \oplus B]$$

$$= ACin + \bar{B}Cin + \bar{A}B\bar{C}in \Rightarrow Cin \oplus (A \oplus B)$$



Implementation of F.A using two H.A's & OR gate:



$= BC_{in} + AC_{in} + AB$

$= AB + AC_{in} + BC_{in}(A+\bar{A})$

$= AB + AC_{in} + ABC_{in} + \bar{A}BC_{in}$

$= AC_{in} + \bar{A}BC_{in} + AB.$

$= AB + AC_{in}(B+\bar{B}) + \bar{A}BC_{in}$

$= \underline{AB + ABC_{in}} + A\bar{B}C_{in} + \bar{A}BC_{in}$

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

Subtractors:

$0-0 = 0$

$0-1 = 1 \quad B = 1$

$1-0 = 1$

$1-1 = 0$

Half Subtractor:

i/p's		o/p's	
A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Diff

	$\bar{B}$	B
$\bar{A}$	0	1
A	1	0

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

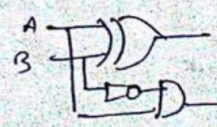
$A \oplus B$

Borrow

0	1
0	0

Borrow $= \bar{A}B$

logic ckt:



$A \oplus B$ = ~~sum~~ diff

$\bar{A}B$ = Borrow.

Full subtractor:

i/p's			o/p's	
A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

D \	$\bar{B}\bar{B}_{in}$	$\bar{B}B_{in}$	BB_{in}	$B\bar{B}_{in}$
$\bar{A}$	0	1	0	1
A	1	0	1	0

Bout \				
	0	1	1	1
	0	0	1	0

$$D = A\bar{B}\bar{B}_{in} + \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}B\bar{B}_{in}$$

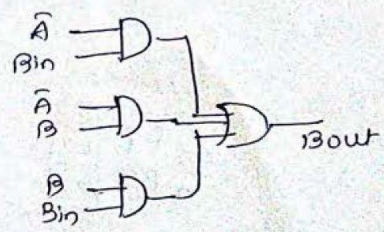
$$= \bar{B}_{in}\left[A\bar{B} + \bar{A}B\right] + B_{in}\left[\bar{A}\bar{B} + AB\right]$$

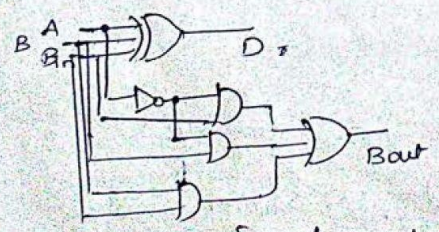
$$= \bar{B}_{in}\left[A \oplus B\right] + B_{in}\left[A \odot B\right]$$

$$= \bar{B}_{in}\left[A \oplus B\right] + B_{in}\left[\overline{A \oplus B}\right]$$

$$= B_{in} \oplus (A \oplus B)$$

$$B_{out} = \bar{A}B_{in} + \bar{A}B + BB_{in}$$





Implementation of full subtractor.

Implementation of F.S using two H.S's. & OR gate.

$$B_{out} = \bar{A}B_{in} + \bar{A}B + BB_{in}$$

$$= \bar{A}B + [B_{in}(\bar{A}+B)]$$

$$= \bar{A}B + [B_{in}(\overline{\bar{A}\bar{B}+\bar{A}B})]$$

$$\Rightarrow \bar{A}B + \bar{A}B_{in}(B+\bar{B}) + BB_{in}$$

$$\Rightarrow \bar{A}B + \bar{A}BB_{in} + \bar{A}\bar{B}B_{in} + BB_{in}$$

$$\Rightarrow \bar{A}\bar{B}B_{in} + \bar{A}B(1+B_{in}) + BB_{in}$$

$$\Rightarrow \bar{A}\bar{B}B_{in} + \bar{A}B + BB_{in}$$

$$\Rightarrow \bar{A}\bar{B}B_{in} + BB_{in}(A+\bar{A}) + \bar{A}B$$

$$\Rightarrow \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}BB_{in} + \bar{A}B$$

$$\Rightarrow \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}B(1+B_{in})$$

$$\Rightarrow \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}B.$$

$$\Rightarrow \bar{A}B + (AB + \bar{A}\bar{B})B_{in}$$

$$\Rightarrow \bar{A}B + (\overline{\bar{A}B + A\bar{B}})B_{in}$$

$$\Rightarrow \bar{A}B + (\overline{A \oplus B})B_{in}$$

$$\overline{\bar{A}B + A\bar{B}}$$
$$= \overline{\bar{A}B} \cdot \overline{A\bar{B}}$$
$$= (\bar{\bar{A}} + \bar{B})(\bar{A} + \bar{\bar{B}})$$
$$= (A + \bar{B})(\bar{A} + B)$$
$$= AB + \bar{A}\bar{B}.$$

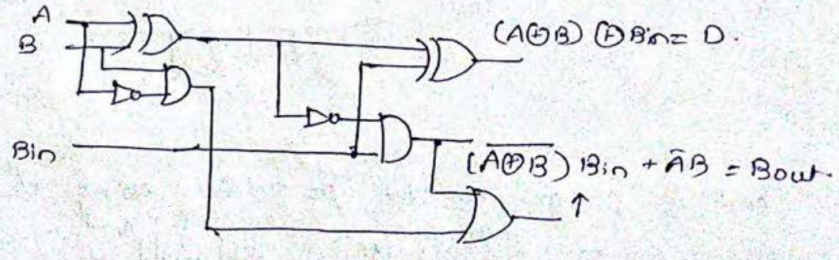
$\bar{A}B + \bar{A}BB_{in} + \bar{A}\bar{B}B_{in} + BB_{in}$

$\Rightarrow \bar{A}B + \bar{A}BB_{in} + BB_{in}(A+\bar{A})$

$\Rightarrow \bar{A}B + \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}BB_{in}$

$= \bar{A}B + \bar{A}\bar{B}B_{in} + ABB_{in}$

$= \bar{A}B + B_{in}[\bar{A}\bar{B} + AB]$

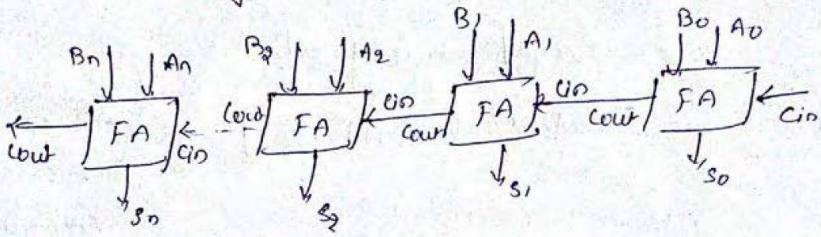


(Ex) 61, 63, 70, 71, 74, 76, 77, 79, 80, 83, 86, 91, 95, 96, 97, 98, 99

A, A6, A7, B7, B4, 1 [Ex. A2]

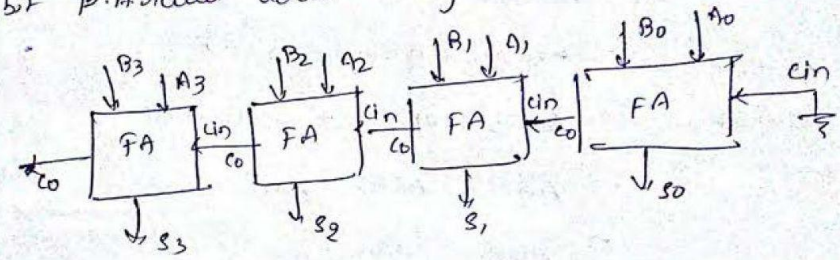
Binary Adder / parallel Adder :

Block dia of nbit parallel adder.



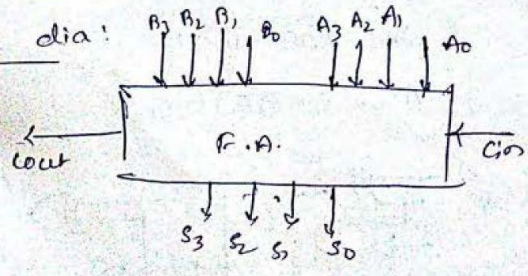
Note : carry = 0 bcz there is no carry into LSB position.

Ex: 4bit p.A parallel adder using F.A.



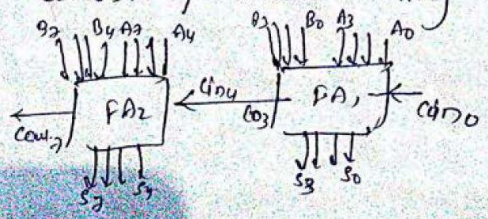
Block dia:

Logic dia:



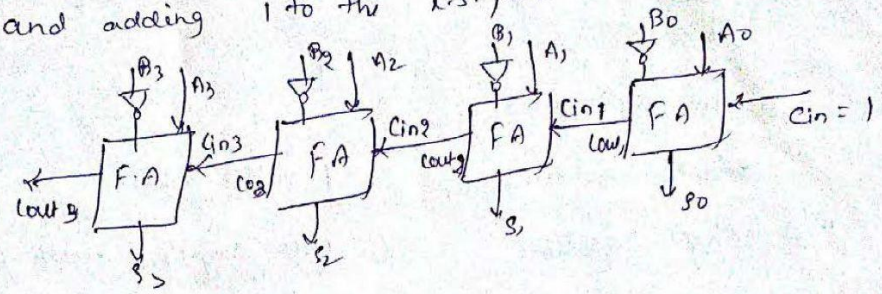
Binary parallel adder (IC 74LS83/74LS283) (Functional sym:)

Ex 2: design an 8bit pa adder using two 74283s.



Binary subtractor / parallel sub:

Her sub can be obtained by taking 2's comp of -ve no., 2's comp can be obtained by taking 1's comp and adding 1 to the l.s. pair bit '1'.



Parallel adder / subtractor:

M = Mode i/p controls the operation (ie) addition/sub.

M = 0 = addition

M = 1 = sub.

when M = 0, one i/p is

each XOR gate receives 1 i/p = M

2's i/p = B

addition ⇒ (ie) M = 0 ⇒ $M \oplus B$ ⇒ $0 \oplus B = 0\bar{B} + 1 \cdot B$

⇒ B

Cin0 = 0

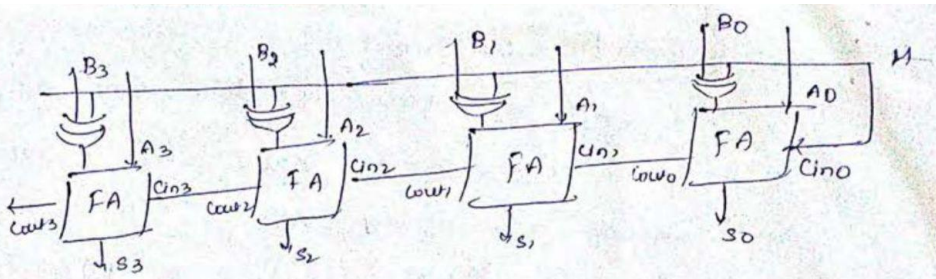
M = 1 ⇒ $M \oplus B$ ⇒ $1 \oplus B = 1\bar{B} + 0 \cdot B$

$= \bar{B}$

Cin0 = 1

ie previous sub bcz

B i/p's are all complemented & 1 is added.



4 bit adder- subtractor:

serial adder	parallel adder.
1. uses shift registers	1. uses registers with parallel load capacity
2. require only 1 full adder ckt	2. No. of F.A. ckt = no of bits in the binary no.
3. It is sequential ckt	3. Excluding reg, It is a purely combinational ckt
4. Time required for addition depends on no. of bits	4. doesn't depend of no. of bits
5. It is slower	5. faster

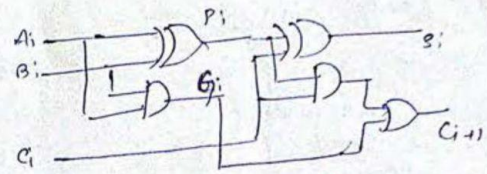
carry look ahead adder:

* In parallel adder the higher bits are generated iff lower bits are added producing sum & carry this results in delay called carry propdelay.

* One method of speeding up the process by eliminating inter stage carry delay is look ahead carry addition

It uses two functions 1. carry generation
2. carry propagation.

Full adder ckt:



$P_i = A_i \oplus B_i$

$G_i = A_i B_i$

$S_i = P_i \oplus C_i$

$C_{i+1} = P_i C_i + G_i$

$i=1$, C_1

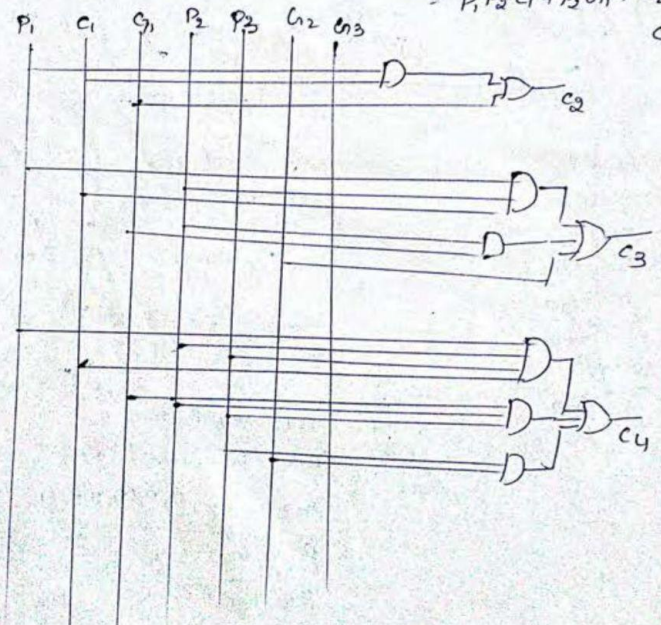
$C_2 = P_1 C_1 + G_1$

$C_3 = P_2 C_2 + G_2 = P_2 (P_1 C_1 + G_1) + G_2$

$= P_1 P_2 C_1 + P_2 G_1 + G_2$

$C_4 = P_3 C_3 + G_3$

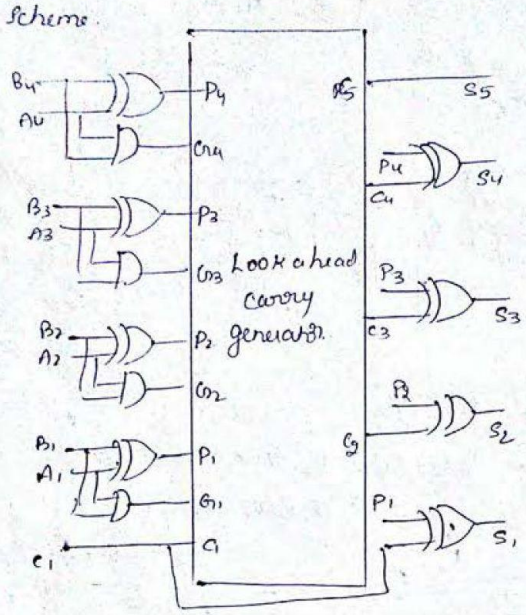
$= P_3 P_2 P_1 C_1 + P_3 P_2 G_1 + P_3 G_2 + G_3$



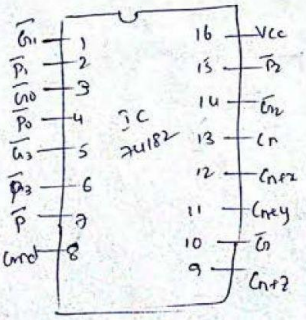
from above it is clear that C_4 ~~has~~ doesnt have to wait untill generation of C_2 & C_3 ..

Vinay (404)

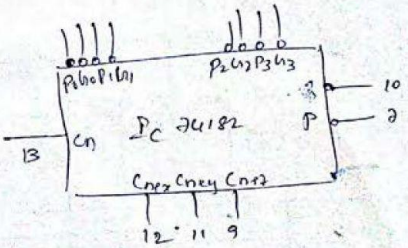
4-bit parallel adder with a look-ahead carry scheme.



Logic sym



Pin dia.



$C_{n+x} = G_0 + P_0 C_n$

$C_{n+y} = G_1 + P_1 G_0 + P_1 P_0 C_n$

$C_{n+z} = G_2 + P_2 P_1 G_0 + P_2 G_1$

$\overline{G} = \overline{G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0}$

$\overline{P} = \overline{P_3 P_2 P_1 P_0}$

Decimal adder:

BCD addition:

1. add two no.s
2. $\leq 9 \Rightarrow$ no correction
3. $> 9 \Rightarrow C = 1$, sum invalid.

then add 0110 to 4bit sum if carry = 1 again add it to the next higher order BCD digit.

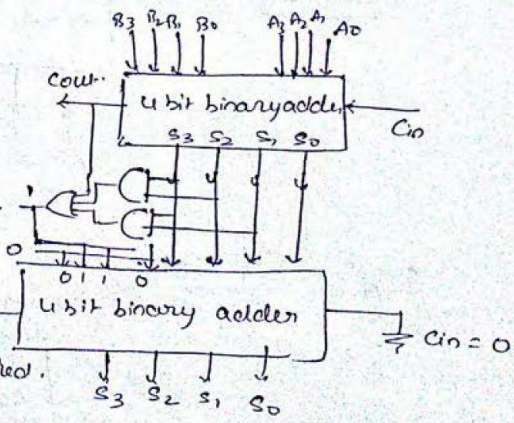
i/p's				o/p's
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

for sum > 9, Y = 1

S_3S_2 \ S_1S_0	00	01	11	10
	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
00 $\bar{A}\bar{B}$	0	0	0	0
01 $\bar{A}B$	0	0	0	0
11 AB	1	1	1	1
10 $A\bar{B}$	0	0	1	1

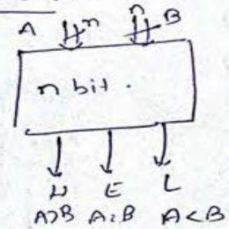
$Y = AB + AC$

ie $\Rightarrow S_3S_2 + S_3S_1$



Magnitude comparator:

Block dia:

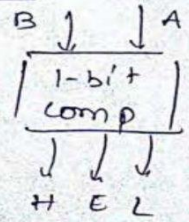


when $H = 1 \Rightarrow A > B$

$E = 1 \Rightarrow A = B$

$L = 1 \Rightarrow A < B$

Single bit comparator:



T.T

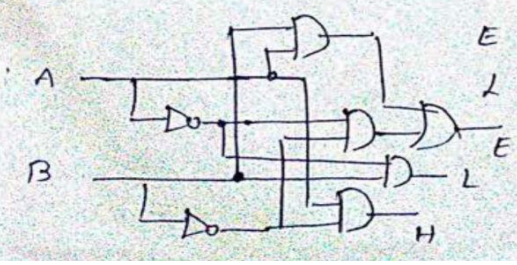
A	B	H	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$H = A\bar{B}$

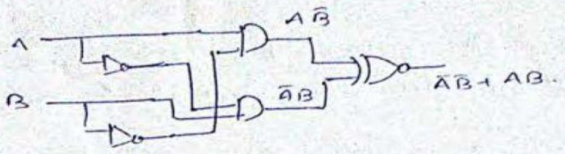
$E = \bar{A}\bar{B} + AB = \overline{A \oplus B}$

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

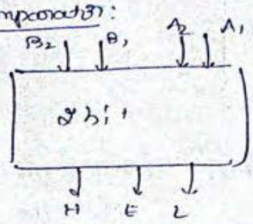
Logic dia:



1981: 3, 5, 8, 15, 29, 37, 39, 51, 59, 64.



Two Bit Comparator:



T.T

A_2	A_1	B_2	B_1
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

o/ps

H	E	L	
0	1	0	— 0
0	0	1	— 1
0	0	1	— 2
0	0	1	— 3
1	0	0	— 4
0	1	0	— 5
0	0	1	— 6
0	0	1	— 7
1	0	0	— 8
1	0	0	— 9
0	1	0	— 10
0	0	1	— 11
1	0	0	— 12
1	0	0	— 13
1	0	0	— 14
0	1	0	— 15

K-map:

H

A_2A_1 \ B_2B_1	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	0
10	1	1	0	0

$H = A_2\bar{B}_2 + A_2\bar{A}_1\bar{B}_1 + A_1\bar{B}_2\bar{B}_1$

E

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

$E = \bar{A}_2\bar{A}_1B_2\bar{B}_1 + \bar{A}_2A_1\bar{B}_2B_1 + A_1A_2B_1B_2 + A_2\bar{A}_1 + B_2\bar{B}_1$

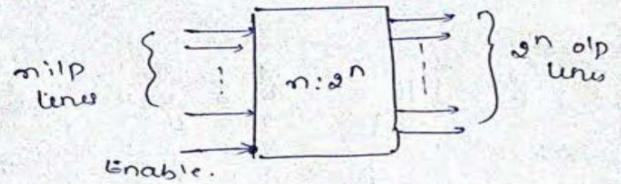
logic dia for H, E, L

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

$L = \bar{A}_2B_2 + \bar{A}_2\bar{A}_1B_1 + B_2B_1\bar{A}_1$

Decoder:

converts an n bit binary i/p code into 2^n o/p lines (ie) only 1 o/p line is active for each one possibility.



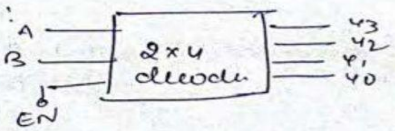
represented by n to m lines (or) n×m lines (or) n:m lines.

n = i/p's

m = o/p's

2 to 4 line decoder:

logic dia:



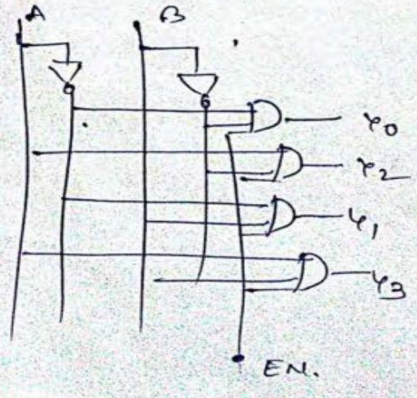
i/p's			o/p's			
EN	A	B	Y_0	Y_1	Y_2	Y_3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$Y_0 = \bar{A}\bar{B}$

$Y_1 = \bar{A}B$

$Y_2 = A\bar{B}$

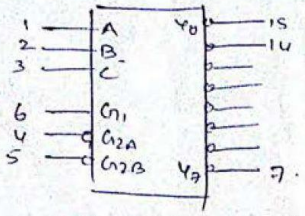
$Y_3 = AB$



3to8 line decoder:

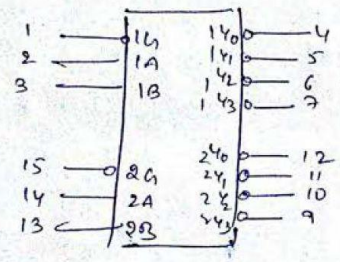
3 i/p's
8 o/p's

74x138 3to8 decoder:



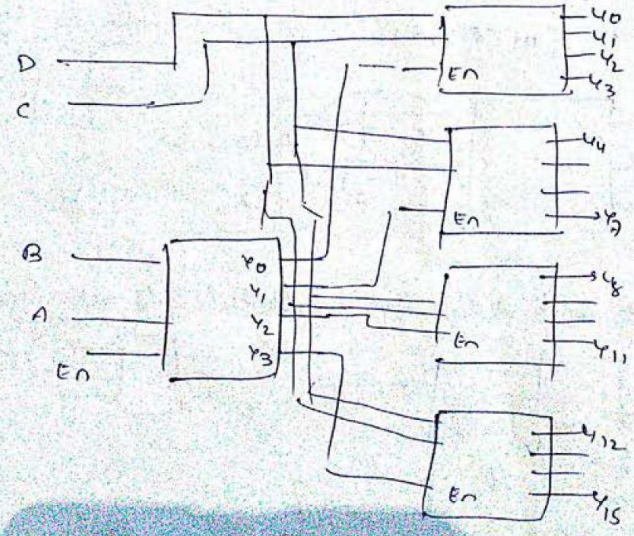
i/p's						o/p's							
G_{2B}	G_{2A}	G_1	C	B	A	$\bar{Y}_7$	$\bar{Y}_6$	$\bar{Y}_5$	$\bar{Y}_4$	$\bar{Y}_3$	$\bar{Y}_2$	$\bar{Y}_1$	$\bar{Y}_0$
1	x	x	x	x	x	1	1	1	1	1	1	1	1
x	1	x	x	x	x	1	1	1	1	1	1	1	1
x	x	0	x	x	x	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	0
0	0	1	0	0	1								
⋮													

74x139 Dual 2to4 decoder

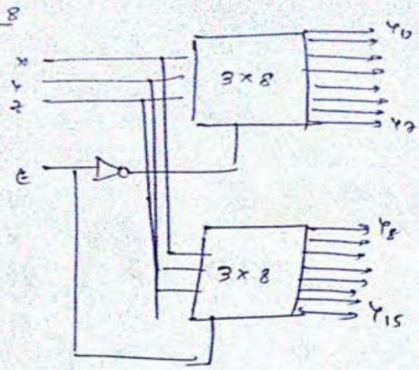


$\bar{G}$	B	A	$\bar{Y}_3$	$\bar{Y}_2$	$\bar{Y}_1$	$\bar{Y}_0$
1	x	x	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	0
0	1	1	0	1	1	1

4to16 using 2to4 decoders:



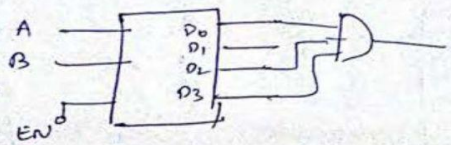
using 3 to 8



Realization of multiple o/p fns using decoder:

~~To realise~~ pos fun: Active high o/p.

Ex: ~~$F(A,B) = \Pi M(0,2,3)$~~ using 2 to 4 decoder with active high o/p.



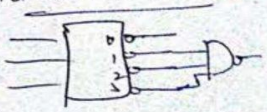
$\overset{4}{0}\overset{2}{0}\overset{1}{0}$ 010 011

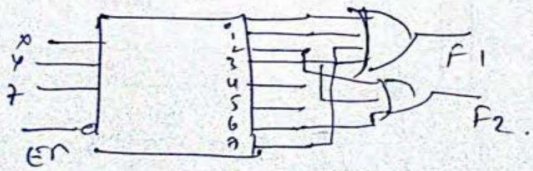
$F = (\bar{A}+\bar{B}+C)(A+\bar{B}+C)(A+\bar{B}+\bar{C})$

Ex: 1. $F_1(x,y,z) = \Sigma m(0,1,3,7)$ (OR gate) $\Sigma m(1,2,3)$

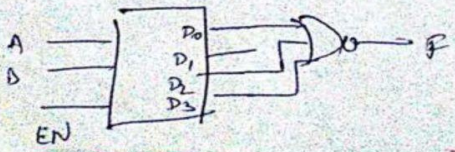
$F_2(x,y,z) = \Sigma m(2,3,7)$

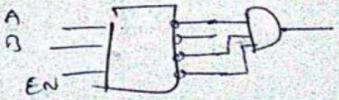
to active low decoder





2. $\Pi M(0,2,3)$ using Active high decoder. to active low:



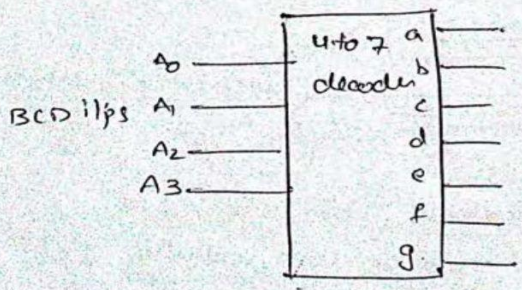


BCD to Decimal decoder:

i/p BCD code				Actual o/p.										
B_3	B_2	B_1	B_0	$\bar{D}_0$	$\bar{D}_1$	$\bar{D}_2$	$\bar{D}_3$	$\bar{D}_4$	$\bar{D}_5$	$\bar{D}_6$	$\bar{D}_7$	$\bar{D}_8$	$\bar{D}_9$	$\bar{D}_{10}$ to $\bar{D}_{11}$ to $\bar{D}_{15}$
0	0	0	0	0	1	1	1	1	1	1	1	1	1	None.
0	0	0	1	1	0	1	1	1	1	1	1	1	1	
0	0	1	0	1	1	0	1	1	1	1	1	1	1	
0	0	1	1	1	1	1	0	1	1	1	1	1	1	
0	1	0	0	1	1	1	1	0	1	1	1	1	1	
0	1	0	1	1	1	1	1	1	0	1	1	1	1	
0	1	1	0	1	1	1	1	1	1	0	1	1	1	
0	1	1	1	1	1	1	1	1	1	1	0	1	1	
1	0	0	0	1	1	1	1	1	1	1	1	0	1	
1	0	0	1	1	1	1	1	1	1	1	1	1	0	
1	0	1	0	—	-	-	-							
1	0	1	1											
1	1	0	0											
1	1	0	1											
1	1	1	0											
1	1	1	1											

NIONE

* BCD to 7 seg Decoder:



common cathode

Decimal digit	BCD A_0	A_1	A_2	A_3	seven seg code a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
⋮											
9	1	0	0	1	1	1	1	1	0	1	1

1) Common cathod type GND. => High = 1 => o/p.

Common anode type Vcc => low : 0 => o/p.

K-maps: CD (for a)

A_0A_1 \ A_2A_3	00	01	11	10
AB 00	1	0	1	1
01	0	1	1	1
11	x	x	x	x
00	1	1	x	1

$a = A_0 + A_2 + A_1A_3 + \overline{A_1}\,\overline{A_3}$

$A + C + BD + \overline{B}\,\overline{D}$

for b : $\overline{A_1} + \overline{A_2}\,\overline{A_3} + A_2A_3$

$= B + \overline{C}\,\overline{D} + CD$

c : $B + \overline{C} + D$

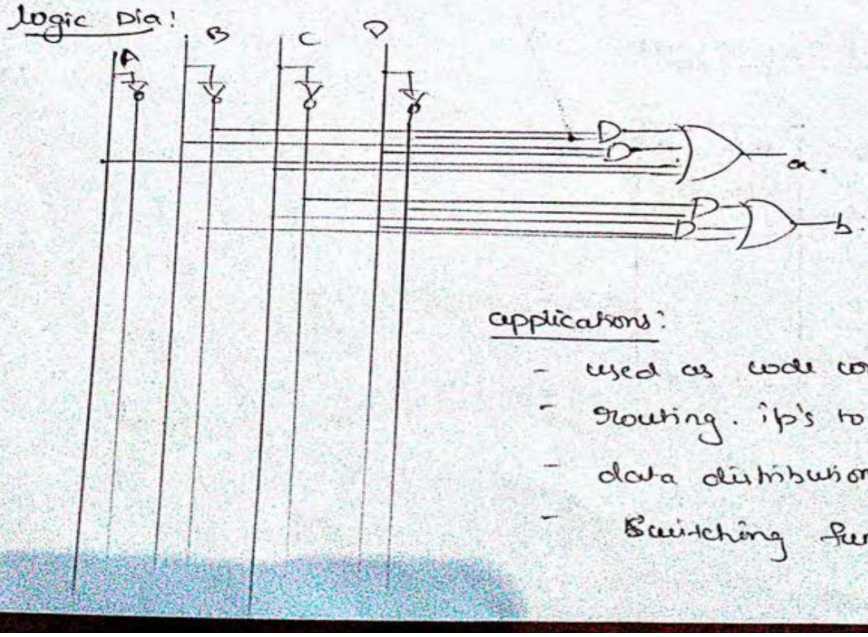
d : $\overline{B}\,\overline{D} + C\overline{D} + B\overline{C}D + \overline{B}C + A$

e : $\overline{B}\,\overline{D} + C\overline{D}$

f : $A + \overline{C}\,\overline{D} + B\overline{C} + B\overline{D}$

g : $A + B\overline{C} + \overline{B}C + C\overline{D}$

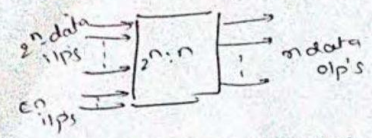
logic Dia:



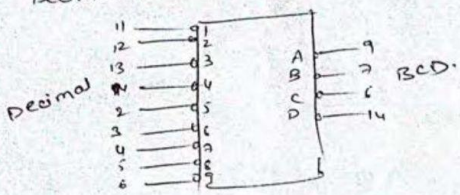
applications:

- used as code converters
- routing. i/p's to specified o/p's
- data distribution. (demux).
- Switching fun.

Encoders:



Decimal to BCD Encoder: (IC 74xx147).



Decimal i/p's

	A	B	C	D	4	5	6	7	8	9
0	1	2	3							
0	1	1	1		1	1	1	1	1	1
1	0	1	1		1	1	1	1	1	1
2	x	0	1		1	1	1	1	1	1
3	x	x	0							

o/p's

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

4 to 2 Encoder:

A_3	A_2	A_1	A_0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

Y_1	Y_0
0	0
0	1
1	0
1	1

$Y_1 = \bar{A}_3 A_2 \bar{A}_1 \bar{A}_0 + A_3 \bar{A}_2 \bar{A}_1 \bar{A}_0$

Priority encoders:

dis: If only 1 i/p is high => o/p is correct in some case

& (or) more i/p's might be high.

Ex: operating a key board, a person by releasing a key he may press the other so problem arises there.

Decimal to BCD priority encoder:

D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	A_3	A_2	A_1	A_0
1	1	1	1	1	1	1	1	1	1	1	1	1
x	x	x	x	x	x	x	x	0	0	1	1	0
x	x	x	x	x	x	x	0	1	0	1	1	1
x	x	x	x	x	x	0	1	1	1	0	0	0

⋮

octal to Binary priority encoder:

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0

⋮

Priority encoder:

priority fun is included.

Ex: if 2 (or) more i/ps = 1 at same time, i/p having highest priority will take precedance

4 bit priority encoder

i/p's				o/p's		
D_0	D_1	D_2	D_3	Y_0	Y_1	V (Valid o/p indicator)
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1

$Y_0 = D_3 + D_1\overline{D_2}$

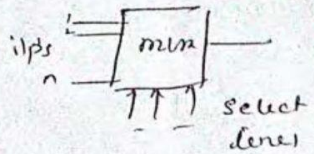
$Y_1 = D_2 + D_3$

$V = D_0 + D_1 + D_2 + D_3$

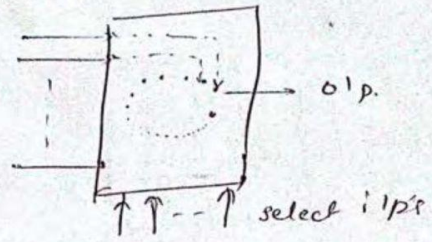
mux :

i/p are 'n'
o/p = 1

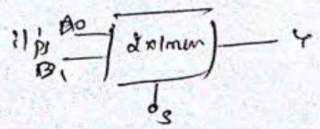
Block dia'



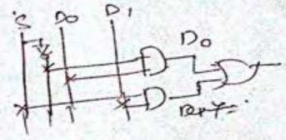
Functional dia'



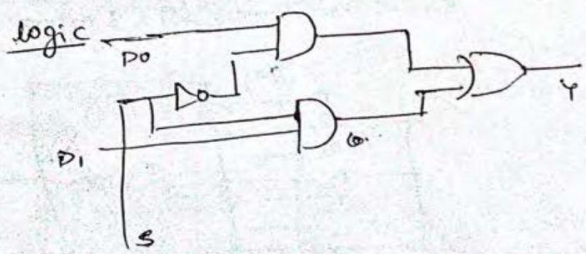
2x1 mux:



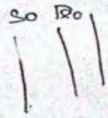
S	A	B	Y
0			D0
1			D1



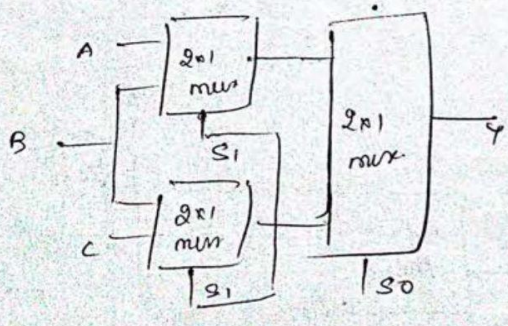
logic



when S = 0
Y = D0
when S = 1
Y = D1



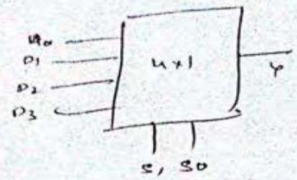
3x1 mux using 2x1 mux



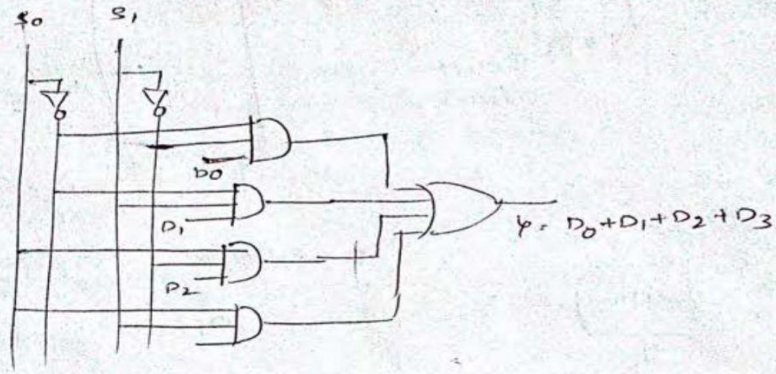
S0	S1	S1	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

S0	S1	Y
0	0	A
0	1	B
1	0	D
1	1	C

4x1 mux:



S_0	S_1	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

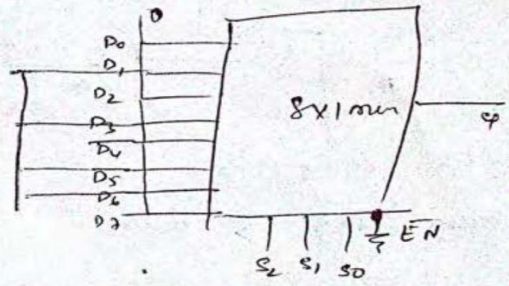


8x1 mux:

Ex: 1) Implement $F = \Sigma m(1,3,5,6)$ using 8x1 mux.

T.T:

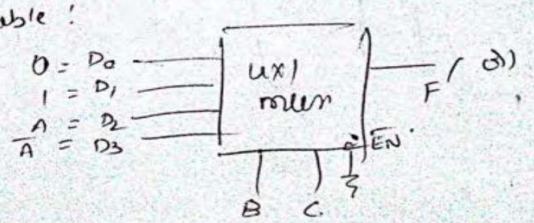
EN	S			o/p
E	S_2	S_1	S_0	Y.
1	x	x	x	0
0	0	0	0	D_0
0	0	0	1	D_1
⋮				
0	1	1	1	D_7



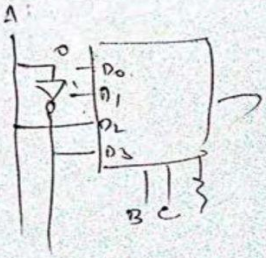
b) $\Sigma m(1,3,5,6)$

Implementation table:

	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	0	D_1	$\bar{A}$	$\bar{A}$



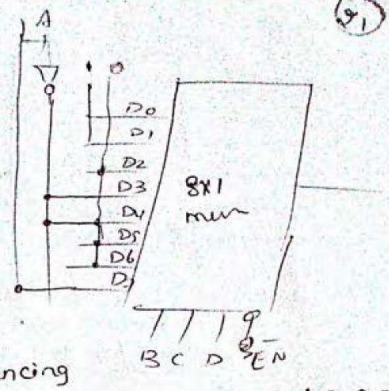
(or)



8x1 mux $\Sigma m(0,1,3,4,8,9,15)$

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1		$\bar{A}$	$\bar{A}$			$\hat{A}$

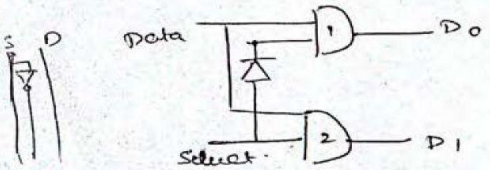
$\Pi M(2,5,6,7,10,11,12,13,14)$



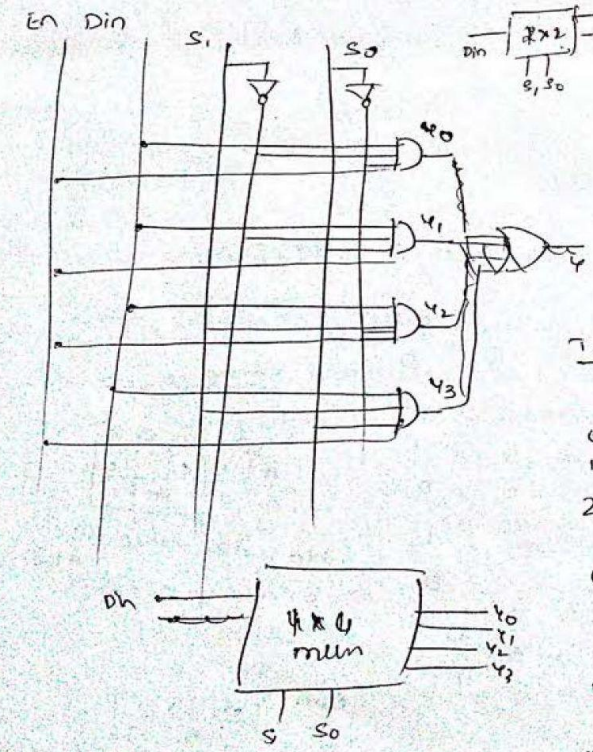
app: 1. 7 seg display 3. Digital counter
2. logic fun gen 4. operating sequencing
5. Data selection, routing 6. parallel & serial conversion
7. code generation.

De mux:

logic dia:



En	S_1	S_0	Din	Y_0	Y_1	Y_2	Y_3
0	x	x	x	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	0
1	1	1	1	0	0	0	1



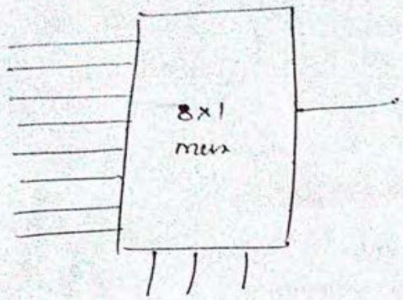
Ex: Implement F.S using demux

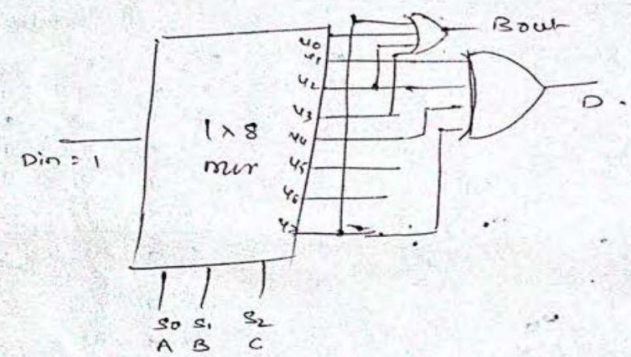
T.T F.F

	A	B	C	D	B_0
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	0
7	1	1	1	1	1

$D = \Sigma m(1,2,4,7)$

$B_0 = \Sigma m(1,2,3,5,7)$





Parity checker / Generator:

TT of parity generator for even & odd parity

A	B	C	Odd	Even
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

K-map:

Odd

	$B\bar{C}$	$\bar{B}\bar{C}$	BC	$B\bar{C}$
$\bar{A}$	1	0	1	0
A	0	1	0	1

$P_{odd}: \bar{A}\bar{B}\bar{C} + ABC + \bar{A}BC + AB\bar{C}$

$\Rightarrow \bar{C}(A \odot B) + C(A \oplus B)$

$\Rightarrow (A \oplus B) \oplus C$

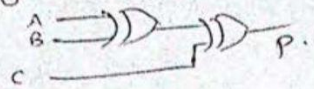
Even

0	1	0	1
1	0	1	0

$P_{even}: A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$

$\Rightarrow (A \oplus B) \oplus C.$

logic dia:



7.7 for Even parity checker:

4 bits received

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

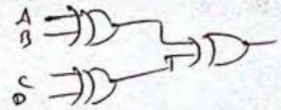
parity error check.

PEC.

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

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

$PEC = (A \oplus B) \oplus (C \oplus D)$



code converter:

Binary to BCD converter:

Binary

D	C	B	A
0	0	0	0
0	0	0	1
0	0	1	0
⋮			
1	0	0	1
1	0	1	0
1	0	1	1

BCD code.

B_4	B_3	B_2	B_1	B_0
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	1	0	0	1
1	0	0	0	0
1	0	0	0	1

k-map.

$B_4 = DC + DB$

$B_3 = D\bar{C}\bar{B}$

$B_2 = \bar{D}C + CB$

$B_1 = DC\bar{B} + \bar{D}B$

$B_0 = A.$

logic dia:

BCD to Binary:

B_4	B_3	B_2	B_1	B_0
0	0	0	0	0
0	0	0	0	1
⋮				
1	0	0	0	0
1	0	0	0	1
1	0	1	0	1
1	0	1	1	0

map

E	D	C	B	A
0	0	0	0	0
0	0	0	0	1
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	0	0	1

5variable K-map:

$A = B_0$

$B_4 = 0$

$B_4 = 1$

$B_3B_2 \backslash B_1B_0$

$B = B_1 \oplus B_4$

$C = \overline{B_4}B_2 + B_2\overline{B_1} + B_4\overline{B_2}B_1$

$D = \overline{B_4}B_3 + B_4\overline{B_3}\,\overline{B_2} + B_4\overline{B_3}\,\overline{B_1}$

$E = B_4B_3 + B_4B_2B_1$

logic dia:

BCD to Excess-3:

Decimal	BCD B_3	B_2	B_1	B_0	Excess-3 E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	1	0	0
1								
⋮								
9	1	0	0	1	1	1	0	0

K-map:

$E_3 = B_3 + B_2(B_0 + B_1)$

$E_2 = B_2\bar{B}_1\bar{B}_0 + \bar{B}_2(B_0 + B_1)$

$E_1 = B_1 \oplus B_0$

$E_0 = \bar{B}_0$

logic dia:

Excess-3 to BCD code converter:

E_3	E_2	E_1	E_0	B_3	B_2	B_1	B_0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
⋮							

K-map:

$B_0 = \bar{E}_0$

$B_1 = E_1 \oplus E_0$

$B_2 = \bar{E}_2\bar{E}_1 + E_2E_1E_0 + E_3E_1\bar{E}_0$

$B_3 = E_3E_2 + E_3E_1E_0$

logic dia:

Binary to Gray code:

Binary B_3	B_2	B_1	B_0	Gray G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
⋮							

K-map:

$G_0 = B \oplus A$

$G_1 = C \oplus B$

$G_2 = D \oplus C$

$G_3 = D$.

logic dia: