

ANNAMACHARYA UNIVERSITY

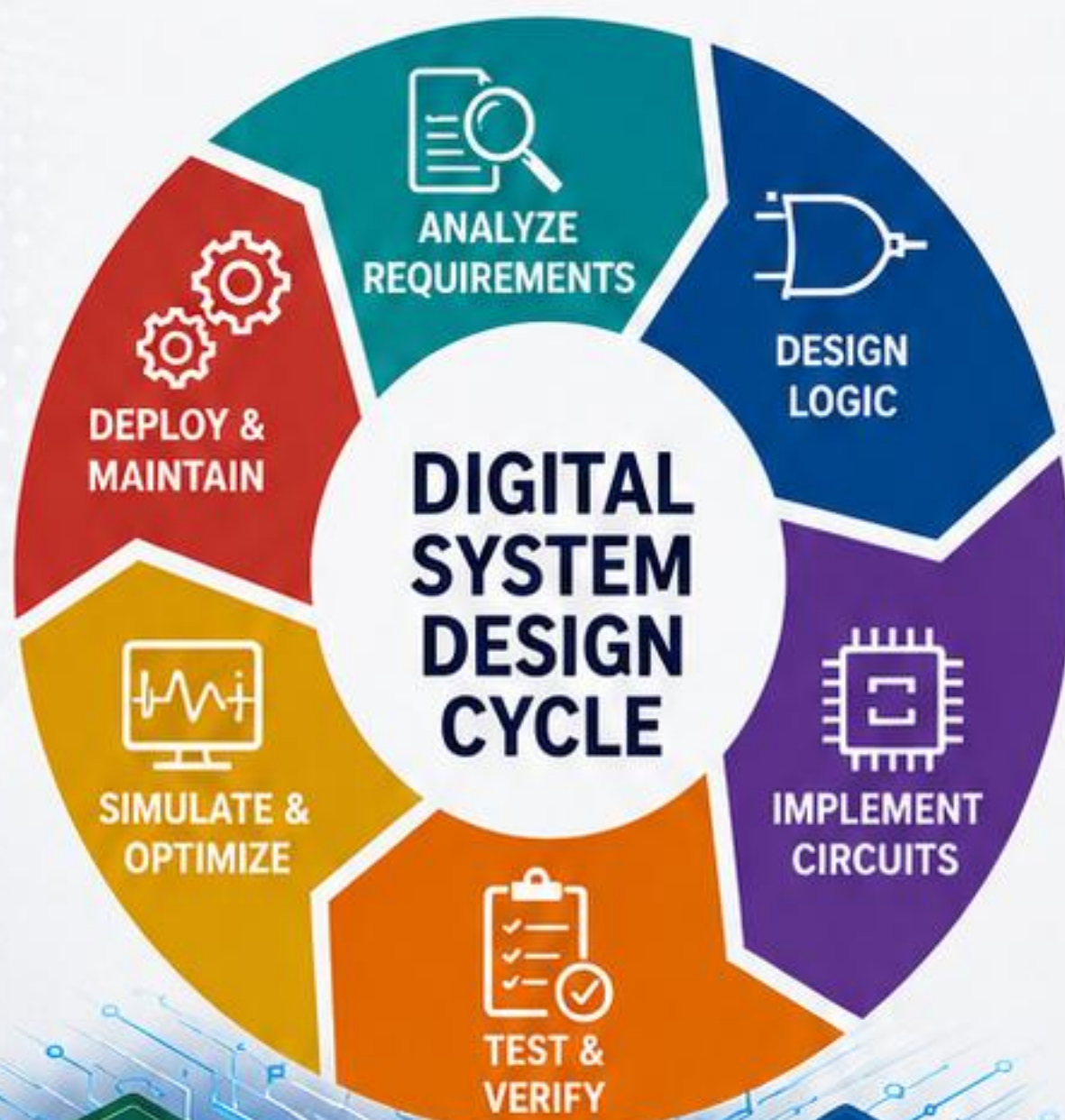
• RAJAMPET •

Department of CSE(AI)

SECOND YEAR B.TECH

DIGITAL LOGIC DESIGN & COMPUTER ORGANISATION

II B.Tech | Sem V | CSE(AI) | R24 Regulations



DIGITAL SYSTEM FUNDAMENTALS

Understand the building blocks of digital systems.



LOGIC DESIGN & OPTIMIZATION

Design and minimize combinational & sequential circuits.



COMPUTER ORGANISATION

Study the structure and function of computer systems.



MEMORY & I/O SYSTEMS

Explore memory hierarchy and I/O mechanisms.



PERFORMANCE AWARENESS

Analyze system performance and design efficient solutions.



• Author •

S. Ramalakshmi

Assistant Professor,
Department of CSE(AI),
Annamacharya University,
Rajampet

UNIT - I

DATA REPRESENTATION

Digital Systems: Digital systems are systems that process and represent information in digital form, using discrete values such as 0s and 1s to represent data, rather than continuous signals like analog systems.

Examples of Digital systems include:

Computers (H/W & S/W), Mobile devices, Digital circuits, Microcontrollers, Digital Cameras, Digital signal processors, Personal, handheld, touch-screen devices etc.

Advantages: High Accuracy, Fast processing and transmission of data, Easy storage and retrieval of data, low noise, High reliability, Flexibility, Ability to perform complex tasks and calculations.

Limitations: Digital noise and errors, Depends on software and hardware reliability, Cyber security risks etc.

Digital Circuits: Digital Circuits are electronic circuits that process and control digital signals, that have only two distinct values, represented by 0s and 1s. These circuits are the building blocks of digital systems and are used in a wide range of applications.

* They are Computers and Laptops, Smartphones and tablets, Digital watches and clocks, Calculators and ALUs, Digital logic gates and Flip-flops, Counters and timers etc.

* Digital circuits are designed using various logic gates and circuits, such as: AND, OR, and NOT gates, NAND, NOR and XOR gates, Flip-flops Multiplexers and demultiplexers etc.

* By using various techniques and tools such as Boolean Algebra, Logic gates and circuits, Truth-tables, K-maps, digital circuits are designed.

→ Data Representation in digital systems can be categorized into several types as Numeric, Character, Image, Audio, Video, Text.

Numeric representation refers to the way numbers are represented in digital systems.

Common Representations are Binary, Decimal, hexadecimal, octal, Fixed-point, Floating-point, signed, unsigned.

Binary numbers: Binary numbers are a way of representing Information using only two digits: 0 & 1.

Binary numbers are made up of bits (0s & 1s)

Each bit can have one of two values: 0 or 1

Binary numbers can be used to represent Integers, Fractions, Characters, Instructions.

Radix (or) Base:

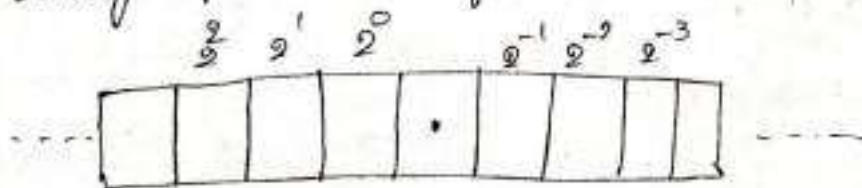
It specifies the number of symbols used for a particular number system.

Radix point:

In any number system, the radix point specifies the dividing line between the Integer part and fractional part.

The base (or) radix of binary numbers is 2. The binary number system is also called as radix-2 number system or base-2 number system.

The binary point in a binary number separates the Integer part and fractional part



In Binary number system, the weights are expressed in terms of the powers of 2.

By adding each digit multiplied by its respective weight in the given binary number, we can obtain the decimal equivalent number.

Ex: Consider a decimal number 7,392 which represents a quantity equal to 7 thousands, plus 3 hundreds, plus 9 tens, plus 2 units

7,392 can be written as

$$7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$$

The decimal number system is said to be of base or radix 10, because it uses 10 digits

In a decimal number, the left most digit which has the highest weight is called as Most Significant Digit (MSD) and the right most digit that has the lowest weight is called as Least Significant Digit (LSD).

Ex: The binary equivalent of 7,392 is

Consider a binary number 11010.11 is 26.75

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 +$$

$$0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$$

$$= (26.75)_{10}$$

$$\begin{array}{r} 2 \overline{) 7392} \\ \underline{21696} \\ 2 \overline{) 848} \\ \underline{2424} \\ 2 \overline{) 212} \\ \underline{2106} \\ 2 \overline{) 106} \\ \underline{253} \end{array}$$

In general, a number expressed in a base- r system has co-efficients multiplied by powers of ' r '.

$$a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + \dots + a_2 \cdot r^2 + a_1 \cdot r + a_0 + a_{-1} \cdot r^{-1} + a_{-2} \cdot r^{-2} + \dots + a_{-m} \cdot r^{-m}$$

The co-efficients a_j range from 0 to $r-1$.

Ex. of a base-5 number is

$$(4021.2)_5 = 4 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1} \\ = (511.4)_{10}$$

The co-efficient values for base-5 can be only 0, 1, 2, 3 & 4.

The octal number system is a base-8 system that have 8 digits: 0, 1, 2, 3, 4, 5, 6, 7.

$$\text{Ex: } (127.4)_8 = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} \\ = (87.5)_{10}$$

Note that the digits 8 & 9 cannot appear in an octal number.

When the base of the number is greater than 10, the letters of the alphabet are used to supplement the 10 decimal digits.

Ex: Hexadecimal (base-16) number system.

Letters A, B, C, D, E & F are used for the

the digits of 10, 11, 12, 13, 14 & 15 respectively.

$$\begin{aligned}(B65F)_{16} &= 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 \\ &= (46,687)_{10}\end{aligned}$$

The conversion from binary to decimal can be obtained by adding only the numbers with powers of two corresponding to the bits that are equal to 1.

$$\begin{aligned}\text{Ex: } (110101)_2 &= 32 \\ &= 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 32 + 16 + 4 + 1 \\ &= (53)_{10}\end{aligned}$$

Fixed-point Representation:

Real numbers can be represented in Computer

in two ways.

1. Fixed point represent.
2. Floating point represent.

① Fixed-point

In this representation, the decimal point is placed in a fixed place.

A number ending with a decimal point is called a whole number/integer and a number starting

with a decimal point is called a fraction.

Ex: 45.00 - whole number/Integer
 0.45 - fraction number
 0.1011 - fraction number
 23.25 - 23 is the whole number/Integer
 $.25$ is the fraction number

② Floating point Representation

In this representation, the position of the decimal point is not specified. By this, a number can be expressed in different ways.

Ex: $(435)_{10}$, this number can be expressed as 435×10^0 or 435×10^1 or 4.35×10^2 etc.
 similarly $(10.1011)_2 \rightarrow 1.01011 \times 2^1$

When the radix point/binary point is shifted to left by a 1 bit position, the exponent will be increased by '1'.

When the radix point/binary point is shifted to right by a 1 bit position, the exponent will be decreased by '1'.

i) $(537.25)_{10} \rightarrow M \times r^e$
 0.53725×10^3
 M - Mantissa, e -> exponent
 r - radix

Number Base Conversions:

The conversion of a decimal number in base '10' to a number in base 'r' is done by dividing the number and all successive quotients by r and accumulating the remainders.

Ex: Convert Decimal 41 to binary.

$$\begin{array}{r} 2 \overline{) 41} \\ 2 \overline{) 20} - 1 \\ 2 \overline{) 10} - 0 \\ 2 \overline{) 5} - 0 \\ 2 \overline{) 2} - 1 \\ 1 - 0 \end{array}$$

$$\therefore (41)_{10} = (101001)_2$$

2) Convert Decimal 153 to Octal.

Here decimal 153 is of base '10'.
Required base is '8', hence 153 is divided by '8' successively.

$$\begin{array}{r} 8 \overline{) 153} \\ 8 \overline{) 19} - 1 \\ 2 - 3 \end{array}$$

$$\therefore (153)_{10} = (231)_8$$

3) Convert $(0.6875)_{10}$ to binary.

Conversion of given decimal fraction to binary is done by a method of multiplication to get a new fraction until the fraction becomes '0' or the number of digits are sufficient.

$$0.6875 \times 2 = 1.3750 \Rightarrow \overset{\text{Integer}}{1}$$

$$0.3750 \times 2 = 0.7500 \Rightarrow 0$$

$$0.7500 \times 2 = 1.5000 \Rightarrow 1$$

$$0.5000 \times 2 = 1.0000 \Rightarrow 1$$

$$\therefore (0.6875)_{10} = (0.1011)_2$$

4) Convert $(0.513)_{10}$ to octal.

$(0.513)_{10} \rightarrow$ Required Radix '8'

$$0.513 \times 8 = 4.104 \Rightarrow \overset{\text{Intg}}{4}$$

$$0.104 \times 8 = 0.832 \Rightarrow 0$$

$$0.832 \times 8 = 6.656 \Rightarrow 6$$

$$0.656 \times 8 = 5.248 \Rightarrow 5$$

$\therefore$ Here 4 digits are sufficient,

$$(0.513)_{10} = (0.4065)_8$$

Practise problem.

1) Convert $(117.23)_{10}$ to octal.

$$\text{Ans. } (65.1656)_8$$

Octal and Hexadecimal numbers

The conversion from and to binary, octal, and hexadecimal numbers plays an important role in digital computers because shorter patterns of hex characters are easier to recognize than long patterns of 1's & 0's.

Since $2^3 = 8$ & $2^4 = 16$, each octal digit corresponds to 3 binary digits and each hexadecimal digit corresponds to four binary digits.

The first 16 numbers in decimal, binary, octal & hexadecimal number systems are listed in table.

Decimal base-10	Binary base-2	Octal base-8	Hexadecimal base-16
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0010	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

(i) Conversion from Binary to Octal Number

→ This conversion is done by partitioning the binary number into groups of three digits each, starting from the binary point and proceeding to the left and to the right.

→ The corresponding octal digit is then assigned to each group.

Ex: Convert binary $(10110001101011.11110000110)_2$ into octal number.

<u>101</u>	<u>100</u>	<u>001</u>	<u>101</u>	<u>011</u>	.	<u>111</u>	<u>100</u>	<u>000</u>	<u>110</u>
5	4	1	5	3	.	7	4	0	6

$\Rightarrow (26153.7406)_8$

(ii) Conversion from Binary to Hexadecimal number

→ This conversion is done by partitioning the binary number into groups of four digits each, starting from the binary point and proceeding to the left and to the right.

→ The corresponding hexadecimal digit is then assigned to each group.

Ex: Convert $(10110001101011.11110010)_2$ in to
Hexadecimal number.

$$\begin{array}{ccccccc} \underline{1011} & \underline{0001} & \underline{1010} & \underline{11} & \cdot & \underline{1111} & \underline{0010} \\ 2 & C & 6 & B & & F & 2 \end{array}$$

$$\Rightarrow (2C6B.F2)_{16}$$

iii) Conversion From Octal to Hexadecimal to binary.

In case of Octal number, each octal digit is converted to its three digit binary equivalent. Similarly, each Hexadecimal digit is converted to its four-digit binary equivalent.

Ex: 1) Octal to binary

$$(673.124)_8 \Rightarrow (?)_2$$

$$\begin{array}{ccccccc} 6 & 7 & 3 & \cdot & 1 & 2 & 4 \\ 110 & 111 & 011 & & 001 & 010 & 100 \end{array}$$

$$\Rightarrow (110111011.001010100)_2$$

2) Hexa to binary

$$(306.D)_{16} \Rightarrow (?)_2$$

$$\begin{array}{cccc} 3 & 0 & 6 & \cdot & D \\ 0011 & 0000 & 0110 & & 1101 \end{array}$$

$$\Rightarrow (001100000110.1101)_2$$

Practise problems:

1) Find the binary representation of $(135)_{10}$

-Ans: $(135)_{10} = (1000\ 0111)_2$

2) Find the octal representation of $(135)_{10}$

-Ans: $(135)_{10} = (207)_8$

3) Convert the following numbers with the indicated bases to decimal.

a) $(1102)_4$ b) $(5134)_6$ c) $(9762)_{14}$

d) $(206)_9$

4) Convert the hexadecimal number DB4F to binary, and then convert it from binary to octal.

5) Express the following numbers in decimal.

a) $(10001.101)_2$ b) $(52.5)_{16}$

c) $(342.54)_8$ d) $(101010.101)_2$

COMPLEMENTS

Complements are used in digital computers to simplify the subtraction operation and for logical manipulation.

- There are two types of complements for each base- r system.
- The Radix Complement and the diminished Radix complement.
- The first is referred to as the r 's complement and the second as the $(r-1)$'s complement.

When the value of base r is substituted in the name, the two types are referred to as the 2's complement and 1's complement for binary numbers and the 10's complement and 9's complement for decimal numbers.

Diminished Radix Complement

Given a number N in base r having ' n ' digits, the $(r-1)$'s complement of N , that is its diminished radix complement is defined as $(r^n - 1) - N$.

For decimal numbers, $r=10$ & $r-1=9$, so the 9's complement of N is $(10^n - 1) - N$. Here 10^n represents a number that consists of a single 1 followed by ' n ' 0's. $10^n - 1$ is a number represented by ' n ' 9's.

Ex: If $n=4$, we have $10^4 = 10,000$ & $10^4 - 1 = 9999$

$\therefore$ The 9's complement of a decimal number is obtained by subtracting each digit from 9.

Ex: i) The 9's Complement of 546700 is

$$\begin{array}{r} 999999 \\ - 546700 \\ \hline 453299 \end{array}$$

ii) The 9's Complement of 012398 is

$$\begin{array}{r} 999999 \\ - 012398 \\ \hline 987601 \end{array}$$

For binary numbers, $r=2$ & $r-1=1$. So the 1's complement of 'N' is $(2^n - 1) - N$. Here 2^n is represented by a binary number that consists of a 1 followed by 'n' 0's. $2^n - 1$ is a binary number represented by 'n' 1's.

Ex: If $n=4$, we have $2^4 = (10000)_2$ & $2^4 - 1 = (1111)_2$

Thus, the 1's complement of a binary number is obtained by subtracting each digit from 1. Therefore, the 1's complement of a binary number is formed by changing 1's to 0's & 0's to 1's.

Ex: i) The 1's complement of 1011000 is 0100111

ii) The 1's complement of 0101101 is 1010010

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

from 7 8 1 5 (decimal 15) respectively.

Practise problems:

Find (a) The diminished radix (9's) complement of $(135)_{10}$

(b) Find 1's complement of the binary numbers

i) 11001101 ii) 10100101 iii) 01001001

Radix Complement

The r 's complement of an n -digit number N in base r is defined as $r^n - N$ for $N \neq 0$ and as 0 for $N = 0$.

The r 's complement is obtained by adding 1 to the $(r-1)$'s complement, since $r^n - N = [(r^n - 1) - N] + 1$.

Ex: The 10's complement of decimal 2389 is :
10's complement is obtained by adding 1 to the 9's complement value.

$\therefore$ 9's complement of 2389 is

$$\begin{array}{r} 9999 \\ 2389 \\ \hline 7610 \\ \text{Adding 1} \Rightarrow + \quad 1 \\ \hline 7611 \end{array}$$

$\therefore$ 10's complement of $(2389)_{10}$ is $(7611)_{10}$.

Ex: i) The 2's Complement of binary $(101100)_2$ is ?

2's Complement is obtained by adding 1 to the 1's Complement value.

$\therefore$ 1's Complement of binary 101100 is 010011

$$\text{Adding 1} \Rightarrow \begin{array}{r} + \quad 1 \\ \hline 010011 \\ \hline 010100 \end{array}$$

Note: The Complement of the Complement restores the number to its original value.

Practise problems:

- i) Find the radix (10's) complement of $(135)_{10}$
- ii) Find the 2's complement of binary numbers
 - a) 01101000 b) 10110100 c) 10100101
- iii) Find the 10's complement of decimal numbers
 - a) $87,000,367$ b) $99,999,000$
 - c) $56,783,223$
- iv) Find the 8's complement of $(1740)_8$
- v) Convert $(1740)_8$ to binary.

Subtraction with Complements (Subtraction with 10's Complement)

Minuend - Subtrahend (M-S)

- 1) Equate the number of digits by padding appropriate number of zeros in front of the numbers.
- 2) Find the 10's complement to subtrahend and add with Minuend.
- 3) If carry is generated, the result is positive and the carry is discarded to get the final result.
- 4) If carry is not generated, the result is negative and the result take 10's complement of the result to get the final result.

Ex: Using 10's complement, subtract $72532 - 3250$

Here $M = 72532$ & $S = 3250$

1) To equate, add '0' at in front of 'S' i.e., 03250

2) 10's Complement of 'S' is obtained by

W.K.T. 10's Complement = 9's Complement + 1

$$\begin{array}{r} 9's \text{ Complement of } S = 99999 \\ - 03250 \\ \hline 96749 \end{array}$$

$$\begin{array}{r} \text{Add } 1 \Rightarrow \\ 96749 \\ + 1 \\ \hline 96750 \end{array}$$

Now Add with 'M'

$$\begin{array}{r} 96750 \\ + 72532 \\ \hline \text{Carry } \swarrow \text{①} \quad 69282 \end{array}$$

3) Here, the carry is generated, the result is positive,
 carry is discarded
 $\therefore$ Final Result is 69282, which is positive.

Practise problems

1) Given the two binary numbers $X = 1010100$ &
 $Y = 1000011$, Perform the subtraction

a) $X - Y$ & b) $Y - X$ by using 2's complement.

Subtraction with (8-1)'s Complement:

Note: To do the subtraction using (8-1)'s complement,
 if the carry is generated, Add that carry to the least
 significant digit position of the result to get the final
 result.

Ex: Given binary numbers $X = 1010100$ & $Y = 1000011$

perform the subtraction using 1's complement.

a) $X - Y$ $X - Y = 1010100 - 1000011$

$$X = 1010100$$

1) 1's complement of $Y = 0111100$

Add with X

$$\begin{array}{r}
 \begin{array}{c} 1111 \\ 1010100 \\ 0010000 \\ \hline 0010001 \end{array} \\
 \text{End-around carry} \rightarrow 1 \\
 \hline
 \end{array}$$

$$X - Y = 0010001.$$

$$b) Y - X = 1000011 - 1010100$$

$$\begin{array}{r} \text{1's comple. of } X = 0101011 \\ \text{Add with } Y = 1000011 \\ \hline 1101110 \end{array}$$

No Carry, the result is -ve, find the 1's complement of result to get final result.

$$-(1101110) \Rightarrow \underline{\underline{-0010001}}$$

Practise problems:

1) perform the subtraction using 2's complement

$$a) 10011 - 10001 \quad b) 1001 - 101000$$

2) perform the subtraction using 10's complement

$$a) 7523 - 4567 \quad b) 230 - 1204$$

Non-Signed Binary numbers

Positive Integers (including zero) can be represented as unsigned numbers.

To represent negative integers, a minus sign is placed in the left most position.

In case of binary numbers, the sign bit is indicated as '0' for positive & '1' for negative.

(11)

If the binary number is signed, then the leftmost bit represent the sign & the rest of the bits represent the number.

If the binary number is assumed to be unsigned, then the leftmost bit is the most significant bit of the number.

Ex: For decimal '9', the binary is 01001

→ Here '9' is unsigned binary 01

+9 is signed binary because left most bit is '0'

→ The string of bits 11001 represent the binary equivalent of 25 considered as an unsigned number & binary equivalent of -9 when considered as a signed number. Because the '1' in the leftmost position designates a negative & the other four bits represents binary 9.

The signed binary numbers are represented in a format called signed magnitude form.

In a signed binary number, the left most bit (i.e., MSB) represents the sign of the number and all the remaining bits represents the magnitude of the number.

Some of the 8-bit signed binary numbers are

$$+6 = 0000\ 0110$$

$$-14 = 1000\ 1110$$

$$+29 = 0001\ 1100$$

$$-64 = 1100\ 0000$$

In unsigned binary number, all the bits represent the magnitude.

→ If the signed binary number is negative, it can be represented in three ways

i) Signed-magnitude form

ii) Signed-1's Complement form

iii) Signed-2's Complement form

→ If the signed binary number is positive, then the Signed magnitude form, Signed 1's Complement form and Signed 2's Complement form all are identical.

Representation of signed binary numbers using 2's Complement and 1's Complement:

i) If the signed binary number is positive, the sign bit '0' is placed. For such numbers, 1's complement and 2's complement are equal to signed magnitude form.

2) If the signed binary number is negative, then the magnitude is represented in 1's Complement

(a) 2's complement form, & then the sign bit '1' is placed in front of MSB.

Ex: Represent '-5' three ways with 8 bits:

a) signed-magnitude

b) Signed 1's Complement

c) signed 2's Complement.

Sol: Given number is '-5'

a) signed-magnitude with 8 bits

Signbit $\underline{1000\ 0101}$

b) signed 1's Complement

$\underline{1111\ 1010}$

c) signed 2's Complement

1's Complement + 1

$$\Rightarrow \begin{array}{r} 1111\ 1010 \\ + \quad \quad \quad 1 \\ \hline 1111\ 1011 \end{array}$$

$$\Rightarrow (1111\ 1011)_2$$

Practise problem

Represent +51 & -51 in signed magnitude form, 1's complement & 2's complement form.

Table: Signed Binary numbers.

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+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	-	1111	1000
-1	1111	1110	1001
-2	1110	1101	1010
-3	1101	1100	1011
-4	1100	1011	1100
-5	1011	1010	1101
-6	1010	1001	1110
-7	1001	1000	1111
-8	1000	-	-

Digital Logic Circuits - I

Digital logic circuits are electronic circuits that process information in the form of binary data (0s & 1s). These are the building blocks of digital systems & are designed to perform specific tasks, such as Arithmetic, logical operations, Data storage and retrieval, Control & Sequencing.

The components of digital logic circuits are Logic gates (eg. AND, OR, NOT), Flip-flops (eg. SR, D, JK, T), Counters, Registers.

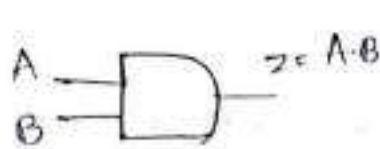
B A particular digital system may define logic 0 as a signal equal to 0V & logic 1 as a signal equal to 5V.

Logic Gates: Logic gates are electronic circuits that operate on one or more physical input signals to produce an output signal.

The Gates are blocks of hardware that produce the equivalent of logic-1 or logic-0 output signals if input logic requirements are satisfied.

There are three basic Logic gates: AND, OR and NOT Gates.

The graphic symbols used to designate these types of gates are shown in Fig.



Two input AND Gate



Two input OR Gate



NOT Gate or Inverter

AND Gate This logic gate performs the product of two or more inputs. This operation is represented by a dot or by the absence of an operator.

For eg., $A.B = z$ or $AB = z$ is read "A AND B is equal to z".

It means that output $z = 1$ if and only if $A = 1$ & $B = 1$. Otherwise output $z = 0$.

The result of the operation or basic logic function of AND Gate is $A.B = z$.

OR Gate; This logic gate performs the addition of two or more inputs. This operation is represented by a plus (+) sign.

For eg., $A+B = z$ is read as "A OR B is equal to z" which means output $z = 1$, if $A = 1$ or

if $B = 1$ or if both $A = 1$ & $B = 1$.

if both $A = 0$ & $B = 0$, then $z = 0$.

NOT Gate: This logic gate performs the inversion of the given input. This operation is represented by an overbar ($-$) / Prime ($'$)

For eg., $A' = Z$ ($\bar{A} = Z$) which read as "Not A is equal to Z"

If $A = 1$, then $Z = 0$, but if $A = 0$, then $Z = 1$.

The NOT Gate is also referred as complement operation since it changes a 1 to 0 and a 0 to 1.

Truth tables: It is a table of all possible combinations of the variables showing the relation between the inputs and outputs.

The truth tables of AND, OR & NOT are given in below.

AND			OR			NOT	
A	B	$A \cdot B$	A	B	$A + B$	A	A'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

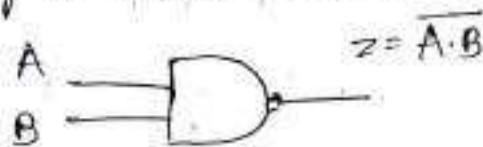
In binary arithmetic, $1+1$ is 10, whereas in binary logic it is $1+1$ is '1'.

Universal Logic Gates:

These gates are a set of logic gates that can be combined to perform any possible logical operation.

There are two universal logic gates.

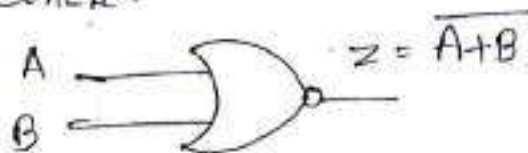
1. NAND Gate It is a combination of AND & NOT operations. It is the complement of the AND function as indicated by AND graphic symbol followed by a small circle.



Truth table

A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

2. NOR Gate It is a combination of OR & NOT operations. It is the complement of OR function as indicated by OR graphic symbol followed by a small circle.



Truth table

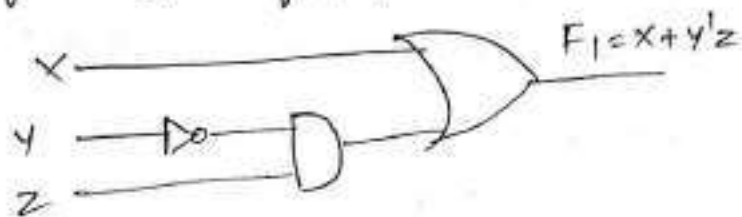
A	B	Z
0	0	1
0	1	0
1	0	0
1	1	0

The Boolean function can be represented in a truth-table. The number of rows in a truth-table is 2^n , where n is the number of variables in Boolean function.

Let truth table of F_1 is

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

Logic Diagram for $F_1 = X + Y'Z$ is



Simplification of Boolean functions:

When a boolean expression is implemented with logic gates, each term requires a gate & each variable within the term designates an input to that gate. A variable within a term, either in complemented form (0) or uncomplemented form is said to be a literal.

The techniques used for simplification of Boolean expression is given below

- 1) Multiply all variables necessary to remove parenthesis.

2) Look for identical terms. Only one of those terms is retained & all other terms are dropped.

$$\text{ex: } AB + AB + AB = AB$$

3) Look for a variable & its negation in the same term. This term can be dropped.

$$\text{ex: } ABcc' = AB(0) = 0$$

4) Look for pairs of terms that are identical except one variable is appearing extra in one of the two terms. Then the larger term can be dropped.

$$ABc' + ABC'd' = ABc'(1+0') = ABc'$$

5) Look for pairs of terms which have the same variables such that a variable in one of the terms is complemented while in the other term it is not, then such terms are combined into a single term by dropping that variable.

$$\begin{aligned} \text{Eg: } ABC'D + AB'C'D &= AC'D(B+B') \\ &= AC'D \end{aligned}$$

Eg: Reduce the expression $f = A[B + c'(AB + Ac')']$

$$\begin{aligned} \text{Sol: Given } f &= A[B + c'(AB + Ac')'] \\ &= A[B + c'(AB)' \cdot (Ac')'] \\ &= A[B + c'(A' + B')(A' + c)] \\ &= A[B + c'(A'A' + A'C + A'B' + B'C)] \\ &= A[B + A'c' + A'cc' + A'B'c' + B'cc'] \\ &= A[B + A'c'(1 + B') + 0] \\ &= AB + A \cdot A' B' c' \\ &= AB \end{aligned}$$

* Any Boolean function can be expressed as a sum of minterms (SOP) as the product of maxterms. Boolean function expressed as a sum of minterms is known as canonical SOP form or minterm canonical form or sum of minterms form.

* Boolean functions expressed as a product of maxterms is known as canonical POS form or maxterm canonical form or product of maxterms form.

Standard forms: In this, each term of a Boolean function may contain one, two or any number of literals.

There are two types of standard forms.

1) Sum of products form (SOP)

2) Product of sum form (POS)

The SOP is a form in which a Boolean function contains AND-terms called product terms, of one or more literals each. The sum denotes the ORing of these terms.

$$\text{Eg: } y' + xy + x'y'z$$

Here, three product terms y' , xy & $x'y'z$ of 1 literal, 2 literals & 3 literals respectively.

The POS is a form in which a Boolean function contains OR-terms called sum terms, of one or more

literals each. The product denotes the ANDing of these terms.

$$\text{Eg: } f_2 = x(y' + z)(x' + y + z' + w)$$

Here sum-terms are x , $(y' + z)$, $(x' + y + z' + w)$ of 1 literal, 2 literals & 4 literals respectively.

Conversion of sop form to canonical form:

To convert sop form to canonical form, steps below followed are:

- 1) Find the missing literal of each product term if any.
- 2) AND each product term that has missing literal/literals with term/terms formed by ORing the literal & its complement.
- 3) Expand the terms by applying distributive law & reorder the literals in the product terms.
- 4) Reduce the expression by omitting the repeated product terms if any.

Eg: Convert the function $f(A, B, C) = AB + BC + AC$ into canonical ^{sop} form & canonical pos form.

$$\begin{aligned}\text{Sol: Given } f(A, B, C) &= AB + BC + AC \\ &= AB(C + C') + (A + A')BC + AC(B + B') \\ &= ABC + ABC' + ABC + A'BC + ABC + AB'C\end{aligned}$$

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

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

$$= m_7 + m_6 + m_3 + m_5$$

$$\therefore \text{Canonical SOP form} = \sum m(3, 5, 6, 7)$$

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

$$\text{Canonical POS form} = \prod M(0, 1, 2, 4) =$$

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

Note: The Canonical POS form function form a Canonical SOP function can be written by writing the missing decimal numbers from the Canonical SOP form function & vice versa.

Canonical POS form from given POS form:

- 1) Find the missing literal in each sum term if any.
- 2) OR each sum term that has missing literal/literals with the term/terms formed by ANDing the literal & its complement.
- 3) Expand the terms by applying distributive law & reorder the literals in the sum terms.
- 4) Reduce the expression by omitting repeated sum terms if any.

Eg: Convert the pos function $F(A,B,c) = (A+B)(B+c)(A+c)$ to canonical pos form & Canonical sop form.

$$\text{Given } F(A,B,c) = (A+B)(B+c)(A+c)$$

$$\begin{aligned} \therefore A+BC &= (A+B)(A+c) \\ &= (A+B+cc')(AA'+B+c)(A+c+BB') \\ &= (A+B+c)(A+B+c')(B+c+A)(B+c+A') \\ &\quad (A+c+B)(A+c+B') \\ &= (A+B+c)(A+B+c')(A+B+c)(A'+B+c) \\ &\quad (A+B+c)(A+B'+c) \\ &= (A+B+c)(A+B+c')(A'+B+c)(A+B'+c) \\ &= M_0 \cdot M_1 \cdot M_4 \cdot M_2 \\ &= \Pi M(0, 1, 2, 4) \end{aligned}$$

$$\text{Canonical pos} = \Pi M(0, 1, 2, 4) = (A+B+c)(A+B+c')(A'+B+c)(A+B'+c)$$

$$\text{Canonical sop} = \Sigma m(3, 5, 6, 7) = A'Bc + AB'C + ABc' + ABC$$

Digital Logic Gates:

AND Gate: An AND Gate has two or more inputs but only one output. The output will be at logic 1 state only when each of its inputs is at logic 1 state. The output is at logic 0 state even if one of its inputs is at logic '0' state.

The logic symbol & truth table of a two input AND Gate is shown in figure.

Gate Level Minimisation

- We know that the Boolean functions can be realized using logic gates. The total number of logic gates and literals can be reduced, if the boolean function is simplified. The simplification of a boolean function, is required for reducing the complexity and cost of designing of its logic circuit.
- During the process of simplification using Boolean algebra, one must know the Boolean laws, rules, properties and theorems thoroughly. And also it is required to predict the successive steps to get the simplest expression.
- The map method gives us the systematic procedure to simplify the given Boolean function. It is also called as Karnaugh map method or K-map method.
- The map method was first proposed by Veitch & modified by Karnaugh, hence map method is also called as Veitch diagram or Karnaugh map.
- The map method (a) Karnaugh-map (b) K-map method.
- The K-map is a diagram made of square basis. Each square box is called as a cell that represents either a minterm or a max-term.
- The simplified function produced using K-map is present in any one of the standard forms i.e., either in product of sum (pos) form or in sum of products (sop) form.

- The simplified function should have less number of terms & each term should have minimum number of literals.
- A K-map contains 2^n cells for its n -variable boolean expression.

Ex: A 4-variable boolean expression contains $2^4 = 16$ cells.

2-variable K-map:

In a 2-variable K-map, there are $2^2 = 4$ cells. Each cell represents a minterm or a max-term.

A two variable K-map with minterm representation & maxterm representation are as shown below

	B	B'	B
A	A'B	A'B'	AB
A'	A'B	A'B'	AB

$A=0 \Rightarrow A'$
 $A=1 \Rightarrow A$

Fig: 2-variable K-map with minterm represent.

	B	B'	B
A	A'B	A'B'	AB
A'	A'B	A'B'	AB

$A=0 \Rightarrow A'$
 $A=1 \Rightarrow A$

Fig: 2-variable K-map with maxterm represent.

3-variable K-map:

A 3-variable K-map contains $2^3 = 8$ cells. Each cell represents a minterm or a maxterm.

Here the minterms or maxterms are arranged in Gray code - sequence but not in ordinary binary sequence.

The advantage of Gray code over normal binary sequence is that only one bit position is having a change

between its present value to previous value & between its present value to its next value. The 3-variable K-map with minterm & maxterm representations are given in below figures.

A \ BC	00	01	11	10
	$A'B'C'$	$A'B'C$	$A'BC$	$A'BC'$
A' 0	0	1	3	2
A 1	4	5	7	6

Fig: 3-variable K-map with minterm represent.

A \ BC	$B+C$	$B+C'$	$B'+C'$	$B'+C$
	$A+B+C$	$A+B+C'$	$A+B'+C'$	$A+B'+C$
A 0	0	1	3	2
A' 1	4	5	7	6

Fig: 3-variable K-map with maxterm represent

4-variable K-map:

A 4-variable K-map contains $2^4 = 16$ cells. Each cell contains either minterm & maxterm.

A 4-variable K-map containing minterm representation & maxterm representation are as shown below

AB \ CD	00	01	11	10
	$A'B'C'D'$	$A'B'C'D$	$A'BC'D'$	$A'BC'D$
A'B' 00	0	1	3	2
A'B 01	4	5	7	6
AB 11	12	13	15	14
AB' 10	8	9	11	10

Fig: 4-variable K-map with minterm representation

AB \ CD	$C+D$	$C+D'$	$C'+D'$	$C'+D$
	$A+B+C+D$	$A+B+C+D'$	$A+B+C'+D'$	$A+B+C'+D$
A+B 00	0	1	3	2
A+B' 01	4	5	7	6
A+B' 11	12	13	15	14
A+B 10	8	9	11	10

Fig: 4-variable K-map with maxterm represent

The following terms are defined with respect to K-map simplification.

Pair: A group of two adjacent minterms (or) maxterms is called as a pair. A pair eliminates one variable from the resultant term.

Quad: A group of four adjacent minterms (or) maxterms is called as a quad. A quad eliminates two variables from its resultant term.

Octet: A group of eight adjacent minterms (or) maxterms is called as an octet. An octet eliminates three variables from its resultant term.

Note: In a K-map, any two cells are said to be adjacent, if their binary equivalent values are having only a one bit position change.

eg: Though cell number '0' and cell number '2' are not looking like adjacent, they are adjacent.

Because, the binary equivalent for '0' & '2' are having a change only in one bit position.

0 = 0000, 2 = 0010, only 2nd LSB bit is getting changed

K-map

simplification

Minimal sop form : (a) simplified .Sop form:

To get the simplified expression in sop form, we have to follow the steps below

- 1) Plot the K-map & place 1's in the cells corresponding to the given minterms in the given Boolean expression.
- 2) Check for the 1's & encircle those 1's which are not adjacent to any other 1's. These are called as isolated 1's.
- 3) Check for the 1's which are adjacent to only one '1' and encircle them as a pair.
- 4) Check for the 1's that are having 4-adjacent 1's (quad) & 8 adjacent 1's (octet), even if some of the 1's among them are already encircled. While doing this make sure that there are less number of groups.
- 5) Form the simplified Boolean function by summing all the product terms of all groups.

Problems: simplify the following two variable Boolean functions in sop using K-map.

1) $f(x, y) = \sum m(0, 1, 3)$ 2) $f(A, B) = \sum m(1, 2, 3)$

3) $f(A, B) = \sum m(1, 3)$ 4) $f(x, y) = \sum m(0, 1, 2, 3)$

Sol: 1) Given Boolean function

$$f(x, y) = \sum m(0, 1, 3)$$

	y'	y
x'	0	1
x	1	1

$$\begin{aligned} & x'y' + x'y \\ &= x'(y' + y) \\ &= x' \cdot 1 \\ &= x' \end{aligned}$$

$$\begin{aligned} & x'y + xy \\ &= y(x' + x) \\ &= y \cdot 1 = y \end{aligned}$$

$\therefore$ Simplified SOP form is obtained by summing each term of each group
 $f(x, y) = x' + y$

2) Given Boolean function

$f(A, B) = \sum m(1, 2, 3)$

	B'	B
A'	0	1
A	1	1

$$\begin{aligned} & AB' + AB \\ &= A(B' + B) \\ &= A \cdot 1 \\ &= A \end{aligned}$$

$$\begin{aligned} & A'B + AB \\ &= B(A' + A) \\ &= B \cdot 1 \\ &= B \end{aligned}$$

$\therefore$ Simplified SOP form is $f(x, y) = A + B$

3) Given Boolean function

$f(A, B) = \sum m(1, 3)$

	B'	B
A'	0	1
A	1	1

$$\begin{aligned} & A'B + AB \\ &= B(A' + A) \\ &= B \cdot 1 \\ &= B \end{aligned}$$

$\therefore$ Simplified SOP form is $f(A, B) = B$

4) $f(x, y) = \sum m(0, 1, 2, 3)$

	y'	y
x'	1	1
x	1	1

In a K-map, if every cell covers a minterm for the given Boolean function, the function will be equal to unity, i.e., $f(x, y) = 1$.

Q) Simplify the following Boolean functions in sop form using k-map

1) $F(x, y, z) = \sum m(2, 3, 4, 5)$ 2) $F(x, y, z) = \sum m(0, 2, 4, 5)$

3) $F(A, B, C) = \sum m(0, 1, 2, 3, 4, 5, 6, 7)$

4) $F(x, y, z) = \sum m(1, 3, 5, 6, 7)$

Sol: 1) Given Boolean function $F(x, y, z) = \sum m(2, 3, 4, 5)$

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

$$\begin{aligned} & x'y'z + x'y'z' \\ & x'y(z+z') \\ & = x'y \end{aligned}$$

$$\begin{aligned} & xy'z' + xy'z \\ & = xy'(z+z') = xy' \end{aligned}$$

$$\therefore F(x, y, z) = x'y + xy'$$

2) Given Boolean function

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

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

$$\begin{aligned} & x'y'z' + x'y'z \\ & x'y'z'(y+y') \\ & = x'y'z' \end{aligned}$$

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

$$\therefore F(x, y, z) = x'y'z' + xy'$$

3) Given

$$F(A, B, C) = \sum m(0, 1, 2, 3, 4, 5, 6, 7)$$

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

$$\begin{aligned} & \Sigma = 1 \\ & F(A, B, C) = 1 \end{aligned}$$

Irrespective of number of variables, if a k-map contains all 1's, the simplified function value is unity.

4) Given $F(x, y, z) = \sum m(1, 3, 5, 6, 7)$

	yz	$y'z'$	$y'z$	yz'
x	00	01	11	10
x' 0	0	1	1	3
x 1	4	5	7	6

$$\begin{aligned} & \text{II} \quad xy'z + xy'z' \\ & = xy(z + z') \\ & = xy \end{aligned}$$

$$\begin{aligned} & \text{I} \quad x'y'z + x'y'z' + x'y'z + x'y'z' \\ & = x'z(y + y') + x'z(y + y') \\ & = x'z + x'z = z(x + x') = z \end{aligned}$$

$$\therefore F(x, y, z) = xy + z$$

Q) Simplify the following Boolean functions into SOP form using K-map.

1) $F(x, y, z) = \sum m(0, 2, 4, 7)$ 2) $F(x, y, z) = \sum m(0, 2, 3, 6, 7)$

3) $F(A, B, C) = \prod M(0, 2, 4, 7)$ 4) $F(x, y, z) = \prod M(3, 5)$

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

6) $F(A, B, C) = \sum m(1, 2, 4, 5, 6, 7)$

sol: 1) Given $F(x, y, z) = \sum m(0, 2, 4, 7)$

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

$$\begin{aligned} & \text{II} \quad x'y'z' + x'y'z' \\ & = x'z'(y + y') = x'z' \end{aligned}$$

$$\begin{aligned} & \text{III} \quad x'y'z' + x'y'z' \\ & = y'z'(x + x') = y'z' \end{aligned}$$

$$\text{I} \quad xyz$$

$$\therefore F(x, y, z) = xyz + x'z' + y'z'$$

2) Given $F(x, y, z) = \sum m(0, 2, 3, 6, 7)$

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

$$\begin{aligned} & x'y'z' + x'y'z + x'y'z' \\ & = x'y'z'(y+y') \\ & = x'y'z' \end{aligned}$$

$$\begin{aligned} & y = x'y'z + x'y'z' + x'yz + x'yz' \\ & = x'y(z+z') + x'y(z+z') \\ & = (x'+x)y = y \end{aligned}$$

$$\therefore F(x, y, z) = x'y'z' + y$$

3) $F(A, B, C) = \prod M(0, 2, 4, 7)$

Given Boolean function is in canonical maxterm form,

Convert into canonical minterm

$$\therefore F(A, B, C) = \sum m(1, 3, 5, 6)$$

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

$$\begin{aligned} & A'B'C + A'BC \\ & = A'C(B+B') = A'C \end{aligned}$$

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

$$\begin{aligned} & A'B'C + A'BC \\ & B'C(A+A') = B'C \end{aligned}$$

$$\therefore F(A, B, C) = ABC' + B'C + A'C$$

4) Given $F(x, y, z) = \prod M(3, 5)$

in canonical minterm form $= \sum m(0, 1, 2, 4, 6, 7)$

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

$$\begin{aligned} & x'y'z' + x'y'z + x'y'z' + x'y'z' \\ & = y'z' + yz' = z' \end{aligned}$$

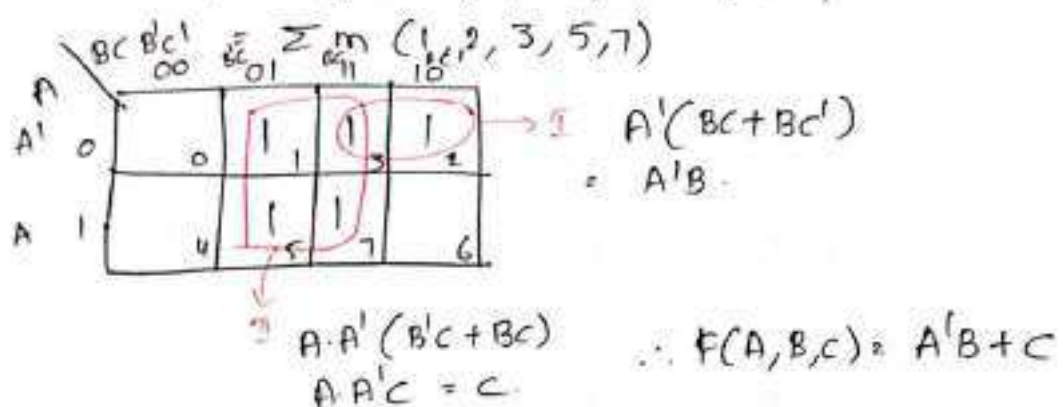
$$\begin{aligned} & x'yz + x'yz' = x'y \\ & x'y'z' + x'y'z = x'y' \end{aligned}$$

$$\therefore F(x, y, z) = x'y' + x'y + z'$$

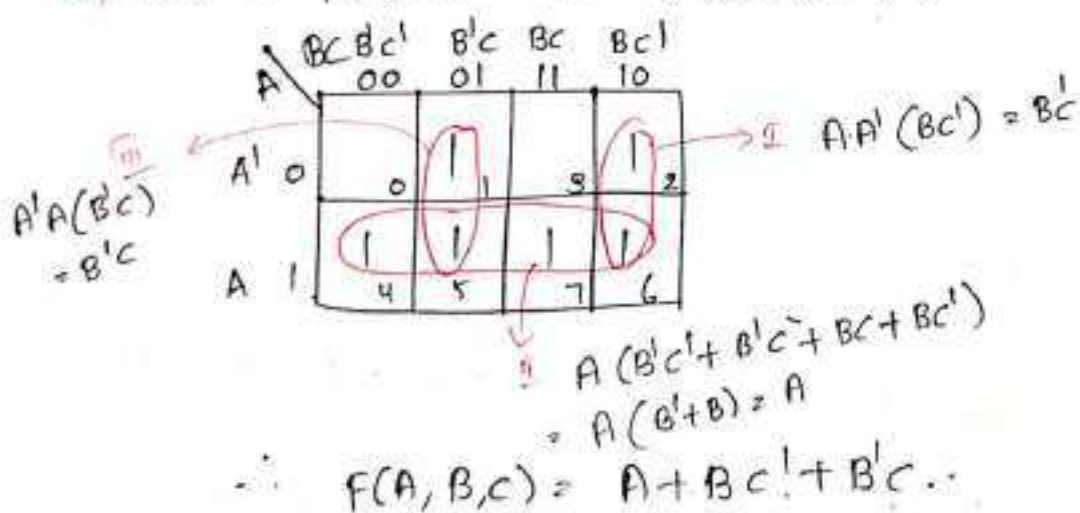
5) Given $F(A, B, C) = A'B + A'C + B'C + ABC$

Given function is not in Canonical form, Convert into minterm Canonical form.

$$\begin{aligned}
 &= A'B(C+C') + A'(B+B')C + (A+A')B'C + ABC \\
 &= A'BC + A'BC' + A'BC + A'B'C + AB'C + A'B'C + ABC \\
 &= A'BC + A'B'C + ABC + A'BC' + AB'C \\
 &= m_3 + m_2 + m_1 + m_5 + m_7
 \end{aligned}$$



6) Given $f(A, B, C) = \Sigma m(1, 2, 4, 5, 6, 7)$



Practice Problem:

Simplify Boolean function into SOP using k-map

- 1) $F(A, B, C) = AB + A'C + ABC$
- 2) $F(x, y, z) = \Sigma m(0, 2, 3, 4, 6, 7)$
- 3) $F(A, B, C) = \Pi M(2, 3, 4, 5)$
- 4) $F(A, B, C) = \Sigma m(0, 2, 4, 5, 7)$

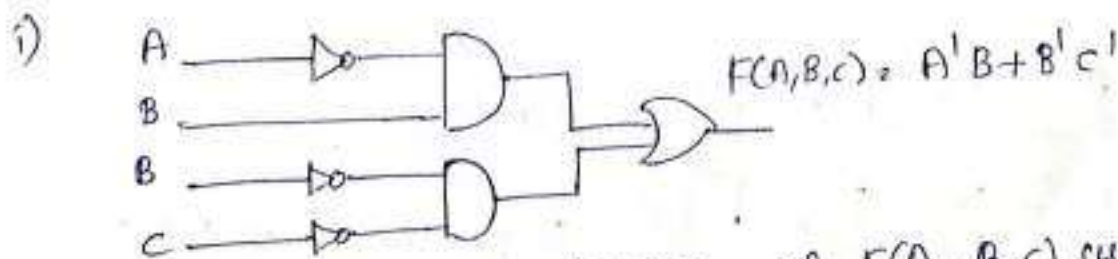
NAND & NOR Implementation of NAND & NOR Realization:

- 1) Draw the Logic Diagram using basic logic gates (AND, OR, NOT)
- 2) If NAND Logic is being implemented, add bubbles to the output of the AND gates and to the inputs of OR gates.
- 3) If NOR logic is being implemented, add bubbles to the output of OR gates & to the inputs of the AND gates.
- 4) Add an inverter (NOT Gate) to each line that receives a bubble in step 2 or step 3.
- 5) Eliminate double inversions & replace bubbled AND by NOR, bubbled OR by NAND.
- 6) Replace single inverter with NAND inverter (in NAND realiz.) / NOR inverter (in NOR realiz.)

Eg: Implement $F(A, B, C) = A'B + B'C'$ using NAND, NOR logic gates.

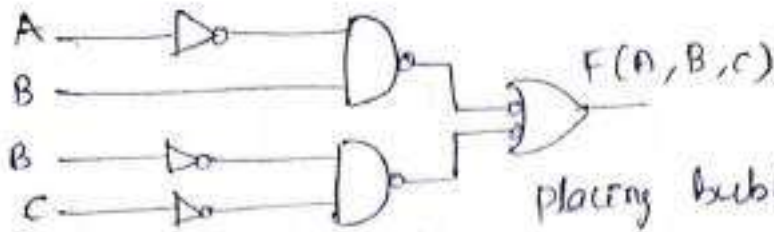
Sol: Given $F(A, B, C) = A'B + B'C'$

NAND Realization:



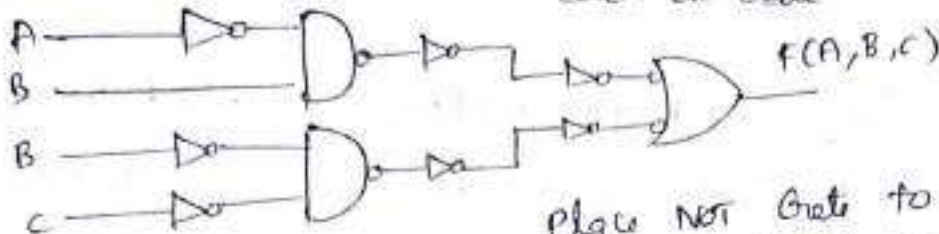
Logic diagram of $F(A, B, C)$ using basic logic gates.

ii)



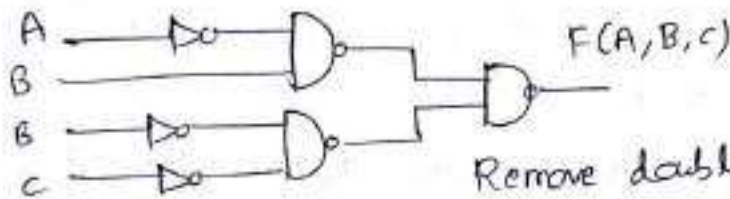
placing bubbles to the o/p of each AND gate & to all i/p's of each OR Gate

iii)



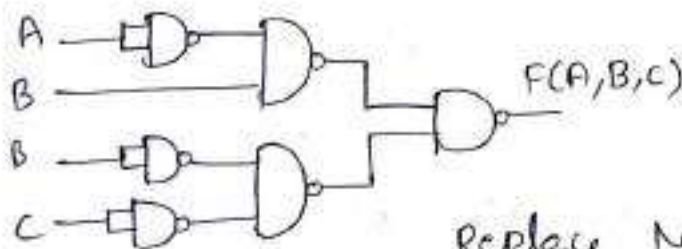
place NOT Gate to each line that receives a bubble, one NOT Gate for each bubble

iv)



Remove double inversions & replace doubled OR with NAND

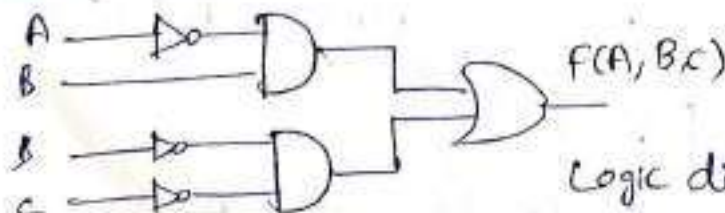
v)



Replace NOT Gates with NAND inverters

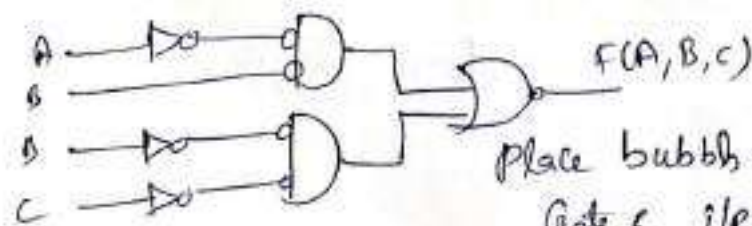
NOR Realization

i)

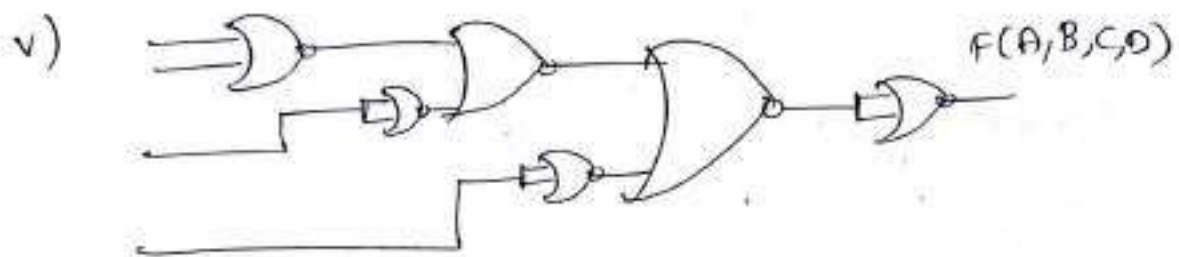
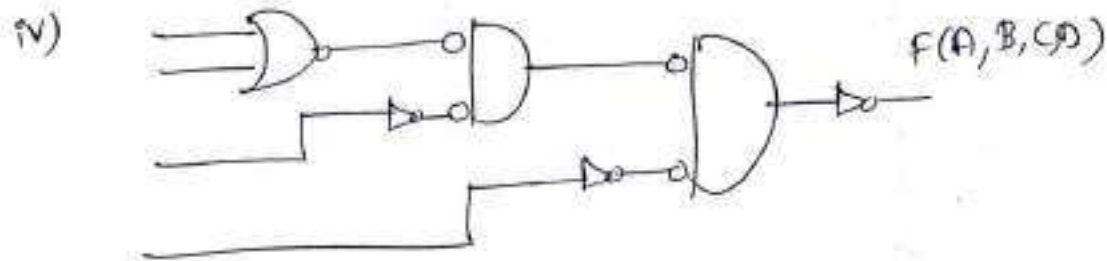
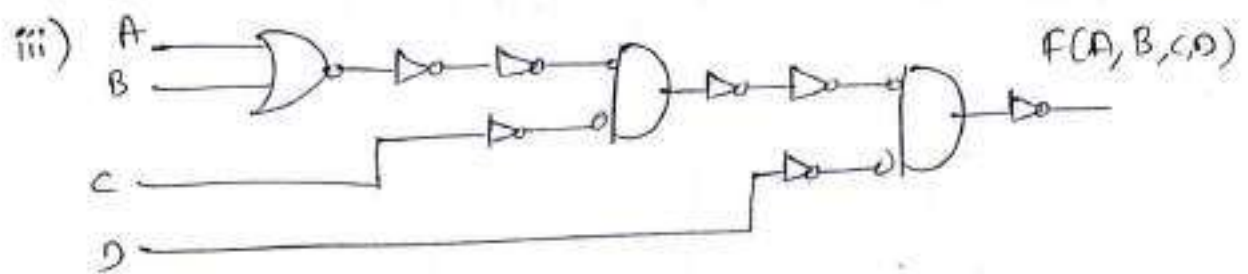


Logic diagram of $F(A,B,C)$

ii)

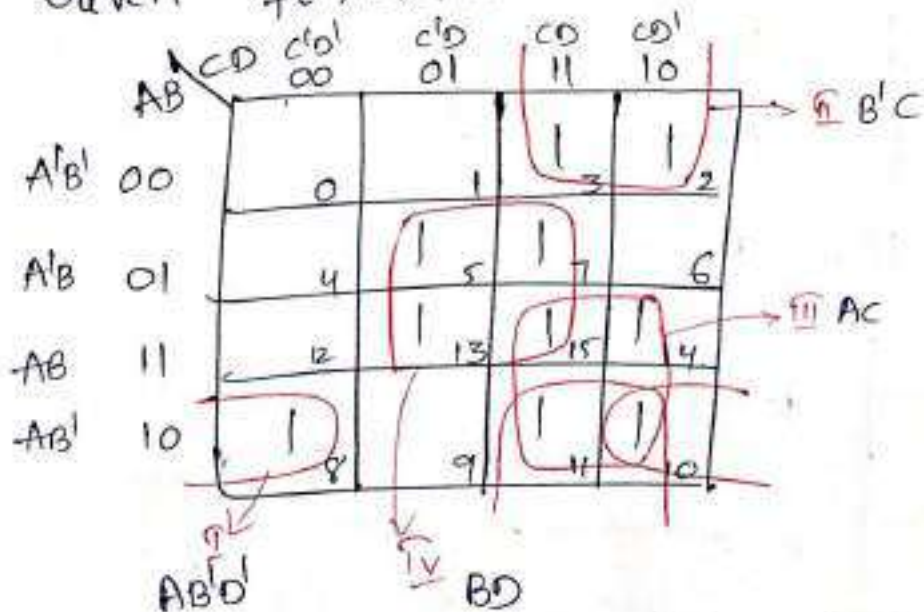


place bubble to o/p of each OR Gate & i/p's of AND Gate



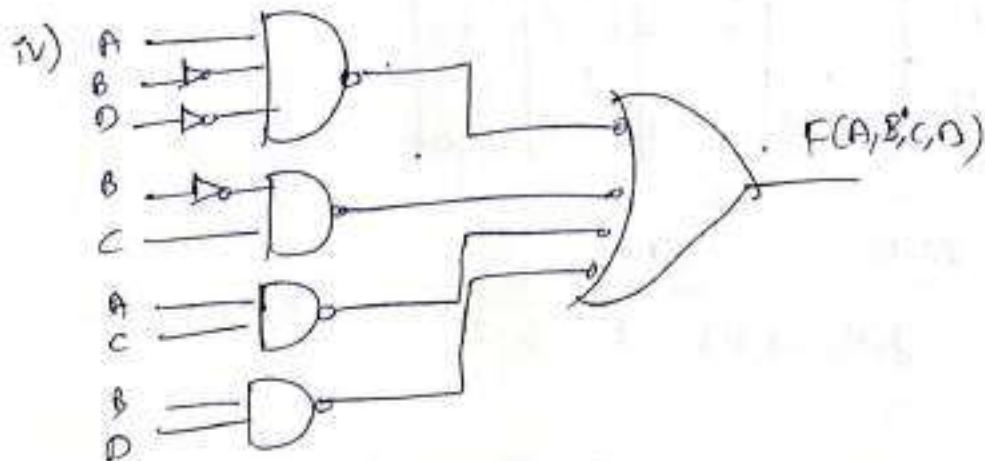
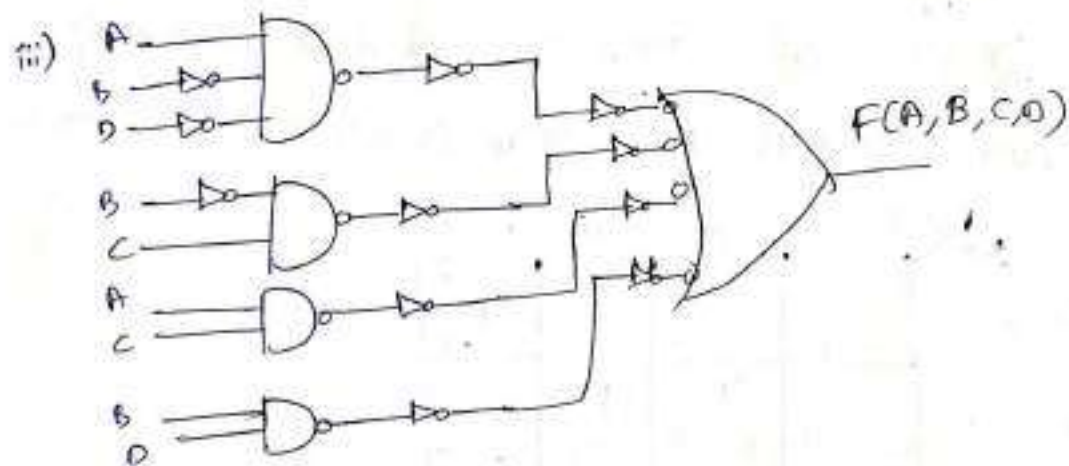
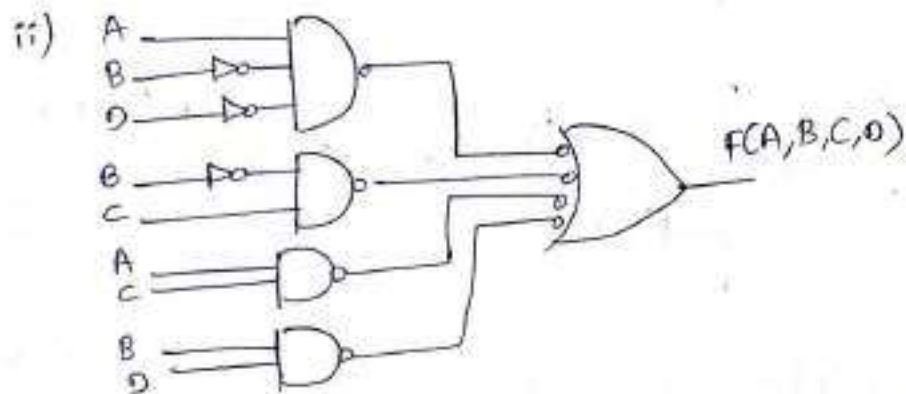
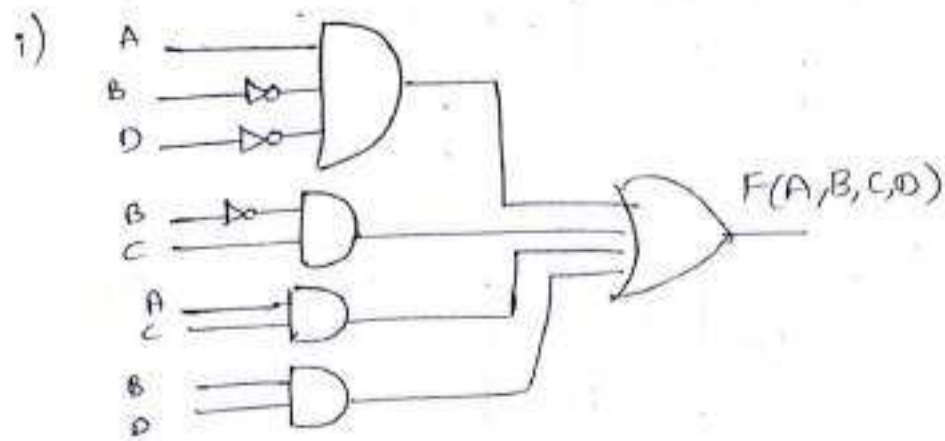
Q) Simplify $f(A, B, C, D) = \sum m(2, 3, 5, 7, 8, 10, 11, 13, 14, 15)$ in SOP form using K-map & implement with NAND Gates.

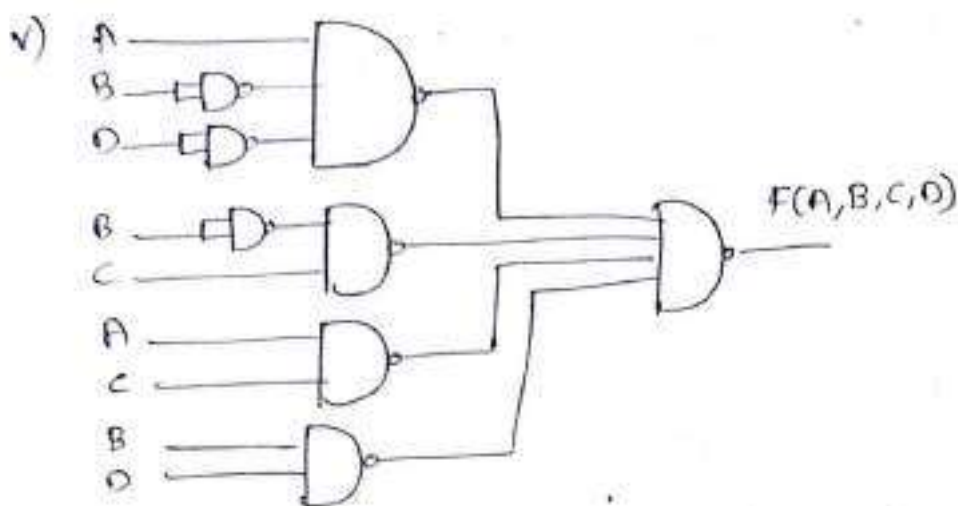
Given $f(A, B, C, D) = \sum m(2, 3, 5, 7, 8, 10, 11, 13, 14, 15)$



$$\therefore f(A, B, C, D) = AB'D + B'C + AC + BD$$

Implementation using NAND Gates





Practise problem:

Implement NAND logic for the simplified SOP form

- i) $F(A, B, C, D) = \prod M(5, 7, 11, 12, 13, 14, 15)$. Use K-map method
- ii) $Y(W, X, Y, Z) = \prod M(0, 1, 3, 5, 6, 7, 10, 14, 15)$ using K-map

Minimal pos form or simplified pos form:

- 1) Plot the K-map & place 0's in the place of maxterms that are given in the given Boolean expression.
- 2) Check for the 0's and encircle those 0's that are not adjacent to any other 0's. These are called as isolated 0's.
- 3) Check for the 0's which are adjacent to only one 0's & encircle them as a pair.
- 4) Check for quads (4 adjacent 0's) & octets (8 adjacent 0's) even if some of the 0's among them are already encircled. While doing this, make sure that there are less number of groups.

2) Form the simplified Boolean exp. by making the product of all sum-terms of all groups.

Eg. $F(A, B, C, D) = \prod M(0, 1, 4, 6, 9, 12)$, simplify this function into pos form using K-map & implement using NOR gates.

Sol: Given $F(A, B, C, D) = \prod M(0, 1, 4, 6, 9, 12)$

	$C'D$	$C'D'$	$C'D$	$C'D'$	$C'D$
AB					
$A+B$ 00	0	1		3	2
$A+B'$ 01	4		5	7	6
$A'+B'$ 11	12		13	15	14
$A'+B$ 10	8	9	11	10	

$$I \rightarrow (A+B)(C+D+C'D')$$

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

$$A+B' + A+B + C+D$$

$$B+C+D$$

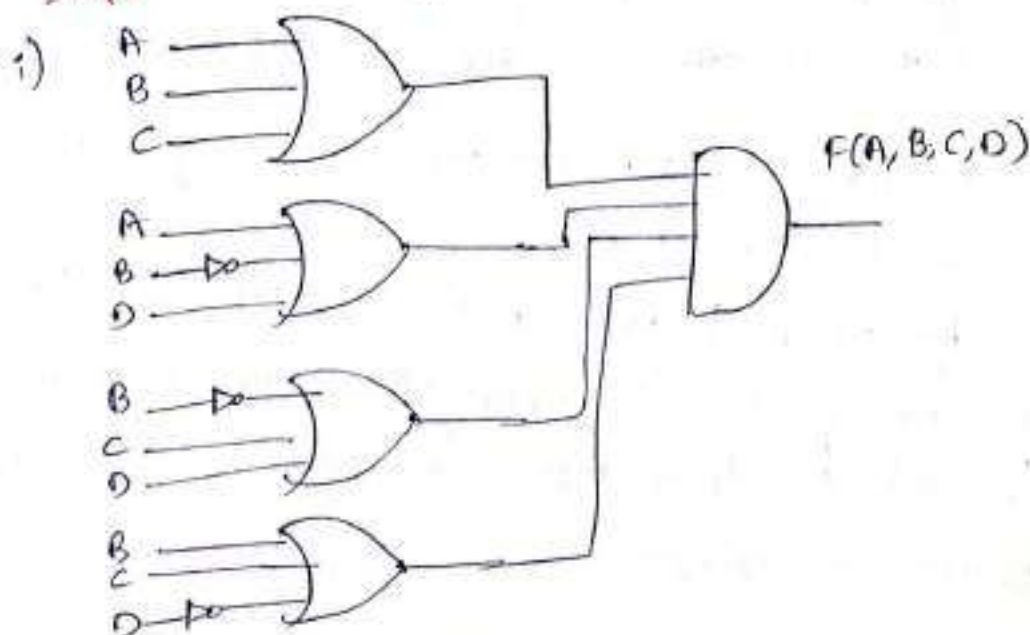
$$II \rightarrow A+B'+D$$

$$III \rightarrow B'+C+D$$

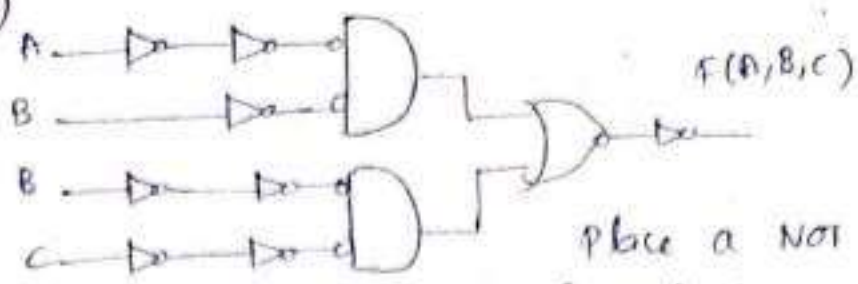
$$IV \rightarrow B+C+D'$$

Simplified Boolean expression in pos form
 $(A+B+C)(A+B'+D)(B'+C+D)(B+C+D')$

Implementation using NOR Gates

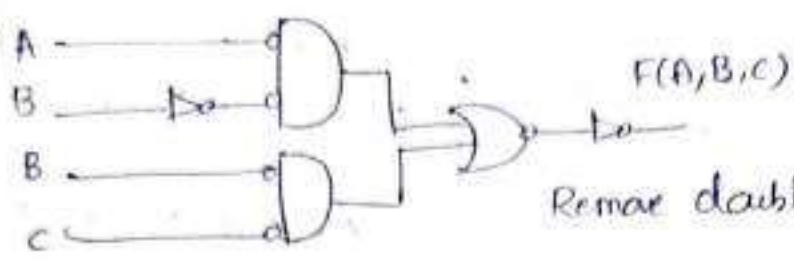


iii)



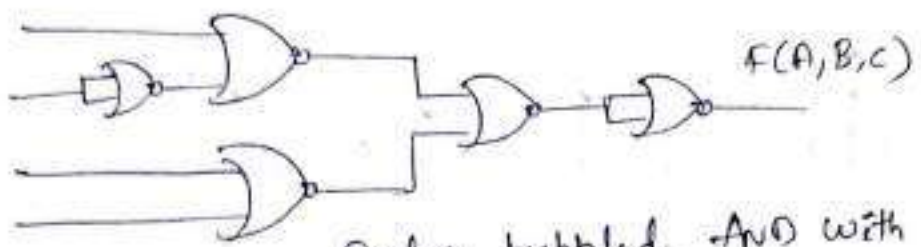
Place a NOT Gate to each line that receives a bubble

iv)



Remove double inversion

v)

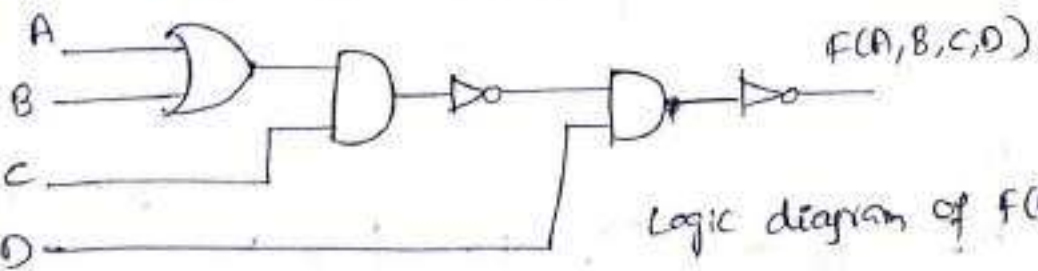


Replace bubbled AND with NOR Gate

2) $F = \overline{(A+B)C \cdot D}$ implement using NAND, NOR logic gates.

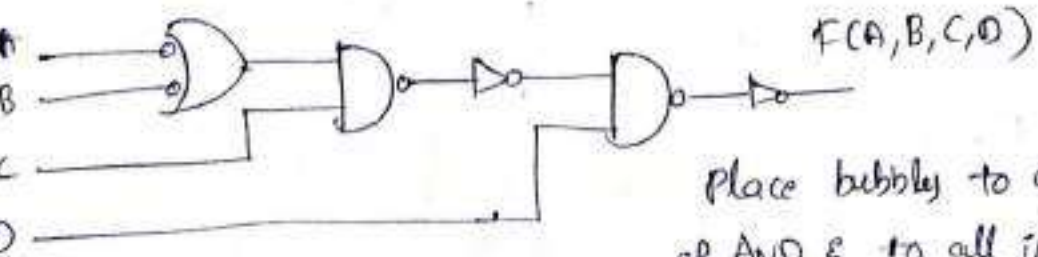
sol: NAND Logic:

i)

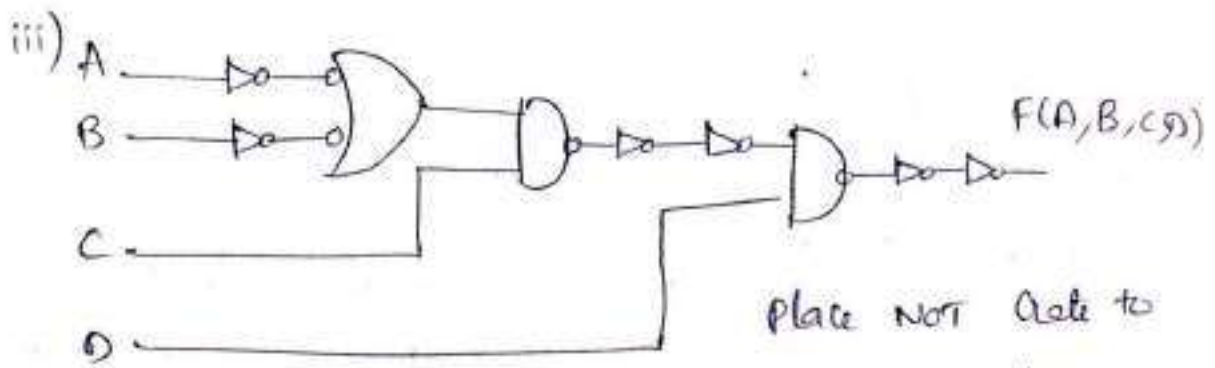


Logic diagram of $F(A,B,C,D)$

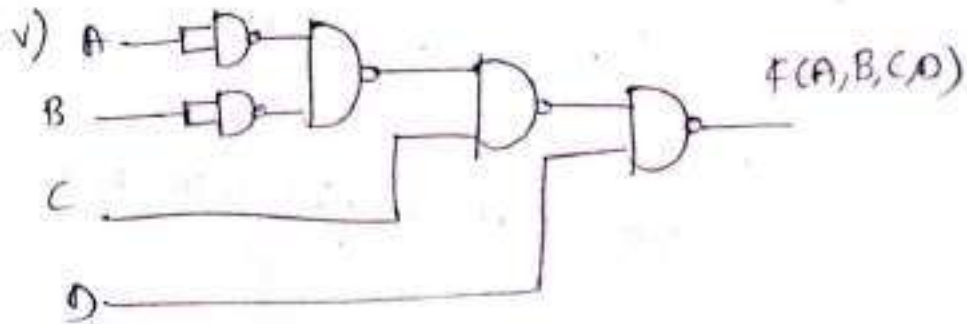
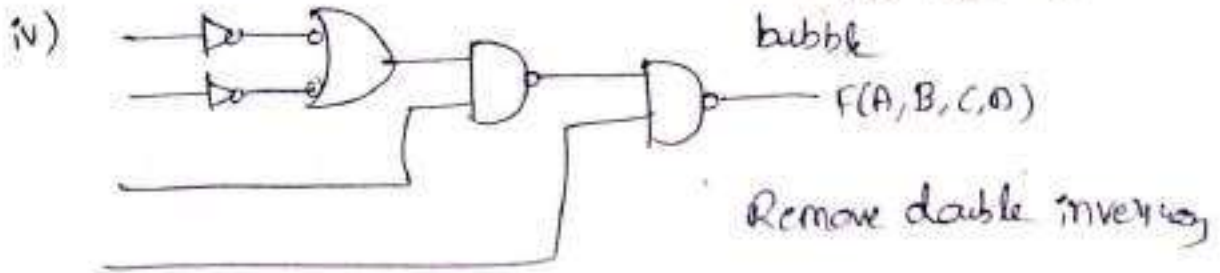
ii)



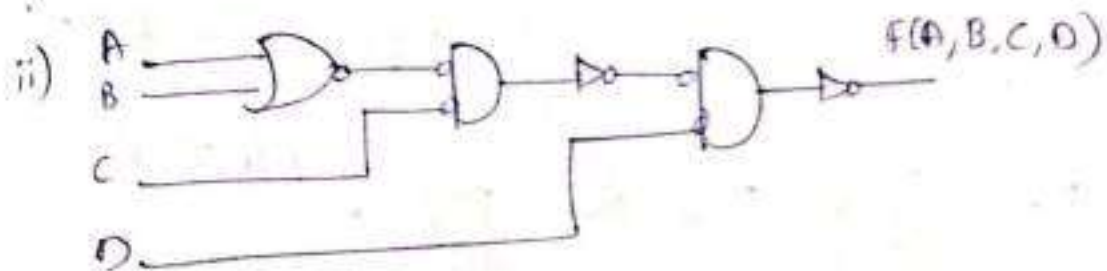
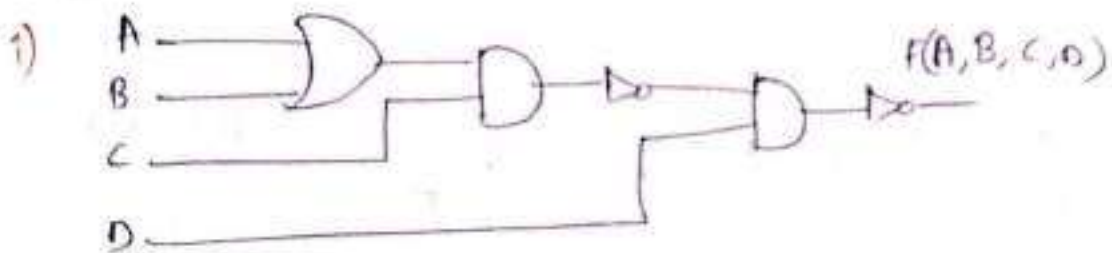
Place bubble to o/p of AND & to all i/p's of OR Gate



Place NOT gate to
each line that has
bubble



NOR Logic



Q) Simplify the function in pos form using k-map

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

using k-map.

Sol: Given

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

It is not in minterm canonical form, so convert to minterm canonical form.

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

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

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

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

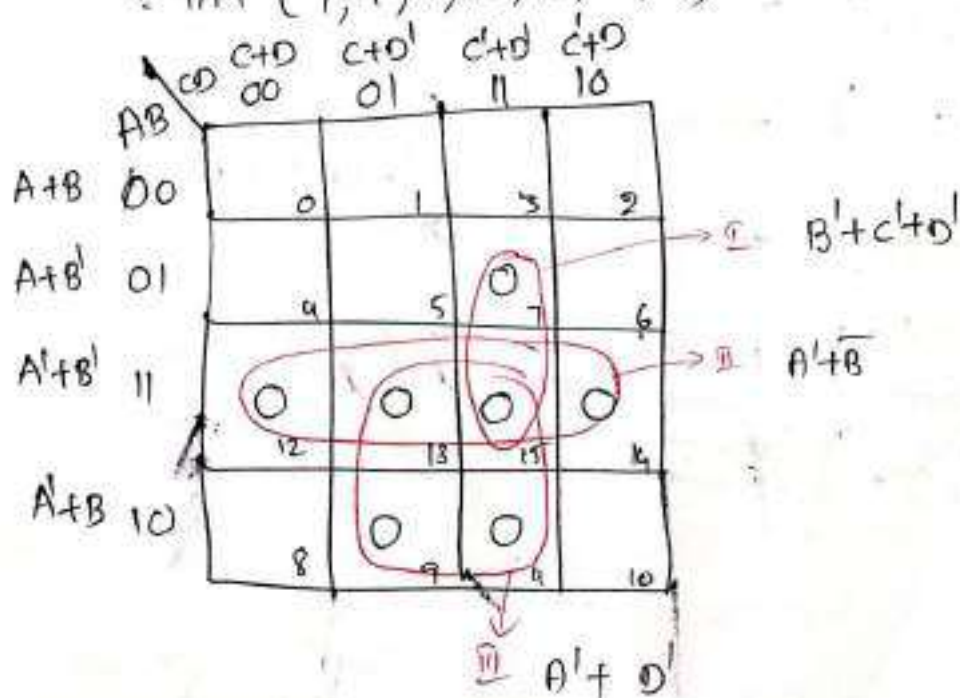
$$= (\bar{A} + B + C + \bar{D})(\bar{A} + B + \bar{C} + \bar{D})(\bar{A} + \bar{B} + \bar{C} + D)(\bar{A} + \bar{B} + \bar{C} + \bar{D})$$

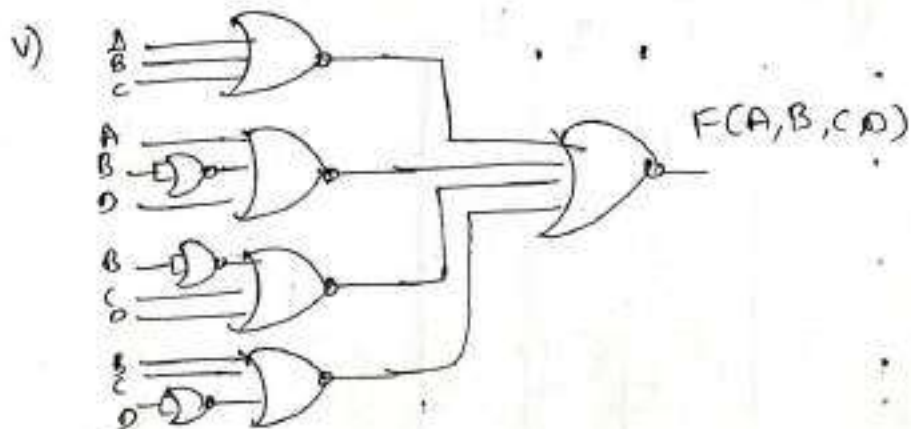
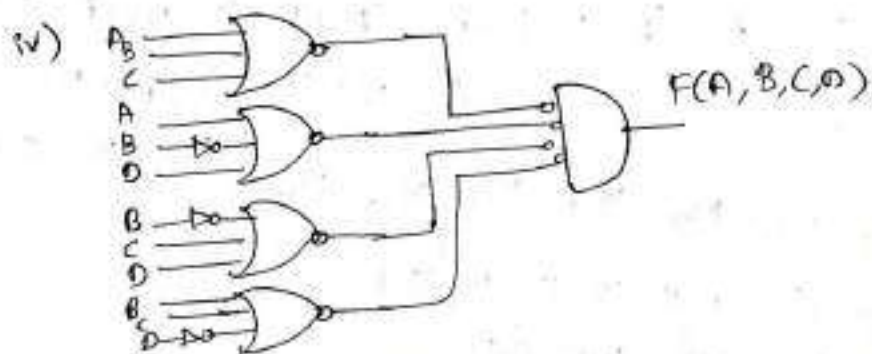
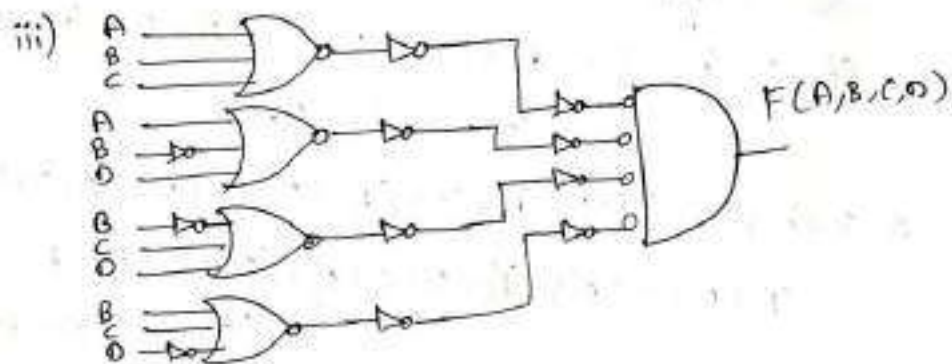
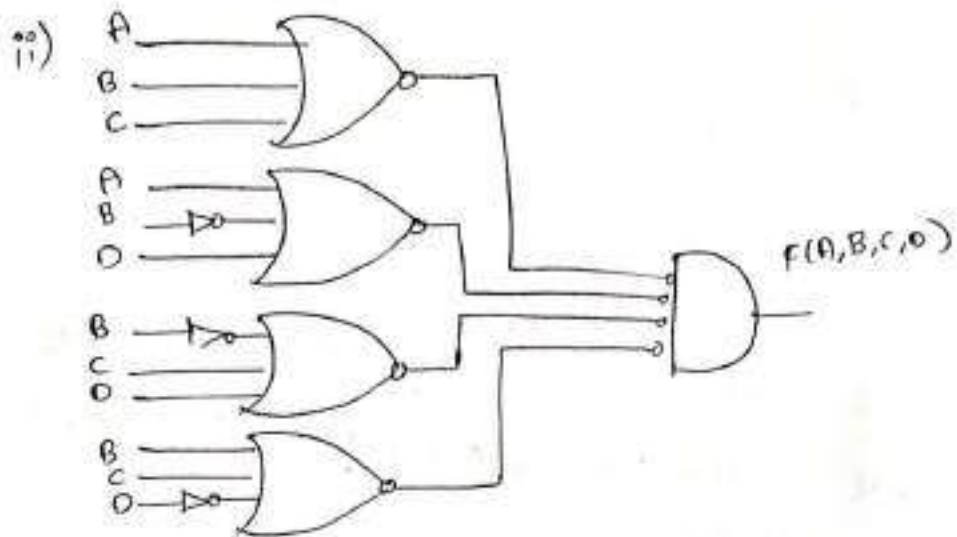
$$(A + \bar{B} + \bar{C} + \bar{D})(A + \bar{B} + C + \bar{D})$$

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

$$= M_9 \cdot M_{11} \cdot M_{14} \cdot M_{15} \cdot M_7 \cdot M_{12} \cdot M_{13}$$

$$= \Pi M(7, 9, 11, 12, 13, 14, 15)$$





Q) Simplify the following Boolean functions into sop using K-map

1) $f(A, B, C, D) = \sum m(2, 4, 5, 9, 11, 12, 13, 14)$ 2) $F(A, B, C, D) = \sum m(0, 1, 3, 7, 11, 13, 14, 15)$

3) $F(W, X, Y, Z) = \sum m(2, 3, 6, 10, 12, 13, 14, 15)$

4) $F(A, B, C, D) = \sum m(3, 4, 5, 7, 9, 13, 14, 15)$

5) $F = \sum m(1, 2, 4, 6, 7, 11, 12, 14)$ 6) $F(W, X, Y, Z) = \sum m(0, 1, 3, 7, 11, 13, 14, 15)$

1) Sol: Given $F(A, B, C, D) = \sum m(2, 4, 5, 9, 11, 12, 13, 14)$

AB \ CD	00	01	11	10
A'B'	0	1	3	2
A'B	1	1	5	6
AB	1	1	13	14
AB'	8	1	9	10

$A'B'CD'$ (m2)
 ABD' (m14)
 $AB'D$ (m9)
 BC' (m4, m5, m12, m13)

$$\therefore F(A, B, C, D) = A'B'CD' + ABD' + AB'D + BC'$$

2) Given $F(A, B, C, D) = \sum m(0, 1, 3, 7, 11, 13, 14, 15)$

AB \ CD	00	01	11	10
A'B'	1	1	1	2
A'B	4	5	1	6
AB	12	1	1	14
AB'	8	9	1	10

$A'B'C'$ (m0, m1, m3)
 ABD (m7, m11, m13, m14)
 ABD' (m9, m10)
 CD (m12, m13)

$$\therefore A'B'C' + ABD + ABD' + CD$$

Note: Sometimes, the given Boolean function may not have variables. In that case, the desired alphabets can be taken as variables, but the number of variables is to be decided based on highest number term decimal equivalent.

3) Given $F(w,x,y,z)$

$$= \sum m(2,3,6,10,12,13,14,15)$$

$w \backslash x \backslash yz$	00	01	11	10
$w'x'$ 00	0	1	3	2
wx 01	4	5	7	6
wx 11	12	13	15	14
wx' 10	8	9	11	10

Groupings:
 - Group I: $w'x'y$ (cells 1, 3, 5, 7)
 - Group II: $w'x'z$ (cells 2, 3, 6, 7)
 - Group III: wx (cells 12, 13, 14, 15)
 - Group IV: wx' (cells 8, 9, 10, 11)

$$\therefore = wx + w'x'y + yz'$$

6) Given $F(w,x,y,z)$

$$= \sum m(2,3,7,9,12,13,14,15)$$

$w \backslash x \backslash yz$	00	01	11	10
$w'x'$ 00	0	1	3	2
wx 01	4	5	7	6
wx 11	12	13	15	14
wx' 10	8	9	11	10

Groupings:
 - Group I: $w'x'y$ (cells 1, 3, 5, 7)
 - Group II: $w'x'z$ (cells 2, 3, 6, 7)
 - Group III: wx (cells 12, 13, 14, 15)
 - Group IV: $wy'z$ (cells 2, 3, 10, 11)

$$\therefore = wy'z + wx + xyz + w'x'y$$

2) Ans: $A'B'C'D + AB'CD + A'CD' + A'BC + BD'$

4) Ans: $A'CD + A'BC' + ABC + AC'D$

practise problems: simplify into SOP using K-map

1) $Y(A,B,C,D) = A'B'C'D' + A'BC'D' + A'BC'D + ABC'D' + ABC'D + AB'C'D$

2) $F(w,x,y,z) = \sum m(0,2,3,7,8,9,10,11)$

Prime Implicants & Essential Prime Implicants:

A prime implicant is a product term that is obtained by combining maximum number of possible adjacent minterms as a group in the K-map.

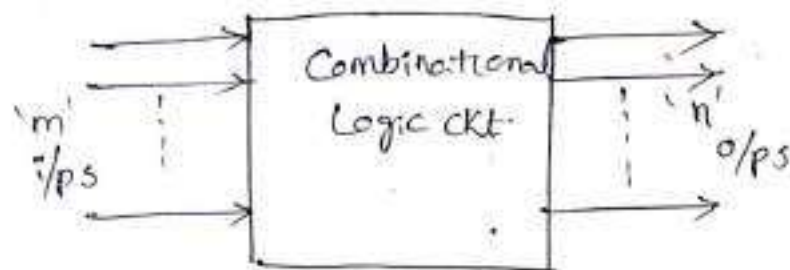
If a minterm is covered only one time in a prime implicant, then that prime implicant is said to be as essential prime implicant.

Combinational circuit

A Combinational circuit is the type of digital logic in which output depends only on the present input combination.

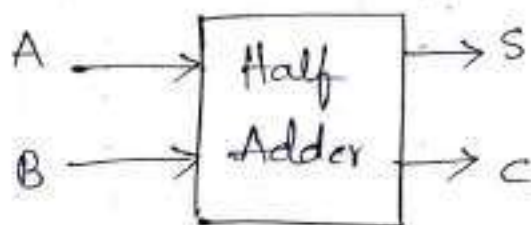
The logic gates are the building blocks for designing a combinational circuit. These circuit may have multiple outputs or single output.

Fig. shows the block representation of a combinational circuit



The various types of combinational logic blocks that performs addition such as half adder, Full adder.

Half Adder A Half Adder is a type of arithmetic logic circuit that adds two binary bits. It produce the results sum (s) & Carry (c) at the output.



Block diagram of Half Adder,

Truth table

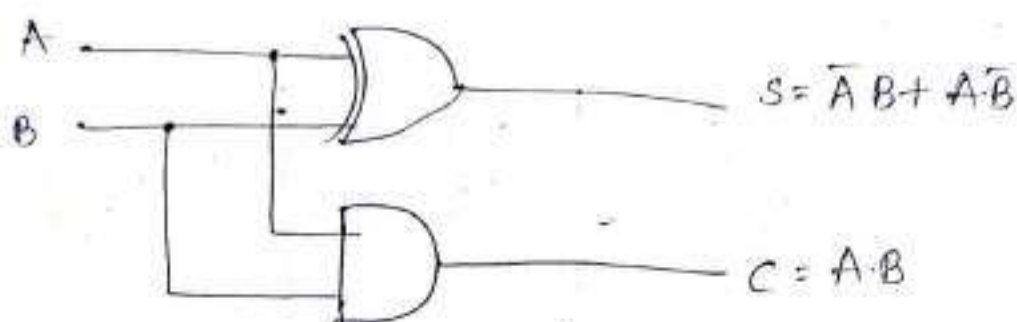
Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The logic expressions of sum (s), carry (c) are

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

$$C = A \cdot B$$

The logic gate implementation of half adder is the combination of a 'XOR' gate for sum & an 'AND' gate for carry outputs.



Full Adder

A Full Adder is a type of arithmetic circuit that adds three binary bits. It produces results sum (s) & carry (Cout) at the output.



Truth table

Inputs			Outputs	
A	B	C _{in}	C _{out}	S
0	0	0	0	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	1	1

The Sum (S), Carry (C_{out}) are given below.

$$S = \sum (m_1, m_2, m_4, m_7)$$

$$C_{out} = \sum (m_3, m_5, m_6, m_7)$$

The simplified expressions are

$$S = C_{in} \oplus A \oplus B$$

$$C_{out} = C_{in} \cdot (A+B) + A \cdot B$$

Encoders

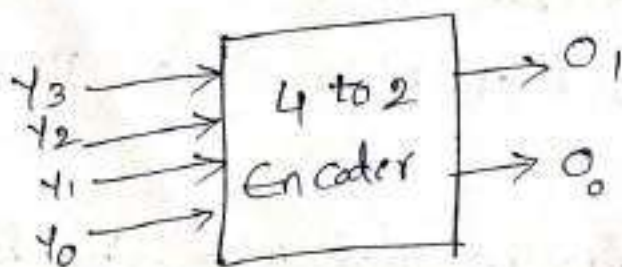
An encoder is a type of combinational circuit that produces the binary equivalent to the input.

It has 2^n input lines & defines n-bit code for the binary information at the output.

It can be represented as $2^n:n$ encoder.

Only one input is activated at a time.

The block diagram of 4:2 encoder is shown



Truth table

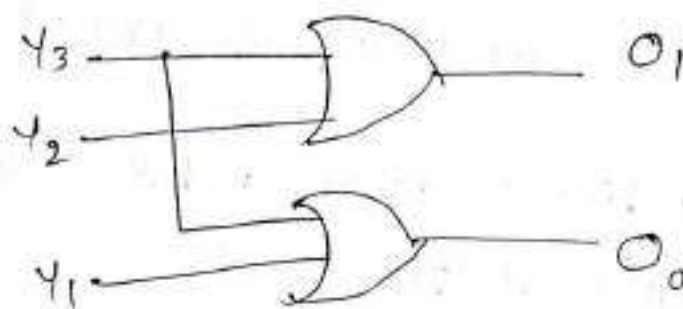
Y_3	Y_2	Y_1	Y_0	O_1	O_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

The Boolean expression for encoder is

$$O_1 = Y_2 + Y_3$$

$$O_0 = Y_1 + Y_3$$

Logic circuit



Decoders

A decoder is a type of Combinational logic circuit that decodes the binary information on 'n' lines to 2^n output lines.

It can be represented as n to 2^n or $n:2^n$ decoder.

A decoder can also be used with enable (E) signal. When enable is HIGH, then it provides

- the outputs according to binary input information.
- The block diagram and truth table of 2:4 decoder is shown in fig.

decoder is shown in fig.

```

    graph LR
      I1 --> Decoder
      I0 --> Decoder
      EN --> Decoder
      subgraph Decoder [2 to 4 Decoder]
        direction TB
        Y3
        Y2
        Y1
        Y0
      end
  
```

Inputs	
I_1	I_0
0	0
0	1
1	0
1	1

Outputs			
Y_3	Y_2	Y_1	Y_0
0	0	0	0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

The Boolean expressions for the outputs are

$$Y_3 = E \cdot I_1 \cdot I_0$$

$$Y_2 = E \cdot I_1 \cdot \bar{I}_0$$

$$Y_1 = E \cdot \bar{I}_1 \cdot I_0$$

$$Y_0 = E \cdot \bar{I}_1 \cdot \bar{I}_0$$

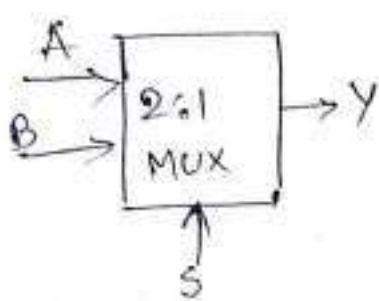
Multiplexers

The term 'Multiplexer' means 'many to one'.

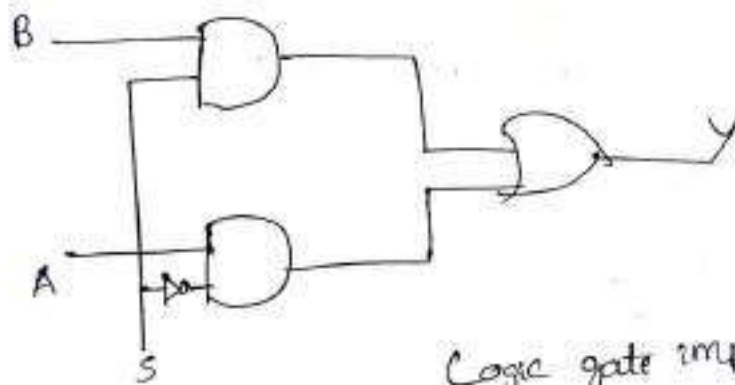
It is a type of combinational circuit which has multiple input lines and one output line. It is also known as data selector or MUX. It is represented as $2^n:1$ multiplexer, where 'n' is the number of select lines, 2^n is the number of input lines and '1' o/p line.

The types of multiplexers are 2:1 MUX, 4:1 MUX, 8:1 MUX and so on.

The block diagram of 2:1 Multiplexer

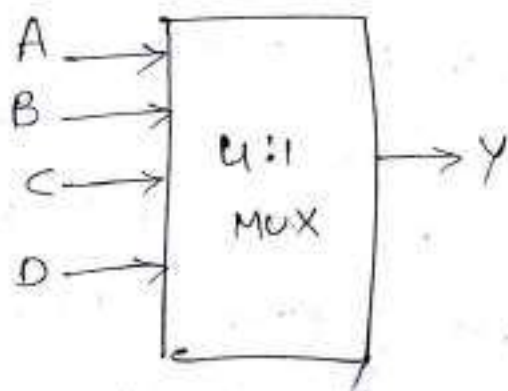


S	Y
0	A
1	B



Logic gate implementation

4 to 1 Multiplexer



S_1	S_0	Y
0	0	A
0	1	B
1	0	C
1	1	D

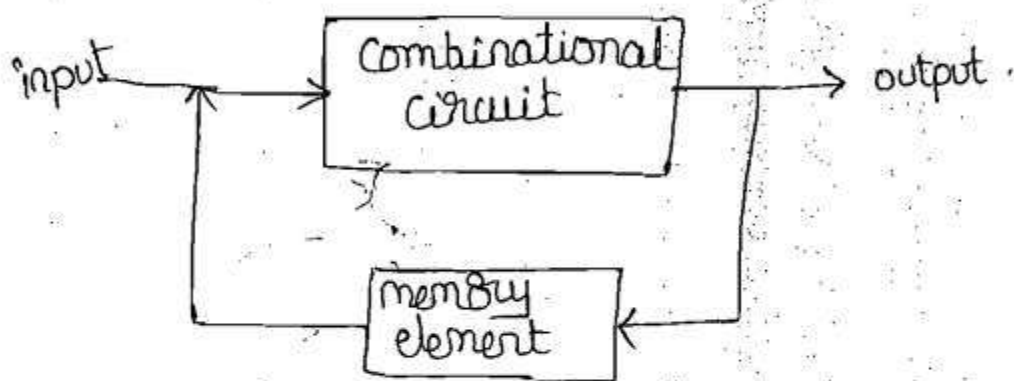
Truth table

$$Y = \bar{S}_1 \cdot \bar{S}_0 \cdot A + \bar{S}_1 \cdot S_0 \cdot B + S_1 \cdot \bar{S}_0 \cdot C + S_1 \cdot S_0 \cdot D$$

Sequential circuits I.

⑤

In sequential logic circuits, the output is a function of the present inputs as well as the past inputs and outputs. Sequential circuits include memory element to store the past data. The flip-flop is a basic element of sequential logic circuits.



Block diagram of sequential circuit.

There are two types of sequential circuits.

- Synchronous circuit:- The sequential circuits which are controlled by a clock are called synchronous sequential circuits. These circuits get actuated only clock signal is present.
- A synchronous circuit:- The sequential circuits which are not controlled by a clock are called asynchronous sequential circuits, that is the sequential circuits in which events can take place any time the inputs are applied are called asynchronous sequential circuits.

Comparison between Synchronous & Asynchronous sequential circuit.

Synchronous sequential circuit.	Asynchronous sequential circuit.
1. In synchronous circuits, the change in input signals can affect memory elements upon activation of clock signal. 2. In synchronous circuits, memory elements are clocked FF's. 3. The maximum operating speed of clock depends on time delays involved. 4. They are easier to design.	1. In asynchronous circuits, change in input signals can affect memory elements at any instant of time. 2. In asynchronous circuits, memory elements are either unclocked FFs or time delay elements. 3. Since the clock is not present, asynchronous circuits can operate faster than synchronous circuits. 4. more difficult to design.

→ Latches & Flip flops :-

- The most important memory element is the flip-flop which is made up of an assembly of logic gates.
- even though a logic gate by itself has no storage capability, several logic gates can be connected together in ways that permit information to be stored.
- Flip-flops are the basic building blocks of most sequential circuits. Actually, flip-flop is an one-bit

memory device and it can store either 1 or 0.

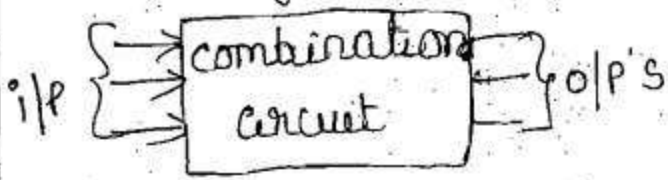
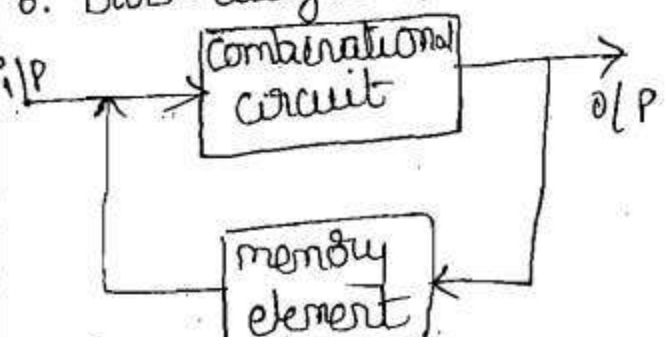
→ Latch is a sequential device that checks all its inputs continuously and changes its outputs accordingly at any time independent of a clock signal.

→ It refers to non-clocked flip-flops, because these flip-flops 'latch on' to a 1 or a 0 immediately upon receiving the input pulse.

→ Difference between latches & flip-flops.

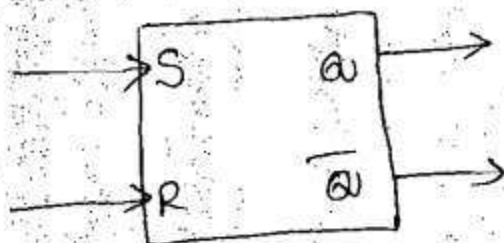
Latch	Flip-flop.
1. A latch is an electronic sequential logic circuit used to store information in an asynchronous arrangement. 2. one latch can store one bit information, but output state changes only in response to data input. 3. latch is an asynchronous device and it has no clock input. 4. latch holds a bit value and it remains constant until new inputs force it to change. 5. latches are level-sensitive and the output tracks the input when the level is high. Therefore as long as the level is logic level 1, the output can change if the input changes.	1. A flip flop is an electronic sequential logic circuit used to store information in an asynchronous arrangement. 2. one flip-flop can store one bit-data, but output state changes with clock pulse only. 3. Flip-flop has clock input and its output is synchronised with clock pulse. 4. Flip flops holds a bit value and it remains constant until a clock pulse is received. 5. Flip flops are edge-sensitive. They can store the input only when there is either a rising or falling edge of the clock.

Difference between combinational, sequential circuits.

Combinational circuit	Sequential circuits.
1. The digital logic circuit whose outputs can be determined using the logic function of current state input are combinational logic circuits. 2. It contains no memory element. 3. It's behaviour is described by the set of output functions. 4. The combinational logic circuits are independent of the clocks. 5. The combinational digital logic circuit don't require any feed back. 6. combinational circuits are easy to design. 7. Combinational circuits are faster because the delay between the input and the output is due to propagation delay of gates only.	1. The digital logic circuits whose outputs can be determined using the logic function of current state inputs and past state inputs are called as sequential logic circuits. 2. It contains memory elements. 3. It's behaviour is described by the set of next state functions and the set of output functions. 4. The maximum sequential logic circuits are uses a clock for triggering the flip-flop operation. 5. The sequential digital logic circuits utilize the feedbacks from outputs to inputs. 6. Sequential circuits are complex to design. 7. Sequential circuits are slower than combinational circuits.
8. Block diagram 	8. Block diagram, 

S-R latch :-

The S-R latch has two inputs, namely SET(S) and RESET(R), and two outputs a and $\bar{a}$, where $\bar{a}$ is the complement of a .

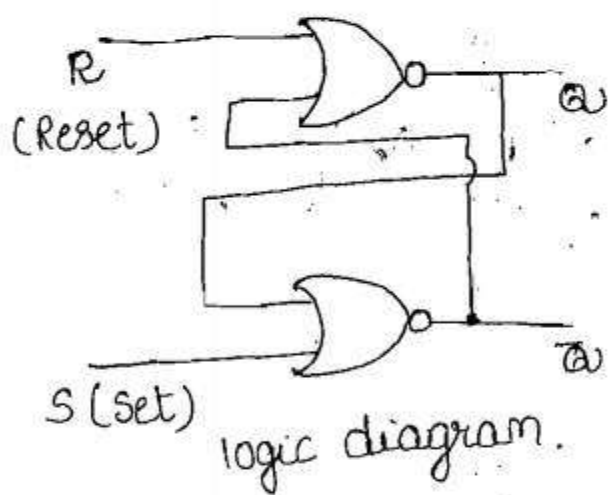


Logic symbol of S-R latch

S-R latch using NOR Gates :- [Active-high S-R latch].

The logic diagram of the S-R latch composed of two cross-coupled NOR gates. S and R are two inputs of S-R latch.

- S stands for set, it means that when S is 1, it stores 1.
- R stands for Reset, and if R=1, latch Reset and its output will be 0. This circuit is called as NOR gate latch or S-R latch.



logic diagram.

Simplified truth table.

S	R	a_{n+1}	State
0	0	a_n	No change
0	1	0	Reset
1	0	1	Set
1	1	X	Invalid state

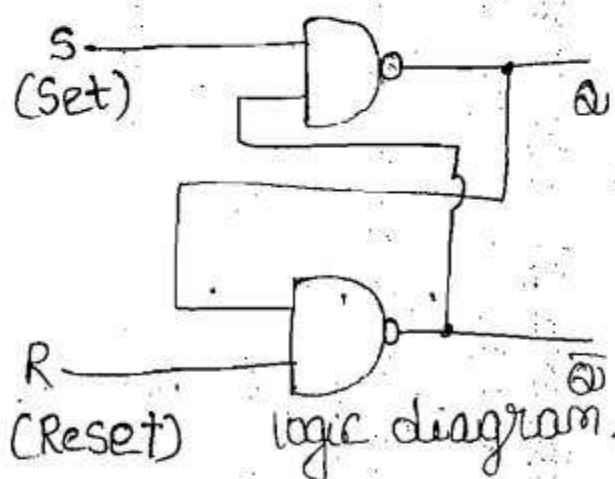
S	R	a_n	a_{n+1}	State
0	0	0	0	No change
0	0	1	1	
0	1	0	0	RESET
0	1	1	0	
1	0	0	1	SET
1	0	1	1	
1	1	0	X	Indetermined (invalid).
1	1	1	X	

Truth table.

1. $SET=0, RESET=0$: This is the normal resting state of the NOR latch and it has no effect on the output state. a and $\bar{a}$ will remain in whatever state they were prior to the occurrence of this input condition.
2. $SET=0, RESET=1$, This will always reset $a=0$, where it will remain even after $RESET$ returns to 0.
3. $SET=1, RESET=0$, This will always set $a=1$, where it will remain even after SET returns to 0.
4. $SET=1, RESET=1$, This condition tries to SET and Reset the latch at the same time, and it produces $a=\bar{a}=0$. If the inputs are returned to zero simultaneously, the resulting output state is erratic and unpredictable. This input condition should not be used. It is indetermined state or invalid state.

S-R latch using NAND Gates:- (active low S-R latch)

→ The logic diagram of the S-R latch composed of two cross-coupled NAND gates.



S	R	Q_{n+1}	State
0	0	X	invalid
0	1	1	Set
1	0	0	Reset
1	1	Q_n	N.C

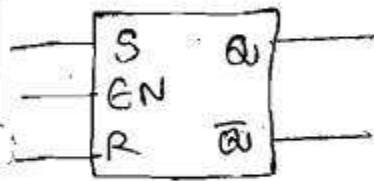
S	R	Q_n	Q_{n+1}	State
0	0	0	X	indetermined (invalid)
0	0	1	X	
0	1	0	1	Set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	0	No change
1	1	1	1	

Truth table.

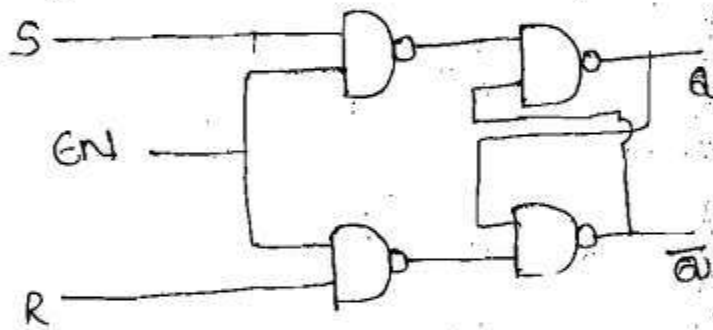
← Simplified truth table.

> Gated latches :-

The gated S-R latch :- The output can change state any time the input conditions are changed, so they are called Asynchronous latches. A gated S-R latch requires an Enable (EN) input. Its S and R inputs will control the state of latch only when the enable is high. when the enable is low, the inputs become ineffective and no change of state can take place.

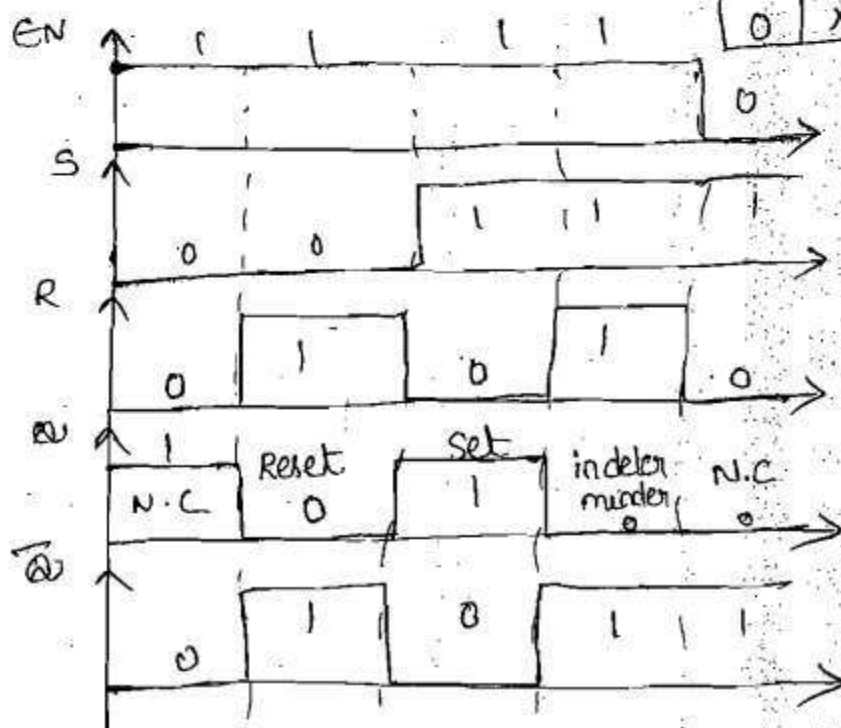


logic symbol.



logic diagram.

EN	S	R	Q_n	Q_{n+1}	State
1	0	0	0	0	No change
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	X	indeterminate (invalid)
1	1	1	1	X	
0	X	X	0	0	No change
0	X	X	1	1	



waveforms for Gated S-R latch.

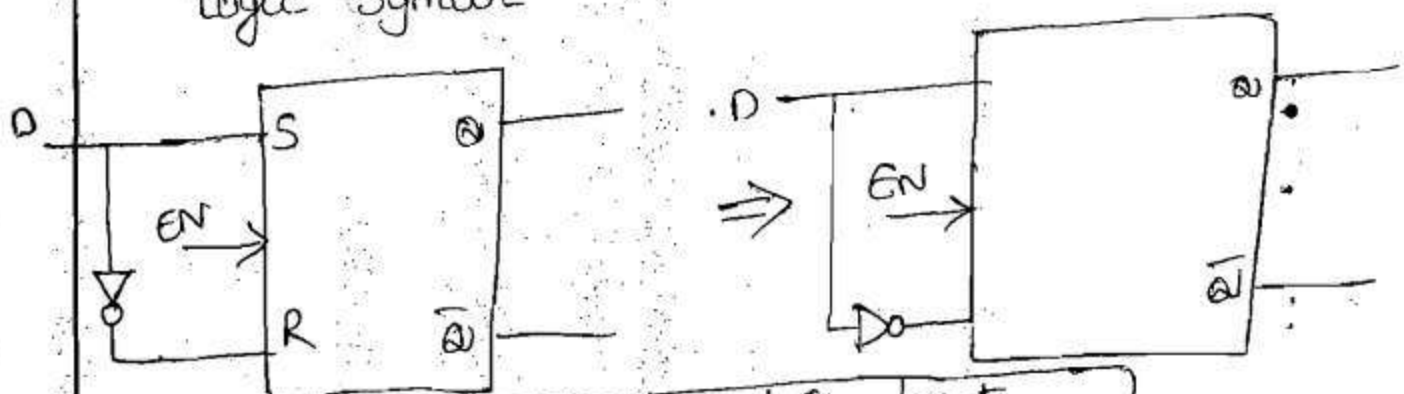
The Gated D-latch; - In many applications, it is not necessary to have separate S and R inputs to a latch. If the input combinations $S=R=0$ and $S=R=1$ are never needed, the S and R are always the complement of each other. So, to construct a latch with a single input (S) and obtain the R input by inverting it. This single input is labelled D (for data) and the device is called a D-latch.

→ when $D=1$, $S=1$ and $R=0$, causing the latch to set when Enabled. when $D=0$, $S=0$ and $R=1$, causing the latch to Reset when enabled. when EN is low, the latch is ineffective, and any change in the value of D input does not affect the output at all.

→ when EN is high, a low D-input makes Q low, i.e. resets the latch and high D input makes Q high, that is sets the latch.

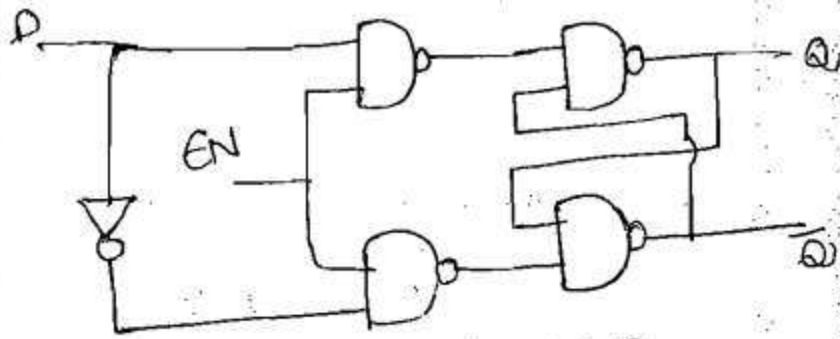
→ The output Q follows the D-input when EN is high. So this latch is said to be transparent.

Logic Symbol

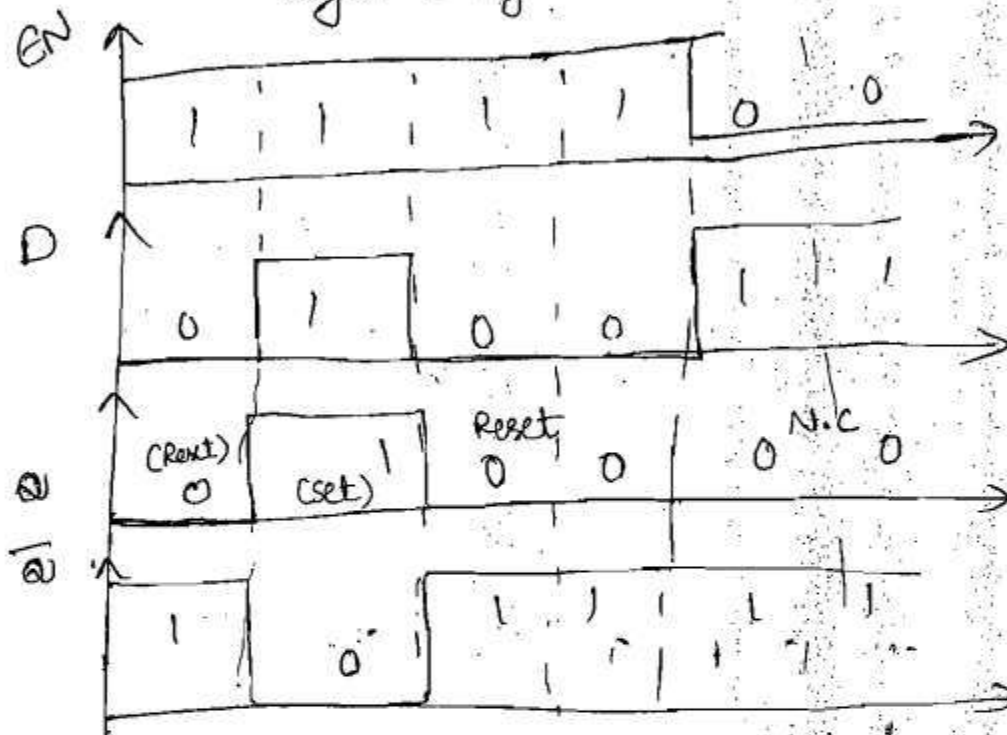


EN	D	Q _n	Q _{n+1}	State
1	0	0	0	Reset
1	0	1	0	
1	1	0	1	Set
1	1	1	1	
0	X	0	0	No change (NC)
0	X	1	1	

Truth table



logic diagram.



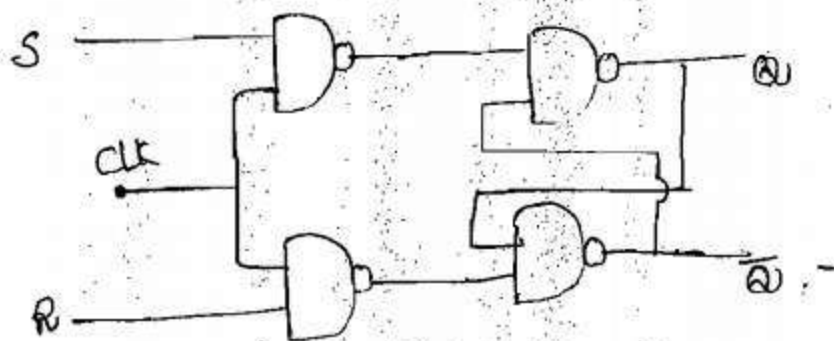
waveforms for Gated D-latch.

Flip-flops :-

Types of flip-flops :-

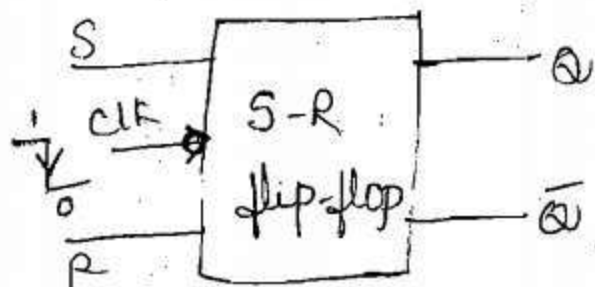
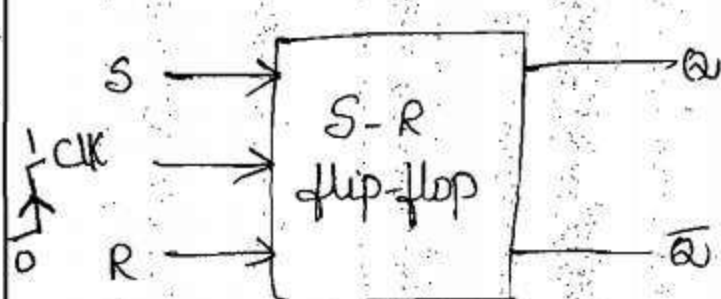
- S-R flip-flop
- D flip-flop
- J-K flip-flop
- T flip-flop.

S-R flip-flop:- logic diagram.



Symbol of +ve edge trigger.

Symbol of -ve edge trigger



Truth table:-

clk	S	R	Q_n	Q_{n+1}	State
↑	0	0	0	0	No change
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	set
↑	1	0	1	1	
↑	1	1	0	X	Invalid state
↑	1	1	1	X	
0	X	X	0	0	No change
0	X	X	1	1	

Characteristic equation:- The characteristic equation of a flip-flop is the equation expressing the next state of a flip-flop in terms of its present state and present excitations. To obtain the characteristic equation of a.

flip-flop write the excitation requirements of the flip-flop, draw a k-map for the next state of the flip-flop in terms of its present state and inputs and simplify it.

S \ R _n	00	01	11	10
0	0	1	3	2
1	1	1	X	X

$$Q_{n+1} = S + \bar{R}Q_n$$

Truth table:-

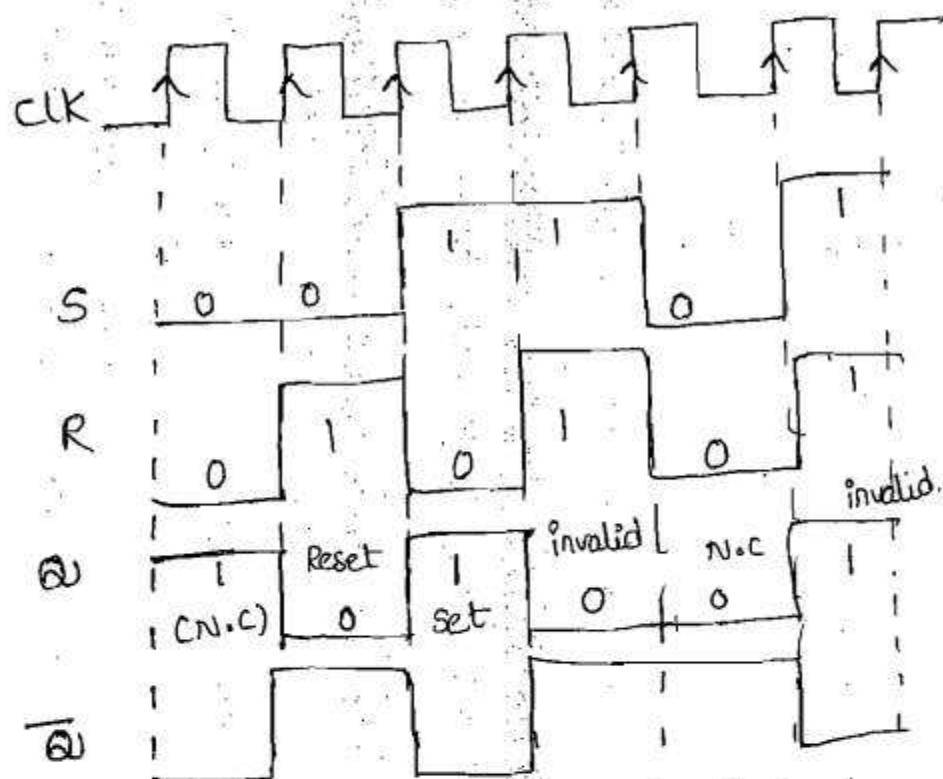
S	R	Q _{n+1}
0	0	Q _n
0	1	0
1	0	1
1	1	X

Excitation table:- A table which lists the present state, the next state and the excitations of a flip-flop is called the excitation table.

→ A table which indicates the excitations required to take the flip-flop from the present state to the next state.

Present State Q _n	next state Q _{n+1}	Required S R	
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

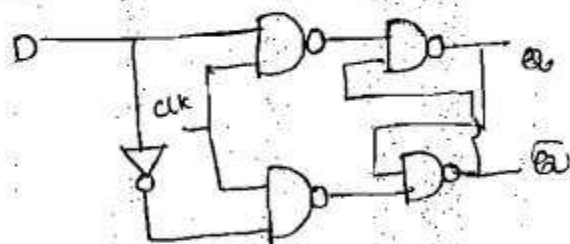
Timing diagram



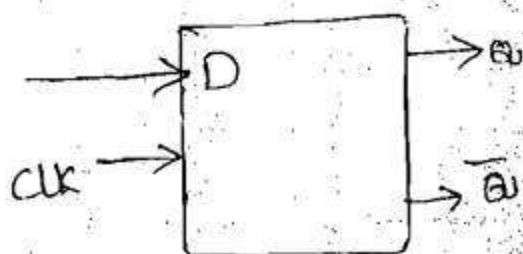
waveforms for S-R flip-flop.

D- flip - flop:-

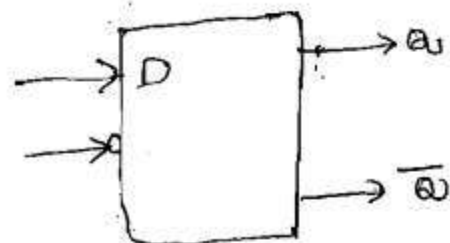
logic diagram



Symbol of +ve edge trigger



Symbol of -ve trigger



Truth table :-

D	Q_{n+1}
0	0
1	1

characteristic Table

clk	D	Q_n	Q_{n+1}	State
↑	0	0	0	Reset
↑	0	1	0	
↑	1	0	1	Set
↑	1	1	1	
↑	X	0	0	No change
↑	X	1	1	

characteristic equation :-

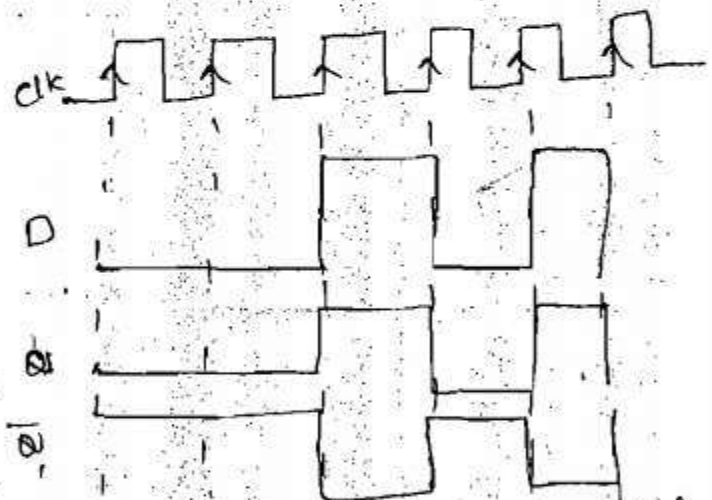
Q_n	0	1
0	0	1
1	1	1

$$Q_{n+1} = D$$

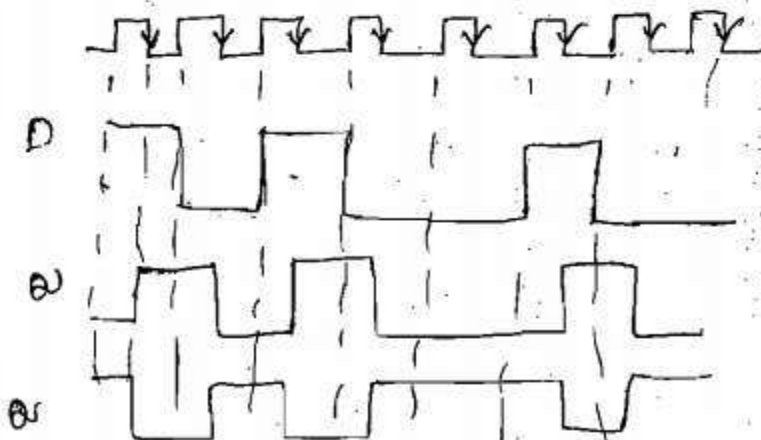
Excitation table :-

Present state Q_n	Next state Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

waveforms

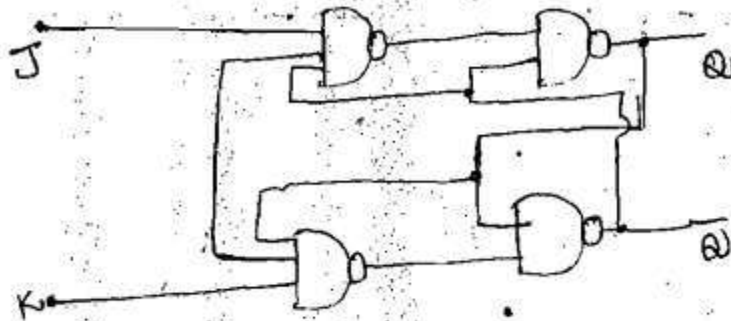


(Positive-edge trigger clk)

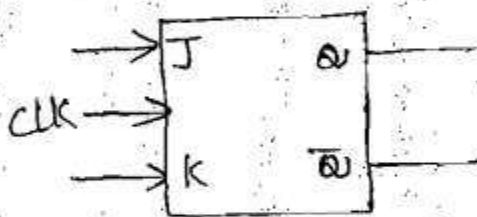


(negative-edge trigger clk)

(J-K - flip flops). logic diagram



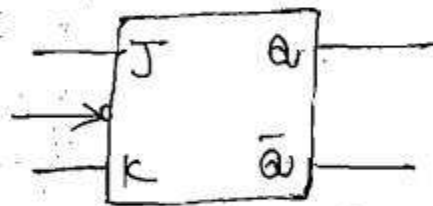
logic symbol .



(positive-edge)

Truth table :-

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Toggle

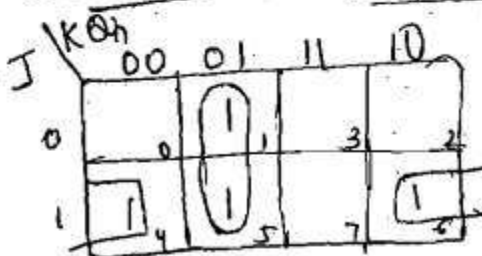


(negative-edge)

characteristic table

clk	J	K	Q_n	Q_{n+1}	State
0	0	0	0	0	No change
	0	0	1	1	
1	0	1	0	0	Reset
	0	1	1	0	
1	1	0	0	1	Set
	1	0	1	1	
1	1	1	0	1	Toggle
	1	1	1	0	
0	X	X	0	0	No change
0	X	X	1	1	

Characteristic equation

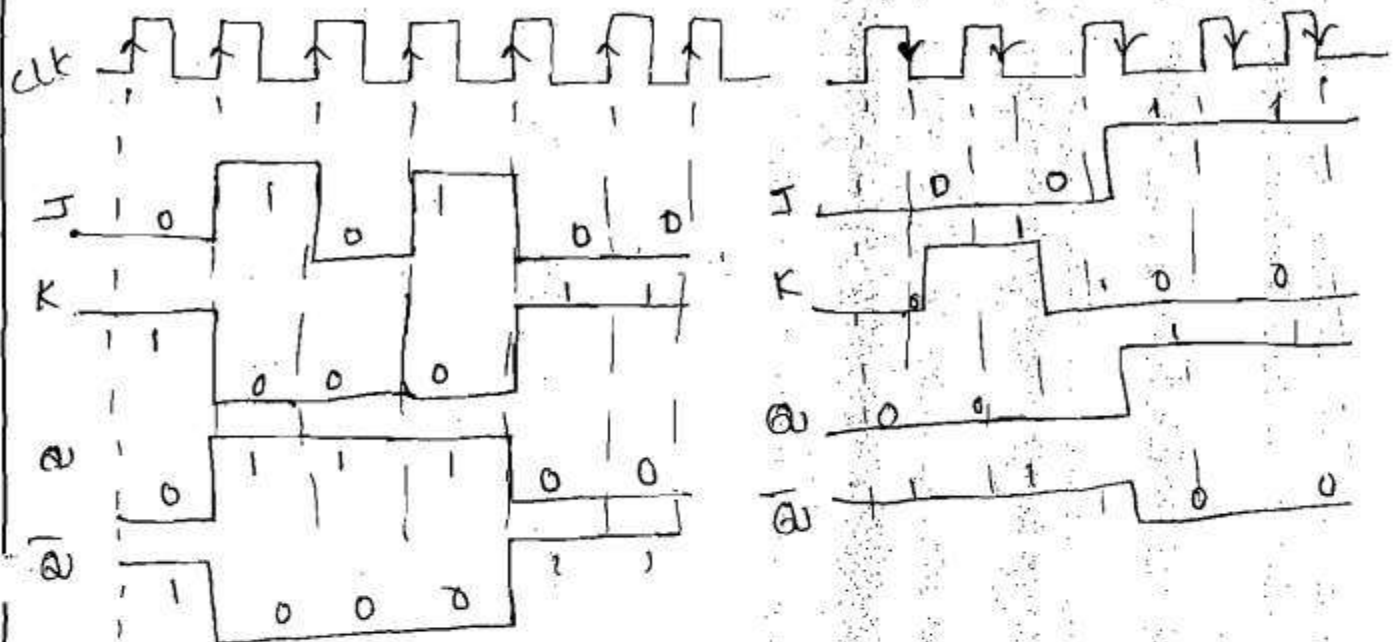


$$Q_{n+1} = \bar{K}Q_n + J\bar{Q}_n$$

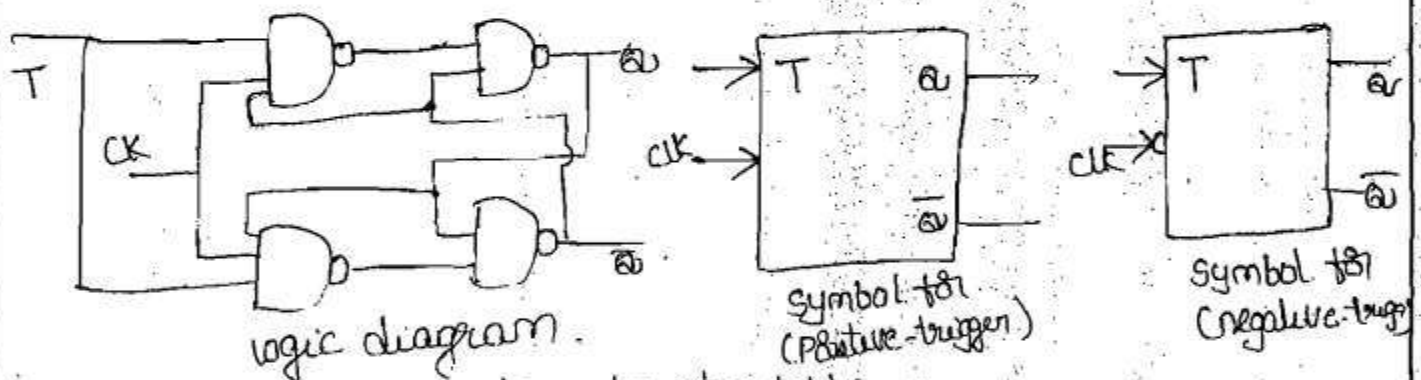
excitation table

Present state	next state	inputs	
		J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Wave-forms for Positive-edge triggering negative-edge triggering



T-flip flop :-



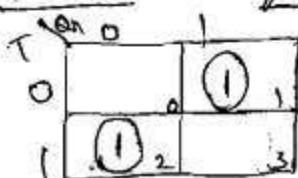
Truth table

T	Q_{n+1}
0	Q_n
1	0

characteristic table

clk	T	Q_n	Q_{n+1}	State
$\downarrow$	0	0	0	No change
$\downarrow$	0	1	1	
$\downarrow$	1	0	1	Toggle
$\downarrow$	1	1	0	
$\otimes$	X	0	0	No change
$\otimes$	X	1	1	

characteristic equation

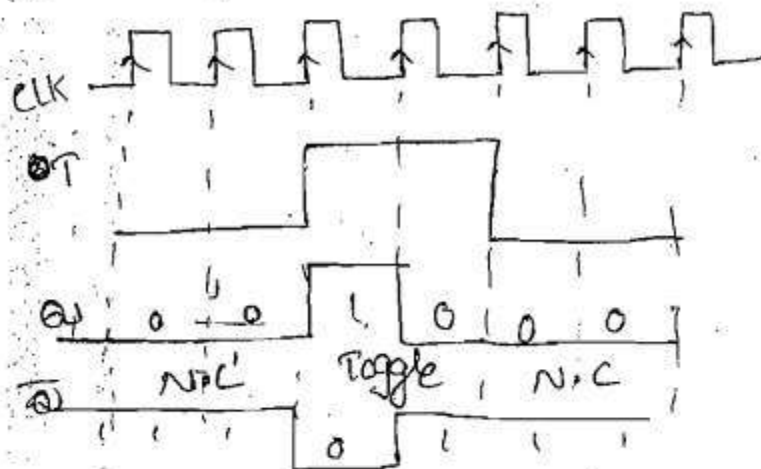


$$Q_{n+1} = T\bar{Q}_n + \bar{T}Q_n$$

Excitation table

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

waveforms



Race - Around condition:-

If the width of the clock pulse t_p is too long, the state of the flip-flop will keep on changing from 0 to 1, 1 to 0, 0 to 1 and so on, and at the end of the clock pulse, its state will be uncertain. This phenomenon is called the race around condition. The outputs Q and $\bar{Q}$ will change on their own if the clock pulse width t_p is too long compared with the propagation delay τ of each NAND Gate.

$$t_p \gg \tau$$

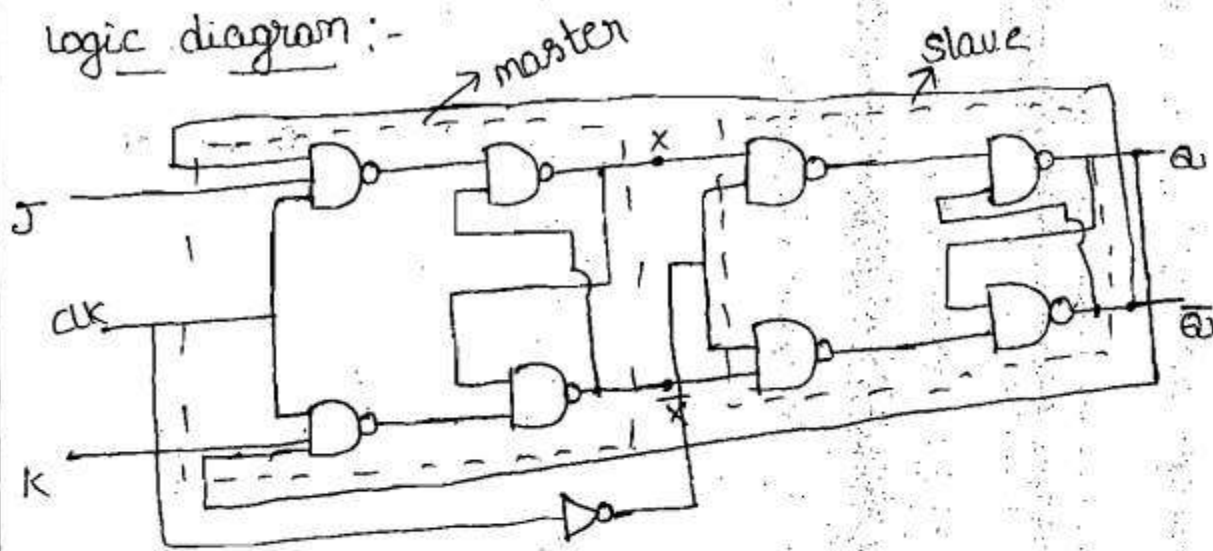
$t_p \rightarrow$ pulse width

$\tau \rightarrow$ propagation delay

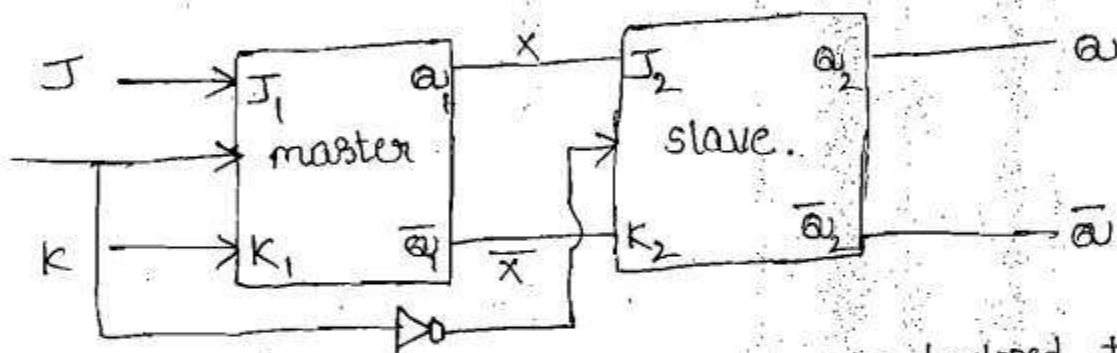
The clock pulse width should be such as to allow only one change to complement the state and not too long to allow many changes resulting in uncertainty about the final state. This is a stringent requirement which cannot be ensured in practice. This problem is eliminated using master-slave flip-flop or edge triggered flip-flop.

master-slave J-K flip-flop :-

logic diagram :-



Symbol :-



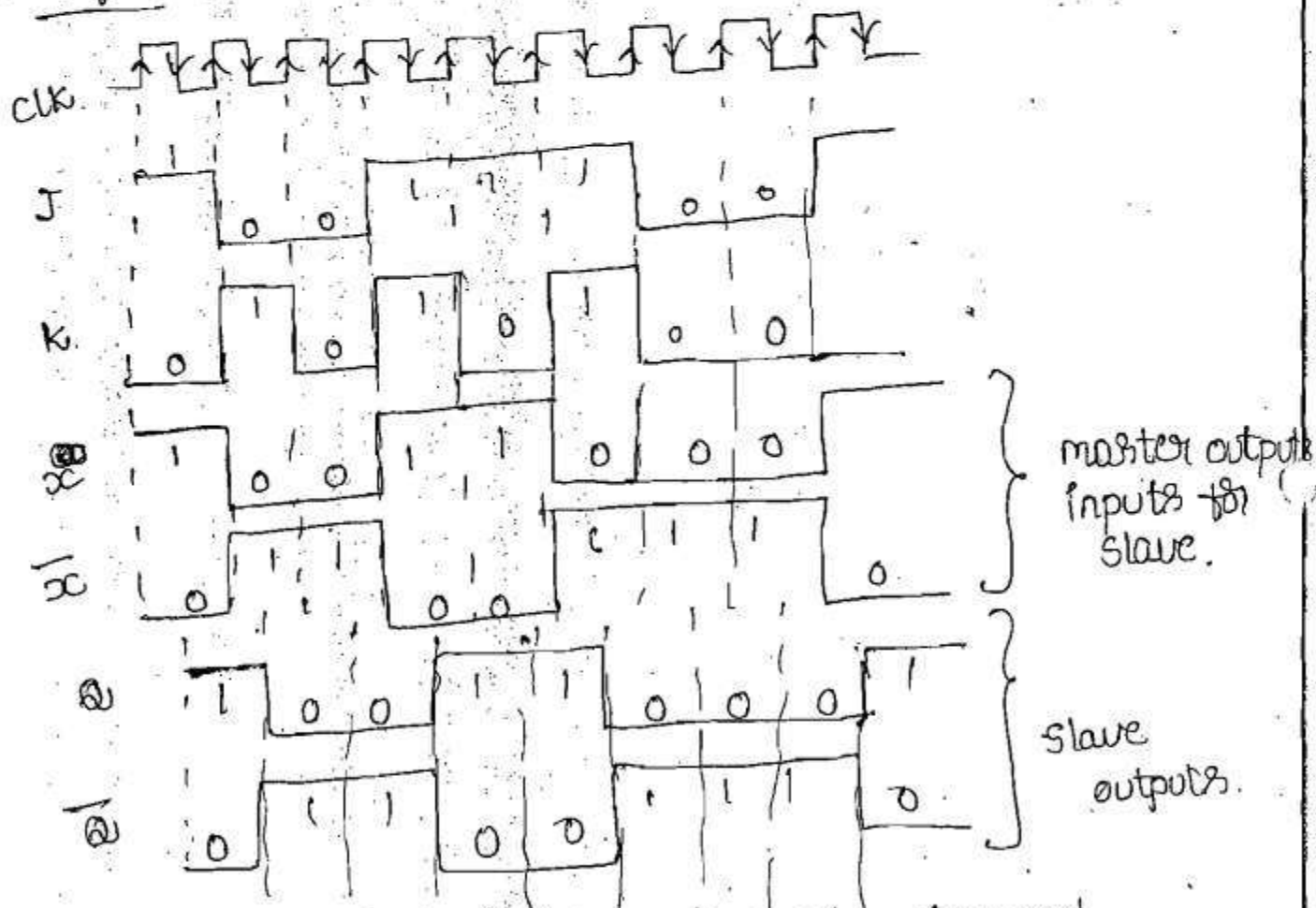
The master-slave flip flop was developed to make the synchronous operation more predictable, that is, to avoid the problems of logic race in clocked flip-flops. This improvement is achieved by introducing a known time delay between the time that the flip-flop responds to a clock pulse and the time the response appears at its output. A master-slave flip-flop is also called a pulse-triggered flip-flop because the length of the time required for its output to change state equals the width of one clock pulse.

In master-slave J-K flip-flop actually contains two flip-flops - a master flip-flop and a slave flip-flop. The control inputs are applied to the master flip-flop and master output is given as input to the

Slave flip-flop. on the rising edge of the clock pulse, the levels on the control inputs are used to determine the output of the master. on the falling edge of the clock pulse, the state of the master is transferred to the slave, whose outputs are awarded.

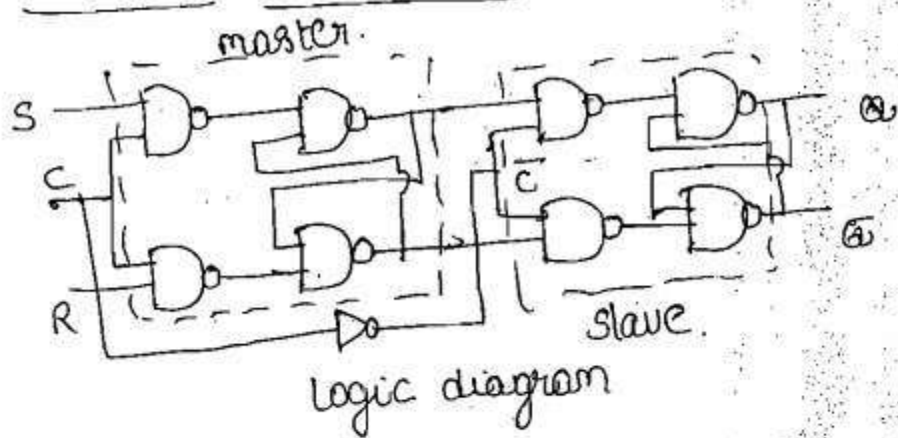
The master-slave flip-flops function very much like the negative-edge triggered flip-flops except for one more disadvantage. The control inputs must be held stable while clk is high, otherwise an unpredictable operation may occur. This problem with the master-slave flip-flop is overcome with an improved master-slave version called the master-slave with data lock-out.

waveforms :-



In slave accept the negative edge triggered signal only. master accept the positive-edge triggered signal only.

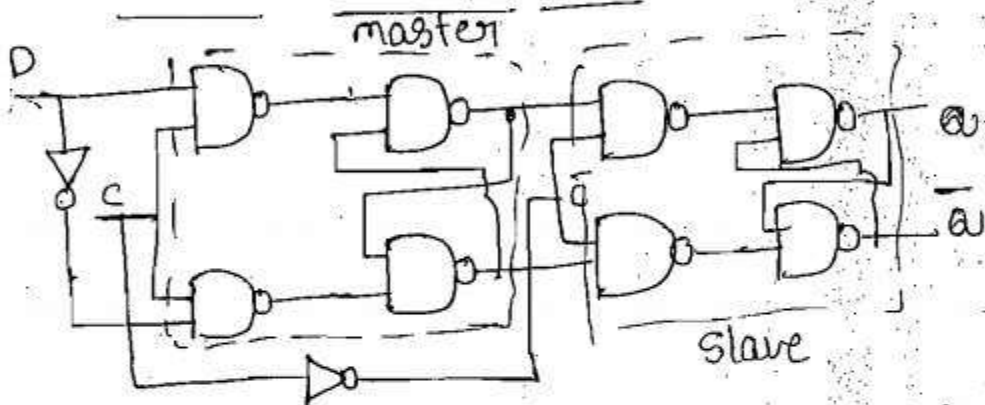
master-slave S-R flip flop :-



S	R	clk	Q	State
0	0		Q ₀	N.C
0	1		0	Reset
1	0		1	Set
1	1		X	Invalid

Truth table

master-slave D flip flop :-



D	clk	Q	State
0		0	Reset
1		1	Set

Truth table

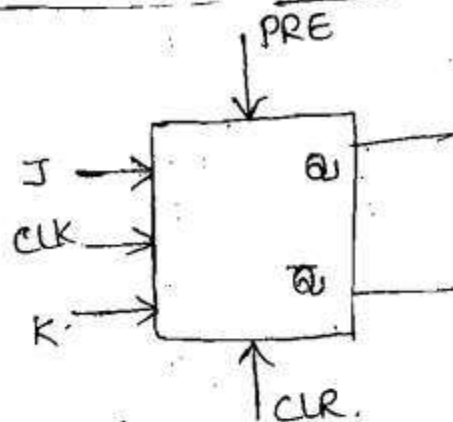
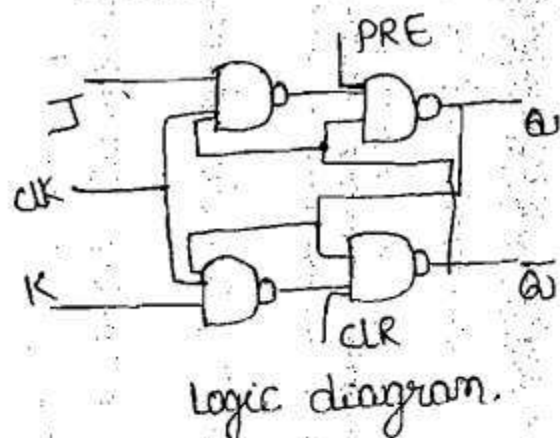
Asynchronous inputs (PRESET and CLEAR)

The S-R, D, and J-K inputs are called synchronous inputs, because their effect on the flip-flop output is synchronized with the clock input.

most IC flip-flops also have one or more asynchronous inputs. These asynchronous inputs affect the flip-flop output independently of the synchronous inputs and the clock input. These asynchronous inputs can be used to SET the flip-flop to the 1 state or RESET the flip-flop to the 0 state at any time regardless of the conditions at the other inputs.

They are normally labelled PRESET or direct SET or DC SET, and CLEAR or direct RESET or DC CLEAR.

J-K flip-flop with Active-high Asynchronous inputs :-



→ When $PRE = 0$, $CLR = 0$ then DC SET = 1 and DC CLEAR = 1, The Asynchronous inputs are inactive and the flip-flop responds freely to J, K and CLK inputs in the normal way. In other words, the clocked operation can take place.

→ When $PRE = 0$, $CLR = 1$ then DC SET = 0 and DC CLEAR = 1. The)^x

→ When $PRE = 0$, $CLR = 0$ then Asynchronous inputs are inactive and the flip-flop responds freely to J, K and CLK inputs.

→ When $PRE = 0$, $CLR = 1$ then Asynchronous input clear is active then flip-flop output is '0'.

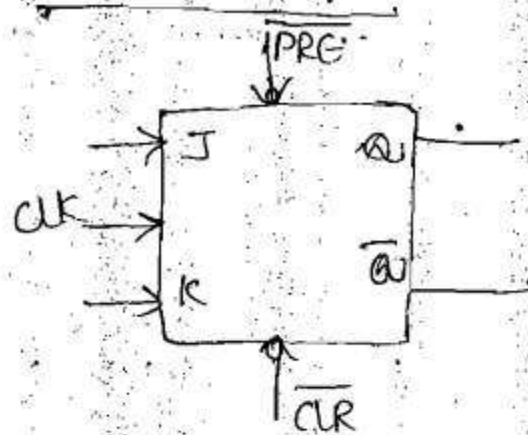
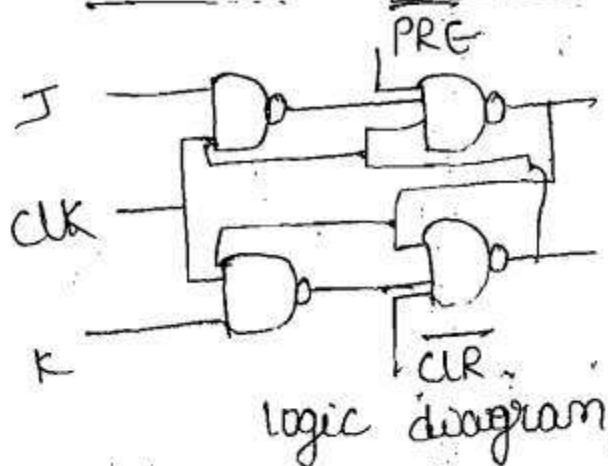
→ When $PRE = 1$, $CLR = 0$ then Asynchronous input PRESET is active the flip-flop output is '1'.

→ When $PRE = 1$, $CLR = 1$. This condition should not be used since it can result in an invalid state.

DC SET (PRE)	DC RESET (CLR)	FF response
0	0	clock operation
0	1	$Q = 0$
1	0	$Q = 1$
1	1	Not used.

Truth table

J-K flip-flop with Active-low Asynchronous inputs



- when $\overline{PRE} = 0$ and $\overline{CLR} = 0$, This condition should not be used. Since it can result in an invalid state.
- when $\overline{PRE} = 0$ and $\overline{CLR} = 1$, then Asynchronous input $\overline{PRESET}$ is active then output is '1'.
- when $\overline{PRE} = 1$ and $\overline{CLR} = 0$, then Asynchronous input $\overline{CLEAR}$ is active then output is '0'.
- when $\overline{PRE} = 1$ and $\overline{CLR} = 1$, Then A synchronous inputs are inactive ~~the~~ and the flip responds freely to J, K and CLK inputs.

DC SET ($\overline{PRE}$)	DC RESET ($\overline{CLR}$)	FF response
0	0	not used
0	1	$Q = 1$
1	0	$Q = 0$
1	1	clock operation

Truth table

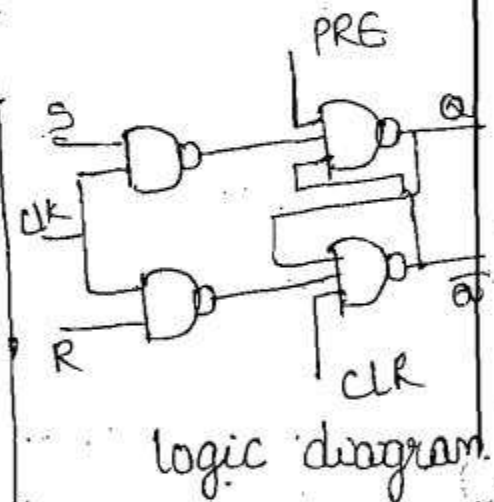
Truth table for J-K flip-flop (both Asynchronous & synchronous inputs).

PRE	CLR	CLK	J	K	Q	$\bar{Q}$
1	1	X	X	X	invalid.	
1	0	X	X	X	1	0
0	1	X	X	X	0	1
0	0	$\downarrow$	0	0	0	1
0	0	$\uparrow$	0	1	0	1
0	0	$\downarrow$	1	0	1	0
0	0	$\uparrow$	1	1	0	1

X - don't care
means either '0'
or '1'.

Similarly S-R flip-flop.

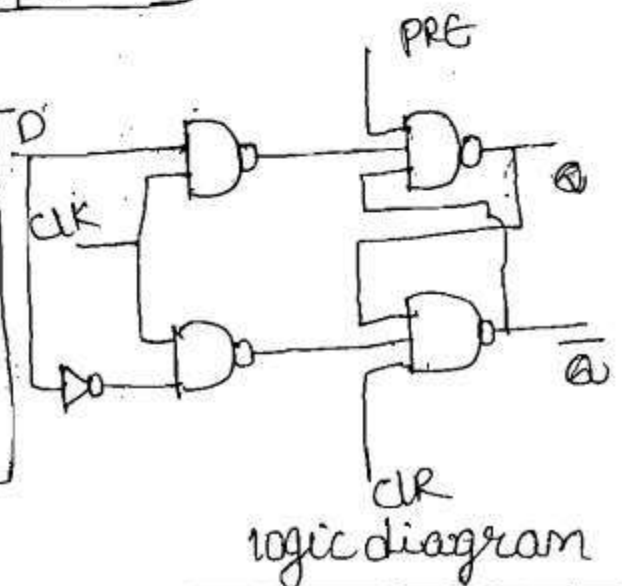
PRE	CLR	CLK	S	R	Q	$\bar{Q}$
1	1	X	X	X	invalid	
1	0	X	X	X	1	0
0	1	X	X	X	0	1
0	0	$\downarrow$	0	0	0	1
0	0	$\downarrow$	0	1	0	1
0	0	$\downarrow$	1	0	1	0
0	0	$\downarrow$	1	1	invalid	
0	0	0	X	X	N.C.	



Similarly for D-flip-flop.

PRE	CLR	CLK	D	Q	$\bar{Q}$
1	1	X	X	invalid	
1	0	X	X	1	0
0	1	X	X	0	1
0	0	$\downarrow$	0	0	1
0	0	$\downarrow$	1	1	0

Truth table



Flip-flop conversions:-

Step 1:- obtain the characteristic table of required flip-flop.

Step 2:- And also obtain the excitation table of available flip-flop.

Step 3:- obtain the expressions for the inputs of the existing flip-flop in terms of the inputs of the required flip-flop and the present state variables of the existing flip-flop and implement them.

Conversion of T-flip-flop to S-R flipflop

Available flip-flop $\rightarrow$ T-flip flop (Excitation table)

Required flip-flop $\rightarrow$ S-R flipflop (characteristic table)

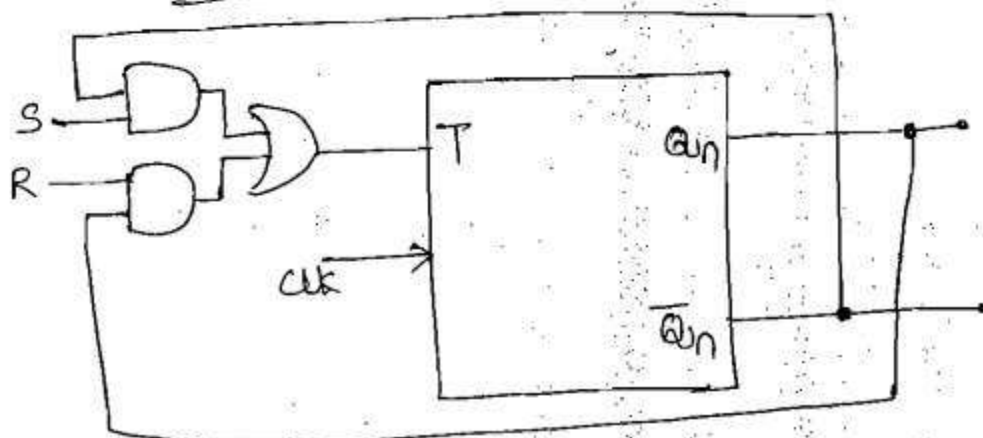
S	R	a_n	a_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	X	X
1	1	1	X	X

K-map for T

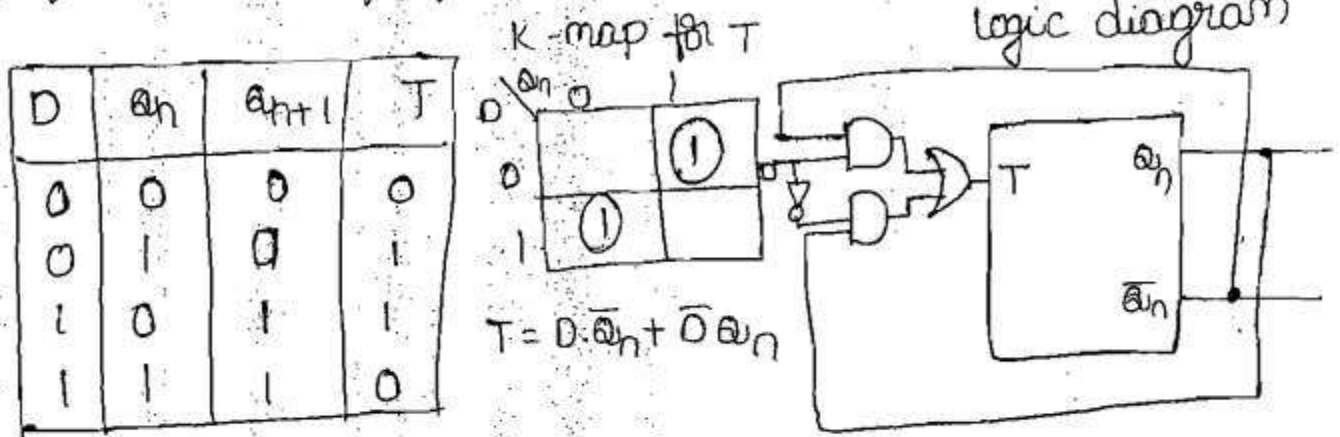
	$\bar{a}_n$	a_n	
S	0	1	
R	0	0	1
1	1	0	X

$$T = S \cdot \bar{a}_n + R \cdot a_n$$

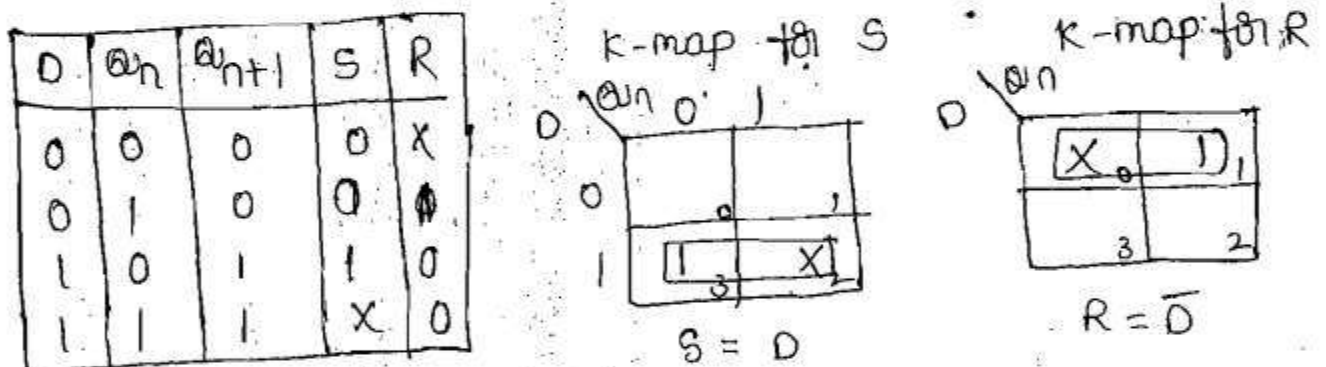
logic diagram



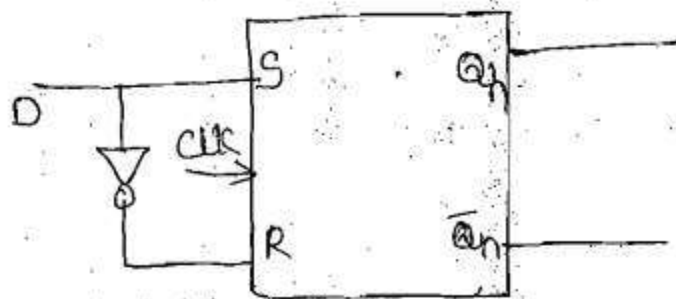
- conversion of T-flip-flop to D-flip-flop.
 Available → T-flipflop → excitation table
 Required → D-flipflop → characteristic table.



- Construct a D-flip flop by ~~so~~ using S-R flip-flop.
 Available → S-R flipflop → excitation table.
 Required → D flip flop → characteristic table.



logic diagram.



→ Realize the S-R flip-flop by using J-K flip-flop.

Available → S-R → excitation table

Required → J-K → characteristic table

J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

K-map for S

$J \backslash Q_n$	0	1	1	0
0	0	X	0	0
1	1	X	0	1

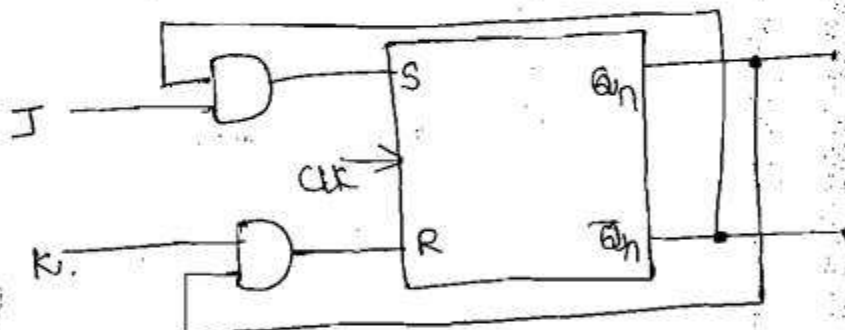
$$S = J\bar{Q}_n$$

K-map for R

$K \backslash Q_n$	0	1	1	0
0	X	0	1	X
1	0	0	1	0

$$R = KQ_n$$

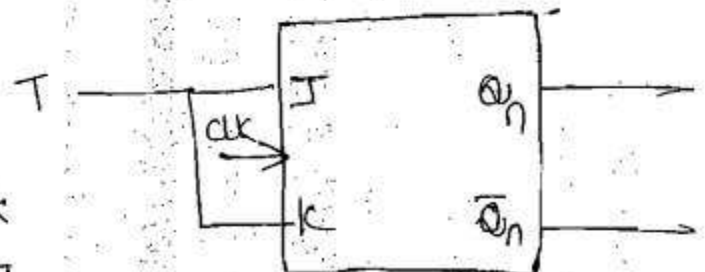
logic diagram:-



→ Convert J-K flip-flop to T flip-flop.

T	Q_n	Q_{n+1}	J	K
0	0	0	0	X
0	1	1	X	0
1	0	1	X	X
1	1	0	X	1

logic-diagram



K-map for J

$T \backslash Q_n$	0	1
0	0	X
1	1	X

$$J = T$$

K-map for K

$T \backslash Q_n$	0	1
0	X	0
1	X	1

$$K = T$$

→ conversion of D-flip flop to T flip-flop.

* Available → D-flip flop → excitation table

* Available → T-flip flop → characteristic table

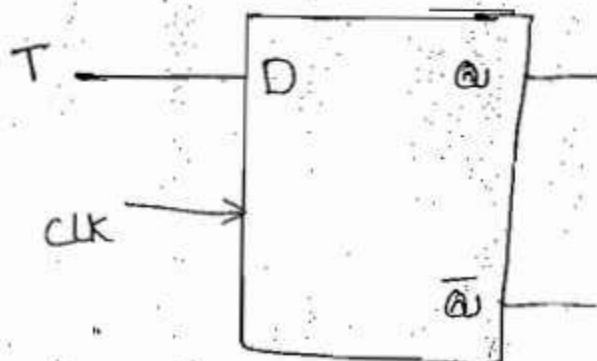
T	a_n	a_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

K-map for D

a_n	0	1
T		
0	0	1
1	0	1

$$D = T$$

logic diagram



Counters :- A digital counter is a set of flip-flops whose states change in response to pulses applied at the input to the counter.

- The name itself it indicates, a counter is used to count pulses.
- counters may be asynchronous counters or synchronous counters. Asynchronous counters are also called ripple counters.
- In asynchronous counters Flipflops are not triggered simultaneously. The clock does not directly control the time at which every stage changes state.
- In synchronous counters are clocked such that each FF in the counter is triggered at the same time.

comparison of synchronous and Asynchronous counters

Asynchronous counters	Synchronous counters.
1. In this type of counter FF's are connected in such a way that the output of first FF drives the clock for the second FF, the output of the second FF to the third FF.	1. In this type of counter there is no connection between the output of first FF and clock input of next FF and so on.
2. All the FF's are not clocked simultaneously.	2. All the FF's are clocked simultaneously.
3. Design and implement is very simple even for more number of states.	3. Design and implementation becomes tedious and complex as the number of states increases.
4. main drawback of these counters is their low speed as the clock is propagated through a number of FF's before it reaches the last FF.	4. since clock is applied to all the FF's simultaneously the total propagation delay is equal to the propagation delay of only one FF. Hence they are faster.

$Q=0$. Since both of these signals don't work with Clock synchronisation, therefore these are called Asynchronous Inputs.

4.4 Digital Counters

A counter is a sequential circuit. It is a type of digital circuit which is used to count pulses. The most widespread use of FFs is on counters. With a clock signal applied, it is a cluster of FFs. There are two types of counters -

- i. Ripple/Asynchronous counters
- ii. Synchronous counters

4.4.1 Ripple/Asynchronous counters.

A cascaded configuration of FFs, known as a ripple counter, is the one in which the output of one FF drives the clock input of the one after it (Rippling the clock input). The modulus of the counter is a parameter that determines how many different logic states the cascaded arrangement passes through before repeating the sequence, determines how many FFs are present.

An asynchronous counter or serial counter are other names for a ripple counter in which the initial FF has clock input signal externally applied and all others are having a rippling clock obtained from previous FF's output. For instance, the output of the 1st FF serves as the clock input for the 2nd FF, the output of the 2nd FF feeds the third FF clock input, and so on. In Fig. 4.16, 3-bit up counter is presented using T-FF. Here Q_0 is LSB while Q_2 is MSB. This counter can be converted to 3-bit down counter using active edge as positive edge of the clock. Fig. 4.17 presents the waveform and state diagram a 3-bit up-counter. This can also be implemented using D FFs by applying Q' input to every D input of the same FF instead of T-FF used in below circuit.

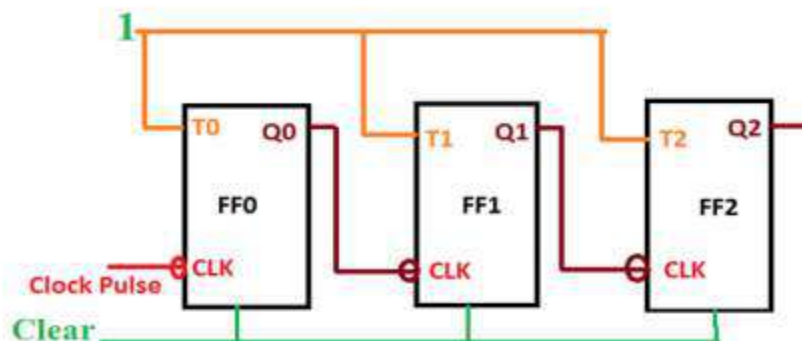


Fig. 4.16: Block Diagram of 3-Bit Ripple Up Counter using T-FFs



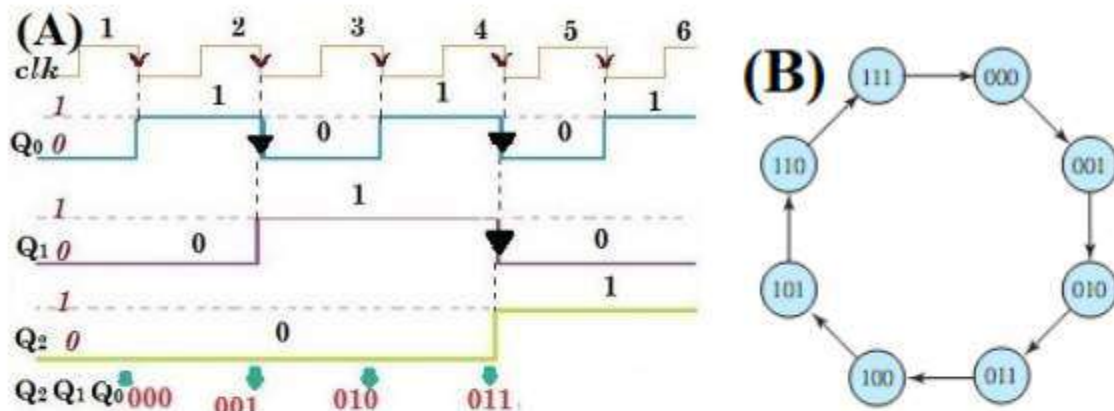


Fig. 4.17: (A) Wave diagram and (B) state diagram for 3-bit up counter

4.4.2 Synchronous Counter

All of the FFs of a synchronous counter are using same clock signal, sometimes referred to as a parallel counter. No matter how many FFs were used to build the counter, the delay in this situation is simply the propagation delay of one FF. In other words, the delay does not rely on how big the counter is.

4.4.2.1 Procedure for Designing Synchronous Sequential Circuits

The procedure for designing synchronous sequential circuits can be summarized as follows:

1. From the word description and specifications of the desired operation, derive a state diagram for the circuit.
2. Reduce the number of states if necessary.
3. Assign binary values to the states.
4. Obtain the binary-coded state table.
5. Choose the type of flip-flops to be used.
6. Derive the simplified flip-flop input equations and output equations.
7. Draw the logic diagram.

Synchronous counters can be designed using the above procedure. This will be discussed in the following subsections with example.

4.4.2.2 Modulus of a Counter

The number of various logic states that a counter passes through before returning to the beginning state and starting the count sequence again is known as the modulus (MOD).

number) of the counter. The modulus of an n -bit counter is the number of states it counts through without skipping any. As we can see, such counters have a modulus that is a power of two that is integral, i.e. 2, 4, 8, 16, and so on. To reach a modulus of less than 2^n , these can be changed with the use of further combinational logic. Find the lowest integer m that is both equal to or larger than the desired modulus and also equal to the number of FFs needed to construct a counter with a specified modulus.

4.4.3 Modulo 3 (MOD-3) counter

Since $2^1 < 3 < 2^2$, therefore, minimum two FFs will be required to design MOD-3 counter and three FFs are required using **one hot encoding**. Assume that the counter counts three states 00, 01, and 10, the forth state 11 is not valid. The state diagram and state excitation table for MOD-3 counter is presented here in Fig. 4.18.

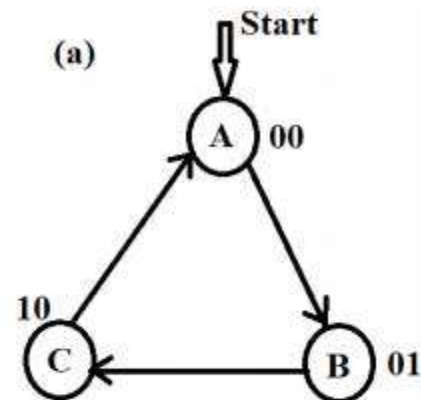
Synchronous counter- In this type counter, all FFs will use the same clock. The procedure to design any sequential circuit is presented in section 4.4.2.1. Table 4.7 presents the state excitation table for all FFs which will be used to create state excitation table for the sequential circuits.

Table 4.7 State excitation table for FFs

S-R FF Excitation Table				D-FF Excitation Table		
Q(t+1)	Q(t)	S	R	Q(t+1)	Q(t)	D
0	0	0	X	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	0
1	1	X	0	1	1	1

JK-FF Excitation Table				T-FF Excitation Table		
Q(t+1)	Q(t)	J	K	Q(t+1)	Q(t)	T
0	0	0	X	0	0	0
0	1	1	X	0	1	1
1	0	X	1	1	0	1
1	1	X	0	1	1	0

For MOD-3 counter, the state diagram, relation between states A, B, C excitation table and Boolean equations derived are in Fig. 4.18. Here we have assigned states A = 00, B = 01, C = 10. We have chosen **minimum bits encoding** to implement the counter, other way is **one hot encoding** in which as many bits as the *number of states* are required and only one bit is HIGH at a time. Here q_1 is MSB and q_0 is LSB for present states and Q_1Q_0 are for next states.



(b)

Present State	Next State
A	B
B	C
C	A

(c)

Present State ($q_1 q_0$)	Next State ($Q_1 Q_0$)	Input to FFs		
		Using $D_1 D_0$	Using JK	
			$J_1 K_1$	$J_0 K_0$
00	01	01	0X	1X
01	10	10	1X	X1
10	00	00	X1	0X

(d) Boolean equations for FF inputs can be written using state excitation table-

$$D_1 = q_0 ; D_0 = q_1' q_0' ;$$

$$J_1 = q_0 ; K_1 = 1 ;$$

$$J_0 = q_1' ; K_0 = 1 ;$$

These expressions can be obtained using k-map.

Fig. 4.18: (a) State diagram, (b) Present state Next state relation and (c) state excitation table (d) Boolean equation for FF inputs for MOD-3 counter.

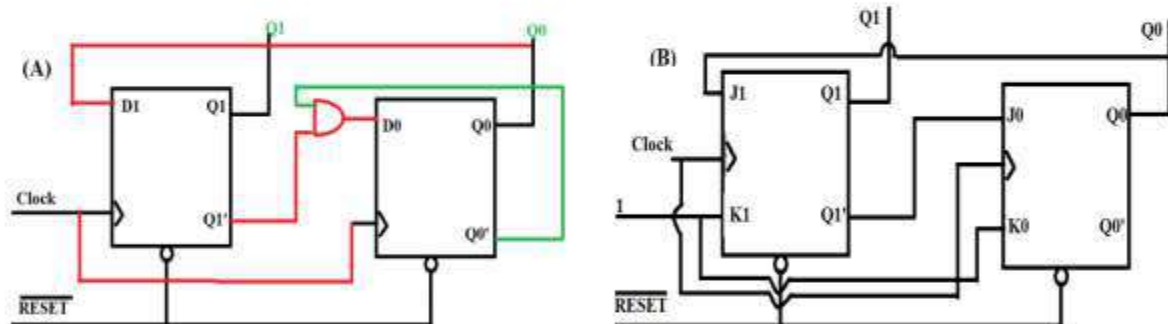


Fig. 4.19: Mod-3 counter (A) using D-FF, (B) using J-K FF.

4.4.3 Modulo 7 (MOD-7) counter

Since $2^2 < 7 < 2^3$, therefore, minimum three FFs will be required to design MOD-7 counter and 7 FFs are required for one hot encoding. Assume that the counter counts seven states from 000, to 110 states, the state 111 is not valid.

Asynchronous Design- This can be designed simply by using asynchronous up-counter. As soon as the counter reaches to the highest state (110 - in this case), the combination (110) can be applied to RESET or clear input which will bring it back to 000 state, as shown in Fig. 4.20.

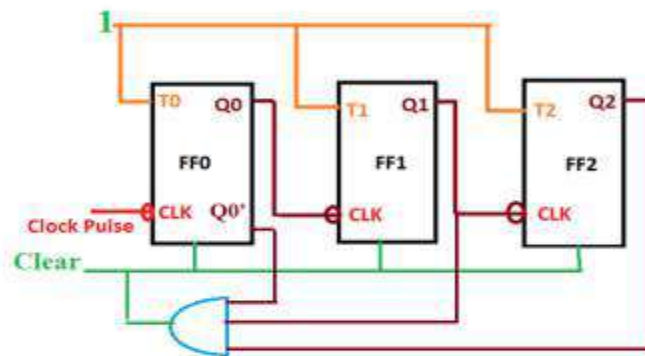


Fig. 4.20: Asynchronous MOD-7 counter

Note- If instead the states used are not continuous then we have to design either using synchronous counter using the given procedure or asynchronous counter with care to skip the intermediate invalid states. For example here instead of 111, assume 101 state is invalid. In this case asynchronous counter can be designed using PRESET and clear. As soon as counter reaches to 100, PRESET the Q_2Q_1 while CLEAR the Q_0 to skip 101 and get 110 state in up-counter.

Locked state: if a sequential circuit once entered in invalid state remain in invalid states only, it is known as **locked state**. This has to be avoided while using don't cares for invalid states. It is explained in example 4.2.

4.4.5 Ring counter

A ring counter is a shift register with each FF's output coupled to its subsequent input, forming a ring. When n FFs are utilised, a single bit pattern is typically cycled to cause the state to repeat every n clock cycles. It is initially configured using PRESET and CLEAR inputs so that just one of its FFs is in state 1, while the rest are in state 0. Ring counter uses one hot encoding; therefore, the number of FFs will be equal to the number of states in the counter. Fig. 4.21 presents logic diagram and truth table for Ring counter. Here FF0 is PRESET to 1 at the starting and all others are CLEARED to 0 using ORI input signal, so starting state is 1000.

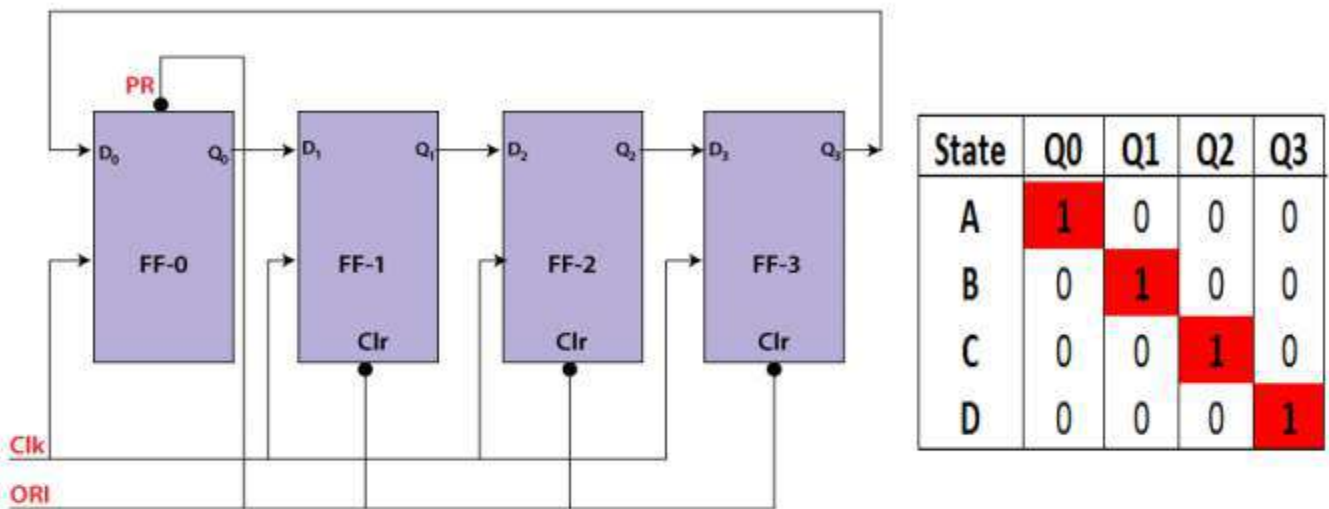


Fig. 4.21: Block Diagram and Truth Table of 4-Bit Ring Counter

4.4.6 Johnson Counter

A modified ring counter is known as a Johnson counter inverts the output of the last step before feeding it back into the initial flop. The register repeatedly cycles through a series of bit patterns that is twice as long as the shift register's length in counter. It may be found in digital-to-analog converters rather frequently. Fig. 4.22 presents logic diagram and truth table for Johnson counter. At starting every FF is CLEARED to 0, so starting state is 0000. In this 4-bit Johnson counter the length is 4 and different states are 8.

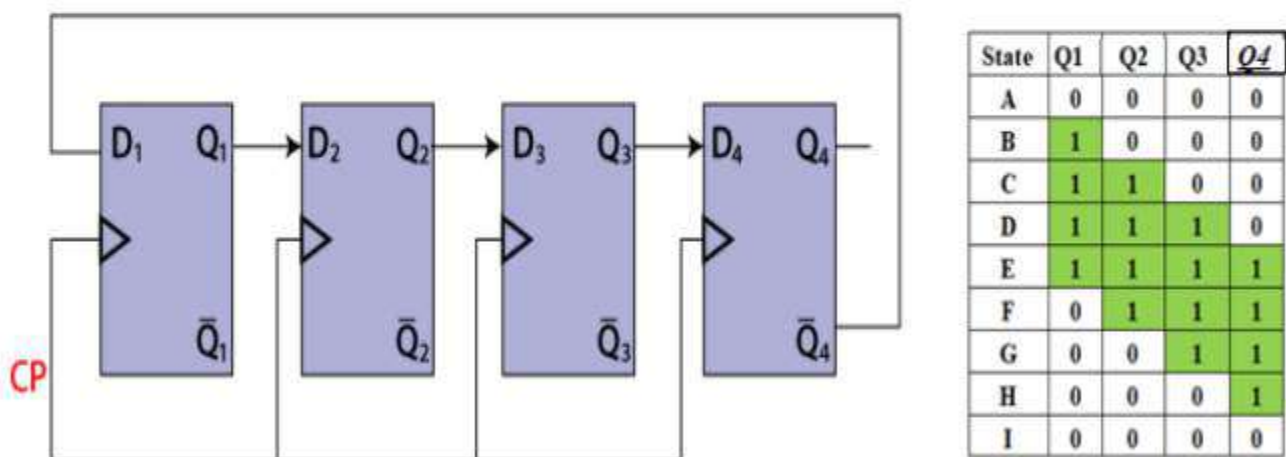


Fig. 4.22: Block Diagram and Truth Table of 4-Bit Johnson Counter

UNIT-I

Computer:

It is a fast-electronic calculating machine which accepts digitized input information and process it according to the instructions stored in the memory and produces results on the output device.

Sequence of instructions stored in the internal storage is called **Computer Program** and internal storage is called **Computer memory**.

Computer Types

Computer types may vary according to the size, cost, Computational Power etc.

Micro Computers

- It is like personal computers, it is widely used in homes, schools, offices etc.
- It provides variety of computing applications such as word processing, photo editing, E-mail and internet etc.
- It is designed to meets the needs of an individual.

Portable Notebook Computers

- These are the compact version of personal computers
- In these all components are integrated as one compact unit.
- It runs on Power Supply or a battery unit.
- It is more expensive than personal computer.

Example: Laptop, tab

Workstations

- It is powerful desktop designed for specialized tasks
- It has more computational power than personal computers.
- It is widely used in engineering applications and in interactive graphics applications.

Main Frames or Enterprise Systems

Mainframe computers are larger and more processing power than some other computers like minicomputer, work stations and personal computers.

These types of computers are used for complex scientific calculations, large data processing applications, military defense control and for complex graphics applications.

Servers

- These computers have large storage unit and faster communication links.
- Servers plays major role in internet communication

Super Computers

- A computer that is considered to be fastest in the world.
- It is used to execute tasks that would take a lot of time for other computers.
- It is used for large scale numerical calculations required in applications such as
- Weather forecasting, aircraft design and simulation

FUNCTIONAL UNITS

A computer consists of 5 functionally independent main parts:

- 1) Input
- 2) Memory
- 3) ALU
- 4) Output &
- 5) Control units.

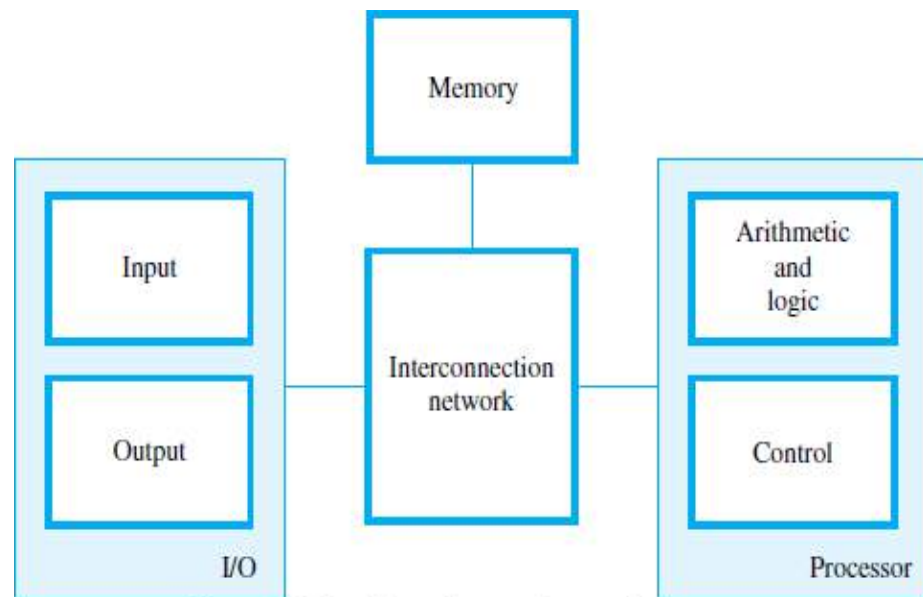


Figure 1.1 Basic functional units of a computer.

Input device accepts the coded information as source program i.e. high-level language. This is either stored in the memory or immediately used by the processor to perform the desired operations. The program stored in the memory determines the processing steps. Basically, the computer converts one source program to an object program. i.e. into machine language.

Finally, the results are sent to the outside world through output device. All of these actions are coordinated by the control unit.

Input unit:

The source program/high level language program/coded information/simply data is fed to a computer through input devices keyboard is a most common type. Whenever a key is pressed, one corresponding word or number is translated into its equivalent binary code over a cable & fed either to memory or processor.

Joysticks, trackballs, mouse, scanners etc are other input devices.

Memory unit:

Its function into store programs and data. It is basically to two types

- **Primary memory**
- **Secondary memory**

1. Primary memory: - Is the one exclusively associated with the processor and operates at the electronics speeds programs must be stored in this memory while they are being executed. The memory contains a large number of semiconductors storage cells. Each capable of storing one bit of information. These are processed in a group of fixed site called **word**.

To provide easy access to a word in memory, a distinct address is associated with each word location. **Addresses** are numbers that identify memory location.

Number of bits in each word is called word length of the computer. Programs must reside in the memory during execution. Instructions and data can be written into the memory or read out under the control of processor.

Memory in which any location can be reached in a short and fixed amount of time after specifying its address is called **random-access memory (RAM)**.

The time required to access one word in called memory access time. Memory which is only readable by the user and contents of which can't be altered is called **read only memory (ROM)** it contains operating system.

Caches are the small fast RAM units, which are coupled with the processor and are often contained on the same IC chip to achieve high performance. Although primary storage is essential it tends to be expensive.

2 Secondary memory: - Is used where large amounts of data & programs have to be stored, particularly information that is accessed infrequently.

Examples: - Magnetic disks & tapes, optical disks (CD-ROM's), floppies etc.,

Arithmetic logic unit (ALU):

Most of the computer operators are executed in ALU of the processor like addition, subtraction, division, multiplication, etc. the operands are brought into the ALU from memory and stored in high speed storage elements called register. Then according to the instructions, the operation is performed in the required sequence.

The control and the ALU are many times faster than other devices connected to a computer system. This enables a single processor to control a number of external devices such as key boards, displays, magnetic and optical disks, sensors and other mechanical controllers.

Output unit:

These actually are the counterparts of input unit. Its basic function is to send the processed results to the outside world.

Examples: Printer, speakers, monitor etc.

Control unit:

It effectively is the nerve centre that sends signals to other units and senses their states. The actual timing signals that govern the transfer of data between input unit, processor, memory and output unit are generated by the control unit

BASIC OPERATIONAL CONCEPTS

- An Instruction consists of 2 parts, 1) Operation code (Opcode) and 2) Operands.

OPCODE	OPERANDS
--------	----------

- The data/operands are stored in memory.
- The individual instruction are brought from the memory to the processor.
- Then, the processor performs the specified operation.
- Let us see a typical instruction

ADD LOCA, R0

- This instruction is an addition operation. The following are the steps to execute the instruction: Step 1: Fetch the instruction from main-memory into the processor.
Step 2: Fetch the operand at location LOCA from main-memory into the processor.
Step 3: Add the memory operand (i.e. fetched contents of LOCA) to the contents of register R0. Step 4: Store the result (sum) in R0.
- The same instruction can be realized using 2 instructions as:

LOAD LOCA, R1

ADD R1, R0

- The following are the steps to execute the instruction:
Step 1: Fetch the instruction from main-memory into the processor.
Step 2: Fetch the operand at location LOCA from main-memory into the register R1. Step 3: Add the content of Register R1 and the contents of register R0.
Step 4: Store the result (sum) in R0.

MAIN PARTS OF PROCESSOR

- The **processor** contains ALU, control-circuitry and many registers.
- The processor contains „n“ general-purpose registers **R₀** through **R_{n-1}**.
- The **IR** holds the instruction that is currently being executed.
- The **control-unit** generates the timing-signals that determine when a given action is to take place.
- The **PC** contains the memory-address of the next-instruction to be fetched & executed.
- During the execution of an instruction, the contents of PC are updated to point to next instruction.
- The **MAR** holds the address of the memory-location to be accessed.
- The **MDR** contains the data to be written into or read out of the addressed location.
- MAR and MDR facilitates the communication with memory
- **IR**: Instruction Register,
- **PC**: Program counter

(**MAR**: Memory Address Register, **MDR**: Memory Data Register)

STEPS TO EXECUTE AN INSTRUCTION

- 1) The address of first instruction (to be executed) gets loaded into PC.
- 2) The contents of PC (i.e. address) are transferred to the MAR & control-unit issues Read signal to memory.
- 3) After certain amount of elapsed time, the first instruction is read out of memory and placed into MDR.
- 4) Next, the contents of MDR are transferred to IR. At this point, the instruction can be decoded & executed.
- 5) To fetch an operand, its address is placed into MAR & control-unit issues Read signal. As a result, the operand is transferred from memory into MDR, and then it is transferred from MDR to ALU.
- 6) Likewise required number of operands is fetched into processor.
- 7) Finally, ALU performs the desired operation.
- 8) If the result of this operation is to be stored in the memory, then the result is sent to the MDR.
- 9) The address of the location where the result is to be stored is sent to the MAR and a Write cycle is initiated.
- 10) At some point during execution, contents of PC are incremented to point to next instruction in the program.

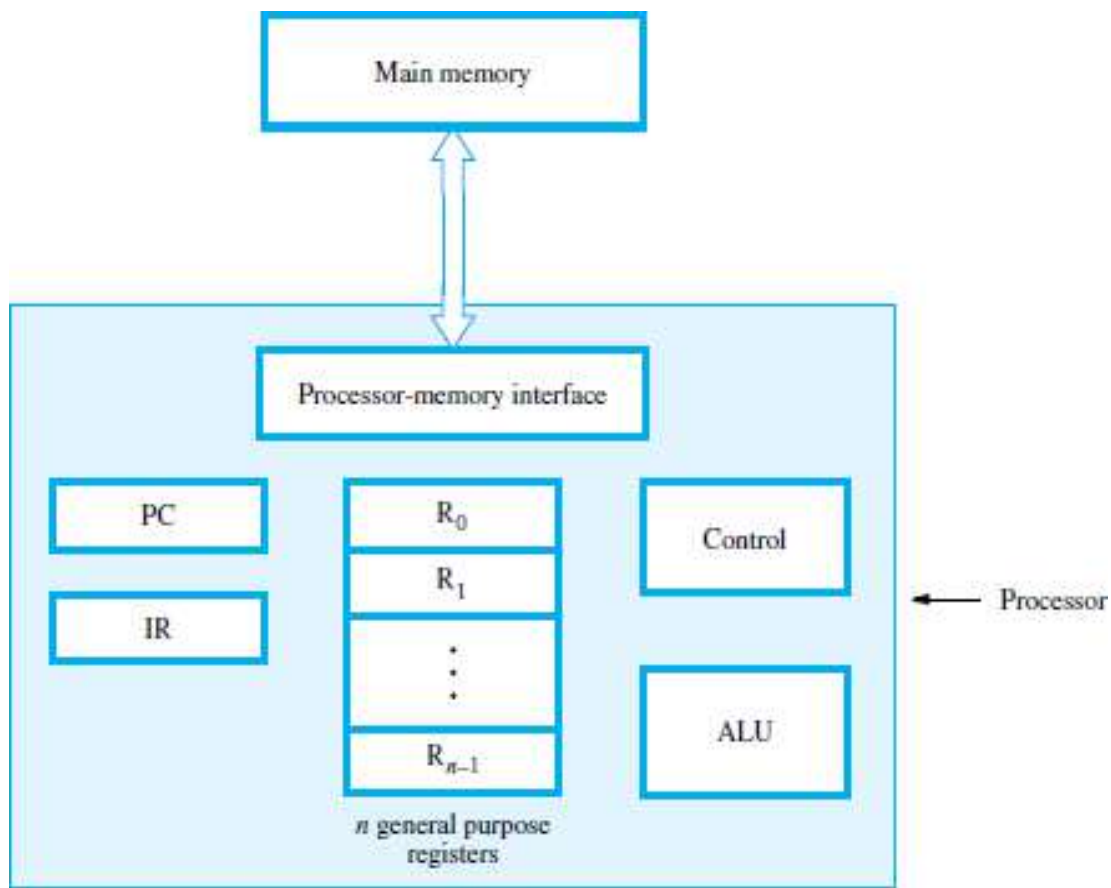


Figure 1.2 Connection between the processor and the main memory.

BUS STRUCTURE

- A bus is a group of lines that serves as a connecting path for several devices.
- A bus may be lines or wires.
- The lines carry data or address or control signal.
- There are 2 types of Bus structures: 1) Single Bus Structure and 2) Multiple Bus Structure.

1) Single Bus Structure

- Because the bus can be used for only one transfer at a time, only 2 units can actively use the bus at any given time.
- Bus control lines are used to arbitrate multiple requests for use of the bus.
- **Advantages:**
 - 1) Low cost &
 - 2) Flexibility for attaching peripheral devices.

2) Multiple Bus Structure

- Systems that contain multiple buses achieve more concurrency in operations.

- Two or more transfers can be carried out at the same time.
- **Advantage:** Better performance.
- **Disadvantage:** Increased cost.

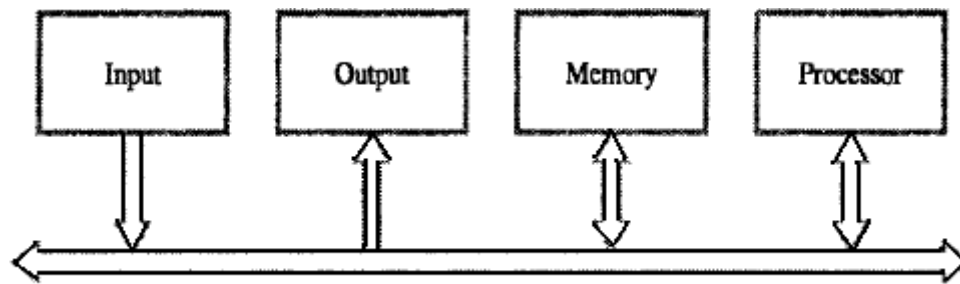


Figure 1.3 Single-bus structure.

- The devices connected to a bus vary widely in their speed of operation.
- To synchronize their operational-speed, buffer-registers can be used.
- **Buffer Registers**
 - are included with the devices to hold the information during transfers.
 - prevent a high-speed processor from being locked to a slow I/O device during data transfers.

SOFTWARE

If a user wants to enter and run an application program, he/she needs a System Software. System Software is a collection of programs that are executed as needed to perform functions such as:

- Receiving and interpreting user commands
- Entering and editing application programs and storing them as files in secondary storage devices
- Running standard application programs such as word processors, spread sheets, games etc...

Operating system - is key system software component which helps the user to exploit the below underlying hardware with the program.

USER PROGRAM and OS ROUTINE INTERACTION

Let's assume computer with 1 processor, 1 disk and 1 printer and application program is in machine code on disk. The various tasks are performed in a coordinated fashion, which is called multitasking. $t_0, t_1 \dots t_5$ are the instances of time and the interaction during various instances as given below:

- t_0 : the OS loads the program from disk to memory.
- t_1 : program executes
- t_2 : program accesses disk
- t_3 : program executes some more.
- t_4 : program accesses printer

t5: program terminates

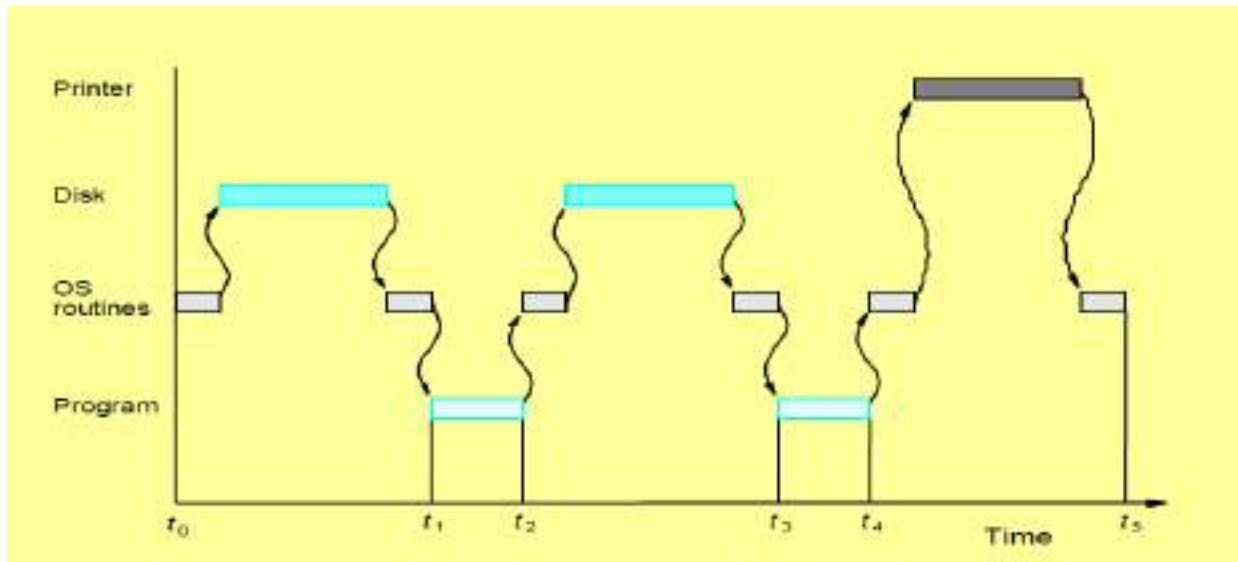


Figure 1.4 : User program and OS routine sharing of the processor

Data Types:

Binary information in digital computers is stored in memory or processor register.

- Registers contain either data or control information .
- Control information is a bit or group of bits used to specify the sequence of command signals needed for data manipulation.
- Data are numbers and other binary-coded information that are operated on .
- Possible data types in registers:
- Numbers used in computations.
- Letters of the alphabet used in data processing
- Other discrete symbols used for specific purposes.
- All types of data, except binary numbers, are represented in binary-coded form .

- A number system of base, or radix, r is a system that uses distinct symbols for r digits.
- Numbers are represented by a string of digit symbols.
- The string of digits 724.5 represents the quantity
$$7 \times 10^2 + 2 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1}$$

The string of digits 101101 in the binary number system represents the quantity.

$$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 45$$

$$(101101)_2 = (45)_{10}$$

We will also use the octal (radix 8) and hexadecimal (radix 16) number systems

GENERATION OF COMPUTERS :Development of technologies used to fabricate the processors, memories and I/O units of the computers has been divided into various generations as given below:

- First generation
- Second generation
- Third generation
- Fourth generation
- Beyond the fourth generation

Computer Organization First generation: 1946 to 1955: Computers of this generation used Vacuum Tubes. The computers were built using stored program concept. Ex: ENIAC, EDSAC, IBM 701. Computers of this age typically used about ten thousand vacuum tubes. They were bulky in size had slow operating speed, short life time and limited programming facilities.

Second generation: 1955 to 1965: Computers of this generation used the germanium transistors as the active switching electronic device. Ex: IBM 7000, B5000, IBM 1401. Comparatively smaller in size About ten times faster operating speed as compared to first generation vacuum tube based computers. Consumed less power, had fairly good reliability. Availability of large memory was an added advantage.

Third generation: 1965 to 1975: The computers of this generation used the Integrated Circuits as the active electronic components. Ex: IBM system 360, PDP minicomputer etc. They were still smaller in size. They had powerful CPUs with the capacity of executing 1 million instructions per second (MIPS). Used to consume very less power consumption. Fourth generation: 1976 to 1990: The computers of this generation used the LSI chips like microprocessor as their active electronic element. HCL horizon III, and WIPRO'S Uniplex+ HCL's Busybee PC etc. They used high speed microprocessor as CPU. They were more user friendly and highly reliable systems. They had large storage capacity disk memories.

Beyond Fourth Generation: 1990 onwards: Specialized and dedicated VLSI chips are used to control specific functions of these computers. Modern Desktop PC's, Laptops or Notebook Computers.

PERFORMANCE

The most important measure of the performance of a computer is how quickly it can execute programs. The speed with which a computer executes program is affected by the design of its hardware. For best performance, it is necessary to design the compiler, the machine instruction set, and the hardware in a coordinated way. The total time required to execute the program is elapsed time is a measure of the performance of the entire computer system. It is affected by the speed of the processor, the disk and the printer. The time needed to execute a instruction is called the processor time.

Just as the elapsed time for the execution of a program depends on all units in a computer system, the processor time depends on the hardware involved in the execution of individual machine instructions. This hardware comprises the processor and the memory which are usually connected by the bus. The pertinent parts of the fig. c is repeated in fig. d which includes the cache memory as part of the processor unit.

Let us examine the flow of program instructions and data between the memory and the processor. At the start of execution, all program instructions and the required data are stored in the main memory. As the execution proceeds, instructions are fetched one by one over the bus into the processor, and a copy is placed in the cache later if the same instruction or data item is needed a second time, it is read directly from the cache. The processor and relatively small cache memory can be fabricated on a single IC chip. The internal speed of performing the basic steps of instruction processing on chip is very high and is considerably faster than the speed at which the instruction and data can be fetched from the main memory. A program will be executed faster if the movement of instructions and data between the main memory and the processor is minimized, which is achieved by using the cache.

For example:- Suppose a number of instructions are executed repeatedly over a short period of time as happens in a program loop. If these instructions are available in the cache, they can be fetched quickly during the period of repeated use. The same applies to the data that are used

repeatedly. **Processor clock:** Processor circuits are controlled by a timing signal called clock. The clock designer the regular time intervals called clock cycles. To execute a machine instruction the processor divides the action to be performed into a sequence of basic steps that each step can be completed in one clock cycle. The length P of one clock cycle is an important parameter that affects the processor performance. Processor used in today's personal computer and work station have a clock rates that range from a few hundred million to over a billion cycles per second.

Basic performance equation: We now focus our attention on the processor time component of the total elapsed time. Let „T“ be the processor time required to execute a program that has been prepared in some high-level language. The compiler generates a machine language object program that corresponds to the source program. Assume that complete execution of the program requires the execution of N machine cycle language instructions. The number N is the actual number of instruction execution and is not necessarily equal to the number of machine cycle instructions in the object program. Some instruction may be executed more than once, which in the case for instructions inside a program loop others may not be executed all, depending on the input data used. Suppose that the average number of basic steps needed to execute one machine cycle instruction is S, where each basic step is completed in one clock cycle. If clock rate is „R“ cycles per second, the program execution time is given by $T = N * S / R$ this is often referred to as the basic performance equation. We must emphasize that N, S & R are not independent parameters changing one may affect another. Introducing a new feature in the design of a processor will lead to improved performance only if the overall result is to reduce the value of T.

Performance measurements: It is very important to be able to access the performance of a computer, comp designers use performance estimates to evaluate the effectiveness of new features. The previous argument suggests that the performance of a computer is given by the execution time T, for the program of interest. Inspite of the performance equation being so simple, the evaluation of „T“ is highly complex. Moreover the parameters like the clock speed and various architectural features are not reliable indicators of the expected performance. Hence measurement of computer performance using bench mark programs is done to make comparisons possible, standardized programs must be used.

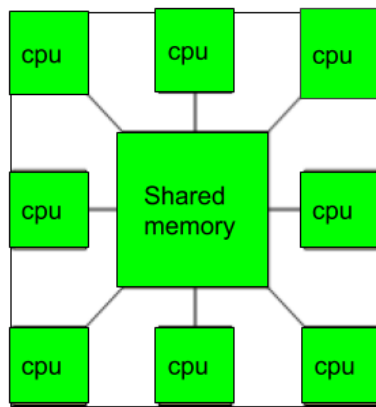
The performance measure is the time taken by the computer to execute a given bench mark. Initially some attempts were made to create artificial programs that could be used as bench mark programs. But synthetic programs do not properly predict the performance obtained when real application programs are run. A non profit organization called SPEC- system performance evaluation corporation selects and publishes bench marks. The program selected range from game playing, compiler, and data base applications to numerically intensive programs in astrophysics and quantum chemistry. In each case, the program is compiled under test, and the running time on a real computer is measured. The same program is also compiled and run on one computer selected as reference. The „SPEC“ rating is computed as follows.

$$\text{SPEC rating} = \frac{\text{Running time on the reference computer}}{\text{Running time on the computer under test}}$$

If the SPEC rating = 50

Multiprocessor and Multicomputer:

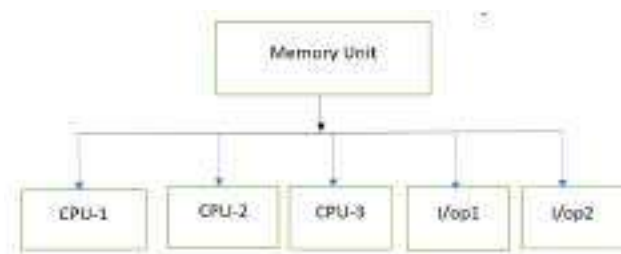
Multiprocessor: A Multiprocessor is a computer system with two or more central processing units (CPUs) share full access to a common RAM. The main objective of using a multiprocessor is to boost the system's execution speed, with other objectives being fault tolerance and application matching. There are two types of multiprocessors, one is called shared memory multiprocessor and another is distributed memory multiprocessor. In shared memory multiprocessors, all the CPUs shares the common memory but in a distributed memory multiprocessor, every CPU has its own private memory.



The interconnection among two or more processor and shared memory is done with three methods:

- 1)Time shared common bus
- 2)Multiport memories
- 3)Crossbar switch network

1)Time shared common bus



As the name itself indicates, in this method, it contains a single shared bus through which all processor & memory unit can be communicated.

Consider CPU-1 is interacting with memory unit using common shared bus; in that case, all other processors must be idle as we have only one bus to communicate.

Advantage:

Simple to implement.

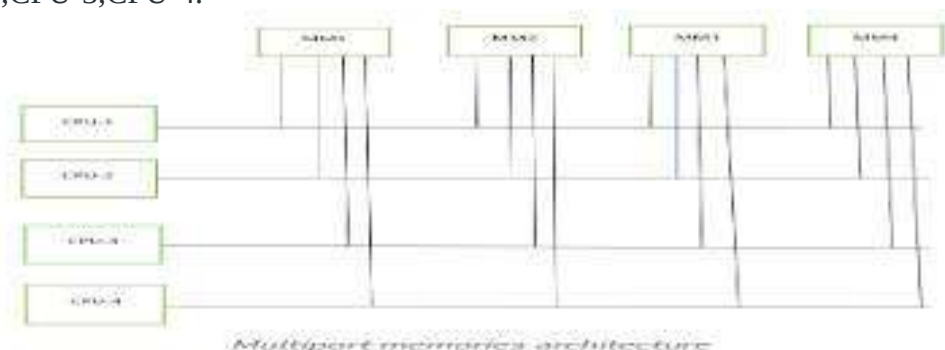
Due to single common bus, the cost to implement is very less.

Disadvantage:

Data transfer rate is slow.

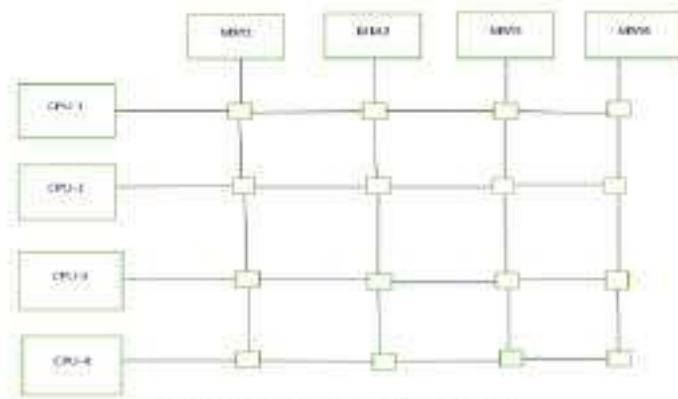
Multiport memories:

Unlike in the shared common bus method, hence it contains separate bus for each processor to communicate with the memory module. Suppose CPU-1 wants to interact with memory module 1; then port mm1 is enabled. Similarly, CPU-4 wants to interact with memory module 4; then port mm4 is enabled. Hence all the processes can be communicated parallelly. If more than one CPU requests for the same time memory module, priority will be given in the order of CPU-1, CPU-2, CPU-3, CPU-4.



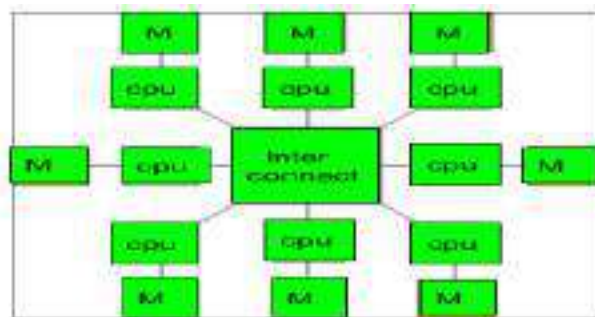
Crossbar switch network:

Here instead multiport unlike in multiport memories, a switch will be installed between memory unit and CPU. Switch is responsible for whether to pass the request to a particular memory module or not based on the request made for



Illustrate cross-bar switch network

Multicomputer: A multicomputer system is a computer system with multiple processors that are connected together to solve a problem. Each processor has its own memory and it is accessible by that particular processor and those processors can communicate with each other via an interconnection network.



As the multicomputer is capable of messages passing between the processors, it is possible to divide the task between the processors to complete the task. Hence, a multicomputer can be used for distributed computing. It is cost effective and easier to build a multicomputer than a multiprocessor. **Difference between multiprocessor and Multicomputer:**

1. Multiprocessor is a system with two or more central processing units (CPUs) that is capable of performing multiple tasks whereas a multicomputer is a system with multiple processors that are attached via an interconnection network to perform a computation task.
2. A multiprocessor system is a single computer that operates with multiple CPUs whereas a multicomputer system is a cluster of computers that operate as a singular computer.
3. Construction of multicomputer is easier and cost effective than a multiprocessor.
4. In multiprocessor system, program tends to be easier whereas in multicomputer system, program tends to be more difficult.
5. Multiprocessor supports parallel computing, Multicomputer supports distributed computing.

Features	Multiprocessor System	Multicomputer System
Definition	It is a system with multiple processors that enables programs to be processed at the same time.	It is a collection of processors linked by a communication network that collaborate to solve a computation task.
Programming	It is easy to program.	It is complex to program.
Computing	It supports parallel computing.	It supports distributed computing.

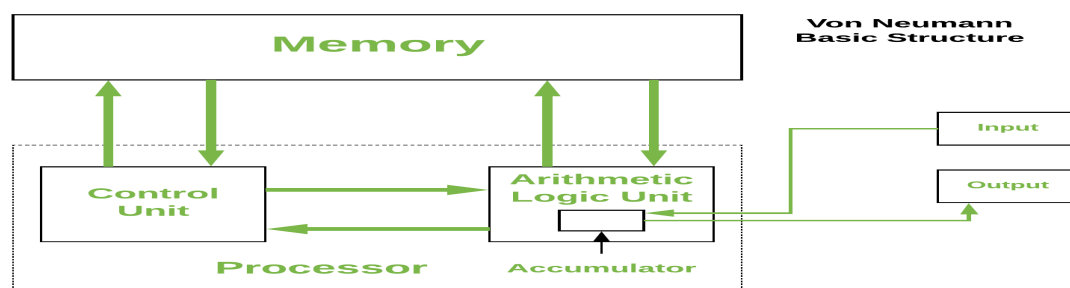
Construction	It is easy and less expensive to develop.	It is complex and expensive to develop.
Type of network	It is a type of dynamic network.	It is a type of static network.
Communication between processing elements	It requires proper communication between the processing components and memory for successful resource allocation.	There is no interaction between processor units or memory resources.
Another name	It is also known as the tightly coupled system.	It is also known as loosely coupled systems.
Example	Sequent symmetry S-81 is an example of a multiprocessor system.	Message-passing multicomputer is an example of the multicomputer system.
Execution	It may execute the programs very quickly.	It may run slowly.

Von Neumann architecture:

Von-Neumann computer architecture design was proposed in 1945. It was later known as Von-Neumann architecture. Historically there have been 2 types of Computers:

1. **Fixed Program Computers** – Their function is very specific and they couldn't be reprogrammed, e.g. Calculators.
2. **Stored Program Computers** – These can be programmed to carry out many different tasks, applications are stored on them, hence the name.

Modern computers are based on a stored-program concept introduced by John Von Neumann. In this stored-program concept, programs and data are stored in the same memory. This novel idea meant that a computer built with this architecture would be much easier to reprogram. The basic structure is like this,



It is also known as **ISA** (Instruction set architecture) computer and is having three basic units:

1. The Central Processing Unit (CPU)
2. The Main Memory Unit
3. The Input/Output Device Let's consider them in detail.

1. Central Processing Unit-

The central processing unit is defined as the it is an electric circuit used for the executing the instruction of computer program.

It has following major components:

1. **Control Unit(CU)**
 2. **Arithmetic and Logic Unit(ALU)**
 3. **variety of Registers**
- **Control Unit**– A control unit (CU) handles all processor control signals. It directs all input and output flow, fetches code for instructions, and controls how data moves around the system.
 - **Arithmetic and Logic Unit(ALU)**– The arithmetic logic unit is that part of the CPU that handles all the calculations the CPU may need, e.g. Addition, Subtraction, Comparisons. It performs Logical Operations, Bit Shifting Operations, and Arithmetic operations.

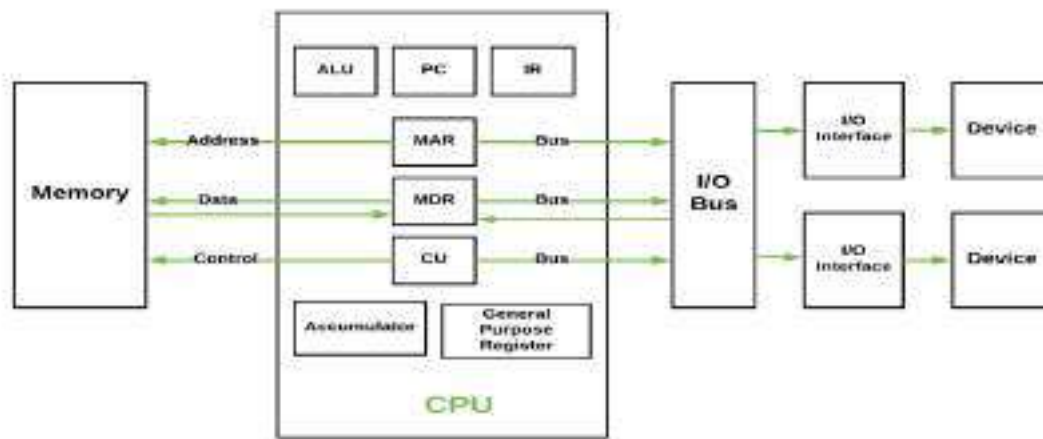


Figure – Basic CPU structure, illustrating ALU

- **Registers** – Registers refer to high-speed storage areas in the CPU. The data processed by the CPU are fetched from the registers. There are different types of registers used in architecture :-
 1. **Accumulator:** Stores the results of calculations made by ALU. It holds the intermediate of arithmetic and logical operations. It acts as a temporary storage location or device.
 2. **Program Counter (PC):** Keeps track of the memory location of the next instructions to be dealt with. The PC then passes this next address to the Memory Address Register (MAR).
 3. **Memory Address Register (MAR):** It stores the memory locations of instructions that need to be fetched from memory or stored in memory.
- Memory Data Register (MDR):** It stores instructions fetched from memory or any data that is to be transferred to, and stored in, memory.
- 5. **Current Instruction Register (CIR):** It stores the most recently fetched instructions while it is waiting to be coded and executed.
- 6. **Instruction Buffer Register (IBR):** The instruction that is not to be executed immediately is placed in the instruction buffer register IBR.
- **Buses** – Data is transmitted from one part of a computer to another, connecting all major internal components to the CPU and memory, by the means of Buses. Types:
 1. **Data Bus:** It carries data among the memory unit, the I/O devices, and the processor.
 2. **Address Bus:** It carries the address of data (not the actual data) between memory and processor.
 3. **Control Bus:** It carries control commands from the CPU (and status signals from other devices) in order to control and coordinate all the activities within the computer.
- **Input / Output Devices** – Program or data is read into main memory from the *input device* or secondary storage under the control of CPU input instruction. *Output devices* are used to output information from a computer. If some results are evaluated by the computer and it is stored in the computer, then with the help of output devices, we can present them to the user.

UNIT-IVCOMPUTER ARITHMETIC

① Topics

②

Addition and Subtraction :-

We can perform addition and subtraction of two binary numbers in three different ways:

- 1) Signed - Magnitude Representation
- 2) Signed - 1's Complement "
- 3) Signed - 2's Complement "

The Signed - 2's complement representation is used for performing arithmetic operations. The signed - magnitude representation is used for floating-point representation.

Addition and Subtraction with Signed - Magnitude data :-

Here, we are going to consider the magnitude of any two numbers i.e., A and B. There are eight different operations which are listed in below:

Operation	Add magnitudes	Subtract Magnitudes		
		When $A > B$	When $A < B$	When $A = B$
$(+A) + (+B)$	$+(A+B)$			
$(+A) + (-B)$		$+(A-B)$	$-(B-A)$	$+(A-B)$
$(-A) + (+B)$		$-(A-B)$	$+(B-A)$	$+(A-B)$
$(-A) + (-B)$	$-(A+B)$			
$(+A) - (+B)$		$+(A-B)$	$-(B-A)$	$+(A-B)$
$(+A) - (-B)$	$+(A+B)$			
$(-A) - (+B)$	$-(A+B)$			
$(-A) - (-B)$		$-(A-B)$	$+(B-A)$	$+(A-B)$

The columns in the table show the actual operation to be performed with the magnitude of the numbers. The last column is needed to prevent a negative zero.

Algorithm :-

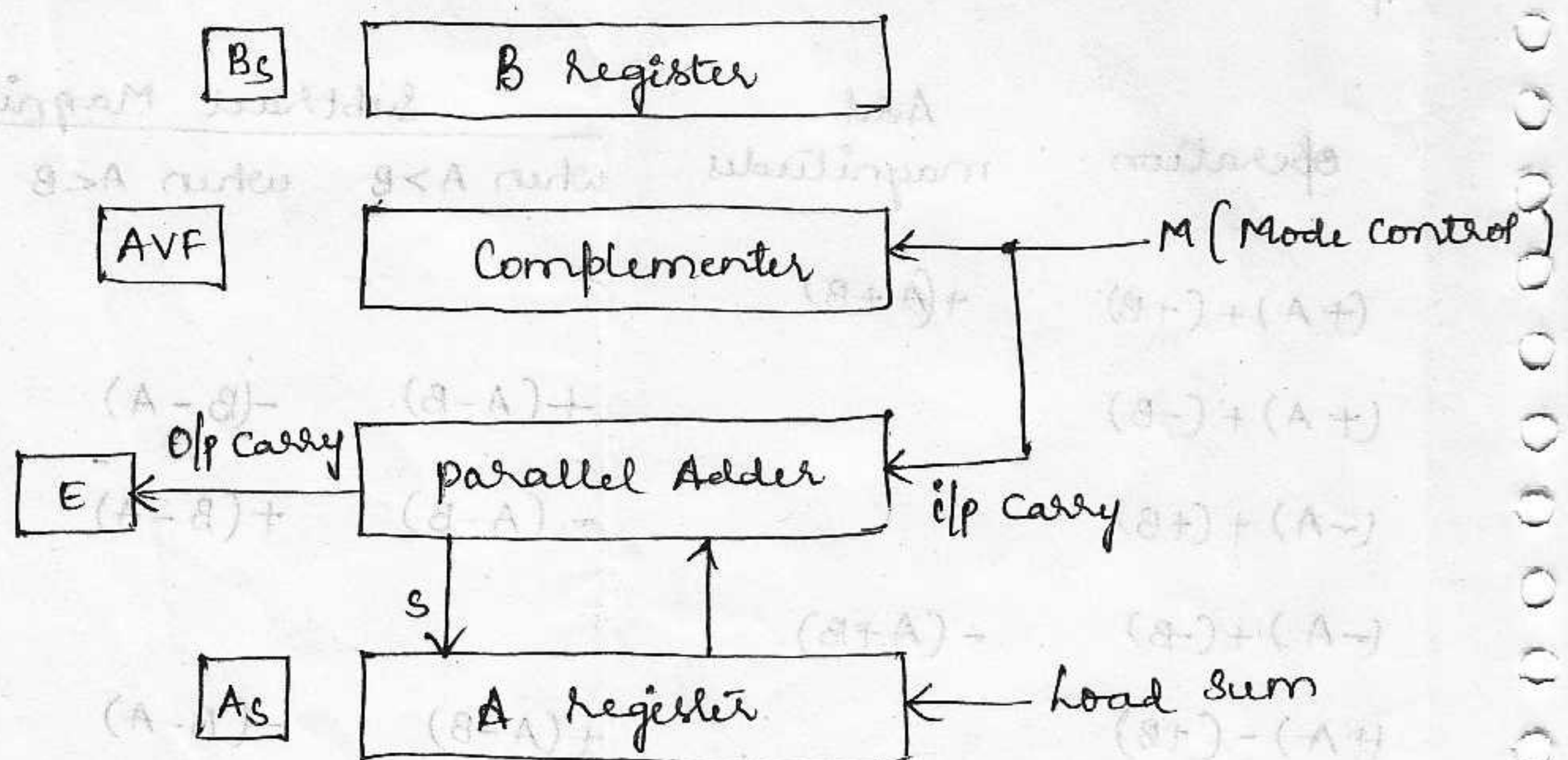
When the signs of A and B are identical, add the two magnitudes and attach the sign of A to the result.

When the signs of A and B are different, compare the magnitudes, subtract the smaller number from the larger.

Hardware Implementation :-

To implement the two arithmetic operations with hardware, it is first necessary that the two numbers be stored in registers.

The h/w implementation for addition & subtraction is shown in fig.



Let A and B be two registers that hold the magnitudes of the numbers and As & Bs be two flip-flops that hold the corresponding signs. The result of the

operation may be transferred to a third register. or the result is transferred into A or A_5 .

First, a parallel-adder is needed to perform micro operation $A+B$. Second a Comparator circuit is needed to establish if $A > B$, $A = B$ or $A < B$. The third, two parallel-subtractor circuits are needed to perform the microoperations, $A-B$ & $B-A$.

The block diagram consists of registers A & B and sign flip-flops A_5 & B_5 . Subtraction is done by adding A to the 2's Complement of B. The o/p carry is transferred to 'E'. The add overflow flip-flop (AVF) holds the overflow bit when A & B are added.

The addition of A plus B is done through the parallel Adder. The S (Sum) o/p of the adder is transferred to A-register. The M (Mode control) signal is also applied to the i/p carry of the adder. When $M=0$, the o/p of B is transferred to the adder, the i/p carry is '0' and the o/p of the adder is equal to the sum $A+B$. When $M=1$, the 1's Complement of B is applied to the adder, the i/p carry is 1,
$$o/p\ S = A + \bar{B} + 1$$

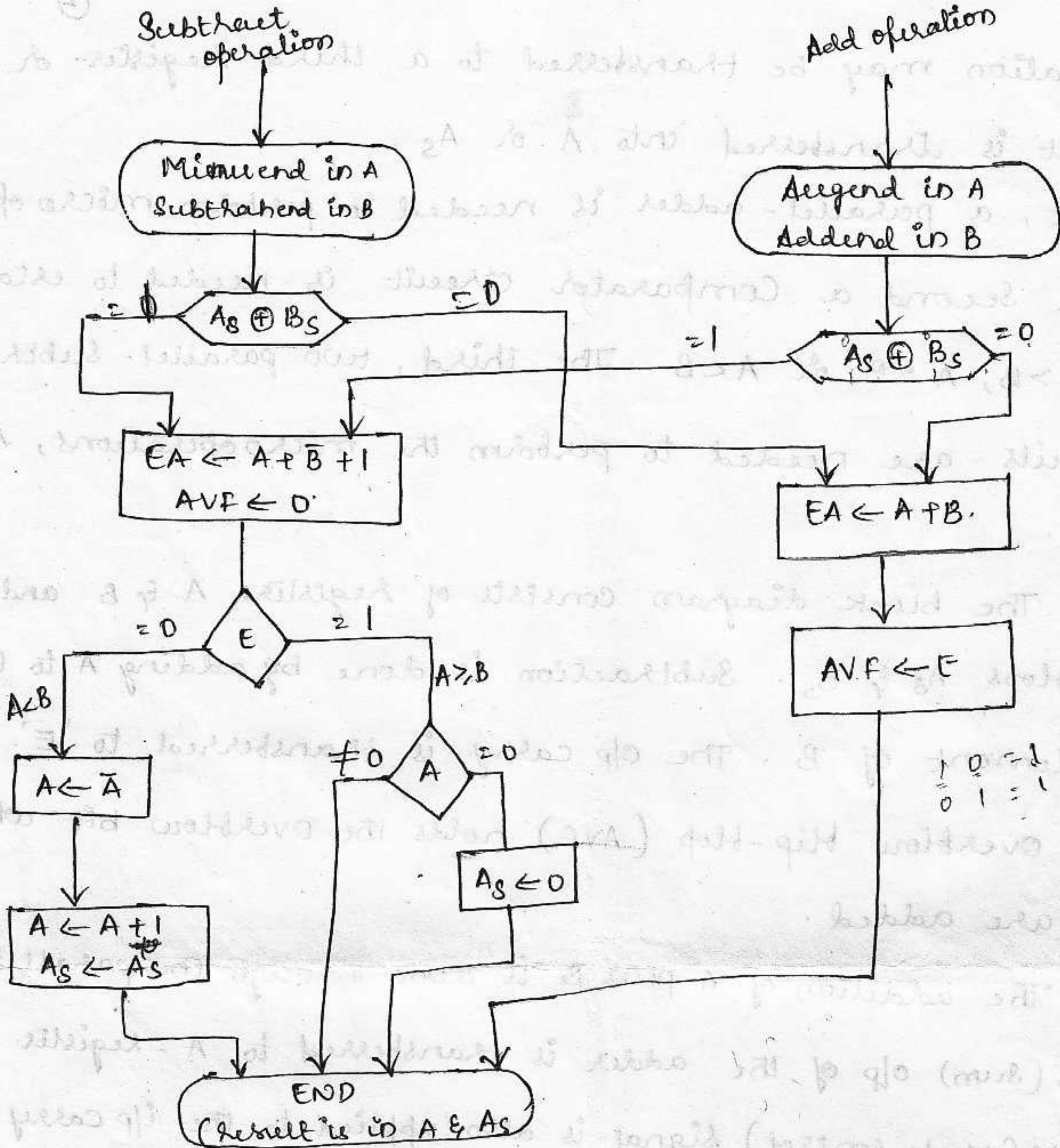
Hardware Algorithm:-

The flowchart for the h/w algorithm is shown in fig.

The two signs A_5 & B_5 are compared by EX-OR gate. If the o/p of the gate is 0, the signs are identical. If the o/p of the gate is 1, the signs are different. For an add operation, the

Subtract operation

Add operation



Identical signs dictate that the magnitudes be added. For a subtract operation, different signs dictate that the magnitudes be added. The magnitudes are added with $EA \leftarrow A + B$, where EA is a register that combines E & A .

The two magnitudes are subtracted if the signs are different for an add operation or identical for a subtract operation. No overflow can occur if the numbers are subtracted so AVF is to '0'.

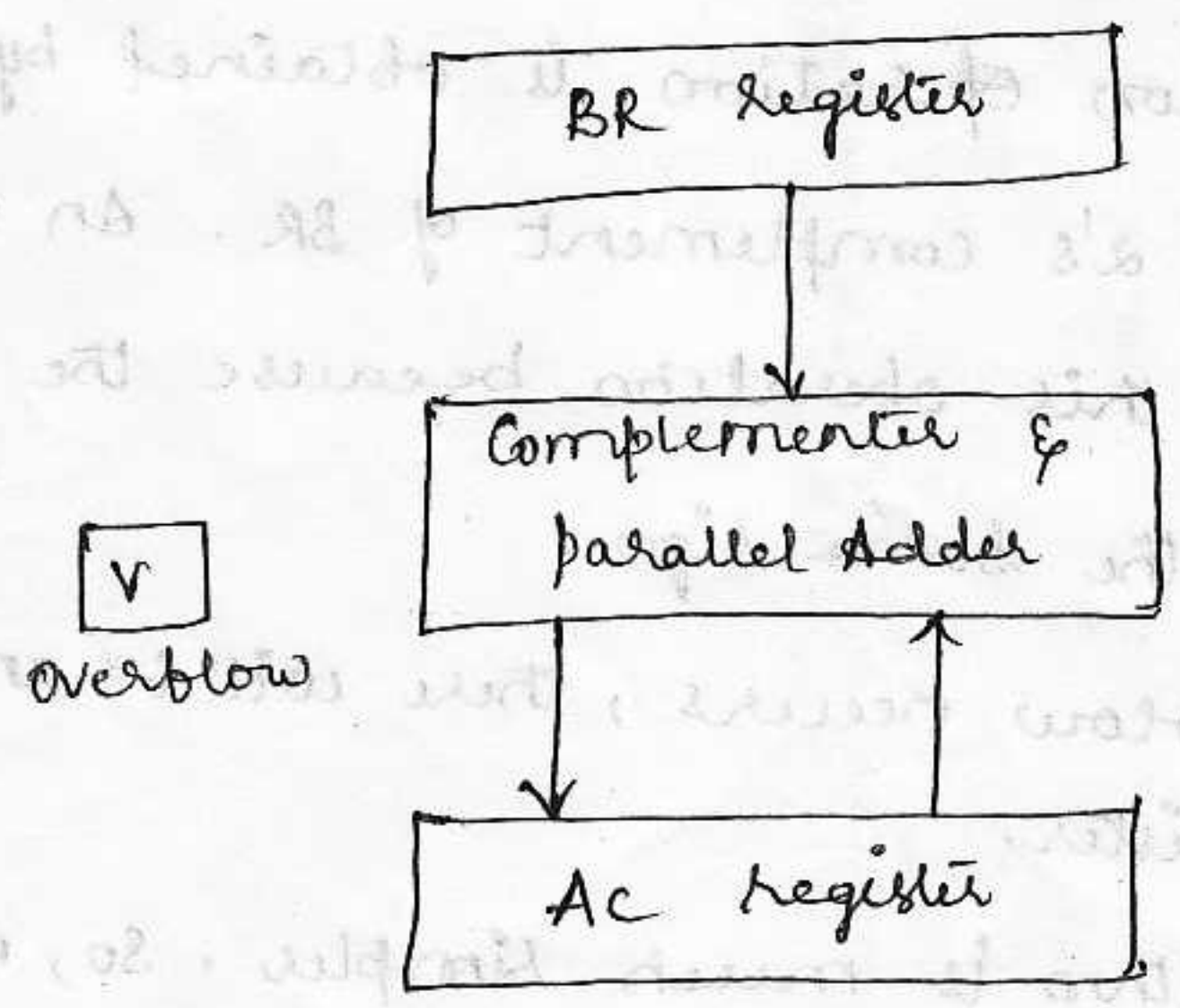
If $E = 1$, then the condition is $A \geq B$, if A is '0' then the result is correct result. If $E = 0$, then the condition is

Condition is $A < B$. For this, we are going to consider the 2's Complement of the value in A. This operation can be done ^{one} microoperation $A \leftarrow \bar{A} + 1$. If the sign of the result is as same as the sign of A, so no change in A_s is required. When $A < B$, the sign of the result is the Complement of the original sign of A. The Complement of A_s is required to get the correct sign. The final result is found in register A & its sign in A_s .

Addition & Subtraction with signed 2's complement data :-

When two numbers of 'n' digits are added and the sum occupies $n+1$ digits, then overflow is occurred. An overflow can be detected by inspecting the last two carries out of the addition. When the two carries are applied to an ex-or gate, the overflow is detected when the o/p of the gate is equal to 1.

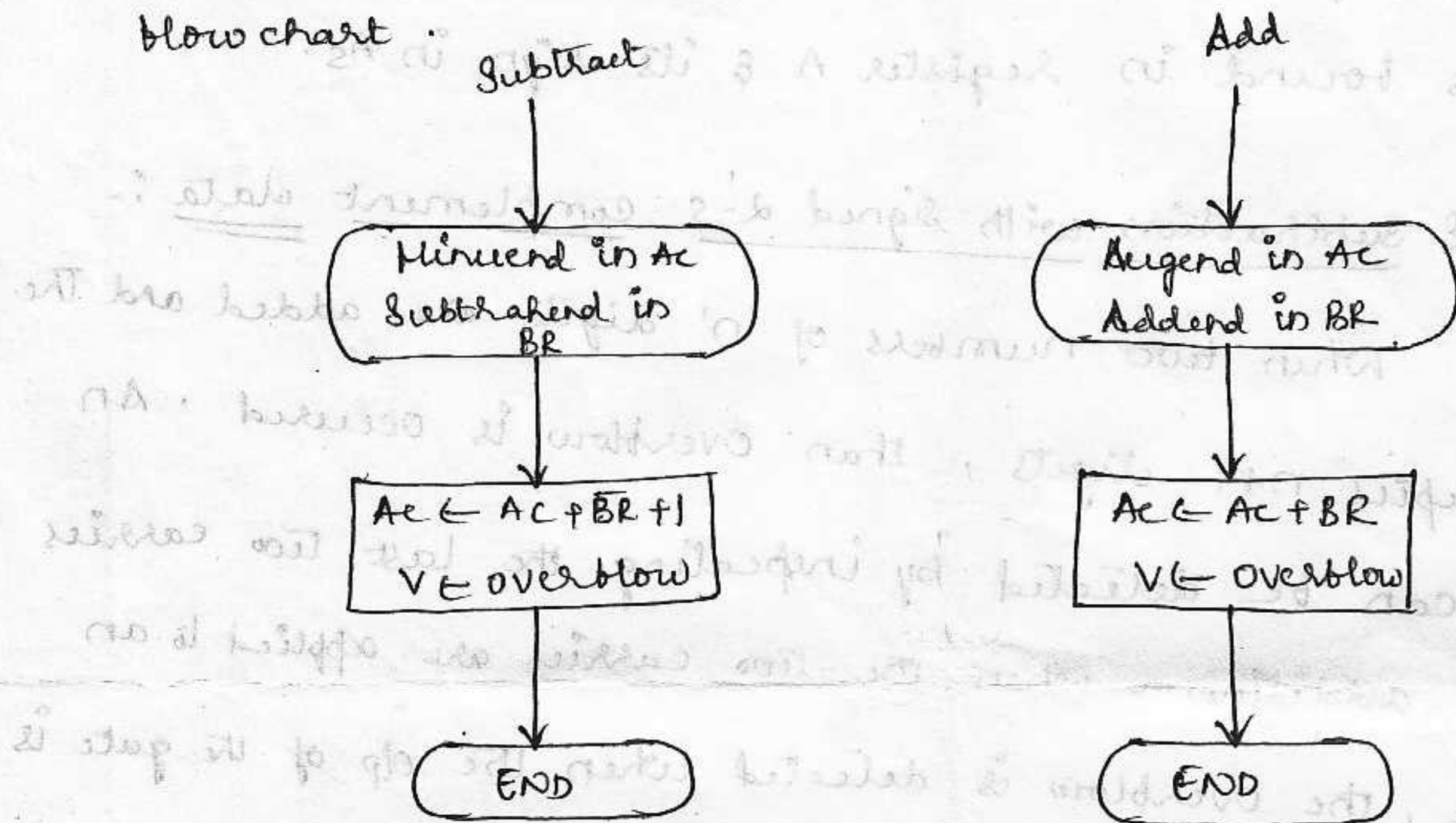
The req. configuration for the h/w implementation is shown in fig.





The leftmost bit in A_c & B_R represent the sign bits of the number. The two sign bits are added or subtracted together with the other bits in the Complementer & parallel Adder. The Overflow flip-flop V is set to 1 if there is an overflow.

The algorithm for adding and subtracting two binary numbers in signed - 2's complement representation is shown in flowchart.



The sum is obtained by adding the contents of A_c & B_R . The Overflow bit is set to 1 if the ex-or of the last two carries is set to 1.

The subtraction operation is obtained by adding the content of A_c to the 2's complement of B_R . An overflow must be checked during this operation because the two numbers added could have the same sign.

If an overflow occurs, there will be an erroneous result in the A_c register.

This algorithm is much simpler. So, most computers use the signed-magnitude representation.

Multiplication Algorithms :-

The sign of the product is determined from the signs of the multiplicand and multiplier. If they are like, the sign of the product is +ve, If they are unlike, the sign of the product is -ve.

H/w implementation for Signed-Magnitude Data :-

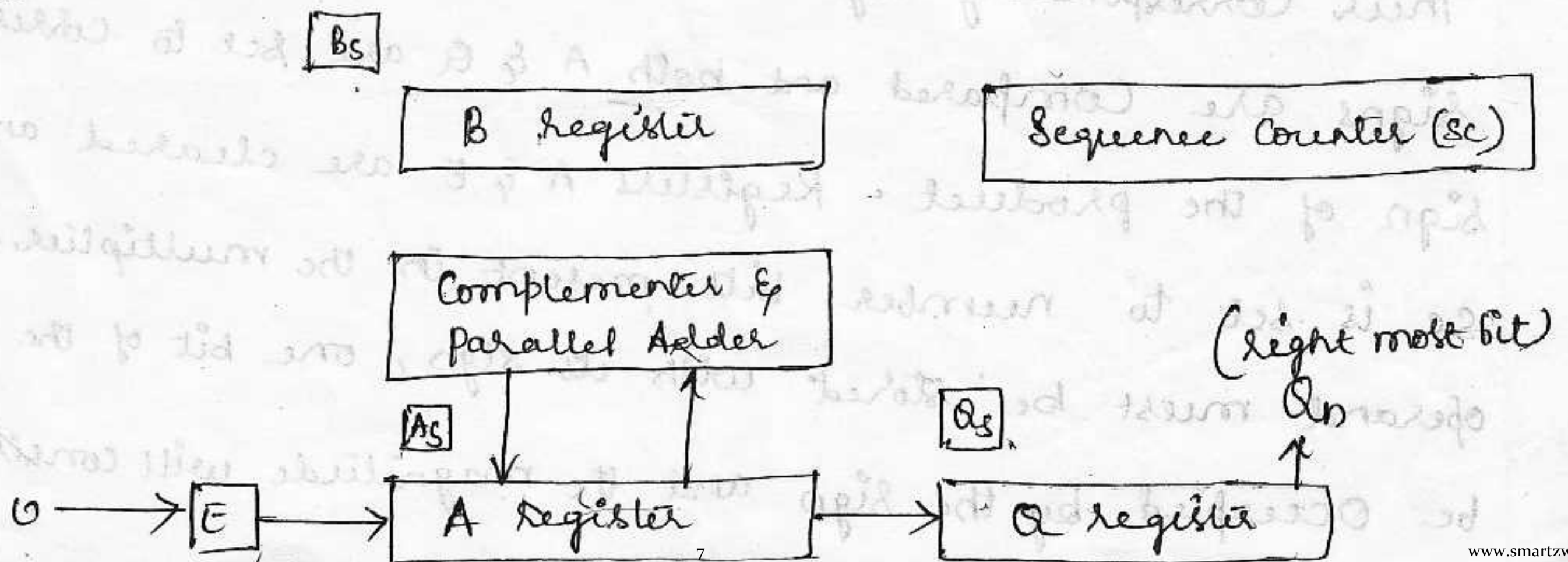
The Multiplication is implemented in a digital computer. First, instead of providing registers to store & add, it is convenient to provide an adder for the summation of only two binary numbers and accumulate the partial products in a register.

Second, instead of shifting the multiplicand to the left, the partial product shifted to right.

Third, when the corresponding bit of the multiplier is 0, there is no need to add all zeros to the partial product.

Ex:
$$\begin{array}{r} 23 \quad 1011 \text{ Multiplicand} \\ 19 \quad \times 1001 \text{ Multiplier} \\ \hline \end{array}$$

H/w & The h/w for multiplication consists is shown in fig.





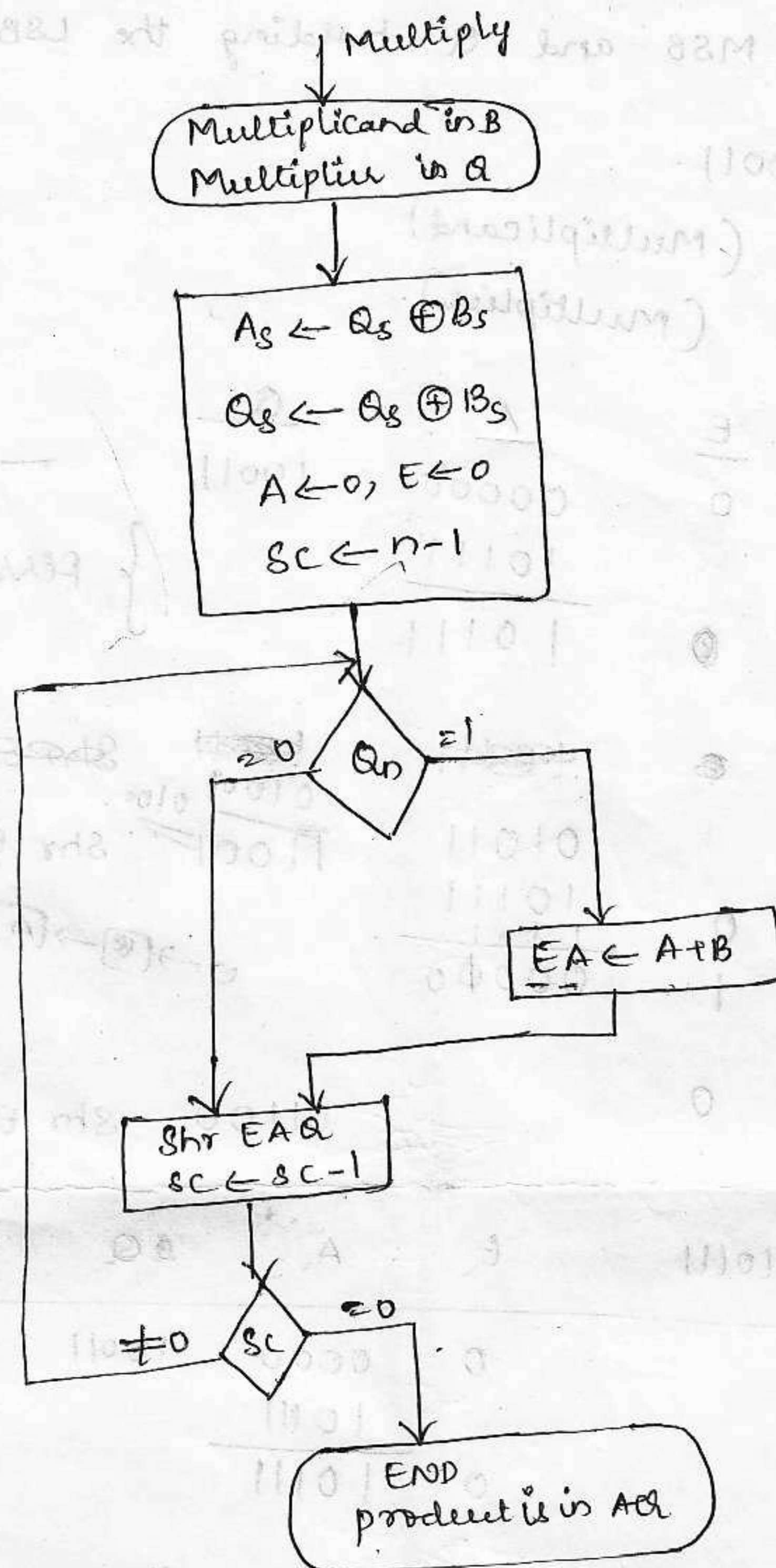
The multiplier stored in the Q register & its sign in Qs. The Sequence Counter (Sc) is initially set to a number equal to the number of bits in the multiplier. The counter is decremented by 1 after forming each partial product. The counter reaches to zero, the product is formed and the process stops.

Initially, the multiplicand is in B reg. and the multiplier is in Q. The sum of A & B forms a partial product which is transferred to the EA register. Both partial product & multiplier are shifted to the right. This shift will be denoted by the statement Shr EAQ . The LSB of A is shifted into MSB of Q. The bit from E is shifted into the MSB of A and '0' is shifted into E. After the shift, one bit of the partial product is shifted into Q, pushing the multiplier bits one position to the right. The right most flip-flop in register Q, designated by Qn, will hold the bit of the multiplier.

How Algorithm:-

The fig. shows the flowchart of the hardware multiply algorithm.

Initially, the multiplicand is in B and the multiplier is in Q. Their corresponding signs are in Bs & Qs respectively. The signs are compared and both A & Q are set to correspond to the sign of the product. Registers A & E are cleared and Sequence Sc is set to number bits present in the multiplier. Since an operand must be stored with its sign, one bit of the word will be occupied by the sign and the magnitude will consist of $n-1$ bits.



After initialization, the low-order bit of the multiplier Q_n is tested. If it is 1, the multiplicand in B is added to the present partial product in A. If it is 0, nothing is done, Register EAA is then shifted once to the right to form the new partial product. The sequence counter is decremented by 1 and its new value is checked. If it is not equal to zero, the process is repeated and if it is equal to zero, the process stops. The final product is available in both A & Q, with

A holding the MSB and Q holding the LSB.

Ex: 10111×10011

B = 10111 (Multiplicand)

Q = 10011 (Multiplier)

B	SC	E	A	Q
10111	5	0	00000	10011
			10111	
		0	10111	

Initial Values

First partial product

4	0	01011	10011	01001	0100	Shr EAA
		10111				
	1	00000				
	0					
				01100	Shr EAA	x

Second partial product

Multiplicand B = 10111

Ans 1 Add

Shr EAA

Ans 1, Add

Shr EAA

Ans 20, Shr EAA

Ans 20, Shr EAA

Ans 21, Add B

8 Shr, EAA

AA = 0110110101

E	A	SA	SC
0	0000	10011	101
	10111		
0	10111		
0	01011	01001	100
	10111		
1	00010		
0	10001	01100	011
0	01000	101100	010
0	00100	01011	001
	10111		
0	11011	01011	001
	01101	10101	000

First partial product

Second partial product

Booth Multiplication Algorithm :-

Booth's algorithm gives a procedure for multiplying binary integers in signed -2's complement representation. It operates on the fact that strings of 0's in the multiplier requires no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as $2^{k+1} - 2^m$.

Eg: The binary number 001110 has a string of 1's from 2^3 to 2^1 .

Here $k=3$, $m=1$. The number can be represented as $2^{k+1} - 2^m$

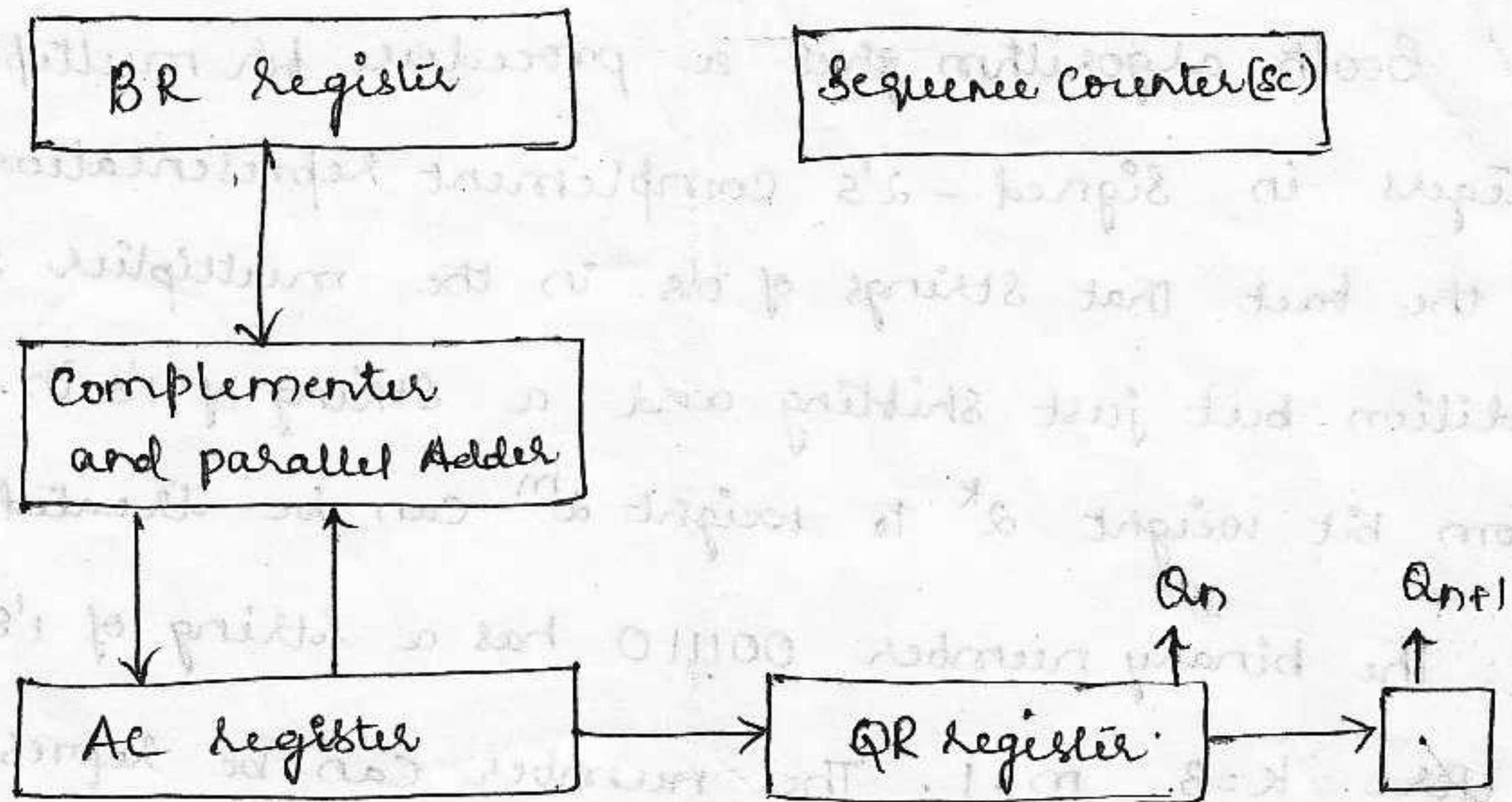
$= 2^4 - 2^1 = 14$. Therefore, the multiplication $M \times 14$, where

M is the multiplicand and 14 is the multiplier. This can be done as $M \times 2^4 - M \times 2^1$, thus the product can be obtained by shifting the binary multiplicand M four times to the left and subtracting ' M ' shifted left once.

Booth's algorithm requires examination of the multiplier bits and shifting the partial product. The following rules are to be required.

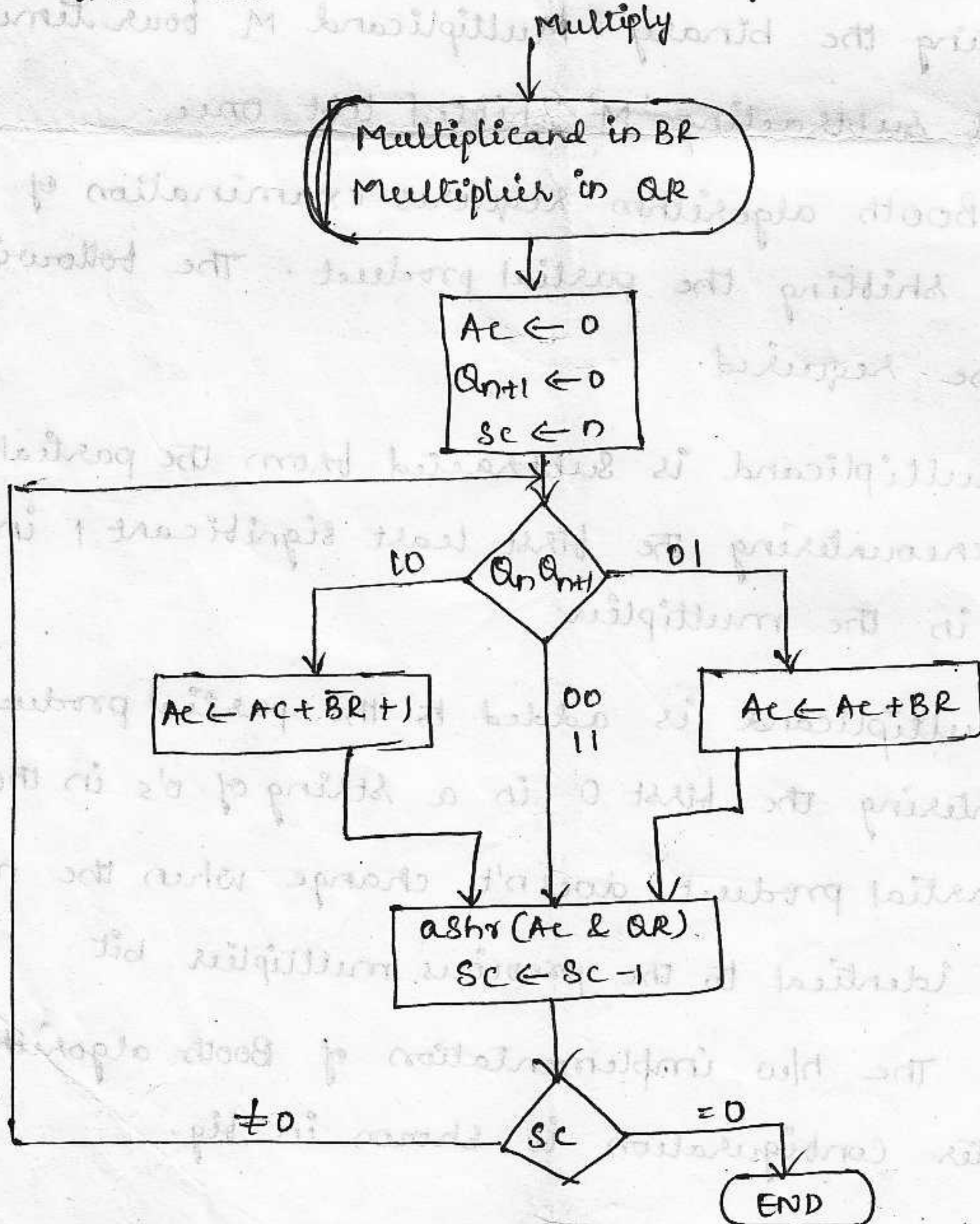
- 1) The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier.
- 2) The multiplicand is added to the partial product upon encountering the first 0 in a string of 0's in the multiplier.
- 3) The partial product doesn't change when the multiplier bit is identical to the previous multiplier bit.

The h/w implementation of Booth's algorithm requires the register configuration is shown in fig.



Here an extra bit - bit Q_{n+1} is appended to QR to facilitate a double bit inspection of the multiplier.

The flow chart for Booth algorithm is shown in fig.



Ex: $(-9) \times (-13) = +117$

Q_n	Q_{n+1}	BR = 1011 $\overline{BR} + 1 = 01001$	Ac	QR	Q_{n+1}	Sc
1	0	Initial	00000	1001	0	101
		Subtract	010001			
			001001			
		ashr	001001	11001	1	100
1	1	ashr	00010	01100	1	011
0	1	Add BR	00010 10111 00100			
			11001			
		ashr	11100	10110	0	010
0	0	ashr	01110	01011	0	001
			01001			
1	0	Subtract BR	10111			
		ashr	01011	10101	1	000

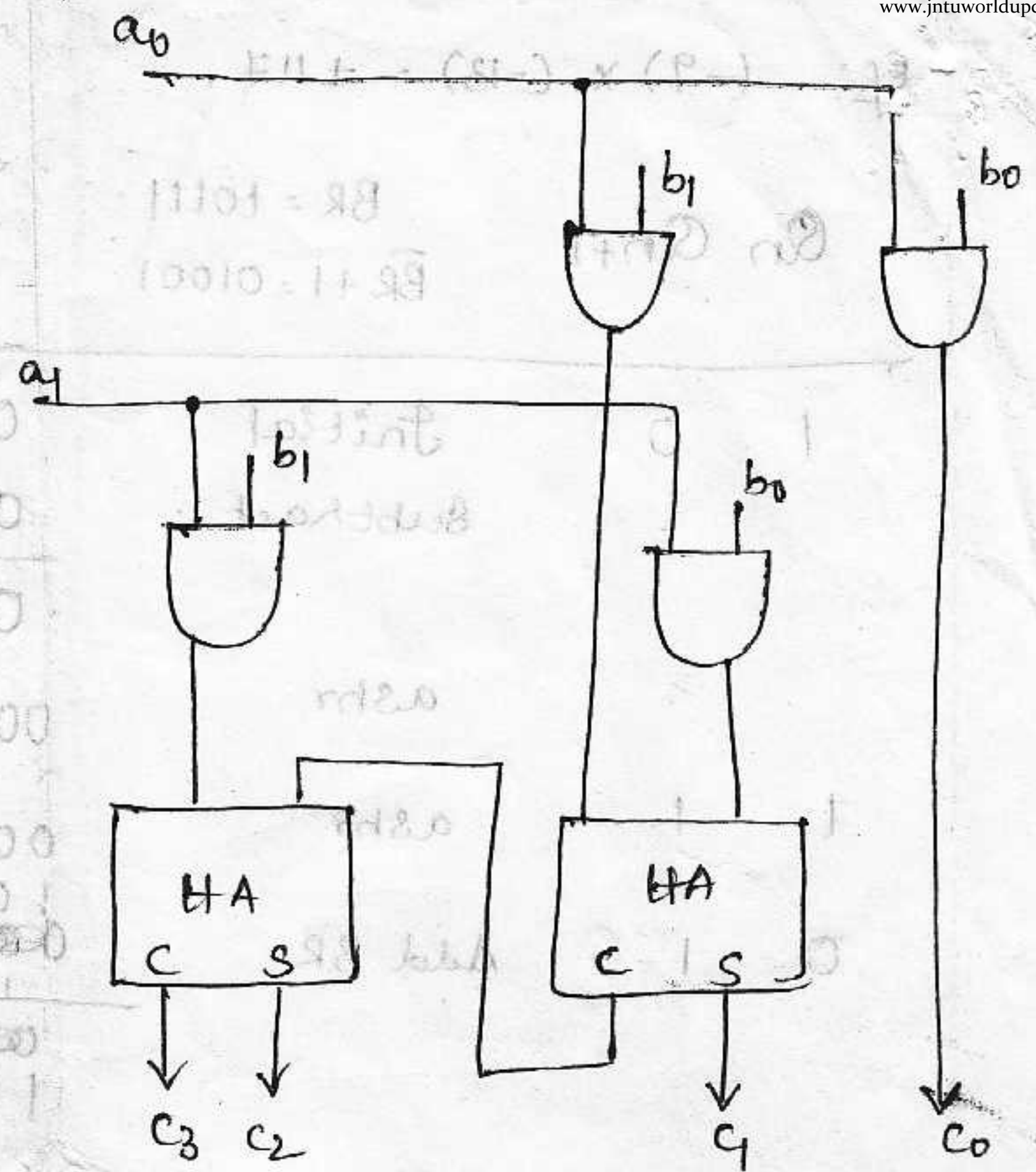
AC QR = 010110101

Array Multiplier :-

Checking the bits of the multiplier one at a time and forming partial products is a sequential operation that requires a sequence of add and shift micro operations. The multiplication of two binary numbers can be done with one micro operation by means of a Combinational circuit that forms the product bit all at once. This is a best way of multiplying two numbers.

An array multiplier can be implemented with a Combinational circuit as shown in fig.

$$\begin{array}{r}
 b_1 \ b_0 \\
 \times a_1 \ a_0 \\
 \hline
 a_0 b_1 \ a_0 b_0 \\
 a_1 b_1 \ a_1 b_0 \\
 \hline
 c_3 \ c_2 \ c_1 \ c_0
 \end{array}$$



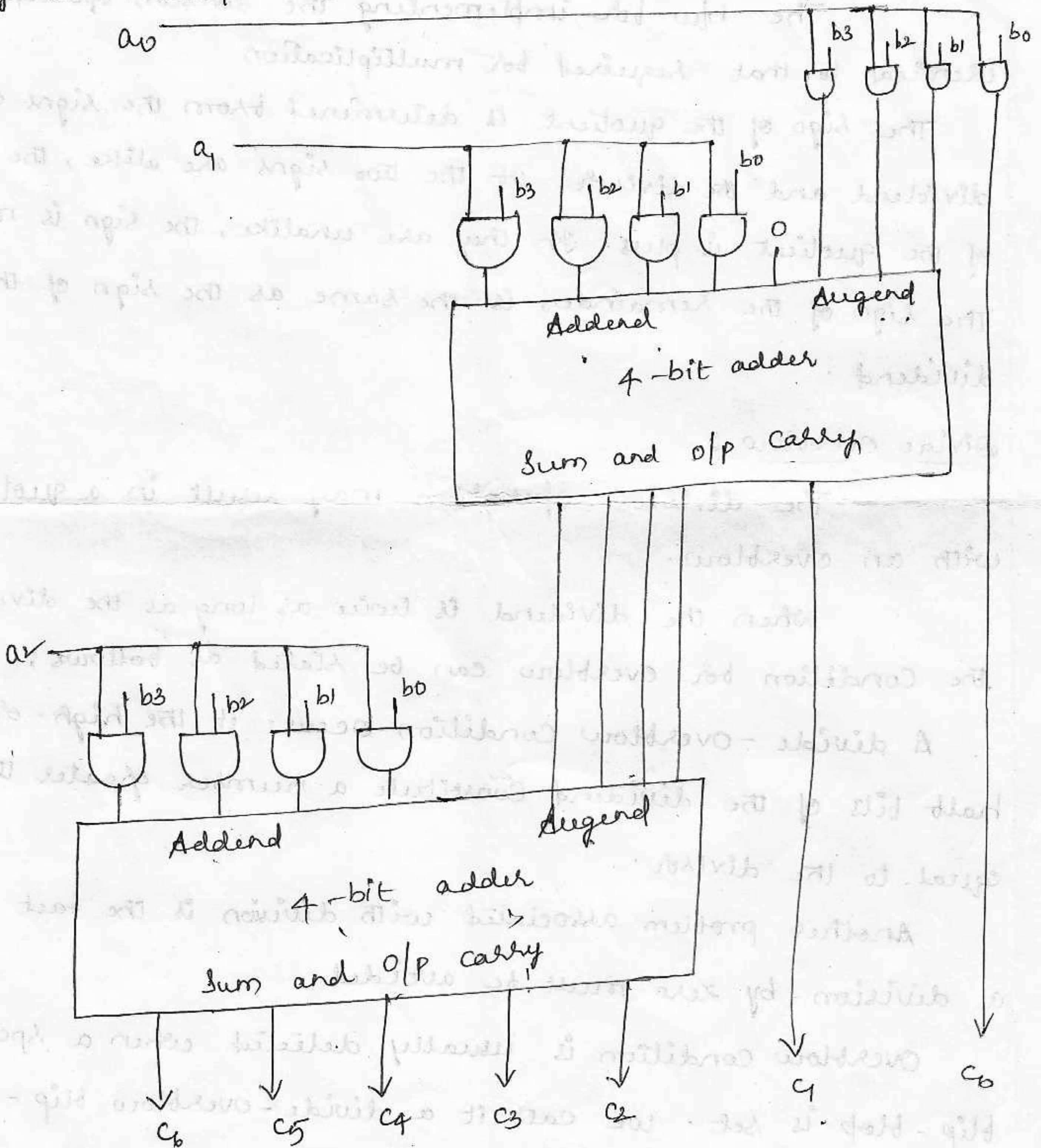
The multiplicand bits are b_1 & b_0 , the multiplier bits a_1 & a_0 and the product is $c_3 \ c_2 \ c_1 \ c_0$. The first partial product is formed by multiplying a_0 by $b_1 \ b_0$. The multiplication of two bits a_0 & b_0 produces '1' if both bits are 1, otherwise, it produces 0. This is identical to 'AND' operation.

The second partial product is formed by multiplying a_1 by $b_1 \ b_0$ and is shifted one position to the left. The two partial products are added with two half adders (HA) circuits. It is necessary of half adders because to produce sum.

A combinational circuit binary multiplier with more bits can be constructed in a similar fashion. The binary output in each level of AND gates is added parallel with the partial product of the previous level to form a new partial product.

For j multiplier bits & ' k ' multiplicand bits we need $j \times k$ AND gates and $(j-1) \ k$ -bit adders to produce a product of $j+k$ bits.

As a second example, Consider a multiplier circuit that multiplies a binary number of four bits with a number of three bits. The multiplicand is represented by $b_3 b_2 b_1 b_0$ and the multiplier by $a_2 a_1 a_0$ since $K=4$ & $J=3$. The logic diagram of the multiplier is shown below.



Division Algorithms :-

Division of two fixed-point binary numbers in signed-Magnitude representation.

H/w implementation for signed-magnitude data :-

The h/w for implementing the division operation is identical to that required for multiplication.

The sign of the quotient is determined from the signs of the dividend and the divisor. If the two signs are alike, the sign of the quotient is plus. If they are unlike, the sign is minus. The sign of the remainder is the same as the sign of the dividend.

Divide Overflow :-

The division operation may result in a quotient with an overflow.

When the dividend is twice as long as the divisor, the condition for overflow can be stated as follows:

A divide-overflow condition occurs if the high-order half bits of the dividend constitute a number greater than or equal to the divisor.

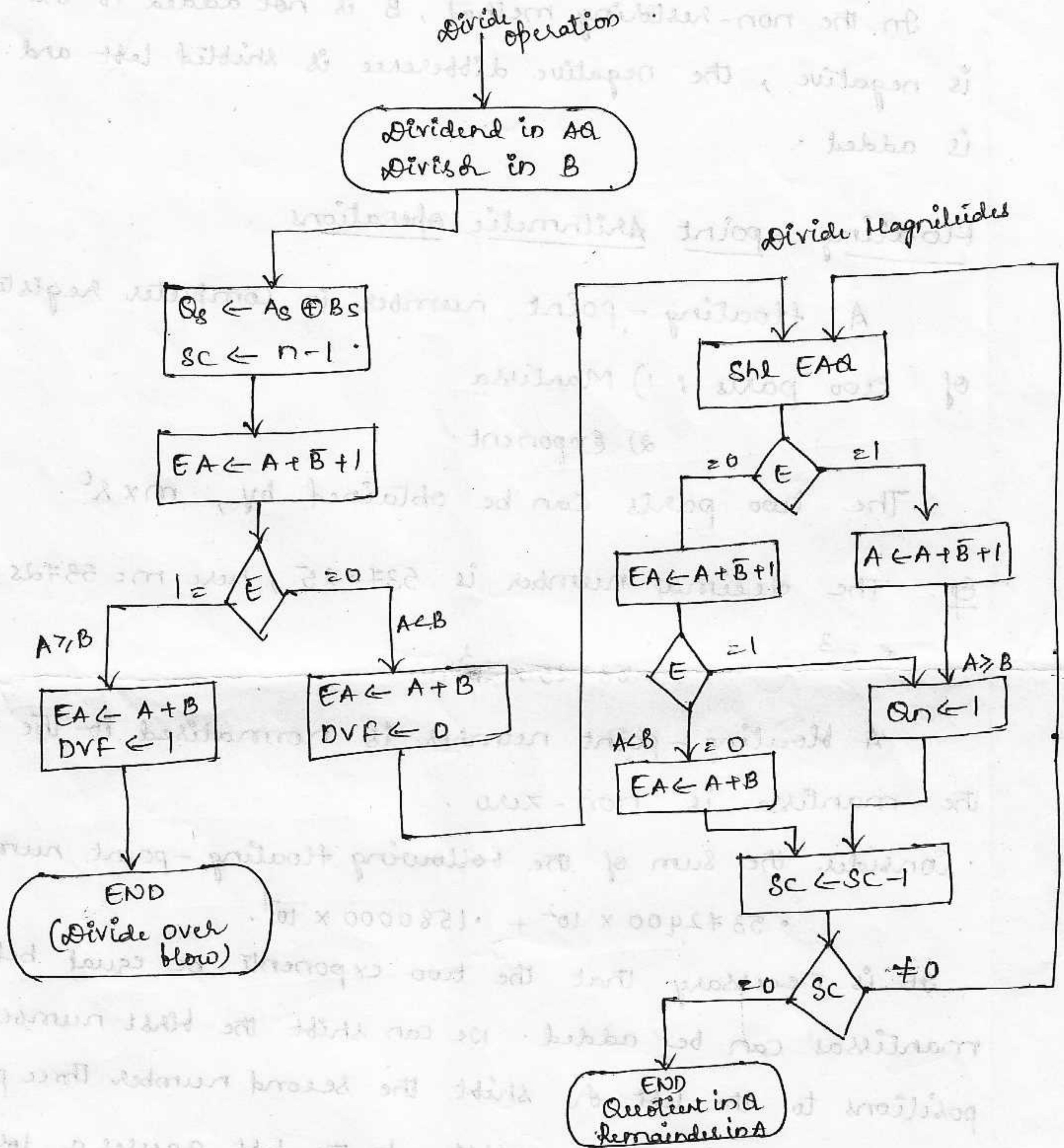
Another problem associated with division is the fact that a division by zero must be avoided.

Overflow condition is usually detected when a special flip-flop is set. We call it a divide-overflow flip-flop and label it DVF.

The best way to avoid a divide overflow is to use floating-point data.

Hardware Algorithm :-

The h/w divide algorithm is shown in the blockchart.



The sign of the result is transferred to Q_s to be part of the quotient.

Other Algorithms

The h/w method which described is called the restoring method. The two other methods are the comparison method and the non restoring method.

When two numbers are added, the sum may contain an overflow digit. An overflow can be corrected by shifting the sum once to the right & incrementing the exponent.

We can subtract two numbers in the following way.

$$\begin{array}{r} .56780 \times 10^5 \\ - .56430 \times 10^5 \\ \hline .00350 \times 10^5 \end{array}$$

A floating-point number that has a '0' in the MSB of the Mantissa is said to have an underflow.

Floating-point multiplication and division don't require an alignment of the mantissas. The product can be formed by multiplying the two mantissas & adding the exponents. Division is accomplished by dividing the mantissa and subtracting the exponents.

The exponent may be represented in any one of three representations: Signed-Magnitude, Signed-2's Complement, or Signed-1's Complement.

A fourth representation is a biased exponent. The sign bit is removed from being a separate entity. The bias is a positive number that is added to each exponent as the floating-point number is formed, so that internally all exponents are positive. For eg, exponent that ranges from -50 to 49. Internally, it is represented by two digits (without a sign) by adding to it a bias of 50. The exponent register contains the number $e + 50$ where 'e' is the actual exponent.

The advantage of biased exponents is that they contain only positive numbers. Another advantage is that the smallest possible biased exponent contains all zero.

Addition & Subtraction :-

The algorithm can be divided into four parts

- 1) Check for zeros
- 2) Align the mantissas
- 3) Add or subtract the mantissas
- 4) Normalize the result.

Multiplication :-

The multiplication algorithm can be divided into four parts

- 1) Check for zeros
- 2) Add the exponents
- 3) Multiply the mantissas
- 4) Normalize the product.

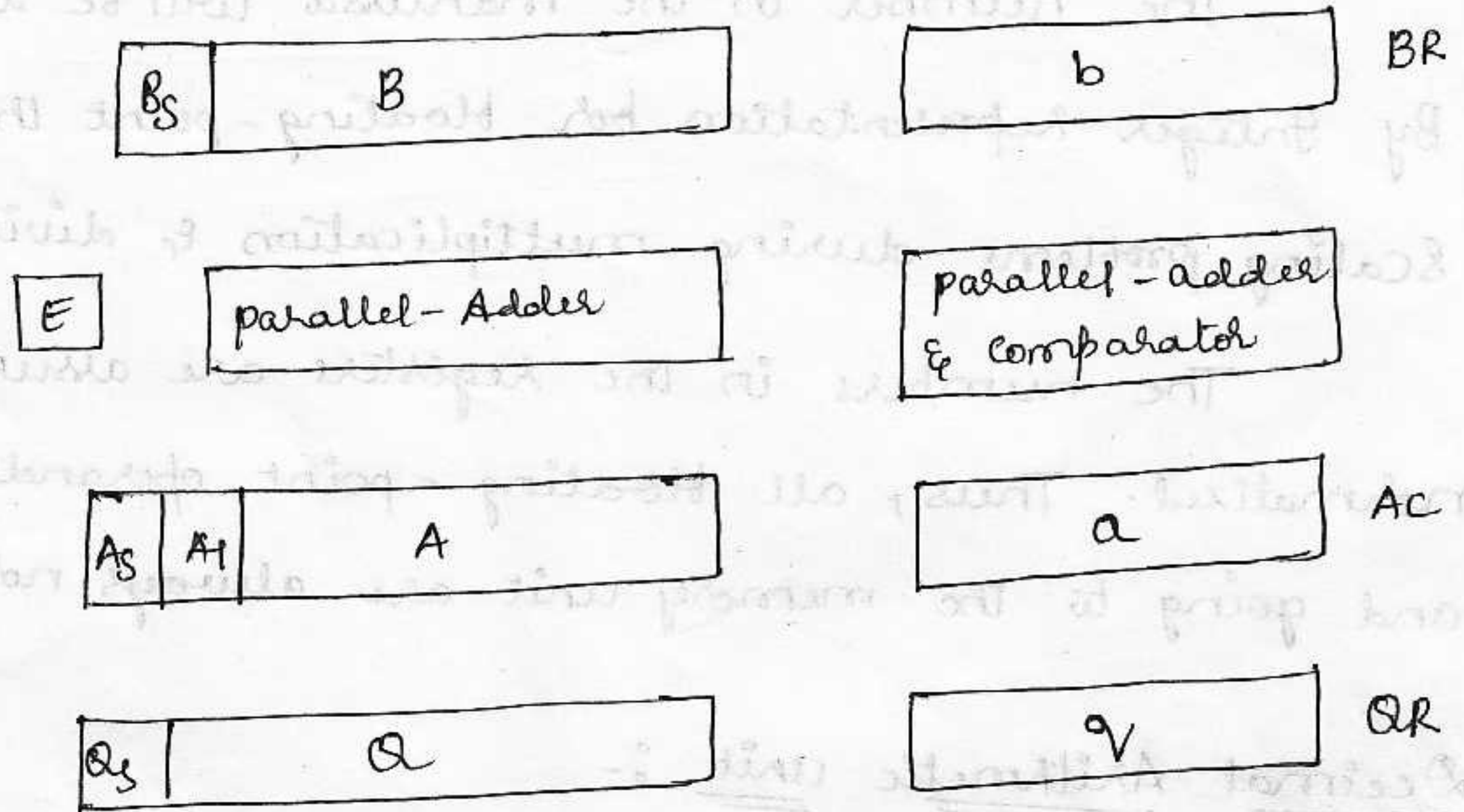
Division :-

The division algorithm can be divided into five parts:

- 1) Check for zeros
- 2) Initialize the registers & evaluate the sign.
- 3) Align the dividend
- 4) Subtract the exponents.
- 5) Divide the mantissas.

Register configuration :-

The register configuration for floating-point operations as shown in fig.



There are three registers BR, AC & QR. Each register is subdivided into two parts. The mantissa part & the exponent part.

The Mantissa part has the same uppercase letter symbols as in fixed-point representation. The exponent part uses the corresponding lower case letter symbol. Each floating-point number has a mantissa in signed-magnitude representation and a biased exponent. Thus AC has mantissa whose sign is in A_s and a magnitude is in A . The exponent is represented in 'a'. The diagram shows the MSB of A , is labeled by A_1 whose value must be '1' to get normalised. In the same way, Register BR is subdivided into B_s , b & B and QR into Q_s , Q , q .

A parallel-adder adds the two mantissas and transfers the sum into A and carry to 'E'. A separate parallel-adder is used for the exponents. It ~~has~~ they don't have a distinct sign bit, but are represented as a biased positive quantity.

The exponents are also connected to a magnitude comparator that provides three binary o/p's to indicate relative magnitude.

The number in the mantissa will be taken as a fraction. By Integer representation for floating-point they may cause scaling problems during multiplication & division.

The numbers in the registers are assumed to be initially normalized. Thus, all floating-point operands coming from and going to the memory unit are always normalized.

Decimal Arithmetic unit :-

To perform arithmetic operations with decimal data, it is necessary to convert the i/p decimal numbers to binary, it is ~~necessary~~ perform all calculations with binary numbers and to convert the results into decimal. It is very efficient Method. The decimal numbers are then applied to a decimal arithmetic unit to executing decimal arithmetic microoperations.

Many Computers have h/w for arithmetic calculations with both binary & decimal data.

A decimal arithmetic unit is a digital function that performs decimal microoperations. A single-stage decimal arithmetic unit consists of nine binary i/p variables & five binary o/p variables, since a minimum of four bits is required to represent each coded decimal digit.

BCD Adder :-

In BCD, each i/p digit doesn't exceed 9, the o/p sum can't

be greater than $9+9+1=19$, the '1' in the sum being an i/p carry.

For eg, we apply two BCD digits to a 4-bit binary adder. The adder will form the sum in binary & produce a result that may range from 0 to 19. These binary numbers are shown in below table and are labeled by symbols k, z_8, z_4, z_2 & z_1 , k is the carry. The first column in the table lists the binary sums as they appear in the O/p of a 4-bit binary adders. The o/p sum of two decimal numbers must be represented in BCD. In the second column, the values of the first value can be converted to the correct BCD digit.

Binary sum					BCD sum					Decimal.
k	z_8	z_4	z_2	z_1	c	s_8	s_4	s_2	s_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	0	1	0	1	0	0	0	10
0	1	0	1	0	1	0	0	0	0	11
0	1	0	1	1	1	0	0	0	1	12
0	1	1	0	0	1	0	0	1	0	13
0	1	1	0	1	1	0	0	1	1	14
0	1	1	1	0	1	0	1	0	0	15
0	1	1	1	1	1	0	1	0	1	16
1	0	0	0	0	1	0	1	1	0	17
1	0	0	0	1	1	0	1	1	1	18
1	0	0	1	0	1	1	0	0	0	19

In the table, when the binary sum is equal to or less than 1001, the corresponding BCD number is identical, and there is no conversion is needed. When the binary is greater than 1001, than nonvalid BCD representation is obtained.

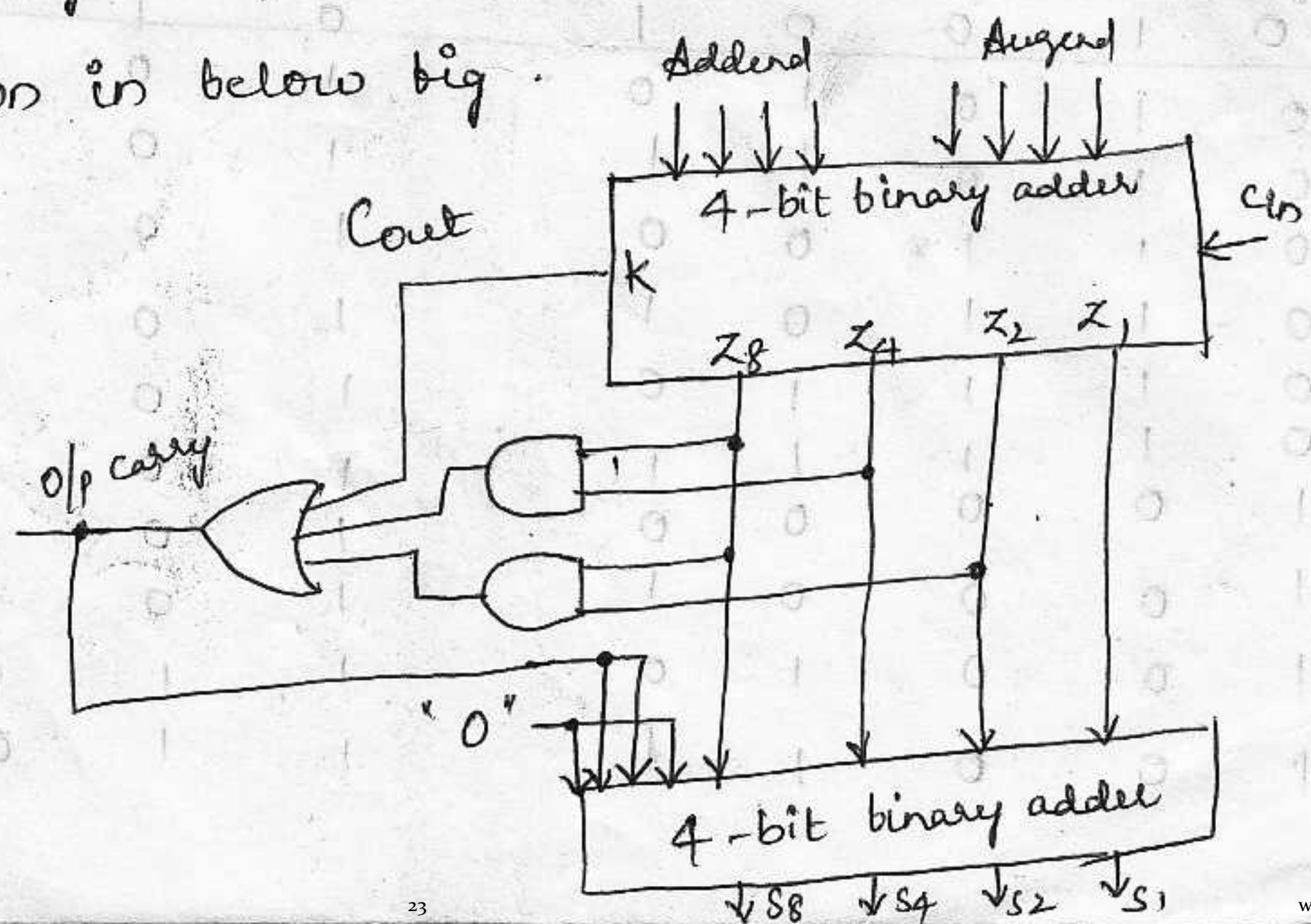
One method of adding decimal numbers in BCD would be to employ one 4 bit binary adder & perform the arithmetic operation one digit at a time. If the result is greater than or ^{equal to} ~~greater than~~ 1010, it corrected by adding 0110 to the binary sum. The second operation will automatically produce an o/p carry for the next pair of significant digits. The procedure is repeated until all decimal digits are added.

The Condition for a correction and an o/p - carry can be expressed by the Boolean function.

$$C = K + Z_8 Z_4 + Z_8 Z_2$$

When $C=1$, it is necessary to add 0110 to the binary sum and provide an output - carry for the next stage.

A BCD adder is a circuit that adds two BCD digits in parallel & produces a sum digit also in BCD. To add 0110 to the binary sum, we use a second 4-bit binary adder as shown in below fig.



The two decimal digits, together with the input-carry, are first added in the top 4-bit binary adder to produce the binary sum. When it is equal to 1, binary 0110 is added to the binary sum through the bottom 4-bit binary adder. The o/p - carry generated from the bottom binary adder may be ignored, since it supplies information already available in the o/p - carry terminal.

A decimal parallel-adder that adds n decimal digits needs n BCD adder stages with the output-carry from one stage connected to the input-carry of the next-higher order stage.

BCD Subtraction :-

BCD is not a Self-Complementing Code, the 9's Complement cannot be obtained by complementing each bit in the code.

The 9's Complement of a decimal digit represented in BCD may be obtained by complementing the bits in the coded representation of the digit provided a correction is included.

There are two possible correction methods.

- 1) binary 1010 (decimal 10) is added to each complemented digit and the carry discarded after each addition
- 2) binary 0110 (decimal 6) is added before the digit is complemented.

Ex: 1) 0111

Complement

$$\begin{array}{r} 1000 \\ 1010 \text{ (add)} \\ \hline 0010 \end{array}$$

2) 0111

$$\begin{array}{r} 0111 \text{ (add)} \\ 0110 \\ \hline 1101 \end{array}$$

$$0010 \text{ (complement)}$$

Complementing each bit of a 4-bit binary number N is identical to the subtraction of the number from 1111 (decimal 15)

1) Adding the binary equivalent of decimal 10 gives

$$15 - N + 10 = 9 - N + 16$$

So, here 16 signifies the carry is discarded. So, the result is $9 - N$

2) Adding the binary equivalent of decimal 6 gives

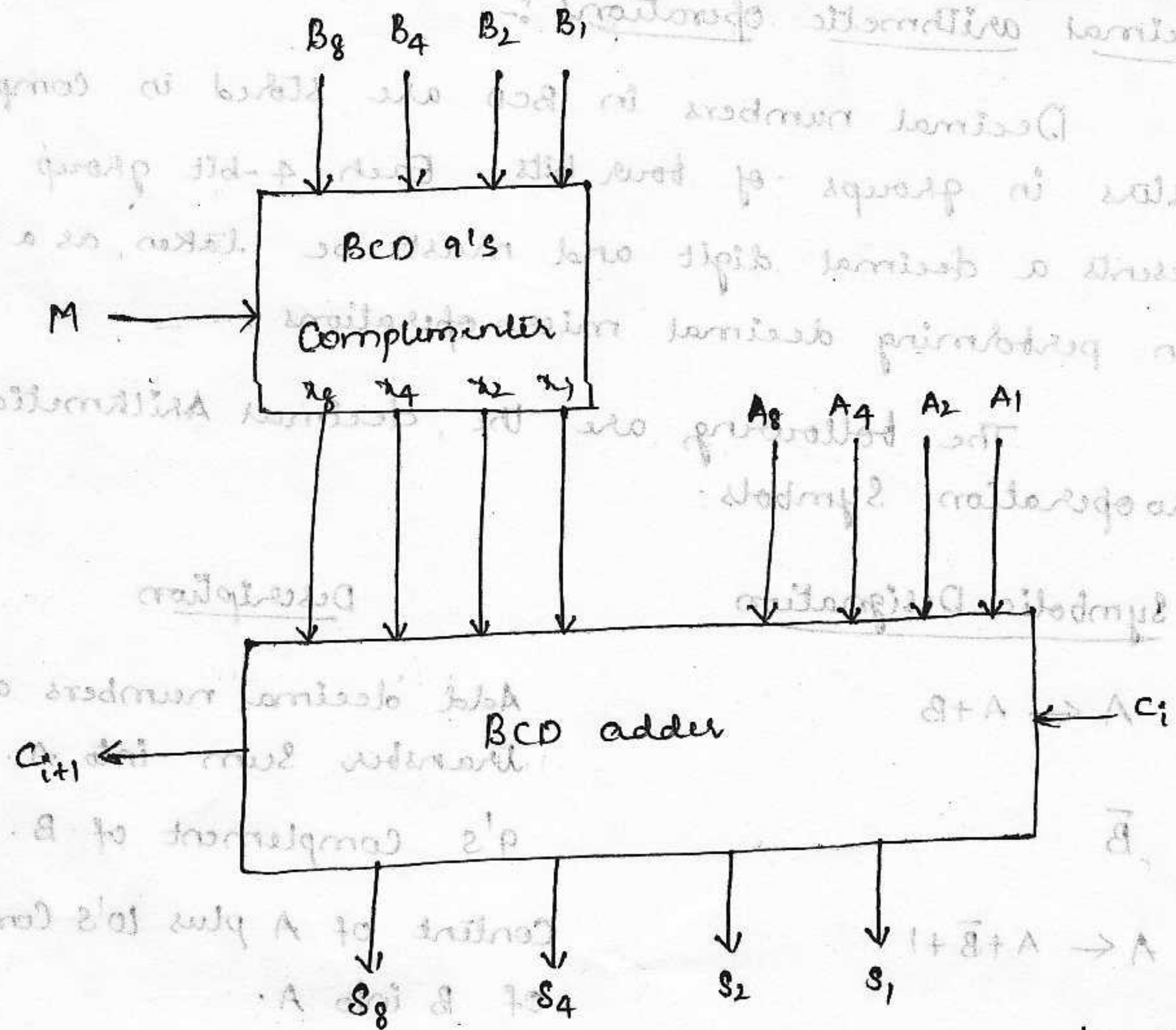
$$15 - (N + 6) = 9 - N$$

The 9's Complement of a BCD digit can also be obtained through a Combinational circuit.

When this circuit is attached to a BCD adder, the result is BCD adder / Subtractor.

Let the subtrahend (or addend) digit be denoted by the four binary variables B_8, B_4, B_2, B_1 . Let M be a mode bit that controls the add / subtract operation.

Let the binary variables x_8, x_4, x_2, x_1 be the o/p's of the 9's Complementer circuit. The below fig. shows the one stage of a decimal arithmetic unit.



It consists of a BCD adder and a 9's Complementer.

The mode 'M' controls the operation of the unit. With $M=0$,

the 's' o/p's form the sum of A and B.

With $M=1$, the 's' o/p's form the sum of A plus the 9's Complement of B.

The o/p carry c_{i+1} from one stage must be connected to the i/p carry c_i of the next higher-order stage.

The way to subtract the two decimal numbers is to let $M=1$ and apply a '1' to the i/p carry c_i of the first stage.

The o/p's will form the sum of A plus the 10's Complement of B which is equivalent to a subtraction operation if the carry-out of the last stage is discarded.

Decimal arithmetic operations :-

Decimal numbers in BCD are stored in computer registers in groups of four bits. Each 4-bit group represents a decimal digit and must be taken as a unit when performing decimal micro operations.

The following are the decimal Arithmetic micro operation Symbols.

Symbolic Designation

Description

$A \leftarrow A + B$

Add decimal numbers and transfer sum into A.

$\bar{B}$

9's Complement of B.

$A \leftarrow A + \bar{B} + 1$

Content of A plus 10's Complement of B into A.

$Q_L \leftarrow Q_L + 1$

Increment BCD number in Q_L

dshr A

Decimal Shift-right register A

dshl A

Decimal Shift-left register A

Incrementing / decrementing a register is the same for binary and decimal except that the ^{binary} counter goes through 16 states from 0000 to 1111, when incremented. The decimal counter goes through 10 states from 0000 to 1001 and back to 0000.

A decimal shift right or left is preceded by the letter 'd' to indicate a shift over the four bits that hold the decimal digits.

Ex: 3421

0011 0100 0010 0001

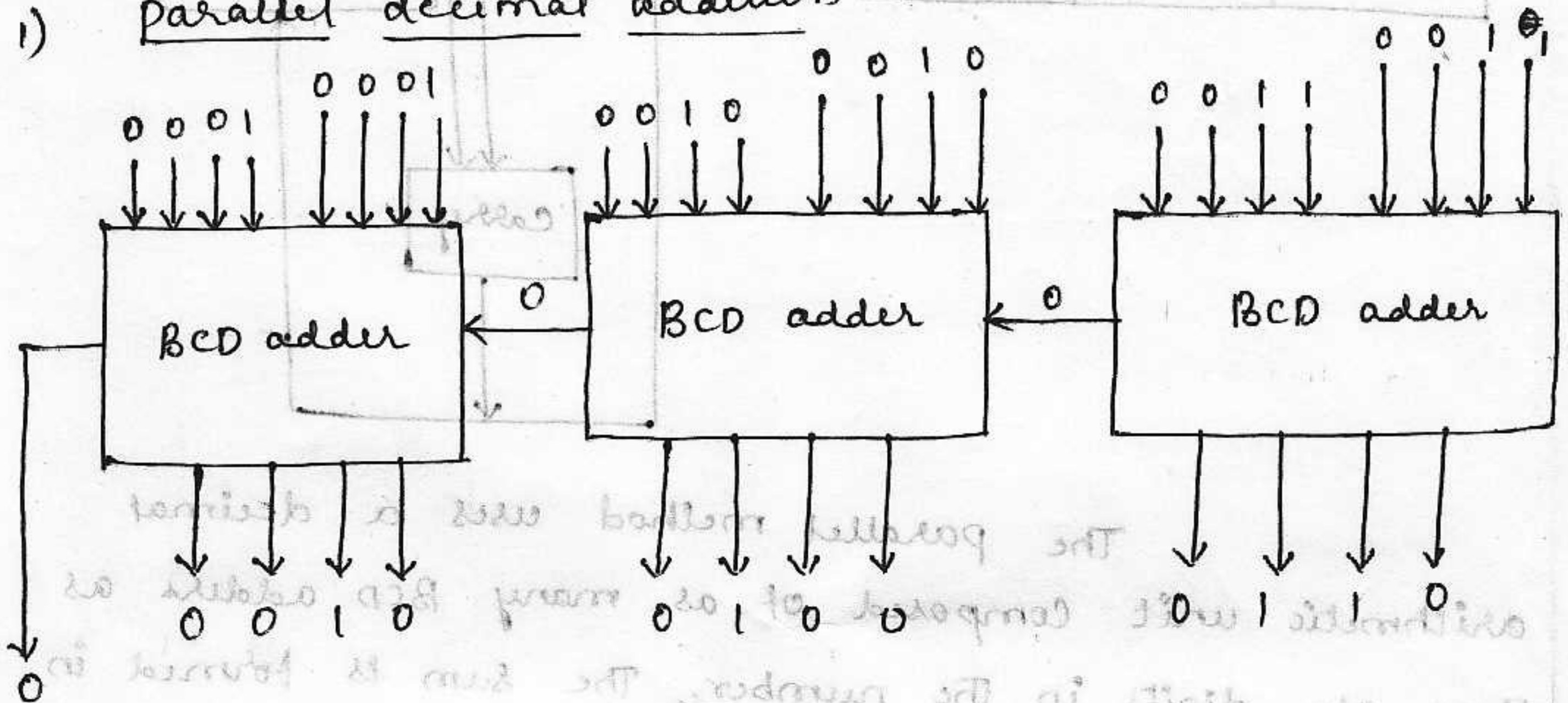
dshr 0000 0011 0100 0010

Addition and Subtraction :-

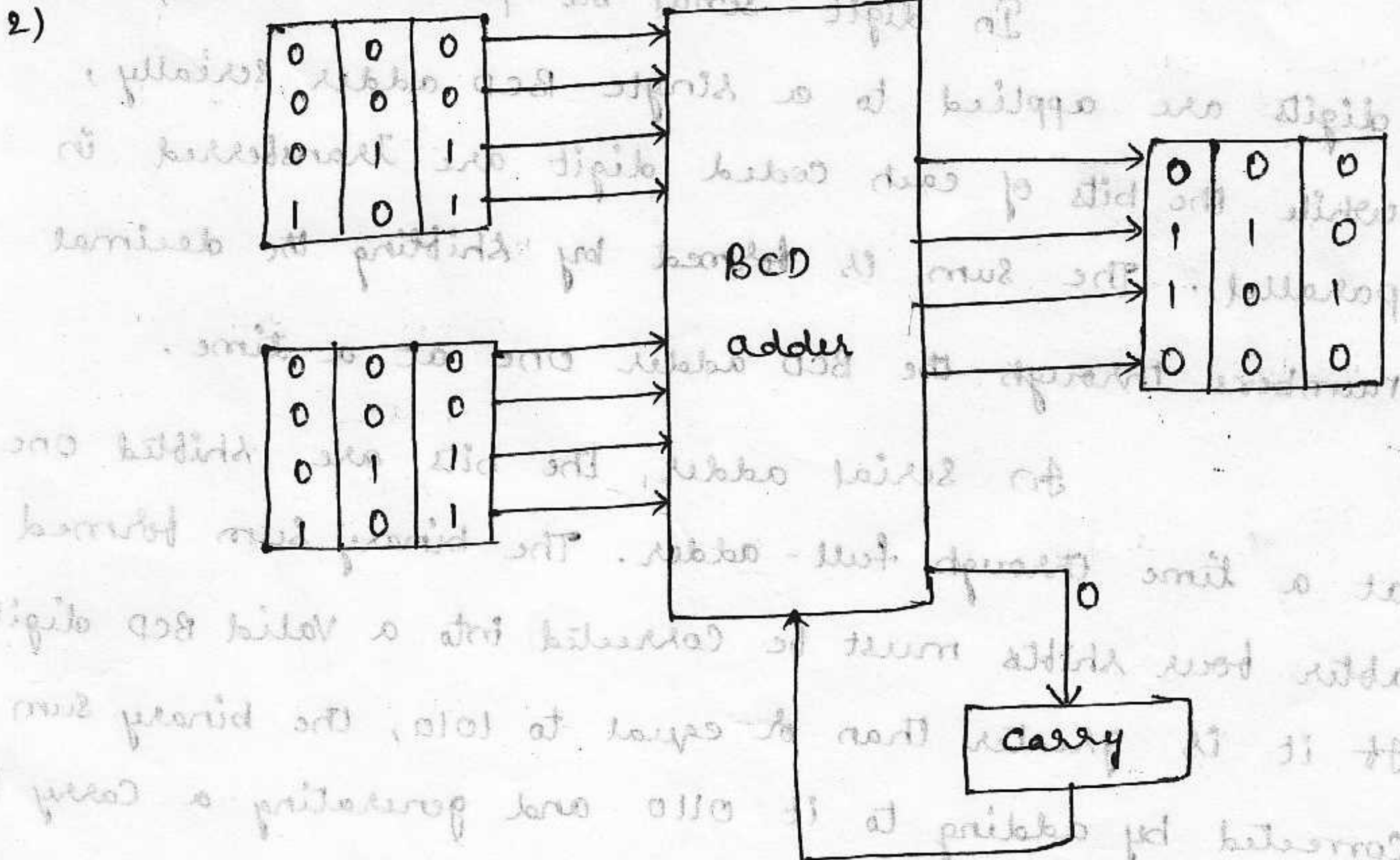
A decimal data can be added in three different ways.

Ex: $123 + 123 = 246$

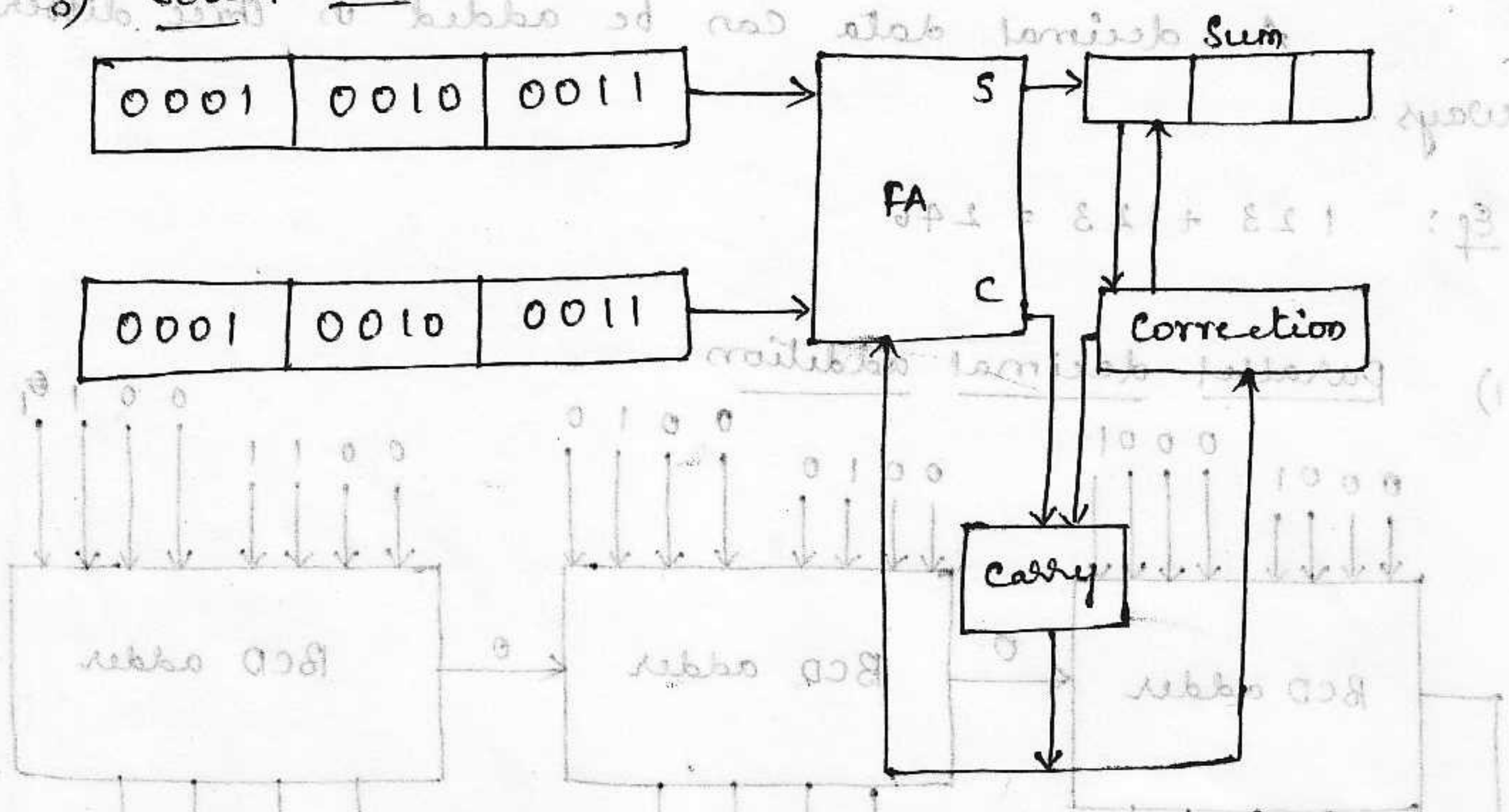
1) Parallel decimal addition



2) Digit - Serial, bit parallel decimal addition



3) Serial adder



The parallel method uses a decimal arithmetic unit composed of as many BCD adders as there are digits in the number. The sum is formed in parallel and requires only one micro operation.

In digit-serial bit-parallel method, the digits are applied to a single BCD adder serially, while the bits of each coded digit are transferred in parallel. The sum is formed by shifting the decimal numbers through the BCD adder one at a time.

In serial adder, the bits are shifted one at a time through full-adder. The binary sum formed after four shifts must be corrected into a valid BCD digit. If it is greater than or equal to 1010, the binary sum is corrected by adding to it 0110 and generating a carry for the next pair of digits.

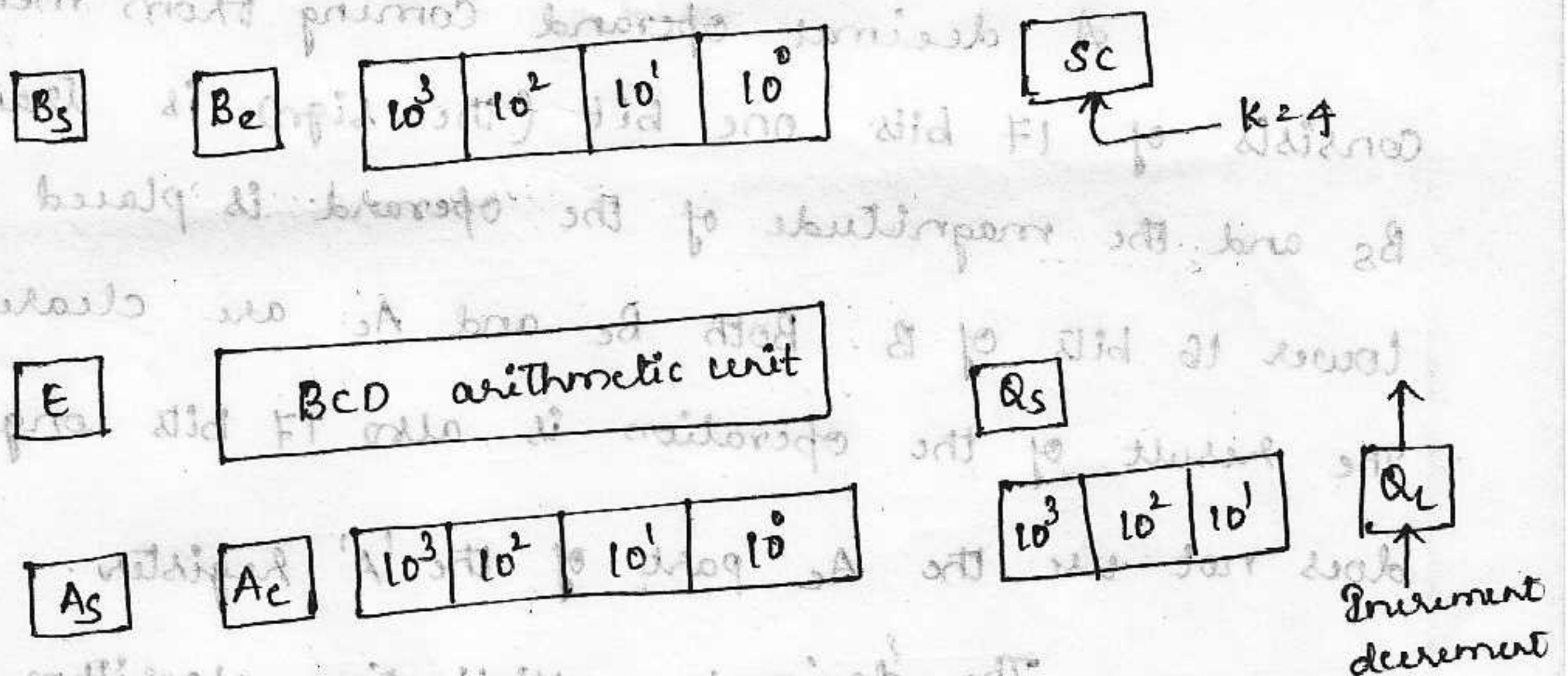
The parallel method is fast but requires a large number of adders.

The digit-serial bit-parallel method requires only one BCD adder, which is shared by all the digits. It is slower than the parallel method because of the time required to shift the digits.

The serial method requires a minimum amount of equipment but is very slow.

Multiplication :-

The registers configuration for the decimal multiplication is shown in fig.



Assuming here four-digit numbers, with each digit occupying four bits, for a total of 16 bits for each number. There are three registers A, B & Q each having a corresponding sign flip-flop A_s , B_s & Q_s .

Registers A and B have four more bits designated by A_e & B_e , that provide an extension of one more digit to the registers.



The BCD arithmetic unit adds the five digits in parallel and places the sum in the five-digit A register. The end carry goes to the flip flop 'E'.

The purpose of digit A_e is to accommodate an overflow while adding the multiplicand to the partial product during multiplication.

The purpose of digit B_e is to form the 9's Complement of the divisor when subtracted from the partial remainder during the division operation.

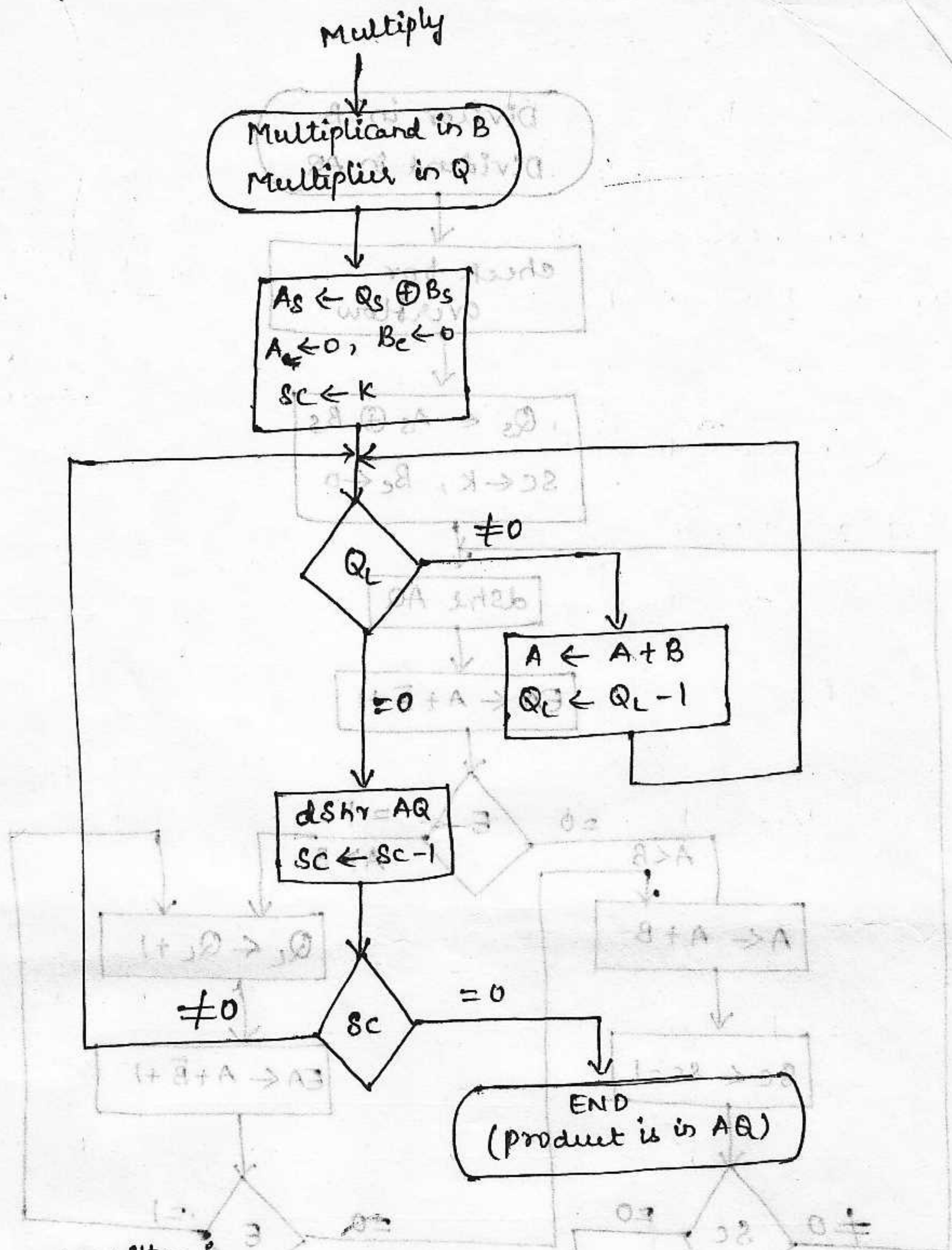
The least significant digit in register Q is denoted by Q_e . This digit can be incremented or decremented.

A decimal operand coming from memory consists of 17 bits. One bit (the sign) is transferred to B_s and the magnitude of the operand is placed in the lower 16 bits of B. Both B_e and A_e are cleared initially.

The result of the operation is also 17 bits long and does not use the A_e part of the 'A' register.

The decimal multiplication algorithm is shown below.

Assuming four four-digit numbers, with each digit occupying four bits, for a total of 16 bits per number. There are three registers A, B & C each having a corresponding sign flip-flop A_s , B_s & C_s . Registers A and B have four more bits designated by A_e & B_e , that provide an extension of one more digit to the registers.

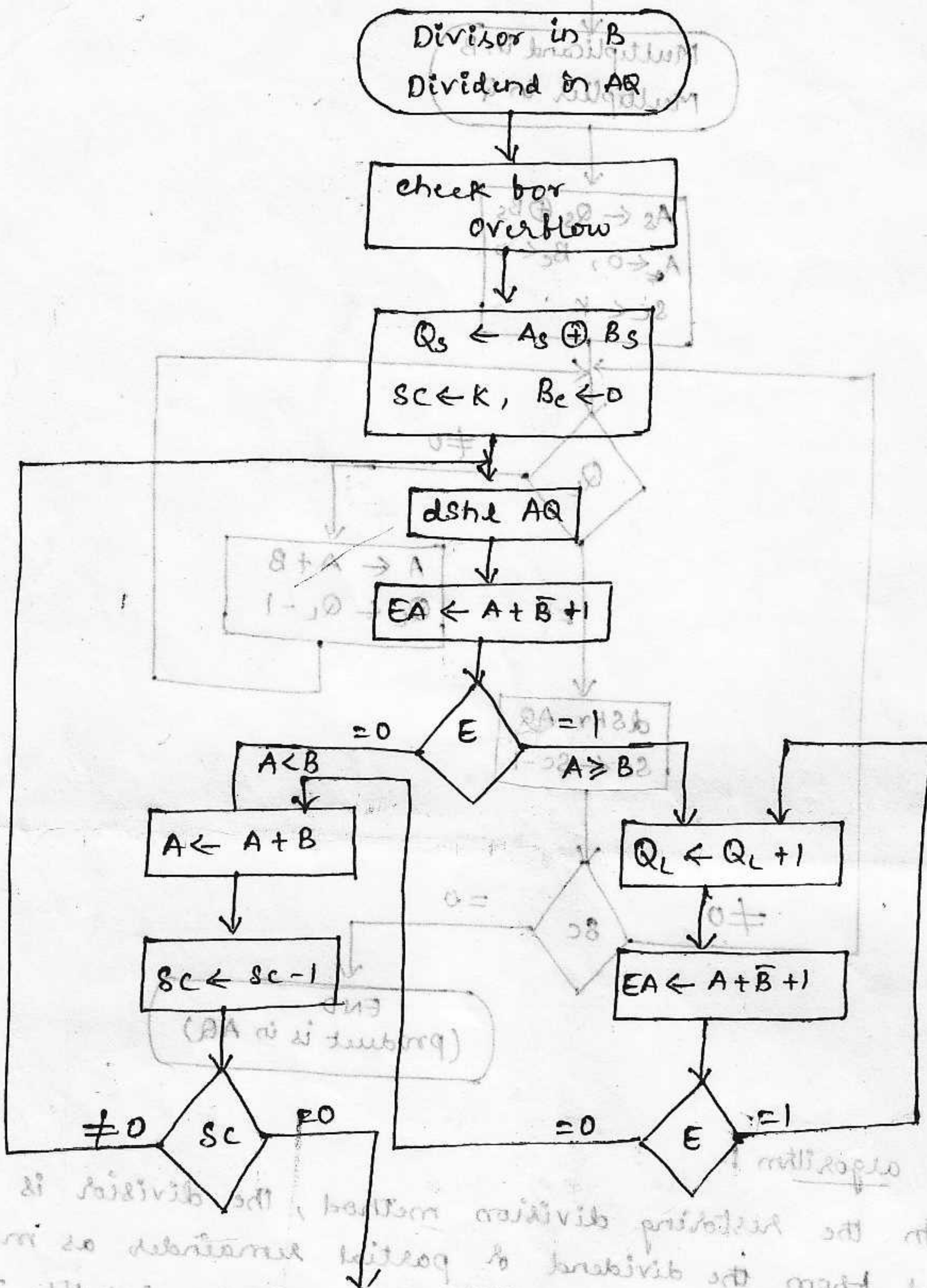


Division algorithm :-

In the restoring division method, the divisor is subtracted from the dividend or partial remainder as many times as necessary until a negative remainder results. The correct remainder is then restored by adding the divisor. The digit in the quotient reflects the no. of subtractions, upto but excluding the one that caused the negative difference.

The decimal division algorithm is shown below.

Algorithm



END
Quotient is in Q
Remainder is in A

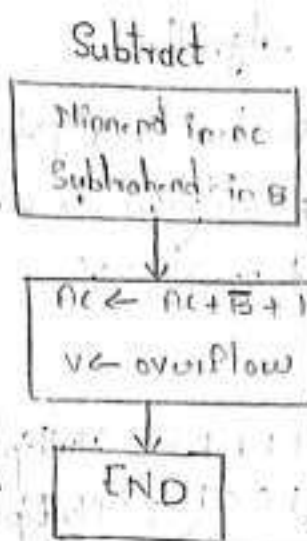
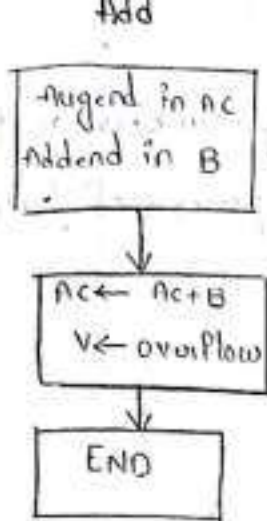
Part-① Computer Arithmetic:-

The data is manipulate (change/control) by using the arithmetic instructions in digital computer. The addition, Subtraction, Multiplication & division are the four basic arithmetic operations. To execute arithmetic operations, there is a separate section called Arithmetic Processing unit in the CPU. There are three ways of representing negative fixed point numbers i.e., Signed magnitude, Signed 1's Complement & Signed 2's Complement.

Addition & Subtraction Algorithm (signed magnitude data):

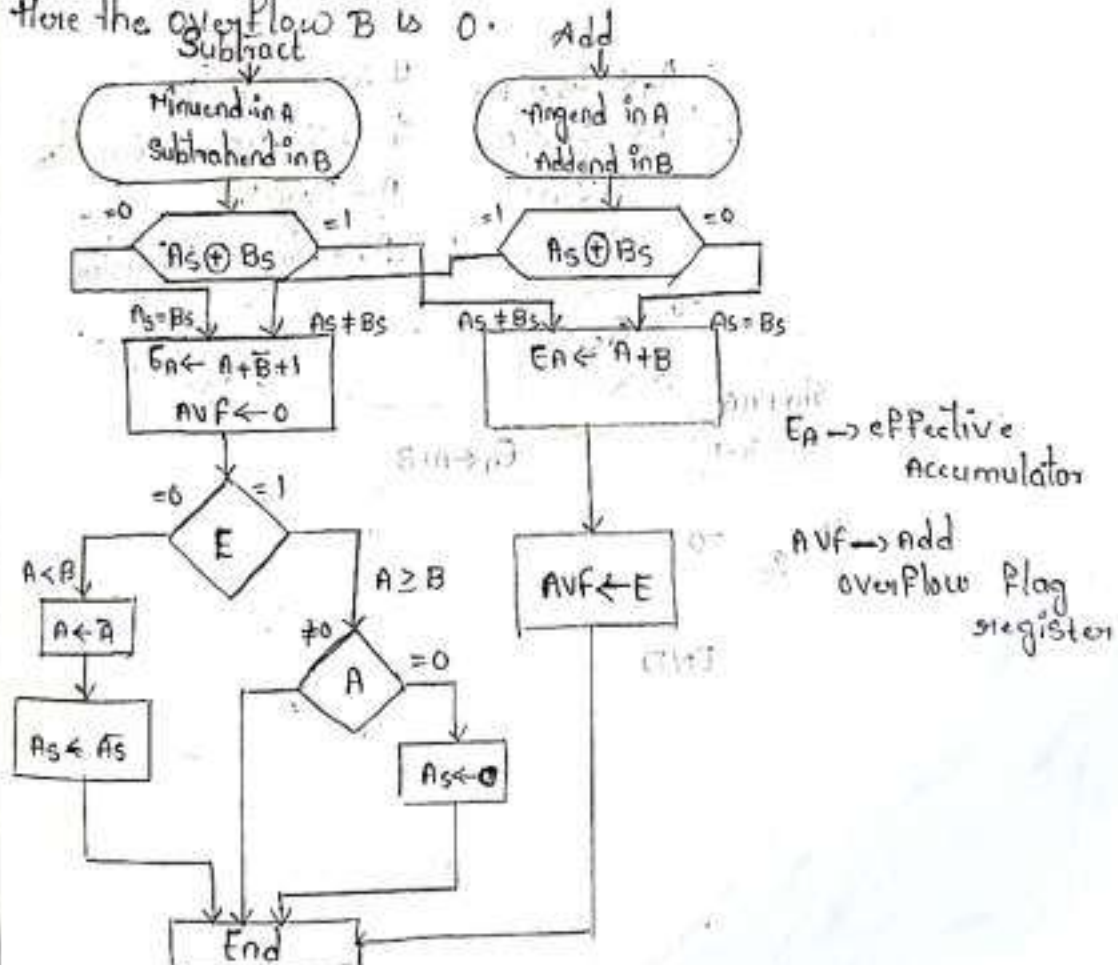
Operation	Add Magnitude	Subtract Magnitude		
		When $A > B$	When $A \leq B$	When $A = B$
$(+A) + (+B)$	$+(A+B)$			
$(+A) + (-B)$		$+(A-B)$	$-(B-A)$	$+(A-B)$
$(-A) + (+B)$		$-(A-B)$	$+(B-A)$	$+(A-B)$
$(-A) + (-B)$	$-(A+B)$			
$(+A) - (+B)$		$+(A-B)$	$-(B-A)$	$+(A-B)$
$(+A) - (-B)$	$+(A+B)$			
$(-A) - (+B)$	$-(A+B)$			
$(-A) - (-B)$		$-(A-B)$	$+(B-A)$	$+(A-B)$

Algorithm:-



To implement the addition & Subtraction the register B & the accumulator AC is taken. to perform addition the Augend & Addend are added and stored in Accumulator AC. If there is any carry the overflow 'V' will be set to '1'. In case of Subtraction the B data has to be Complemented and added with minuend and stored in accumulator AC.

Here the overflow V is 0.



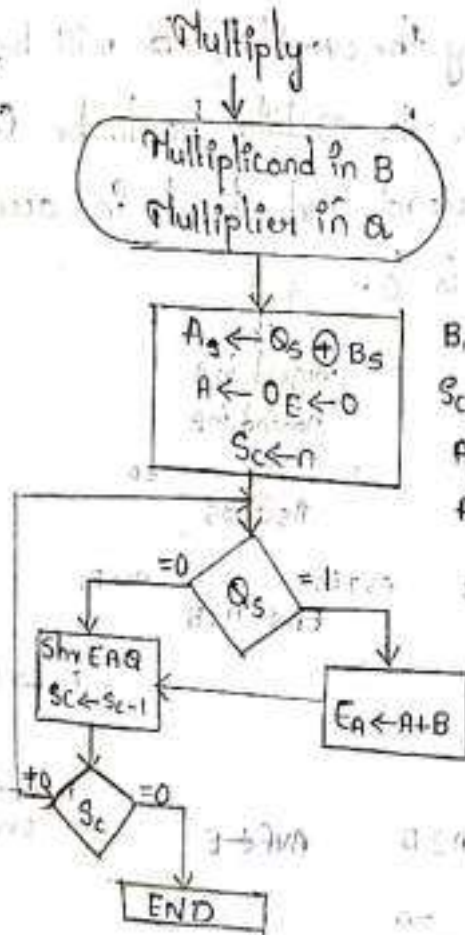
Multiplication Algorithm:-

Multiplication of two fixed point binary numbers in signed magnitude representation is done by a process of successive shift and add operations.

Ex:-

$$\begin{array}{r} 15 \rightarrow 1111 \rightarrow \text{Multiplicand} \\ \times 9 \rightarrow \times 1001 \rightarrow \text{Multiplier} \\ \hline \end{array}$$

$$\begin{array}{r} 1111 \\ 00000 \\ 00000 \\ (+) 1111 \\ \hline 10000111 \end{array}$$

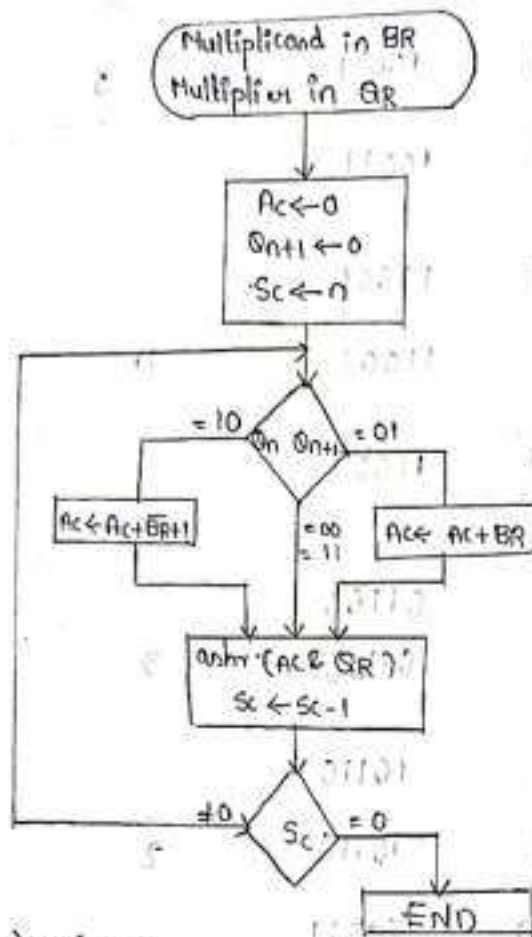


B, A → Registers
S_c → Sequence Counter
A - Register
A_s - Signed Register

E	A	Q	SC	B = 1011 Q = 1001
0	00000	10011	5	
0	01011 10111	10011		eliminated
0	01011	11001		
0	00000 01011	11001	4	
1	10111 00010	11001		eliminated
0	10001	01100		
0	10001 01100	01100	3	eliminated
0	01000	10110		
0	01000 00100	10110	2	eliminate
0	00100	01011		
0	00100 10111	01011	1	
0	11011 01101	01011		eliminate
✓ 0	01101	10101		

Booth's Algorithm:-

The Booth's algorithm is a procedure for multiplying binary integers in signed 2's Complement representation.



Ac - auxiliary carry

Ex:- $(+7) \times (+3) =$
 $(0111) (0011)$

Ac	Qn	Qn+1	Operations	Sc
0000	0011	0	BR = 0111 BR = 1000	
01001		0		
1001		0		
1001	0011	0		
1100	1001	1	shifted	4
0110	0100	1	shifted	3
0001	0100	1		2
0000	1010	0		
0000	0101	0	shifted	1

Ex-2: $(+8) \times (+4) =$

1000 0100

Ac	Qn	Qn+1	Operation	Sc
0000	0100	0	shifted	4
0000	0010	0	shifted	3
0000	0001	0	shifted	2
0000	0001	0	Br = 1000 Br = 0111 0 1 1000	2
1100	0000	1	shifted	
1100	0000	1		1
0100	0000	1		
0010	0000	0	shifted	

Fast Multiplication:-

For speeding up the multiplication operation there are two techniques:

- 1) Bit pair Recoding of Multiplier
- 2) Carry Save Addition of Summand.

Bit pair recoding of Multiplier

In this the maximum number of partial Product will get halved.

i.e., $n = n/2$

Multiplier bit Pair		Multiplier bit on the right	Multiplier selected at position i
$i+1$	i	$i-1$	
0	0	0	$0 \times M$
0	0	1	$+1 \times M$
0	1	0	$+1 \times M$
0	1	1	$+2 \times M$
1	0	0	$-2 \times M$
1	0	1	$-1 \times M$
1	1	0	$-1 \times M$
1	1	1	$0 \times M$

Ex:- Multiply -11 & +27

$$A = 1011$$

$$B = 011011$$

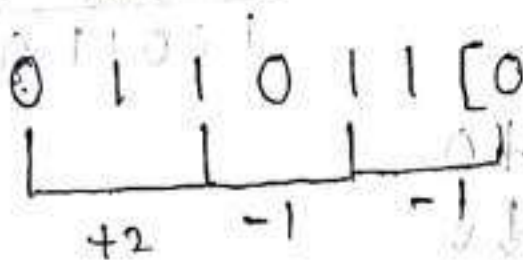
equating A & B.

$$A = 01011$$

$$B = 11011$$

→ A = $\overset{\text{-ve sign}}{\textcircled{1}}10101 \rightarrow$ Multiplicand

B = $\overset{\text{+ve sign}}{\textcircled{0}}11011 \rightarrow$ Multiplier.



$$110101$$

$$\times \begin{matrix} +2 & -1 & -1 \end{matrix}$$

$$\begin{array}{r} 0000000001011 \\ 00000001011 \\ \hline 11101010 \end{array}$$

$$111011010111$$

For checking, find the 2's complement for the solution

$$\begin{array}{r} 01011 \\ \downarrow \downarrow \downarrow \downarrow \downarrow \\ 10100 \\ (+) \quad 1 \\ \hline 10101 \end{array}$$

For (-1) → 2's complement

$$\begin{array}{r} 110101 \\ \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \\ 001010 \\ + \quad 1 \\ \hline 001011 \end{array}$$

For (+2) multiply by '10'

$$\begin{array}{r} 110101 \\ (x) \quad 10 \\ \hline 000000 \\ 110101 \\ \hline 1101010 \end{array}$$

Perform the bit pair recoding of multiplier to the values +13 & -6

+13 & -6

(13) A = 1101

Find 2's Complement Pos 6

(6) B = 0110

add sign bits

```

0 1 1 0
↓ ↓ ↓ ↓
1 0 0 1
(+) 0 1
-----
1 0 1 0

```

(+13) A = 01101 - Multiplicand

(-6) B = 11010 - Multiplier

added bit

```

  1 1 1 0 1 0 [0]
  |   |   |   |
 0   -1  -2

```

stored values

~~11010~~

the value is extended by 0
MSB bit
discarded

```

01101
00000
11100
00000
-----
1110110010

```

Pair bits extension to match an

For (-2) Find 2's complement & multiply with it

```

0 1 1 0 1
↓ ↓ ↓ ↓ ↓
1 0 0 1 0
(+) 1
-----
1 0 0 1 1
1 0
-----
0 0 0 0 0
1 0 0 1 1
-----
1 0 0 1 1 0

```

2's complement of result

```

1 1 1 1 0 1 1 0 0 1 0
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
0 0 0 0 1 0 0 1 1 0 1
(+) 1
-----
0 0 0 0 1 0 0 1 1 1 0
2^10 2^9 2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

```

= 64 + 8 + 4 + 2

1011010

Method-2:-

Method
carry Save addition of Summands:-

Ex:- 11 & 13

$$11 = 1011$$
 $13 = 1101$

= "1011"

Q. 1101

[illegible]
$$A+B = \begin{array}{r} 1011 \\ 00000 \end{array}$$

01011 - 5

00000 - C1

$$S_1 + C_1 + S_2 = S_3, C_3$$

0001011

00 00000

100 000 000 000

[illegible]

0000000 - c₃

63

6- 0110

3-0011

$$= 0.1101$$
 $x, 0.01-1$
$$\begin{array}{r} 1110n \\ 1110n \end{array}$$

01,100 B

[illegible]
$$\begin{array}{r} 00000000 \\ \hline 0010010 \end{array}$$

1000

643

μV

$C + D = 101100$

⑥ 101 1000

$$10000100 = 52$$

1110100 - S2

00000000 - C2

$$S_3 + C_3 + C_2 = S_4, C_4$$

1 1 1 1 1 1 1

0 0 0 0 0 0 0

0000000000

- 1 1 0 0 1 1 - 54

00000000 - cu

10001111

$$\begin{array}{r}
 A+B = 0110 \\
 (+) 01100 \\
 \hline
 01010 - S_1 \\
 01000 - C_1
 \end{array}$$

$$\begin{array}{r}
 C+D = 000000 \\
 00000000 \\
 \hline
 00000000 - S_2 \\
 00000000 - C_2
 \end{array}$$

$$S_1 + C_1 + S_2 = S_3, C_3$$

$$\begin{array}{r}
 01010 \\
 01000 \\
 0000000 \\
 \hline
 0000010 - S_3 \\
 0010000 - C_3
 \end{array}$$

$$S_3 + C_3 + C_2 = S_4, C_4$$

$$\begin{array}{r}
 0000010 \\
 0010000 \\
 0000000 \\
 \hline
 0010010
 \end{array}$$

bit pair recoding of multiplier:-

$$\begin{array}{cc}
 \text{Ex:- } +5 \times -8 \\
 A \quad B
 \end{array}$$

$$(5) A = 0101 \quad \text{2's complement of '8'}$$

$$(8) B = 1000$$

$$\begin{array}{r}
 1000 \\
 \downarrow \downarrow \downarrow \downarrow \\
 0001 \\
 \hline
 1000
 \end{array}$$

$$(+5) A = 00101 - \text{Multiplicand}$$

$$(-8) B = 11000 - \text{multiplier}$$

$$\begin{array}{ccccccc}
 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
 \hline
 & 0 & -2 & & & &
 \end{array}$$

Recoding values:-

$$\begin{array}{r}
 00101 \\
 0 \quad -2 \quad 0 \\
 \hline
 000000000000 \\
 11110010 \\
 000000 \\
 \hline
 1111001000
 \end{array}$$

2's complement -

$$\begin{array}{r}
 1111001000 \\
 \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \\
 0000000000 \\
 (+) 0000000000 \\
 \hline
 0000000000
 \end{array}$$

2's complement:-

$$\begin{array}{r}
 00101 \\
 \downarrow \downarrow \downarrow \downarrow \downarrow \\
 11010 \\
 \hline
 11010
 \end{array}$$

= 4

$$+9 \times -6$$

A B

$$A = 1001$$

$$B = 0110$$

2's Complement of 6

$$\begin{array}{r} 0110 \\ \downarrow \downarrow \downarrow \downarrow \\ 1001 \\ \hline 01 \\ \hline 1010 \end{array}$$

$$\rightarrow (+9) = 01001 - \text{multiplicand}$$

$$(-6) = 11010 - \text{multiplier}$$

$$\begin{array}{r} 111010 \\ \hline 0 \quad -1 \quad -2 \end{array}$$

$$\begin{array}{r} 01001 \\ \hline 0 \quad -1 \quad -2 \end{array}$$

$$\begin{array}{r} 1111101110 \\ 11110111 \\ \hline 0000000000 \\ \hline 1111001010 \end{array}$$

$$\begin{array}{r} 01001 \\ \downarrow \downarrow \downarrow \downarrow \downarrow \\ 10111 \\ \hline 10111 \\ \hline 00000 \\ 10111 \\ \hline 101110 \end{array}$$

2's Complement -

$$\begin{array}{r} 1111001010 \\ \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \downarrow \\ 0000110101 \\ \hline 01 \\ \hline 0000110110 \end{array}$$

$$32 + 16 + 4 + 2$$

$$\begin{array}{r} 20 \\ 30 \\ \hline 50 \end{array}$$

Carry Save Addition:-

$$11 \text{ \& } 15$$

$$11 = 1011$$

$$15 = 1111$$

$$11 \times 15 =$$

$$\begin{array}{r} 1011 \\ 1111 \\ \hline 01011 \quad A \\ 01011 \quad B \\ 0101100 \quad C \\ 1011000 \quad D \\ \hline 10100101 \end{array}$$

$$\begin{array}{r}
 A+B: \quad \textcircled{0} \\
 \quad 1011 \\
 \quad 10110 \\
 \hline
 11101 - s_1 \\
 00100 - c_1
 \end{array}$$

$$\begin{array}{r}
 c_1 + 0 = \textcircled{0} \\
 \quad 101100 \\
 \quad 1011000 \\
 \hline
 1110100 - s_2 \\
 0010000 - c_2
 \end{array}$$

$$\begin{array}{r}
 s_1 + c_1 + s_2 = \textcircled{0} \\
 \quad 11101 \\
 \quad 000100 \\
 \quad 1110100 \\
 \hline
 1101101 - s_3 \\
 0101000 - c_3
 \end{array}$$

$$\begin{array}{r}
 s_3 + c_3 + c_2 = \textcircled{0} \\
 \quad 1101101 \\
 \quad 0101000 \\
 \quad 0010000 \\
 \hline
 1010101 - s_4 \\
 1010000 - c_4 \\
 \hline
 10100101
 \end{array}$$

Integer division:-

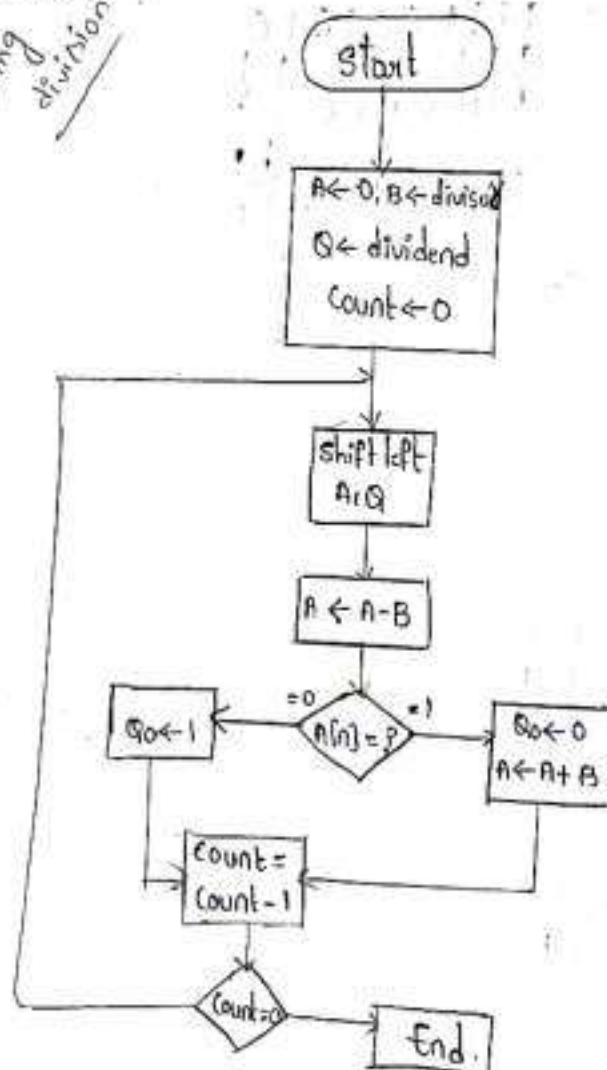
there are two types of divisions

① Restoring division

② non-restoring division

flowchart:-

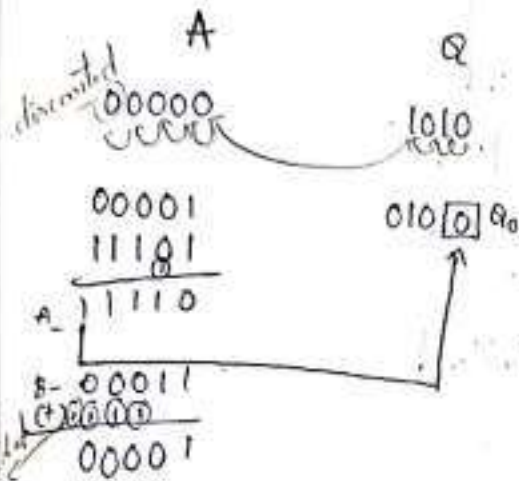
Restoring
division



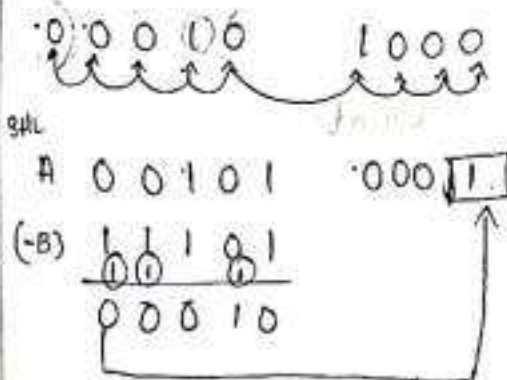
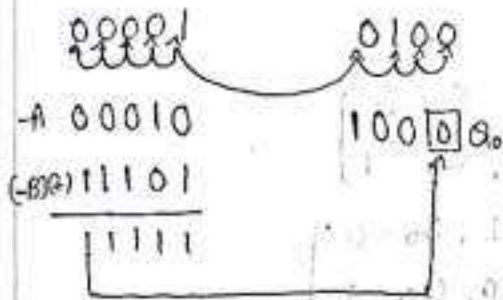
Ex:- Perform the integer division of 10 & 3.

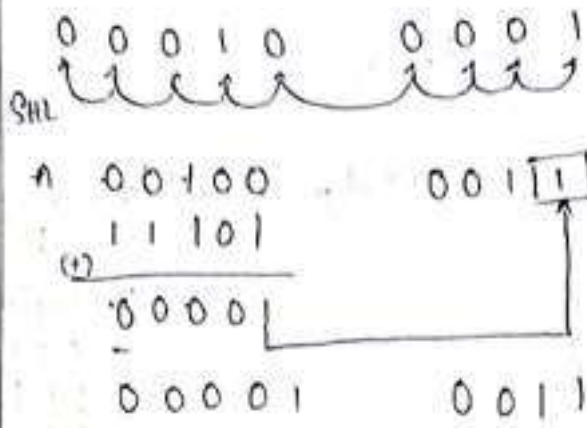
dividend
(a) 10 & 3 (b)
1010 0011

In 1st step 'A' must have (min) bits



count 2's complement
0 0011
↓ ↓ ↓ ↓
1 1 0 0
1 1 0 1

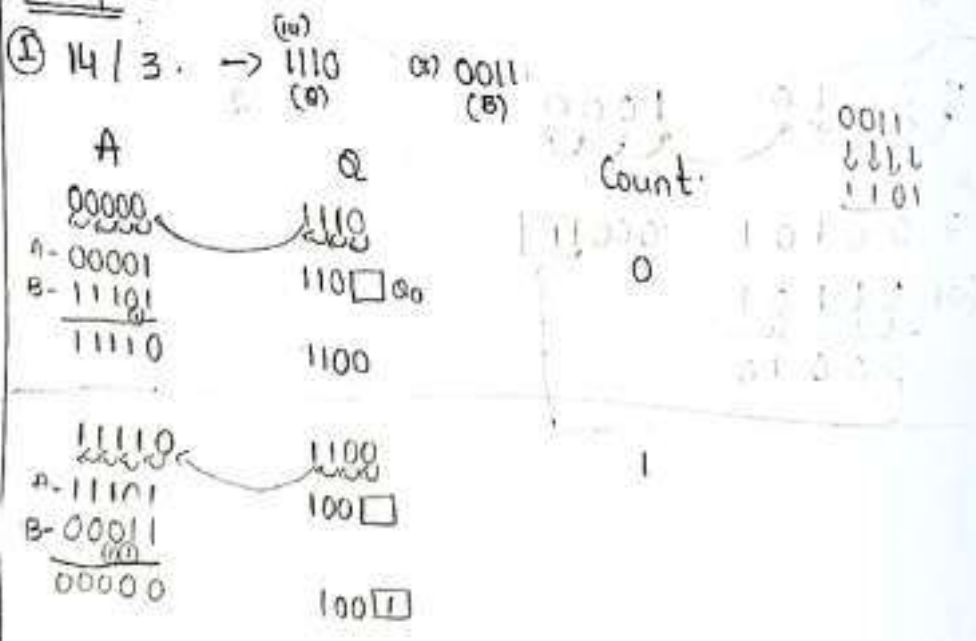




Non-Restoring Division:-

1. $A \leftarrow 0$, $B \leftarrow \text{Divisor}$, $Q \leftarrow \text{Dividend}$
2. $\text{Count} \leftarrow n$
3. Shift left A & Q
3. IF sign of A is '0'
 - Yes $\Rightarrow A \leftarrow A - B$ $\begin{cases} A \text{ is } 1, Q_0 = 0 \\ A \text{ is } 0, Q_0 = 1 \end{cases}$
 - No $\Rightarrow A \leftarrow A + B$ $\begin{cases} A \text{ is } 1, Q_0 = 0 \\ A \text{ is } 0, Q_0 = 1 \end{cases}$
4. Repeat 2 & 3 for 'n' times
5. IF sign of A is 1, add divisor to A.

Example:-



$$\begin{array}{r} 00000 \\ 00001 \\ 5-11101 \\ \hline 11110 \end{array}$$

$$\begin{array}{r} 1001 \\ 001 \square \\ 0010 \end{array}$$

2.

$$\begin{array}{r} 11110 \\ 11100 \\ 00011 \\ \hline 11111 \end{array}$$

$$\begin{array}{r} 0010 \\ 010 \square \end{array}$$

3

add binary 3 - 0011

$$\begin{array}{r} 11111 \\ 00011 \\ \hline 00010 \\ \downarrow \end{array}$$

$$0100$$

if the remainder is all 1's add binary 3 to the remainder to get the final remainder.

if the Quotient is all 1's add 1 to make the final quotient.

$$\frac{11}{11} \div 3$$

dividend (A)=11, divisor (B)=3

A: B Q: Count

$$\begin{array}{r} 00000 \\ 00001 \\ 11101 \\ \hline 11110 \end{array}$$

$$\begin{array}{r} 011 \\ 011 \square 0_0 \\ 011 \square 0_0 \end{array}$$

0

$$\begin{array}{r} 11110 \\ 11100 \\ 00011 \\ \hline 11111 \end{array}$$

$$\begin{array}{r} 0110 \\ 110 \square 0_0 \\ 110 \square 0_0 \end{array}$$

$$\begin{array}{r}
 11111 \\
 1100 \\
 \hline
 11111 \\
 00011 \\
 \hline
 00010
 \end{array}
 \quad
 \begin{array}{r}
 1100 \\
 100 \square 00 \\
 100 \boxed{1} 00
 \end{array}$$

2

$$\begin{array}{r}
 00010 \\
 00101 \\
 11101 \\
 \hline
 00010
 \end{array}
 \quad
 \begin{array}{r}
 1001 \\
 001 \square 00 \\
 0011
 \end{array}$$

3

↓
remainder
2

quotient
3

floating point numbers:-

To represent the fractional binary number it is necessary to consider binary point:-
 → the position of the binary point is variable therefore the binary point is said to be float and the numbers are called floating point numbers

$$B = \underbrace{b_0 \times 2^0}_{\text{Significant bit}} + \underbrace{b_{-1} \times 2^{-1} + b_{-2} \times 2^{-2} + \dots + b_{-(n-1)} \times 2^{-(n-1)}}_{\text{Scaling Factor}}$$

The floating point representation has three fields.

- 1) Sign bit
- 2) Significant bits (mantissa)
- 3) Exponent

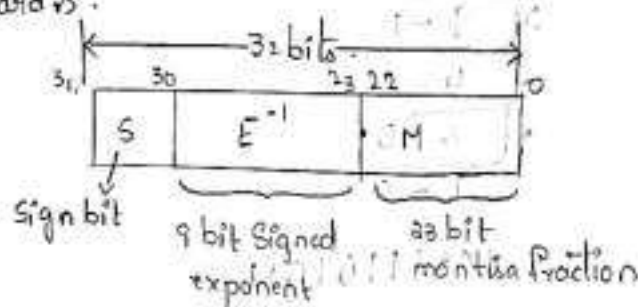
Example:-

$100111 \cdot 110110$
 $01 \cdot 00111110110 \times 2^5$
 sign bit significant digit exponent Scaling Factor
 normalized

② $11110000 \cdot 11101$
 $011 \cdot 1000011101 \times 2^3$
 sign bit significant digit exponent Scaling Factor
 normalized.

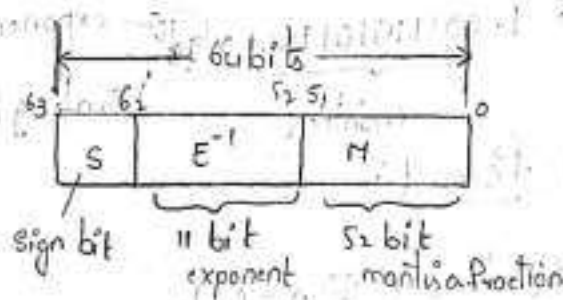
IEEE Standard for floating point numbers:-

The Institute of Electrical & Electronics Engineers have been developed the floating point representation in 32 bits and 64 bits which is referred as IEEE 754 Standards.



0: Signifies '+'
 1: Signifies '-'

(a) Single Precision Format



Here, $E^{-1} = E + \text{bias}$
 bias = added value.

(b) Double precision Format.

→ The 32 bit format has the bias of 127

→ The 64 bit format has bias of 1023.

Represent $(1259.125)_{10}$ in Single Precision & double precision format.

Steps:-

① Convert the decimal to binary format.

Integer part:-

$$\begin{array}{r}
 2 \overline{) 1259} \\
 \underline{2 \ 6029} \quad -1 \\
 2 \overline{) 3014} \quad -1 \\
 2 \overline{) 1507} \quad -0 \\
 2 \overline{) 753} \quad -1 \\
 2 \overline{) 376} \quad -1 \\
 2 \overline{) 188} \quad -0 \\
 2 \overline{) 94} \quad -0 \\
 2 \overline{) 47} \quad -0 \\
 2 \overline{) 23} \quad -1 \\
 2 \overline{) 11} \quad -1 \\
 2 \overline{) 5} \quad -1 \\
 2 \overline{) 2} \quad -1 \\
 1 \quad -0
 \end{array}$$

$$\begin{array}{r}
 2 \overline{) 1259} \\
 \underline{2 \ 609} \quad -1 \\
 2 \overline{) 314} \quad -1 \\
 2 \overline{) 157} \quad -0 \\
 2 \overline{) 78} \quad -1 \\
 2 \overline{) 39} \quad -0 \\
 2 \overline{) 19} \quad -1 \\
 2 \overline{) 9} \quad -1 \\
 2 \overline{) 4} \quad -1 \\
 2 \overline{) 2} \quad -0 \\
 1 \quad -0
 \end{array}$$

$$= 10011101011$$

Fractional part:-

$$0.125 \times 2 = 0.250 \rightarrow 0$$

$$0.250 \times 2 = 0.500 \rightarrow 0$$

$$0.500 \times 2 = 1.000 \rightarrow 1$$

$$= (0.001)_2$$

$$= (10011101011.001)_2$$

$$= \underbrace{1.0011101011001}_{\text{mantissa}} \times \underbrace{2^{10}}_{\text{scaling factor}} \quad \text{exponent}$$

Single Precision Format:-

$$E=10, S=0 \text{ [Pos '1' Sign = 0]}$$

$$\begin{array}{r}
 2 \overline{) 137} \\
 \underline{2 \ 68} \quad -1 \\
 2 \overline{) 34} \quad -0 \\
 2 \overline{) 17} \quad -0 \\
 2 \overline{) 8} \quad -1 \\
 2 \overline{) 4} \quad -0 \\
 2 \overline{) 2} \quad -0 \\
 1 \quad -0
 \end{array}$$

$$E' = E + \text{bias}$$

$$= 10 + 127 = 137$$

$$= 10001001$$

$$= \underbrace{0.10001001}_E \underbrace{0011101011001 \dots 0}_{\text{mantissa}}$$

Second Precision Format:-

$$E = 10, \quad S = 0$$

$$\begin{array}{r} 2 \overline{) 1033} \\ 2 \overline{) 516-1} \\ 2 \overline{) 258-0} \\ 2 \overline{) 129-0} \\ 2 \overline{) 64-1} \\ 2 \overline{) 32-0} \\ 2 \overline{) 16-0} \\ 2 \overline{) 8-0} \\ 2 \overline{) 4-0} \\ 2 \overline{) 2-0} \\ 2 \overline{) 1-0} \end{array}$$

$$E_1 = E + \text{bias}$$

$$E_1 = 10 + 1033$$

$$= 1033$$

$$= 10000001001$$

$$E' = 10000001001$$

$$= 0 \underbrace{10000001001}_{E'}$$

$$\underbrace{00111010101100 \dots}_{\text{mantissa}}$$

$$\begin{array}{r} 2 \overline{) 1033} \\ 2 \overline{) 516-1} \\ 2 \overline{) 258-0} \\ 2 \overline{) 129-0} \\ 2 \overline{) 64-1} \\ 2 \overline{) 32-0} \\ 2 \overline{) 16-0} \\ 2 \overline{) 8-0} \\ 2 \overline{) 4-0} \\ 2 \overline{) 2-0} \\ 2 \overline{) 1-0} \end{array}$$

Represent $-(307.1875)_{10}$ in Single & double Precision Format.

Step-1:- Convert the decimal into binary format.

Integer part

$$\begin{array}{r} 2 \overline{) 307} \\ 2 \overline{) 153-1} \\ 2 \overline{) 76-1} \\ 2 \overline{) 38-0} \\ 2 \overline{) 19-0} \\ 2 \overline{) 9-0} \\ 2 \overline{) 4-1} \\ 2 \overline{) 2-0} \\ 2 \overline{) 1-0} \end{array}$$

Fractional part:

$$0.1875 \times 2 = 0.3750 \rightarrow 0$$

$$0.375 \times 2 = 0.750 \rightarrow 0$$

$$0.75 \times 2 = 1.5 \rightarrow 1$$

$$0.5 \times 2 = 1.0 \rightarrow 1$$

$$1 = 0.0011 = 6 \times 7 = 0$$

$$= (100110011 \cdot 0011)_{2^{10-0}}$$

$$100110011$$

$$= 11 \cdot 001100110011 \times 2^{100}$$

signed bit mantissa scaling factor

Single precision:-

$$E = 8, \quad S = 1$$

$$\Rightarrow E' = E + \text{bias} \quad (32 \text{ bits})$$

$$E' = 8 + 127 = 135$$

$$= 10000111$$

$$= \underbrace{1}_{\text{Sign bit}} \underbrace{0000111}_{E'} \underbrace{001100110011 \dots}_{\text{mantissa}} 0$$

$$\begin{array}{r} 2 \overline{) 135} \\ 2 \overline{) 64-1} \\ 2 \overline{) 33-1} \\ 2 \overline{) 16-1} \\ 2 \overline{) 8-0} \\ 2 \overline{) 4-0} \\ 2 \overline{) 2-0} \\ 2 \overline{) 1-0} \end{array}$$

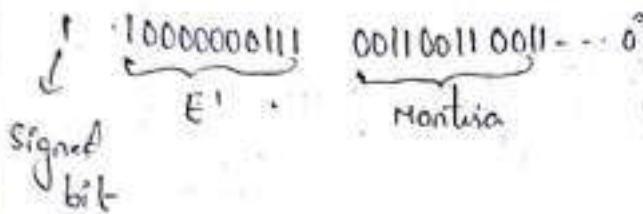
Double Precision:

$$F=6, S=1$$

$$(60b1) \Rightarrow E' = [1608$$

$$= 8 + 1023 = 1031$$

$$= 10000000111$$


$$\begin{array}{r} 2 \overline{) 1031} \\ \underline{2} 515 - 1 \\ \underline{2} 297 - 1 \\ \underline{1} 25 - 1 \\ \underline{0} 64 - 0 \\ \underline{3} 2 - 0 \\ \underline{1} 6 - 0 \\ \underline{8} - 0 \\ \underline{4} - 0 \\ \underline{2} - 0 \\ 1 - 0 \end{array}$$

(ii) $(0.0625)_{10}$

① Convert the decimal into binary.

$$0.0625 \times 2 = 0.125 \rightarrow 0$$

$$0.125 \times 2 = 0.25 \rightarrow 0$$

$$0.25 \times 2 = 0.5 \rightarrow 0$$

$$0.5 \times 2 = 1.0 \rightarrow 1$$

$$= (0.0001)_{10} \cdot 10000(0.01)_{10} =$$

$$= 0.0001 \times 2^{\text{exponent}}$$

Single Precision:-

$$\mathcal{I} \neq \emptyset, \quad \mathcal{I} \neq \emptyset$$

$$E' = E + b\bar{c}a_s$$

$$= 0 + 107 = 107$$

2. 11111111

$$0-11111-0001-----0^9$$

$$\begin{array}{r} 2 \overline{) 127} \\ \underline{2} 63 - \\ \underline{2} 31 - \\ \underline{2} 15 - \\ \underline{2} 7 - \\ \underline{2} 3 - \\ 1 - \end{array}$$

$$= \underbrace{0}_{\text{Signed bit}} \underbrace{111111}_{E'} \underbrace{0001}_{\text{Mantissa}} \dots 0^{\circ}$$

Double Precision:-

$$S=0, E=0$$

$$E' = E + \text{bias} \quad (6 \text{ bits})$$

$$= 0 + 1023$$

$$= 1023$$

$$= 1111111111$$

$$0 \underbrace{1111111111}_{E'} \underbrace{0001}_{\text{Mantissa}} \dots 0^{\circ}$$

2	1023
2	511 - 1
2	255 - 1
2	127 - 1
2	63 - 1
2	31 - 1
2	15 - 1
2	7 - 1
2	3 - 1
2	1 - 1

consider two floating point numbers $X = M_1 \times E_1$, $Y = M_2 \times E_2$

Assume $E_1 \geq E_2$

rules:-

Step-①:- Select the number with a smaller exponent and shift its mantissa to right, a number of steps equal to the difference in exponents $|E_2 - E_1|$

Step-②:- Set the exponent of the result equal to the larger exponent

Step-③:- Perform addition / Subtraction on the mantissas and determine the sign of the result.

Step-④:- Normalise the result if necessary.

Ex:-

Add Single Precision Floating point numbers A & B

where $A = 44900000H$ & $B = 48A00000H$
heradecimal number

Sol Steps:-

① represent the numbers in Single Precision Format

$$A = 44900000 = \underbrace{01000100}_s \underbrace{100100000}_{\text{mantissa}} \dots 0$$

$\downarrow \quad \downarrow \quad \downarrow$
 0100 0100 1001

$$B = 48A00000 = \underbrace{01000010}_s \underbrace{101000000}_{\text{mantissa}} \dots 0$$

$\downarrow \quad \downarrow \quad \downarrow$
 0100 0010 1010

Exponent for A = $10001001 = 137 = E'$

$$E' = E + \text{bias}$$

$$E = E' - \text{bias} = 137 - 127 = 10$$

Exponent for B = $10000101 = 133 = E'$

$$E = E' - \text{bias}$$

$$E = 133 - 127 = 6$$

$$\begin{array}{r}
 2^7 \ 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0 \\
 \hline
 128 \ 64 \ 32 \ 16 \ 8 \ 4 \ 2 \ 1 \\
 \hline
 10 \times 8 \\
 \hline
 80 \\
 \hline
 133
 \end{array}$$

② Mantissa of B = $01000000 \dots 0$

$\underbrace{0000} \ 01000000 \dots 0$

< shift the mantissa of B by 4 bits

Step 3:-

Add mantissas of A & B

Mantissa of A = $00100000 \dots 0$

Mantissa of B = $00000100 \dots 0$

$$\begin{array}{r}
 00100000 \\
 (+) \ 00000100 \\
 \hline
 00100100
 \end{array}$$

replace

Add the result mantissa into the mantissa of A

$$= \frac{0}{4} \frac{10001001}{4} \frac{00100100}{9} \frac{\dots}{2} \dots 0$$

$$= (449200000H)$$

required hexadecimal number.

Ex: Subtract the single precision floating point numbers A & B where $A = 44900000H$ & $B = 42A00000H$

Step-1 represent the number in the single precision format

$$A = 44900000H = \frac{0}{5} \frac{10001001}{\text{exponent}} \frac{00100000 \dots}{\text{mantissa}} 0^{\circ}$$

$$B = 42A00000H = \frac{0}{7} \frac{10000101}{\text{exponent}} \frac{01000000 \dots}{\text{mantissa}} 0^{\circ}$$

Exponent for A = $10001001 = 137 = E'$

$$E' = E + \text{bias}$$

$$E = E' - \text{bias} = 137 - 127 = 10$$

Exponent for B = $10000101 = 133 = E'$

$$E = E' - \text{bias}$$

$$E = 133 - 127 = 6$$

2) Mantissa of B = $01000000 \dots 0^{\circ}$

$$\frac{0000}{\text{mantissa}} \frac{01000000 \dots}{\text{mantissa}} 0^{\circ}$$

← Add 4 zeros to mantissa of B because mantissa of A is 10 & B is 6.

3) Subtract mantissas of A & B.

Mantissa of A = $00100000 \dots 0^{\circ}$

$$\leftarrow \frac{00000100 \dots}{\text{result of mantissa}} 0^{\circ}$$

$$\text{result of mantissa } 00011100 \dots 0^{\circ}$$

$$\leftarrow 0 - 0 = 0$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$0 - 1 = \text{diff} - 1$$

borrow 1

4) replace mantissa of A with result mantissa

$$\frac{0}{4} \frac{10001001}{4} \frac{00011100}{8} \frac{\dots}{E} 0^{\circ}$$

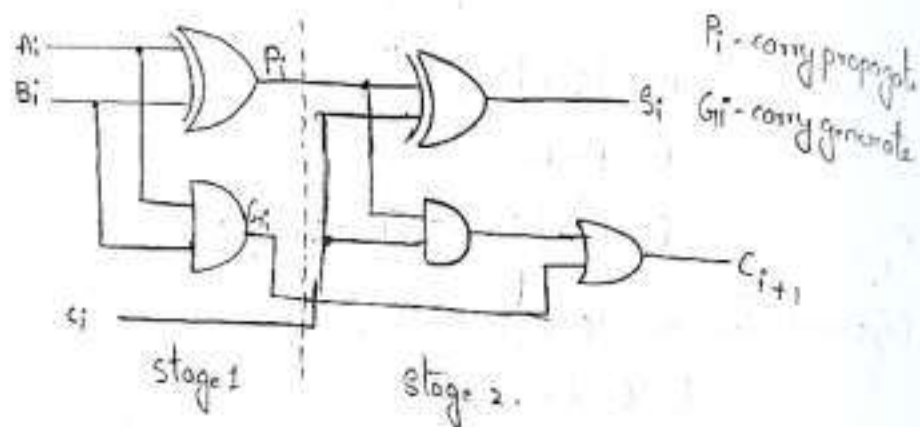
$$= (448E00000H)$$

required hexadecimal number.

Design of fast adder:-

Carry look ahead generator/adder:-

- In a full adder the carry output of each full adder is connected to the carry input of the next stage.
- The sum & the carry outputs of any stage cannot be produced until the input carry occurs, this leads to a time delay in the addition process. This delay is known as carry propagation delay.
- Consider the circuit of full adder.



$$P_i = A_i \oplus B_i$$

$$G_i = A_i \cdot B_i$$

$$\text{O/p Sum } S_i = P_i \oplus C_i$$

$$\text{O/p Carry } C_{i+1} = G_i + P_i \cdot C_i$$

$$i=0, C_1 = G_0 + P_0 C_{in}$$

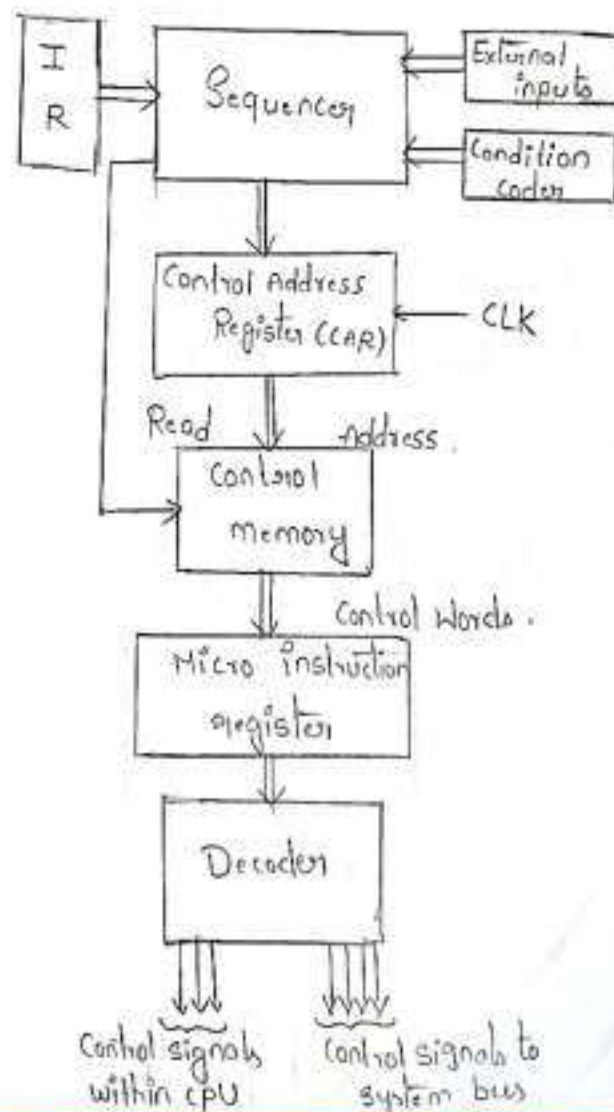
$$i=1, C_2 = G_1 + P_1 C_1$$
$$= G_1 + P_1 G_0 + P_1 P_0 C_{in}$$

$$i=2, C_3 = G_2 + P_2 C_2$$

$$= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$$

Micro Programmed Control:-

- Micro Programming:- It is a method of Control unit design in which the Control Signal Selection and Sequencing information is stored in a RAM or ROM called a CM (Control Memory).
- The Control Signals to be activated at any time or specified by a micro instruction.
 - A Sequence of 1 or more micro operations designed to control specific operations such as addition, multiplication etc is called a micro Program.



- The control address register holds the address of the next instruction to be read. every time a new instruction is loaded into IR, the output of the Sequencer is loaded in CAR (Control Address register).
- When address is available in CAR, the Sequencer issues Read Command Control memory (CM).
- After issue of read, the word from the addressed location is read into micro instruction register.
- The control address register (CAR) is automatically increases by the clock and the contents of micro instruction register generates control signals which are delivered to various parts of the processor in the correct sequence.

The Memory System

Mr. C. J. Brown

The Memory System [Unit-III] ①

Basic Concepts, Semiconductor RAM Memories, Read-Only Memories, Speed, Size and Cost, Cache Memories, Performance Considerations, Virtual Memories, Memory Management Requirements, Secondary Storage.

Some Basic Concepts:-

The maximum size of the memory that can be used in any Computer is determined by the addressing Scheme.

For eg. a 16-bit Computer that generates 16-bit addresses is capable of addressing upto $2^{16} = 64K$ memory locations. Similarly machines whose instructions generate 32-bit addresses can utilize a memory that contains $2^{32} = 4G$ (giga) memory locations. Similarly 40-bit $\Rightarrow 2^{40} = 1T$ (tera) locations.

Most modern Computers are byte addressable (Show in fig) shows the possible address assignment for a byte-addressable 32-bit Computer.

The memory is usually designed to store and retrieve data in word-length quantities. In fact, number of bits actually stored or retrieved in one memory access is the most common definition of the word

length of a Computer.

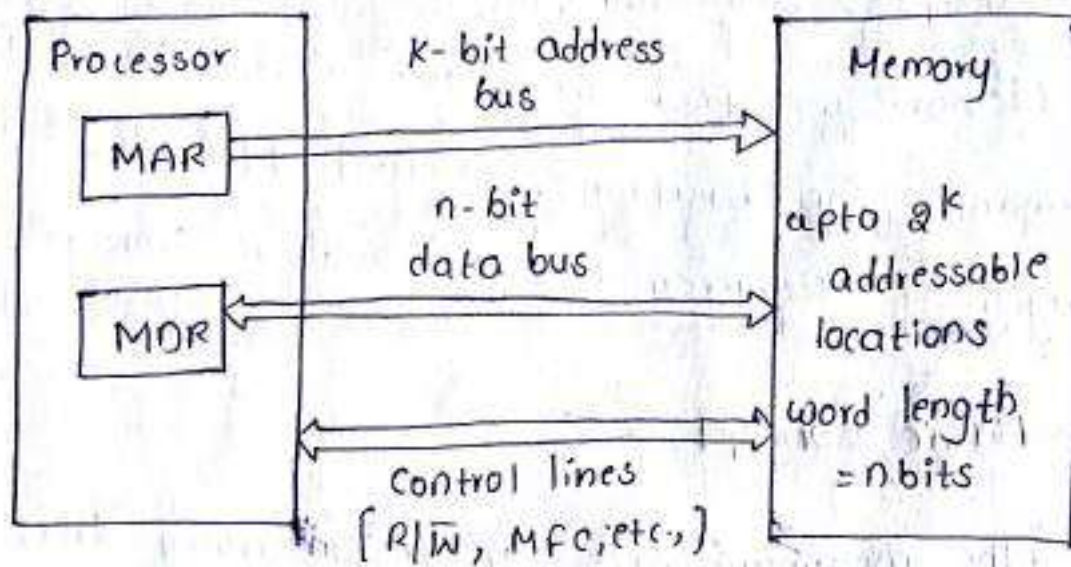


Fig: Connection of memory to the processor

From the above fig, it shows that the memory and processor are represented in two different blocks. Data transfer between the memory and the processor takes place through the use of two processor registers, usually called MAR (Memory Address Register) and MDR (Memory Data Register).

If MAR is k -bits long then MDR is n -bits long. then the memory unit may contain up to 2^k addressable locations. During the memory cycle, n bits of data are transferred between the memory and the processor.

This transfer takes place over the processor bus, which has k -address lines and n -data lines. The bus also includes $R/\bar{W}$ and memory function

Completed (MFC) for co-ordinating data transfers. (9)

If read or write operations involve consecutive address locations in the main memory, then a "block transfer" operation can be performed. In which the only address sent to the memory is the one that identifies the first location.

A useful measure of the speed of the memory units is the time that elapses between the initiation of an operation and the completion of that operation. This can be referred to as "memory access time".

Another important measure is the "memory cycle time", in which is the minimum time delay required between the initiation of two successive memory operations.

One way to reduce the memory access time is to use a "cache memory". This is a small, fast memory that is inserted between the larger, slower main memory and the processor. It holds the currently active segment of a program and their data.

The address generated by the processor is referred to as "virtual" or "logic address." The mapping function is implemented by a special memory control circuit,

often called the "memory management Unit".

Semiconductor RAM memories:-

The semiconductor memory includes a memory cell array including a no. of memory cells arranged in a rows and columns, a no. of word lines each connecting all memories cells in a row and a no. of bit cells each one connecting all memory cells of a column.

Semiconductor memories are available in wide range of speeds. Their cycles times range from 10ns to less than 1ns. When first introduced in the late 1960's they were much more expensive than the magnetic core memories they replaced. Because of rapid advances in VLSI [very Large Scale Integration] technology, the cost of semiconductor memories has dropped dramatically.

Internal Organization of Memory Chips:-

Memory cells are usually organized in the form of array, in which each cell is capable of storing one bit of information.

Each row of cells constitute of a memory word and all cells of a row are connected to a common line referred to as the "word line" which is driven

During the read operation, these circuit sense, or read, the information stored in the cells selected by a word line and transmits the information to the output data lines.

the Output data lines.

During write Operation the sense/write circuit receive input information and store in the cells of the selected word.

Static Memories:-

Memories that consists of circuits capable of retaining their state as long as power is applied are known as "Static Memories".

The fig illustrated how Static Ram or SRAM cell may be implemented. Two inverters are cross-connected to form a latch. The latch is connected to two bit lines by transistors T_1 and T_2 . These transistors act as switches that can be opened or closed under control of the word line. When the word line is at the ground level, the transistor are turned off and the latch retains its state.

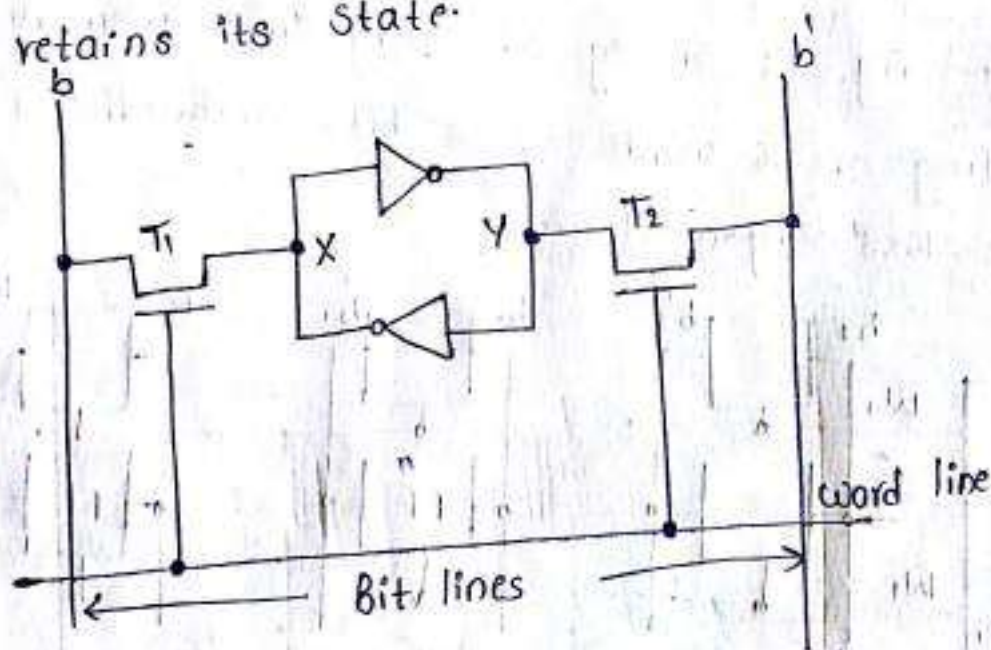


fig: A Static RAM cell.

Read Operation:-

In order to read the state of the SRAM cell, the word line is activated to close switches T_1 and T_2 . If the cell is in state 1, the signal on bit line b is high and

and the signal on bit line b' is low. The opposite is true if the cell is in state 0. Thus b and b' are complements of each other. (4)

Write Operation:-

The state of the cell is set by placing the appropriate value on the bit line b and its complement on b' , and then activating the word line. This forces the cell into corresponding state. The required signals on the bit lines are generated by the Sense/Write Circuit.

CMOS Cell:-

The CMOS realization of the memory cell is given below the fig. Transistor pairs (T_3, T_5) and (T_4, T_6) from the inverters in the latch. The state of the cell is read or written as just explained.

For eg. in state 1, the voltage at point x is maintained by high volt by having transistors T_3 & T_6 on, while T_4 & T_5 are off. Thus if T_1 & T_2 are turned on [closed], bit lines b & b' will have high and low signals respectively.

The power supply voltage V_{supply} is 5V in older CMOS SRAMs or 3.3V is new low-voltage

versions. If the power is interrupted the cell's content will be lost. When power is restored, the latch will settle into stable state, but it will not necessarily be the same state the cell was in before the interruption.

Hence, SRAMs are said to be "Volatile memories" because their contents are lost when power is interrupted.

A major advantage of CMOS SRAMs is their very low power consumption because current flows in the cell only when the cell is being accessed.

SRAMs can be accessed very quickly. Access time of just a few nanoseconds are found commercially available chips.

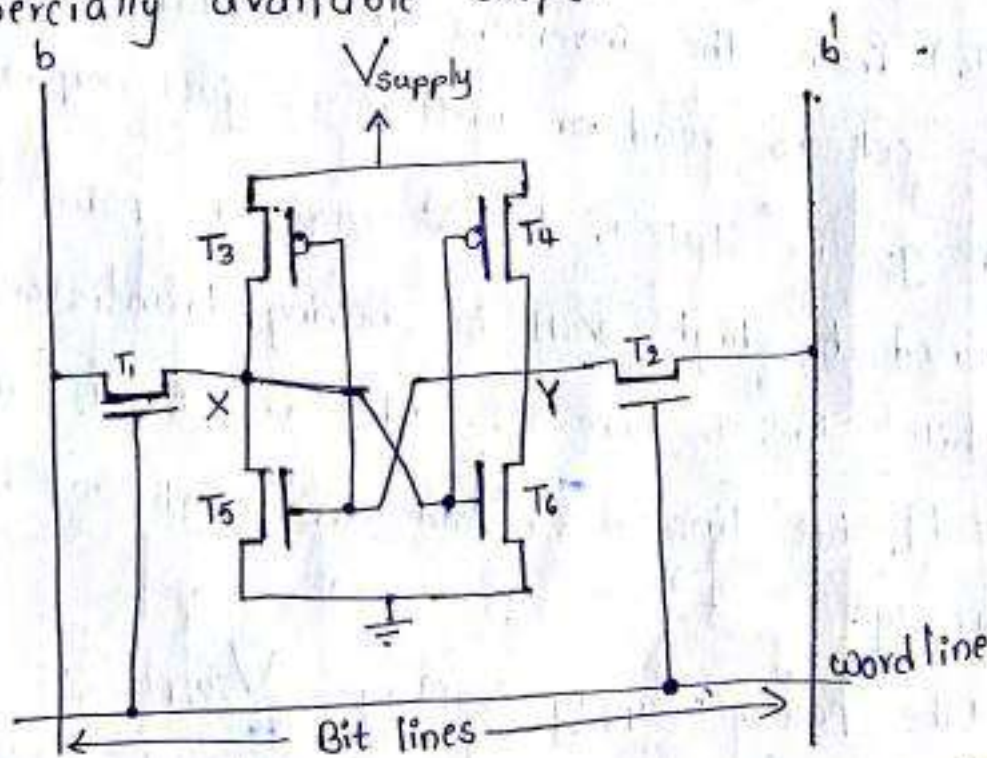


Fig: CMOS cell. (Complementary metaloxide semi-conductor)

Asynchronous DRAMs:-

Static RAM's are fast, but they come at high cost because their cells require several transistors. Less expensive RAM's can be implemented if simpler cells are used. However, such cells do not retain their state indefinitely; hence they are called "dynamic RAM's" (DRAM's).

Information stored in DRAM memory cell is in the form of a charge on a capacitor, and this charge can be maintained for only ten of milliseconds. Since, the cell is required to store information for much more longer time.

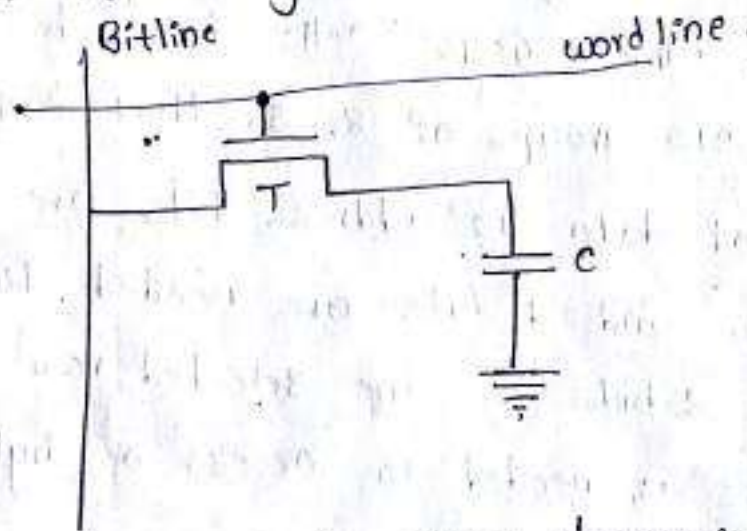
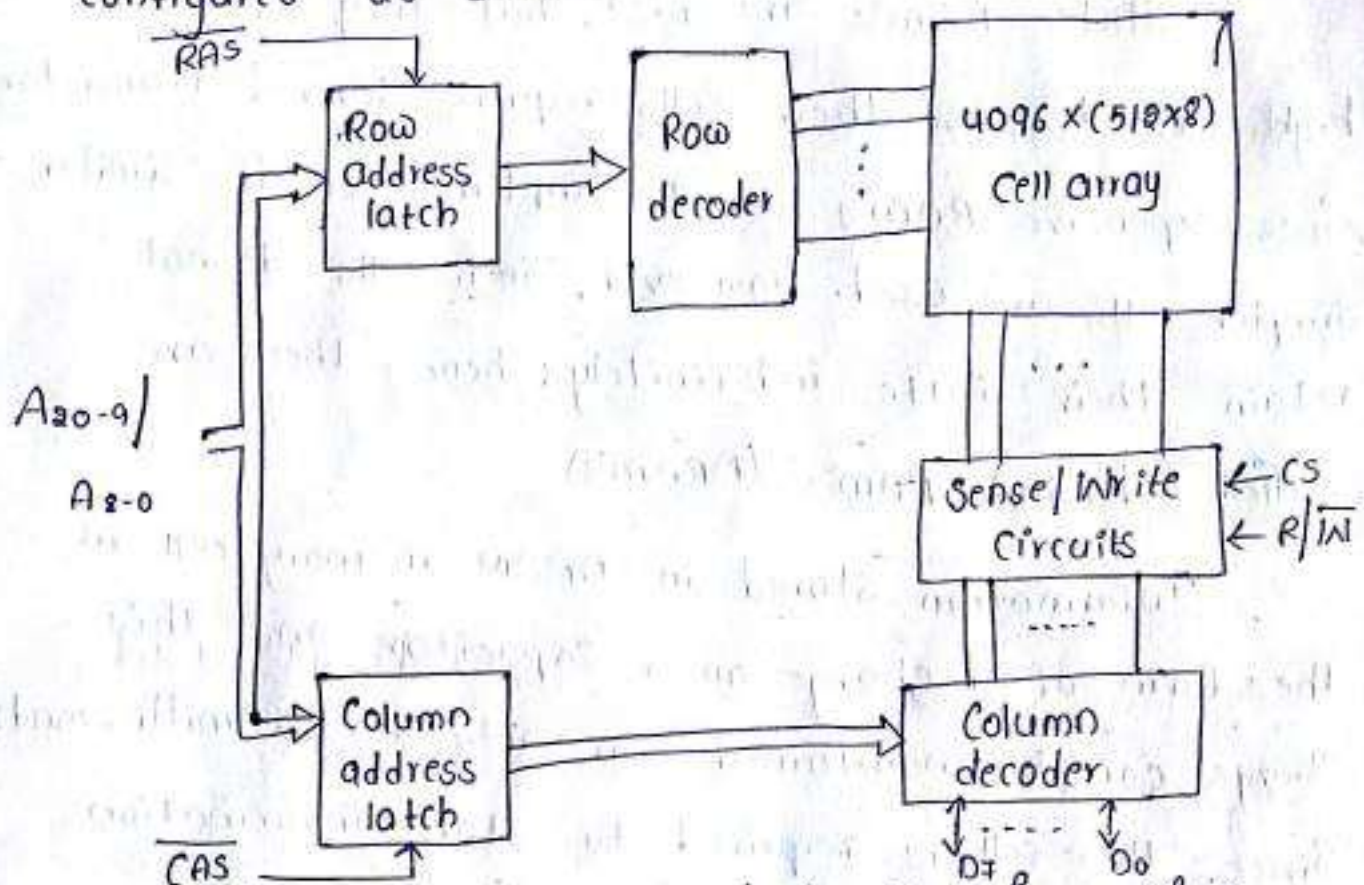


fig : A single transistor dynamic memory cell.

The above fig shows A single transistor dynamic memory cell. T is turned on and an appropriate voltage is applied to the bit line. This causes a known amount of charge to be stored in capacitor.

The following fig shows the 16-Mega bit DRAM chip,

Configured as 2Mx8.



The cells are organized in the form of a 4Kx4K Array. The 4096 cells in each row are divided into 512 groups of 8, so that each row can store 512B of data. 12 address bits are needed to select a row and 9-bits are needed to specify a group of 8-bits in the selected row. Thus, a 21-bit address is needed to access a byte in this memory.

The higher order 12-bits and low-order of 9-bits of the address constitute the row & Column address of a byte. During a read/write operation, the row address is applied first.

It is loaded into the row address latch is referred to a signal pulse on the "Row Address Strobe (RAS)" input of the chip. Then Read Operation is initiated, in which all cells on the selected row are read & refreshed. (6)

After the row address is loaded, the Column is applied to the address pins are loaded into the Column address latch under the control of "Column Address Strobe (CAS)" signal.

The information in the latch are decoded and the appropriate group of 8 sense/write are selected. If the control signal indicates the Read Operation the o/p values of the selected circuits are transferred to the datalines D_7-0 . For write operation the information on the D_7-0 lines is transferred to the selected circuits. This information is then used to overwrite the contents of the selected cells in the corresponding 8-columns.

The timing of the memory device is controlled asynchronously. A specialized memory controller circuit provides the necessary control signals. RAS & CAS signals. The processor must take into account the delay in the response of the memory. Such memories are referred to as "Asynchronous DRAM's".

Advantages :-

- ⇒ It has high density & low cost, these are widely used.
- ⇒ Available Chips are of size 1M to 256M bits & larger chips are developed.
- ⇒ A DRAM chip is Organized to read/write a number of bits in parallel.
- ⇒ It provides flexibility in designing memory systems.

Fast Page Mode :-

When the DRAM is accessed, the contents of all 4096 cells in the selected row are sensed, but only 8-bits are placed on the data lines D_7-0 . This byte is selected by the column address bits A_8-0 . A simple modification can make it possible that is a latch can be added to the o/p of the sense amplifier in each column.

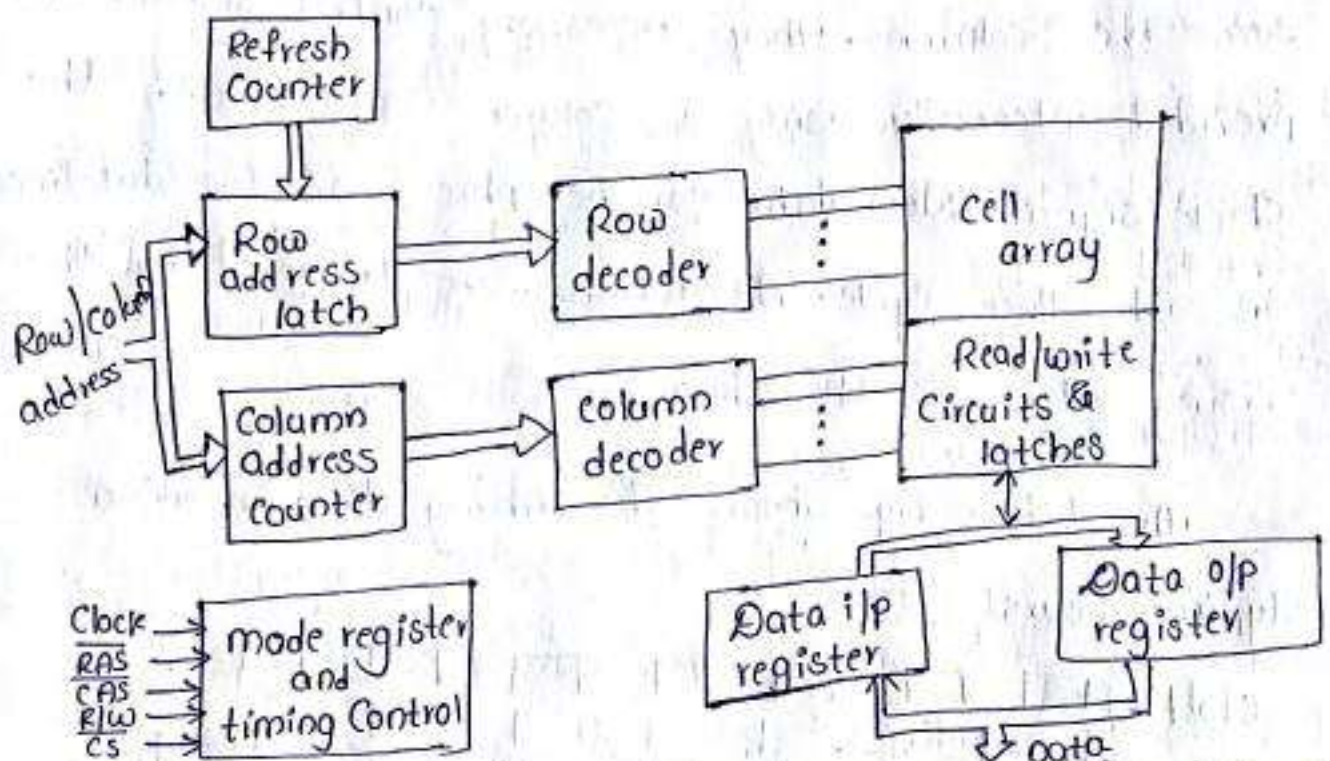
The most useful arrangement is to transfer the bytes in sequential order, which is achieved by applying a consecutive sequence column addresses under the control of successive CAS signals. This scheme allows transferring a block of data at a much faster rather than can be achieved for transfers involving random addresses.

The block transfer capability is referred to as "Fast Page Mode" features. ⑦

Synchronous DRAMs:-

More recently developments in memory technology have resulted in DRAMs whose operation is directly synchronized with a clock signal. Such memories are known as Synchronous DRAMs (SDRAMs).

The below fig shows the structure of SDRAM:-



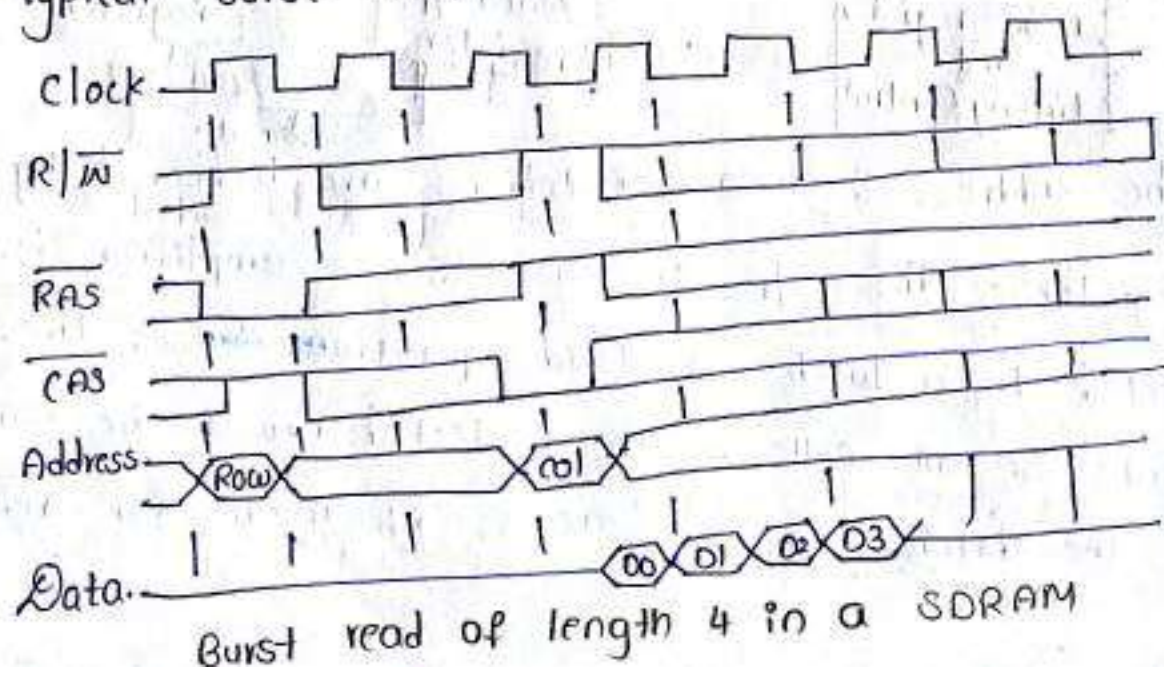
The address & data connections are buffered by means of registers. The o/p of each sense amplifier is connected to a latch. A Read operation causes the contents of all cells in the selected row to be loaded into the latches. If an access is made for refreshing

purposes but it will not change the contents of these latches and it will refresh the contents of the cell. Data held in the latches and it will that corresponds to the selected columns are transferred into the data o/p register.

SRAM's have several different modes of operation, which can be selected by using control information into "mode" register.

IR In SDRAM's it is not necessary to provide externally generated pulses on the cas line to select successive columns. The necessary control signals are provided internally using a column counter and the clock signal. New data can be placed on the datalines in each clock cycle. All actions are triggered by the rising edge of the clock.

The below fig. Shows the timing diagram of a typical burst read.



First, the row address is latched under control of the $\overline{\text{RAS}}$ signal. The memory typically takes 2 or 3 clock cycles. (8)

Latency & Band Width:-

Transfer between the memory and the processor involve single words for data or small blocks of words. Large blocks, constituting a page of data are transferred between the memory and the disks. The speed and efficiency of these transfers have large impact on the performance of the computer system. A good indication of the performance is given by two parameters: "Latency & Bandwidth".

The term "memory Latency" is used to refer the amount of time it takes to transfer a word of data to or from the memory. In this case of reading or writing a single word of data, the latency provides a complete indication of memory performance.

When transferring blocks of data, it is of interest to know how much time is needed to transfer entire block. Since blocks can be variable in size, it is useful to define a performance measure in

terms of number of bits or bytes that can be transferred in one second. This measure is often referred as 'Bandwidth'. The bandwidth of a memory unit depends on the speed of access to the stored data on the number of bits that can be accessed in parallel.

Double Data Rate SDRAM:-

The standard SDRAM perform all actions on the rising edge of the clock signal. A similar memory device is available, which accesses the cell array in the same way, but transfers data on both edges of the clock. The latency of these devices is the same for standard SDRAM's. But, since they transfer data on both edges of the clock, their bandwidth is essentially doubled for long burst transfers. Such devices are known as "Double Data Rate SDRAM's) or (DDR SDRAM's)

Structure of Larger Memories:-

Static Memory System:-

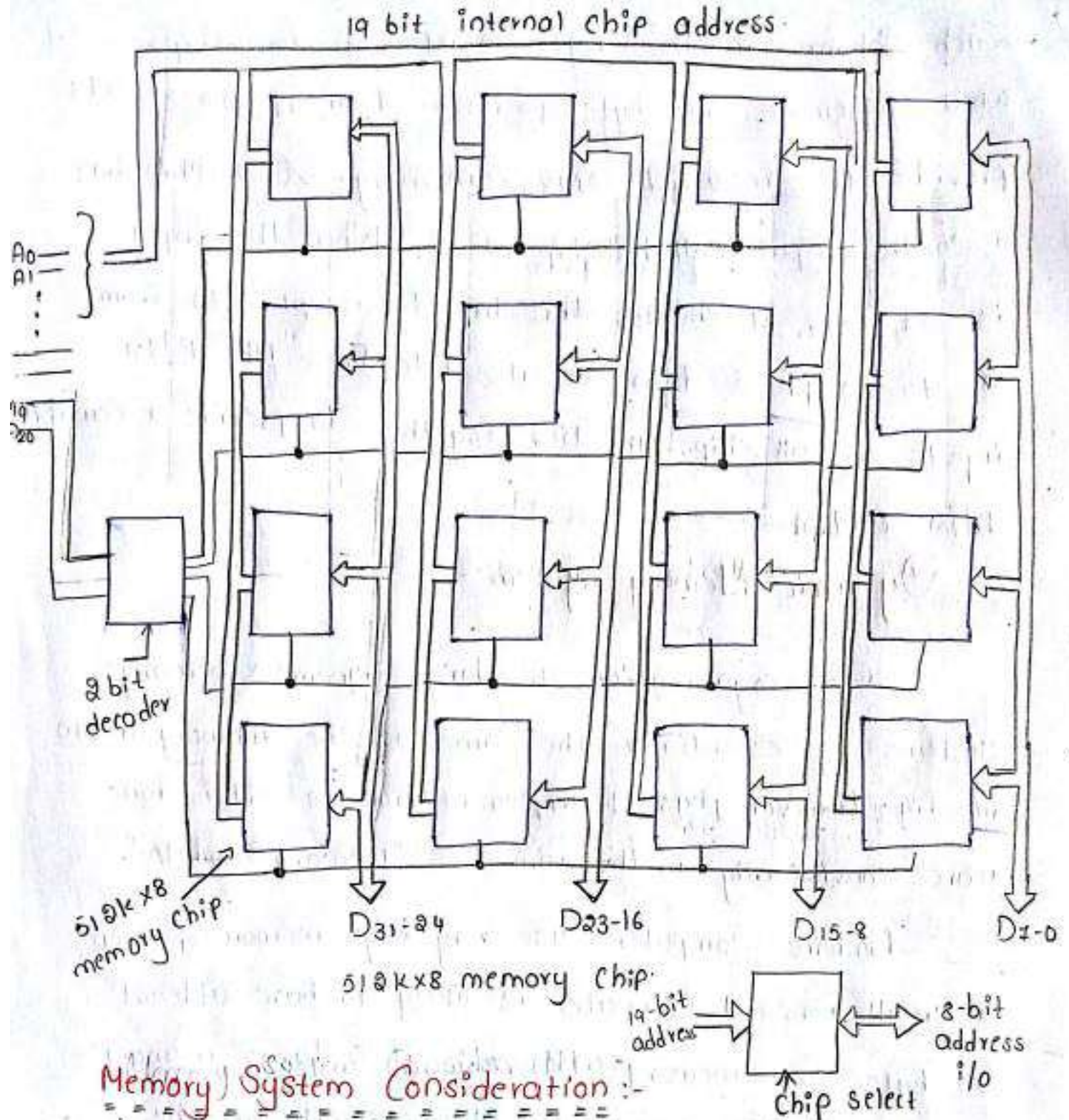
Consider a memory consisting of 2^M ($2,048,150$) words of 32 bit each. The fig. shows how we can implement this memory using $512K \times 8$ Static memory chips.

Each Column in the fig Consists of four chips. (a)
which implement one byte position. Four of these sets
provide the required $2M \times 32$ memory. Each chip has
a control input called "Chip Select". When the input
is set to 1, it enables the chip to accept data from
or to place data on its data lines. The $R/\overline{W}$
inputs of all chips are tied together to provide a common
 $R/\overline{W}$ control.

Dynamic Memory System:-

The organization of large dynamic memory
Systems is essentially the same as the memory shown
in fig. However, physical implementation is often done
more conveniently in the form of "memory modules".

Moderns Computers use very large memories; even
a small personal Computer is likely to have atleast
 32 bytes of memory. However, if a large memory
is built by placing DRAM chips directly on the
main memory system printed-circuit board that
contains the processor, often referred to as "mother
Board", it will occupy an unacceptably large amount
of space on the board.



Memory System Consideration:-

The choice of a RAM chip for a given application depends on several factors. Foremost among these factors are the cost, power dissipation, and size of the chip.

SRAM's are generally used only when very fast operation is the primary requirement. DRAM's are

the predominant choice for implementing computer main memories. The high densities achievable in these chips make large memories economically feasible.

Memory Controller:-

To reduce the no. of pins, the dynamic memory chips use multiplexed address inputs. The address is divided into two parts. The higher-order address bits, which select a row in the cell array, are provided first and latched into the memory chip under control of the RAS signal. Then the lower order address bits, which select a column, are provided on the same address pins and latched together CAS signal.

A typical processor issues all bits of an address at the same time. The required multiplexing of address bits is usually performed by a memory controller circuit, which is interposed between the processor and the dynamic memory as shown in fig. The controller accepts a complete address and the $R/\overline{W}$ signal from the processor under control of a Request signal which indicates that a memory access operation is needed. The controller provides the RAS-CAS timing, in addition

to its address multiplexing function. It also sends the $R/\overline{W}$ and CS signals to the memory.

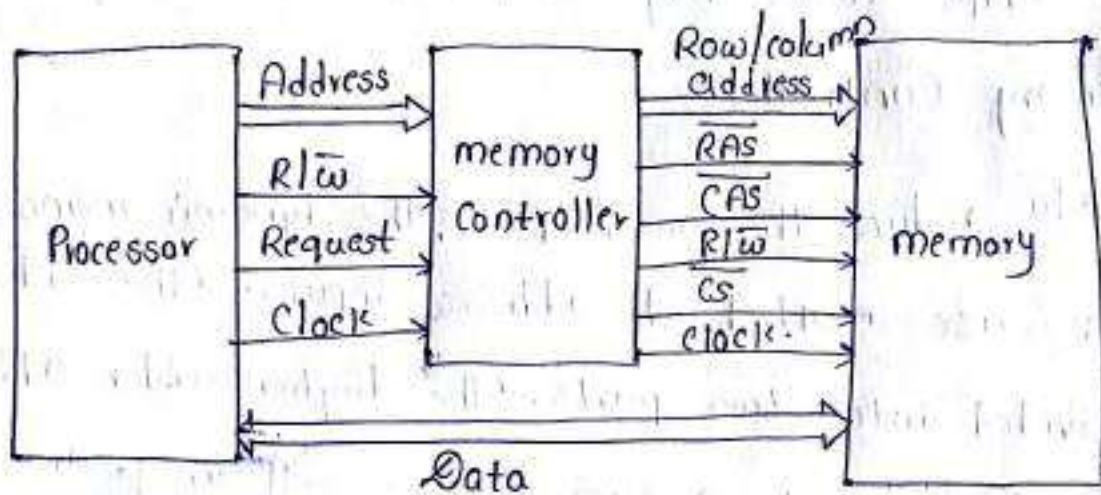


Fig use of a memory Controller

The CS signal is usually active low, hence it is shown as $\overline{CS}$ in the fig. Data lines are connected directly between the processor and the memory.

Refresh Overhead: Rambus Memory:-

The performance of a dynamic memory is characterized by its latency and bandwidth. Since all dynamic memory chips are similar organizations for their cell arrays, their latencies are tend to be similar if the chips are produced using the same manufacturing process.

DDR, SDRAM's and Standrand SDRAM's are connected to the processor bus. Thus, the speed of transfers is not just a function of the speed

of the memory device - it also depends on the speed of the bus. (11)

A very wide bus is expensive and requires a lot of space on the motherboard. An alternative approach is to implement a narrow bus that is much faster. This approach was very used by Rambus to develop a proprietary design known as "Rambus". The key feature of Rambus technology is a fast signalling method used to transfer information between chips.

The reference voltage is about ± 0.3 V and the two logic values are represented by 0.3 V swings above and below V_{ref} . This type of signalling is generally known as "differential Signalling".

Differential signaling and high transmission rates require special techniques for the design of wire connections that serve as communication links. These requirements make it difficult to make the bus wide. It is also necessary to design special circuit interfaces to deal with the differential signals. Rambus provides a complete specification for the design of such communication links, called the "Rambus Channel".

Circuitry needed to interface to the Rambus Channel is included on the chip. Such chips are known as "Rambus DRAM's (RDRAM's)".

The original Specification of Rambus provided for a Channel Consisting of 9 data lines and no. of Control and power supply. 8 of data lines are intended for transferring a byte of data. The 9th data line can be used for purposes such as parity checking. A 2-channel Rambus is also known as "Direct RDRAM, has 18 lines intended. (transfers 2 bytes of data).

Communication b/w the processor, or some other device that can serve as a master, and RDRAM modules, which serve as 'Slaves', is carried out by means of packets transmitted on the data lines. There are 3 types of packets; request, acknowledge, and data.

Rambus technology competes directly with the DDR and SDRAM technology. Each has certain advantages & disadvantages. Finally ~~as~~ in the memory market, assuming that the performance is adequate, the decisive factor is often the price of components.

Read Only Memories:-

(12)

Both SRAM and DRAM chips are volatile, which means that they lose the stored information if power is turned off. There are different types of ROM's they are:- PROM, EPROM, EEPROM.

ROM:-

The below fig shows the Configuration for a ROM cell. A logic value "0" is stored in the cell if the transistor is connected to ground P, otherwise a "1" is stored. The bit line is connected through a resistor to the power supply. To read the state of the cell, the word line is activated. Thus the transistor switch is closed and the voltage on the bit line drops to near zero if there is a connection b/w the transistor and ground. A sense circuit at the end of the bit line generates the proper output value. Data are written into a ROM when it is manufactured.

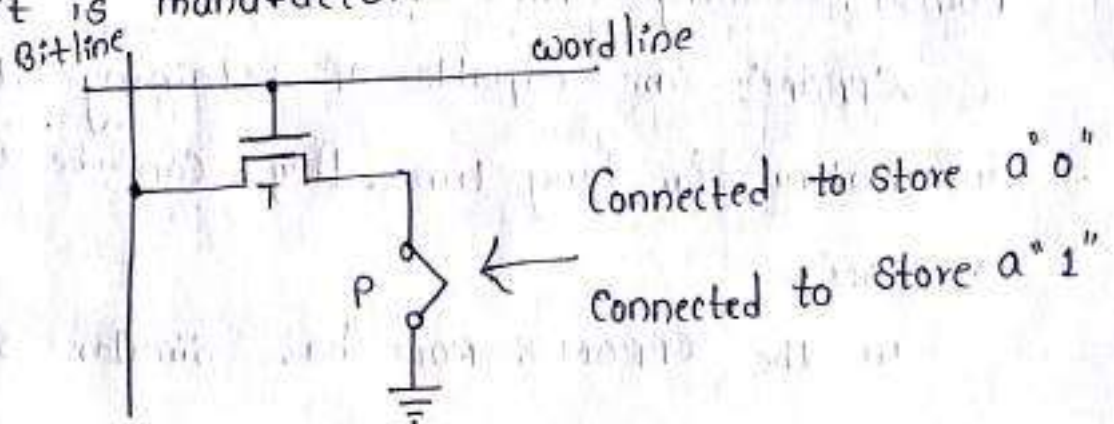


Fig: ROM Cell.

PROM:-

For small quantities, we are going to use a type of ROM called "Programmable Read Only Memory".

In this, the programmability is achieved by inserting a fuse at point p (as shown in ROM fig). Before it is programmed, the memory contains all zero's (0's).

The user can insert 1's at the required locations by burning out the fuses at the locations using high-current pulses.

* PROM's provide flexibility & Convenience.

* PROM's provide faster & less expensive because they can be programmed directly by the users. The ROM and EPROM are irreversible.

EPROM:-

This is an type of ROM, which allows the stored data to be erased and new data to be loaded. Such an erasable, Reprogrammable Rom is usually called as "EPROM".

EPROM's are Capable of retaining stored information for long time, they can be used in place of ROM's.

In the EPROM & ROM has similar structure. In

EPROM cell, the connection to ground is always made at point "P" and a special transistor is used, which has the ability to function as either as a normal transistor or as disabled transistor that is always turned off i.e, the transistor can be programmed to behave as a permanently open switch.

The important advantage is that their contents can be erased and reprogrammed.

The disadvantage is that a chip is removed physically from the circuit for reprogramming and the entire contents is erased by exposure to u.v (Ultra Violet) rays. light.

EEPROM :-

The another version of erasable PROM's that can be both programmed and erased electrically. Such chips are called EEPROM's. It is possible to erase the contents selectively.

The disadvantage is that different voltages are needed for erasing, writing & reading the stored data.

Flash Memory:-

Flash memory is intermediate b/w EPROM & EEPROM. Like EEPROM, flash memory uses an electrical erasing technology.

An entire flash memory can be erased in one or a few seconds which is much faster than EPROM.

Flash memory uses only one transistor per bit and so achieves the high density.

Single flash chips don't provide sufficient storage capacity for applications mentioned.

Larger memory modules consisting of a no. of chips are needed. There are two popular choices for implementation of such modules: They are,

(i) Flash Cards

(ii) Flash Drives

Speed Size & Cost:-

It is very clear that very fast memory can be implemented if SRAM chips are used. But these chips are expensive because their basic cells have '6' transistors, which precludes packing a very large no. of cells into one single chip.

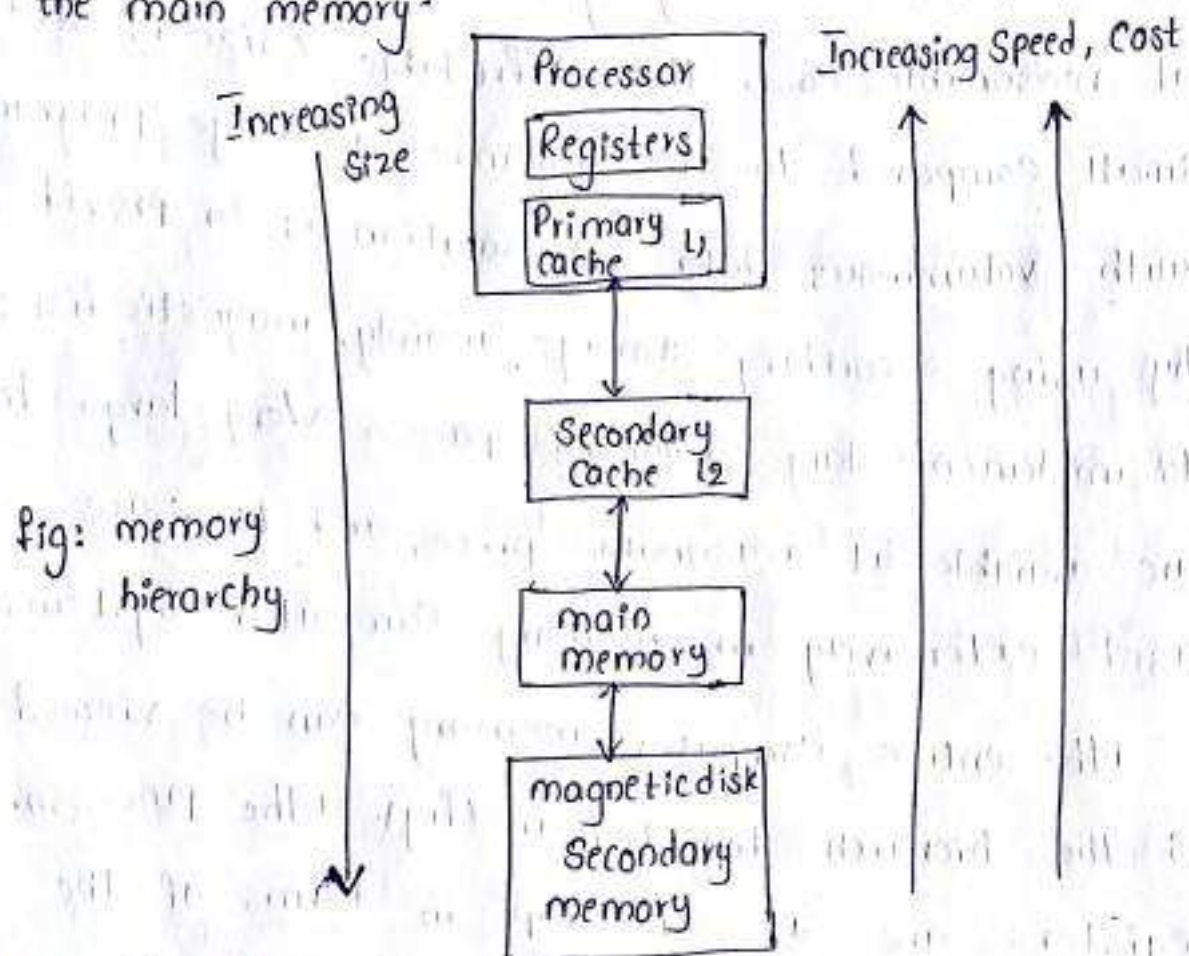
The alternative use is to use "Dynamic Ram (14) chips", which have much simpler basic cells and thus are much ^{less} expensive. But such memories are significantly slower.

Although dynamic memory units in the range of hundred's of megabytes can be implemented at reasonable cost, the affordable size is still small compared to the demands of large programs with voluminous data. A solution is to provide by using Secondary storage mainly, magnetic disks to implement large memory spaces. Very large disks are available at reasonable price, and they are used extensively in memory Computer systems.

The entire Computer memory can be viewed as the hierarchy depicted in (fig). The processor registers are at the top in terms of the speed of access. The next level of the hierarchy is relatively small amount of memory that can be implemented directly on the processor chip. This memory is called a "Processor Cache". Another next level of hierarchy is called the "main memory".

This rather large memory, is implemented using dynamic memory components, typically in the form of SIMM's, DIMM's or RIMM's.

Disk devices provide a huge amount of inexpensive storage. They are very slow compared to the semiconductor devices used to implement the main memory.



Cache Memories:-

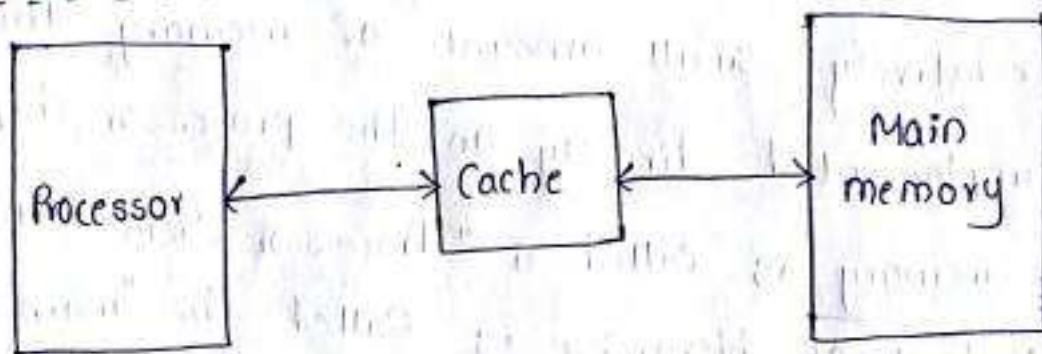


fig: Cache memory

The speed of the memory is very low, because (15)
for good performance, the processor cannot spend
much of its time waiting to access instructions
and data in main memory.

An efficient of cache mechanism is based on
a property of Computer programs called "locality of
reference". Analysis of programs shown the most of
their execution time is spent on routines in which
many instructions are executed repeatedly.

The instructions is localized areas of the
program are executed repeatedly during some time
period and the remainder of the program is
accessed relatively infrequently. This is referred
to as "locality of reference". This is divided into two
types (i) Temporal (ii) Spatial

The "temporal" means that a recently executed
instructions is likely to be executed again.

The "spatial" means that instructions in
close proximity to a recently executed instructions
(Based on instruction address).

If the active segments of a program can be
placed in a fast cache memory, then the total

execution time is reduced.

The temporal aspect of the locality of reference that whenever an information item is first needed, this item should be brought into cache where it is needed.

The spatial aspect suggests that instead of fetching one item from the main memory to the cache, it is useful to fetch several items that reside at adjacent addresses.

The term 'block' to refer to a set of contiguous address locations of some size. Another item is refer to a cache block is "cache line".

The cache memory can store a reasonable no. of blocks at any given time, but this number is small compared to the total no. of blocks in the main memory.

The Correspondence b/w the main memory blocks and those in the cache is specified by a 'mapping function'.

The Cache Control must decide which block should be removed to create space for new block that contains the referenced word. The

Collection of rules for making this decision is (16)
"replacement Algorithm".

The R/W operation is performed on the appropriate cache location. In read operation, the main memory is not involved.

The write operation may be proceed into two ways- (i) Write-through Protocol

(ii) Write-back or Copy back
(back) (back)

In write-through protocol, both cache memory and the main memory locations are updated simultaneously.

In write back or copy back, the cache memory is only updated and the updated part is marked with its associated flag bit. The bit is known as "dirty or modified bit". The main memory location is updated later, when the block containing this marked word is to be removed from the cache to make room for a new block.

When the addressed word in a Read operation is not in the Cache, a "read miss" occurs.

The later approach, which is called load-through or early restart. In this, it reduces the processor's waiting period but it is more complex circuitry.

During write operation, if the addressed word is not in cache, a "write miss" occurs. Then, the write-through is used, the information is written directly into the main memory.

In this case of write back protocol, the block containing the addressed word is first brought into the cache, and then the desired word in the cache is overwritten with the new information.

Performance Considerations:-

Two key factors of a Computer are performance & cost. Performance depends on how fast instructions can be brought into the processor for execution & how fast they can be executed.

An effective way to introduce parallelism is to use an interleaved organization.

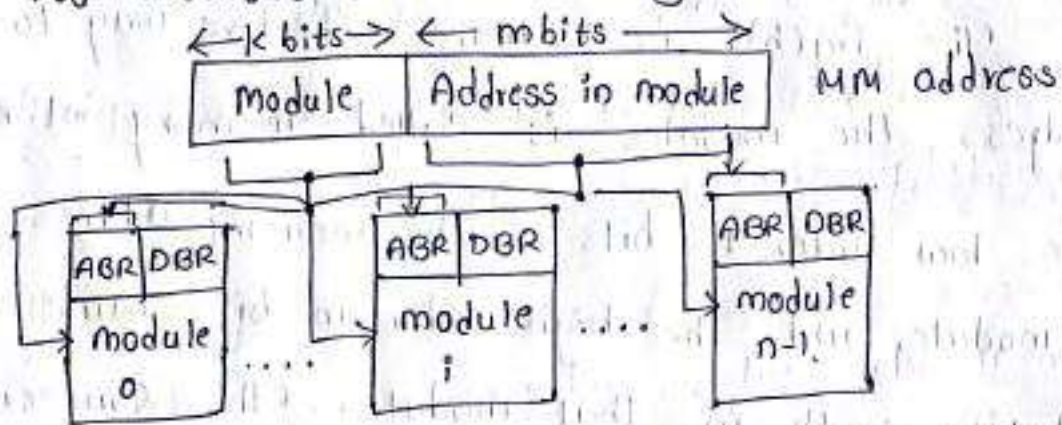
Interleaving:-

If the main memory of a Computer is

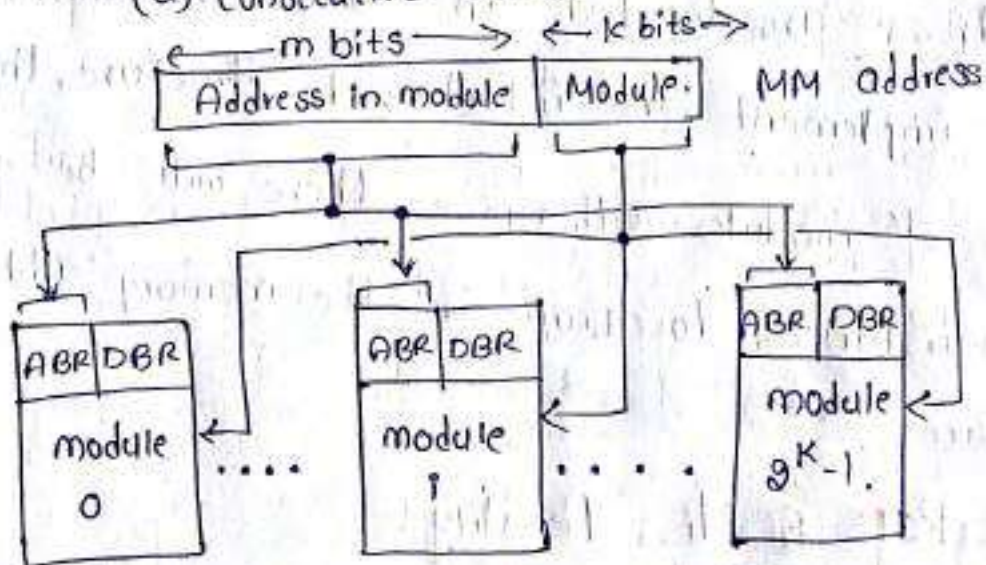
Structured as Collection of physically separate modules, each with its own address bytes are address Buffer Registers (ABR). and data Buffer Register (OBR).

The memory access operations may proceed in more than one module at the same time. Thus, the aggregate rate of transmission of words to and from the main memory system can be increased.

Two methods of address layout as shown in fig.



(a) Consecutive words in module.



(b) Consecutive words in Consecutive modules

In fig(a), the memory address generalized by the processor is decoded. The high order k bits name one of ' n ' modules and the low-order ' m ' bits name a particular word in that module.

When consecutive locations are accessed, when a block of data is transferred to a cache, only a one module is involved. At the same time, however, devices with DMA ability may be accessing information in other memory modules.

The fig(b), is a more effective way to address the modules is called "memory interlaving". The low order k bits of the memory address select a module, and the high-order m bits name or location within that module. The consecutive addresses are located in successive modules.

To implement the interleaved structure, there must be 2^k modules otherwise, there will be gaps of nonexistence location in the memory address space.

Hit Rate & Miss Penalty:-

An excellent indicator of the effectiveness of a particular implementation of the memory

hierarchy is the success rate in accessing information at various levels of the hierarchy. A successful access (18) to data in a cache is called a "hit". The no. of hits stated as a fraction of all attempted accesses is called the "hit rate", and the "miss rate" is the no. of misses stated as a fraction of attempted accesses.

The entire memory hierarchy would appear to the processor as a single memory unit that has to access time of a cache on the processor chip and the size of a magnetic disk.

Performance is adversely affected by the actions that must be taken after a miss. The extra time needed to bring the desired information into the cache is called the "miss penalty". In general, the miss penalty is the time needed to bring a block of data from a slower unit in the memory hierarchy to a faster unit.

The rough estimate of the cache can be obtained as follows:

Time without Cache

Time with Cache.

Caches on the Processor chip :-

When information is transferred between different chips, considerably delays are introduced in driver and receiver gates on the chips. Unfortunately, space on the processor chip is needed for many other functions, this limits the size of the cache that can be accommodated.

The average access time experienced by the processor in a system with two levels of cache is

$$t_{ave} = h_1 C_1 + [1-h_1] h_2 C_2 + [1-h_1] [1-h_2] M$$

where

h_1 = hit rate in the L_1 cache

h_2 = hit rate in the L_2 cache

C_1 = time to access information in the L_1 cache

C_2 = time to access information in the L_2 cache

M = time to access information in main memory.

$(1-h_1)(1-h_2)$ is should be low. If both h_1, h_2 are in the 90% range, then the no. of misses will be less.

Other Enhancements:-

Write Buffer:-

When the write through protocol is used, each write operation results in writing a new value

into the main memory. The processor must wait for the memory function to be completed. The processor (19) typically does not immediately depend on the result of a write operation, so it is not necessary for the processor to wait for write requested to be completed. To improve performance, a write buffer can be included for temporary storage of write requests. The write buffer may hold a no. of write requests. Afterward, the contents of the buffer are written into the main memory. Thus, the write buffer also works well for the "write-back protocol".

Prefetching:-

The new data are brought into the cache when they are first needed. The processor has to pause until the new data arrive, which is the effect of miss penalty.

To avoid stalling the processor, it is possible to prefetch the data into the cache before they are needed. The simplest way to do this is through software.

A special prefetch instruction may be provided in the instructions set of the processor. A prefetch instruction is inserted in a program, to cause the data to be loaded in the cache by the time they are needed in the program. Prefetch instructions can be inserted into a program either by the programmer or by the compiler. It is obviously preferable to have the compiler insert these instructions, which can be done with good success for many applications.

Prefetching can also be done through hardware. This involves adding circuitry that attempts to discover a pattern in memory references and then prefetches data according to this pattern.

Look up Free Cache :-

The software prefetching scheme does not work well if it interferes significantly with the normal execution of instructions. This is the case if the action of prefetching stops other accesses to the cache until the prefetch is completed. A cache of this type is said to be locked while it services a miss. A cache that can support multiple outstanding misses is called "look up free". Since it

Can service only one miss at a time it must
circuitry that keeps track of all outstanding
misses. Lock up free caches were first used in
the early 1980's in the cyber series of Computers
manufactured by "Control Data Company".

Virtual Memories:-

In most modern Computer Systems, the physical
main memory is not as large as the address
space spanned by. an address is issued by the
processor. The size of the main memory in a
typical computer ranges from a few hundred
megabytes to 1G bytes. When a program does not
completely fit into the main memory, the parts
of it not currently being executed are stored on
secondary storage devices, such as "magnetic disks".

Techniques that automatically move program and
data blocks into the physical main memory when
they are required for execution is called "Virtual
techniques".

The binary address that the processor issues for
either instructions or data are called "Virtual" or
Logical Addresses.

These addresses are translated into physical addresses by a Combination of h/w & s/w Components.

If a virtual address refers to a part of the program or data space that is currently in the physical memory, then the contents of the appropriate location in the main memory are accessed immediately.

If the referenced address is not in the main memory, its contents must be brought into a suitable location in the memory.

The below fig, Shows an Organization that implements virtual memory.

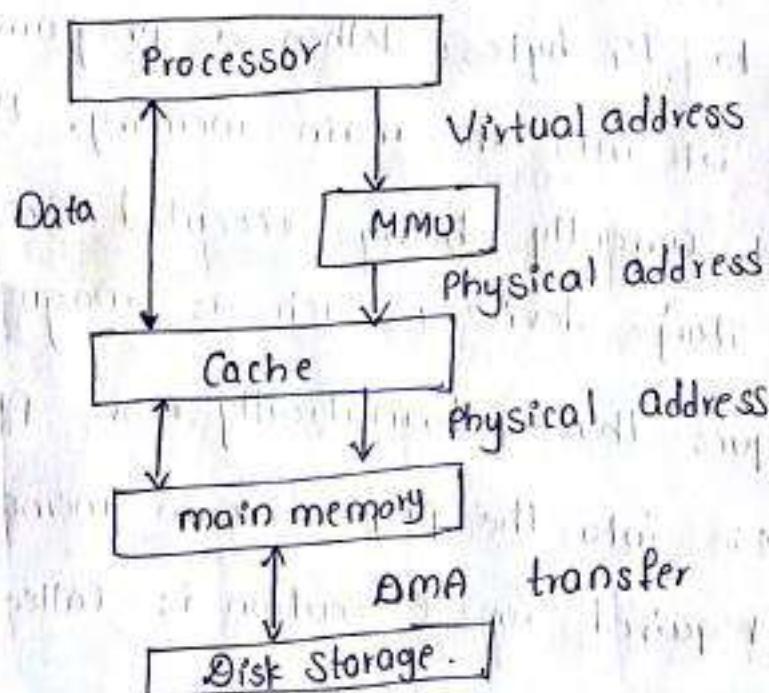


Fig: Virtual memory Organization

A special hardware unit, called the "Memory Management Unit" (MMU), translates Virtual

address to physical address. When the desired data (or instructions) are in the main memory the data is fetched as and transfers to Cache memory. (21)

If the data are not in the main memory, the MMU causes the OS to bring the data into the memory from the disk. Transfer of data b/w the disk & main memory is performed using DMA.

Address Translation:-

A simple method for translating virtual address into physical address is to assume that all programs and data are composed of fixed-length unit called "pages", each of which consisting of block of words that occupy contiguous locations in the main memory. Pages commonly range from 2k to 16k bytes in length. Pages should not be too small, because the access time of magnetic disk is much longer than the access time of the main memory. If pages is too large it is possible that a substantial portion may not be used, yet this unnecessary data will occupy valuable space in the memory.

A virtual memory address translation method

based on the concept of fixed length pages (is shown in fig). Each virtual address generated by the processor, whether it is for an instruction fetch or an operand fetch/store operation is interpreted as "Virtual Page Number" (higher order bits) followed by an offset (low-order bits) that specifies the location of a particular byte (or word) within a page.

This information includes the main memory address where the page is stored and the current status of the page. An area in the main memory that can hold one page is called "Page frame". The starting address of the page table is kept in a "page table base register".

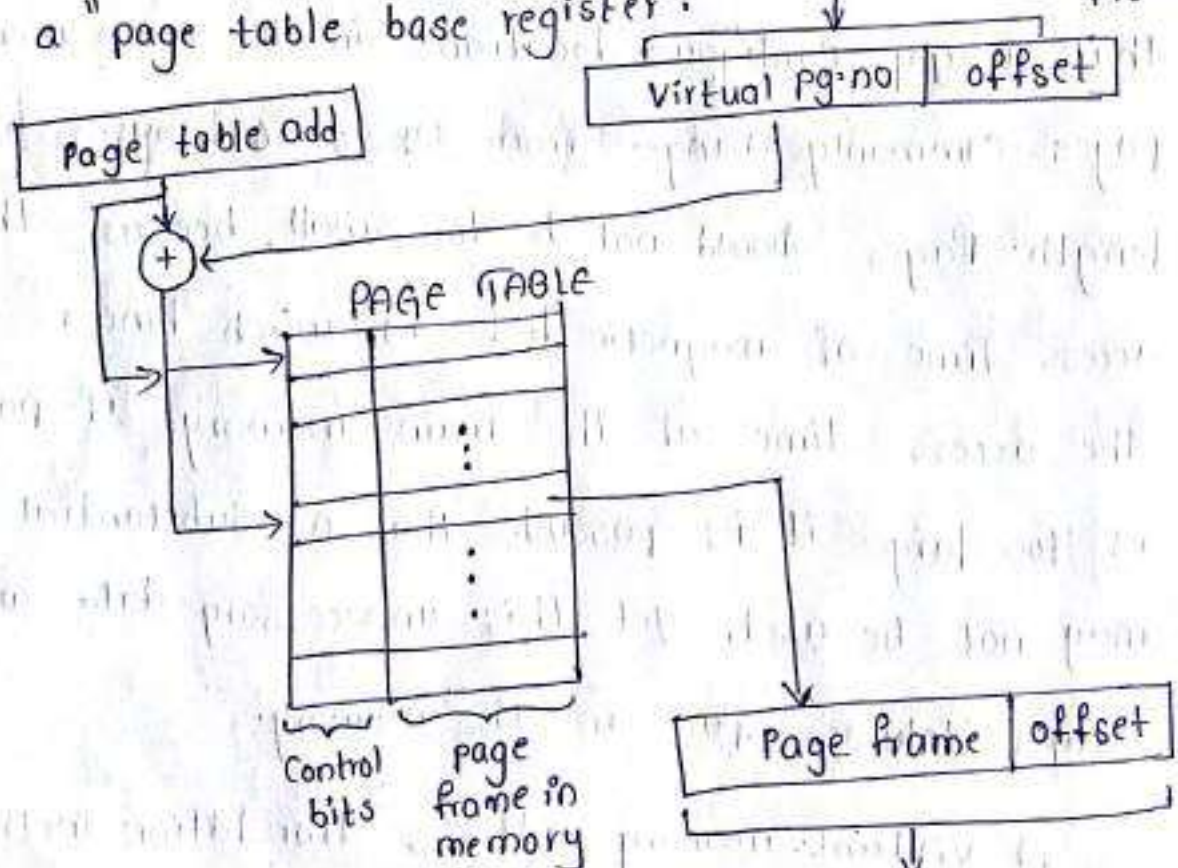
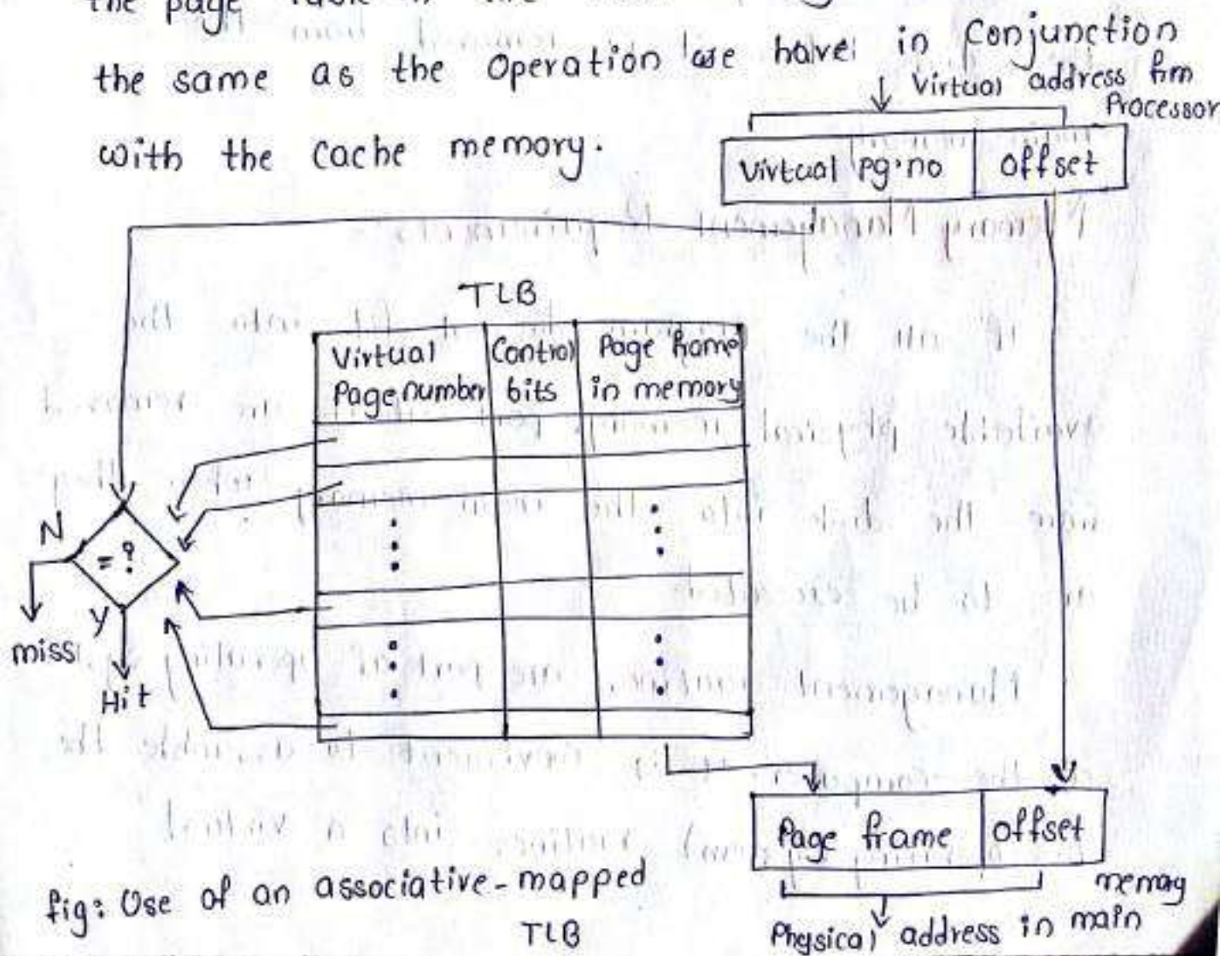


fig: Virtual-memory address

The page table information is used by MMU for every read & write access, so ideally, the page table should be situated within the MMU. Unfortunately the page table may be rather large, and since the MMU is normally implemented as part of the processor chip, it is impossible to include a complete page table on this chip. A small cache usually called the "Translation Lookaside Buffer" (TLB) is incorporated into the MMU for this purpose. The operation of the TLB with respect to the page table in the main memory is essentially the same as the operation we have in conjunction with the cache memory.



When a program generates an access request to a page that is not in the main memory, a "page fault" have Occured. The whole page must be brought from the disk into the memory before access can proceed. When it detects the page fault the MMU asks the operating System to intervene by raising an exception (interrupt).

It is essential to ensure that the interrupted task can continue correctly when it resumes execution.

A modified page has to be written back to the disk before it is removed from the main memory.

Memory Management Requirements:-

If all the program doesnot fit into the available physical memory, parts of it are removed from the disk into the main memory when they are to be executed.

Management routines are part of operating System of the Computer. It is convinient to assemble the o.s (operating System) routines into a virtual

address space, called the system space, that is separate from the virtual space in which user application programs reside. The latter space is called the "user space". The MMU uses a page table base register to determine the address of the table used in the translation process.

In any computer system in which independent user programs coexist in the main memory, the notion of "protection" must be addressed. No program should be allowed to destroy either the data or instructions of other programs in the memory. The processor has two states of protection, the "supervisor state" and the "user state". As the name suggests, the processor is usually placed in the supervisor state when operating system routines are being executed and in user state to execute user programs. These privileged instructions, which include such operations as modifying the page table base register, can only be executed while the processor is in the supervisor state.

It is sometimes desirable for one application program to have access to certain pages belonging to another program. The operating system can arrange this by causing these pages to appear in both spaces.

Secondary Storage:-

Large storage requirements of most computer systems are economically realized in the form of magnetic disks, optical and magnetic tapes which are usually referred to as secondary storage devices.

Magnetic Hard Disk:-

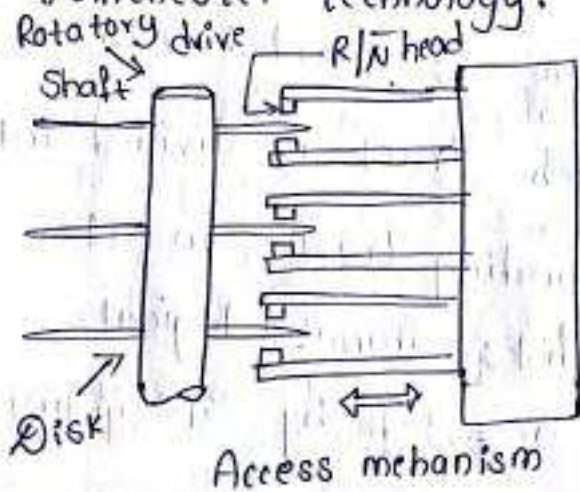
As the name implies, the storage medium in a magnetic disk system consists of one or more disk mounted on a common spindle. A thin magnetic film is deposited on each disk, usually on both sides.

Digital information can be stored on the magnetic film by applying current pulses of suitable polarity to the magnetized coil. Using the clock signal as reference, the data stored on other tracks can be read correctly.

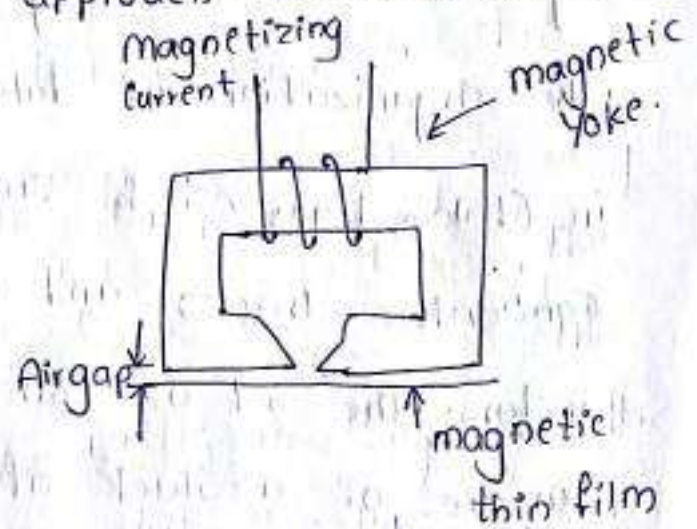
The modern approach is to combine the clocking (24) information with the data. Several different techniques have been developed for such encoding. One simple scheme depicted in (fig.c) is known as "phase encoding or Manchester encoding". The drawback of Manchester encoding is its poor bit storage density.

In most modern disk units, the disks and the read/write heads are placed in a sealed, air-filtered enclosure. This approach is known as

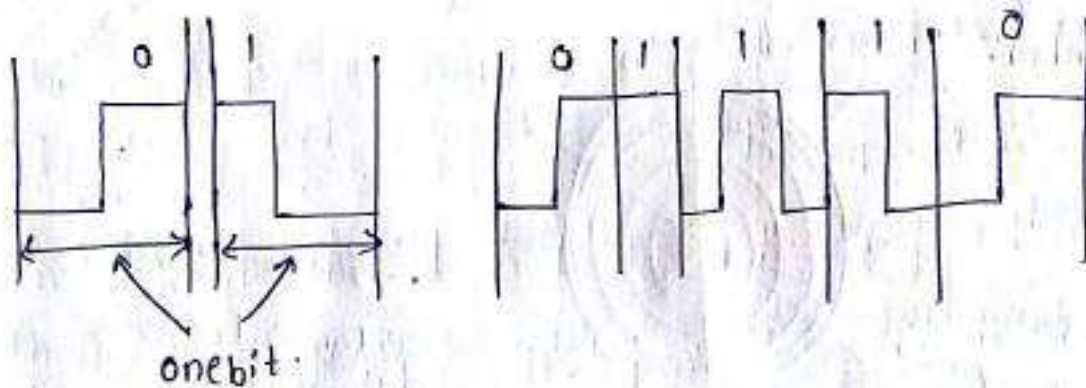
"Winchester technology."



(a) Mechanical Structure



(b) R/W head detail



(c) Bit representation by phase encoding

The disk system consists of three key parts. One part is the assembly of disk platters, which is usually referred as the disk. The second part comprises the electromechanical mechanism that spins the disk & moves the read/write heads; it is called "disk drive". The 3rd part is a electronic circuitry that controls the operation of system, which is called "disk controller".

Organization & Accessing Data on a Disk:-

The Organization of data on a disk is illustrated in (below fig). Each surface is divided into concentric tracks, and each track is divided into sectors. The set of corresponding tracks on all surfaces of a stack of disks form a logical cylinder. The data are accessed by specifying the surface number, track number and the sector number.

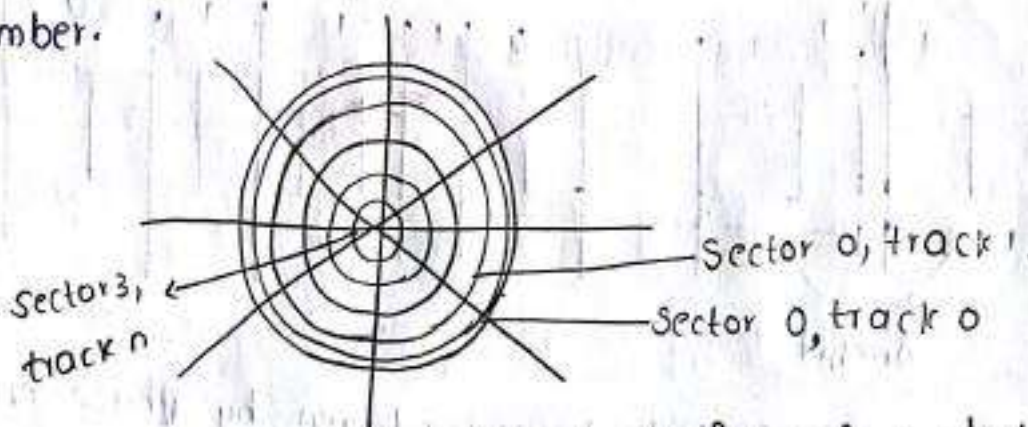


fig: Organization of one surface of a disk.

• Data bits are stored serially on each track. Each ⁽²⁵⁾ Sector usually contains 512 bytes of data, but other sizes may be used. The data are preceded by a sector header that contains identification information used to find the desired sector and on the selected track. There are additional bits that constitute an "Error Correcting Code" (ECC). The ECC bits are used to detect & correct errors that may have occurred in writing or reading of the 512 data bytes. To easily distinguish b/w two consecutive sectors, there is small intersector gap. In a typical computer, the disk is subsequently divided into logical partitions.

Access Time :-

There are 2 components involved in the time delay b/w receiving an address and the beginning of the actual data transfer. The first, called the "seek time", is the time required to move the R/W head to the proper track.

The second component called "rotational delay", also called "latency time". This is the amount of time that elapses after the head is positioned over the

Correct track until the starting position of the addressed sector passes under the R/W head.

The sum of these two delays is called "disk access time".

Disk Controller :-

Operation of a disk drive is controlled by a disk Controller circuit. Which also provides an interface b/w the disk drive and the bus that connects it to the rest of Computer system. The disk Controller may be used to control more than one drive.

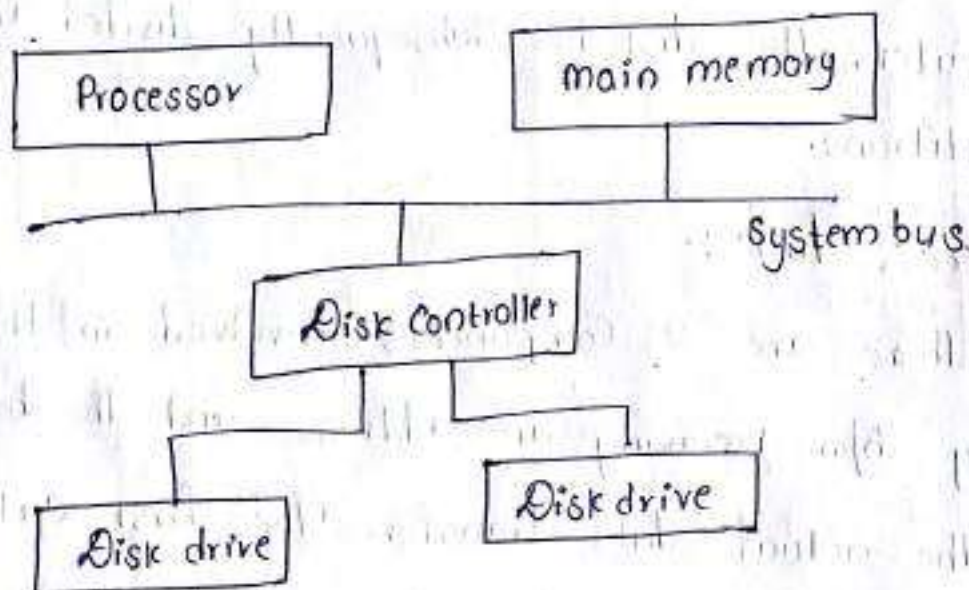


Fig: Disk Connected to the System bus.

The disk Controller uses the DMA scheme to transfer b/w the disk and the main memory.

Actually, these transfers are from/to the data buffer, which is implemented as a part of disk

Controller module.

(26)

Main memory Address:- The address of the first main memory location of the block of words involved in the transfer.

Disk Address:- The location of the sector containing the beginning of the desired block of words.

Word Count:- The no. of words in the block to be transferred.

on the disk drive side, the Controller's major functions are:-

Seek:- Causes the disk drive to move the read/write head from its current position to the desired track.

Read:- Initiates a Read Operation starting at the address specified in the disk address register.

Data read serially from the disk are assembled into words and placed into the data buffer for transfer to the main memory.

Write:- Transfers data to the disk, using a control method similar to that for Read Operations.

Error Checking:- Computes the Ecc value for the data read from a given sector & compares it

with the corresponding Ecc value read from the disk.

Software & Operating System Implications:-

All data transfers activities involving disks are initiated by the Operating System. The disk is a nonvolatile storage medium, so, the OS itself is stored on a disk.

When power is turned off, the contents of the main memory are lost. When the power is turned on again, the OS has to be loaded into the main memory. Which takes place a part of process known as "booting". ROM stores a small monitor program that can read and write main memory locations as well as read one block of data stored on the disk at address 0. This block referred to as "boot block", contains a loaded program.

After the boot block is loaded into the memory by the ROM monitor program, it loads the main parts of the OS into the main memory.

In a Computer system that has multiple disks, the OS may require transfer from several disks.

Floppy Disk:-

The devices previously discussed are known as hard or rigid disk units. "Floppy disks" are smaller, simpler and cheaper disk units that consist of flexible, removable, plastic diskette coated with magnetic material. The diskette is enclosed in a plastic jacket which has opening where the R/W head makes contact with the diskette.

One of the simplest schemes used in the first floppy disks for recording data is phase or Manchester encoding. Disk encoded in this way are said to have "single density". A more complicated variant of this scheme, called "double density".

The main feature of floppy disks is their low cost & shipping convenience. However, they have much smaller storage capacities, longer access times, and higher failure rates than hard disks.

Current standard floppy disks are 3.25 inches in diameter and store 144 or 2M bytes of data.

Raid Disk Arrays:-

Semiconductor memory speeds have

improved more modestly. The smallest relative improvement in terms of speed has been in disk storage devices, for which access times are still on the order of milliseconds.

High performance devices tend to be expensive. Sometimes it is possible to achieve very high performance at a reasonable cost by using a no. of low cost devices operating in parallel.

In 1988, researchers at the university of California - Berkeley proposed a storage system based on multiple disks. They called it "RAID", for "Redundant Array of Inexpensive Disk". Using multiple disks also makes it possible to improve the reliability of overall system. 6 different configurations were proposed. They are known as RAID levels even though there is no hierarchy involved.

RAID 0 is the basic configuration intended to enhance performance. A single large file is stored in several separate disk units by breaking the file up into a no. of smaller pieces. and storing these pieces on different disks. This is called "data striping". When the file is accessed for a read, all disks can deliver their data in parallel.

RAID 1 is intended to provide better reliability (28) by storing identical copies of data on two disks rather than just one. The two disks are said to be mirrors of each other. Then, if one disk drive fails, all read & write operations are directed to its mirror drive.

RAID 2, RAID 3, RAID 4 levels achieve increased reliability through various parity checking schemes without requiring a full duplication of disks. All of parity information is kept in one disk.

RAID 5 also makes use of a parity-based error-recovery scheme. However, the parity information is distributed among all disks, rather than being stored on one disk.

Commodity Disk Considerations:-

Most disk units are designed to be connected to standard busses. The performance of a disk unit depends on its internal structure and the interface used to connect it to the rest of the system. The cost depends largely on the storage capacity, but it is also affected greatly by the sales volume of a particular product.

The different types of disks are i) ATA/EIDE Disks
ii) SCSI Disks
iii) RAID Disks.

Optical Disks :-

Large storage devices can also be implemented using optical means. The familiar Compact Disk (CD), used for audio system, was the first practical application of this technology. Soon after, the optical technology was adapted to the computer environment to provide high-capacity read-only storage referred to as "CD-ROM".

The first generation of CD's was developed in mid 1980's by the Sony & Philips Companies, which also published a complete specification for these devices.

CD Technology :-

The optical technology that is used for CD system is based on laser light source. A laser beam is directed onto the surface of the spinning disk. Physical indentations in the surface are arranged along the tracks of the disk.

They reflect the focused beam toward a photodetector, which detects the stored binary pattern.

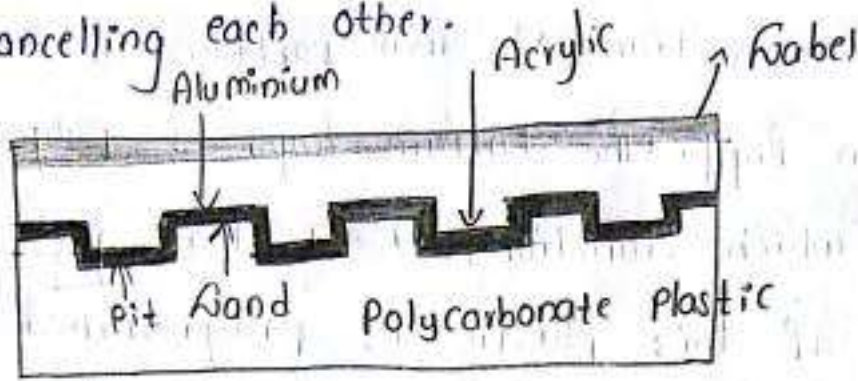
A cross-section of small portion of CD is [shown in fig]. The bottom layer is polycarbonate plastic, which functions as a clear glass base. The surface of this plastic is programmed to store data by indenting it with "pits". The unindented parts are called "lands". A thin layer of reflecting aluminium material is placed on top of a programmed disk.

The laser source and the photodetector are positioned below the polycarbonate plastic. The emitted beam travels through this plastic, reflects off the aluminium layer, and travels back toward the photodetector.

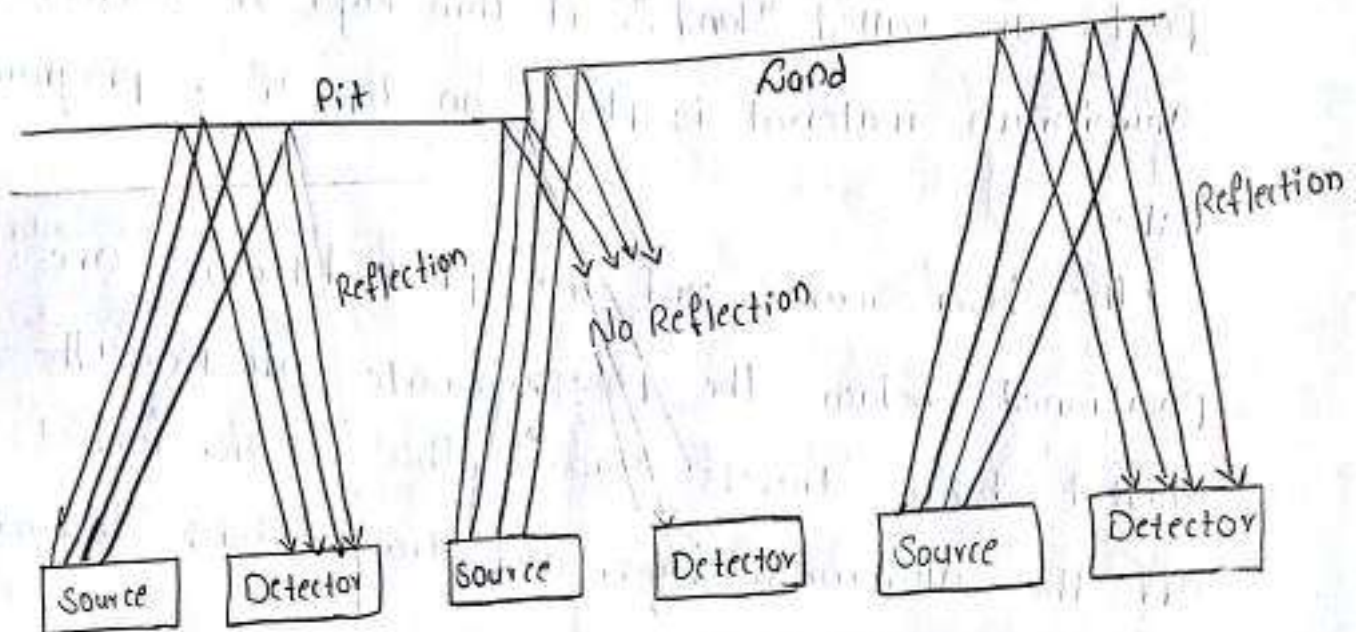
When the light reflects from the solely from the pit, or solely from the land, the detector will see the reflected beams as a bright spot.

But, a different situation arises when the beam moves through the edge where the pit changes to the land, and vice versa. The pit width is recessed on quarter of the wavelength of the light. Thus

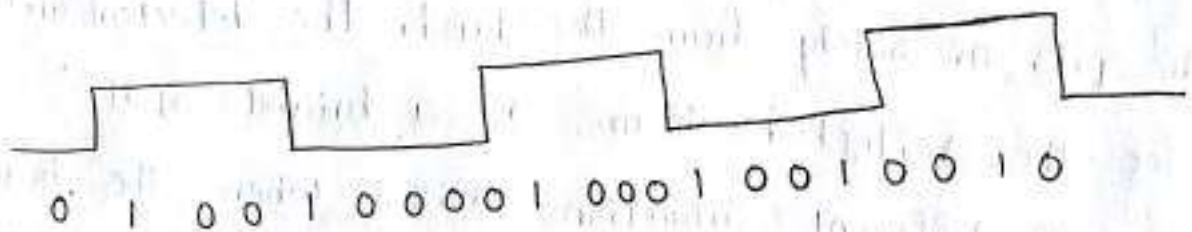
the reflected wave from the pit will be 180° out of phase with the wave reflected from the land, Cancelling each other.



(a) Cross Section



(b) Transition from pit to land.



(c) Stored binary pattern

fig: Optical Disk.

At the pit-land & land-pit transitions the detector will not see a reflected beam & will detect a dark spot. (30)
The CD is 120 mm in diameter. There is a 15-mm hole in centre. Data are stored on tracks that cover pits area from 25-mm radius to 58-mm radius. The space b/w the tracks is 1.6 microns. pits are 0.5 microns wide and 0.8 to 3 microns long. There are more than 15,000 tracks on a disk. If the entire track spiral were unraveled, it would be over 5 km long!

CD: Rom:-

Since information is stored in binary form in CD's they are suitable for use as a storage medium in computer system. The biggest problem is to ensure the integrity of stored data. Because the pits are very small, it is difficult to implement all of the pits perfectly. It is necessary to use additional bits to provide error checking and correcting capability. CD's are used in computer applications have such capability. They are called CD-Rom's, because after manufacture their contents can only be read, as with semiconductor ROM chips.

Stored data are organized on CD ROM tracks in

the form of blocks that are called sectors. There are several different formats for a sector. One format is known as, Mode 1, uses 2352-byte sectors. There is a 16-byte header that contains a synchronization field used to detect the beginning of the sector and addressing information used to identify the sector.

Error detection and correction is done at more than one level. As in CD's each byte of stored information is encoded using a 14-bit code that has some error correcting capability. This code can correct single bit errors.

CD-ROM drives operate at a no. of different rotational speeds. The basic speed, known as 1x, is 75 sectors per second. The importance of CD-ROM's for computer systems stems from their large storage capacity and fast access times compared to other inexpensive portable media, such as floppy disks & magnetic tapes. They are widely used for the distribution of software, databases, application programs and video games.

CD-Recordables:-

A new type of CD was developed in the

late 1990's on which data can be easily recorded by a Computer user. It is known as CD-Recordable [CD-R]. A Spiral track is implemented on a (31) disk during manufacturing process. A laser in a CD-R drive is used to burn pits into a organic dye on the track. When a burned spot is heated beyond a critical temperature, it becomes opaque. The written data stored permanently. Unused portions of disk can be used to store additional data on later time.

CD - Rewritables :-

The most flexible CD's are those that can be written multiple times by user. They are known as CD-RW's [CD-Rewritables].

The basic structure of CD-RW's is similar to the structure of CD-Rs. Instead of using organic dye in the recording layer, an alloy of silver, indium, antimony and tellurium is used. This alloy has interesting and useful behaviour when it is heated and cooled.

The CD-RW drive uses three different laser powers. The highest power is used to record the pits. The middle power is used to put the alloy

into the crystalline state; it is referred to as "erase power". The lowest power is used to read the stored information. There is a limit on how many times a CD-RW disk can be rewritten. Presently this can be done upto 1000 times.

CD-RW's provide a low-cost storage medium. They are suitable for archival storage of information that may range from databases to photographic images. The CD-RW drives are used for low-volume distribution of information just like CD-R's.

DVD Technology :-

The success of CD technology and the continuing quest for greater storage capability has led to the development of "Digital Versatile Disk" [DVD]. The first DVD standard was defined in 1996 by Consortium companies. The objective is to be able to store a full length movie on one side of DVD disk.

The physical size of DVD disk is the same as for CD's. The disk is 1.2 mm thick, and it is 120 mm in diameter. Its storage capacity is made larger than that of CD's by several design changes:

⇒ A red light laser with a wavelength of 635 nm is (32) used instead of the infrared light laser used in CD's which has a wavelength of 780 nm. The shorter wavelength makes it possible to focus the light to a smaller spot.

⇒ Pits are smaller, having a minimum length of 0.4 micron.

⇒ Tracks are placed close together, the distance between tracks is 0.74 micron.

Using these improvements leads to DVD capacity of 4.7 Gbytes.

Access time for DVD drives are similar to CD drives. However, when DVD disk rotates at the same speed, the data transfer rates are much higher because of the higher density of pits.

DVD-RAM:-

A rewritable version of DVD devices known as DVD-RAM, has also been developed. It provides a large storage capacity. Its only disadvantages are higher price and relatively slow writing speed. To ensure that the data have been recorded correctly on the disk, a process known as write verification is performed.

Magnetic Tape Systems:-

Magnetic tapes are suited for off-line storage of large amount of data. They are typically used for hard disk backup purpose and for archival storage.

Magnetic tape recording uses the same principles are used in magnetic-disk recording. The main difference is that the magnetic film is deposited on a very thin 0.5 or 0.25 inch wide plastic tape.

A separate R/W head is provided for each bit position on the tape, so that all bits of a character can be done read or written in parallel. One of the character bits is used as parity bit.

Data on the tape are organized in the form of records separated by gaps [as shown in fig]. The record gaps are long enough to allow the tape to attain its normal speed before the beginning of the next record is reached. To help users to organize large amount of data, a group of related books records is called "file". The beginning of file is identified by "file mark".

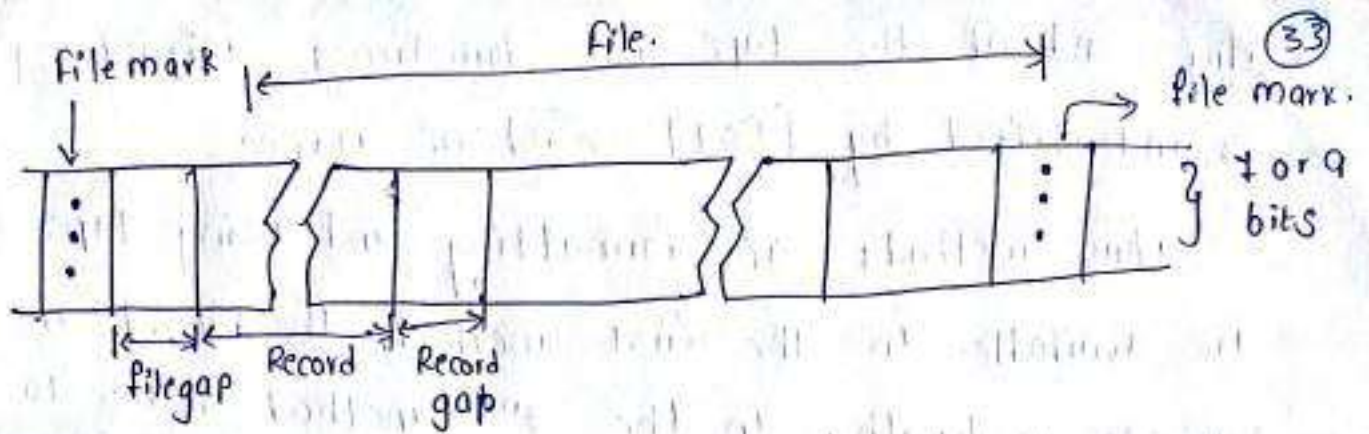


Fig: Organization of data on magnetic tape.

The first record following a file mark can be used as "header" or "identifier" for this file.

The Controller of magnetic tape drive enables the execution of a no. of Control Commands in addition to R/W Commands. Control Commands include the following operations:

- ⇒ Rewind tape
- ⇒ Rewind and upload tape
- ⇒ Erase tape
- ⇒ Write tape mark
- ⇒ Forward space one record
- ⇒ Backspace one record
- ⇒ Forward space one file
- ⇒ Backspace one file.

The tape mark referred to in operation 'write tape mark' is similar to the file mark except that it is used for identifying the beginning of the tape.

The end of the tape is sometimes defined by or identified by [EOT] End of Tape.

Two methods of formatting and using tapes are available. In the first method, the records are variable in length. In the 2nd method is use to fixed-length records, In this case it is possible to update records in place.

Cartridge Tape System:-

Tape systems have been developed for backup of on-line disk storage. One such system use an 8-mm video format tape housed in cassette. These units are called "Cartridge tapes". They have capacities in range of 2 to 5 gigabytes and handle data transfers at the rate of few hundred kilobytes per second. Multiple Cartridge systems are available that automate the loading and unloading of cassettes so that 10's of gigabytes of on-line storage can be backed up unattended.

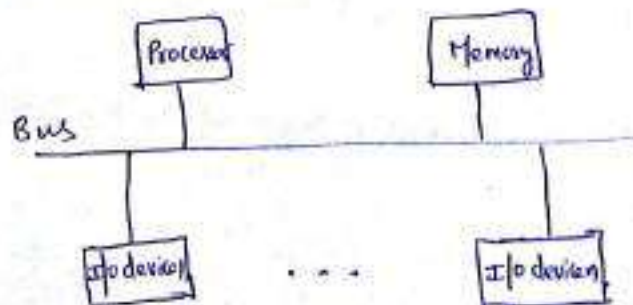
I/O Organization

Syllabus :

- Accessing I/O devices
- Interrupts
 - Interrupt Hardware
 - Enabling and Disabling Interrupts
 - Handling Multiple devices. ✓
- Direct Memory Access ✓
- Buses
 - Synchronous Bus
 - Asynchronous Bus ✓
- Interface Circuits ✓
- Standard I/O Interface ✓
 - Peripheral Component Interconnect (PCI) Bus
 - Universal Serial Bus (USB)

① Accessing I/O devices :

- The basic feature of a computer is its ability to exchange data with other devices
- The bus enables all the devices connected to it to exchange information.
- A single bus structure is depicted as



- ~~But~~ The bus consists of 3 sets of lines
 - Address lines
 - Data lines
 - Control lines

- Each I/O device is assigned a unique set of addresses
- When I/O device and the memory share the same address space, the arrangement is called Memory-mapped I/O.

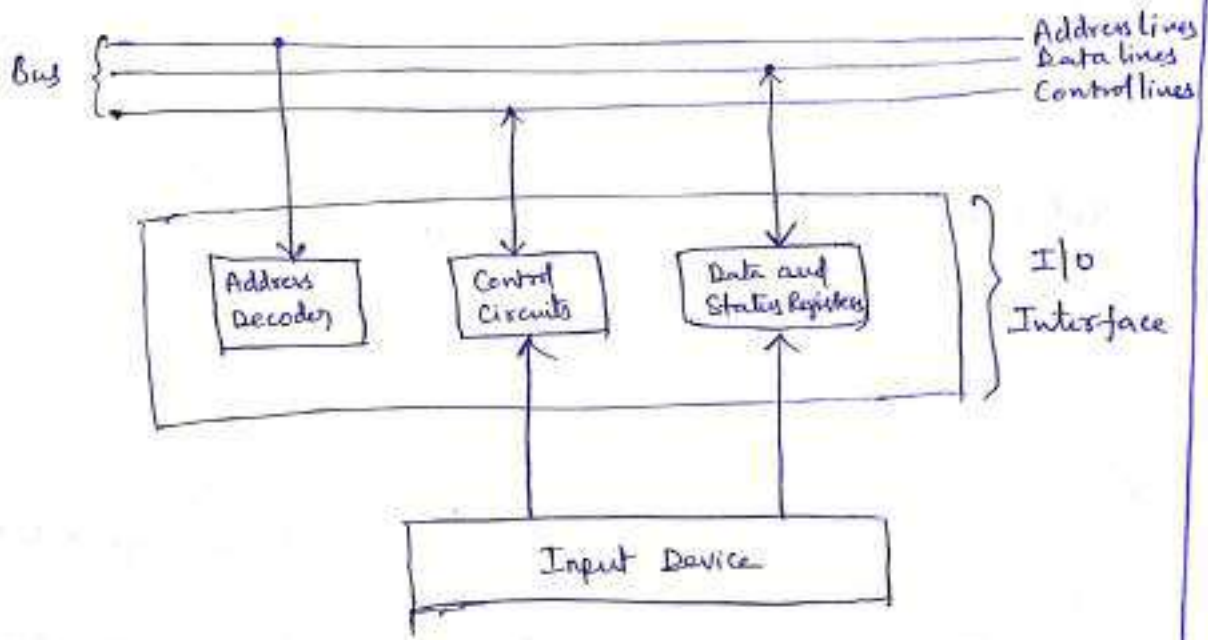
Memory-mapped I/O:

- With Memory-mapped I/O, any machine instruction that can access memory can be used to transfer data to (or) from an I/O device.

Example:

MOVE DATIN, R0 (Input Device to Processor Register)
MOVE R0, DATAOUT (Processor Register to Output Device)

- The I/O Interface for an Input device is depicted as



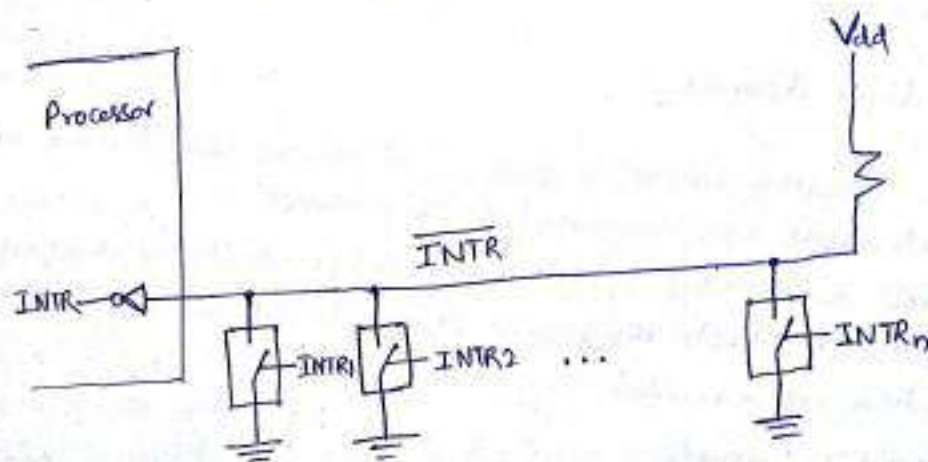
- Here, the address decoder enables the device to recognize its address when this address appears on the address lines.
- The data register holds the data being transferred to (or) from the processor
- The status register contains information relevant to the operation of the I/O device.

② Interrupts :

- An Interrupt ^{will} stop the continuous progress of an activity (or) process.
- An Interrupt is a signal to the processor emitted by hardware (or) software indicating an event that needs immediate attention.
- ~~An interrupt~~ It alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing.
- The bus control line, called interrupt-request line is used for interrupts.

i) Interrupt Hardware :

- An I/O device requests an interrupt by activating a bus line called interrupt-request.
- Most computers are likely to have several I/O devices that can request an interrupt.
- A single interrupt request line may be used to serve n -devices.
- An equivalent circuit for an open-drain bus used to implement a common interrupt-request line is depicted as



- All devices are connected to the line via switches to ground.
- To request an interrupt, a device closes its associated switch.
- Thus, if all interrupt-request signals INTR₁ to INTR_n are inactive, i.e., if all switches are open, the voltage on the interrupt-request line will be equal to V_{dd}.
- This is the inactive state of the line.
- Since the closing of one (or) more switches will cause the line voltage to drop to 0, the value of INTR is the logical OR of the requests.

from individual devices, i.e.,

$$INTR = INTR_1 + INTR_2 + \dots + INTR_n$$

→ The $\overline{INTR}$ signal is active when in the low voltage state.

(ii) Enabling and Disabling Interrupts:

→ The sequence of events involved in handling interrupt request from a single device are,

- The device raises an interrupt request.
- The processor interrupts the program currently being executed.
- Interrupts are disabled by changing the control bits in the PS (Processor Status register)
- The device is informed that its request has been recognized, and in response, it deactivates the interrupt request signal.
- The action requested by the interrupt is performed by the interrupt-service routine.
- Interrupts are enabled and execution of the interrupt program is resumed.

(iii) Handling Multiple devices:

→ Consider the situation where a number of devices ~~capable~~ capable of initiating interrupts are connected to the processor.

→ Because these devices are operationally independent, there is no definite order in which they will generate interrupts.

→ This situation is handled by 3 methods.

- (a) Vectored Interrupts
- (b) Interrupt Nesting
- (c) Simultaneous Requests

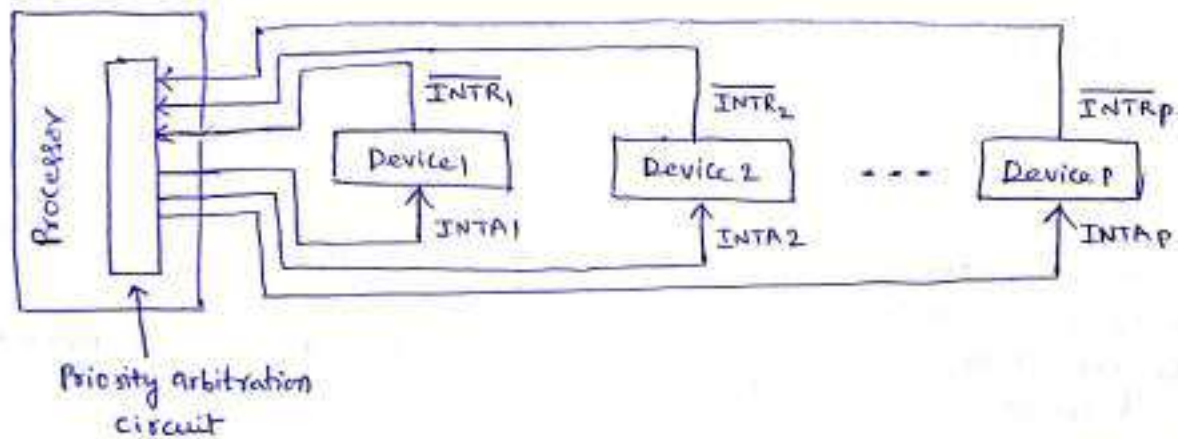
a) Vectored Interrupts:

- A device requesting an interrupt may identify directly to the processor.
- Then the processor can immediately start executing the corresponding interrupt-service routine.
- This interrupt handling scheme is known as Vectored Interrupts.

- A commonly used scheme is to allocate permanently an area in the memory to hold the addresses of interrupt-service routines.
- These addresses are referred as Interrupt vectors, and they are said to constitute the interrupt-vector table.
- For example, 128 bytes may be allocated to hold a table of 32 interrupt vectors.

b) Interrupt Nesting:

- Implementation of Interrupt priority using individual interrupt-request and Acknowledge lines is depicted as,

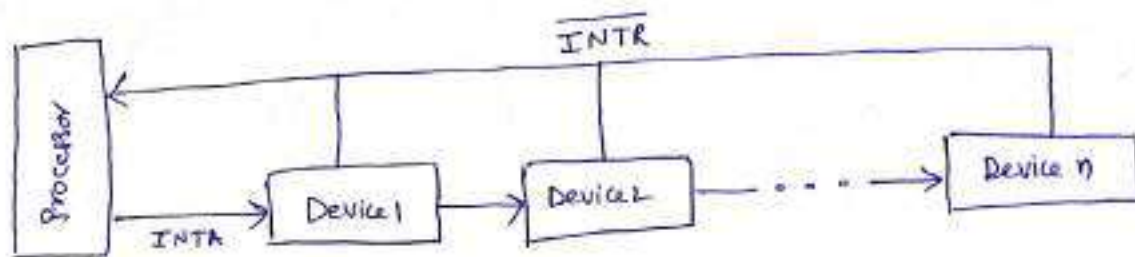


- Here, Each of the interrupt-request lines is assigned a different priority level.
- Interrupt requests received over these lines are sent to a priority arbitration circuit in the processor.
- A request is accepted only if it has a higher priority level than that currently assigned to the processor.

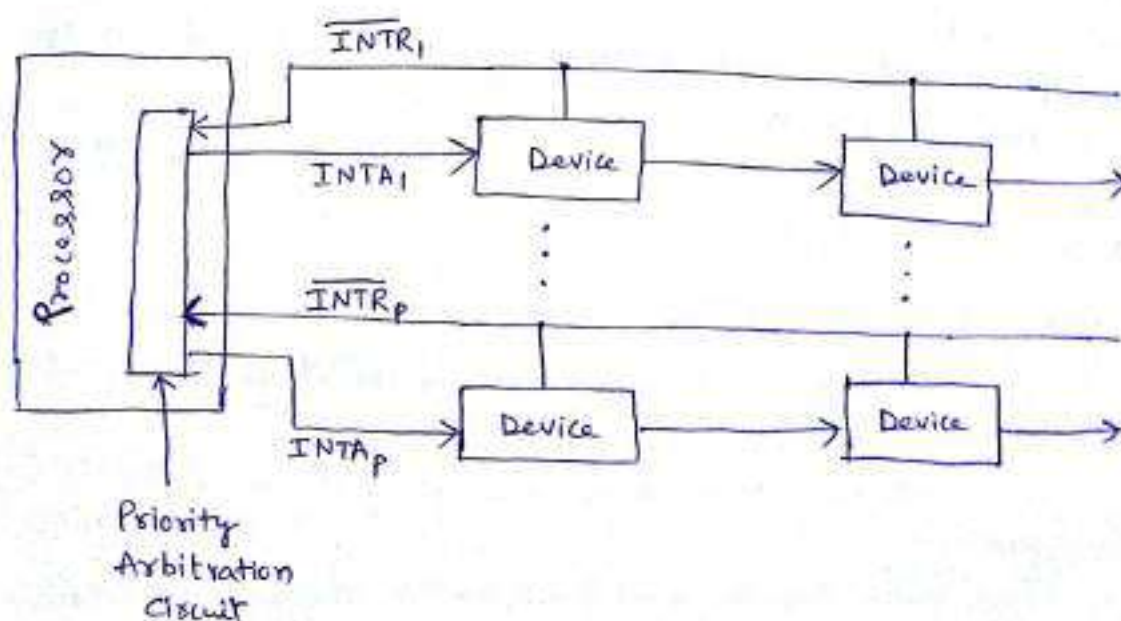
c) Simultaneous Requests:

- Consider the problem of simultaneous arrivals of interrupts requests from two (or) more devices.
- The processor must have some means of deciding which request to service first.
- The processor simply accepts the request having the highest priority.
- Priority is determined by the order in which the devices are polled, i.e., the polling the status registers of the I/O devices.

→ A Daisy chain priority interrupt Scheme is depicted as,



- The Interrupt-request line $\overline{INTR}$ is common to all devices.
- The Interrupt-Acknowledge line $INTA$, is connected in a daisy-chain fashion.
- The $INTA$ Signal propagates serially through the devices.
- When several devices raise an interrupt request and the $\overline{INTR}$ line is activated, the processor responds by setting the $INTA$ line to 1.
- This signal is received by device 1.
- Device 1 passes the signal on to device 2 only if it does not require any service.
- In daisy-chain arrangement, the device that is electronically closest to the processor has the highest priority.
- The second device along the chain has second highest priority, and so on.
- The arrangement of priority groups is depicted as,



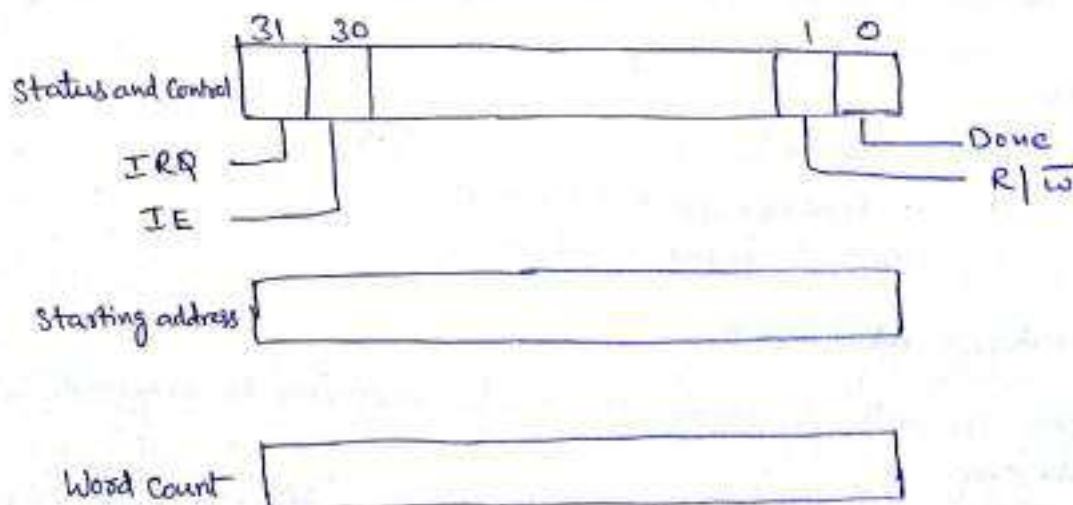
- Here, Devices are organized in groups, and each group is connected at a ~~at~~ different priority level.
- Within a group, devices are connected in a daisy-chain
- This organization is used in many Computer Systems.

③ DMA: (Direct Memory Access)

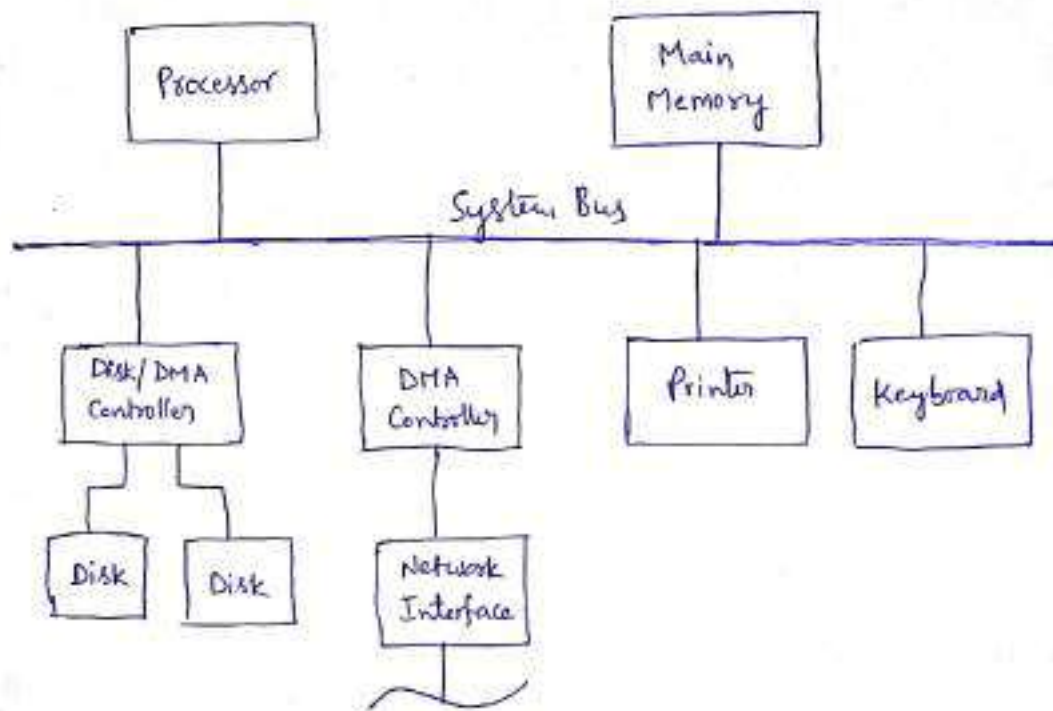
- To transfer large blocks of data at high speed, a special control unit may be provided to allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor.
- This approach is called Direct Memory Access (DMA)
- DMA transfers are performed by a control unit that is part of the I/O device interface, called DMA Controller.

(*) DMA Controller:

- The DMA Controller performs the functions that would normally be carried out by the processor when accessing the main memory.
- For each word transferred, it provides the memory address and all the bus signals that ~~are~~ control data transfer.
- Although the DMA controller can transfer data without intervention by the processor, its operation must be under the control of a program executed by the processor.
- The Registers used in DMA Controller are depicted as,



→ The use of DMA Controllers in a Computer System is depicted as,

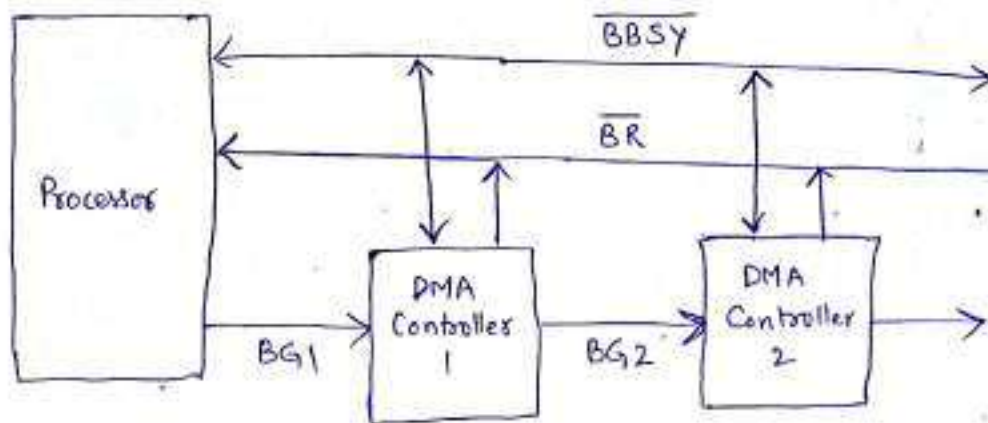


(*) Bus Arbitration :

- The device that is allowed to initiate data transfers on the bus at any given time is called the Bus Master.
- Bus arbitration is the process by which the next device to become the bus master is selected and bus mastership is transferred to it.
- The selection of the bus master must take into account the needs of various devices by establishing a priority system for gaining access to the bus.
- There are two approaches to bus arbitration
 - (i) Centralized Arbitration
 - (ii) Distributed Arbitration

(i) Centralized Arbitration :

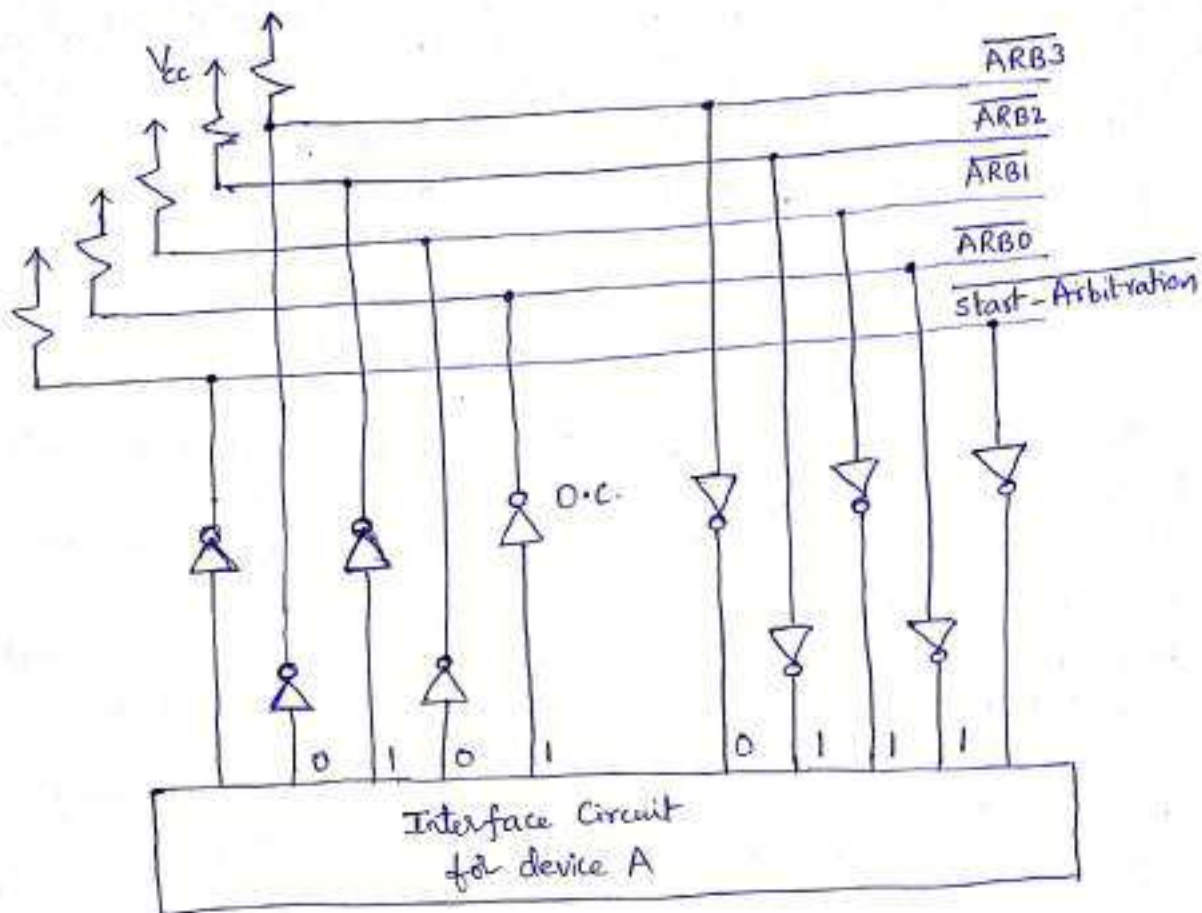
- In this approach, a single bus arbiter performs the required arbitration.
- A simple arrangement $\blacklozenge$ for bus arbitration using a daisy chain is depicted as



- In this case, the processor is normally the bus master unless it grants bus mastership to one of the DMA controllers.
- A DMA Controller indicates that it needs to become the bus master by activating the Bus-Request line, $\overline{BR}$.
- When Bus-Request is activated, the processor activates the Bus-Grant Signal BG_1 , and this signal is connected to all DMA controllers using a daisy-chain arrangement.
- The current bus master indicates to all devices that it is using the bus by activating another open-collector line called Bus Busy, $\overline{BBSY}$.
- Hence, after receiving the Bus-Grant signal, a DMA controller waits for Bus-Busy to become inactive, then assumes mastership of the bus.
- At this time, it activates Bus-Busy to prevent other devices from using the bus at the same time.

(ii) Distributed Arbitration:

- In this approach, all devices participate in the selection of the next bus master.
- Distributed arbitration means that all devices waiting to use the bus have equal responsibility in carrying out the arbitration process, without using a central arbiter.
- A distributed arbitration scheme is depicted as,



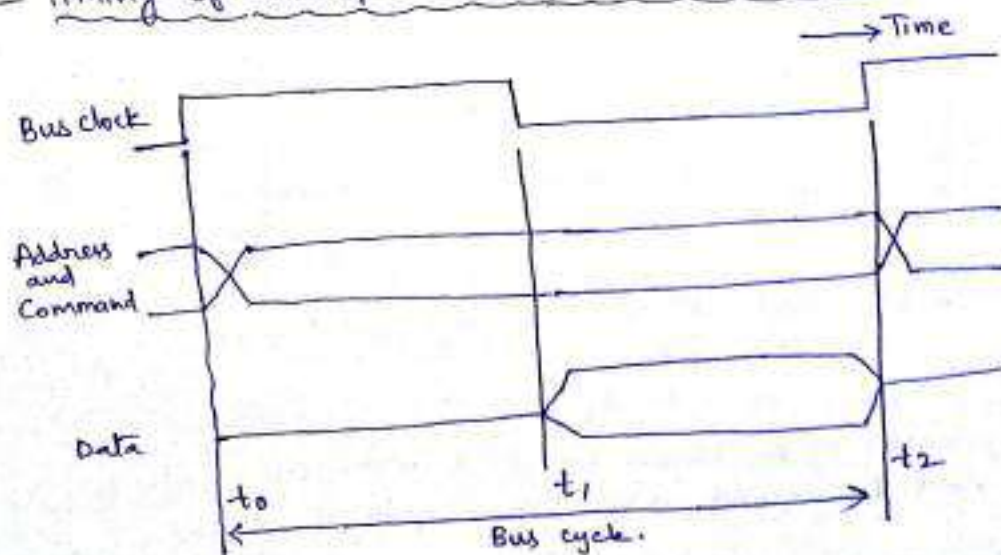
- Each device on the bus is assigned a 4-bit identification number.
- When one (or) more devices request the bus, they assert the Start-Arbitration signal and place their 4-bit ID numbers on four open-collector lines, $\overline{ARB0}$ through $\overline{ARB3}$.
- A winner is selected as a result of the interaction among the signals transmitted over these lines by all contenders.
- The net outcome is that the code on the four lines represents the request that has the highest ID number.

④ BUSES :

- The processor, main memory, and I/O devices can be interconnected by means of a common bus.
- The Bus primary function is to provide a communication path for the transfer of data.
- The bus includes the lines needed to support interrupts and arbitration.
- The bus lines used for transferring data may be grouped into three types.
 - data lines
 - address lines
 - Control lines
- In any data transfer operation, one device plays the role of a master.
- This is the device that initiates data transfers by issuing read (or) write commands on the bus, hence, it may be called an initiator.
- Normally, the processor (or) devices with DMA capability become bus masters.
- The device addressed by the master is referred to as a slave (or) target.

(i) Synchronous Bus:

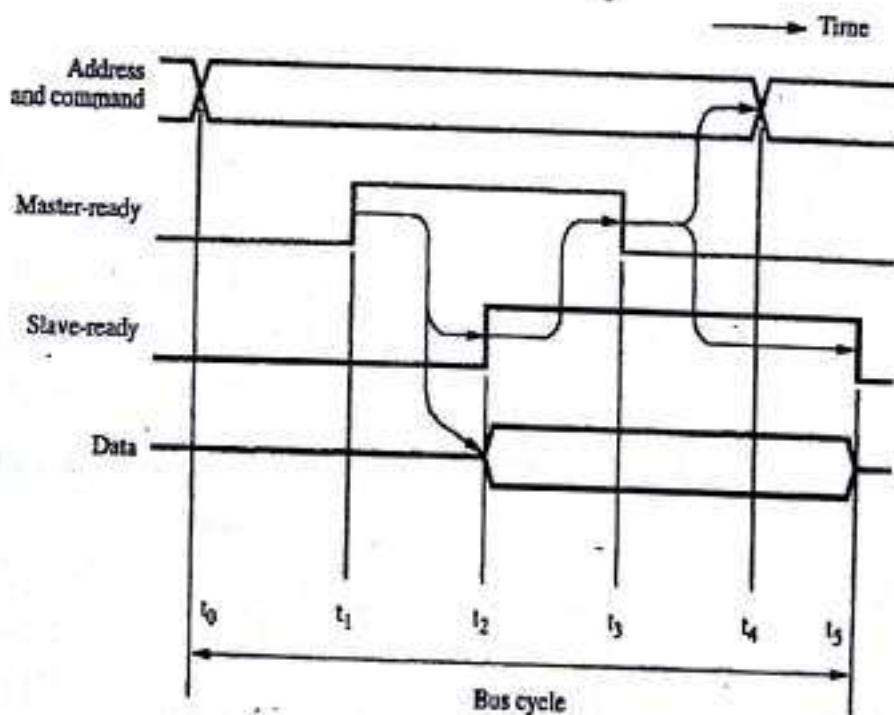
- In a synchronous bus, all devices derive timing information from a common clock line.
- Equally spaced pulses on this line define equal time intervals.
- In the simplest form of a synchronous bus, each of these intervals constitutes a bus cycle during which one data transfer can take place.
- Timing of an input transfer on a Synchronous bus is depicted as



- The address and datalines are high and low at the same time.
- This is a common convention indicating that some lines are high and ~~low~~ some low, depending on the particular address (or) data pattern being transmitted.
- The crossing points indicate the times at which these patterns change.
- A signal line in an indeterminate (or) high-impedance state is represented by an intermediate level half-way between the low and high signal levels.

(ii) Asynchronous Bus:

- An alternative scheme for controlling data transfer on the bus is based on the use of a handshake between the master and the slave.
- Handshake control of data transfer during an input operation is depicted as,

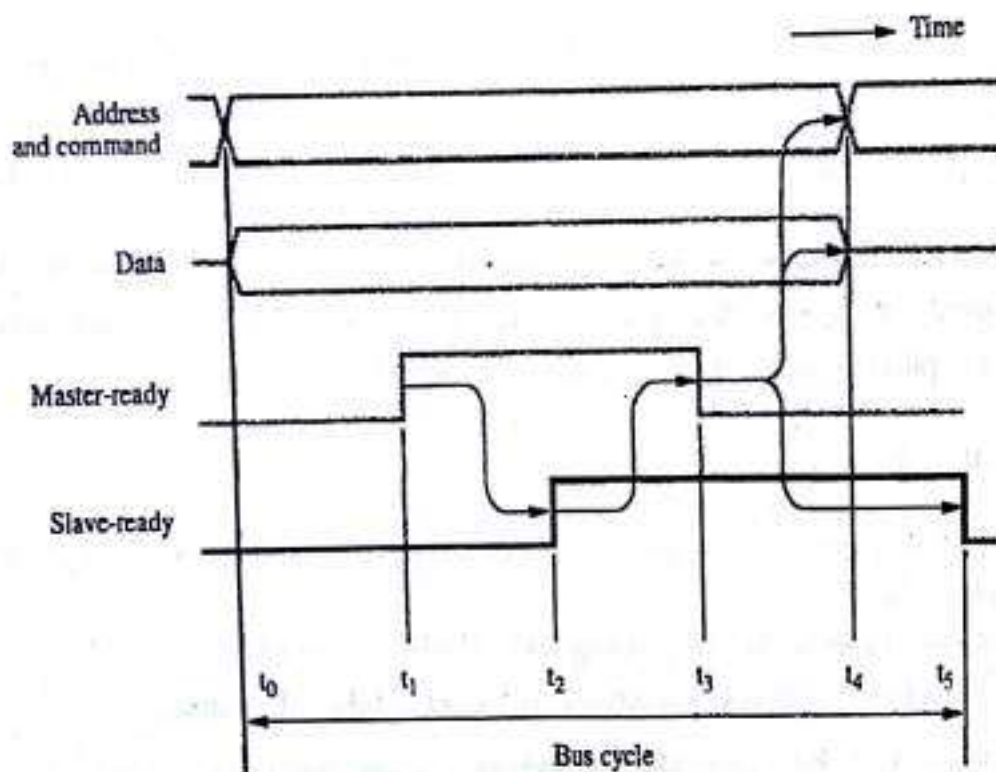


- The common clock is replaced by two timing control lines
 - Master-Ready
 - Slave-Ready
- The first is asserted by the master to indicate that it is ready for a transaction, and the second is a response from the slave.

→ In principle, a data transfer controlled by a handshake proceeds as,

- The master places the address and Command information on the bus.
- Then it indicates to all devices that it has done so by activating the Master-ready line.
- This causes all devices on the bus to decode the address.
- The selected slave performs the required operation and informs the processor it has done so by activating the Slave-ready line.
- The master waits for Slave-ready to become asserted before it removes its signals from the bus.
- In the case of a read operation, it also strobes the data into its input buffer.

→ Handshake Control of data transfer during an output operation is depicted as



- In this case, the master places the output data on the data lines at the same time that it transmits the address and Command information.
- The selected slave strobes the data into its output buffer when it receives the Master-ready signal and indicates that it has done so by setting the Slave-ready signal to 1.
- The remainder of the cycle is identical to the input operation.

⑤ Interface Circuits:

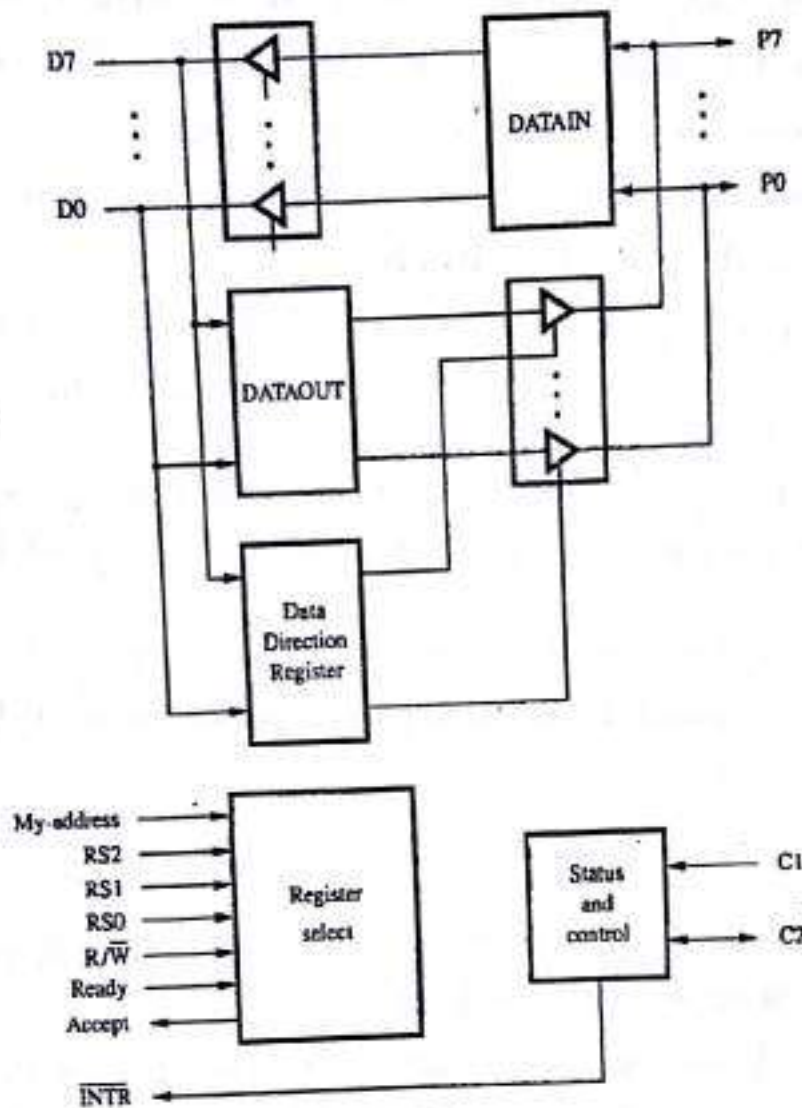
- An I/O interface consists of the circuitry required to connect an I/O device to a computer bus.
- On one side of the interface, we have the bus signals for address, data and control.
- On the other side we have a data path with its associated controls to transfer data between the interface and the I/O device.
- This is called a port.
- It can be classified as

- (i) Parallel Port
- (ii) Serial Port

- A parallel port transfers data in the form of a number of bits, typically 8 or 16, simultaneously to or from the device.
- A Serial port transmits and receives data one bit at a time
- Communication with the bus is the same for both formats, the conversion from the parallel to the serial format, and vice versa, takes place inside the interface circuit.

(i) Parallel Port:

- A parallel port is a type of interface found on computers, for connecting peripherals.
- The name refers to the way the data is sent
- parallel ports send multiple bits of data at once,
- A general 8-bit parallel interface is depicted as,

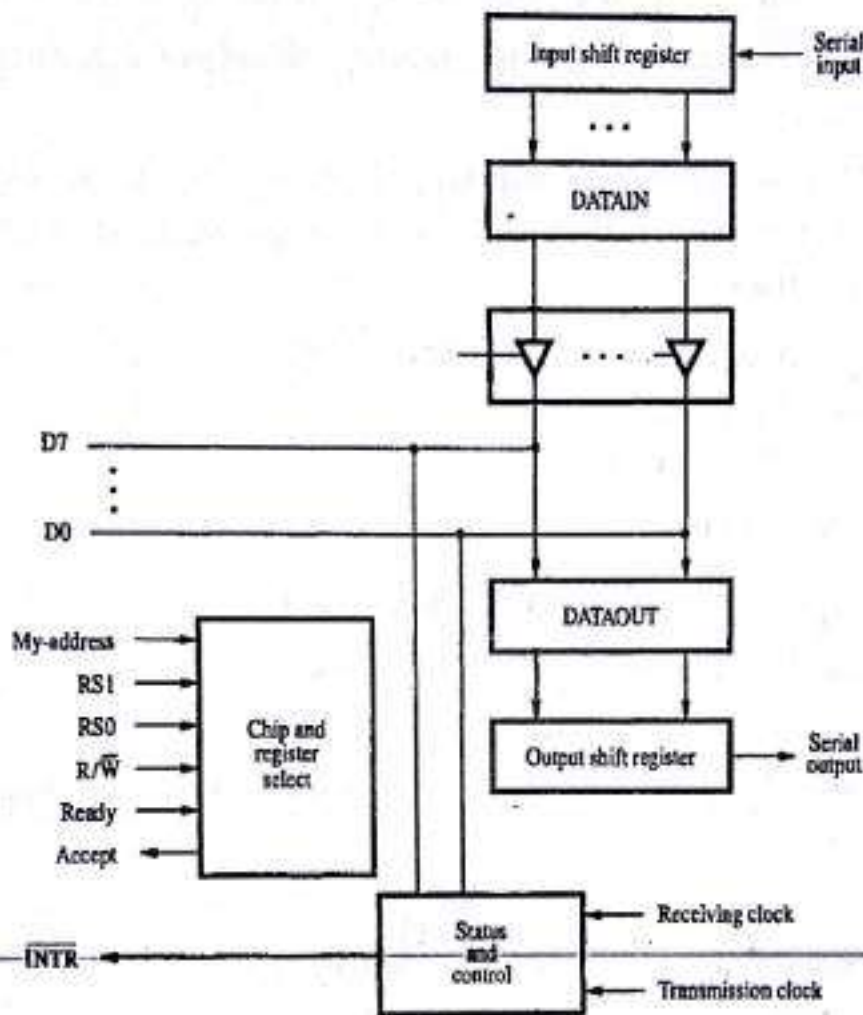


- Data lines P7 through P0 can be used for either input (or) output purposes.
- For increased flexibility, the circuit makes it possible for some lines to ~~have~~ serve as inputs and some lines to serve as outputs, under program control.

- The DATAOUT register is connected to these lines via three-state drivers that are controlled by a data direction register, DDR.
- For a given bit, if the DDR value is 1, the corresponding data line acts as an output line, otherwise, it acts as an input line.
- Two lines C1 and C2, are provided to control the interaction between the interface circuit and the I/O device it serves.
- Line C2 is bidirectional to provide several different modes of signaling, including the handshake.
- The Ready and Accept lines are the handshake control lines on the processor bus side, and hence would be connected to Master-ready and Slave-ready.
- The input signal My-address should be connected to the output of an address decoder that recognizes the address assigned to the interface.
- An interrupt request output, $\overline{INTR}$, is also provided.
- It should be connected to the interrupt-request line on the Computer bus.

(ii) Serial Port :

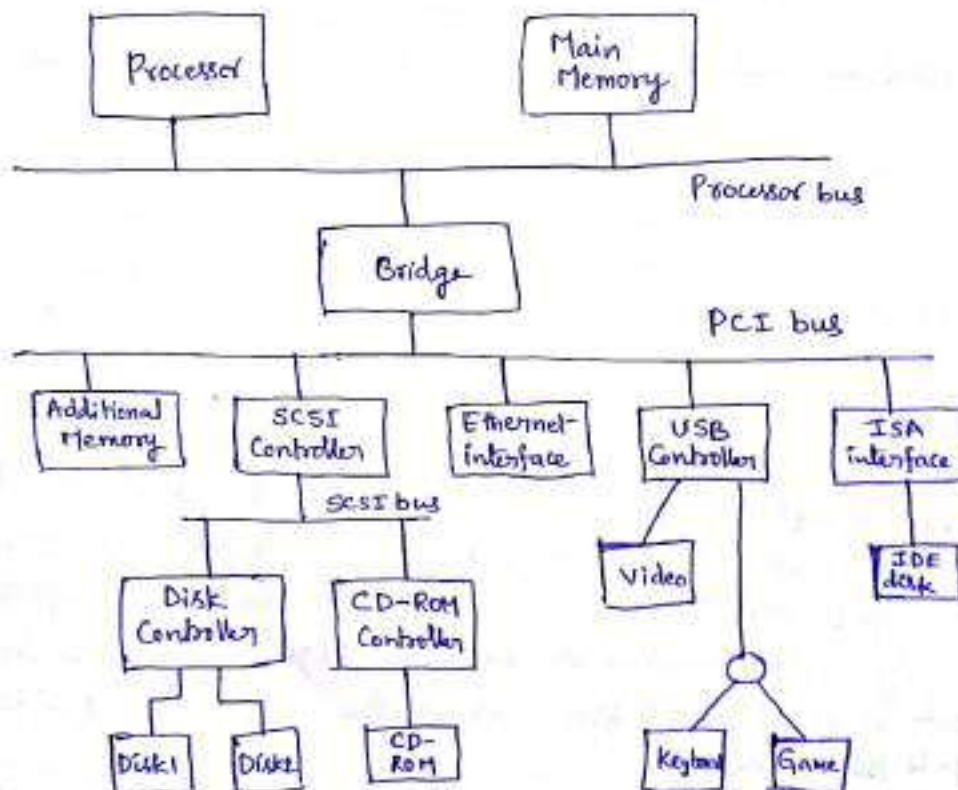
- A Serial port is used to connect the processor to I/O devices that require transmission of data one bit at a time.
- The key feature of an interface circuit for a serial port is that it is capable of communicating in a bit-serial fashion on the device side and in a bit-parallel fashion on the bus side.
- The transformation between the parallel and serial formats is achieved with shift registers that have parallel access capability.
- A Serial interface is depicted as,



- It includes the familiar DATAIN and DATAOUT registers.
- The input shift register accepts bit-serial input from the I/O device.
- When all 8 bits of data have been received, the contents of this shift register are loaded in parallel into the DATAIN register.
- Similarly, output data in the DATAOUT register are loaded into the output shift register, from which the bits are shifted out and sent to the I/O device.

⑥ Standard I/O Interfaces:

- A different interface may have to be designed for every combination of I/O device and computer, resulting in many different interfaces.
- The most practical solution is to develop standard interface signals and protocols
- The two buses are interconnected by a circuit, which we will call a bridge, that translates the signals and protocols of one bus into those of the other.
- The different standard I/O Interfaces are,
 - (i) PCI bus
 - (ii) SCSI bus
 - (iii) USB
- PCI - Peripheral component Interconnect
- SCSI - Small Computer System Interface
- USB - Universal Serial Bus
- An example of a computer system using different interface standards is depicted as,



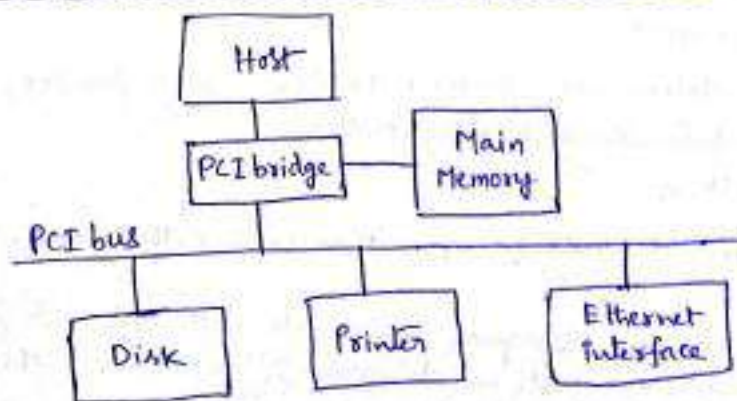
- ISA - Industry Standard Architecture
- IDE - Integrated Device Electronics
- The PCI standards defines an expansion bus on the motherboard.
- SCSI and USB are used for connecting additional devices, both inside and outside the computer box.

(i) PCI (Peripheral Component Interconnect) Bus:

- The PCI bus is a good example of a system bus that grew out of the need for standardization.
- It supports the functions found on a processor bus but in a standardized format that is independent of any particular processor.
- Devices connected to the PCI bus appear to the processor as if they were connected directly to the processor bus.
- They are assigned addresses in the memory address space of the processor.

Data transfer:

- When the processor specifies an address and requests a read operation from the main memory, the memory responds by sending a sequence of data words starting at that address.
- Similarly, during a write operation, the processor sends a memory address followed by a sequence of data words, to be written in successive memory locations starting at that address.
- The PCI is designed primarily to support this mode of operation.
- A read (or) write operation involving a single word is simply treated as a burst of length one.
- The use of a PCI bus in a Computer System is depicted as,



- The PCI bridge provides a separate physical connection for the main memory.
- For electrical reasons, the bus may be further divided into segments connected via bridges.
- However, regardless of which bus segment a device is connected to, it may still be mapped into the processor's memory address space.
- The Data Transfer Signals on the PCI bus is given as

Name	Function
CLK	A 33-MHz (or) 66-MHz clock
FRAME#	Sent by the initiator to indicate the duration of a transaction
AD	32 address/data lines, which may be optionally increased to 64
C/BE#	4 Command / Byte-Enable lines (8 for 64-bit bus)
IRDY#, TRDY#	Initiator-Ready, Target-Ready Signals
IDSEL#	Initialization Device Select

Device Configuration:

- The PCI simplifies the process by incorporating in each I/O device interface a small Configuration ROM memory that stores information about that device.
- The Configuration ROMs of all devices are accessible in the Configuration address space.
- The PCI initialization software reads these ROMs whenever the system is powered up (or) reset.
- In each case, it determines whether the device is a printer, a keyboard, an Ethernet Interface, (or) a disk controller.

Electrical Characteristics:

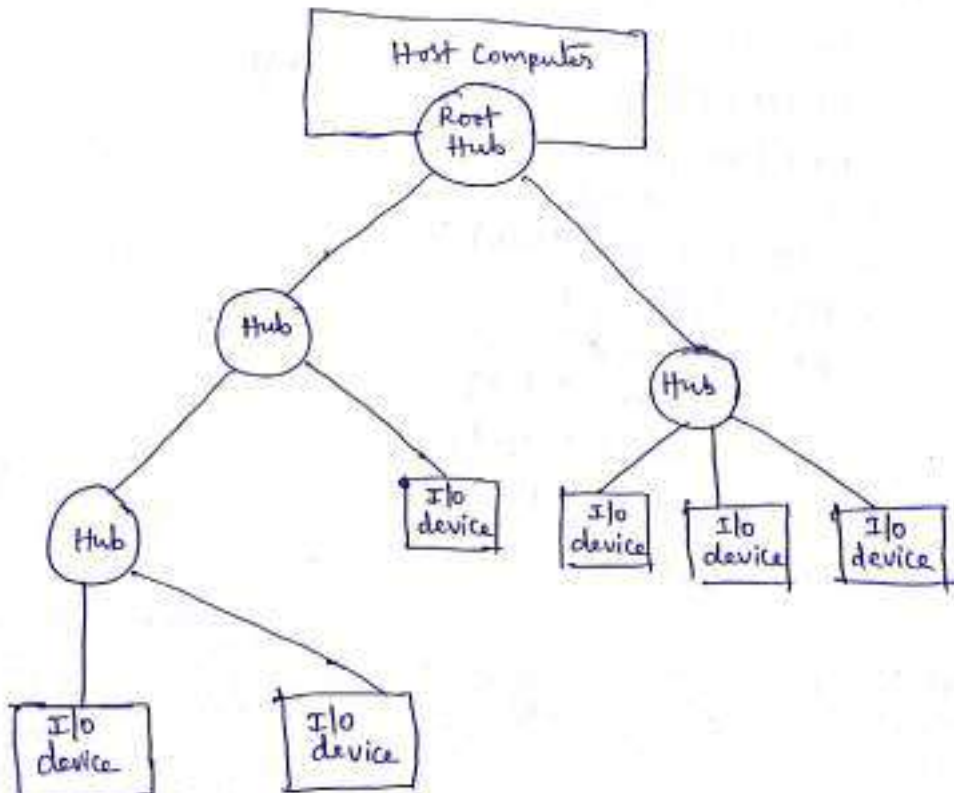
- The PCI bus has been defined for operation with either a 5-V (or) 3.3V power supply.
- The motherboard may be designed to operate with either signaling system.
- Connectors on expansion boards are designed to ensure that they can be plugged only in a compatible motherboard.

(ii) SCSI Bus :

- SCSI stands for Small Computer System Interface.
- It refers to a standard bus defined by the ANSI (American National Standards Institute) under the designation X3.131
- In the original specifications of the standard, devices such as disks are connected to a computer via a 50-wire cable, which can be up to 25 meters in length and can transfer data at rates up to 5 megabytes/s
- A SCSI bus may have 8 datalines, in which case it is called a narrow bus and transfers data one byte at a time.
- A wide SCSI bus has 16 data lines and transfers data 16-bits at a time.
- The SCSI bus standard has undergone many revisions, and its data transfer capability has increased very rapidly, almost doubling every two years.
- SCSI-2, SCSI-3 have been defined, and each has several options.
- The different SCSI bus signals are
 - DB (Data lines)
 - DB(P) (Parity bit for the data bus)
 - BSY (Busy)
 - SEL (Selection)
 - C/D (Control/Data)
 - MSG (Message)
 - REQ (Request)
 - ACK (Acknowledge)
 - I/O (Input/Output)
 - ATN (Attention)
 - RST (Reset)

Q11) USB (Universal Serial Bus):

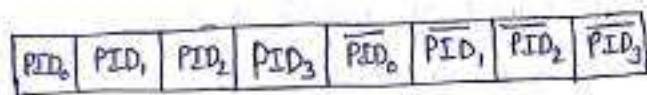
- A simple, low-cost mechanism to connect the devices to the computer is possible using USB.
- The USB supports two speeds of operation
 - * Low-speed (1.5 Mb/s)
 - * Full-speed (12 Mb/s)
- The recent development is USB 2.0
- The USB has been designed to meet several key objectives.
 - provide a simple, low-cost, and easy to use interconnection system.
 - Accommodate a wide range of data transfer characteristics for I/O devices, including telephone and Internet connections.
 - Enhance user convenience through a "plug-and-play" mode of operation.
- USB Tree structure is depicted as,



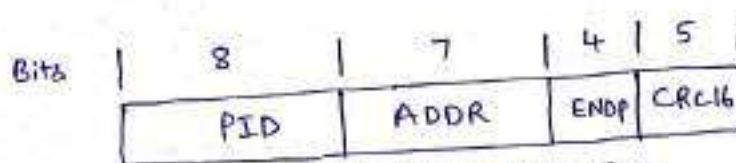
- To accommodate a large number of devices that can be added (or) removed at any time, the USB has the tree structure.
- Each node of the tree has a device called a hub, which acts as an intermediate control point between the host and the I/O devices.
- At the root of the tree, a root hub connects the entire tree to the host computer.
- The leaves of the tree are the I/O devices being served (for example, keyboard, Internet connection, speaker (or) digital TV), which are called functions in USB terminology.

USB protocols:

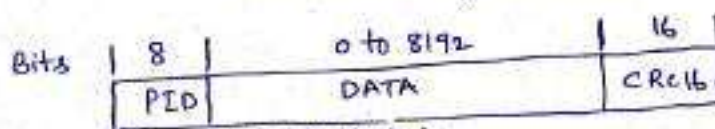
- All information transferred over the USB is organized in packets, where a packet consists of one (or) more bytes of information.
- The information ~~divided~~ transferred on the USB can be divided into two broad categories.
 - Control
 - Data
- Control packets perform such tasks as addressing a device to initiate data transfer, acknowledging that data have been received correctly, (or) indicating error.
- Data packets carry information that is delivered to a device.
- A packet contains one or more fields with different kinds of information.
- The first field of any packet is called the packet identifier, PID, which identifies the type of that packet.
- USB packet formats is depicted as,



(a) Packet Identifier field



(b) Token Packet, IN (or) Out



(c) Data packet

- ADDR - Address
- ENDP - Ending Packet
- CRC - Cyclic Redundancy ~~Check~~ Check

Isochronous Traffic on USB:

- One of the key objectives of the USB is to support the transfer of isochronous data, such as sample voice, in a simple manner.
- Devices that generate (or) receive isochronous data require a time reference to control the sampling process.
- To provide this reference, transmission over the USB is divided into frames of equal length.
- A frame is 1 ms long for full and low speed data.
- The root hub generates a Start Of Frame (SOF) control packet precisely once every 1 ms to mark the beginning of a new frame.
- USB frames depicted as,



(a) SOF Packet



(b) Frame Example

Electrical characteristics,

- The cables used for USB connections consists of four wires
 - Two are used to carry power, +5V and Ground.
 - The other two wires are used to carry data

① Discuss about Single bus structure.
Page 4.1

② Explain about Memory-mapped I/O.
Page 4.2

③ What is programmed I/O?

Programmed I/O:

→ Programmed I/O is a method of transferring data between the CPU and a peripheral.

Advantages:

- I/O Command is issued by the processor to the respective I/O module.
- Requested I/O instruction is executed at the earliest.
- I/O module switches to the next task as soon as it completes the task.

Disadvantages:

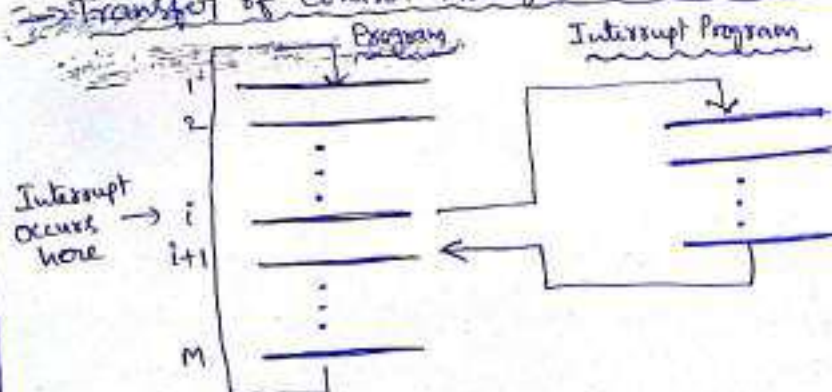
- Processor has to wait until the I/O module is ready for performing data transfer.
- Processor have to interrogate several times regarding the status of I/O module.

④ What is interrupt-initiated data transfer?

Interrupt-initiated Data transfer:

- When the I/O device is ready to transfer data, instead CPU keep asking, does not check the flag.
- When the flag in the interface is set, the interface will initiate an interrupt, and the CPU will drops what it is doing and do the I/O transfer.
- Until the I/O transfer is done, it returns to what it was doing

→ Transfer of control through the use of Interrupts is depicted as,



⑤ Discuss Daisy-chain priority Interrupt
page 4.6

⑥ Explain a) Interrupt b) Exception

a) page 4.3

b) Exception:

- An exception is an abnormal (or) unusual termination
- An exception is a condition that results from software and prevents the processor from executing the current instruction stream.
- It affects the normal execution of an instruction.

⑦ What is DMA?

page 4.7

⑧ What is the need for Bus Arbitration?

page 4.8

⑨ Discuss handshaking (or) Asynchronous Data transfer (or) Asynchronous Bus

page 4.12

⑩ Explain PCI bus.

page 4.19

⑪ Explain SCSI bus.

page 4.21

⑫ Explain USB.

page 4.22