INTRODUCTION TO DIGITAL COMPUTER:

Concept of Digital Computer:

Digital systems move, store and process information. Information represents broad range of parameters. The parameters can be continuous or discrete. Continuous parameters are capable of taking all possible values over a defined range.

Analog signal - It is a continuous signal which takes all possible values over a defined range.

Digital signal - It is a discrete signal that occupy only a finite number of values over a range. The signals in most present-day electronic digital systems use just two discrete values and are therefore said to be binary. Typically two discrete values are represented as HIGH and LOW. (1 and 0 respectively).

Positive logic - Here we assume HIGH/1 is associated with higher voltage range. LOW/0 is associated with lower voltage range.

Negative logic - LOW/0 is associated with how higher voltage range. HIGH/1 is associated with lower voltage range.

The Digital Computer:

A block diagram of a digital computer is shown in Fig ①.

The memory stores programs as well as input, output and intermediate data. The data path performs arithmatic and other data-processing operations as specified by the program.

The control unit supervises the flow of information between the various units. A datapath, when combined with the control unit, forms a component called Central Processing Unit (CPU).

The program and data prepared by the user are transfered into memory by means of an input device. (Eg Keyboard). An output device, (Eg LCD), displays the result of the computations and presents them to the user.

These input and output devices use digital logic circuits, but can often include analog circuits.

The control unit in the CPU retrieves the instructions one by one, from the program stored in the memory. For each instruction, the control unit manipulates the datapath to execute the operations specified by the instruction. Both program and data are stored in memory. A digital computer can perform arithmatic computations, manipulate strings of alphabetic characters, and be programmed to make decisions based on external and internal conditions.

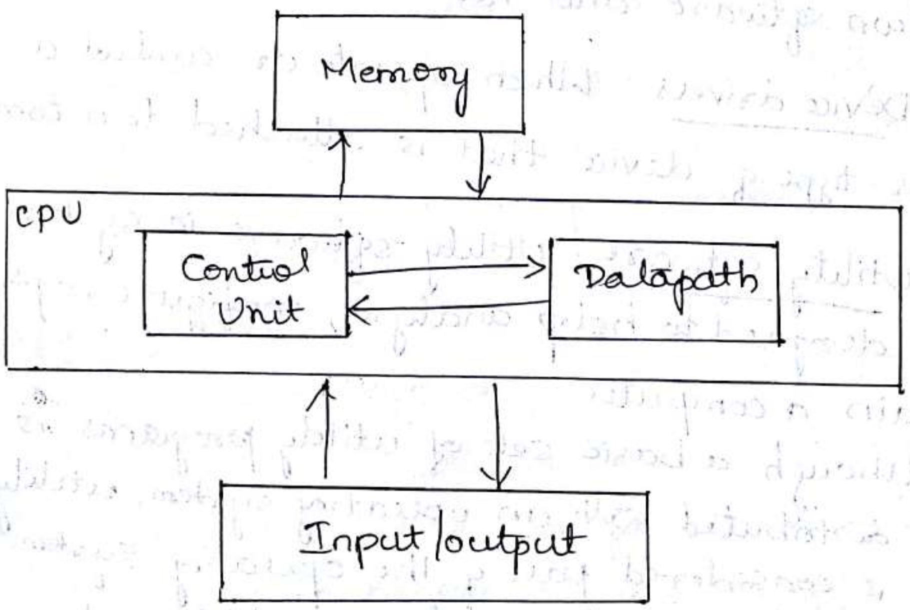


Fig ① Block diagram of a Digital Computer

Software (Computer Software):

It is a collection of data or computer instruction that tell the computer how to work. Program is a called of computer instructions, many programs collectively makes up a software. It includes computer programs, libraries and related non-executable data, such as online documentation or digital media.

Types of Software:

Based on the purpose or domain of use software is divided as:

1. System software 2. Application software:

1. System software - It is a software that directly operates the computer hardware to provide basic functionality needed by users and other software, and to provide a platform for running application software. It includes operating systems, device drives and utility software. including developmet software.

a. operating systems - It is a collection of software that manages the interaction between application software and hardware.

b. Device drivers: Which operate or control a particular type of device that is attached to a computer

c. Utility software: Utility software is system software designed to help analyse, configure, optimize or maintain a computer.

Although a basic set of utility programs is usually distributed with an operating system, utility software is considered part of the operating system users often install replacements or additional utilities.

It provides additional facilities to carry out tasks which are beyond the capabilities of the operating system

- Anti virus - utilities scan for computer viruses and remove them.

- Disk checkers - scan an operating system hard drive and check for logical or physical errors.

- Backup software - makes copies of all information stored on a disk and restores either the entire disk or selected files.

- Cryptographic - utilities encrypt and decrypt streams and files.

d) Development software -

These are the software which helps in development of other programs or software for the programmer/ developer. Eg. Compiler, Assembler.

When a software is called a System software!!

1) Running of other software.
2) Communicating with peripheral devices such as printers, card readers, tape drives.
3) Development of other types of software.
4) Monitoring the use of various hardware resources such as memory, peripherals, CPU, etc.

(2) Application software:

An application software is computer software designed to perform a group of coordinated functions, tasks or activities for the benefit of the user.

Eg. word processors, Web browsers, Spreadsheets, database programs.

(*) Some important questions:

1. Write the differences between System Software & application software.
2. What are the types of Software. Give examples for each.
3. What is software. Explain the types.
4. What is System Software. Explain when a software is called system software.

Compiler:

Compiler is a computer software that translate source program from a high level programming language to a lower level language. (Assembly language or machine Level language). It produces an executable file after translation.

Eg. Turbo C compiler.

Interpreter:

Interpreter is a computer program / software that executes instructions written in a programming or scripting language without requiring them previously to have been compiled into a machine language program. It translates one line at a time.

(*) Differentiate between compiler and Interpreter.

Linker:

Linker is a program that takes one or more object files generated by a compiler and combines them into a single executable file.

It also links a particular module into system library.

Assembler:

It is a program for converting instructions written in low-level symbolic code into machine code. The assembly level language is converted to machine level language. Assembly level language uses mnemonic to represent each low-level machine instruction.

Loader -

Loader is a program that loads programs and libraries to the main memory for execution. It reads the contents of executable file containing the program instructions into memory and then carry out other required preparatory tasks to prepare the executable for running.

SYSTEM BUS STRUCTURE :-

A system bus is a single computer bus that connects the major components of a computer system, combining the functions of a data bus, address bus and control bus.

Fig (2)

A bus is a communication system that transfers data between components inside a computer or between computers.

Data bus - A data bus is a system within a computer or device consisting of a connector or set of wires, that provides transportation for data.

Address bus - An address bus is a computer bus (a series of lines connecting two or more devices) that is used to specify a physical address. When a processor needs to read or write to a memory location, it specifies that memory address on the address bus.

Eg A system with 32-bit address bus can address 2^{32} memory locations.

(Control bus –

A control bus is a computer bus that is used by the CPU to communicate with devices that are contained within the computer. The CPU transmits a variety of control signals to components and devices, to transmit control signals to the CPU using the control bus.

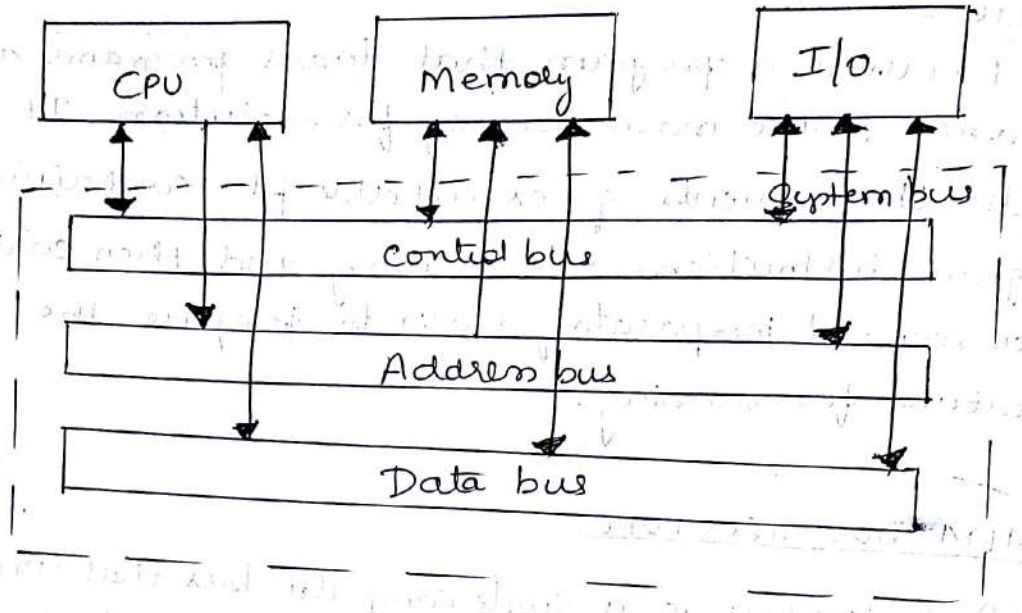


Fig ② System bus Structure.

Memory :-

Memory refers to hardware integrated circuits that store information for immediate use in a computer.

Types of memory : 1) Primary | Main memory – It holds only those data and instructions on which the computer is currently working. It has limited capacity and data is lost when power is switched off. Eg. RAM.

2) Secondary memory – (External | non-volatile memory). These are used for storing data | information permanently. It is slower than main memory. Eg. DVD, CD-ROM.

3) Cache memory – Cache memory acts like a buffer between the CPU and main memory. Which speed up the CPU.

Module-2

Number System and Logic design minimization Techniques

1. Introduction to number system

Number System -

Number system is used to represent numbers by strings of digits Depending upon the position in the string, each digit has an associated value of an integer raised to the power of specific number system. Eg The number in decimal number system 724.5 is interpreted as 7 hundreds plus 2 tens plus 4 units plus 5 one by 10-ths.

$$724.5 = 7\times10^2 + 2\times10^1 + 4\times10^1 + 5\times10^{-1}$$

Decimal Number System: The digits used 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. 10 digits are used so the base is 10 also called radix

Binary number system – Radix or base is 2, Digits used 0 and 1

Octal - Radix or base 8, Digits used 0, 1, 2, 3, 4, 5, 6, 7

Hexadecimal - Radix or base 16, Digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
(10)(11)(12)(13)(14)(15).

Conversions between the Number Systems

1. Decimal to Binary. – Method used is repeated division by 2 till the quotient becomes 0 for whole numbers and taking in reverse order the reminders.

Eg. $35_{(10)}$

2	35
2	18 – 1
2	9 – 0
2	4 – 1
2	2 – 0
2	1 – 0
	0 – 1

$\therefore 35_{(10)} = 0100101_{(2)}$

Try for: $128_{(10)}$, $6612_{(10)}$

2. Binary to decimal :-

Eg. $10100_{(2)}$

Method: Multiply every digit with the powers of 2, increasing from left to right.

Eg. $10100_{(2)}$

$1\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0$

$16 + 8 + 4 + 0 + 0$

$\therefore 10100_{(2)} = 28_{(10)}$

Try for: ① $1011111_{(2)}$ ② $10001001_{(2)}$

(Module) Unit-2
Boolean Algebra and Logic gates

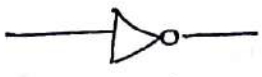
Logic gates

- A logic gate is an electronic circuit with one or more logical inputs and with only one output.
- It is the basic building block of digital system.
- Logic design is interconnection of gates to perform variety of logical operations.
- Truth table consists of all combinations of i/p conditions with corresponding o/p logic level.

NOT, OR, AND - basic gates
NAND, NOR - Universal gates.

1. NOT gate :

Symbol :



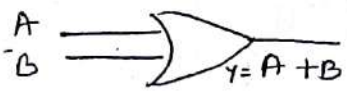
Truth table:

I/ps	O/P
A	$Y = \bar{A}$
1	0
0	1

The NOT gate complements or inverts the logical level i/ps. If i/p is 1 the o/p will be 0. and vice-versa. It can be called as an inverter.
The complement of A is represented by A' or $\bar{A}$.

2. OR gate :-

Symbol :



Truth table:

I/ps		O/P
A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

The OR gate performs logical addition operation on 2 or more variables.

- The OR function yields o/p as logic one when any 1 or all i/ps are high. It produces logic 0 when all i/ps are logic zero.

3. AND gate :-

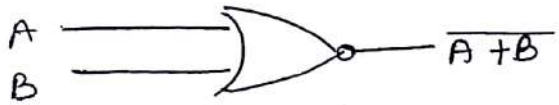
Symbol: 

∴ Truth table

I/ps		O/P
A	B	Y = A·B
0	0	0
0	1	0
1	0	0
1	1	1

The AND gate performs and function by logical multiplication (·) operation on 2 or more variables. It produces an output 1 when all the i/ps are 1 else 0. (Logic zero)

4) NOR gate



Truth table :

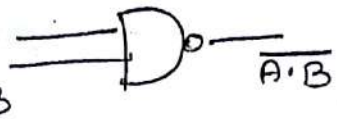
I/ps		O/P
A	B	$Y = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

The NOR gate is equivalent to OR with its o/p complement. It is also called an universal gate because using NOR gate we can realise all other gates.

Lavanya B. Koppal

When any one of the i/p is logic one (1) the output is logic 0. else logic 1.

NAND gate:- Symbol



Truth table:

I/ps A	I/ps B	O/p $Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

NAND gate is equivalent to AND gate with output complemented.

The output of NAND gate is logic 0 when all the i/ps are logic 1.

They are also called universal gate as they can be used to realise other gates.

XOR gate (Exclusive OR gate)

Symbol:

I/ps A	I/ps B	O/p $Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

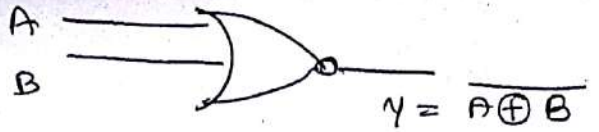
$Y = A \oplus B = \bar{A}B + \bar{B}A$

When there are odd no. of 1s the o/p is 1 else o/p is logic 0.

It is also called anti coincidence gate. A 2 i/p XOR gate is called inequality detector as it produces 1 output only when its 2 i/ps are not equal.

Lavanya B. Rappal

XNOR gate :-

Symbol: 

Truth table.

I/P		O/P
A	B	$Y = \overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

$Y = \overline{\overline{A}B + \overline{B}A} = \overline{A \oplus B}$

It is an coincidence gate. It produces a 1 when the i/ps are same. It is also called equality detector as it produces one O/p only when 2 i/ps are equal.

Realisation of other gates using NAND gate.

NOT gate using NAND

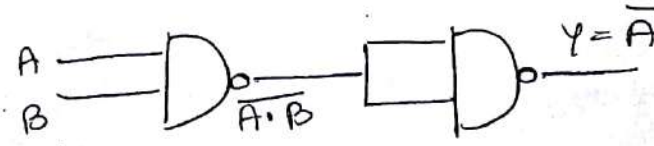
A NAND gate is used to make NOT gate by connecting all the input together.

Symbol: 

Exercise:
Mention the truth table.

AND gate using NAND

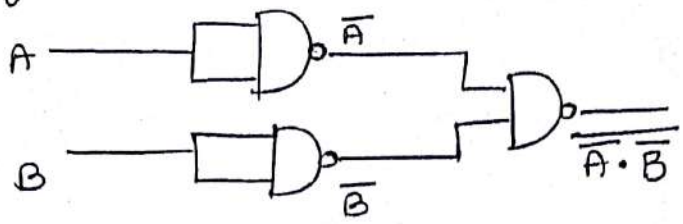
For AND gate we need 2 NAND gates. One used as normal NAND gate and other used as NOT gate.



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

Lavanya B. Koppal

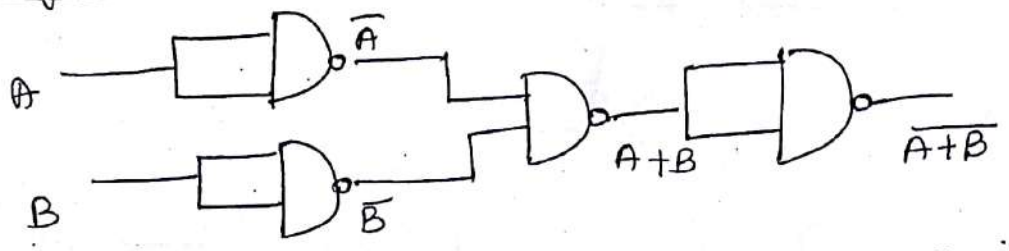
OR gate using NAND



Two NAND gates are used to invert the 2 i/ps A & B The complemented i/ps are applied to NAND gate. By applying De Morgan's theorm we get the o/p as OR.

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

NOR gate using NAND

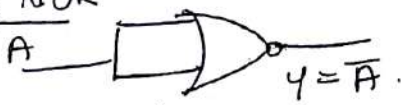


2 NAND gates are used to invert the 2 i/p variables and output of it is applied to NAND gate. Again the output of it is provided to the NOT gate form of NAND gate to produce NOR

Exercise: Realise XOR and XNOR using NAND gate.

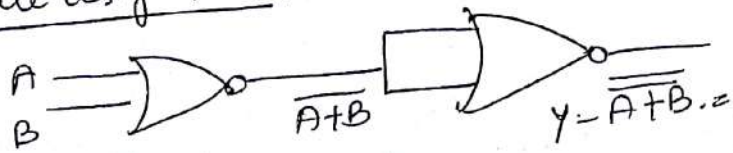
Realisation of other gates using NOR gate:

NOT gate using NOR



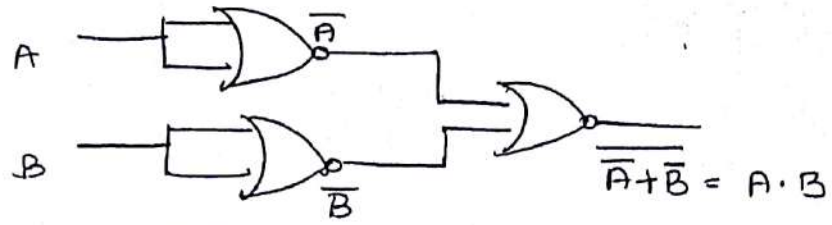
A NOR gate can be used as NOT gate by connecting all its i/p terminals together.

OR gate using NOR :-



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

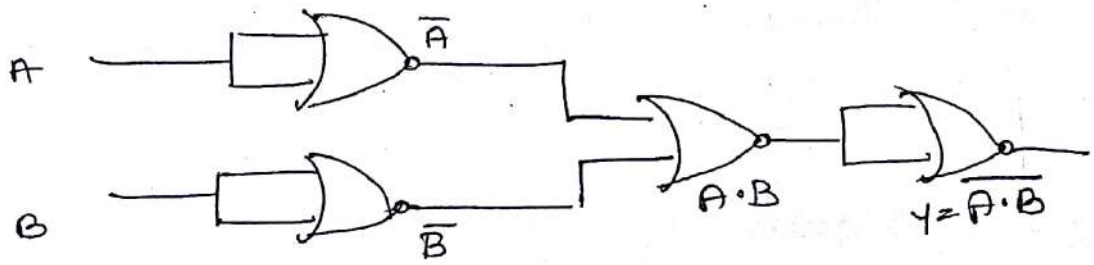
⑥ AND gate using NOR :-



$Y = \overline{\overline{A}+\overline{B}} = \overline{\overline{A}} \cdot \overline{\overline{B}} = A \cdot B$ using De Morgan's Theorem.

2 NOR gates is used to invert 2 input variables & applied to one more NOR gate. The AND function is derived by using De Morgan's Theorem.

NAND gate using NOR.



2 NOR gates are used to invert 2 input variable and applied to one more NOR gate. This result is again given to NOT gate form of NOR gate to get NAND function output.

Exercise: Realise XOR and XNOR using NOR gate

BOOLEAN ALGEBRA :-

Boolean algebra is a mathematical tool used to describe relation between logic circuit output and its inputs. The basic rules, laws and theorms of boolean algebra will help to maintain and minimize the logical expressions that represent circuits.

In boolean algebra the variables used in the functions are called boolean variables.

Lavanya B. Koppal

operations | Functions used in boolean algebra.

1. Complement of a variable [NOT operation]

The complement of a variable is represented by a bar over the letter. or a dash after the letter (variable)

$\bar{A}$, A' respectively.

Rules for complementing : $\bar{0} = 1$, $\bar{1} = 0$.

2. Boolean Addition :- [OR operation]

$A + B$.

Rules for Addition :- $0+0=0$ $1+0=1$
$0+1=1$ $1+1=1$

3. Boolean multiplication [AND operation]

$A \cdot B$

Rules for multiplication : $0 \cdot 0 = 0$ $1 \cdot 0 = 0$
$0 \cdot 1 = 0$ $1 \cdot 1 = 1$

Advantages of Boolean algebra:

1. It is easy to work as only 2 values (0 or 1) is used.
2. There are no fractions, decimal, negative numbers square roots, logarithm and imaginary no.
3. There are only 3 basic operations - Complement, boolean addition, boolean multiplication.

Basic laws of boolean Algebra:

1. Commutative law
2. Associative law
3. Distributive law.

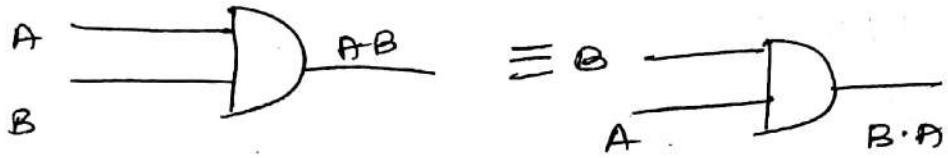
Commutative law:

a) of boolean addition states that the order in which variables are ORed can be reversed.

Lavanya B. Koppal

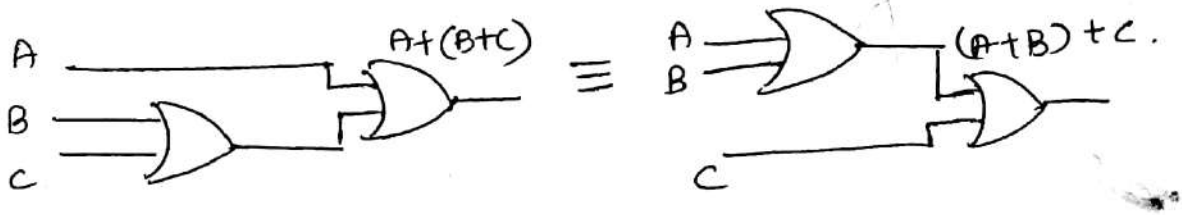
i.e $A+B = B+A$

b) <u>of multiplication</u> :- States that the order in which the variables are ANDed can be reversed.



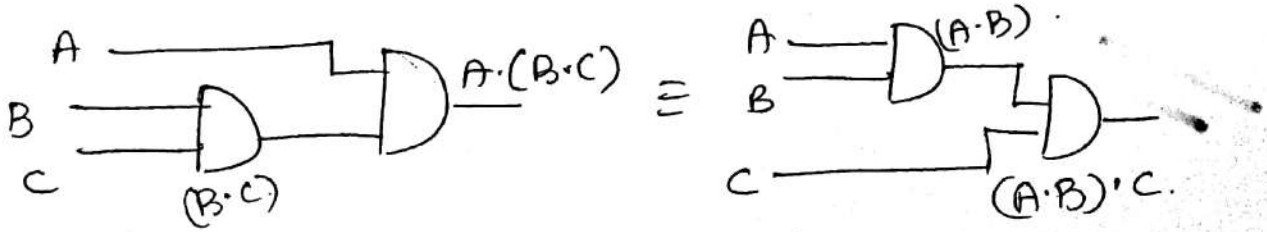
<u>Associative law of boolean addition</u> :- It states that in ORing of several variables the result is same regardless of the grouping of variables.

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



<u>Associative law of boolean multiplication</u> :- It states that in AND ing of several variables the result is same regardless of grouping of variables

i.e. $A\cdot(B\cdot C) = (A\cdot B)\cdot C$.



<u>Distributive law</u> :

It states that ORing several variables and ANDing the result with single variable $A(B+C)$

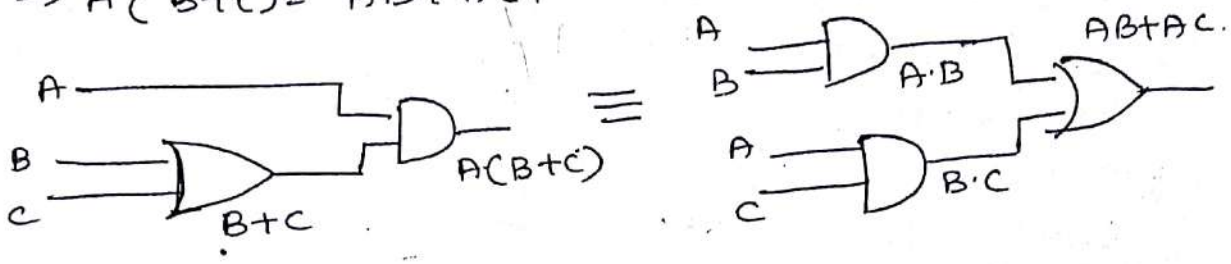
· is equal to ANDing the single variable with each of the several variables and ORing the products.

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

For 4 variables,

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

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



Basic Rules of Boolean Algebra:

Rule 1: A variable ORed with 0 is equal to variable: $A+0=A$.

if $A=1$: $1+0=1=A$

if $A=0$: $0+0=0=A$

Rule 2: A variable ORed with 1 is equal to 1

i.e. $A+1=1$

if $A=1$: $1+1=1$

if $A=0$: $0+1=1=1$

Rule 3: A variable ANDed with 0 is equal to 0

ie. $A\cdot 0=0$

if $A=1$: $1\cdot 0=0$

if $A=0$: $0\cdot 0=0$

Rule 4: A variable ANDed with 1 is equal to A.

i.e $A\cdot 1=A$

if $A=1$: $1\cdot 1=1=A$

if $A=0$: $0\cdot 1=0=A$.

Rule 5 : If variable ORed with itself the o/p is a variable.

ie $A+A=A$

if $A=1$: $1+1=1=A$

if $A=0$: $0+0=0=A$.

Rule 6 : If a variable and its complement are ORed the result is 1

ie $A+\overline{A}=1$

If $A=0$: $0+1=1$

If $A=1$: $1+0=1$

Rule 7 : If a variable is And with itself then the result is equal to that variable.

ie $A \cdot A = A$

if $A=0$: $0 \cdot 0 = 0 = A$

if $A=1$: $1 \cdot 1 = 1 = A$.

Rule 8 : If a variable is ANDed with its complement the result = 0.

ie $A \cdot \overline{A} = 0$

if $A=1$: $1 \cdot 0 = 0$

if $A=0$: $0 \cdot 1 = 0$.

Rule 9 :- If the variable is complemented twice the result is variable itself.

ie $\overline{\overline{A}} = A$.

if $A=0$: $\overline{\overline{0}} = 0 = A$

$A=1$: $\overline{\overline{1}} = 1 = A$.

Rule 10 : $A+AB$

$A(1+B)$ by distributive law.

$A(1)$ by rule $1+B=1$

Rule 11 : $A + \bar{A}B = A + B$ (simplified)

$= A + AB + \bar{A}B$ Using rule 10 $A + AB = A$

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

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

$= A + B$ By rule $B \cdot 1 = B$

Rule 12 : $(A+B)(A+C) = A + BC$. (simplified)

$= A \cdot A + A \cdot C + B \cdot A + B \cdot C$

$= A + A \cdot C + B \cdot A + B \cdot C$ (By $A \cdot A = A$)

$= A(1 + C) + B(A + C)$ By rule $1 + A = 1$

$= A + BA + BC$

$= A(1 + B) + BC$ (By rule $1 + A = 1$)

$= A + BC$.

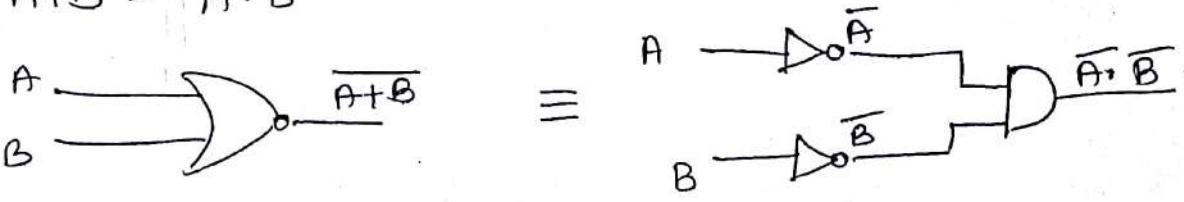
De Morgan's Theorem :-

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

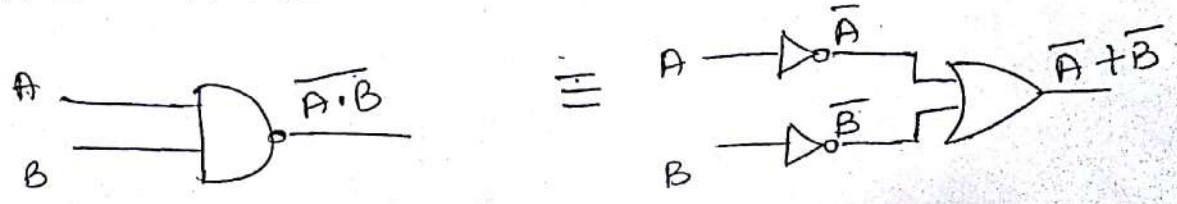
1) $\overline{A+B} = \bar{A}\bar{B}$ The complement of sum is equal to the product of individual complement.

2) $\overline{A \cdot B} = \bar{A} + \bar{B}$ The complement of multiplication is sum of the individual complement.

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



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



Exercise : write the truth table to check

(12) De Morgan's theorem. (Both 1 & 2).

2 important forms of boolean expression:

① Sum of products

② Products of sum.

① Sum of products (SOP) : is the products of 2 or more variables are summed up together. Several AND functions ORed together.

Ex. $AB+BC+CA$ Sum of products

② Product of Sum (POS) is the sum of 2 or more variables that are multiplied together. Several OR functions are logically ANDed together.

Ex. $(A+B)(B+C)$.

Simplify the expressions using laws, rules and Theorms:

① $C+\bar{B}C$

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

Using the rule $\bar{A}+1=1$

$= C(1)$

Using the rule

$A \cdot 1 = A$

$= C$

② $A\bar{B}C + A\bar{B}\bar{C}$

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

Using the rule $A+\bar{A}=1$

Using the rule

$A \cdot 1 = A$

$= A\bar{B}$

Exercise

① simplify:

$(A+C)(AD+A\bar{D})+AC+C$.

② simplify:

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

Note: Please mention the rules on the right hand side that is being used

Lavanya B. Koppal

Module 2 .. cont.
Conversions

1. Binary to decimal conversions :

When converting from binary to decimal, each digit must be multiplied by its weight and the resulting products should be added.

Eg $1101101_{(2)}$

Weight : $2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Binary number : 1 1 0 1 1 0 1

$= 1\times2^6 + 1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$= 64 + 32 + 8 + 4 + 1 = 109$

Exercise

Convert binary to decimal:

① 0.1011_2 ② 10010001_2

③ 10111101.011_2

2. Octal to Decimal Conversion

When converting from octal to decimal each digit must be multiplied by its weight and resulting products must be added. (Powers of 8 is used)

Eg: 2375_8 to decimal

$= 2 \times 8^3 + 3 \times 8^2 + 7 \times 8^1 + 5 \times 8^0$

$= 2 \times 512 + 3 \times 64 + 7 \times 8 + 5 \times 1$

$= 1277_{10}$

Exercise Convert octal to decimal :—

① $0.325_{(8)}$

② $24.6_{(8)}$

Hexadecimal to decimal conversion :-

When converting from hexadecimal to decimal, each digit must be multiplied by its weight and the resulting products should be added. (Powers of 16 is used)

Eg Convert $A85_{16}$ to decimal.

$= A \times 16^2 + 8 \times 16^1 + 5 \times 16^0$

$= 10 \times 16^2 + 8 \times 16 + 5 \times 1$

$= 2693_{10}$

Exercise : Convert hexadecimal to decimal.

① $F8E6.39_{16}$

② $86A.45_{16}$

Decimal to Binary conversion :-

* For whole numbers,
- Conversion from decimal to binary is performed by dividing decimal number repeatedly by 2 and writing down the reminders.
- When a quotient is 0, the conversion is complete.
- The reminders taken in the reverse order is the binary number.

* For fractional numbers,
- If there is a fractional part in the decimal numbe then it is to be multiplied repeatedly by 2 and carry is recorded in the integer position
- The process of multiplication is continued till the desired accuracy (precision) is reached.
- Reading the carry downward gives the binary fraction.

Ex.

Convert 45_{10} to Binary

$2 \overline{)45}$ 22, 44, 1 (LSB)

$2 \overline{)22}$ 11, 22, 0

$2 \overline{)11}$ 5, 10, 1

$2 \overline{)5}$ 2, 4, 1

$2 \overline{)2}$ 1, 2, 0

$2 \overline{)1}$ 0, 0, 1 (MSB)

Stopping criteria = Quotient = 0

Stopping criteria is reached.

MSB LSB

101101

$45_{10} = 101101_2$

Exercise : ① Convert 99_{10} to Binary

② $25_{10} =$ ——— (2)

⇒ 0.85 to Binary (4 point precision)

$0.85 \times 2 = 1.7$	1	MSB
$0.7 \times 2 = 1.4$	1	↓
$0.4 \times 2 = 0.8$	0	↓
$0.8 \times 2 = 1.6$	1	LSB

Carry

4 point precision is reached.

$\therefore 0.85_{10} = 0.1101_2$

Exercise :-

① 25.35_{10} to Binary ② 0.65_{10} to Binary

③ 0.9375_{10} to Binary ④ 35.62_{10} Binary.

For fractional, provide upto 4 point precision.

Decimal to octal conversion:

For, whole numbers,

1. To convert decimal to octal, the decimal number has to be divided by 8 repeatedly until quotient becomes 0. Reminder lies between 0-7.
2. The reminders generated forms the octal number (from MSD to LSD).

For fractional numbers,

- The fractional part of the decimal number should be multiplied by 8 repeatedly and the carry is considered in the integral part.

- Process is repeated until desired accuracy is reached. (precision)
- Reading the carry downwards gives the octal number.

Eg. 359_{10} to octal.

```
      44          5          0
8 ) 359     8 ) 44     8 ) 5
    352         40         0
    ---          4         ---
     7          ---         5
    ---                    ---
    LSB                    MSB
```

MSB LSB

547_8

Exercise : ① 0.16_{10} ② $89H.H5_{10}$.

⇒ Decimal to hexadecimal conversion:-

Same as decimal to binary/octal, only change is while dividing the integer part should divide by 16 and while multiplying the fractional part multiply by 16.

Ex : $H23_{10}$ to Hexadecimal.

```
      26          1          0
16 ) 423    16 ) 26    16 ) 1
    H16         16         0
    ---         ---        ---
     07         10 (A)      1
    ---         ---        ---
    LSB                    MSB
```

$1A7_{16}$

Exercise : ① $89H_{10}$ to hexadecimal
② $0.H5_{10}$ to hexadecimal
③ 37.033_{10} to hexadecimal.

⇒ Octal to Binary Conversion

When converting from octal to binary, change each octal digit to its 3 bit binary equivalent

Octal digit	3 bit binary Equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Ex: Convert $527H_8$ to binary

5 2 7 H octal

101 010 111 100 Binary equivalent

$527H_8 = 101010111100_2$

Exercise: 73.42_8 , $67H2.25_8$

⇒ Binary to octal conversion:-

When converting from binary to octal, group the bits in threes starting at the binary point. Then convert each group of 3 bits to its octal equivalent. 0's can be added to the left if necessary.

Eg: 1011100110_2 to octal

001 011 100 110 $1011100110_2 = 1346_8$

1 3 4 6

Exercise: 0.1011_2 to octal ③ 11.10001_2 to octal.

⇒ Hexadecimal to binary conversion: When converting from hexadecimal to binary, each hexadecimal digit is converted to its 4 bit binary equivalent.

Hexadecimal digit	4 bit binary equivalent
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Eg: $A2FH_{16}$ to binary

A 2 F H

Binary equivalent 1010 0010 1111 0100.

$A2FH_{16} = 1010001011110100_2$

Exercise: $9AF.EDB_{16}$ to binary

⇒ Binary to hexadecimal conversion:

When converting from binary to hexadecimal, group the bits in fours starting at the binary point. Then convert each group of four bits to its hexadecimal equivalent. 0's can be added if necessary to the left.

1) Convert 111111000101101000_2

$0011\ 1111\ 0001\ 0110\ 1000_2$

3 F 1 6 8 $_{16}$.

Signed binary representation:-

1. Sign-magnitude form:

The left most bit in a signed number is the sign bit. This is sign-magnitude form.

0 for positive
1 for negative

Eg

+25 ⓪0011001 (8 bit signed 25)
-25 ①0011001 (8 bit signed -25)

Sign bit

2. 1's complement form:-

In the 1's complement form, a negative number is the 1's complement of the corresponding positive number.

-39 as 8 bit binary number.

① First write 8 bit number of +39

00100111

② In 1's complement 00100111 becomes 11011000 which is -39.

∴ -39 = 11011000 in 1's complement.

3. 2's complement form:-

In 2's complement form, a negative number is 2's complement of corresponding +ve number.

-39.

① First write 8 bit number of +39

00100111

② Find 1's complement : 11011000

③ Find 2's complement : 11011001.

∴ -39 = 11011001 in 2's complement.

Exercise: Represent -399 in all the signed magnitude forms.

Codes

1. The ASCII Code:

At one time, manufacturers used their own alphanumeric codes which led to confusions. Eventually, industry settled on an input-output code known as the American Standard Code for Information Interchange (ASCII) which is an alphanumeric code (letters, numbers and other symbols). It allows to standardize computer hardware such as keyboard, printers and video displays.

ASCII code is a 7-bit code,

$X_6 X_5 X_4 X_3 X_2 X_1 X_0$ (X is a 0 or 1)

$X_3X_2X_1X_0$ \ $X_6X_5X_4$	010	011	100	101	110	111
0000	SP(space)	0	@	P	`	p
0001	!	1	A	Q	a	q
0010	"	2	B	R	b	r
0011	#	3	C	S	c	s
0100	$	4	D	T	d	t
0101	%	5	E	U	e	u
0110	&	6	F	V	f	v
0111	'	7	G	W	g	w
1000	(	8	H	X	h	x
1001	)	9	I	Y	i	y
1010	*	:	J	Z	j	z
1011	+	;	K	[	k	{
1100	,	<	L	\	l	|
1101	-	=	M	]	m	}
1110	.	>	N	^	n	~
1111	/	?	O	_	o	

Ex: Capital A.

$X_6X_5X_4 = 100$ $X_3X_2X_1X_0 = 0001$ $= 1000001$

Small a

$X_6X_5X_4 = 110$ $X_3X_2X_1X_0 = 0001$ $= 1100001$

2. BCD (Binary Coded Decimal)

A binary coded decimal is a type of binary representation for decimal values where each digit is represented by a 4 bits binary.

Decimal digits : 0 to 9.

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

Ex: 69 in BCD

6 9

0110 1001 = $01101001_{(BCD)}$

3. The Excess-3 Code

To convert any decimal number into its excess-3 code, add 3 to each decimal digit and then convert the sum to a BCD number.

Ex: Convert 12 to Excess-3 code.

$\begin{array}{r} 1 \\ +3 \\ \hline 4 \end{array}$ $\begin{array}{r} 2 \\ +3 \\ \hline 5 \end{array}$ → Convert 45 to BCD

0100 0101 = 01000101 (Excess

Excess-3 helps in BCD arithmetic as 9's compleme

4. The Gray Code

1. Binary to Gray converter.

$B_3 B_2 B_1 B_0$ be the binary bits.

$$G_3 = B_3$$
$$G_2 = B_3 \oplus B_2$$
$$G_1 = B_2 \oplus B_1$$
$$G_0 = B_1 \oplus B_0.$$

2. Gray to binary converter.

$$B_3 = G_3$$
$$B_2 = G_3 \oplus G_2$$
$$B_1 = B_2 \oplus G_1$$
$$B_0 = B_1 \oplus G_0 \quad (B_1 \oplus G_0)$$

Ex. $B_3 B_2 B_1 B_0 = 1011$

$G_3 = 1$

$G_2 = 1 \oplus 0 = 1$

$G_1 = 0 \oplus 1 = 1$

$G_0 = 1 \oplus 1 = 0$

$\therefore G_3 G_2 G_1 G_0 = 1110$

$G_3 G_2 G_1 G_0 = 1110$

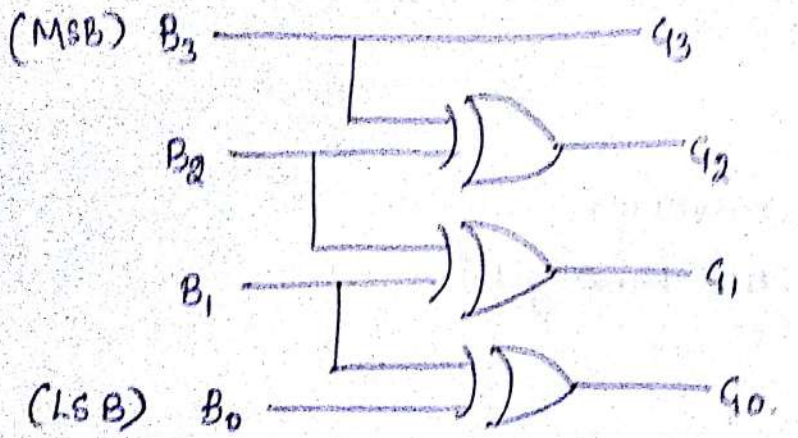
$B_3 = 1$

$B_2 = 1 \oplus 1 = 0$

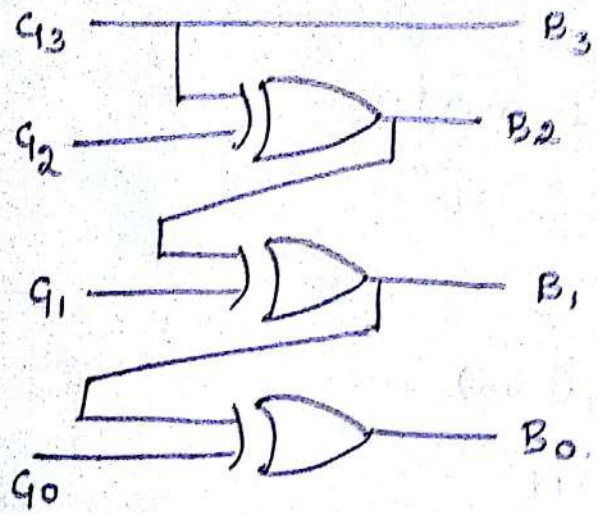
$B_1 = 0 \oplus 1 = 1$

$B_0 = 1 \oplus 0 = 1$

$\therefore B_3 B_2 B_1 B_0 = 1011$



Binary to Gray Converter.



Gray to binary Converter.

Arithmatic operations

1. Binary Addition:

$0+0=0$

$0+1=1$

$1+0=1$

$1+1=10$

Ex. Add 11100 and 11010.

	11100
+	11010
	110110

We need to consider either 8bit or 16 bit binary for arithmatic operations.

Ex. 0101 0111
+ 0011 0101
1000 1100

Binary subtraction

0 − 0 = 0
1 − 0 = 1
1 − 1 = 0
Borrow (1) 0 − 1 = 1

It is difficult to add or subtract signed binary numbers. So we go for 2's complement form of addition and subtraction of binary numbers. (signed).

1. Addition (using 2's complement)

Case 1: Both numbers are positive.

Convert both numbers to 8 bit (or 16 bit) binary and use binary addition.

Ex Add +83 and +16.

2	83
2	41 − 1
2	20 − 1
2	10 − 0
2	5 − 0
2	2 − 1
2	1 − 0
	0 − 1

+83 = $(01010011)_2$

2	16
2	8 − 0
2	4 − 0
2	2 − 0
2	1 − 0
	0 − 1

+16 = $(00010000)_2$

Add 01010011
+ 00010000
01100011

= 99(10)

Case 2: Larger positive number, smaller negative number.

1. Convert larger positive number to 8 bit (or 16 bit) binary.
2. Convert smaller negative number to 2's complement form (8 bit or 16 bit).
3. Add both
4. Discard the final carry generated.

Result will be in binary.

Ex. Add +125 and -68.

2 | 125	
2 | 62 - 1	
2 | 31 - 0	
2 | 15 - 1	
2 | 7 - 1	
2 | 3 - 1	
2 | 1 - 1	
0 - 1	

$+125 = (01111101)_2$

2 | 68	
2 | 34 - 0	
2 | 17 - 0	
2 | 8 - 1	
2 | 4 - 0	
2 | 2 - 0	
2 | 1 - 0	
0 - 1	

$+68 = (01000100)_2$

Convert +68 to 2's complement

$$\begin{array}{r} 10111011 \\ + \quad\quad 1 \\ \hline 10111100 \end{array}$$

1's complement

2's complement

$$\begin{array}{rr} \text{Add} & \overset{1}{0}\overset{1}{1}\overset{1}{1}\overset{1}{1}\overset{1}{1}101 \\ & + \; 10111100 \\ \hline \text{discard} \leftarrow & (1)\,00111001 \end{array} \equiv (57)_{10}$$

Case 3:- Larger negative number, smaller +ve number.

1. Convert larger negative number to 2's complement form. (8 bit or 16 bit)
2. Convert smaller +ve number to 8 or 16 bit binary
3. Add both.
4. The result will be in 2's complement form.

To show the result in decimal convert the result to 2's complement and prefix with -ve sign.

Add.

Ex. −115 and +37

2	115
2	57 −1
2	28 −1
2	14 −0
2	7 −0
2	3 −1
2	1 −1
	0 −1

+115 = 01110011

2	37
2	18 −1
2	9 −0
2	4 −1
2	2 −0
2	1 −0
	0 −1

$+37 = (00100101)_2$

+115 in 2's complement

```
  10001100   1's complement.
+        1
----------
  10001101   2's complement
```

Add

```
  00100101
  10001101
----------
  10110010   2's complement  (−78)
```

Proof: (To convert to decimal)

```
  01001101
+        1
----------
  01001110  = 78
```

Prefix with −ve sign (−78).

Case 4: Both are negative.

1) Convert both numbers to 2's complement form and add.
2) Result will be in 2's complement.
3) Discard the final carry generated.

To show the result in decimal convert the result to 2's complement and prefix with −ve sign.

Ex. Add -43 and -78.

+43

2	43
2	21 - 1
2	10 - 1
2	5 - 0
2	2 - 1
2	1 - 0
	0 - 1

$+43 = (00101011)_2$

+78

2	78
2	39 - 0
2	19 - 1
2	9 - 1
2	4 - 1
2	2 - 0
2	1 - 0
	0 - 1

$+78 = (01001110)_2$

+43 in 2's complement

11010100 is complement

$$\begin{array}{r} 11010100 \\ + \quad\quad 1 \\ \hline 11010101 \end{array}$$ 2's complement

+78 in 2's complement

10110001 is complement

$$\begin{array}{r} 10110001 \\ + \quad\quad 1 \\ \hline 10110010 \end{array}$$ 2's complement

Add

$$\begin{array}{r} {\scriptstyle 1\,1\,1} \\ 11010101 \\ 10110010 \\ \hline \text{discard} \leftarrow (1)\,10000111 \end{array}$$ 2's complement. (-121)

Proof (to convert to decimal)

$$\begin{array}{r} 01111000 \\ + \quad\quad 1 \\ \hline 01111001 \end{array}$$ 2's complement. = +121

Prefix with -ve sign = -121

Subtraction.

case 1.

Ex. 1. Subtract 212 from 81

$$\begin{array}{r} 81 \\ -212 \\ \hline \end{array}$$

This is similar to case 3 of addition

(smaller +ve number
larger negative number).

Case 2:

Ex. Subtract 20 from 40.

$$\begin{array}{r} 40 \\ -20 \\ \hline \end{array}$$

This is similar to case 2 of addition (Large positive number small negative number)

Case 3:

Ex: Subtract −30 from 39

$$\begin{array}{r} 39 \\ -(-30) \\ \hline \end{array}$$

Negative (−) Negative becomes +ve

$$\therefore \begin{array}{r} 39 \\ +30 \\ \hline \end{array}$$

Calculate using case 1 of addition (Positive numbers addition).

Case 4: Subtract 39 from −30.

$$\begin{array}{r} -30 \\ -39 \\ \hline \end{array}$$

This is similar to case 4 of addition (Both numbers −ve).

Combinational circuits

Combinational circuits are the circuits where the output is a pure function of the present inputs only. Logic gates are the building blocks of combinational circuits.

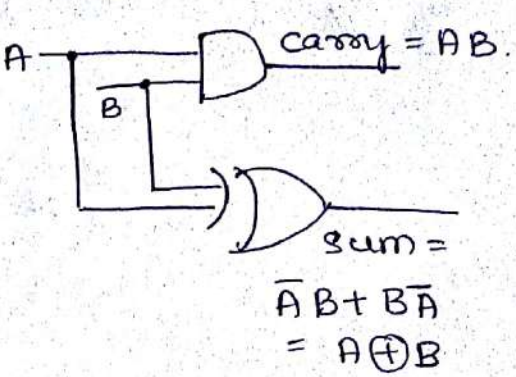
Half adder

Half adder is an arithmetic, combinational circuit. When we add two binary numbers, we start with the least significant column. We have to add two bits with the possibility of a carry.

Truth table

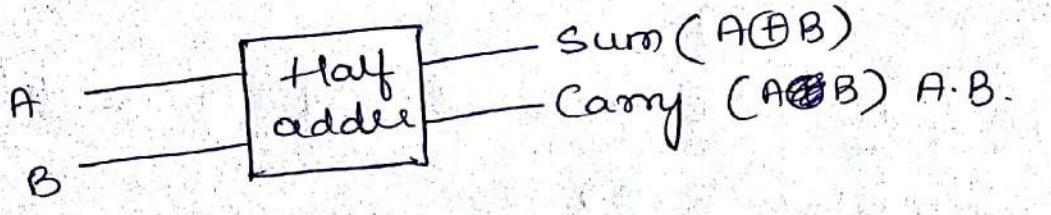
A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Logic diagram (Circuit)



The output of XOR gate is called the sum.
The output of AND gate is the carry.

Block diagram



Full adder

For higher order columns, we have to use a full-adder, a logic circuit that can add 3 bits at a time. The 3rd bit is the carry from a lower column. It is also an arithematic, combinational circuit.

Truth table.

A	B	C_{in}	Cout (Carry)	Sum
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

Carry.

$C / \bar{C}_{in}$ \ AB	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}_{in}$	0	0	1	0
C_{in}	0	1	1	1

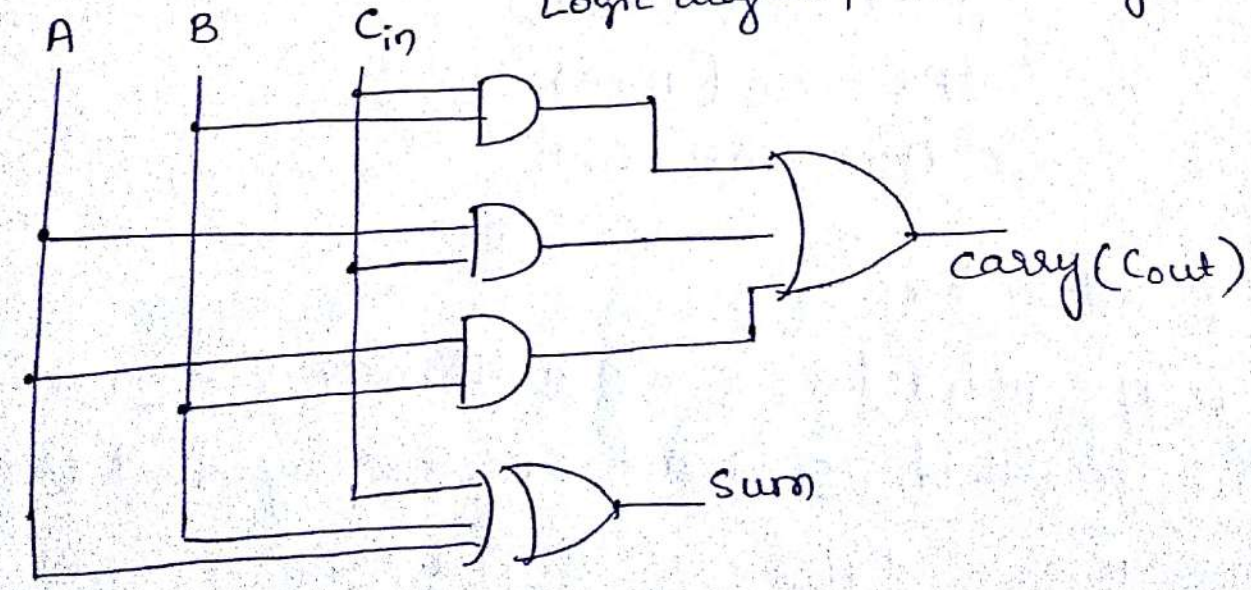
$Carry = AB + BC_{in} + AC_{in}$

Sum

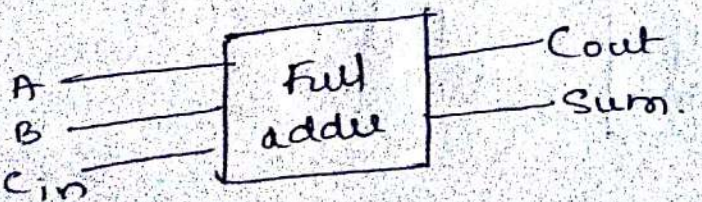
$C / \bar{C}$ \ AB	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}$	0	1	0	1
C	1	0	1	0

$Sum = A \oplus B \oplus C.$

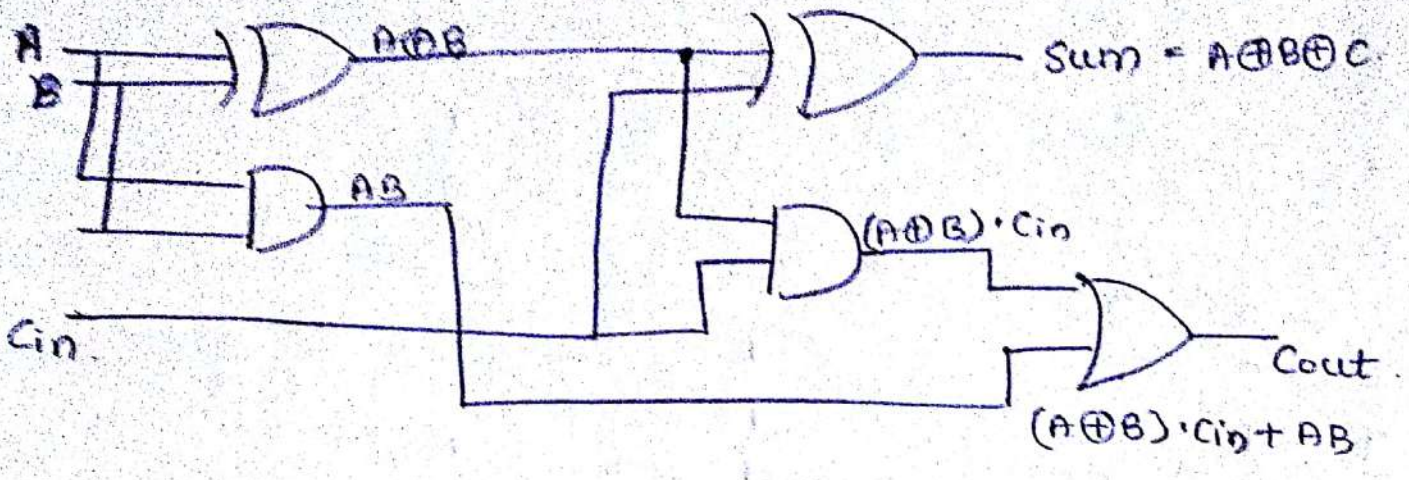
Logic diagram / circuit diagram



Block diagram



Full adder using two half adders :-

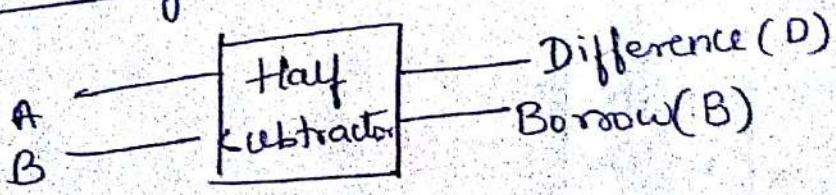


$$\begin{aligned}
Cout &= (A \oplus B)Cin + AB \\
&= (\bar{A}B + \bar{B}A)Cin + AB \\
&= \bar{A}BCin + \bar{B}ACin + AB \\
&= B(\bar{A}Cin + A) + \bar{B}A\,cin \\
&= B(A + Cin) + \bar{B}A\,cin \\
&= BA + BCin + \bar{B}ACin \\
&= AB + Cin(B + \bar{B}A) \\
&= AB + cin(A + B) \\
Cout &= AB + CinA + CinB
\end{aligned}$$

Half Subtractor

Half subtractor is an arithmetic, combinational circuit used to subtract one binary digit from another to give difference output and a Borrow output

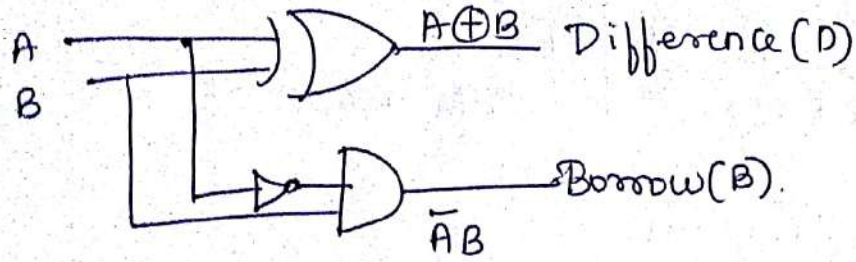
Block diagram



Truth table

A	B	Difference (D)	Borrow (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

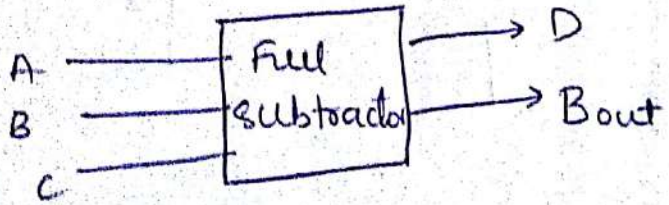
Logic / circuit diagram



Full Subtractor

Full subtractor is an arithmetic, combinational circuit that performs subtraction of two bits taking into account borrow of the previous adjacent lower bit.

Block diagram



Truth table

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

1 1 1 1 1

K-map

Difference
~~Borrow~~ = $A \oplus B \oplus C$

borrow
~~Difference~~ = $\bar{A}B + \bar{A}C + BC$.

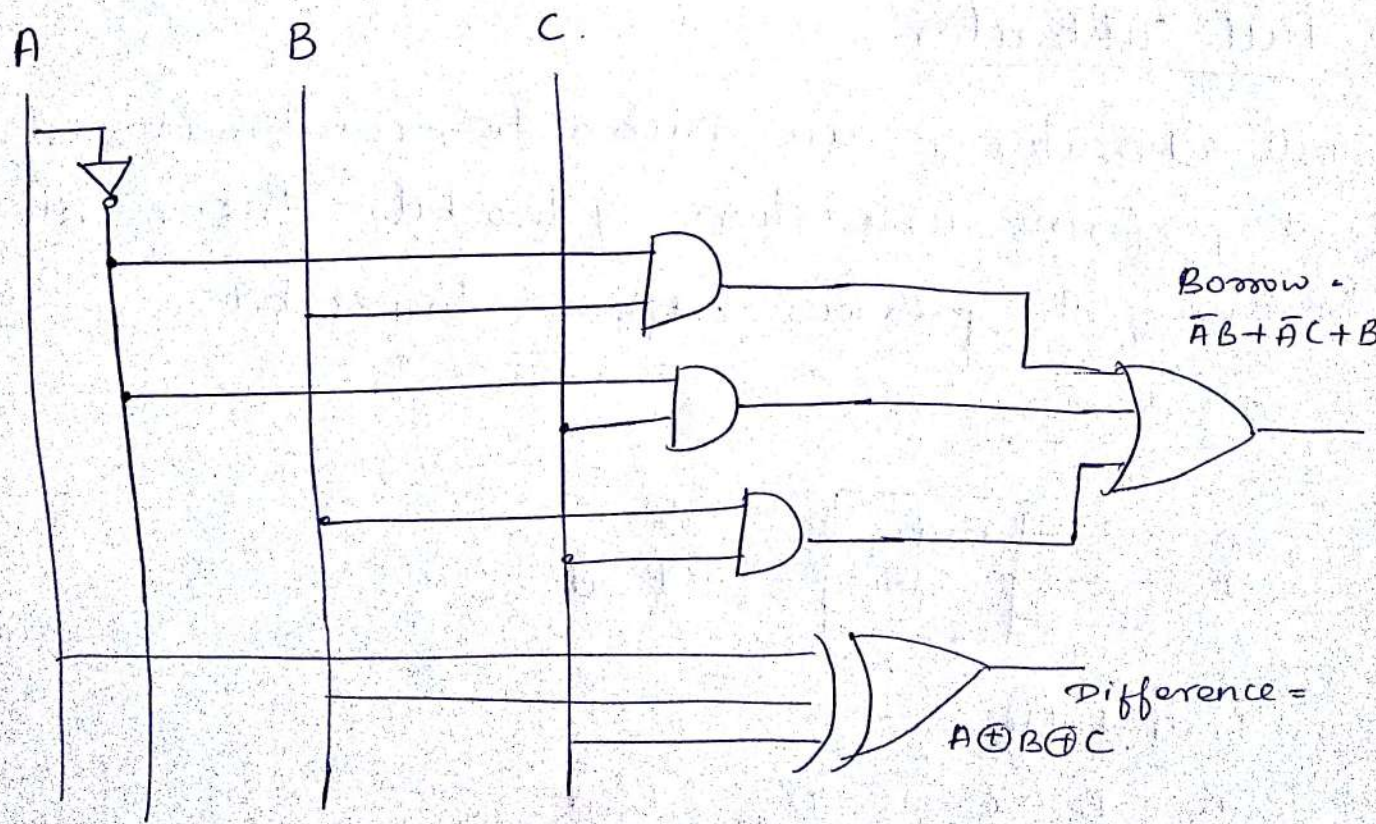
AB \ C	$\bar{C}$	C
$\bar{A}\bar{B}$	0	1
$\bar{A}B$	1	1
AB	0	1
$A\bar{B}$	0	0

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

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

Difference = $A \oplus B \oplus C$.

Circuit / Logic diagram



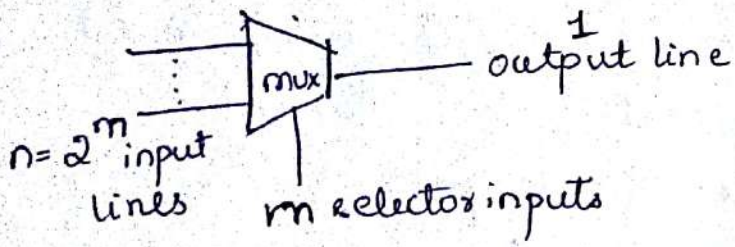
Multiplexer

A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.

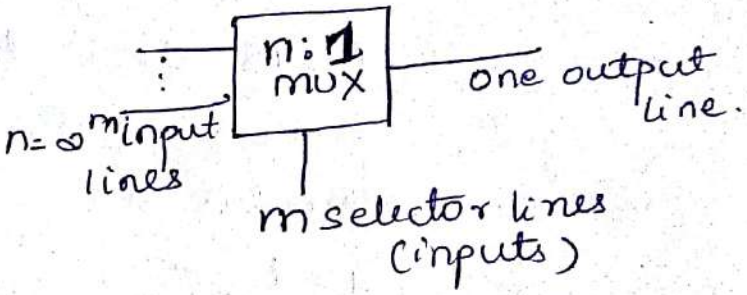
The selection of a particular input line is controlled by a set of selection lines called selector inputs.

Normally, there are 2^m input lines and m selector inputs.

Symbol for multiplexer:



Block diagram



2:1 multiplexer

2:1 multiplexer connects one of two 1 bit sources to a common destination.

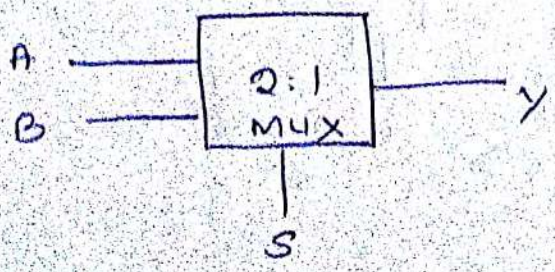
Truth table.

S	A	B
0	✓	
1		✓

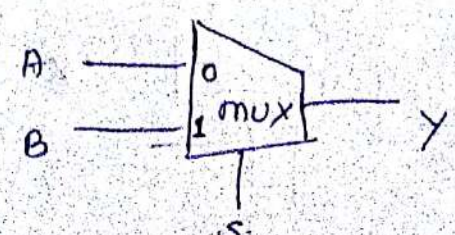
When S=0, A is the output

When S=1 B is the output.

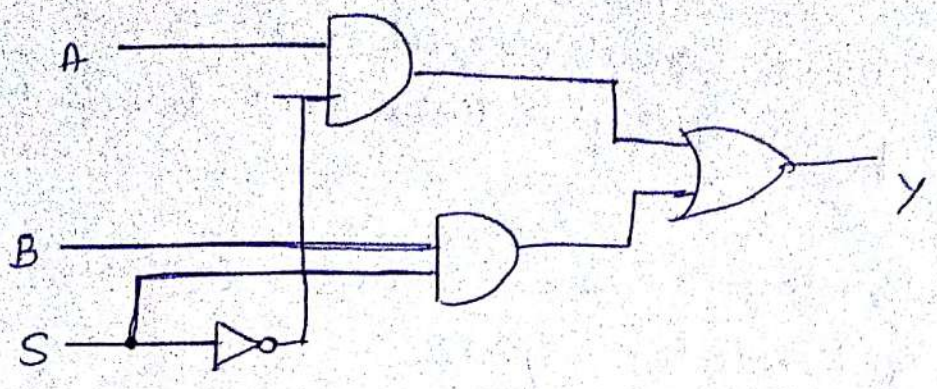
Block diagram



Symbol



Logic diagram / Circuit diagram



4:1 multiplexer

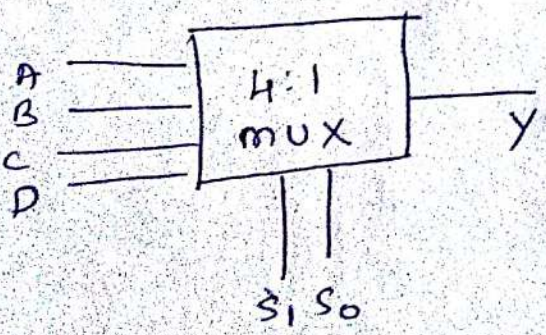
There are 4 inputs, 1 output. Each of the four inputs, A through D, is applied to one ~~output~~ input of an AND gate. Selection inputs S_1 and S_0 are decoded to select a particular AND gate. The outputs of the AND gates are applied to a single OR gate that provides one-line output.

Truth table

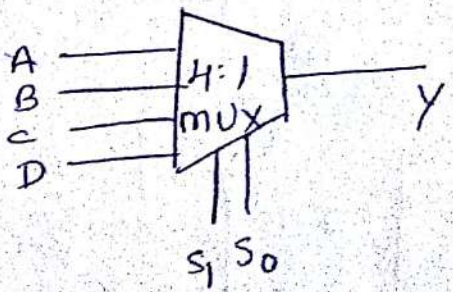
S_1	S_0	Y
0	0	A
0	1	B
1	0	C
1	1	D

Ex If $S_1S_0 = 10$.
The output associated = C.

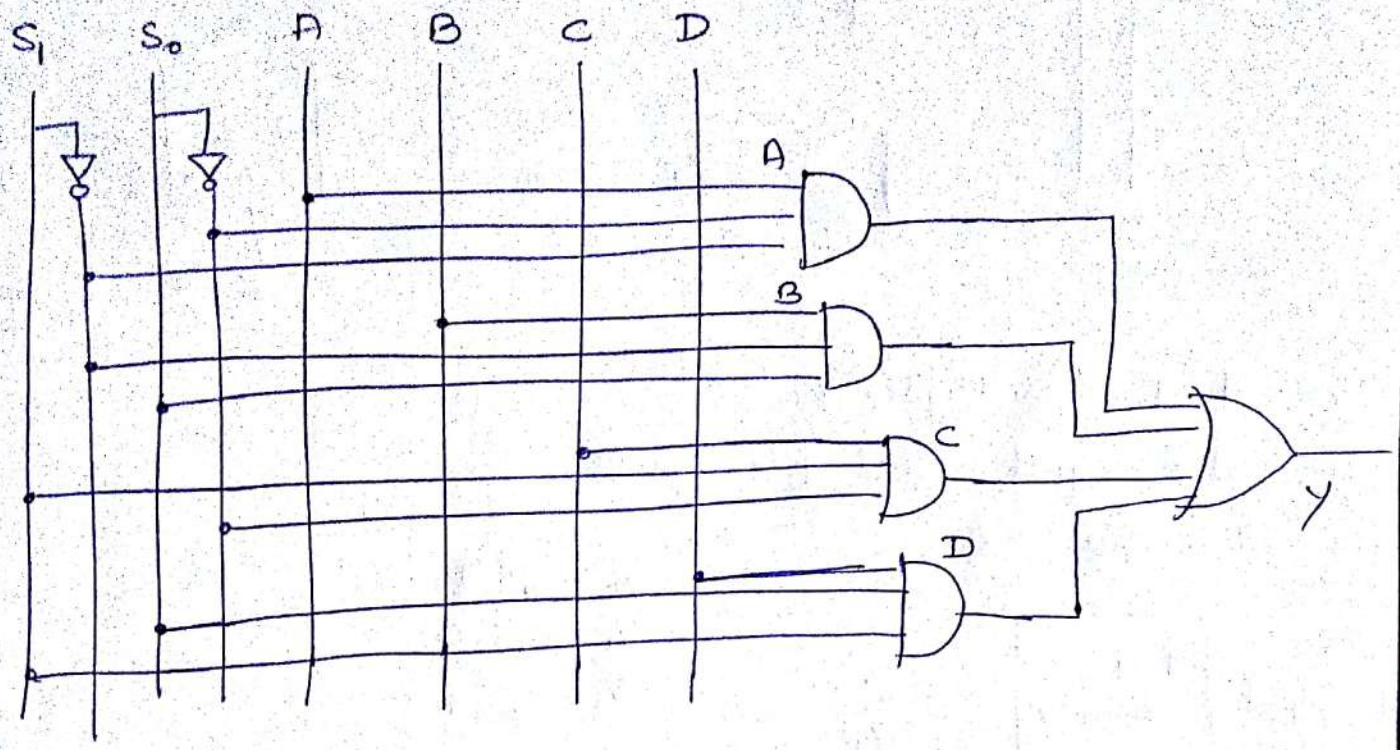
Block diagram



Symbol



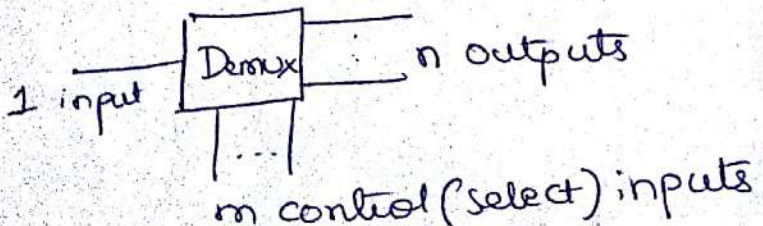
Logic diagram / circuit



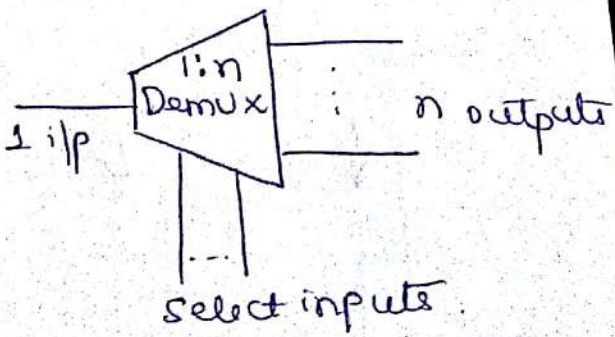
Demultiplexer

Demultiplexer is a combinational circuit with one input and many outputs. The circuit has 1 input signal, m select signals (inputs) and n output signals.

Block diagram



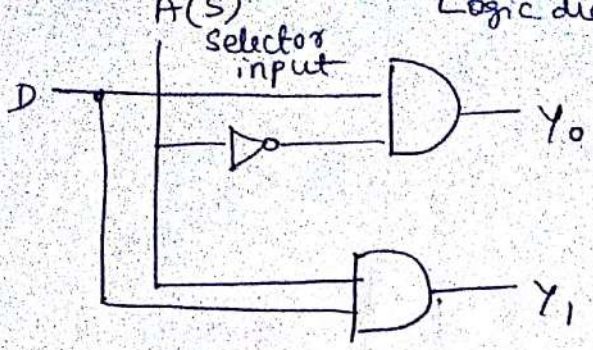
Symbol



1:2 demultiplexer

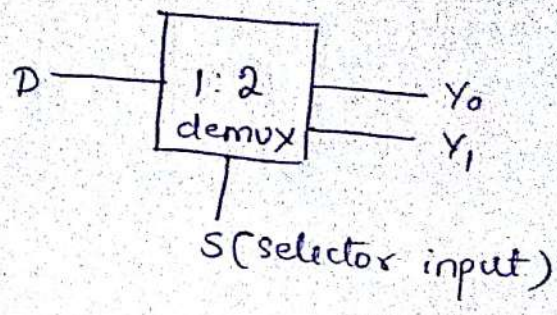
Truth table

Selection Input (S)	Y_1	Y_0 (output)
0	0	D
1	D	0



Logic diagram/ circuit diagram

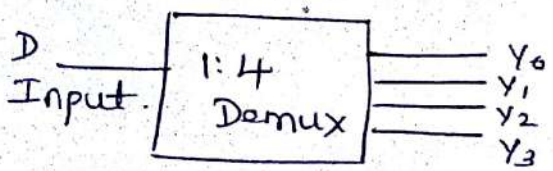
Block diagram



1:4 demultiplexer

One input and 4 outputs.

Block diagram



$Y_0 = \overline{S_1}\,\overline{S_0}\,D$

$Y_1 = \overline{S_1}\,S_0\,D$

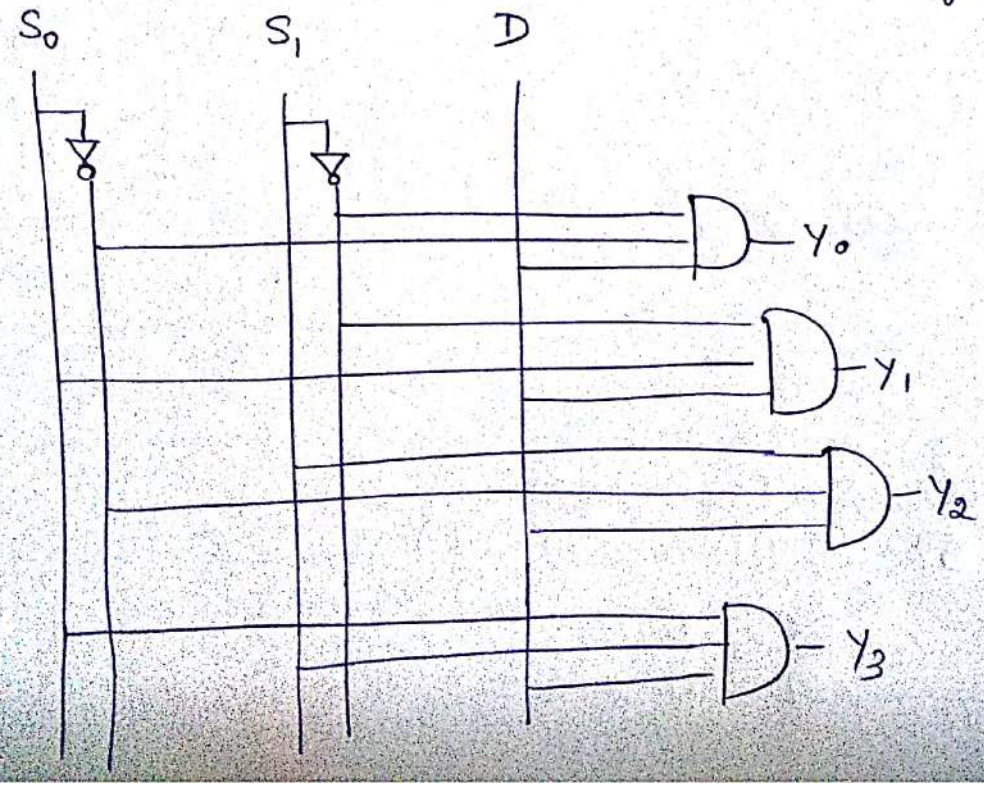
$Y_2 = S_1\,\overline{S_0}\,D$

$Y_3 = S_1\,S_0\,D$

Truth table

S_1 S_0	Y_0	Y_1	Y_2	Y_3
0 0	D	0	0	0
0 1	0	D	0	0
1 0	0	0	D	0
1 1	0	0	0	D

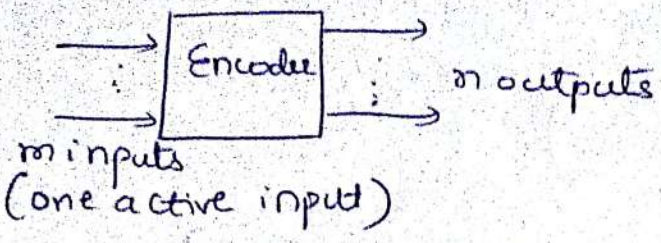
circuit/Logic diagram.



Encoders

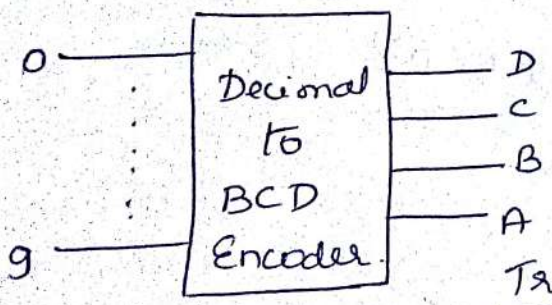
Encoder is a combinational circuit that converts numbers or symbols to coded format. Eg Conversion from decimal to BCD, Conversion from octal to binary. It contains one active input signal.

Block diagram



Decimal to BCD Encoder

Decimal number system consists of digits from 0-9. Every digit can be converted to equivalent BCD using this encoder. By making one input as active signal encoding can be done.

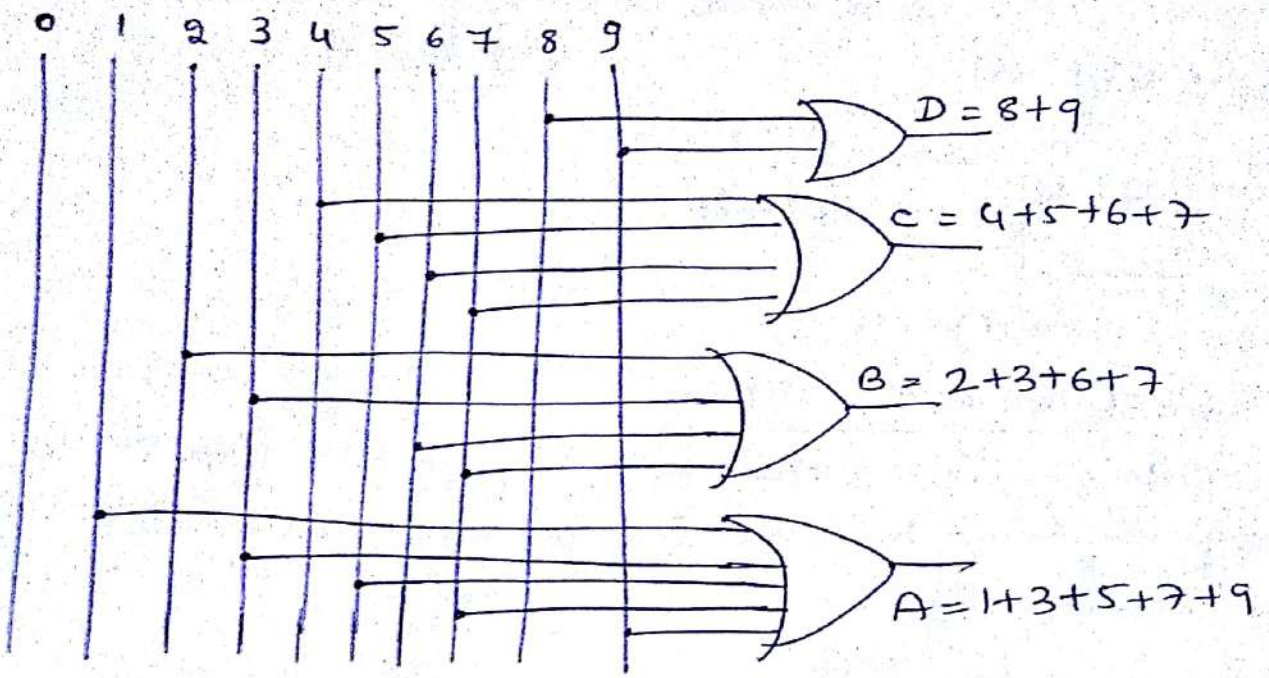


Truth table

Decimal (High)	BCD outputs D C B A
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Ex If input signal is high for 1
BCD output = 0001
all others will be of low signal.

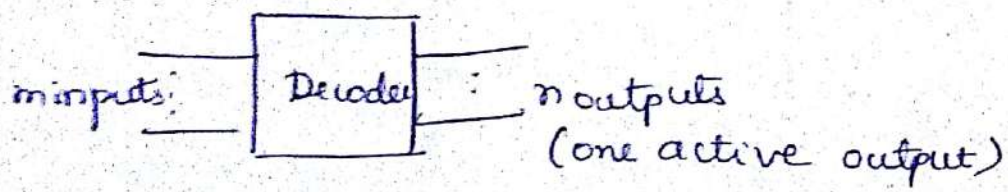
Logic diagram / circuit



Ex If 4 is high, all other will be low, only C will become high so DCBA = 0100 (BCD)

Decoders

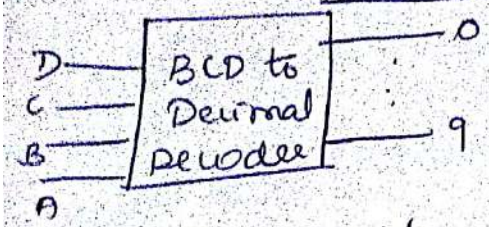
Decoder is a combinational circuit that is used to translate (decode) or convert coded formal to numbers / symbols. Eg Conversion from BCD to decimal, conversion from binary to octal. It contains one active output signal.



BCD to Decimal Decoder

4 bits of BCD - input
1 digit decimal - output
(one active signal)

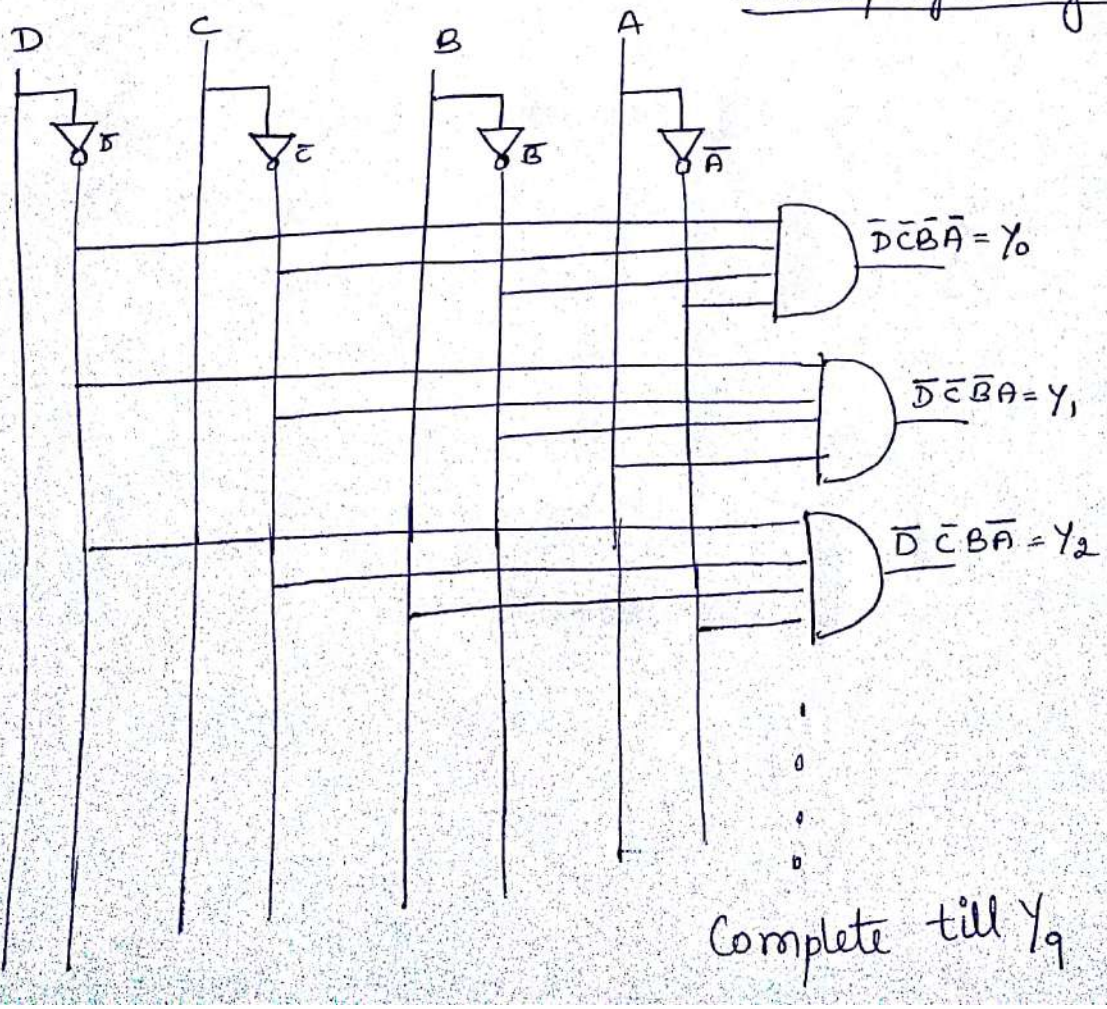
Block diagram



Truth table

Inputs				Output	
D	C	B	A	Decimal	Y
0	0	0	0	$\bar{D}\bar{C}\bar{B}\bar{A}$	0
0	0	0	1	$\bar{D}\bar{C}\bar{B}A$	1
0	0	1	0	$\bar{D}\bar{C}B\bar{A}$	2
0	0	1	1	$\bar{D}\bar{C}BA$	3
0	1	0	0	$\bar{D}C\bar{B}\bar{A}$	4
0	1	0	1	$\bar{D}C\bar{B}A$	5
0	1	1	0	$\bar{D}CB\bar{A}$	6
0	1	1	1	$\bar{D}CBA$	7
1	0	0	0	~~$\bar{D}\bar{C}\bar{B}\bar{A}$~~ $D\bar{C}\bar{B}\bar{A}$	8
1	0	0	1	$D\bar{C}\bar{B}A$	9

circuit/logic diagram



Complete till Y_9

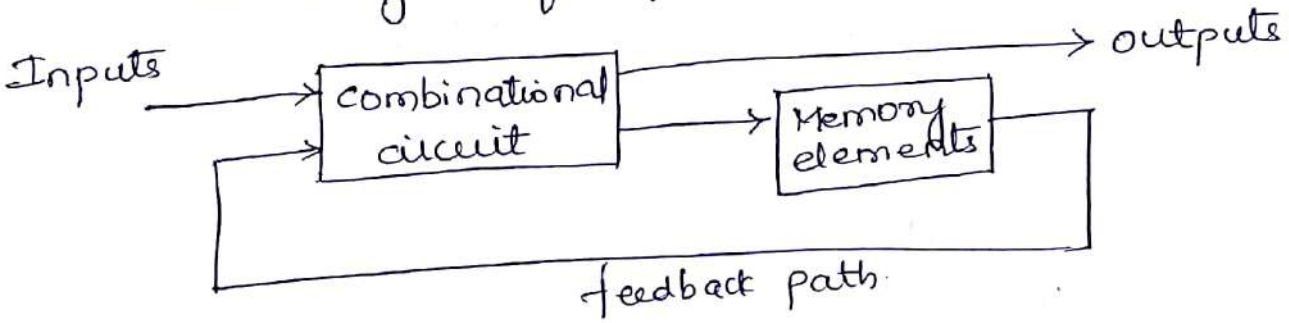
Module - 4 continued.

Sequential circuits

A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs.

Sequential circuits are the circuits where present outputs depends on previous output as well, including present inputs. They employ storage elements in addition (memory) to logic gates.

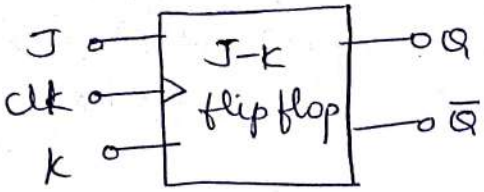
Block diagram of sequential circuit

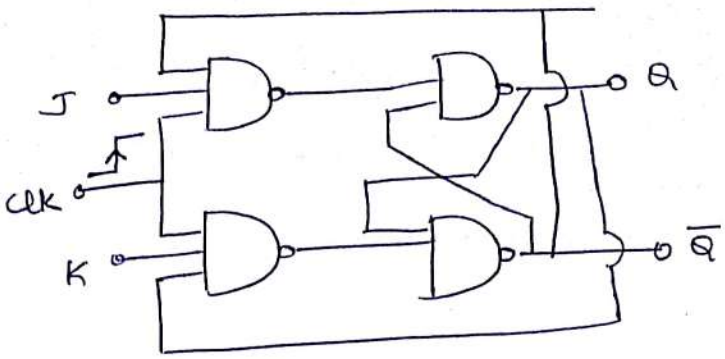


A clock signal is a particular type of signal that oscillates between a high and a low state. The circuits which are clocked are called synchronous Sequential circuits.

The storage elements (memory) used in clocked Sequential circuits are called flip flops. A flip flop is a binary storage device capable of storing one bit of information. In a stable state, the output of a flip flop is either 0 or 1.

JK flip flop





JK flip flop operates when clock is high (leading edge).

Truth table

clk	J	K	Q_{n+1}	
0	x	x	Q_n (memory)	Nochange.
1	0	0	Q_n (memory)	
1	0	1	0	
1	1	0	1	
1	1	1	Toggle ($\bar{Q}_n$)	

If clock = 0 (low) whatever the values of J & K is there will be no change. Q_n (memory) will be the output.

If clock = 1 and J = 0 K = 0 there will be no change Q_n (memory) will be the output.

If clock = 1, J = 0, K = 1, $Q_{n+1} = 0$ $\bar{Q}_{n+1} = 1$

If clock = 1, J = 1, K = 0 $Q_{n+1} = 1$ $\bar{Q}_{n+1} = 0$

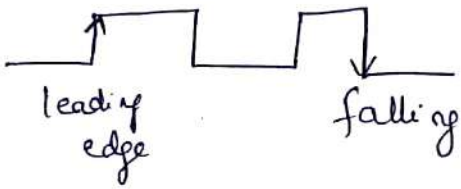
If clock = 1, J = 1, K = 1, Q_{n+1} Toggles with Q_n & $\bar{Q}_n$.

Counters

Counter is a sequential circuit which stores the number of times a particular event or process has occured, often in relationship to a clock signal.

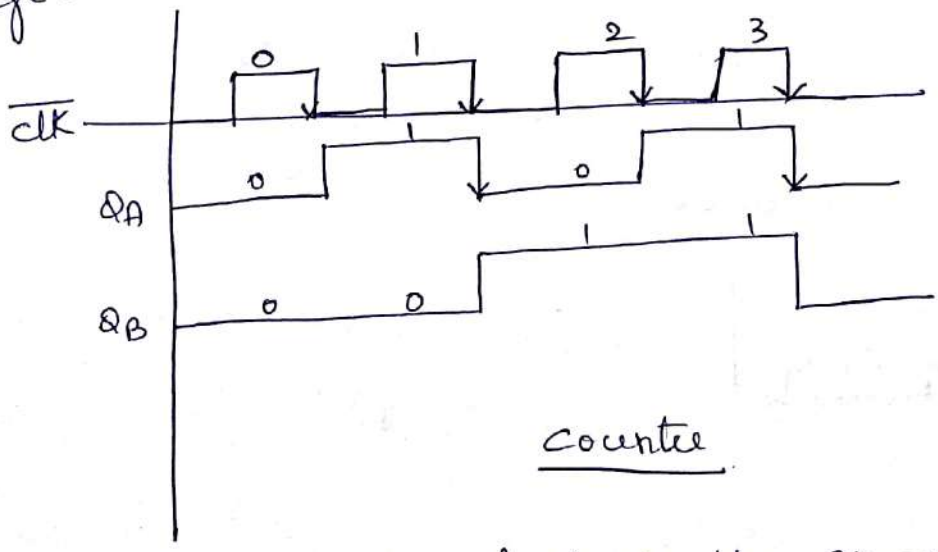
Asynchronous (ripple counter) – When the output of a flip flop is used as the clock input for the next flip flop, it is called asynchronous or ripple counter.

An electronic oscillator called clock generates a sequence of repetitive pulses called clock signal which is distributed to all the memory elements in the circuit. It has a leading edge and falling edge.



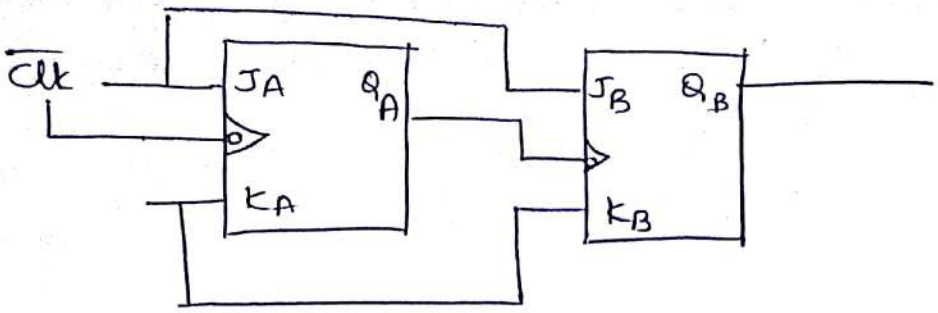
Positive clock - It is the one where the leading edge is the active edge.

Negative clock - Here the falling edge is the active edge.



Counter

Counter uses negative clock as the signal is active on falling edge of the clock.



clk	Q_B	Q_A
0	0	0
1	0	1
2	1	0
3	1	1

To counter from 0-3 (0, 1, 2, 3) $= 2^{(2)}$ 2 flip flops are required.

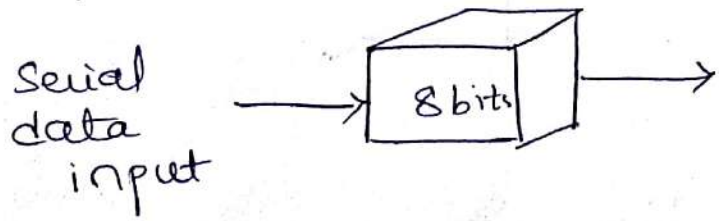
Registers

A register is a group of flip-flops that can be used to store a binary number. There must be one flip flop for each bit in the binary number. For instance, a register used to store an 8-bit binary number must have eight flip-flops.

Types of registers

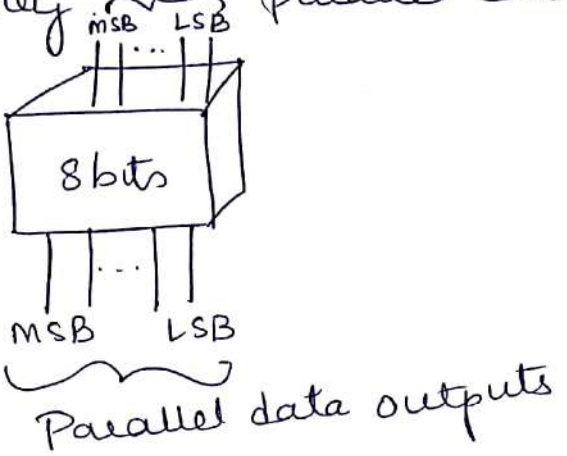
1. Serial in Serial out

The register is loaded with serial data, one bit at a time, with the stored data being available at the output in serial form only.



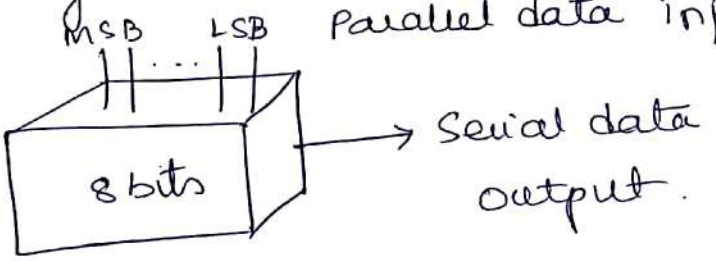
2. Parallel in parallel out

The register is loaded with parallel data, and the stored data being available at the output in parallel form only.



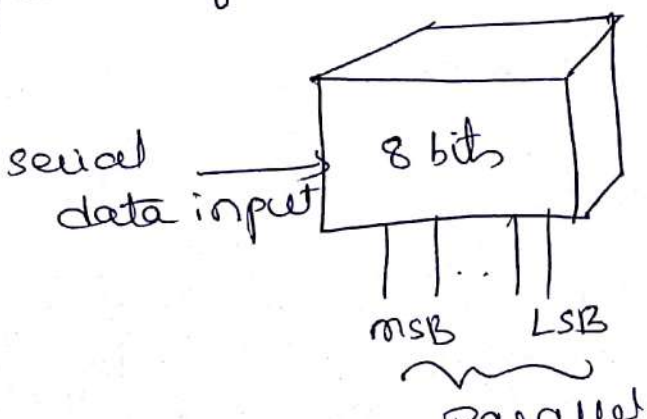
3. Parallel in serial out

The register is loaded with parallel data, and the stored data being available at the output in serial form.



4. Serial in parallel out

The register is loaded with serial input, one bit at a time and stored data being available at the output in parallel form.



Module 5	**INPUT/ OUTPUT AND MEMORY ORGANIZATION OF COMPUTER** I/O Devices, Memory Systems, Interrupts and instruction cycle, Addressing modes

I/O devices :

A **peripheral device** connects to a computer system to add functionality. Examples are a mouse, keyboard, monitor, printer and scanner. Input device: Input device is a peripheral hardware device that sends data to a computer, allowing you to interact with and control it.

Examples:

Keyboard

Mouse

Joy Stick

Light pen

Track Ball

Scanner

Graphic Tablet

Microphone

Magnetic Ink Card Reader(MICR)

Optical Character Reader(OCR)

Bar Code Reader

Optical Mark Reader(OMR)

Keyboard: Keyboard is the most common and very popular input device which helps to input data to the computer. The layout of the keyboard is like that of traditional typewriter, although there are some additional keys provided for performing additional functions.

Keyboards are of two sizes 84 keys or 101/102 keys, but now keyboards with 104 keys or 108 keys are also available.

The keys on the keyboard are as follows –

1.Typing Keys

These keys include the letter keys (A-Z) and digit keys (09) which generally give the same layout as that of typewriters.

2. Numeric Keypad

It is used to enter the numeric data or cursor movement. Generally, it consists of a set of 17

keys that are laid out in the same configuration used by most adding machines and calculators.

3.Function Keys

The twelve function keys are present on the keyboard which are arranged in a row at the top of the keyboard. Each function key has a unique meaning and is used for some specific purpose.

4.Control keys

These keys provide cursor and screen control. It includes four directional arrow keys. Control keys also include Home, End, Insert, Delete, Page Up, Page Down, Control(Ctrl), Alternate(Alt), Escape(Esc).

5.Special Purpose Keys

Keyboard also contains some special purpose keys such as Enter, Shift, Caps Lock, Num Lock, Space bar, Tab, and Print Screen.

Working of keyboard:

A keyboard has its own processor and circuitry that carries information to and from that processor. A large part of this circuitry makes up the key matrix. The key matrix is a grid of circuits underneath the keys. In all keyboards each circuit is broken at a point below each key. When you press a key, it presses a switch, completing the circuit and allowing a tiny amount of current to flow through. The mechanical action of the switch causes some vibration, which the processor filters out. If you press and hold a key, the processor recognizes it as the equivalent of pressing a key repeatedly.

When the processor finds a circuit that is closed, it compares the location of that circuit on the key matrix to the character map in its read-only memory (ROM). A character map is basically a comparison chart or lookup table. It tells the processor the position of each key in the matrix and what each keystroke or combination of keystrokes represents. For example, the character map lets the processor know that pressing the a key by itself corresponds to a small letter "a," but the Shift and a keys pressed together correspond to a capital "A."

Mouse :

A mouse is an input peripheral hardware device that a computer user pushes across a desk surface in order to point to a place on a display screen and to select one or more actions to take from that position.

Mouse actions:

Action	**Definition**
Click	Press and release the left mouse button, without moving the mouse.
Click-and-hold	Press and do not release the left mouse button.
Left-click	Same as **click**. Left-click clarifies the action when there might be confusion with **right-click**.

Action	Definition
Middle-click	Press and release the middle mouse button, without moving the mouse.
Right-click	Press and release the right mouse button, without moving the mouse.
Double-click	Press and release the left mouse button twice in rapid succession without moving the mouse.
Drag	Click-and-hold a mouse button, then move an object. For example, you can drag a window or an icon. The left and middle mouse buttons can perform drag actions.
Drag-and-drop	Click-and-hold a mouse button, then move an object. For example, you can drag-and-drop a window or an icon. Release the mouse button to place the object in a new location.
Grab	Point to an item that you can move, and click-and-hold on the mouse button. For example, you can grab the titlebar of a window, and then drag the window to a new location.

A mechanical mouse is a computer mouse that contains a metal or rubber ball on its under side. When the ball is rolled in any direction, sensors inside the mouse detect this motion and move the on-screen mouse pointer in the same direction. An optical mouse is a computer mouse which uses a light source, typically a light-emitting diode (LED), and a light detector, to detect movement relative to a surface.

Working of mechanical mouse:

The mechanical mouse contains a ball which partially protrudes from the bottom surface, so that the ball rotates as the device is moved, internally actuating two pick-up elements which convert the motion to its `x' and `y' components and sends signal to the computer that causes a mouse-responsive program to reposition a cursor on the display screen.

Working of optical mouse:

An optical mouse shines a bright light down onto the desk from an LED (light-emitting diode) mounted on the bottom of the mouse. The light bounces straight back up off the desk into a photocell(photoelectric cell), also mounted under the mouse, a short distance from the LED. The photocell has a lens in front of it that magnifies the reflected light, so the mouse can respond more precisely to the hand movements. As the mouse is pushed around your desk, the pattern of reflected light changes, and the chip inside the mouse uses this to figure out the movement.

JOYSTICK

A joystick is an input device consisting of a stick that pivots on a base and reports its angle or direction to the device it is **controlling**.

LIGHT PEN

A handheld pen-like photosensitive input device held to the display screen of a computer terminal for passing information to the computer.

TRACK BALL

A trackball is a pointing device consisting of a ball held by a socket containing sensors to detect a rotation of the ball about two axes—like an upside-down mouse with an exposed protruding ball.

SCANNER

A scanner is an input device that optically scans images, printed text, handwriting or an object and converts it to a digital image.

OUTPUT DEVICES

An output device is any peripheral that receives data from a computer, usually for display or projection.

Examples

3D Printer

Headphones

Monitor

Plotter

Printer (dot matrix printer, inkjet printer, and laser printer).

Projector

Speakers

Monitor

Alternatively referred to as a VDT (video display terminal) and VDU (video display unit), a monitor is an output device that displays video images and text. A monitor is made up of circuitry, a screen, a power supply, buttons to adjust screen settings, and casing that holds all of these components. The first computer monitors were comprised of a CRT (cathode ray tube) and a fluorescent screen. Today, all monitors are created using flat panel display technology, usually backlit with LEDs.

Printer

a. Impact printer
b. Non impact printer

Impact printers:

An impact printer makes contact with the paper. It usually forms the print image by pressing an inked ribbon against the paper using a hammer or pins. Following are some examples of impact printers.

Dot-Matrix Printers

The dot-matrix printer uses print heads containing from 9 to 24 pins. These pins produce patterns of dots on the paper to form the individual characters. The 24 pin dot-matrix printer produces more dots that a 9 pin dot-matrix printer, which results in much better quality and clearer characters. The general rule is: the more pins, the clearer the letters on the paper. The pins

strike the ribbon individually as the print mechanism moves across the entire print line in both directions, from left to right, then right to left, and so on. The user can produce a color output with a dot-matrix printer. Dot-matrix printers are inexpensive and typically print at speeds of 100-600 characters per second.

Daisy-wheel printers

Line printers

Chain printers

Non-impact printers

Non-impact printers do not use a striking device to produce characters on the paper and because these printers do not hammer against the paper they are much quieter. Following are some non-impacted printers.

Ink-jet printers

Ink-jet printers work in the same fashion as dot-matrix printers in the form images or characters with little dots. However, the dots are formed by tiny droplets of ink. Ink-jet printers form characters on paper by spraying ink from tiny nozzles through an electrical field that arranges the charged ink particles into characters at the rate of approximately 250 characters per second. The ink is absorbed into the paper and dries instantly. Various colors of ink can also be used.

Laser printers

A laser printer works like a photocopy machine. Laser printers produce images on paper by directing a laser beam at a mirror which bounces the beam onto a drum. The drum has a special coating on it to which toner (an ink powder) sticks. Using patterns of small dots, a laser beam conveys information from the computer to a positively charged drum to become neutralized. From all those areas of drum which become neutralized, the toner detaches. As the paper rolls by the drum, the toner is transferred to the paper printing the letters or other graphics on the paper. A hot roller bonds the toner to the paper.

Memory

The function of the memory is to store programs, data and results.

Primary memory

Primary memory is computer memory that is accessed directly by the CPU. This includes several types of memory, such as the processor cache and system ROM. In most cases, primary memory refers to system RAM. This memory is also known as main memory.

RAM (Random Access Memory)

RAM(Random Access Memory) is a type of Primary memory. It consists of one or more memory modules that temporarily store data while a computer is running. RAM is volatile memory, it is erased when the power is turned off. Therefore, each time you start up your computer, the operating system must be loaded from secondary memory (such as a hard drive) into the primary memory. Similarly, whenever you launch an application on your computer, it is loaded into RAM.

ROM(Read Only memory)

ROM stands for Read Only Memory.ROM can only be read by the processor. New data cannot be written into ROM. Data to be stored into ROM is written during the manufacturing phase itself. They contain data that does not need to be altered, like booting sequence of a computer or algorithmic tables for mathematical applications. ROM is slower and hence cheaper than RAM. It retains its data even when power is switched off, i.e. it is non-volatile. ROM cannot be altered the way RAM can be but technologies are available to program the ROMs –

Programmable Read-Only Memory (PROM)

This type of ROM can be re-programmed by using a special device called a PROM programmer. Generally, a PROM can only be changed or updated once.

Erasable Programmable Read-Only Memory (EPROM)

This type of ROM can have its contents erased by ultraviolet light and then reprogrammed by an RPROM programmer. This procedure can be carried out many times however , the constant erasing and rewriting will eventually render the chip useless.

Electrically Erasable Programmable Read-Only Memory (EEPROM)

This type of ROM works in a similar way to Flash memory in that it can its contents can be 'flashed' for erasure and then written to without having to remove the chip from its environment. EEPROMs are used to store a computer system's BIOS, and can be updated without returning the unit to the factory.

Flash memory

Flash memory, also known as flash storage, is a type of nonvolatile memory that erases data in units called blocks. A block stored on a flash memory chip must be erased before data can be written or programmed to the microchip. Flash memory retains data for an extended period of time, regardless of whether a flash-equipped device is powered on or off.

Secondary memory

We know that processor memory, also known as primary memory, is expensive as well as limited. The faster primary memory are also volatile. If we need to store large amount of data or programs permanently, we need a cheaper and permanent memory. Such memory is called secondary memory. Examples:Hard disk,Floppy disk,CD ROM,Pen Drive,Blu ray Disk,DVD.

Primary memory	Secondary memory
○ Faster compared to secondary memory ○ Expensive ○ Low capacity ○ Works directly with the processor	○ Slower when compared to primary memory ○ Cheap ○ Large capacity ○ Not connected directly to the processor

HARD DISK

A hard disk drive (HDD), hard disk, is type of secondary memory and an electromechanical data storage device that uses magnetic storage to store and retrieve digital information using one or more rigid rapidly rotating disks (platters) coated with magnetic material. The platters are paired with magnetic heads, usually arranged on a moving actuator arm, which read and write data to the platter surfaces. Data is accessed in a random-access manner, meaning that individual blocks of data can be stored or retrieved in any order and not only sequentially.

Interrupts

In system programming, an interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. An interrupt alerts the processor to a high-priority condition requiring the interruption of the current code the processor is executing. The processor responds by suspending its current activities, saving its state, and executing a function called an interrupt handler (or an interrupt service routine, ISR) to deal with the event. This interruption is temporary, and, after the interrupt handler finishes, the processor resumes normal activities. There are two types of interrupts: hardware interrupts and software interrupts.

Hardware interrupts are used by devices to communicate that they require attention from the operating system. Internally, hardware interrupts are implemented using electronic alerting signals that are sent to the processor from an external device, which is either a part of the computer itself, such as a disk controller, or an external peripheral. For example, pressing a key on the keyboard or moving the mouse triggers hardware interrupts that cause the processor to read the keystroke or mouse position. The act of initiating a hardware interrupt is referred to as an interrupt request (IRQ).

Software interrupt is caused either by an exceptional condition in the processor itself, or a special instruction in the instruction set which causes an interrupt when it is executed. The former is often called a trap or exception and is used for errors or events occurring during program execution that are exceptional enough that they cannot be handled within the program itself. For example, a divide-by-zero exception will be thrown if the processor's arithmetic logic unit is commanded to divide a number by zero as this instruction is an error and impossible. The operating system will catch this exception, and can decide what to do about it: usually aborting the process and displaying an error message. Software interrupt instructions can function similarly to subroutine calls and are used for a variety of purposes, such as to request services from device drivers, like interrupts sent to and from a disk controller to request reading or writing of data to and from the disk.

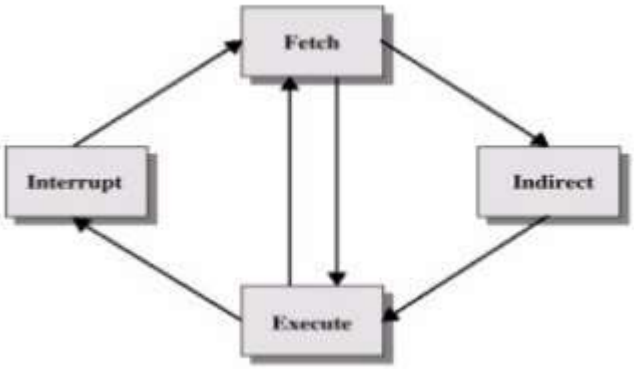
Instruction-Fetch Cycle(Instruction Cycle)

1.Fetch instruction (Fetch cycle) :It brings the instruction or opcode from the program memory.

2.Decode instruction (decode cycle) : decodes the opcode.

3.Read effective address the computer checks whether it's a direct or indirect memory operation.

4.Execute instruction(Execute cycle) : Finally the execution is done in execute cycle.



Addressing modes:

Addressing modes are the schemes for specifying locations of operands. They are intended to provide more flexibility in specifying operand addresses. The following are the commonly used addressing modes:

1. Immediate addressing mode : Here the operand is given explicitly in the instruction.
   Ex. MOVE #200,R1

This instruction places the value 200 in register R1. The immediate mode is used to specify the value of the source operand.

R1← 200

2. Register addressing mode : Here the operand is in one of the processor registers.
   Ex: ADD R1,R2

The address of the general purpose processor registers R1 and R2 are given explicitly as a part of the instruction.

R2← [R1]+[R2]

3. Absolute(Direct addressing mode) :
   Here the operand is in the main memory location.
   Ex: LOAD X

The address of the memory location X is given explicitly as a part of the instruction.

Accumulator ← X

4. Indirect addressing :
   In the indirect addressing mode the effective address of the operand is in the register or main memory location. Ex: ADD (A),R0

   Memory address A holds the effective address of the source operand.
   R0← [[A]]+R0

5. Index addressing mode:

   In this addressing mode , the effective address of the operand is generated by adding a

constant value to the contents of the register, referred to as index register.
It is denoted as X(Ri) where EA= X+[Ri]

Ex: ADD 20(R1),R2

EA= 20+[R1]

6. Base with Index Addressing mode :

The effective address of the operand is generated by adding the contents of an index register with contents of a base register.

Ex: ADD (R1,R2),R3

R1 and R2 hold the offset and base address respectively of the source operand. Result is stored in R3.

7. Base with index and offset addressing mode:
   Here the effective address of the operand is generated by adding the contents of an index register with contents of a base register and a constant provided in the instruction.
   Ex: ADD 10(R1,R2),R3
   Effective address = 10+[R1]+[R2]

8. Relative addressing mode :

   The effective address of the operand is generated by adding the contents of program counter with the constant provided in the instruction. Program counter identifies the current execution point in a program.
   X(PC)
   Effective address = X+ [PC]
   Ex: ADD 20(PC) , R2
9. Autoincrement addressing mode:
   The effective address of the operand is the contents of a register specified in the instruction. After accessing the operand, the contents of this register are incremented to point to the next item in the list.
   Ex:
   ADD [R1]+,R2
   R2← [[R1]]+[R2]
   R1←[R1]+1
10. Autodecrement addressing mode :
    The contents of a register specified in the instruction are decremented.Then these contents are used as the effective address of the operand.
    Ex:
    ADD –[R1],R2
    R1← [R1]-1
    R2← [[R1]]+[R2]