



Powered By



Futureense

Revolutionising B.Tech



Powered By

Futureense

Module 1: Fundamentals and basics in COA

Course Name: Computer Architecture and organization[22CSE104]

Total Hours : 12

Table of Content

- Aim
- Objectives
- Functional Units
- Basic Operational Concepts
- Bus Structure
- Performance
- Multi Processors and Multi-computers
- Encoders
- Demultiplexers
- Programmable Logic Arrays(PLAs)
- Digital Logic Circuits
- Basic Logic Functions
- Synthesis of Logic Functions Using AND,OR and NOT Gates
- Minimization of Logic Expressions

Table of Content

- Synthesis with NAND and NOR Gates
- Flip Flops
- Self Assessments
- Activities
- Did You Know
- Summary
- Terminal Questions

Aim



To equip students in the fundamentals and understanding the Concepts of Interconnections of Computers and make them to design the Logic Gates.



Objective

- a. Discuss on the various basic concepts and Structure of Computers.
- b. Understanding different types of Logic Functions and Synthesis Techniques
- c. Understanding and practice of Designing Encoders and Demultiplexer



Computer Organization

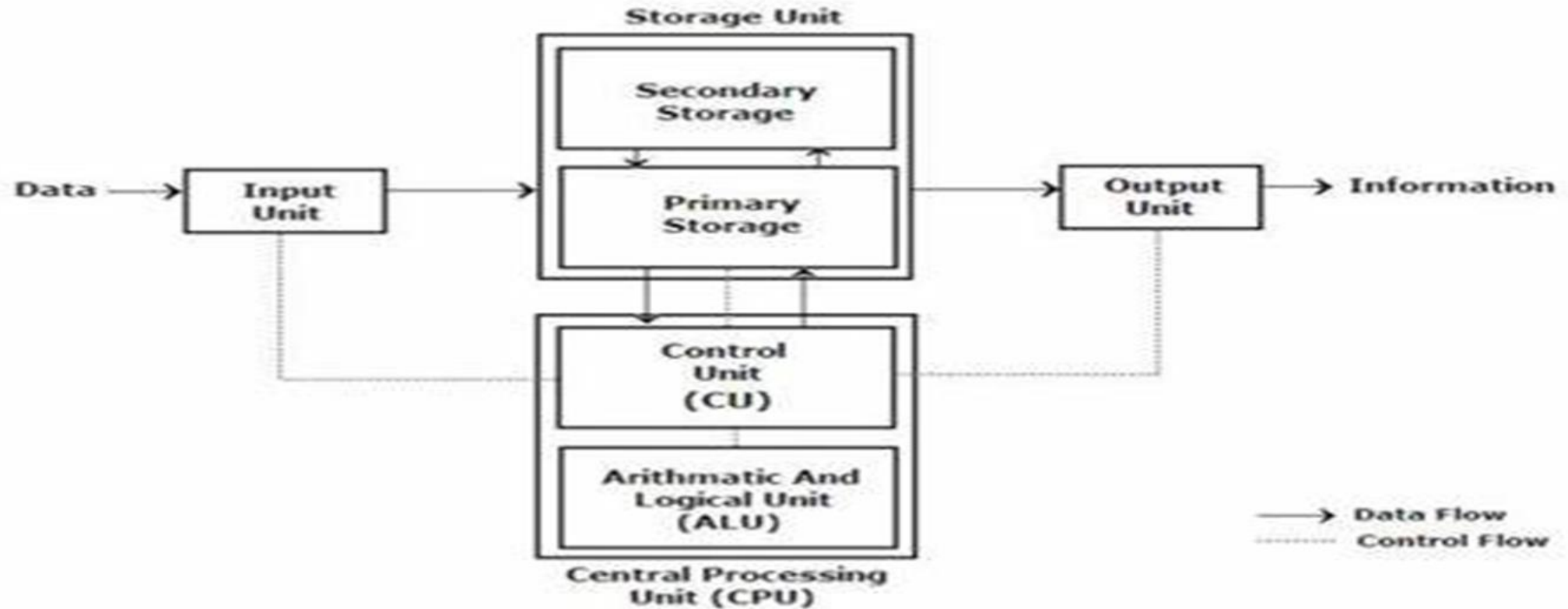
- Computer organization refers to the operational units and their interconnections .
- It deals with how the components of a computer system are arranged and how they interact to perform the required operations.
- Computer organization is concerned with the physical implementation like Circuit Design, Peripherals and Adders.
- Interconnection and communication between components, such as the bus structure, memory hierarchy, and input/output systems.
- It is frequently called as Micro Architecture.

Computer Architecture

- Computer Architecture deals with structure and behavior of Computer System.
- Computer architecture refers to the design of the internal workings of a computer system, including the CPU, memory, and other hardware components.
- It involves the logical functions such as Instruction sets, Data types, Registers and Addressing modes.
- Computer architecture is concerned with optimizing the performance of a computer system and ensuring that it can execute instructions quickly and efficiently.
- Computer Architecture is also called Instruction Set Architecture (ISA).

Functional Units of a Computer

Block diagram of computer



Input Unit

- Computers accept coded information through input units.
- The most common input device is the keyboard.
- When we give the input the corresponding letter or digit is automatically translated into its corresponding binary code and transmitted to the processor.
- Human-computer interaction are available, including the touchpad, mouse, joystick, and trackball.
- Graphic input devices in conjunction with displays.
- Microphones can be used to capture audio input which is then sampled and converted into digital codes for storage and processing.
- Cameras can be used to capture video input.
- Digital communication facilities, such as the Internet, can also provide input to a computer from other computers and database servers.

Memory Unit

- The function of the memory unit is to store programs and data.
- There are two classes of storage, called primary and secondary.
- Primary Memory, also called main memory, is a fast memory that operates at electronic speeds.
- The number of bits in each word is referred to as the **word length of the computer**, typically 16, 32, or 64 bits.
- A memory in which any location can be accessed in a short and fixed amount of time after specifying its address is called **a random-access memory (RAM)**.
- The time required to access one word is called the **memory access time. (MAT)**
- This time is independent of the location of the word being accessed. It typically ranges from a few nanoseconds (ns) to about 100 ns for current RAM units

Cache memory

- Smaller, faster RAM unit, called a **cache**, is stored between CPU and main memory.
- The cache is tightly coupled with the processor and is usually contained on the same integrated-circuit chip.
- The purpose of the cache is to facilitate high instruction execution rates at faster Execution rates.
- At the start of program execution, the cache is empty.
- As execution proceeds, **instructions are fetched into the processor chip, and a copy of each is placed in the cache.**
- When the execution of an instruction requires data, **located in the main memory, the data are fetched and copies are also placed in the cache.**

Secondary Storage

- Secondary Storage are additional, less expensive is used to store large amounts of data .
- Access times for secondary storage are longer than for primary memory.
- The devices available are including magnetic disks, optical disks (DVD and CD), and flash memory devices.

Arithmetic and Logic Unit(ALU)

- Computer operations are executed in the arithmetic and logic unit (ALU) of the processor.
- Arithmetic operations Addition, Subtraction, Multiplication, Division.
- Logical Operations AND,OR,NOT
- When operands are brought into the processor, they are stored in high-speed storage elements called registers.
- Each register can store one word of data.
- Access times to registers are even shorter than access times to the cache unit on the processor chip.

Control Unit

- The control unit controls and coordinate all the activities that sends control signals to other units.
- Memory, arithmetic and logic, and I/O units store and process information and perform input and output operations.
- Control units are responsible for generating the timing signals that govern the transfers.
- Data transfers between the processor and the memory are also managed by the control unit through timing signals.
- A large set of control lines carries the signals used for timing and synchronization of events in all units.

Output Unit

- Output unit function is to send processed results to the outside world.
- A familiar example of such a device is a printer.
- laser printers, or ink jet streams. Such printers may generate output at speeds of 20 or more pages per minute.
- Graphic displays provide both an output function showing text and graphics and an input function, through touchscreen capability.

Summarisation of Operations of a Computer

- The operation of a computer can be summarized as follows:
- The computer accepts information in the form of programs and data through an input unit and stores it in the memory.
- Information stored in the memory is fetched and send to arithmetic and logic unit where it is processed.
- Processed information leaves the computer through an output unit.
- All activities in the computer are directed by the control unit.

A Typical Instruction

- **Add LOCA, R0**
- Add the operand at memory location LOCA to the operand in a register R0 in the processor.
- Place the sum into register R0.
- The original contents of LOCA are preserved.
- The original contents of R0 is overwritten.
- Instruction is fetched from the memory into the processor – the operand at LOCA is fetched and added to the contents of R0 – the resulting sum is stored in register R0.

Separate Memory Access and ALU Operation

- Load LOCA, R1
- Add R1, R0
- Whose contents will be overwritten?

Connection Between the Processor and the Memory

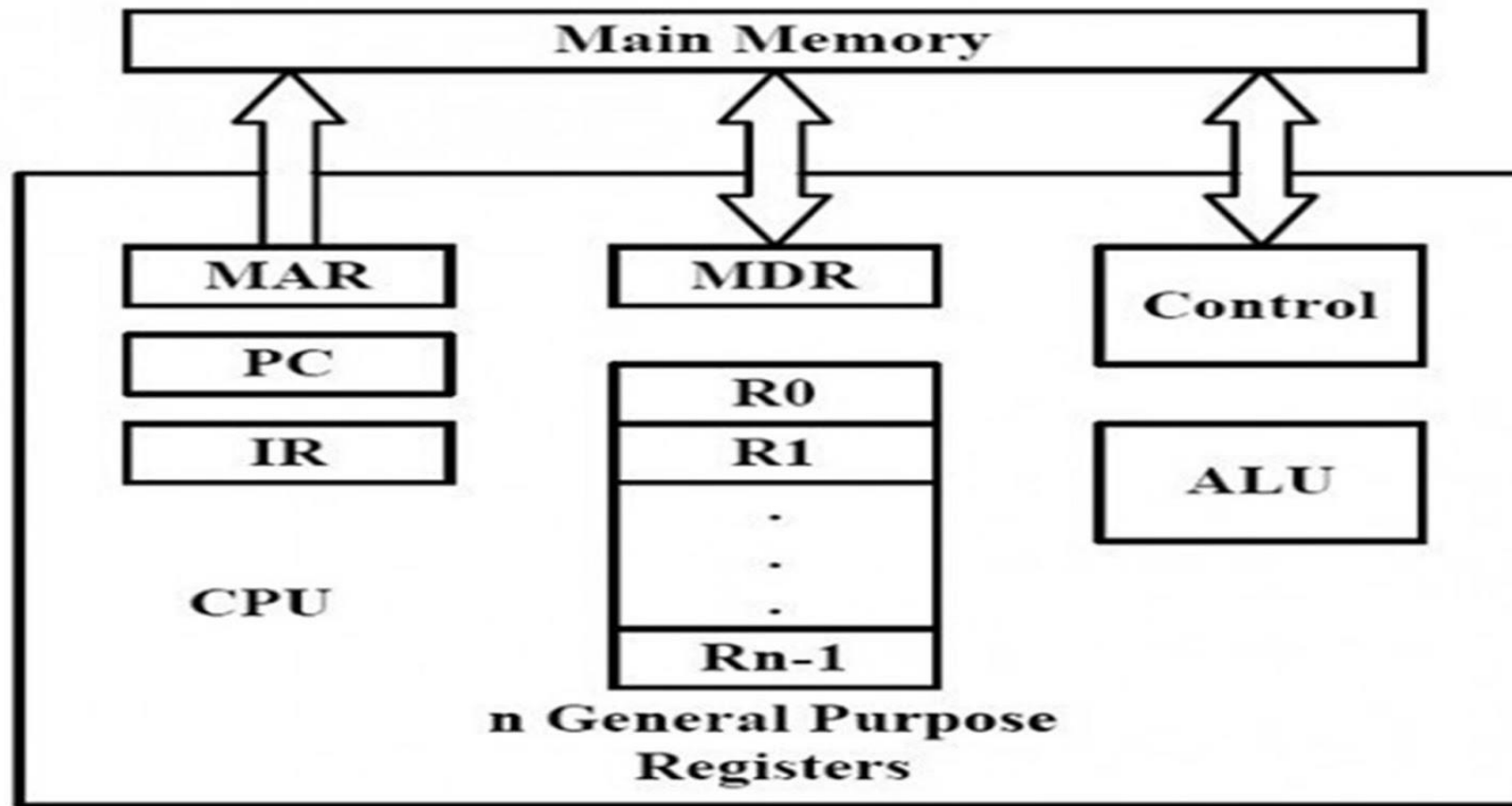


Figure: Connection between MM & Processor

Registers

- Instruction register (IR)
- Program counter (PC)
- General-purpose register ($R_0 - R_{n-1}$)
- Memory address register (MAR)
- Memory data register (MDR)

Typical Operating Steps

- Programs reside in the memory through input devices
- PC is set to point to the first instruction
- The contents of PC are transferred to MAR
- A Read signal is sent to the memory
- The first instruction is read out and loaded into MDR
- The contents of MDR are transferred to IR
- Decode and execute the instruction

Typical Operating Steps (Cont')

- Get operands for ALU
 - General-purpose register
 - Memory (address to MAR – Read – MDR to ALU)
- Perform operation in ALU
- Store the result back
 - To general-purpose register
 - To memory (address to MAR, result to MDR – Write)
- During the execution, PC is incremented to the next instruction

Interrupt

- Normal execution of programs may be preempted if some device requires urgent servicing.
- The normal execution of the current program must be interrupted – the device raises an *interrupt* signal.
- Interrupt-service routine
- Current system information backup and restore (PC, general-purpose registers, control information, specific information)

Bus Structure

Single Bus

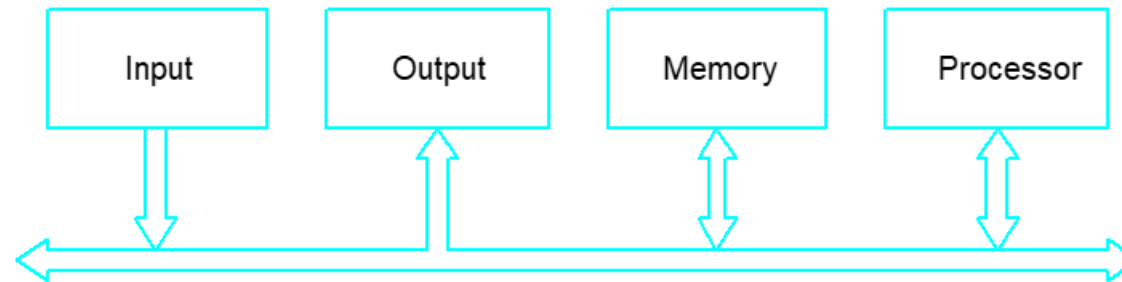


Figure 1.3. Single-bus structure.

Speed Issue

- Different devices have different transfer/operate speed.
- If the speed of bus is bounded by the slowest device connected to it, the efficiency will be very low.
- How to solve this?
- A common approach – use buffers.

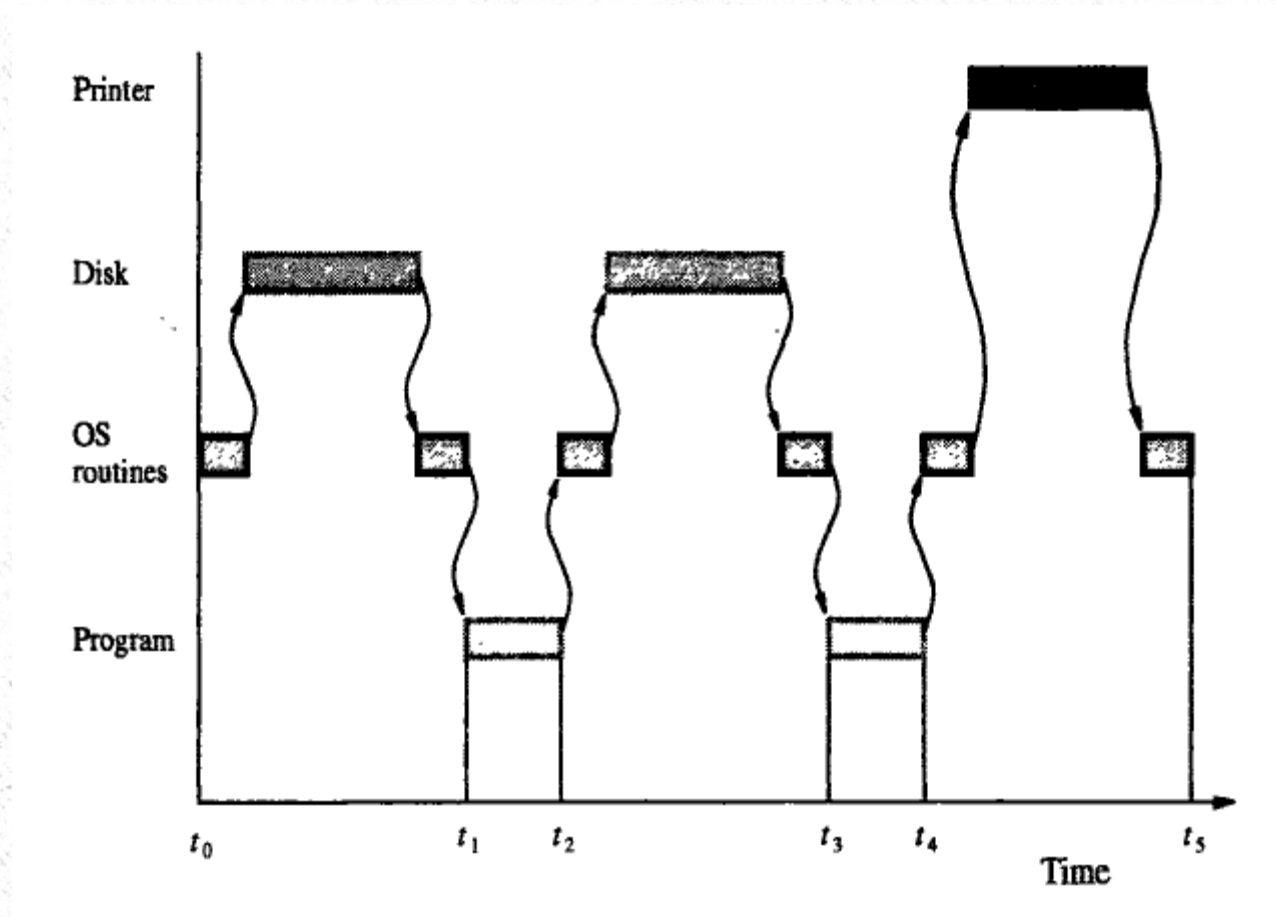
SYSTEM SOFTWARE

- **Compiler**
- **Text Editor**
- **Operating System:** Large program or collection of routines that is used to control the sharing of and interaction among various computer units as they execute the application programs.
- **OS routines:** Performs the task required to assign computer resources to individual application programs.

Application Task to be Performed

1. Reading a Data file from the Disk to the memory.
2. Performing some computation on the data.
3. Printing the results.

User Program and OS routine sharing of the processor.



Performance

- The most important measure of a computer is how quickly it can execute programs.
- Three factors affect performance:
 - Hardware design
 - Instruction set
 - Compiler

Performance

- Processor time to execute a program depends on the hardware involved in the execution of individual machine instructions.

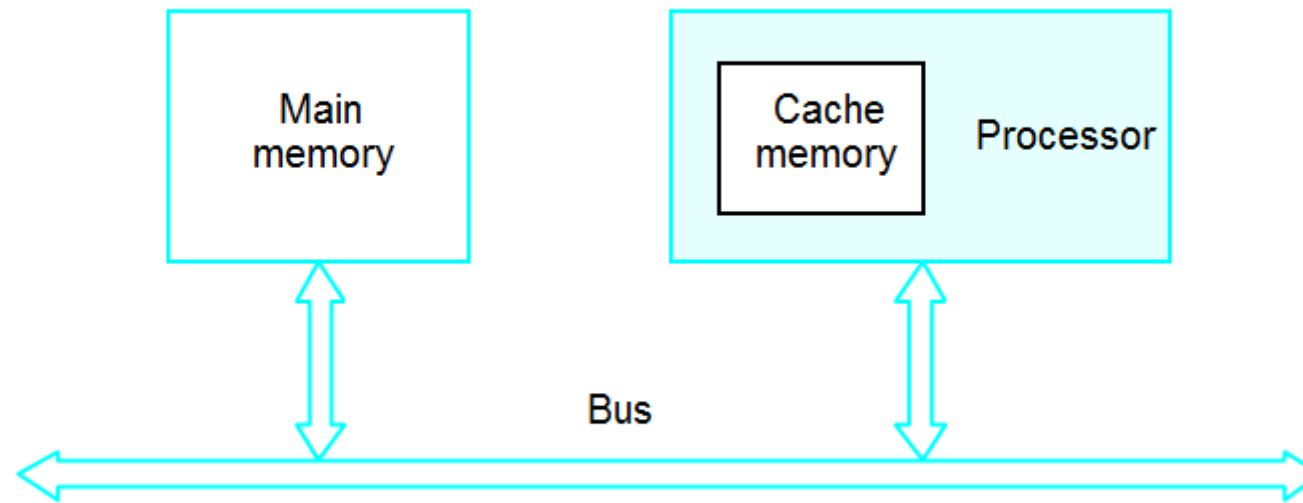


Figure 1.5. The processor cache.



Performance

- The processor and a relatively small cache memory can be fabricated on a single integrated circuit chip.
- Speed
- Cost
- Memory management

Processor Clock

- Clock, clock cycle, and clock rate
- The execution of each instruction is divided into several steps, each of which completes in one clock cycle.
- Hertz – cycles per second

Basic Performance Equation

- T – processor time required to execute a program that has been prepared in high-level language
- N – number of actual machine language instructions needed to complete the execution (note: loop)
- S – average number of basic steps needed to execute one machine instruction. Each step completes in one clock cycle
- R – clock rate
- Note: these are not independent to each other

$$T = \frac{N \times S}{R}$$

How to improve T?

Pipeline and Superscalar Operation

- Instructions are not necessarily executed one after another.
- The value of S doesn't have to be the number of clock cycles to execute one instruction.
- Pipelining – overlapping the execution of successive instructions.
- Add R1, R2, R3
- Superscalar operation – multiple instruction pipelines are implemented in the processor.
- Goal – reduce S.

Clock Rate

- Increase clock rate
 - Improve the integrated-circuit (IC) technology to make the circuits faster
 - Reduce the amount of processing done in one basic step (however, this may increase the number of basic steps needed)
- Increases in R that are entirely caused by improvements in IC technology affect all aspects of the processor's operation equally except the time to access the main memory.

CISC and RISC

- Tradeoff between N and S
- A key consideration is the use of pipelining
 - S is close to 1 even though the number of basic steps per instruction may be considerably larger
 - It is much easier to implement efficient pipelining in processor with simple instruction sets
- Reduced Instruction Set Computers (RISC)
- Complex Instruction Set Computers (CISC)

Compiler

- A compiler translates a high-level language program into a sequence of machine instructions.
- To reduce N, we need a suitable machine instruction set and a compiler that makes good use of it.
- Goal – reduce $N \times S$
- A compiler may not be designed for a specific processor; however, a high-quality compiler is usually designed for, and with, a specific processor.

Performance Measurement

- Measure computer performance using benchmark programs.
- System Performance Evaluation Corporation (SPEC) selects and publishes representative application programs for different application domains, together with test results for many commercially available computers.
- Compile and run (no simulation)
- Reference computer

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

$$SPEC \text{ rating} = \left(\prod_{i=1}^n SPEC_i \right)^{\frac{1}{n}}$$

Performance Evaluation of CPU

- **Example:** In a computer the clock speed of processor is 4 GHz. Suppose you are executing a program with 100 instruction . And each instruction require 1.2 clock cycle to complete.What will be the execution time for the computer (in ns)?

➤ Number of instruction in program or instruction count (IC) = 100, Number of clock cycle require per instruction (CPI) = 1.2

Clock speed = 4GHz

Clock cycle time = $1 / (4 \times 10^9)$ sec = 0.25×10^{-9} sec = 0.25 ns Then,

CPU execution time $T = IC \times CPI \times$ Clock cycle time

$$= 100 \times 1.2 \times 0.25$$

$$T = 30 \text{ ns}$$

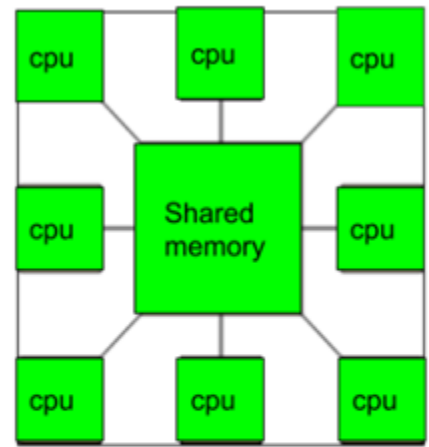
Multiprocessors and Multicomputers

- **Multiprocessor computer**

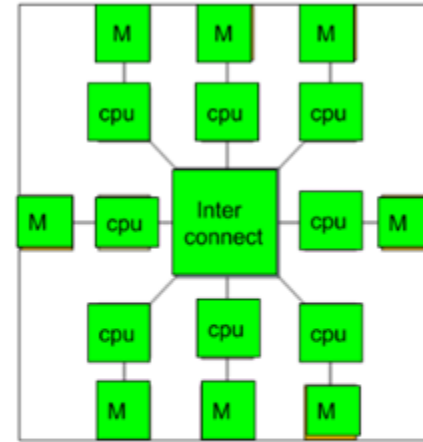
- Execute a number of different application tasks in parallel
- Execute subtasks of a single large task in parallel
- All processors have access to all of the memory – shared-memory multiprocessor
- Cost – processors, memory units, complex interconnection networks

- **Multicomputers**

- Each computer only have access to its own memory
- Exchange message via a communication network – message-passing multicomputers



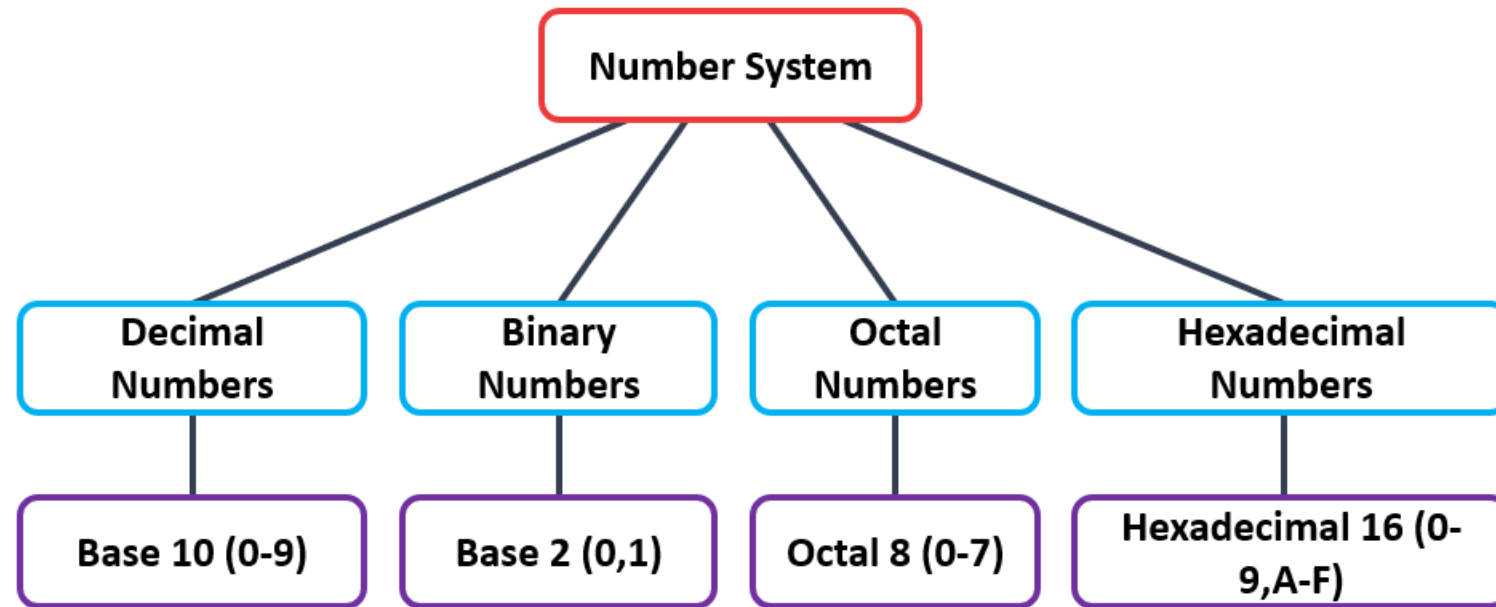
Multiprocessor



Multicomputer

Number System

- A number system defines a set of values used to represent quantity.



Different Number Systems

✓ Decimal Number System

- Base 10

✓ Binary Number System

- Base 2

✓ Octal Number System

- Base 8

✓ Hexadecimal Number System

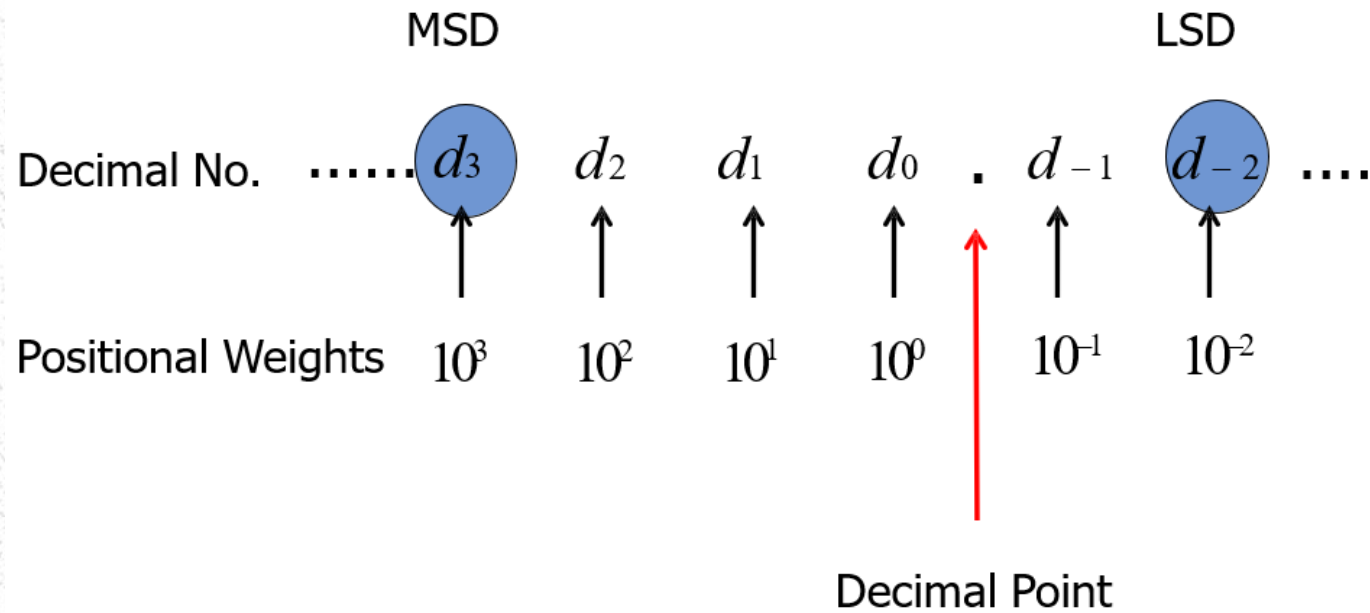
-Base 16

Decimal Number System

- ✓ Decimal number system contains ten unique symbols 0,1,2,3,4,5,6,7,8 and 9
- ✓ Since counting in decimal involves ten symbols, we can say that its base or radix is ten.
- ✓ It is a positional weighted system

Decimal Number System

Structure:

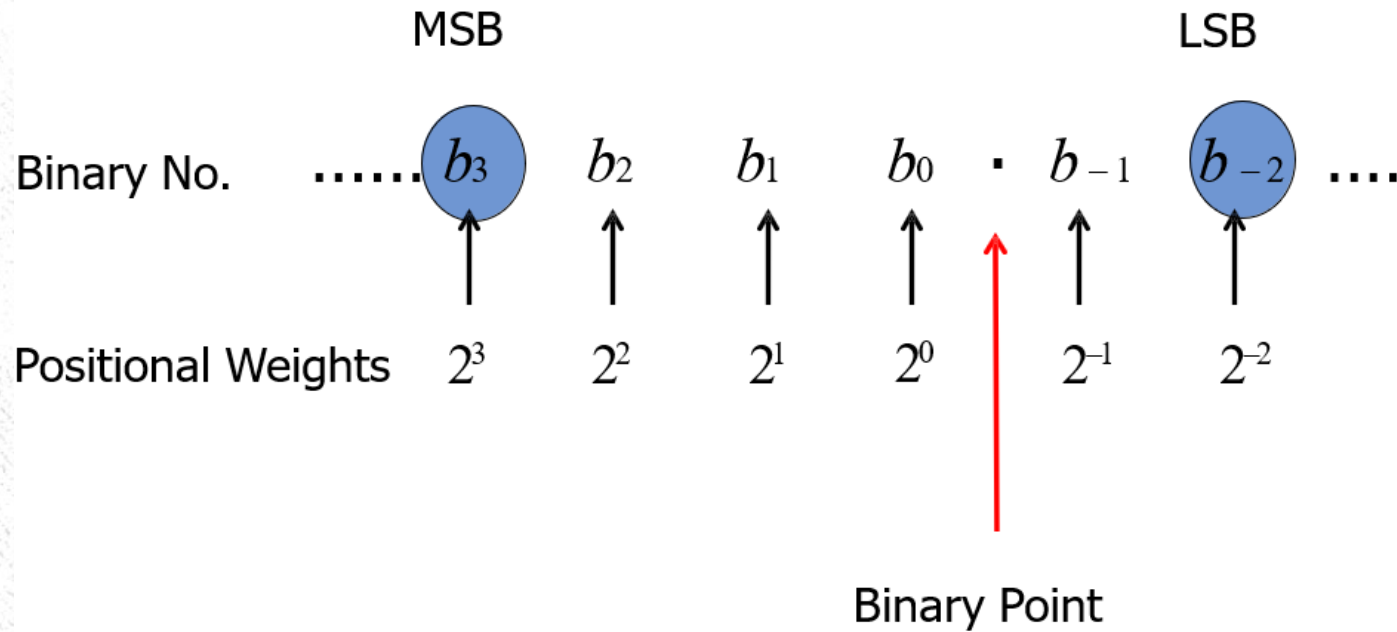


Binary Number System

- ✓ Binary number system is a positional weighted system
- ✓ It contains two unique symbols 0 and 1
- ✓ Since counting in binary involves two symbols, we can say that its base or radix is two.

Binary Number System

Structure:



Binary Number System

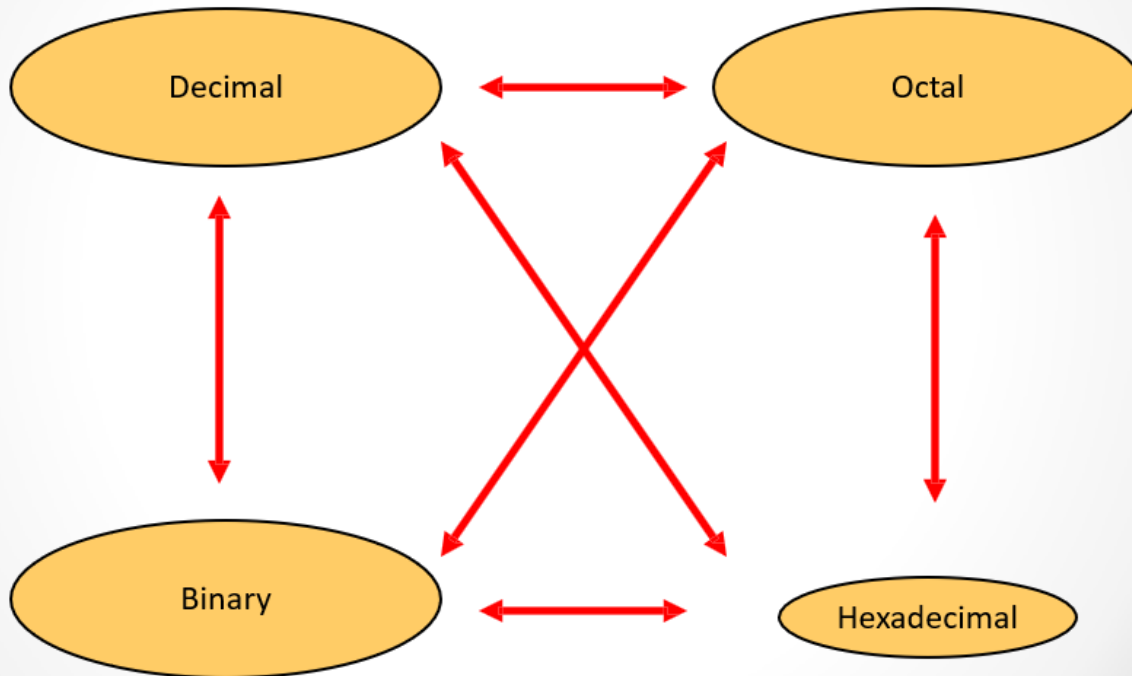
Decimal No.	Binary No.
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Decimal No.	Binary No.
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111



Conversion Among Bases

Possibilities



Conversion of Decimal number into Binary number (Integer Number)

Procedure:

1. Divide the decimal no by the base 2, noting the remainder.
2. Continue to divide the quotient by 2 until there is nothing left, keeping the track of the remainders from each step.
3. List the remainder values in reverse order to
• d the number's binary equivalent •

Example: Convert 105 decimal number in to it's equivalent binary number.

2	105	
2	52	1
2	26	0
2	13	0
2	6	1
2	3	0
2	1	1
	0	1

LSB

MSB

$$(105)_{10} = (1101001)_2$$

Conversion of Decimal number into Binary number (Fractional Number)

Procedure:

1. Multiply the given fractional number by base 2.
2. Record the carry generated in this multiplication as MSB.
3. Multiply only the fractional number of the product in step 2 by 2 and record the carry as the next bit to MSB.
4. Repeat the steps 2 and 3 up to 5 bits. The last carry will represent the LSB of equivalent binary number

Example: Convert 0.42 decimal number in to it's equivalent binary number.

$0.42 \times 2 = 0.84$	0	<div>MSB ↓ LSB</div>
$0.84 \times 2 = 1.68$	1	
$0.68 \times 2 = 1.36$	1	
$0.36 \times 2 = 0.72$	0	
$0.72 \times 2 = 1.44$	1	

$$(0.42)_{10} = (0.01101)_2$$

- Convert following Decimal Numbers in to its equivalent Binary Number:

1. $(1248.56)_{10} = (?)_2$

2. $(8957.75)_{10} = (?)_2$

3. $(420.6)_{10} = (?)_2$

4. $(8476.47)_{10} = (?)_2$

Conversion of Binary Number into Decimal Number

Procedure:

1. Write down the binary number.
2. Write down the weights for different positions.
3. Multiply each bit in the binary number with the corresponding weight to obtain product numbers to get the decimal numbers.
4. Add all the product numbers to get the decimal

**Example: Convert 1011.01 binary number
in to it's equivalent decimal number.**

Binary No.	1	0	1	1	·	0	1
	↑	↑	↑	↑		↑	↑
Positional Weights	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}

$$\begin{aligned} &= (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) \\ &= 8 + 0 + 2 + 1 + 0 + 0.25 \\ &= 11.25 \end{aligned}$$

$$(1011.01)_2 = (11.25)_{10}$$

Exercise

- Convert following Binary Numbers in to its equivalent Decimal Number:
 - $(1101110.011)_2 = (?)_{10}$
 - $(1101.11)_2 = (?)_{10}$
 - $(10001.01)_2 = (?)_{10}$

Introduction to Logic Gates

- Logic gates are the basic building blocks of any digital system. Logic gates are electronic circuits having one or more than one input and only one output.
- The relationship between the input and the output is based on a certain logic. Based on this, logic gates are named as
 - 1) AND gate
 - 2) OR gate
 - 3) NOT gate
 - 4) NAND gate
 - 5) NOR gate
 - 6) Ex-OR gate
 - 7) Ex-NOR gate

AND Gate

- It is an electronic circuit, which generates an output signal of 1 if and only if all input signals are also 1.
- An AND gate is the physical realization of the logical multiplication (AND operation)

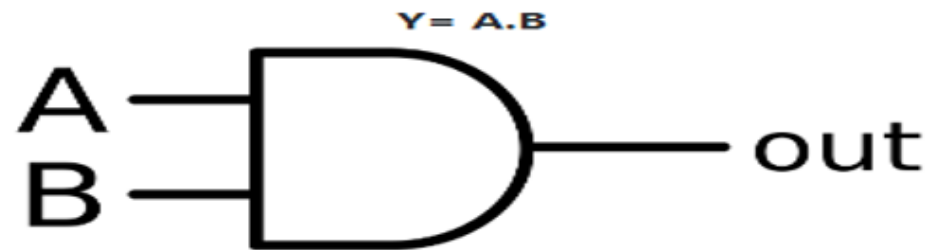


Figure-1: Logic Symbol of AND Gate

Input		Output
A	B	$Y = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

Figure-2: Truth Table of AND Gate

OR gate

- The OR gate is an electronic circuit that gives a high output (1) if one or more of its inputs are high. A plus (+) is used to show the OR operation.

$$Y = A + B$$

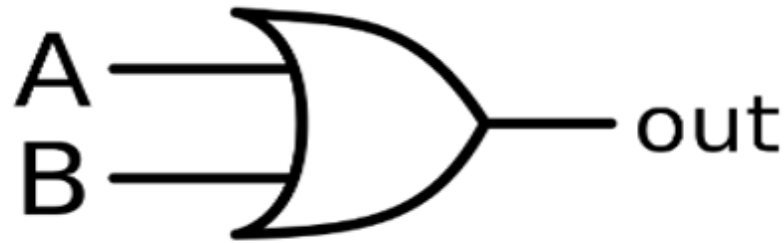


Figure-4: Logic Symbol of OR Gate

Input		Output
A	B	$Y = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

Figure-5: Truth Table of OR Gate

NOT gate

- The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an inverter.
- If the input variable is A, the inverted output is known as NOT A. This is also shown as A' or A with a bar over the top, as shown at the outputs.

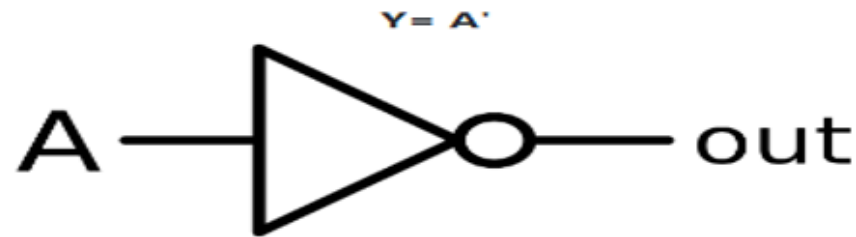


Figure-7: Logic Symbol of NOT Gate

Input	Output
A	Y
0	1
1	0

Figure-8: Truth Table of NOT Gate

NAND gate

- This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if any of the inputs are low.
- The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

$$Y = \overline{AB}$$



Figure-10: Logic Symbol of NAND Gate

Input		Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Figure-11: Truth Table of NAND Gate

NOR gate

This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if any of the inputs are high.

- The symbol is an OR gate with a small circle on the output. The small circle represents inversion.

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

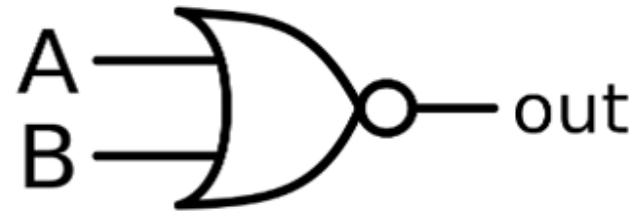


Figure-13: Logic Symbol of NOR gate

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

Figure-14: Truth Table of NOR gate

- **Ex-OR gate**

The 'Exclusive-OR' gate is a circuit which will give a high output if either, but not both of its two inputs are high.

- An encircled plus sign (\oplus) is used to show the Ex-OR operation.

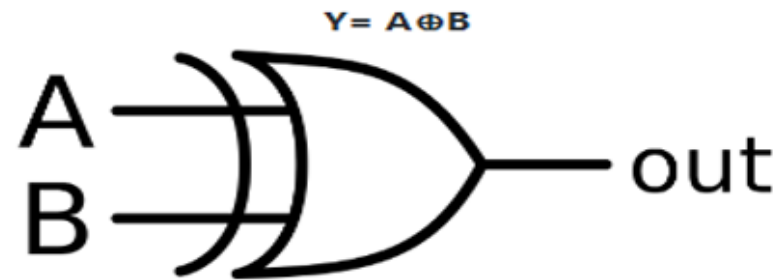


Figure-16: Logic Symbol of Ex-OR gate

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Ex-NOR gate

- The 'Exclusive-NOR' gate circuit does the opposite to the EX-OR gate. It will give a low output if either, but not both of its two inputs are high.
- The symbol is an EX-OR gate with a small circle on the output. The small circle represents inversion.

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

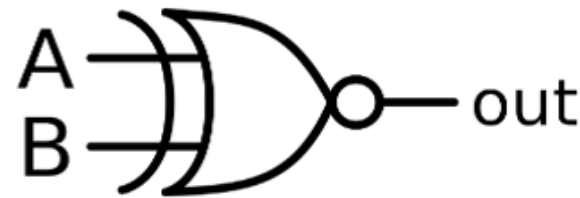


Figure-19: Logic Symbol of Ex-NOR gate

XNOR Truth Table		
A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

NAND gates as NOT gate

- A NOT produces complement of the input. It can have only one input, tie the inputs of a NAND gate together. Now it will work as a NOT gate.
- Its output is $Y = (A.A)'$
 $Y = (A)'$



NOT (inverter)

Figure-1:NAND gates as NOT gate

Input	Output
A	A'
0	1
1	0

Figure-2:Truth table of NOT

NAND gates as AND gate

- A NAND produces complement of AND gate. So, if the output of a NAND gate is inverted, overall output will be that of an AND gate.

$$Y = ((A.B)')'$$

$$Y = (A.B)$$

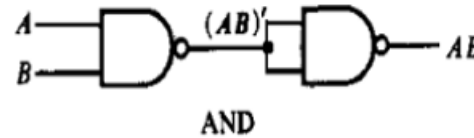


Figure-3:NAND gates as AND gate

Input		Output
A	B	$F = A.B$
0	0	0
0	1	0
1	0	0
1	1	1

Figure-4:Truth table of AND

NAND gates as OR gate

- From DeMorgan's theorems:

$$(A.B)' = A' + B'$$

$$(A'.B')' = A'' + B'' = A + B$$

So, give the inverted inputs to a NAND gate, obtain OR operation at output.

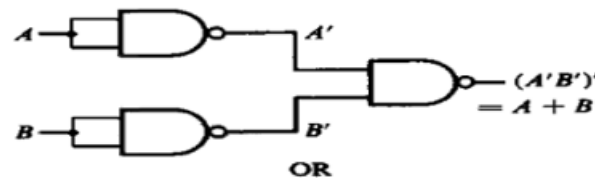


Figure-5:NAND gates as OR gate

A	B	X = A+B
0	0	0
0	1	1
1	0	1
1	1	1

Figure-6:Truth table of OR

NOR gates as NOT gate

- A NOT produces complement of the input. It can have only one input, tie the inputs of a NOR gate together. Now it will work as a NOT gate.
- Its output is

$$Y = (A+A)'$$

$$Y = (A)'$$

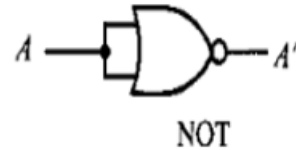


Figure-12:NOR gates as NOT gate

Input	Output
A	A'
0	1
1	0

Figure-13:Truth table of NOT

NOR gates as OR gate

- A NOR produces complement of OR gate. So, if the output of a NOR gate is inverted, overall output will be that of an OR gate.

$$Y = ((A+B)')$$

$$Y = (A+B)$$

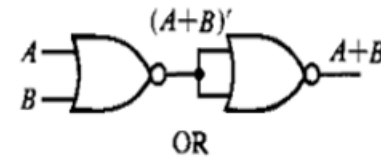


Figure-14: NOR gates as OR gate

A	B	$X = A+B$
0	0	0
0	1	1
1	0	1
1	1	1

Figure-15: Truth table of OR

NOR gates as AND gate

- From DeMorgan's theorems:

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

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

So, give the inverted inputs to a NOR gate, obtain AND operation at output.

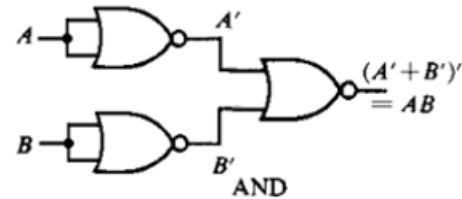


Figure-16: NOR gates as AND gate

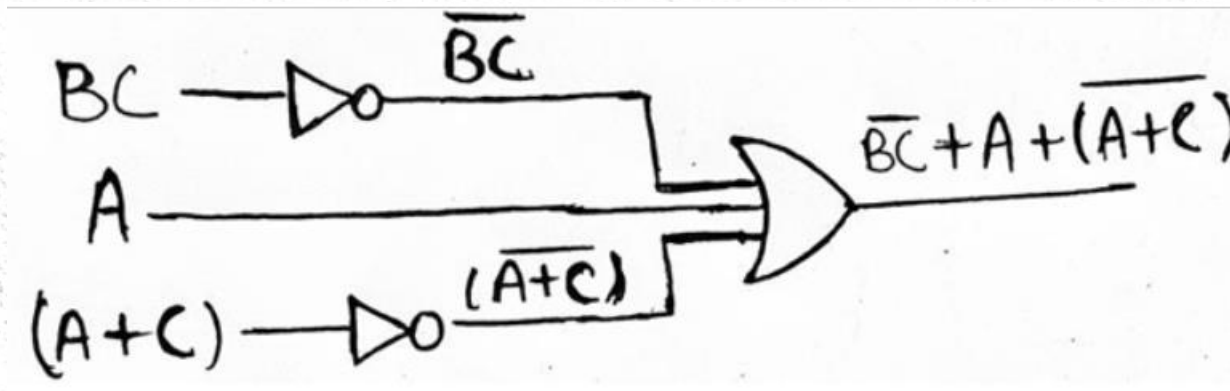
Input		Output
A	B	F = A.B
0	0	0
0	1	0
1	0	0
1	1	1

Figure-17: Truth table of AND

Realization of Boolean expressions using Basic Gates

Example 1

Realize the Boolean Expression $\overline{BC} + A + \overline{(A + C)}$ using AOI logic



Realise the logic expression $Y = \overline{BC} + \overline{AC} + \overline{AB}$ using basic gates.

$$Y = (A+AB). (B+BC). (C+AB),$$

Simplification of Boolean Functions using K-map

What is K-Map

- It's similar to truth table; instead of being organized (i/p and o/p) into columns and rows, the K-map is an array of cells in which each cell represents a binary value of the input variables.
- The cells are arranged in a way so that simplification of a given expression is simply a matter of properly grouping the cells.
- K-maps can be used for expressions with 2, 3, 4, and 5 variables.

1,2,3,4 variable map

	A
\bar{A}	\bar{A}
A	A

1-Variable map

	B	\bar{B}	B
\bar{A}	$\bar{A}\bar{B}$	$\bar{A}B$	
A	$A\bar{B}$	AB	

2-Variable map

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

3-Variable map

	CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}C\bar{D}$	$\bar{A}\bar{B}CD$	
$\bar{A}B$	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BC\bar{D}$	$\bar{A}BCD$	
AB	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABC\bar{D}$	$ABCD$	
$A\bar{B}$	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}C\bar{D}$	$A\bar{B}CD$	

4-Variable map

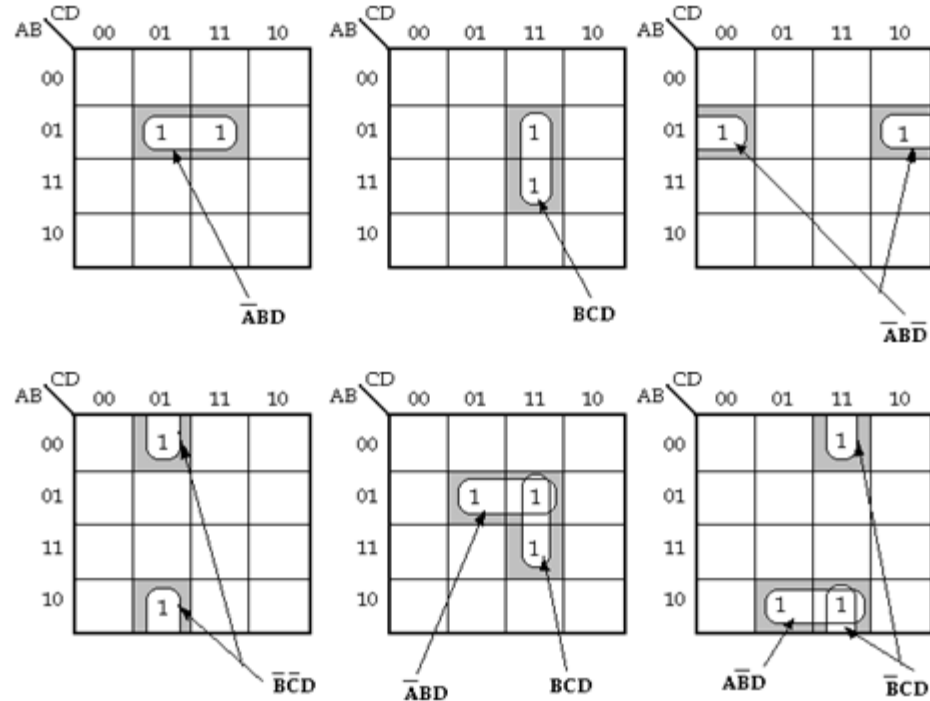
● Grouping cells for Simplification

- The grouping is nothing but combining terms in adjacent cells.
- The simplification is achieved by grouping adjacent 1's or 0's in groups of 2^i , where $i = 1, 2, \dots, n$ and n is the number of variables.
- When adjacent 1's are grouped then we get result in the sum of product form; otherwise we get result in the product of sum form.

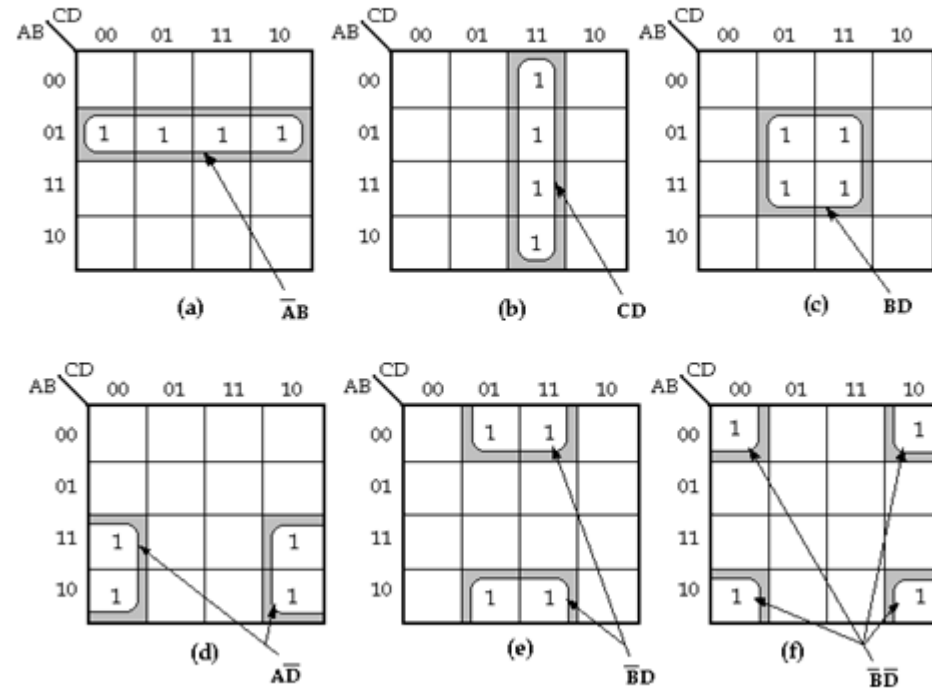
Steps to solve expression using K-map-

1. Select K-map according to the number of variables.
2. Identify minterms or maxterms as given in problem.
3. For SOP put 1's in blocks of K-map respective to the minterms (0's elsewhere).
4. For POS put 0's in blocks of K-map respective to the maxterms (1's elsewhere).
5. Make rectangular groups containing total terms in power of two like 2, 4, 8 .. (except 1) and try to cover as many elements as you can in one group.
6. From the groups made in step 5 find the product terms and sum them up for SOP form.

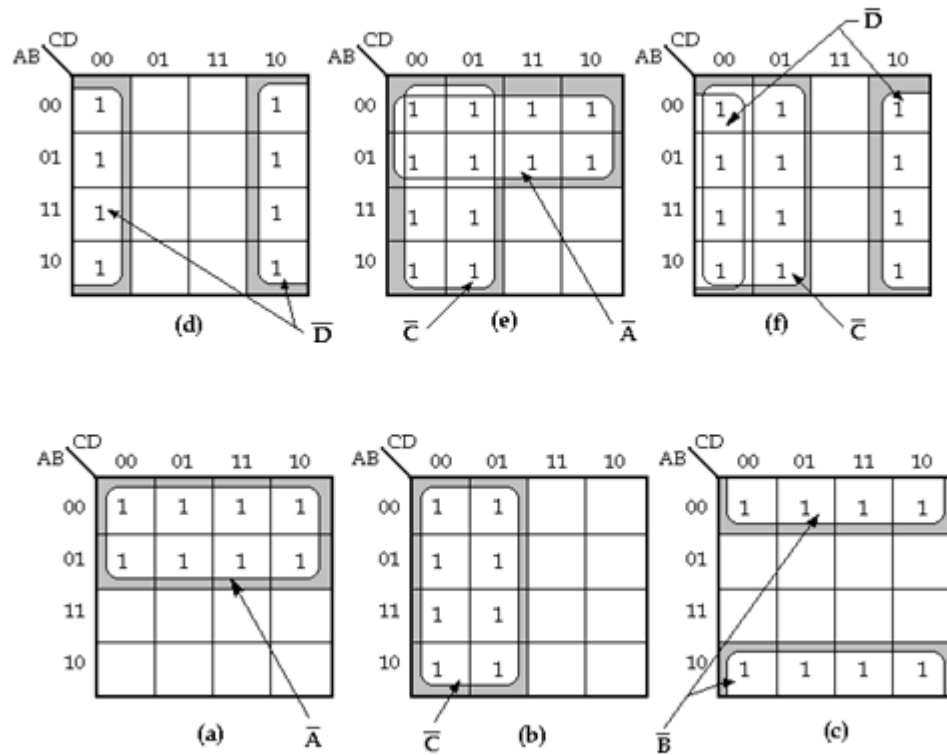
Grouping Two Adjacent 1's: (Pair)



Grouping Four Adjacent 1's: (Quad)



Grouping Eight Adjacent 1's: (Octet)



1. Simplify the Boolean expression using K-Map

$$F(A,B,C,D) = \sum m(0,2,3,7)$$

Use K MAP Reduction technique and obtain simplified logical expression for the function

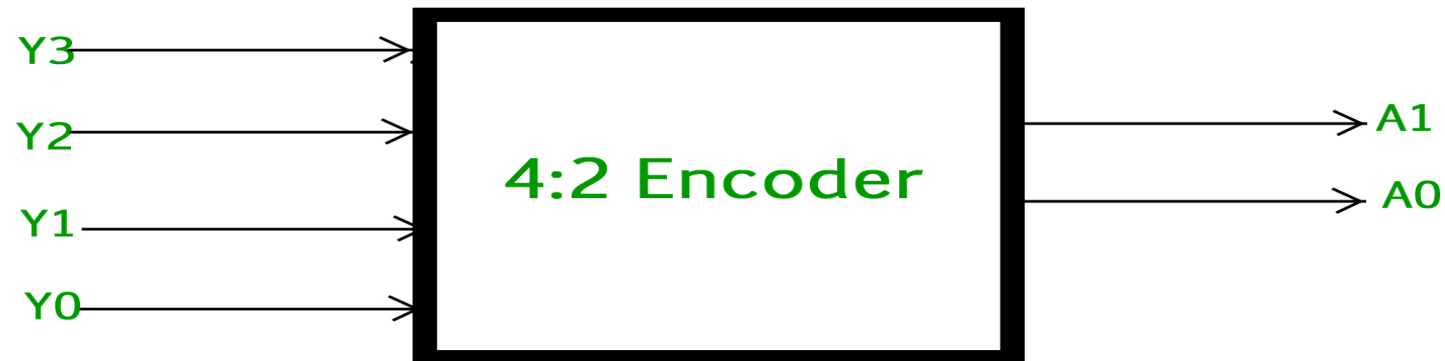
$$F(A, B, C, D) = ABCD + AB'C'D' + AB'C + AB$$

Encoder

- An encoder is a digital circuit that converts a set of binary inputs into a unique binary code.
- The binary code represents the position of the input and is used to identify the specific input that is active.
- Encoders are commonly used in digital systems to convert a parallel set of inputs into a serial code.
- An Encoder is a **combinational circuit** that performs the reverse operation of Decoder.
- It has maximum of **2^n input lines** and '**n**' **output lines**, hence it encodes the information from 2^n inputs into an n-bit code. It will produce a binary code equivalent to the input, which is active High.
- Therefore, the encoder encodes 2^n input lines with 'n' bits.

4:2 Encoder

- **4 : 2 Encoder**
- The 4 to 2 Encoder consists of **four inputs Y3, Y2, Y1 & Y0** and **two outputs A1 & A0**. At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output. The figure below shows the logic symbol of 4 to 2 encoder :



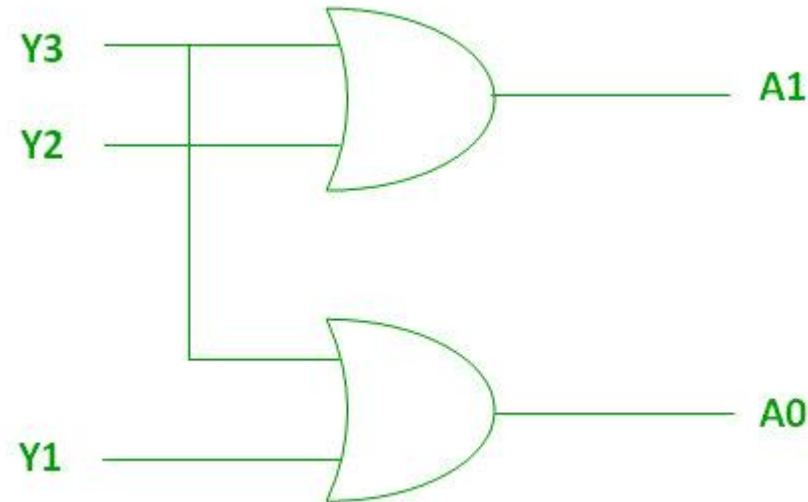
Truth Table

- 4:2 Encoder

INPUTS				OUTPUTS	
Y3	Y2	Y1	Y0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Implementation

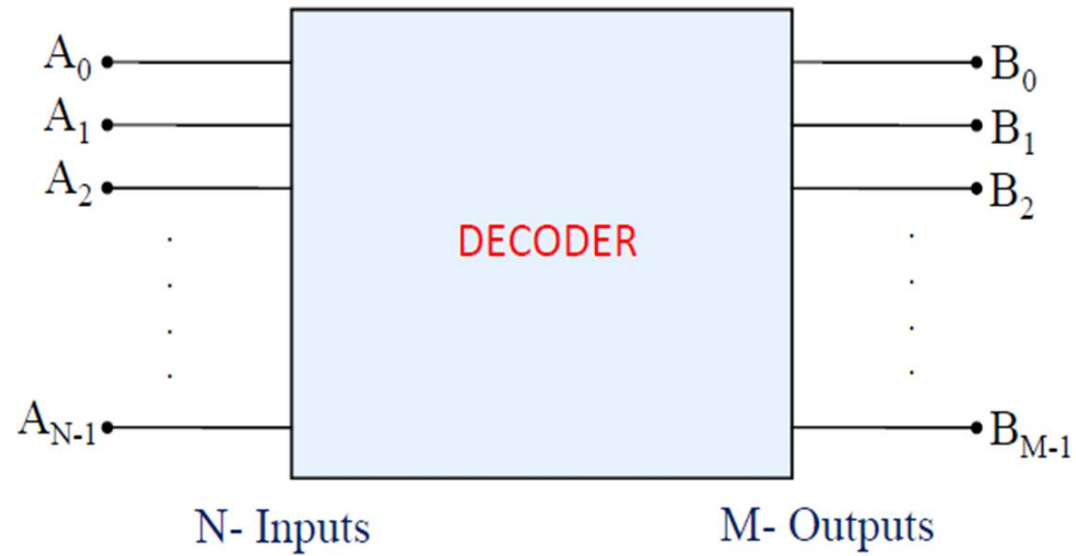
- Logical Expression for A0 and A1
- $A1 = Y3 + Y2$
- $A0 = Y3 + Y1$



DECODER

- A decoder is a combinational circuit.
- A decoder accepts a set of inputs that represents a binary number and activates only that output corresponding to the input number. All other outputs remain inactive.
- Fig. 1 shows the block diagram of decoder with 'N' inputs and 'M' outputs.
- There are 2^N possible input combinations, for each of these input combination only one output will be HIGH (active) all other outputs are LOW
- Some decoder have one or more ENABLE (E) inputs that are used to control the operation of decoder.

BLOCK DIAGRAM OF DECODER



Only one output is High for each input

Fig. 1

2 to 4 Line Decoder:

- Block diagram of 2 to 4 decoder is shown in fig. 2
- A and B are the inputs. (No. of inputs =2)
- No. of possible input combinations: $2^2=4$
- No. of Outputs : $2^2=4$, they are indicated by D_0 , D_1 , D_2 and D_3
- From the Truth Table it is clear that each output is “1” for only specific combination of inputs.

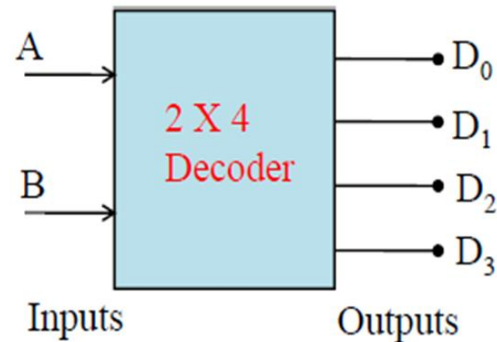


Fig. 2

TRUTH TABLE

INPUTS		OUTPUTS			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

BOOLEAN EXPRESSION:

From Truth Table

$$D_0 = \bar{A} \bar{B}$$

$$D_1 = \bar{A} B$$

$$D_2 = A \bar{B}$$

$$D_3 = AB$$

LOGIC DIAGRAM:

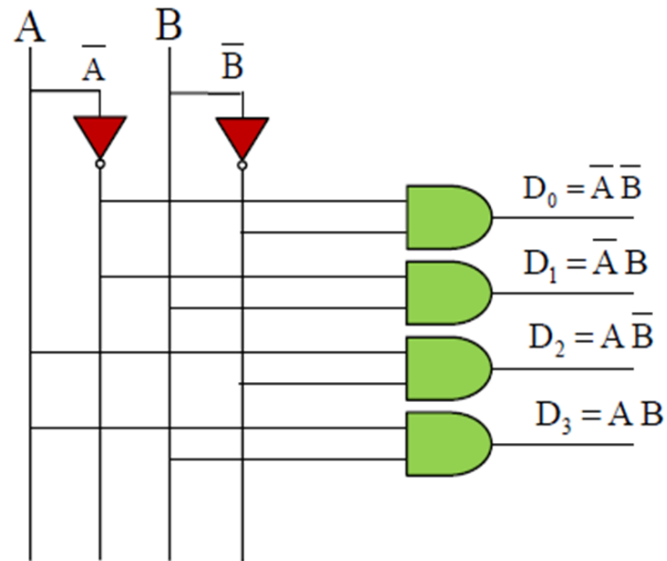
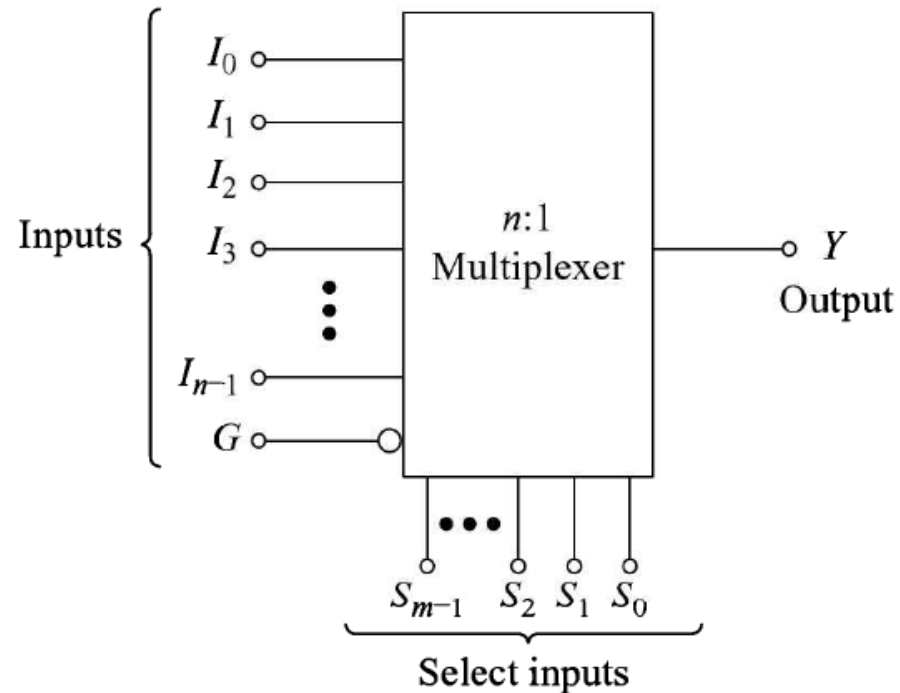


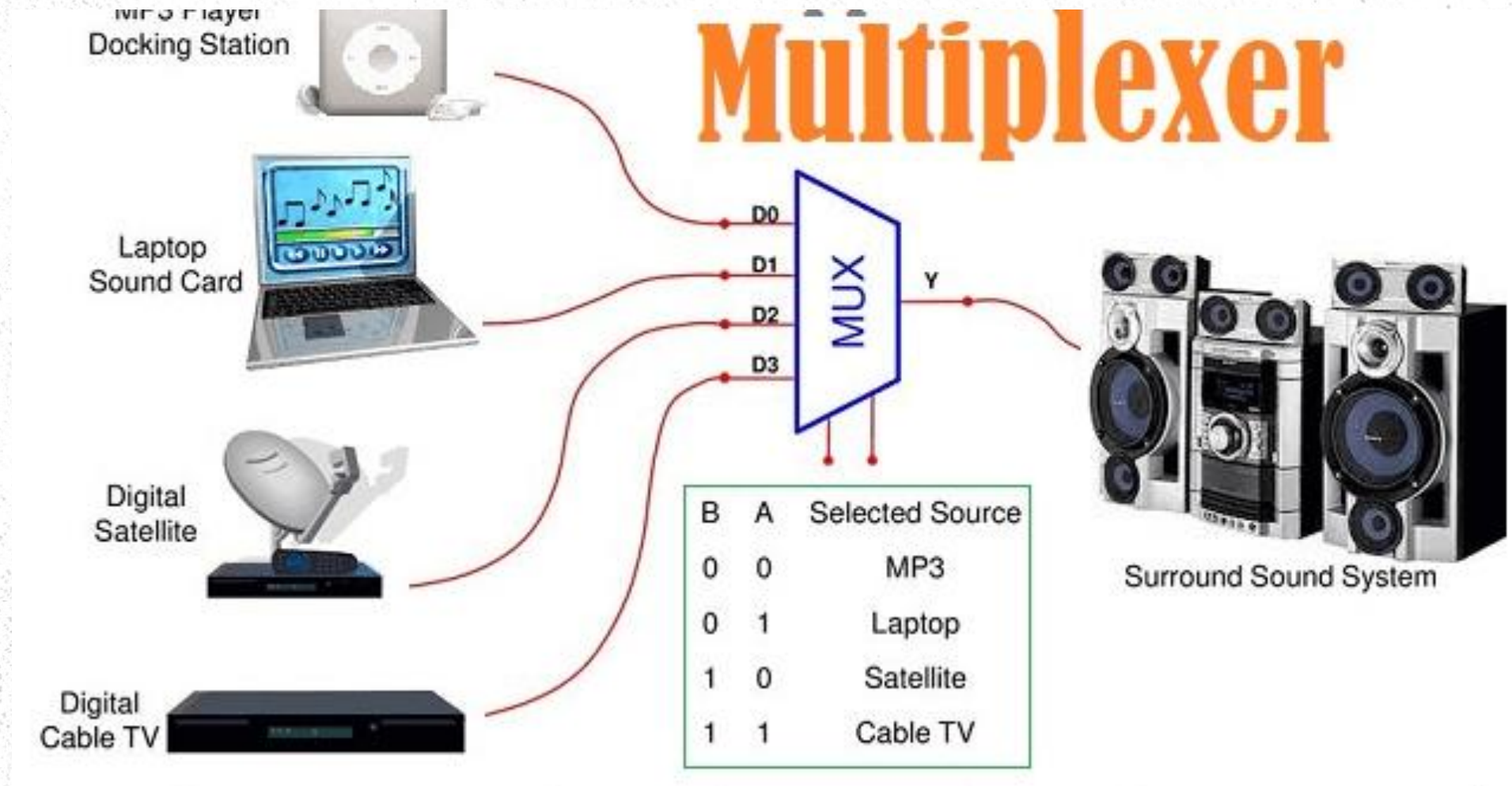
Fig. 3

MULTIPLEXER

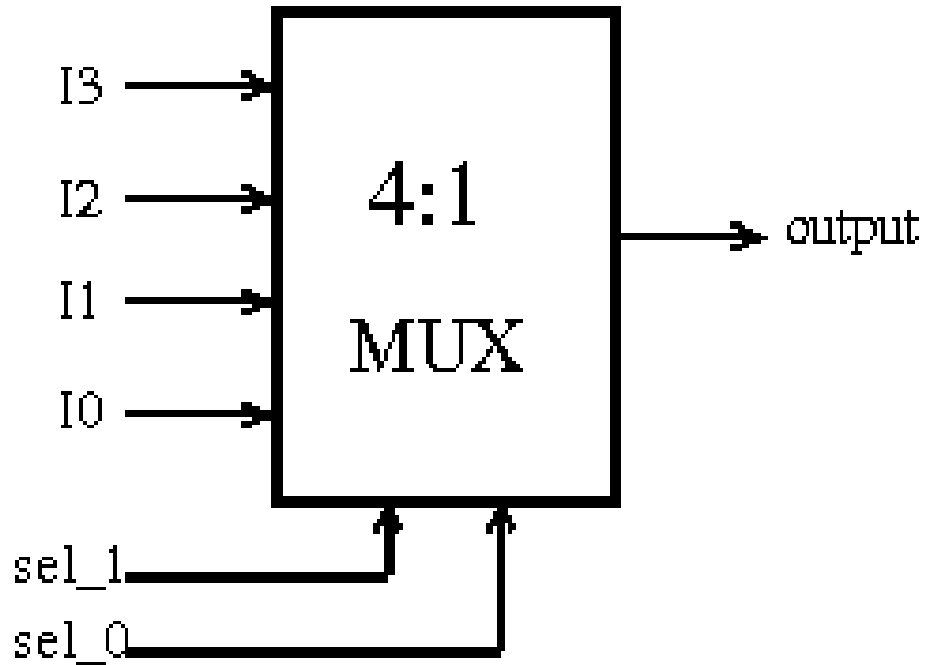
- A Multiplexer is combinational circuit with more than one input and one output.
- Multiplexers are also called as Data Selectors.
- Multiplexers selects any one of the inputs to the output based on the status of the select Lines.
- i.e The output of the Multiplexer depends on the Select lines not on the inputs.



MULTIPLEXER



4:1 Multiplexer



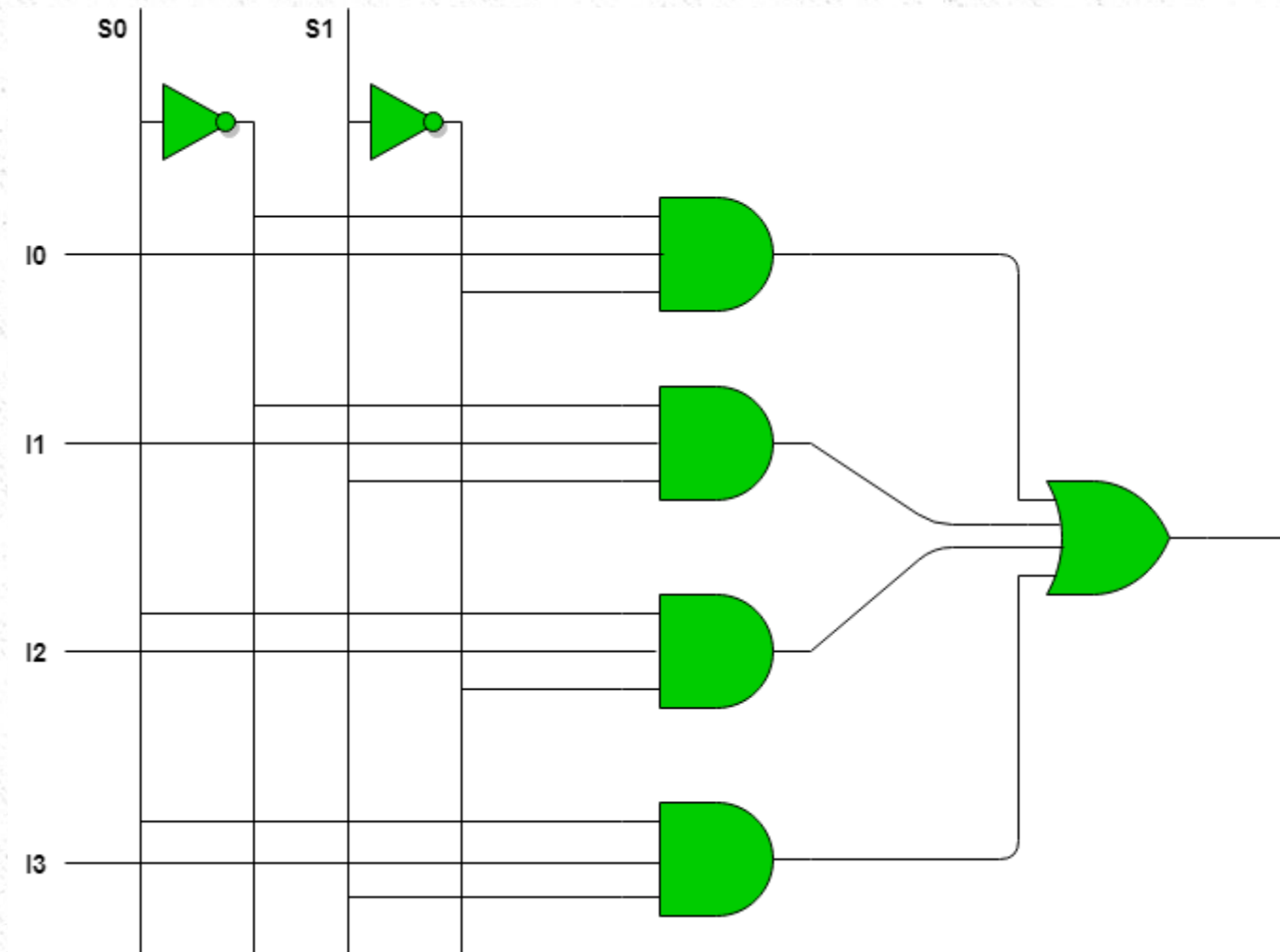
Truth Table

S0	S1	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

So, final equation,

$$Y = S0'.S1'.I0 + S0'.S1.I1 + S0.S1'.I2 + S0.S1.I3$$

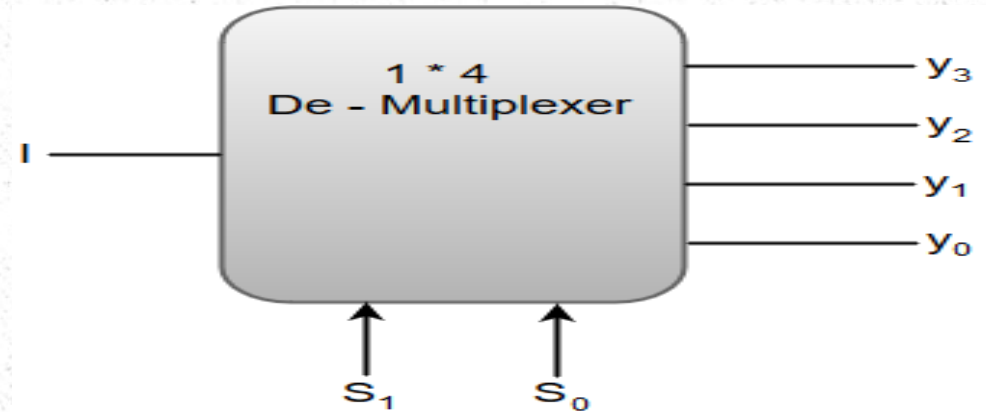
LOGIC CIRCUIT



De-Multiplexers

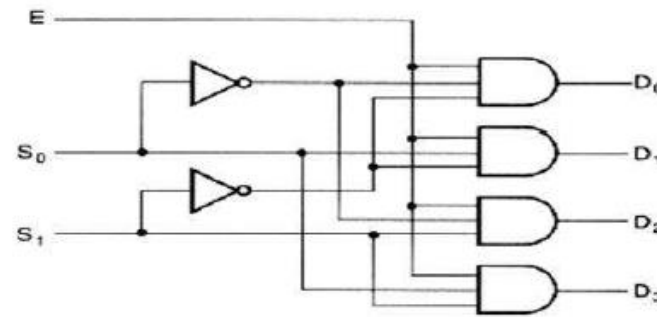
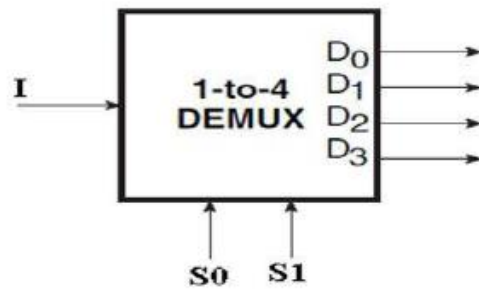
- De-Multiplexers

- A De-multiplexer (De-Mux) can be described as a combinational circuit that performs the reverse operation of a Multiplexer.
- A De-multiplexer has a single input, 'n' selection lines and a maximum of 2^n outputs.
- The following image shows the block diagram of a $1 * 4$ De-multiplexer.



Function Table 1*4

1- to- 4 line demultiplexer



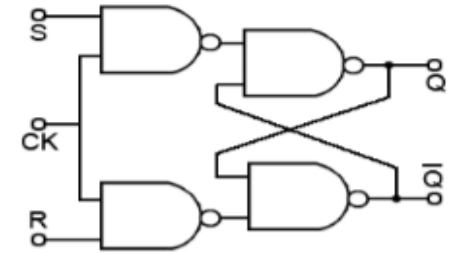
I	Select		O/P			
	S0	S1	D0	D1	D2	D3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Flip Flop

- Flip-flop is a circuit that maintains a state until directed by input to change the state. A basic flip-flop can be constructed using four-NAND or four-NOR gates. **Types of flip-flops:**

1. SR Flip Flop
2. JK Flip Flop
3. D Flip Flop
4. T Flip Flop

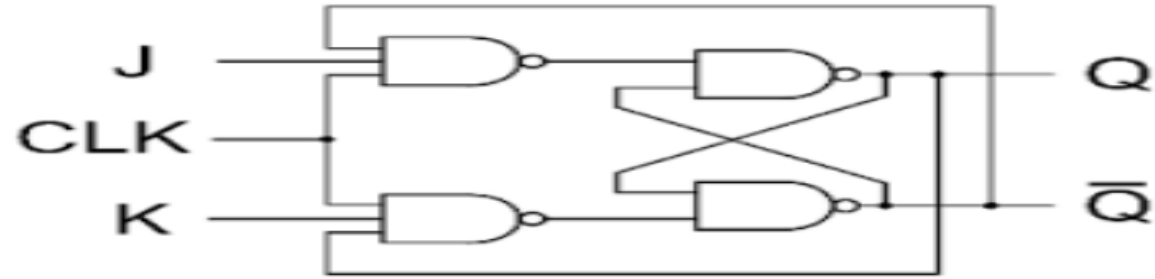
- Logic diagrams and truth tables of the different types of flip-flops
- are as follows:
- S-R Flip Flop :**
- Characteristics Equation for SR Flip Flop: $Q_{N+1} = Q_N R' + SR'$
-



TRUTH TABLE

S	R	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

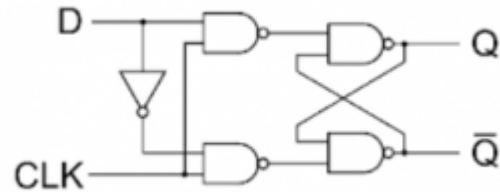
JK FLIP FLOP



TRUTH TABLE

J	K	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

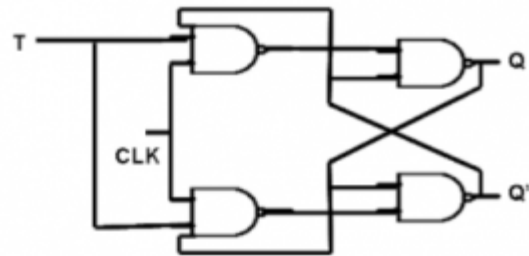
D-Flip Flop



Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

Characteristics Equation for D Flip Flop: $Q_{N+1} = D$









T FLIP FLOP



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

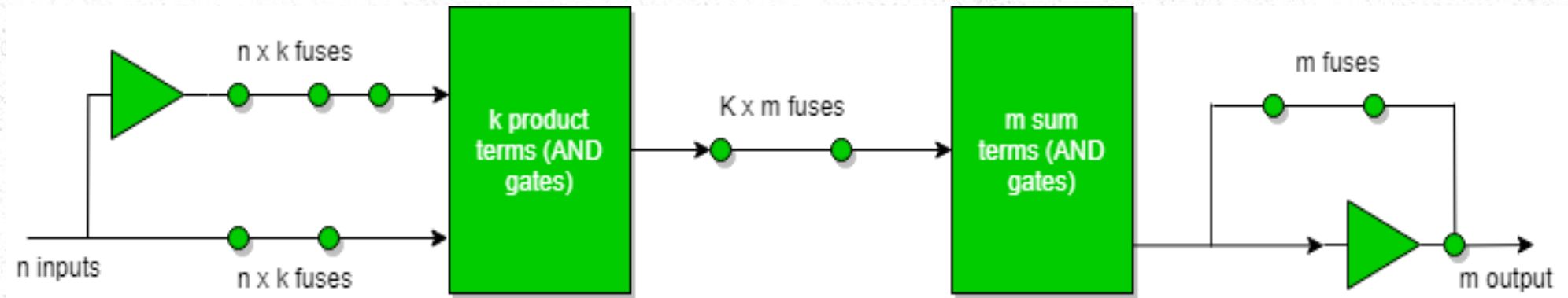
LOGIC GATES



Name	Graphic symbol	Algebraic function	Truth table															
AND		$F = x \cdot y$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	0	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = x + y$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	1
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
Inverter		$F = x'$	<table><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x	F	0	1	1	0									
x	F																	
0	1																	
1	0																	
Buffer		$F = x$	<table><tr><th>x</th><th>F</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	x	F	0	0	1	1									
x	F																	
0	0																	
1	1																	
NAND		$F = (xy)'$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	1	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = (x + y)'$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	0
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	<table><tr><th>x</th><th>y</th><th>F</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Programmable Logic Array(PLA)

- Programmable Logic Array(PLA) is a fixed architecture logic device with programmable AND gates followed by programmable OR gates.
- **Basic block diagram for PLA:**

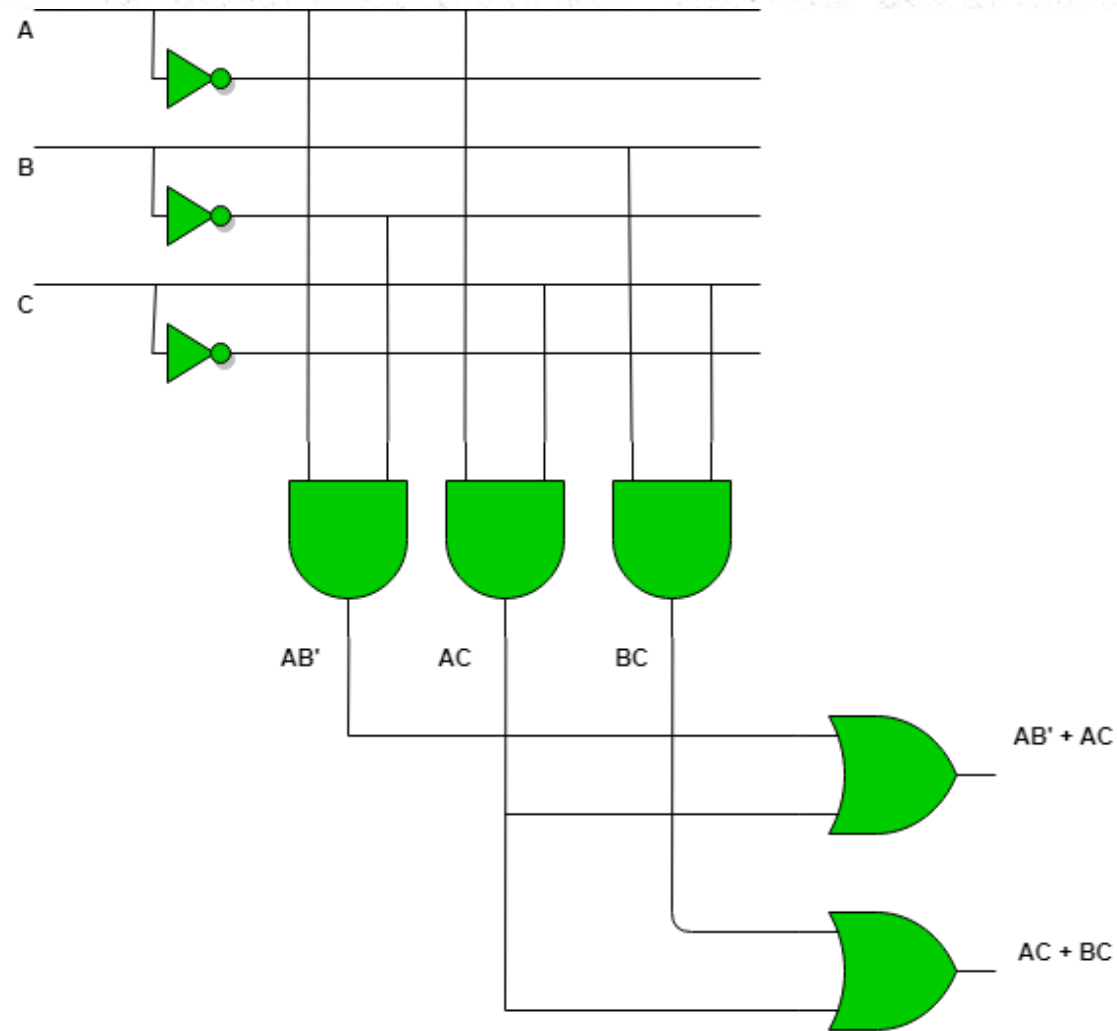


Truth Table

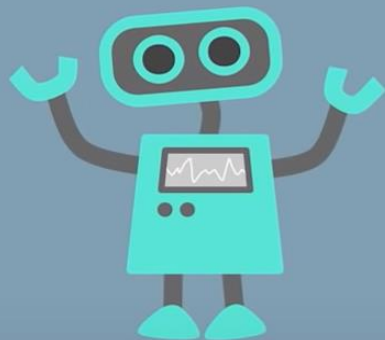
- $F1 = AB'C' + ABC' + ABC$
on simplifying we get : $F1 = AB + AC'$
- $F2 = A'BC + AB'C + ABC$
on simplifying we get: $F2 = BC + AC$

A	B	C	F1	F2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

Circuit Diagram



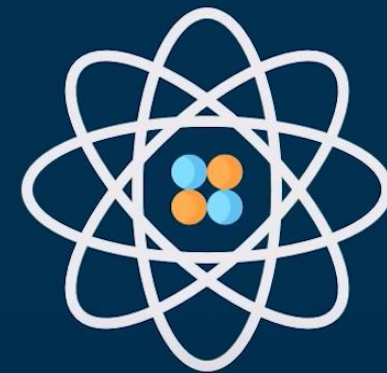
Did You Know?



Registers are fast
Computer Memory



An encoder converts a set
of binary inputs into a
unique binary code.



Summary



Outcomes:

- a. Discuss the theory functionality and basic architecture of CPU
- b. Discuss the Design Issues on the basis of speed, Technology, cost and performance.
- c. Illustrate the different Logic Gates and Minimization of Logic gates.

Thank you