

---

**CSE211**

**Computer Organization and  
Design**

**Lecture : 3**

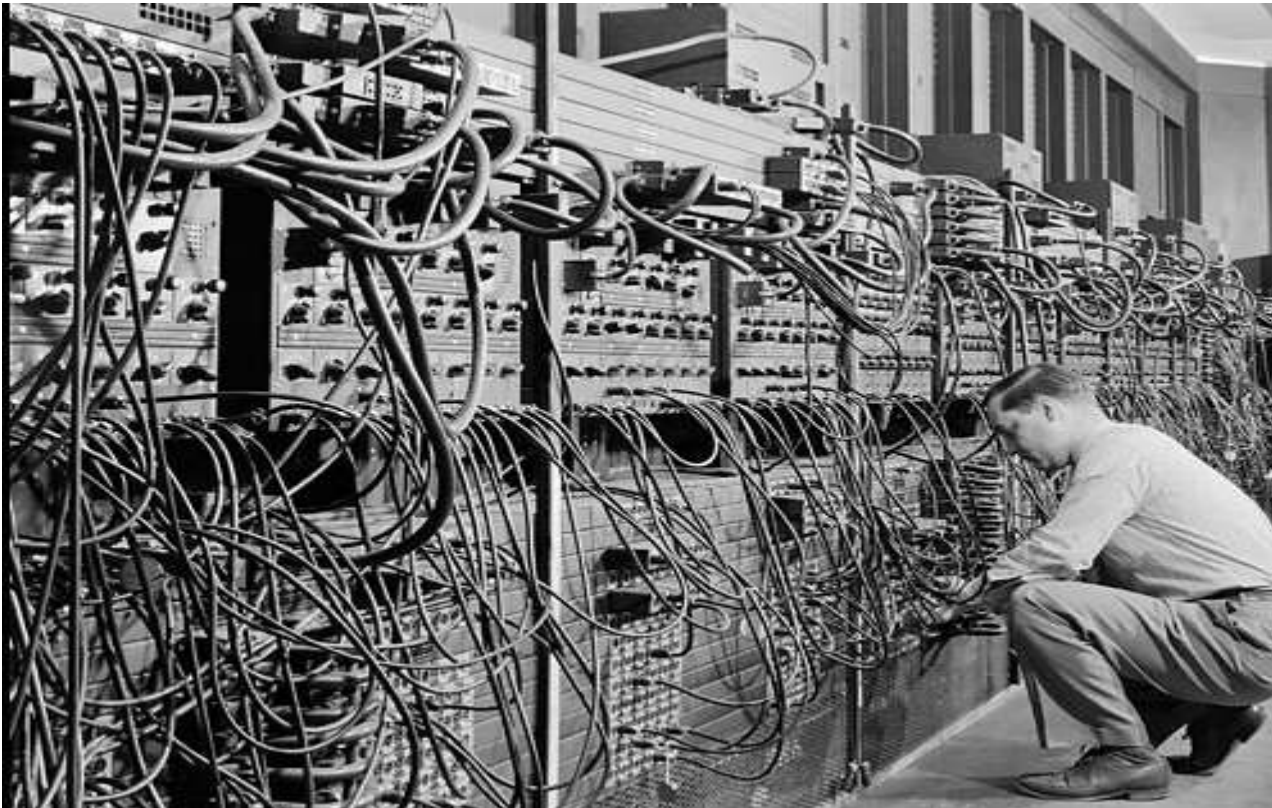
**Tutorial: 1**

**Practical: 0**

**Credit: 4**

# Historical Perspective

## ❖ First generation Computers (1941-1956):



# Characteristics

- Vacuum Tubes
- Magnetic Drums
- Slow Operating Systems
- Production of the heat
- Machine language was used for programming
- First generation computers were unreliable
- They were difficult to program and use

**\*Von Neumann Computers**

## ❖ Second Generation Computers (1956-1963):



# Characteristics

- Use of transistors
- Reliable in comparison to first generation computers
- Smaller size as compared to first generation computers
- Generated less heat as compared to first generation computers
- Consumed less electricity as compared to first generation computers
- Faster than first generation computers
- Still very costly
- **AC required**
- Supported machine and assembly languages

## ❖ Third Generation Computers (1964-1971)



# Characteristics

- **IC used**
- More reliable
- Smaller size
- Generated **less heat**
- Consumed lesser electricity
- Supported high-level language

# ❖ Fourth Generation (1971-2010)





# Characteristics

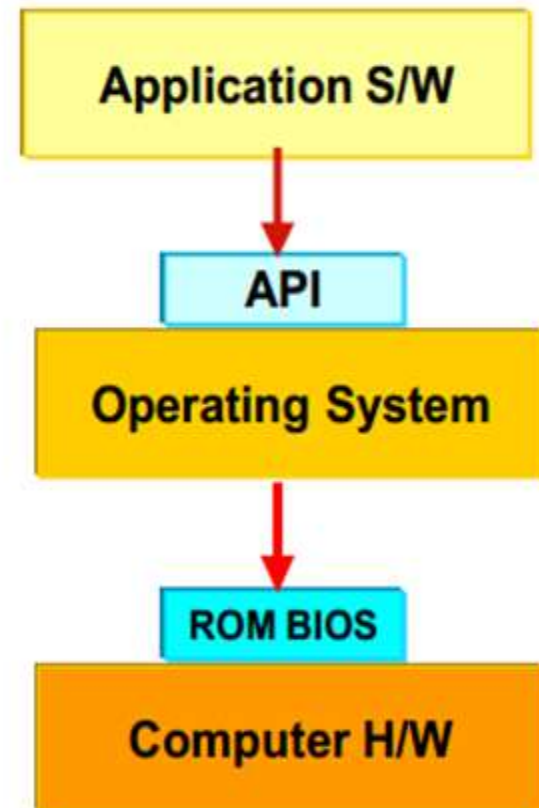
- VLSI technology used
- Very cheap
- Portable and reliable
- Use of PCs
- Very small size

## **2010- : Fifth Generation – Artificial Intelligence**

- Computer devices with artificial intelligence are still in development, but some of these technologies are beginning to emerge and be used such as voice recognition.
- AI is a reality made possible by using parallel processing and superconductors. Leaning to the future, computers will be radically transformed again by quantum computation, molecular and nano technology.
- The essence of fifth generation will be using these technologies to ultimately create machines which can process and respond to natural language, and have capability to learn and organise themselves.

# 1-1 Digital Computers

- Digital – A limited number of discrete value
- Bit – A Binary Digit
- Program – A Sequence of instructions
  
- Computer = H/W + S/W
- Program(S/W)
  - ◆ A sequence of instruction
  - ◆ S/W = Program + Data
    - The data that are manipulated by the program constitute the data base
  - ◆ Application S/W
    - DB, word processor, Spread Sheet
  - ◆ System S/W
    - OS, Firmware, Compiler, Device Driver



# 1-1 Digital Computers

---

## ■ Computer Hardware

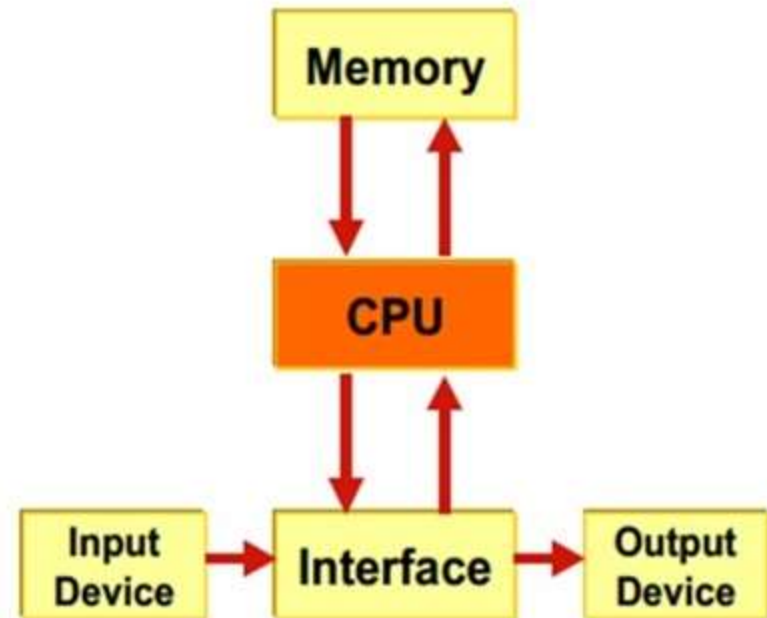
### ◆ CPU

### ◆ Memory

- Program Memory(ROM)
- Data Memory(RAM)

### ◆ I/O Device

- Interface
- Input Device: Keyboard, Mouse, Scanner
- Output Device: Printer, Plotter, Display
- Storage Device(I/O): FDD, HDD, MOD



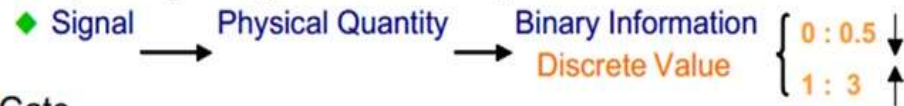
*Figure*     *Block Diagram of a digital Computer*

Logic gates form the building/ elementary blocks of a **Combinational Circuit**.

Flip-flops form the building/ elementary blocks of a **Sequential Circuit**.

## 1-2 Logic Gates

- ADC(Analog to Digital Conversion)







- Gate

- ◆ The manipulation of binary information is done by logic circuit called "gate".





- *Fig. Digital Logic Gates*

- ◆ AND, OR, INVERTER, BUFFER, NAND, NOR, XOR, XNOR

## 1-2 Logic Gates

| Name   | Symbol   | Function                          | Truth Table  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|--------|--|-----------------------------------|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AND    | <div><div>A</div><div>B</div></div> | $X = A \cdot B$<br>or<br>$X = AB$ | <table><tr><th>A</th><th>B</th><th>X</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> | A | B | X | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| A      | B  | X                                 |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0      | 0  | 0                                 |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0      | 1  | 0                                 |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1      | 0  | 0                                 |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1      | 1  | 1                                 |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| OR     | <div><div>A</div><div>B</div></div> | $X = A + B$                       | <table><tr><th>A</th><th>B</th><th>X</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> | A | B | X | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| A      | B  | X                                 |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0      | 0  | 0                                 |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0      | 1  | 1                                 |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1      | 0  | 1                                 |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1      | 1  | 1                                 |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| I      | <div><div>A</div></div>             | $X = A'$                          | <table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>   | A | X | 0 | 1 | 1 | 0 |   |   |   |   |   |   |   |   |   |
| A      | X  |                                   |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0      | 1  |                                   |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1      | 0  |                                   |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| Buffer | <div><div>A</div></div>           | $X = A$                           | <table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>   | A | X | 0 | 0 | 1 | 1 |   |   |   |   |   |   |   |   |   |
| A      | X  |                                   |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0      | 0  |                                   |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1      | 1  |                                   |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

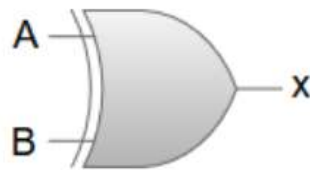
## 1-2 Logic Gates

| Name                                    | Symbol   | Function                                     | Truth Table  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|--|--|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NAND                                    |   | $X = (AB)'$                                  | <table><tr><th>A</th><th>B</th><th>X</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> | A | B | X | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| A                                       | B  | X  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                                       | 0  | 1  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                                       | 1  | 1  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1                                       | 0  | 1  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1                                       | 1  | 0  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| NOR                                     |   | $X = (A + B)'$                               | <table><tr><th>A</th><th>B</th><th>X</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> | A | B | X | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| A                                       | B  | X  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                                       | 0  | 1  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                                       | 1  | 0  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1                                       | 0  | 0  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1                                       | 1  | 0  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| XOR<br>Exclusive OR                     |   | $X = A \oplus B$<br>or<br>$X = A'B + AB'$    | <table><tr><th>A</th><th>B</th><th>X</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> | A | B | X | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| A                                       | B  | X  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                                       | 0  | 0  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                                       | 1  | 1  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1                                       | 0  | 1  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1                                       | 1  | 0  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| XNOR<br>Exclusive NOR<br>or Equivalence |  | $X = (A \oplus B)'$<br>or<br>$X = A'B' + AB$ | <table><tr><th>A</th><th>B</th><th>X</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> | A | B | X | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| A                                       | B  | X  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                                       | 0  | 1  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0                                       | 1  | 0  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1                                       | 0  | 0  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 1                                       | 1  | 1  |  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

## Exclusive-OR/ XOR GATE:

The 'Exclusive-OR' gate is a circuit which will give a high output if one of its inputs is high but not both of them. The XOR operation is represented by an encircled plus sign.

### XOR Gate:



Algebraic Function:  $x = A \oplus B$   
or  
 $x = A'B + AB'$

Truth Table:

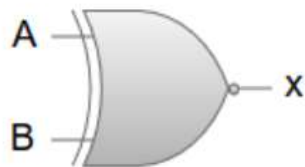
| A | B | x |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



## EXCLUSIVE-NOR/Equivalence GATE:

The 'Exclusive-NOR' gate is a circuit that does the inverse operation to the XOR gate. It will give a low output if one of its inputs is high but not both of them. The small circle represents inversion.

### Exclusive-NOR Gate:



Algebraic Function:  $x = (A \oplus B)'$   
or  
 $x = A'B' + AB$

Truth Table:

| A | B | x |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

- VIDEO LINK FOR BETTER UNDERSTANDING OF LOGIC GATES IN Digital Electronics:

- <https://www.youtube.com/watch?v=dpmnqSOGv4o>

# Combinational Circuits

- In this- output depends only **upon present input.**
- Speed is fast.
- It is designed easy.
- There is no feedback between input and output.
- This is time independent.
- Elementary building blocks: Logic gates
- Used for arithmetic as well as boolean operations.
- Combinational circuits don't have capability to store any state.
- As combinational circuits **don't have clock**, they don't require triggering.
- These circuits do not have any memory element.

Examples – Adder, Subtractor, Encoder, Decoder,  
Multiplexer, Demultiplexer

# Combinational circuits

- Full adders are used in many digital devices, including **calculators and computers**.
- Multiplexer- used in many applications, including data communication and computer memory.
- Combinational circuits can be made using various **logic gates**, such as AND gates, OR gates, and NOT gates.

# Sequential Circuits

- In this output depends upon present as well as past input.
- Speed is slow comparative to combinational circuits.
- It is designed tough as compared to combinational circuits.
- There exists a feedback path between input and output.
- This is time dependent.
- Elementary building blocks: **Flip-flops** Mainly used for storing data.
- Sequential circuits have capability to store any state or to retain earlier state.
- As sequential circuits are clock dependent they need triggering.
- These circuits have memory element.

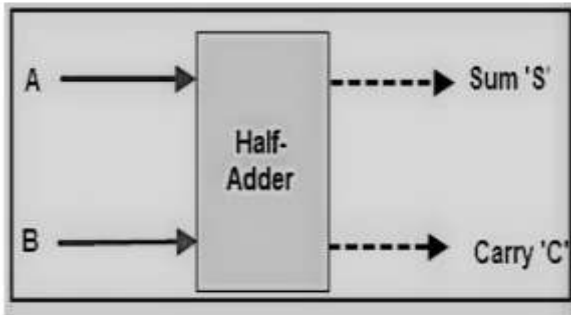
Examples – **Flip-flops, Counters**

## Difference Between Combinational and Sequential Circuit

| Parameters             | Combinational Circuit   | Sequential Circuit  |
|------------------------|---|---|
| Meaning and Definition | It is a type of circuit that generates an output by relying on the input it receives at that instant, and it stays independent of time.                         | It is a type of circuit in which the output does not only rely on the current input. It also relies on the previous ones.   |
| Feedback               | A Combinational Circuit requires no feedback for generating the next output. It is because its output has no dependency on the time instance.                   | The output of a Sequential Circuit, on the other hand, relies on both- the previous feedback and the current input. So, the output generated from the previous inputs gets transferred in the form of feedback. The circuit uses it (along with inputs) for generating the next output. |
| Performance            | We require the input of only the current state for a Combinational Circuit. Thus, it performs much faster and better in comparison with the Sequential Circuit. | In the case of a Sequential Circuit, the performance is very slow and also comparatively lower. Its dependency on the previous inputs makes the process much more complex.  |
| Complexity             | It is very less complex in comparison. It is because it basically lacks implementation of feedback.   | This type of circuit is always more complex in its nature and functionality. It is because it implements the feedback, depends on previous inputs and also on clocks.   |
| Elementary Blocks      | Logic gates form the building/ elementary blocks of a Combinational Circuit.  | Flip-flops form the building/ elementary blocks of a Sequential Circuit.  |
| Operation              | One can use these types of circuits for both- Boolean as well as Arithmetic operations.   | You can mainly make use of these types of circuits for storing data.  |

# Half adder, Full adder- combinational logic circuits

- add two binary digits
- provides the output along with a carry value (if any).
- designed by connecting an EX-OR gate and one AND gate.



HA Functional Diagram

The 2-bit **half adder truth table** is as below:

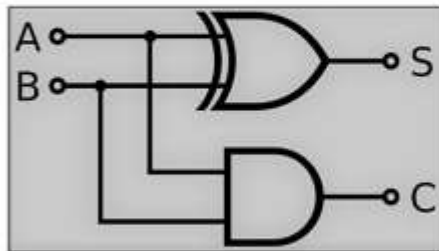
| INPUTS |   | OUTPUTS |       |
|--------|---|---------|-------|
| A      | B | SUM     | CARRY |
| 0      | 0 | 0       | 0     |
| 0      | 1 | 1       | 0     |
| 1      | 0 | 1       | 0     |
| 1      | 1 | 0       | 1     |

Half Adder Truth Table

The simplest expression uses the exclusive OR function:

$$\text{Sum} = A \text{ XOR } B$$

$$\text{Carry} = A \text{ AND } B$$



HA Logical Diagram

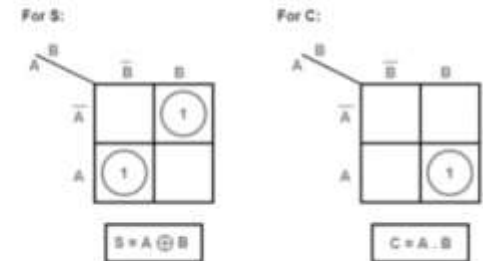
$$0+0=0$$

$$0+1=1$$

$$1+0=1$$

$$1+1=10$$

The half adder K-map is



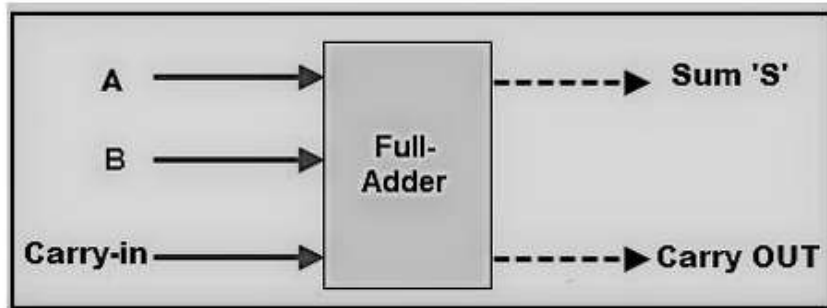
HA K-Map

And an equivalent expression in terms of the basic AND, OR, and NOT is:

$$\text{SUM} = A \cdot B + A \cdot B'$$



# Full Adder

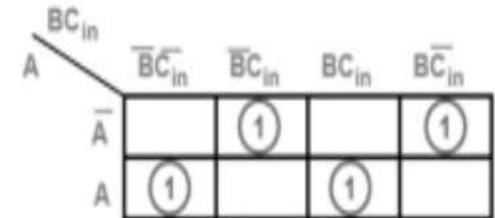


Full Adder Functional Diagram

| INPUTS |   |      | OUTPUT |   |
|--------|---|------|--------|---|
| A      | B | C-IN | C-OUT  | S |
| 0      | 0 | 0    | 0      | 0 |
| 0      | 0 | 1    | 0      | 1 |
| 0      | 1 | 0    | 0      | 1 |
| 0      | 1 | 1    | 1      | 0 |
| 1      | 0 | 0    | 0      | 1 |
| 1      | 0 | 1    | 1      | 0 |
| 1      | 1 | 0    | 1      | 0 |
| 1      | 1 | 1    | 1      | 1 |

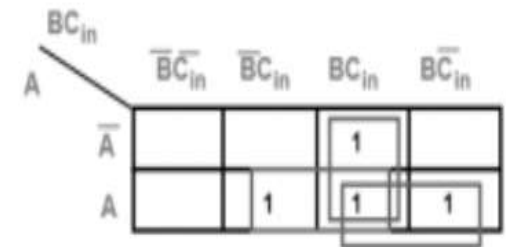
FA Truth Table

For S:



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

For C<sub>in</sub>:

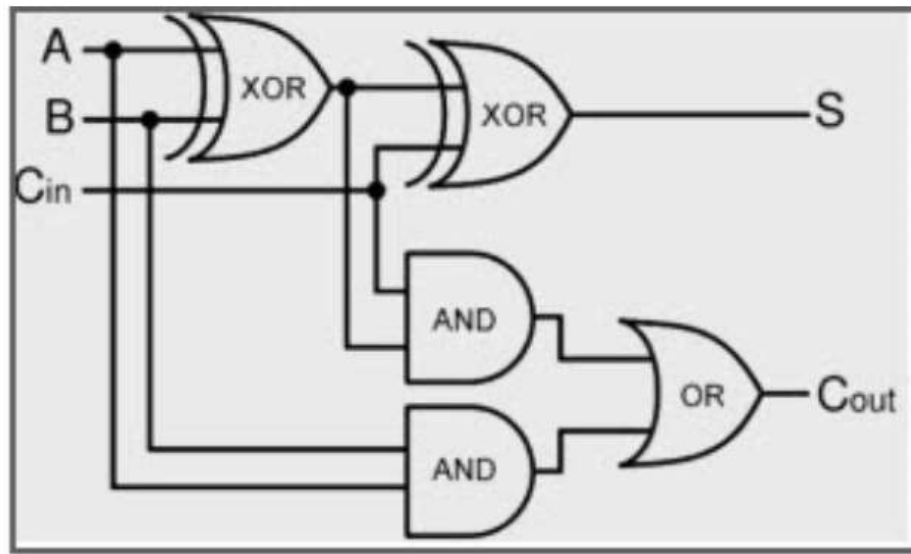


$$C_{out} = AB + BC_{in} + C_{in}A$$

FA K-Map

# Full Adder

- In FA- The SUM 'S' is produced in two steps:
  - By XORing the provided inputs 'A' and 'B'
  - The result of A XOR B is then XOR with the C-IN



*Full Adder Logical Diagram*

# Flip Flops- a type of sequential circuit

- a circuit with two stable states
- used to store binary data. (a storage element)
- used in computers, communications, and many other types of systems.

# Types of flip flops

- There are basically 4 types of flip-flops:
- SR Flip-Flop
- JK Flip-Flop
- D Flip-Flop
- T Flip-Flop

- SR-
  - When S is active- o/p Q is high. Q' is low
- JK- an improvement on the SR flip-flop. Now, S=R=1 is not a problem.

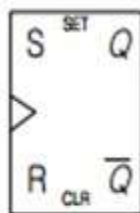
# 1-6 Flip-Flops

Combinational Circuit = Gate  
Sequential Circuit = Gate + F/F

## ■ Flip-Flop

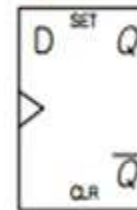
- ◆ The *storage elements* employed in clocked *sequential circuit*
- ◆ A binary cell capable of storing one bit of information

## ■ SR(*Set/Reset*) F/F



| S | R | Q(t) | Q(t+1)        |
|---|---|------|---------------|
| 0 | 0 | Q(t) | no change     |
| 0 | 1 | 0    | clear to 0    |
| 1 | 0 | 1    | set to 1      |
| 1 | 1 | ?    | Indeterminate |

## ■ D(*Data*) F/F

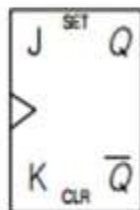


| D | Q(t) | Q(t+1)     |
|---|------|------------|
| 0 | 0    | clear to 0 |
| 1 | 1    | set to 1   |

- ◆ "no change" condition

- 1) Disable Clock
- 2) Feedback output into input

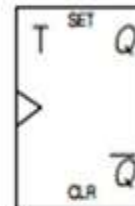
## ■ JK(*Jack/King*) F/F



| J | K | Q(t) | Q(t+1)     |
|---|---|------|------------|
| 0 | 0 | Q(t) | no change  |
| 0 | 1 | 0    | clear to 0 |
| 1 | 0 | 1    | set to 1   |
| 1 | 1 | Q(t) | Complement |

- ◆ JK F/F is a refinement of the SR F/F
- ◆ The indeterminate condition of the SR type is defined in complement

## ■ T(*Toggle*) F/F

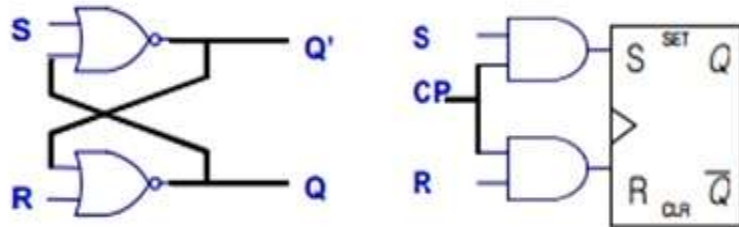


| T | Q(t)  | Q(t+1)     |
|---|-------|------------|
| 0 | Q(t)  | no change  |
| 1 | Q'(t) | Complement |

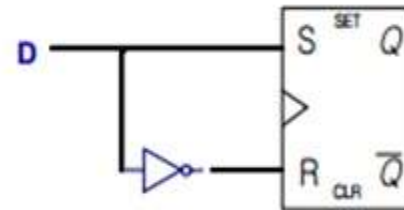
- ◆ T=1(J=K=1), T=0(J=K=0)

## 1-6 Flip-Flops

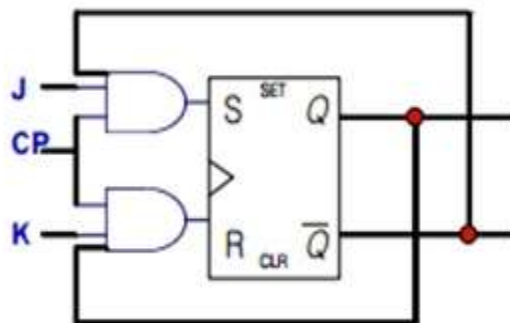
### ■ SR(*Set/Reset*) F/F



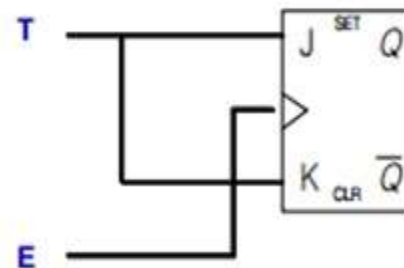
### ■ D(*Data*) F/F



### ■ JK(*Jack/King*) F/F

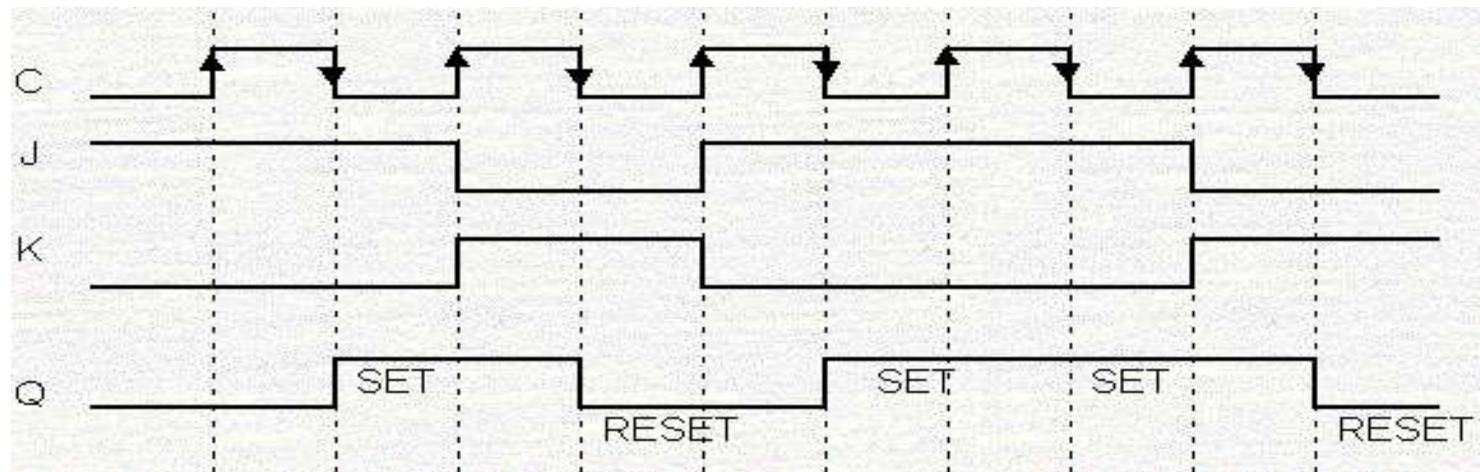
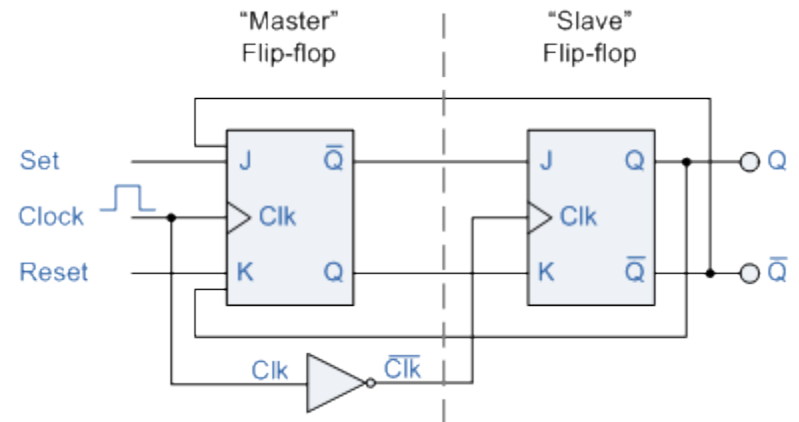
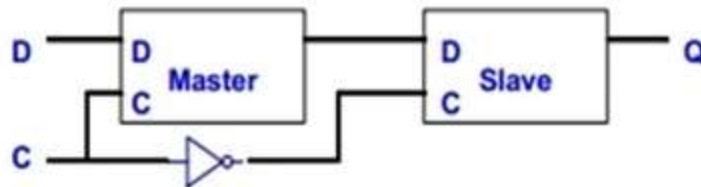


### ■ T(*Toggle*) F/F



# 1-6 Flip-Flops

## ■ Master – Slave D(Data) F/F





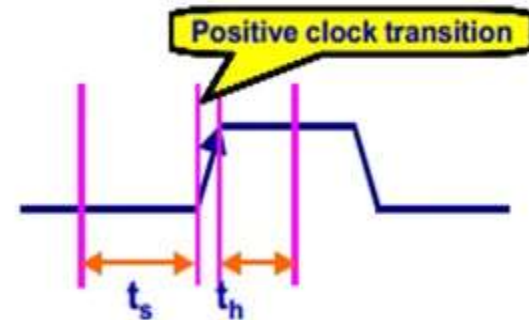


# 1-6 Flip-Flops

## ■ Edge-Triggered F/F

### ◆ State Change : *Clock Pulse*

- Rising Edge(positive-edge transition) 
- Falling Edge(negative-edge transition) 



### ◆ Setup time(20ns)

- minimum time that D input must remain at constant value before the transition.

### ◆ Hold time(5ns)

- minimum time that D input must not change after the positive transition.

### ◆ Propagation delay(max 50ns)

- time between the clock input and the response in Q

### ◆ Master-Slave F/F

# Integrated Circuits

---

An IC is a small silicon semiconductors crystal called chip containing the electronic components for digital gates.

- Various gates are interconnected inside chip to form required circuit.
- Chip is mounted in ceramic/plastic container connected to external pin

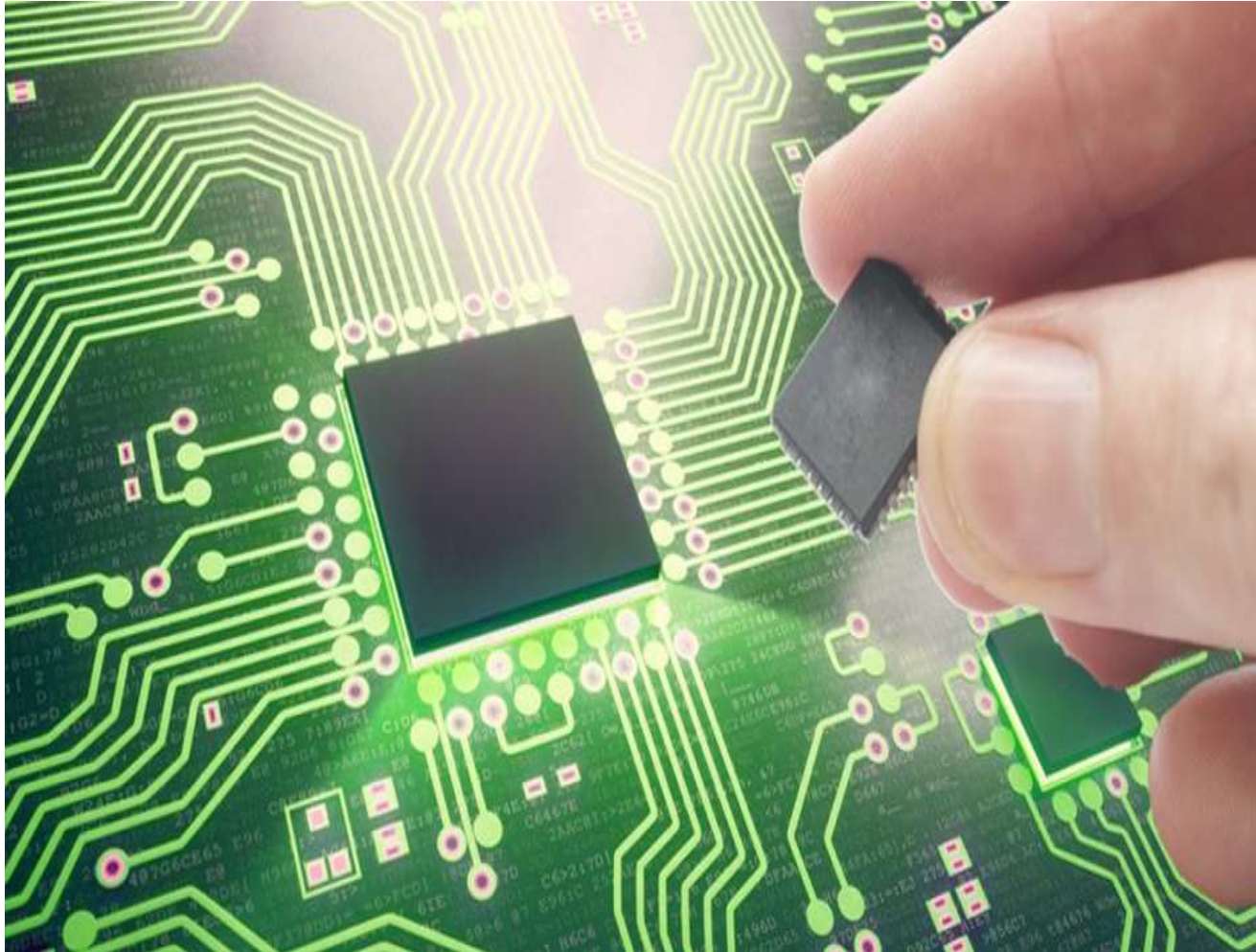
**Small scale Integration (SSI)** : less than 10 gates

**Medium Scale Integration(MSI)** : between 10 to 200 gates  
(decoders, adders, registers)

**Large Scale Integration(LSI)** : between 200 and few thousands gates  
( Processors, Memory Chips)

**Very Large Scale Integration (VLSI)** : Thousands of gate within single package ( Large Memory Arrays, Complex Microcomputer Chips)

# Very Large Scale Integration (VLSI)



- Simple Flip Flop Circuit
  - <https://www.instructables.com/Simple-Flip-Flop-Circuit/>
  - The Flip Flop circuit is ready! at this link

# More about FF.....

- flip-flop is an edge-triggered
- latch is a level-triggered type (level 1 or 0).
- edge triggering is preferred, it is good for clocks

## EDGE TRIGGERING

Type of triggering that allows a circuit to become active at the positive edge or the negative edge of the clock signal

An event occurs at the rising edge or falling edge

Flip flops are edge triggered

## LEVEL TRIGGERING

Type of triggering that allows a circuit to become active when the clock pulse is on a particular level

An event occurs during the high voltage level or low voltage level

Latches are level triggered

# FF/Latch

| Parameter                   | Flip-Flop   | Latch  |
|-----------------------------|---|--|
| Basic Principle             | Flip-flop utilizes an edge triggering approach.   | Latch follows a level triggering approach.   |
| Clock Signal                | The clock signal is present.  | The clock signal is absent.  |
| Designed Using              | You can design it using Latches along with a clock.   | You can design it using Logic gates.   |
| Type of Operation Performed | Flip-flop performs Synchronous operations.  | Latch performs Asynchronous operations.  |
| Types                       | J-K, S-R, D, and T Flip-flops.  | J-K, S-R, D, and T Latches.  |
| Uses                        | They constitute the building blocks of many sequential circuits such as counters.                                       | Users can utilize these for designing sequential circuits. But they are still not generally preferred. |
| Input and Output            | A flip-flop checks the inputs. It only changes the output at times defined by any control signal like the clock signal. | The latch responds to the changes in inputs continuously as soon as it checks the inputs.              |
|                             | A flip-flop is synchronous.   | A latch is asynchronous. It  |



- A latch is **an electronic device that changes its output immediately on the basis of the applied input**. One can use it to store either 0 or 1 at a specified time. A latch contains two inputs- SET and RESET, and it also has two outputs. They complement each other. One can use a latch for storing one bit of data.



# 1-1 Digital Computers

---

- 3 different point of view(Computer Hardware)
  - ◆ Computer Organization
    - H/W components operation/connection
  - ◆ Computer Design
    - H/W Design/Implementation
  - ◆ Computer Architecture
    - Structure and behavior of the computer as seen by the user
    - Information format, Instruction set, memory addressing, CPU, I/O, Memory
- ISA(Instruction Set Architecture)
  - ◆ the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls, the logic design, and the physical implementation.
    - Amdahl, Blaaw, and Brooks(1964)