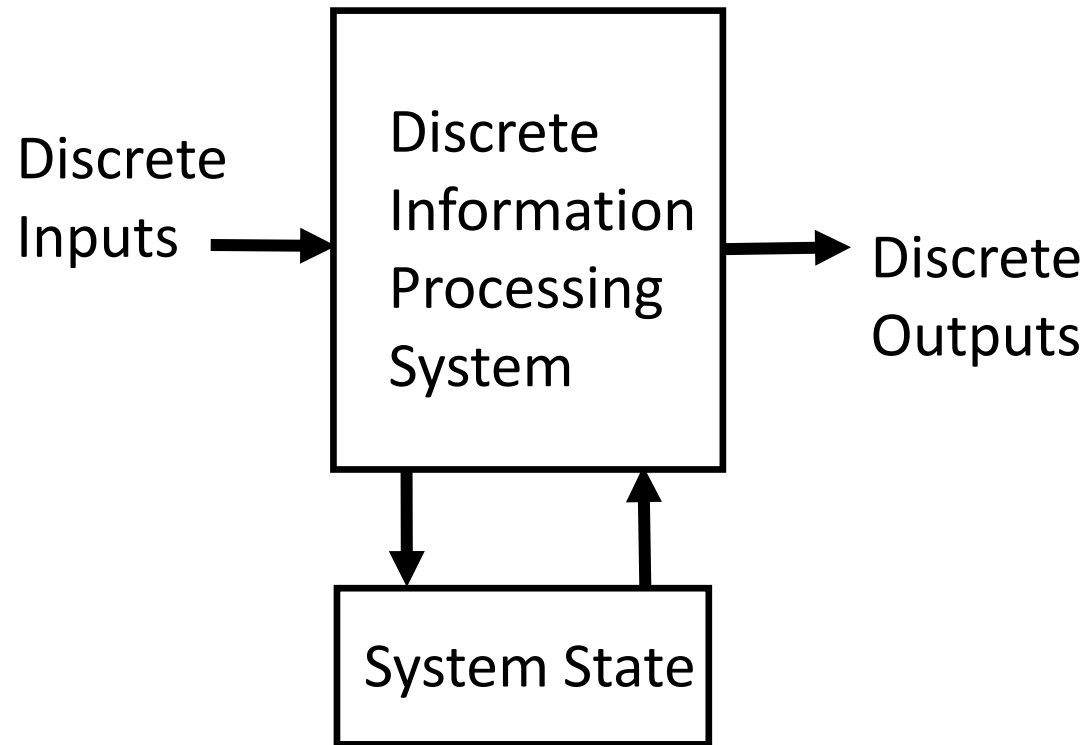


Computer System Architecture

Unit 1

- Digital computers = Digital system + Computational tasks.
- **Digital system takes a set of discrete information inputs and discrete internal information (system state) and generates a set of discrete information outputs.**



- Digital system takes discrete value , while analog system takes continuous values.
- Square waves vs sine waves
- E.g.- computer, CD, Analog- radio voice.
- Digital computer uses binary system.
- Its has 2 digit : 0,1
- Digit is called bit and any info is represented in forms of group of bits.
- Bits not only used to represent number but also alphabets, decimal digit, instruction set.
- Computer System = Hardware +software
- Hardware= electronic and electromechanical components
- Software= programs

- Why to study CSA:

Since program execution requires direct hardware involvement(faster execution) , it is good to have knowledge about hardware operation csa.

- Computer Architecture(CA) vs organization(CO) vs design(CD)

CA

- CA deals with instruction set architecture
- Mainly useful for machine or assembly language programmer.
- Thus, it enable programmers to understand instruction set architecture, different registers and their role, way of accessing and operating data and i/o operations
- Thus deals with external view of computer operation.

CO

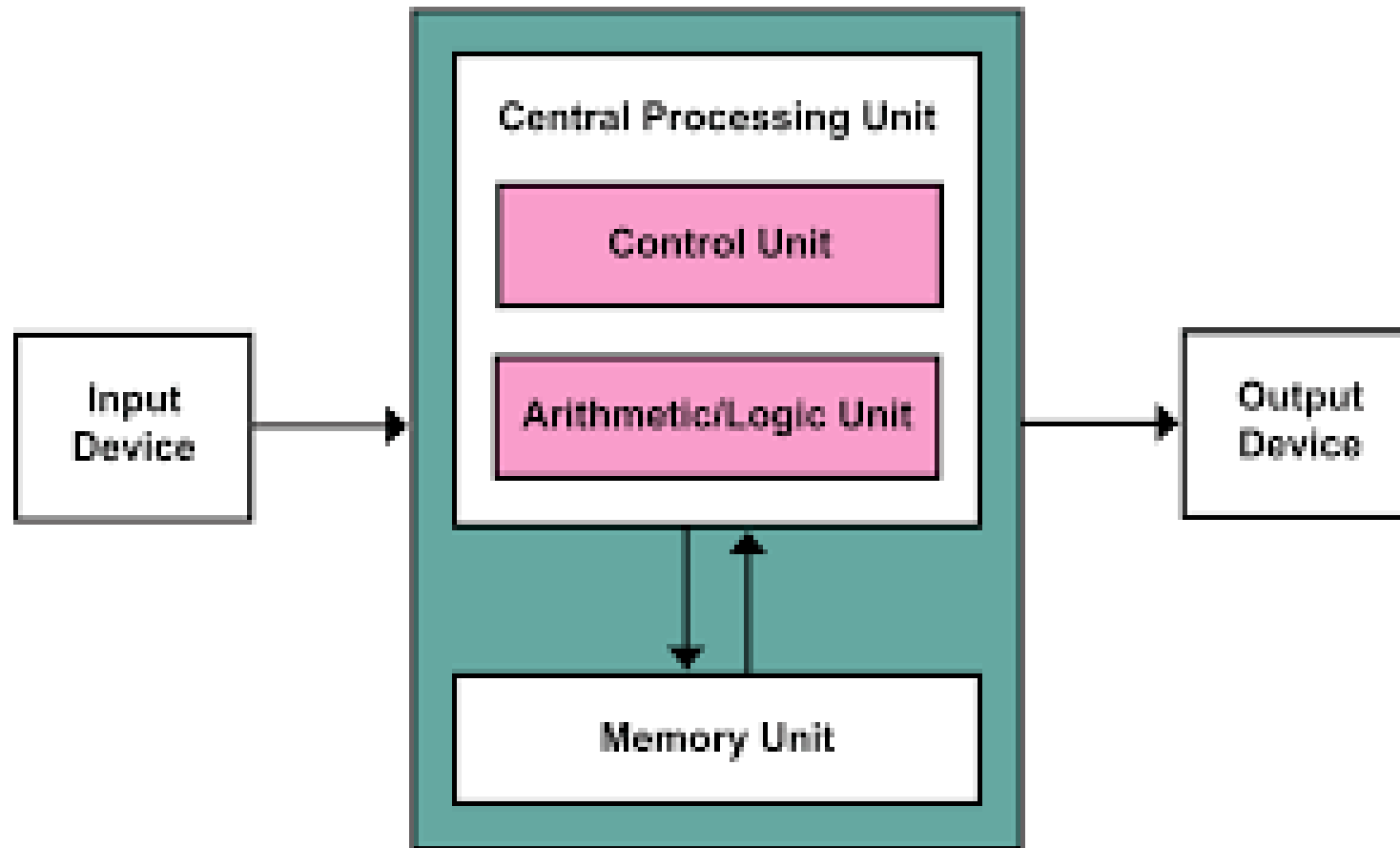
- Way which hardware components connect and operate.
- Deals with internal view of computer working
- How different components such as processor, memory, I/O devices work

- Processor- Instruction execution and cycle
- Memory- cache, primary and secondary organisation

CD

- It deals with designing and use of different types of hardware.
- Connection among hardware

Von Neumann Computers



- Earlier one instruction at a time of program is executed
- An operator is required, that used to set up instruction by flipping switches.
- Von neumann proposed that instruction can be stored and encoded just like data in memory.
- During program execution, instruction are fetched , decoded and executed.
- Control unit of processor perform the above task.
- Thus with stored instruction and control, no need of operator
- Hence faster execution of program is seen.
- Whole decoding to output generation process is known as instruction cycle.
- Upto 1980 architecture works fine
- Shortcoming of Von Neumann – at a time either instruction or data is fetched. Thus no parallel execution of instructions. (bottleneck problem)

Basic Organization of computer

- Processor
 - ALU, Registers, Control unit
- Memory
 - Primary, Secondary
- Output devices-Monitor, printer
- Input devices- Keyboard, mouse
- Bus
 - Set of wires- control, address and data

Evolution of Computer

| Generations of Computer | Time-Period | Evolving Hardware |
|-------------------------|------------------|-------------------------------|
| First Generation | 1940s – 1950s | Vacuum Tube Based |
| Second Generation | 1950s – 1960s | Transistor Based |
| Third Generation | 1960s – 1970s | Integrated Circuit Based |
| Fourth Generation | 1970s – Present | Microprocessor Based |
| Fifth Generation | Present – Future | Artificial Intelligence Based |



| Characteristics | Components |
|----------------------------------|--|
| Main electronic component | Vacuum tube. |
| Programming language | Machine language. |
| Main memory | Magnetic tapes and magnetic drums. |
| Input/output devices | Paper tape and punched cards. |
| Speed and size | Very slow and very large (often taking up an entire room). |
| Examples of the first generation | IBM 650, IBM 701, ENIAC, UNIVAC1, etc |

| Characteristics | Components |
|-----------------------------------|--|
| Main electronic component | Transistor. |
| Programming language | Machine language and assembly language. |
| Memory | Magnetic core and magnetic tape/disk. |
| Input/output devices | Magnetic tape and punched cards. |
| Power and size | Smaller in size, had low power consumption, and generated less heat (in comparison with the first-generation computers). |
| Examples of the second generation | PDP-8, IBM1400 series, IBM 7090 and 7094, UNIVAC 1107, CDC 3600, etc. |

| Characteristics | Components |
|----------------------------------|---|
| Main electronic component | Integrated circuits (ICs). |
| Programming language | High-level language. |
| Memory | Large magnetic core, magnetic tape/disk. |
| Input/output devices | Magnetic tape, monitor, keyboard, printer, etc. |
| Examples of the third generation | IBM 360, IBM 370, PDP-11, NCR 395, B6500, UNIVAC 1108, etc. |

| Characteristics | Components |
|--------------------------------------|---|
| Main electronic component | Very-large-scale integration (VLSI) and the microprocessor (VLSI has thousands of transistors on a single microchip). |
| Memory | semiconductor memory (such as RAM , ROM , etc.). |
| Input/output devices | pointing devices, optical scanning, keyboard, monitor, printer, etc. |
| Examples of the fourth generation | IBM PC, STAR 1000, APPLE II, Apple Macintosh, Alter 8800, etc. |

| Characteristics | Components |
|---------------------------------|--|
| Main electronic component | Based on artificial intelligence, uses the Ultra Large-Scale Integration (ULSI) technology and parallel processing method (ULSI has millions of transistors on a single microchip and the Parallel processing method use two or more microprocessors to run tasks simultaneously). |
| Language | Understand natural language (human language). |
| Size | Portable and small in size. |
| Input/output device | Trackpad (or touchpad), touchscreen, pen, speech input (recognize voice/speech), light scanner, printer, keyboard, monitor, mouse, etc. |
| Example of the fifth generation | Desktops, laptops, tablets, smartphones, etc. |

Number System and Conversion

| | General | Decimal | Binary |
|-----------------|-----------------------|-------------------|-------------------|
| Radix (Base) | r | 10 | 2 |
| Digits | $0 \Rightarrow r - 1$ | $0 \Rightarrow 9$ | $0 \Rightarrow 1$ |
| Powers of Radix | 0 | r^0 | 1 |
| | 1 | r^1 | 2 |
| | 2 | r^2 | 4 |
| | 3 | r^3 | 8 |
| | 4 | r^4 | 16 |
| | 5 | r^5 | 32 |
| | -1 | r^{-1} | 0.5 |
| | -2 | r^{-2} | 0.25 |
| | -3 | r^{-3} | 0.125 |
| | -4 | r^{-4} | 0.0625 |
| | -5 | r^{-5} | 0.03125 |

Commonly occur Bases

| Name | Base | Digits |
|-------------|------|-------------------|
| Binary | 2 | 0,1 |
| Decimal | 10 | 0-9 |
| Octal | 8 | 0-7 |
| Hexadecimal | 16 | 0-9, A, B,C,D,E,F |

Decimal to Binary Conversion

- $(112) = (1110000)_2$
- For fractional decimal multiply by 2 and append the integer part
- Repeat the step until fractional part becomes zero.
- Converting 0.6875 as fractional part:
 - $0.6875 * 2 = 1.3750$ int = 1
 - $0.3750 * 2 = 0.7500$ int = 0
 - $0.7500 * 2 = 1.5000$ int = 1
 - $0.5000 * 2 = 1.0000$ int = 1

| Division | Remainder (R) |
|----------------|---------------|
| $112 / 2 = 56$ | 0 |
| $56 / 2 = 28$ | 0 |
| $28 / 2 = 14$ | 0 |
| $14 / 2 = 7$ | 0 |
| $7 / 2 = 3$ | 1 |
| $3 / 2 = 1$ | 1 |
| $1 / 2 = 0$ | 1 |

Decimal to Octal and Hexadecimal

- Divide decimal number by 8 and 16.

| Division | Remainder (R) |
|----------------|---------------|
| $210 / 8 = 26$ | 2 |
| $26 / 8 = 3$ | 2 |
| $3 / 8 = 0$ | 3 |

| Multiplication | Resultant integer part |
|--|------------------------|
| $0.140869140625 \times 8 = 0.12695313$ | 1 |
| $0.12695313 \times 8 = 0.01562504$ | 1 |
| $0.01562504 \times 8 = 0.12500032$ | 0 |
| $0.12500032 \times 8 = 0.00000256$ | 1 |
| $0.00000256 \times 8 = 0.000020544$ | 0 |
| and so on | |

| Division | Remainder (R) |
|-----------------|---------------|
| $540 / 16 = 33$ | $12 = C$ |
| $33 / 16 = 2$ | 1 |
| $2 / 16 = 0$ | 2 |
| $0 / 16 = 0$ | 0 |

| Multiplication | Resultant integer part |
|---------------------------------|------------------------|
| $0.06640625 \times 16 = 1.0625$ | 1 |
| $0.0625 \times 16 = 1.0$ | 1 |
| $0 \times 16 = 0.0$ | 0 |

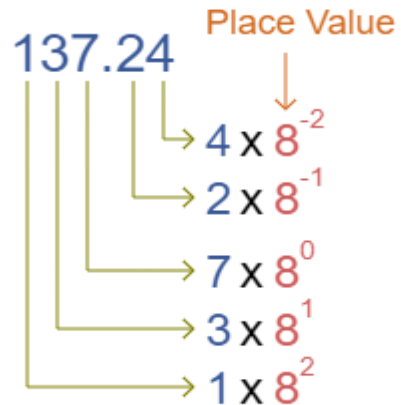
Binary, octal and hexadecimal to Decimal

- $111001_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 57_{10}$

.....

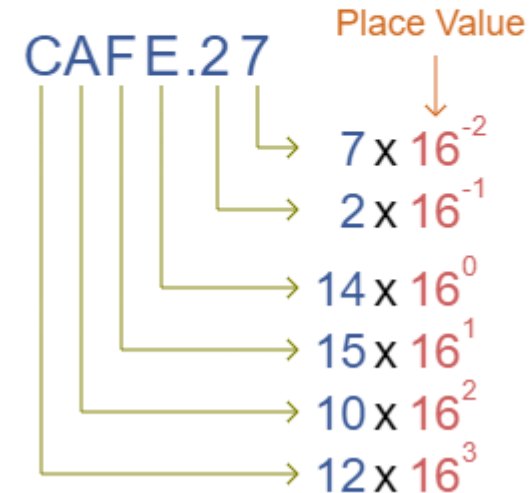
$$(\mathbf{137.24})_8 = (\mathbf{95.3125})_{10}$$

DESCRIPTIONS



$$(\mathbf{CAFE.27})_{16} = (\mathbf{51966.15234375})_{10}$$

DESCRIPTIONS



Binary to Octal and Hexadecimal

- Group 3 bit starting from LSB to MSB and so on and convert them to equivalent octal digit
- Group 4 bit starting from LSB to MSB and so on and convert them to equivalent octal digit

- Eg 1: $(101010101)_2$
 $= (101)(010)(101)$
 $= (525)_8$

Eg 2: $(101010101)_2$
 $= (1)(0101)(0101)$
 $= (155)_{16}$

- Octal to hexadecimal : convert octal to equivalent binary and then group in 4 bit

Moore's Law

- **Moore's law** is the observation that the number of transistors in an integrated circuit (IC) doubles about every two years.
- In 1965, Gordon E. Moore, the co-founder of Intel gives this law.
- As a results that we can expect the speed and capability of our computers to increase every two years because of this, yet we will pay less for them.

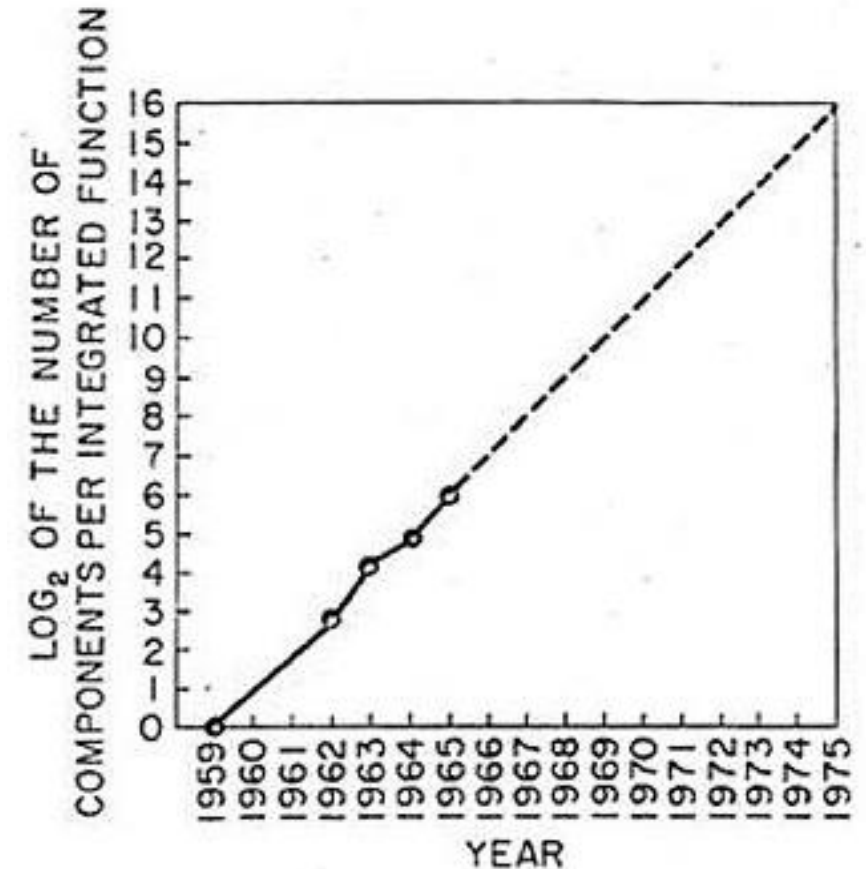


Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.

Memory Organization

- In the computer system, we need computer memory to store various types of data like text, images, video, audio, documents, etc. It is an enhancement to organize the memory such that it can minimize the access time
- multiple levels present in the memory, each one having a different size, different cost, etc.
- Memory – Primary , secondary, cache, register.
- Secondary memory is also known as external memory in which the computer data and program can be saved on a long term basis.
- However, it is cheaper and slower than the main memory.
- Unlike primary memory, secondary memory cannot be accessed directly by the CPU.
- Instead of that, secondary memory data is first loaded into the RAM (Random Access Memory) and then sent to the processor to read and update the data.
- hard disk and floppy disks, CDs and magnetic tapes.

- Primary memory is also known as the computer system's main memory that communicates directly within the CPU.
- Primary- RAM, ROM.

Random Access Memory (RAM)

- is one of the faster types of main memory to temporarily store data, programs or program results
- It is volatile, which means if a power failure occurs or the computer is turned off, the information stored in RAM will be lost.

Read-Only Memory (ROM)

- ROM is a memory device or storage medium that is used to **permanently** store information inside a chip.
- It is a read-only memory that can only read stored information, data or programs, but we cannot write or modify anything.

Cache Memory

- Cache memory is a small, fast memory unit located close to the CPU.
- It stores frequently used data and instructions that have been recently accessed from the main memory.

Registers

- Registers are small, high-speed memory units located in the CPU.
- They are used to store the most frequently used data and instructions.









| Level | 1 | 2 | 3 | 4 |
|-------------|--------------------|-----------------|------------------|------------------|
| Name | Register | Cache | Main Memory | Secondary Memory |
| Size | <1 KB | less than 16 MB | <16GB | >100 GB |
| Access Time | 0.25ns to 0.5ns | 0.5 to 25ns | 80ns to 250ns | 50 lakh ns |
| Bandwidth | 20000 to 1 lakh MB | 5000 to 15000 | 1000 to 5000 | 20 to 150 |
| Managed by | Compiler | Hardware | Operating System | Operating System |
| | | | | |

Logic Gates

- In digital computers , binary information is represented in form of signal.
- For instances if 3 volt signal is input circuit output 1 else output 0.

Gates:

- To perform manipulation on binary information, logic circuits i.e. gates is used.
- Different gates have different symbol and their operations is described with algebraic expression. For instances AND gate with A, B input has algebraic expression $A \cdot B$.
- Truth Table: A tabular representation of binary input/output relationship for gate

| 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 | | | | | | | | | | | | | | | | |

- Boolean Function:
 - It is algebraic expression with logic gate operations.
 - For e.g. $F=x+y'z$, F = boolean function, x, y, z are inputs.
 - F produces output = 1 if $x=1$ or $y'z=1$.

| | Postulate / Theorem | Dual |
|---------------------|-----------------------------|----------------------------------|
| 1. Identity Element | $x + 0 = x$ | $x \cdot 1 = x$ |
| 2. Complementation | $x + x' = 1$ | $x \cdot x' = 0$ |
| 3. Idem potency | $x + x = x$ | $x \cdot x = x$ |
| 4. Null Law | $x + 1 = 1$ | $x \cdot 0 = 0$ |
| 5. Involution | $(x')' = x$ | - |
| 6. Commutative | $x + y = y + x$ | $xy = yx$ |
| 7. Associative | $x + (y + z) = (x + y) + z$ | $x(yz) = (xy)z$ |
| 8. Distributive | $x + yz = (x + y)(x + z)$ | $x(y + z) = xy + xz$ |
| 9. De Morgan | $(x + y)' = x' y'$ | $(xy)' = x' + y'$ |
| 10. Absorption | $x + xy = x$ | $x(x + y) = x$ |
| 11. Simplification | $x + x'y = x + y$ | $x(x' + y) = xy$ |
| 12. Consensus | $xy + x'z + yz = xy + x'z$ | $(x+y)(x'+z)(y+z) = (x+y)(x'+z)$ |

Boolean Function Simplification

K-Map Method (Karnaugh Map)

→ It is pictorial arrangement of Truth Table, that allow easy interpretation for choosing the minimum number of terms to express the function algebraic.

① Minterm: Each combination of variables in a truth table.

for n -variable we have 2^n possible minterms
→ Boolean function equal to 1 for some min terms and 0 for others

→ for compact representation, decimal equivalent of minterm is considered.

For eg: $f(x, y, z) = \sum (1, 4, 5, 6, 7)$
Sum of Minterm \downarrow while for this f produces of $p=12$.
Missing decimal equivalent minterm = 0.

For 2 variables

| A \ B | 0 | 1 |
|-------|-----------|-----|
| | \bar{B} | B |
| A 0 | 0 | 1 |
| A 1 | 2 | 3 |

For three variables

| A \ BC | 00 | 01 | 11 | 10 |
|--------|------------------|------------|------|------------|
| | $\bar{B}\bar{C}$ | $\bar{B}C$ | BC | $B\bar{C}$ |
| A 0 | 0 | 1 | 3 | 2 |
| A 1 | 4 | 5 | 7 | 6 |

For 4 variables

| AB \ CD | 00 | 01 | 11 | 10 |
|---------|------------------|------------|------|------------|
| | $\bar{C}\bar{D}$ | $\bar{C}D$ | CD | $C\bar{D}$ |
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 12 | 13 | 15 | 14 |
| 10 | 8 | 9 | 11 | 10 |

→ Minterm numbers are assigned in such a way ^{adjacent} minterm differ by one variable.

→ Square containing 1's are group of adjacent squares.

→ The group of square must be in power of 2.

For eg:

$$f(A, B, C) = \sum(3, 4, 6, 7)$$

| A \ BC | 00 | 01 | 11 | 10 |
|--------|----|----|----|----|
| | 0 | 1 | 3 | 2 |
| 0 | 0 | 1 | 3 | 2 |
| 1 | 4 | 5 | 7 | 6 |

$$f = A\bar{C} + BC$$

→ This Boolean expression is known as Sum of Product
First Product → Then Sum Them.

Product of Sums - Simplification

Product terms are AND terms (\cdot)
Sum terms are OR terms ($+$)

→ Minterm $\begin{matrix} \diagup 1 \\ \diagdown 0 \end{matrix}$

→ So here we consider the minterm with equal to zero.

→ So we proceed with minterm equal to zero, to obtain complement of f' . Then we take complement of f' to produce an expression of f in product of sum forms.

For eg:

$$f(A, B, C, D) = \sum (0, 1, 2, 5, 8, 9, 10)$$

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

$$f = B'D' + B'C' + A'C'D$$

$$f' = AB + CD + BD'$$

Take complement of f' , to obtain Product of sum form

$$f = (A+B') \cdot (C'+D') \cdot (B'+D)$$

Don't Care conditions

Represented by X.

Minterms that may produce either 0 or 1 for the function are said to be don't care method
 → Don't care can take 0 or 1 based on further simplification

$$f(A, B, C) = \sum (9, 2, 16)$$

$$d(A, B, C) = \sum (1, 3, 5)$$

$$f = A' + BC'$$

| BC \ A | 0 | 1 | 10 |
|--------|---|---|----|
| 00 | 1 | X | 1 |
| 01 | X | X | 1 |
| 10 | X | 1 | 1 |

| A | 0 | 1 | 10 |
|---|---|---|----|
| 0 | 1 | X | 1 |
| 1 | X | X | 1 |

~~8~~

Circuit

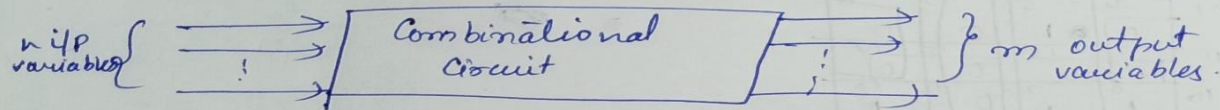
- Circuit is electronic components that enables interconnection of various gates.
- Generally, it is of two types
 - Combinational
 - Sequential
- Combinational circuit is type of circuit where output is dependent on the input only.

E.g. : half adder, full adder, decoder, encoder, multiplexer, demultiplexer
- Sequential circuit is type of circuit where output dependent on both input and previous state.

E.g.: Register , counter

Combinational Circuits

→ Combinational logic is type of digital logic implemented by Boolean circuits, where the output is a pure function of the present input



Eg: of 2 combinational circuit.

Half Adder

↳ Addition of two binary digits.

I/p → x, y

O/p → $\text{sum}(s), \text{carry}(c)$

| I/p | | O/p | |
|-----|-----|-----|-----|
| x | y | s | c |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

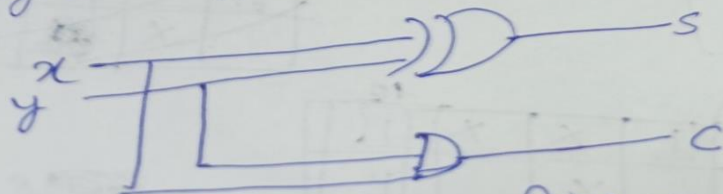
$$s = x'y + xy'$$
$$c = xy$$

Step 1: Assign I/p and o/p to variable name.

Step 2: Draw Truth Table.

Step 3: Simplified Boolean functions for each o/p.

Step 4: Logic Diagram is drawn.



Half Adder logic Diagram

Full Adder

- It performs addition of 3 bit input bit.
→ I/p → 3 i/p — x, y, z
2 o/p — Sum and Carry

Truth Table

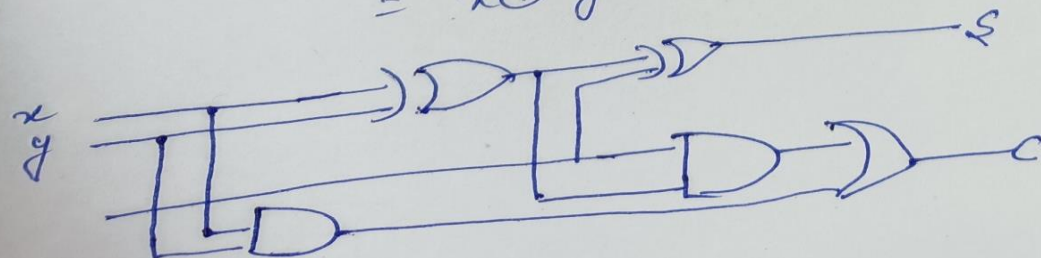
| Inputs | | | Outputs | |
|--------|-----|-----|---------|-----|
| x | y | z | C | S |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

| | | | |
|-----|-----|-----|---|
| x | y | z | |
| | | 1 | 1 |
| 1 | | | 1 |

$$S = x'y'z + x'yz' + xy'z' + xyz$$
$$= x \oplus y \oplus z$$

| | | | |
|-----|-----|-----|---|
| x | y | z | |
| | | 1 | 1 |
| | 1 | 1 | 1 |

$$C = xy + xz + yz$$



Decoder

- It is type of combinational circuit that converts n-bit coded information to maximum of 2^n output.
- It is represented as n X m decoder where $m \leq 2^n$. Here n –input and m= no. of output.
- For e.g. : 3 X 8 decoder.
- Let says 3 input line X, Y, Z.
- Converts analogue signal into digital data, which can then be processed by a computer

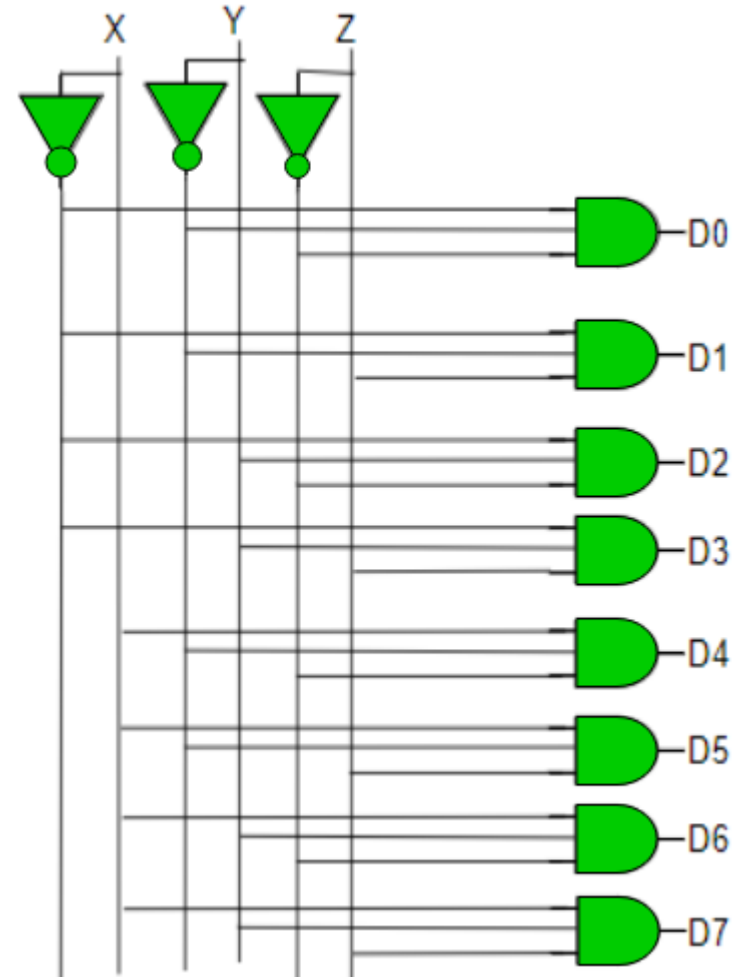
| X | Y | Z | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |
|---|---|---|----|----|----|----|----|----|----|----|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Applications:

- Tv setup box
- FM radio
- address decoders in CPU memory location identification

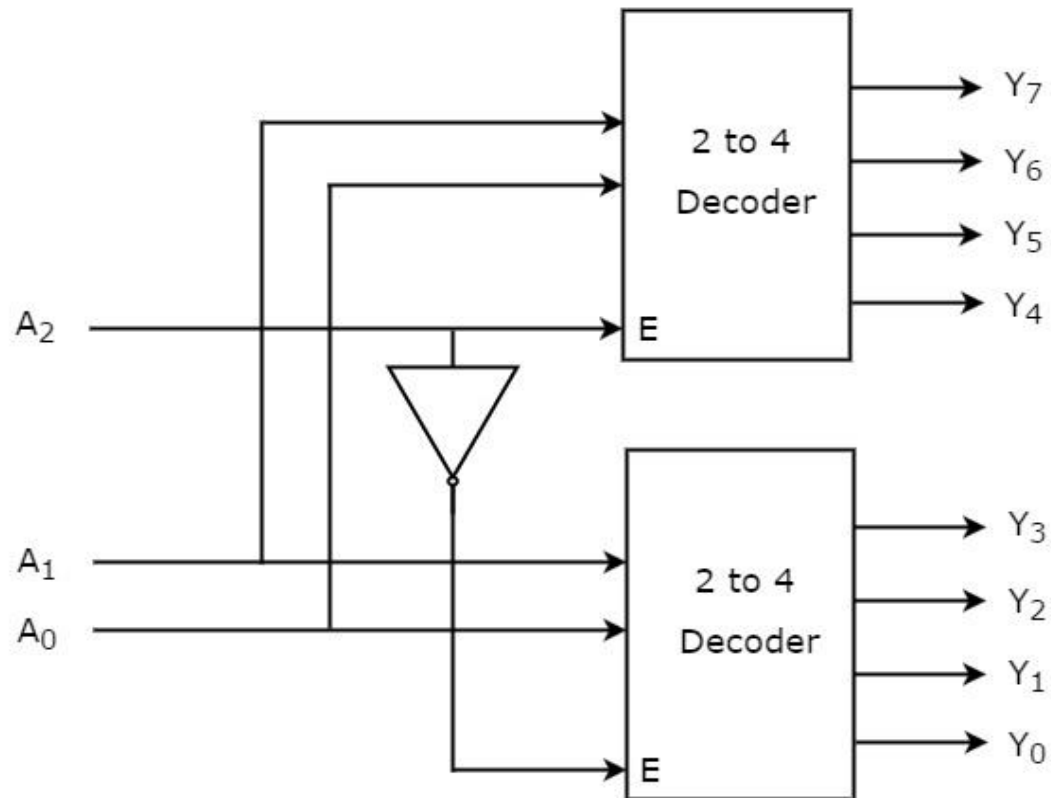
Advantages:

- Improve performance and reliability



3 X 8 Decoder

Decoder Expansion

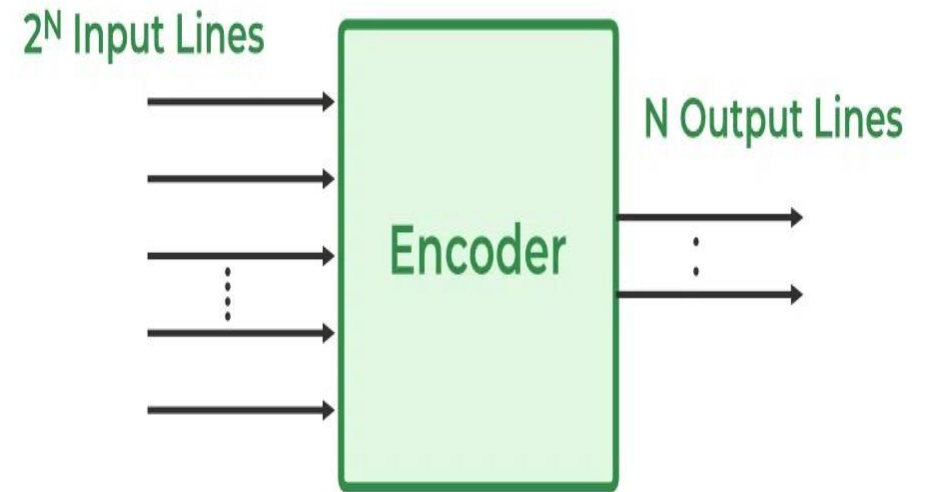


| E (A2) | A1 | A0 | Y0 | Y1 | Y2 | Y3 | Y4 | Y5 | Y6 | Y7 |
|-----------|----|----|----|----|----|----|----|----|----|----|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

- Sometimes certain size decoder is needed but only smaller sizes are available.
- Thus, with combination of smaller decoder(enable input) we can achieve functionality of larger decoder.
- For eg: 2- 2X4 decoder= 1-3X8 decoder
- Enable signal –MSB
- Two input signal –LSB.

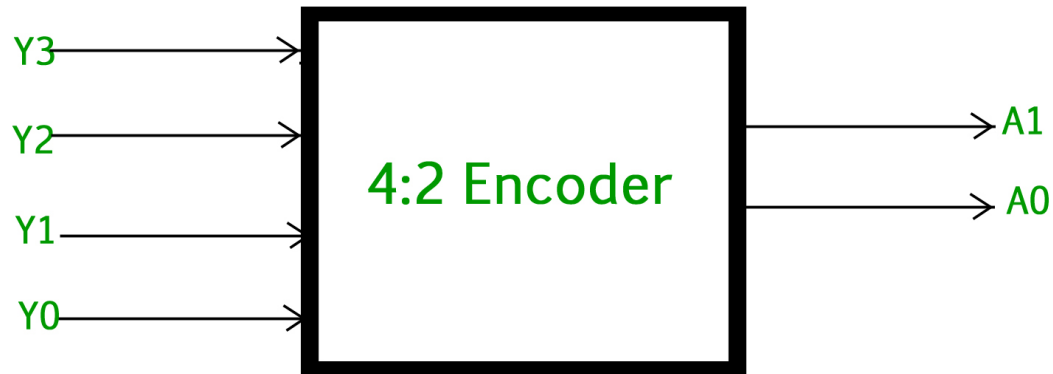
Encoder

- a digital circuit that converts a set of binary inputs into a unique binary code
- Encoders are commonly used in digital systems to convert a parallel set of inputs into a serial code.
- 2^n input \rightarrow n output.
- It will produce a binary code equivalent to the input, which is active High.
- E.g. 4 to 2 Encoder
 - Octal to Binary Encoder (8 to 3 Encoder)
 - Decimal to BCD Encoder



• 4 to 2 Encoder

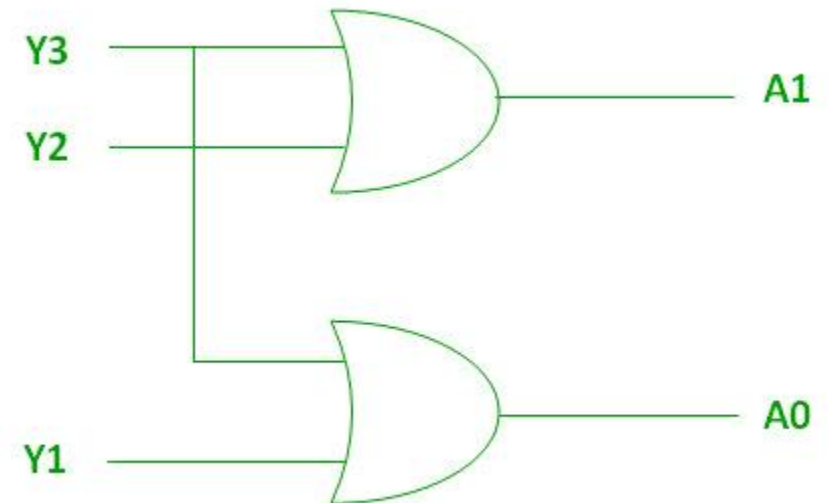
- The 4 to 2 Encoder consists of **four inputs Y3, Y2, Y1 & Y0**, and **two outputs A1 & A0**.
- At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output.
- The figure below shows the logic symbol of the 4 to 2 encoder
- Real life e.g. -



| 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 |

Logical expression for A1 and A0:

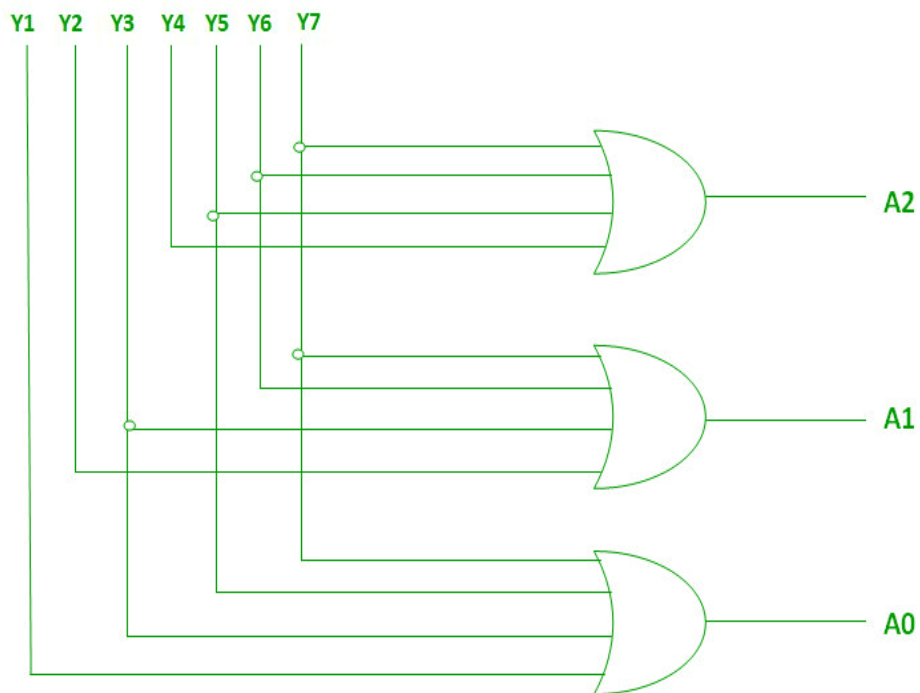
$$A1 = Y3 + Y2 \quad A0 = Y3 + Y1$$



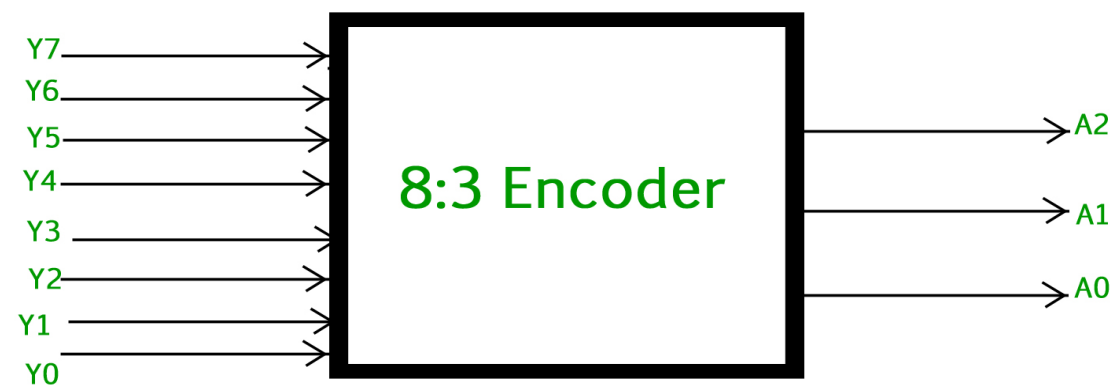
- **Octal to Binary Encoder (8 to 3 Encoder)**

- **8 inputs:** Y7 to Y0 and **3 outputs:** A2, A1 & A0.

- Each input line corresponds to each octal digit and three outputs generate corresponding binary code.
- $A2 = Y7 + Y6 + Y5 + Y4$
- $A1 = Y7 + Y6 + Y3 + Y2$
- $A0 = Y7 + Y5 + Y3 + Y1$

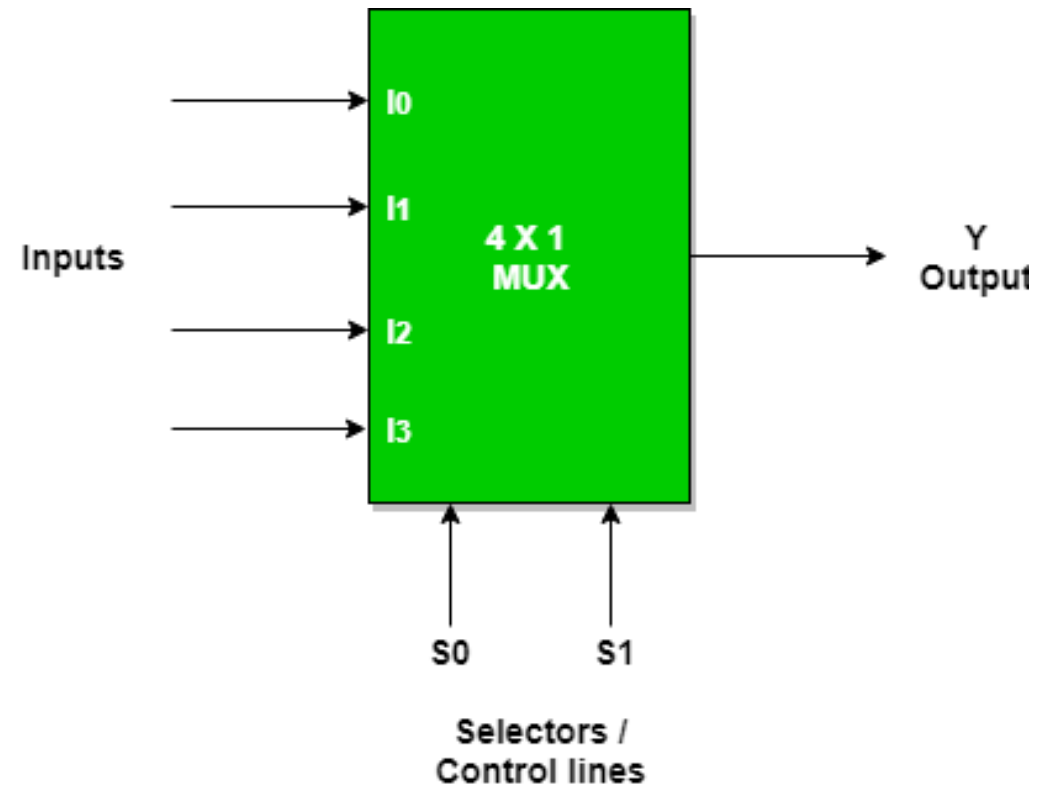


| INPUTS | | | | | | | | OUTPUTS | | |
|--------|----|----|----|----|----|----|----|---------|----|----|
| Y7 | Y6 | Y5 | Y4 | Y3 | Y2 | Y1 | Y0 | A2 | A1 | A0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |



Multiplexer

- It is a combinational circuit which have many data inputs and single output depending on control or select inputs.
- For N input lines, $\log_2 n$ (base2) selection lines, or we can say that for 2^n input lines, n selection lines are required.
- Multiplexers are also known as **“Data n selector, parallel to serial convertor, many to one circuit, universal logic circuit”**.
- Multiplexers are mainly used to increase amount of the data that can be sent over the network within certain amount of time and bandwidth.



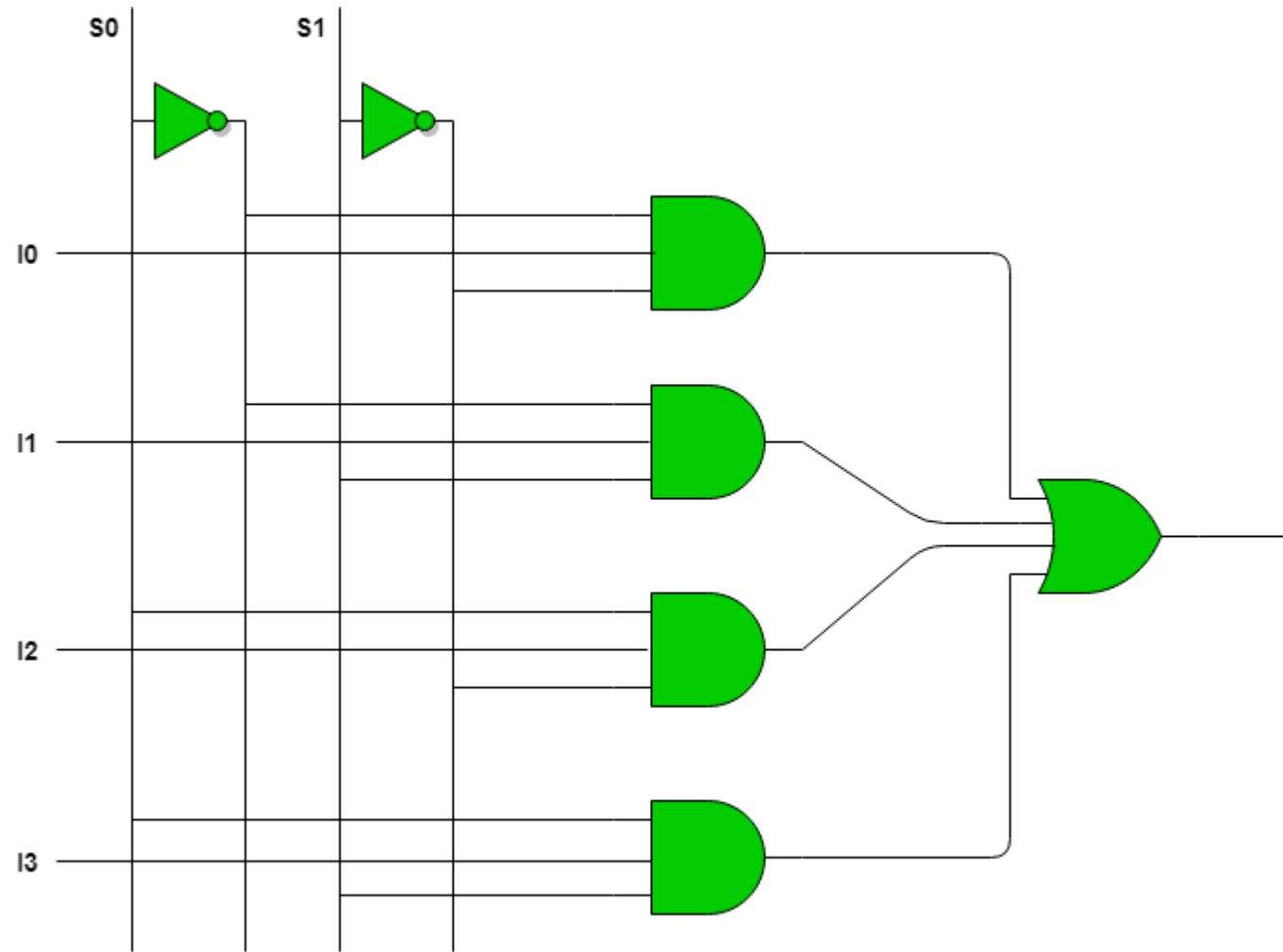
- 4:1 Multiplexer

Truth Table

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

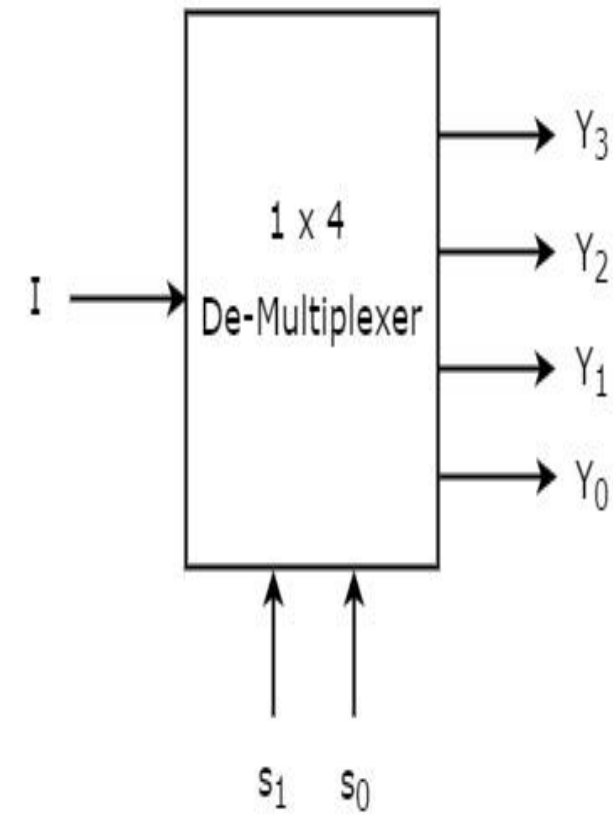
So, final equation,

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



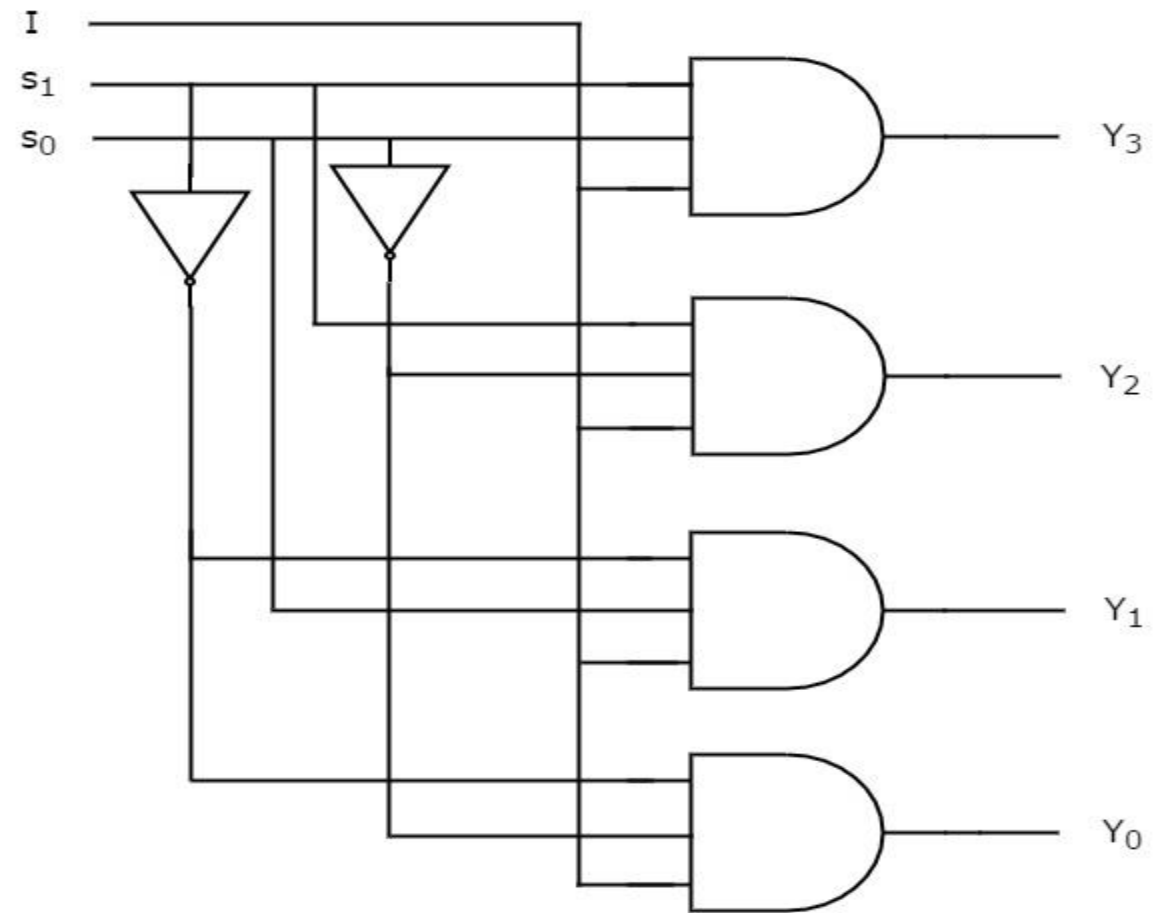
Demultiplexer

- **De-Multiplexer** is a combinational circuit that performs the reverse operation of Multiplexer.
- It has single input, 'n' selection lines and maximum of 2^n outputs. The input will be connected to one of these outputs based on the values of selection lines.
- Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones.
- So, each combination can select only one output. De-Multiplexer is also called as **De-Mux**.
-



| Selection Inputs | | Outputs | | | |
|------------------|----------------|----------------|----------------|----------------|----------------|
| S ₁ | S ₀ | Y ₃ | Y ₂ | Y ₁ | Y ₀ |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

$$\begin{aligned}
 Y_3 &= s_1 s_0 \cdot I \\
 Y_2 &= s_1 s_0' \cdot I \\
 Y_1 &= s_1' s_0 \cdot I \\
 Y_0 &= s_1' s_0' \cdot I
 \end{aligned}$$



Sequential Circuit

- This circuit depends upon both input and previous state.
- Thus to store previous state, a memory element is required.

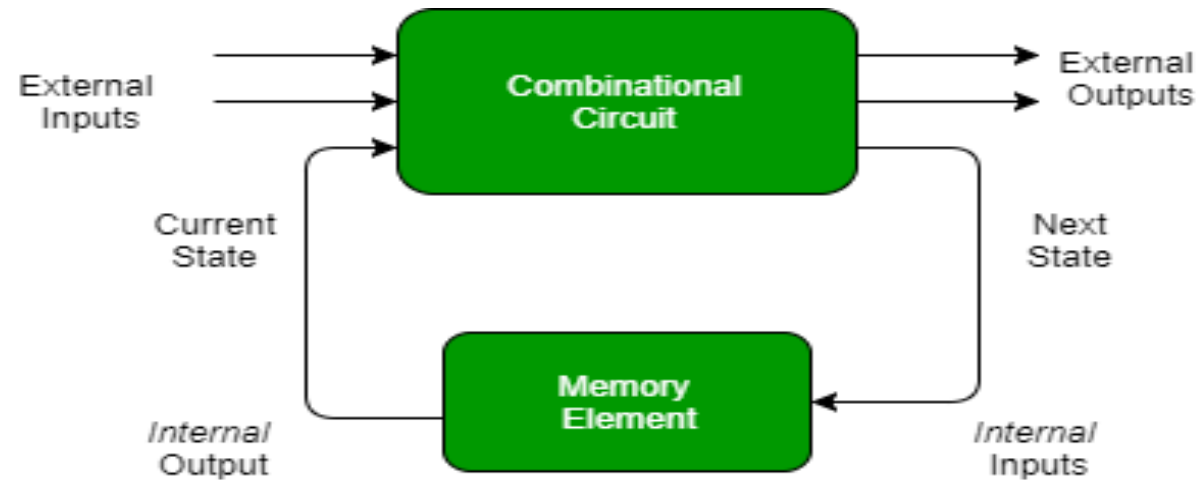


Figure: Sequential Circuit

- Flip flop (FF) is a basic digital memory circuit, which stores one bit of information.

- The state of flip-flop changes at active state of clock pulses and remains unaffected when the clock pulse is not active.
- Clocking causes the flip-flop either to change or to retain its output signal based upon the values of the input signals at the transition.
- flip-flop can be in one of two stable states, typically represented as 0 or 1. The flip-flop changes its state based on the input signals it receives, and this change occurs only at specific points in time, typically controlled by a clock signal.
- When the clock signal transitions from one state to another (e.g., from low to high or high to low), the input signals are examined, and the flip-flop's output may change accordingly. The specific behavior of a flip-flop depends on its type and the input conditions.
- Flip-flops are used such as storing data in registers, implementing counters, designing finite state machines, and building memory units.

Types of Sequential Circuits

Asynchronous sequential circuit

- These circuits **do not use a clock signal** but uses the pulses of the inputs.
- change their state immediately when there is a change in the input signal.
- Use when speed of operation is important and **independent** of internal clock pulse
- But these circuits are more **difficult** to design and their output is **uncertain**.

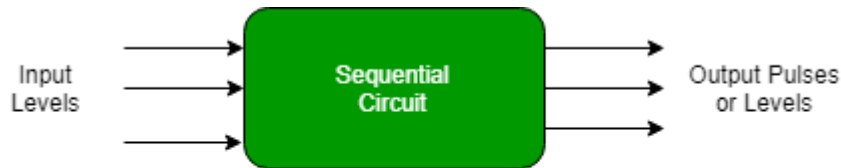


Figure: Asynchronous Sequential Circuit

Synchronous sequential circuit:

- These circuits **uses clock signal** and level inputs (or pulsed).
- bit **slower** compared to asynchronous.
- output changes only when next input or clock pulse.
- Synchronized and deterministic output

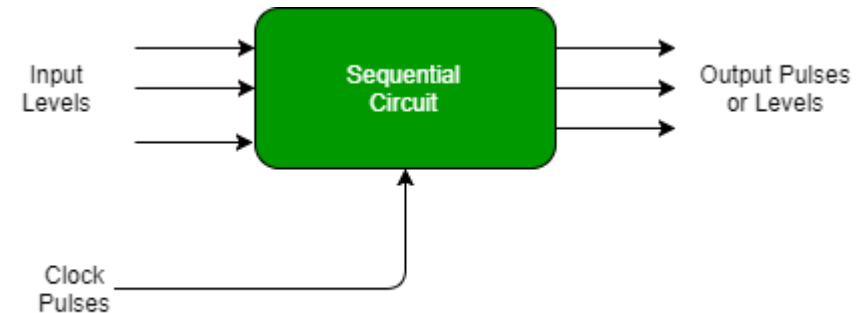


Figure: Synchronous Sequential Circuit

Types of Flip flop

SR flip-flop:(Set-Reset flip-flop)

- It has two inputs, commonly labeled S (set) and R (reset), and two outputs, Q (normal output) and \bar{Q} (complement output)
- When both inputs are low (0), the flip-flop maintains its current state, and the outputs remain unchanged.
- When the S input is high (1) and the R input is low (0), the flip-flop is "set" or "1", and the Q output goes high (1) while the \bar{Q} output goes low (0).
- Conversely, when the R input is high (1) and the S input is low (0), the flip-flop is "reset" or "0", and the Q output goes low (0) while the \bar{Q} output goes high (1).
- Characteristics Eqn:

$$Q(t+1) = S + R'Q(t)$$

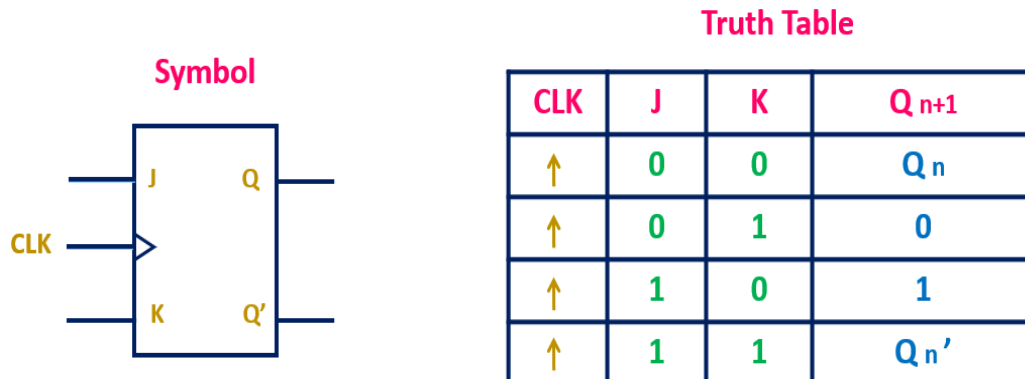
| S | R | Q _{t+1} |
|---|---|---------------------|
| 0 | 0 | Q |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | -(Forbidden State) |

Characteristics Table

| S | R | Q(t) | Q(t+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 | X |
| 1 | 1 | 1 | X |

JK Flip Flop

- It is a modification of the SR flip-flop with an additional input, known as the "toggle" or "J-K" input.
- The JK flip-flop has two inputs: J (set) and K (reset), along with a clock input (CLK) and two outputs: Q (the current state) and Q' (the complement of the current state).



Characteristics Eqn:

$$Q(t+1) = JQ(t)' + K'Q(t)$$

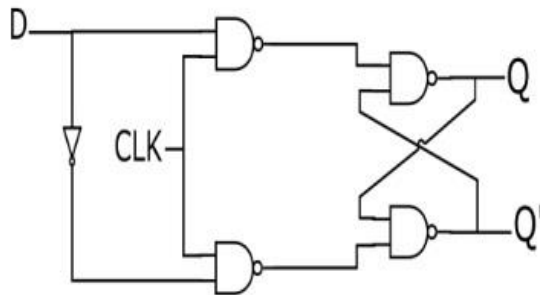
| J | K | Q(t) | Q(t+1) |
|---|---|------|--------|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

D flip flop

- A D flip-flop, also known as a data flip-flop or delay flip-flop.
- a D flip-flop simply transfers the input data D to the output Q on the rising edge of the clock signal.

| D | CLK | Q(t) | Q(t+1) |
|-------|-----|------|--------|
| ----- | | | |
| 0 | ↑ | Q(t) | 0 |
| 1 | ↑ | Q(t) | 1 |

Characteristics Eqn.
 $Q(t+1)=D$

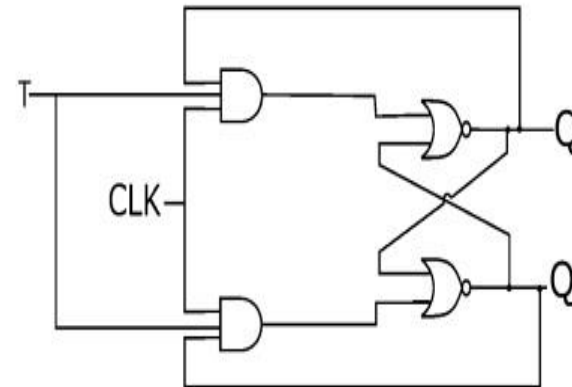


T flip flop

- It has a single input called T (toggle) and two outputs: Q (the current state) and Q' (the complement of the current state).

| T | CLK | Q(t) | Q(t+1) |
|-------|-----|------|-------------|
| ----- | | | |
| 0 | ↑ | Q(t) | Q(t) |
| 1 | ↑ | Q(t) | $\sim Q(t)$ |

Characteristics Eqn.
 $Q(t+1)=T'Q(t)+TQ(t)'=T\oplus Q(t)$

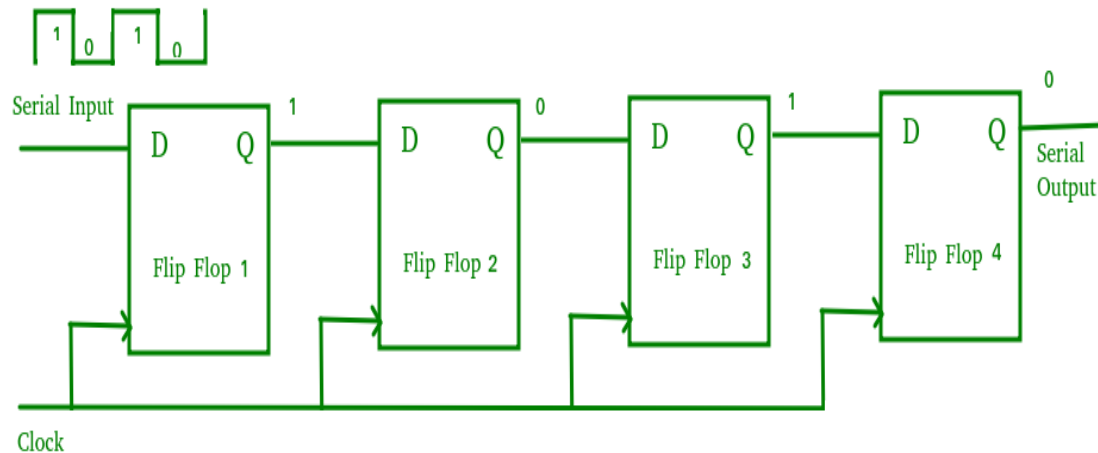


Registers

- a register is a storage element within a processor or a separate circuit that can store and hold binary data.
- A Register is a device that is used to store such information. It is a group of flip-flops connected in series used to store multiple bits of data.
- The information stored within these registers can be transferred with the help of shift registers.
- Registers typically consist of a group of flip-flops (such as D flip-flops) that store individual bits of data.
- The number of flip-flops determines the size or width of the register, and each flip-flop holds one bit of information. For example, an 8-bit register would have eight flip-flops, allowing it to store 8 bits of data.
- Registers serve several important functions in digital systems
 - Data Storage
 - Data Transfer
 - Addressing
 - Control Signals

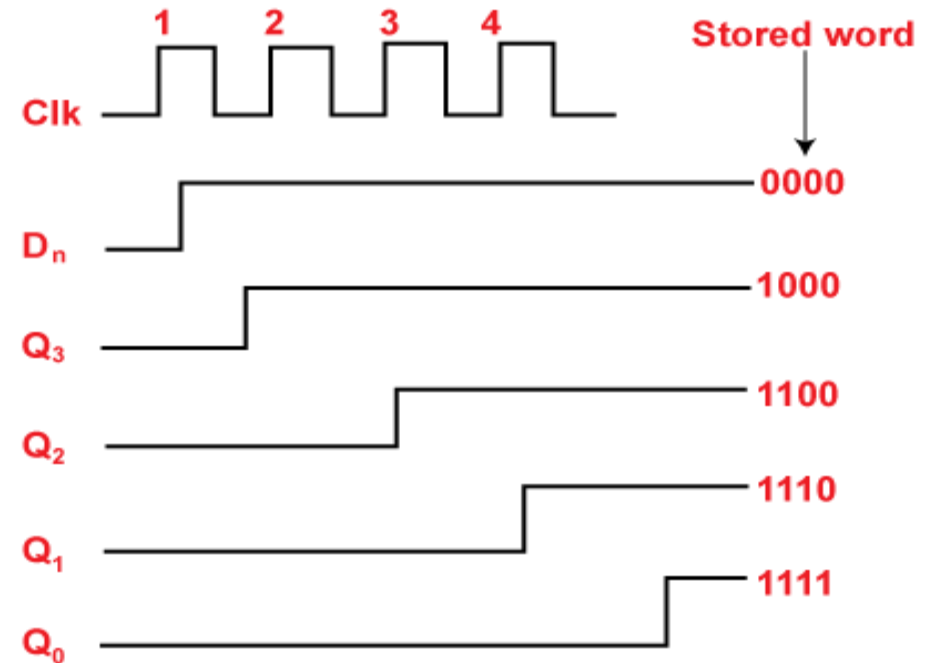
Shift registers

- The bits stored in such registers can be made to move within the registers and in/out of the registers by applying clock pulses.
- Generally, of two types:
 - Shift right register
 - Shift left register



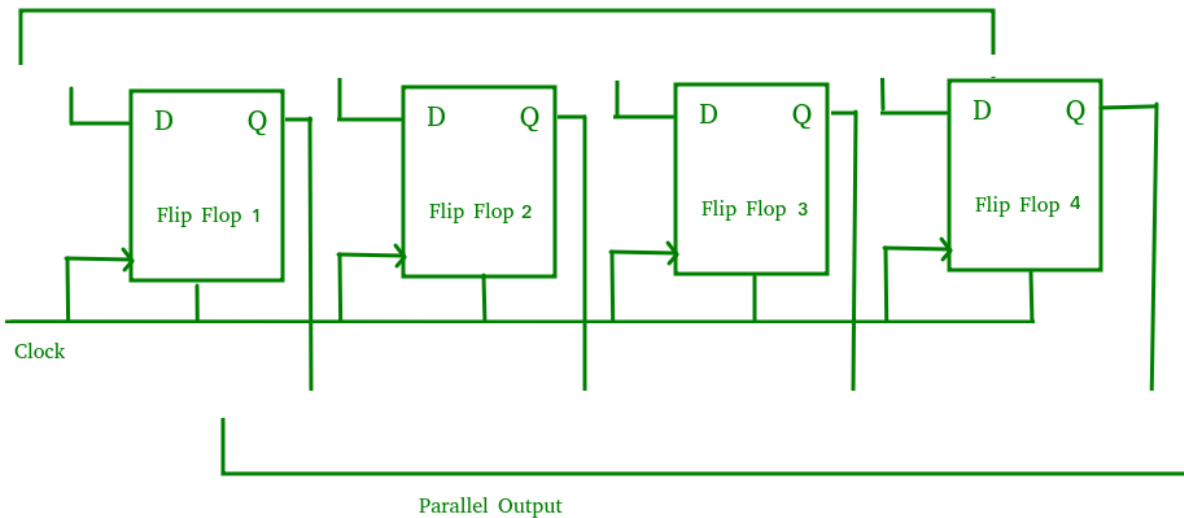
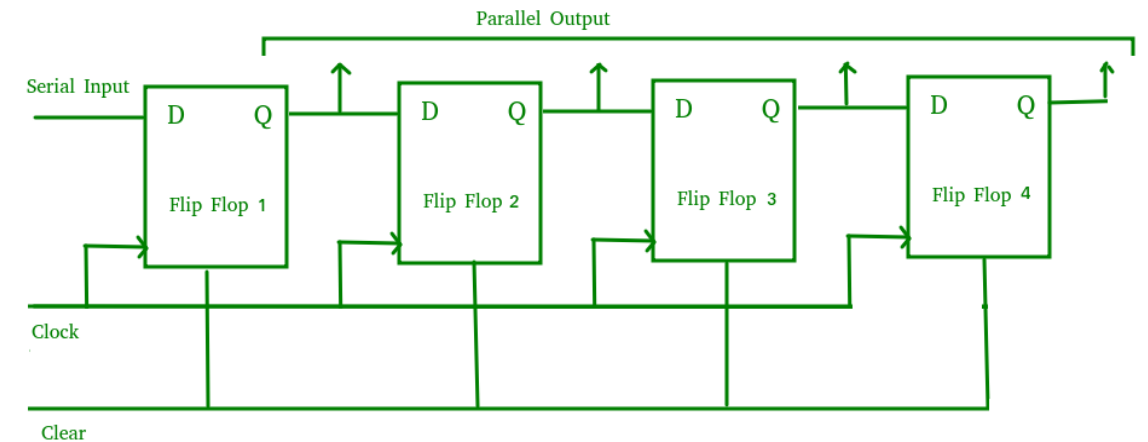
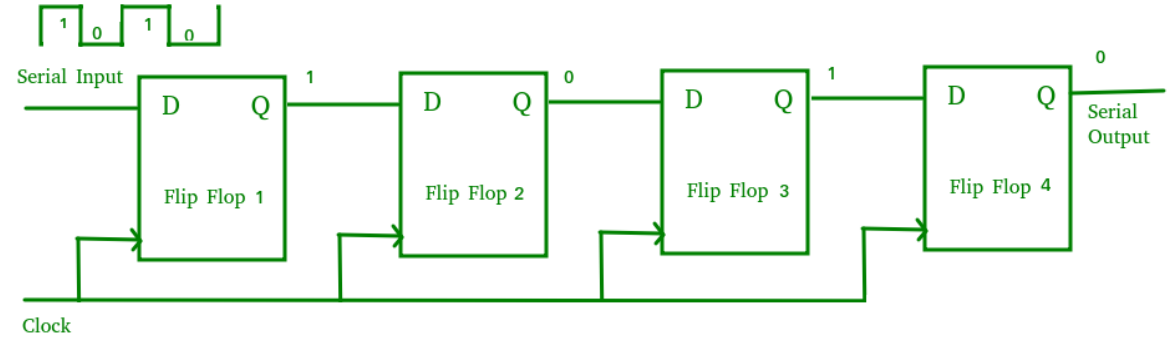
| | Clk | $D_n = Q_3$ | $Q_3 = D_2$ | $Q_2 = D_1$ | $Q_1 = D_0$ | Q_0 |
|-----------|-----|-------------|-------------|-------------|-------------|-------|
| Initially | | | 0 | 0 | 0 | 0 |
| (1) | ↓ | 1 → | 1 | 0 | 0 | 0 |
| (2) | ↓ | 1 → | 1 | 1 | 0 | 0 |
| (3) | ↓ | 1 → | 1 | 1 | 1 | 0 |
| (4) | ↓ | 1 → | 1 | 1 | 1 | 1 |

→ Direction of data travel



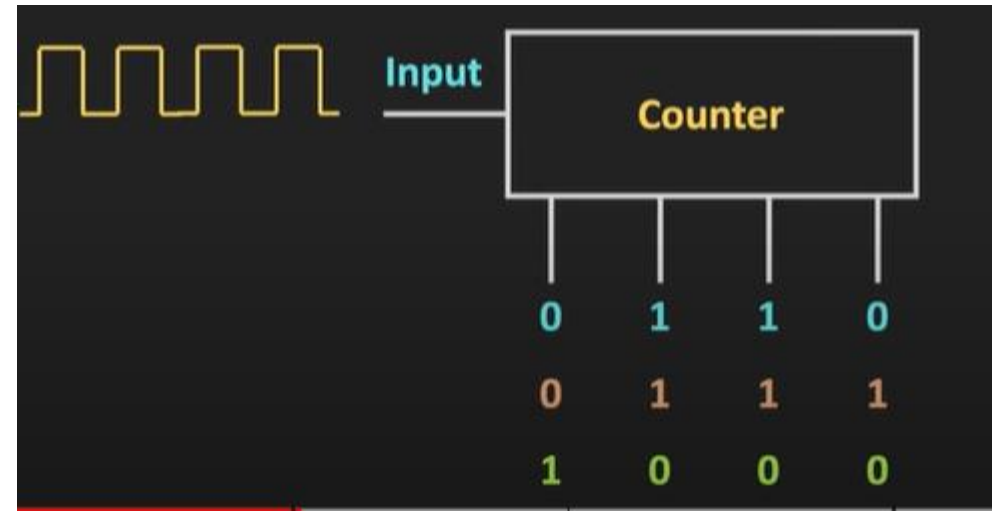
Types of serial registers

- SISO: Serial-In Serial-Out Shift Register (SISO)
- SIPO: Serial-In Parallel-Out Shift Register (SIPO)
- PISO: Parallel-In Serial-Out Shift Register
- PIPO: Parallel-In Parallel-Out Shift Register



Counters

- Counter is a device that stores the number of times an event occurs.
- E.g. timer in washing machine, ac
- It is constructed by number of flip flops connected in cascade.
- Trigger in flip flop is controlled by clock pulse.(low to high or high to low)
- If the output of counter is binary number sequence, then it is binary counter.
- Binary up – 000- 111
- Binary down – 111-000
- N bit binary counter = n flip flop and we count from 0 to $2^n - 1$
- For reading the output of counter we need decoding logic. (whether particular count is reached or not)



Mod of counter:

- No of different output through which counter goes before returning to its final state
- For eg 4 bit – binary system = Mod 16
- BCD counter = Mod 10

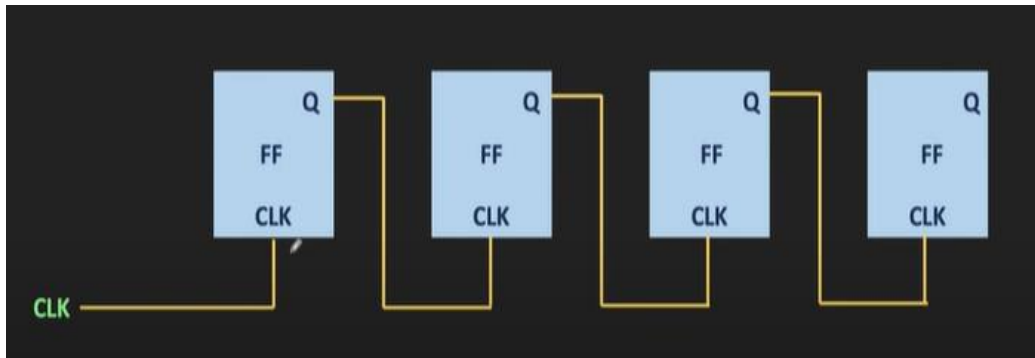
Applications

- Timer
- Time measurement
- Analog to digital
- Frequency division

Types of counter

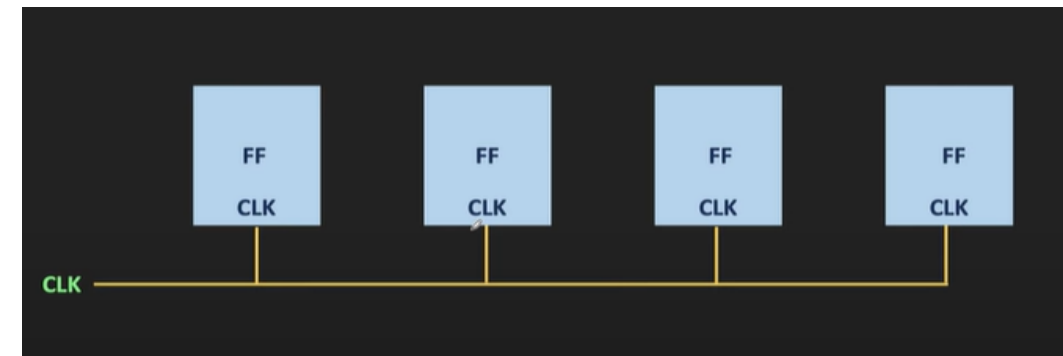
Asynchronous

- Here the clock pulse is directly applied to first FF
- Output of one FF is input to clock of next FF
- E.g.-Ripple counter
- Cannot operate for high-speed operations
- Sequence is always follows



Synchronous

- Here clock pulse is applied to each FF
- Fast as compared to asynchronous
- Sequence may or may not.
- E.g.- Ring counter



RISC vs CISC

- **Reduced Instruction Set Architecture (RISC)** –
The main idea behind this is to make hardware simpler by using an instruction set composed of a few basic steps for loading, evaluating, and storing operations just like a load command will load data, a store command will store the data.

Characteristic of RISC –

1. Simpler instruction, hence simple instruction decoding.
2. Instruction comes undersize of one word.
3. Instruction takes a single clock cycle to get executed.
4. More general-purpose registers.
5. less Addressing Modes.
6. Low cost and low powerful
7. A pipeline can be achieved.
8. Microprogrammed

e.g. LOAD A, 2

LOAD B, 5

PROD A, B

STORE, A

Complex Instruction Set Architecture (CISC) –
The main idea is that a single instruction will do all loading, evaluating, and storing operations just like a multiplication command will do stuff like loading data, evaluating, and storing it, hence it's complex.

Characteristic of CISC –

1. Complex instruction, hence complex instruction decoding.
2. Instructions are larger than one-word size.
3. Instruction may take more than a single clock cycle to get executed.
4. Less number of general-purpose registers as operations get performed in memory itself.
5. More Addressing Modes.
6. High cost and powerful
7. Hardwired

E.g.- MULT 2,5

Advantages of RISC:

- **Simpler instructions:** easier to decode and execute quickly. This results in faster processing times.
- **Faster execution:** simpler instruction set - execute instructions
- **Lower power consumption:** RISC processors consume less power than CISC processors, making them ideal for portable devices.

Disadvantages of RISC:

- **More instructions required:** RISC processors require more instructions to perform complex tasks than CISC processors.
- **Increased memory usage:** RISC processors require more memory to store the additional instructions needed to perform complex tasks.
- **Higher cost:** Developing and manufacturing RISC processors can be more expensive than CISC processors.

Advantages of CISC:

- **Reduced code size:** CISC processors use complex instructions that can perform multiple operations, reducing the amount of code needed to perform a task.
- **More memory efficient:** fewer instructions to perform complex tasks, which can result in more memory-efficient code.
- **Widely used:** CISC processors have been in use for a longer time than RISC processors, so they have a larger user base and more available software.

Disadvantages of CISC:

- **Slower execution:** more complex instructions and need more time to decode them.
- **More complex design:** more complex instruction sets, which makes them more difficult to design and manufacture.
- **Higher power consumption:** CISC processors consume more power than RISC processors because of their more complex instruction sets.

Types of Computer

- We will discuss the type of computers on the basis of size and data handling capabilities.
 - Super Computer
 - [Mainframe computer](#)
 - Mini Computer
 - Workstation Computer
 - Personal Computer (PC)
 - Server Computer
 - Analog Computer
 - Digital Computer
 - Hybrid Computer
 - Tablets and Smartphone

Supercomputer

- They are the biggest and fastest computers (in terms of speed of processing data).
- process a huge amount of data, like processing trillions of instructions or data just in a second.
- scientific and engineering applications such as weather forecasting, scientific simulations, and nuclear energy research.
- first developed by Roger Cray in 1976.

Mainframe computer

- can support hundreds or thousands of users at the same time.
- It also supports multiple programs simultaneously.
- ideal for big organizations like banking, telecom sectors, etc..

Mini-Computer

- In this type of computer, there are two or more processors, and it supports 4 to 200 users at one time.
- places like institutes or departments for different work like billing, accounting, inventory management, etc

Workstation Computer

- A workstation computer is designed for technical or scientific applications.
- It consists of a fast microprocessor, with a large amount of RAM and a high-speed graphic adapter. It is a single-user computer.

Personal Computer (PC)

- Personal Computers is also known as a microcomputer.
- It is basically a general-purpose computer designed for individual use.
- E.g. laptop

Server Computer

- Server Computers are computers that are combined data and programs. Electronic data and applications are stored and shared in the server computer.
- E.g. Wikipedia

Analog Computer

- Analog Computers are particularly designed to process analog data. Continuous data that changes continuously and cannot have discrete values are called analog data.
- For e.g. speedometer, mercury thermometer, etc.

Digital Computer

- perform calculations and logical operations at high speed.
- E.g. laptops, desktops

Hybrid Computer

- combination of both analog and digital computers.
- E.g. – petrol pump (fuel flow into quantity and price)

Tablet and Smartphones

- Tablets and Smartphones are the types of computers that are pocket friendly and easy to carry is these are handy.

Unit I: Introduction

11 lecture hours

- Evolution of Computer Systems, Von Neumann Architecture, Moore's Law, Computer Types, Functional Units, Devices (Input, Output, Storage & Communication Devices), Memory System (RAM, ROM, Cache, VM, etc.), Introduction to Logic Gates, Truth Table, K-Map, Latch Flip Flops (J, K & D), Encoder & Decoder, MUX & DEMUX, Registers & Counters, Binary Number system, Overview of RISC/CISC, RISC vs. CISC.

Unit II: ALU Design

8 lecture hours

- Computer Organization and Design, Instruction Codes, Op-Code, Computer registers, Computer Instructions, CPU stack Organization, Instruction Formats, Instruction types, Timing and control, Instruction and Instruction sequencing, Instruction Cycle, Memory Reference Instructions, Addressing modes, Program Control, Types of Interrupts, Adder & Subtractor.