Memory.

Memory — means for storing data or information in the form of binary words. Data can be numerical or alphanumerical types or programs.

$$\text{Memory} \begin{cases} \text{Data memory} \rightarrow \text{to store data} \\ \text{program memory} \rightarrow \text{to store program.} \end{cases}$$

small special purpose computers may have no data memory whereas general purpose computers have a large portion for data storage.

The capacity of a memory is the no: of bits it can store. It is usually specified in terms of bytes. These memories are made up of storage elements like FFs or Capacitors in semiconductor system & magnetic domain in magnetic memories, each of which stores one bit of data. A storage element is called a cell.

eg:- Magnetic tapes, magnetic disks, MBM (magnetic bubble memory.

Computer memories can also be classified into

1) main. & 2) peripheral

Main Memory :- It is ~~the~~ an internal part of the computer. & is very fast. It serves as program memory. The program to be currently executed & data used by the program are usually stored in the main memory ~~in~~ eg:- Semiconductor memories are used as main memory. Its also called as core memory.

2) Peripheral Memory :- Also called as auxillary / add on / Mass memory. It's a typically add on memory for very large storage capacity, & is much slower than

~~data~~ Main memory usually consists of magnetic tapes or disks. It can store millions of bits of data without electric power.

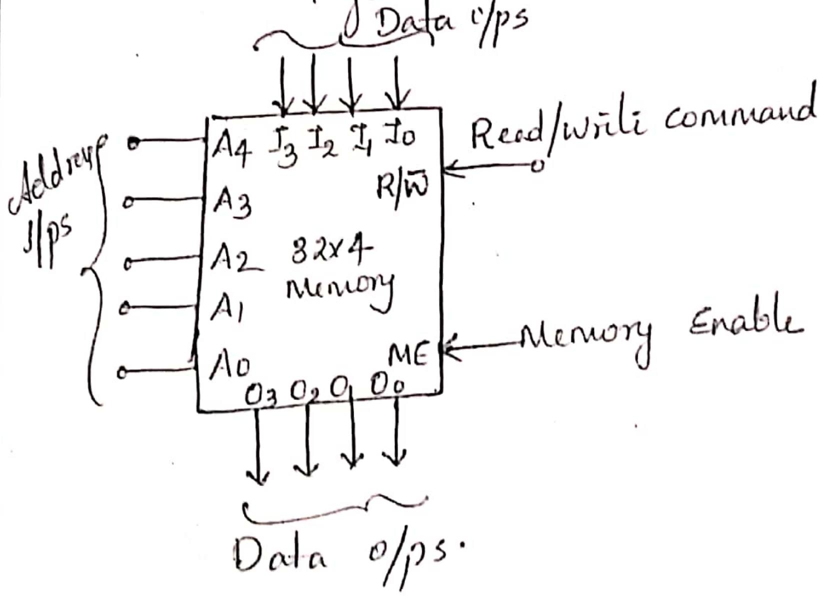
Memory Organization & operation:

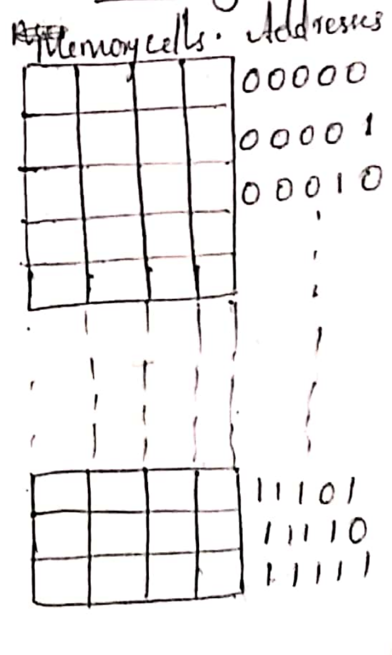
In each memory, informations are stored as bits. The no: of bits in a location may differ based on the type of the computers system used. The group of bits in a location is called a word, & word size is no: of bits in a word. It may vary from 4 to 64 or more bits. If word is of 8 bit, ie 1 byte, that location can store 8 bits. Thus a memory location is defined as a set of devices capable of storing one word.

eg:- 8 bit micro-computer - each location can store 8 bits, so 8 latches to store in a cell.

The capacity or size of a memory is the total no: of bits it can store. For convenience the size of memory is expressed as a multiple of $2^{10} = 1024 = 1K$. So $2^{11} = 2 \cdot 2^{10} = 2 \cdot K$ ie 2 Kilo bytes

Figure illustrates ~~the~~ a 32x4 memory.





word size is 4 bits. so data handling capacity is 4 bits at a time. ie four i/p lines & 4 o/p lines. 32 such cells are available. ~~ie~~ Each cell in a memory device is identified by a unique address for data transfer. so 32 different addresses are needed. For generating 32 different addresses, minimum 5 address lines are required, ie $2^5 = 32$.

Thus $A_0 - A_4$ forms address lines
$I_0 - I_3$ " I/p lines &
$O_0 - O_3$ " o/p lines.

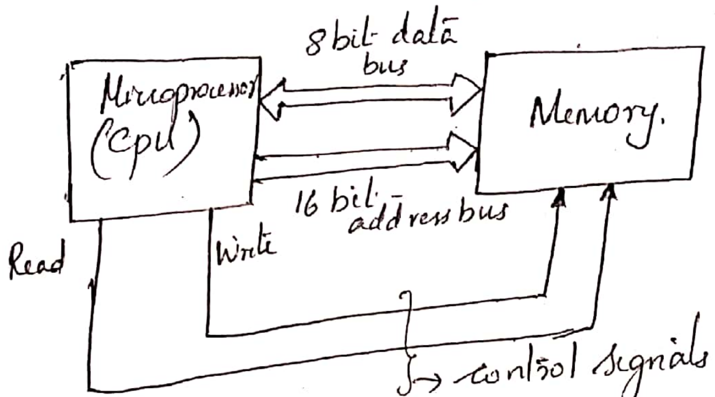
To store some data into the memory, the location is selected by giving proper address through the address lines. After this the data is to be transferred to that location & this process is called a write operation. During the write operation, data to be written must be available at the i/p lines. this operation is enabled by an active low signal at the R/$\overline{W}$ command line.

To retrieve data from a memory device, the location is identified by selecting the proper address & keep the R/$\overline{W}$ logic ~~inkep~~ at high level, ~~the~~ the data will be now available at the O/p lines. This process is called as Read operation.

To activate the Memory chip, there is a Memory Enable input line. If it is kept disabled, it will not respond to other inputs. To select a memory chip among a no: of memories, the i/p line can be used.

Read/Write Logic

Process of storing data in memory is called write operation & the process of retrieving data from memory is called read operation. Fig shows how reading & writing is accomplished in an 8 bit microprocessor system. (μp)



μp system consists of a cpu, a no: of registers & logic circuits to perform read, write & programme execution. Read & Write are the two control signals to control the respective operations. Data transfer is through 8 bit data lines from D_0 to D_7. To identify the memory locations there 16 address lines $A_0 - A_{15}$ so that 2^{16} different locations can be addressed.

when a data is to stored in memory, the cpu activates the write control & places the data to be stored on data bus & the address to which it is to be placed on the address bus. The address is transmitted through the address lines & the respective location is identified. Then data from data bus is placed in that location erasing the old data. Similarly to read data from a location, the address is transmitted over the address lines. from that location data is taken & ~~placed~~ transferred to cpu ~~by~~ through the data bus.

Types of Memory.

1) Read/write Memory (RWM) eg:- RAM
~~ard~~ both read & write operations are possible

2) ROM - Read only Memory. only for read operations.
∴ used for permanent storage of programs or data.
It is a ready to use set of combinational circuits.
It is a previously programmed 'Decoder-OR array based logic device with appropriate masks at the manufacturing stage. Data on ROM do not change or ~~have~~ get erased when power is interrupted. so it is a non-volatile type of memory.

Applications :- where non-volatile data storage is required.

✓ Microcomputer program storage (firmware)
Microcomputer programs stored in ROMs are called as firmware, becoz they are not subject to change.
eg:- electronic games, electronic cash registers

✓ Boot strap memory :- Boot strap program is stored in ROM.

✓ Data tables - for storing tables that do not change
eg:- trigonometric tables, code conversion tables.

✓ Data converters -
eg:- Binary to BCD

✓ character generators - stores dot pattern codes for each character at an address corresponding to the ASCII code for that character. eg:- to display or print alphanumeric characters.

✓ function generator — which produces waveforms of different functions like sine, triangular etc.

ROM special version programmable logic devices are called as PLDs. The main PLD is ~~a~~ Programmable Read Only Memory (PROM). Details are given in the next page.

Volatile & Non volatile Memory

Volatile — Data stored in RWMs made from semiconductor devices will be lost if power is removed. Such memory is said to be volatile.

Non Volatile — Data stored in memory is retained even after the power is switched off. Such memory is said to be non volatile — for eg:- ROM

Random Access & Sequential Access Memory.

Random Access Memory are memory whose locations can be accessed directly & immediately. ie time taken to access a data from ~~to~~ any location is same.

Sequential Access Memory:— to access a data from a location, the system has to through a series of address before reaching the address desired. for eg:- magnetic tape — to read a data, ~~the~~ ~~tape~~ it is necessary to wind or unwind the tape & go through a series of addresses before reaching the address desired.

Memory constituents.

Made of semiconductor devices like BJT or MOSFET.

MOSFET Technology	BJT Technology.
✓ Easy technique	✓ not so easy
✓ less space	✓ large space
✓ so large capacity Memory IC's	✓ less storage
✓ slow compared to BJT	✓ faster than MOS memory
✓ Every type of IC is available	✓ not every type in BJT technology
✓ used where greater capacity is more important than speed	✓ used for high speed applications
✓ Dynamic memory is possible	✓ Dynamic memory design cannot be done.

i^2L technology is also available which gives higher speed than MOS & greater capacity than BJT.

Static & Dynamic Memory.

Static — Available in both BJT & MOSFET technologies, which uses conventional storage devices such as latches.

eg:- SRAM

Dynamic — Available only in MOSFET circuits. Here storage of data is done ~~as~~ as charge on capacitances. Thus it comprises a very large no: of memory cells in a single circuit and very low power consumption. periodic refreshment of the circuit is essential. so complicated circuit. eg:- DRAM.

Random-access memory (RAM /ræm/) is a form of computer data storage that stores data and machine code currently being used. A random-access memory device allows data items to be read or written in almost the same amount of time irrespective of the physical location of data inside the memory. In contrast, with other direct-access data storage media such as hard disks, CD-RWs, DVD-RWs and the older magnetic tapes and drum memory, the time required to read and write data items varies significantly depending on their physical locations on the recording medium, due to mechanical limitations such as media rotation speeds and arm movement.

RAM contains multiplexing and demultiplexing circuitry, to connect the data lines to the addressed storage for reading or writing the entry. Usually more than one bit of storage is accessed by the same address, and RAM devices often have multiple data lines and are said to be "8-bit" or "16-bit", etc. devices.

In today's technology, random-access memory takes the form of integrated circuits. RAM is normally associated with volatile types of memory (such as DRAM modules), where stored information is lost if power is removed, although non-volatile RAM has also been developed.[1] Other types of non-volatile memories exist that allow random access for read operations, but either do not allow write operations or have other kinds of limitations on them. These include most types of ROM and a type of flash memory called *NOR-Flash*.

The two widely used forms of modern RAM are static RAM (SRAM) and dynamic RAM (DRAM). In SRAM, a bit of data is stored using the state of a six transistor memory cell. This form of RAM is more expensive to produce, but is generally faster and requires less dynamic power than DRAM. In modern computers, SRAM is often used as cache memory for the CPU. DRAM stores a bit of data using a transistor and capacitor pair, which together comprise a DRAM cell. The capacitor holds a high or low charge (1 or 0, respectively), and the transistor acts as a switch that lets the control circuitry on the chip read the capacitor's state of charge or change it. As this form of memory is less expensive to produce than static RAM, it is the predominant form of computer memory used in modern computers.

Both static and dynamic RAM are considered *volatile*, as their state is lost or reset when power is removed from the system. By contrast, read-only memory (ROM) stores data by permanently enabling or disabling selected transistors, such that the memory cannot be altered. Writeable variants of ROM (such as EEPROM and flash memory) share properties of both ROM and RAM, enabling data to persist without power and to be updated without requiring special equipment. These persistent forms of semiconductor ROM include USB flash drives, memory cards for cameras and portable devices, and solid-state drives. ECC memory (which can be either SRAM or DRAM) includes special circuitry to detect and/or correct random faults (memory errors) in the stored data, using parity bits or error correction codes.

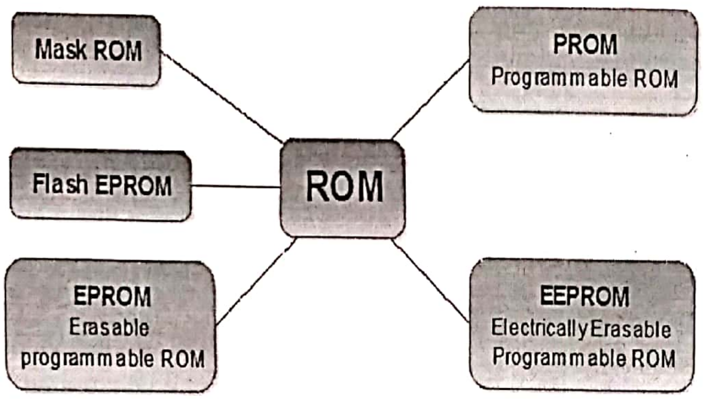
In general, the term *RAM* refers solely to solid-state memory devices (either DRAM or SRAM), and more specifically the main memory in most computers. In optical storage, the term DVD-RAMis somewhat of a misnomer since, unlike CD-RW or DVD-RW it does not need to be erased before reuse. Nevertheless, a DVD-RAM behaves much like a hard disc drive if somewhat slower.

ROM is a type of memory that does not lose its contents when the power is turned off. For this reason, ROM is also called non volatile memory.

Different Types of ROM

There are different types of read-only memory, such as

1. PROM (Programmable ROM)
2. EPROM (Erasable Programmable ROM)
3. EEPROM (electrically erasable programmable ROM)
4. Flash EPROM
5. Mask ROM



Mask ROM
PROM Programmable ROM
Flash EPROM
ROM
EPROM Erasable programmable ROM
EEPROM Electrically Erasable Programmable ROM

PROM (programmable ROM) and OTP

PROM refers to the kind of ROM that the user can burn information into. In other words, PROM is a user-programmable memory:

For every bit of the PROM, there exists a fuse. PROM is programmed by blowing the fuses. If the information burned into PROM is wrong, that PROM must be discarded since its internal fuses are blown permanently. For this reason, PROM is also referred to as OTP (One Time Programmable).

Programming ROM, also called burning ROM, requires special equipment called a ROM burner or ROM programmer.

EPROM (erasable programmable ROM) and UV-EPROM

EPROM was invented to allow making changes in the contents of PROM after it is burned.

In EPROM, one can program the memory chip and erase it thousands of times. This is especially necessary during development of the prototype of a microprocessor-based project.

A widely used EPROM is called UV-EPROM, where UV stands for ultraviolet. The only problem with UV-EPROM is that erasing its contents can take up to 20 minutes.

All UV-EPROM chips have a window through which the programmer can shine ultraviolet (UV) radiation to erase the chip's contents. For this reason, EPROM is also referred to as UV-erasable EPROM or simply UV-EPROM.

Programming a UV-EPROM

To program a UV-EPROM chip, the following steps must be taken:

1. Its contents must be erased. To erase a chip, remove it from its socket on the system board and place it in EPROM erasure equipment to expose it to UV radiation for 5—20 minutes.

All UV-EPROM chips have a window through which the programmer can shine ultraviolet (UV) radiation to erase the chip's contents. For this reason, EPROM is also referred to as UV-erasable EPROM or simply UV-EPROM.

Programming a UV-EPROM

To program a UV-EPROM chip, the following steps must be taken:

1. Its contents must be erased. To erase a chip, remove it from its socket on the system board and place it in EPROM erasure equipment to expose it to UV radiation for 5—20 minutes.
2. Program the chip. To program a UV-EPROM chip, place it in the ROM burner (programmer). To burn code or data into EPROM, the ROM burner uses 12.5 volts or higher, depending on the EPROM type. This voltage is referred to as V_{pp} in the UV-EPROM data sheet.
3. Place the chip back into its socket on the system board.

As can be seen from the above steps, not only is there an EPROM programmer (burner), but there is also separate EPROM erasure equipment. The main problem, and indeed the major disadvantage of UV-EPROM, is that it cannot be erased and programmed while it is in the system board. To provide a solution to this problem, EEPROM was invented. '

EEPROM (electrically erasable programmable ROM)

EEPROM has several advantages over EPROM, such as the fact that its method of erasure is electrical and therefore instant. as opposed to the 20-minute erasure time required for UV-EPROM.

In addition, in EEPROM one can select which byte to be erased, in contrast to UV-EPROM, in which the entire contents of ROM are erased.

However, the main advantage of EEPROM is that one can program and erase its contents while it is still in the system board. it does not require physical removal of the memory chip from its socket. In other words, unlike UV-EPROM, EEPROM does not require an external erasure and programming device.

To utilize EEPROM fully, the designer must incorporate the circuitry to program the EEPROM into the system board. In general, the cost per bit for EEPROM is much higher than for UV-EPROM.

Flash memory EPROM

Since the early 1990s, Flash EPROM has become a popular user-programmable memory chip. and for good reasons.

- First, the erasure of the entire contents takes less than a second, or one might say in a flash, hence its name, Flash memory.
- In addition, the erasure method is electrical, and for this reason it is sometimes referred to as Flash EEPROM. To avoid confusion, it is commonly called Flash memory.

The major difference between EEPROM and Flash memory is that when Flash memory's contents are erased, the entire device is erased, in contrast to EEPROM, where one can erase a desired byte.

Although in many Flash memories recently made available the contents are divided into blocks and the erasure can be done block by block, unlike EEPROM, Flash memory has no byte erasure option.

Because Flash memory can be programmed while it is in its socket on the system board, it is widely used to upgrade the BIOS ROM of the PC. Some designers believe that Flash memory will replace the hard disk as a mass storage medium.

This would increase the performance of the computer tremendously, since Flash memory is semiconductor memory with access time in the range of 100 ns compared with disk access time in the range of tens of milliseconds. For this to happen, Flash memory's program/erase cycles must become infinite, just like hard disks.

Program/erase cycle refers to the number of times that a chip can be erased and reprogrammed before it becomes unusable. At this time, the program/erase cycle is 100,000 for Flash and EEPROM, 1000 for UV-EPROM, and infinite for RAM and disks.

Mask ROM

Mask ROM refers to a kind of ROM in which the contents are programmed by the IC manufacturer. In other words, it is not a user-programmable ROM.

The term mask is used in IC fabrication. Since the process is costly, mask ROM is used when the needed volume is high (hundreds of thousands) and it is absolutely certain that the contents will not change.

It is common practice to use UV-EPROM or Flash for the development phase of a project, and only afier the code/data have been finalized is the mask version of the product ordered.

The main advantage of mask ROM is its cost, since it is significantly cheaper than other kinds of ROM, but if an error is found in the data/code, the entire batch must be thrown away. It must be noted that all ROM memories have 8 bits for data pins; therefore, the organization is x8.

ROM - Internal Circuit - consists of an array of semiconductor devices that are interconnected to store an array of binary data. Figure shows a 'n' i/p, 'm' o/p ROM, with $n = 3$ & $m = 4$. The no: i/p lines represent address lines & no: of o/p lines the no: of bits in each location (ie word size).

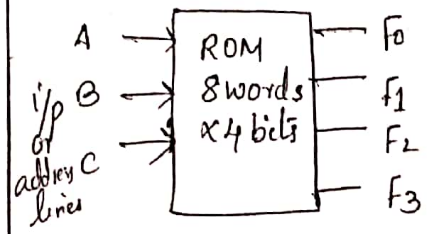


fig. 1. - 8X4 ROM

Since 3 address lines. 2^3 different locations (ie 8 different words can be stored) ie the data o/p ~~value~~ can take ~~a different~~ is $F_3 F_2 F_1 F_0$ (MSB) (LSB) any value as stored in ROM.

The address ranges from 000 to 111

It's internal circuit usually consists of a decoder and a memory array. The decoder o/p select one of the o/p word (ie the memory location) as per the i/p address and the bit pattern in the location is given as o/p by the o/p lines.

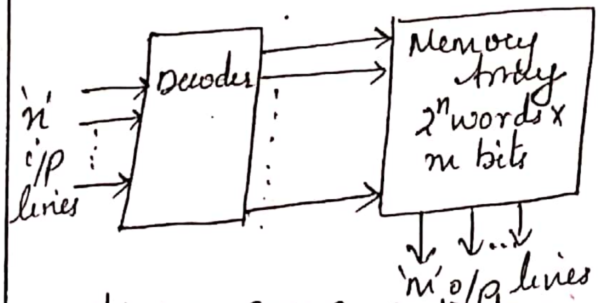


fig: 2 - ROM Basic structure

The detailed internal structure of the array with switching element as shown in fig. 3. The decoder o/p is shown with respective minterms for a 3 address line ROM, ie 3 inputs to the decoder. The decoder o/p line is called as word line. If the ROM c/s of 4 o/p lines, that can be drawn with pull down resistors at the top as in fig. 3. A switching element is placed at the intersection of a word line & an o/p line if the corresponding minterm is to be included in the o/p function. Otherwise switching elem is not connected. ~~If an o/p line~~ If a switching element connects an o/p line to a word line which is 1, the o/p value will be 1. otherwise the pull down resistors at the top cause the o/p line to be 0. So the switching elements in the memory array form an OR gate for each of the o/p functions. Let the o/p functions of the ROM in fig 3 is as follows.

$F_0 = \sum m(0,1,4,6)$, $F_1 = \sum m(2,3,4,6)$; $F_2 = \sum m(0,1,2,6)$

$F_3 = \sum m(2,3,5,6,7)$.

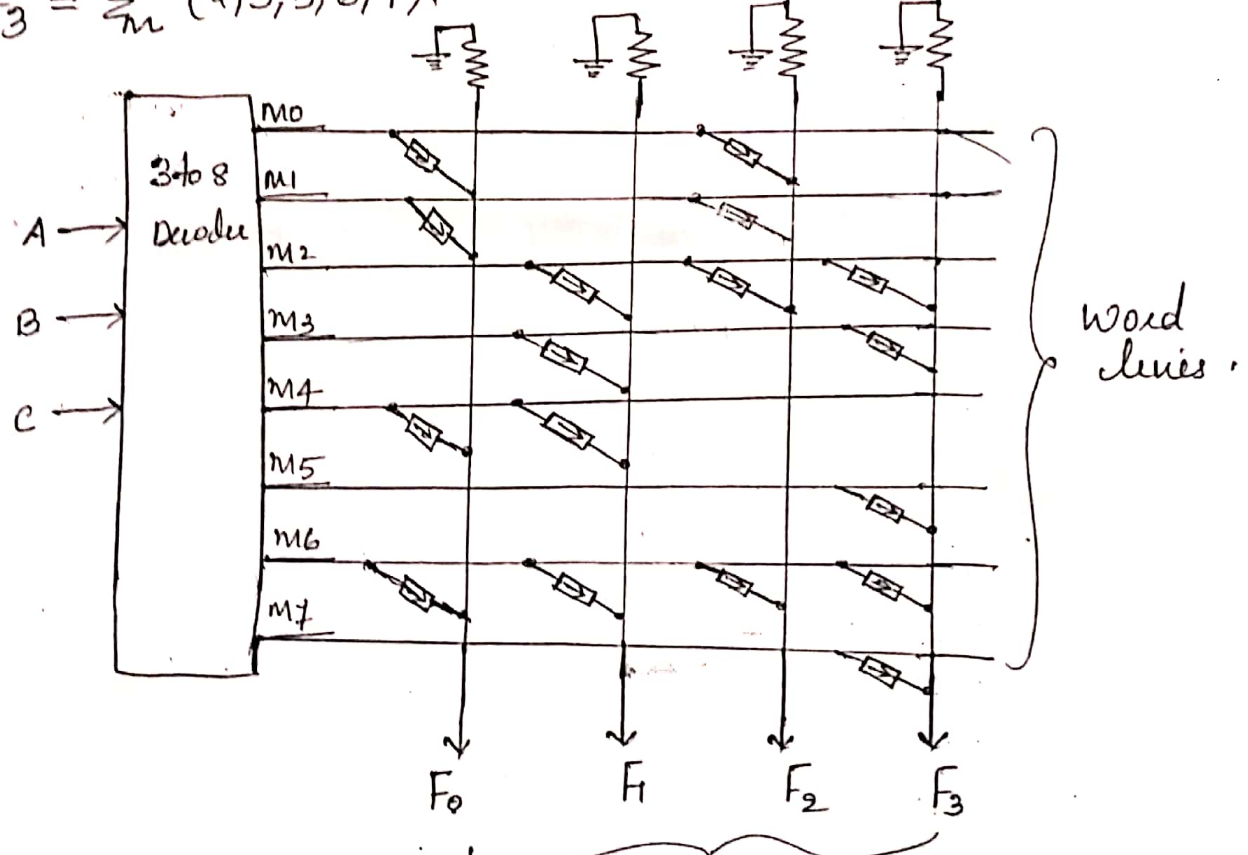


Fig : 3 — Memory Array of a ~~[illegible]~~ 8 word x 4 bit ROM.

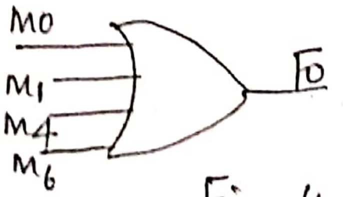
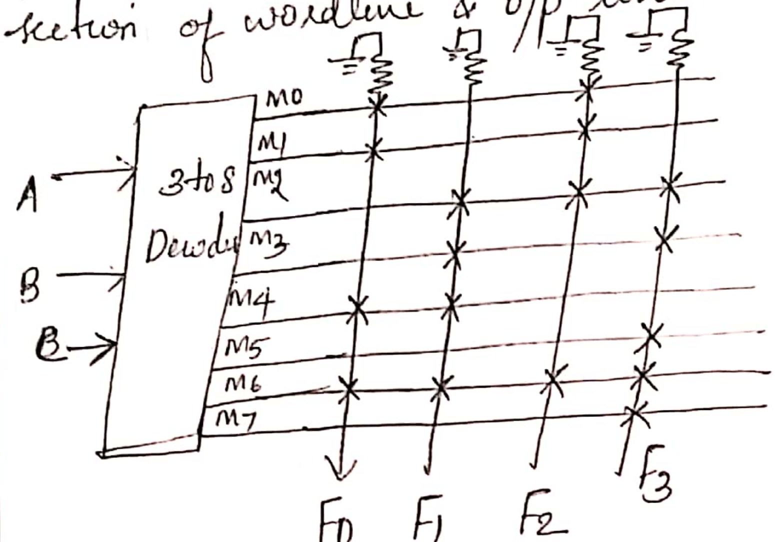
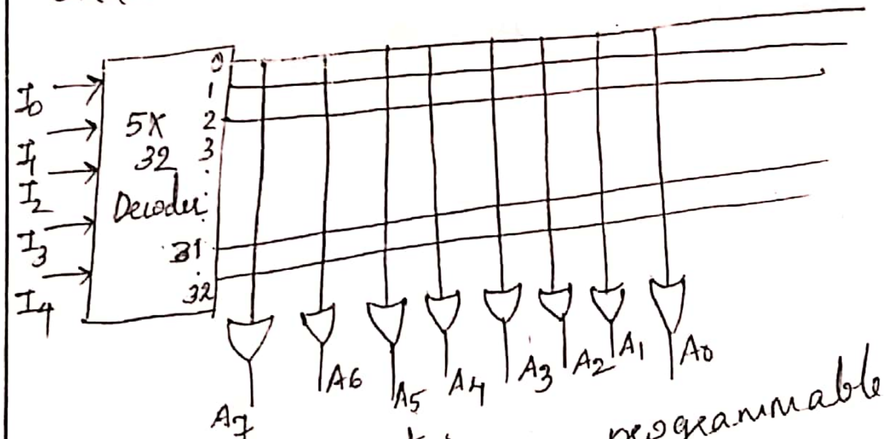


Fig. 4 — Equivalent OR gate for F0.

The above circuit can also be drawn by indicating the present minterms using a 'X' symbol at the point of inter section of wordline & o/p line.



Since the memory array of ROM is an OR array a 32X8 ROM can be shown as below. (32 segments or o/p lines with 8 bits in each location)



so 256 intersections are programmable.

(Q) Design a combinational circuit using a ROM. Circuit accepts a 3 bit number and generates an o/p binary number equal to square of the i/p number.

i/ps are of 3 bits, let $A_2 \cdot A_1 A_0$ so numbers can be from 0 to 7 in decimal i.e 000 to 111. o/p is square of respective numbers so max value of o/p in decimal is 49. no: of o/p bits is 6.

Scanned with CamScanner

I/p's			Decimal o/p.		binary o/p					
					2^5	2^4	2^3	2^2	2^1	2^0
A_2	A_1	A_0			B_5	B_4	B_3	B_2	B_1	B_0
0	0	0	0	$\longrightarrow$	0	0	0	0	0	0
0	0	1	1	$\longrightarrow$	0	0	0	0	0	1
0	1	0	4	$\longrightarrow$	0	0	0	1	0	0
0	1	1	9	$\longrightarrow$	0	0	1	0	0	1
1	0	0	~~4~~ 16	$\longrightarrow$	0	1	0	0	0	0
1	0	1	25	$\longrightarrow$	0	1	~~0~~1	0	0	1
1	1	0	36	$\longrightarrow$	1	0	0	1	0	0
1	1	1	49	$\longrightarrow$	1	1	0	0	0	1

From the table,

$B_0 = A_0$

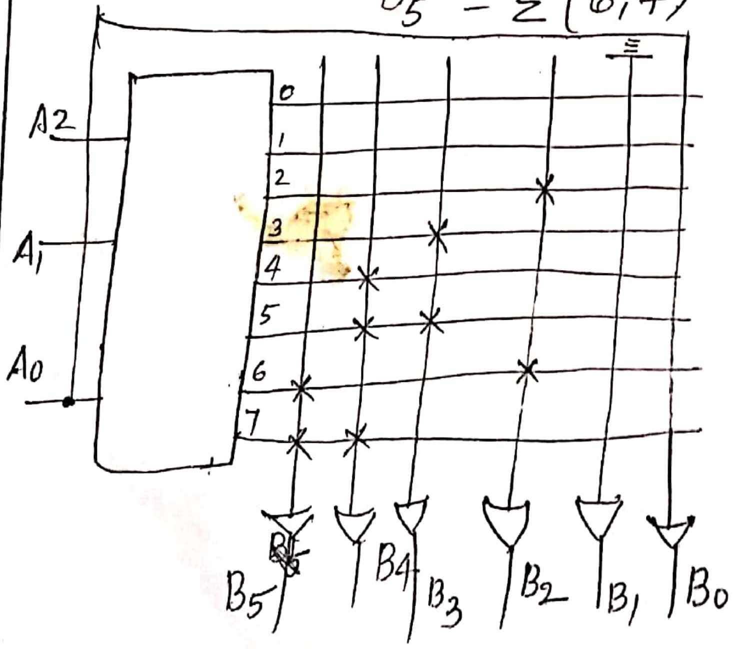
$B_1 = 0$

$B_2 = \Sigma(2,6)$

$B_3 = \Sigma(3,5)$

$B_4 = \Sigma(4,5,7)$

$B_5 = \Sigma(6,7)$



Programmable Logic Devices: (PLD)

It is the general name for a digital Integrated Circuit ~~capable of~~ that can be programmed for a variety of different logic functions. It can be combinational type or sequential type. A PLD may contain thousands of gates & flipflops. Thus a single PLD can replace a large no: of integrated circuits, which lowers cost. Another advantage is that in PLD change in design can be made by changing the program without changing the wiring.

Programmable Logic Array (PLA) – is a kind of PLD which does the same function as a ROM; but with different internal structure. Here decoder is replaced by an AND array which realizes selected product terms of the input variables. It's then followed by an OR array, which ORs the product term needed to form the o/p functions. so PLA implements SOP expressions, while a ROM directly implements the truthtable.

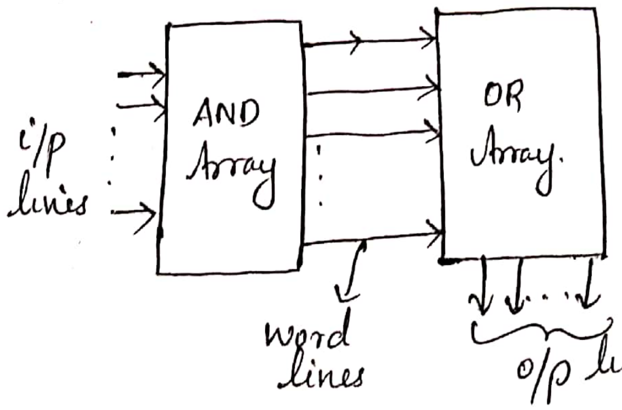
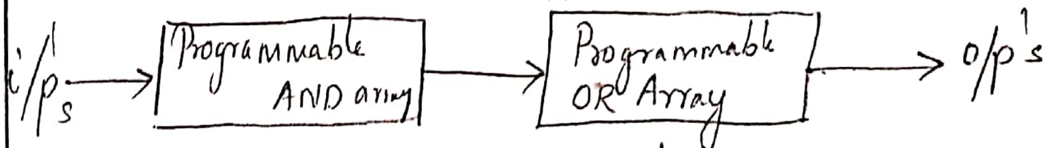


Fig: 4
PLA - internal structure



PAL structure

eg:- full Adder using PLA.

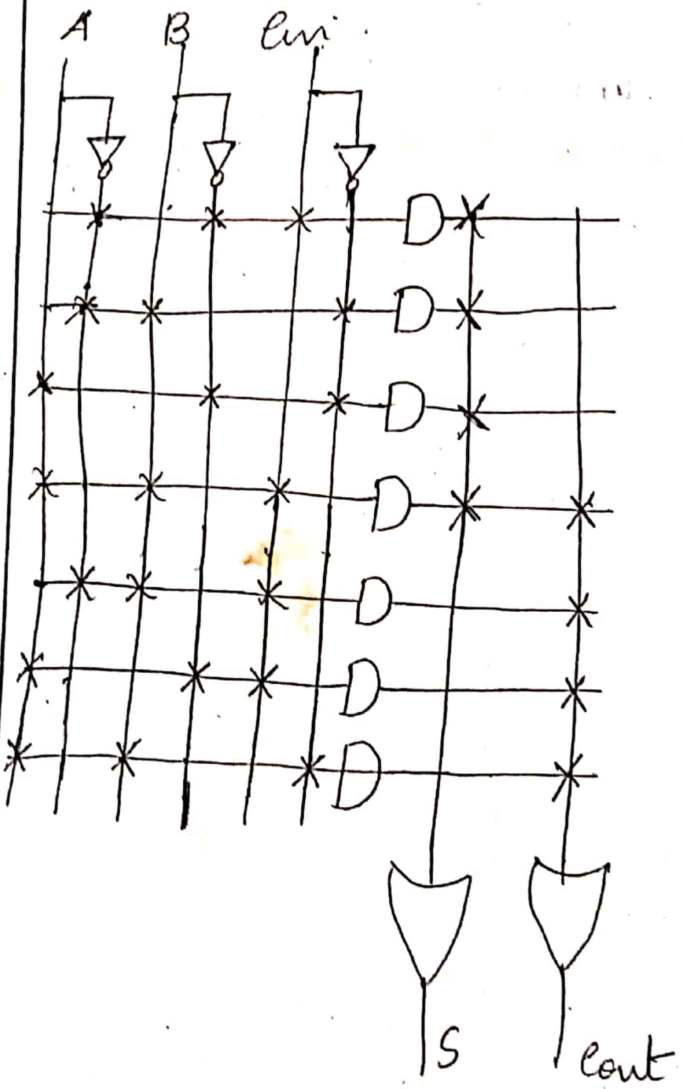
$S = \bar{A}\bar{B}\,Cin + \bar{A}B\overline{Cin} + A\bar{B}\,\overline{Cin} + AB\,Cin$

~~Cout = AB + ACin + BCin~~

(i.e. from truthtable)

$Cout = \bar{A}B\,Cin + A\bar{B}\,Cin + AB\,\overline{Cin} + AB\,Cin.$

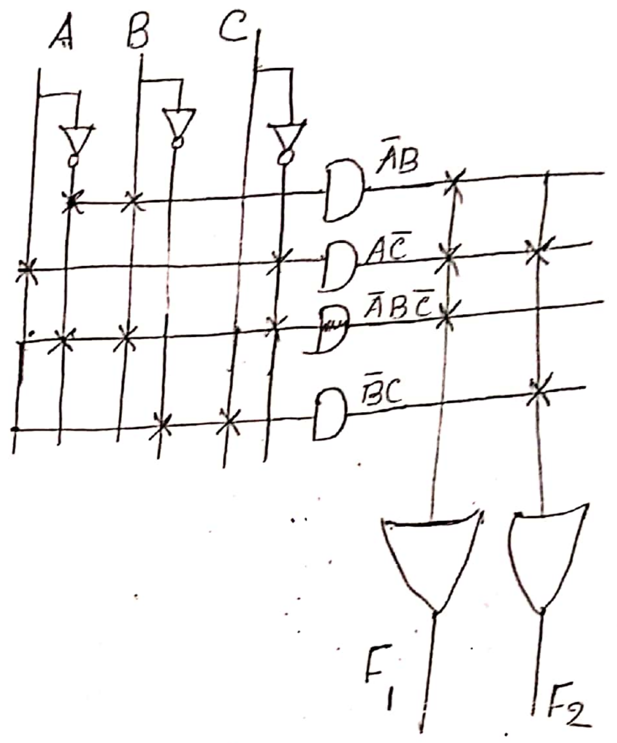
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



9) Implement using PLA.

$F_1 = \bar{A}B + A\bar{C} + \bar{A}B\bar{C}$, $F_2 = A\bar{C} + \bar{B}C$

pdt term	I/p's A	B	C	o/p's F1(T)	F2(C)
$\bar{A}B$	0	1	-	1	-
$A\bar{C}$	1	-	0	1	1
$\bar{A}B\bar{C}$	0	1	0	1	-
$\bar{B}C$	-	0	1	-	1



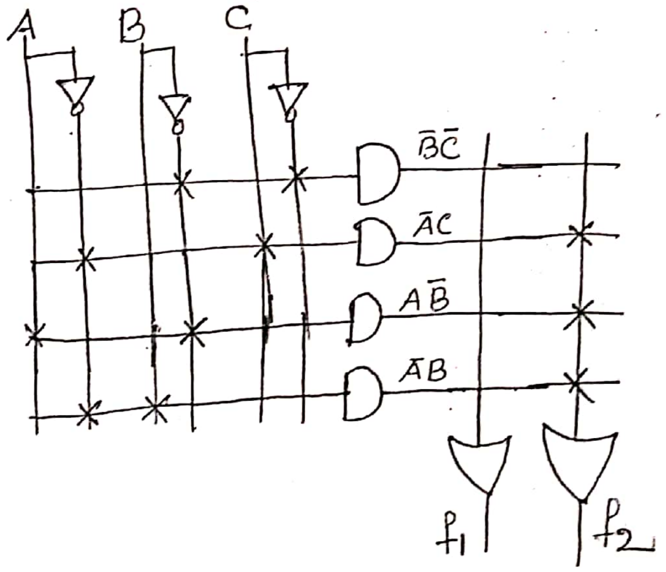
② $F_1(A,B,C) = \Sigma m(0,1,3,4)$
$F_2(A,B,C) = \Sigma m(1,2,3,4,5)$

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

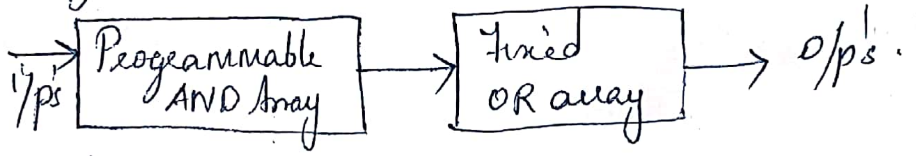
$F_1 = \bar{B}\bar{C} + \bar{A}C$

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

$F_2 = \bar{A}B + A\bar{B} + \bar{A}C$.



Programmable Array Logic (PAL)

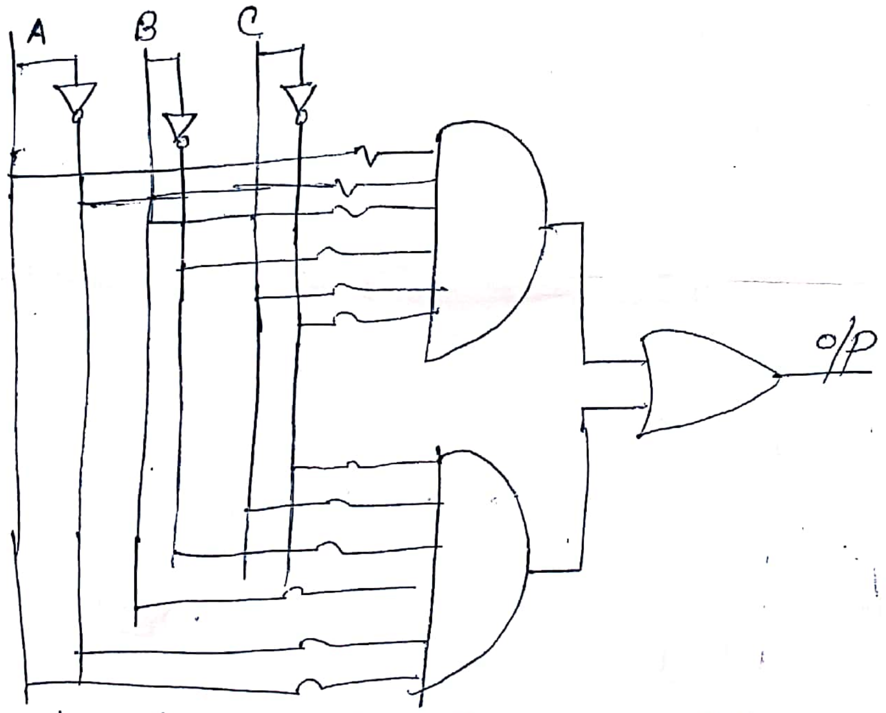


✓ less expensive

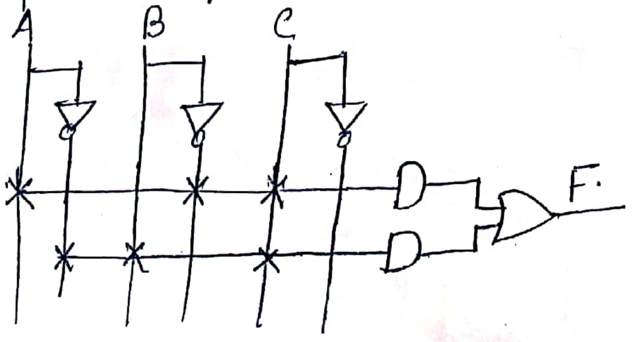
✓ Easier to program

It's a special case of PLA in which the AND array is programmable & OR array is fixed.

Basic structure

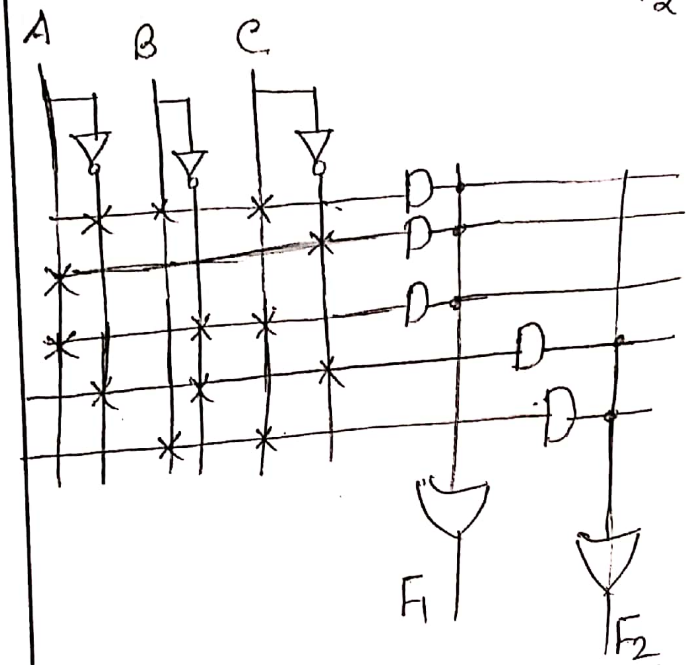


1) Implement $F = A\bar{B}C + \bar{A}BC$ using PAL.



Q) Implement using PAL. $F_1 = \bar{A}BC + A\bar{C} + A\bar{B}C$

$F_2 = \bar{A}\bar{B}\bar{C} + BC$.



Field Programmable Gate Arrays (FPGA).

It Consists of an array of identical logic cells called Configurable logic blocks (CLBs) surrounded by a ring of i/p-o/p interface blocks, which connects the CLB signals to IC pins. The users can program the functions realized by each logic cells & the connections between the cells. Generally an FPGA may contain function generators, flip flops, multiplexers etc.

~~VHDL~~

Implementation of half adder, and gate, or gate, full adder using (VHDL).

Preset and Clear of counters

Preset will make the output high irrespective of input conditions if taken low. **Clear** will make the output low irrespective of input conditions if taken low. If not in use both **preset and clear** pins are tied to Logic High.

Synchronous Counter

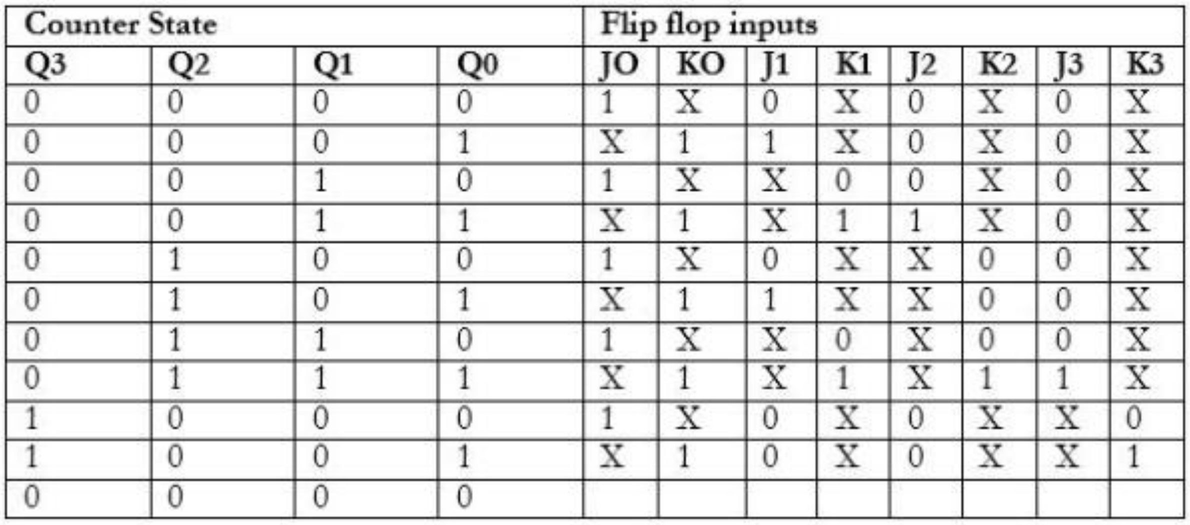
In the previous Asynchronous binary counter, we saw that the output of one counter stage is connected directly to the clock input of the next counter stage and so on along the chain. The result of this is that the Asynchronous counter suffers from what is known as “Propagation Delay” in which the timing signal is delayed a fraction through each flip-flop.

However, with the **Synchronous Counter**, the external clock signal is connected to the clock input of EVERY individual flip-flop within the counter so that all of the flip-flops are clocked together simultaneously (in parallel) at the same time giving a fixed time relationship. In other words, changes in the output occur in “synchronisation” with the clock signal. The result of this synchronisation is that all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay.

Modulo 10 or Decade or BCD Counter

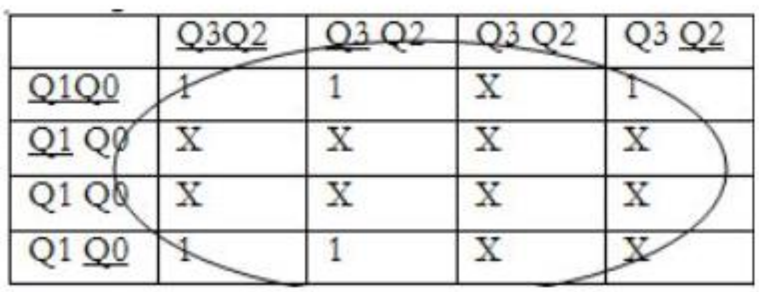
Design mod-10 synchronous counter using JK Flip Flops.Check for the lock out condition.If so,how the lock-out condition can be avoided? Draw the neat state diagram and circuit diagram with Flip Flops.

1) Truth Table:



Counter State				Flip flop inputs							
Q3	Q2	Q1	Q0	JO	KO	J1	K1	J2	K2	J3	K3
0	0	0	0	1	X	0	X	0	X	0	X
0	0	0	1	X	1	1	X	0	X	0	X
0	0	1	0	1	X	X	0	0	X	0	X
0	0	1	1	X	1	X	1	1	X	0	X
0	1	0	0	1	X	0	X	X	0	0	X
0	1	0	1	X	1	1	X	X	0	0	X
0	1	1	0	1	X	X	0	X	0	0	X
0	1	1	1	X	1	X	1	X	1	1	X
1	0	0	0	1	X	0	X	0	X	X	0
1	0	0	1	X	1	0	X	0	X	X	1
0	0	0	0								

2) K-maps:



	$\overline{Q3}\,\overline{Q2}$	$\overline{Q3}\,Q2$	$Q3\,Q2$	$Q3\,\overline{Q2}$
$\overline{Q1}\,\overline{Q0}$	1	1	X	1
$\overline{Q1}\,Q0$	X	X	X	X
$Q1\,Q0$	X	X	X	X
$Q1\,\overline{Q0}$	1	1	X	X

J0=1

	$\underline{Q3Q2}$	$\underline{Q3}$ Q2	Q3 Q2	Q3 $\underline{Q2}$
$\underline{Q1Q0}$	X	X	X	X
$\underline{Q1}$ Q0	1	1	X	1
Q1 Q0	1	1	X	X
Q1 $\underline{Q0}$	X	X	X	X

K0=1

	$\underline{Q3Q2}$	$\underline{Q3}$ Q2	Q3 Q2	Q3 $\underline{Q2}$
$\underline{Q1Q0}$	0	0	X	0
$\underline{Q1}$ Q0	1	1	X	0
Q1 Q0	X	X	X	X
Q1 $\underline{Q0}$	X	X	X	X

J1=Q0 Q3

	$\underline{Q3Q2}$	$\underline{Q3}$ Q2	Q3 Q2	Q3 $\underline{Q2}$
$\underline{Q1Q0}$	X	X	X	X
$\underline{Q1}$ Q0	X	X	X	X
Q1 Q0	1	1	X	X
Q1 $\underline{Q0}$	0	0	X	X

K1=Q0

	$\underline{Q3Q2}$	$\underline{Q3}$ Q2	Q3 Q2	Q3 $\underline{Q2}$
$\underline{Q1Q0}$	0	X	X	0
$\underline{Q1}$ Q0	0	X	X	0
Q1 Q0	1	X	X	X
Q1 $\underline{Q0}$	0	X	X	X

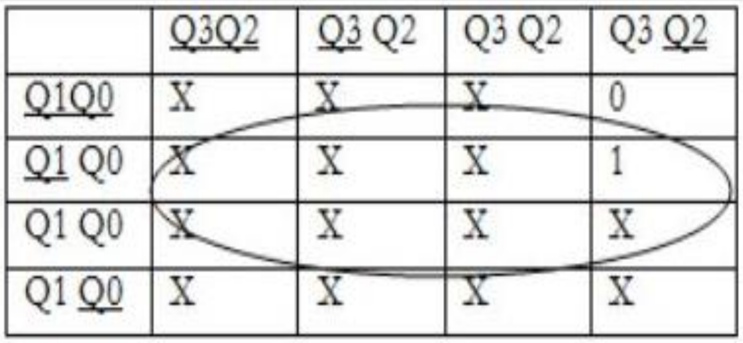
J2=Q0 Q1

	$\underline{Q3Q2}$	$\underline{Q3}$ Q2	Q3 Q2	Q3 $\underline{Q2}$
$\underline{Q1Q0}$	X	0	X	X
$\underline{Q1}$ Q0	X	0	X	X
Q1 Q0	X	1	X	X
Q1 $\underline{Q0}$	X	0	X	X

K2=Q0 Q1

	$\underline{Q3Q2}$	$\underline{Q3}$ Q2	Q3 Q2	Q3 $\underline{Q2}$
$\underline{Q1Q0}$	0	0	X	X
$\underline{Q1}$ Q0	0	0	X	X
Q1 Q0	0	1	X	X
Q1 $\underline{Q0}$	0	0	X	X

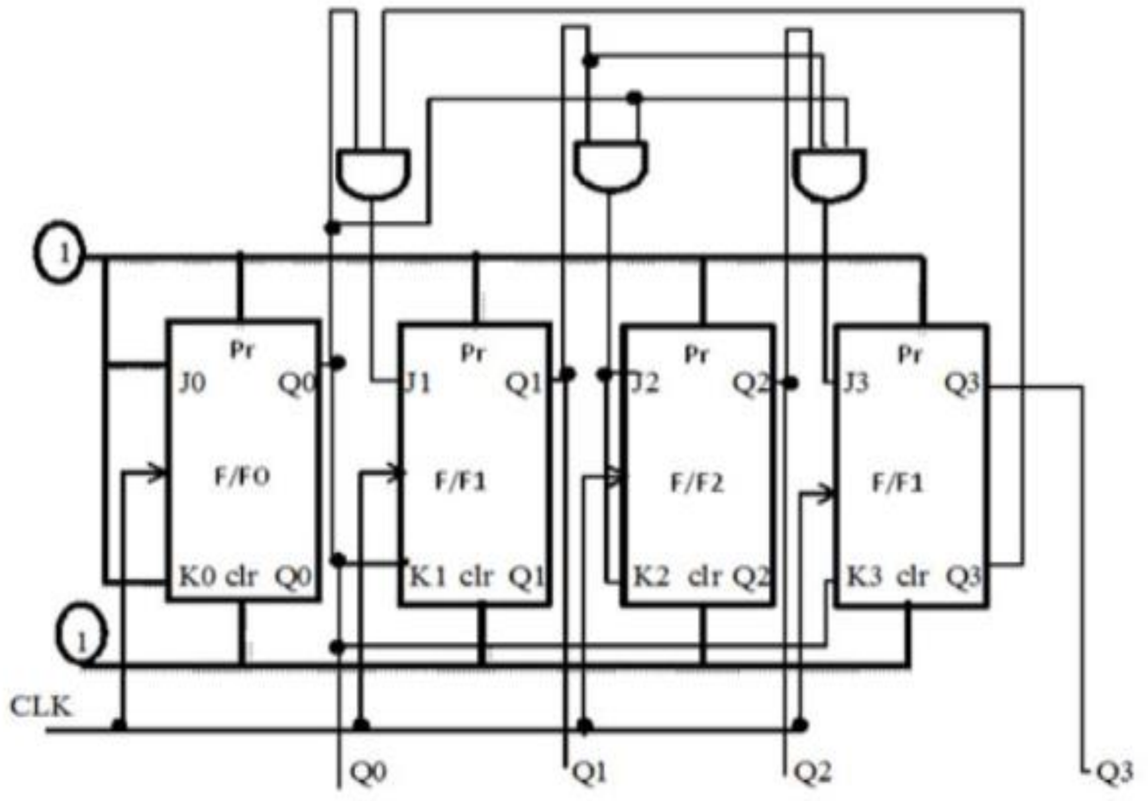
J3=Q0 Q1 Q2



	$\underline{Q3Q2}$	$\underline{Q3}$ Q2	Q3 Q2	Q3 $\underline{Q2}$
$\underline{Q1Q0}$	X	X	X	0
$\underline{Q1}$ Q0	X	X	X	1
Q1 Q0	X	X	X	X
Q1 $\underline{Q0}$	X	X	X	X

K3=Q0

2) Logic Circuit:



3) Lock out condition:

- In the above counter the logic states 1010, 1011, 1100, 1101, 1110 and 1111 are not used. If by chance, the counter happens to find itself in any one of the unused states, its next state would not be known. It may just be possible that the counter might go from one unused state to another and never arrive at a used state. A counter whose unused states have this feature is said to suffer from LOCK OUT.
- To avoid lock out and make sure that at the starting point the counter is in its initial state or it comes to its initial state within few clock cycles, external logic circuitry is to be provided and so we design the counter assuming the next state to be the initial state, from each unused states.

Sequential Machines

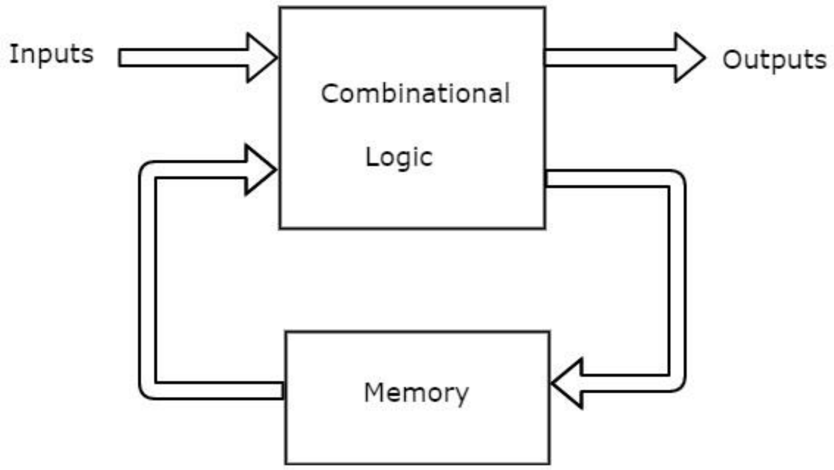
We know that synchronous sequential circuits change affect their states for every positive 0r negative transition of the clock signal based on the input. So, this behavior of synchronous sequential circuits can be represented in the graphical form and it is known as **state diagram**.
A synchronous sequential circuit is also called as **Finite State Machine** FSM, if it has finite number of states. There are two types of FSMs.

- Mealy State Machine
- Moore State Machine

Now, let us discuss about these two state machines one by one.

Mealy State Machine

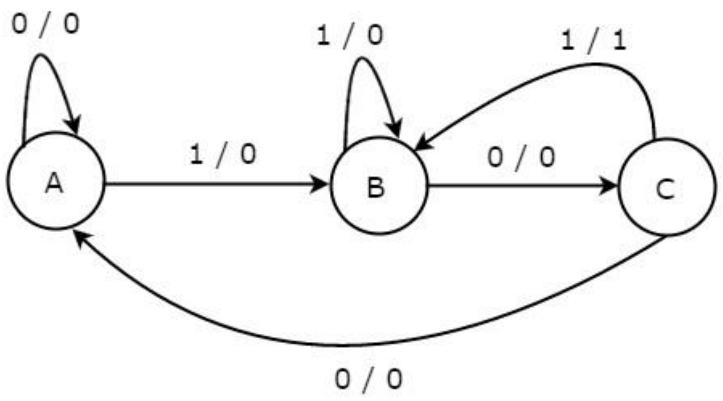
A Finite State Machine is said to be Mealy state machine, if outputs depend on both present inputs & present states. The **block diagram** of Mealy state machine is shown in the following figure.



As shown in figure, there are two parts present in Mealy state machine. Those are combinational logic and memory. Memory is useful to provide some or part of previous outputs and present states as inputs of combinational logic. So, based on the present inputs and present states, the Mealy state machine produces outputs. Therefore, the outputs will be valid only at positive or negative transition of the clock signal.

Mealy Diagram –

$$\begin{tikzcd}
A \arrow[loop above, "0/0"] \arrow[r, "1/0"] & B \arrow[loop above, "1/0"] \arrow[r, "0/0"] & C \arrow[l, bend right, "1/1"'] \arrow[ll, bend left, "0/0"]
\end{tikzcd}$$

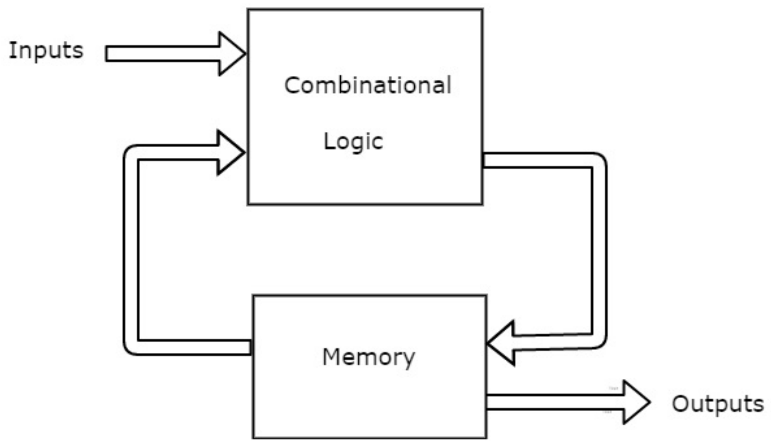


In the above figure, there are three states, namely A, B & C. These states are labelled inside the circles & each circle corresponds to one state. Transitions between these states are represented with directed lines. Here, 0 / 0, 1 / 0 & 1 / 1 denotes **input / output**. In the above figure, there are two transitions from each state based on the value of input, x.

In general, the number of states required in Mealy state machine is less than or equal to the number of states required in Moore state machine. There is an equivalent Moore state machine for each Mealy state machine.

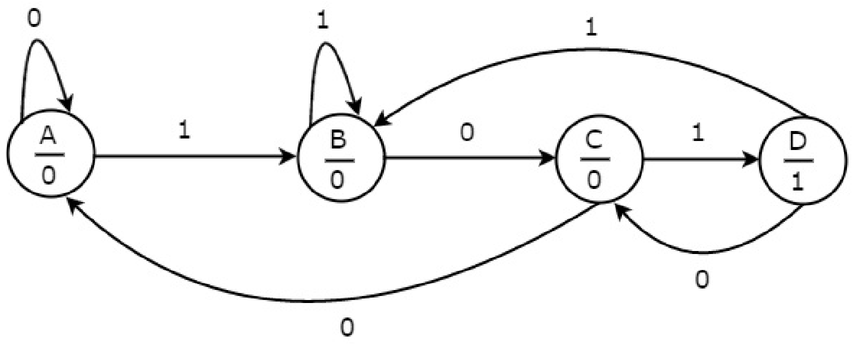
Moore State Machine

A Finite State Machine is said to be Moore state machine, if outputs depend only on present states. The **block diagram** of Moore state machine is shown in the following figure.



As shown in figure, there are two parts present in Moore state machine. Those are combinational logic and memory. In this case, the present inputs and present states determine the next states. So, based on next states, Moore state machine produces the outputs. Therefore, the outputs will be valid only after transition of the state.

The **state diagram** of Moore state machine is shown in the following figure.



In the above figure, there are four states, namely A, B, C & D. These states and the respective outputs are labelled inside the circles. Here, only the input value is labeled on each transition. In the above figure, there are two transitions from each state based on the value of input, x.

In general, the number of states required in Moore state machine is more than or equal to the number of states required in Mealy state machine. There is an equivalent Mealy state machine for each Moore state machine. So, based on the requirement we can use one of them.

Moore Machine –

1. Output depends only upon present state.
2. If input changes, output does not change.
3. More number of states are required.
4. There is more hardware requirement.
5. They react slower to inputs(One clock cycle later)
6. Synchronous output and state generation.
7. Output is placed on states.
8. Easy to design.

Mealy Machine –

1. Output depends on present state as well as present input.
2. If input changes, output also changes.
3. Less number of states are required.
4. There is less hardware requirement.
5. They react faster to inputs.
6. Asynchronous output generation.
7. Output is placed on transitions.
8. It is difficult to design.

VHDL Introduction

VHDL stands for very high-speed integrated circuit hardware description language. It is a programming language used to model a digital system by dataflow, behavioral and structural style of modeling.

Describing a Design

In VHDL an entity is used to describe a hardware module. An entity can be described using,

- Entity declaration
- Architecture
- Configuration
- Package declaration
- Package body

Entity Declaration

It defines the names, input output signals and modes of a hardware module.

Syntax −

```
entity entity_name is
   Port declaration;
end entity_name;
```

An entity declaration should start with ‘entity’ and end with ‘end’ keywords. The direction will be input, output or inout.

In	Port can be read
Out	Port can be written

Inout	Port can be read and written
Buffer	Port can be read and written, it can have only one source.

Architecture −

Architecture can be described using structural, dataflow, behavioral or mixed style.

Syntax −

```
architecture architecture_name of entity_name
architecture_declarative_part;

begin
   Statements;
end architecture_name;
```

Here, we should specify the entity name for which we are writing the architecture body. The architecture statements should be inside the ‘begin’ and ‘énd’ keyword. Architecture declarative part may contain variables, constants, or component declaration.

Data Flow Modeling

In this modeling style, the flow of data through the entity is expressed using concurrent (parallel) signal. The concurrent statements in VHDL are WHEN and GENERATE.

Besides them, assignments using only operators (AND, NOT, +, *, sll, etc.) can also be used to construct code.

Finally, a special kind of assignment, called BLOCK, can also be employed in this kind of code.

In concurrent code, the following can be used −

- Operators
- The WHEN statement (WHEN/ELSE or WITH/SELECT/WHEN);
- The GENERATE statement;
- The BLOCK statement

Behavioral Modeling

In this modeling style, the behavior of an entity as set of statements is executed sequentially in the specified order. Only statements placed inside a PROCESS, FUNCTION, or PROCEDURE are sequential.

PROCESSES, FUNCTIONS, and PROCEDURES are the only sections of code that are executed sequentially.

However, as a whole, any of these blocks is still concurrent with any other statements placed outside it.

One important aspect of behavior code is that it is not limited to sequential logic. Indeed, with it, we can build sequential circuits as well as combinational circuits.

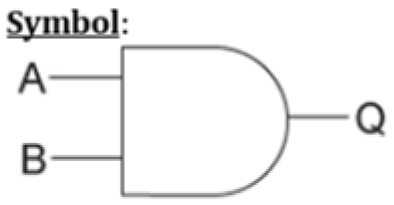
The behavior statements are IF, WAIT, CASE, and LOOP. VARIABLES are also restricted and they are supposed to be used in sequential code only. VARIABLE can never be global, so its value cannot be passed out directly.

Structural Modeling

In this modeling, an entity is described as a set of interconnected components. A component instantiation statement is a concurrent statement. Therefore, the order of these statements is not important. The structural style of modeling describes only an interconnection of components (viewed as black boxes), without implying any behavior of the components themselves nor of the entity that they collectively represent.

In Structural modeling, architecture body is composed of two parts − the declarative part (before the keyword begin) and the statement part (after the keyword begin).

Logic Operation – AND GATE



X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

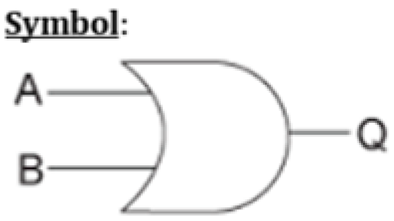
VHDL Code:

```
Library ieee;
use ieee.std_logic_1164.all;

entity and1 is
   port(x,y:in bit ; z:out bit);
end and1;

architecture virat of and1 is
begin
   z<=x and y;
end virat;
```

Logic Operation – OR Gate



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

```
VHDL Code:
Library ieee;
use ieee.std_logic_1164.all;

entity or1 is
  port(x,y:in bit ; z:out bit);
end or1;

architecture virat of or1 is
begin
  z<=x or y;
end virat;
```

VHDL Code for a Half-Adder

```
VHDL Code:

Library ieee;
use ieee.std_logic_1164.all;

entity half_adder is
  port(a,b:in bit; sum,carry:out bit);
end half_adder;

architecture data of half_adder is
begin
  sum<= a xor b;
  carry <= a and b;
end data;
```

VHDL Code for a Full Adder

```
Library ieee;
use ieee.std_logic_1164.all;

entity full_adder is port(a,b,c:in bit; sum,carry:out bit);
end full_adder;

architecture data of full_adder is
begin
```

```
   sum<= a xor b xor c;
   carry <= ((a and b) or (b and c) or (a and c));
end data;
```