UNIT 5 PROGRAMMABLE LOGIC DEVICES, MEMORY

Memories: ROM, PROM, EPROM, PLA, PLD,

5.1 INTRODUCTION

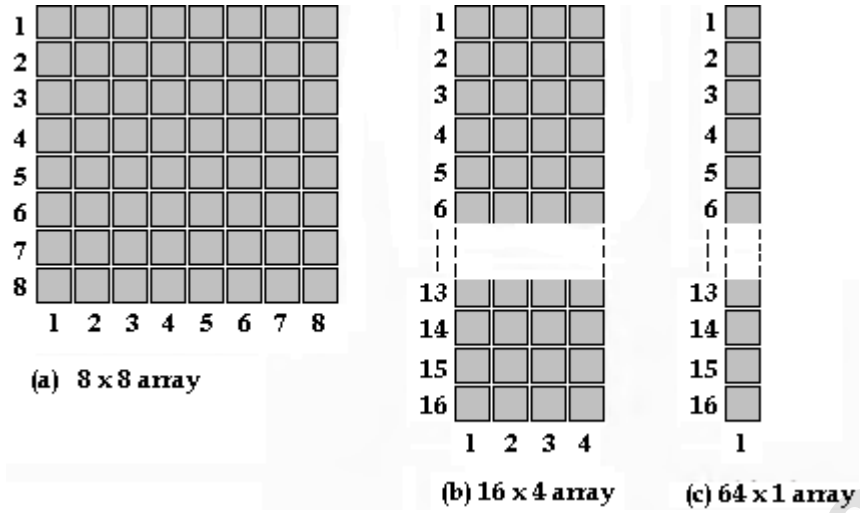
A memory unit is a collection of storage cells with associated circuits needed to transfer information in and out of the device. The binary information is transferred for storage and from which information is available when needed for processing. When data processing takes place, information from the memory is transferred to selected registers in the processing unit. Intermediate and final results obtained in the processing unit are transferred back to be stored in memory.

5.2 Units of Binary Data: Bits, Bytes, Nibbles and Words

As a rule, memories store data in units that have from one to eight bits. The smallest unit of binary data is the **bit**. In many applications, data are handled in an 8-bit unit called a **byte** or in multiples of 8-bit units. The byte can be split into two 4-bit units that are called **nibbles**. A complete unit of information is called a **word** and generally consists of one or more bytes. Some memories store data in 9-bit groups; a 9-bit group consists of a byte plus a parity bit.

5.3 Basic Semiconductor Memory Array

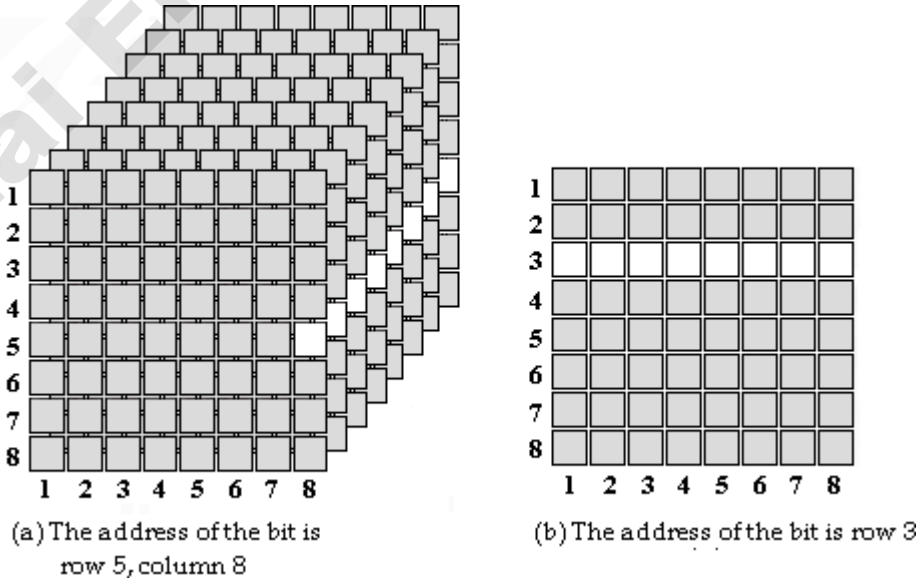
Each storage element in a memory can retain either a 1 or a 0 and is called a **cell**. Memories are made up of arrays of cells, as illustrated in Figure below using 64 cells as an example. Each block in the memory array represents one storage cell, and its location can be identified by specifying a row and a column.



A 64-cell memory array organized in three different ways

5.4 Memory Address and Capacity

The *location* of a unit of data in a memory array is called its **address**. For example, in Figure (a), the address of a bit in the 3-dimensional array is specified by the row and column. In Figure (b), the address of a byte is specified only by the row in the 2-dimensional array. So, as you can see, the address depends on how the memory is organized into units of data. Personal computers have random-access memories organized in bytes. This means that the smallest group of bits that can be addressed is eight.



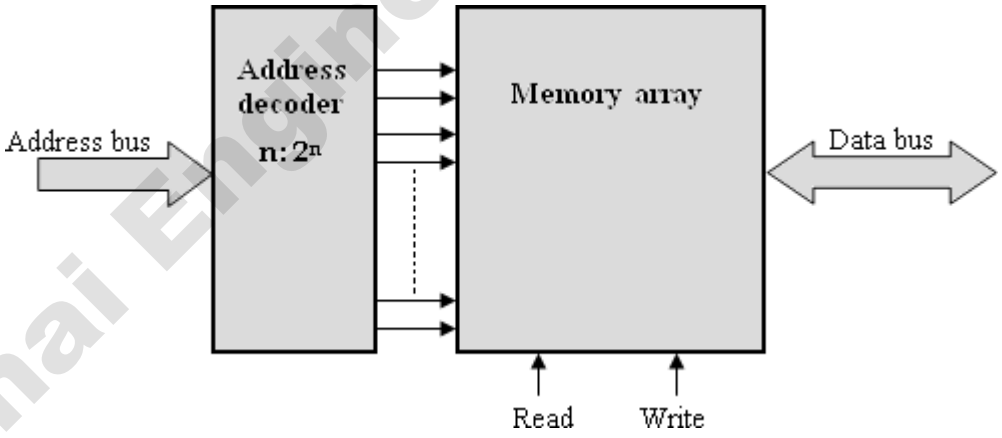
Examples of memory address

The **capacity** of a memory is the total number of data units that can be stored. For example, in the bit-organized memory array in Figure (a), the capacity is 64 bits. In the byte-organized memory array in Figure (b), the capacity is 8 bytes, which is also 64 bits. Computer memories typically have 256 MB (megabyte) or more of internal memory.

5.5 Basic Memory Operations

Since a memory stores binary data, data must be put into the memory and data must be copied from the memory when needed. The write operation puts data into a specified address in the memory, and the read operation copies data out of a specified address in the memory. The addressing operation, which is part of both the write and the read operations, selects the specified memory address.

Data units go into the memory during a write operation and come out of the memory during a read operation on a set of lines called the *data bus*. As indicated in Figure, the data bus is bidirectional, which means that data can go in either directional (into the memory or out of the memory).



Block diagram of memory operation

For a write or a read operation, an address is selected by placing a binary code representing the desired address on a set of lines called the address bus. The address code is decoded internally and the appropriate address is selected. The number of lines in the address bus depends on the capacity of the memory. For example, a 15-bit address code can select 32,768 locations (2^{15}) in the memory; a 16-bit address code can select 65,536 locations (2^{16}) in the memory and so on.

In personal computers a 32-bit address bus can select 4,294,967,296 locations (2^{32}), expressed as 4GB.

5.5.1 Write Operation

To store a byte of data in the memory, a code held in the address register is placed on the address bus. Once the address code is on the bus, the address decoder decodes the address and selects the specified location in the memory. The memory then gets a write command, and the data byte held in the data register is placed on the data bus and stored in the selected memory address, thus completing the write operation. When a new data byte is written into a memory address, the current data byte stored at that address is overwritten (replaced with a new data byte).

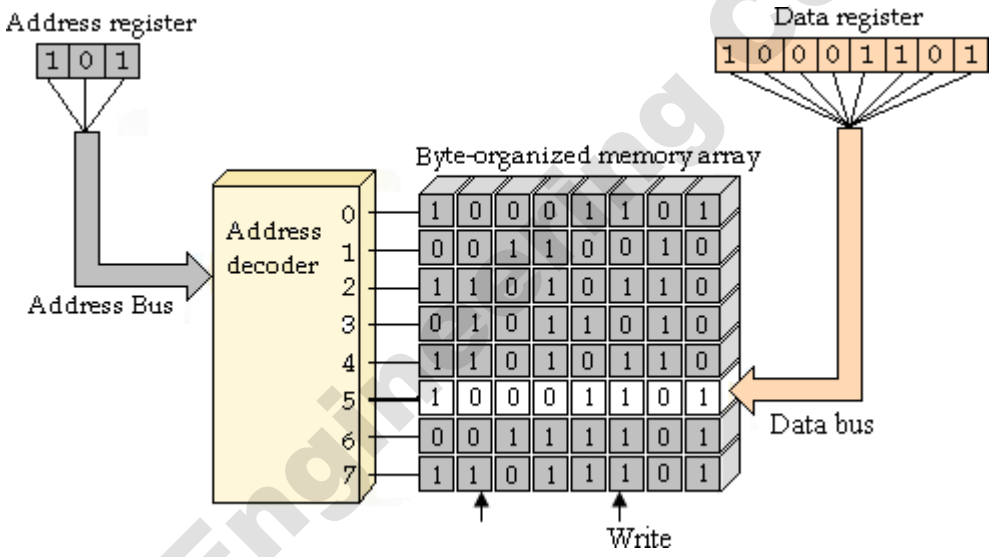


Illustration of the Write operation

5.5.2 Read Operation

A code held in the address register is placed on the address bus. Once the address code is on the bus, the address decoder decodes the address and selects the specified location in the memory. The memory then gets a read command, and a "copy" of the data byte that is stored in the selected memory address is placed on the data bus and loaded into the data register, thus completing the read operation. When a data byte is read from a memory address, it also remains stored at that address. This is called *nondestructive* read.

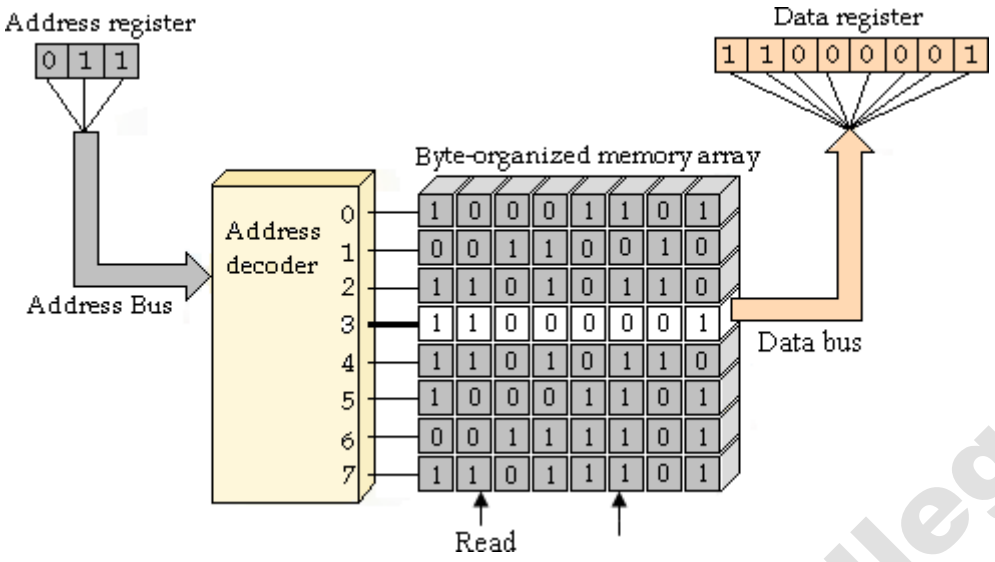


Illustration of the Read operation

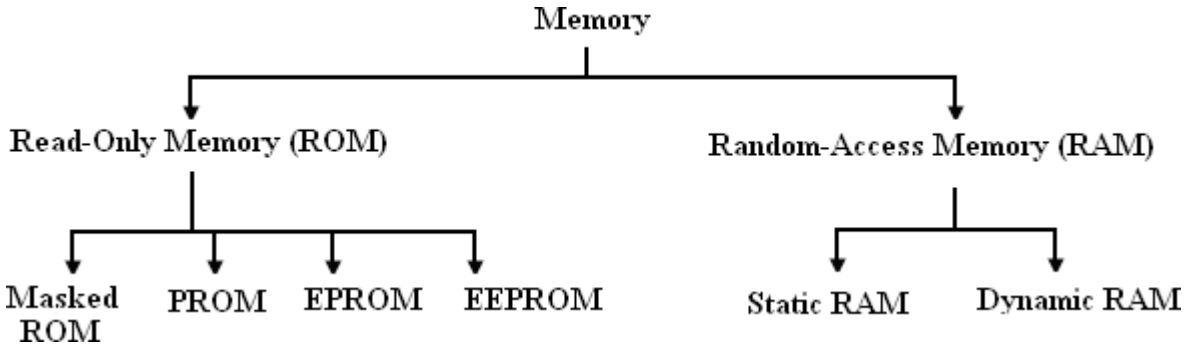
5.6 Classification of Memories

There are two types of memories that are used in digital systems:

- Random-Access Memory (RAM),
- Read-Only Memory (ROM).

RAM (random-access memory) is a type of memory in which all addresses are accessible in an equal amount of time and can be selected in any order for a read or write operation. All RAMs have both read and write capability. Because RAMs lose stored data when the power is turned off, they are ***volatile*** memories.

ROM (read-only memory) is a type of memory in which data are stored permanently or semi permanently. Data can be read from a ROM, but there is no write operation as in the RAM. The ROM, like the RAM, is a random-access memory but the term RAM traditionally means a random-access read/write memory. Because ROMs retain stored data even if power is turned off, they are ***nonvolatile*** memories.



Classification of memories

5.6.1 RANDOM-ACCESS MEMORIES (RAMS)

RAMs are read/write memories in which data can be written into or read from any selected address in any sequence. When a data unit is written into a given address in the RAM, the data unit previously stored at that address is replaced by the new data unit. When a data unit is read from a given address in the RAM, the data unit remains stored and is not erased by the read operation. This nondestructive read operation can be viewed as copying the content of an address while leaving the content intact.

A RAM is typically used for short-term data storage because it cannot retain stored data when power is turned off.

The two categories of RAM are the ***static* RAM** (SRAM) and the ***dynamic* RAM** (DRAM). Static RAMs generally use flip-flops as storage elements and can therefore store data indefinitely *as long as dc power is applied*. Dynamic RAMs use capacitors as storage elements and cannot retain data very long without the capacitors being recharged by a process called **refreshing**. Both SRAMs and DRAMs will lose stored data when dc power is removed and, therefore, are classified as ***volatile memories***.

Data can be read much faster from SRAMs than from DRAMs. However, DRAMs can store much more data than SRAMs for a given physical size and cost because the DRAM cell is much simpler, and more cells can be crammed into a given chip area than in the SRAM.

5.6.1.1 Static RAM (SRAM)

Storage Cell:

All static RAMs are characterized by flip-flop memory cells. As long as dc power is applied to a static memory cell, it can retain a 1 or 0 state indefinitely. If power is removed, the stored data bit is lost.

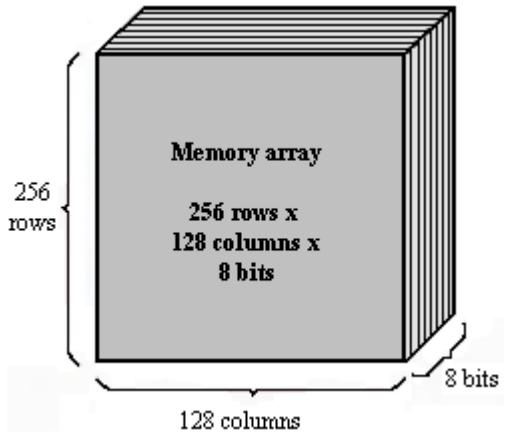
The cell is selected by an active level on the Select line and a data bit (l or 0) is written into the cell by placing it on the Data in line. A data bit is read by taking it off the Data out line.

Basic SRAM Organization:

Basic Static Memory Cell Array

The memory cells in a SRAM are organized in rows and columns. All the cells in a row share the same Row Select line. Each set of Data in and Data out lines go to each cell in a given column and are connected to a single data line that serves as both an input and output (Data I/O) through the data input and data output buffers.

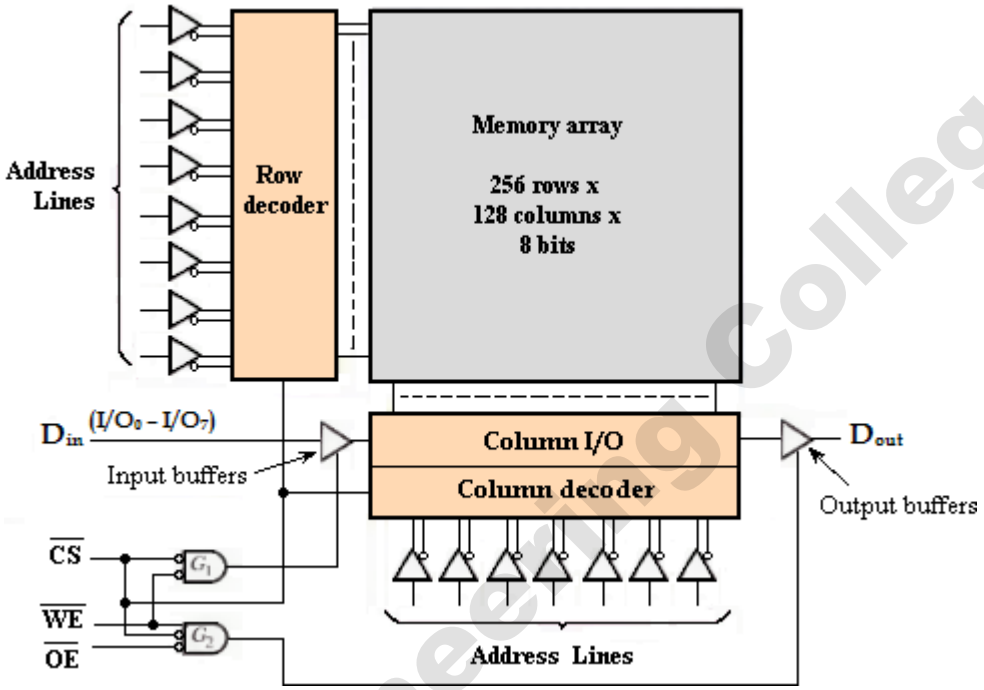
SRAM chips can be organized in single bits, nibbles (4 bits), bytes (8 bits), or multiple bytes (16, 24, 32 bits, etc.). The memory cell array is arranged in 256 rows and 128 columns, each with 8 bits as shown below. There are actually 2^{15} = 32,768 addresses and each address contains 8 bits. The capacity of this example memory is 32,768 bytes (typically expressed as 32 Kbytes).



Memory array configuration

Operation:

The SRAM works as follows. First, the chip select, CS, must be LOW for the memory to operate. Eight of the fifteen address lines are decoded by the row decoder to select one of the 256 rows. Seven of the fifteen address lines are decoded by the column decoder to select one of the 128 8-bit columns.



Memory block diagram

Read:

In the READ mode, the write enable input, WE' is HIGH and the output enable, OE_ is LOW. The input tri state buffers are disabled by gate **G_1**, and the column output tristate buffers are enabled by gate **G_2**. Therefore, the eight data bits from the selected address are routed through the column I/O to the data lines (I/O_1 through I/O_7), which are acting as data output lines.

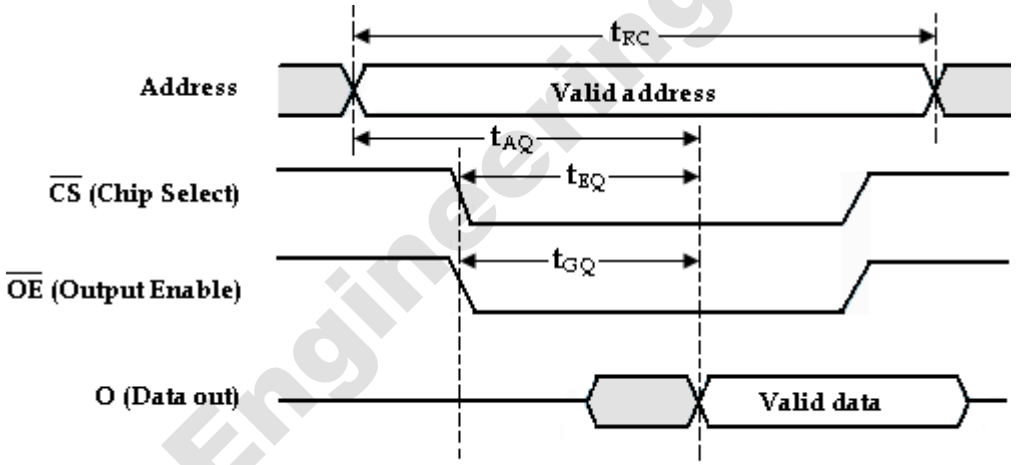
Write:

In the WRITE mode, WE' is LOW and OE' is HIGH. The input buffers are enabled by gate G_1, and the output buffers are disabled by gate G_2. Therefore the eight input data bits on the data lines are routed through the input data control and the column I/O to the selected address and stored.

Read and Write Cycles:

For the read cycle shown in part (a), a valid address code is applied to the address lines for a specified time interval called the ***read cycle time***, t_{WC}. Next, the chip select (CS) and the output enable (DE) inputs go LOW. One time interval after the DE input goes LOW; a valid data byte from the selected address appears on the data lines. This time interval is called the ***output enable access time***, t_{GQ}. Two other access times for the read cycle are the ***address access time***, t_{AQ}, measured from the beginning of a valid address to the appearance of valid data on the data lines and the **chip enable access time**, t_{EQ}, measured from the HIGH-to-LOW transition of CS to the appearance of valid data on the data lines.

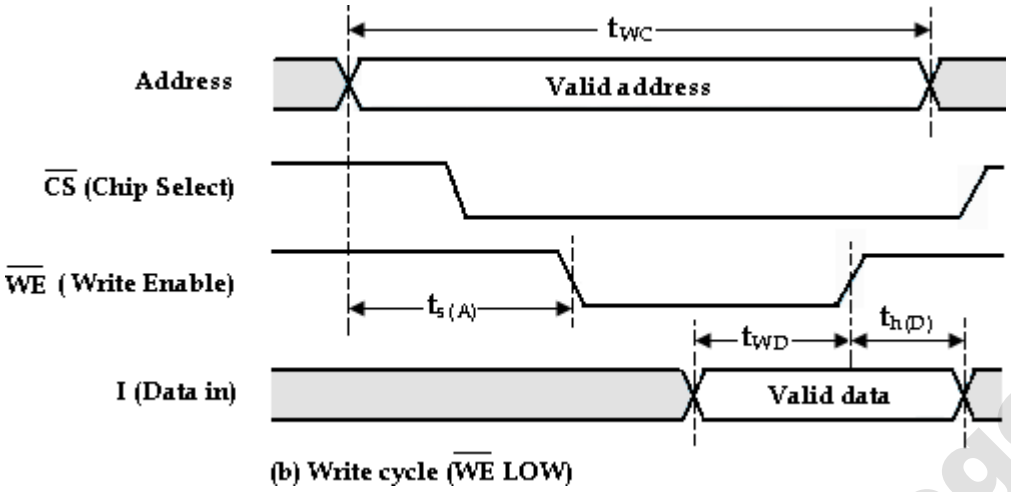
During each read cycle, one unit of data, a byte in this case is read from the memory.



(a) Read Cycle ($\overline{WE}$ HIGH)

For the write cycle shown in Figure (b), a valid address code is applied to the address lines for a specified time interval called the ***write cycle time***, t_{WE} . Next, the chip select (CS) and the write enable (WE) in puts go LOW. The required time interval from the beginning of a valid address until the WE input goes LOW is called the ***address setup time***, $t_{s(A)}$. The time that the WE input must be LOW is the write pulse width. The time that the input WE must remain LOW after valid data are applied to the data inputs is designated t_{WD}; the time that the valid input data must remain on the data lines after the WE input goes HIGH is the data hold time, $t_{h(D)}$.

During each write cycle, one unit of data is written into the memory.



(b) Write cycle ($\overline{WE}$ LOW)

5.6.2 READ- ONLY MEMORIES (ROMS)

A ROM contains permanently or semi-permanently stored data, which can be read from the memory but either cannot be changed at all or cannot be changed without specialization equipment. A ROM stores data that are used repeatedly in system applications, such as tables, conversions, or programmed instructions for system initialization and operation. ROMs retain stored data when the power is OFF and are therefore nonvolatile memories.

The ROMs are classified as follows:

i. Masked ROM (ROM)

ii. Programmed ROM (PROM)

iii. Erasable PROM (EPROM)

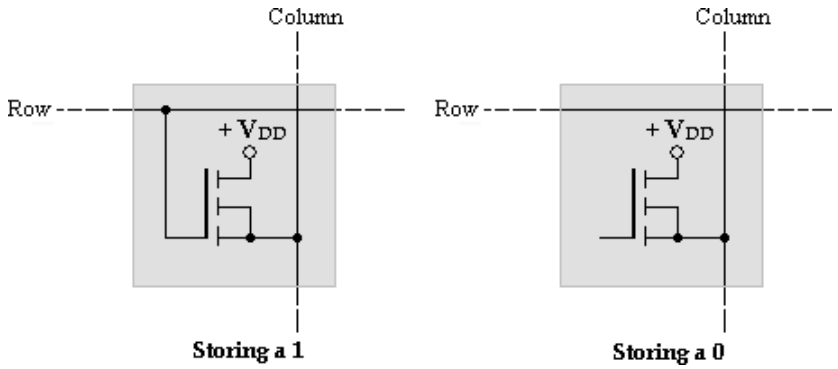
iv. Electrically Erasable PROM (EEPROM)

5.6.2.1 Masked ROM

The mask ROM is usually referred to simply as a ROM. It is permanently programmed during the manufacturing process to provide widely used standard functions, such as popular conversions, or to provide user-specified functions. Once the memory is programmed, it cannot be changed.

Most IC ROMs utilize the presence or absence of a transistor connection at a row/column junction to represent a 1 or a 0. The presence of a connection from a row line to the gate of a transistor represents a 1 at that location because when the row line is taken HIGH; all transistors with a gate connection to that row line turn on

and connect the HIGH (1) to the associated column lines.



ROM Cells

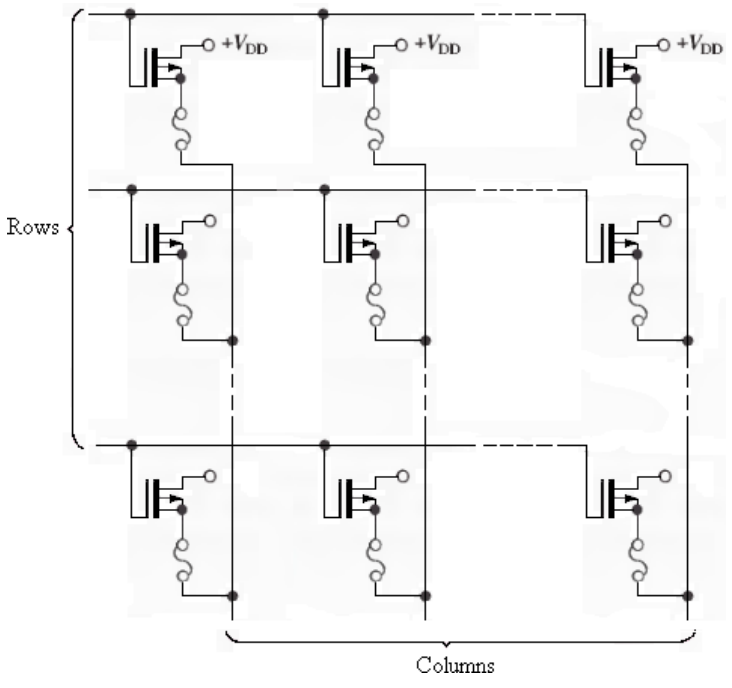
At row/column junctions where there are no gate connections, the column lines remain LOW (0) when the row is addressed.

5.6.2.2 PROM (Programmable Read-Only Memory)

The PROM (Programmable Read-only memory), comes from the manufacturer unprogrammed and are custom programmed in the field to meet the user's needs.

A PROM uses some type of fusing process to store bits, in which a memory link is burned open or left intact to represent a 0 or a 1. The fusing process is irreversible; once a PROM is programmed, it cannot be changed.

The fusible links are manufactured into the PROM between the source of each cell's transistor and its column line. In the programming process, a sufficient current is injected through the fusible link to bum it open to create a stored O. The link is left intact for a stored 1. All drains are commonly connected to V_{DD}.



PROM array with fusible links

Three basic fuse technologies used in PROMs are metal links, silicon links, and pn junctions. A brief description of each of these follows.

1. **Metal links** are made of a material such as ***nichrome***. Each bit in the memory array is represented by a separate link. During programming, the link is either "blown" open or left intact. This is done basically by first addressing a given cell and then forcing a sufficient amount of current through the link to cause it to open. When the fuse is intact, the memory cell is configured as a logic 1 and when fuse is blown (open circuit) the memory cell is logic 0.

2. **Silicon links** are formed by narrow, notched strips of ***polycrystalline silicon***. Programming of these fuses requires melting of the links by passing a sufficient amount of current through them. This amount of current causes a high temperature at the fuse location that oxidizes the silicon and forms insulation around the now-open link.

3. **Shorted junction**, or avalanche-induced migration, technology consists basically of two pn junctions arranged back-to-back. During programming, one of the diode junctions is avalanched, and the resulting voltage and heat cause aluminum ions to migrate and short the junction. The remaining junction is then used as a forward- biased diode to represent a data bit.

5.6.2.3 EPROM (Erasable Programmable ROM)

An EPROM is an erasable PROM. Unlike an ordinary PROM, an EPROM can be reprogrammed if an existing program in the memory array is erased first.

An EPROM uses an NMOSFET array with an isolated-gate structure. The isolated transistor gate has no electrical connections and can store an electrical charge for indefinite periods of time. The data bits in this type of array are represented by the presence or absence of a stored gate charge. Erasure of a data bit is a process that removes the gate charge.

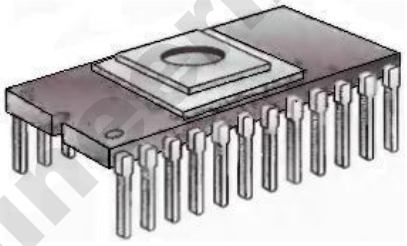
Two basic types of erasable PROMs are the ultraviolet erasable PROM (UV EPROM) and the electrically erasable PROM (EEPROM).

- **UV EPROM:**

You can recognize the UV EPROM device by the transparent quartz lid on the package, as shown in Figure below. The isolated gate in the FET of an ultraviolet EPROM is "floating" within an oxide insulating material. The programming process causes electrons to be removed from the floating gate. Erasure is done by exposure of the memory array chip to high-intensity ultraviolet radiation through the quartz window on top of the package.

The positive charge stored on the gate is neutralized after several minutes to an hour of exposure time. In EPROM's, it is not possible to erase selective information, when erased the entire information is lost. The chip can be reprogrammed.

It is ideally suited for product development, college laboratories, etc.



Ultraviolet Erasable PROM

During programming, address and datas are applied to address and data pins of the EPROM. The program pulse is applied to the program input of the EPROM. The program pulse duration is around 50msec and its amplitude depends on EPROM IC. It is typically 11.5V to 25V.

In EPROM, it is possible to program any location at any time- either individually, sequentially or at random.

5.6.2.4 EEPROM (Electrically Erasable PROM)

The EEPROM (Electrically Erasable PROM), also uses MOS circuitry. Data is stored as charge or no charge on an insulating layer, which is made very thin (< 200Å). Therefore a voltage as low as 20- 25V can be used to move charges across the thin barrier in either direction for programming or erasing ROM.

An electrically erasable PROM can be both erased and programmed with electrical pulses. Since it can be both electrically written into and electrically erased, the EEPROM can be rapidly programmed and erased in-circuit for reprogramming.

It allows selective erasing at the register level rather than erasing all the information, since the information can be changed by using electrical signals.

It has chip erase mode by which the entire chip can be erased in 10 msec. Hence EEPROM's are most expensive.

Advantages of RAM:

1. Fast operating speed **(< 150 nsec**),
2. Low power dissipation **(< 1mW**),
3. Economy,
4. Compatibility,
5. Non-destructive read-out.

Advantages of ROM:

1. Ease and speed of design,
2. Faster than MSI devices (PLD and FPGA)
3. The program that generates the ROM contents can easily be structured to handle unusual or undefined cases,
4. A ROM's function is easily modified just by changing the stored pattern, usually without changing any external connections,
5. More economical.

Disadvantages of ROM:

1. For functions more than 20 inputs, a ROM based circuit is impractical because of the limit on ROM sizes that are available.
2. For simple to moderately complex functions, ROM based circuit may be costly: consume more power; run slower.

Comparison between RAM and ROM:

S.No	RAM	ROM
1	RAMs have both read and write capability.	ROMs have only read operation.
2	RAMs are volatile memories.	ROMs are non-volatile memories.
3	They lose stored data when the power is turned OFF.	They retain stored data even if power is turned off.
4	RAMs are available in both bipolar and MOS technologies.	RAMs are available in both bipolar and MOS technologies.
5	**Types**: SRAM, DRAM, EEPROM	**Types**: PROM, EPROM.

Comparison between SRAM and DRAM:

S.No	Static RAM	Dynamic RAM
1	It contains less memory cells per unit area.	It contains more memory cells per unit area.
2	Its access time is less, hence faster memories.	Its access time is greater than static RAM
3	It consists of number of flip-flops. Each flip-flop stores one bit.	It stores the data as a charge on the capacitor. It consists of MOSFET and capacitor for each cell.
4	Refreshing circuitry is not required.	Refreshing circuitry is required to maintain the charge on the capacitors every time after every few milliseconds. Extra hardware is required to control refreshing.
5	Cost is more	Cost is less.

Comparison of Types of Memories:

Memory type	Non- Volatile	High Density	One- Transistor cell	In-system writability
SRAM	No	No	No	Yes
DRAM	No	Yes	Yes	Yes
ROM	Yes	Yes	Yes	No
EPROM	Yes	Yes	Yes	No
EEPROM	Yes	No	No	Yes

5.8 PROGRAMMABLE LOGIC DEVICES:

5.8.1 INTRODUCTION:

A combinational PLD is an integrated circuit with programmable gates divided into an AND array and an OR array to provide an AND-OR sum of product implementation. The PLD's can be reprogrammed in few seconds and hence gives more flexibility to experiment with designs. Reprogramming feature of PLDs also makes it possible to accept changes/modifications in the previously design circuits.

The advantages of using programmable logic devices are:

1. Reduced space requirements.
2. Reduced power requirements.
3. Design security.
4. Compact circuitry.
5. Short design cycle.
6. Low development cost.
7. Higher switching speed.
8. Low production cost for large-quantity production.

According to architecture, complexity and flexibility in programming in PLD's are classified as—

- PROMs : Programmable Read Only memories,
- PLAs : Programmable Logic Arrays,
- PAL : Programmable Logic Array,
- FPGA : Field Programmable Gate Arrays,
- CPLDs : Complex Programmable Logic Devices.

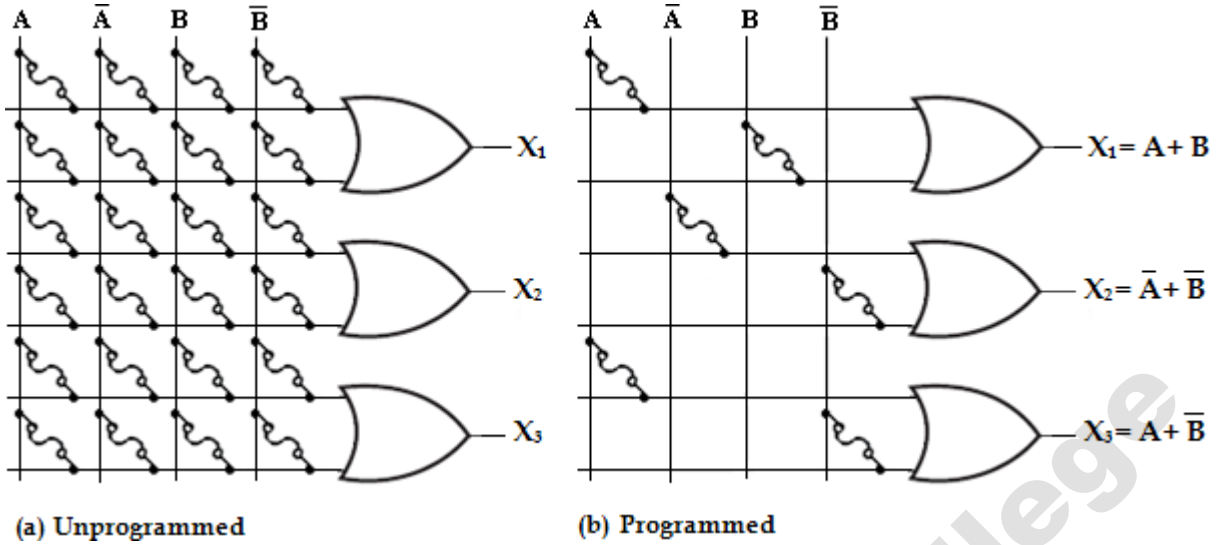
Programmable Arrays:

All PLDs consists of programmable arrays. A programmable array is essentially a grid of conductors that form rows and columns with a fusible link at each cross point. Arrays can be either fixed or programmable.

The OR Array:

It consists of an array of OR gates connected to a programmable matrix with fusible links at each cross point of a row and column, as shown in the figure below. The array can be programmed by blowing fuses to eliminate selected variables from the output functions. For each input to an OR gate, only one fuse is left intact in order to connect the desired variable to the gate input. Once the fuse is blown, it cannot be reconnected.

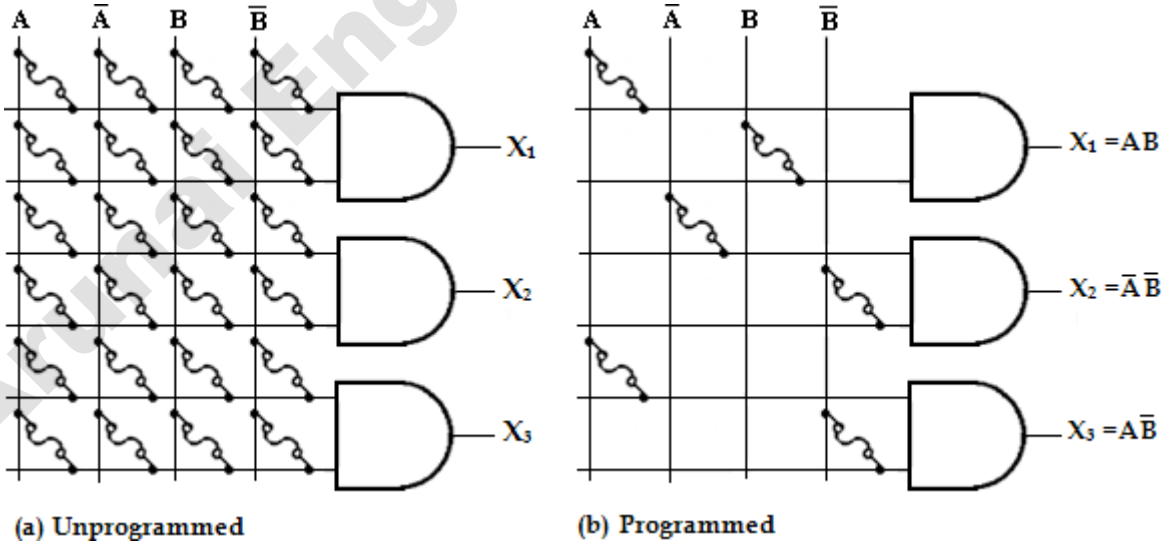
Another method of programming a PLD is the antifuse, which is the opposite of the fuse. Instead of a fusible link being broken or opened to program a variable, a normally open contact is shorted by ―melting‖ the antifuse material to form a connection.



An example of a basic programmable OR array

The AND Array:

This type of array consists of AND gates connected to a programmable matrix with fusible links at each cross points, as shown in the figure below. Like the OR array, the AND array can be programmed by blowing fuses to eliminate selected variables from the output functions. For each input to an AND gate, only one fuse is left intact in order to connect the desired variable to the gate input. Also, like the OR array, the AND array with fusible links or with antifuses is one-time programmable.



An example of a basic programmable AND array

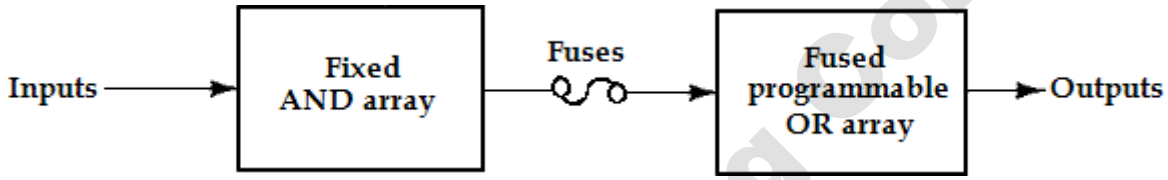
5.8.2 Classification of PLDs

There are three major types of combinational PLDs and they differ in the placement of the programmable connections in the AND-OR array. The configuration of the three PLDs is shown below.

1. Programmable Read-Only Memory (PROM):

A PROM consists of a set of fixed (non-programmable) AND array constructed

as a decoder and a programmable OR array. The programmable OR gates implement the Boolean functions in sum of minterms.



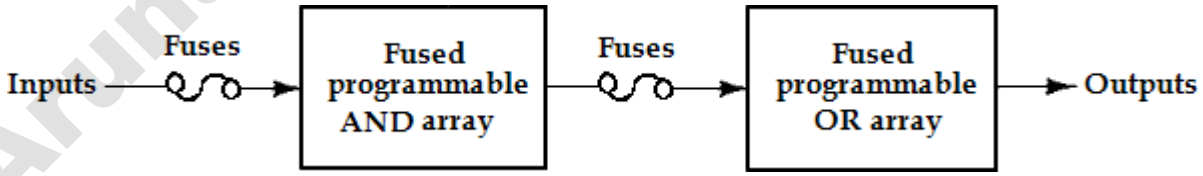
(a) Programmable read- only memory (PROM)

2. Programmable Logic Array (PLA):

A PLA consists of a programmable AND array and a programmable OR array.

The product terms in the AND array may be shared by any OR gate to provide the required sum of product implementation.

The PLA is developed to overcome some of the limitations of the PROM. The PLA is also called an FPLA (Field Programmable Logic Array) because the user in the field, not the manufacturer, programs it.

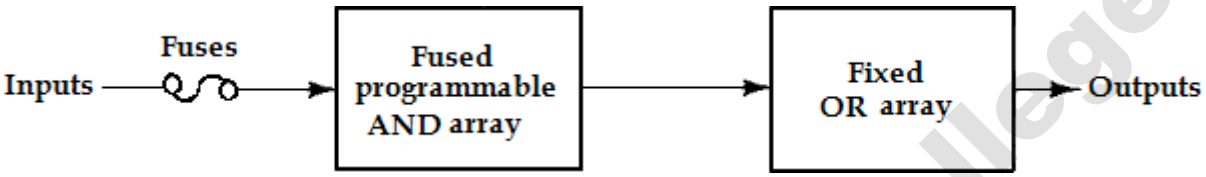


Programmable Logic Array (PLA)

3. Programmable Array Logic (PAL):

The basic PAL consists of a programmable AND array and a fixed OR array. The AND gates are programmed to provide the product terms for the Boolean functions, which are logically summed in each OR gate.

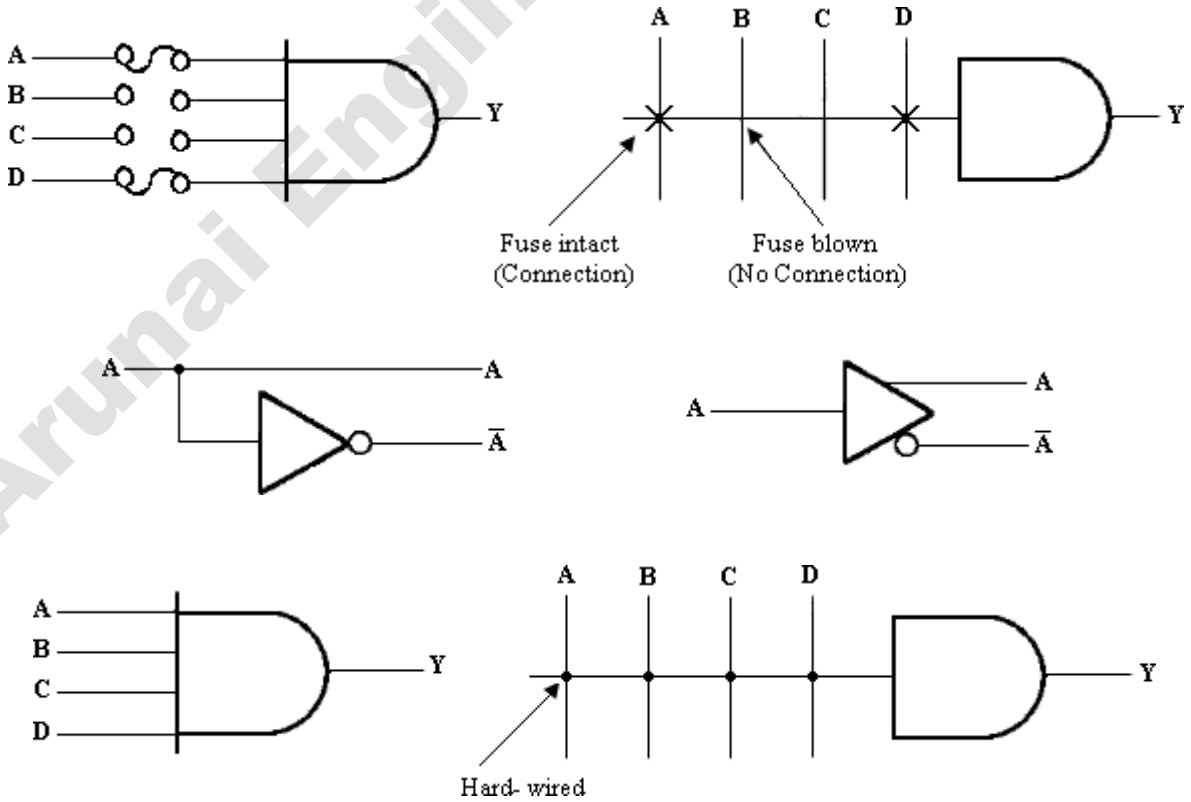
It is developed to overcome certain disadvantages of the PLA, such as longer delays due to the additional fusible links that result from using two programmable arrays and more circuit complexity.



Programmable Array Logic (PAL)

Array logic Symbols:

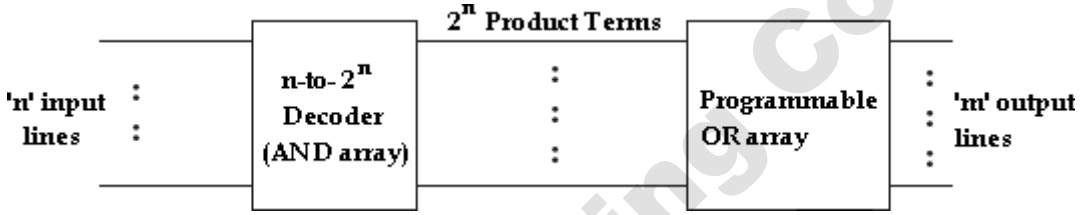
PLDs have hundreds of gates interconnected through hundreds of electronic fuses. It is sometimes convenient to draw the internal logic of such device in a compact form referred to as **array logic**.



5.8.3 PROGRAMMABLE ROM:

PROMs are used for code conversions, generating bit patterns for characters and as look-up tables for arithmetic functions.

As a PLD, PROM consists of a fixed AND-array and a programmable OR array. The AND array is an n-to-2^n decoder and the OR array is simply a collection of programmable OR gates. The OR array is also called the memory array. The decoder serves as a minterm generator. The n-variable minterms appear on the 2^n lines at the decoder output. The 2^n outputs are connected to each of the _m' gates in the OR array via programmable fusible links.



2^n x m PROM

5.8.4 Implementation of Combinational Logic Circuit using PROM

1. Using PROM realize the following expression

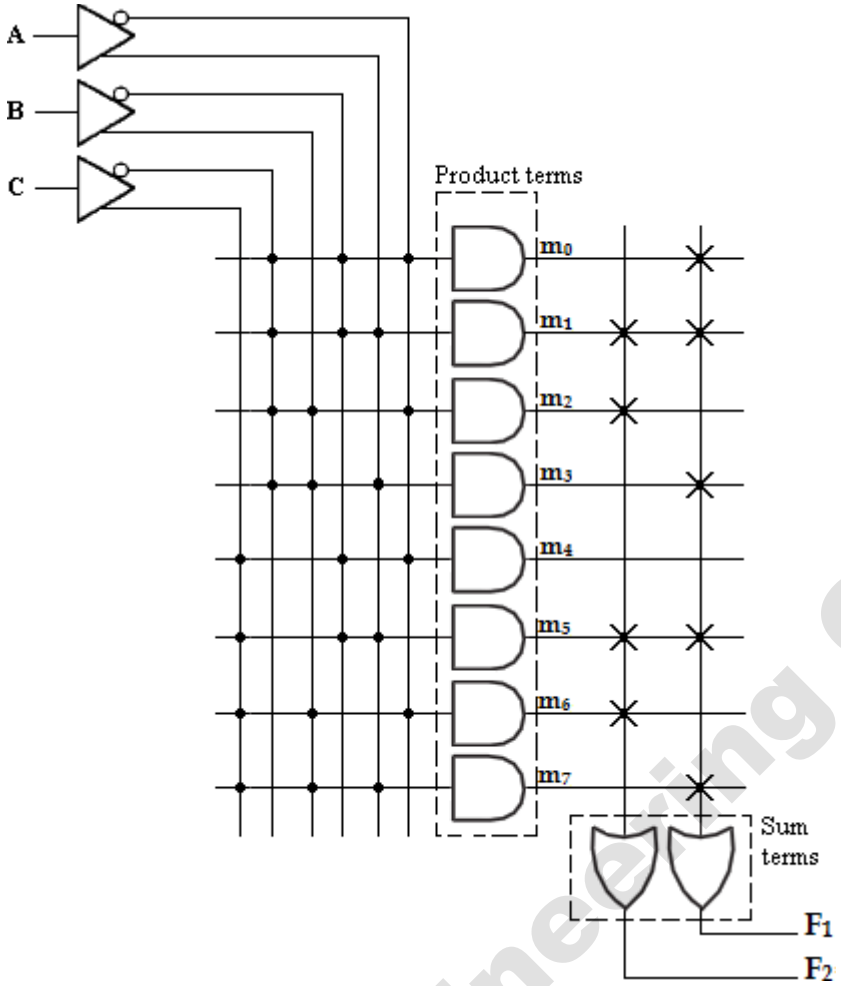
   F_1 (A, B, C) = ∑m (0, 1, 3, 5, 7)

   F_2 (A, B, C) = ∑m (1, 2, 5, 6)

Step1: Truth table for the given function

A	B	C	F_1	F_2
0	0	0	1	0
0	0	1	1	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	0

Step 2: PROM diagram

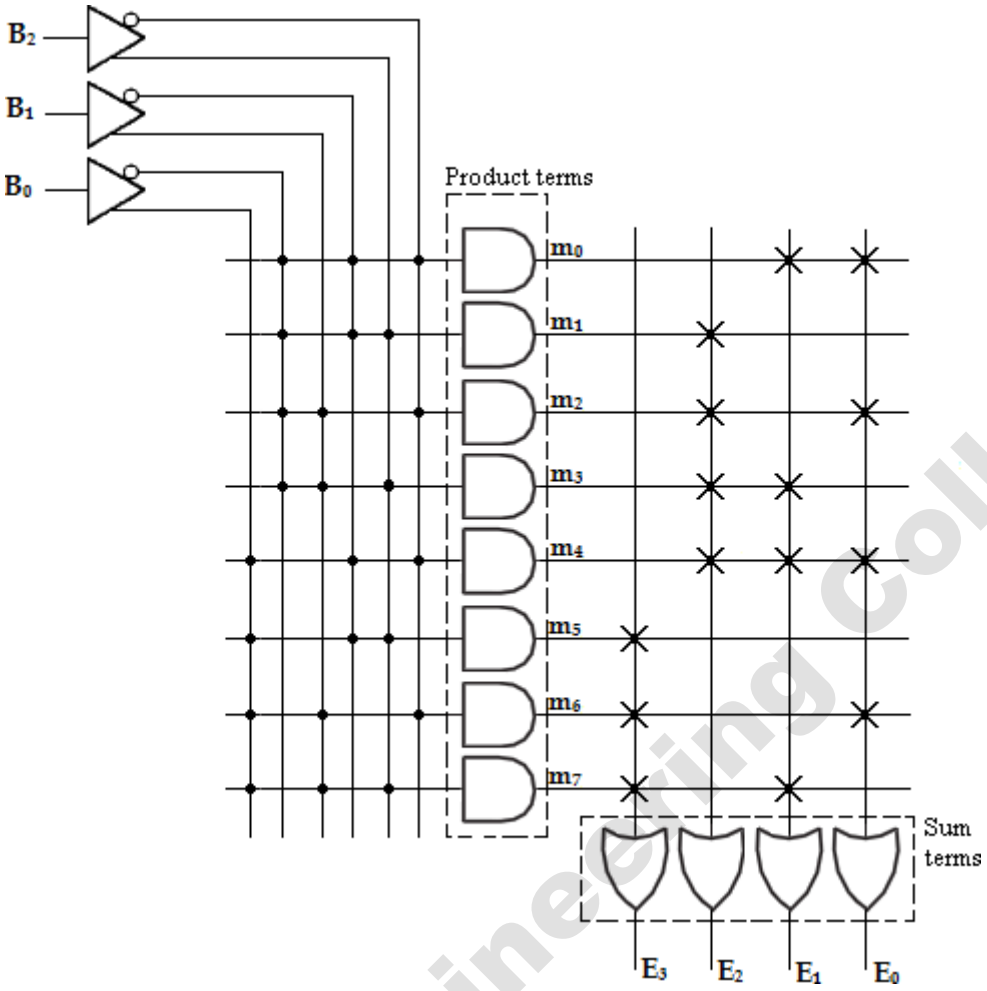


2. Design a combinational circuit using PROM. The circuit accepts 3-bit binary and generates its equivalent Excess-3 code.

Step1: Truth table for the given function

B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	1	1
0	0	1	0	1	0	0
0	1	0	0	1	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	0	0
1	1	0	1	0	0	1
1	1	1	1	0	1	0

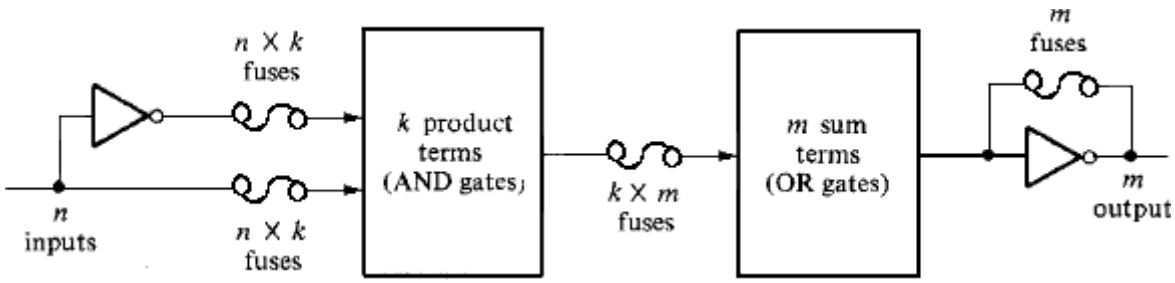
Step 2: PROM diagram



5.8.5 PROGRAMMABLE LOGIC ARRAY: (PLA)

The PLA is similar to the PROM in concept except that the PLA does not provide full coding of the variables and does not generate all the minterms.

The decoder is replaced by an array of AND gates that can be programmed to generate any product term of the input variables. The product term are then connected to OR gates to provide the sum of products for the required Boolean functions. The AND gates and OR gates inside the PLA are initially fabricated with fuses among them. The specific boolean functions are implemented in sum of products form by blowing the appropriate fuses and leaving the desired connections.



PLA block diagram

The block diagram of the PLA is shown above. It consists of _n' inputs, _m' outputs, _k' product terms and _m' sum terms. The product terms constitute a group of _k' AND gates and the sum terms constitute a group of _m' OR gates. Fuses are inserted between all _n' inputs and their complement values to each of the AND gates. Fuses are also provided between the outputs of the AND gate and the inputs of the OR gates.

Another set of fuses in the output inverters allow the output function to be generated either in the AND-OR form or in the AND-OR-INVERT form. With the inverter fuse in place, the inverter is bypassed, giving an AND-OR implementation. With the fuse blown, the inverter becomes part of the circuit and the function is implemented in the AND-OR-INVERT form.

5.8.6 Implementation of Combinational Logic Circuit using PLA

1. **Implement the combinational circuit with a PLA having 3 inputs, 4 product terms and 2 outputs for the functions.**

   F_1 (A, B, C) = ∑m (0, 1, 2, 4)

   F_2 (A, B, C) = ∑m (0, 5, 6, 7)

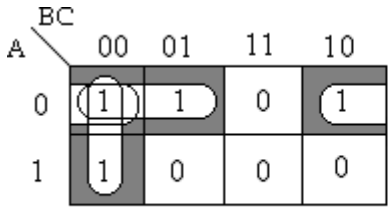
Solution:

Step 1: Truth table for the given functions

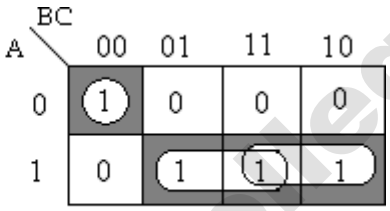
A	B	C	F_1	F_2
0	0	0	1	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0

1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

Step 2: K-map Simplification

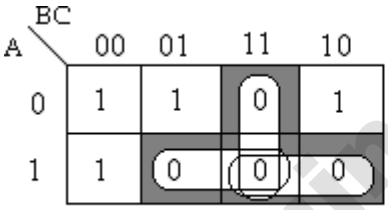


$F_1 = A'B' + A'C' + B'C'$

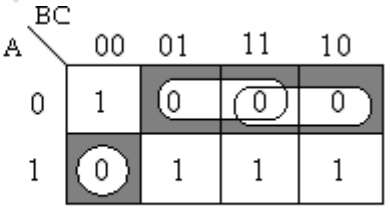


$F_2 = AC + AB + A'B'C'$

With this simplification, total number of product term is 6. But we require only 4 product terms. Therefore find out F_1' and F_2'.



$F_1' = AC + BC + AB$



$F_2' = A'C + A'B + A'B'C'$

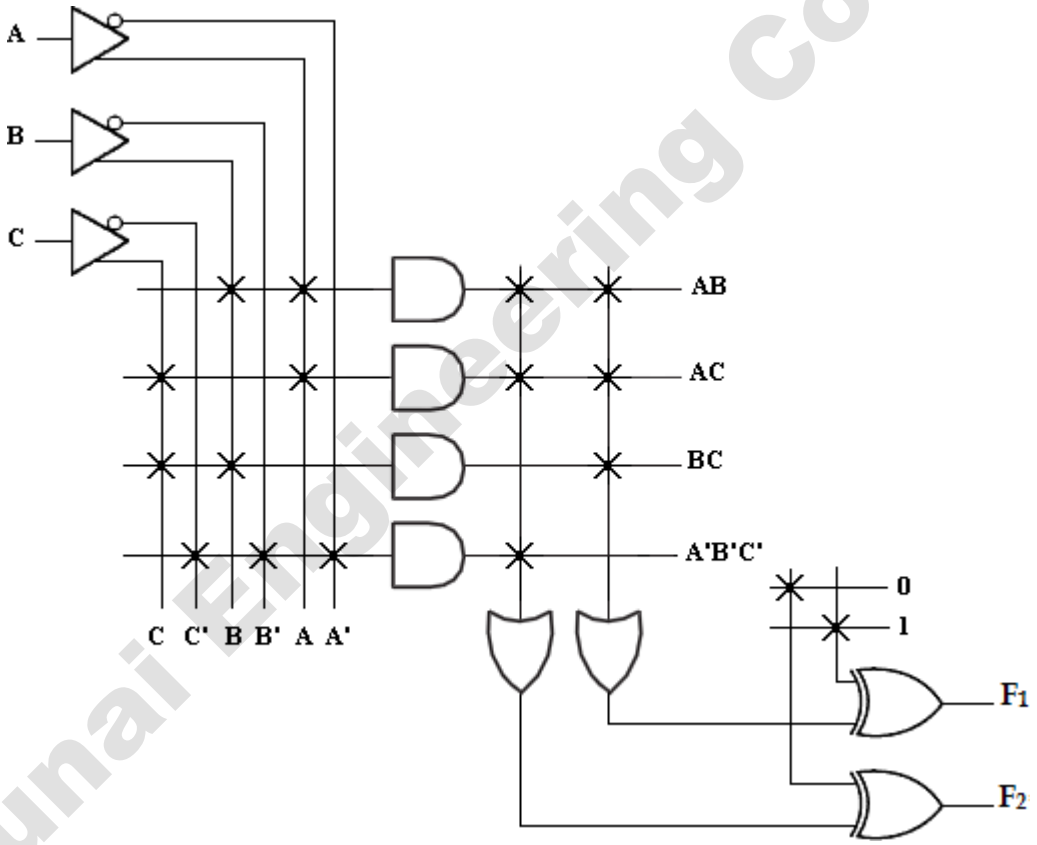
Now select, F_1' and F_2, the product terms are AC, AB, BC and A'B'C'

Step 3: PLA Program table:

	Product term	Inputs			Outputs	
		A	B	C	F_1 (C)	F_2 (T)
AB	1	1	1	-	1	1
AC	2	1	-	1	1	1
BC	3	-	1	1	1	-
A'B'C'	4	0	0	0	-	1

In the PLA program table, first column lists the product terms numerically as 1, 2, 3, and 5. The second column (Inputs) specifies the required paths between the AND gates and the inputs. For each product term, the inputs are marked with 1, 0, or - (dash). If a variable in the product form appears in its normal form, the corresponding input variable is marked with a 1. If it appears complemented, the corresponding input variable is marked with a 0. If the variable is absent in the product term, it is marked with a dash (-). The third column (output) specifies the path between the AND gates and the OR gates. The output variables are marked with 1's for all those product terms that formulate the required function.

Step 4: PLA Diagram



The PLA diagram uses the array logic symbols for complex symbols. Each input and its complement is connected to the inputs of each AND gate as indicated by the intersections between the vertical and horizontal lines. The output of the AND gate are connected to the inputs of each OR gate. The output of the OR gate goes to an EX-OR gate where the other input can be programmed to receive a signal equal to either logic 1 or 0.

The output is inverted when the EX-OR input is connected to 1 ie., **(x ⊕1= x')**. The output does not change when the EX-OR input is connected to 0 ie., **(x ⊕0= x)**.

2. Implement the combinational circuit with a PLA having 3 inputs, 4 product terms and 2 outputs for the functions.

F_1 (A, B, C) = ∑m (3, 5, 6, 7)

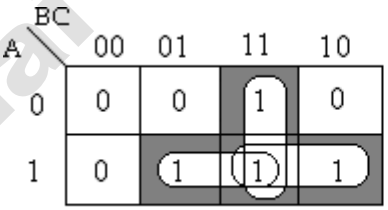
F_2 (A, B, C) = ∑m (0, 2, 4, 7)

Solution:

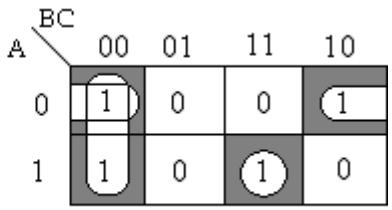
Step 1: Truth table for the given functions

A	B	C	F_1	F_2
0	0	0	0	1
0	0	1	0	0
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

Step 2: K-map Simplification

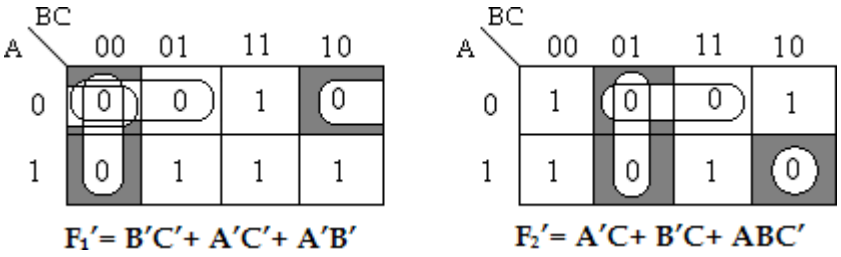


F_1= AC+ AB+ BC



F_2= B'C'+ A'C'+ ABC

With this simplification, total number of product term is 6. But we require only 4 product terms. Therefore find out F_1' and F_2'.

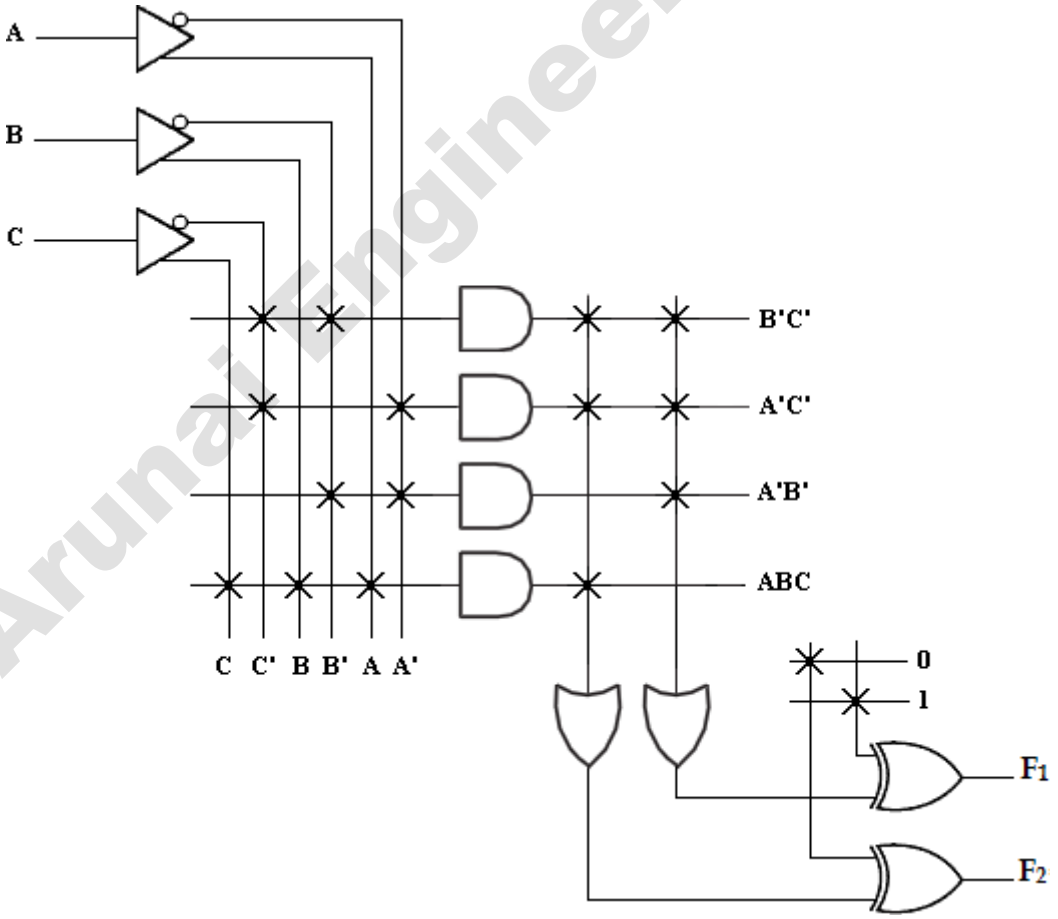


Now select, F_1' and F_2, the product terms are **B′C′, A′C′, A′B′ and ABC**.

Step 3: PLA Program table

	Product term	Inputs			Outputs	
		A	B	C	F_1 (C)	F_2 (T)
B′C′	1	-	0	0	1	1
A′C′	2	0	-	0	1	1
A′B′	3	0	0	-	1	-
ABC	4	1	1	1	-	1

Step 4: PLA Diagram



3. Implement the following functions using PLA.

F_1 (A, B, C) = ∑m (1, 2, 4, 6)

F_2 (A, B, C) = ∑m (0, 1, 6, 7)

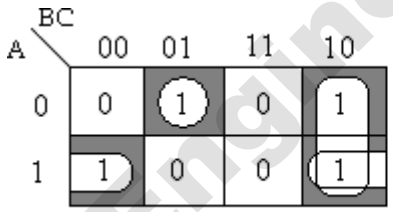
F_3 (A, B, C) = ∑m (2, 6)

Solution:

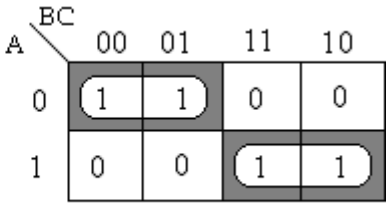
Step 1: Truth table for the given functions

A	B	C	F_1	F_2	F_3
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	1	0

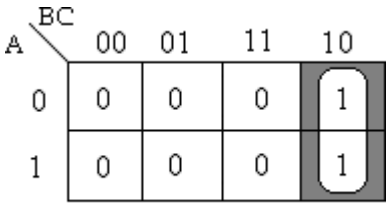
Step 2: K-map Simplification



F_1= A′B′C+ AC′+ BC′



F_2= A′B′+ AB

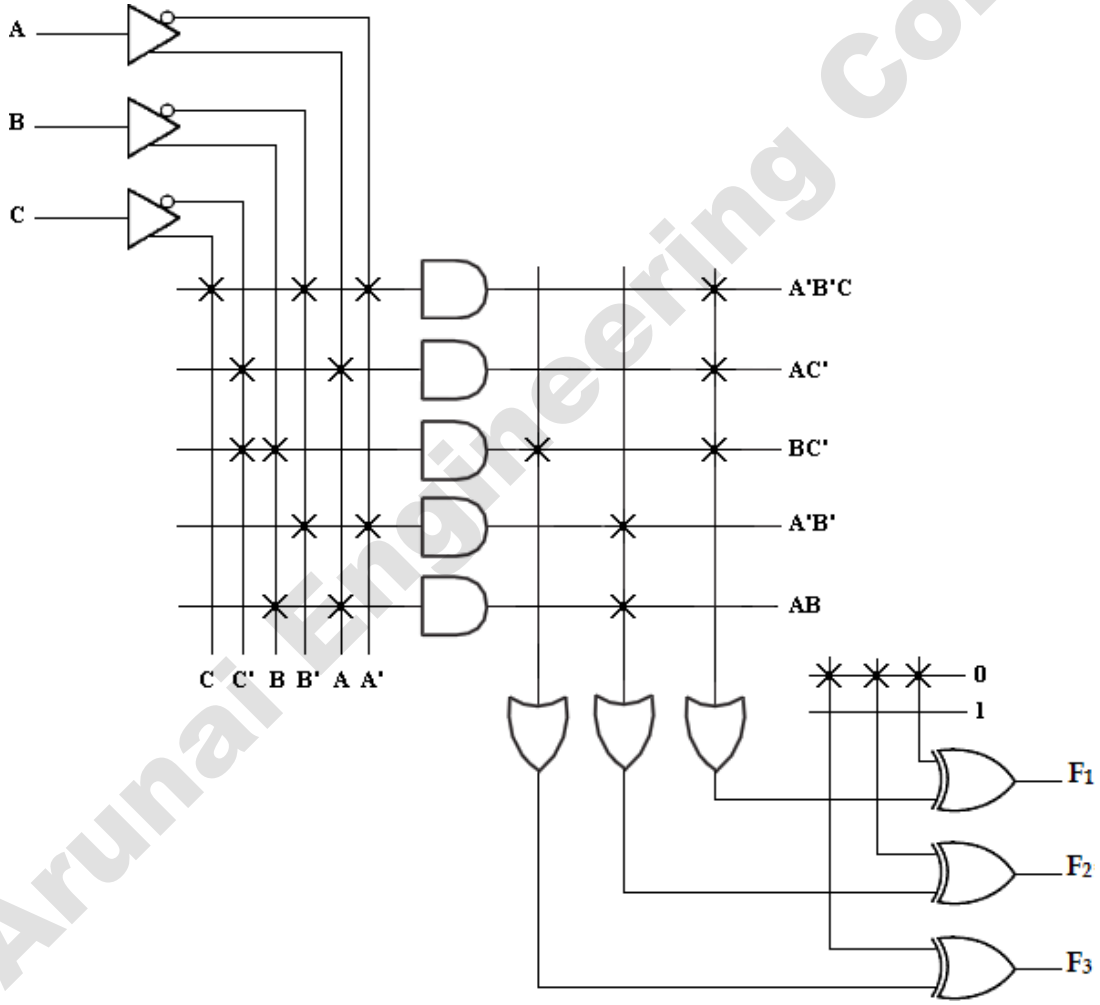


F_3= BC′

Step 3: PLA Program table

	Product term	Inputs			Outputs		
		A	B	C	F_1 (T)	F_2 (T)	F_3 (T)
A′B′C	1	0	0	1	1	-	-
AC′	2	1	-	0	1	-	-
BC′	3	-	1	0	1	-	1
A′B′	4	0	0	-	-	1	-
AB	5	1	1	-	-	1	-

Step 4: PLA Diagram



4. **A combinational circuit is designed by the function**

   F_1 (A, B, C) = ∑m (3, 5, 7)

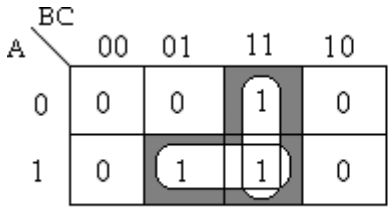
   F_2 (A, B, C) = ∑m (4, 5, 7)

Solution:

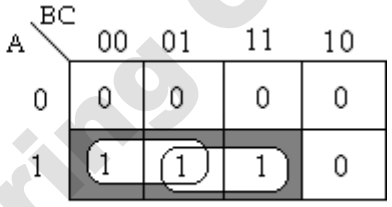
Step 1: Truth table for the given functions

A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

Step 2: K-map Simplification



$F_1 = AC + BC$

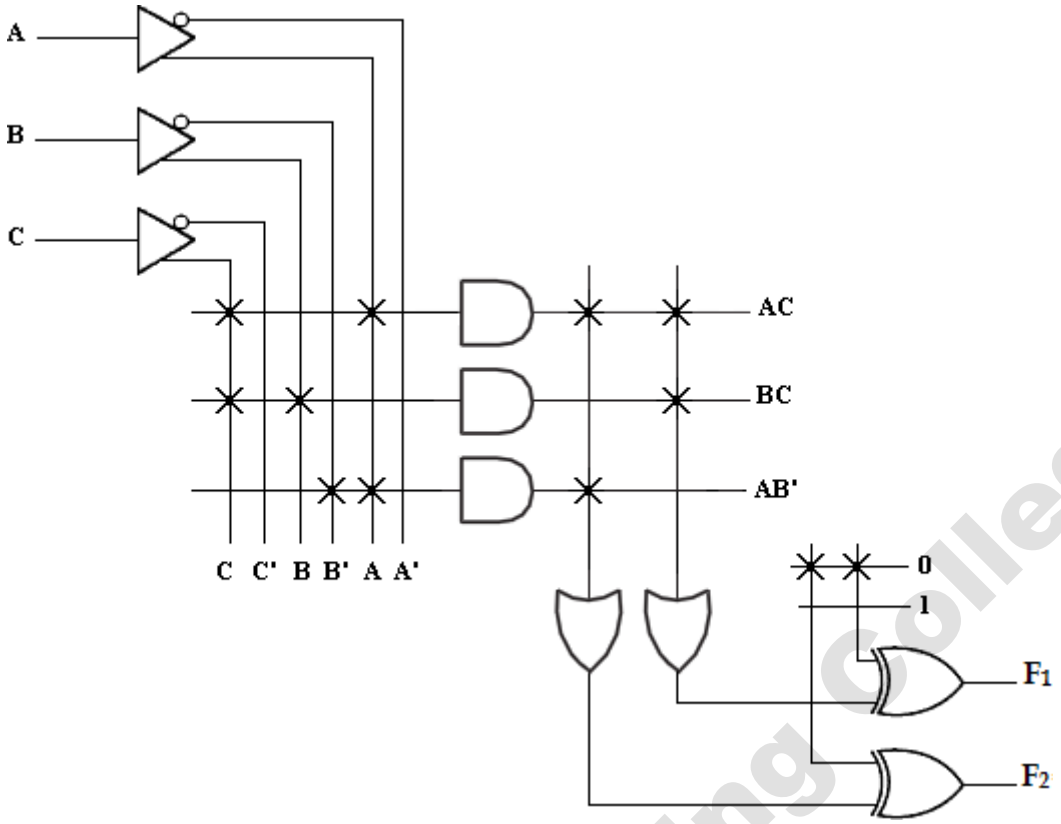


$F_2 = AB' + AC$

Step 3: PLA Program table

	Product term	Inputs			Outputs	
		A	B	C	F_1 (C)	F_2 (T)
AC	1	1	-	1	1	1
BC	2	-	1	1	1	-
AB‘	3	1	0	-	-	1

Step 4: PLA Diagram



5. A combinational circuit is defined by the functions,

$F_1(A, B, C) = \sum m(1, 3, 5)$

$F_2(A, B, C) = \sum m(5, 6, 7)$

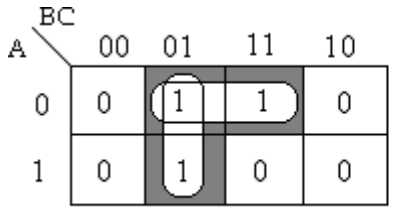
Implement the circuit with a PLA having 3 inputs, 3 product terms and 2 outputs.

Solution:

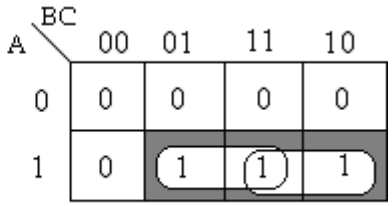
Step 1: Truth table for the given functions

A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	0	1
1	1	1	0	1

Step 2: K-map Simplification

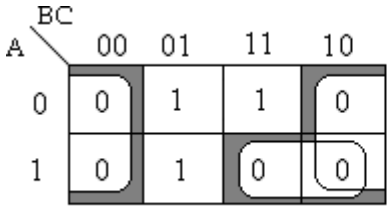


$F_1 = B'C + A'C$

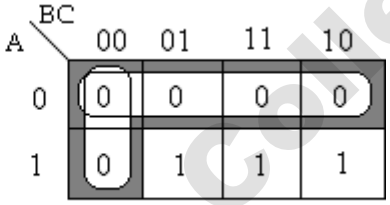


$F_2 = AC + AB$

With this simplification, total number of product term is 5. But we require only 3 product terms. Therefore find out F_1' and F_2'.



$F_1' = C' + AB$



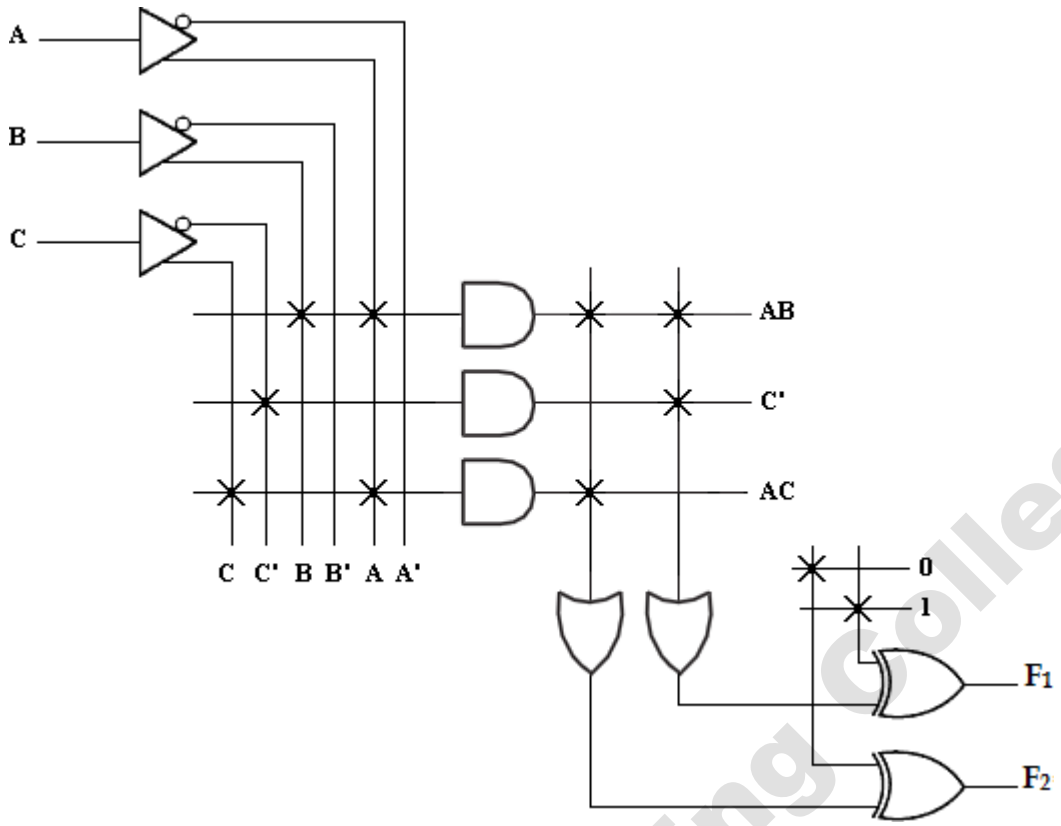
$F_2' = A' + B'C'$

Now select, F_1' and F_2, the product terms are **AC, AB and C'**.

Step 3: PLA Program table

	Product term	Inputs			Outputs	
		A	B	C	F_1 (C)	F_2 (T)
AB	1	1	1	-	1	1
C'	2	-	-	0	1	-
AC	3	1	-	1	-	1

Step 4: PLA Diagram



6. A combinational circuit is defined by the functions,

$$F_1(A, B, C) = \sum m(0, 1, 3, 4)$$

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

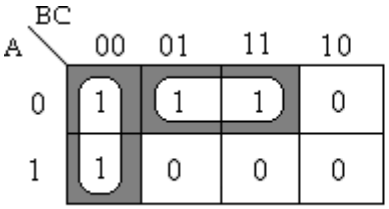
Implement the circuit with a PLA having 3 inputs, 4 product terms and 2 outputs.

Solution:

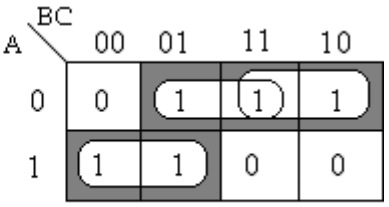
Step 1: Truth table for the given functions

A	B	C	F_1	F_2
0	0	0	1	0
0	0	1	1	1
0	1	0	0	1
0	1	1	1	1
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	0	0

Step 2: K-map Simplification



$F_1 = B'C' + A'C$



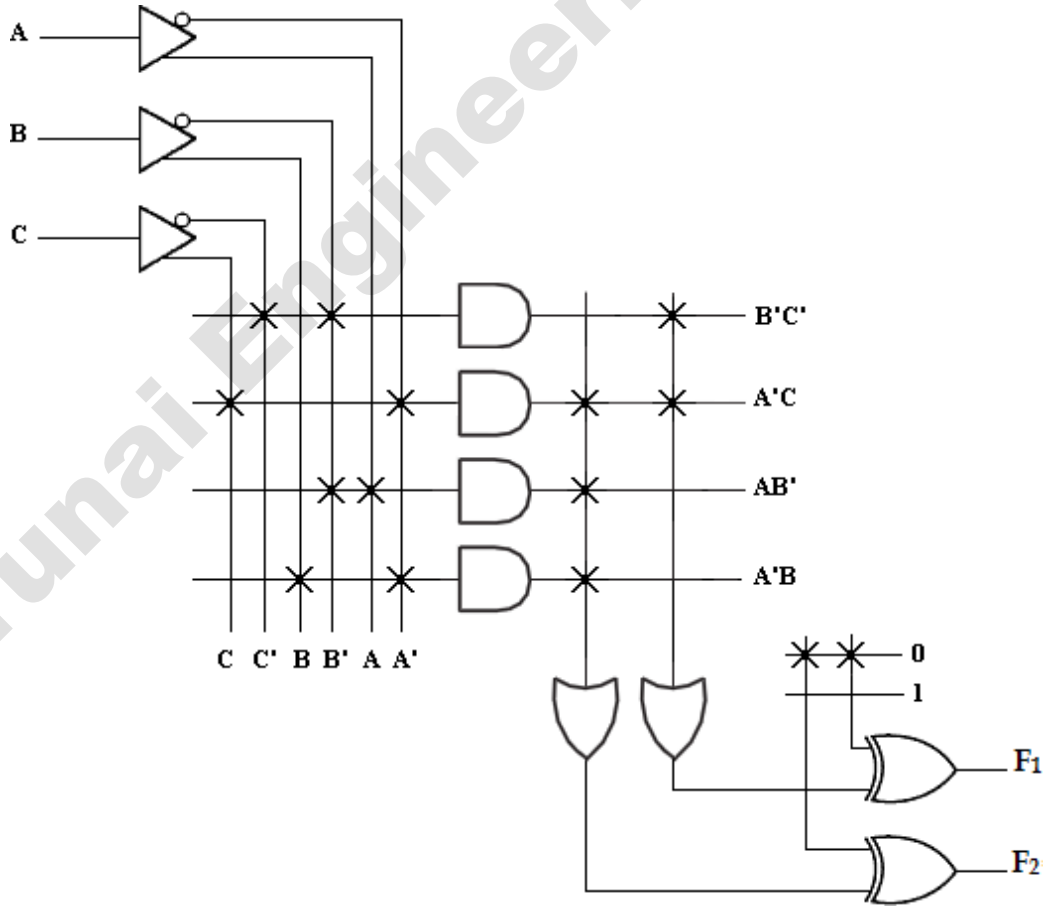
$F_2 = AB' + A'C + A'B$

The product terms are **B'C', A'C, AB' and A'B**.

Step 3: PLA Program table

	Product term	Inputs			Outputs	
		A	B	C	F_1 (T)	F_2 (T)
B'C'	1	-	0	0	1	-
A'C	2	0	-	1	1	1
AB'	3	1	0	-	-	1
A'B	4	0	1	-	-	1

Step 4: PLA Diagram



7. A combinational logic circuit is defined by the function,

F (A, B, C, D) = ∑m (3, 4, 5, 7, 10, 14, 15)

G (A, B, C, D) = ∑m (1, 5, 7, 11, 15)

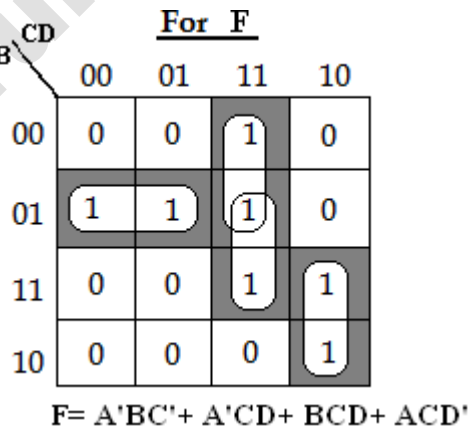
Implement the circuit with a PLA having 4 inputs, 6 product terms and 2 outputs.

Solution:

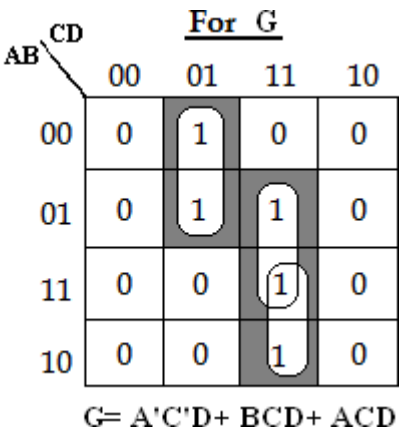
Step 1: Truth table for the given functions

A	B	C	D	F	G
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	0
0	0	1	1	1	0
0	1	0	0	1	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	1	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	1	0
1	1	1	1	1	1

Step 2: K-map Simplification



F= A'BC'+ A'CD+ BCD+ ACD'



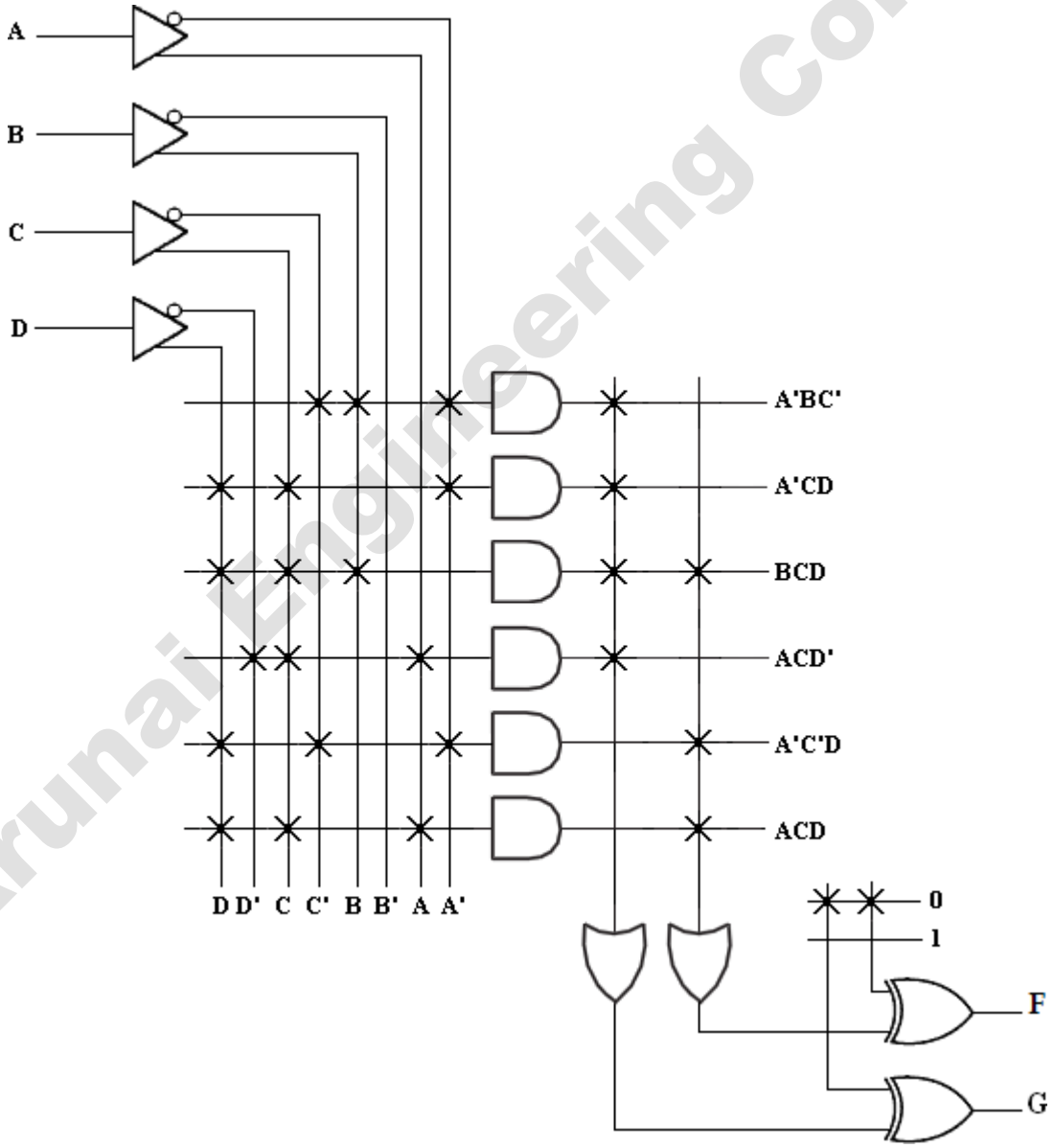
G= A'C'D+ BCD+ ACD

The product terms are A‘BC‘, A‘CD, BCD, ACD‘, A‘C‘D, ACD

Step 3: PLA Program table

	Product term	Inputs				Outputs	
		A	B	C	D	F (T)	G (T)
A‘BC‘	1	0	1	0	-	1	-
A‘CD	2	0	-	1	1	1	-
BCD	3	-	1	1	1	1	1
ACD‘	4	1	-	1	0	1	-
A‘C‘D	5	0	-	0	1	-	1
ACD	6	1	-	1	1	-	1

Step 4: PLA Diagram



8. Design a BCD to Excess-3 code converter and implement using suitable PLA.

Solution:

Step 1: Truth table of BCD to Excess-3 converter is shown below,

Decimal	BCD code				Excess-3 code			
	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Step 2: K-map Simplification

For E_3

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	x	x	x	x
10	1	1	x	x

$E_3 = B_3 + B_2B_0 + B_2B_1$

For E_2

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	x	x	x	x
10	0	1	x	x

$E_2 = B_2B_1'B_0' + B_2'B_0 + B_2'B_1$

For E_1

B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	x	x	x	x
10	1	0	x	x

$E_1 = B_1'B_0' + B_1B_0$

For E_0

B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	x	x	x	x
10	1	0	x	x

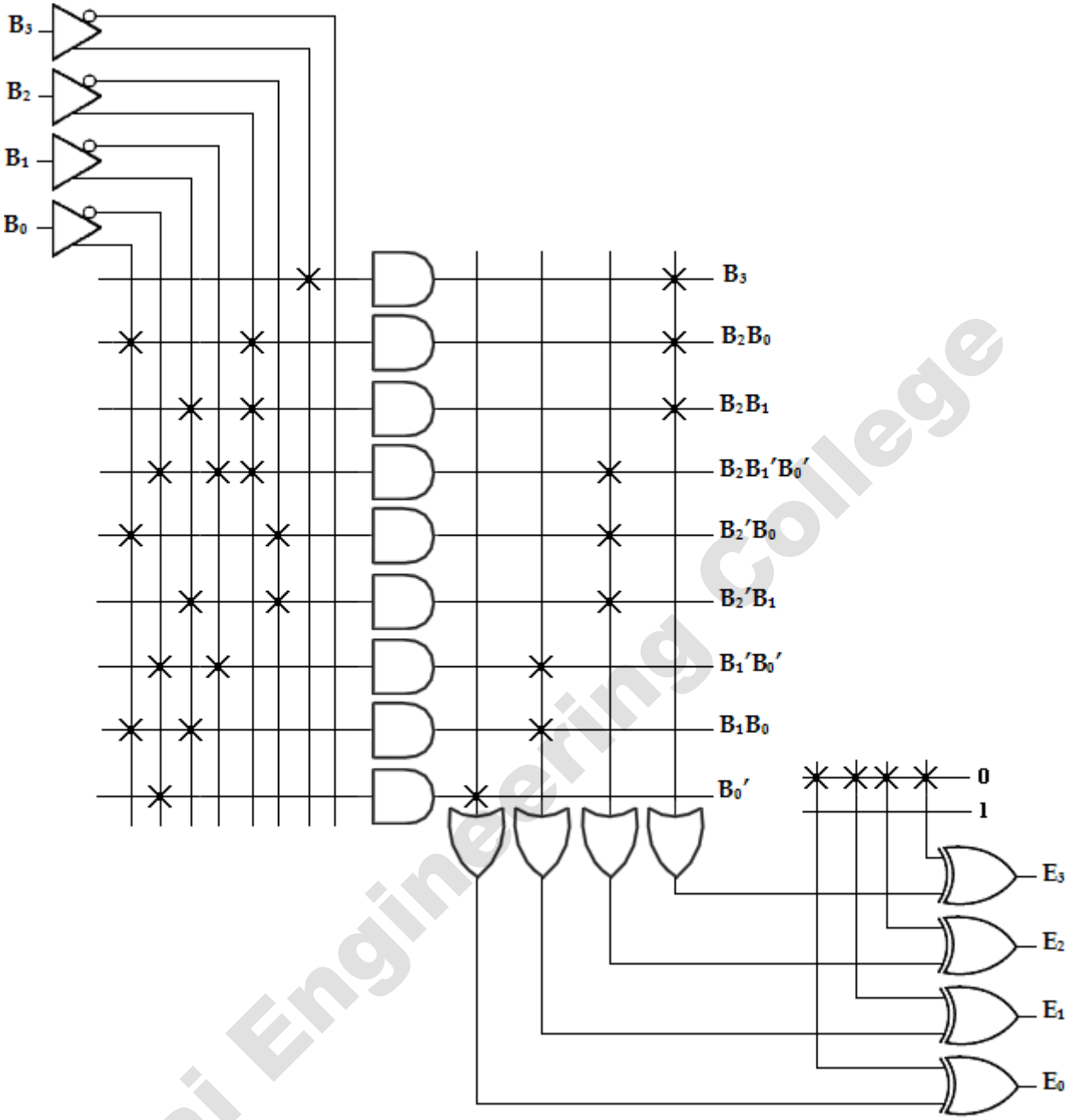
$E_0 = B_0'$

The product terms are B_3, B_2B_0, B_2B_1, $B_2B_1'B_0'$, $B_2'B_0$, $B_2'B_1$, $B_1'B_0'$, B_1B_0, B_0'

Step 3: PLA Program table

	Product terms	Inputs				Outputs			
		B_3	B_2	B_1	B_0	E_3 (T)	E_2 (T)	E_1 (T)	E_0 (T)
B_3	1	1	-	-	-	1	-	-	-
B_2B_0	2	-	1	-	1	1	-	-	-
B_2B_1	3	-	1	1	-	1	-	-	-
$B_2B_1'B_0'$	4	-	1	0	0	-	1	-	-
$B_2'B_0$	5	-	0	-	1	-	1	-	-
$B_2'B_1$	6	-	0	1	-	-	1	-	-
$B_1'B_0'$	7	-	-	0	0	-	-	1	-
B_1B_0	8	-	-	1	1	-	-	1	-
B_0'	9	-	-	-	0	-	-	-	1

Step 4: PLA Diagram



Comparison between PROM, PLA, and PAL:

S.No	PROM	PLA	PAL
1	AND array is fixed and OR array is programmable	Both AND and OR arrays are programmable	OR array is fixed and AND array is programmable
2	Cheaper and simpler to use	Costliest and complex	Cheaper and simpler

3	All minterms are decoded	AND array can be programmed to get desired minterms	AND array can be programmed to get desired minterms
4	Only Boolean functions in standard SOP form can be implemented using PROM	Any Boolean functions in SOP form can be implemented using PLA	Any Boolean functions in SOP form can be implemented using PLA