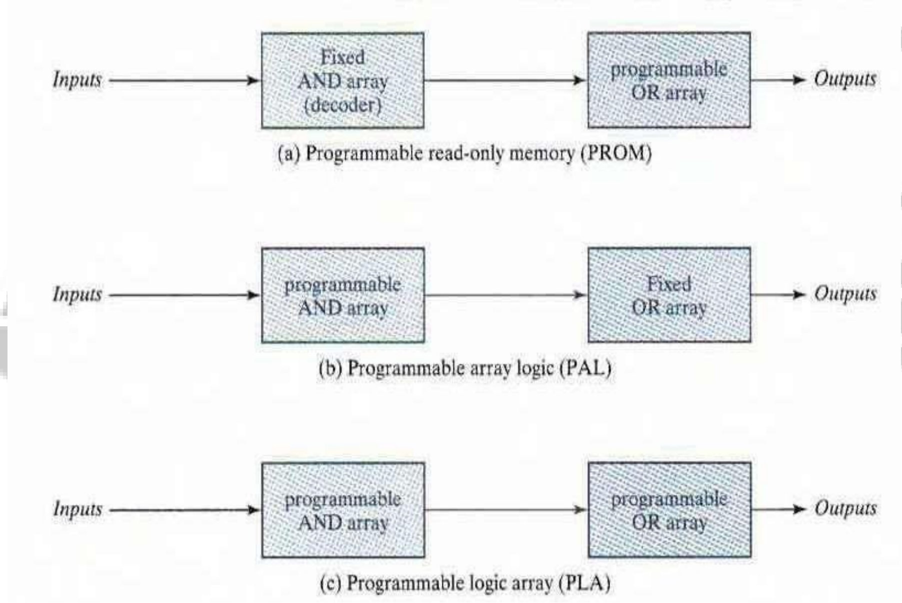
# EC 3352 – DIGITAL SYSTEM DESIGN

## UNIT – V : LOGIC FAMILIES AND PROGRAMMABLE LOGIC DEVICES

### 5.3 PROGRAMMABLE LOGIC DEVICES COMBINATIONAL PLDs

The PROM is a combinational programmable logic device (PLD) - an integrated circuit with programmable gates divided into an AND array and an OR array to provide an AND-OR sum of- product implementation. There are three major types of combinational PLDs differing in the placement of the programmable connections in the AND-OR array. The PROM has a fixed AND array constructed as a decoder and a programmable OR array. The programmable OR gates implement the Boolean functions in sum-of-minterms form. The PAL has a programmable ASD 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. The most flexible PLD is the PLA, in which both the AND and OR arrays can be programmed. The product terms in the AND array may be shared by any OR gate to provide the required sum-of-products implementation.



(a) Programmable read-only memory (PROM)

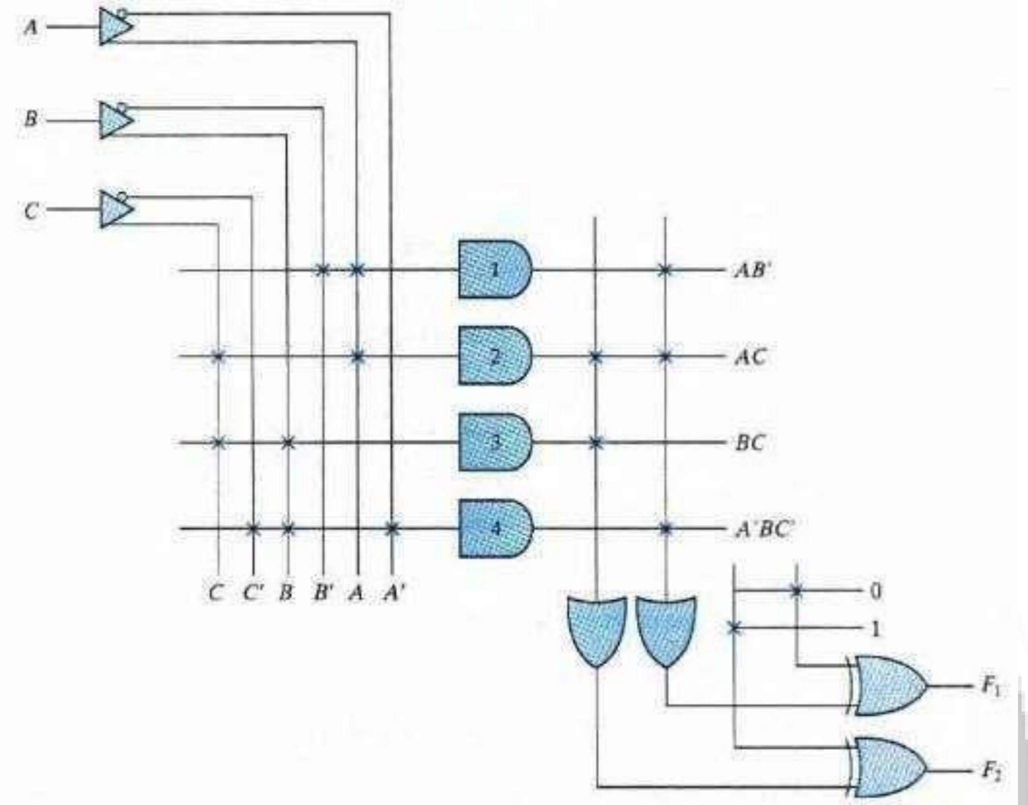
(b) Programmable array logic (PAL)

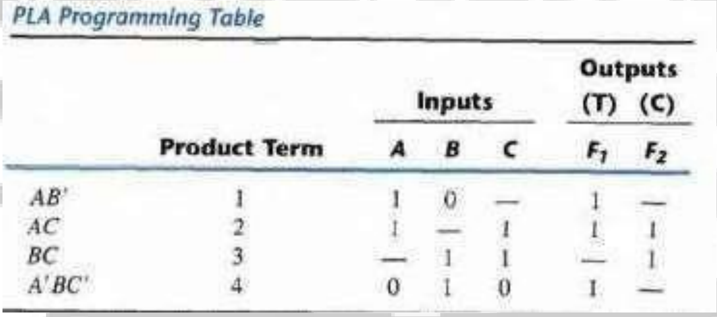
(c) Programmable logic array (PLA)

## PROGRAMMABLE LOGIC ARRAY

The PLA is similar in concept to the PROM, except that the PLA does not provide full decoding 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 terms are then connected to OR gates to provide the sum of products for the required Boolean functions. The diagram uses the array logic graphic symbols for complex circuits. Each input goes through a buffer-inverter combination, shown in the diagram with a composite graphic symbol, that has both the true and complement outputs. Each input and its complement are connected to the inputs of each ANDgate, as indicated by the intersections between the vertical and horizontal lines. The outputs of the AND gates are connected to the inputs of each OR gate. The output of the OR gate goes to an XOR gate, where the other input can be programmed to receive a signal equal to either logic 1 or logic 0. The output is inverted when the XOR input is connected to 1. The output does not change when the XOR input is connected to 0. The particular Boolean functions implemented in the PLA of below figure are,

$$F_1 = AB' + AC + A'BC'$$
$$F_2 = (AC + BC)'$$





PLA Programming Table

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

## PROGRAMMABLE ARRAY LOGIC

The PAL is a programmable logic device with a fixed OR array and a programmable AND array. Because only the AND gates are programmable, the PAL is easier to program than, but is not as flexible as the PLA. The below figure shows, the logic configuration of a typical PAL with four inputs and four outputs. Each input has a buffer- inverter gate, and each output is generated by a fixed OR gate. There are four sections in the unit. each composed of an AND-OR array that is three wide, the term used to indicate that there are three programmable AND gates in each section and one fixed OR gate. Each AND gate has 10 programmable input connections, shown in the diagram by 10 vertical lines intersecting each horizontal line. The horizontal line symbolizes the multiple-input configuration of the AND gate. One of the outputs is connected to a buffer-inverter gate and then fed back into two inputs of the AND gates.

In designing with a PAL, the Boolean functions must be simplified to fit into each section. Unlike the situation with a PLA, a product term cannot be shared among two or more OR gates. Therefore, each function can be simplified by itself, without regard to common product terms. The number of product terms in each section is fixed and if the number of terms in the function is too large, it may be necessary to use two sections to implement one Boolean function.

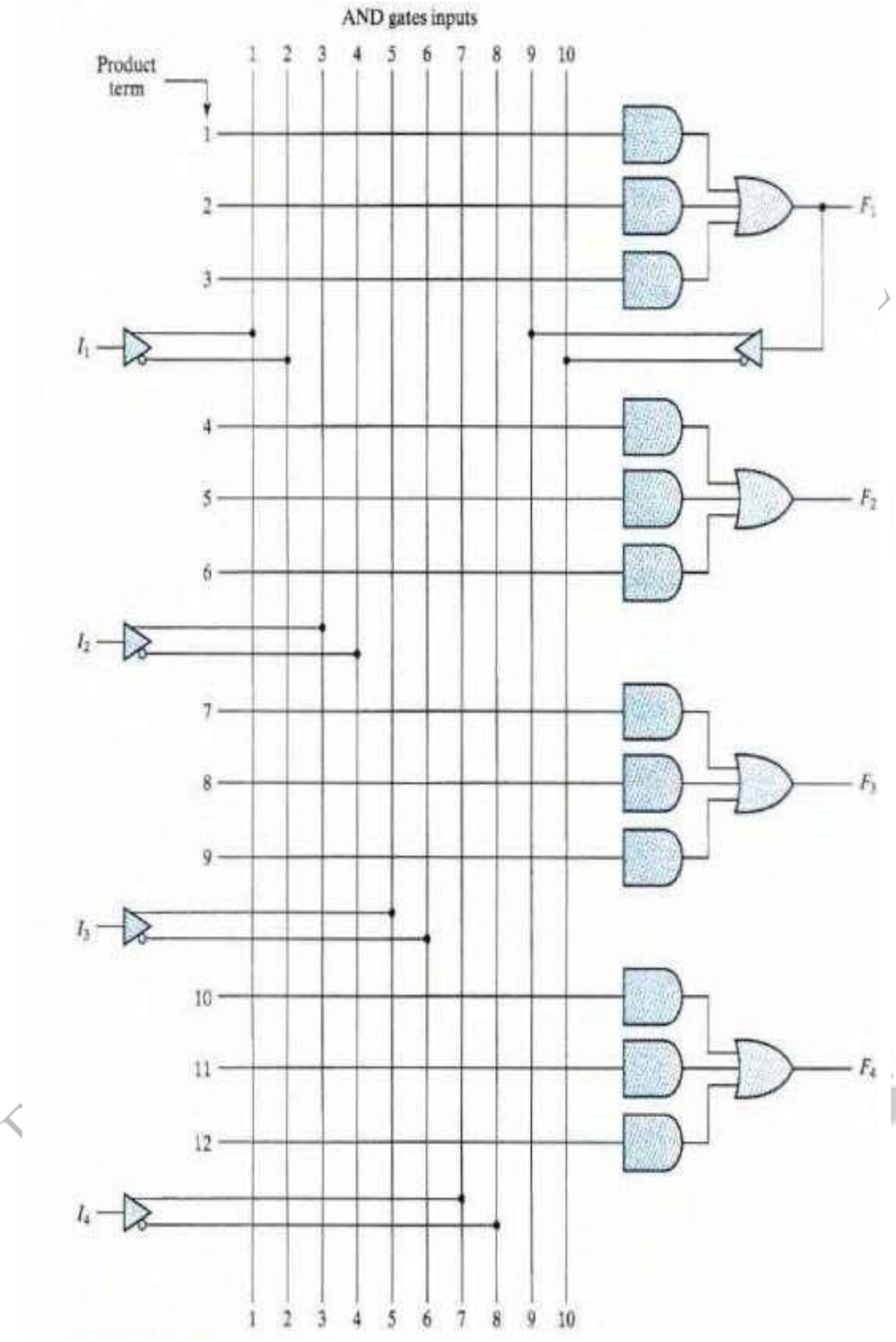
As an example of using a PAL in the design of a combinational circuit, consider the following Boolean functions, given in sum-of-minterms form:

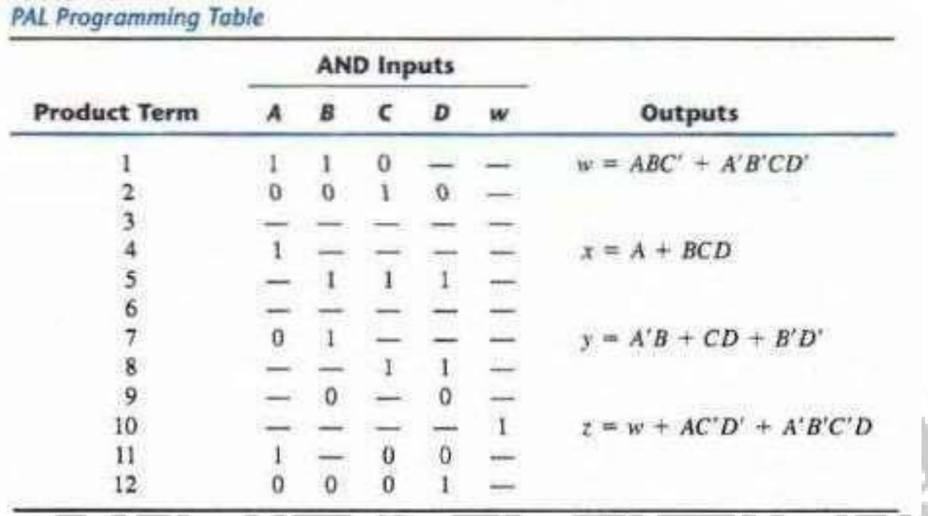
$$w(A, B, C, D) = \Sigma(2, 12, 13)$$
$$x(A, B, C, D) = \Sigma(7, 8, 9, 10, 11, 12, 13, 14, 15)$$
$$y(A, B, C, D) = \Sigma(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)$$
$$z(A, B, C, D) = \Sigma(1, 2, 8, 12, 13)$$

Simplifying the four functions to a minimum number of terms results in the following

$$w = ABC' + A'B'CD'$$
$$x = A + BCD$$
$$y = A'B + CD + B'D'$$
$$z = ABC' + A'B'CD' + AC'D' + A'B'C'D$$
$$= w + AC'D' + A'B'C'D$$

Boolean functions:





PAL Programming Table

| | AND Inputs | | | | | |
|---|---|---|---|---|---|---|
| **Product Term** | ***A*** | ***B*** | ***C*** | ***D*** | ***w*** | **Outputs** |
| 1 | 1 | 1 | 0 | — | — | $w = ABC' + A'B'CD'$ |
| 2 | 0 | 0 | 1 | 0 | — | |
| 3 | — | — | — | — | — | |
| 4 | 1 | — | — | — | — | $x = A + BCD$ |
| 5 | — | 1 | 1 | 1 | — | |
| 6 | — | — | — | — | — | |
| 7 | 0 | 1 | — | — | — | $y = A'B + CD + B'D'$ |
| 8 | — | — | 1 | 1 | — | |
| 9 | — | 0 | — | 0 | — | |
| 10 | — | — | — | — | 1 | $z = w + AC'D' + A'B'C'D$ |
| 11 | 1 | — | 0 | 0 | — | |
| 12 | 0 | 0 | 0 | 1 | — | |