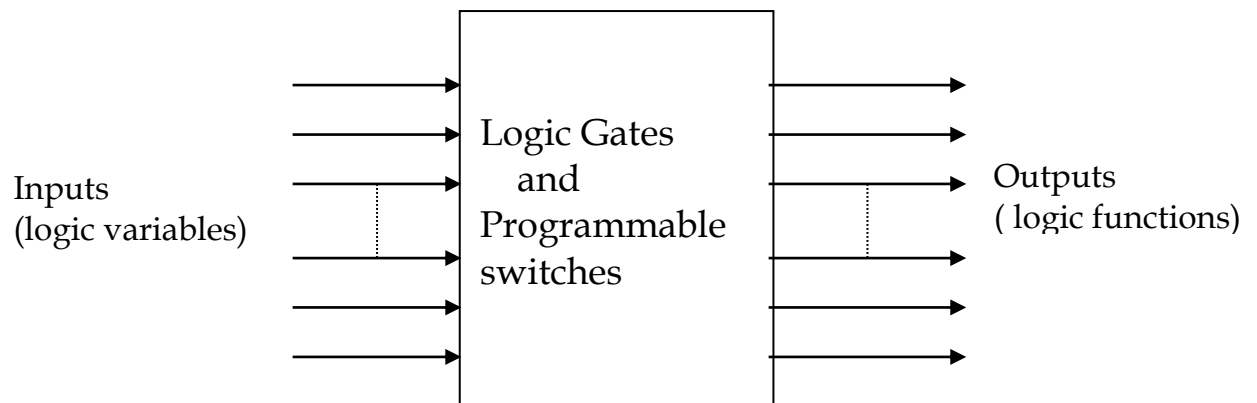


## **Programmable Logic Devices (PLDs)**

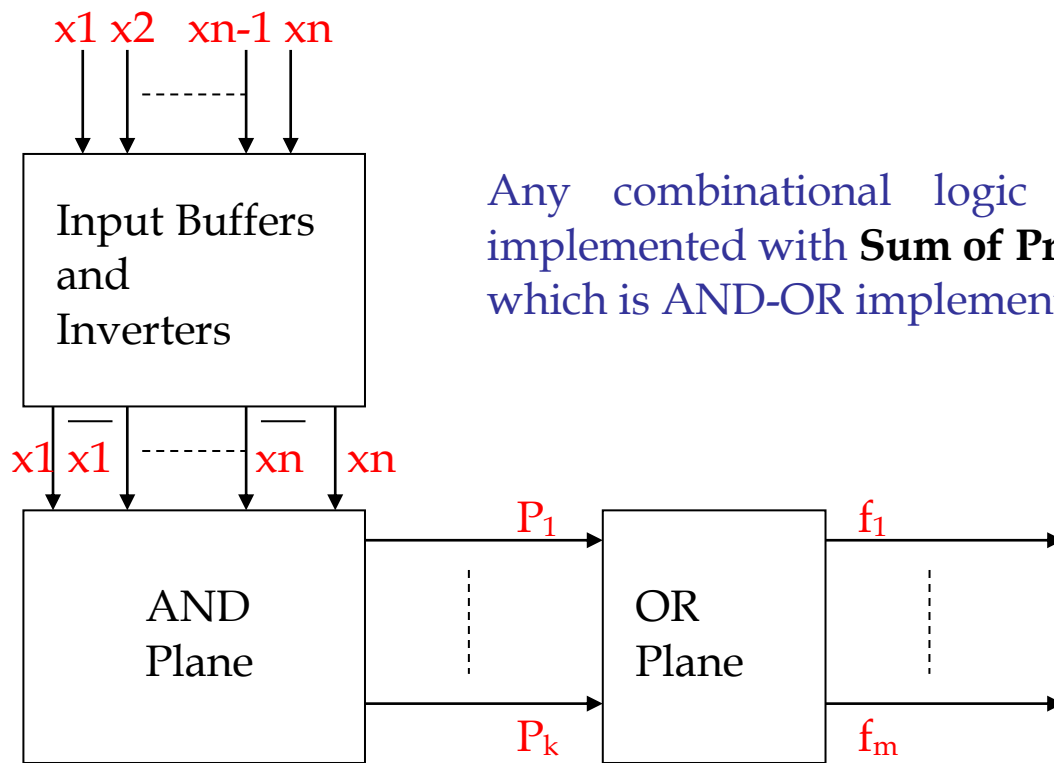
**PLDs basically store binary information in a volatile/nonvolatile device. Data is specified by designer and physically inserted (Programmed) into the device.**

**Any Boolean function can be expressed in Sum of Products. SOP. In turn the SOP can be implemented in an AND-OR form.**

**ROM, PAL, PLA are different optimized implementations of a given circuit using the AND-OR planes.**



**Programmable Logic Device as a black box**



Any combinational logic can be implemented with **Sum of Product** which is AND-OR implementation.

## Programming the AND OR Plane

**ROM: AND Fixed, OR Programmable**

**PAL: AND Programmable, OR fixed**

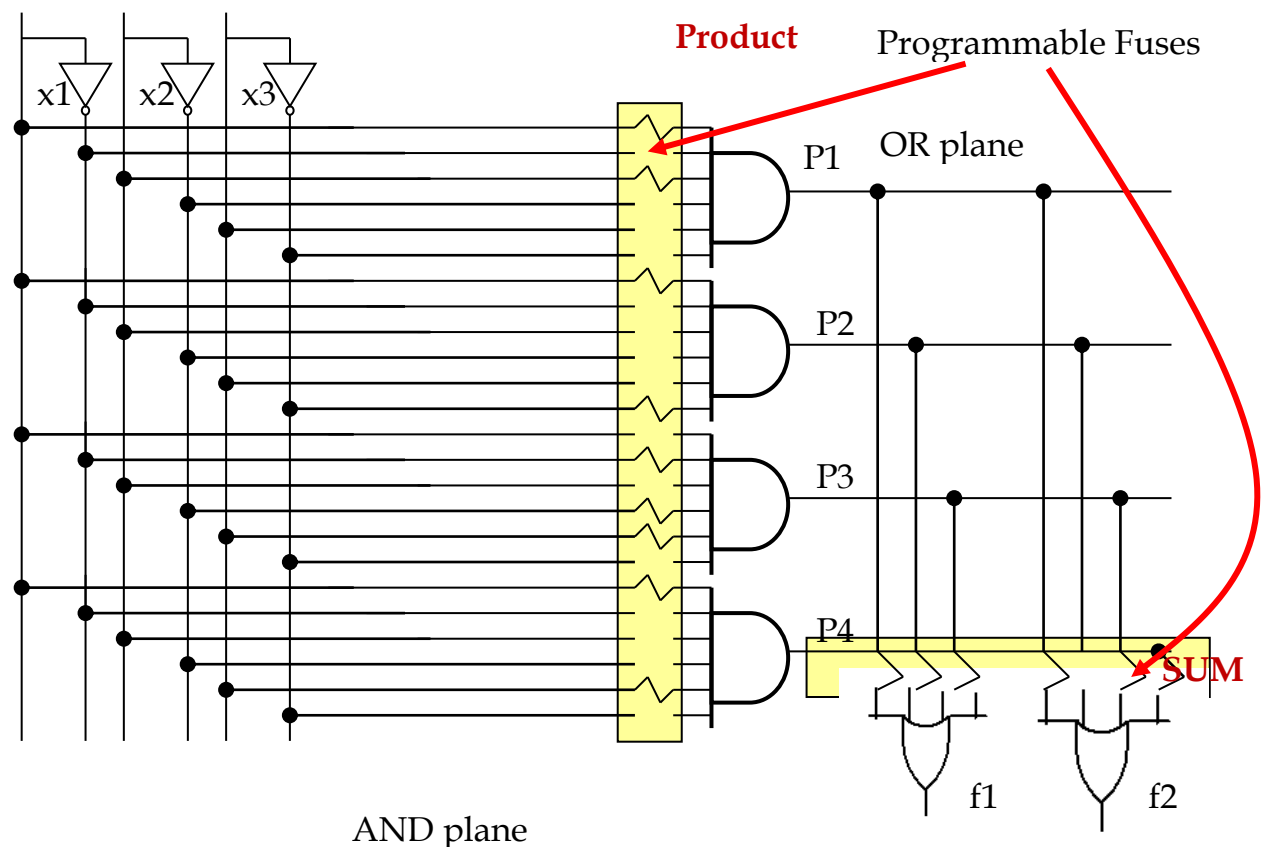
**PLA: AND Programmable, OR Programmable**

**FPGA: Programmable Logic Blocks, Programmable Interconnect**

Example of programming the AND-OR planes

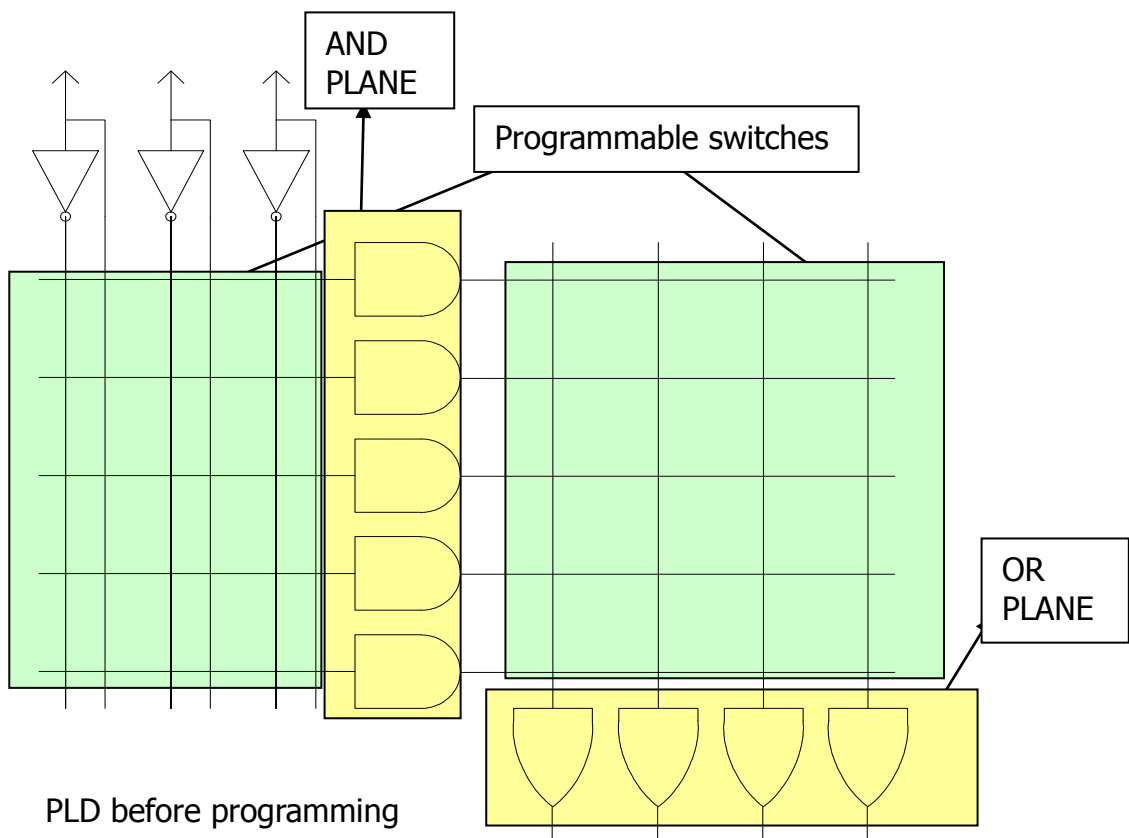
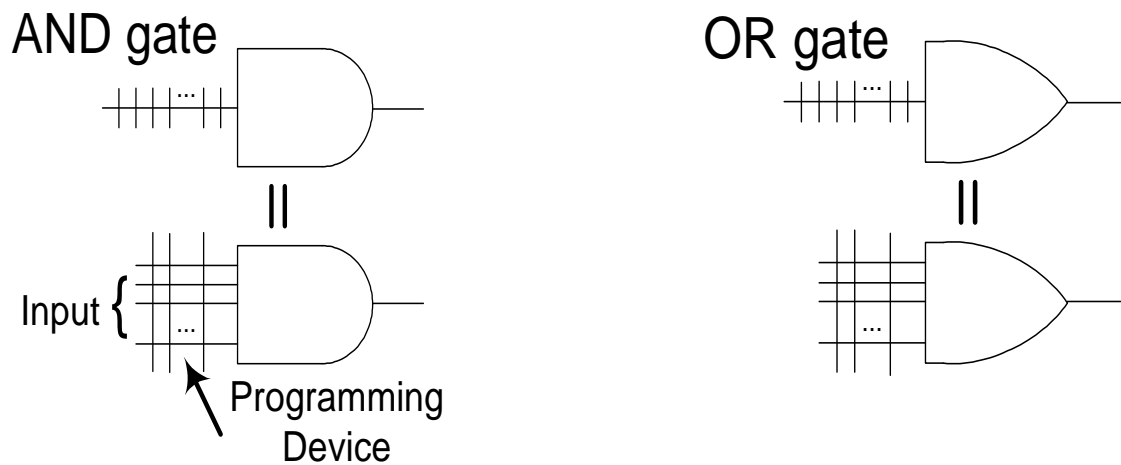
$$f_1(x_1, x_2, x_3) = x_1 \cdot x_2 + x_1 \cdot x_3' + x_1' \cdot x_2' \cdot x_3$$

$$f_2(x_1, x_2, x_3) = x_1 \cdot x_2 + x_1' \cdot x_2' \cdot x_3 + x_3$$



## PLD basic cells:

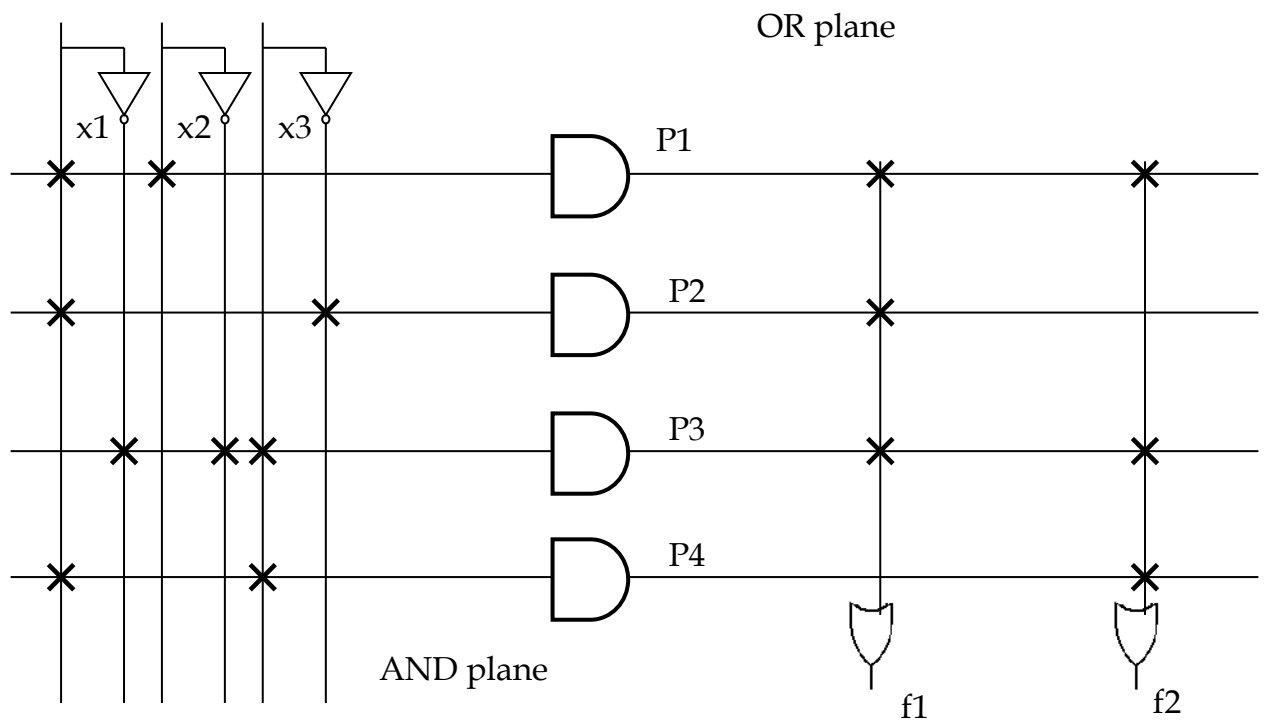
The number of inputs and outputs are usually high. To avoid drawing all the inputs and outputs all the inputs to a gate is shown with one line only and the actual inputs are shown as an intersection as shown below



So the previous example after programming will look like this

$$f_1(x_1, x_2, x_3) = x_1 \cdot x_2 + x_1 \cdot x_3' + x_1' \cdot x_2' \cdot x_3$$

$$f_2(x_1, x_2, x_3) = x_1 \cdot x_2 + x_1' \cdot x_2' \cdot x_3 + x_1 \cdot x_3$$



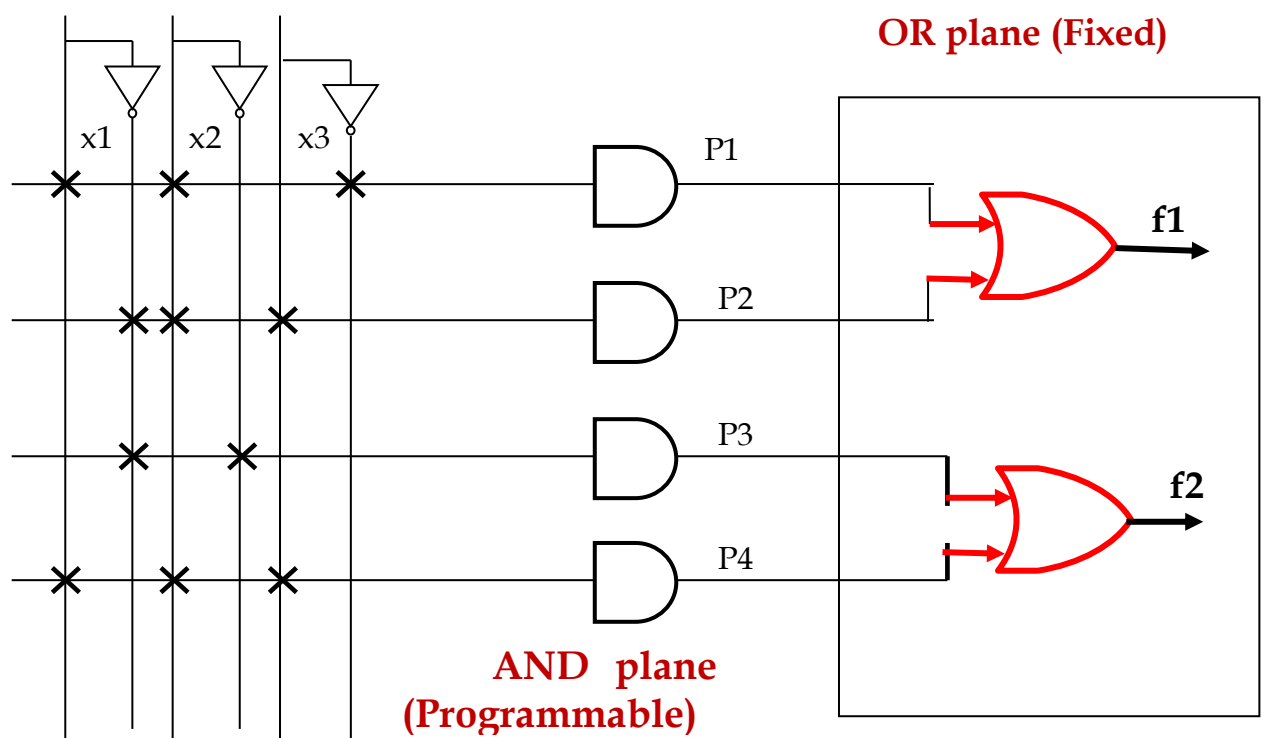
# Programmable Array Logic (PAL )

In this category of Programmable Logic devices the AND Plane is programmable and the OR Plane is fixed

Previous Example:

$$f1(x1,x2,x3) = x1 x2 x3' + x1' .x2 x3$$

$$f2(x1,x2,x3) = x1' .x2' + x1 .x2 x3$$



## Programmable Logic Array (PLA)

Both AND-Plane and the OR-Plane are programmable.

### Example:

Implement the following functions on a PLA

$$\left. \begin{aligned} F_0 &= A\bar{C} + AB \\ F_1 &= A + \bar{B}\bar{C} \\ F_2 &= \bar{B}\bar{C} + AB \\ F_3 &= \bar{B}C + A \end{aligned} \right\}$$

Size of the Device related to:

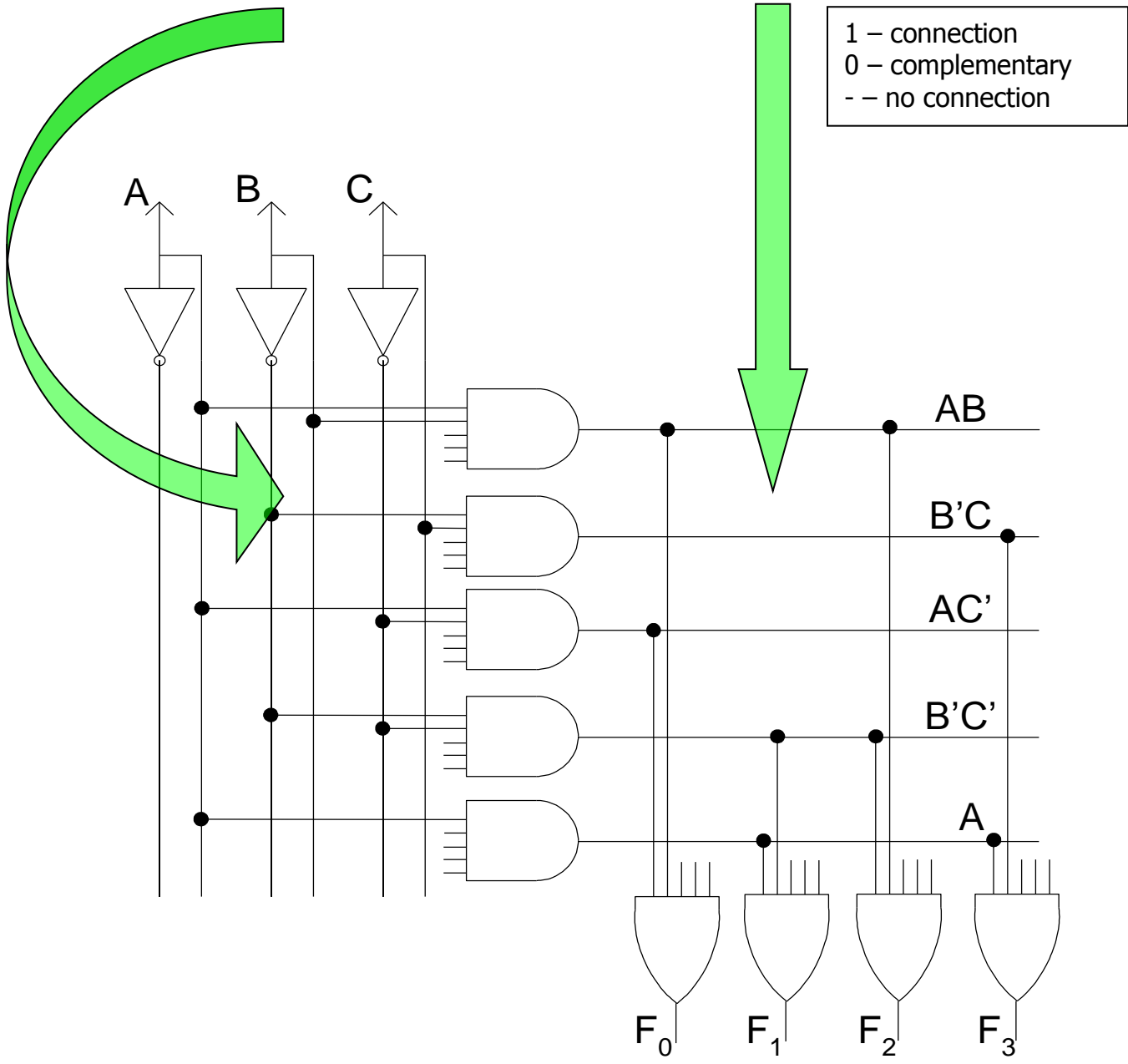
- cost;
  - speed;
  - power.
- ?

### You should choose:

- 3 variable input
- 5 product terms -> 5 AND gates
- 4 outputs -> 4 OR gates

Personality Matrix

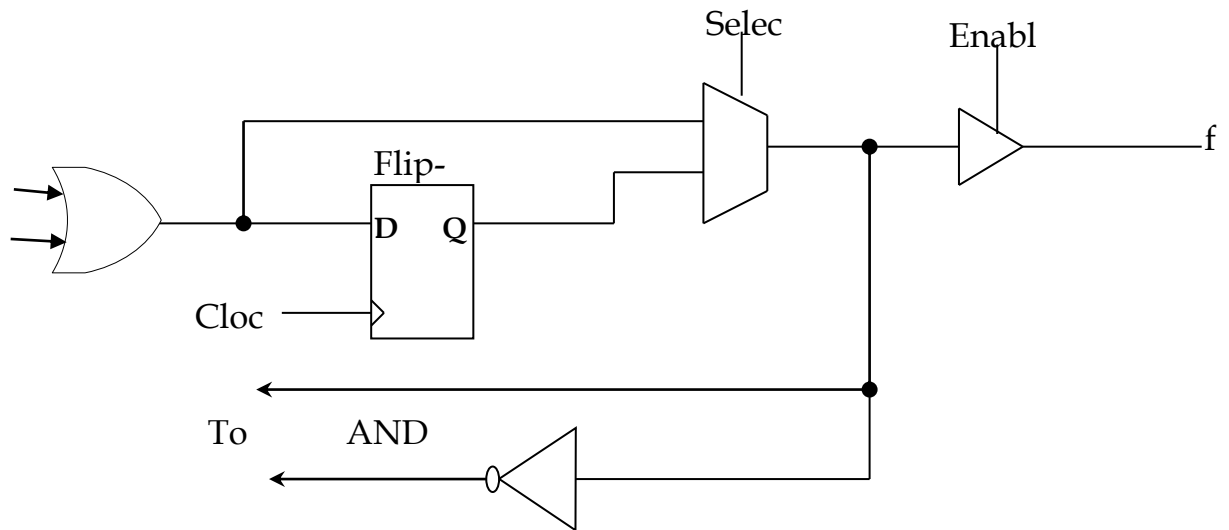
Product	Input			Output			
	A	B	C	F0	F1	F2	F3
$AB$	1	1	-	1	-	1	-
$\overline{B}C$	-	0	1	-	-	-	1
$A\overline{C}$	1	-	0	1	-	-	-
$\overline{B}\overline{C}$	-	0	0	-	1	1	-
$A$	1	-	-	-	1	-	1
	For AND gate programming			For OR gate programming			



After programming  
 (\* unused inputs of AND gates are connected to "1";  
 \*\* unused inputs of OR gates are connected to "0")



When it is necessary to implement a Finite state machine, then we need a PAL or PLA that has extra circuitry such as a register and a Feedback path, Then the following general circuitry can be added to some of the PLD devices.



**Example:**

Implement the Finite state Machine given by the following equations on a suitable PAL. The PAL has SR flip Flops. These equations were derived for a particular state machine using SR-Flip Flops.

Excitation Vector:

$$S_2 = P' Q y_1, \quad R_2 = y_2, \quad S_1 = P' Q', \quad R_1 = Q + P$$

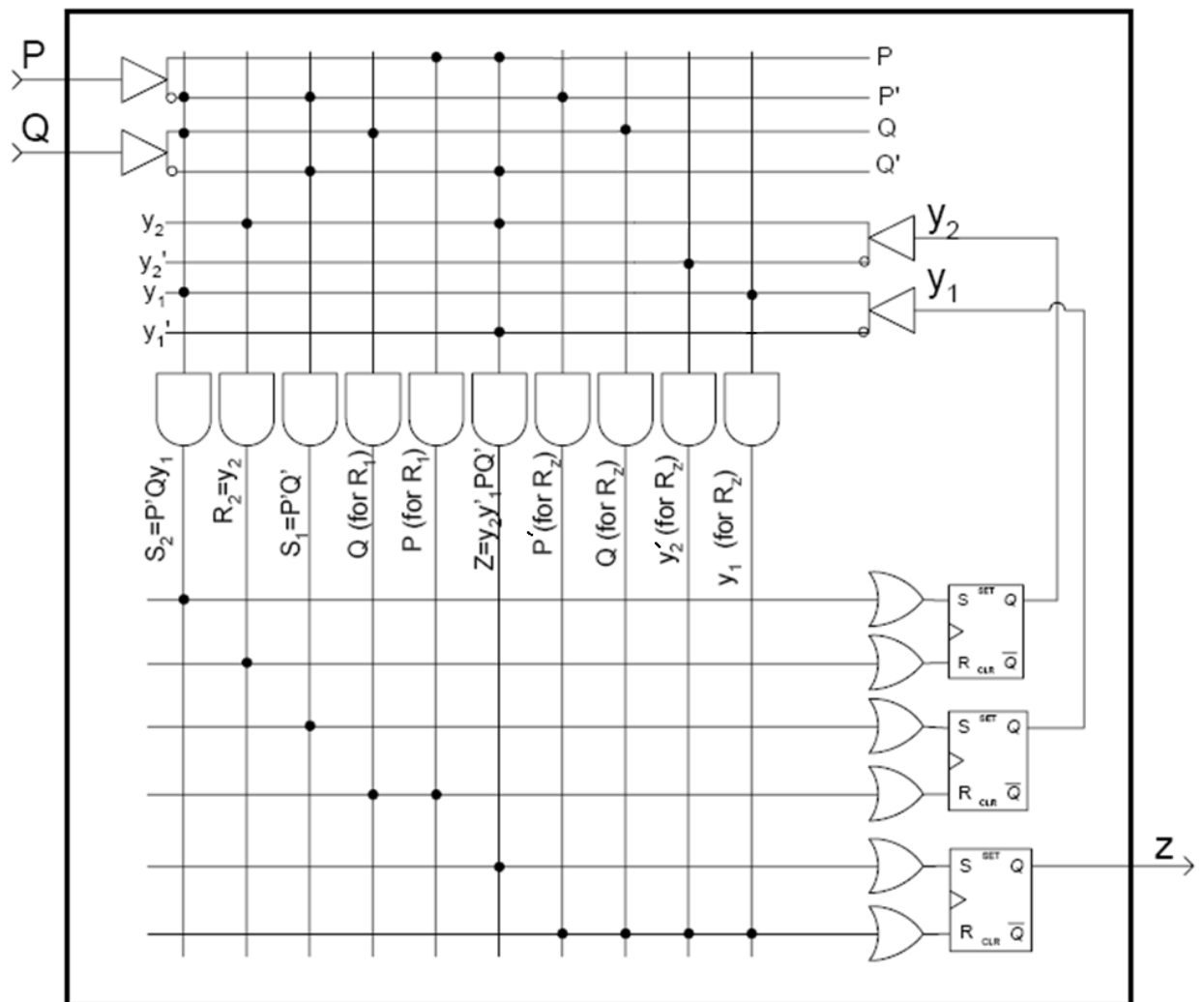
Output :

$$Z = y_2 y_1' P Q'$$

P & Q – are inputs

$y_2$  &  $y_1$  are the states

z is the output



## Programming the devices:

### Programming the PAL

Logic expressions first must be minimized before programming the Pal. PALs are programmed electrically using binary patterns (as [JEDEC ASCII/hexadecimal](#) files) and by using electronic programming device from a certain manufacturer such as [Data I/O Corporation's](#) Model 60A Logic Programmer.

User circuits are implemented in the programmable devices by configuring or programming these devices. Due to the large number of programmable switches in commercial chips; it is not feasible to specify manually the desired programming state for each switch. CAD systems are used to solve this problem.

Computer system that runs the CAD tools is connected to a programming unit. After design of a circuit has been completed, CAD tool generates a file (programming file or fuse map) that specifies the state of each switch in PLD. PLD is then placed into the programming unit and the programming file is transferred from the computer system to the unit. Programming unit then programs each switch individually.

Most General of all PLDs is the FPGA or Field Programmable Gate Arrays, where, both cells of combinational logic and the routing can be programmed. Current FPGA's are available in 10nm technology having millions of gates.



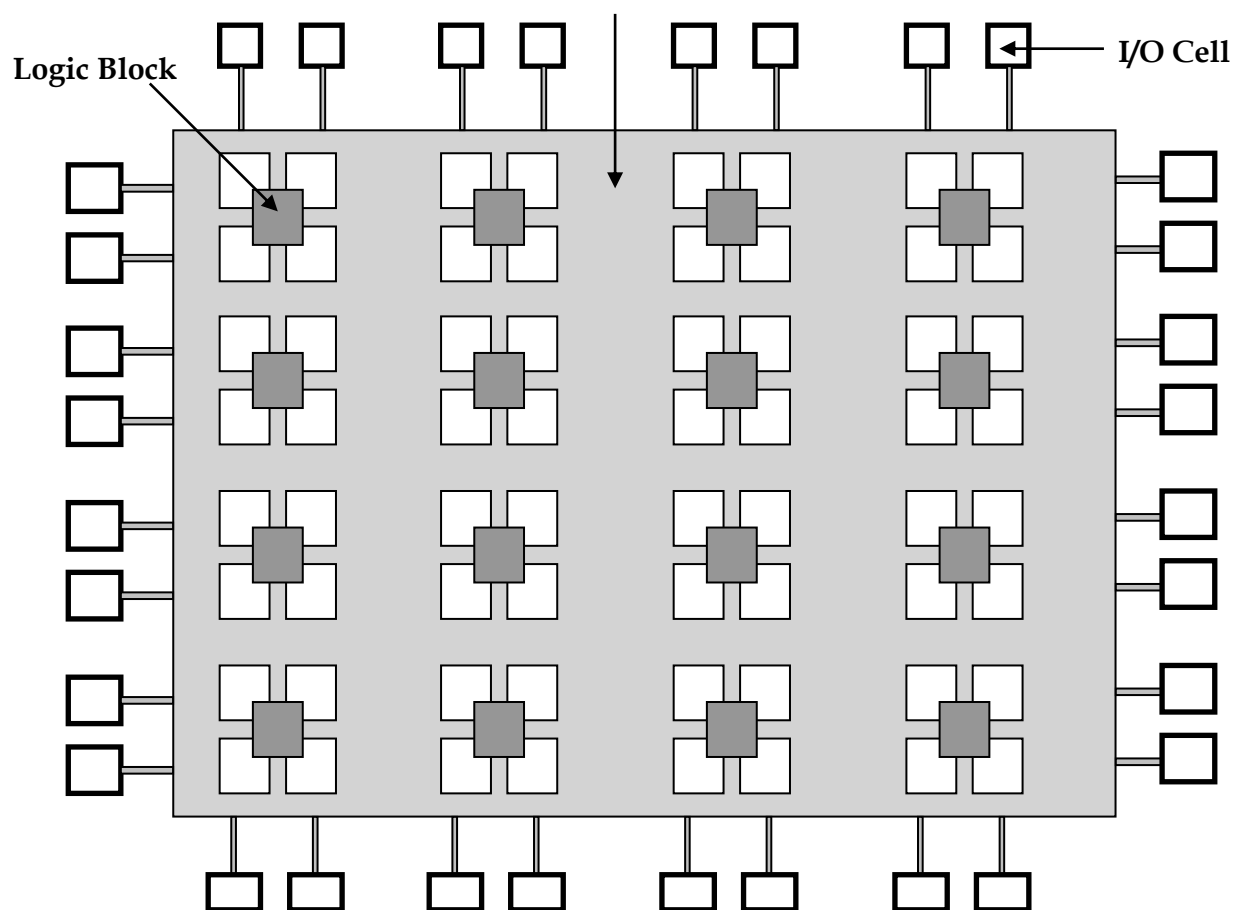
Prices range from \$500 \$1000

So all you need is a Personal computer, PLD CAD tool, The programming device and its software and the kind of PLD that the device accepts.

Tutorial for PAL can be found at

[http://courses.cs.washington.edu/courses/cse370/06au/tutorials/Tutorial\\_PAL.htm](http://courses.cs.washington.edu/courses/cse370/06au/tutorials/Tutorial_PAL.htm)

Schematic of an FPGA is shown below.



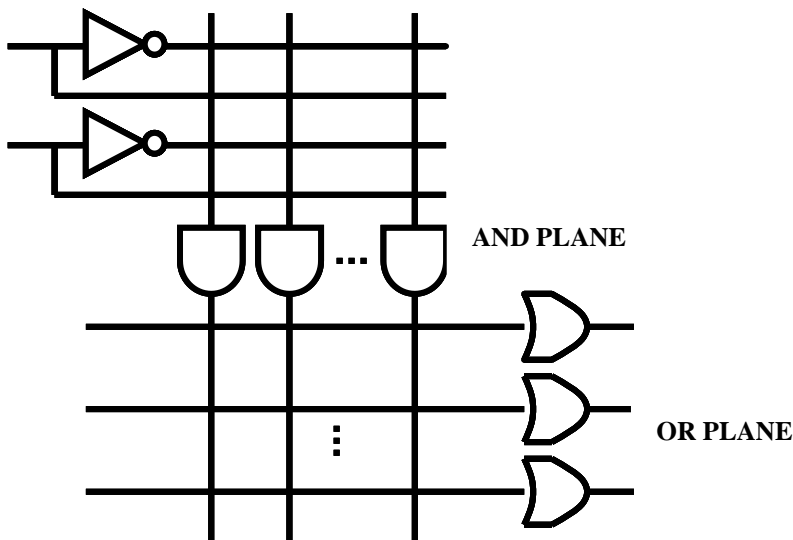
## Example:

Design a PLA, PAL and ROM at a gate level to realize the following sum of product functions:

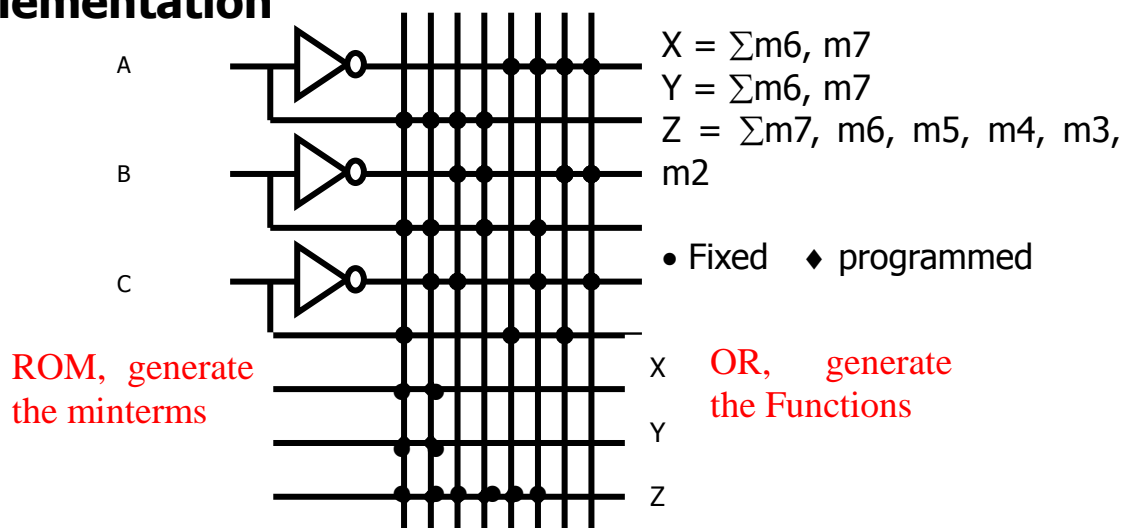
$$X(A,B,C) = A.B + A.B.C + A.B.C$$

$$Y(A,B,C) = A.B + A.B.C$$

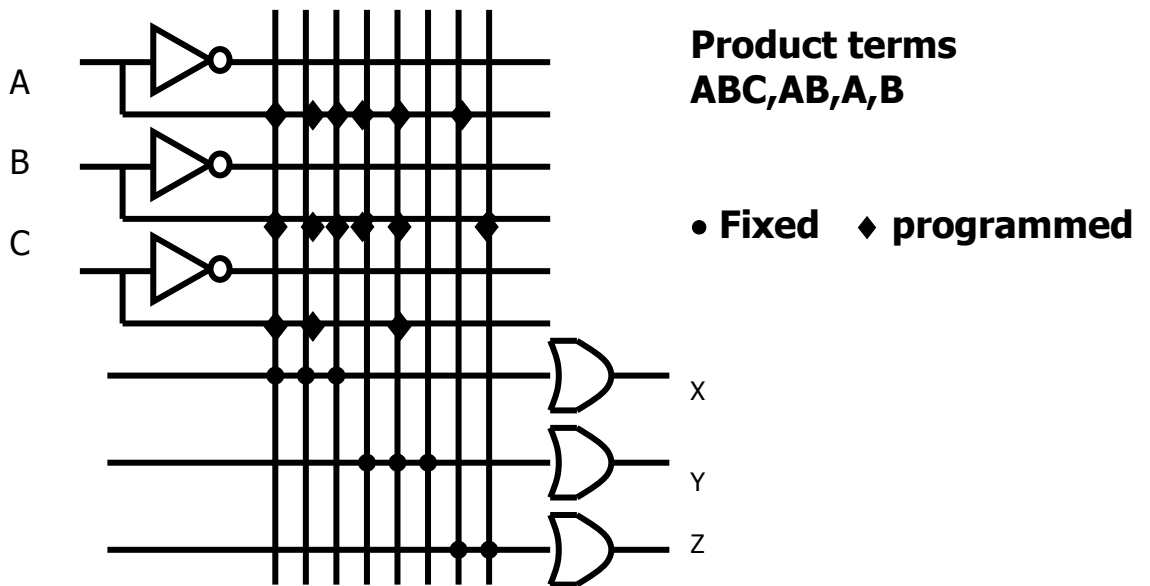
$$Z(A,B,C) = A + B$$



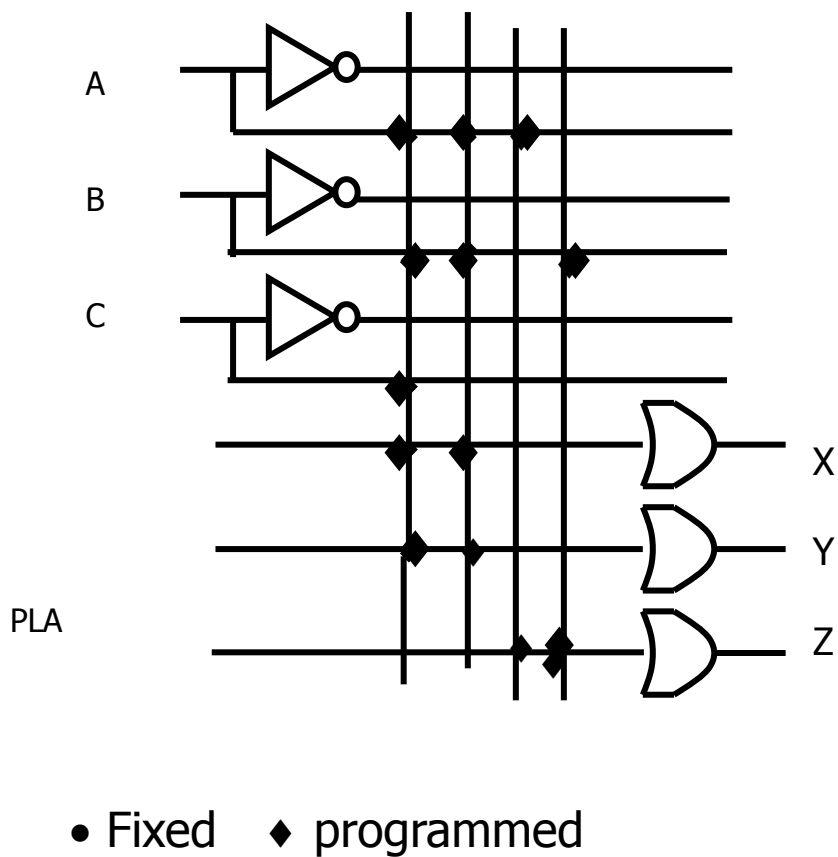
## ROM Implementation



## PAL Implementation



## PLA Implementation



## Example 2

Design a function generator  $F=x^2$  where  $x$  is a 3-bit unsigned binary number. What size of ROM is required for this generator. Show your ROM and its content.

Answer:

For the 3-bit word there is 8 possibilities, we draw the Truth Table

$x_2 x_1 x_0$	$x$	$f_5 f_4 f_3 f_2 f_1 f_0$	$F$
0 0 0	0	0 0 0 0 0 0	0
0 0 1	1	0 0 0 0 0 1	1
0 1 0	2	0 0 0 1 0 0	4
0 1 1	3	0 0 1 0 0 1	9
1 0 0	4	0 1 0 0 0 0	16
1 0 1	5	0 1 1 0 0 1	25
1 1 0	6	1 0 0 1 0 0	36
1 1 1	7	1 1 0 0 0 1	49

Therefore the size of the ROM will be 8 words each of 6 bits.

It may be further minimized by noticing that  $f_0=x_0$  and that  $f_1$  is=0. If we take these into account the size of the ROM is 8 words of 4 bits each. ●

