Programmable Logic Devices (PLDs)

Lesson Objectives:

In this lesson you will be introduced to some types of Programmable Logic Devices (PLDs):

- PROM, PAL, PLA, CPLDs, FPGAs, etc.
- How to implement digital circuits using PLAs and PALs.

Introduction:

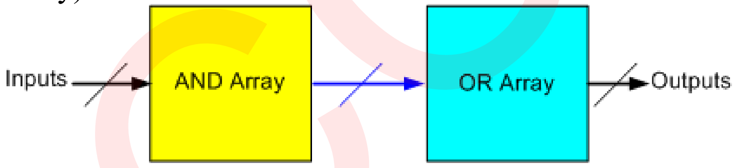
An IC that contains large numbers of gates, flip-flops, etc. that can be *configured by the user* to perform different functions is called a ***Programmable Logic Device (PLD)***.

The internal logic gates and/or connections of PLDs can be changed/configured by a programming process.

One of the simplest programming technologies is to use fuses. In the original state of the device, all the fuses are intact.

Programming the device involves blowing those fuses along the paths that must be removed in order to obtain the particular configuration of the desired logic function.

PLDs are typically built with an *array* of AND gates (AND-array) and an *array* of OR gates (OR-array).



Advantages of PLDs:

Problems of using standard ICs:

Problems of using standard ICs in logic design are that they require hundreds or thousands of these ICs, considerable amount of circuit board space, a great deal of time and cost in inserting, soldering, and testing. Also require keeping a significant inventory of ICs.

Advantages of using PLDs:

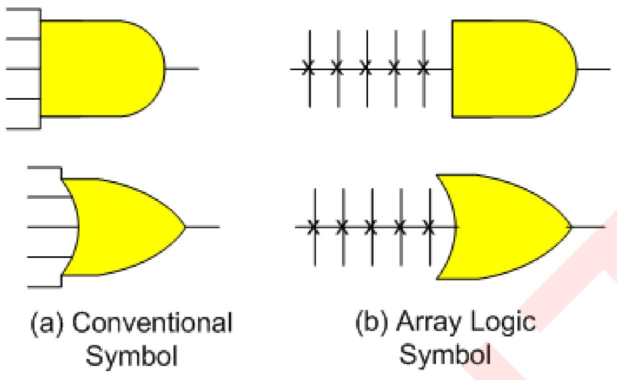
Advantages of using PLDs are less board space, faster, lower power requirements (i.e., smaller power supplies), less costly assembly processes, higher reliability (fewer ICs and circuit connections means easier troubleshooting), and availability of design software.

There are three fundamental types of standard PLDs: ***PROM, PAL,*** and ***PLA***.

A fourth type of PLD, which is discussed later, is the ***Complex Programmable Logic Device (CPLD),*** e.g., ***Field Programmable Gate Array (FPGA).***
A typical PLD may have hundreds to millions of gates.

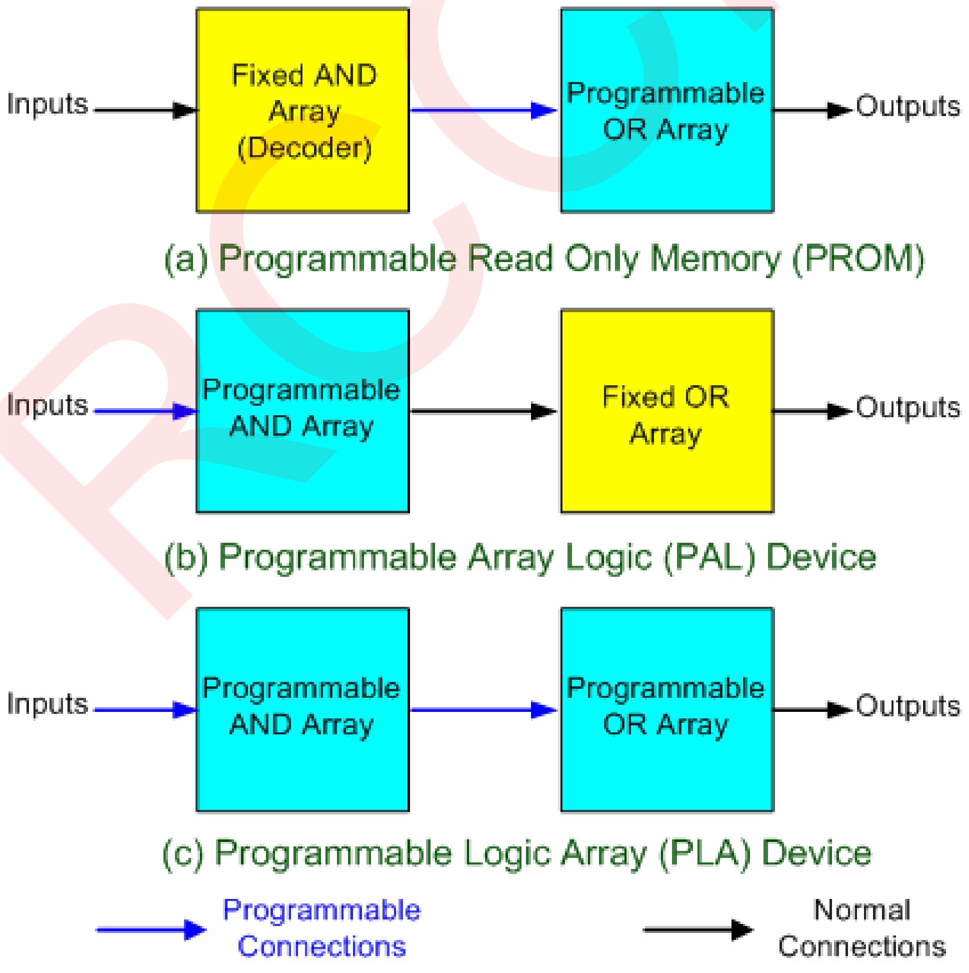
In order to show the internal logic diagram for such technologies in a concise form, it is necessary to have special symbols for array logic.

Figure shows the conventional and array logic symbols for a multiple input AND and a multiple input OR gate.



Three Fundamental Types of PLDs:

The three fundamental types of PLDs differ in the placement of programmable connections in the AND-OR arrays. Figure shows the locations of the programmable connections for the three types.



- The ***PROM (Programmable Read Only Memory)*** has a fixed AND array (constructed as a decoder) and programmable connections for the output OR gates array. The PROM implements Boolean functions in sum-of-minterms form.

- The ***PAL (Programmable Array Logic)*** device has a programmable AND array and fixed connections for the OR array.

- The ***PLA (Programmable Logic Array)*** has programmable connections for both AND and OR arrays. So it is the most flexible type of PLD.

The ROM (Read Only Memory) or PROM (Programmable Read Only Memory):
The input lines to the AND array are hard-wired and the output lines to the OR array are programmable.

Each AND gate generates one of the possible AND products (i.e., minterms).

In the previous lesson, you have learnt how to implement a digital circuit using ROM.

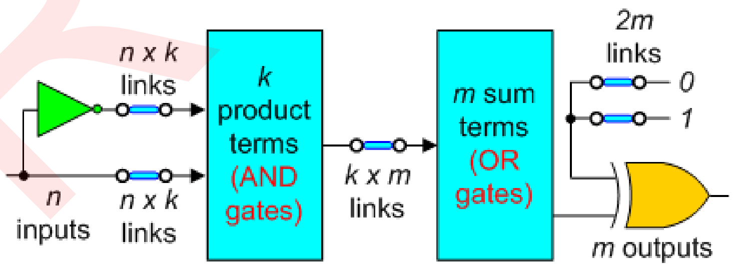
The PLA (Programmable Logic Array):
In PLAs, instead of using a decoder as in PROMs, a number (***k***) of AND gates is used where $k < 2^n$, (***n*** is the number of inputs).

Each of the AND gates can be programmed to generate a product term of the input variables and does not generate all the minterms as in the ROM.

The AND and OR gates inside the PLA are initially fabricated with the links (fuses) among them.

The specific Boolean functions are implemented in sum of products form by opening appropriate links and leaving the desired connections.

A block diagram of the PLA is shown in the figure. It consists of ***n*** inputs, ***m*** outputs, and ***k*** product terms.



The product terms constitute a group of ***k*** AND gates each of $2n$ inputs.

Links are inserted between all ***n*** inputs and their complement values to each of the AND gates.

Links are also provided between the outputs of the AND gates and the inputs of the OR gates.

Since PLA has ***m***-outputs, the number of OR gates is ***m***.

The output of each OR gate goes to an XOR gate, where the other input has two sets of links, one connected to logic 0 and other to logic 1. It allows the output function to be generated either in the **true** form or in the **complement** form.

The output is inverted when the XOR input is connected to 1 (since $X \oplus 1 = X'$). The output does not change when the XOR input is connected to 0 (since $X \oplus 0 = X$).

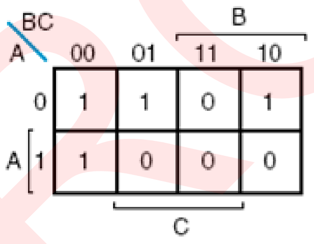
Thus, the total number of programmable links is $2n \times k + k \times m + 2m$.

The size of the PLA is specified by the number of inputs ***(n)***, the number of product terms ***(k)***, and the number of outputs ***(m)***, (the number of sum terms is equal to the number of outputs).

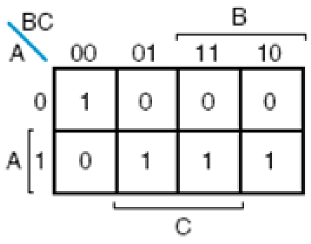
Example:
Implement the combinational circuit having the shown truth table, using PLA.

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

Each product term in the expression requires an AND gate. To minimize the cost, it is necessary to simplify the function to a minimum number of product terms.



$$F_1 = \bar{A}\bar{B} + \bar{A}\bar{C} + \bar{B}\bar{C}$$
$$\bar{F}_1 = AB + AC + BC$$



$$F_2 = AB + AC + \bar{A}\bar{B}\bar{C}$$
$$\bar{F}_2 = \bar{A}C + \bar{A}B + A\bar{B}\bar{C}$$

Designing using a PLA, a careful investigation must be taken in order to reduce the distinct product terms. Both the true and complement forms of each function should be simplified to see which one can be expressed with fewer product terms and which one provides product terms that are common to other functions.

The combination that gives a minimum number of product terms is:
$F_1' = AB + AC + BC$ *or* $F_1 = (AB + AC + BC)'$
$F_2 = AB + AC + A'B'C'$

This gives only 4 distinct product terms: *AB, AC, BC,* and *A'B'C'*.

So the PLA table will be as follows:

PLA programming table

	Product term	Inputs A B C	Outputs (C) F_1	Outputs (T) F_2
AB	1	1 1 –	1	1
AC	2	1 – 1	1	1
BC	3	– 1 1	1	–
$\overline{A}\overline{B}\overline{C}$	4	0 0 0	–	1

For each product term, the inputs are marked with 1, 0, or – (dash). If a variable in the product term appears in its normal form (unprimed), the corresponding input variable is marked with a 1.

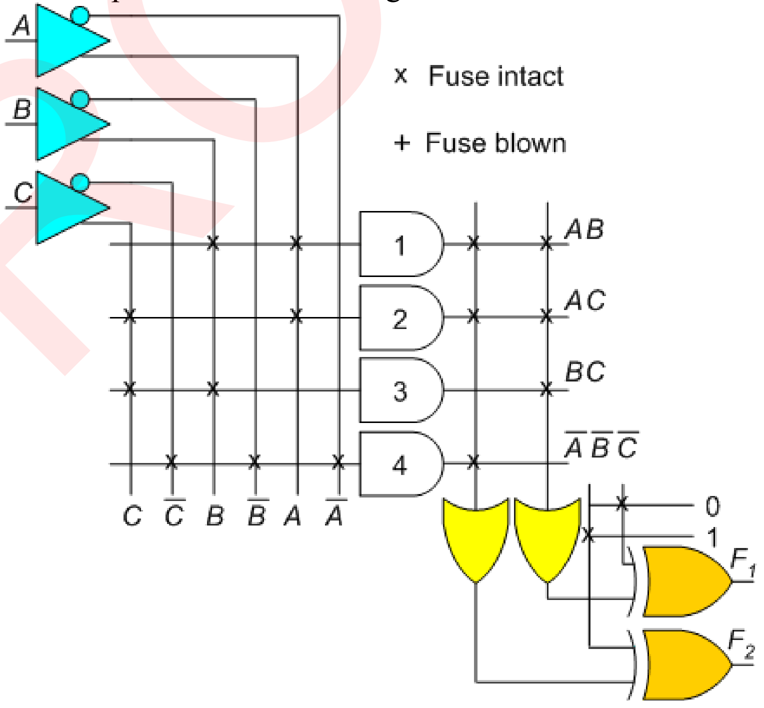
A 1 in the **Inputs** column specifies a path from the corresponding input to the input of the AND gate that forms the product term.

A 0 in the **Inputs** column specifies a path from the corresponding complemented input to the input of the AND gate. A dash specifies no connection.

The appropriate fuses are blown and the ones left intact form the desired paths. It is assumed that the open terminals in the AND gate behave like a 1 input.

In the Outputs column, a **T (true)** specifies that the other input of the corresponding XOR gate can be connected to 0, and a **C (complement)** specifies a connection to 1.

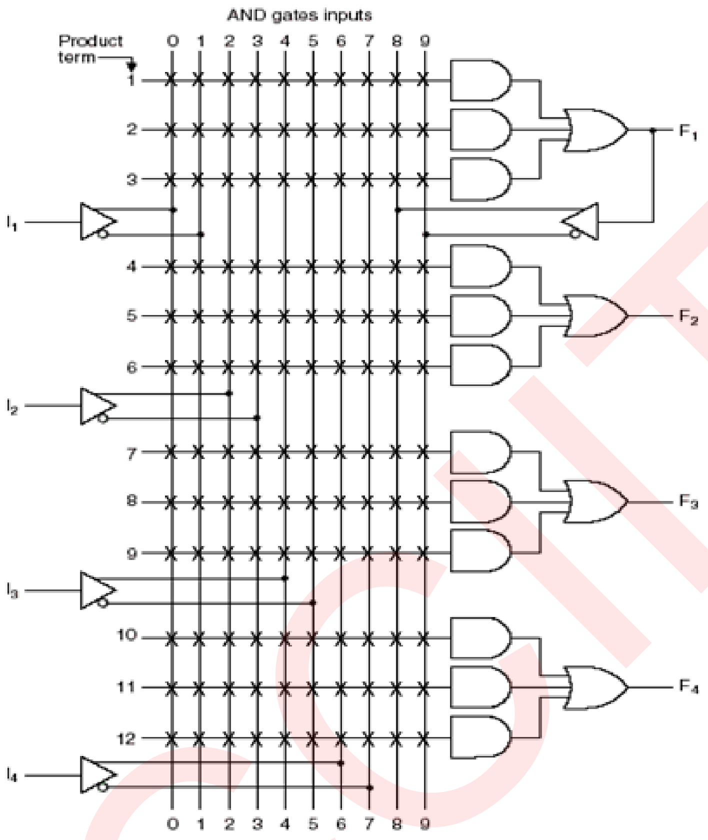
Note that output ***F_1*** is the normal (or true) output even though a **C** (for complement) is marked over it. This is because ***F_1'*** is generated with AND-OR circuit prior to the output XOR. The output XOR complements the function *F_1'* to produce the true *F_1* output as its second input is connected to logic 1.



The PAL (Programmable Array Logic):

The PAL device is a PLD with a fixed OR array and a programmable AND array.

As only AND gates are programmable, the PAL device is easier to program but it is not as flexible as the PLA.



The device shown in the figure has 4 inputs and 4 outputs. Each input has a buffer-inverter gate, and each output is generated by a fixed OR gate.

The device has 4 sections, each composed of a 3-wide AND-OR array, meaning that there are 3 programmable AND gates in each section.

Each AND gate has 10 programmable input connections indicating by 10 vertical lines intersecting each horizontal line. The horizontal line symbolizes the multiple input configuration of an AND gate.

One of the outputs F_1 is connected to a buffer-inverter gate and is fed back into the inputs of the AND gates through programmed connections.
(see animation in authorware version)

Designing using a PAL device, the Boolean functions must be simplified to fit into each section.

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 or more sections to implement one Boolean function.

Example:
Implement the following Boolean functions using the PAL device as shown above:

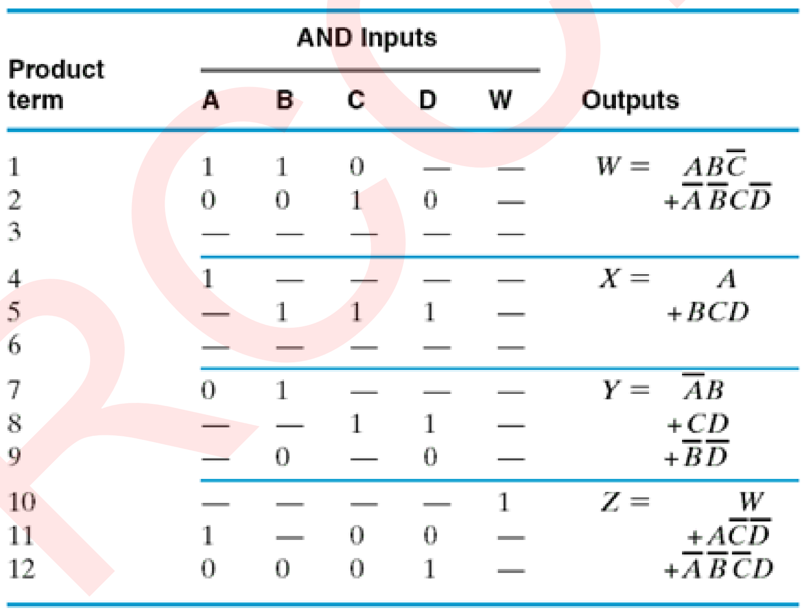
W(A, B, C, D) = ∑m(2, 12, 13)
X(A, B, C, D) = ∑m(7, 8, 9, 10, 11, 12, 13, 14, 15)
Y(A, B, C, D) = ∑m(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)
Z(A, B, C, D) = ∑m(1, 2, 8, 12, 13)

Simplifying the 4 functions to a minimum number of terms results in the following Boolean functions:

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

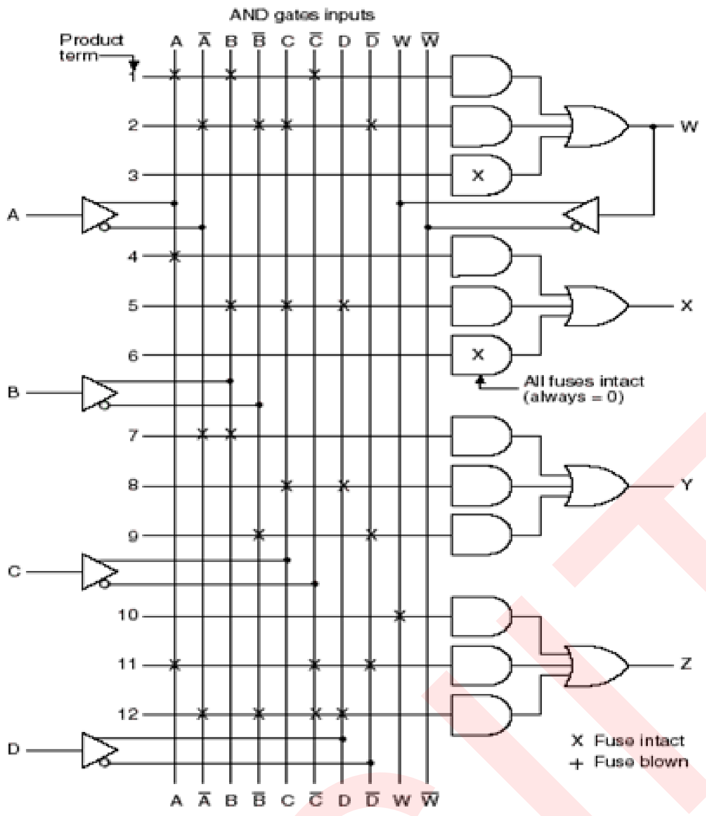
Note that the function for ***Z*** has four product terms. The logical sum of two of these terms is equal to ***W***. Thus, by using ***W***, it is possible to reduce the number of terms for ***Z*** from four to three, so that the function can fit into the given PAL device.

The PAL programming table is similar to the table used for the PLA, except that only the inputs of the AND gates need to be programmed.



Product term	AND Inputs					Outputs
	A	B	C	D	W	
1	1	1	0	—	—	$W = AB\overline{C}$
2	0	0	1	0	—	$+\overline{A}\,\overline{B}C\overline{D}$
3	—	—	—	—	—	
4	1	—	—	—	—	$X = A$
5	—	1	1	1	—	$+BCD$
6	—	—	—	—	—	
7	0	1	—	—	—	$Y = \overline{A}B$
8	—	—	1	1	—	$+CD$
9	—	0	—	0	—	$+\overline{B}\,\overline{D}$
10	—	—	—	—	1	$Z = W$
11	1	—	0	0	—	$+A\overline{C}\,\overline{D}$
12	0	0	0	1	—	$+\overline{A}\,\overline{B}\,\overline{C}D$

The figure shows the connection map for the PAL device, as specified in the programming table.
(see animation in authorware version)

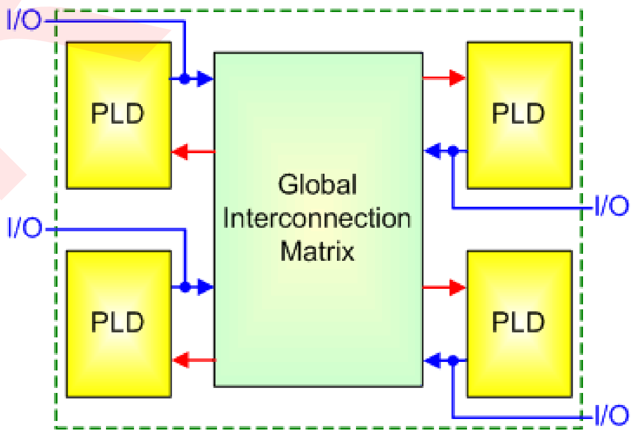

AND gates inputs
Product term
A $\bar{A}$ B $\bar{B}$ C $\bar{C}$ D $\bar{D}$ W $\bar{W}$
1
2
3
4
5
6
7
8
9
10
11
12
A
B
C
D
W
X
Y
Z
All fuses intact (always = 0)
X Fuse intact
+ Fuse blown

Since both W and X have two product terms, third AND gate is not used. If all the inputs to this AND gate left intact, then its output will always be 0, because it receives both the true and complement of each input variable i.e., AA' =0

Complex Programmable Logic Devices (CPLDs):

A CPLD contains a bunch of PLD blocks whose inputs and outputs are connected together by a global interconnection matrix.

Thus a CPLD has two levels of programmability: each PLD block can be programmed, and then the interconnections between the PLDs can be programmed.



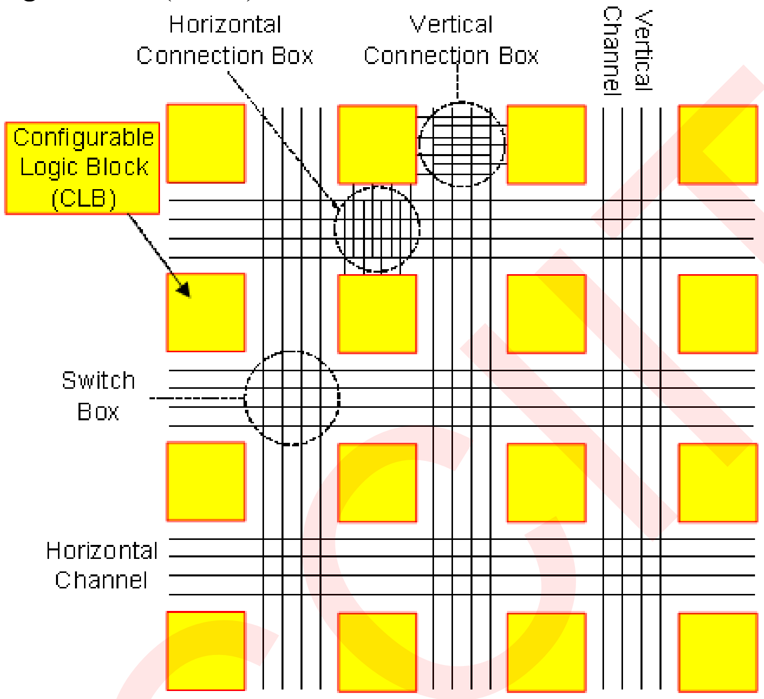
Field Programmable Gate Arrays (FPGAs):

The FPGA consists of 3 main structures:

1. Programmable logic structure,
2. Programmable routing structure, and
3. Programmable Input/Output (I/O).

1. Programmable logic structure

The programmable logic structure FPGA consists of a 2-dimensional array of configurable logic blocks (CLBs).



Each CLB can be configured (programmed) to implement *any* Boolean function of its input variables. Typically CLBs have between 4-6 input variables. Functions of larger number of variables are implemented using more than one CLB.

In addition, each CLB typically contains 1 or 2 FFs to allow implementation of sequential logic.

Large designs are partitioned and mapped to a number of CLBs with each CLB configured (programmed) to perform a particular function.

These CLBs are then connected together to fully implement the target design. Connecting the CLBs is done using the FPGA programmable routing structure.

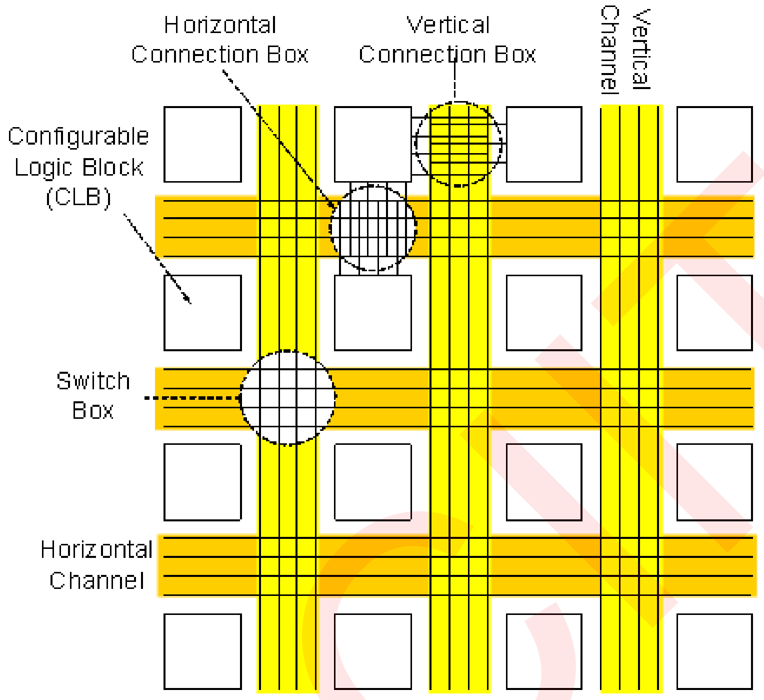
2. Programmable routing structure

To allow for flexible interconnection of CLBs, FPGAs have 3 *programmable* routing resources:

1. Vertical and horizontal routing channels which consist of different length wires that can be connected together if needed. These channel run vertically and horizontally between columns and rows of CLBs as shown in the Figure.

2. Connection boxes, which are a set of programmable links that can connect input and output pins of the CLBs to wires of the vertical or the horizontal routing channels.

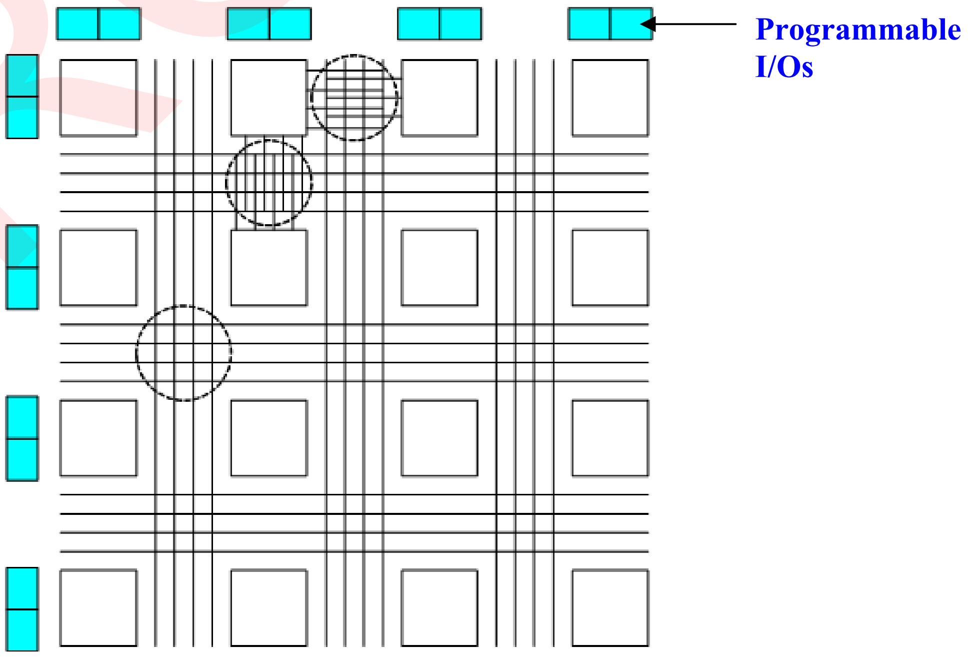
3. Switch boxes, located at the intersection of the vertical and horizontal channels. These are a set of programmable links that can connect wire segments in the horizontal and vertical channels. **(see animation in authorware version)**



3. Programmable I/O

These are mainly buffers that can be configured either as input buffers, output buffers or input/output buffers.

They allow the pins of the FPGA chip to function either as input pins, output pins or input/output pins.


Programmable I/Os