# Exploring PLDs

Sayak Haldar(111205038)
sayakhaldar@ymail.com

Shuvam Bosana(111205042)
shuvambosana0705@gmail.com

Subham Banerjee(111205014)
subhambansphs@gmail.com

April 20, 2014

**Abstract**

This is an associated document to the main report we have submitted for our project. The main reason behind creating this associated document is that the students who will work on this project later, can easily understand the basics very clearly.

## Contents

# 1 Introduction

In this document we will discuss-

- History of PLDs and different types of it
- Advantages of using PLDs over normal ICs we have used in labs
- Structure of PLDs

## 2 History of Digital Logic Design

- **Discrete Devices:**1940-50s.
- **Discrete logic gates:**1950-60s.
- **Integrated circuits(IC)**
  - eg. TTL series 74LSxx, 1960s-...
    100s of different devices
  - Simple programmble logic devices 1970s-...
    2 levels of logic (SPLD)
  - Mask-programmable gate arrays 1970s-...
    used to implement application-specific
    integrated circuits (ASICs)
  - Large programmable logic devices 1980s-...
    Multiple SPLDs on a chip (CPLD)
    Programmable array of logic cells +
    Programmable interconnect (FPGA)

Now, we will focus on the History of PLDs

## 3 History of PLDs

A Programmable Logic Device (PLD) is an electronic component used to build reconfigurable digital circuits. Now before proceed into any kind of details **first,**

**we need to understand what is the basic difference of PLDs with Logic Gates?**

⇒ Logic gate has a fixed function i.e it has fixed output for various input combinations. But a PLD has an undefined function at the time of manufacture. Before the PLD can be used in a circuit it must be programmed, that is, reconfigured.
**Now there's another question, how are those programmable devices configured?**

⇒ One of the most common and simplest programming tehniques is to use fuse. In the original state of device, all the fuses are intact.
Programming the device involves blowing those fuses along the path that must be removed in order to obtain the perticular configuration of the desired logic function.

**Before PLDs were invented, we could use a ROM as a PLD...**

## 3.1 Using a ROM as a PLD

Before PLDs were invented, Read-Only Memory (ROM) chips were used to create arbitrary combinational logic functions of a number of inputs. Consider a ROM with m inputs (the address lines) and n outputs (the data lines). When used as a memory, the ROM contains 2m words of n bits each.

Now imagine that the inputs are driven not by an m-bit address, but by m independent logic signals. Theoretically, there are 22m possible Boolean functions of these m input signals. By Boolean function in this context is meant a single function that maps each of the 2m possible combinations of the m Boolean inputs to a single Boolean output. There are 22m possible distinct ways to map each of 2m inputs to a Boolean value, which explains why there are 22m such Boolean functions of m inputs.

Now, consider that each of the n output pins acts, independently, as a logic device that is specially selected to sample just one of the possible 22m such functions. At any given time, only one of the 2m possible input values can be present on the ROM, but over time, as the input values span their full possible domain, each output pin will map out its particular function of the 2m possible input values, from among the 22m possible such functions. Note that the structure of the ROM allows just n of the 22m possible such Boolean functions to be produced at the output pins. The ROM therefore becomes equivalent to n separate logic circuits, each of which generates a chosen function of the m inputs.

The advantage of using a ROM in this way is that any conceivable function of all possible combinations of the m inputs can be made to appear at any of the n outputs, making this the most general-purpose combinational logic device available for m input pins and n output pins.

Also, PROMs (Programmable ROMs), EPROMs (ultraviolet-erasable PROMs) and EEPROMs (Electrically Erasable PROMs) are available that can be programmed using a standard PROM programmer without requiring specialised hardware or software.

However, there are several disadvantages:-

- They are usually much slower than dedicated logic circuits.
- They cannot necessarily provide safe "covers" for asynchronous logic transitions so the PROM's outputs may glitch as the inputs switch.
- They consume more power.
- They are often more expensive than programmable logic, especially if high speed is required.

Since most ROMs do not have input or output registers, they cannot be used stand-alone for sequential logic. An external TTL register was often used for sequential designs such as state machines. Common EPROMs, for example the 2716, are still sometimes used in this way by hobby circuit designers, who often have some lying around. This use is sometimes called a 'poor man's PAL'.

## 3.2 Evolution of PLDs

The first user-programmable chip that could implement logic circuits was the Programmable Read Only Memory (PROM), in which address lines serve as logic circuit inputs and data lines as outputs. Logic functions, however, rarely require more than a few product terms, and a PROM contains a full decoder for its address inputs. PROMs are thus inefficient for realizing logic circuits. So, designers rarely use them for that purpose. The first device developed specifically for implementing logic circuits was the field-programmable logic array, or simply PLA for short. A PLA consists of two levels of logic gates: a programmable, wired-AND plane followed by a programmable, wired OR plane. A PLA's structure allows any of its inputs (or their complements) to be ANDed together in the AND plane; each AND plane output can thus correspond to any product term of the inputs. Similarly, users can configure each OR plane output to produce the logical sum of any AND plane output. With this structure, PLAs are well-suited for implementing logic functions in sum-ofproducts form. They are also quite versatile, since both the AND and OR terms can have many inputs (product literature often calls this feature “wide AND and OR gates”). When Philips introduced PLAs in the early 1970s, their main drawbacks were expense of manufacturing and somewhat poor speed performance. Both disadvantages arose from the two levels of configurable logic; programmable logic planes were difficult to manufacture and introduced significant propagation delays. To overcome these weaknesses, Monolithic Memories (MMI, later merged with Advanced Micro Devices) developed PAL devices. As Figure 1 shows, PALs feature only a single level of programmability—a programmable, wired-AND plane that feeds fixed-OR gates. To compensate for the lack of generality incurred by the fixed-OR plane, PALs come in variants with different numbers of inputs and outputs and various sizes of OR gates. To implement sequential circuits, PALs usually contain flip-flops connected to the OR gate outputs. The introduction of PAL devices profoundly affected digital hardware design, and they are the basis of some of the newer, more sophisticated architectures like SPLD(Simple Programmable Logic Device) , CPLD (Complex Programmable Logic Decice) anf FPGA and the techonogies whch are invented after FPGA.

## 3.3 Advantages of PLDs

⇒ **Problem of Using Standard ICs:**
If you want to design a small combinational or sequential cicuit then it’s better to use available ICs. But suppose you are implementing a cicuit which requires say hundred or thousand of those ICs.Then you need considerabe amount of cicuit board space also a great deal of time and cost in inserting, soldering, and testing. Now, suppose after testing it you find that you have done something wrong. then again it needs a a lot of time to find the exact position of error(i.e in which postition you have made a connection wrong) and again you need to reconnect it. That means the troubleshooting becomes a work of great difficulty.

⇒ **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(like we use Quartus II for programming an FPGA).

## 3.4 Three Fundamental PLDs

Three fundamental PLDs are PROM,PAL and PLA. Before proceed into the details that how to implement a function using those,we must first understand the basic differences between them.

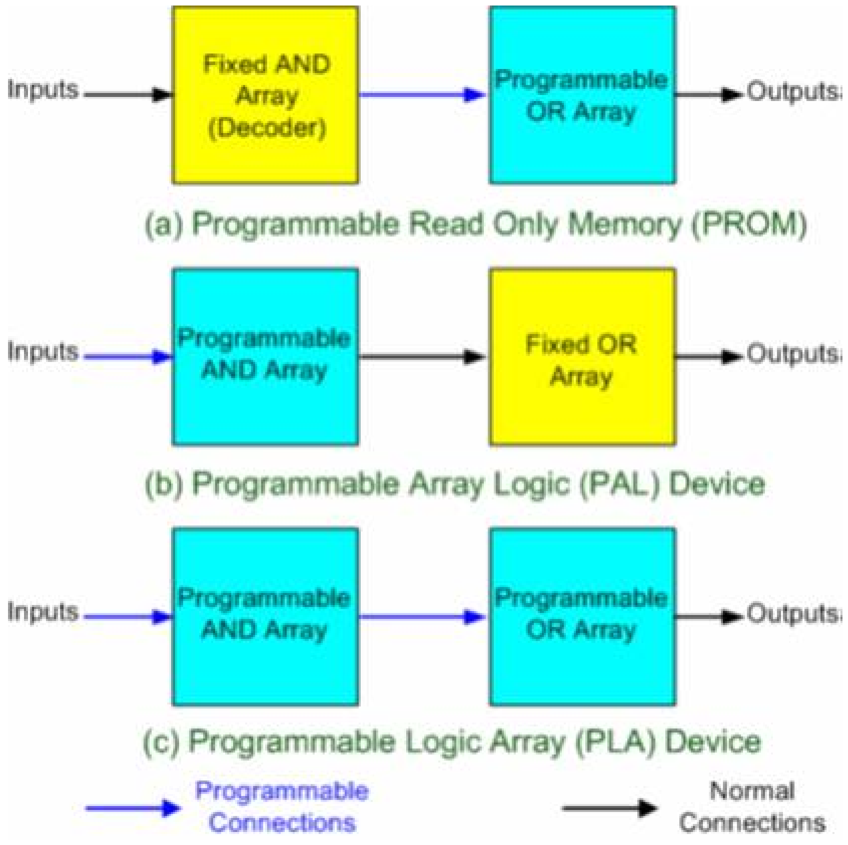


Figure 1: Three Fundamental PLDs

Now, clearly each of them consists of two level logic gates.

- PROM(Programmable Read Only Memory):-It Has a fixed AND Array followed by fused programmable OR Array(we have previously discussed about the fuse,which is a common programming technology). Now this fixed AND Array is provided by means of a Decoder.
- PAL(Programmable Array Logic):-It has a Fused Programmable AND Array followed by an Fixed OR Array.
- PLA(Programmable LOgic Array):-It has a Fused Programmable AND Array followed by Fused Programmable OR Array.

Now,there's difference between the conventional and array logic symbols for a multiple input AND and a multiple input OR gate. See the next picture

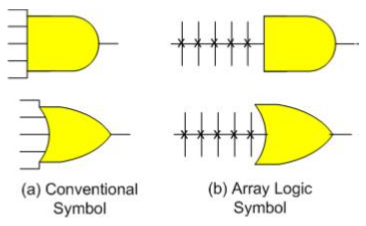


Figure 2: Difference Between Conventional and Array Logic Symbol

Now we will go into details of PROM,PAL and PLA.

### 3.4.1 Programmable Read Only Memory(PROM)

A Programmable Read-Only Memory (PROM) or Field Programmable Read-Only Memory (FPROM) or One-Time Programmable Non-Volatile Memory (OTP NVM) is a form of digital memory where the setting of each bit is locked by a fuse or antifuse.

⇒ Now we must understand one thing why it can can also be called Field Programmable:- because this is an integrated circuit designed to be configured by a customer or a designer after manufacturing,i.e this can be configured at the working field.

**Differences between ROM and PROM**
**ROM**

1. Non-programmable.
2. Not flexible(because data can't be changed.
3. It is comparatively slower.
4. Economical only when produced in large volumes.
5. Used in PC's.
6. Writing of cell contains mainly done by 'masking; mechanisms.

**PROM**

1. Programmable.
2. Flexible(Field-Programmable i.e. can be programmed in any place of work).
3. It is comparatively faster.
4. Economical even when produced in small volume.
5. Mainly used for reserarche and development purpose.
6. Mainly done electrically.

**Why PROM is called Non-Volatile?**
Because it can store information even when it is not powered.
**Programming of a PROM:**
A typical PROM comes with all bits reading as "1". Burning a fuse bit during Programming causes the bit to read as "0". The memory can be programmed just once after manufacturing by "blowing" the fuses, which is an irreversible process. Blowing a fuse opens a connection while programming an antifuse closes a connection (hence the name). While it is impossible to "unblow" the fuses, it is often possible to change the contents of the memory after initial programming by blowing additional fuses, changing some remaining "1" bits in the memory to "0"s. (Once all of the bits are "0", no further programming change is possible.)

The bit cell is programmed by applying a high-voltage pulse not encountered during normal operation across the gate and substrate of the thin oxide transistor (around 6V for a 2 nm thick oxide, or 30MV/cm) to break down the oxide between gate and substrate. The positive voltage on the transistor's gate forms an inversion channel in the substrate below the gate, causing a tunneling current to flow through the oxide. The current produces additional traps in the oxide, increasing the current through the oxide and ultimately melting the oxide and forming a conductive channel from gate to substrate. The current required to form the conductive channel is around $100\mu$A/100nm2 and the breakdown occurs in approximately $100\mu$s or less.

### 3.4.2 Programmable Array Logic(PAL)

Programmable Array Logic (PAL) is a family of Programmable Logic Device Semiconductors used to implement logic functions in digital circuits introduced by Monolithic Memories, Inc. (MMI) in March 1978.
PAL devices also are field programmable.

But, early Programmable Devices including PROM,PAL are one time programmable. That means though they can be configured after manufacture,they could not be updated and reused after its initial programming,i.e. they can be configured once.

**PAL Architecture:**

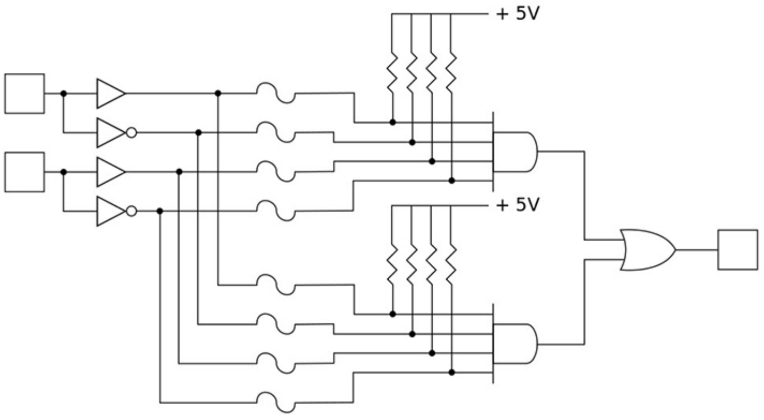


Figure 3: A Simplified PAL Device

⇒ **Programmable Logic Plane:** The Programmable Logic Plane is a Programmable Read-Only Memory (PROM) array that allows the signals present on the devices pins (or the logical complements of those signals) to be routed to an output logic macrocell.
PAL devices have arrays of transistor cells arranged in a "fixed-OR, programmable-AND" plane used to implement "Sum-Of-Products" binary logic equations for each of the outputs in terms of the inputs and either synchronous or asynchronous feedback from the outputs.

⇒ **Output Logic:** The early 20-pin PALs had 10 inputs and 8 outputs. The outputs were active low and could be registered or combinational. Members of the PAL family were available with various output structures called "output logic macrocells" or OLMCs. Prior to the introduction of the "V" (for "variable") series, the types of OLMCs available in each PAL were fixed at the time of manufacture. (The PAL16L8 had 8 combinational outputs and the PAL16R8 had 8 registered outputs. The PAL16R6 had 6 registered and 2 combinational while the PAL16R4 had 4 of each.) Each output could have up to 8 product terms (effectively AND gates), however the combinational

outputs used one of the terms to control a bidirectional output buffer. There were other combinations that had fewer outputs with more product terms per output and were available with active high outputs. The 16X8 family or registered devices had an XOR gate before the register. There were also similar 24-pin versions of these PALs.
AMD 22V10 Output Macrocell- this fixed output structure often frustrated designers attempting to optimize the utility of PAL devices because output structures of different types were often required by their applications. (For example, one could not get 5 registered outputs with 3 active high combinational outputs.) So, in June 1983 AMD introduced the 22V10, a 24 pin device with 10 output logic macrocells.[5] Each macrocell could be configured by the user to be combinational or registered, active high or active low. The number of product terms allocated to an output varied from 8 to 16. This one device could replace all of the 24 pin fixed function PAL devices. Members of the PAL "V" ("variable") series included the PAL16V8, PAL20V8 and PAL22V10.

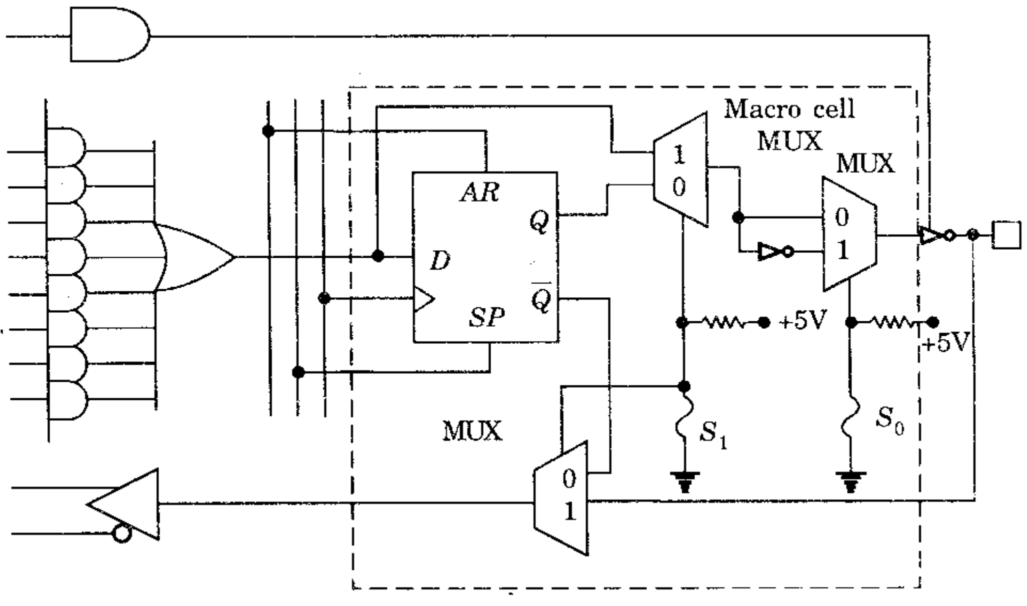


Figure 4: AMD 22V10 Output Macrocell

### Programming PAL

Though some engineers programmed PAL devices by manually editing files containing the binary fuse pattern data, most opted to design their logic using a Hardware Description Language (Verilog or VHDL) such as Data I/O's ABEL, Logical Devices' CUPL,or MMI's PALASM. These were computer-assisted design(CAD) (now referred to as "design automation") programs which translated (or "compiled") the designers' logic equations into binary fuse map files used to program (and often test) each device.

**Illustration of implementing a boolean function using PAL**

The device shown in the figure has 4 inputs and 4 outputs. Each input has a

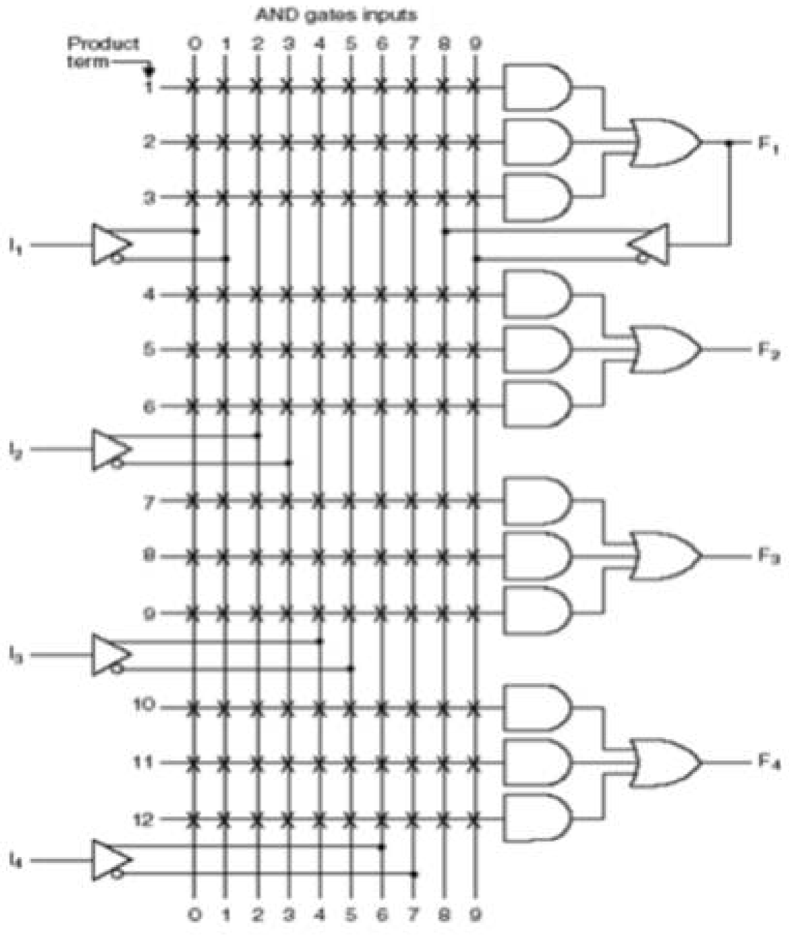


Figure 5: An Inside View Of A PAL Circuitry

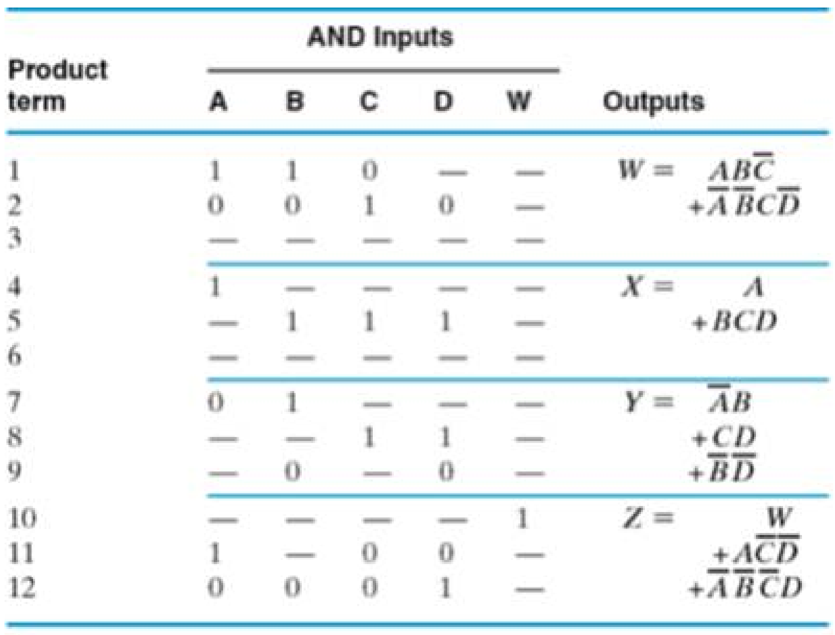
buffer-inverter gate, and each output is generated by a fixed OR gate. The device has 4 sections, each composed of a 3-wide AND-OR array, meaning that there are 3 programmable AND gates in each section.

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 F1 is connected to a buffer-inverter gate and is fed back into

the inputs of the AND gates through programmed connections.
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:
$W(A,B,C,D) = \sum m(2,12,13)$
$X(A,B,C,D) = \sum m(7,8,9,10,11,12,13,14,15)$
$Y(A,B,C,D) = \sum m(0,2,3,4,5,6,7,8,10,11,15)$
$Z(A,B,C,D) = \sum m(1,2,8,12,13)$

Simplifying the 4 functions to a minimum number of terms results in the following Boolean functions(By Means of Karnaugh map):
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
**Now,by analysis procedure we can reduce** Z to 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. The figure below shows the connection map for the PAL device, as specified in the programming table.



| 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 | — | $+\overline{A}\,\overline{B}\,\overline{C}D$ |

Figure 6: Connection Map For PAL Device

Since both W and X have two product terms, third AND gate is not used(the figure below). 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

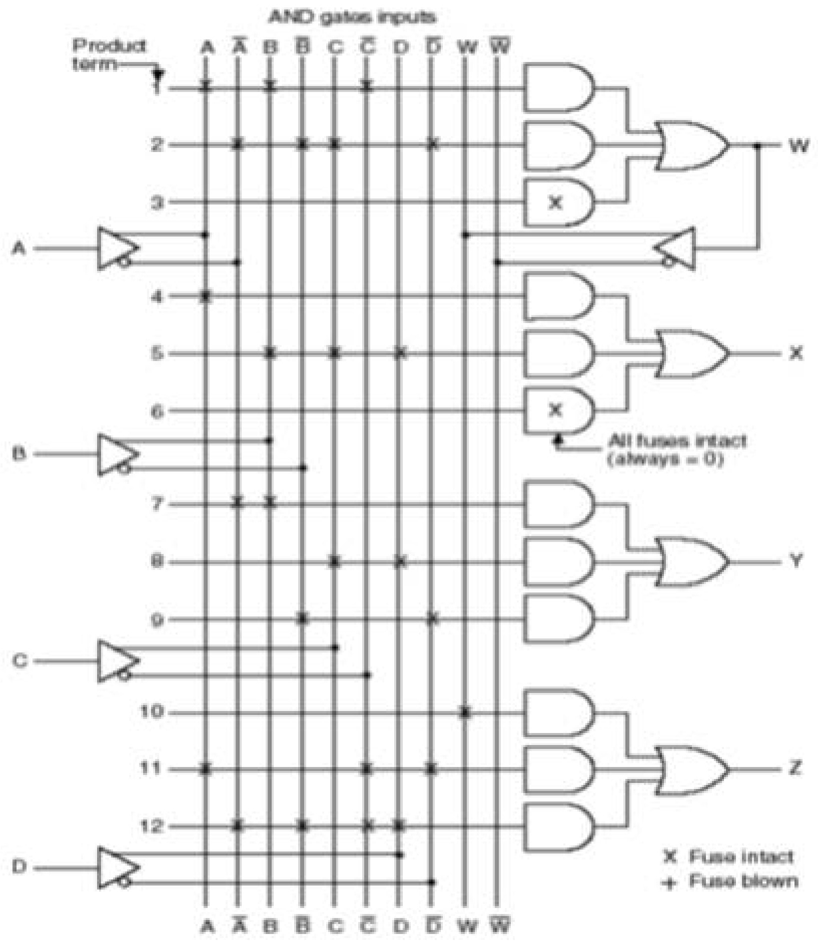


Figure 7: Programming The PAL Device Using Fuse Technology

In the initial stage, all fuses are intact(at the time of manufacture,when it is not configured).Now,fuses are blown accordingly to impement the boolean function with it.

**PAL- Extra circuitry: Macrocell**
To provide additional flexibility in PAL, extra circuitry is added at the output of

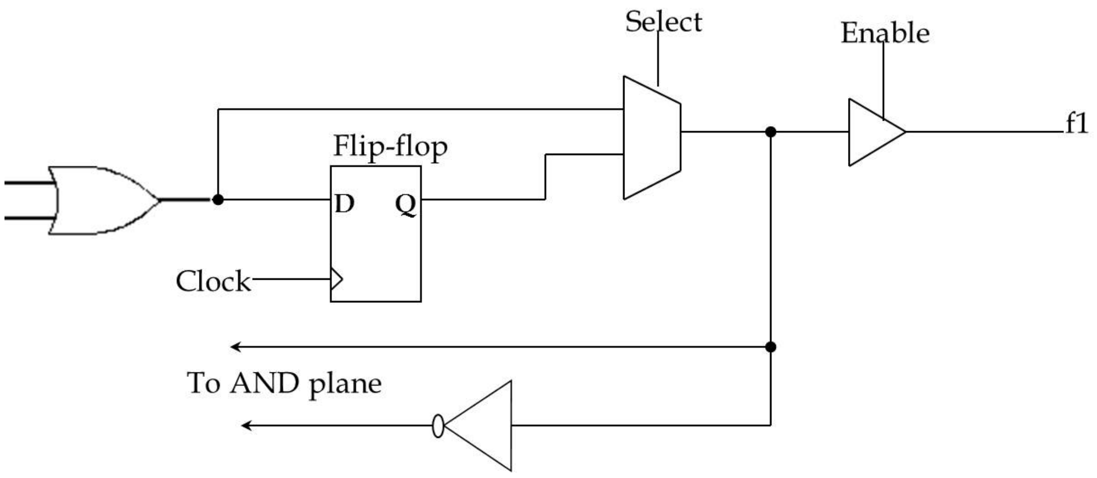


Figure 8: Macrocell In PAL

each OR gate. This is also referred to macrocell.

**Note:-**Macrocell Arrays are an approach to the design and manufacture of ASICs.

**Advantages of Using PAL**

- Less board space
- Fewer printed circuit board
- Smaller enclosures
- Lower power requirements(i.e. smaller power supplies)
- Faster and less costly assembly processes
- Higher reliabilty(fewer ICs and circuit connections=¿easier troubleshooting)
- Availability of design software
- Increase in speed
- Better security(copying is less likely to take place)
- Low production cost as compare to PLA

- More flexibility to designer
- Modification can be carried out within a short span of time
- Implementation of combinational and sequential circuits can be done with the help of PAL.
- For given internal complexity, a PAL can have larger inputs and implement a number of functions.

### 3.4.3 Programmable Logic Array(PLA)

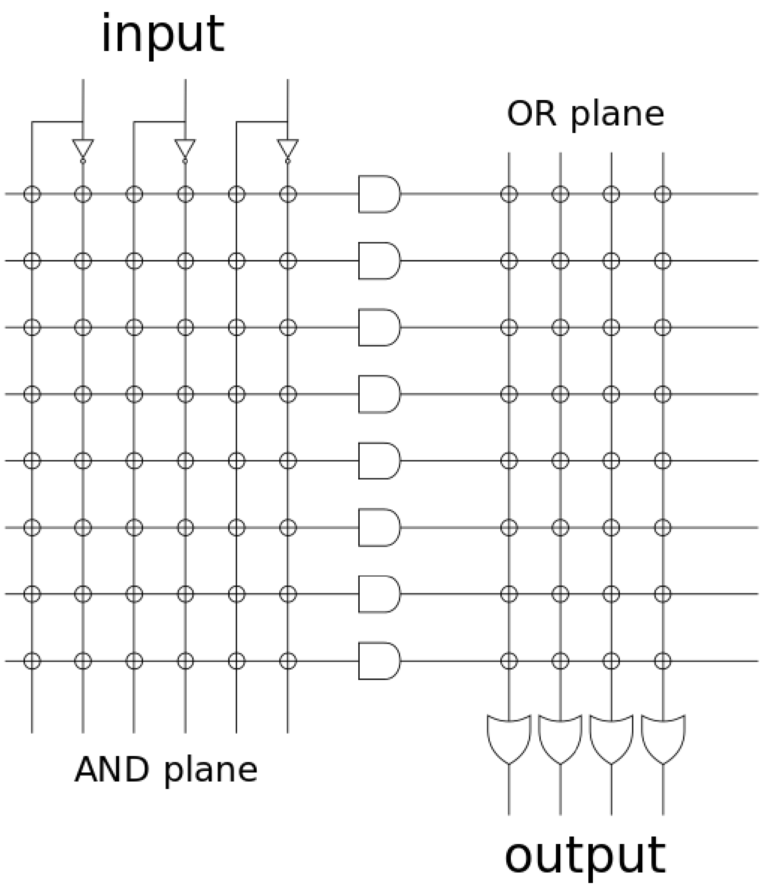


Figure 9: PLA schematic example

A programmable Logic Array (PLA) is a kind of Programmable Logic Device used to implement combinational logic circuits. The PLA has a set of programmable AND gate planes, which link to a set of programmable OR gate planes,

which can then be conditionally complemented to produce an output. This layout allows for a large number of logic functions to be synthesized in the sum of products (and sometimes product of sums) canonical forms.
PLA's differ from Programmable Array Logic devices (PALs and GALs) in that both the AND and OR gate planes are programmable.

**Implementation Procedure of a Boolean function using PAL**

⇒ Prepare the truth table

⇒ Write the Boolean expression in SOP (sum of products) form.

⇒ Obtain the minimum SOP form to reduce the number of product terms to a minimum.

⇒ Decide the input connection of the AND matrix for generating the required product term.

⇒ Then decide the input connections of OR matrix to generate the sum terms. Decide the connections of invert matrix.

⇒ Program the PLA.

In PLAs, instead of using a decoder as in PROMs, a number (k) of AND gates is used where k<2n, (n is the number of inputs). This is another architectural difference between PROM and PLA.
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.
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.

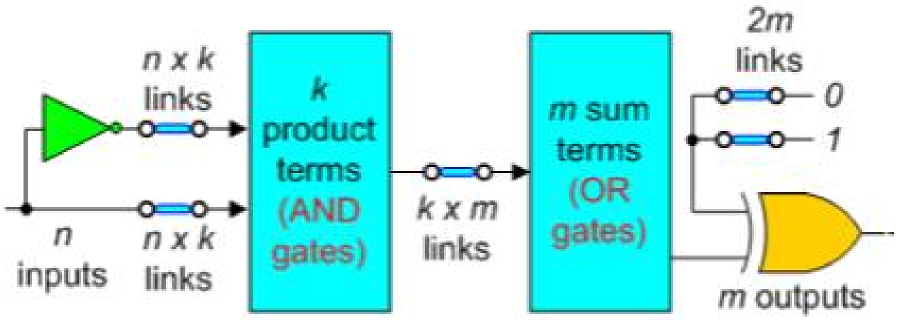


Figure 10: Block Diagram Of PLA

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 x k + k x 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).

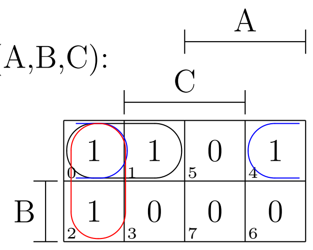
**Example:**

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 |

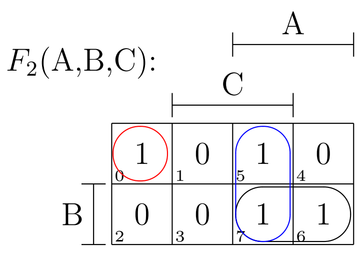
Each product term in the expression requires an AND gate. To minimize the cost, it is necessary to simplify the function to a minimum number of product terms.

$F_1$(A,B,C):



What we obtain from this Karnaugh map, is that $F_1$ =A'B'+A'C'+B'C'(by combining al the 1's in the map.

$F_1$' =AB+BC+AC(combining all the 0's n the map).



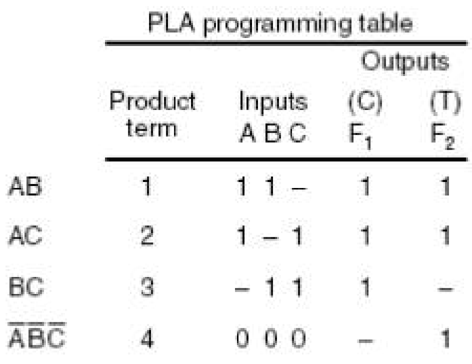
we obtain, $F_2$=AB+AC+A'B'C'(combining all the 1's in map)
and $F_2$'=A'C+A'B+AB'C'(combining all the 0's in map)

The combination that gives a minimum number of product terms is:

$F_1$'=AB+BC+AC or $F_1$=(AB+BC+AC)' & $F_2$ = 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) $F_1$ | (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 |

Figure 11: PLA Programming Table

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.

**Advantages of using PLA over Read Only Memory**

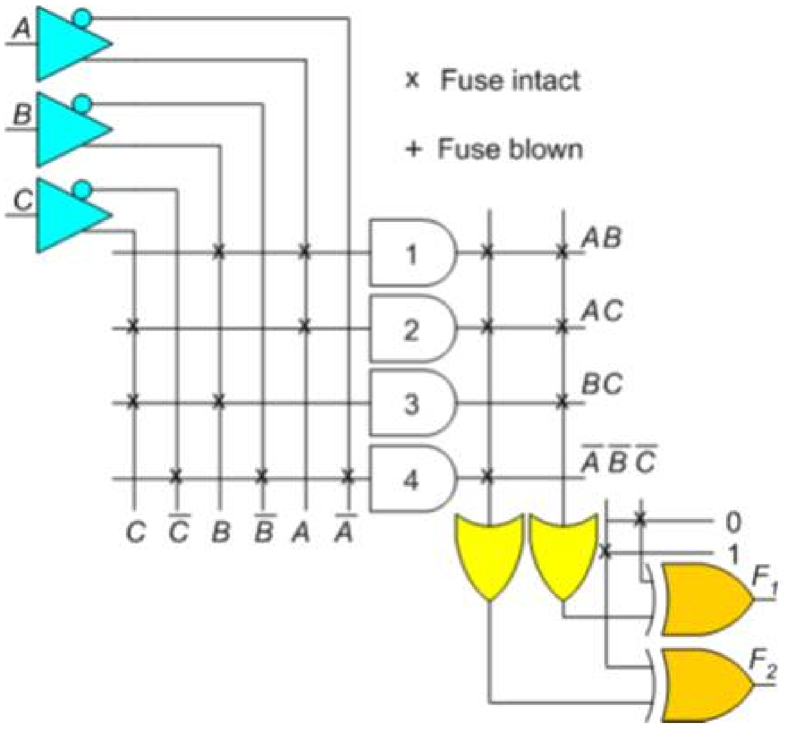


Figure 12: Condition of PLA After The Function Is Implemented

The desired outputs for each combination of inputs could be programmed into a Read-Only Memory, with the inputs being loaded onto the address bus and the outputs being read out as data. However, that would require a separate memory location for every possible combination of inputs, including combinations that are never supposed to occur, and also duplicating data for ”don’t care” conditions (for example, logic like ”if input A is 1, then, as far as output X is concerned, we don’t care what input B is”: in a ROM this would have to be written out twice, once for each possible value of B, and as more ”don’t care” inputs are added, the duplication grows exponentially); therefore, a Programmable Logic Array can often implement a piece of logic using fewer transistors than the equivalent in Read-Only Memory. This is particularly valuable when it is part of a processing chip where transistors are scarce (for example, the original 6502 chip contained a PLA to direct various operations of the processor).

**Applications of PLA**

- One application of a PLA is to implement the control over a datapath. It defines various states in an instruction set, and produces the next state (by conditional branching). [e.g. if the machine is in state 2, and will go to state 4 if the instruction contains an immediate field; then the PLA should define the actions of the control in state 2, will set the next state to be 4 if the instruction contains an immediate field, and will define the actions of the control in state 4]. Programmable logic arrays should correspond to a state diagram for the system.

- Other commonly used programmable logic devices are PAL, CPLD and FPGA. Note that the use of the word "programmable" does not indicate that all PLAs are field-programmable; in fact many are mask-programmed during manufacture in the same manner as a mask ROM. This is particularly true of PLAs that are embedded in more complex and numerous integrated circuits such as microprocessors. PLAs that can be programmed after manufacture are called FPGA (Field-Programmable Gate Array), or less frequently FPLA (Field-Programmable Logic Array)..

- The Commodore 64 home computer released in 1982 used a "906114-01 PLA" to handle system signals.

## 4 Generic Array Logic

---

After PROM,PAL and PLA,GAL was invented.

⇒ The GAL was an improvement on the PAL because one device was able to take the place of many PAL devices or could even have functionality not covered by the original range.

⇒ **The main difference between GAL and the earlier Programmable Device** is that unlike earlier device which can only be configured once,it can be reprogrammable and erasable which make things easier for the ones who are working with it.

⇒ Lattice GALs combine CMOS and electrically erasable $E^2$ floating gate technology for a high-speed, low-power logic device.

⇒ A similar device called a PEEL (programmable electrically erasable logic) was introduced by the International CMOS Technology (ICT) corporation.

Upto this all PLDs can be categorized into SPLDs(Simple Programmable Logic Device) which only contain a hundred of logic gates.

## 5 Mask Programmable Gate Array

The highest capacity general purpose logic chips available today are the traditional gate arrays sometimes referred to as Mask-Programmable Gate Arrays (MPGAs). The most prevalent style of Mask Programmed Gate Array (MPGA) in current use is the Sea-Of-Gates (SOG) or Sea-Of-Transistors Architecture. The core of an SOG MPGA is composed of a continuous array of transistors in fixed positions, surrounded by I/O circuits and bonding pads. Wafers containing the core design are pre-fabricated up to the final metalization steps. These master or base wafers are stocked by the vendor until a customer design is received. Then, one or more custom masks are created to define the user's circuit with design specific metalization and contacts. The cost of gate arrays is lower than standard cell or full custom designs because:

1. A single "base wafer" design can be used for many different customer designs,
2. Only 2-5 masks are required (compared to >14 for full custom or standard cell),
3. Automated Placement and Routing tools keep design costs low,
4. Use of standard bonding patterns and packages keeps packaging costs low,

5. Design turn-around time is short because only top metalization steps are run, and
6. Use of common test fixtures and programs keeps testing costs low.

MPGAs are available with varying core array sizes, package pin counts, and fabrication technologies. 24 Package pin counts range from 64 to 752, and manufacturers claim maximum usable gate counts up to 1,750,000. Table 10 indicate the range of vendors and options available for MPGAs in Bipolar/GaAs technologies.The propagation delay per gate is up to an order of magnitude faster than in CMOS gate arrays.
Mask Programmable Logic Devices are interconnection between Simple Programmable Logic Devices and Large Programmable Logic Devices.
typical SPLD contains say 100 of logic gates.For bigger logic circuits, There are two approaches. CPLD and FPGA.

## 6 Concept of Simple PLDs and Large PLDs :Introduction to CPLD and FPGA

---

**Simple PLDs**

**Single AND_OR plane**

It is configured by programming the AND and OR plane, or may be the Flip Flop inclusion and feedback selection
Usually has less than 32 I/O
They are available in DIP (Dual in line package), PLCC (Plastic Lead Chip Carrier up to 100 pins. Usually less than 100 equivalent gates.

**Large PLDs**

**Multiple AND_OR Plane**

Extend the concept of the simple PLDs further by incorporating architectures that contain several multiple logic block PAL models.There are two approaches for it. CPLD and FPGA.

- **CPLD**
  CPLD use huge no. of PAL like logic blocks.
  It has two level of programming. Each PAL like PLD blocks can be programmed as well as interconnection between them can be ptogrammed.
  Can accommodate from 1000 to 10,000 equivalent gates.
  Are available in PLCC and QFP (Quad Flap Pack) up to 200 pins.

- **FPGA**
  Whereas an FPGA has 3 level of programmings. Programmable Logic bocks, Programmable interconnects and Programmable Input/Output blocks.
  can acomodate upto 2-3 billion of logic gates.

## 6.1 Comlex Programmable Logic Device(CPLD)

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.Those PLD blocks are PAL like blocks.Commercial CPLDs range in size from only 2 PAL-like blocks to more than 100 PALlike blocks. They are available in a variety of packages.
It can be reconfigured after used once(like GAL,technologies from now on, can be reprogrammed which is additional feature for them).The devices are programmed using programmable elements that, depending on the technology of the manufacturer, can be EPROM cells, EEPROM cells, or Flash EPROM cells.

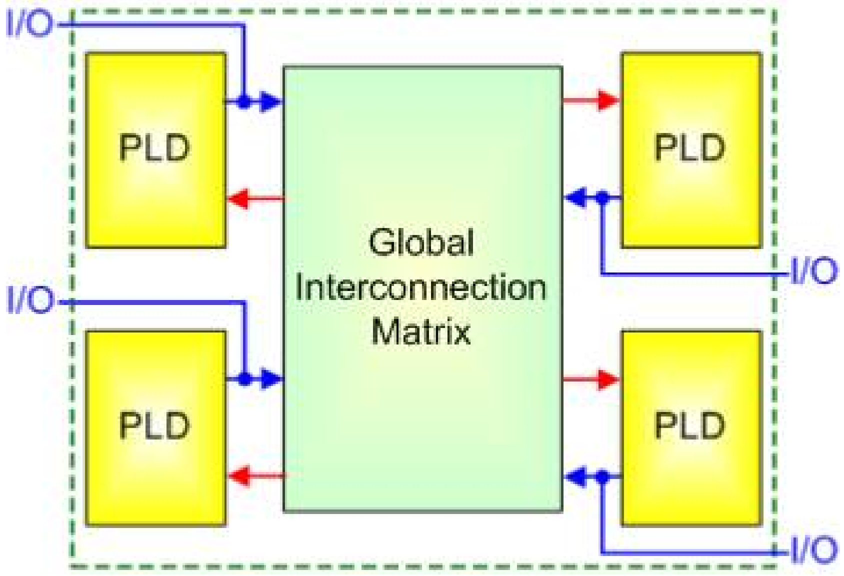
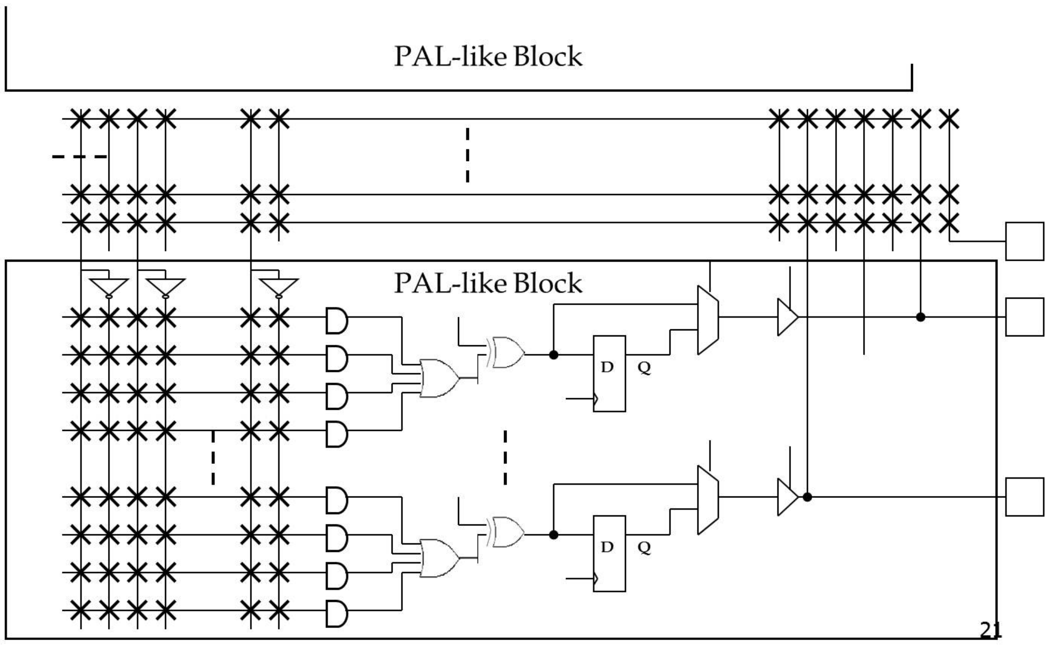


Figure 13: Block Diagram Of CPLD



A section of Configurabe Logic Block of CPLD

### 6.1.1 Architecture:

⇒ **CPLD functional blocks(CLBs or Configurable Logic Blocks):-** Those are also called macrocells.The AND plane of CPLD can accept inputs from the I/O blocks, other function blocks, or feedback from the same function block. The terms and then ORed together using a fixed number of OR gates, and terms are selected via a large multiplexer. The outputs of the mux can then be sent straight out of the block, or through a clocked flip-flop. This particular block includes additional logic such as a selectable exclusive OR and a master reset signal, in addition to being able to program the polarity at different stages. Usually, the function blocks are designed to be similar to existing PAL architectures, such as the 22V10, so that the designer can use familiar tools or even older designs without changing them.

⇒ **I/O Blocks:-**The I/O block of a CPLD is used to drive signals to the pins of the CPLD device at the appropriate voltage levels with the appropriate current. Usually, a flip-flop is included, as shown in the figure. This is done on outputs so that clocked signals can be output directly to the pins without encountering significant delay. It is done for inputs so that there is not much delay on a signal before reaching a flip-flop which would increase the device hold time requirement. Also, some small amount of logic is included in the I/O block simply to add some more resources to the device.

⇒ **Interconnect:-**The CPLD interconnect is a very large programmable switch matrix that allows signals from all parts of the device go to all other parts of the device. While no switch can connect all internal function blocks to all other function blocks, there is enough flexibility to allow many combinations of connections.

⇒ **Programmable Elements:-**Different manufacturers use different technologies to implement the programmable elements of a CPLD. The common technologies are Electrically Programmable Read Only Memory (EPROM), Electrically Erasable PROM (EEPROM) and Flash EPROM. These technologies are similar to, or next generation versions of, the technologies that were used for the simplest programmable devices, PROMs.

### 6.1.2 Common Features with PAL

- Non-volatile configuration memory. Unlike many FPGAs, an external configuration ROM isn't required, and the CPLD can function immediately on system start-up.

- For many legacy CPLD devices, routing constrains most logic blocks to have input and output signals connected to external pins, reducing opportunities for internal state storage and deeply layered logic. This is usually not a factor for larger CPLDs and newer CPLD product families.

### 6.1.3 Common Features with FPGA

- Large number of gates available. CPLDs typically have the equivalent of thousands to tens of thousands of logic gates, allowing implementation of moderately complicated data processing devices. PALs typically have a few hundred gate equivalents at most, while FPGAs typically range from tens of thousands to several million.

- Some provisions for logic more flexible than sum-of-product expressions, including complicated feedback paths between macro cells, and specialized logic for implementing various commonly used functions, such as integer arithmetic.

The most noticeable difference between a large CPLD and a small FPGA is the presence of on-chip non-volatile memory in the CPLD. The characteristic of non-volatility makes the CPLD devices used in modern digital designs for performing "boot loader" functions before handing over control to other devices not having this capability. A good example is where a CPLD is used to load configuration data for an FPGA from non-volatile memory.

### 6.1.4 CPLD Architectural Issues

When considering a CPLD for use in a design, the following issues should be taken into account:

1. The programming technology:-
   EPROM, EEPROM, or Flash EPROM? This will determine the equipment needed to program the devices and whether they came be programmed only once or many times.
2. The function block capability
   - How many function blocks are there in the device?
   - How many product and sum terms can be used?
   - What are the minimum and maximum delays through the logic?
   - What additional logic resources are there such as XNORs, ALUs, etc.?
   - What kind of register controls are available (e.g., clock enable, reset, preset, polarity control)? How many are local inputs to the function block and how many are global, chipwide inputs?
   - What kind of clock drivers are in the device and what is the worst case skew of the clock signal on the chip. This will help determine the maximum frequency at which the device can run.
3. The I/O capability
   - How many I/O are independent, used for any function, and Introduction to FPGA Design
   - How many are dedicated for clock input, master reset, etc.?
   - What is the output drive capability in terms of voltage levels and current?
   - What kind of logic is included in an I/O block that can be used to increase the functionality of the design?

### 6.1.5 CPLD Programming

CPLD uses quad flat pack (QFP) type of package. QFP package has pins on all four sides and the pins extend outward from the package with a downward-curving shape. Moreover, QFP pins are much thinner and hence, they support a larger number of pins when compared to the PLCC packing.

Most CPLDs contain the same type of switch as in PLDs. Here, a separate

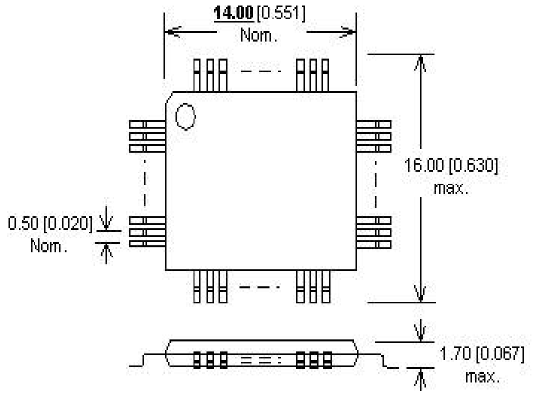


Figure 14: Block diagram of a QFP

programming unit is not used due to two main reasons. Firstly, CLPDs contain 200 + pins on the package, and these pins are often fragile and easily bent. Secondly, a socket would be required to hold the chip. Sockets are usually quite expensive and hence, add to the overall cost incurred.
CLPD usually support the ISP technique. A small connector is included on the PCB and is connected to a computer system. CLPD is programmed by transferring the programming information from the CAD tool to into the CLPD. The circuitry on the CLPD that allows this type of programming is called JTAG, Joint Test Action Group port, and is standardized by the IEEE.
JTAG is a non-volatile type of programming i.e programmed state is retained permanently (for example, in case of power failure, CLPD retains the program).
The programming design of CPLD is first coded in Hardware Description Language (Verilog or VHDL), once the code is validated (simulated and synthesized). During synthesis the target device(CPLD model) is selected, and a technology-mapped net list is generated. The net list can then be fitted to the actual CPLD architecture using a process called place-and-route, usually performed by the CPLD company's proprietary place-and-route software. Then the user will do some verification processes. If every thing is fine, he will use the CPLD, else he will reconfigure it.

### 6.1.6 Example CPLD Families

Some CPLD families from different vendors are listed below:

- Altera MAX 7000 and MAX 9000 families
- Atmel ATF and ATV families
- Lattice ispLSI family
- Lattice (Vantis) MACH family
- Xilinx XC9500 family

### 6.1.7 Application of CPLD

⇒ Their high speeds and wide range of capacities make CPLDs useful for many applications, from implementing random glue logic to prototyping small gate arrays. An important reason for the growth of the CPLD market is the conversion of designs that consist of multiple SPLDs into a smaller number of CPLDs.

⇒ CPLDs can realize complex designs such as graphics, LAN, and cache controllers. As a rule of thumb, circuits that can exploit wide AND/OR gates and do not need a large number of flip-flops are good candidates for CPLD implementation. Finite state machines are an excellent example of this class of circuits. A significant advantage of CPLDs is that they allow simple design changes through reprogramming (all commercial CPLD products are reprogrammable). In-system programmable CPLDs even make it possible to reconfigure hardware (for example, change a protocol for a communications circuit) without powering down.

⇒ Designs often partition naturally into the SPLD-like blocks in a CPLD, producing more predictable speed performance than a design split into many small pieces mapped into different areas of the chip. Predictability of circuit implementation is one of the strongest advantages of CPLD architectures.

## 6.2 Field Programmable Gate Array

A field-programmable gate array (FPGA) is an integrated circuit designed to be configured by a customer or a designer after manufacturing—hence "field-programmable".
FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare).
Contemporary FPGAs have large resources of logic gates and RAM blocks to implement complex digital computations. As FPGA designs employ very fast I/Os and bidirectional data buses it becomes a challenge to verify correct timing of valid data within setup time and hold time. Floor planning enables resources allocation within FPGA to meet these time constraints. FPGAs can be used to implement any logical function that an ASIC could perform. The ability to update the functionality after shipping, partial re-configuration of a portion of the design and the low non-recurring engineering costs relative to an ASIC design (notwithstanding the generally higher unit cost), offer advantages for many applications.
FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together"—somewhat like many (changeable) logic gates that can be inter-wired in (many) different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory
Some FPGAs have analog features in addition to digital functions. The most common analog feature is programmable slew rate and drive strength on each output pin, allowing the engineer to set slow rates on lightly loaded pins that would otherwise ring unacceptably, and to set stronger, faster rates on heavily loaded pins on high-speed channels that would otherwise run too slowly.[4][5] Another relatively common analog feature is differential comparators on input pins designed to be connected to differential signaling channels. A few "mixed signal FPGAs" have integrated peripheral analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) with analog signal conditioning blocks allowing them to operate as a system-on-a-chip.[6] Such devices blur the line between an FPGA, which carries digital ones and zeros on its internal programmable interconnect fabric, and field-programmable analog array (FPAA), which carries analog values on its internal programmable interconnect fabric.

### 6.2.1 FPGAs Classifications

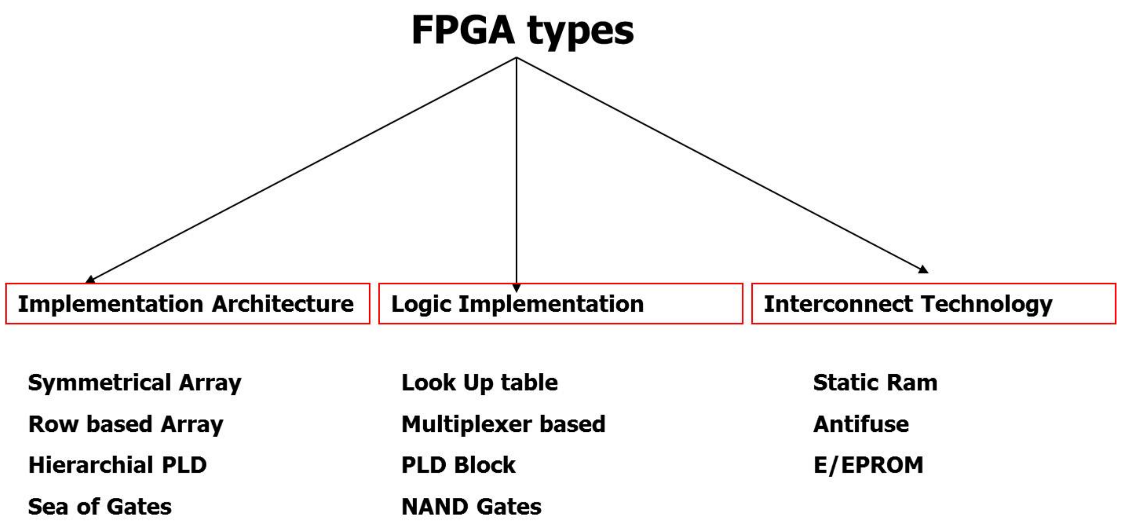


Figure 15: Classifications Of FPGA According To Different Criterias

### 6.2.2 FPGA Architecture in general

The FPGA consists of 3 main structures:

1. Programmable logic structure,
2. Programmable routing structure, and
3. Programmable Input/Output (I/O). Though there are some additional structure:
4. Clock circuitry

⇒ **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.

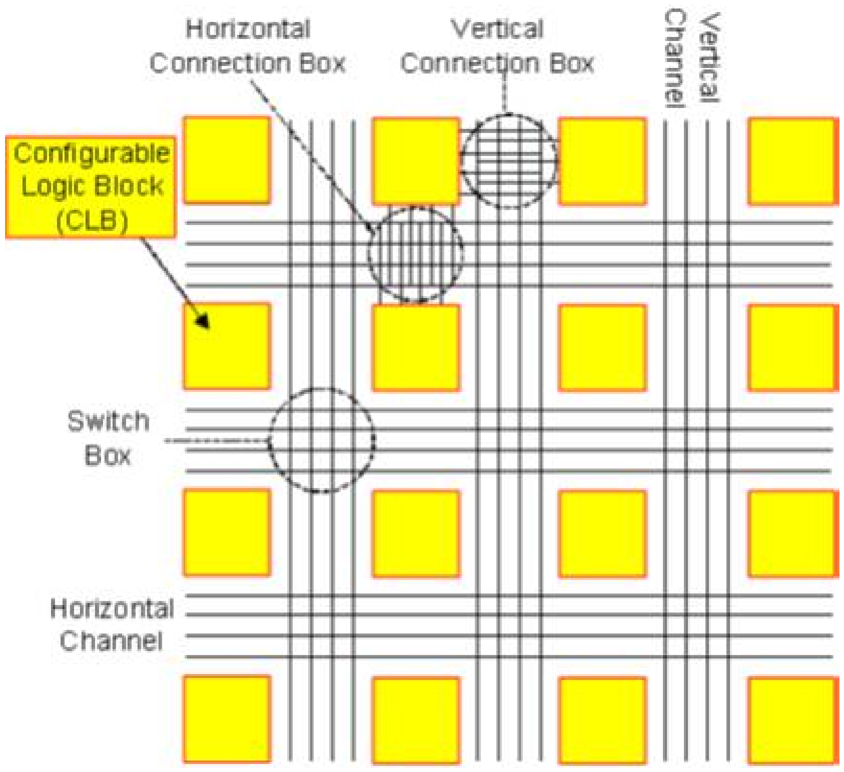


Figure 16: Programmable Logic Structure

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.The Logic bolcks(or configurable logic blocks of an FPGA can be implemented by any of the following:-

- Transistor pairs
- combinational gates like basic NAND gates or XOR gates
- n-input Lookup tables
- Multiplexers
- Wide fan-in And-OR structure

According to the arrangement of logic blocks FPGAs(or according to the implementation architecture we can say) FPGAs can be classified into four types.

- Symmetrical Array
- Row based Array
- Hierarchical PLD
- Sea of Gates

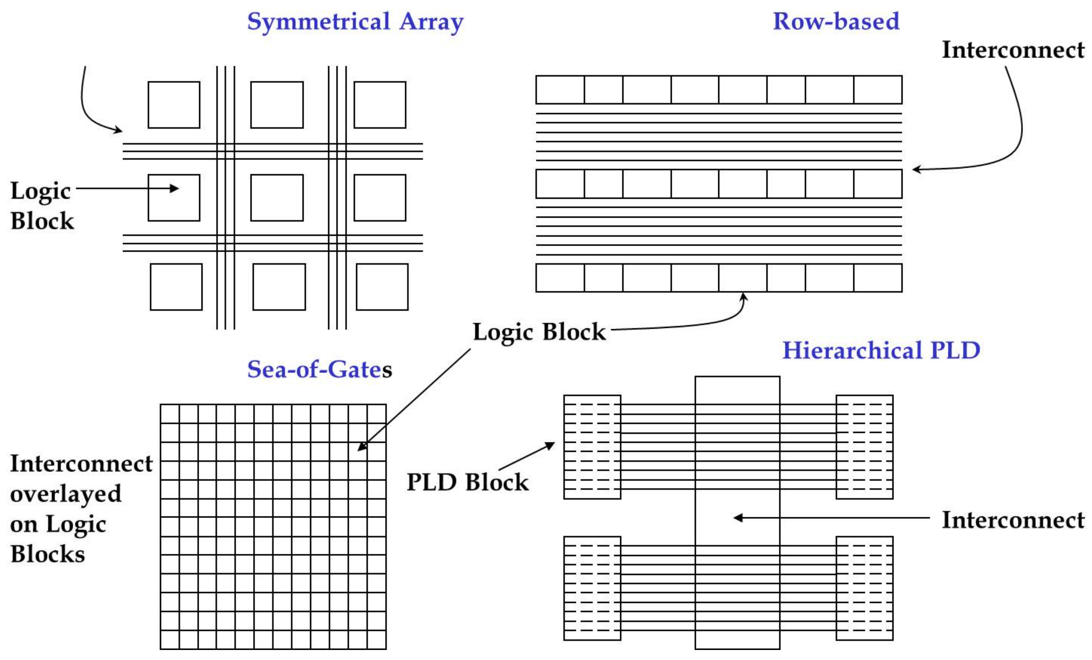


Figure 17: Classification of FPGAs According To Implementation Architecture

- **Symmetrical arrays:** This architecture consists of logic elements (called CLBs) arranged in rows and columns of a matrix and interconnect laid out between them. This symmetrical matrix is surrounded by I/O blocks which connect it to outside world. Each CLB consists of n-input Lookup table and a pair of programmable flip flops. I/O blocks also control functions such as tri-state control, output transition speed. Interconnects provide routing path. Direct interconnects between adjacent logic elements have smaller delay compared to general purpose interconnect.
  **Symmetrical Array FPGA Example:** Xilinx
- **Row based architecture:** Row based architecture consists of alternating rows of logic modules and programmable interconnect tracks. Input output blocks is located in the periphery of the rows. One row may be connected to adjacent rows via vertical interconnect. Logic

modules can be implemented in various combinations. Combinatorial modules contain only combinational elements which Sequential modules contain both combinational elements along with flip flops. This sequential module can implement complex combinatorial-sequential functions. Routing tracks are divided into smaller segments connected by anti-fuse elements between them.
**Row Based FPGA Example:** Actel

- **Hierarchical PLDs** This architecture is designed in hierarchical manner with top level containing only logic blocks and interconnects. Each logic block contains number of logic modules. And each logic module has combinatorial as well as sequential functional elements. Each of these functional elements is controlled by the programmed memory. Communication between logic blocks is achieved by programmable interconnect arrays. Input output blocks surround this scheme of logic blocks and interconnects. **Hierarchical PLD Example:** Altera
- **Sea Of Gates:** The basic building block of the sea of gates design is a configurable cell containing multiplexers and a function unit. the function unit is preceded by multiplexers which select the source for the inputs. The function unit is capable of generating any logic function of the two inputs, or of operating as a D-type latch.
**Sea of Gates FPGA Example:** Algotronix

**Now FPGAs can also be classified according to logic implementation. Like it is mentioned in the tree diagram of FPGA types.**
According to logic implementation,FPGAs can be classified into:-

- Look Up Table.
- Multiplexer based
- PLD block.
- NAND gates.

Now, this is correlated with the classification of FPGA according to architecture implementation and different models of FPGA.
**Look Up Table:**
A k input LUT can implement any Boolean function of k variables. The inputs are used as addresses that can retrieve the 2k by 1-bit memory that stores the truth table of the Boolean function. Since the size of the memory increases with the number of inputs, k, in order to optimize this mapping and reduce the size of the memory there are a variety of algorithms that

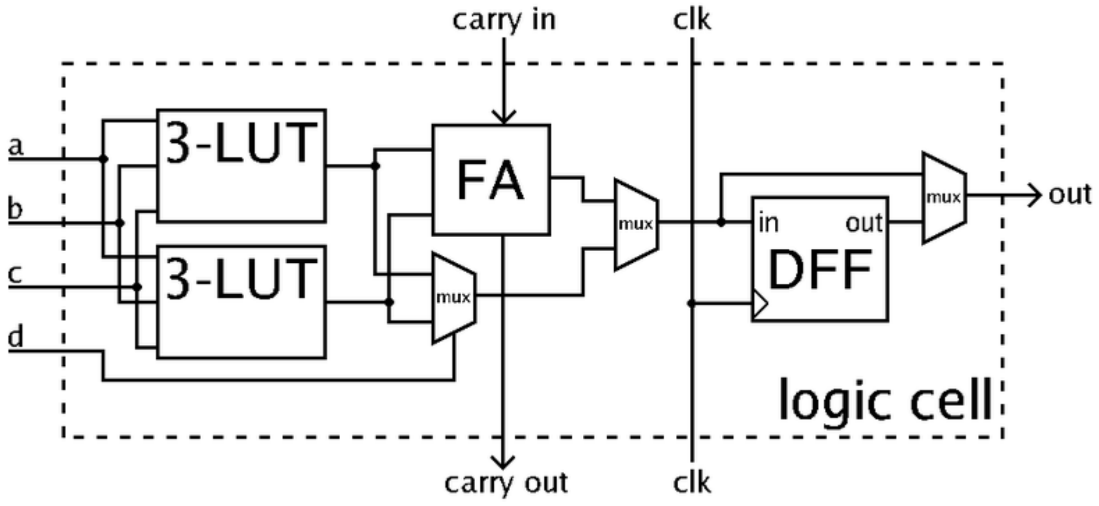


Figure 18: Simplified Example Illustration Of A Logic Cell.

map a Boolean network, from a given equation, into a circuit of k-input LUT. These algorithms minimize either the total number of LUTs or the number of levels of LUTs in the final circuit. Minimizing the total number of LUTs reduces the CLB requirements while minimizing the levels of LUTs improves the delay.

n general, a logic block (CLB or LAB) consists of a few logical cells (called ALM, LE, Slice etc.). A typical cell consists of a 4-input LUT, a Full adder (FA) and a D-type flip-flop, as shown below. The LUTs are in this figure split into two 3-input LUTs. In normal mode those are combined into a 4-input LUT through the left mux. In arithmetic mode, their outputs are fed to the FA. The selection of mode is programmed into the middle multiplexer. The output can be either synchronous or asynchronous, depending on the programming of the mux to the right, in the figure example. In practice, entire or parts of the FA are put as functions into the LUTs in order to save space.

ALMs and Slices usually contains 2 or 4 structures similar to the example figure, with some shared signals.

CLBs/LABs typically contains a few ALMs/LEs/Slices.

In recent years, manufacturers have started moving to 6-input LUTs in their high performance parts, claiming increased performance.

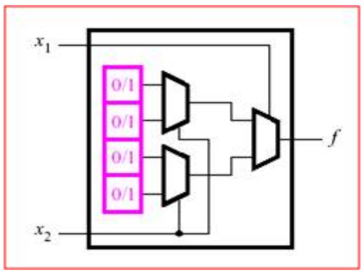
**Understanding of LUT**

f1= x1'x2' + x1 x2

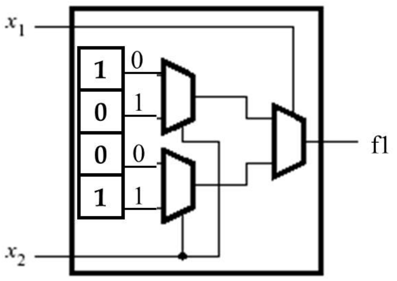
Function to be implemented with LUT

Now, clearly it is a 2 variable boolean function. so it will be implemented by 2 input LUT.

| $X_1$ | $X_2$ | $f_1$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



(a) 2-Input LUT Before Programming



(b) 2-Input LUT After Programming

Figure 19: Condition Of LUT Before and After Programming

**Note:-**Storage cells are actually latches. which can hold the value of 0 or 1 according to the programming.

**FPGA logic blocks can be multiplexer based like we discussed before**. We already know the concept of MUX very well. So, no need of further discussion about it.

Example of a mux based logic block configuration:-**Actel logic block**

**FPGA logic blocks can be PLD logic blocks too.**

Example of PLD logic blocks:**Altera Logic block.**

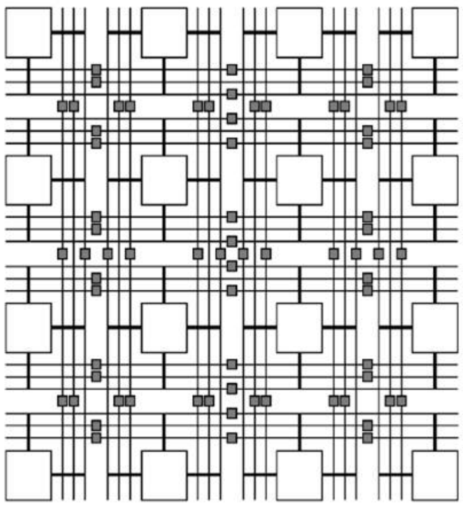
Altera's logic block has evolved from earlier PLDs. It consists of wide fan in (up to 100 input) AND gates feeding into an OR gate with 3-8 inputs. The advantage of large fan in AND gate based implementation is that few logic blocks can implement the entire functionality thereby reducing the amount of area required by interconnects. On the other hand disadvantage is the low density usage of logic blocks in a design that requires fewer input logic. Another disadvantage is the use of pull up devices (AND gates) that consume static power. To improve power manufacturers provide low power consuming logic blocks at the expense of delay. Such logic blocks have gates with high threshold as a result they consume less power. Such logic blocks can be used in non-critical paths.

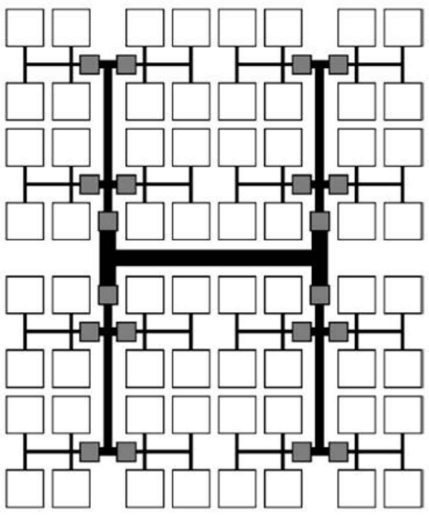
**FPGA logic blocks can be NAND gate based too**

Example of NAND gate constructed FPGA logic blocks:-**Plessey FPGA.**

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

- 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.
- 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.
- 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.


Segmented Routing


Hierarchical routing

So far, we described ”Segmented Routing”. But, routing can be hiearchical too.
**Difference between two types of routing:**

– **Segmented Routing**
  Short wires accommodate local traffic.
  Short wires can be connected together using switch boxes to emulate longer wires.
  Also contain long wires to allow efficient communication without passing through switches

– **Hiearchical Routing**
  Routing within a “group” of logic blocks occur at the local level.
  Longer hierarchical wires connect different groups.

⇒ **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.

⇒ **Clock Circuitry:**
Special I/O blocks with special high drive clock buffers, known as clock drivers, are distributed around the chip. These buffers are connect to clock input pads and drive the clock signals onto the global clock lines described above. These clock lines are designed for low skew times and fast propagation times. As we will discuss later, synchronous design is a must with FPGAs, since absolute skew and delay cannot be guaranteed. Only when using clock signals from clock buffers can the relative delays and skew times be guaranteed.

### 6.2.3 FPGA design and programming

To define the behavior of the FPGA, the user provides a Hardware Description Language (HDL) or a schematic design. The HDL form is more suited to work with large structures because it’s possible to just specify them numerically rather than having to draw every piece by hand. However, schematic entry can allow for easier visualisation of a design.

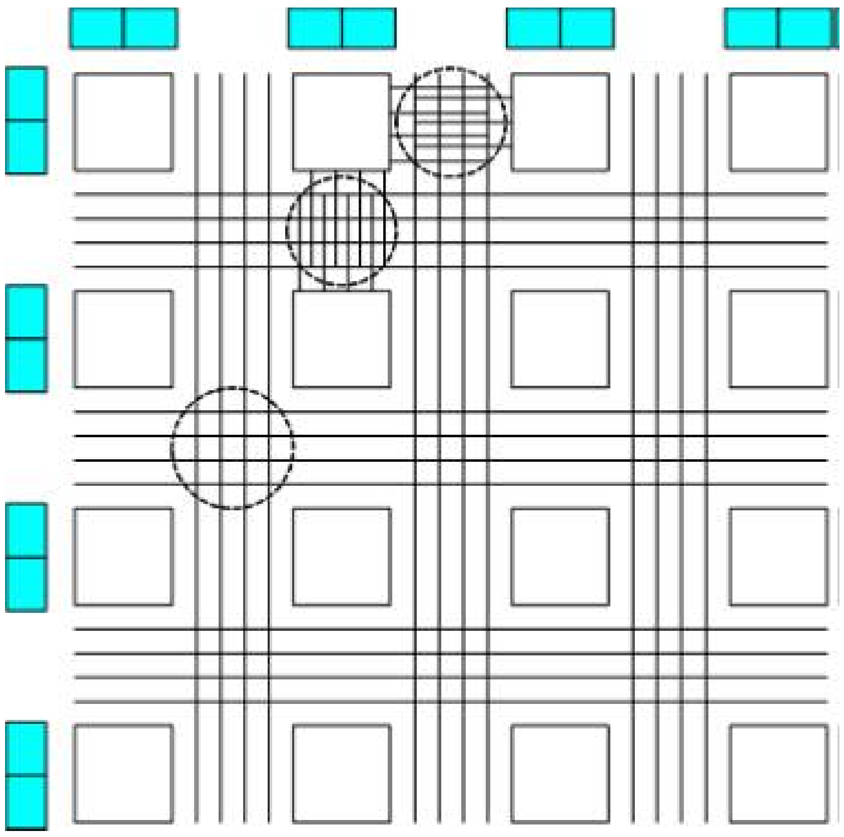


Figure 20: Programmable I/O

Then, using an electronic design automation tool, a technology-mapped netlist is generated. The netlist can then be fitted to the actual FPGA architecture using a process called place-and-route, usually performed by the FPGA company's proprietary place-and-route software. The user will validate the map, place and route results via timing analysis, simulation, and other verification methodologies. Once the design and validation process is complete, the binary file generated (also using the FPGA company's proprietary software) is used to (re)configure the FPGA. This file is transferred to the FPGA/CPLD via a serial interface (JTAG) or to an external memory device like an EEPROM.

The most common HDLs are VHDL and Verilog, although in an attempt to reduce the complexity of designing in HDLs, which have been compared to the equivalent of assembly languages, there are moves to raise the abstraction level through the introduction of alternative languages. National Instruments' LabVIEW graphical programming language (sometimes referred to as "G") has an FPGA add-in module available to target and program FPGA hardware.

To simplify the design of complex systems in FPGAs, there exist libraries of predefined complex functions and circuits that have been tested and optimized to speed up the design process. These predefined circuits are commonly called IP cores, and are available from FPGA vendors and third-party IP suppliers (rarely free, and typically released under proprietary licenses). Other predefined circuits are available from developer communities such as OpenCores (typically released under free and open source licenses such as the GPL, BSD or similar license), and other sources.

In a typical design flow, an FPGA application developer will simulate the design at multiple stages throughout the design process. Initially the RTL description in VHDL or Verilog is simulated by creating test benches to simulate the system and observe results. Then, after the synthesis engine has mapped the design to a netlist, the netlist is translated to a gate level description where simulation is repeated to confirm the synthesis proceeded without errors. Finally the design is laid out in the FPGA at which point propagation delays can be added and the simulation run again with these values back-annotated onto the netlist.
**The design flow of an FPGA (step by step) is provided below:**

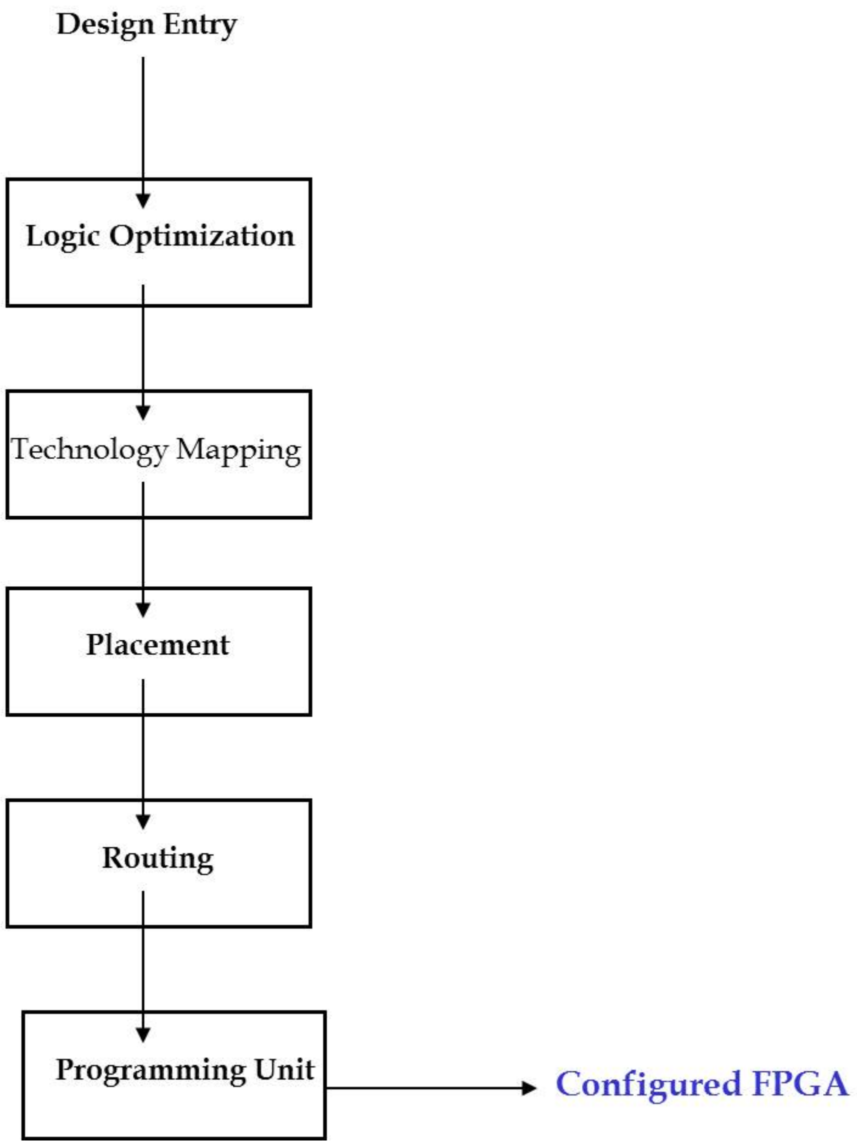


Figure 21: The Design Flow Of An FPGA

### 6.2.4 Architectural Resources

**Storage elements:**

- Like flip flops and latches.
- Gives Pipelined designs.

**RAM using Function Generator**

- Each CLB makes the memory look up tables in 'F' and 'G' function generators and can be used as an array of Read/Write memory cells depending on the selected Operation Mode.
- Operational modes are level-sensitive,edge-triggered and dual port edge-triggered.
- Depending upon the selected mode the CLB can be configured as either a $16 \times 2$,$16 \times 1$ or $32 \times 1$ bit array.

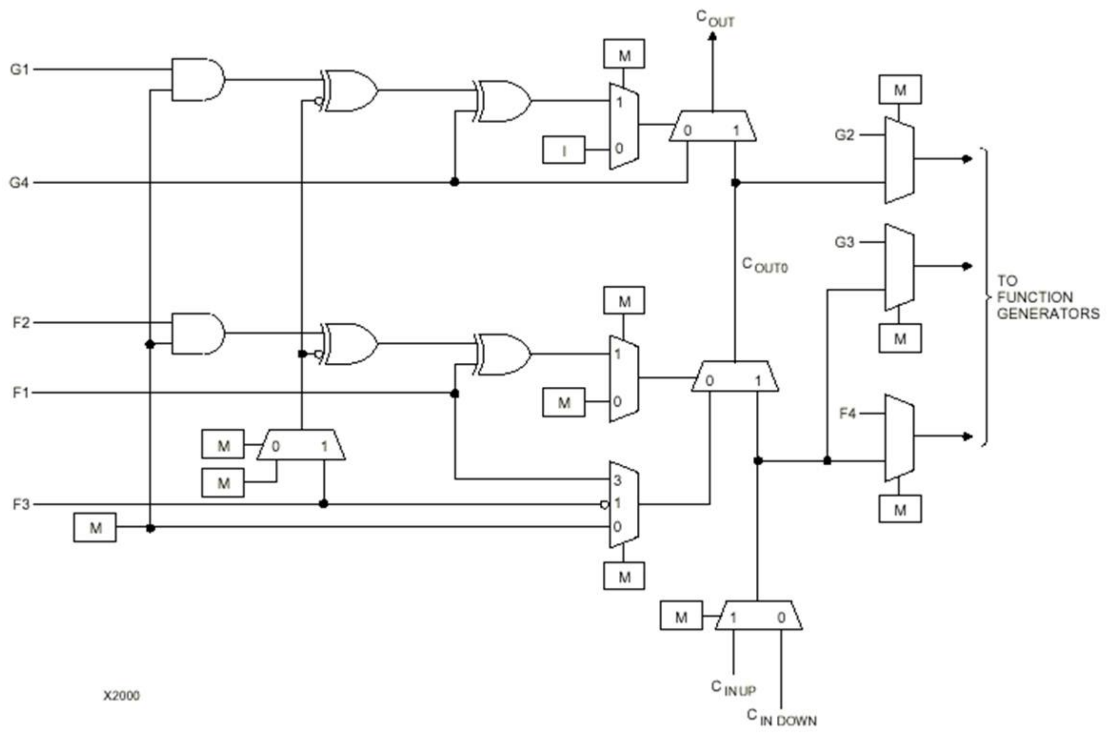
**First Array Logic** There is a dedicated arithmatic logic for the fast generation of carry and borrow signals in each CLB's F and G function generators.
Carry chain is used or first carry logicwhich isindependent of normal routing resources.
Between each CLB there are 8 permanent connections i.e carry chain.
This first carry logic greatly increases the efficiency and performance of adders,counters,subtractors,accumulators and comparators.

**Tri State buffers**

- A pair of 3-states buffers is associated with each CLB in the array.
- These can be used to implement multiplexed or bidirectional buses on the horizontal longlines to save other logic resources.
- Programmable pull-up resistors attached to these longlines helped to implement a wired-and logic function.
- Week Keeper avoids data connection while writing data and latches previous value.

**Wide-edge Decoders**

- Used to decode specific values from a larger no. of bits e.g address decoding of large microprocessors.
- These are seperate from CLB and do not use CLB resources.



The Circuit Diagram Of First Carry Logic

- These are implemented as a Wired-AND gate.
- Four Programmable Decoders are located on each edge of the device.
- Each row or column of CLBs provides up to three variables or their compliments.
- Each decoder generates a high o/p(Resistive Pull Up) when the AND gate condition is true.

**On-chip Oscillator**

- It is used to clock the power-on-time-out for configuration memory clearing and as the source of CCLK in master configuration modes.
- The counter runs at 8MHz nominal frequency which varies with VCC and temperature. The O/P freqency falls between 4 to 10 MHz.
- On chip clock is available to load the data from EPROM to configuration memory.

**Power Distribution in FPGA**

- Power is distributed through a grid to achieve high noise immunity and isolation between logic and I/O.
- Dedicated ground and Vcc ring surrounding the logic array provides power to I/O drivers.
- An independent matrix of VCC and ground supplies the internal logic of the device.
- Typically 0.1 microfarad capacitor is connected between VCC and ground for decoupling.

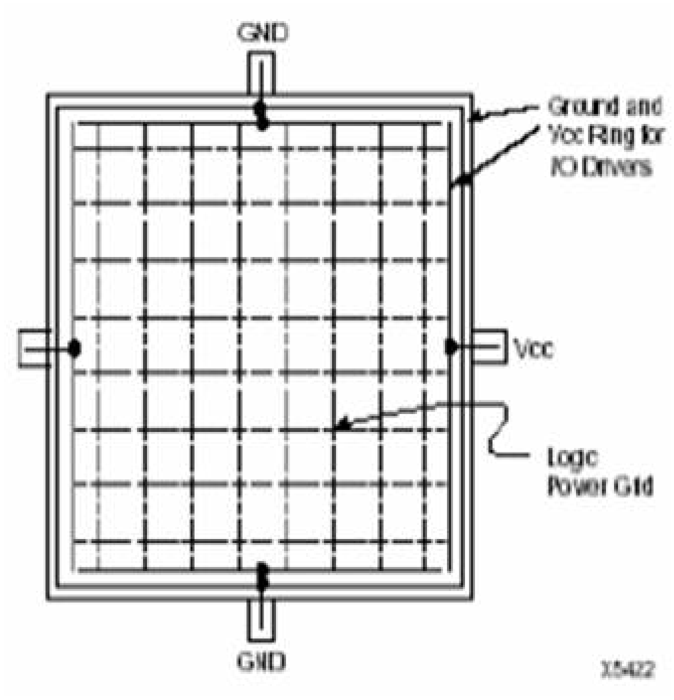


Figure 22: Power Distribution

### 6.2.5 Configuration of an FPGA

- **Configuration is a process of**
  - Loading a specific programming data to an FPGA.
  - To determine the functional operation of the internal blocks and their interconnects.

- **Special purpose pins or configuration:**
  - Dedicated pins.
  - Special function user I/O pins.
  - Unrestricted user programmable I/O pins.

**Dedicated pins**

- **CCLK:-**Acts as a configuration clock. Internal oscillator generates CCLK. Its selected as 1MHz(default) or 8MHz.
- **DONE:-**It's a bidirectional signal and indicates the completion of a configuration process.
- **PROGRAM:-**Clears FPGA configuration memory and initiates configura
- **VCC:-**Eight or more connections to nominal 5v supply.
- **GND:-**Eight or more connections to nominal 5v supply.

**Unrestricted user programmable I/O pins**

- Special function user I/O pins can be used as user programmable I/Os after configuration.
- All outputs not usedfor configuration process are tri-stated with a 50k-100k pull up resistor.
- After configuration if an IOB is unused then then it is configured as an input with a 50k-100k pull up resistor.

**Special function user I/O pins**

- $M_0$,$M_1$,$M_2$**:**Used for configuration mode selection.
- **INIT:**A bi-directional signal used to delay the configuration. A low on this pin indicates that configuration data error has occurred.
- **A0 to A17:**These pins address the configuration EPROM during master parallel configuration.
- **D0 to D7:**Recieves configuration data during Master parallel and peripheral mode.
- **DIN:**Serial configuration data input that recieves data on the rising edge of clock.

- **DOUT:**Serial configuration data output. In daisy chaining acts as a DIN for next FPGA in chain.
- **TDI,TDO,TMS,TCK:**Pins for boundary scanning or as I/O if not used

### 6.2.6 Applications of FPGA

- FPGAs have gained rapid acceptance over the past decade because users can apply them to a wide range of applications: random logic, integrating multiple SPLDs, device controllers, communication encoding and filtering, small- to medium-size systems with SRAM blocks, and many more.
  Another interesting FPGA application is prototyping designs to be implemented in gate arrays by using one or more large FPGAs. (A large FPGA corresponds to a small gate array in terms of capacity). Still another application is the emulation emulation of entire large hardware systems via the use of many interconnected FPGAs. QuickTurn4 and others have developed products consisting of the FPGAs and software necessary to partition and map circuits for hardware emulation.

- An application only beginning development is the use of FPGAs as custom computing machines. This involves using the programmable parts to execute software, rather than compiling the software for execution on a regular CPU. For information, we refer readers to the proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines, held for the last four years.

- As mentioned earlier, pieces of designs often map naturally to the SPLDlike blocks of CPLDs. However, designs mapped into an FPGA break up into logic-block-size pieces distributed through an area of the FPGA. Depending on the FPGA's interconnect structure, the logic block interconnections may produce delays. Thus, FPGA performance often depends more on how CAD tools map circuits into the chip than does CPLD performance.

## 7 User-programmable switch technologies

Now, we have to understand what is a User-programmable switch. In short, we can tell a user-programmable switch that can connect a logic element to an interconnect wire or one interconnect wire to another.

User-programmable switches are the key to user customization of FPDs. The first user-programmable switch developed was the fuse used in PLAs. Although some smaller devices still use fuses, we will not discuss them here because newer technology is quickly replacing them. For higher density devices, CMOS dominates the IC industry, and different approaches to implementing programmable switches are necessary. For CPLDs, the main switch technologies (in commercial products) are floating gate transistors like those used in EPROM (erasable programmable read-only memory) and EEPROM (electrically erasable PROM). For FPGAs, they are SRAM (static RAM) and antifuse. Below this, we provide the most important characteristics of these programming technologies.

## 7.1 EPROM-technology

Erasable Programmable Read-Only Memory
EPROM cells are non-volatile memory cells.Similar to the technology used in standard EPROM memory devices. EPROM cells are electrically programmed in a device programmer.EPROM-based devices are erasable using ultra-violet light (UV-light), if they are in a windowed package.EPROM-based devices in a standard package are only one-time programmable (OTP).
**Typical data retention time :** greater than 10 . . . 20 years
**Typical erase/program cycles :** OTP . . . 10,000 times
**Typical erase/program times :**some minutes UV-light / about 0.1 msec. per cell

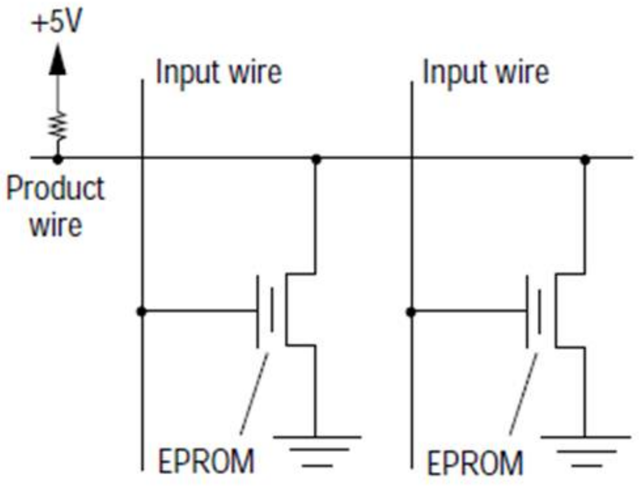


Figure 23: EPROM controlled programmable switches

## 7.2 EEPROM-technology

Electrically-Erasable Programmable Read-Only Memory
EEPROM cells are non-volatile memory cells. An EEPROM memory cell is physically larger (about 3 times) than an EPROM cell, but offers the advantage of being erased electrically. Cause of this great advantage many vendors implemented the in-system programmable (ISP) capability .
**Typical data retention time :** greater than 10. . . 20 years
**Typical erase/program cycles:** greater than 1,000 .. 10,000 times
**Typical erase/program times :** some milliseconds per cell / about 0.1 msec. per cell

## 7.3 Flash EPROM-technology

Flash Erasable Programmable Read-Only Memory
Flash EPROM cells are non-volatile memory cells. FLASH has the electrically-erasable benefits of EEPROM but a smaller physical cell size (about 50 %). Most FLASH-based devices are in-system programmable (ISP).
**Typical data retention time :** greater than 10 . . . 20 years
**Typical erase/program cycles :** greater than 50 . . . 10,000 times
**Typical erase/program times :** about 1 sec. for whole chip / about 0.1 msec. per cell

## 7.4 SRAM-technology

Static Random Access Memory
Similar to the technology used in static RAM devices, but with a few modifications for maximum stability instead of read/write performance. Because SRAM-technology is volatile (the contents disappear after power-down) the devices have to be bootet (configured) after power-up. This makes the devices in-system programmable (ISP) and reconfigurable during operation mode. Most SRAM-based FPGAs can configure themselves automatically at power-up from an external or internal (on-chip) configuration ROM.
**Typical data retention time :** only at stable power-on (volatile)
**Typical erase/program cycles :** unlimited
**Typical erase/program times :** about some milliseconds / milliseconds . . . minutes for whole chip (depends on ROM-interface)

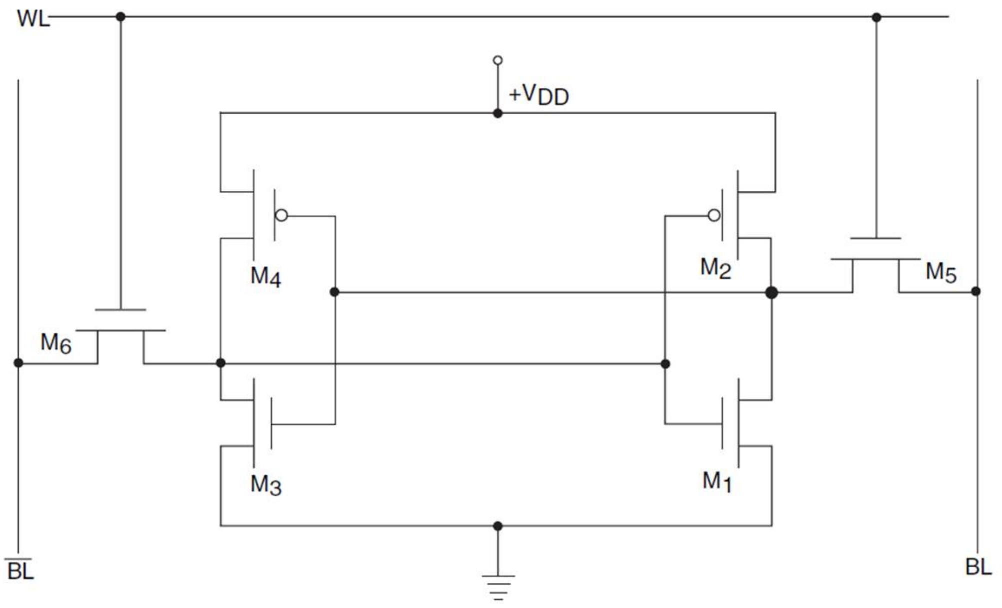


Figure 24: SRAM cell

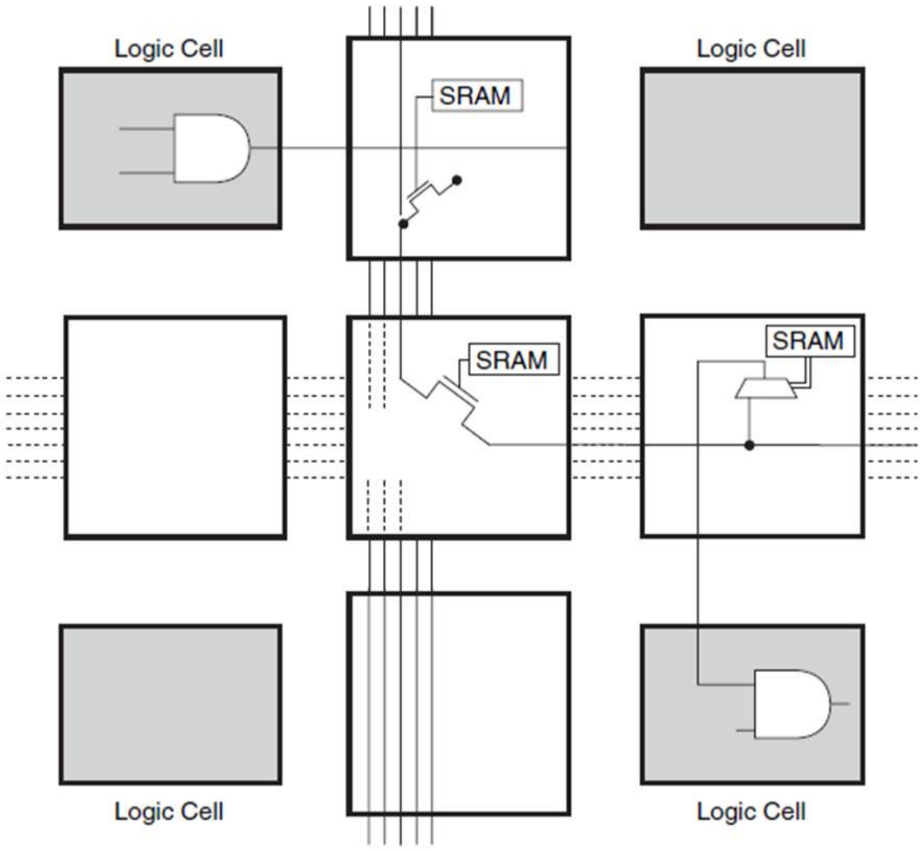


Figure 25: SRAM controlled Interconncet

## 7.5 Antifuse technology

Antifuse cells are non-volatile and only one-time programmable (OTP). Instead of breaking a metal connection by passing current through (like fuse-technology), a link is grown to make a connection. Antifuse-based devices are very good for high reliability applications, cause of their unlimited data-retention time. Antifuse cells are electrically programmed in a device programmer. There are different kinds of CMOS-based PLD-Antifuse-technologies known, like PLICE, ViaLink or MicroVia.

**Typical data retention time :** unlimited
**Typical erase/program cycles :** 1 time (OTP)
**Typical erase/program times :** not erasable / some minutes for whole chip (depends on chip complexity)

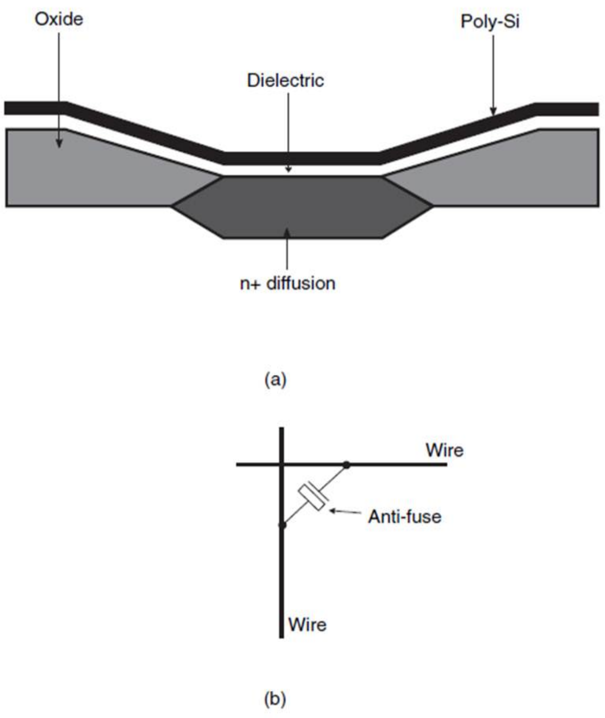


Figure 26: (a) Actel's Antifuse and (b) The Antifuse As A Programmable Interconnect

### 7.6 Fuse-technology

Fuse cells are non-volatile and only one-time programmable (OTP). Fuse-technology was the original programming technology for programmable logic. A fuse is a metal link that can be programmed (blown) by passing a current through. Fuse cells are electrically programmed in a device programmer.

**Typical data retention time :** unlimited
**Typical erase/program cycles :** 1 time (OTP)
**Typical erase/program times :** not erasable / some minutes for whole chip (depends on chip complexity)

**Now, we have frequently used the term data retention time**
The definition of it:-
**Data Retention Time:-**Data retention time is the time a memory bit retain its data state regardless of whether the part is powered on or powered off.

**Comparison of different switch programmable technologies**

| Switch type | Reprogrammable? | Volatile | Technology |
|---|---|---|---|
| Fuse | No | No | Bipolar |
| Eprom | Yes(out of Cicuit) | No | UVCMOS |
| EEPROM | Yes(in circuit) | No | EECMOS |
| SRAM | Yes(in circuit) | Yes | CMOS |
| Antifuse | No | No | CMOS+ |

## 8 Classification of FPGAs according to switch programmable technologies

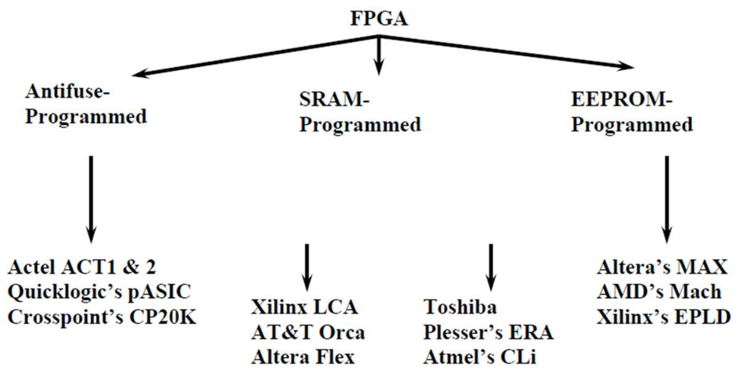


Figure 27: Classification Of FPGAs Based On Switch Programmable Technology

## 8.1 Antifuse Based FPGA

The antifuse based cell is the highest density interconnect by being a true cross point. Thus the designer has a much larger number of interconnects so logic modules can be smaller and more efficient. Place and route software also has a much easier time. These devices however are only one-time programmable and therefore have to be thrown out every time a change is made in the design. The Antifuse has an inherently low capacitance and resistance such that the fastest parts are all Antifuse based. The disadvantage of the antifuse is the requirement to integrate the fabrication of the antifuses into the IC process, which means the process will always lag the SRAM process in scaling. Antifuses are suitable for FPGAs because they can be built using modified CMOS technology. As an example, Actel's antifuse structure is depicted in Fig. 20.9. The figure shows that an antifuse is positioned between two interconnect wires and physically consists of three sandwiched layers: the top and bottom layers are conductors, and the middle layer is an insulator. When unprogrammed, the insulator isolates the top and bottom layers, but when programmed the insulator changes to become a low-resistance link. It uses Poly-Si and n+ diffusion as conductors and ONO as an insulator, but other antifuses rely on metal for conductors, with amorphous silicon as the middle layer.

## 8.2 SRAM Programmed FPGA

The major advantage of SRAM based device is that they are infinitely re-programmable and can be soldered into the system and have their function changed quickly by merely changing the contents of a PROM. They therefore have simple development mechanics. They can also be changed in the field by uploading new application code, a feature attractive to designers. It does however come with a price as the interconnect element has high impedance and capacitance as well as consuming much more area than other technologies. Hence wires are very expensive and slow. The FPGA architect is therefore forced to make large inefficient logic modules (typically a look up table or LUT).The other disadvantages are: They needs to be reprogrammed each time when power is applied, needs an external memory to store program and require large area. Fig. 20.8 shows two applications of SRAM cells: for controlling the gate nodes of pass-transistor switches and to control the select lines of multiplexers that drive logic block inputs. The figures gives an example of the connection of one logic block (represented by the AND-gate in the upper left corner) to another through two pass-transistor switches, and then a multiplexer, all controlled by SRAM cells . Whether an FPGA uses pass-transistors or multiplexers or both depends on the particular product.

### 8.3 EEPROM Programmed FPGA

The EEPROM/FLASH cell in FPGAs can be used in two ways, as a control device as in an SRAM cell or as a directly programmable switch. When used as a switch they can be very efficient as interconnect and can be reprogrammable at the same time. They are also non-volatile so they do not require an extra PROM for loading. They, however, do have their detractions. The EEPROM process is complicated and therefore also lags SRAM technology.

## 9 Today's FPGA

Todays generation of FPGAs consist of various mixes of configurable embedded Ips (large blocks) such as: SRAM, transceivers, I/Os, logic blocks, Arithematic units such as adders and multipliers and routing. Most FPGAs contains programmable logic components called logic elements (LEs) and a hierarchy of reconfigurable interconnects You can configure LEs to perform complex combinational functions, or merely simple logic gates. Most FPGAs, include memory elements, which may be simple flipflops or complete blocks of memory.

## 10 A question answer session

⇒ **What is the fullform of PLD?What are the early programmable devices?**
PLD:-Programmable Logic Device. The early programmable devices are:-

1. Programmable Read Only Memory.
2. Programmable Array Logic.
3. Programmable Logic Array.

They tend to execute SOP logic.

⇒ **What does OTP mean?What are the OTP Programmable Logic devices?**
OTP:-One Time Programmable.
The One Time Programmable logic devices are:-PROM,PAL and PLA.

⇒ **What is the basic difference between PROM,PAL and PLA?**
We have already discussed it. (Referred to 3.4)

⇒ **What is the first Programmable Logic device which is erasasble and reprogrammable?**
Generic Array Logic.

⇒ **A GAL is essentially a _____.**
Reprogrammable PAL.

⇒ **Which technology is used used at the inputs of PAL/GAL devices in order to prevent input loading from a large number of AND gates?**
Latches are used to do so.

⇒ **Why some of the PLDs are called "Field-Programmable"?**
Some of the PLDs are designed to be configured by a customer or a designer after manufacturing. Hence they are called "Field-Programmable".

⇒ **What is the basic difference between Mask programmable logic devices and Field Programmable Logic Devices?**
Basic Differene is that Mask Programmable Logic Devices are programmed by manufacturers. Whereas Field Programmable Logic Devices can be programmed by the user.

⇒ **What kinds of programmable logic devices are available today?**
There are a few major programmable logic architecture available today. Each architecture typically has vendor-specific sub-variants within each type. The major types include:
Simple Programmable Logic Devices (SPLDs),
Complex Programmable Logic Devices (CPLDs)
Field Programmable Gate Arrays (FPGAs)
Field Programmable InterConnect Connection(FPICs)

⇒ **Which logic devices are there in the family of SPLD?**
PROM,PAL,PLA & Gal belong to the family of SPLD.

⇒ **Describe EPIC in a few words.**
An FPIC is not really a logic device but rather a programmable "wiring" device. Through programming, an FPIC connects one pin on the device to another on the device providing programmable interconnect. FPICs use either SRAM or anti-fuse programming technology.
I-Cube provides FPIC devices as stand-alone components. Aptix sells their FPIC devices as part of their hardware emulation system.

⇒ **What is Volatile Memory?**
Volatile memory or volatile storage is computer memory that requires power

to maintain the stored information. It retains its contents while powered, but when power is interrupted stored data is immediately lost. Non-volatile memory is memory that maintains its content even when unpowered.e.g:- Volatile Memory is used in RAM Drive.

⇒ **What is non-volatile memory?** Non-volatile memory, nonvolatile memory, NVM or non-volatile storage is computer memory that can get back stored information even when not powered. Examples of non-volatile memory include read-only memory, flash memory, ferroelectric RAM (F-RAM), most types of magnetic computer storage devices (e.g. hard disks, floppy disks, and magnetic tape) etc.

⇒ **What is ISP?**
In System Programmable.
It refers to the ability to reconfigure the logic and functionality of a Device. It can be done before,after or during the manufacture.

⇒ **What are the features of ISP?**

- Flexible and Easy to modify Hardware.
- Design Upgradaton is simple.
- No special manufacturing flow is requires.
- 20-year program retention ability.
- A minimum of 10000 program erase cycles.

⇒ **What are the advantages of ISP?**

- Faster time to market.
- Internal test and board configuration.
- Superior prototyping with multiple h/w designs.
- Security feature allows density security. i.e. A secured device cant be read back until it has been erased.

⇒ **What is the use of JTAG port in CPLD?**
The term JTAG stands for Joint Test Action Group.
The programming of CPLD is done by transerring the programming information generated by CAD system through the cable, from the computer into the CPLD.The circuitry on the CPLD that allows this type of programming

has been standardized by IEEE and is usually called a JTAG port.
It uses four wires to transfer information between the program and the device being programmed.
JTAG is a non-volatile type of programming i.e programmed state is retained permanently (for example, in case of power failure, CLPD retains the program).

⇒ **What logic is inferred when there are multiple assign statements targeting the same wire in FPGA?**
It is illegal to specify multiple assign statements to the same wire in a synthesizable code that will become an output port of the module. The synthesis tools give a syntax error that a net is being driven by more than one source. However, it is legal to drive a three-state wire by multiple assign statements.

⇒ **What are different types of FPGA programming modes?**
Before powering on the FPGA, configuration data is stored externally in a PROM or some other nonvolatile medium either on or off the board. After applying power, the configuration data is written to the FPGA using any of five different modes: Master Parallel, Slave Parallel, Master Serial, Slave Serial, and Boundary Scan (JTAG). The Master and Slave Parallel modes Mode selecting pins can be set to select the mode.

⇒ **Difference between FPGA and CPLD?**
**FPGA:**

1. SRAM based technology.
2. Segmented connection between elements.
3. Usually used for complex logic circuits.
4. Must be reprogrammed once the power is off.
5. Costly

**CPLD:**

1. Flash or EPROM based technology.
2. Continuous connection between elements.
3. Usually used for simpler or moderately complex logic circuits.
4. Need not be reprogrammed once the power is off.

5. Cheaper

⇒ **Name some of the FPGAs and CPLDs.**

**FPGA:-**

- LCA:-Logic Cell Array.
- ACT:- Actel.
- FLEX,APEX :-Altera.

**CPLD:**

- EPLD:-Erasable Programmable Logic Device.
- PEEL:-Programmable Electrically Erasable Device.
- EEPLD:-Electrically Erasable Programmable Logic Device.
- MAX:-Multiple Array Matrix of Altera.

⇒ **What is CLB,Slice & LUT in an FPGA?**

The Configurable Logic Blocks (CLBs) constitute the main logic resource of FPGA for implementing synchronous as well as combinatorial circuits.
CLB are configurable logic blocks and can be configured to combo,ram or rom depending on coding style

CLB consist of slices and each slice consist of two 4-input LUT (look up table) F-LUT and G-LUT generally.(This is a general answer. The nos of Slices and LUTs can be varied from model to model).

⇒ **How can we classify an FPGA structure?**

FPGAs are commercially available in many different architectures and organizations. Although each company's offerings have unique characteristics, FPGA architectures can be generically classified into one of four categories: Symmetrical Array, Row Based, Hierarchical PLD, and Sea of Gates.

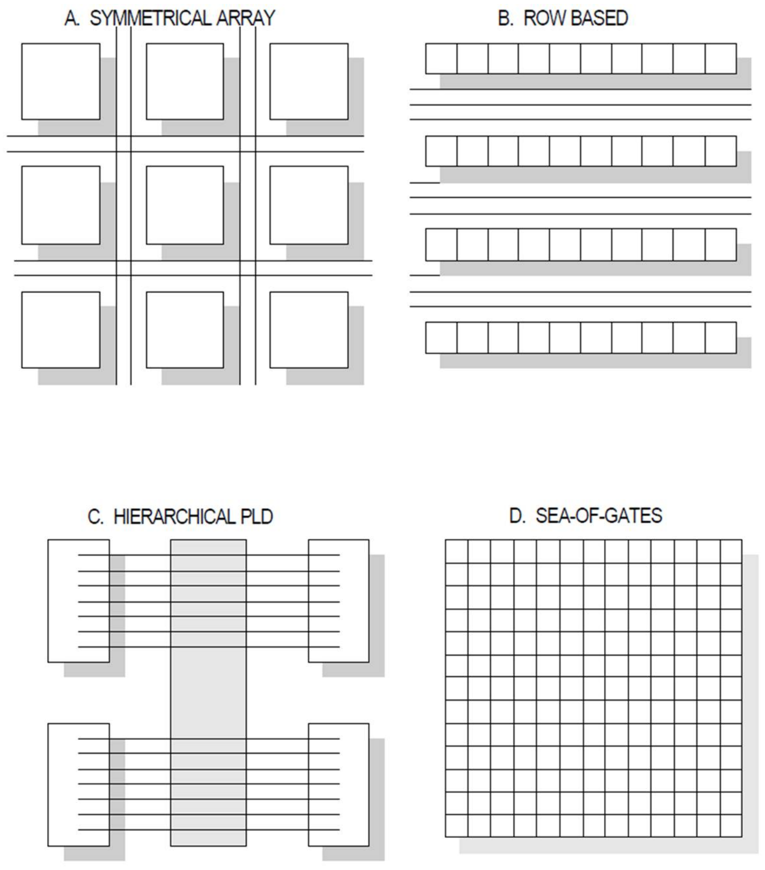


Figure 28: Classfication Of FPGAS According To Their Structure

⇒ Can a clb configured as ram?
Yes.
The configuration design is provided below.

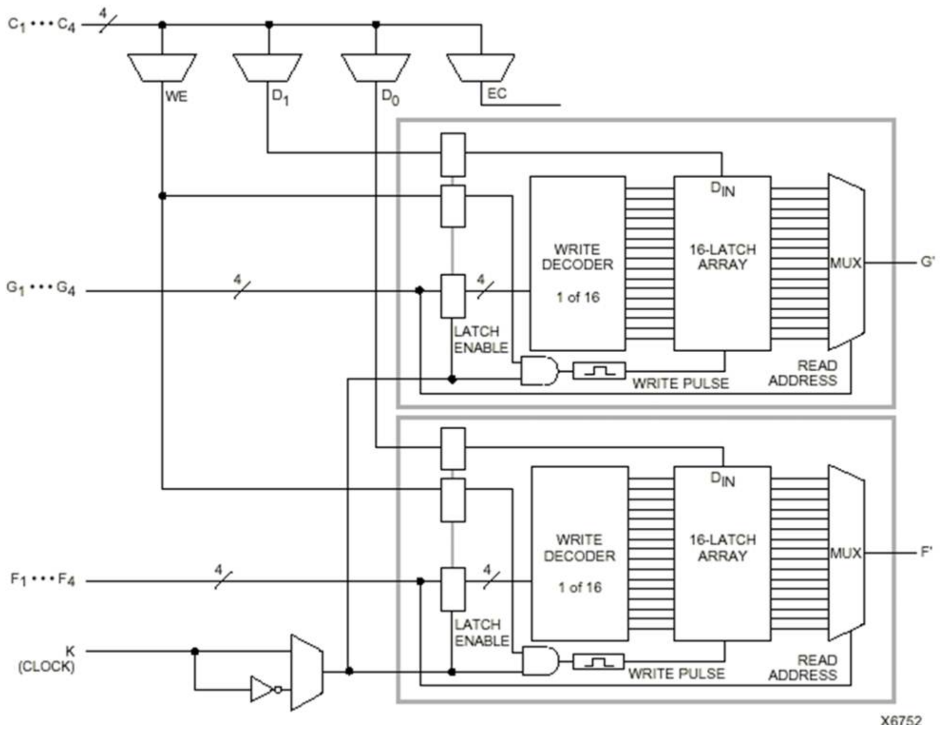


Figure 29: 16×2(or16×1)Edge triggered single port RAM

⇒ **What is LUT?**
LUT:-Look Up Table.
Look-up tables (LUTs) are most commonly used logic blocks.It contains storage cell that are used to implement a simple logic function.Each cell is capable of holding a single logic value 0 or 1. The store valueis produced as the output of the storage cell.It is used to implement function generators in CLBs. Four independent inputs are provided to each of two function generators (F1-F4 and G1-G4). These function generators can implement any arbitrarily defined Boolean function of four inputs.

⇒ **How the propagation delay is reduced in an FPGA?**
It is done by increasing the circuit coplexity in such a manner that the propagation delay is reduced. One of those methods to reduce circit complexity

is First carry Logic method. The circuit diagram of the respective method is in the respective section.

⇒ **What is the fullform of ASIC?Tell about ASIC in a few words.**
Application Specific Integrated Circuit.
It is a special type of integrated circuit which is customized for a particular use.

⇒ **What are the different categories of ASIC?**
There are three different categories of ASICS:

- **Full-Custom ASICS:** These are custom-made from scratch for a specific application. Their ultimate purpose is decided by the designer. All the photolithographic layers of this integrated circuit are already fully defined, leaving no room for modification during manufacturing.
- **Semi-Custom ASICs:** These are partly customized to perform different functions within the field of their general area of application. These ASICS are designed to allow some modification during manufacturing, although the masks for the diffused layers are already fully defined.
- **Platform ASICs:** These are designed and produced from a defined set of methodologies, intellectual properties and a well-defined design of silicon that shortens the design cycle and minimizes development costs. Platform ASICs are made from predefined platform slices, where each slice is a premanufactured device, platform logic or entire system. The use of premanufactured materials reduces development costs for these circuits.

⇒ **What is the application of FPGA in ASICs**

Leading-edge ASIC designs are becoming more expensive and time-consuming because of the increasing cost of mask sets and the amount of engineering verification required. Getting a device right the first time is imperative. A single missed deadline can mean the difference between profitability and failure in the product life cycle.

Using an FPGA to prototype an ASIC or ASSP for verification of both register transfer level (RTL) and initial software development has now become standard practice to both decrease development time and reduce the risk of first silicon failure. An FPGA prototype accelerates verification by allowing testing of a design on silicon from day one, months in advance of final silicon becoming available. Code can be compiled for the FPGA, downloaded, and debugged in hardware during both the design and verification phases using a variety of techniques and readily available solutions. Whether you're doing RTL validation, initial software development, or system-level testing, FPGA prototyping platforms provide a faster, smoother path to delivering an end working product.

This is the application of an FPGA in an ASIC.

## 11 References

- en.wikipedia.org
- google.co.in
- http://www.fpga-guide.com/
- http://only-vlsi.blogspot.in/
- http://corelis.com/blog/jtag-programming-of-cplds-aamp-fpgas/
- Digital Logic and Computer Design by Morris Mano
- Introduction to CPLD and FPGA Design By Bob Zeidman
- Lecture 12(a tutorial lecture) of Concordia University.
- Design of Embedded Processors A tutorial of IIT kharagpur.
- Architecture of Field Programmable Gate Arrays by Jonathon Rose,member,IEEE. Abbas El Gamal, Senior member,IEEE and Alberto Sangiovanni-Vincentelli,Fellow,IEEE
- FPGA and CPLD Architectures: A Tutorial by STEPHEN BROWN & JONATHAN ROSE,Toronto University
- Reconfigurable Computing (EN2911X, Fall07) Lecture 02: RC Principles: Programmable Logic Technology (1/3) by Prof. Sherief Reda Division of Engineering, Brown University
- Programmable Logic and Application Specific Integrated Circuits by Dave Landis, Ph.D., P.E. Professor of Electrical Engineering The Pennsylvania State University