



Programmable Logic Devices

INTRODUCTION

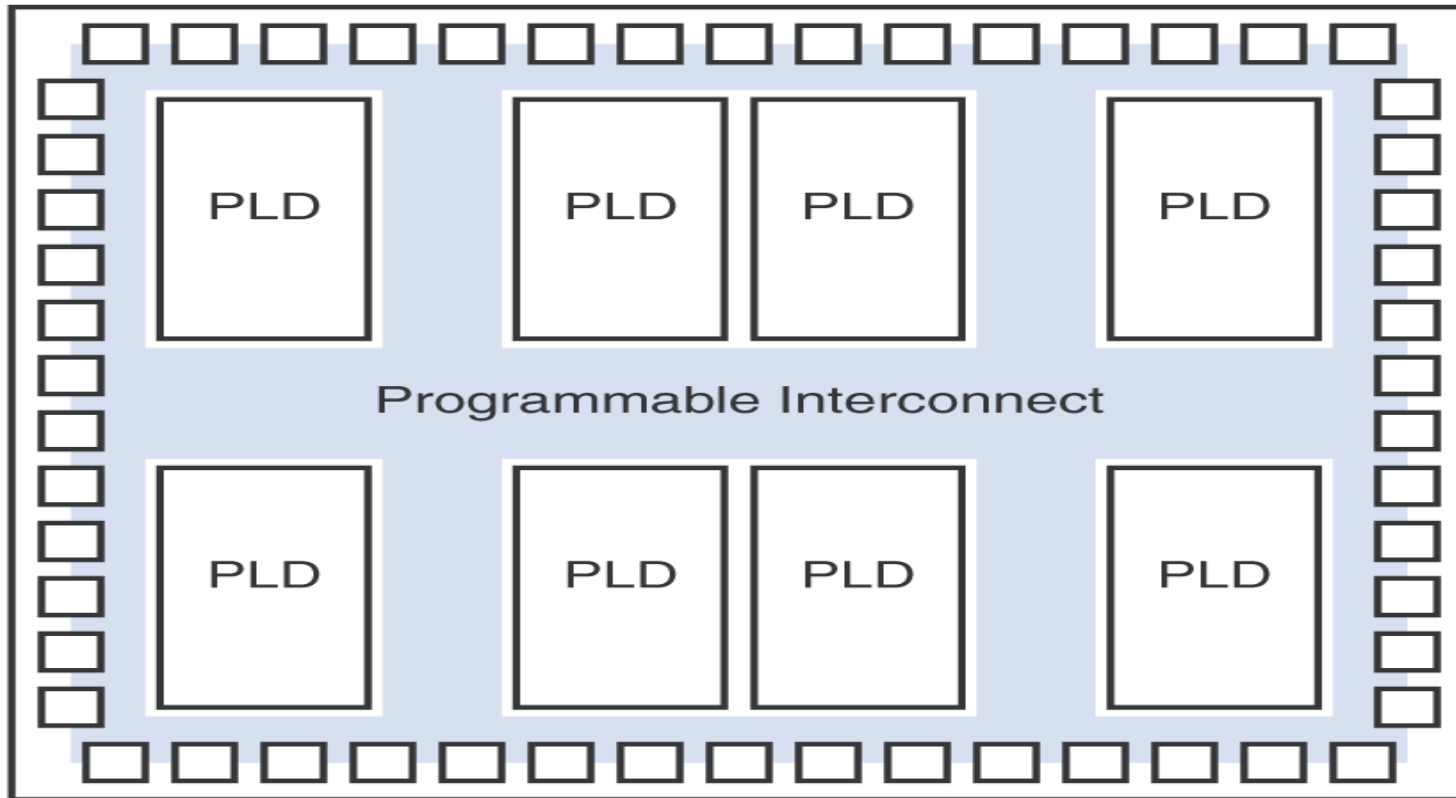
- A programmable logic device or PLD is an electronic component used to build reconfigurable digital circuits. Unlike a logic gate, which has a fixed function, a PLD has an undefined function at the time of manufacture. Before the PLD can be used in a circuit it must be programmed, that is, reconfigured.

CPLD

- Complex Programmable Logic Devices (CPLD)
- SPLDs (PLA, PAL) are limited in size due to the small number of input and output pins and the limited number of product terms
- CPLDs contain multiple circuit blocks on a single chip
 - Each block is like a PAL: PAL-like block
 - Connections are provided between PAL-like blocks via an interconnection network that is programmable
 - Each block is connected to an I/O block as well

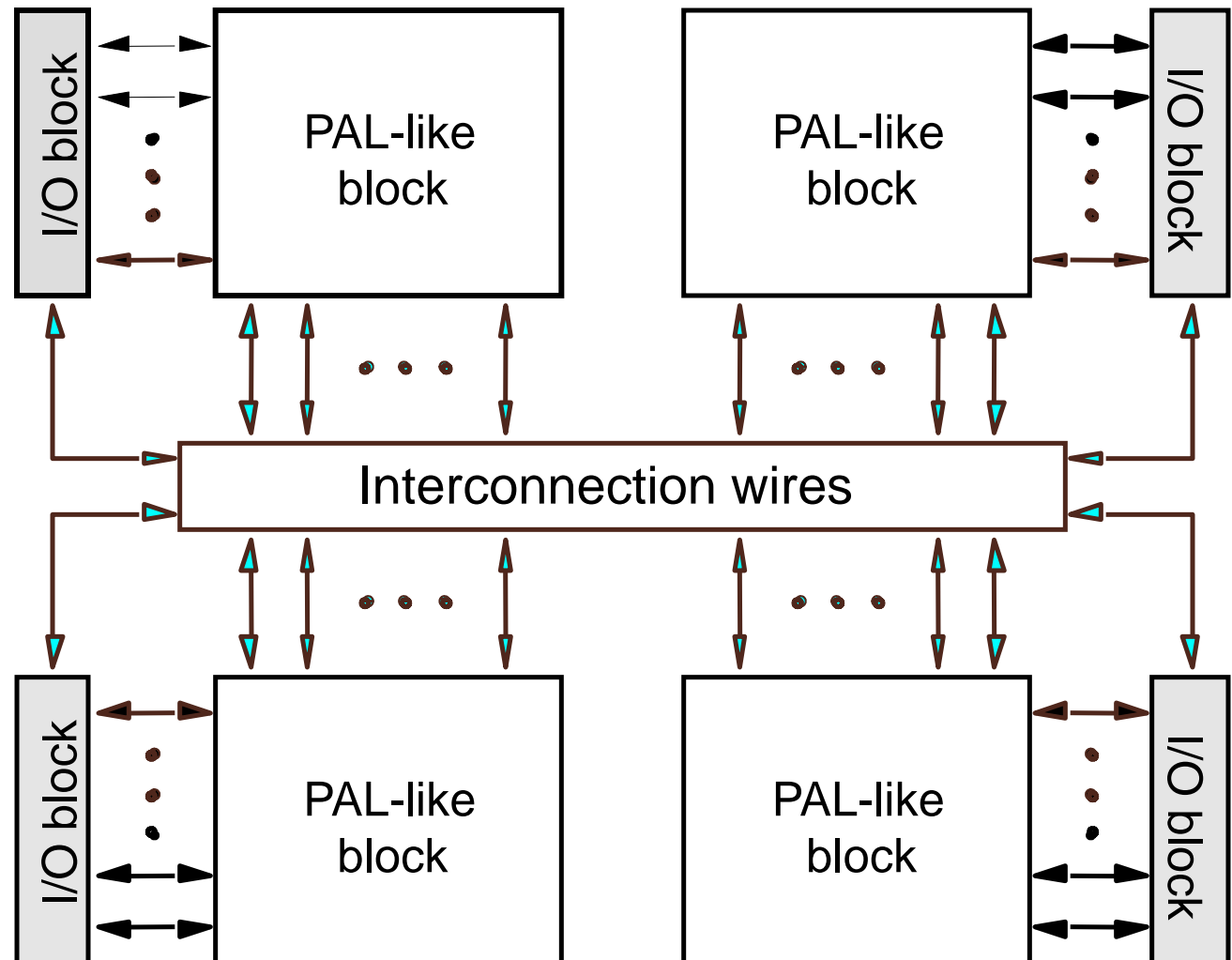
CPLD

- A CPLD is just a collection of individual PLDs on a single chip, accompanied by a programmable interconnection structure.
- This structure allow the PLDs to be interconnected in the same way as can be done off-chip.



□ = input/output block

Structure of a CPLD: A Closer Look



CPLD Logic Block

- Inputs
- Product-term array
- Macro-cells (registers)
- Logic blocks executes sum-of-product expressions and stores the results in micro-cell registers
- CPLDs have many pins – large ones have > 200
 - Removal of CPLD from a PCB is difficult without breaking the pins
 - Use ISP (in system programming) to program the CPLD
 - JTAG (Joint Test Action Group) port used to connect the CPLD to a computer

Why CPLDs?

- For larger applications, we could simply increase the number of inputs and outputs in a conventional SPLD ...
- Problems:
 - too costly
 - logic gets slower as number of inputs to AND array increases
- Solution:
 - multiple PLDs with a relatively small (fast) programmable interconnect
 - less general than a single large PLD, but we can use software to partition our design into smaller PLD blocks

- To create a CPLD device:
 - put a lot of Simple PLDs on the same chip
 - add “wires” between them whose connections can be programmed (interconnect)
 - use EEPROM technology for the connections

CPLD featured in common FPGA:-

- i. Large number of gates available.
- ii. Can include complicated feedback path.

CPLD application:-

- i. Address coding
- ii. High performance control logic
- iii. Complex finite state machines

FPGA

- SPLDs and CPLDs are relatively small and useful for simple logic devices
 - Up to about 20000 gates
- Field Programmable Gate Arrays (FPGA) can handle larger circuits
 - No AND/OR planes
 - Provide logic blocks, I/O blocks, and interconnection wires and switches
 - Logic blocks provide functionality
 - Interconnection switches allow logic blocks to be connected to each other and to the I/O pins

Field Programmable Logic Devices (FPGAs)

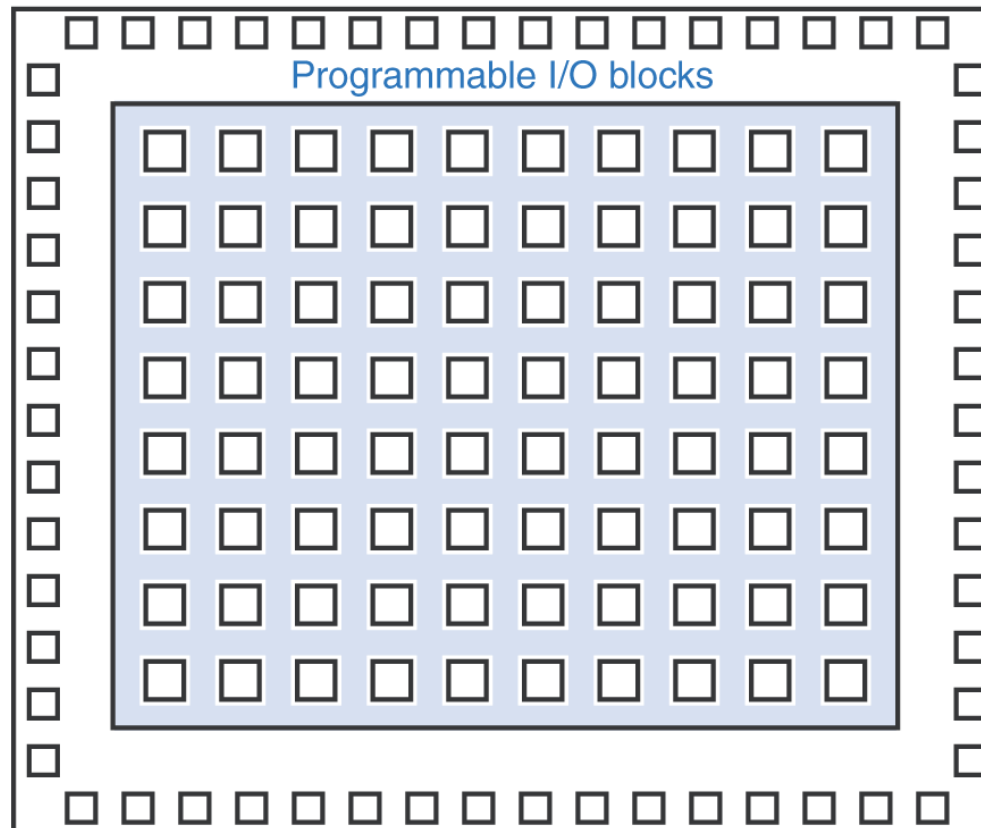
- Gate arrays are non-programmable devices that can be manufactured more cheaply than other types of IC, because they contain a standard grid of logic gates whose interconnections are specified by the customer.
- When a customer orders a new type of chip, the manufacturer does not have to design it from scratch, but can just take a standard gate array and modify it to the customer's requirement.

Field Programmable Logic Devices (FPGAs)

- FPGAs use a similar grid of logic gates, but the programming is done by the customer, not by the manufacturer.
- The term "field-programmable" may be obscure to some, but the "field" is just an engineering term for the world outside the factory where customers live.

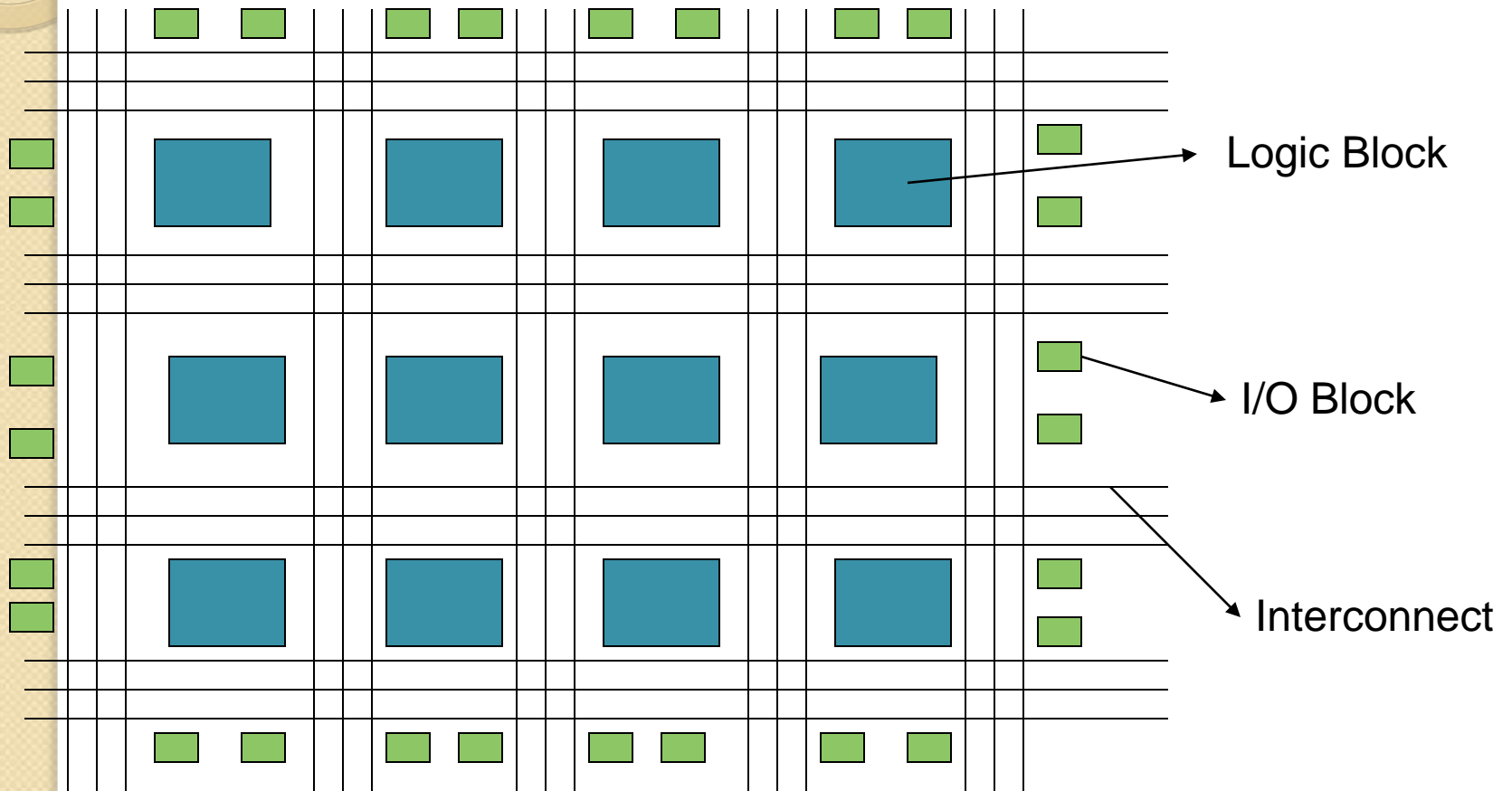
General FPGA Architecture

- Field Programmable Gate Array



- = Programmable interconnect
- = Programmable logic block
- = I/O pad

Structure of FPGA (Xilinx)

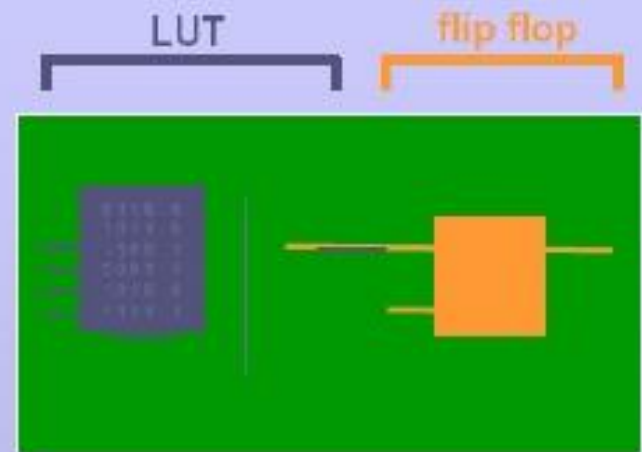


Programming an FPGA

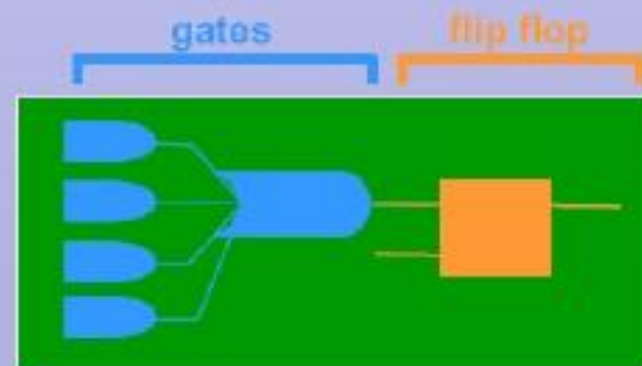
- In System Programming method is used
 - None of the other PLD technologies are volatile
 - FPGA storage cells are loaded via a PROM when power is first applied

FPGA Types

- **2 types of FPGAs**
- **Reprogrammable (SRAM-based)**
 - Xilinx, Altera, Lattice, Atmel
- **One-time Programmable (OTP)**
 - Actel, Quicklogic



SRAM logic cell



OTP logic cell

FPGA advantages

- Highly flexible
 - Control your design down to the individual gate level
 - Designs can be updated without rewiring
 - Quick turnaround during development
 - Easy to try alternative solutions
- Excellent real-time performance
 - No interrupt handler context switch overhead
 - Fully parallel operation

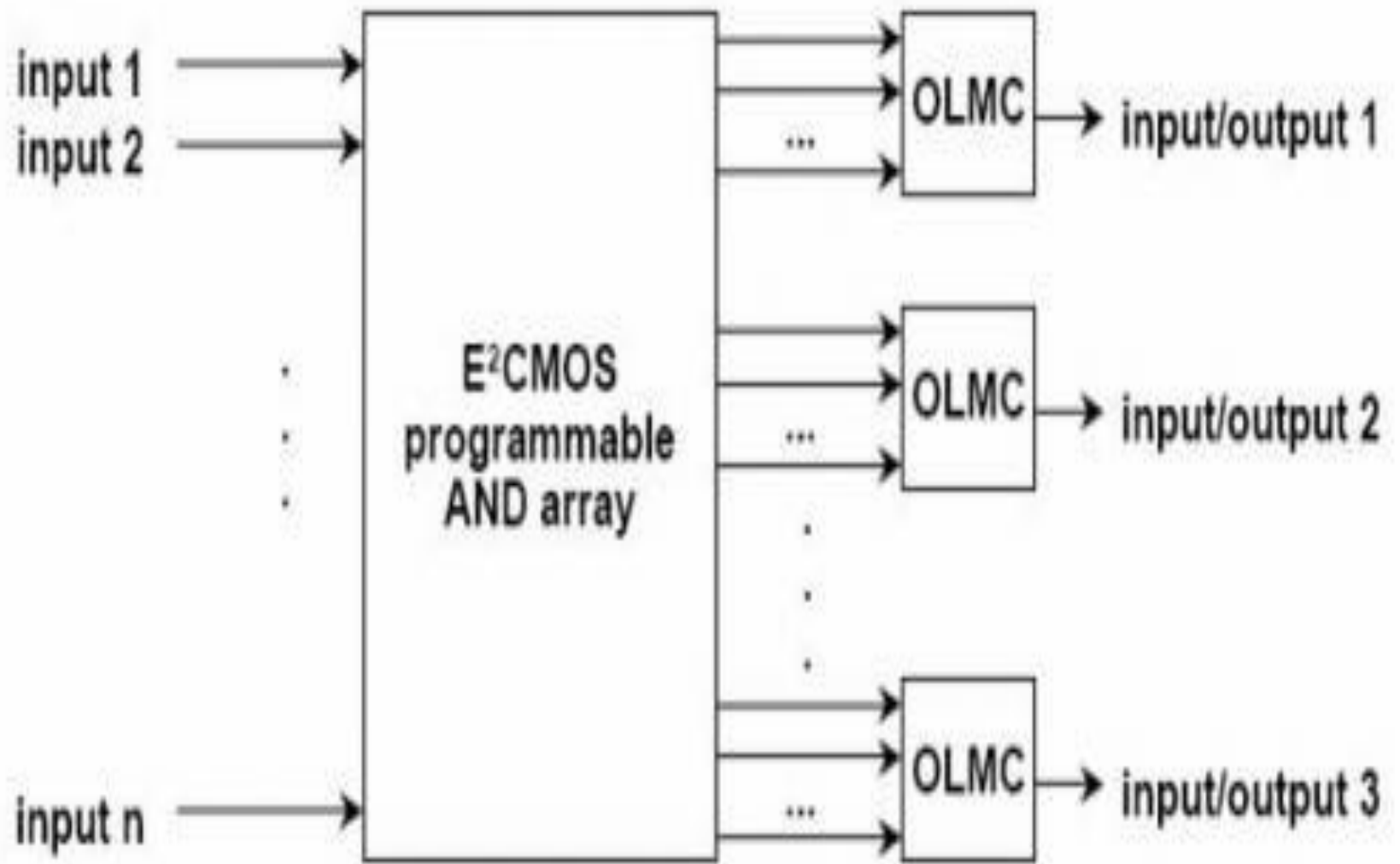
FPGA

FPGA applications:-

- i. DSP
- ii. Software-defined radio
- iii. Aerospace
- iv. Defense system
- v. ASIC
- vi. Medical Imaging
- vii. Computer vision
- viii. Speech Recognition
- ix. Bioinformatic
- x. And others.

Generic Array Logic (GAL):

- Generic array logic family consists of electrically erasable programmable devices designed by lattice semiconductor.
- Same logic properties as PAL but can be erased and reprogrammed.
- Programmed and reprogrammed using a PAL programmer
- It has a fixed OR array and a programmable And array the reprogrammable array is essentially a grid of conductors forming rows and columns with an electrically erasable CMOS (E2CMOS) cell at each cross point.
- The GAL has the programmable logic and the OLMC (Output Logic Macro cell) Logic that excludes OR gates and flip-flops.




Generic Array Logic – Macrocell

- I/O circuit that can be configured as a registered output, a combinational output, or a dedicated input as required.
- Outputs can also be specified as active-HIGH or active-LOW.

PEEL (Programmable Electrically Erasable Logic) Device :

- Programmable Electrically Erasable Logic
- Introduced by the international CMOS technology (ICT) corporation.
- ICT offer the most flexible PLD solutions for lower pin count application.
- Include PEEL devices, PEEL array and PEEL development tool.
- PEEL devices are another family of devices that are intended as PAL replacements the PEEL is available in 20 pin different packages with speeds ranging from 5ns to 25ns.
- The PEEL architecture allows it to replace over 20 standard 20 pin PLDs (PAL, GAL, etc.)

- 
- Features of PEEL:
 - 1. Speed ranging from 5ns to 25ns
 - 2. Low Power consumption.
 - 3. CMOS Electrically Erasable Technology.
 - 4. Reduces development Cost
 - 5. Flexible architecture.

Design implementation using CPLD and FPGAs

- Implementing a logic design with the FPGA or CPLD development software usually consists of the following steps :
 1. You enter a description of your logic circuit using a *hardware description language* (HDL) such as VHDL or Verilog. You can also draw your design using a schematic editor.
 2. You use a *logic synthesizer* program to transform the HDL or schematic into a *netlist*. The netlist is just a description of the various logic gates in your design and how they are interconnected.

Cont..

3. You use the *implementation tools* to map the logic gates and interconnections into the FPGA. The configurable logic blocks in the FPGA can be further decomposed into *look-up tables* that perform logic operations. The CLBs and LUTs are interwoven with various *routing resources*. The mapping tool collects your netlist gates into groups that fit into the LUTs and then the *place & route tool* assigns the gate collections to specific CLBs while opening or closing the switches in the routing matrices to connect the gates together.
4. Once the implementation phase is complete, a program extracts the state of the switches in the routing matrices and generates a *bitstream* where the ones and zeroes correspond to open or closed

Cont..

5. The bitstream is *downloaded* into a physical FPGA chip (usually embedded in some larger system such as an XS Board). The electronic switches in the FPGA open or close in response to the binary bits in the bitstream. Upon completion of the downloading, the FPGA will perform the operations specified by your HDL code or schematic. You can apply input signals to the I/O pins of the FPGA to check the operation of your design.

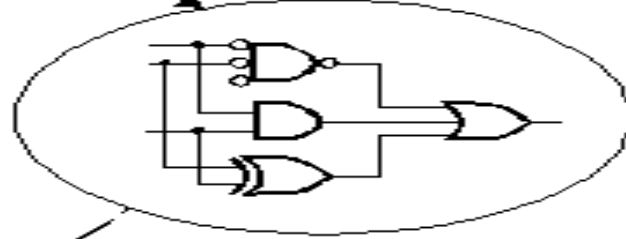
VHDL Source Code

```
entity leddcd is
  port(
    d: in std_logic_vector(3 downto 0);
    s: out std_logic_vector(6 downto 0);
  );
end;

architecture leddcd_arch of leddcd is
begin
  s <= "1110111" when d="0000" else
    "0010010" when d="0001" else
    "1101101";
end leddcd_arch;
```

Synthesize

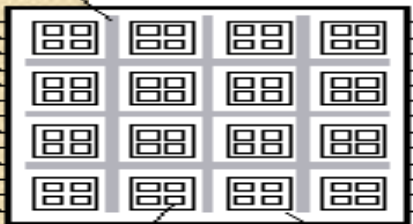
Netlist



Map, Place & Route

routing resources

FPGA



look-up table

configurable logic block

Generate Bitstream

Bitstream

```
101010010101100101
01011010101010110101
010110100101101011
01010101001010101010
10101010101001101010
11011011010101001010
110100101011001011
001011001010101001
010101101001101001
011001100010101010
101010100110010101
```

Download and Test

XSA Board

