University Of Diyala
College of Engineering
Computer Engineering Department

Introduction to CPLD and FPGA

Dr.Yasir Amer Abbas

Third Class

2020

Types of PLD's

PLD
Programmable Logic Device

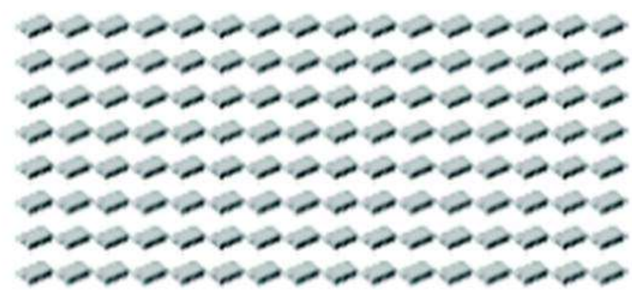
- **SPLD** – Simple Programmable Logic Device
  - PAL
  - GAL
  - PLA
- **CPLD** – Complex Programmable Logic Device
  - CPLD
  - EPLD
  - EEPLD
  - SPLD
  - XPLD
- **FPGA** – Field Programmable Gate Array
  - FPGA
  - LCA
  - pASIC
  - SPGA
  - XPGA

Problems Using SPLD

- Current trend is
  - Increasing gate count
  - Increase design complexity
  - Requirement for smaller size due to lower cost, lower power and higher reliability
  - Fast prototyping for quick design verification
  - PROM, PLA and PAL not used much except in small designs!
- Solution:
  - CPLD for intermediate complexity
  - FPGA for very complex designs (up to millions of gates)

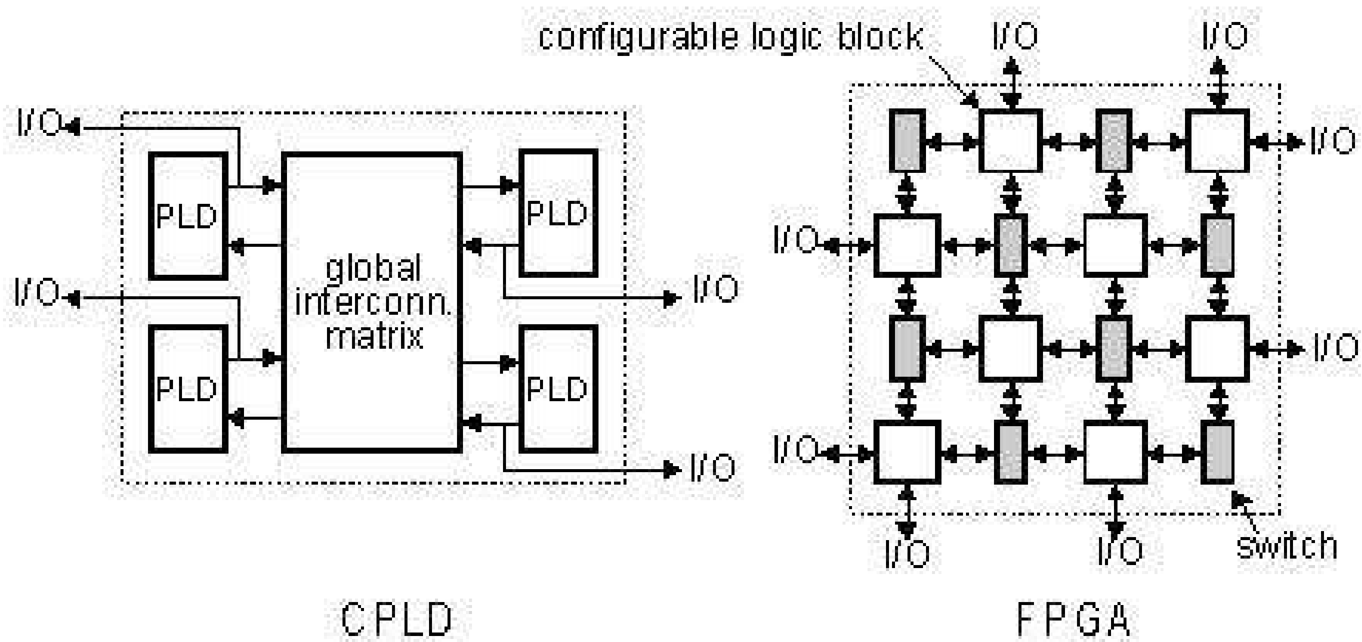
Advantages of FPGA & CPLDs

- Reduced circuit board space utilization, and significant cost savings
  - A 44-pin CPLD is equivalent to 600 gates
  - Some CPLDs have gate equivalents in the millions and over 1000 pins.



- Universal inventory, as one IC can be programmed for various applications

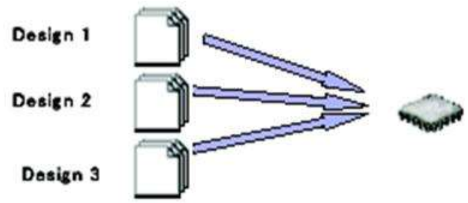
CPLD and FPGA Architectures



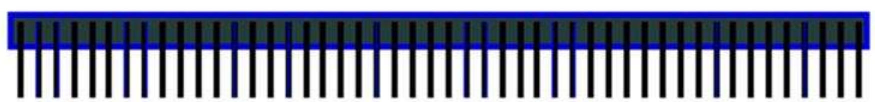
CPLD and FPGA Architectures

Advantages of FPGA & CPLDs

- Reprogrammability
  - FPGA & CPLD can be reprogrammed hundreds of times.



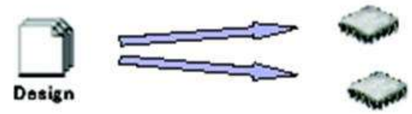
- Number of I/Os



  - FPGA & CPLD have large amounts of programmable input/output contacts.

Advantages of FPGA & CPLDs

- Easiest to modify
  - Modify the software to modify the hardware
- Easy to duplicate



- Direct entry of conceptual design into functional circuit (Quick time-to-market)
  - Streamlined design to prototype process

Advantages of FPGA & CPLDs

- Software and Language
  - Manufacturers of FPGA & CPLDs supply design software (basic software, like the Max+ PlusII , is a free download)
  - VHDL & verilog is a standards-based language that most manufacturers conform to.
- Simple Interface
  - Devices can be interfaced directly to a computer with a serial or USB connection .
- In-circuit modifications
  - With the proper interface connections, the FPGA & CPLD logic can be edited in-circuit.
- Transportable design
  - As a designer you may easily exchange your designs and design modifications (email, CD, web site, etc etc…).
- Large amounts of logic gates, registers, RAM and routing resources

– Low-cost, High density, High speed, Flexible

Difference between CPLD & FPGA

Sr. No.	CPLD	FPGA
1.	Complex Programmable Logic Device	Field Programmable Gate Array
2.	It is collection PLDs.	It is collection of CLBs.
2.	Gate density up to 10000 gates	1000-few hundred 1000s gates
4.	Interconnection wise it is complex so called as CPLD.	Programmable at field or site so called as FPGA.
5.	PAL like blocks used as PLDs.	CLBs used as building blocks.
6.	AND-OR arrays are there in PAL like blocks.	LUTs are there in CLBs.
7.	Configuration context is stored in ROM.	Configuration context is stored in RAM.
8.	Configuration context is Non-volatile.	Configuration context is volatile.

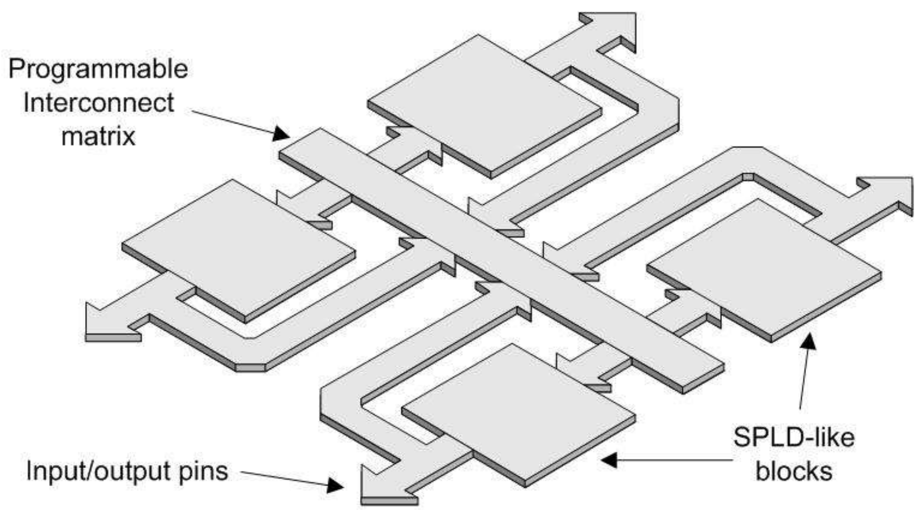
Complex Programmable Logic Devices (CPLDs)

CPLDs

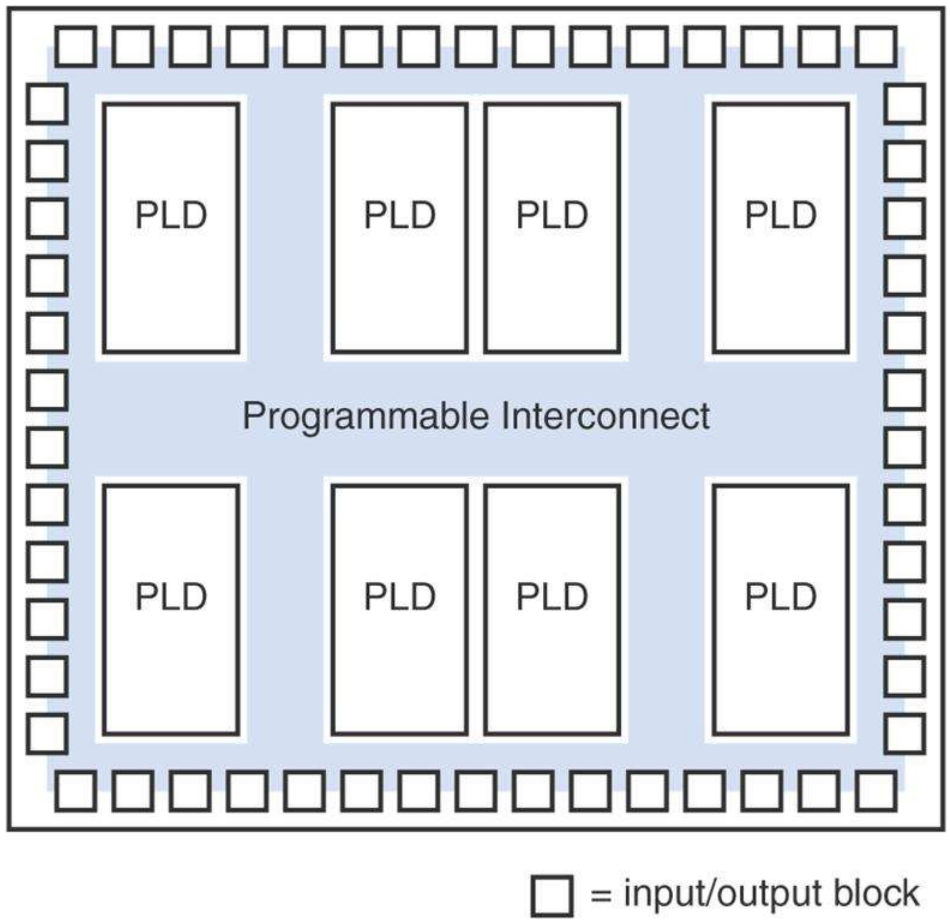
- Complexity of CPLD is between FPGA and PLD.
- CPLDs evolved from PAL\PLAs - as chip densities increased.
- Larger sizes of CPLDs allow either more logic equations or more complicated designs to be realized.
- Current devices have additional logic and registers units and more complex routing topologies than PAL/PLAs.
- Complex routing provides greater flexibility.
- Clock management, clock dividers + global routing lines.
- Sleep\standby power facilities available in modern devices.

CPLDs

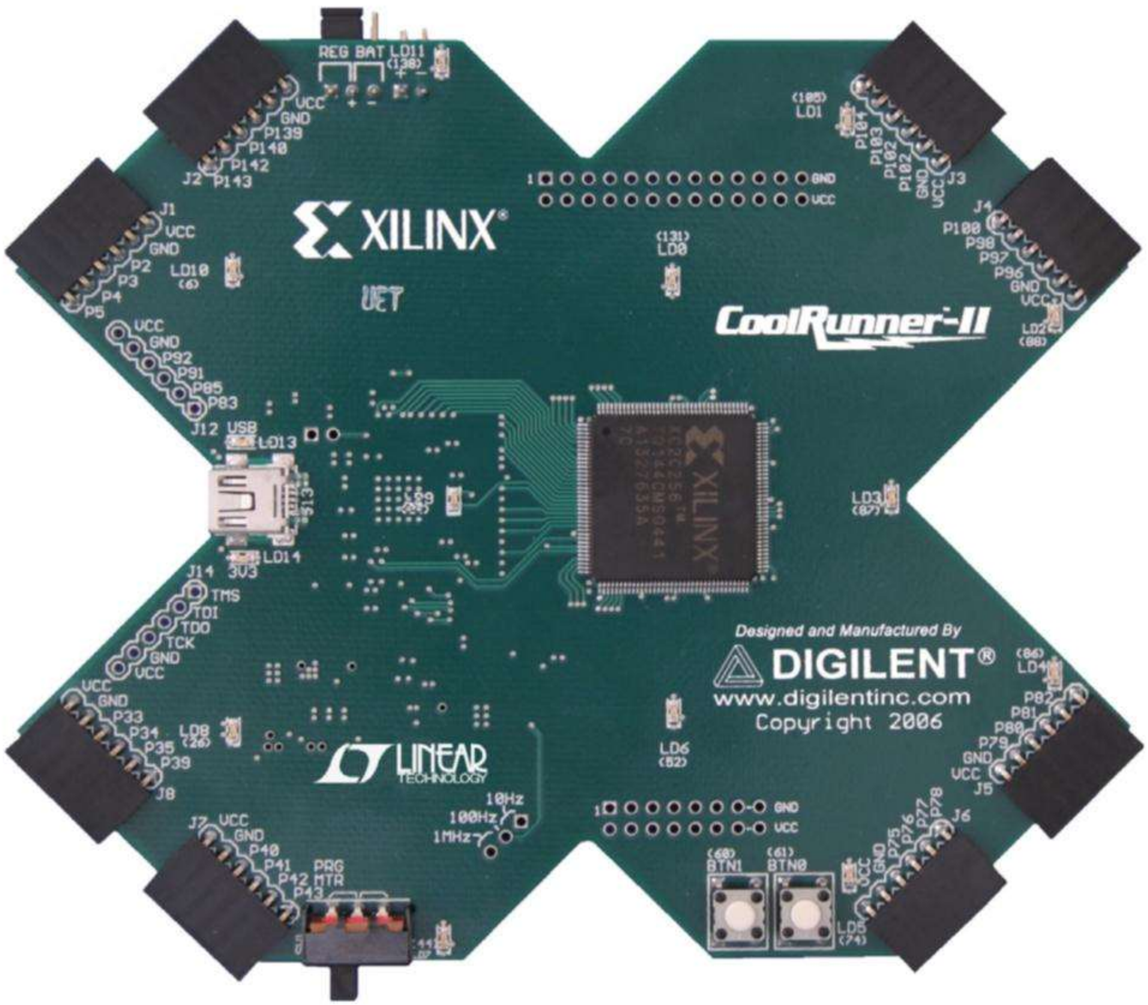
- Core voltage supplies have dropped as low as 1.8v.
- Cheap solution for small and simple tasks, around £3-8.
- Manufacturers include Xilinx, Altera, Atmel, Actel, Lattice.
- Generic structure - each of the four logic blocks are equivalent to one PLD.



General CPLD Architecture



CPLD
Architecture Examples

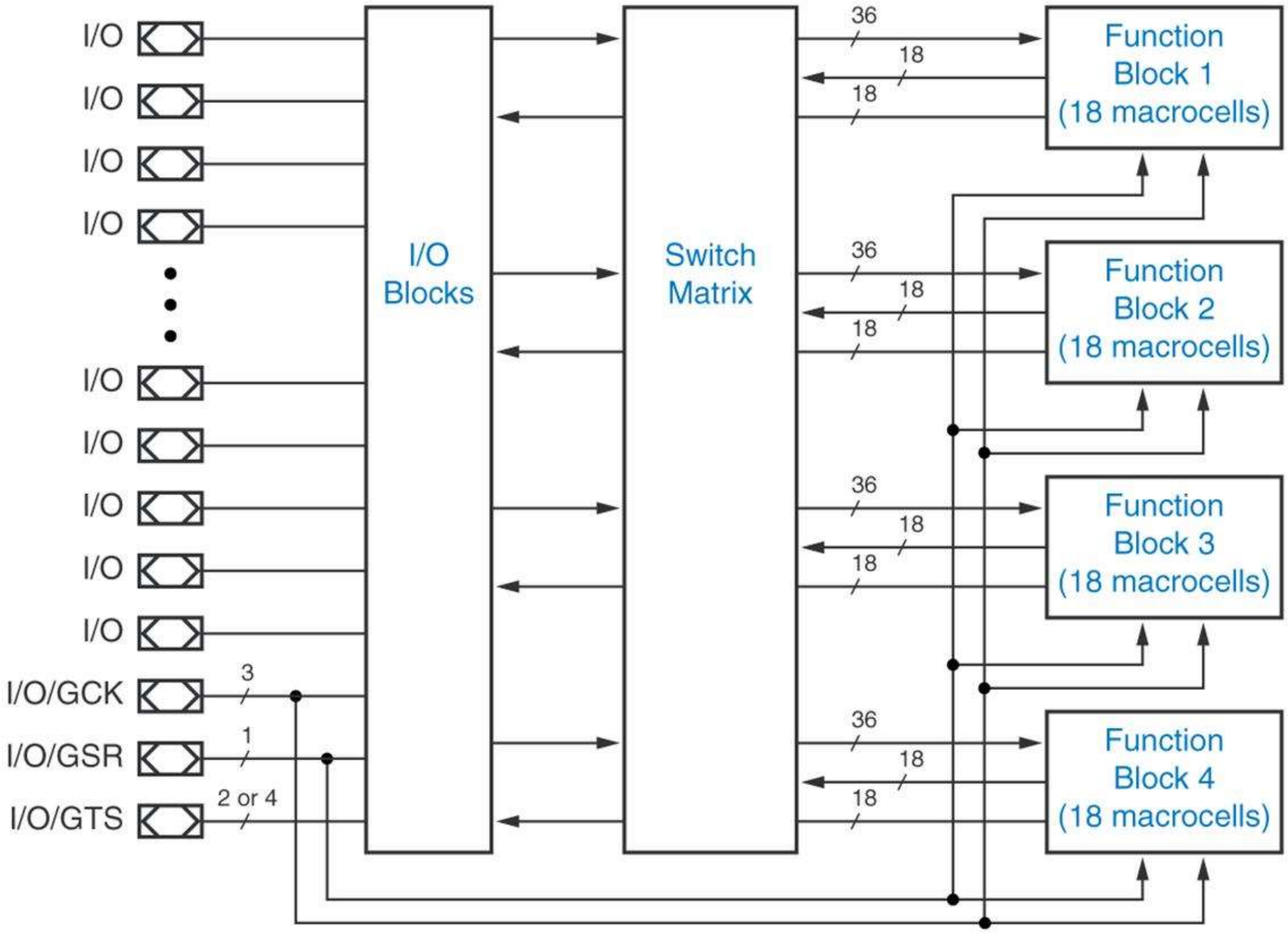


Example CPLD Device

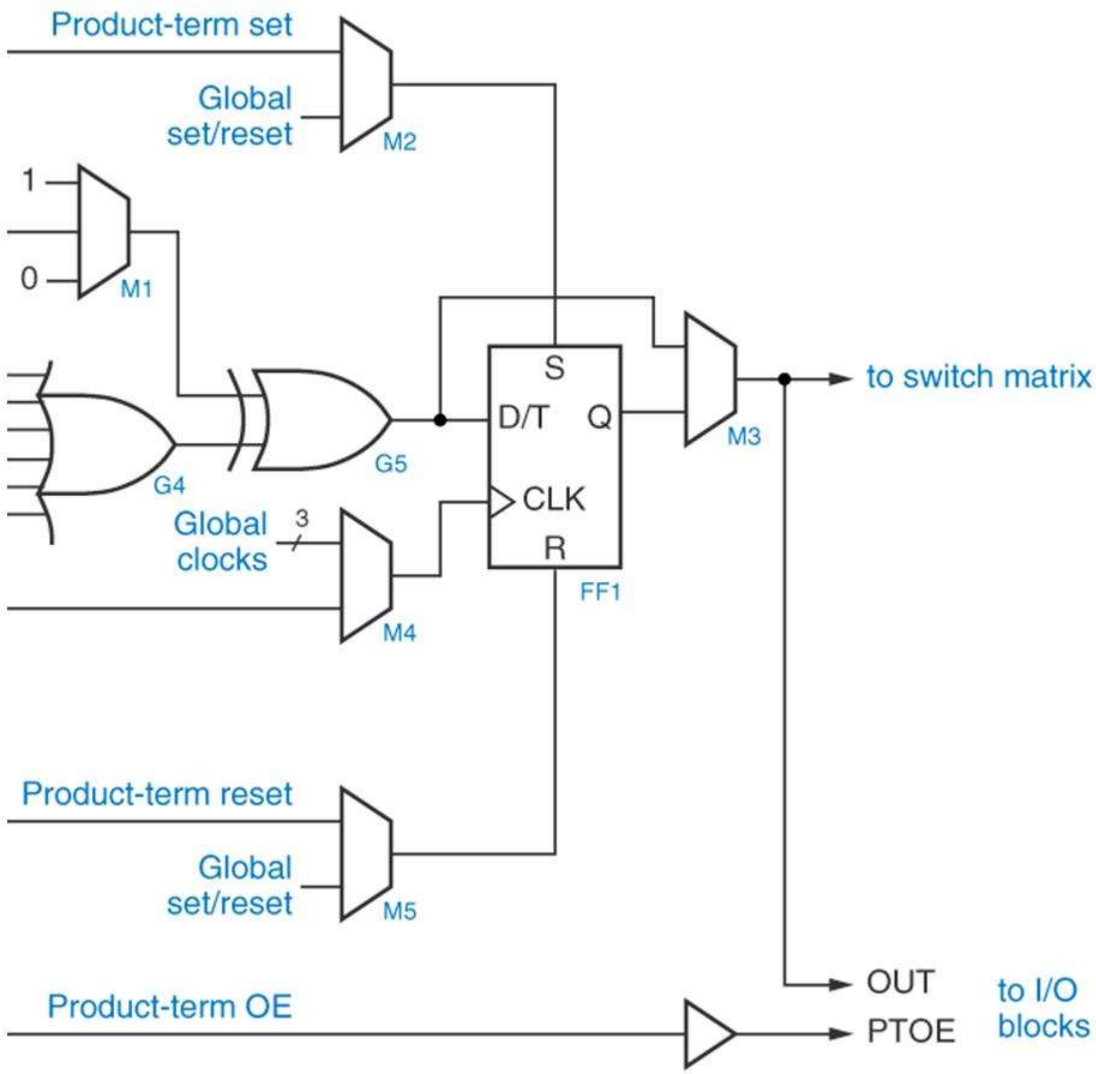
- Example CPLD - Xilinx XC2C512 CoolRunner-II

  - 180 Nanometer technology
  - clocking speed 217MHz (Max 333MHz in family)
  - core voltage 1.8v, (I/O requires 3.3v)
  - *Macro cell* based structure (512 capacity)
  - advanced routing with global clocks + clock dividers
  - low power, typically several hundred micro amps through DataGate feature
  - Flash programming (20 yrs data retention)
  - integrates with Xilinx FPGA software
  - low cost solution, less than £5

Xilinx CPLD Architecture

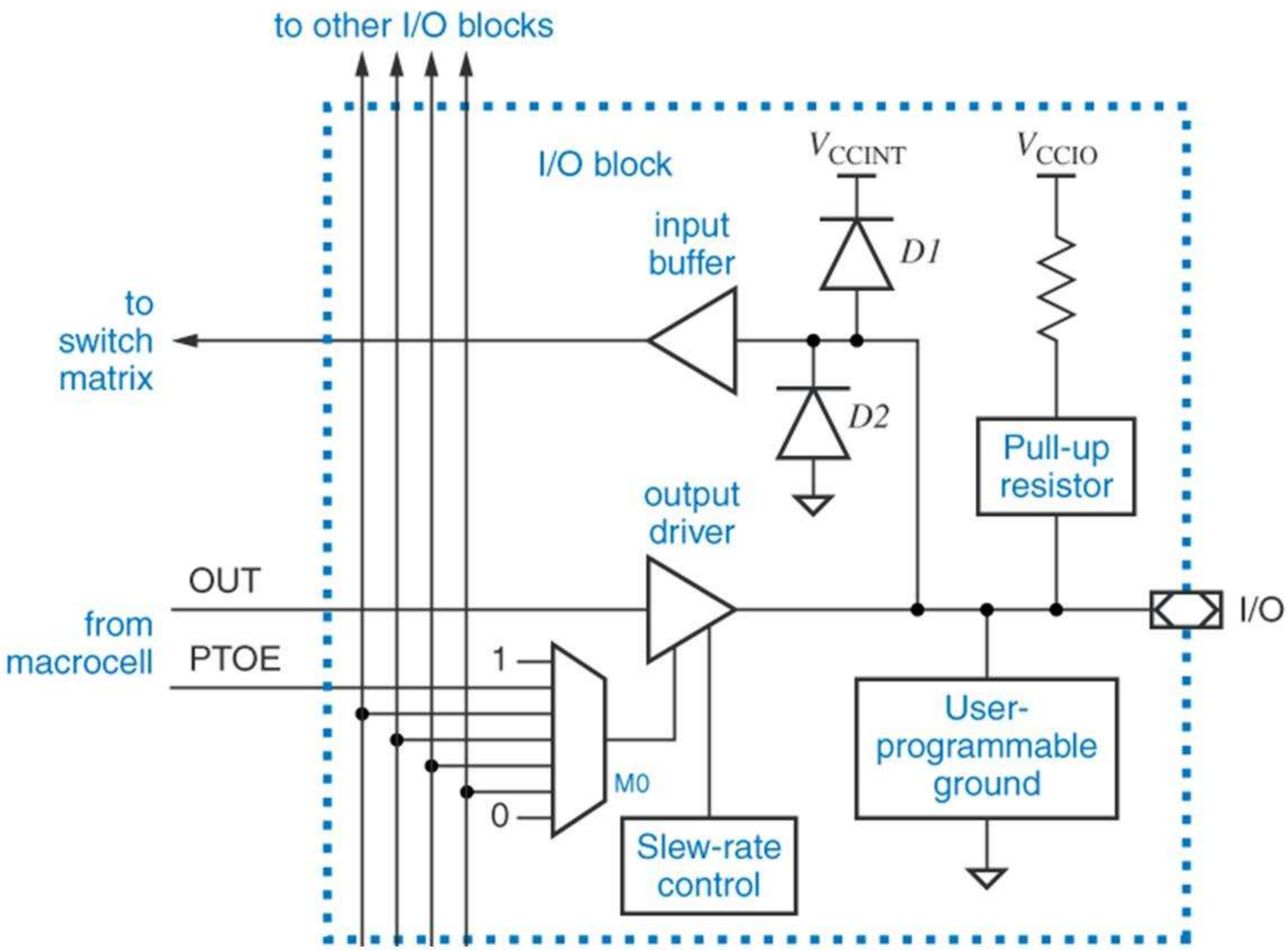


Macrocells



Product-term set
Global set/reset
M2
1
0
M1
S
D/T
Q
CLK
R
FF1
M3
to switch matrix
G4
G5
Global clocks
3
M4
Product-term reset
Global set/reset
M5
Product-term OE
OUT
PTOE
to I/O blocks

I/O Blocks

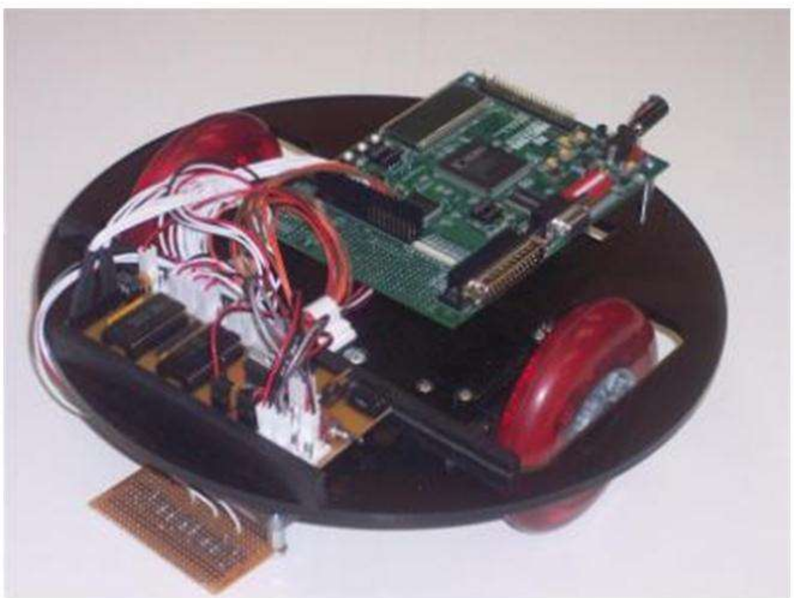


to other I/O blocks
I/O block
V_{CCINT}
V_{CCIO}
input buffer
D1
D2
to switch matrix
Pull-up resistor
output driver
OUT
from macrocell
PTOE
I/O
1
0
M0
User-programmable ground
Slew-rate control

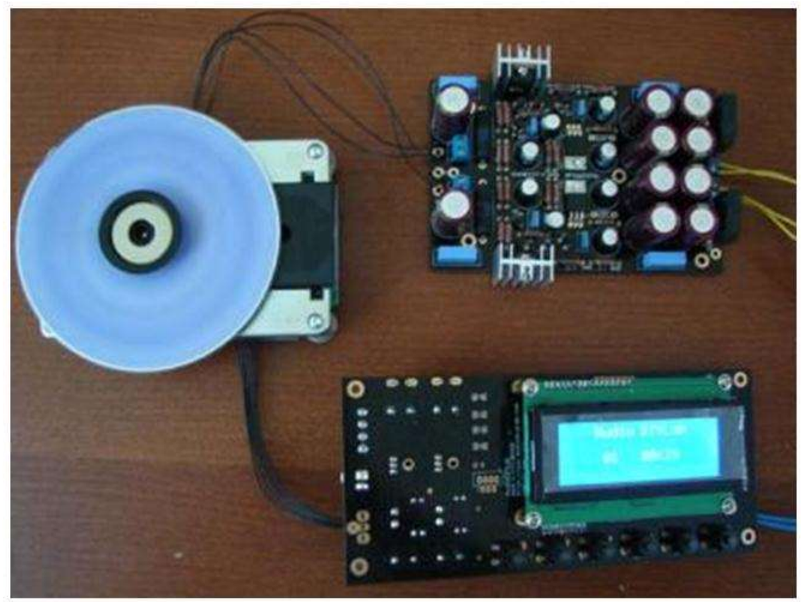
CPLDs - Applications

- CPLDs are still occasionally used for simple applications like address decoding, but more often contain high-performance control-logic or complex finite state machines.

- Robot controller



- CD/audio controller

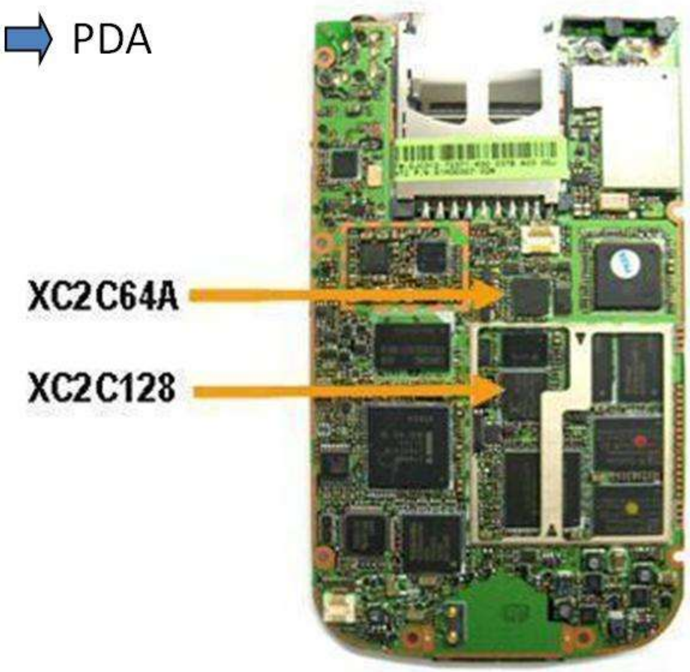


CPLDs - Applications

GPS Navigation Systems

- Hard disk controller
- GPIO interface
- Timing configuration

PDA



- LCD Timing Control
- GPIO Expansion
- Power Management

CPLDs - Applications

- GSM Phone

- Keypad scanner
- Logic consolidation

- Printer

- Controller and interface conversion
- Interface expansion
- Simple logic

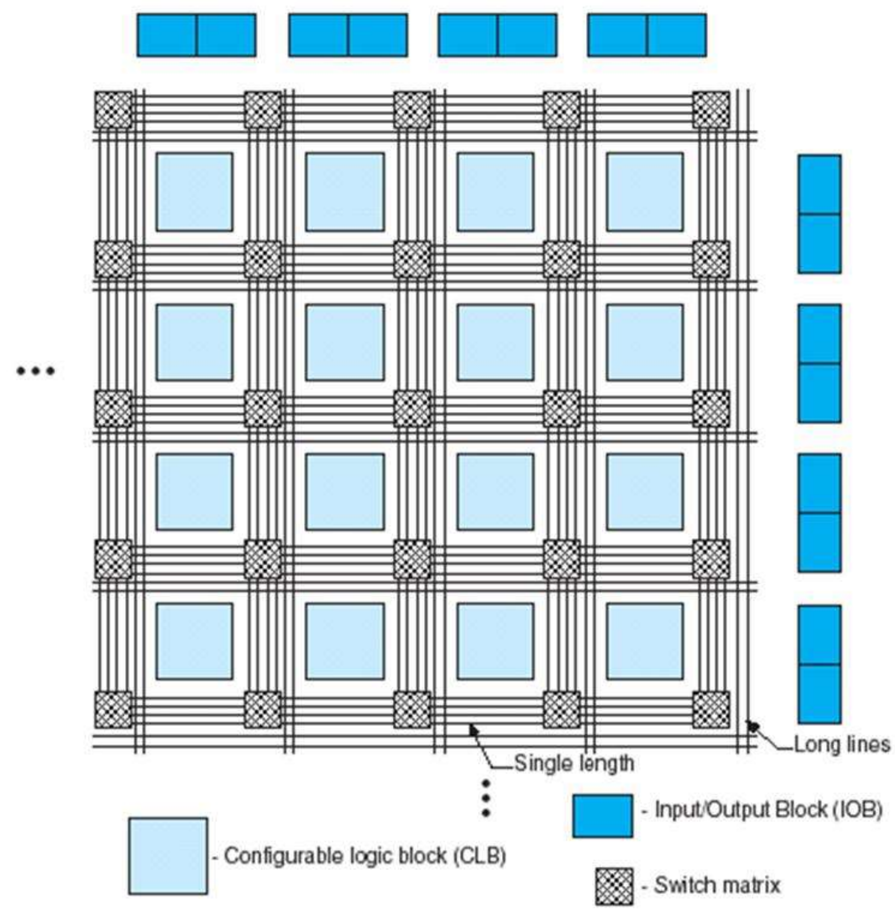
Field Programmable Gate Array (FPGA)

what does 'Field Programmable' mean?

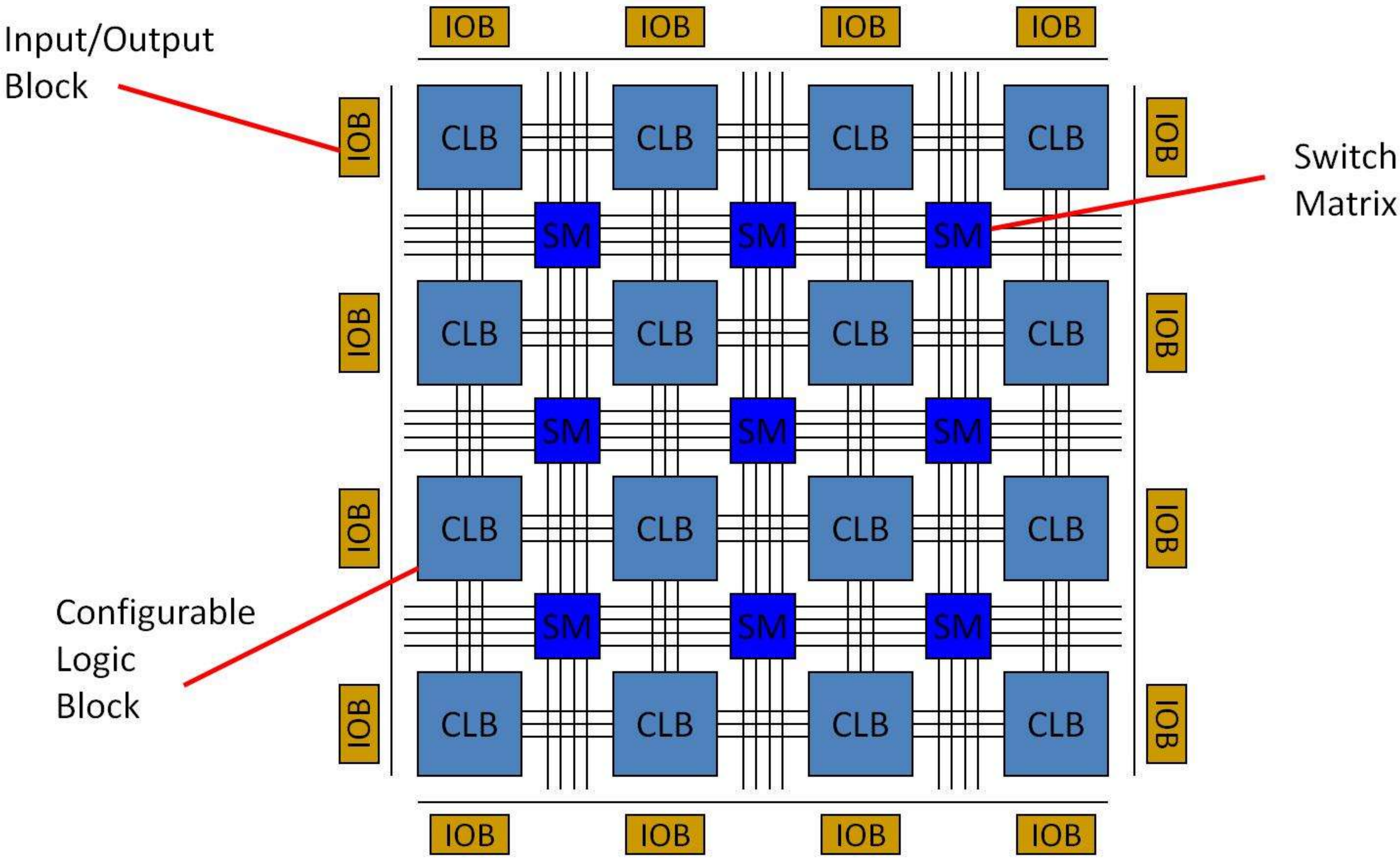
- Field Programmable means that the FPGA's function is defined by a user's program rather than by the manufacturer of the device.
- A typical integrated circuit performs a particular function defined at the time of manufacture. In contrast, the FPGA's function is defined by a program written by someone other than the device manufacturer.
- This user programmability gives the user access to complex integrated designs without the high engineering costs associated with application specific integrated circuits(ASIC).

FPGA

- Consists of CLBs and IOBs along with Switch matrix
- Substantial amounts of uncommitted combinational logic
- Pre-implemented flip-flops
- Programmable interconnections between these two



FPGA Structure



The structure of FPGA

<u>The basic elements of the FPGA structure:</u>

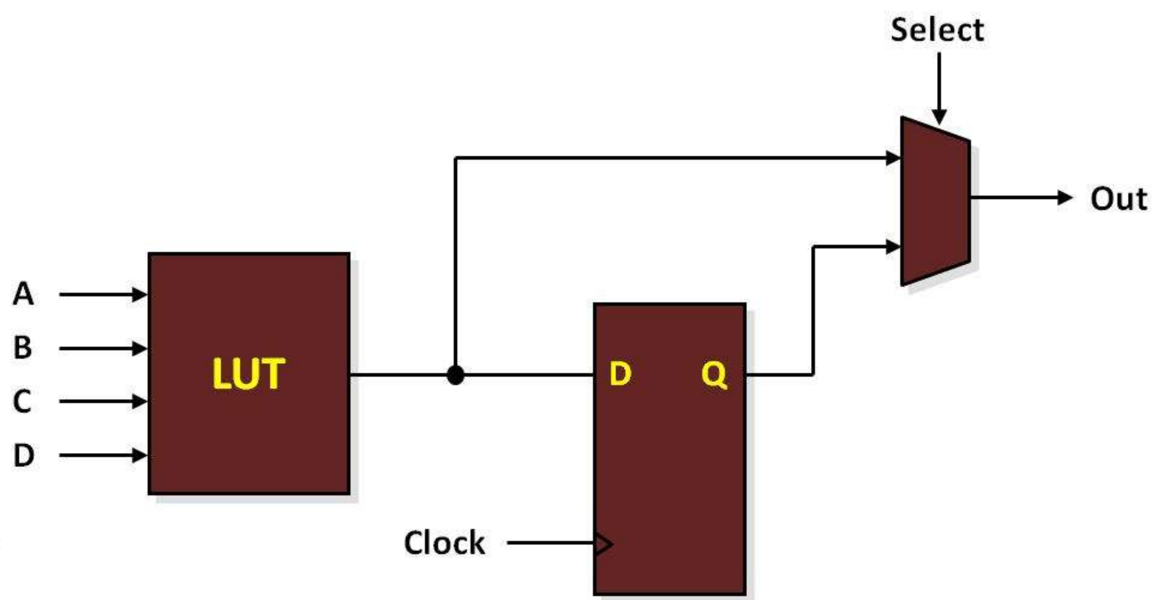
1-Programmable Logic blocks
Based on memories *(LUT – Lookup Table)* Xilinx
Based on multiplexers *(Multiplexers)* Actel
Based on PAL/PLA Altera

2-**Programmable Interconnection Resources**
Symmetrical FPGA-s
Row-based FPGA-s
Hierarchical **FPGA-s (***CPLD)*
3-Programmable Input-output cells (*I/O Cell)*
Possibilities for programming :

- **Input**
- **Output**
- **Bidirectional**

1-Basic Logic block

- LUT to implement combinatorial logic
- Register for sequential circuits
- Additional logic (not shown):
  - Carry logic for arithmetic functions
  - Expansion logic for functions requiring more than 4 inputs

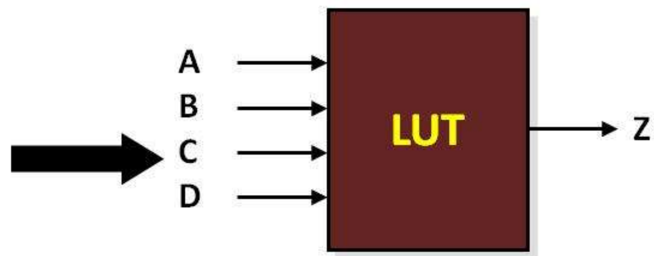


Look-Up Tables (LUT)

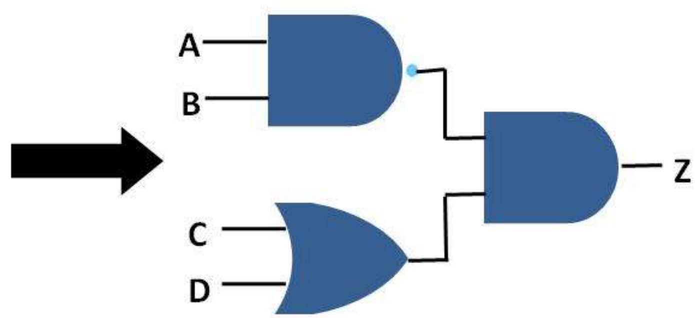
- Look-up table with N-inputs can be used to implement any combinatorial function of N inputs
- LUT is programmed with the truth-table

A	B	C	D	Z
0	0	0	0	**0**
0	0	0	1	**1**
0	0	1	0	**1**
0	0	1	1	**1**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**0**
1	0	0	1	**1**
1	0	1	0	**1**
1	0	1	1	**1**
1	1	0	0	**0**
1	1	0	1	**0**
1	1	1	0	**0**

Truth-table

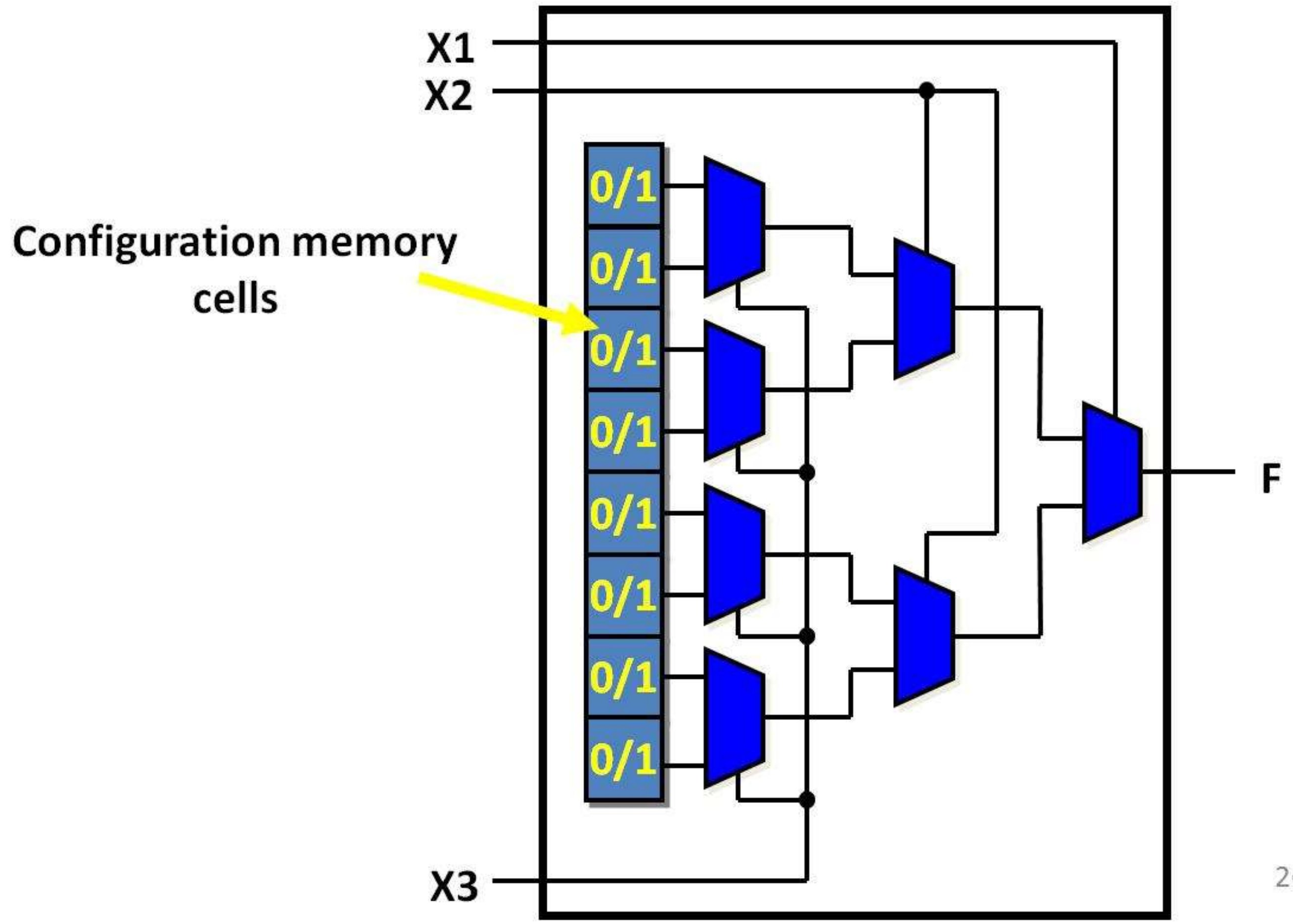


LUT implementation



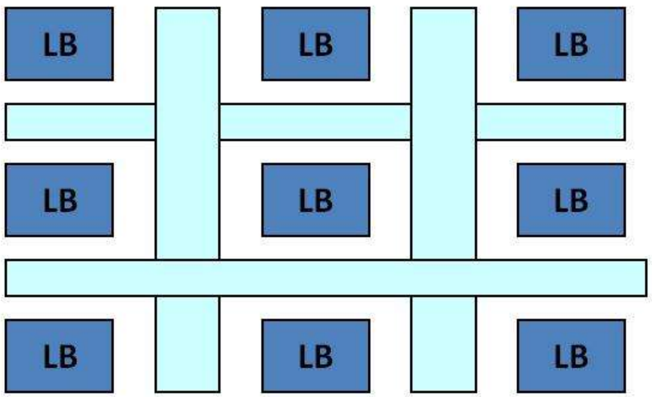
Gate implementation

multiplexers Implementation

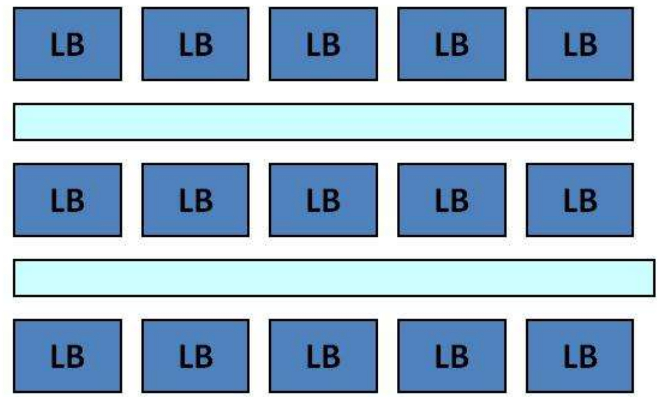


2-Programmable Interconnect

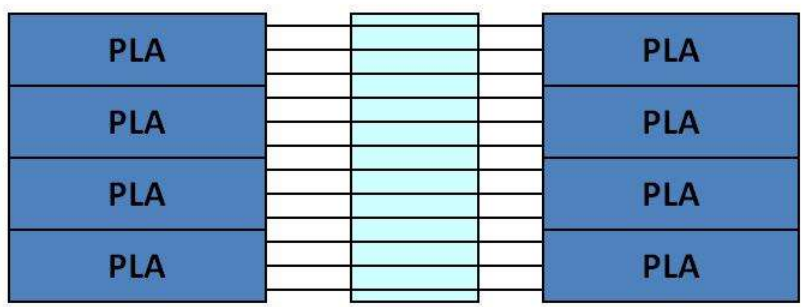
Symmetrical Array



Row-based

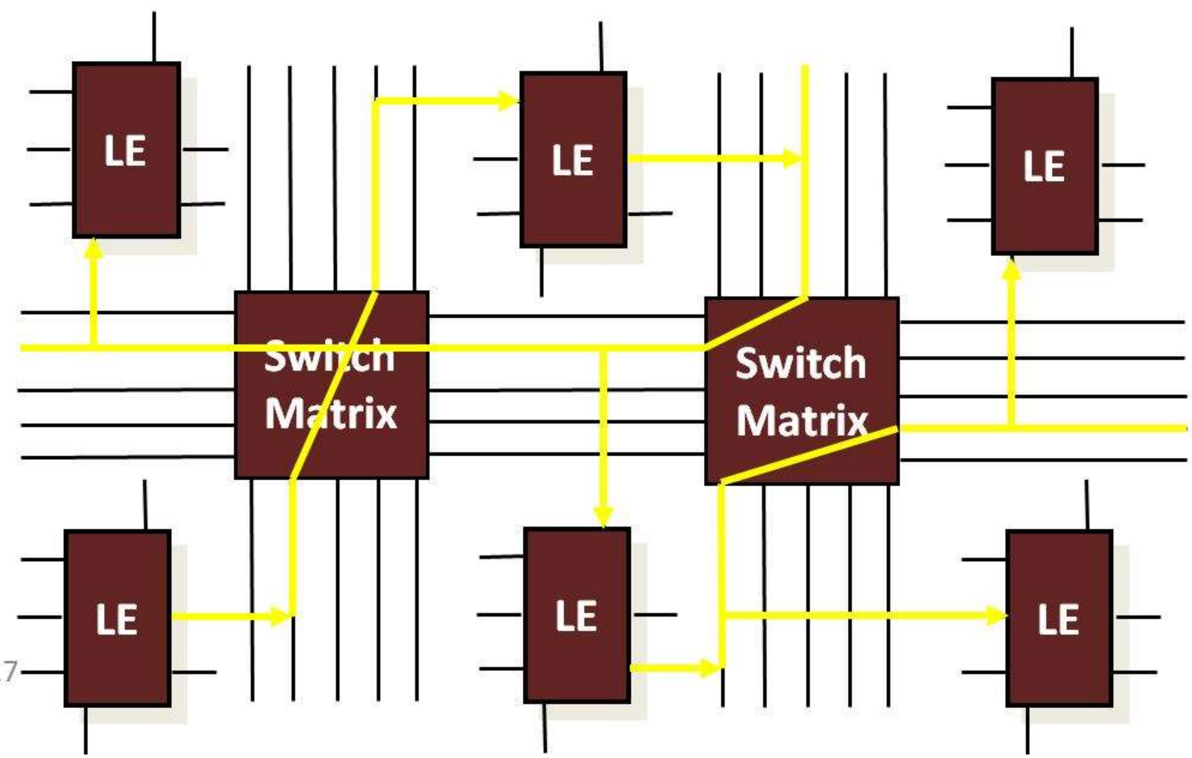


Hierarchical (CPLD)



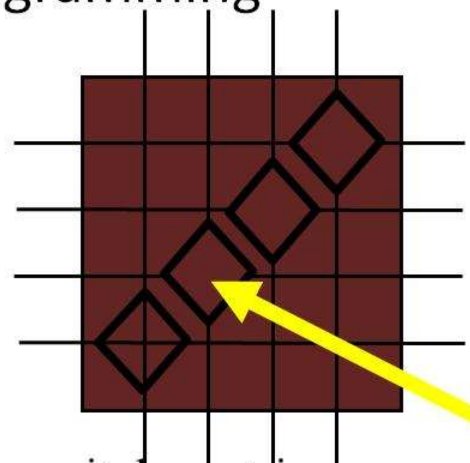
2-Programmable Interconnect

- Interconnect hierarchy
  - Horizontal and vertical lines of various lengths

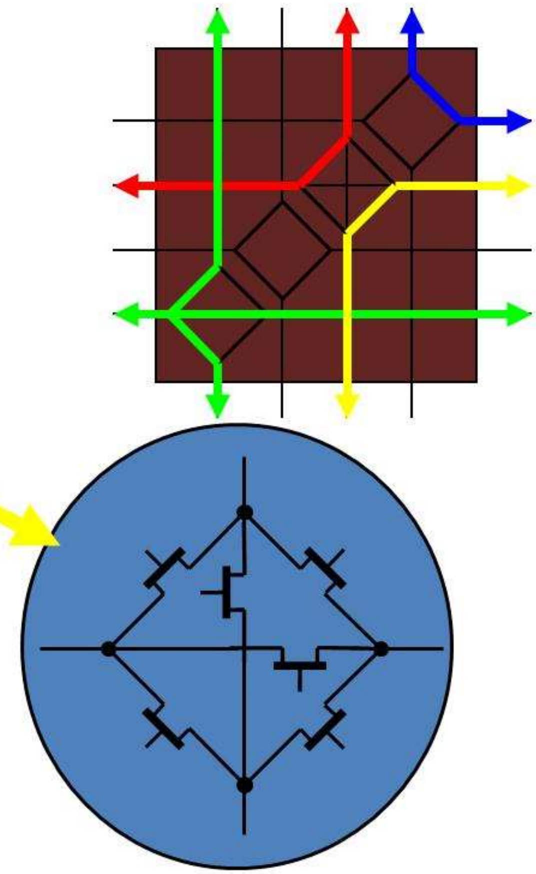


Switch Matrix Operation

Before Programming

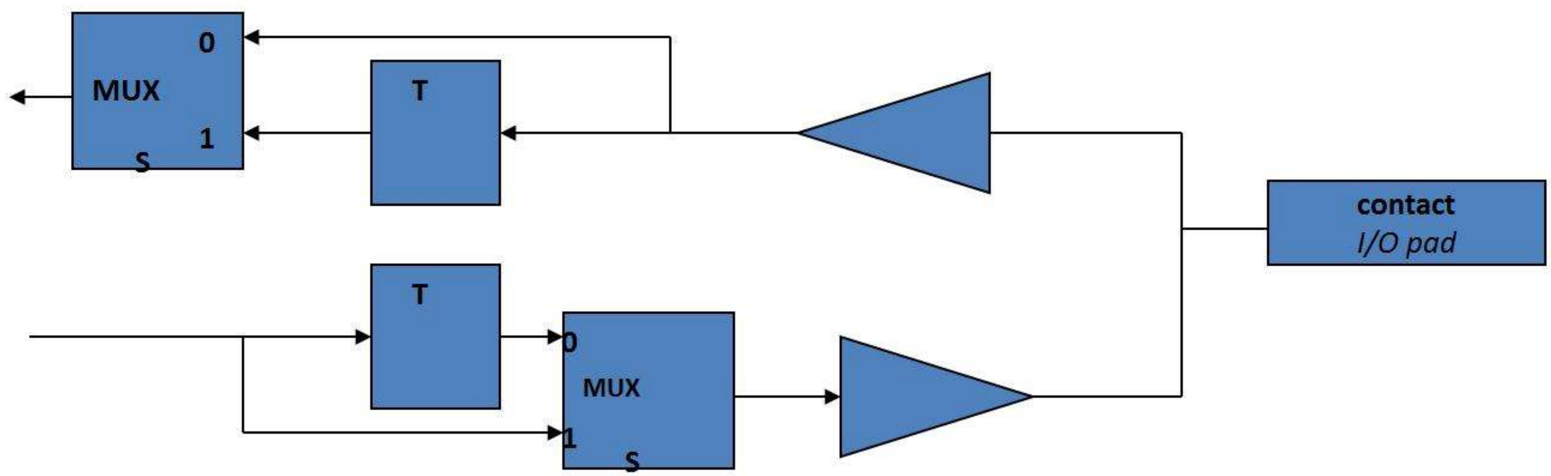


After Programming



- 6 pass transistors per switch matrix interconnect point
- Pass transistors act as programmable switches
- Pass transistor gates are driven by configuration memory cells

3- I/O cells



Other FPGA Building Blocks

- Clock distribution
- Embedded memory blocks
- Special purpose blocks:
  - DSP blocks (Hardware multipliers, adders and registers).
  - Embedded microprocessors/microcontrollers
  - High-speed serial transceivers

Configuration Storage Elements

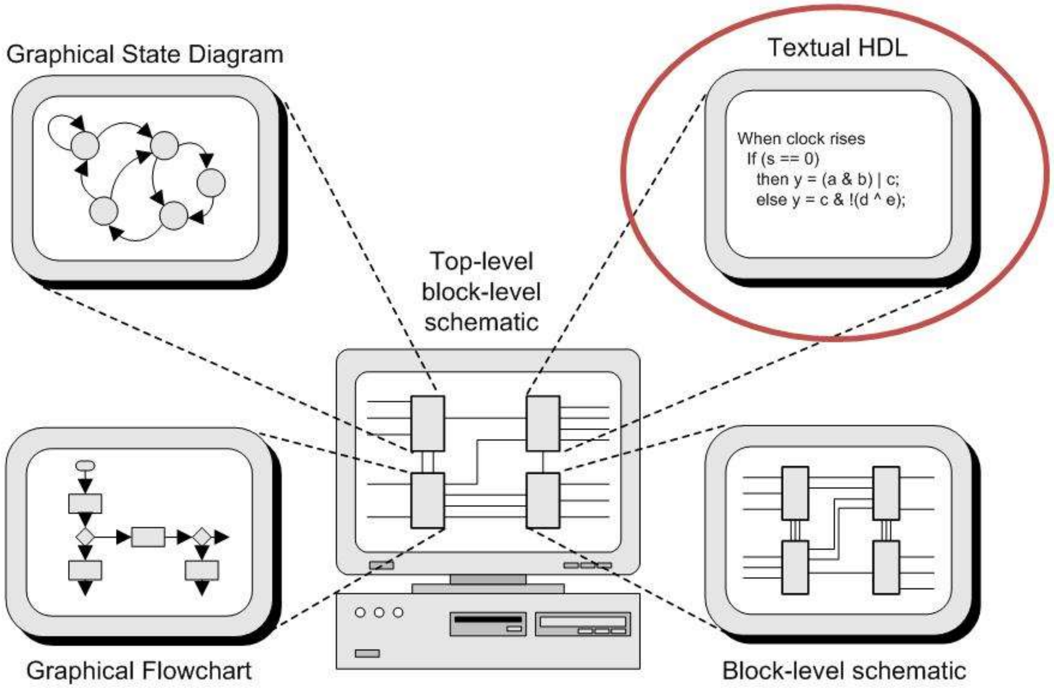
- Static Random Access Memory (SRAM)
  - each switch is a pass transistor controlled by the state of an SRAM bit
  - FPGA needs to be configured at power-on
- Flash Erasable Programmable ROM (Flash)
  - each switch is a floating-gate transistor that can be turned off by injecting charge onto its gate. FPGA itself holds the program
  - reprogrammable, even in-circuit
- Fusible Links ("Antifuse")
  - Forms a forms a low resistance path when electrically programmed
  - one-time programmable in special programming machine
  - radiation tolerant

FPGA Vendors & Device Families

- Xilinx
  - Virtex-II/Virtex-4: Feature-packed high-performance SRAM-based FPGA
  - Spartan 3: low-cost feature reduced version
  - CoolRunner: CPLDs
- Altera
  - Stratix/Stratix-II
    - High-performance SRAM-based FPGAs
  - Cyclone/Cyclone-II
    - Low-cost feature reduced version for cost-critical applications
  - MAX3000/7000 CPLDs
  - MAX-II: Flash-based FPGA
- Actel
  - Anti-fuse based FPGAs
    - Radiation tolerant
  - Flash-based FPGAs
- Lattice
  - Flash-based FPGAs
  - CPLDs (EEPROM)
- QuickLogic
  - ViaLink-based FPGAs

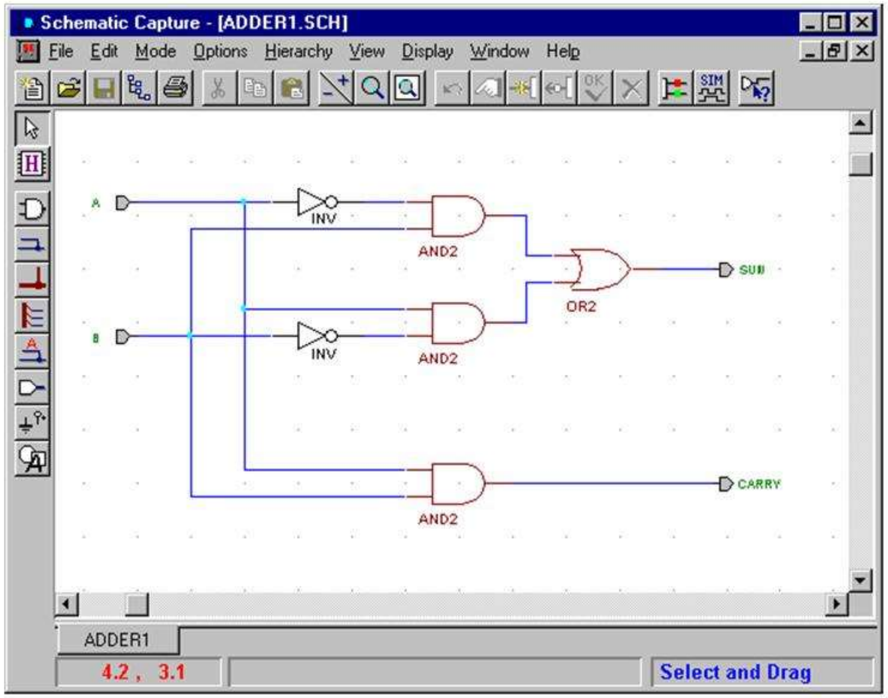
Designing Logic with FPGAs

- High level Description of Logic Design.
  - Graphical descriptions (Schematics)
  - Hardware Description Language (Textual)



Graphical descriptions

- ☺ Schematics are intuitive. They match our use of gate-level or block diagrams.
- ☺ Somewhat physical. They imply a physical implementation.
- ☹ Require a special tool (editor).
- ☹ Unless hierarchy is carefully designed, schematics can be confusing and difficult to follow.



What is HDL?

- Hardware Description Language(HDL)
- A software programming language used to model the intended operation of a piece of hardware

- Language describing hardware (Engineers call it FIRMWARE)
- Doesn't behave like "normal" programming language 'C/C++'
- Describe Logic as collection of Processes operating in Parallel
- Language Constructs for Synchronous Logic
- Compiler (Synthesis) Tools recognise certain code constructs and generates appropriate logic
- 2 Popular languages are VHDL , VERILOG

Why HDL?

- Why HDL?
  - Text-based design rather than Schematic design
    - ASIC complexity increase
    - faster time-to-market
  - Simulation
  - Logic Synthesis
  - Documentation

Shortly About the VHDL

- VHDL: VHSIC Hardware Description Language
- VHSIC: Very High Speed Integrated Circuit
- VHDL is sponsored by Dept. of Defense (USA) 1983
- A Formal Language for Specifying the Behavior and Structure of a Digital Circuit
- Allows Top-Down Design
- First IEEE standard: IEEE 1076-1987. (VHDL'87)
- Second IEEE standard: IEEE 1076-1993. (VHDL'93)
- Commercial simulation and synthesis tools based on IEEE 1076-1993 is available in 1996
- IEEE 1076.3 1076.4 (VITAL) 1997 (VHDL Initialtive Towards ASIC Libraries)
- VHDL-AMS (analog mixed signal) 1997
- Most major CAD frameworks now support both VHDL and Verilog.

Shortly About the Verilog

- Verifying Logic
- Originally designed in 1983/1984 as a proprietary verification/simulation
- Verilog-XL simulator in 1986
- Synopsis Synthesis Tool in 1988
- Afraid of losing market share, Cadence opened Verilog to the public in 1990.
- An IEEE working group was established in 1993, and ratified IEEE Standard 1394 (Verilog) in 1995.
- Verilog is language of choice of Silicon Valley companies, initially because of high-quality tool support and its similarity to C-language syntax.
- Last revision in 2001
  - IEEE Std-1364-2001
- Ongoing work for adding
  - Mixed-signal constructs: Verilog-AMS
  - System-level constructs: SystemVerilog

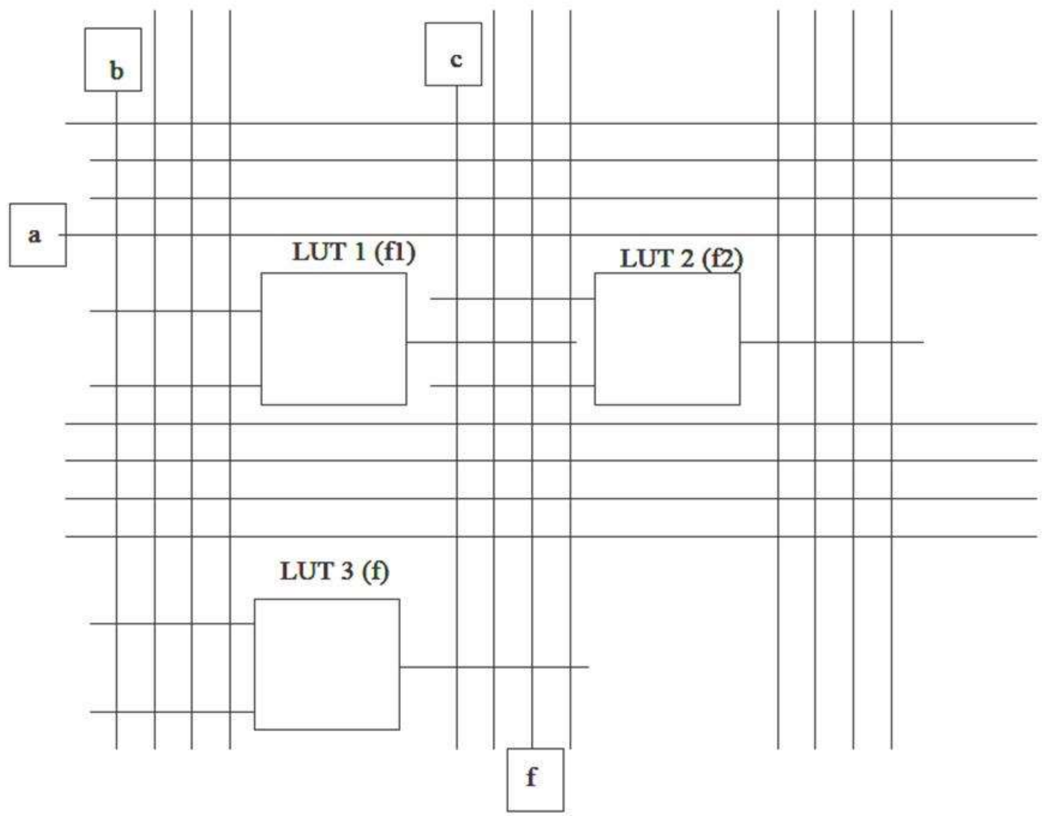
VHDL vs. Verilog

VHDL	Verilog
All abstraction levels	All abstraction levels
Most FPGA design in VHDL	Most ASIC design in Verilog
Based on Pascal language	Based on C language
Lots of data types	Few data types
User-defined package & library	No user-defined packages
Full design parameterization	Simple parameterization
Easier to handle large designs	
USA(IBM, TI, AT&T, INTEL), Europe, Korea	USA (Silicon Valley), Japan

FPGA applications:-

i. DSP
ii. Aerospace
iii. Defense system
iv. ASIC Prototyping
v. Medical Imaging
vi. Computer vision
vii. Speech Recognition
viii. Cryptography
ix. Bioinformatics
x. And others............

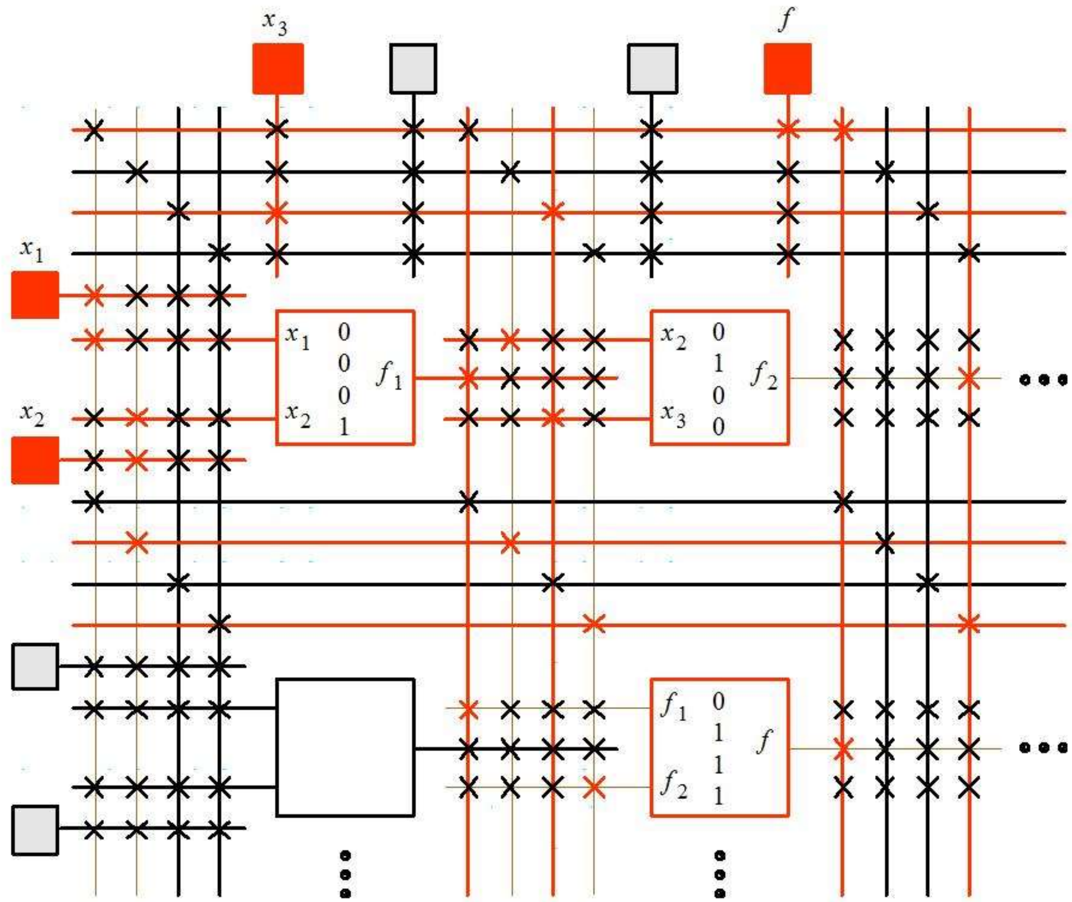
Q1: you are asked to program an FPGA, whose LUTs and inter-connection wires are shown in **Figure below.** The function to be implemented is f = f1 _ f2, where f1 = a + b and f2 = a + c. LUT 1 should implement f1. LUT 2 should implement f2 and LUT 3 should implement f1-f2. The horizontally and vertically placed interconnection wires are fabricated in different planes. In order to depict a connection between these wires at a cross-point, place a cross-mark (X). The inputs a; b; c and the output f have already been connected to the "input-output pads



- Example FPGA

  – Use an FPGA with 2 input LUTS to implement the function f $= x_1x_2 + x_2'x_3$

    - $f_1 = x_1x_2$
    - $f_2 = x_2'x_3$
    - $f = f_1 + f_2$



- Another Example FPGA

  - Use an FPGA with 2 input LUTS to implement the function f $= x_1x_3x_6' + x_1x_4x_5x_6' + x_2x_3x_7 + x_2x_4x_5x_7$

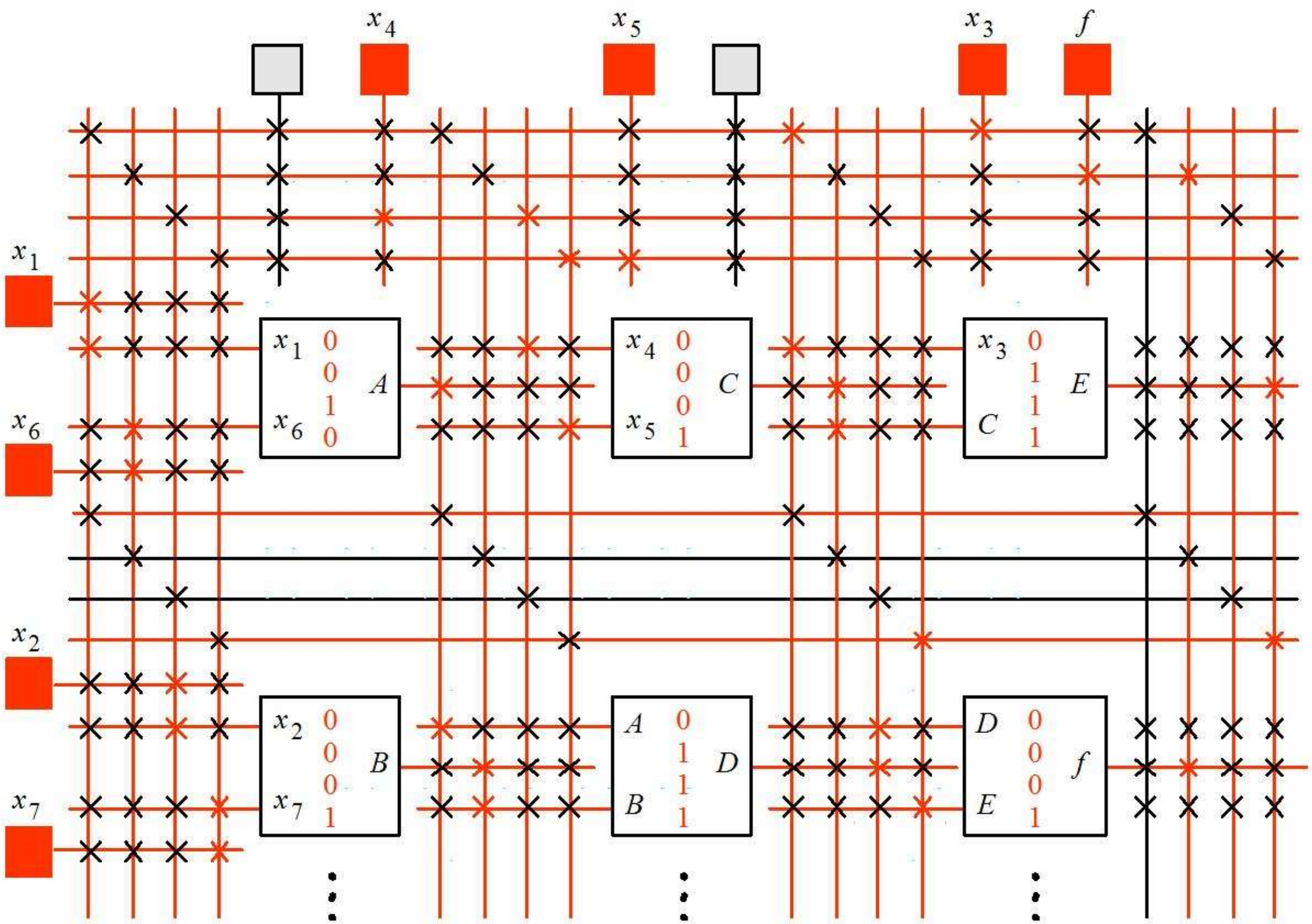
    - Fan-in of expression is too large for FPGA (this was simple to do in a CPLD)

    - Factor f to get sub-expressions with max fan-in = 2
      - $f = x_1x_6'(x_3 + x_4x_5) + x_2x_7(x_3 + x_4x_5)$
        $= (x_1x_6' + x_2x_7)(x_3 + x_4x_5)$

    - Could use Shannon's expansion instead
      - Goal is to build expressions out of 2-input LUTs

- FPGA Implementation

$f = (x_1x_6' + x_2x_7)(x_3 + x_4x_5)$

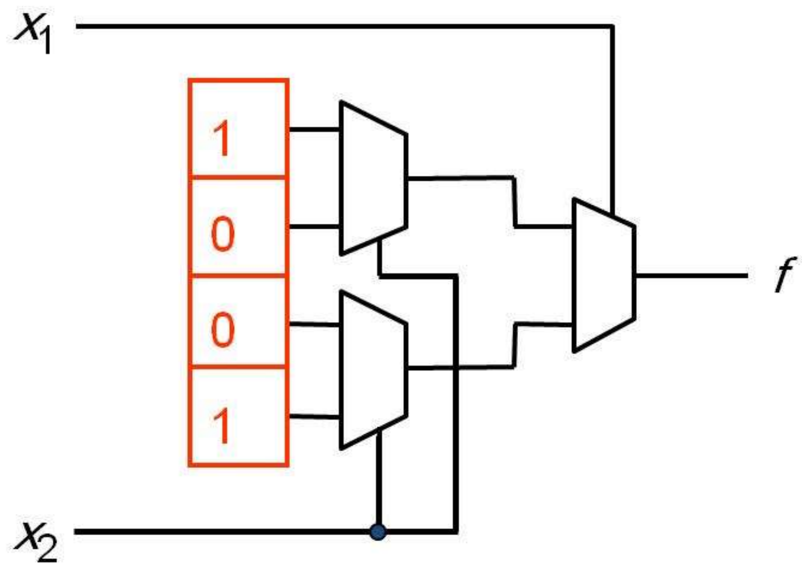


- Example 2 Input LUT

x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	1

$f = x_1'x_2' + x_1x_2$, or using Shannon's expansion:

$$f = x_1'(x_2') + x_1(x_2)$$
$$= x_1'(x_2'(1) + x_2(0)) + x_1(x_2'(0) + x_2(1))$$



- 3 Input LUT
  - 7 2x1 MUXes and 8 storage cells are required
  - Commercial LUTs have 4-5 inputs, and 16-32 storage cells