# Circuit Design with VHDL

## Chapters 4, 5: PLDs, Introduction to VHDL

Instructor: Ali Jahanian

# Outline

- I-CIRCUIT DESIGN

  1 Introduction

  1.1 About VHDL

  1.2 Design Flow

  1.3 EDA Tools

  1.4 Translation of VHDL Code into a Circuit

  1.5 Design Examples

- Appendix A: Programmable Logic Devices

# About VHDL

- VHDL is a language for describing digital hardware used by industry worldwide
  - **VHDL is an acronym for**

Pedroni_Chapter

**VHSIC (Very High Speed Integrated Circuit)**
**Hardware Description Language**

# Hardware Language Requirements

- Main features
  - Concurrency
  - Hardware Semantics
- Highlights of modern HDL
  - Encapsulate the concepts of entity, connectivity, concurrency, and timing.
  - Consist of constructs for structural implementation
  - Incorporate constructs for behavioral description, gate level and RT level.
  - Consist of constructs to support hierarchical design process

# About VHDL …

- **Genesis of VHDL**
  - State of the art circa 1980
    - Multiple design entry methods and hardware description languages in use
    - No or limited portability of designs between CAD tools from different vendors
    - Objective: shortening the time from a design concept to implementation from 18 months to 6 months

# About VHDL …

- **A Brief History of VHDL**
  - June 1981: Woods Hole Workshop
  - July 1983: contract awarded to develop VHDL
    - Intermetrics
    - IBM
    - Texas Instruments
  - August 1985: VHDL Version 7.2 released
  - December 1987: VHDL became IEEE Standard 1076-1987 and in 1988 an ANSI standard
  - Four versions of VHDL:
    - IEEE-1076 1987
    - IEEE-1076 1993 **(most commonly supported by CAD tools)**
    - IEEE-1076 2000 (minor changes)
    - IEEE-1076 2002 (minor changes)

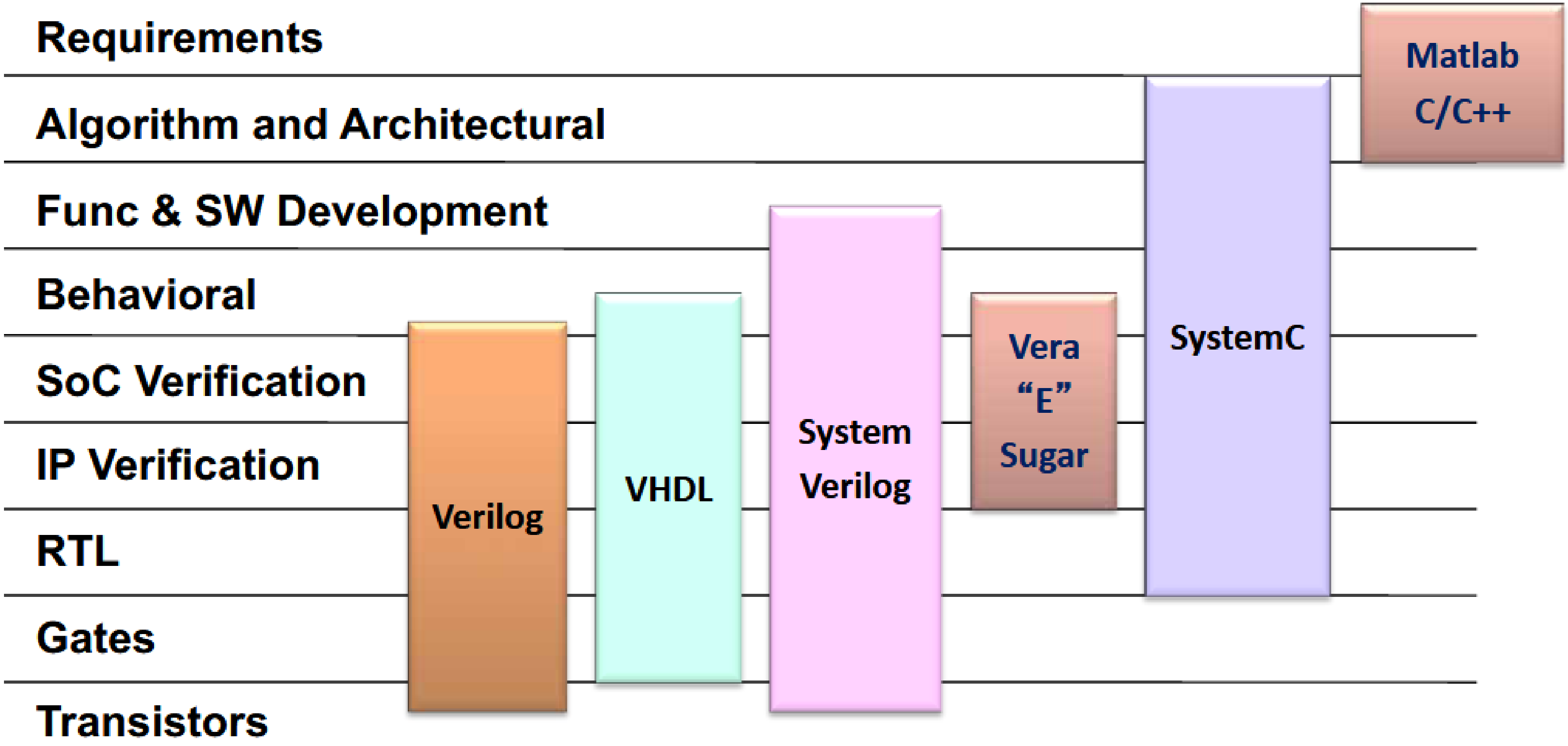
# VHDL Competitor

- **Verilog**
  - Essentially identical in function to VHDL
    - No generate statement
  - Simpler and syntactically different
    - C-like
  - Gateway Design Automation Co., 1983
  - Early de facto standard for ASIC programming
  - Open Verilog International Standard

# VHDL vs. Verilog

| VHDL | Verilog |
|---|---|
| Government Developed | Commercially Developed |
| Ada Based | C Based |
| Difficult to Learn | Easier to Learn |
| More Powerful | Less Powerful |

# HDLs Usage



Requirements
Algorithm and Architectural
Func & SW Development
Behavioral
SoC Verification
IP Verification
RTL
Gates
Transistors
Verilog
VHDL
System Verilog
Vera
"E"
Sugar
SystemC
Matlab
C/C++

# Features of VHDL and Verilog

- Technology/vendor independent
- Portable
- Reusable

# Example 1- Full Adder

```
ENTITY fa IS
    PORT (    a, b    : IN        bit;
              cin     : IN        bit;
              s       : OUT  bit;
              cout    : OUT  bit);
END fa;
ARCHITECTURE dataflow OF fa IS
    S      <= a XOR b XOR cin;

    Cout<= (a AND b) OR (a AND cin) OR (b AND cin);
END fa;
```

- Main structure
- Ports
- Entity & Architecture
- Concurrency
- Semantic of <=

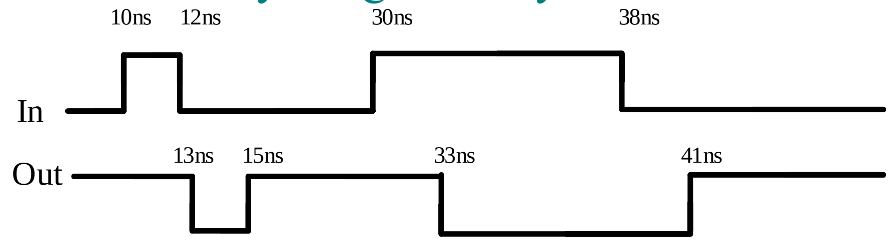
# Example 1- Full Adder

```
ENTITY fa IS
    PORT (     a, b      : IN              bit;
               cin       : IN              bit;
               s         : OUT   bit;
               cout      : OUT   bit);
END fa;
ARCHITECTURE dataflow OF fa IS
    S     <= a XOR b XOR cin AFTER 10 ns;
    Cout<= (a AND b) OR (a AND cin) OR (b AND cin) AFTER 15 ns;
END fa;
```

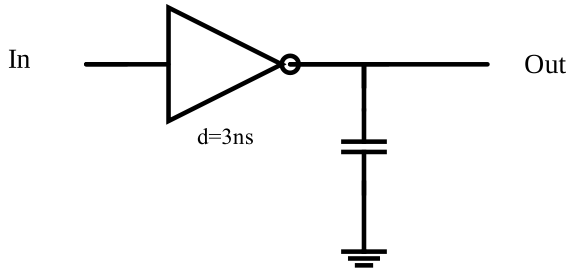
- Delay
- Delay mechanism
- Source of delay
- Usage of Delay in HDLs
- Synthesis & delay

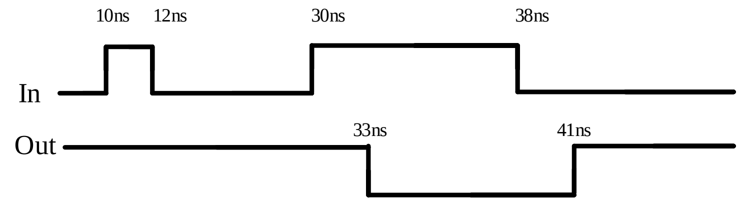
# Delay Mechanisms in VHDL

- Transport delay
  - Suitable for transmission line simulation
    - The output value of a gate is shifted by the gate delay value




- Inertial delay
  - Suitable for normal capacitive wires
    - The output value of a gate is shifted by the gate delay value but low width pulses are rejected



# Delay Mechanisms in VHDL

- Transport delay
    - y <= TRANSPORT NOT x AFTER 3 ns

- Inertial delay
    - y <= REJECT 2 ns INERTIAL NOT x AFTER 3 ns;
        - Gate delay = 3ns but the pulses whose width < 2ns are rejected.
    - Y <= INERTIAL NOT x AFTER 3 ns;
        - Gate delay = 3ns but the pulses whose width < 3ns are rejected.
    - y <= NOT x AFTER 3 ns;
        - Gate delay = 3ns but the pulses whose width < 3ns are rejected.

# Digital Circuit Simulation

- Oblivious simulation
  - All the circuit is evaluated at T=0.
  - Time is increased by a time step (fixed or dynamic).
  - All the circuit is evaluated in each time step.
- Event-Driven simulation
  - All the circuit is evaluated at T=0.
  - Time is jumped to the next event.
  - Only the modules are evaluated whose inputs are changed.
- Cycle-based simulation
  - All the circuit is evaluated at T=0.
  - Time is jumped to the next clock edge.
  - Only the modules are evaluated whose inputs are changed.

# Example of Event Driven Simulation

- Event Driven algorithm:
  - Evaluate all the elements at T=0.
  - While there is transaction LOOP
    - Evaluate the gates that their inputs have events and generate new transaction.
  - END LOOP
  - Definition of event and transactions
    - Event is a transaction that change the value of signal at firing time

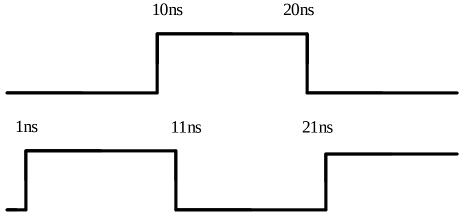
**a <= ‘0’, ‘1’ AFTER 10 ns, ‘0’ AFTER 20 ns;**
**b <= NOT a AFTER 1 ns;**

**Transaction list:**

**a: (‘0’, 0ns) (‘1’, 10ns) (‘0’, 20ns)**

**b: (‘1’, 1ns) (‘0’, 11ns) (‘1’, 21ns)**

T=0



# Delta delay δ

- Delta-delay shows the order of events in one instant.
- Note that delta delay has not physical dimension and just an scalar value.

```
a <= '0', '1' AFTER 10 ns, '0' AFTER 20 ns;
b <= NOT a;
c <= NOT b;
```

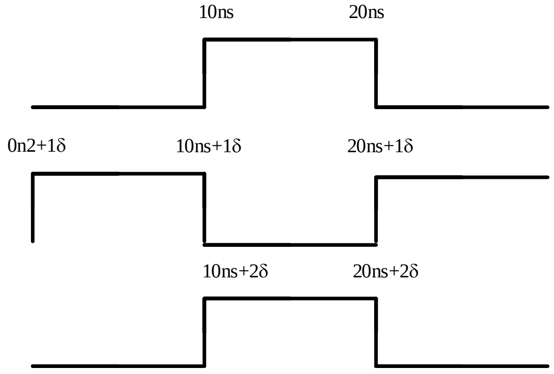
**Transaction list:**

a: ('0', 0ns) ('1', 10ns) ('0', 20ns)

b: ('1', 0ns+1δ) ('0', 10ns+1δ) ('1', 20ns+1δ)

c: ('0', 0ns+2δ) ('1', 10ns+2δ) ('0', 20ns+2δ)

T=0



# Design Flow

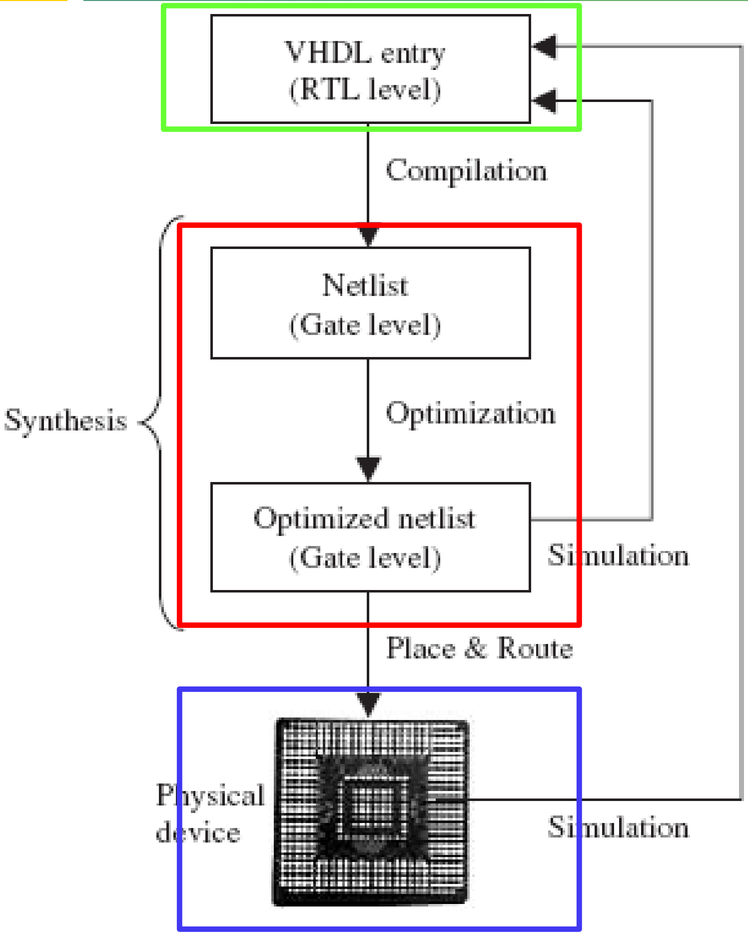
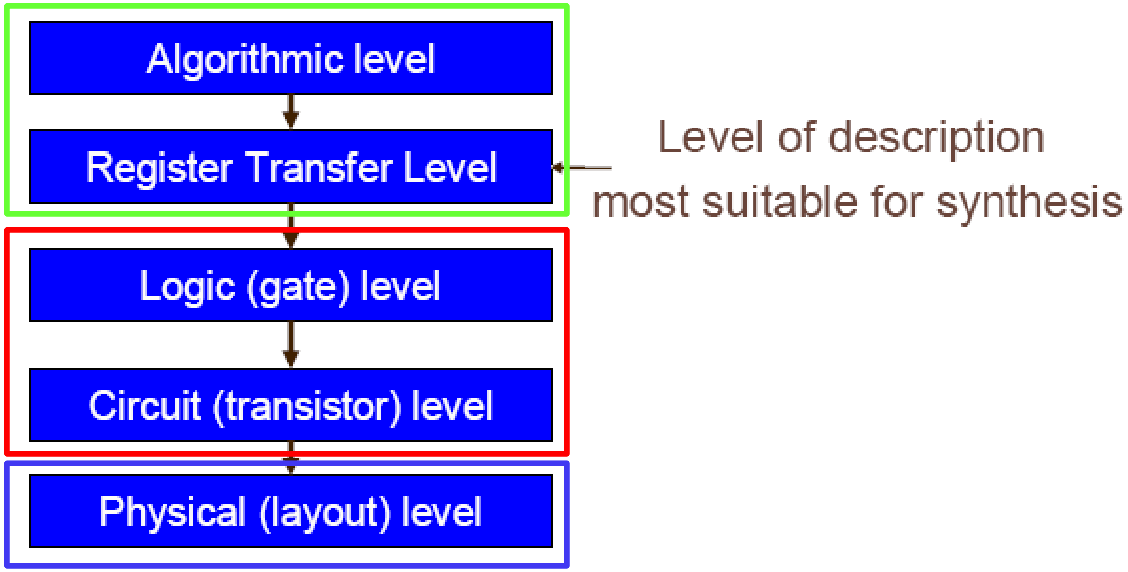
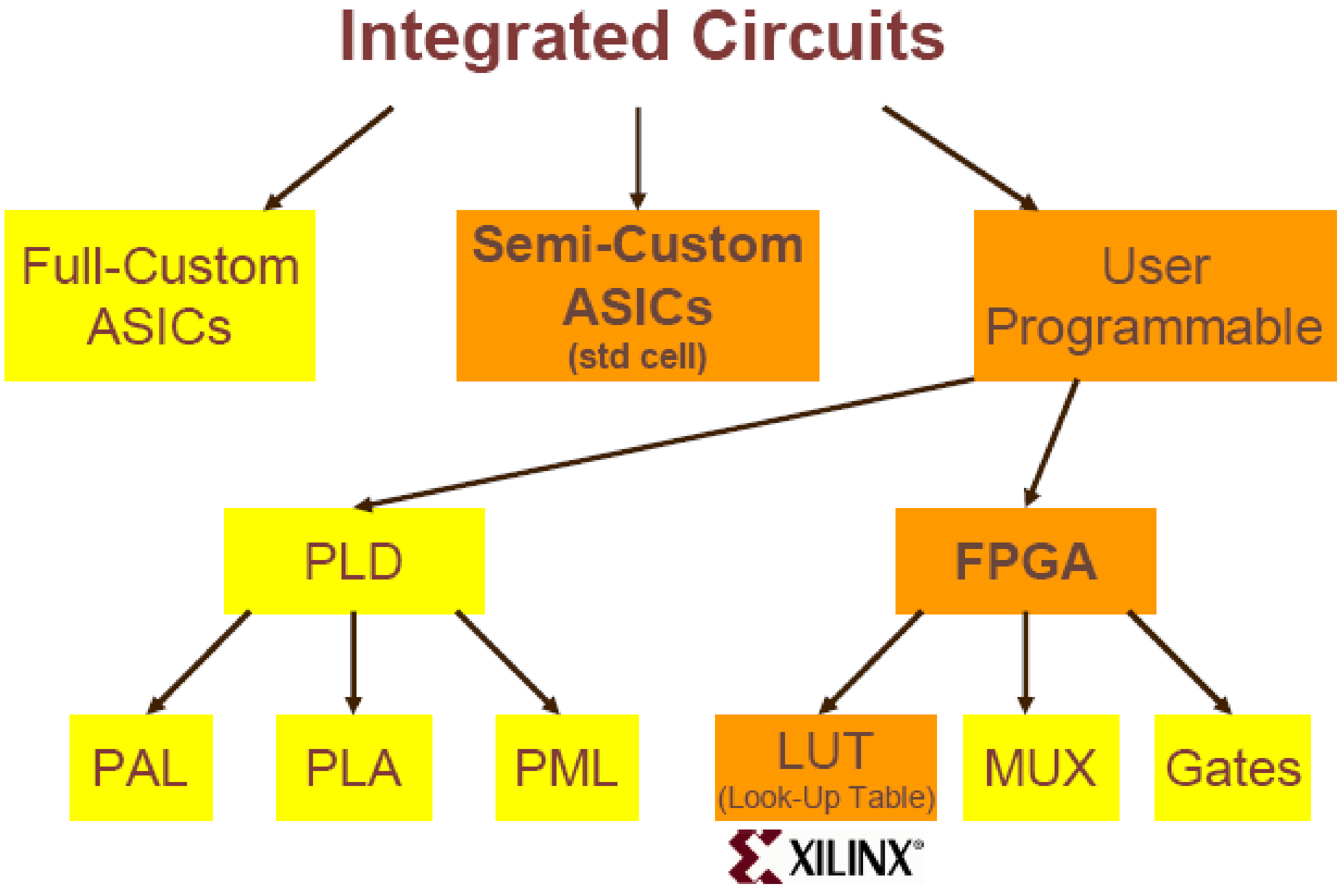



**Figure 1.1**
Summary of VHDL design flow.



# Physical device

- World of Integrated Circuits

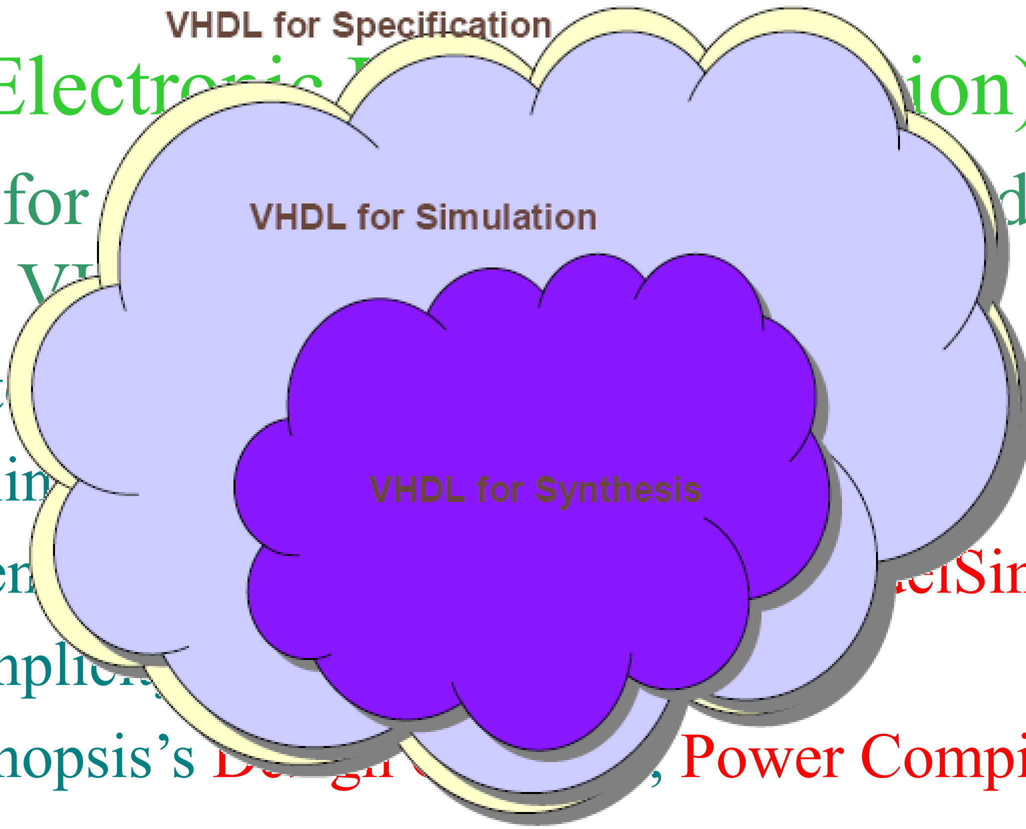


# ASIC Vs. FPGA

| ASIC<br>Application Specific Integrated Circuit | FPGA<br>Field Programmable Gate Array |
|---|---|
| • designs must be sent for expensive and time consuming fabrication in semiconductor foundry | • bought off the shelf and reconfigured by designers themselves |
| • designed all the way from behavioral description to physical layout | • no physical layout design; design ends with a bitstream used to configure a device |

# EDA Tools

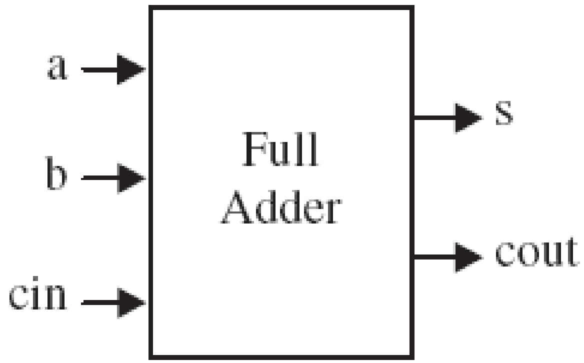
- EDA (Electr ion)
  - tools for d simulation using V
    - Alt
    - Xilin
    - Men elSim
    - Synplic
    - Synopsis's D , Power Compiler, …



# EDA Tools …

- Appendix B: Xilinx ISE & ModelSim Tutorial
- Appendix C: Altera MaxPlus II & Advanced Synthesis Software Tutorial
- Appendix D: Altera Quartus II Tutorial

# Translation of VHDL Code into a Circuit




| a | b | cin | s | cout |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

**Figure 1.2**
Full-adder diagram and truth table.

# Translation of VHDL Code into a Circuit …

```
ENTITY full_adder IS
PORT (a, b, cin: IN BIT;
s, cout: OUT BIT);
END full_adder;
-------------------------------------
ARCHITECTURE dataflow OF full_adder IS
BEGIN
s <= a XOR b XOR cin;
cout <= (a AND b) OR (a AND cin) OR (b AND cin);
END dataflow;
```

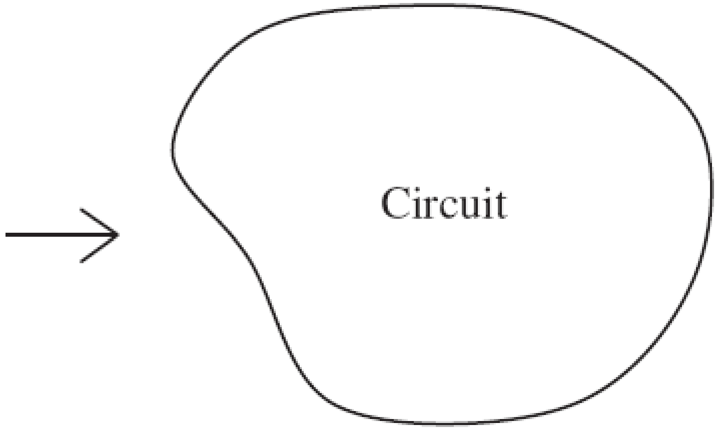



**Figure 1.3**
Example of VHDL code for the full-adder unit of figure 1.2.

# Translation of VHDL Code into a Circuit …

- compiler/optimizer
- target technology

# Translation of VHDL Code into a Circuit …

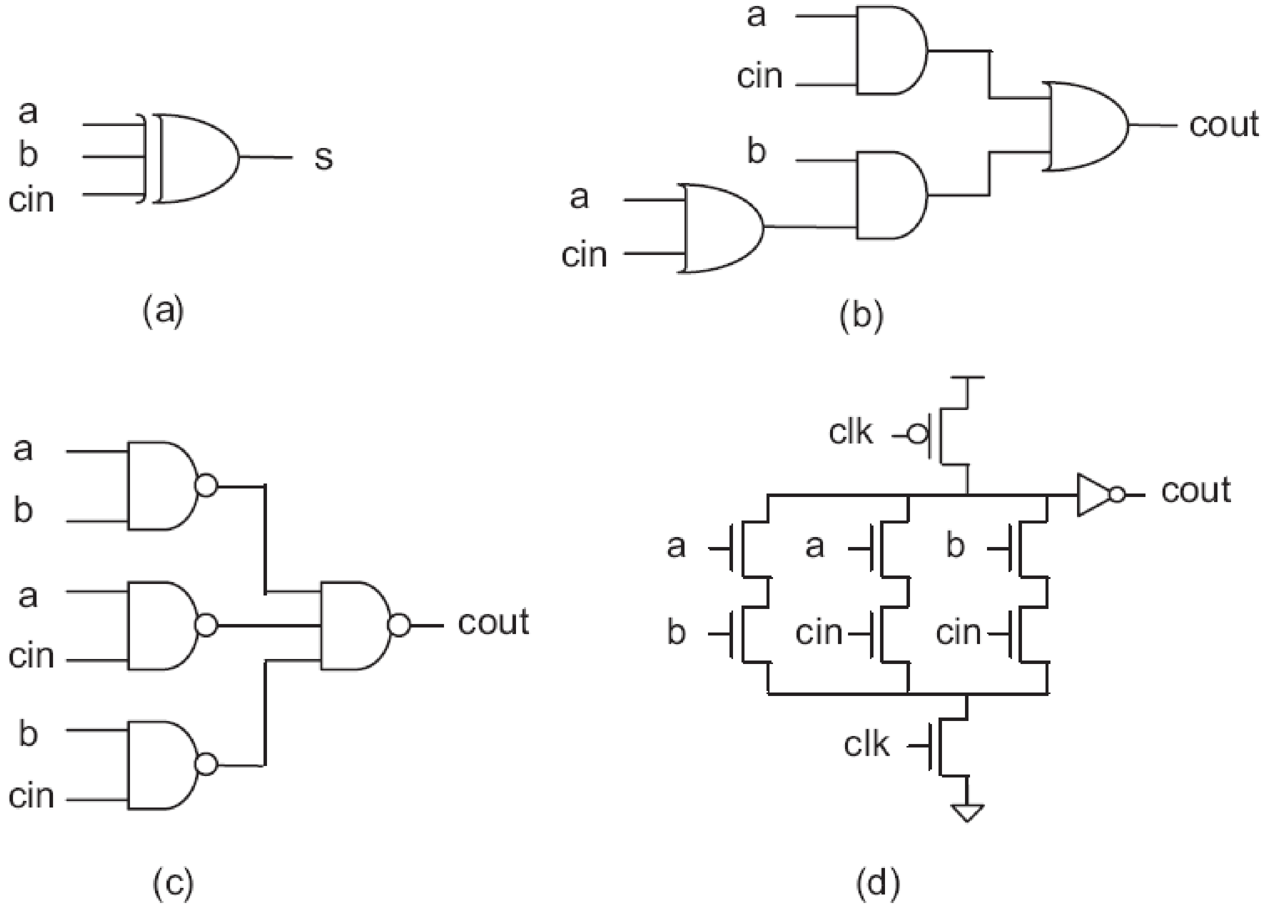



**Figure 1.4**
Examples of possible circuits obtained from the full-adder VHDL code of figure 1.3.

# Appendix A: Programmable Logic Devices

| | | |
|---|---|---|
| PLDs | Simple PLD (SPLD) | PAL<br>PLA<br>Registered PAL/PLA<br>GAL |
| | Complex PLD (CPLD) | |
| | FPGA | |

# PAL architecture

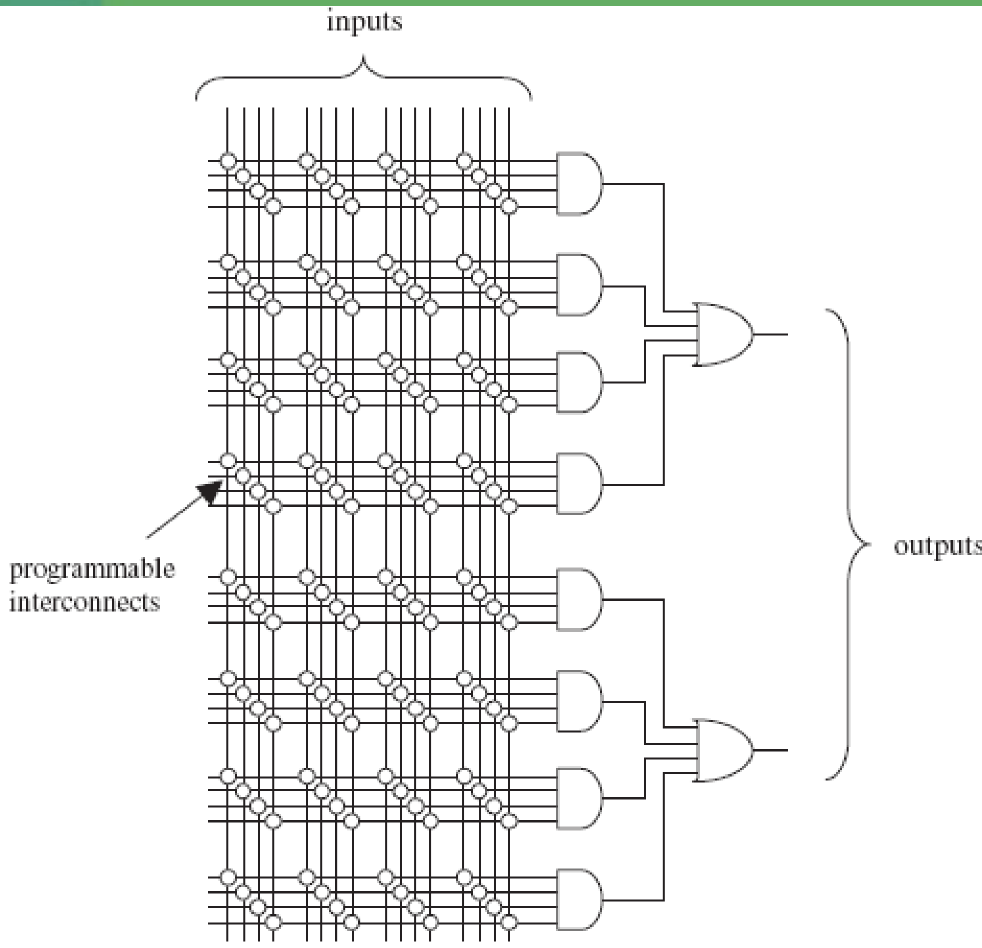



Figure A1
Illustration of PAL architecture.

# PLA architecture

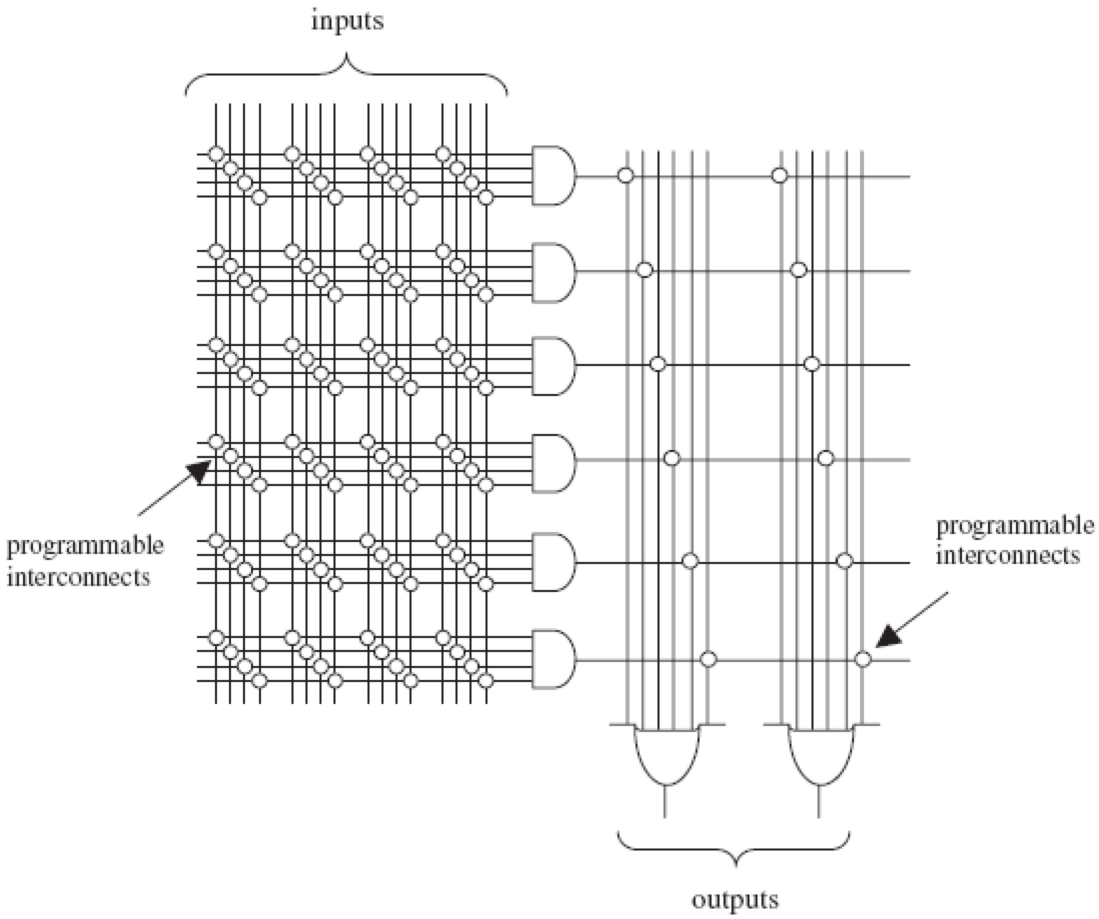



**Figure A2**
Illustration of PLA architecture.

# GAL 16V8 chip (Generic PAL)

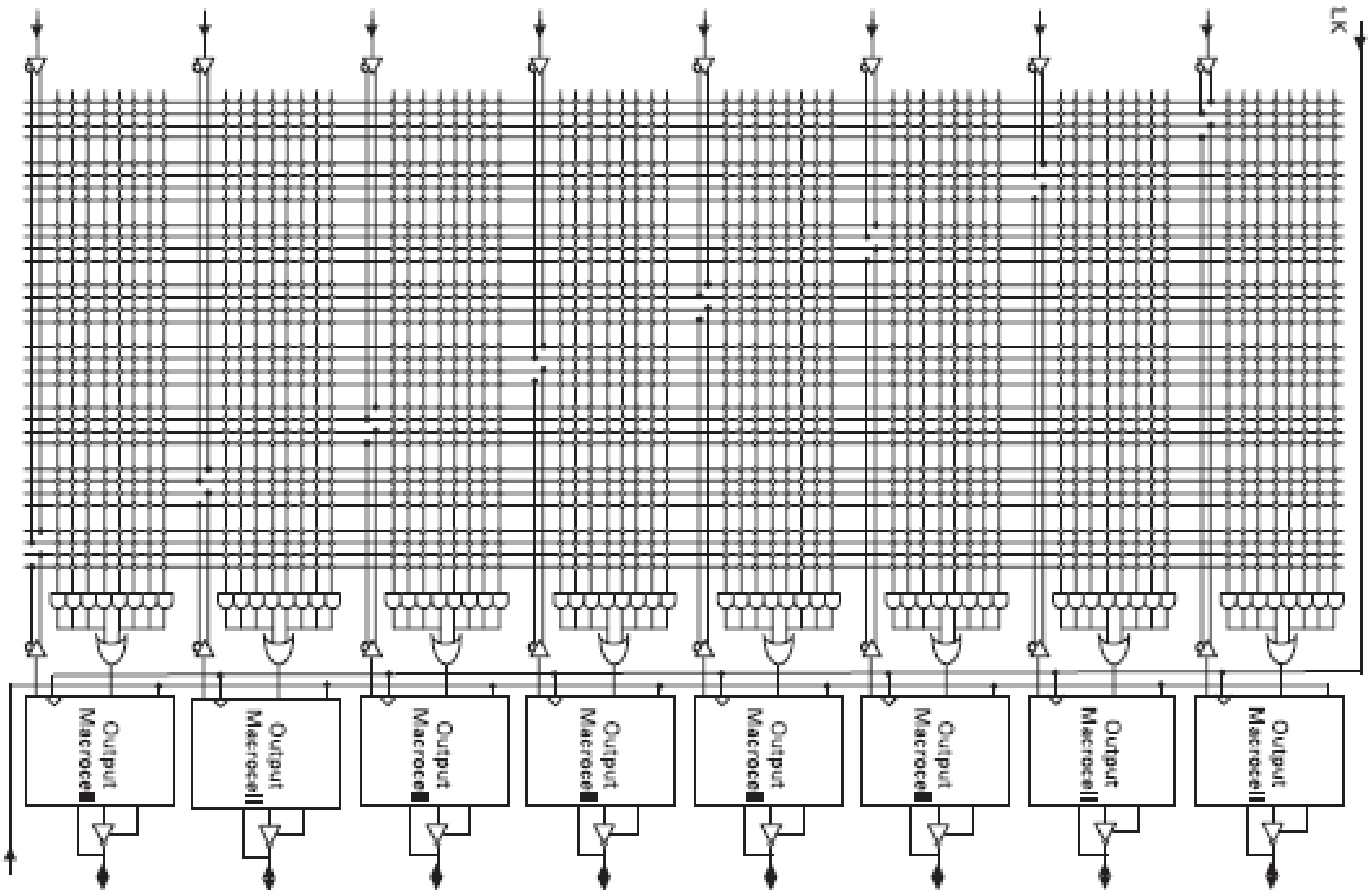



Figure A3
GAL 16V8 chip.

# CPLD architecture (Complex PLD)

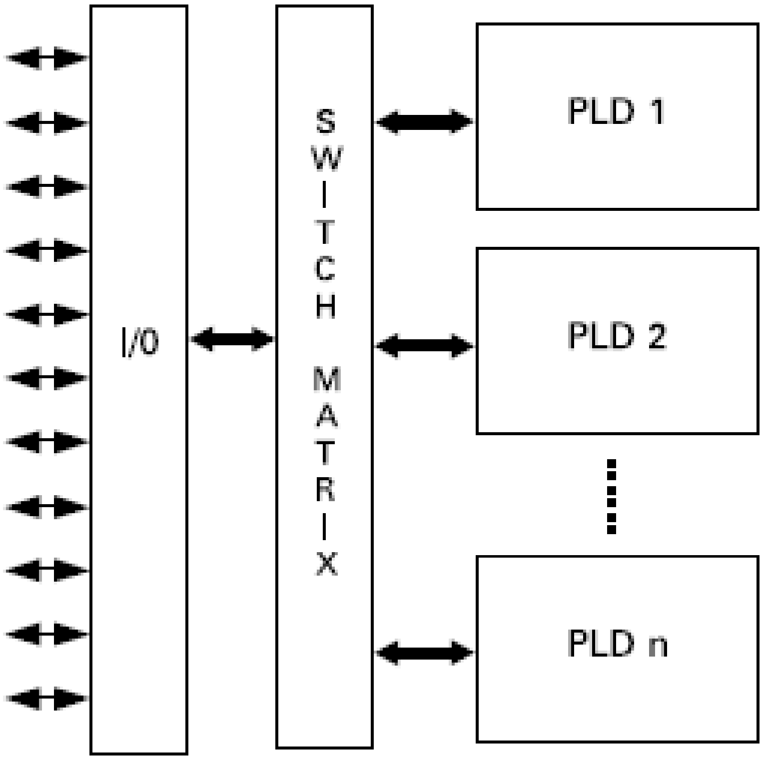



Figure A4
CPLD architecture.

# FPGA architecture

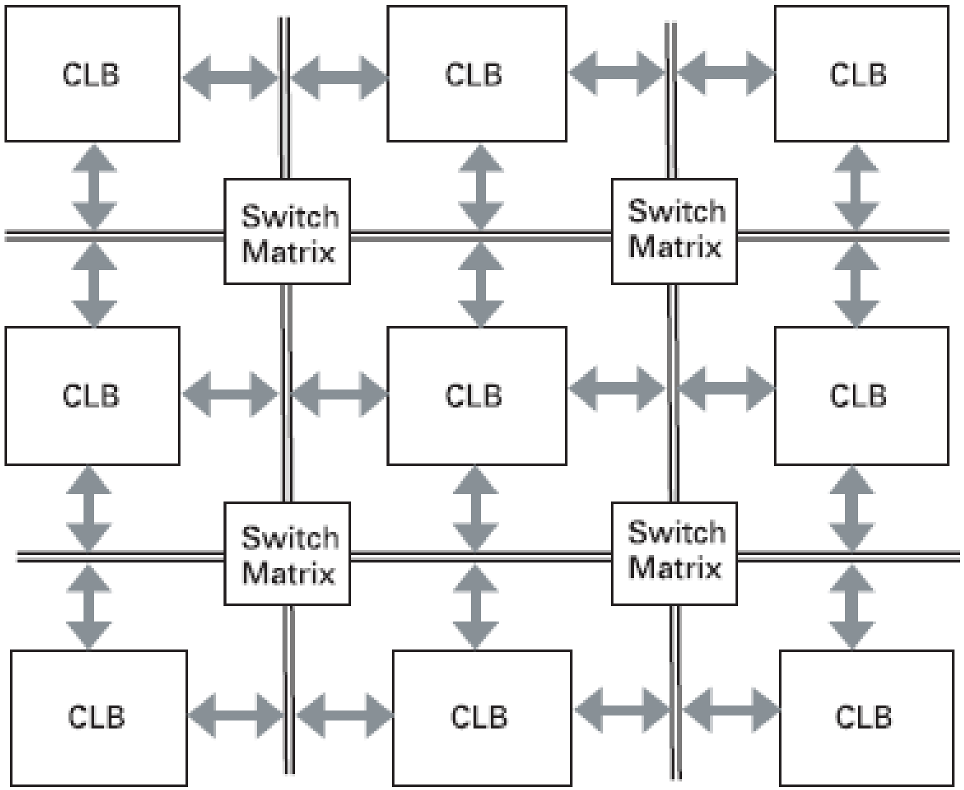



Figure A5
FPGA architecture.

# Intel FPGA Architecture

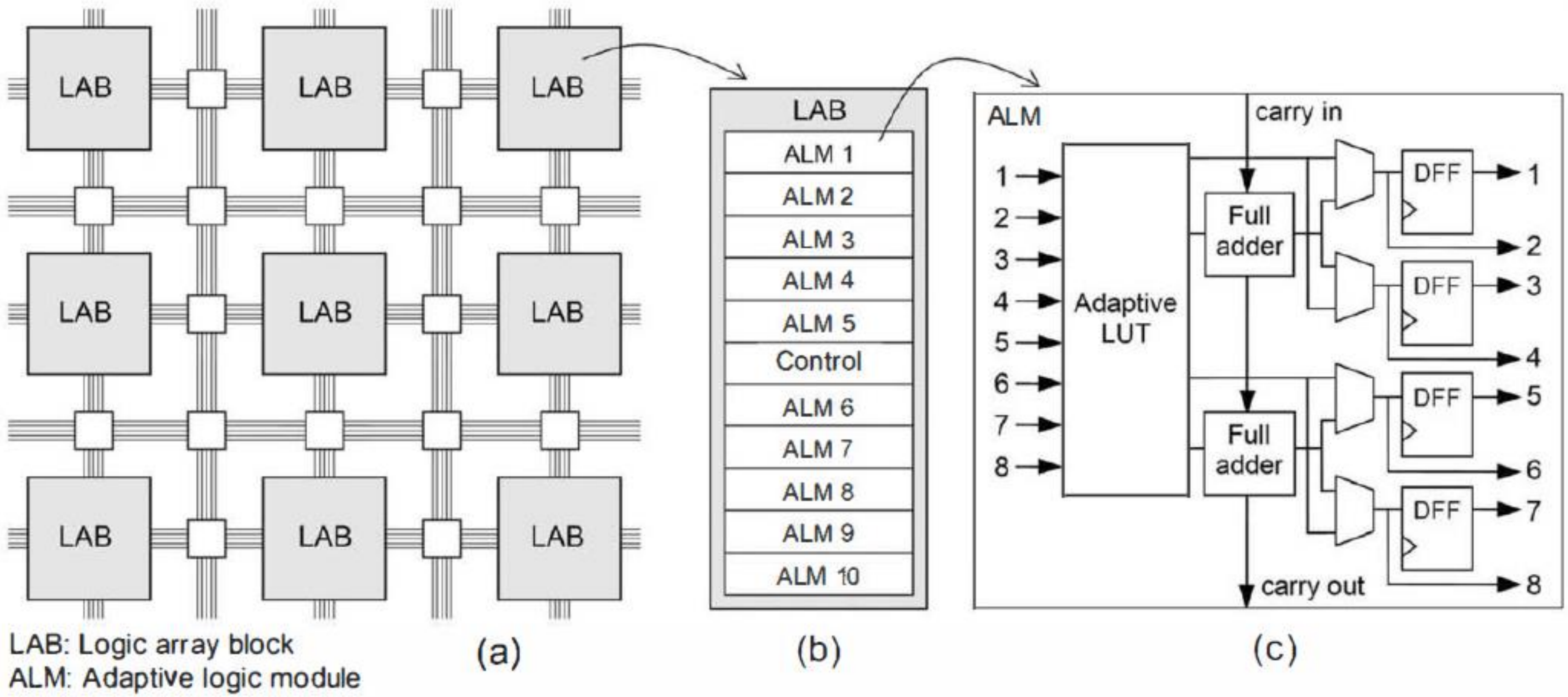



**Figure 4.8**

Intel architecture for the logic part of Stratix 10 and other FPGAs: (a) FPGA grid (array of LAB blocks); (b) LAB contents; (c) ALM contents.

# Xilinx FPGA Architecture

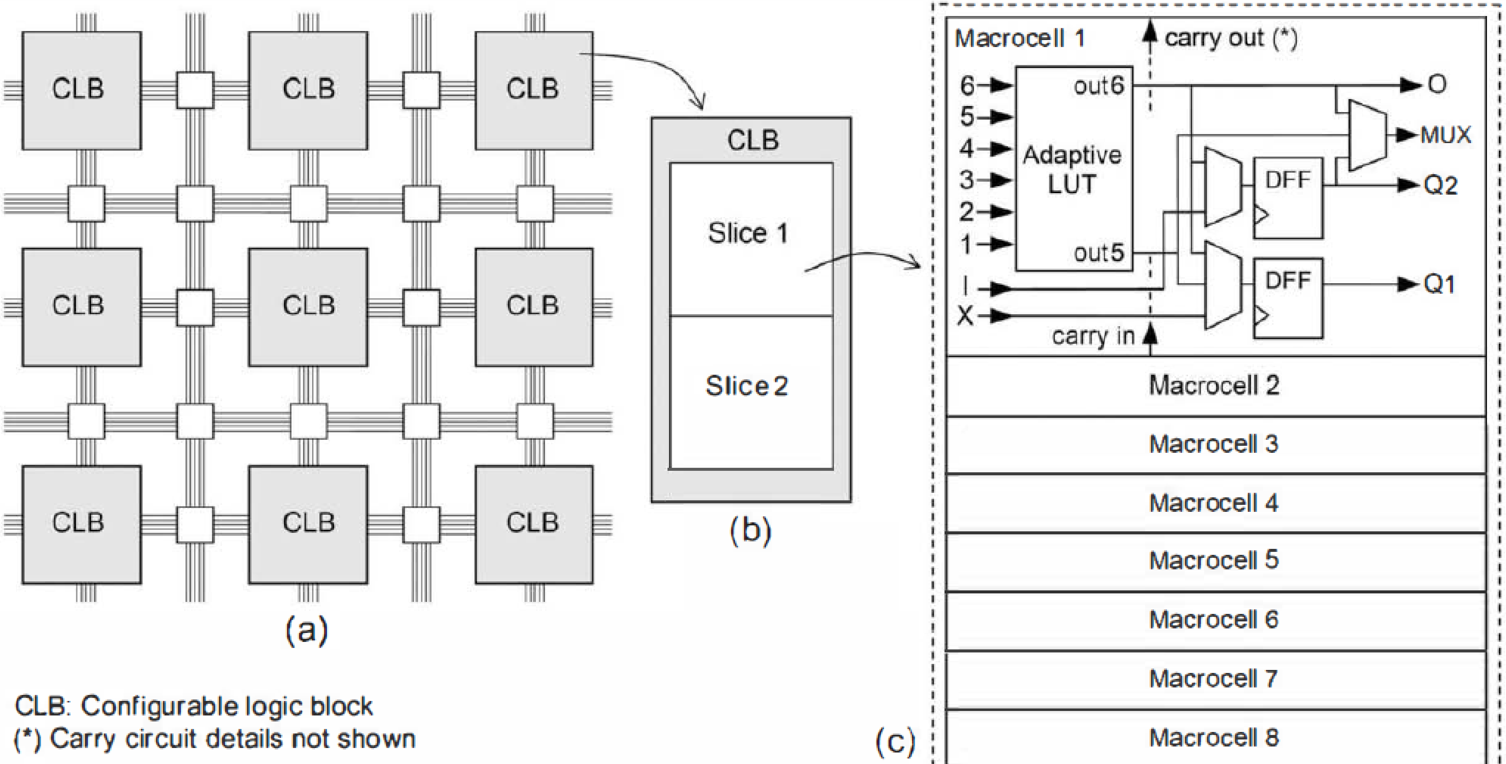



**Figure 4.9**

Xilinx architecture for the logic part of UltraScale+ and other FPGAs: (a) FPGA grid (array of CLB blocks); (b) CLB contents; (c) Slice contents.

# Genrations of PLDs

| Type | Architecture | Logic functions implementation | Main config. technologies | Introduced by |
|---|---|---|---|---|
| SPLD | PLA | AND-OR array, both programmable | EEPROM | Signetics, mid 1970s |
| | PAL | AND-OR array, only AND programmable | EEPROM | Monolitic Memories, mid/late 1970s |
| | GAL | AND-OR array, only AND programmable | EEPROM | Lattice, early 1980s |
| CPLD | Small array of GAL-like blocks | AND-OR array, only AND programmable | EPROM, then EEPROM, then Flash | Altera, 1984 |
| | Simplified FPGA, with nonvolatile config. memory (*) | LUT | | Altera, early/mid 2000s (MAX II, then V and 10) |
| FPGA | Large array of small GAL-like clusters, with LUT in place of AND-OR array and volatile config. memory | LUT | SRAM | Xilinx, 1984 |
| | | | Antifuse | Actel, late 1980s |
| | | | Flash | Several companies |

# FPGAs, From Beginning

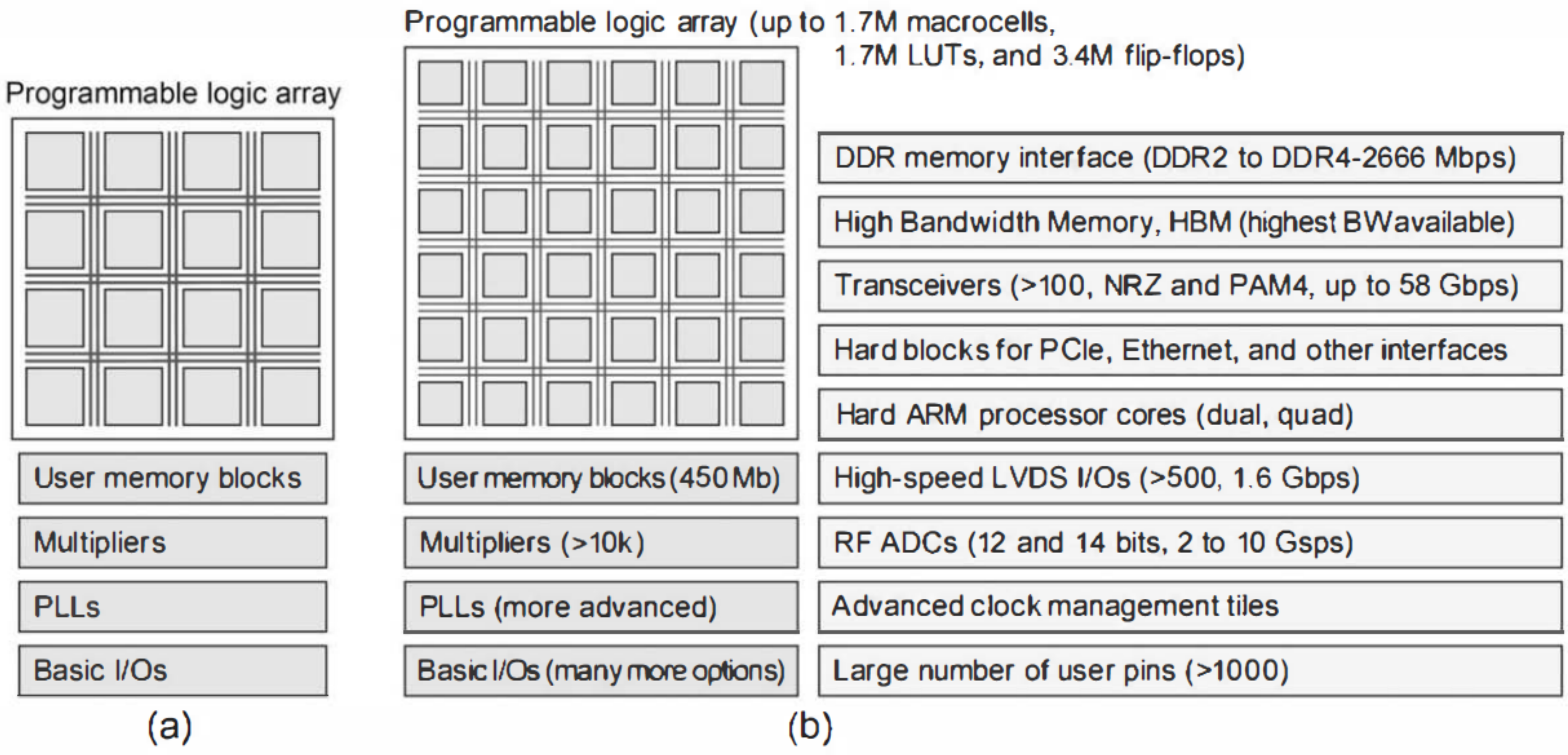



**Figure 4.7**
(a) FPGAs in the beginning and (b) FPGAs today.

# Next session

- 2 Code Structure
  - 2.1 Fundamental VHDL Units
  - 2.2 LIBRARY Declarations
  - 2.3 ENTITY
  - 2.4 ARCHITECTURE
  - 2.5 Introductory Examples
  - 2.6 Problems