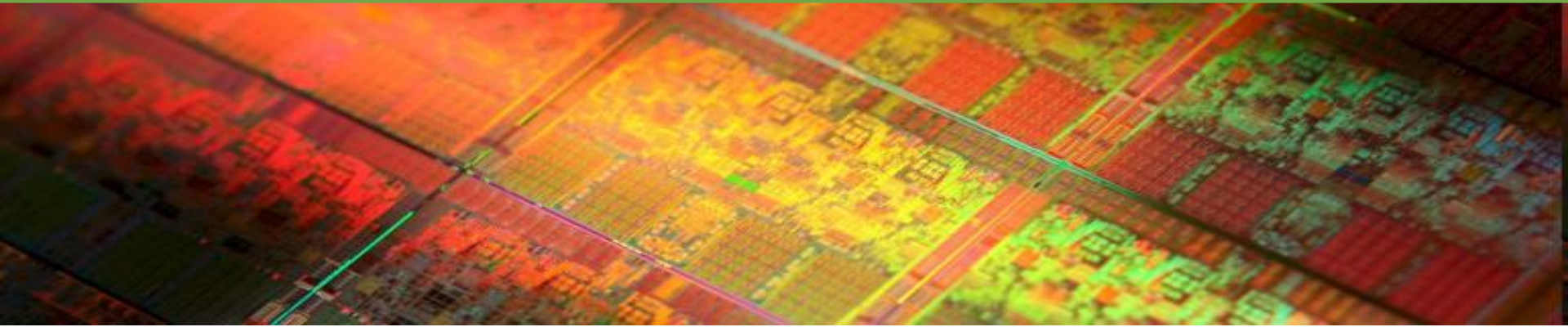


EE 2000

Logic Circuit Design (Fall 2018)



Lecture 1: Introduction to Field Programmable Gate Array (FPGA)

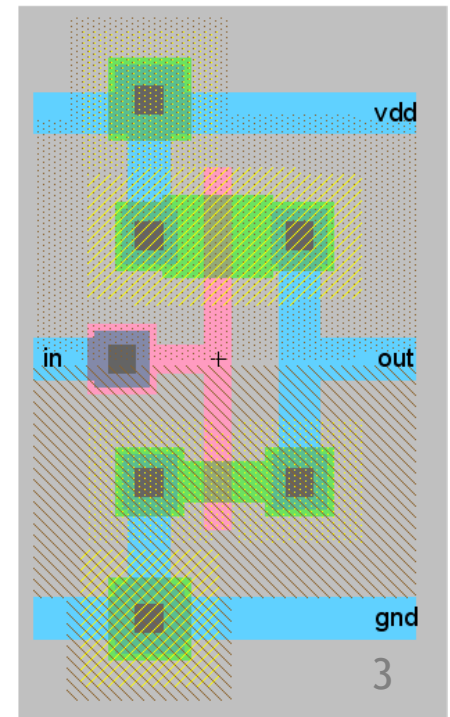
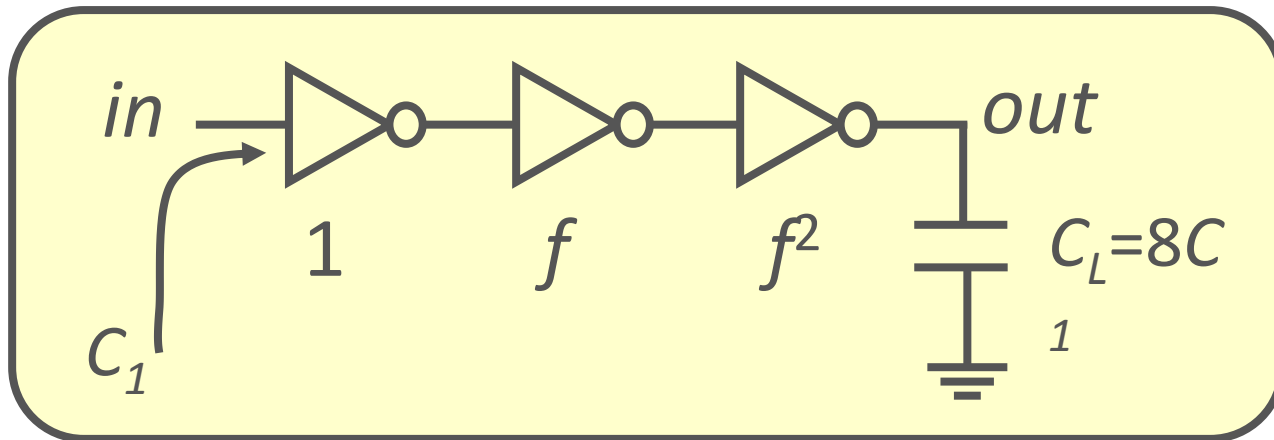
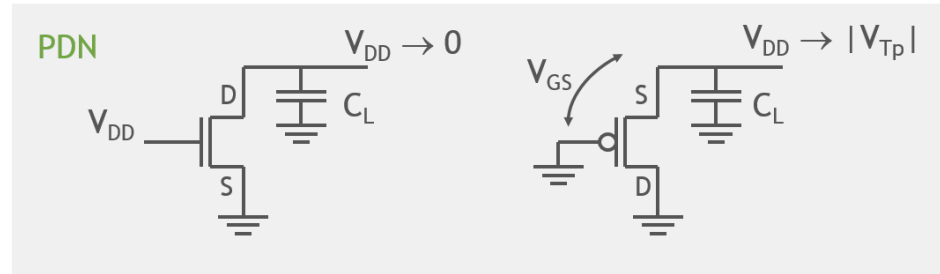
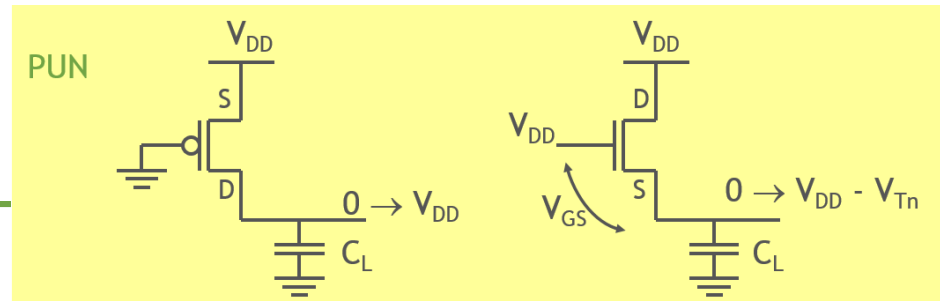
Instructor: Dr. Ray C.C. Cheung

Outline

- Administrative issues
 - 3-week (VHDL + FPGA), 3-week (Sequential Logic by Dr. Lau), 4-week (VHDL + FPGA)
 - Lab 1 - Week 6 (Oct 7 - 13)
 - Lab 2, 3, 4, 5 - Week 8, 11, 12, 13
- Logic Design → Programmable Logic Design
- Reconfigurable computing
 - FPGA, Xilinx, Amazon EC2 F1, Intel
- VHDL, Verilog, SystemC, ...

Outline

- Not covered
 - Gate Delay Model
 - Layout
 - Transistor level design

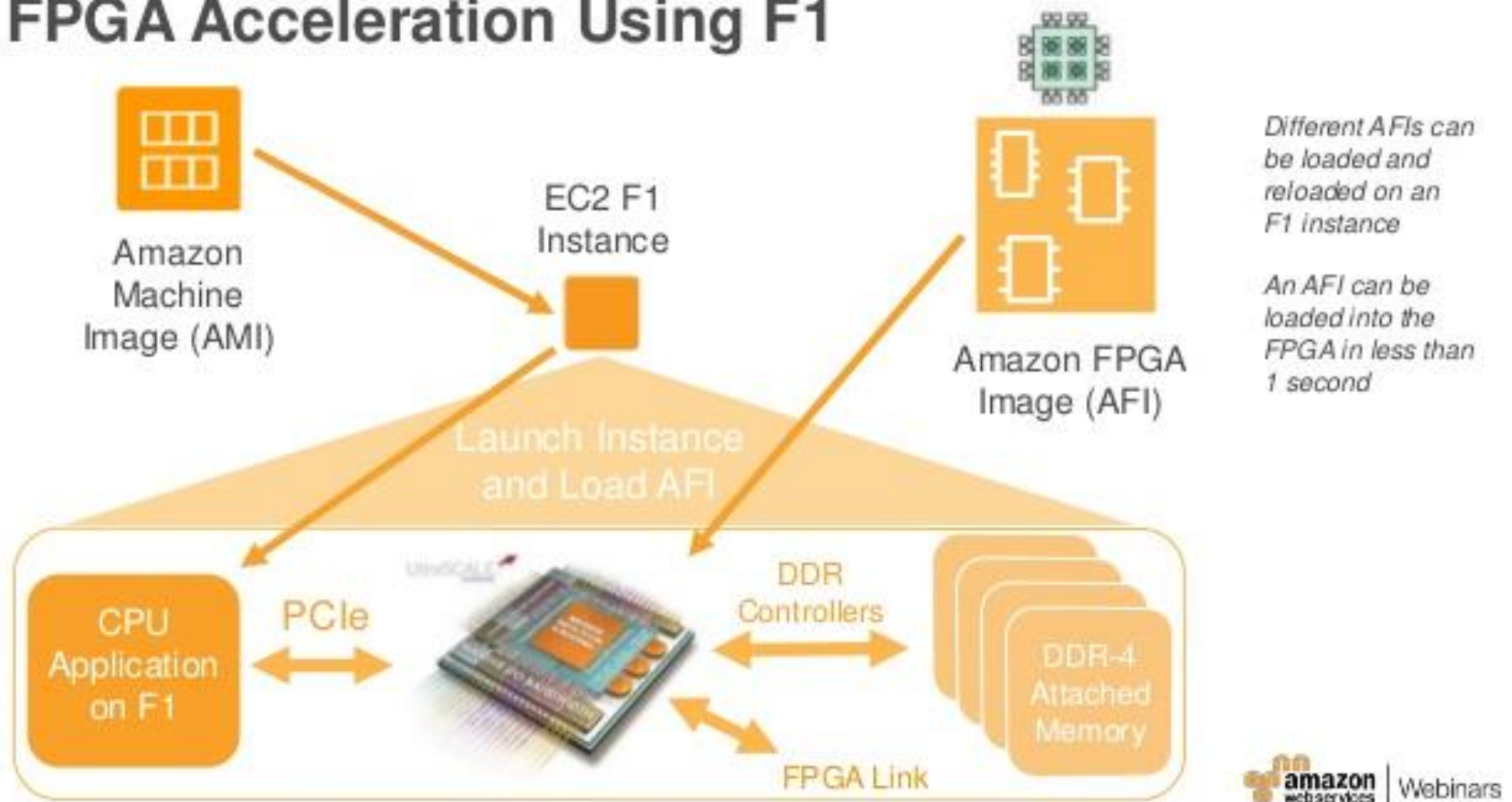


Outline

- Reconfigurable computing & Applications
- HDL Design Styles
- Overview of Configurable Logic
- Overview of FPGA
- Details of FPGA

Amazon EC2 F1

FPGA Acceleration Using F1



Intel + Altera

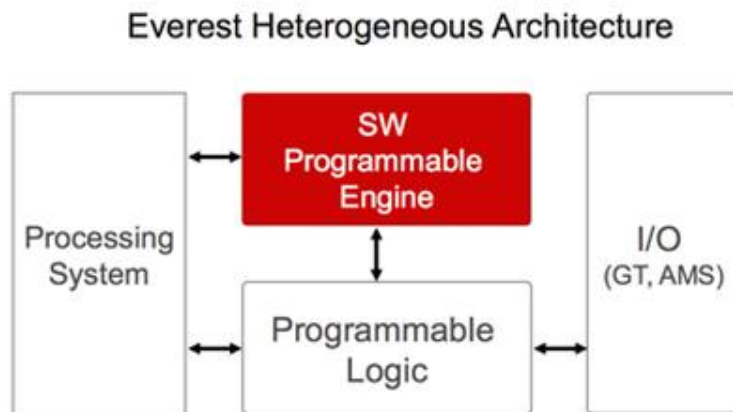


<https://www.intel.com/content/www/us/en/fpga/devices.html>

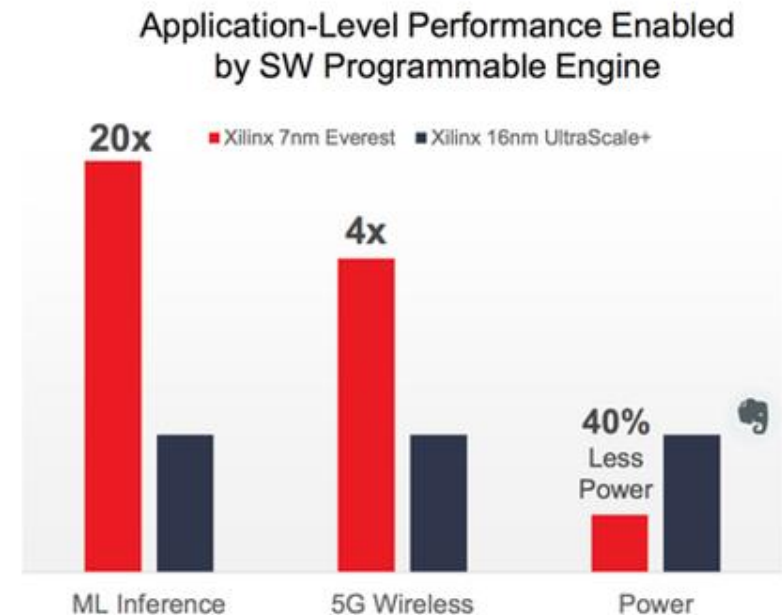
7nm vs. 16nm Xilinx FPGA

- Xilinx acquires DeePil, 3-yr BJ startup in Deep learning

Heterogenous Computing Architecture



- > Compute Efficiency
- > Reduced Power
- > Software Programmable



7nm Everest is the First Product to Integrate this New Architecture

Tape Out in 2018

Source: Forbes

FPGA Developers in Demand

Developer - Low Latency Trading (FPGA/C++)

 Morgan Stanley  New York, NY, United States

JOB DETAILS

Company Profile

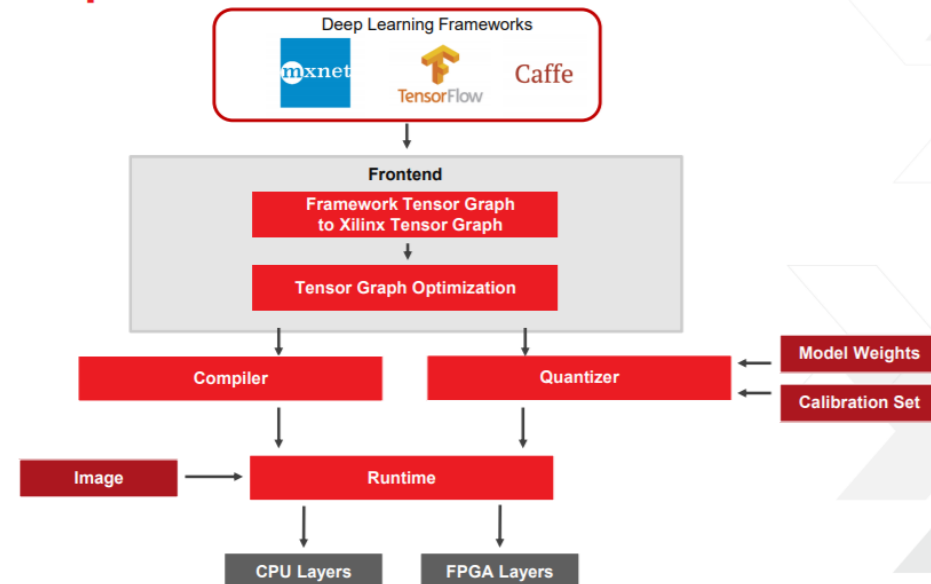
Morgan Stanley is a leading global financial services firm providing investment banking, securities, investment management and services. The Firm's employees serve clients worldwide including governments and individuals from more than 1,200 offices in 40 countries. As a market leader, the talent and passion of our people is critical to our success. Together, we share a common set of values rooted in integrity and team ethic. Morgan Stanley can provide a superior foundation for your professional career - a place for people to learn, to achieve and to grow. A culture that balances personal lifestyles, perspectives and needs is at the heart of our culture.



Using FPGA Supercomputers to Build a High Precision Robot

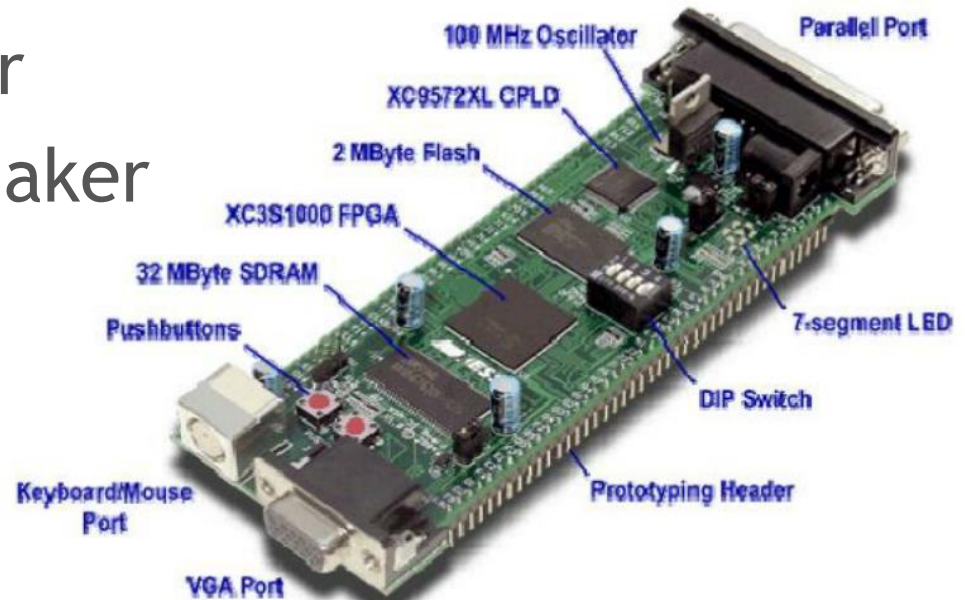
Kent Gilson | Haddington Dynamics Inc.

xDNN Compiler + Runtime



FPGA Board / FPGA Chip

- Xilinx is a chip maker
- Digilent is a board maker

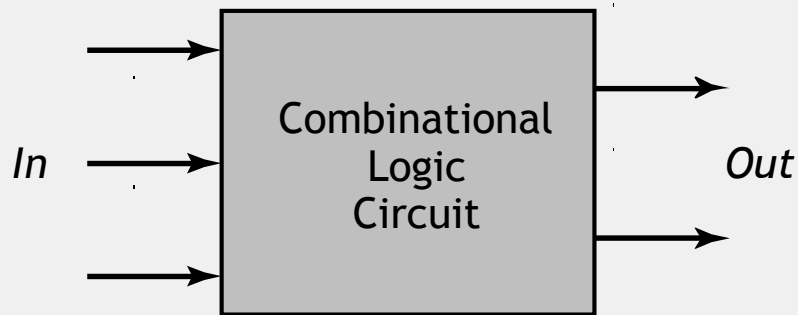


Video - Xilinx 7

- <https://www.youtube.com/watch?v=G4eBQYyxJ4s>

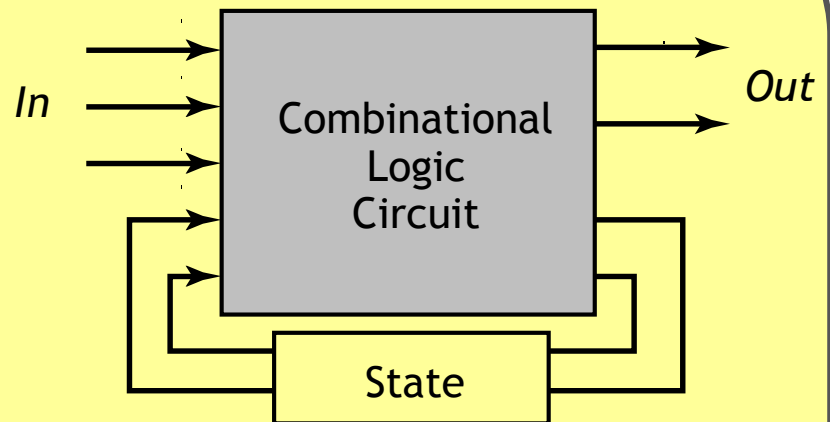


Combinational vs. Sequential Logic



Combinational

Output = $f(\text{input})$



Sequential

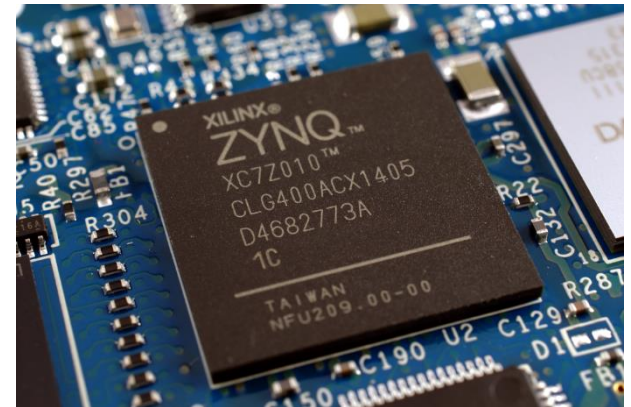
Output = $f(\text{input}, \text{state})$

Reconfigurable Computing

- Hardware Description Language
 - VHDL, Verilog, System-C, System-Verilog
 - CAD Tools for synthesis of FPGAs, ASIC
- Programmable Logic (FPGA/CPLD)
 - Configurable logic blocks + programmable routing
 - Can download a circuit design to the chip in seconds / minutes
 - The synthesis, place and route time will take longer
- Design digital circuits in much less time than traditional methods (MSI logic / PCB / Schematic)
 - Rapid Prototyping of Digital Systems

FPGA Applications

- Fast computations for (examples):
 - Long integer arithmetic
 - RSA encryption / decryption
 - DNA sequence matching
 - Heat and Laplace equations
 - Neural network computation
 - Video compression
- Can achieve up to over 1,000 times speedup compared to CPU
- Why?



FPGAs: Applications

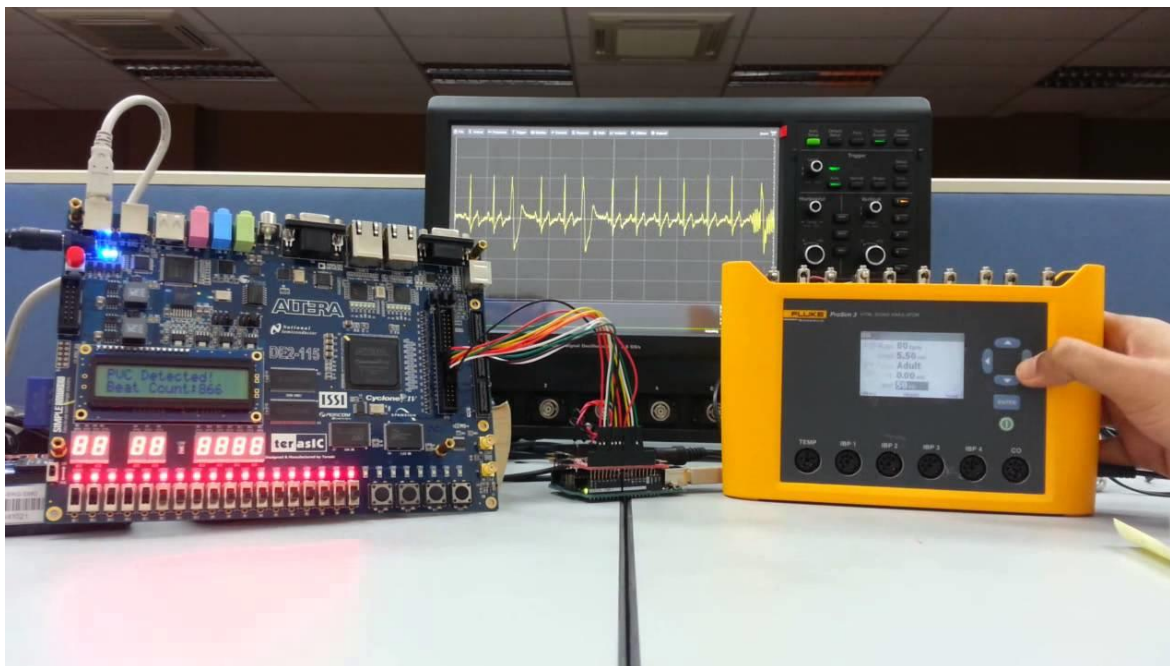
Rapid prototyping:

- FPGAs can contain 10 million or more gates, so many large real-world systems can be prototyped using a single FPGA.
- Multiple FPGAs can be interconnected to realize large systems.
- Accomplished by using boards with multiple FPGAs and plugging multiple boards into a backplane (motherboard).



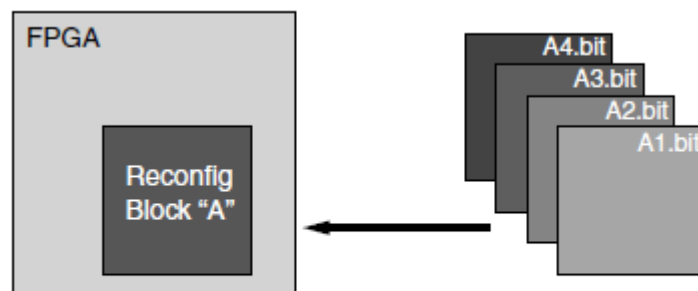
FPGAs: Applications

- As Final Product in Medium-Speed Systems:
 - When 150-200 MHz is sufficient, FPGAs can be used as the final product.
 - Enhancements to the system can be done as software updates rather than as hardware changes.



FPGAs: Applications

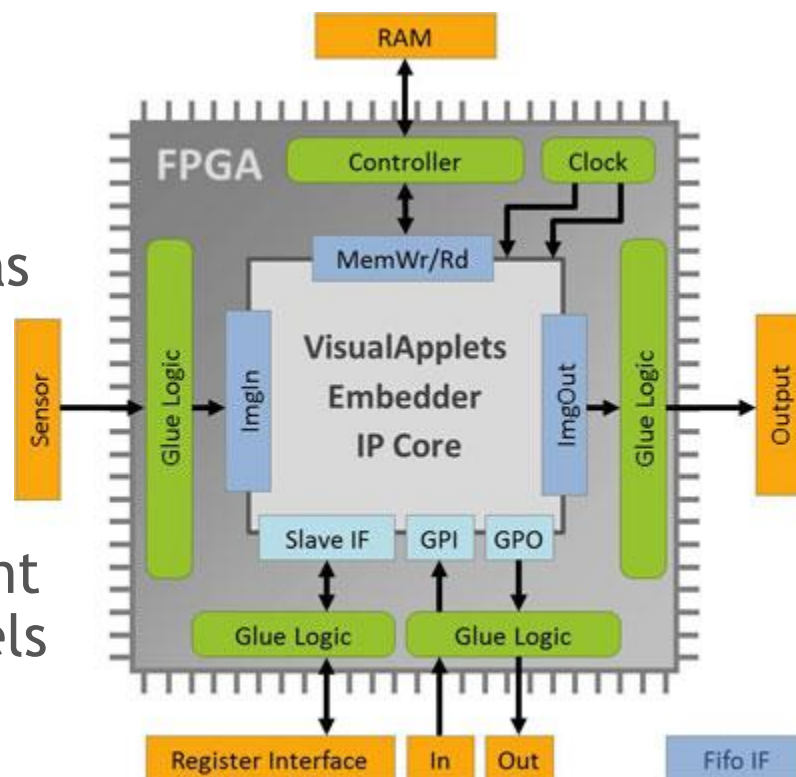
- Reconfigurable Circuits and Systems:
 - Can build dynamically reconfigurable circuits and systems.
 - SRAM-based FPGAs make it possible to implement “soft” hardware.
 - Used to design circuits and systems that need multiple functionalities at various times.
 - Example: reprogrammable Tomahawk missile.



Basic Premise of Partial Reconfiguration

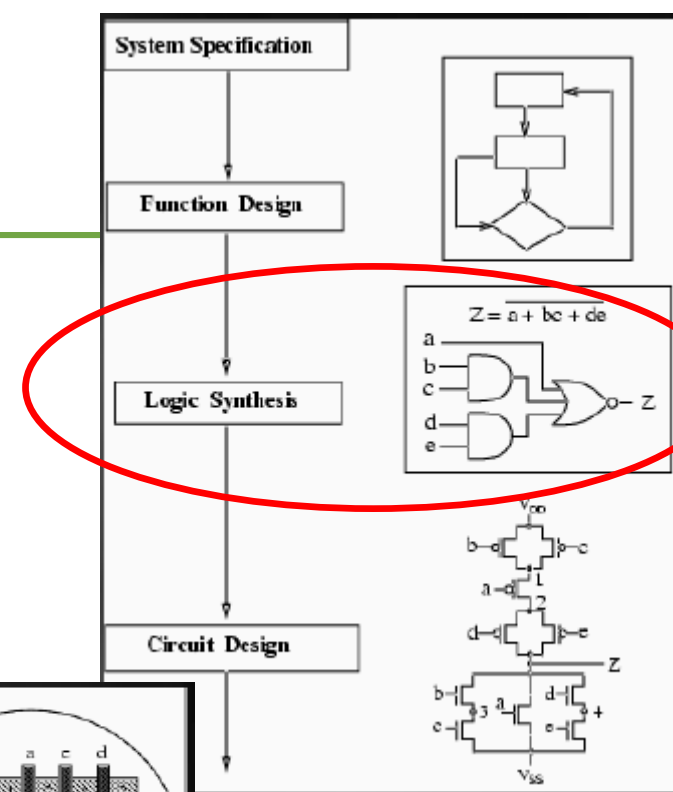
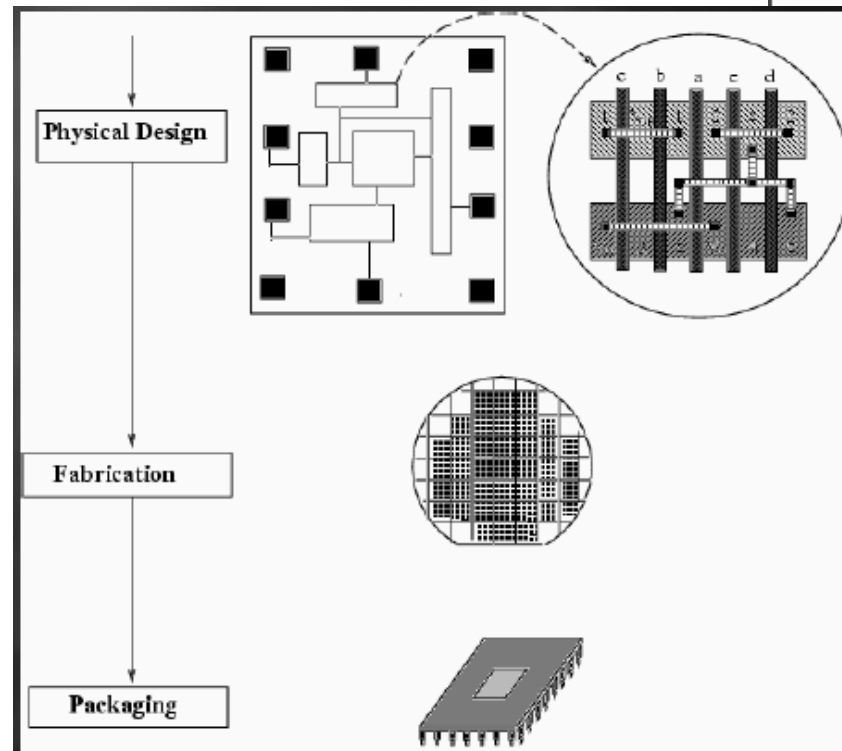
FPGAs: Applications

- Glue Logic:
 - New interface logic can be implemented on the same FPGA as in a software update.
- Hardware Accelerators/Coprocessors:
 - An FPGA can be used to implement the key kernel, as different kernels can dynamically be programmed into the FPGA.
 - Examples: computer architecture simulator acceleration, emulation boards, and hardware test/verification.



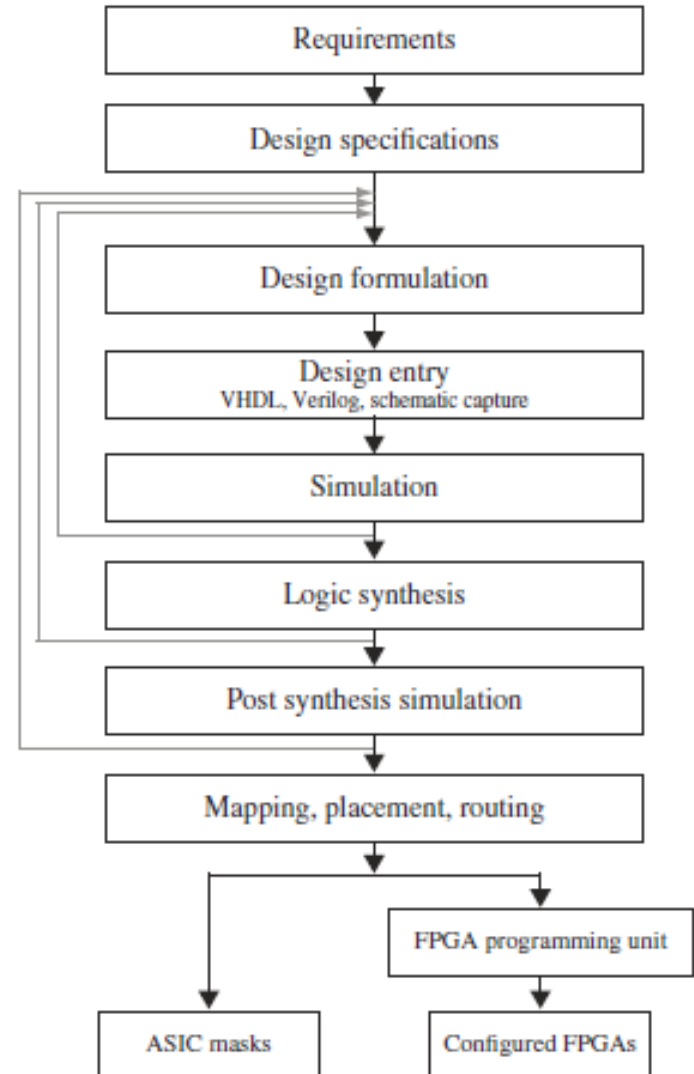
IC Design Cycle

- Design Specification
- Functional Design
- Logic Design
- Circuit Design
- Physical Design
- Fabrication
- Packaging
- IC delivery



Computer-Aided Design (CAD)

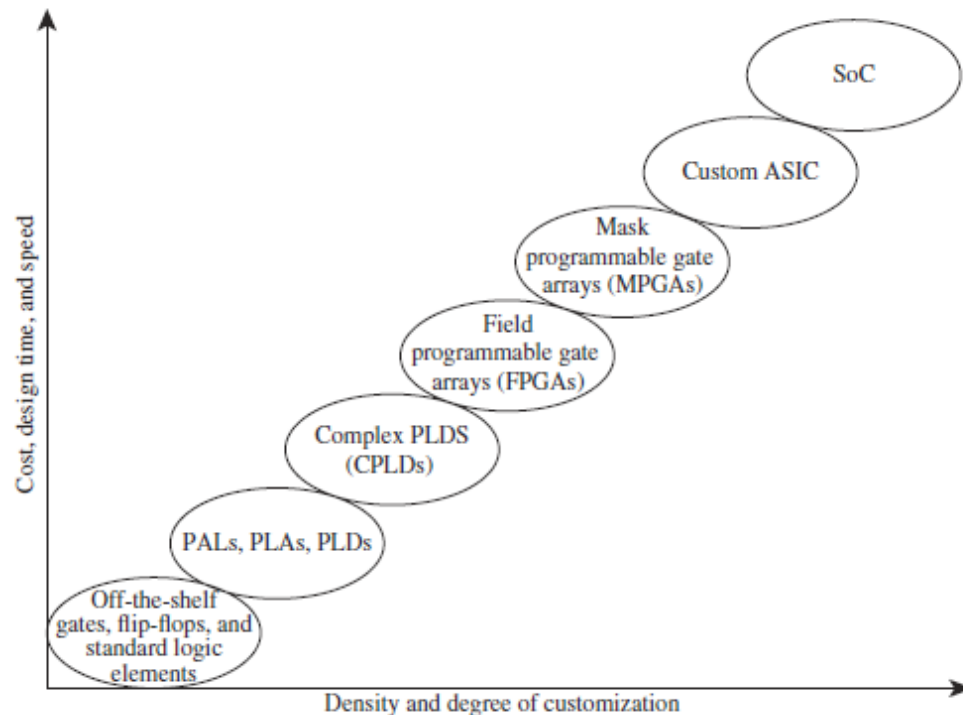
- Steps in modern digital system design:



CAD

- Target technologies that are available:

FIGURE 2-2: Spectrum of Design Technologies

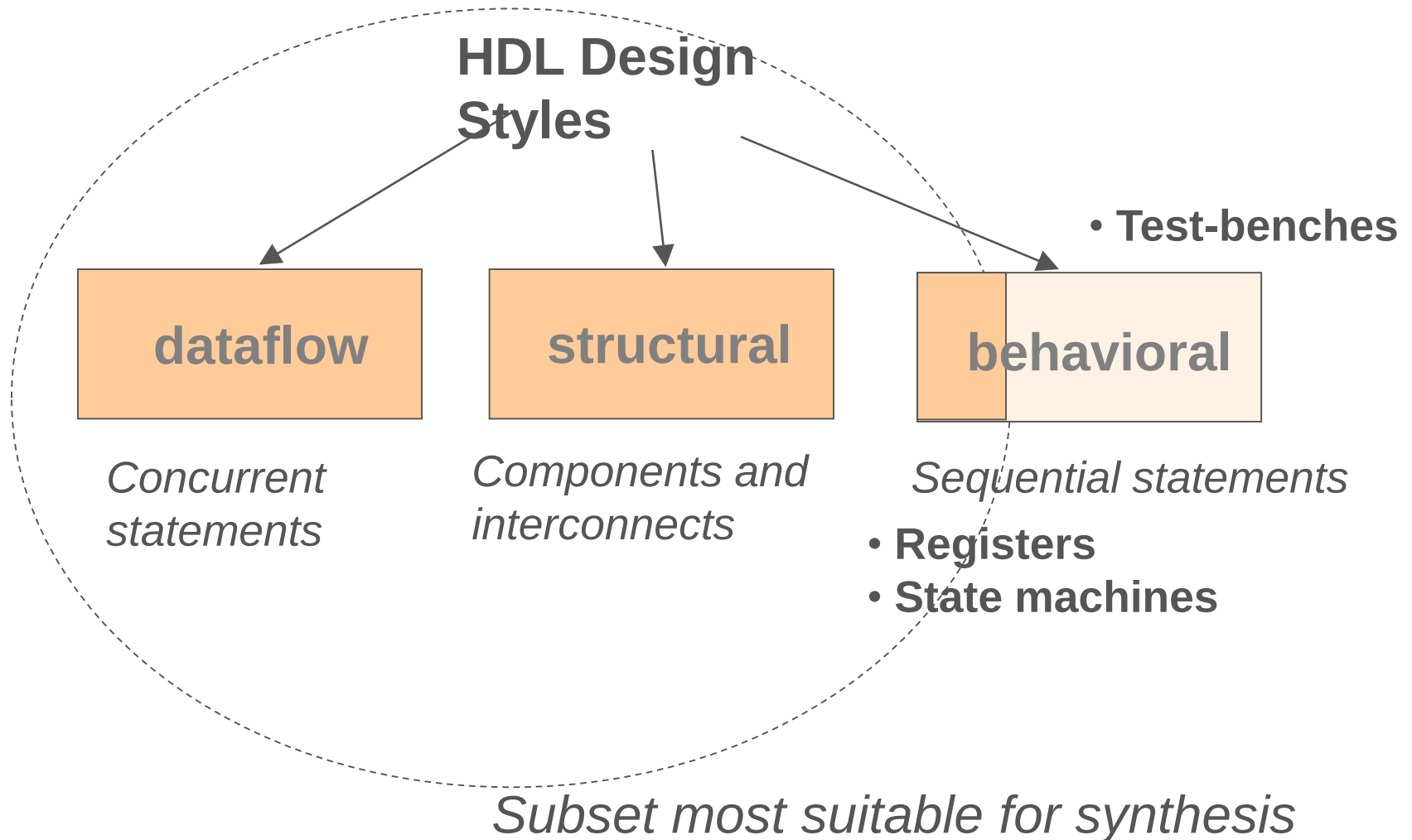


- Most common: field programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs).

Outline

- Reconfigurable computing & Applications
- **HDL Design Styles**
- Overview of Configurable Logic
- Overview of FPGA
- Details of FPGA

HDL Design Styles



Abstraction Level

- Behavioral or algorithmic level

This is the highest level of abstraction. A module can be implemented in terms of the desired design algorithm without concern for the hardware implementation details. Designing at this level is very similar to C programming.

- Dataflow level / RTL

The module is designed by specifying the data flow. The designer is aware of how data flows between hardware registers and how the data is processed in the design.

- Gate level / Structural

The module is implemented in terms of logic gates and interconnections between these gates. Design at this level is similar to describing a design in terms of a gate-level logic diagram.

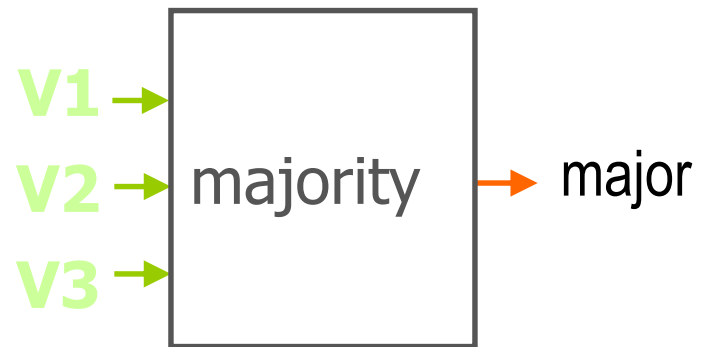
Behavioral Level

- This level describes a system by concurrent algorithms (Behavioral).
- Each algorithm itself is sequential, that means it consists of a set of instructions that are executed one after the other.
- Functions, Tasks and Always blocks are the main elements. There is no regard to the structural realization of the design.

Behavioral Example

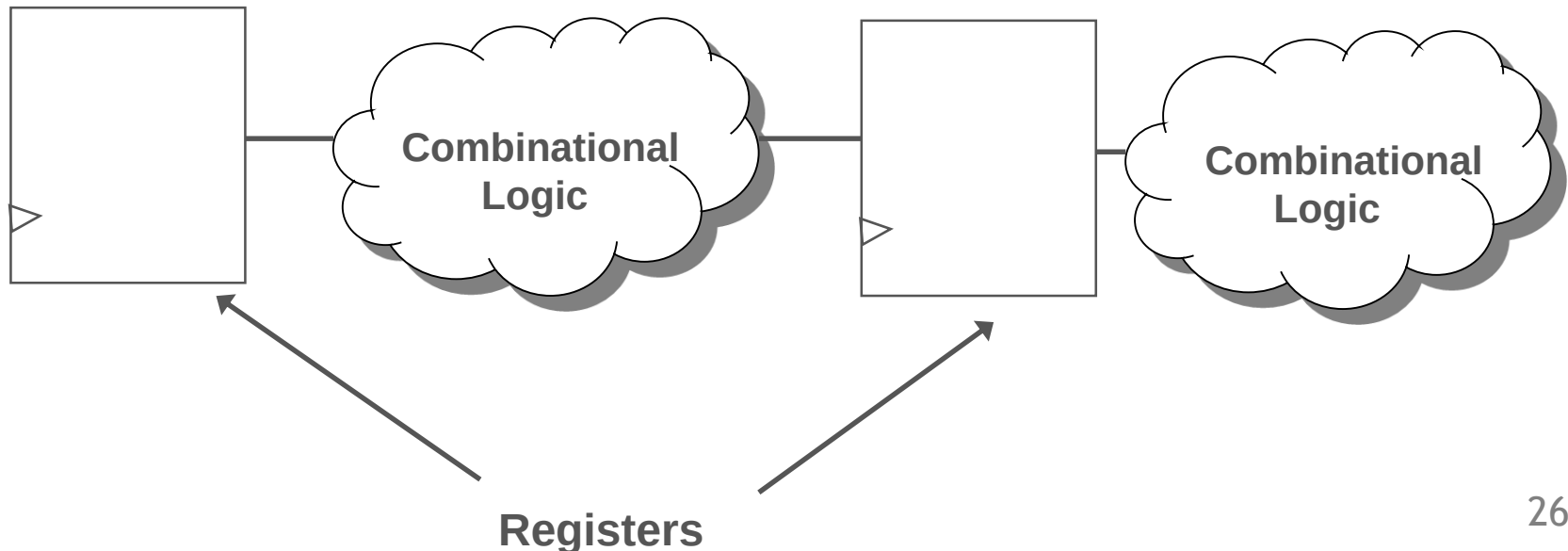
```
module majority (major, V1, V2, V3) ;  
  
output reg major ;  
input V1, V2, V3 ;  
  
always @(V1, V2, V3) begin  
    if (V1 && V2 || V2 && V3 || V1 && V3)  
        major = 1;  
    else  
        major = 0;  
end  
  
endmodule
```

Look familiar? C Language?



Register Transfer Level (RTL)

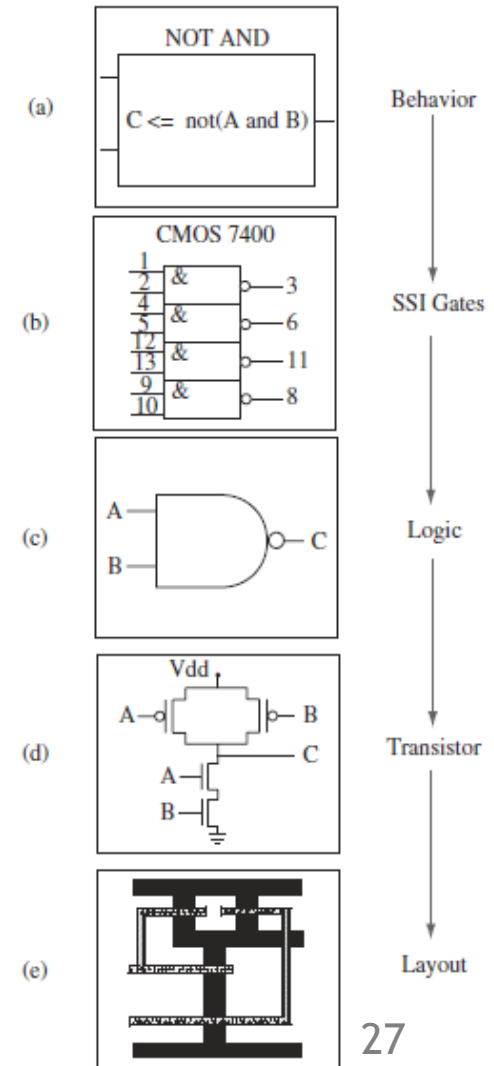
- ❑ **Register Transfer Level (RTL)** description is a way of describing the operation of a synchronous digital circuit. In RTL design, a circuit's behavior is defined in terms of the flow of signals (or transfer of data) between hardware registers, and the logical operations performed on those signals.
- ❑ An explicit clock is used. RTL design contains exact timing bounds: operations are scheduled to occur at certain times.



Behavioral and Structural VHDL

- Any circuit or device can be represented in multiple forms of abstraction.
- Example:

FIGURE 2-50: Different Levels of Abstraction of a NAND Device



Behavioral and Structural VHDL

- 3 Models:
 - **Behavioral:**
 - Specifies only the behavior at a higher level of abstraction.
 - Does not imply any particular structure or technology.
 - **Structural:**
 - Specifies more details.
 - Components used and the structure of the interconnection between the components are clearly specified.
 - At a low level of abstraction.
 - **Dataflow (Register Transfer Language [RTL]):**
 - Data path and control signals are specified.
 - System is described in terms of the data transfer between registers.
 - At an intermediate level of abstraction.

Behavioral and Structural VHDL

- Modeling a sequential machine: 3 approaches:
 - 1. Behavioral model:

```
process (CLK) -- State Register
begin
    if CLK = '1' then -- rising edge of clock (simulation)
        State <= Nextstate;
    end if;
end process;
```

```
process (CLK) -- State Register
begin -- (synthesis)
    if CLK'event and CLK = '1' then -- rising edge of clock
        State <= Nextstate;
    end if;
end process;
```

Behavioral and Structural VHDL

- 2. Data Flow:

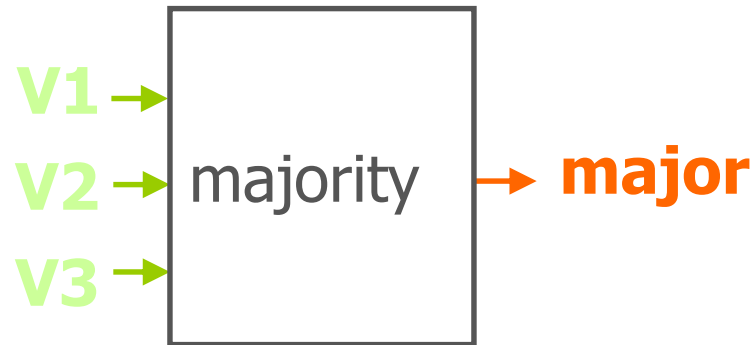
```
entity Code_Converter is  
  port(X, CLK: in bit;  
        Z: out bit);  
end Code_Converter;
```

```
architecture Equations of Code_Converter is  
  signal Q1, Q2, Q3: bit;  
begin  
  process(CLK)  
  begin  
    if CLK = '1' and CLK'event then -- rising edge of clock  
      Q1 <= not Q2 after 10 ns;  
      Q2 <= Q1 after 10 ns;  
      Q3 <= (Q1 and Q2 and Q3) or (not X and Q1 and not Q3) or  
        (X and not Q1 and not Q2) after 10 ns;  
    end if;  
  end process;  
  Z <= (not X and not Q3) or (X and Q3) after 20 ns;  
end Equations;
```

Behavioral and Structural VHDL

- 3. Structural model:
 - Describes gates and flip-flops in the circuit.
 - Gates and flip-flops can be defined in a separate VHDL module.
- Behavioral design is often used for quick time-to-market.

RTL Example



```
module majority (major, V1, V2, V3) ;
```

```
output major ;
```

```
input V1, V2, V3 ;
```

```
assign major = V1 & V2 | V2 & V3 | V1 & V3;
```

```
endmodule
```

Structural Level

- A schematic in text form
 - Within the logic level the characteristics of a system are described by logical links and their timing properties
- Build up a circuit from gates/flip-flops
 - Gates are primitives (part of the language)
- Structural design
 - Create module interface
 - Instantiate the gates in the circuit
 - Declare the internal wires needed to connect gates
 - Put the names of the wires in the correct port

Structural Example

```
module majority (major, V1, V2, V3) ;
```

```
output major ;
```

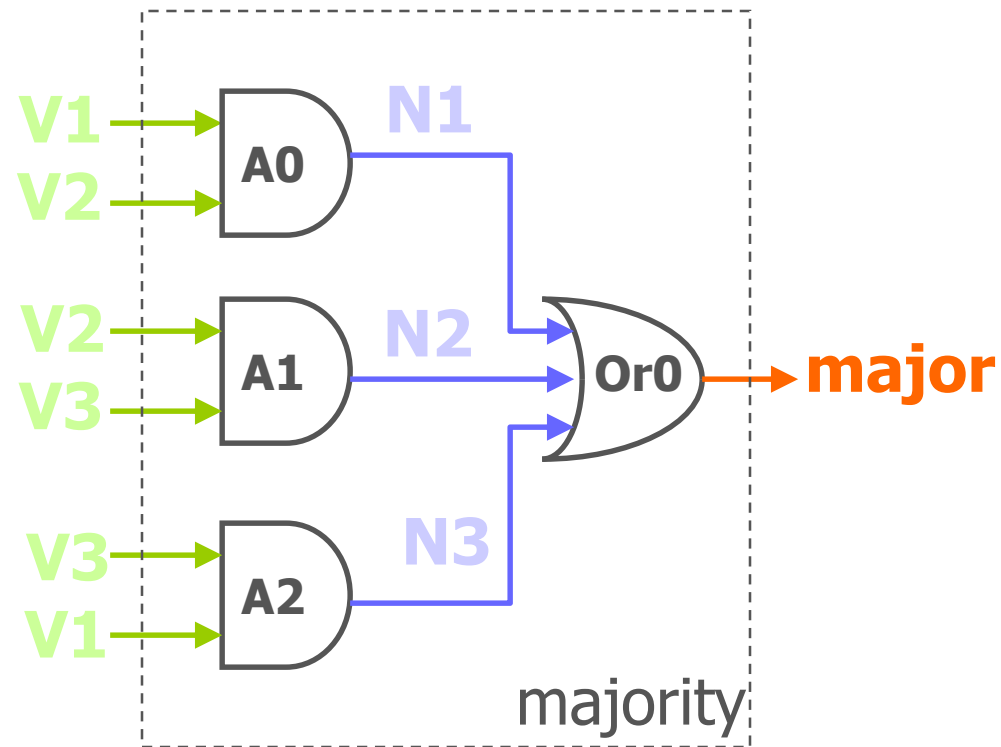
```
input V1, V2, V3 ;
```

```
wire N1, N2, N3;
```

```
and A0 (N1, V1, V2),  
      A1 (N2, V2, V3),  
      A2 (N3, V3, V1);
```

```
or Or0 (major, N1, N2, N3);
```

```
endmodule
```



Typical FPGA Chip

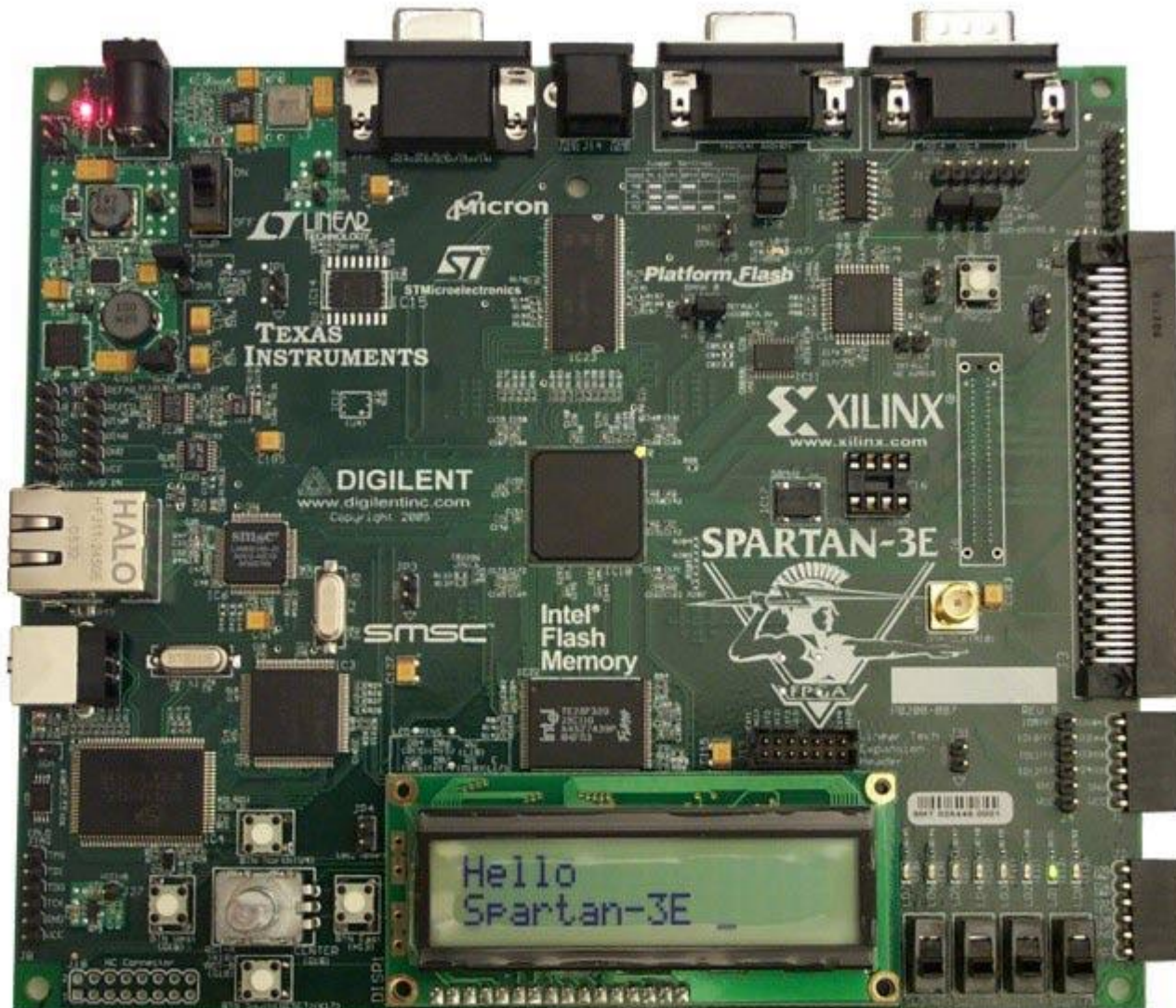
Virtex® UltraScale+™ VU9P FPGA



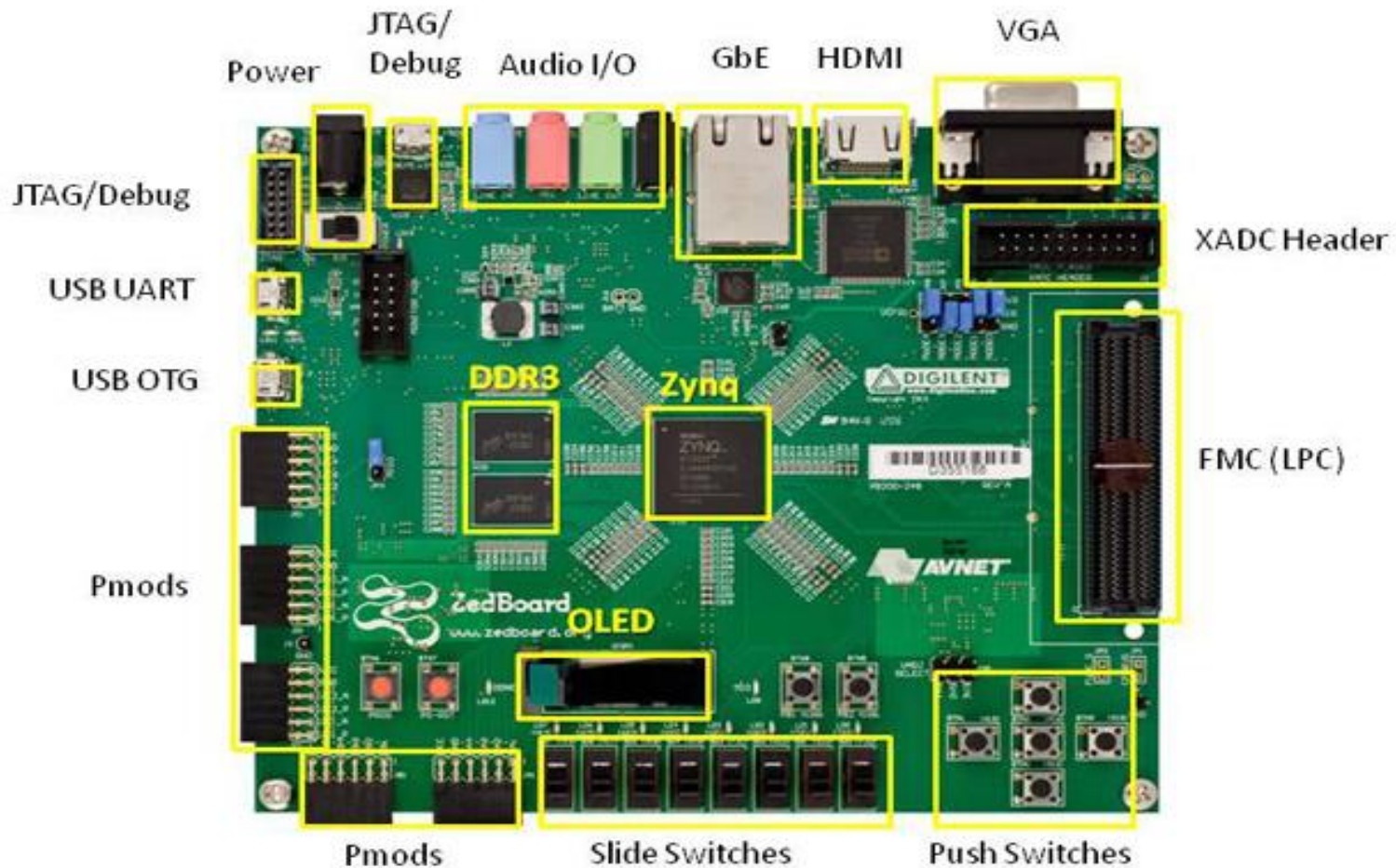
Virtex® UltraScale+™ FPGA VCU1525
Developer Board and Reference Design

- > 16nm TSMC FF+ FPGA
- > 2.5M System Logic Cells
- > 6840 DSP Blocks (18x27 MACs)
- > 382 Mbit On-Die SRAM
- > 4 DDR4-2400 x72 Channels
- > VU9P Virtex UltraScale+ FPGA
- > 21 TOPS (INT8)
- > 382 Mbit on-chip SRAM
- > 64 GByte on-board DRAM
- > 75W

An FPGA Board with Spartan

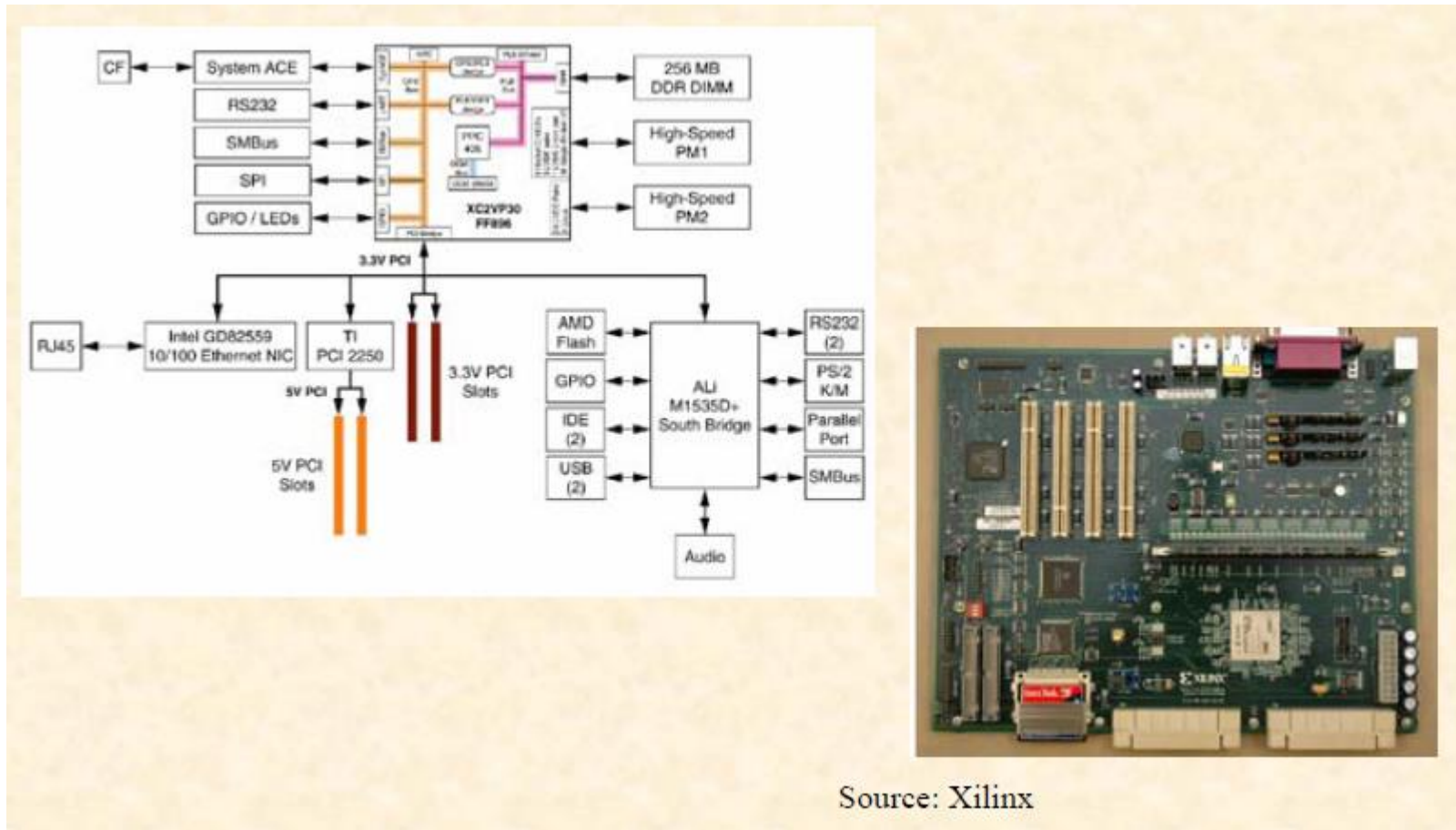


The Board we use in this Course



* SD card cage and QSPI Flash reside on backside of board

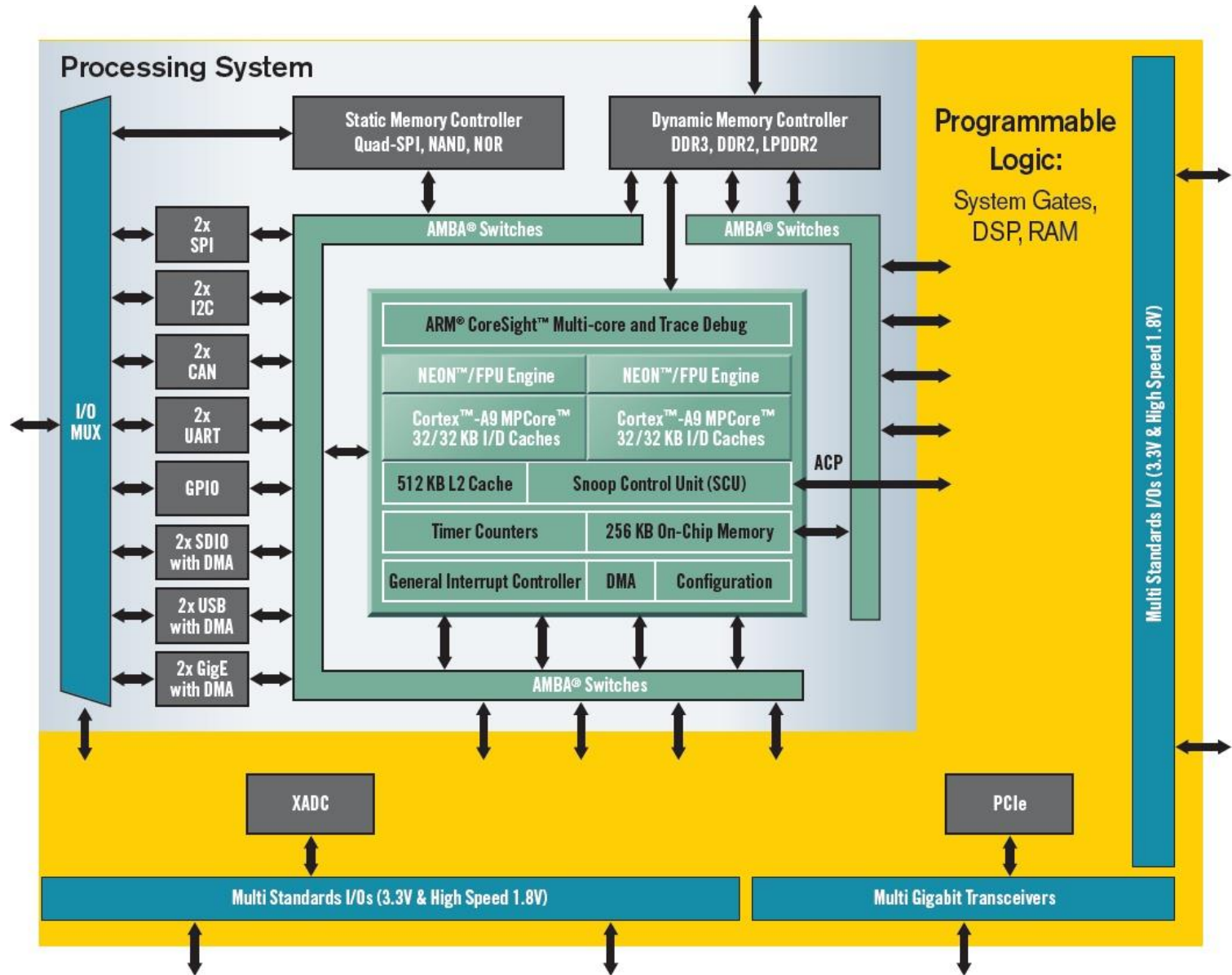
Xilinx FPGA boards



Source: Xilinx

Very much like a normal mother board in your PC

FPGA + ARM Processor



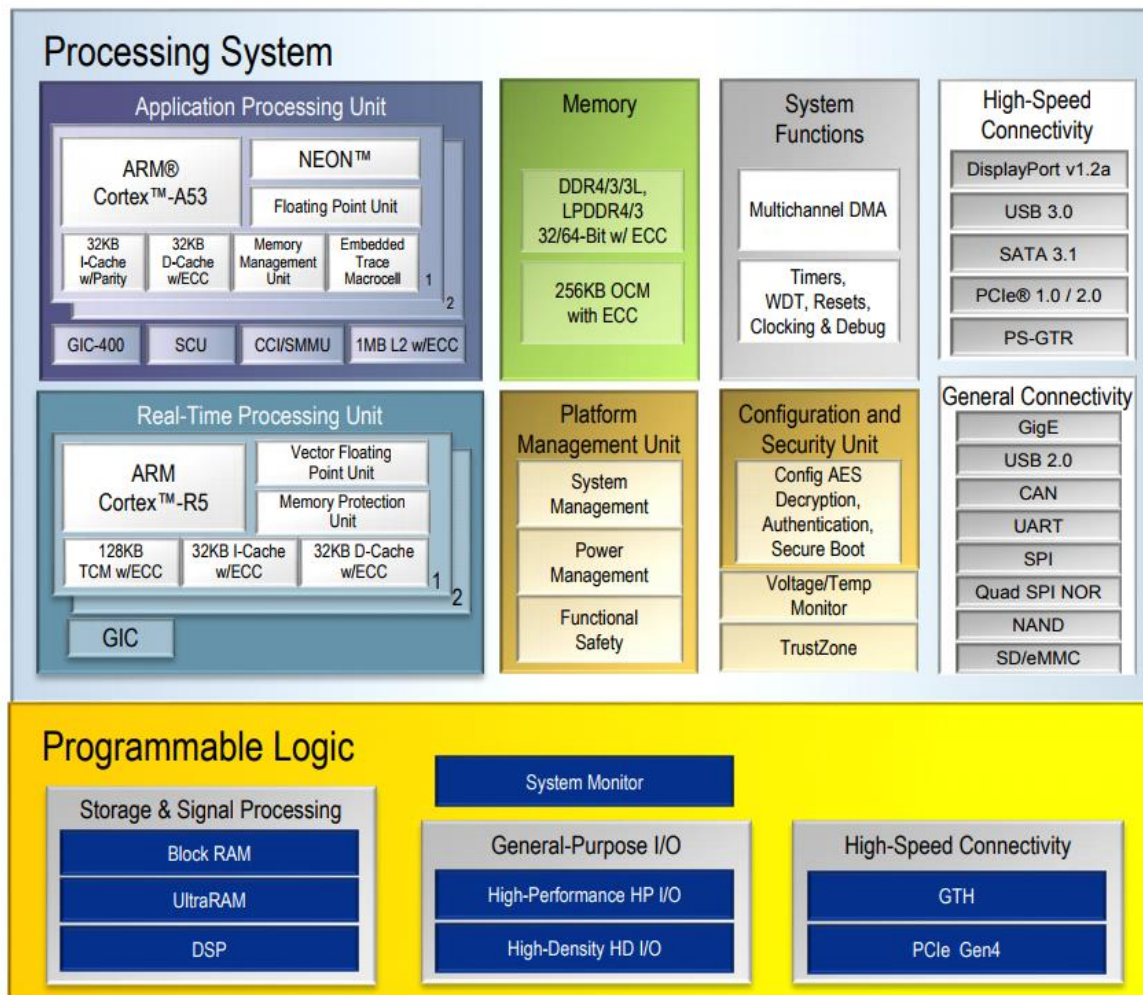
Multi-processor System-on-Chip (MPSoC)



All Programmable
Heterogeneous MPSoC

Zynq - MPSoC Details

Zynq® UltraScale+™ MPSoCs: CG Block Diagram



Outline

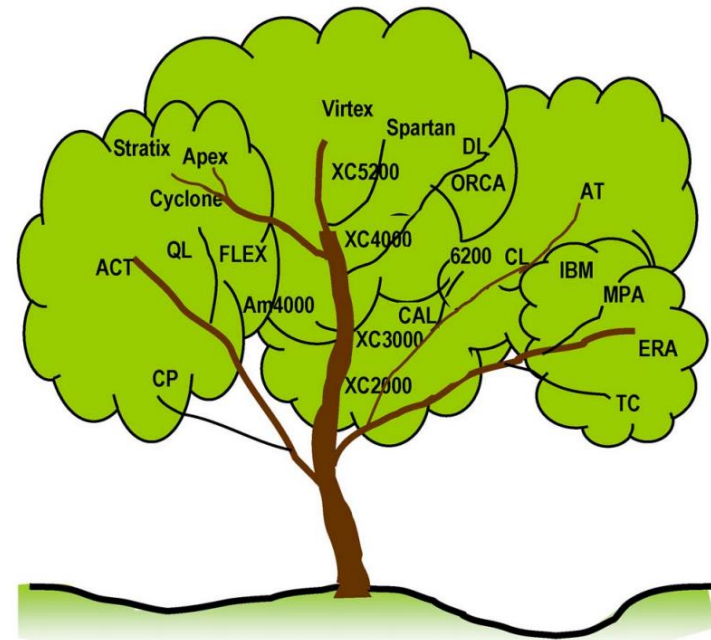
- Reconfigurable computing & Applications
- HDL Design Styles
- **Overview of Configurable Logic**
- Overview of FPGA
- Details of FPGA

Introduction to Logic Devices

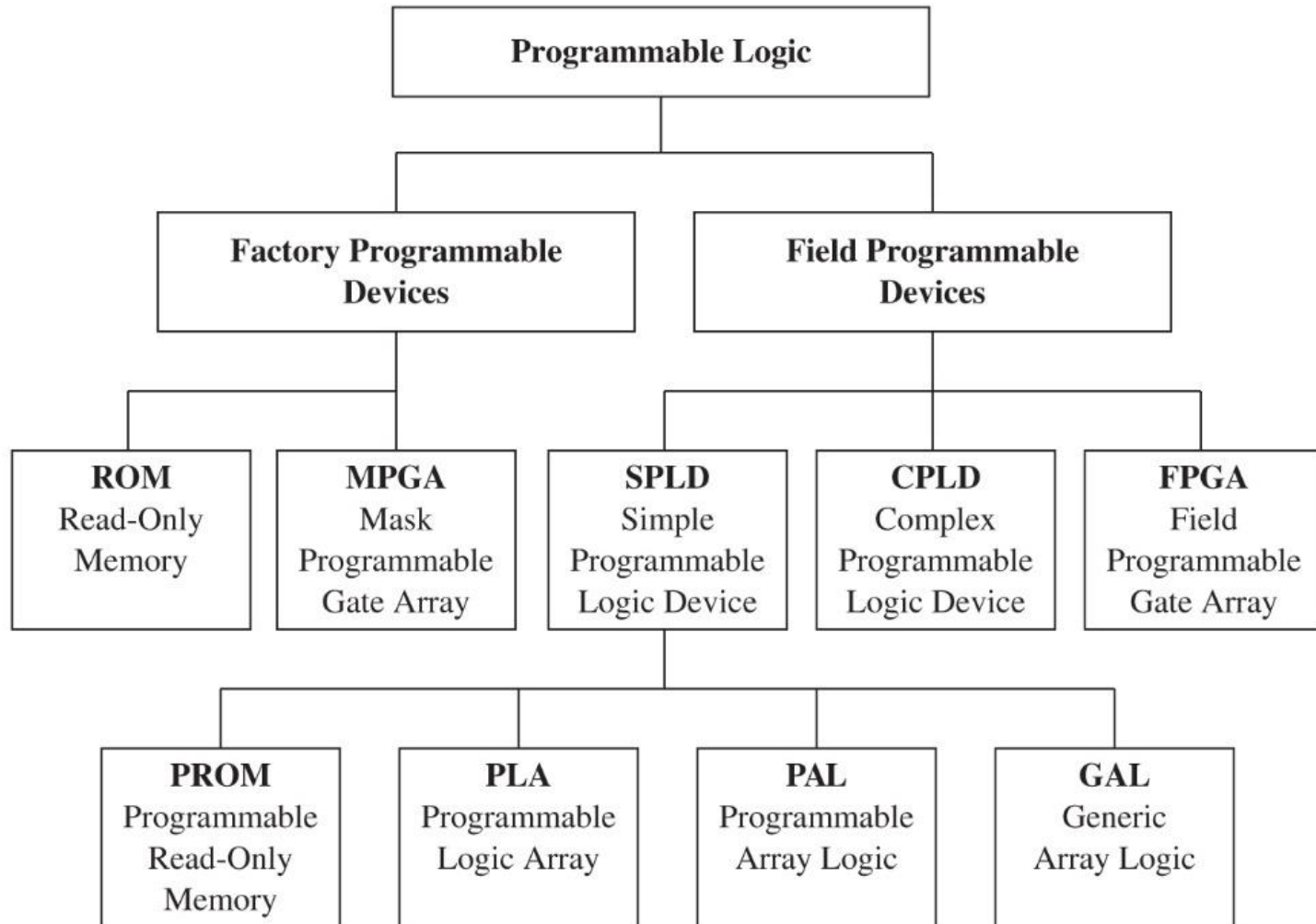
- Use of programmable logic devices allows the implementation of complex logic functions on a single IC chip.
- Programmable logic devices discussed:
 - Read-only memories (ROMs)
 - Programmable logic arrays (PLAs)
 - Programmable array logic (PAL)
 - Complex programmable logic devices (CPLDs)
 - Field-programmable gate arrays (FPGAs)

Overview of Programmable Logic Devices

- Programmable logic falls into two different types:
 - 1. Devices that can be programmed only once.
 - Factory programmable
 - 2. Devices that can be reprogrammed multiple times.
 - Field programmable



Overview of Programmable Logic Devices



Overview of Programmable Logic Devices

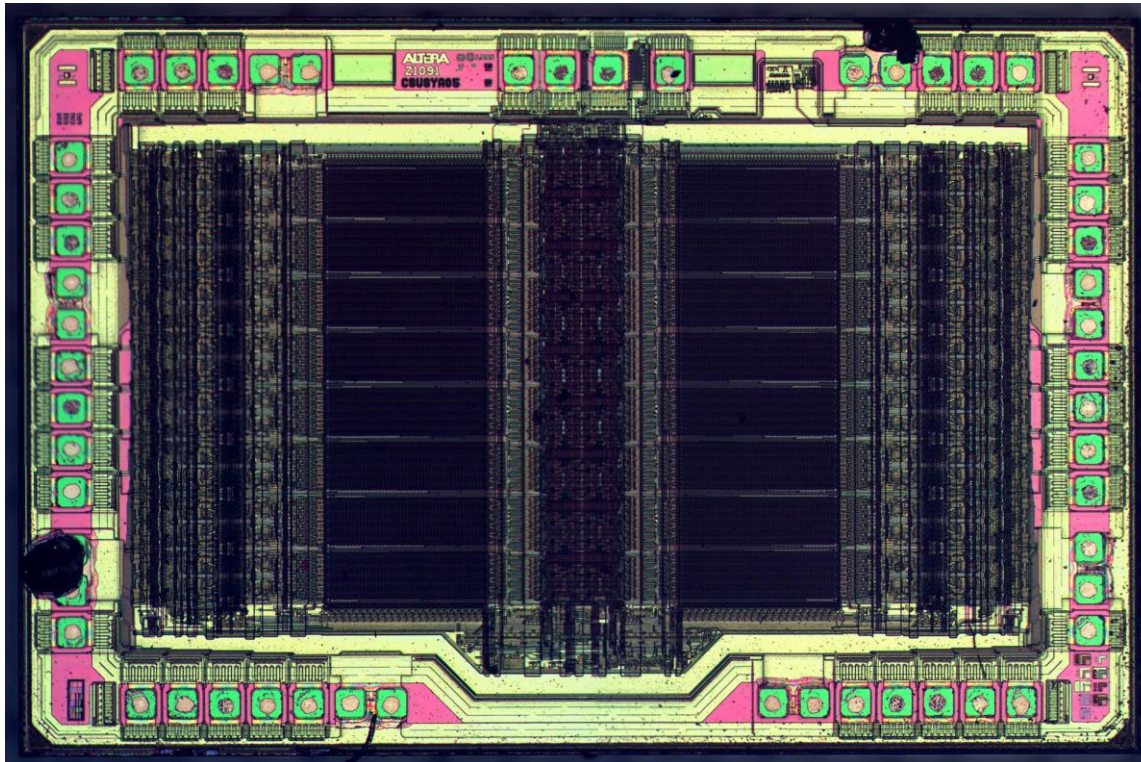
- Simple programmable logic devices:
 - Programmable Logic Arrays (PLAs):
 - There is a programmable AND array and a programmable OR array, allowing users to implement combinational functions in two levels of gates.
 - Programmable Array Logic (PAL):
 - A special case of a PLA, in that the OR array is fixed and only the AND array is programmable.
 - Many also contain flip-flops.
 - PLAs and PALs were popular in the 1970s and 1980s due to ease of design.

Overview of Programmable Logic Devices

- Simple programmable logic devices:
 - Programmable Logic Devices (PLDs)/Generic Array Logic (GAL):
 - Flash erasable/reprogrammable PALs.
 - Contain macroblocks with arrays of gates, multiplexers, flip-flops, or other standard building blocks. Several of these macroblocks appear in a PLD.
 - Lattice Semiconductor created a line of these: GALs

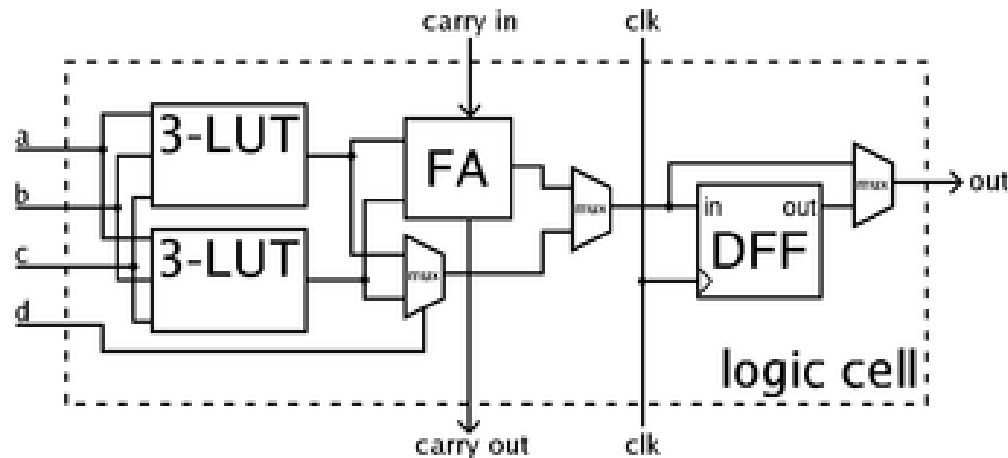
Overview of Programmable Logic Devices

- Complex Programmable Logic Devices (CPLDs):
 - Have more integration capability than SPLDs.
 - Come in sizes ranging from 500 to 16,000 gates.
 - Multiple PLDs placed into the same chip.



Overview of Programmable Logic Devices

- Field-Programmable Gate Arrays (FPGAs)
 - During the late 1980s, Xilinx started using static RAM storage elements to hold configuration information.
 - Larger and more complex blocks containing static RAMs and multiplexers.
 - With improved technology over the past 15 years, FPGAs can now contain more than 100 million gates.



Overview of Programmable Logic Devices

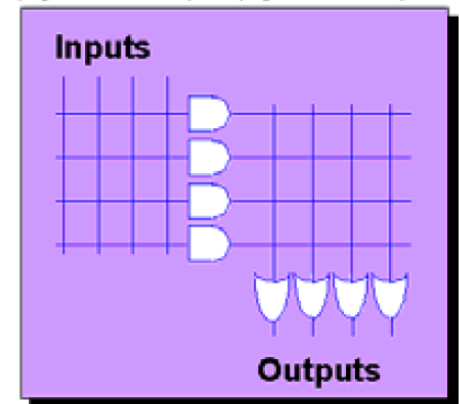
- Comparison of devices:

	SPLD	CPLD	FPGA
Density	Low Few hundred gates	Low to Medium 500 to 12,000 gates	Medium to High 3,000 to 5,000,000 gates
Timing	Predictable	Predictable	Unpredictable
Cost	Low	Low to Medium	Medium to High
Major Vendors	Lattice Semiconductor Cypress AMD	Xilinx Altera	Xilinx Altera Lattice Semiconductor Microsemi (previously Actel)
Example Device Families	Lattice Semiconductor GAL16LV8 GAL22V10 Cypress PALCE16V8 AMD 22V10	Xilinx CoolRunner XC9500 Altera MAX	Xilinx Kintex Artix Virtex Spartan Altera Arria Cyclone Stratix Lattice Mach ECP Microsemi Accelerator

Simple Programmable Logic Devices (SPLDs): Introduction

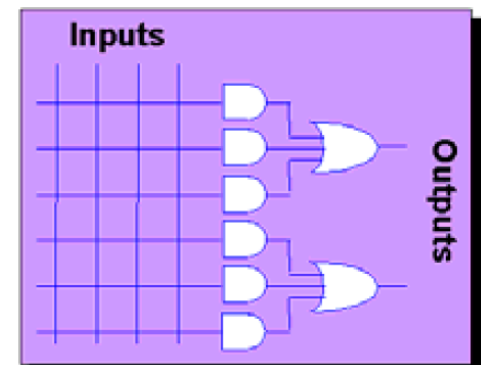
- Simple Programmable Logic Devices (SPLDs):
 - Read-only memories (ROMs)
 - Programmable Logic Arrays (PLAs)
 - Programmable Array Logic (PAL)
 - Programmable Logic Devices (PLDs)
 - Generic Array Logic (GAL)

The PLA has both a programmable AND array and a programmable OR array and looks



Signetics - Cline - 1975

The PAL has a programmable AND array with a fixed OR array and looks like this:



Which one is PLA, or PAL?

SPLDs: Read-Only Memories (ROMs)

- Array of semiconductor devices that are interconnected to store an array of binary data.
- Once data is stored in the ROM, it can be read out but it cannot be changed.
- Outputs can be looked up: truth table.
- Basic types: mask-programmable ROMs, user-programmable ROMs (PROMs), erasable programmable ROMs (EPROMs), electrically erasable and programmable ROMs (EEPROMs), and flash memories.

SPLDs: ROMs

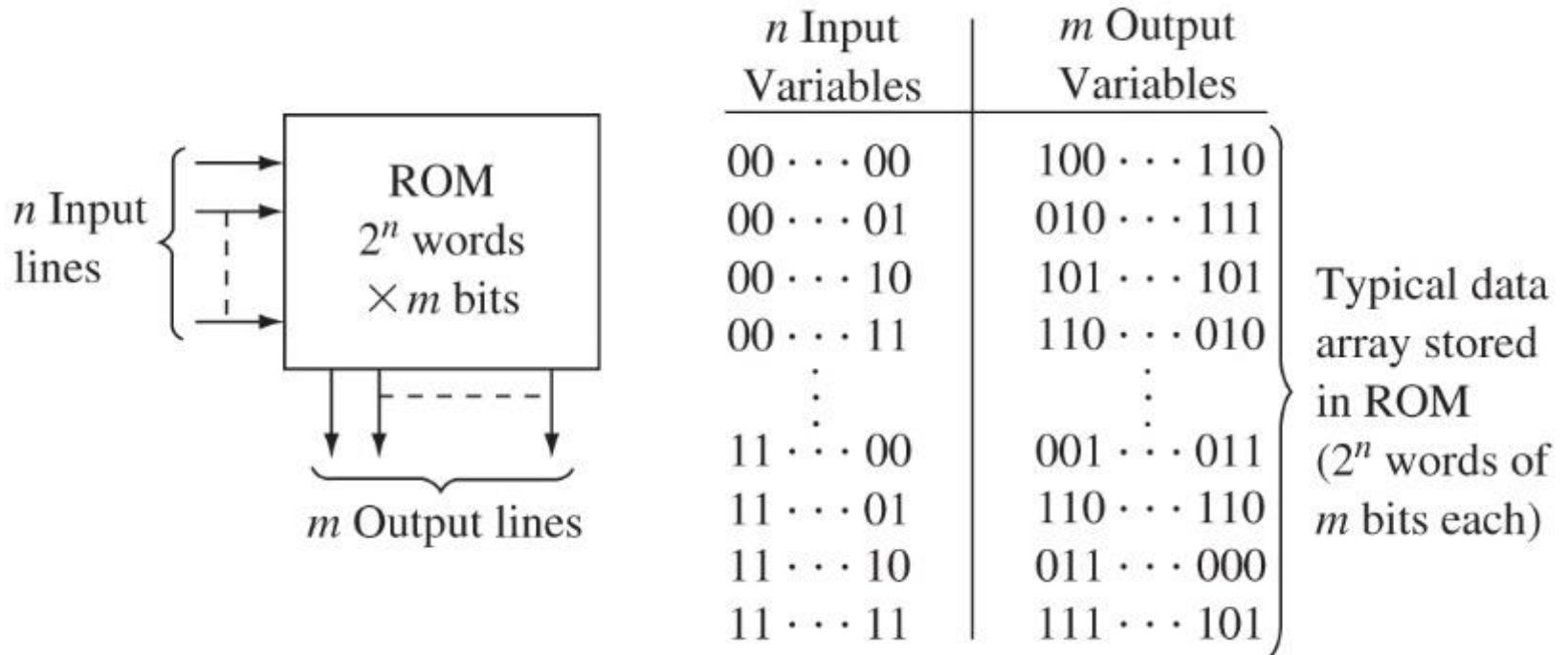
- Types:
 - Mask-programmable ROMs: data array is permanently stored at the time of manufacture.
 - PROMs: one-time programmable.
 - EPROMs: use a special charge-storage mechanism to enable or disable the switching elements in the memory array. Can be erased using ultraviolet light.
 - EEPROMs: similar to EPROM except that erasure is done using electrical pulses. Can be erased and reprogrammed only a limited number of times.
 - Flash memories: similar to EEPROMs, except that they use a different charge-storage mechanism. Usually have built-in programming and erasure capability.

SPLDs: ROMs

- A ROM that has n input lines and m output lines contains an array of 2^n words, and each word is m bits long.
- A $2^n \times m$ ROM can realize m functions of n variables since it can store a truth table with 2^n rows and m columns.
- Typical sizes for commercially available ROMs: 32 words x 4 bits to 512K words x 8 bits, or larger.

SPLDs: ROMs

- ROM with n inputs and m outputs:



SPLDs: ROMs

- Example:
 - Compute the size of the ROM required to implement an 8-to-3 priority encoder, one additional output 'd' to indicate invalid outputs.

SPLDs: ROMs

- Answer:
 - Please refer to the lecture.

SPLDs: Programmable Logic Arrays

- Performs the same basic function as a ROM.
- A PLA with n inputs and m outputs can realize m functions of n variables.
- The internal organization of the PLA is different from that of the ROM: the decoder is replaced with an AND array that realizes selected product terms of the input variables. The OR array OR's together the product terms needed to form the output functions.

SPLDs: PLAs

- Example of PLA that realizes these functions:

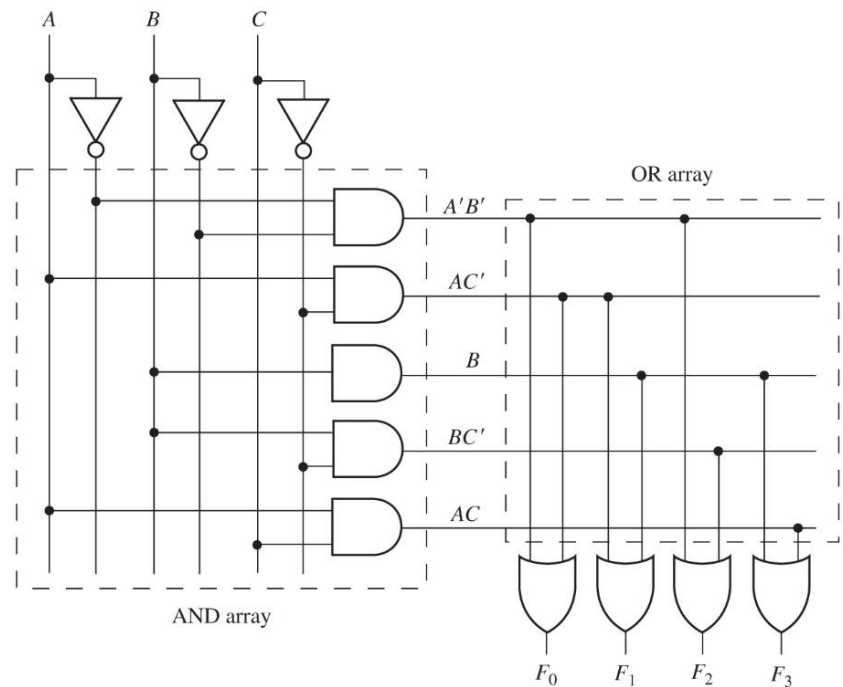
$$F_0 = \sum m(0,1,4,6) = A'B' + AC'$$

$$F_1 = \sum m(2,3,4,6,7) = B + AC'$$

$$F_2 = \sum m(0,1,2,6) = A'B' + BC'$$

$$F_3 = \sum m(2,3,5,6,7) = AC + B$$

Product Term	Inputs			Outputs			
	A	B	C	F_0	F_1	F_2	F_3
$A'B'$	0	0	—	1	0	1	0
AC'	1	—	0	1	1	0	0
B	—	1	—	0	1	0	1
BC'	—	1	0	0	0	1	0
AC	1	—	1	0	0	0	1



SPLDs: PLAs

- From the preceding example, realize the functions using a PLA:

$$F_1 = \sum m(2,3,5,7,8,9,10,11,13,15)$$

$$F_2 = \sum m(2,3,5,6,7,10,11,14,15)$$

$$F_3 = \sum m(6,7,8,9,13,14,15)$$

- Minimize each function:

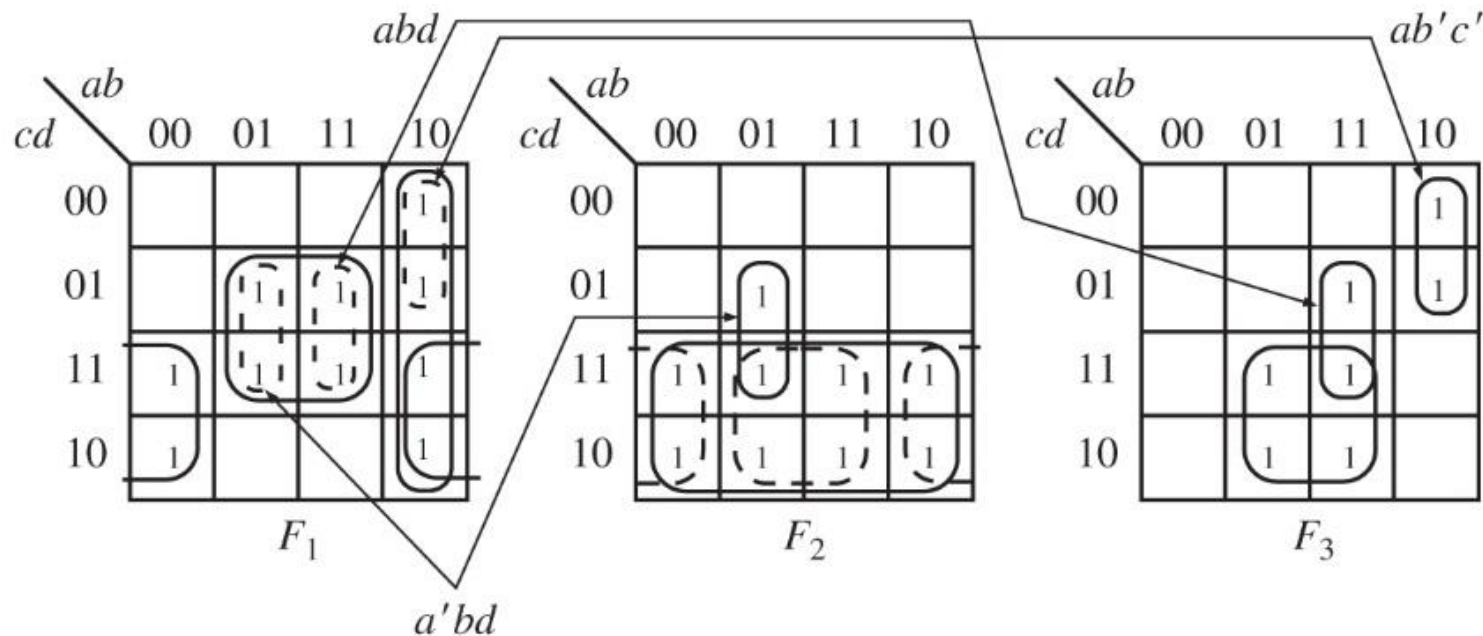
$$F_1 = bd + b'c + ab'$$

$$F_2 = c + a'bd$$

$$F_3 = bc + ab'c' + abd$$

SPLDs: PLAs

- Equations from preceding plotted on Karnaugh maps:



SPLDs: PLAs

- Instead of minimizing each function separately, we want to minimize the total number of rows in the PLA table.
- Reduced PLA table and equations:

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>F</i> ₁	<i>F</i> ₂	<i>F</i> ₃
0	1	—	1	1	1	0
1	1	—	1	1	0	1
1	0	0	—	1	0	1
—	0	1	—	1	1	0
—	1	1	—	0	1	1

$$F_1 = a'bd + abd + ab'c' + b'c$$

$$F_2 = a'bd + b'c + bc$$

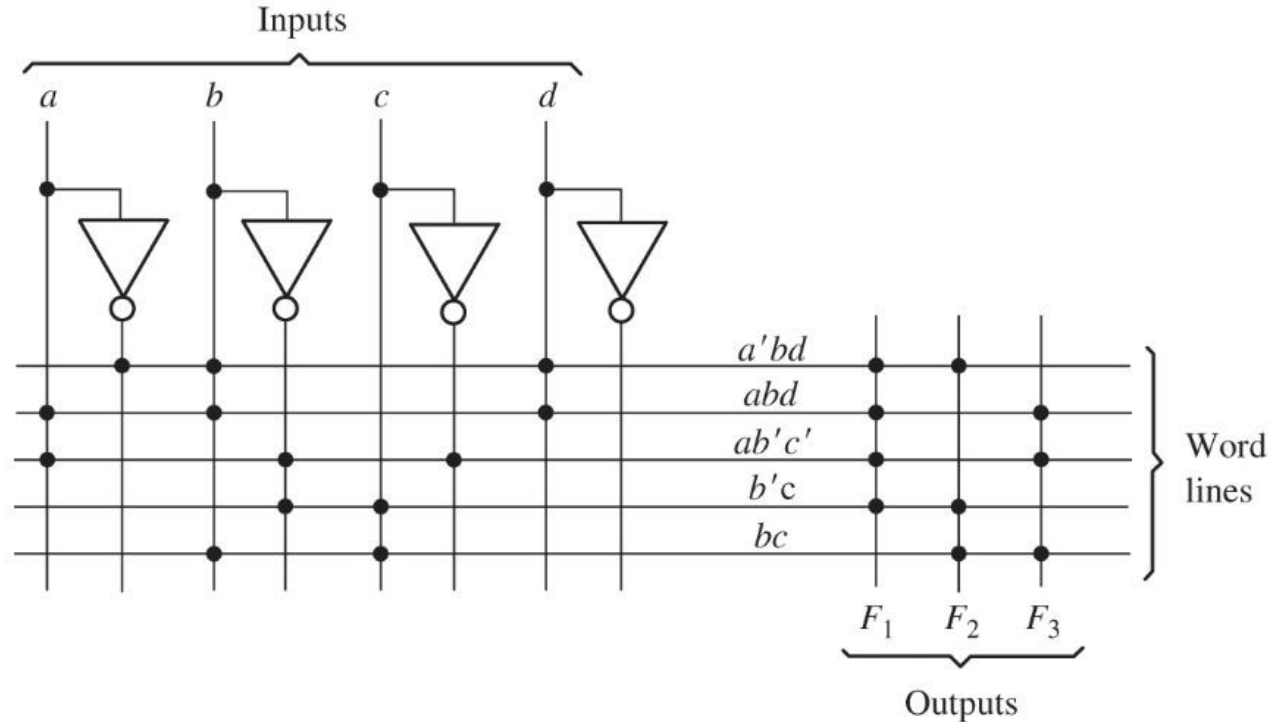
$$F_3 = abd + ab'c' + bc$$

- Note: a PLA table is different from a truth table for a ROM, as each row represents a product term.

SPLD: PLAs

- Only five different product terms are needed.

FIGURE 3-15: PLA
Realization of
Equations (3-4)



SPLDs: Programmable Array Logic

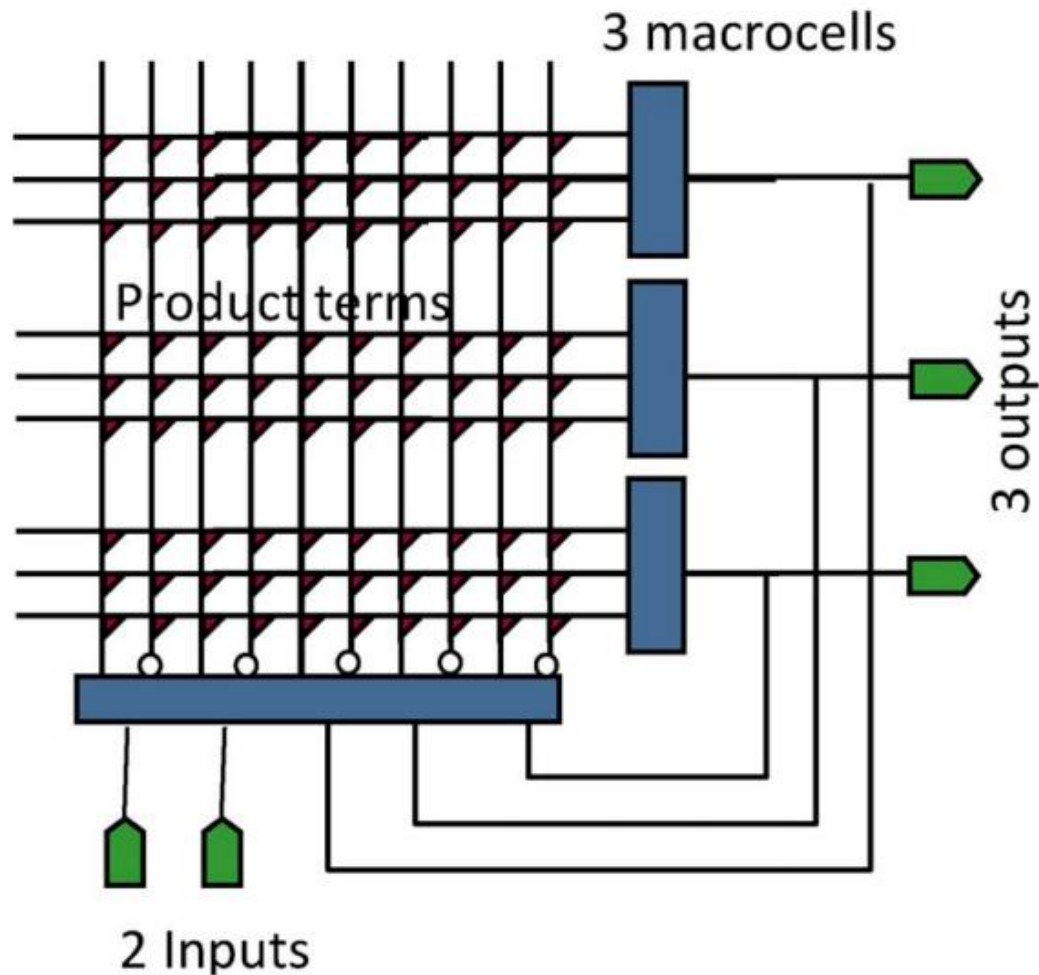
The AND array is programmable and the OR array is fixed. Less expensive than the PLA because only the AND array is programmable; for this reason, designers use PALs to replace individual logic gates when several logic functions must be realized.

When designing with PALs, simplify logic equations.

- AND terms cannot be shared among two or more OR gates.
- The number of AND terms that feed each output OR gate is fixed and limited.
- If the number of AND terms in a simplified function is too large, we may be forced to choose a PAL with more gate inputs and fewer outputs.

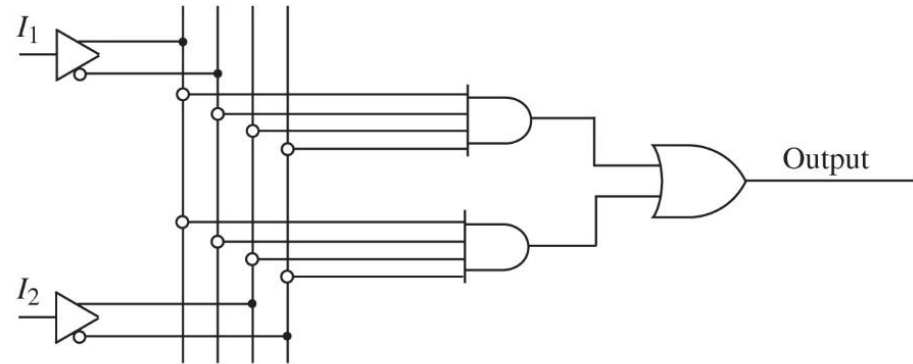
PAL Example

- Generic PAL Architecture

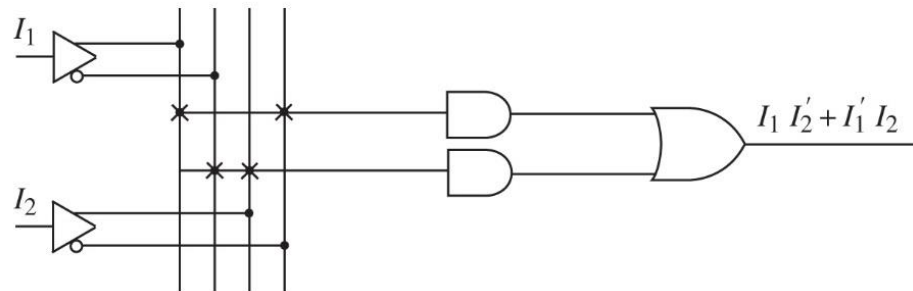


SPLDs: PAL

- Example:
 - $I_1 I'_2 + I'_1 I_2$
 - X's in (b) indicate connections.



(a) Unprogrammed

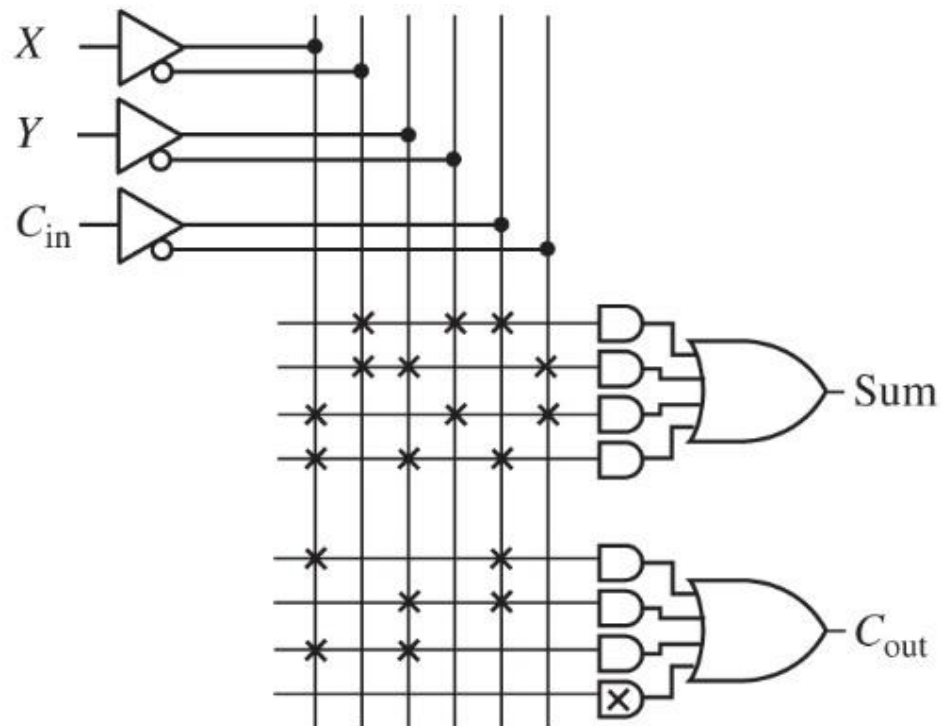


(b) Programmed

SPLDs: Full Adder using a PAL

- Each OR Gate is driven by four AND gates
- Typical PALs have 10-20 inputs, 2-10 outputs

FIGURE 3-17:
Implementation of
a Full Adder Using a
PAL

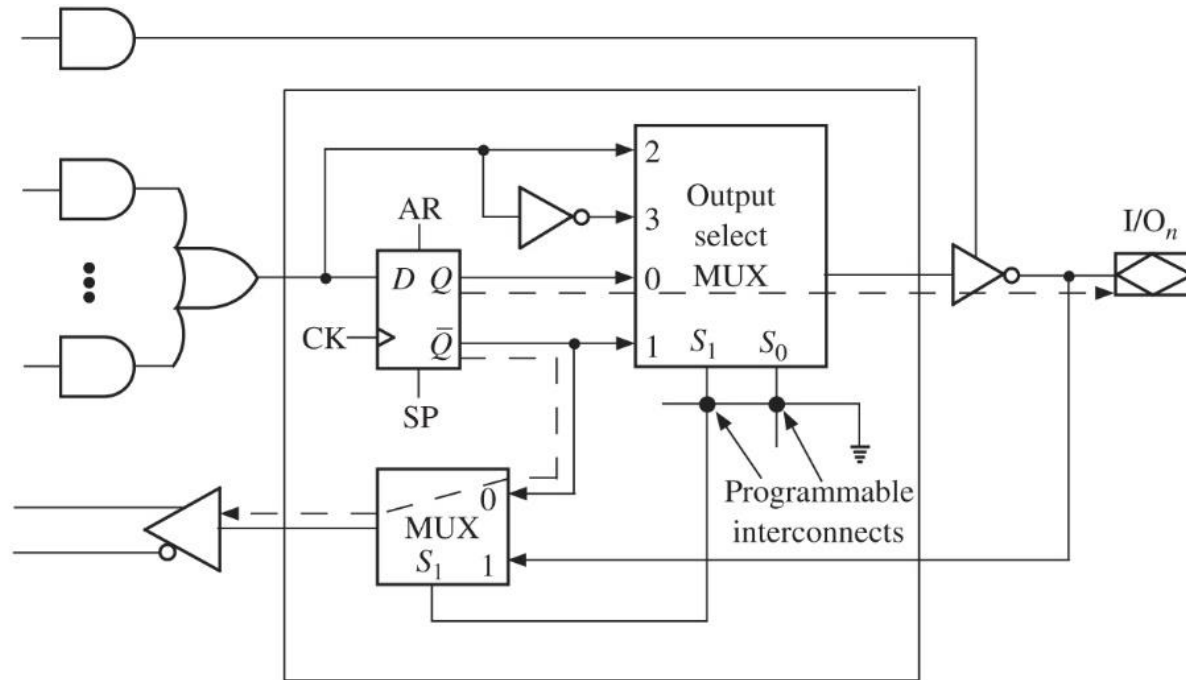


SPLDs: Programmable Logic Devices (PLDs)/Generic Array Logic

- Flash erasable/reprogrammable.
- In addition to the AND-OR arrays that PALs have, most PLDs have some type of a macroblock that contains some multiplexers and some additional programmability. These PLDs are named with reference to their input and output capability.
- All typically have tristate buffers at the outputs and some of them have a dedicated output enable.

SPLDs: PLDs/GAL

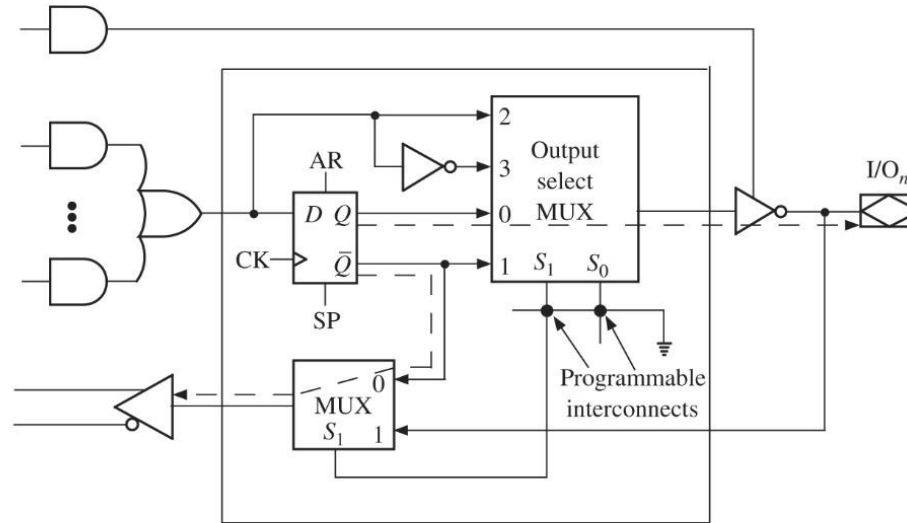
- PLD 22CEV10 output macrocell:



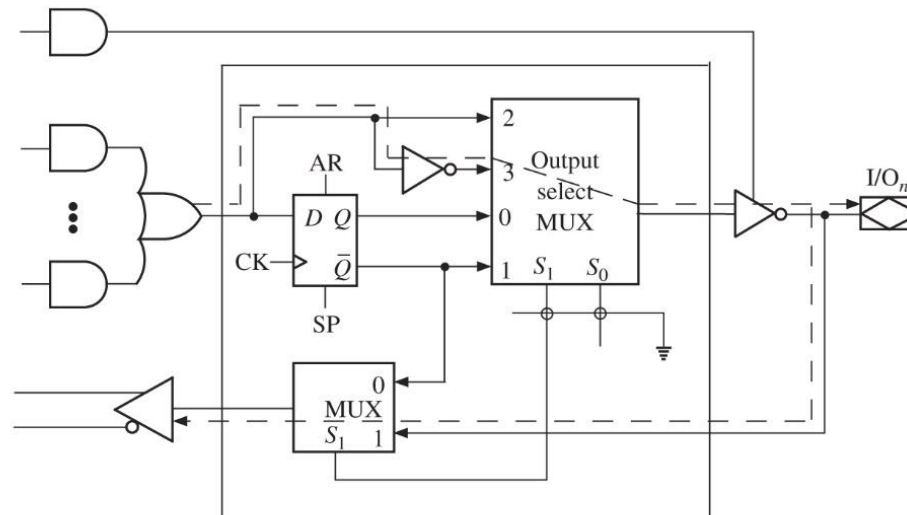
(a) Paths with $S_1 = S_0 = 0$

PLD Output Macrocell Example

FIGURE 3-20: PLD Output Macrocell



(a) Paths with $S_1 = S_0 = 0$



(b) Paths with $S_1 = S_0 = 1$

Complex Programmable Logic Devices (CPLDs)

- Programmable ICs equivalent to several PLDs (that are interconnected using crossbar-like switch) in the same chip.
- When storage elements such as flip-flops are also included on the same IC, a small digital system can be implemented with a single CPLD.
- Essentially, a CPLD is an IC that consists of a number of PAL-like logic blocks together with a programmable interconnect matrix.

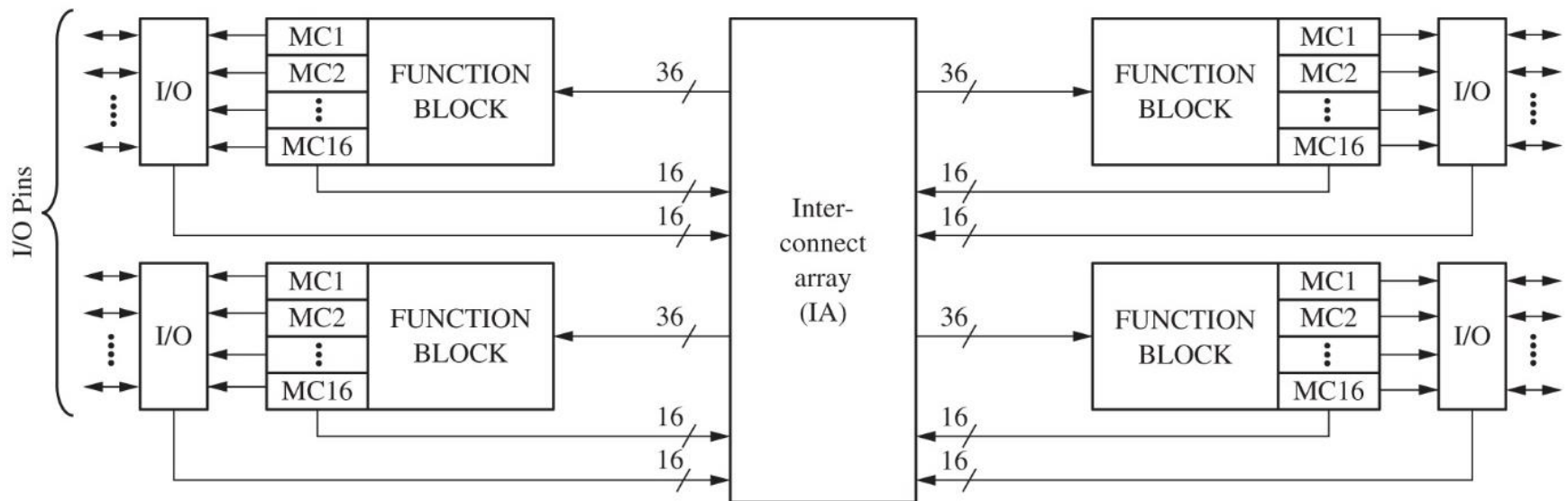
CPLDs

- $N \times M$ crossbar switch: each of the N input lines can be connected to any of the M output lines simultaneously.
 - It is expensive to build the switches, but using them results in predictable timing.
- Many CPLDs are electronically erasable and reprogrammable: EPLDs (erasable PLDs).
- A CPLD contains macrocells grouped into function blocks.
- Some are based on PALs and some on PLAs.

CPLDs



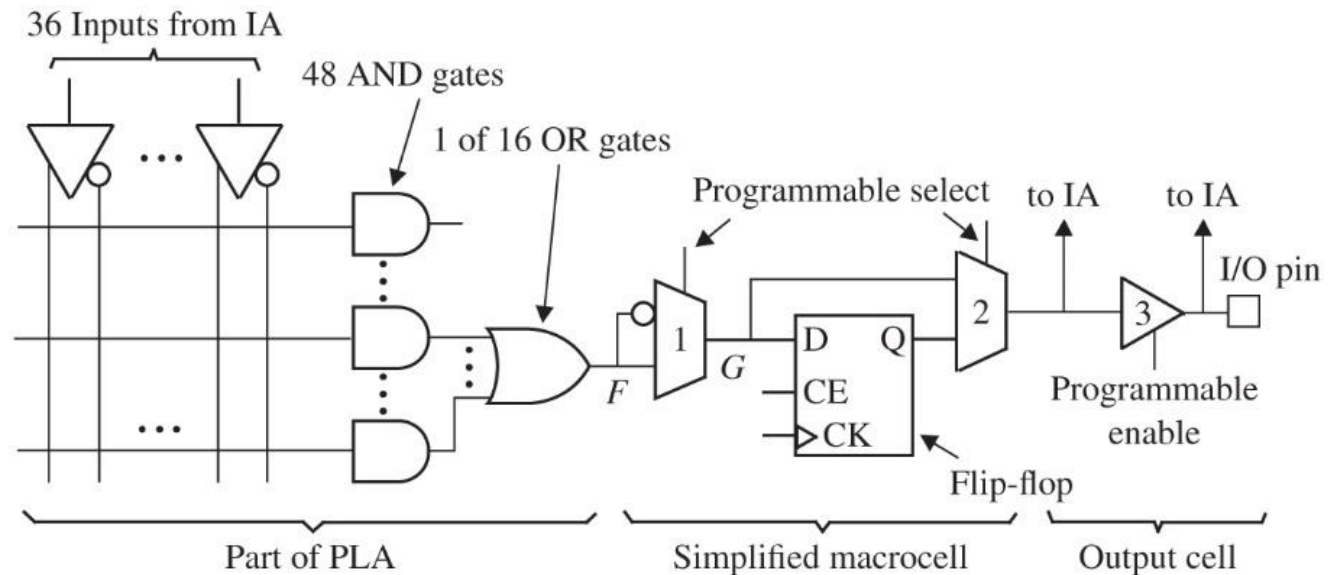
- Example: Xilinx CoolRunner. Has 4 function blocks and each block has 16 macrocells.



CPLD Function Block

- Each OR gate can accept up to 48 product term inputs from the AND Array.

FIGURE 3-22: CPLD Function Block and Macrocell (Simplified Version of XCR3064XL)



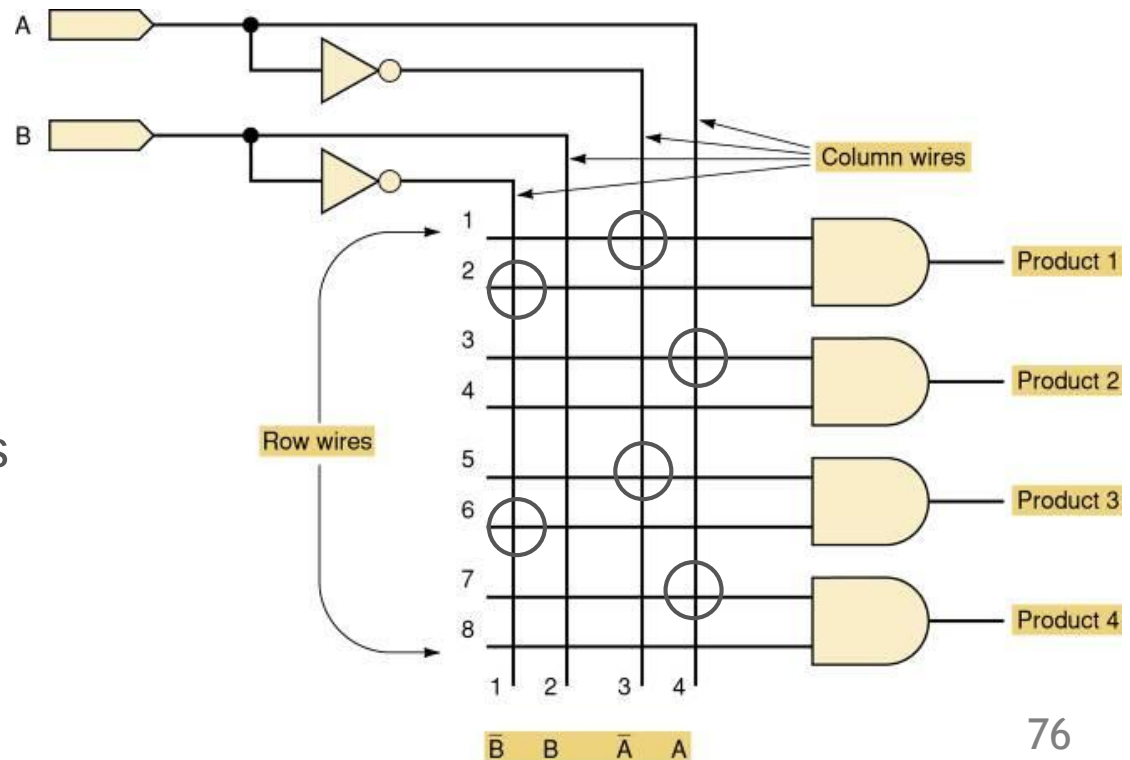
Programmable Logic Devices

- The concept behind programmable logic devices is simple — lots of logic gates in a single IC.
 - Control of the interconnection of these gates electronically.
- PLDs allow the design process to be automated. Designers identify inputs, outputs, and logical relationships.
 - PLDs are electronically configured to form the defined logic circuits.

Programmable Logic Devices

- PLDs use a switch matrix that is often referred to as a programmable array.

By deciding which intersections are connected & which are not, we can “program” the way the inputs are connected to the outputs of the array.



Programmable Logic Devices

- For out-of-system programming the PLD is placed in a programmer, connected to a PC.
- PC software translates and loads the information.



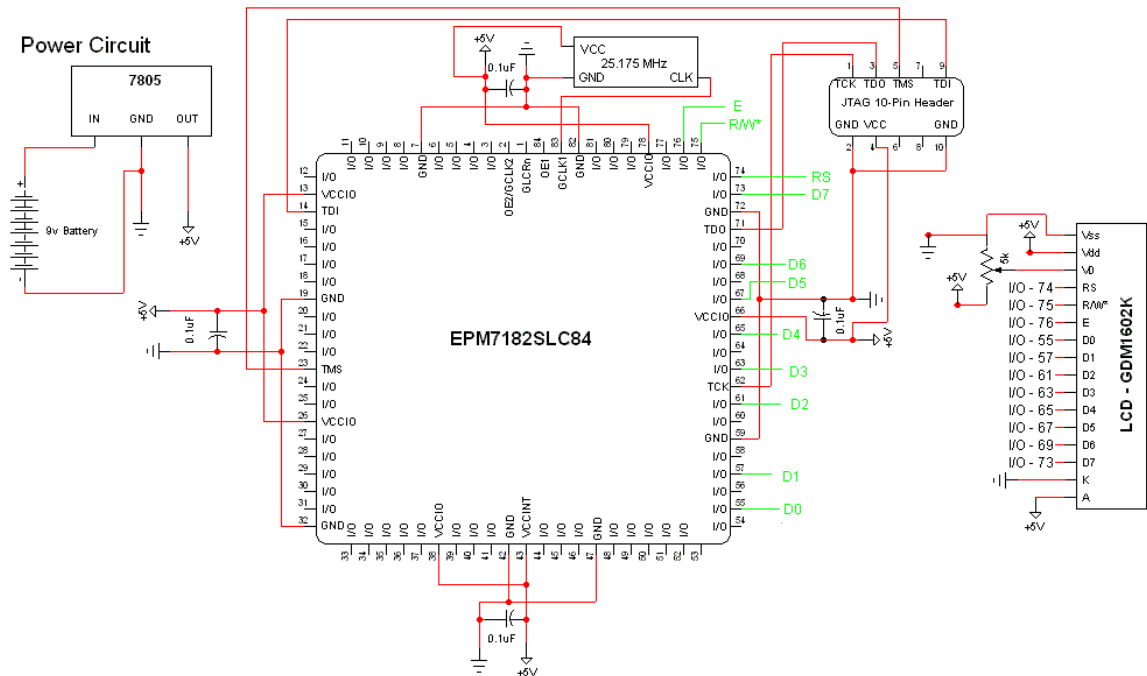
Programmable Logic Devices

- In-system programming is done by connecting directly to “portal” pins while the IC remains in the system.
 - An interface cable connects the PLD to a PC running the software that loads the device.



Programmable Logic Devices

- Logic circuits can be described using schematic, logic equations, truth tables, and HDL.
 - PLD development software can convert any of these descriptions into 1s and 0s and loaded into the PLD.



Outline

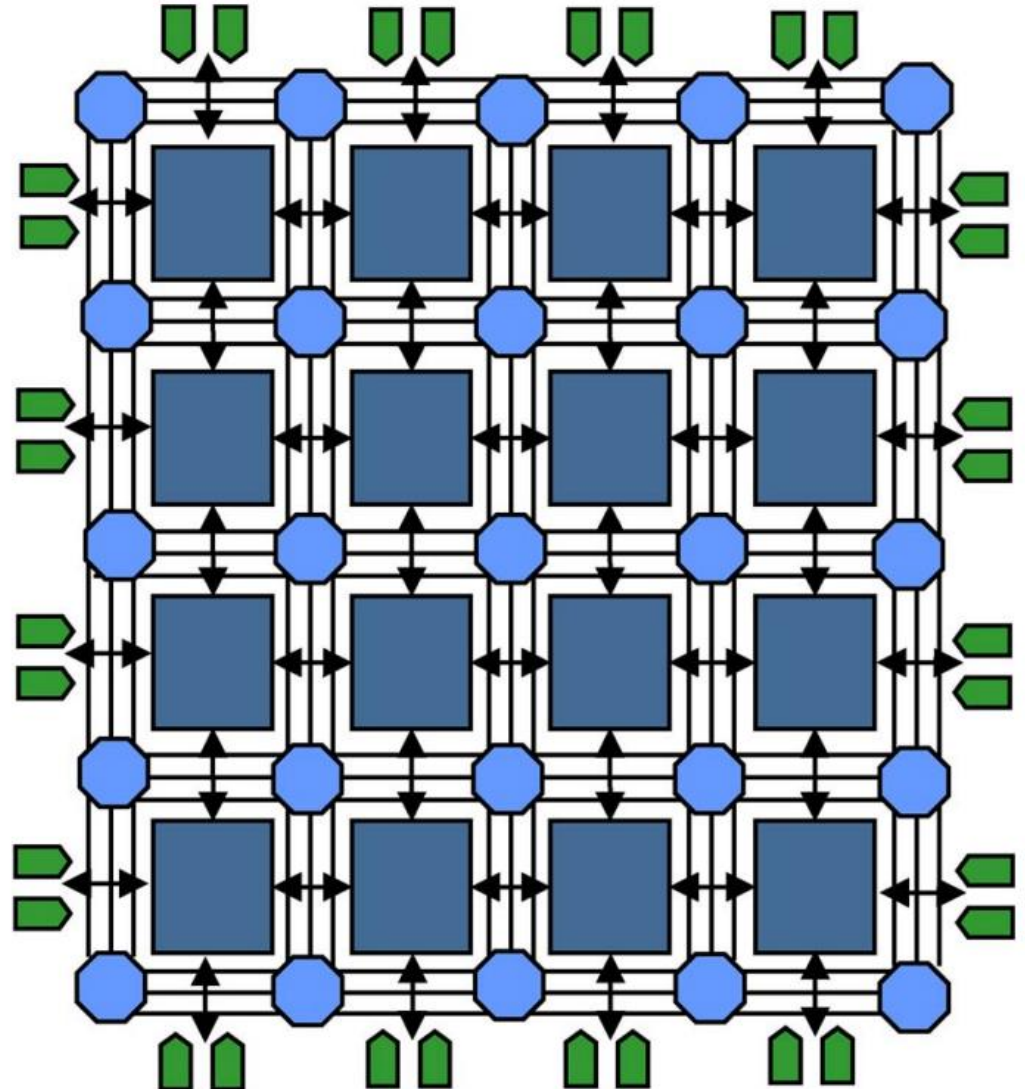
- Reconfigurable computing & Applications
- HDL Design Styles
- Overview of Configurable Logic
- **Overview of FPGA**
- Details of FPGA

Field-Programmable Gate Arrays (FPGAs): Introduction

- ICs that contain an array of identical logic blocks with programmable interconnections.
- Three major programmable elements in FPGAs: the logic block, the interconnect (routing), and the input/output block.
- Programmable logic blocks are created by using multiplexers, look-up tables, and AND-OR or NAND-NAND arrays.
- Have revolutionized the way prototyping and designing is done in the world due to the flexibility offered as it is reprogrammable.
- Vendors: Xilinx, Altera, Lattice Semiconductor, and Microsemi, Atmel, ...

FPGA Overview

- 3 wiring channel per row and column
- Switches
- IO Pads
- Logic cells



FPGA History

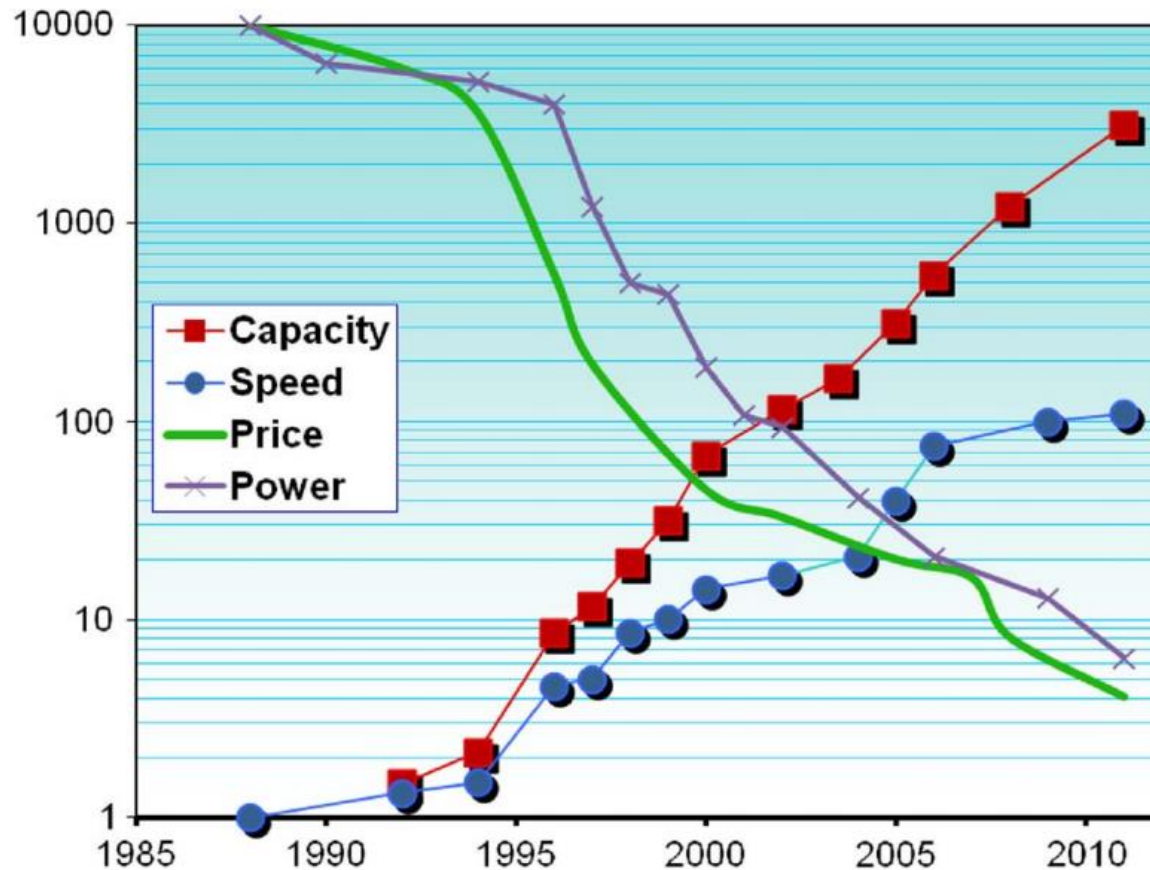
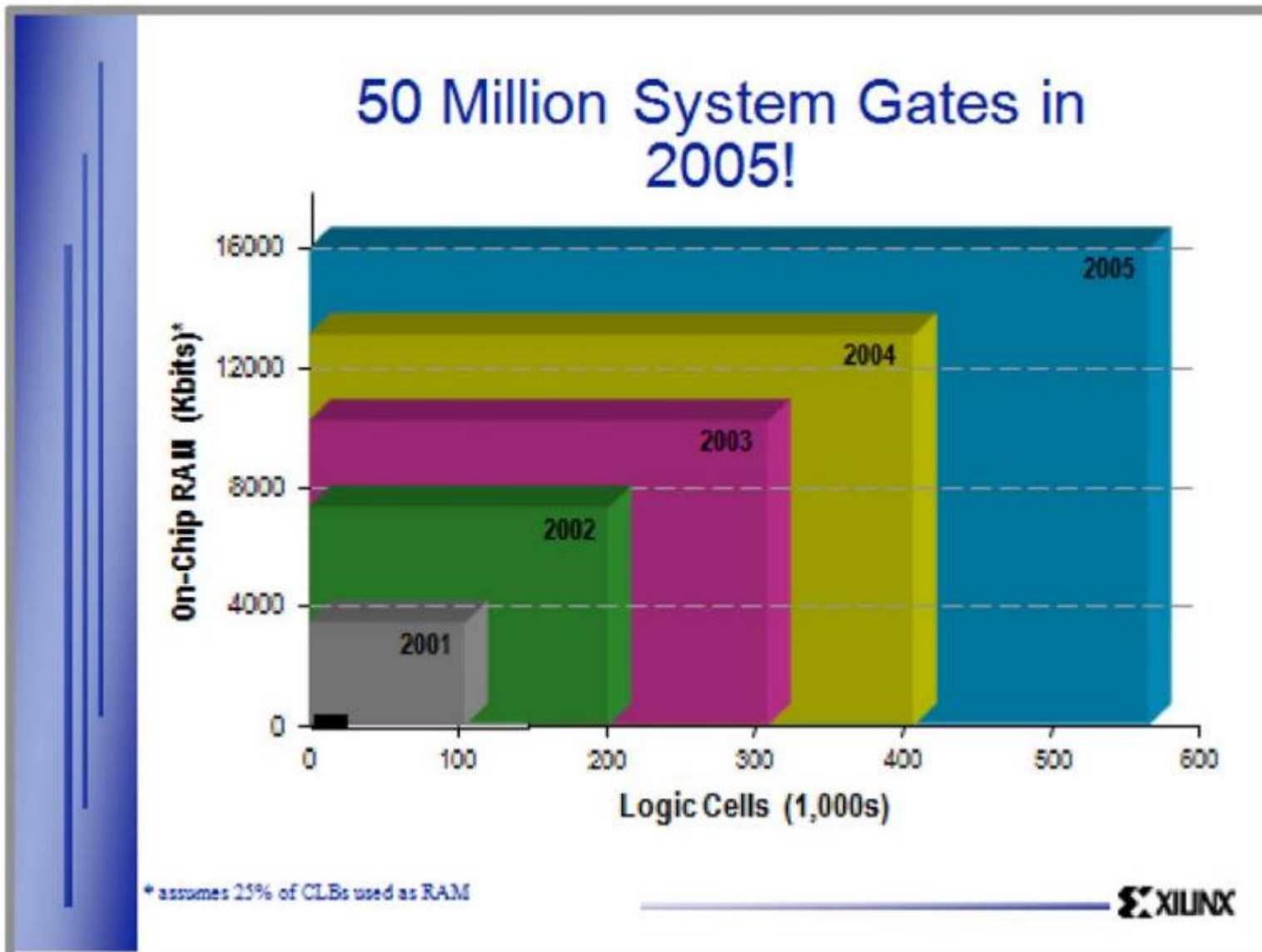
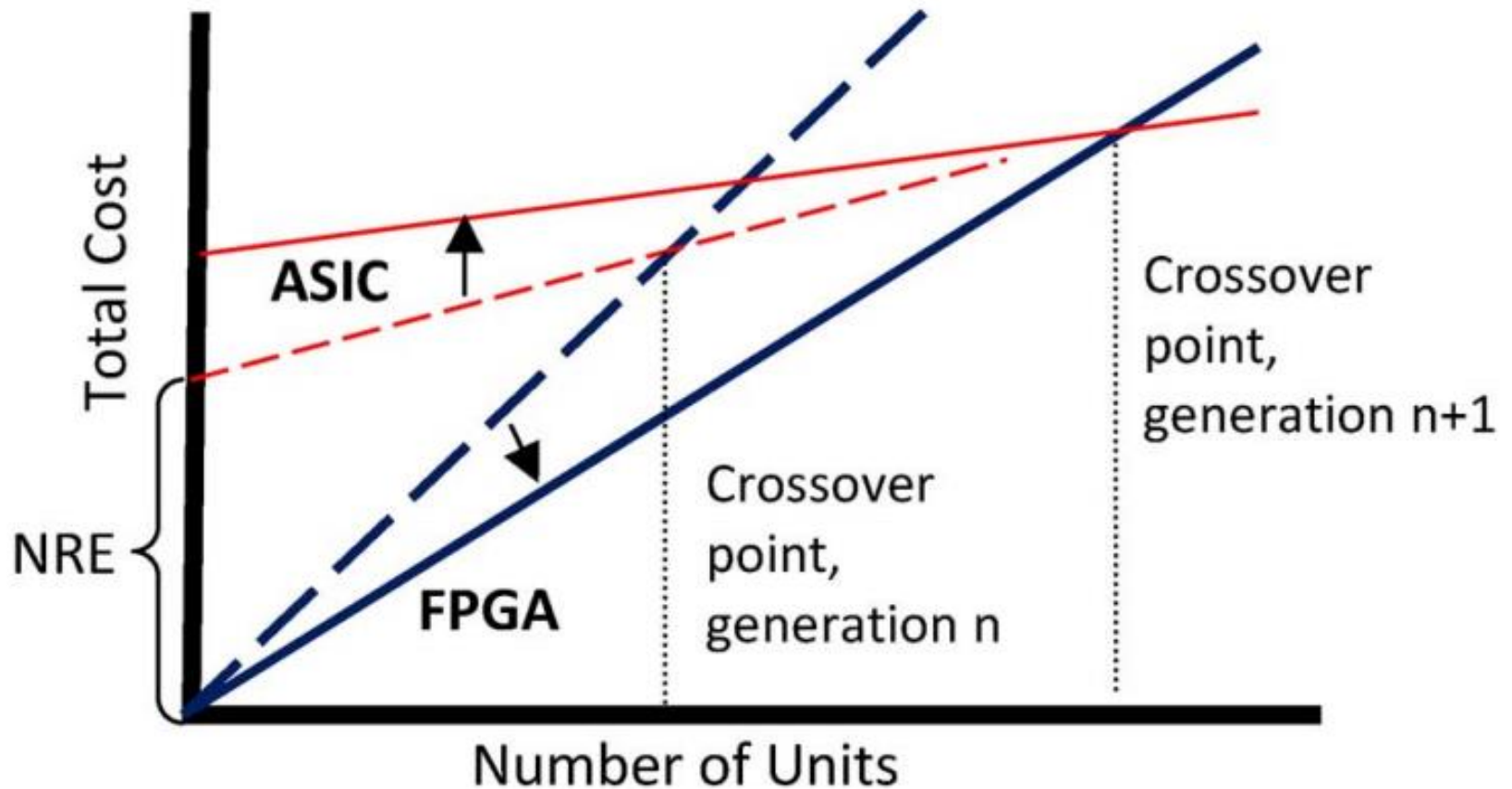


Fig. 1. Xilinx FPGA attributes relative to 1988. Capacity is logic cell count. Speed is same-function performance in programmable fabric. Price is per logic cell. Power is per logic cell. Price and power are scaled up by 10 000 $\times$. Data: Xilinx published data.

How many Gates?



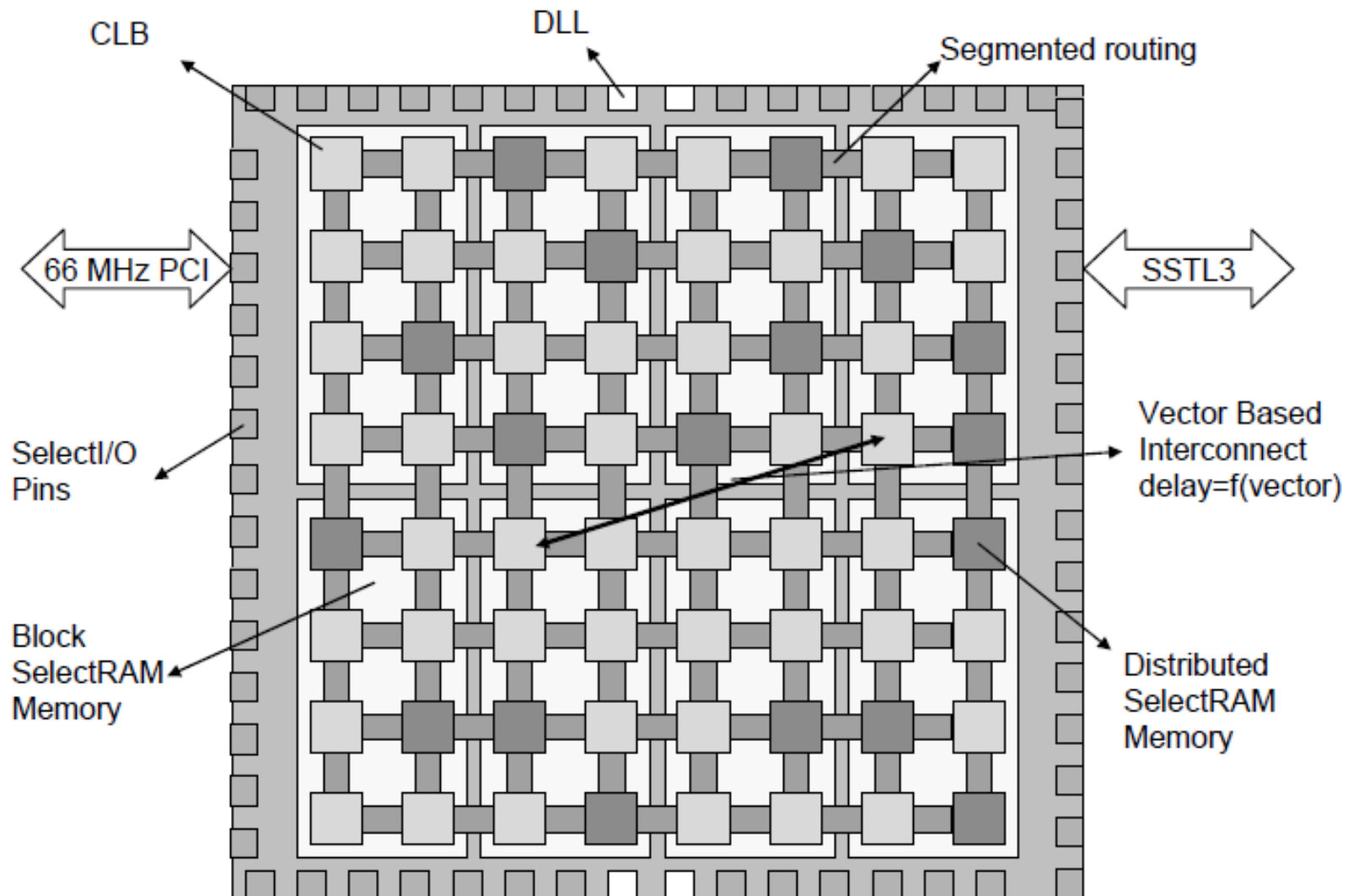
ASIC vs. FPGA



FPGA history

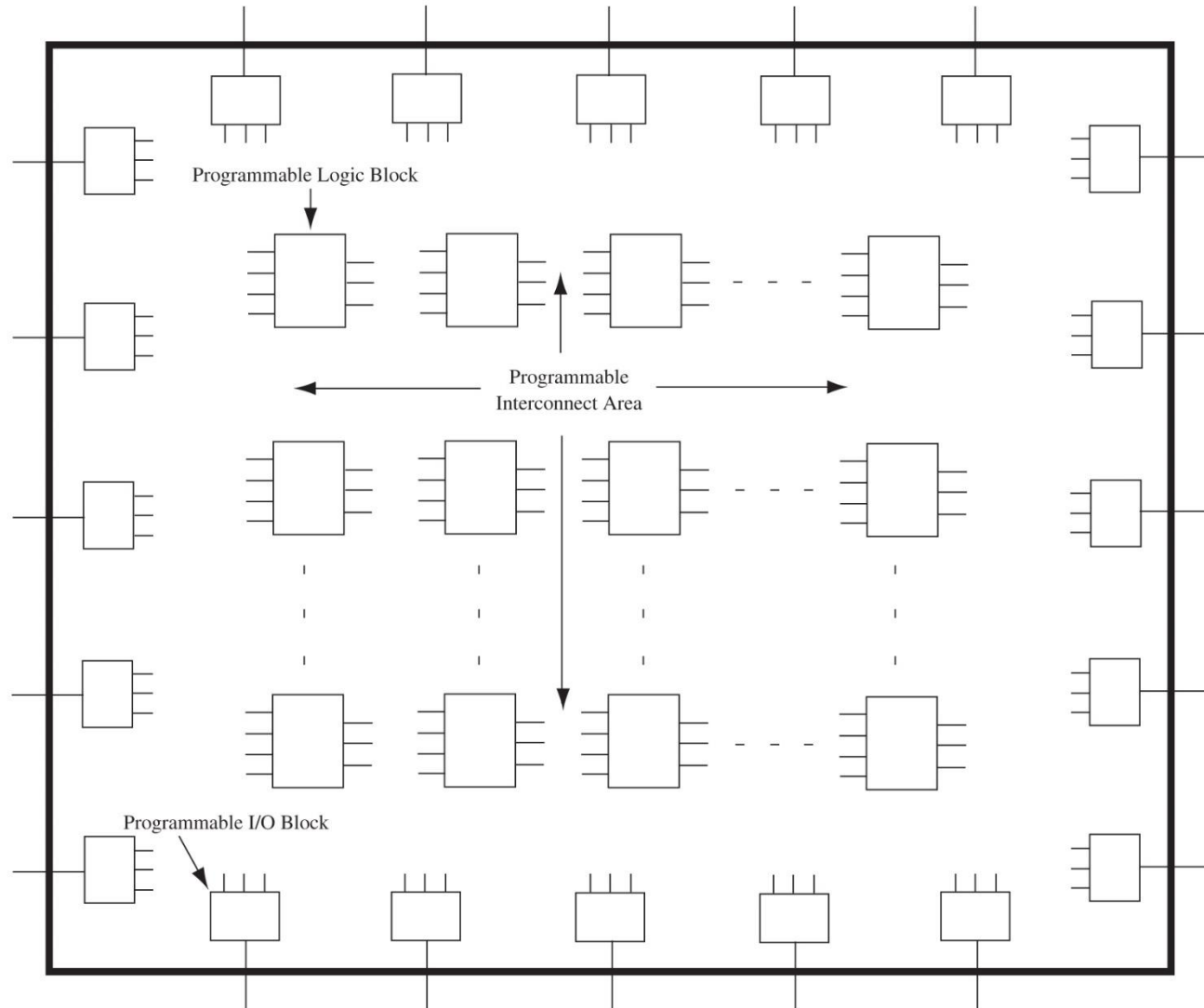
- Array of simple logic cells (4-input LUT) + programmable interconnect (much more fine grained)
- Reconfigurable via static RAM!
 - we can easily change the configuration as many times as we want
 - Rapid Prototyping of Digital Systems
- Very large devices are available
 - XC4062XL has 2,304 slices and equivalent of 130K logic gates (2000)
 - XCV1000 has 12,288 slices and equivalent of 1.5 million gates (2001)
 - XC2V8000 has 46,592 slices and equivalent of 8 million gates (2002)
 - XC2VP125 has 62,568 slices, 4 PowerPC processors and 24 3.125 Gbps transceivers (2003)
 - XC2VP70 has 37,223 slices, 2 PowerPC processors and 20 10.3125 Gbps transceivers (2004)
 - XC4VFX140 has 70,704 slices, 2 PowerPC processors, 24 11.1 Gbps transceivers, 4 ethernet MACs and 192 multipliers (2005)

FPGA Chip Overview

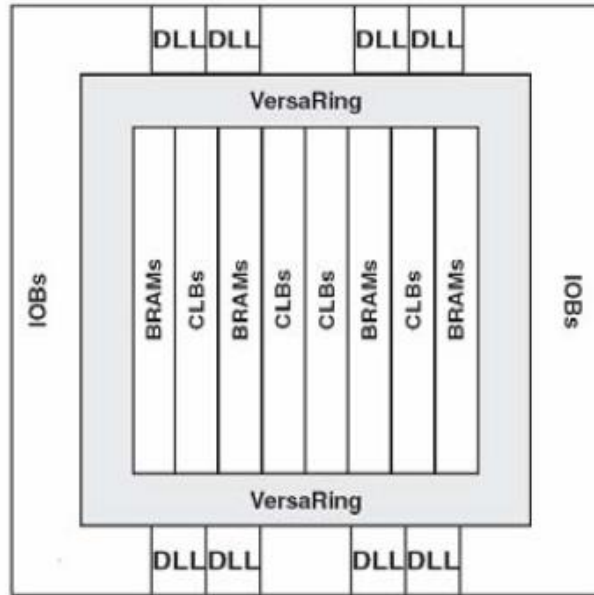


FPGA Layout

FIGURE 3-26: Layout of a Typical FPGA

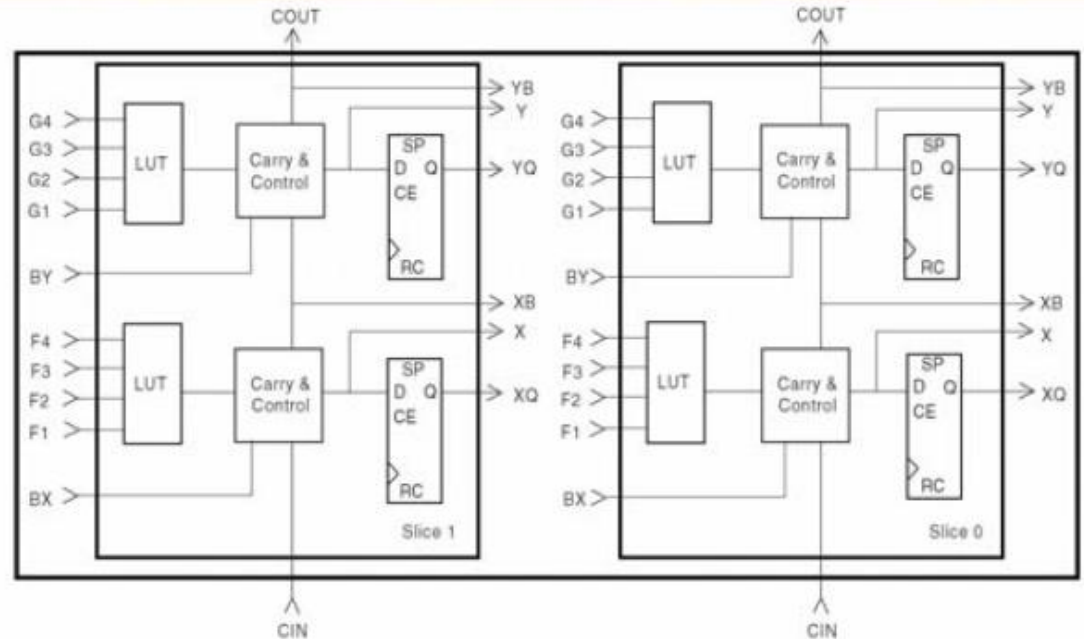


Xilinx Virtex E



ds022_01_121000

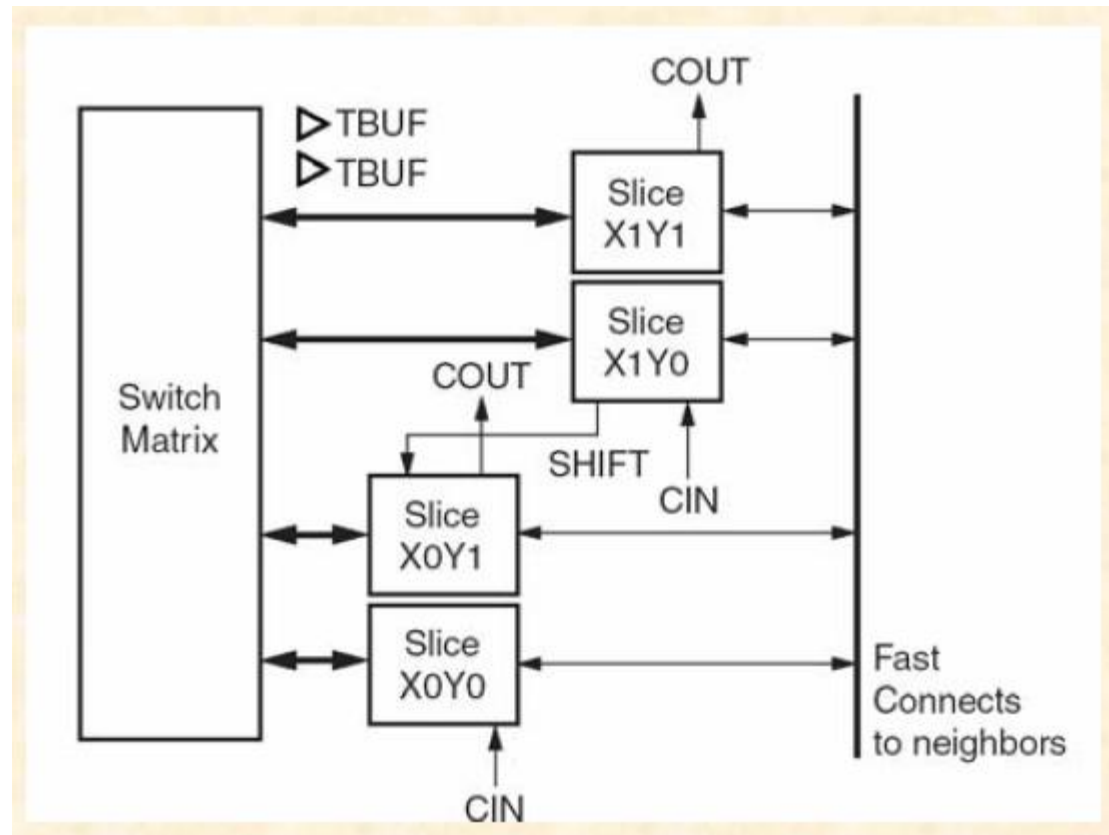
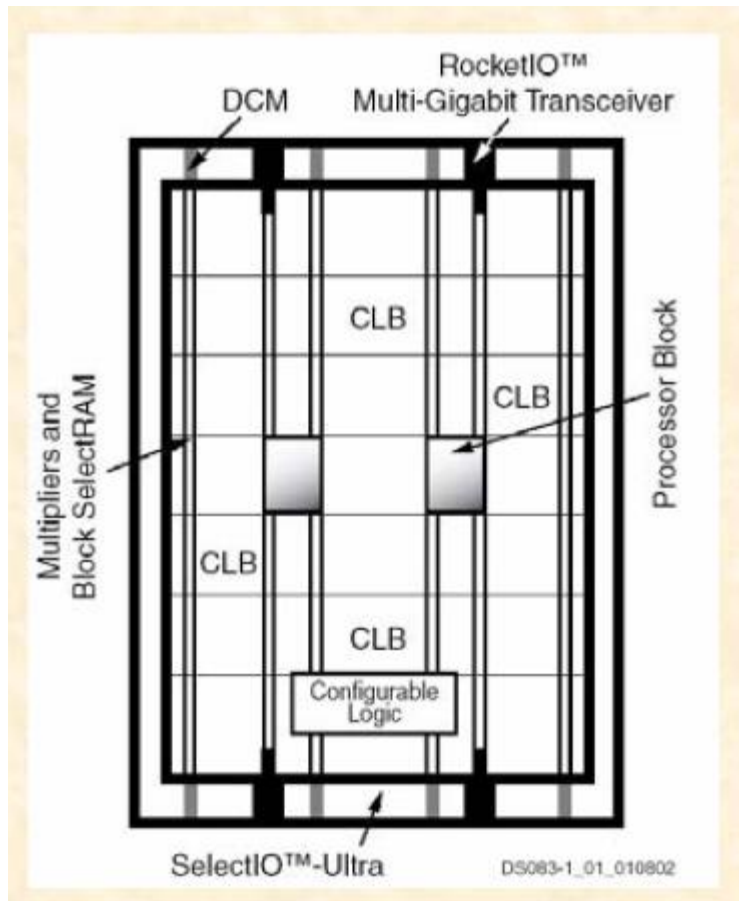
Figure 1: Virtex-E Architecture Overview



ds022_04_121799

Figure 4: 2-Slice Virtex-E CLB

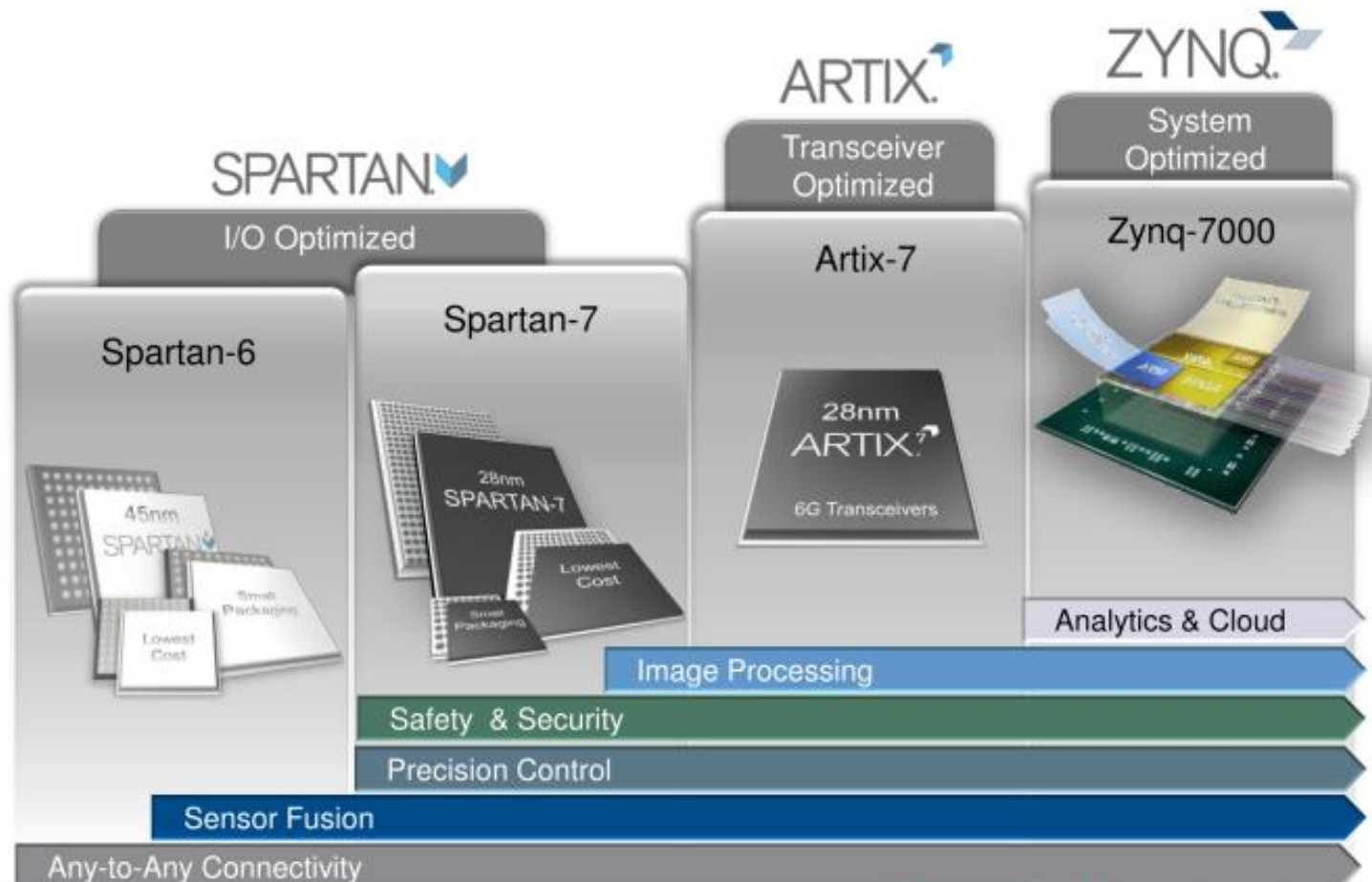
Xilinx Virtex II Pro



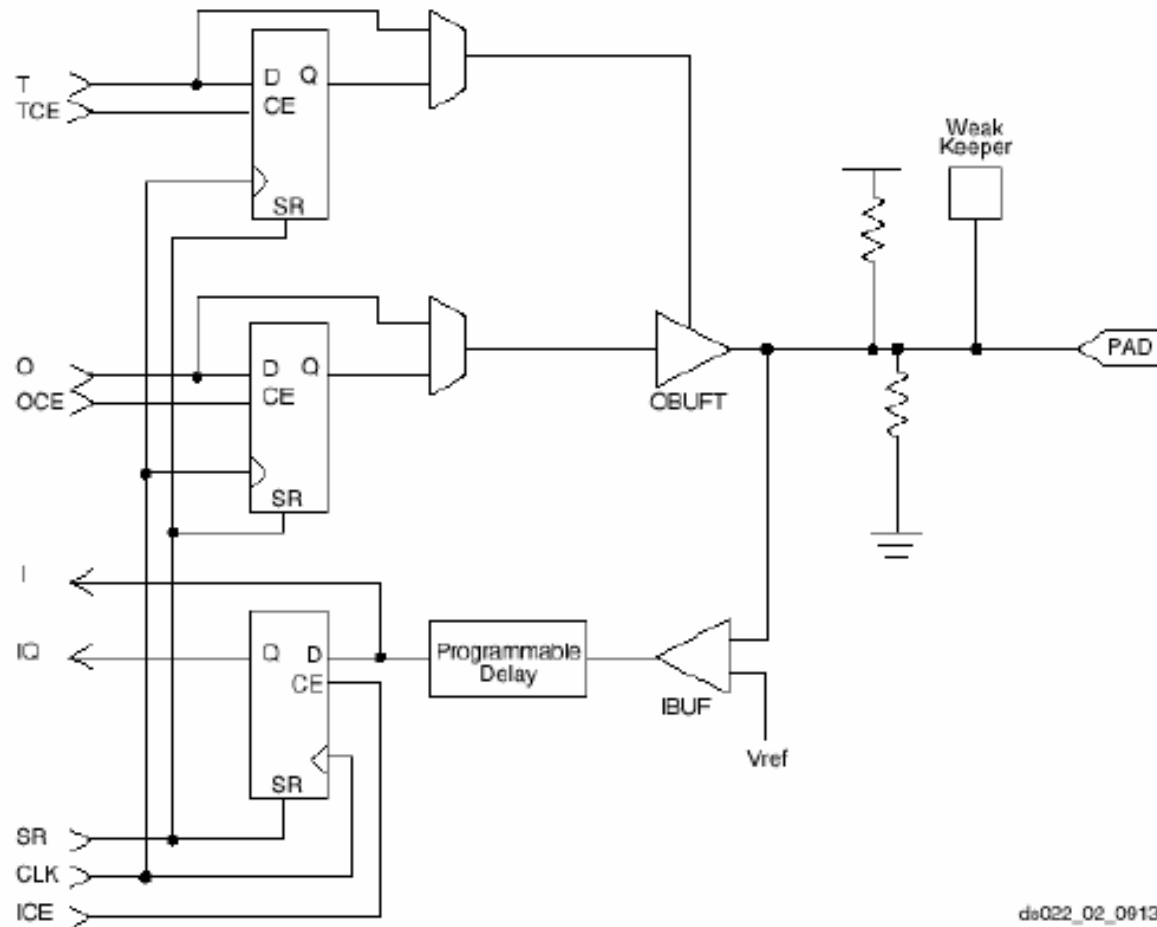
Xilinx Artix



Broadest Cost-Optimized All Programmable Portfolio



Simplified IOB

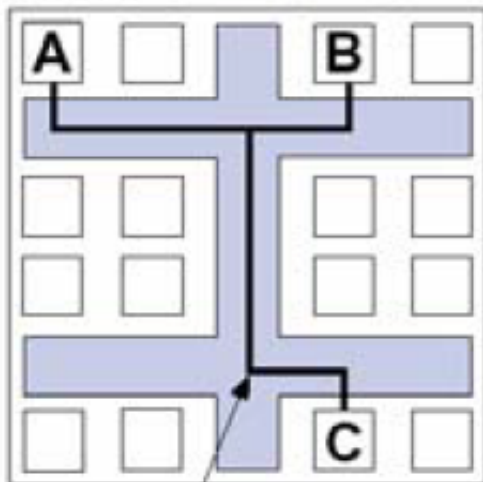


ds022_02_091300

FPGA vs. CPLD

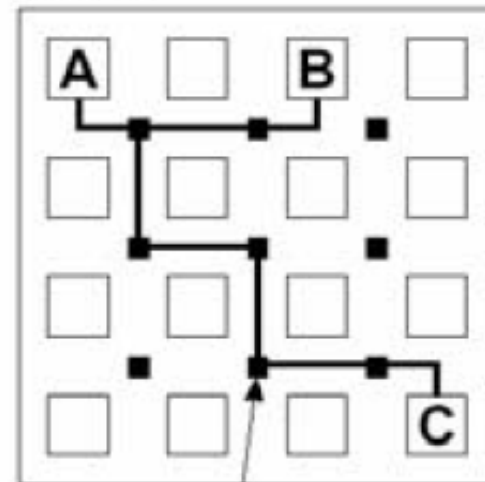
Figure 3. CPLD vs. FPGA Routing Scheme

CPLD Continuous Interconnect Structure



Fixed/Predictable Delay

FPGA Segmented Interconnect Structure



Variable/Unpredictable Delay

Interconnect structures affect the following device characteristics:

- ☐ Performance predictability
- ☐ In-system performance
- ☐ Logic utilization

FPGAs: Advantages

- Advantages:
 - Reduction in manufacturing time as one adopts FPGAs instead of MPGAs.
 - Easier design iterations.
 - Less costly to correct design mistakes or specification changes.
 - Reduced prototyping costs.
 - At low volumes, FPGAs are cheaper than MPGAs.

FPGAs: Disadvantages

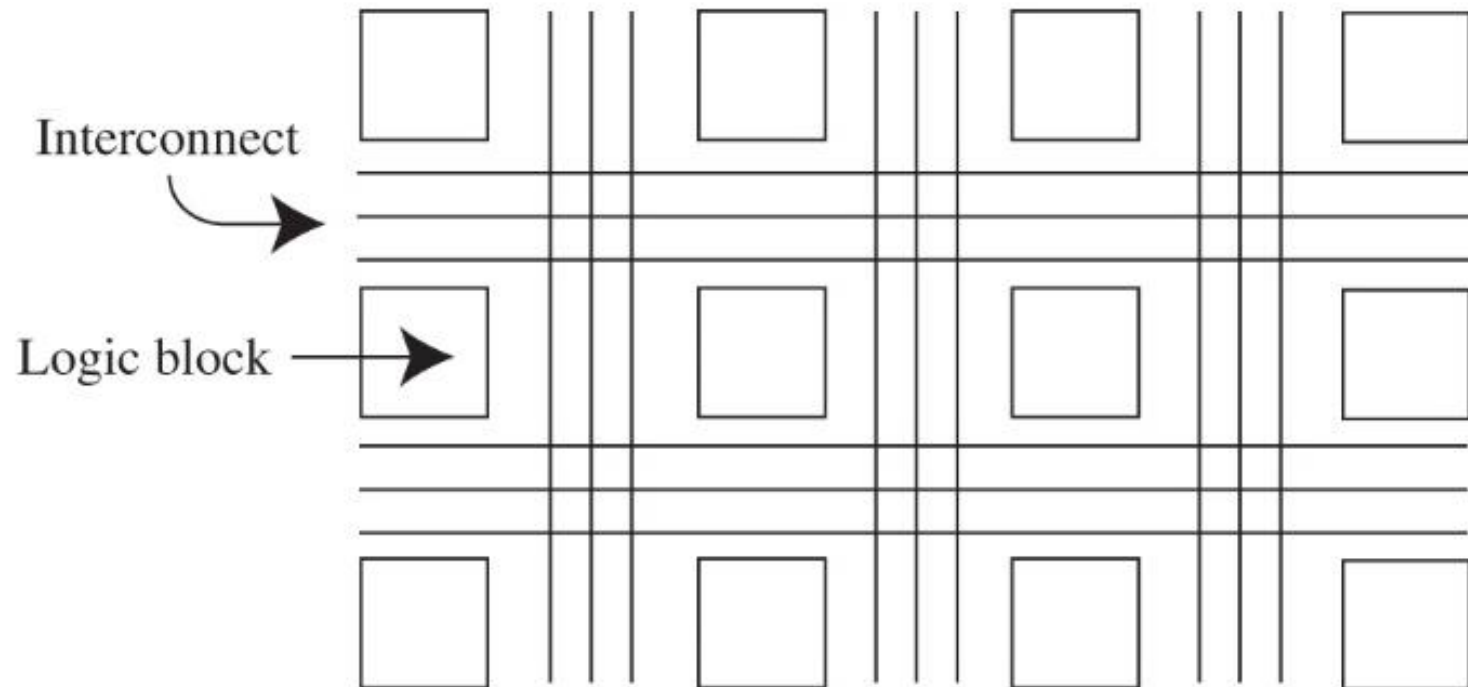
- Disadvantages:
 - Are less dense than traditional gate arrays (MPGAs).
 - A lot of resources are needed to achieve programmability.
 - MPGAs have better performance than FPGAs.
 - Interconnection delays are unpredictable in FPGAs.
 - PLDs such as PALs and GALs are simple and inexpensive.
 - CPLDs are faster, cheaper, and more predictable in timing than FPGAs.
 - However, FPGA is the mainstream now.

FPGAs: Organization

- Matrix-based (symmetrical array):
 - Manufacturer: Most Xilinx FPGAs.
 - Large granularity: capable of implementing 4-variable functions or more.
 - Typically contain 8 x 8 arrays in the smaller chips and 100 x 100 or larger arrays in the bigger chips.
 - Routing: two-dimensional channeled (horizontal and vertical).

FPGAs: Organization

- Matrix-based (symmetrical array):



FPGAs: Organization

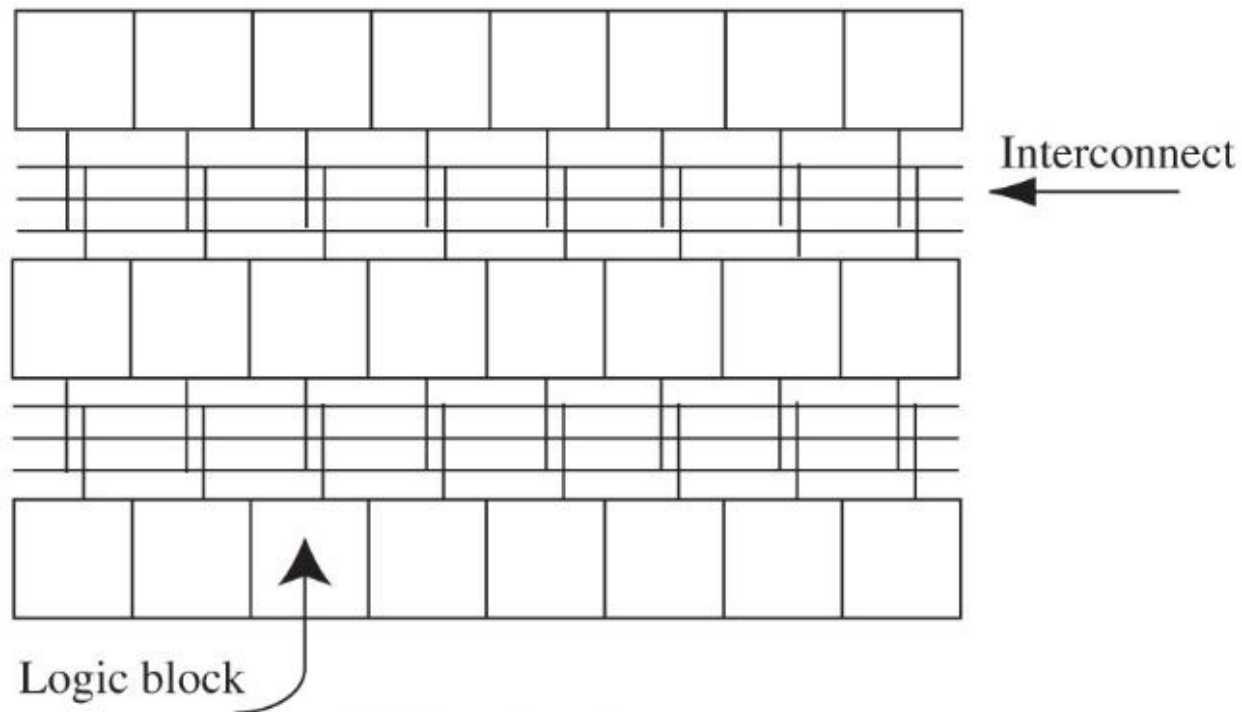
Row-based:

- Inspired by traditional gate arrays.
- Traditional mask-programmable gate arrays use very similar architectures.
- Routing: one-dimensional channeled routing, as the routing resources are located as a channel in between rows of logic resources.
- Manufacturer: some Microsemi FPGAs employ this architecture.



FPGAs: Organization

Row-based:



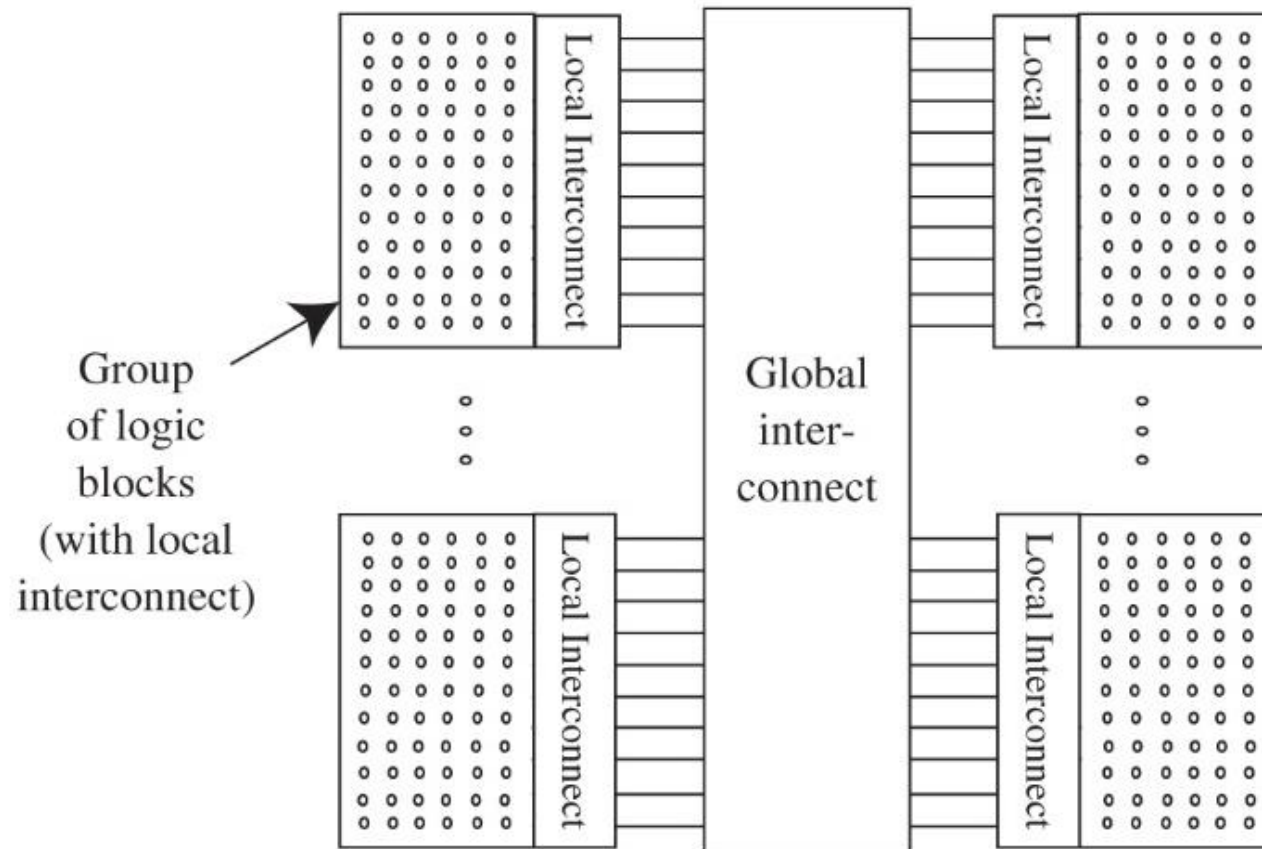
FPGAs: Organization

- Hierarchical:
 - Blocks of logic cells are grouped together by a local interconnect, and several such groups are interconnected by another level of interconnect.
 - Manufacturer: Altera APEX20 and APEX II.



FPGAs: Organization

- Hierarchical:



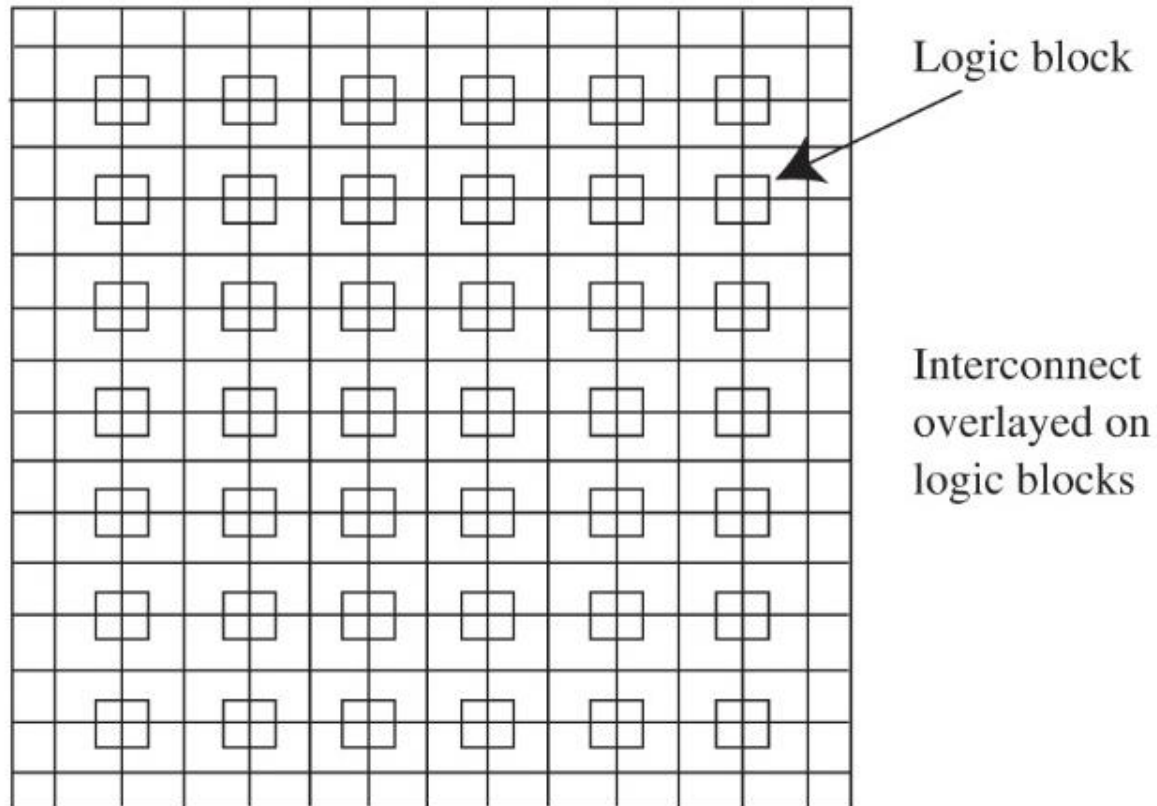
FPGAs: Organization

- Sea-of-gates:
 - Consists of a large number of gates with an interconnect superimposed on the sea of gates.
 - Manufacturers: Plessey - mid-1990's (sea-of-gates), Microsemi Fusion (sea of tiles).



FPGAs

- Sea-of-gates:



FPGAs: Programming Technologies

- FPGAs consist of a large number of logic blocks interspersed with a programmable interconnect.
- Programmable:
 - The logic block: the same building block can be “programmed” or “configured” to create any desired circuitry.
 - Interconnections between the logic blocks.

FPGAs: Programming Technologies

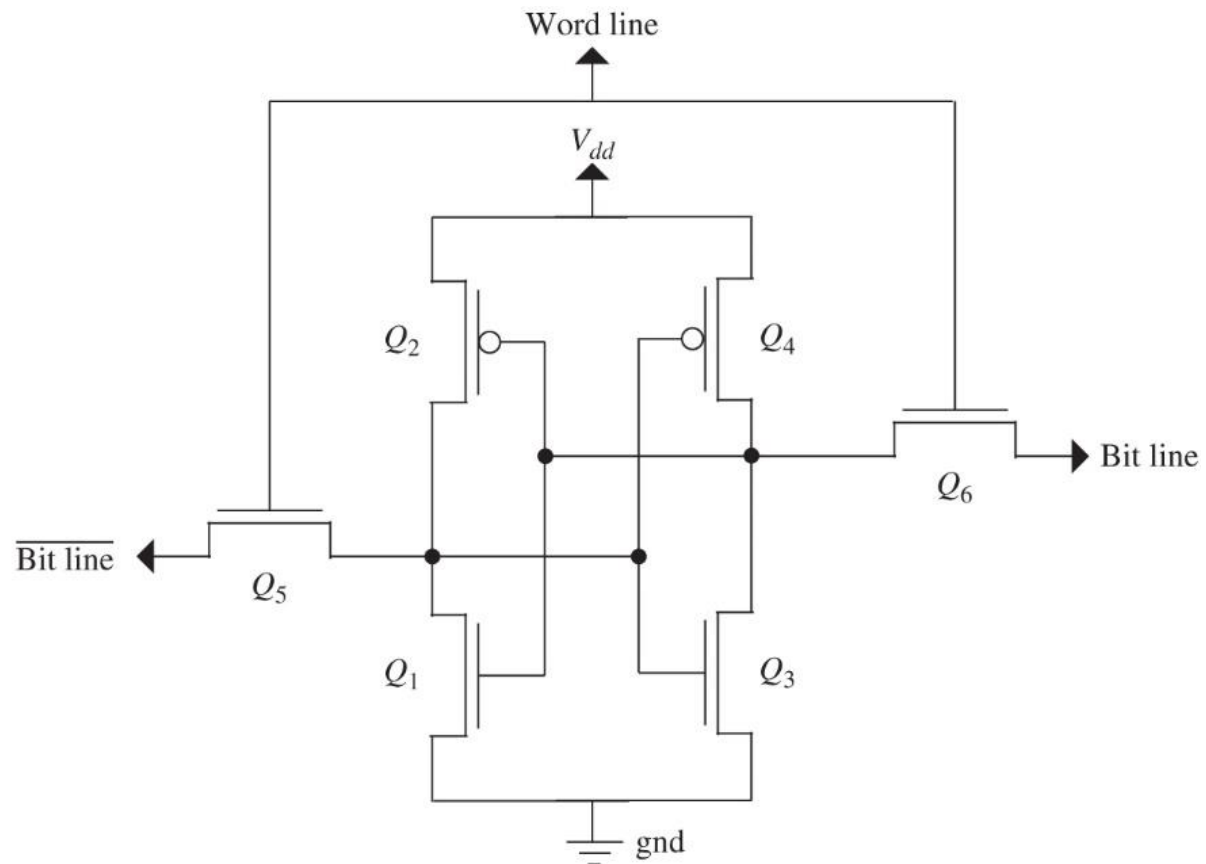
- StaticRAM (SRAM):
 - Involves creating reconfigurability by bits stored in SRAM cells.
 - Logic blocks, I/O blocks, and interconnects can be made programmable by using configuration bits stored in SRAM.
 - Reconfigurable logic blocks can easily be implemented as look-up tables (LUTs).
 - Bits that are stored in the SRAM for deciding the LUT functionality or interconnection are called **configuration bits**.

FPGAs: Programming Technologies

- SRAM:
 - A cell usually takes six transistors.
 - Write Operation: when the *Word Line* is set to high, the values on the *Bit Line* will be latched into the cell.
 - Read operation: done by precharging the *Bit Line* and *Bit Line* to a logic 1 and then setting *Word Line* to high. The contents stored in the cell will then appear on the *Bit Line*.

FPGAs: Programming Technologies

- Typical 6-transistor SRAM cell: (operation details can be learned in a more advanced course)



FPGAs: Programming Technologies

- SRAM Advantages:
 - Volatile: new contents can be written again and again.
 - Fabrication steps are the same for making SRAM cells and for making logic.
- SRAM Disadvantages:
 - Cost: five or six transistors are used for every SRAM cell.
 - Volatile: another device (such as an EPROM) is needed to store the configuration bits.

FPGAs: Programming Technologies

- EPROM/EEPROM:
 - Used to control programmable connections.
 - Compared with SRAMs:
 - Are slower than SRAMs; SRAMs can be programmed faster.
 - EPROMs also require more processing steps than SRAM.
 - EEPROM is similar to EPROM, but removal of the gate charge can be done electrically.

FPGAs: Programming Technologies

- EPROM/EEPROM:
 - Flash memory:
 - A form of EEPROM.
 - Allows multiple locations/segments to be erased in one operation by pulling electrons off.
 - Stores information in floating gate transistors as in EPROM.
 - Cell is read by placing a specific voltage on the control gate.

FPGAs: Programming Technologies

- Antifuse:
 - Programming: changes from high resistance (open) to low resistance (closed) when a high voltage is applied to it.
 - Normally OFF.
 - One-time programmable.
- Antifuse Advantages:
 - Area consumed is small.
 - Connections are faster than SRAM and EEPROM.
- Antifuse Disadvantages:
 - Not reprogrammable.

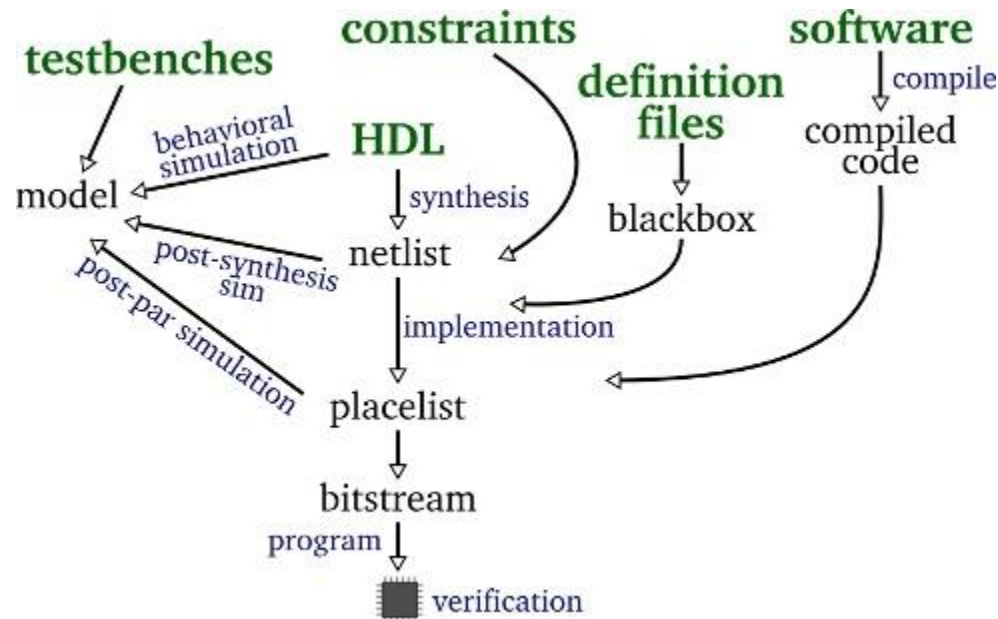
FPGAs: Programming Technologies

Comparison of technologies:

Programming Area Technology	Volatility	Programmability	Overhead	Resistance	Capacitance
SRAM	Volatile	In-circuit reprogrammable	Large	Medium to high	High
EPROM	Nonvolatile	Out-of-circuit reprogrammable	Small	High	High
EEPROM/Flash	Nonvolatile	In-circuit reprogrammable	Medium to high	High	High
Antifuse	Nonvolatile	Not reprogrammable	Small	Small	Small

Download Bitstream

- Can you use your mobile device to check what is the latest device name from Xilinx?
- How do we download the bitstream to the FPGA board?
 - Xilinx software is used to download the “0011... 010101...”
 - Can we protect our bitstream to ensure the copyright?



Outline

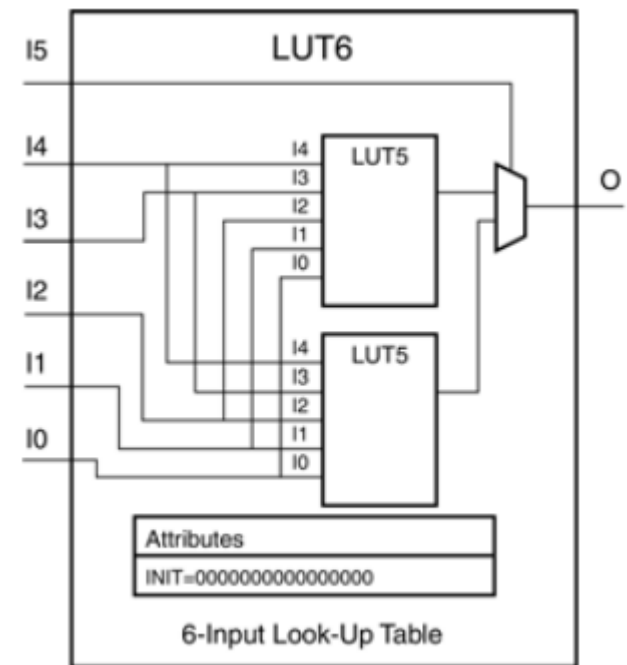
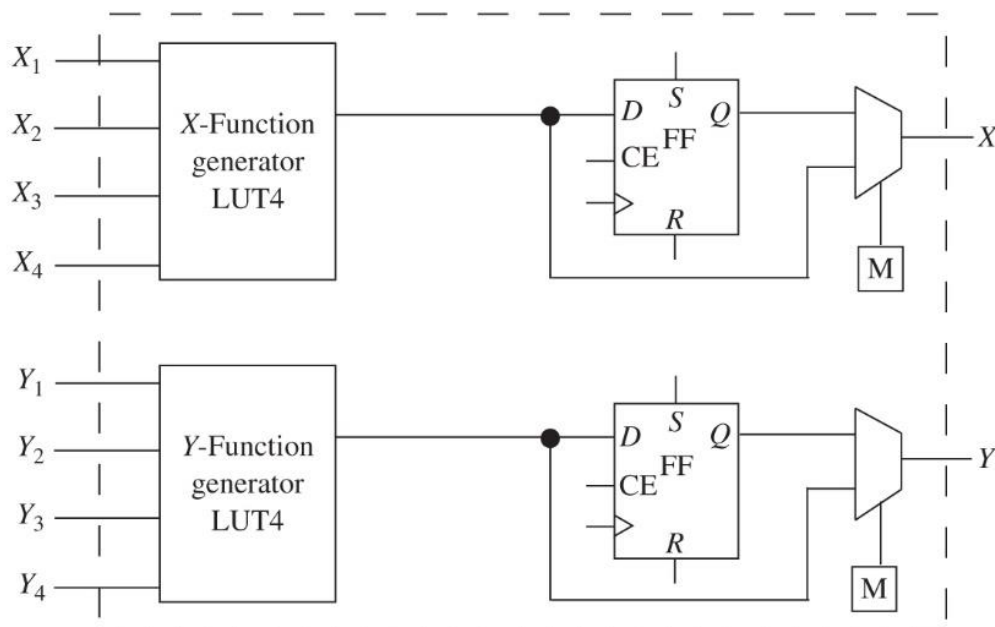
- Reconfigurable computing & Applications
- HDL Design Styles
- Overview of Configurable Logic
- Overview of FPGA
- **Details of FPGA**

FPGAs: Programmable Logic Block Architectures

- Look-Up-Table-Based Programmable Logic Blocks:
 - Many LUT-based FPGAs use a 4-variable look-up table (LUT4) plus a flip-flop as the basic element and then combine them in different ways.
 - Very common for Xilinx and Altera.

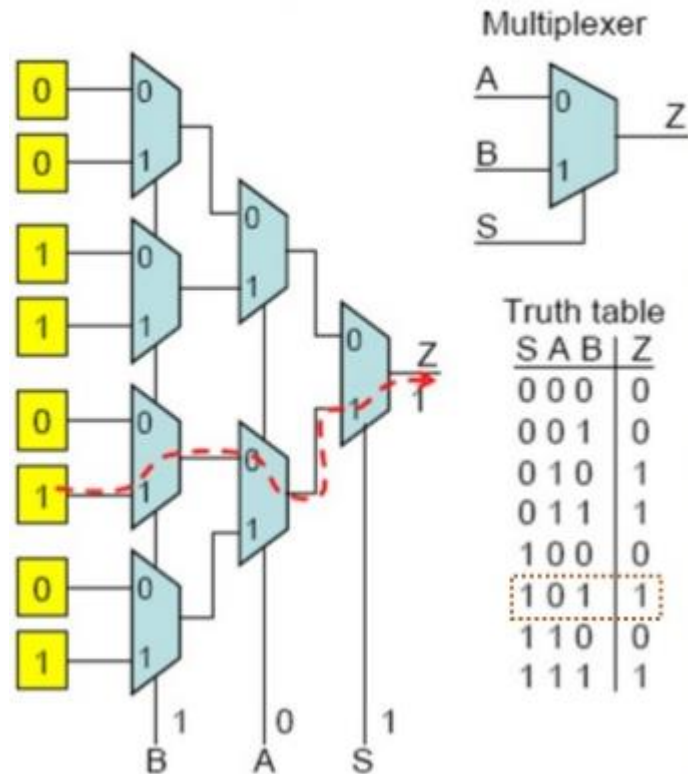
FPGAs: Programmable Logic Block Architectures

- Look-Up-Table-Based Programmable Logic Blocks:



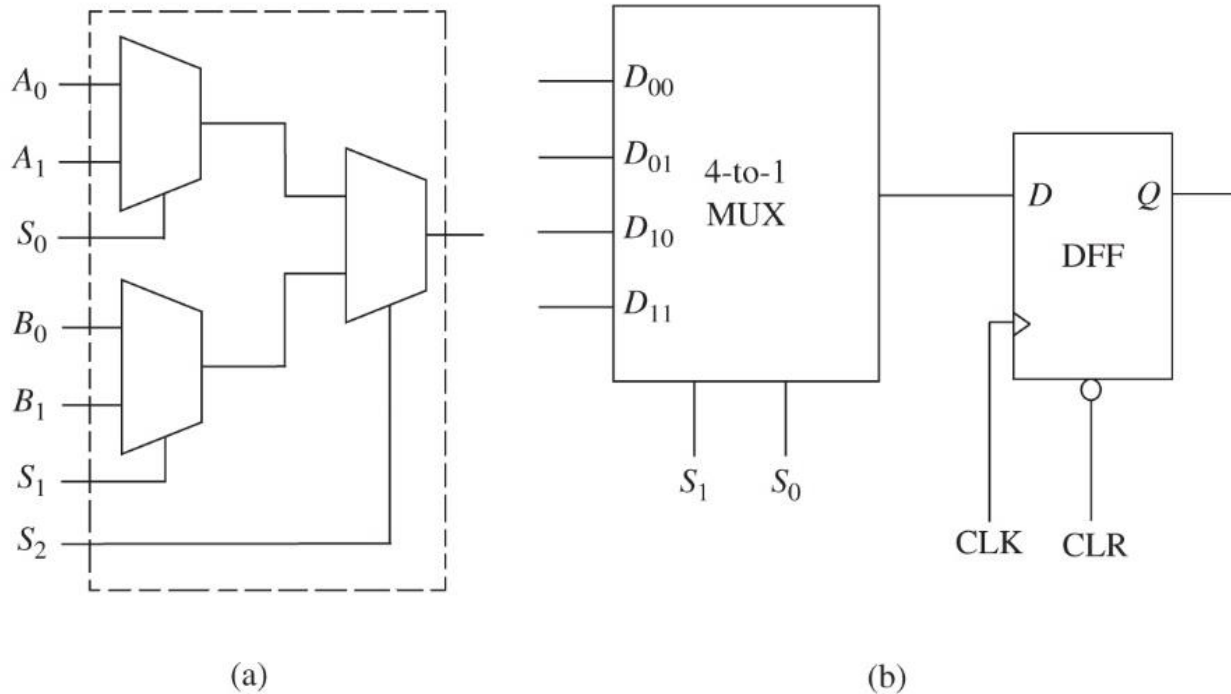
FPGAs: Programmable Logic Block Architectures

- Logic Blocks Based on Multiplexers and Gates:
 - Any combinational function can be implemented using multiplexers alone.
 - Microsemi is a manufacturer of multiplexer-based FPGAs.



FPGAs: Programmable Logic Block Architectures

- Logic Blocks Based on Multiplexers and Gates:



FPGAs: Programmable Interconnects

- There are different types of interconnection resources in all commercial FPGAs. Every vendor has its own specific names for the different types of interconnects in their FPGA.
- Main types:
 - Interconnects in Symmetric Array
 - Row-based

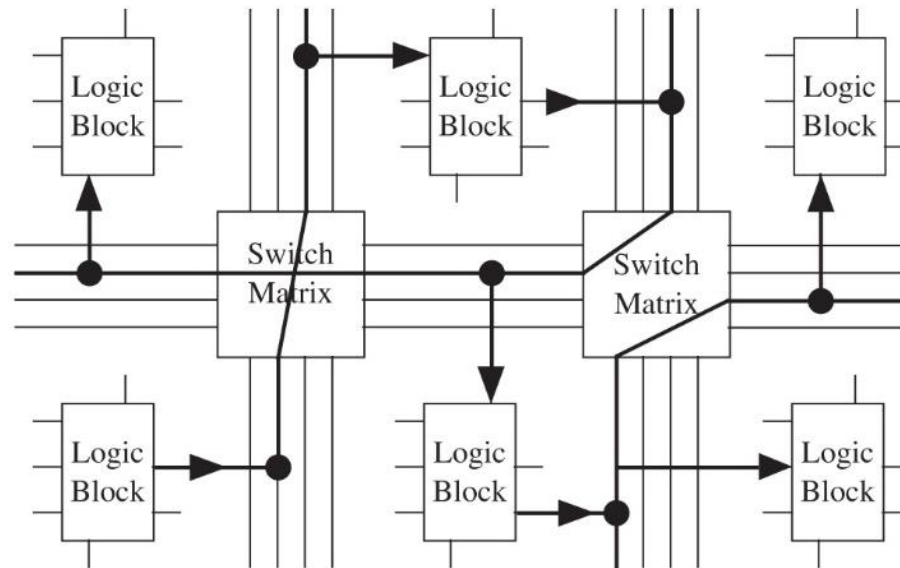
FPGAs: Programmable Interconnects

- Interconnects in Symmetric Array FPGAs:
 - General-purpose
 - Direct interconnects
 - Global lines

FPGAs: Programmable Interconnects

General-purpose:

- Use switch matrices that provide interconnections between routing wires connected to the switch matrix.



FPGAs: Programmable Interconnects

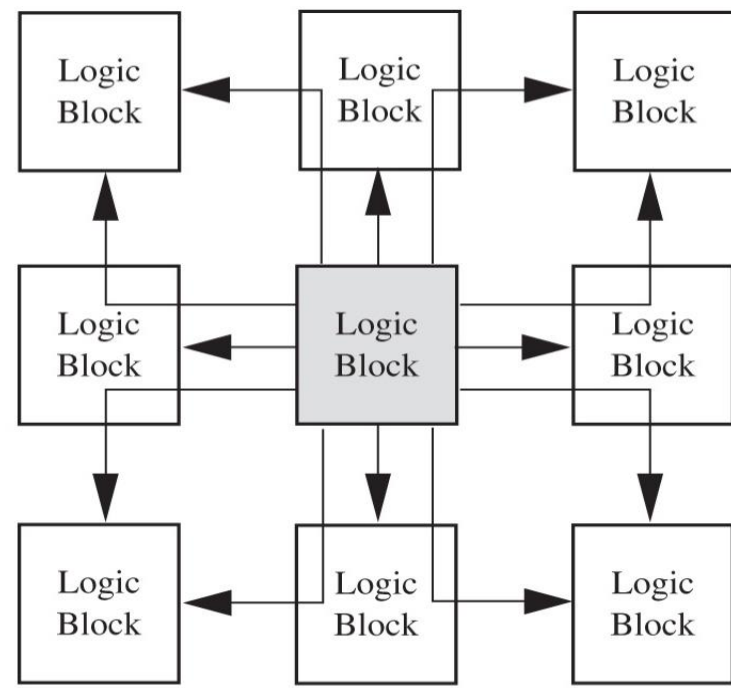
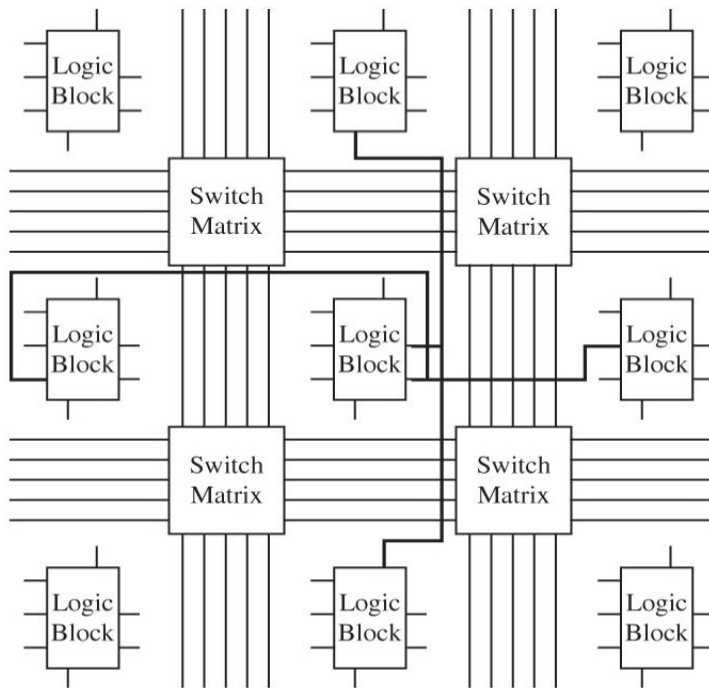
Direct interconnects:

- Special connections between adjacent logic blocks. Interconnects are fast because they do not go through the routing matrix.
- Direct interconnections are to the four nearest neighbors or, in some cases, to eight neighboring blocks.

FPGAs: Programmable Interconnects

Direct interconnects:

- Nearest 4 and 8 neighboring blocks

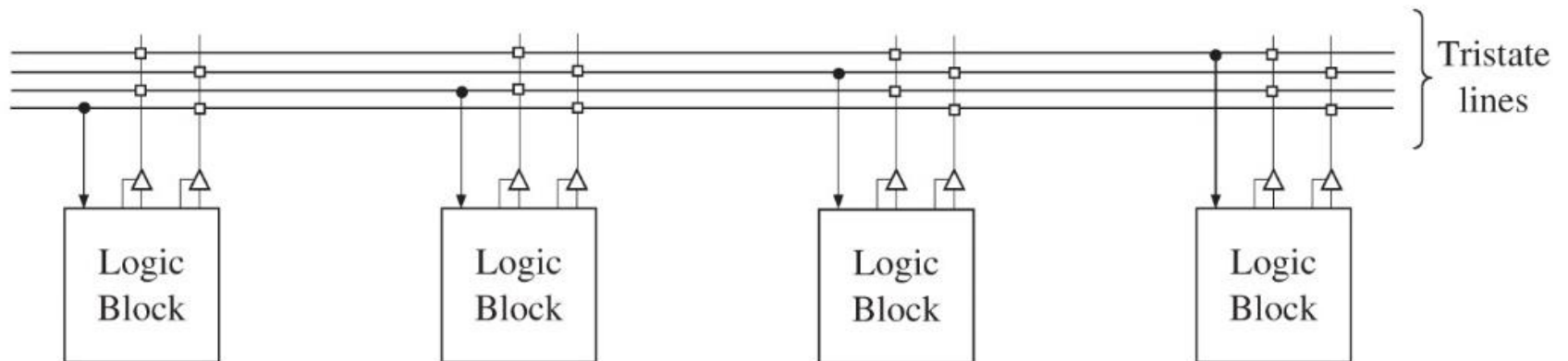


FPGAs: Programmable Interconnects

- Global lines:
 - Routing lines span the entire width/height of device.
 - A limited number (two or four) of such global lines are provided by many FPGAs in horizontal and vertical directions.

FPGAs: Programmable Interconnects

- Global lines:

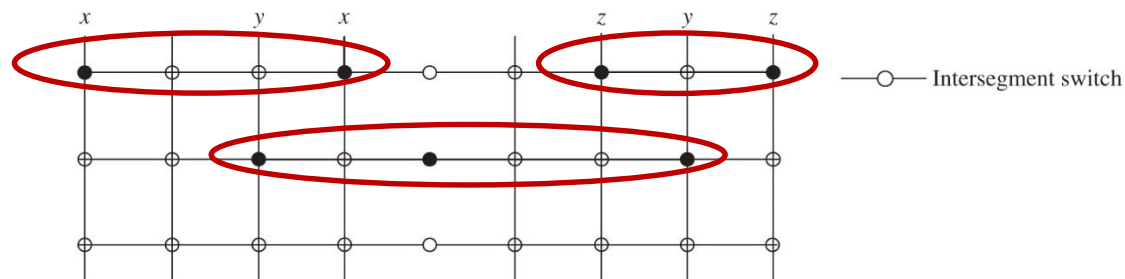
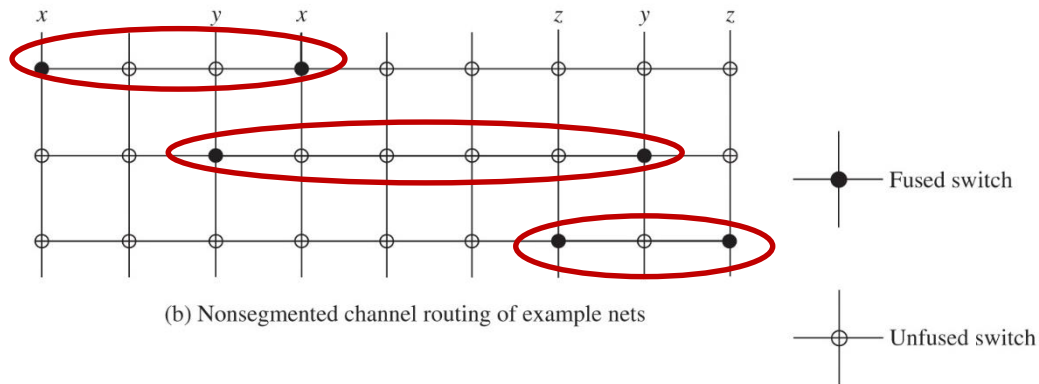


FPGAs: Programmable Interconnects

- Row-Based:
 - Has rows of logic blocks and channels of switches to enable connections between the logic blocks.
 - Routing resources in these are similar to routing in traditional gate arrays.
 - Interconnects are either non-segmented or segmented.
 - Non-segmented has a high area overhead compared to segmented.

FPGAs: Programmable Interconnects

- Row-Based:
 - Non-segmented vs. segmented



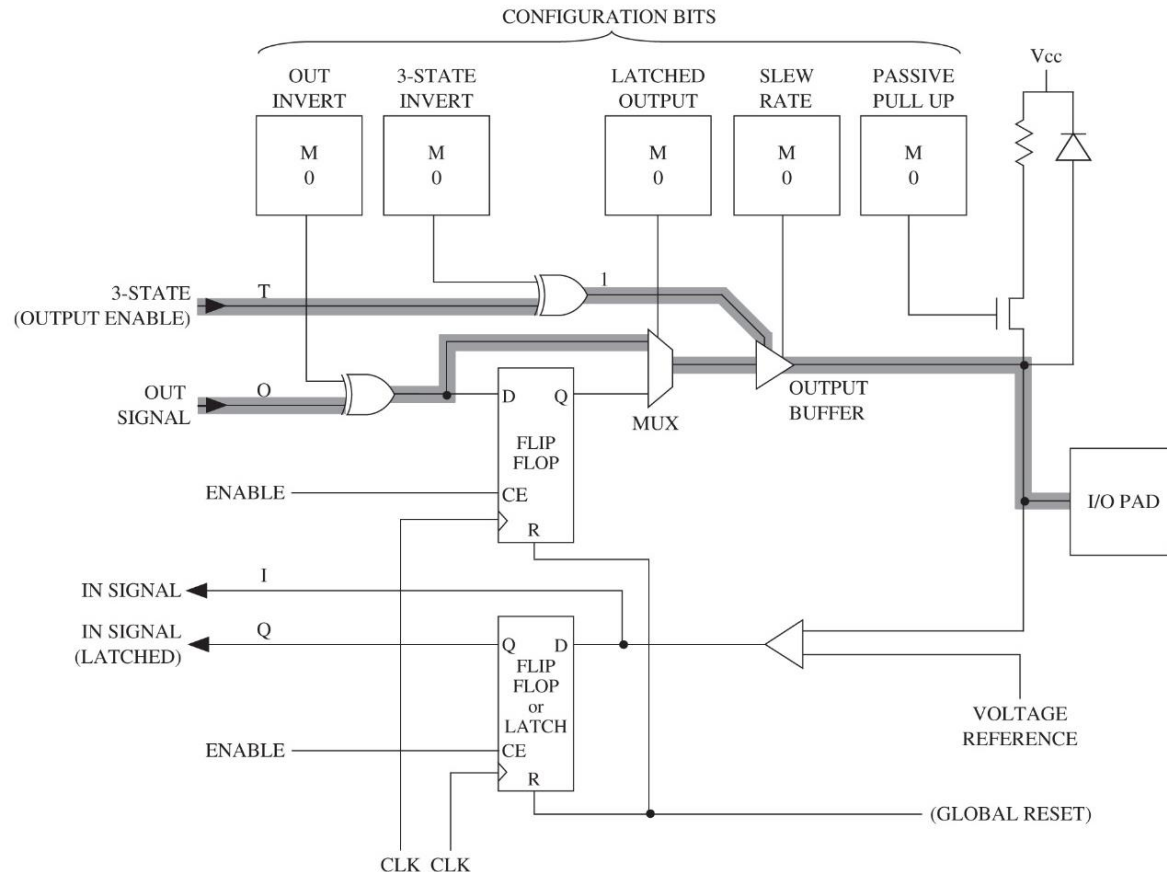
(c) Segmented channel routing of example nets

FPGAs: Programmable I/O Blocks

- I/O blocks on modern FPGAs allow use of the pin as input and/or output, in direct (combinational) or latched forms, in tristate true or inverted forms, and with a variety of I/O standards.

FPGAs: Programmable I/O Blocks

- Programmable I/O block (based on Xilinx):



FPGAs: Programmable I/O Blocks

- I/O Standards:

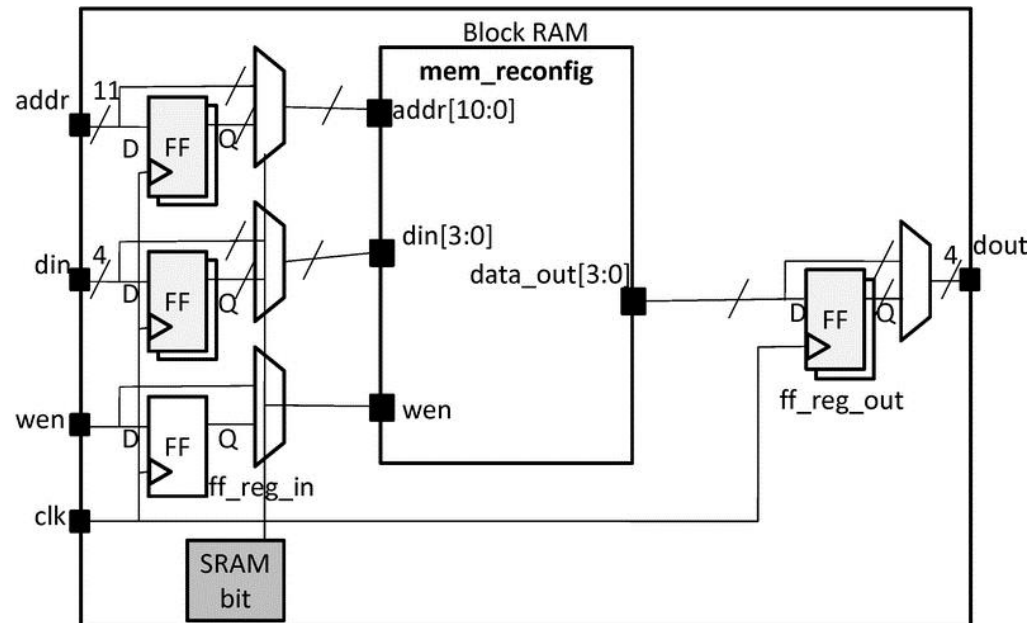
- LVTTTL: low-voltage transistor-transistor logic; 3.3-V standard that can tolerate 5-V signals.
- PCI: peripheral component interconnect; has 5-V and 3.3-V versions.
- LVCMOS: low-voltage complementary metal-oxide semiconductor; LVCMOS2, a 2.5-V standard that can tolerate 5-V signals.
- LVPECL: low-voltage positive emitter-coupled logic
- SSTL: stub-series terminated logic
- AGP: advanced graphics port
- CTT: center tap terminated
- GTL: gunning transceiver logic
- HSTL: high-speed transceiver logic

Dedicated Logic on FPGA

Gates, Registers and Routing	XC2000
Three-State Bus	XC3000
Dedicated Arithmetic	XC4000
Memory	XC4000
RAM Blocks	FLEX
Dynamic Reconfiguration	CAL/XC6200
Universal I/O	Virtex
LVDS I/O	Virtex
Programmable Clock	Virtex
Microprocessor	Excalibur
Source-Synchronous Transceiver	Virtex-II
Bitstream Encryption	Virtex-II
Transceiver	Virtex-II Pro
Multiplier	Virtex-4
Ethernet MAC / PCI Express	Virtex-4
System Monitor	Virtex-6
Analog to Digital Converter	Virtex-7
Floating Point Arithmetic	Stratix-10

FPGAs: Dedicated Specialized Components

- Dedicated Memory:
 - A key feature of modern FPGAs is the embedding of dedicated memory blocks (RAM) onto the chip.
 - Modern FPGAs include 16K to 10M bits of memory. The width of the embedded RAM often can be adjusted.



FPGAs: Dedicated Specialized Components

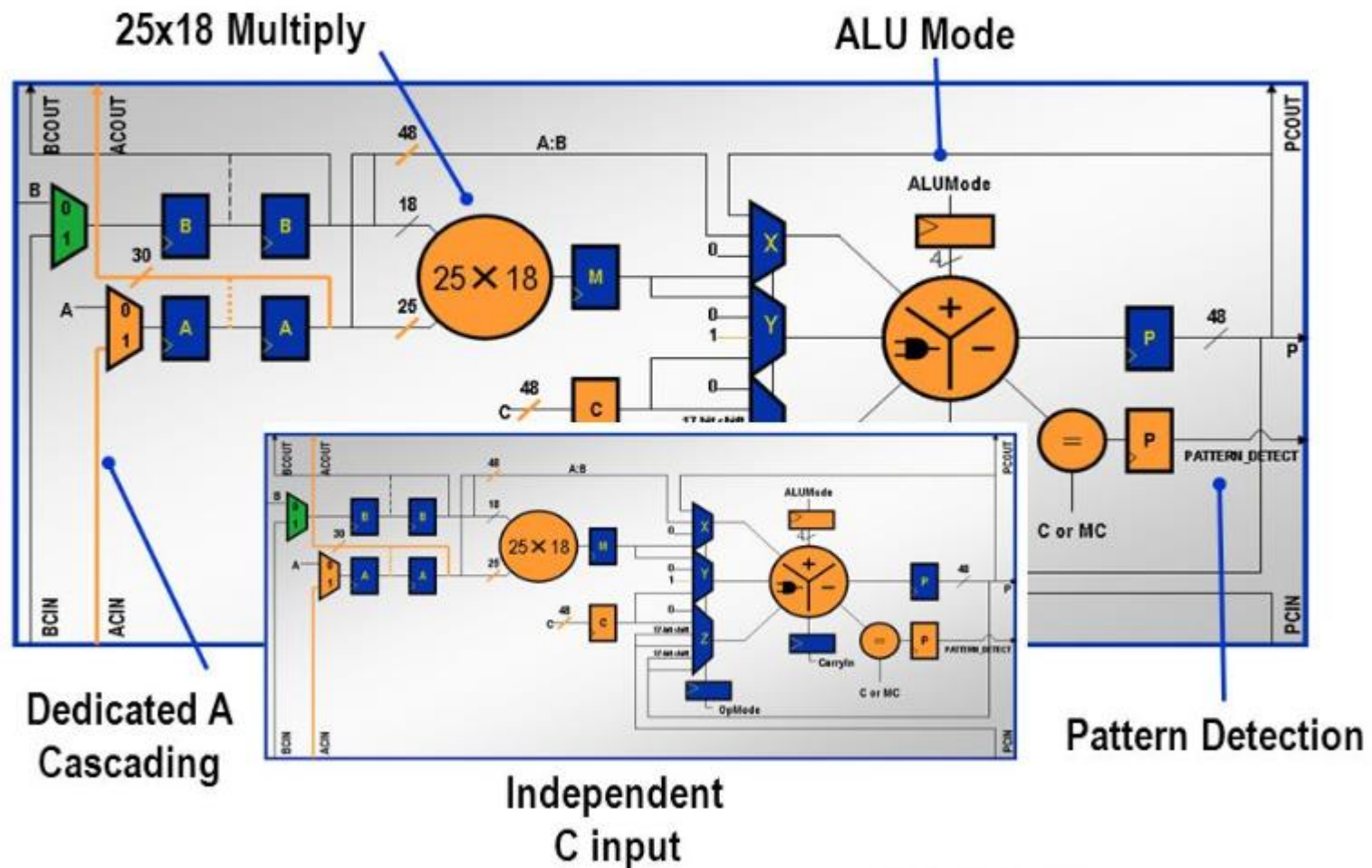
- Dedicated Arithmetic Units:
 - Used to implement arithmetic logic.
 - Implementation generally takes more area and power and is slower than custom implementations. It is therefore of benefit to provide dedicated fast-carry logic to create fast adders.
 - Many FPGAs also contain dedicated multipliers that are more efficient than those implemented using the programmable logic in the FPGA.

FPGAs: Dedicated Specialized Components

- Digital Signal Processing (DSP) Blocks:
 - Dedicated multipliers help DSP applications.
 - Example: Xilinx Virtex 5
 - Carry chains to facilitate addition.
 - Example: Altera Stratix IV
 - Also provided: DSP building blocks such as hardware for fast Fourier transforms (FFTs), finite impulse response (FIR) filters, infinite impulse response (IIR) filters, encryption/ decryption, compression/decompression, and security functions.

Xilinx DSP Block

DSP Slice



FPGAs: Dedicated Specialized Components

- Embedded Processors:
 - Many modern FPGAs contain an entire processor core which is useful when designers use hybrid solutions.
 - Some also include: the core of a small MIPS processor (i.e., MIPS R 4000), an embedded version of the IBM PowerPC processor, or custom processors (i.e., MicroBlaze from Xilinx).

FPGAs: Dedicated Specialized Components

- Content Addressable Memories (CAM):
 - Memory blocks can be used as content addressable memories (CAMs).
 - A CAM is a special kind of memory in which the content, not the address, is used to search the memory.
 - CAMs contain more logic than RAMs.

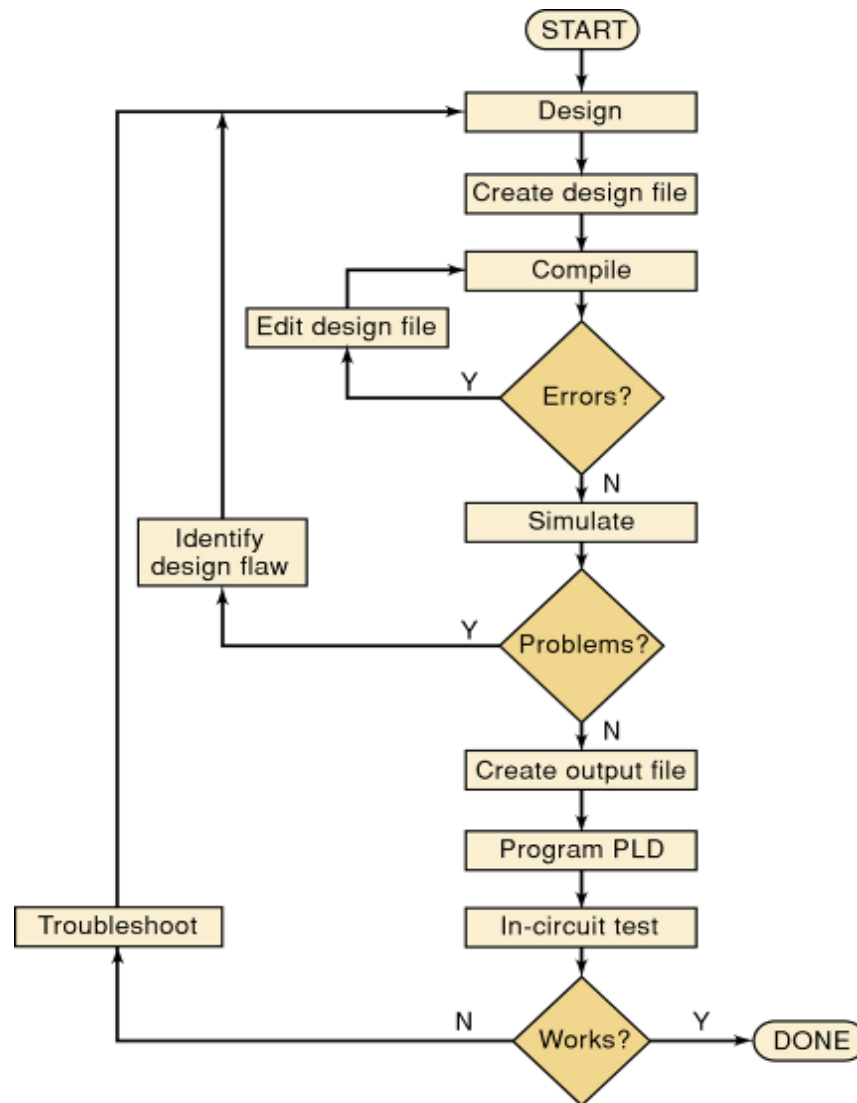
FPGAs: Design Flow

- Automatic synthesis tools are available that will take a VHDL description of the system as an input and generate an interconnection of gates and flip-flops to realize the system.
- Behavioral models can be translated into design implementations reasonably efficiently.

FPGAs: Design Flow

- Design steps for a digital system:
 - 1. Create a behavioral, RTL, or structural model of the design in a hardware description language such as Verilog or VHDL.
 - 2. Simulate and debug the design.
 - 3. Synthesize the design targeting the desired device.
 - 4. Run a mapping/partitioning program.
 - 5. Run an automatic place and route program.
 - 6. Run a program that will generate the bit pattern necessary to program the FPGA.
 - 7. Download the bit pattern into the internal configuration cells in the FPGA, and test the operation of the FPGA.

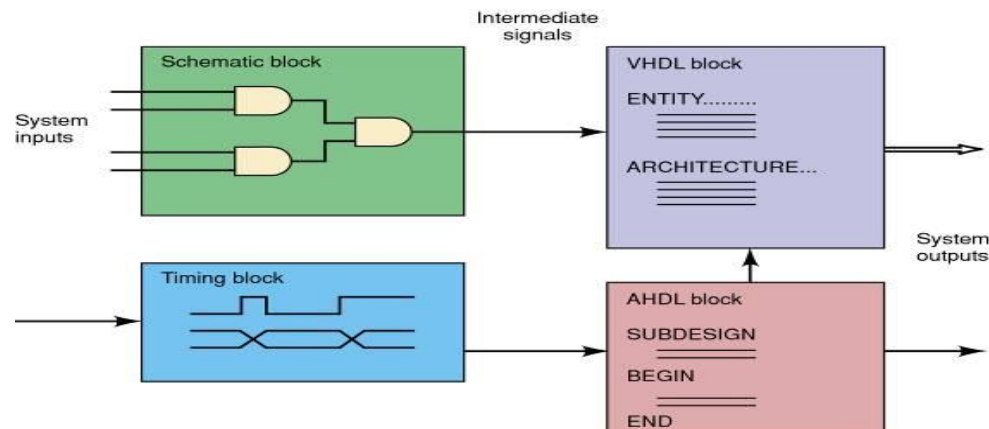
FPGAs: Design Flow



FPGAs: Design Flow

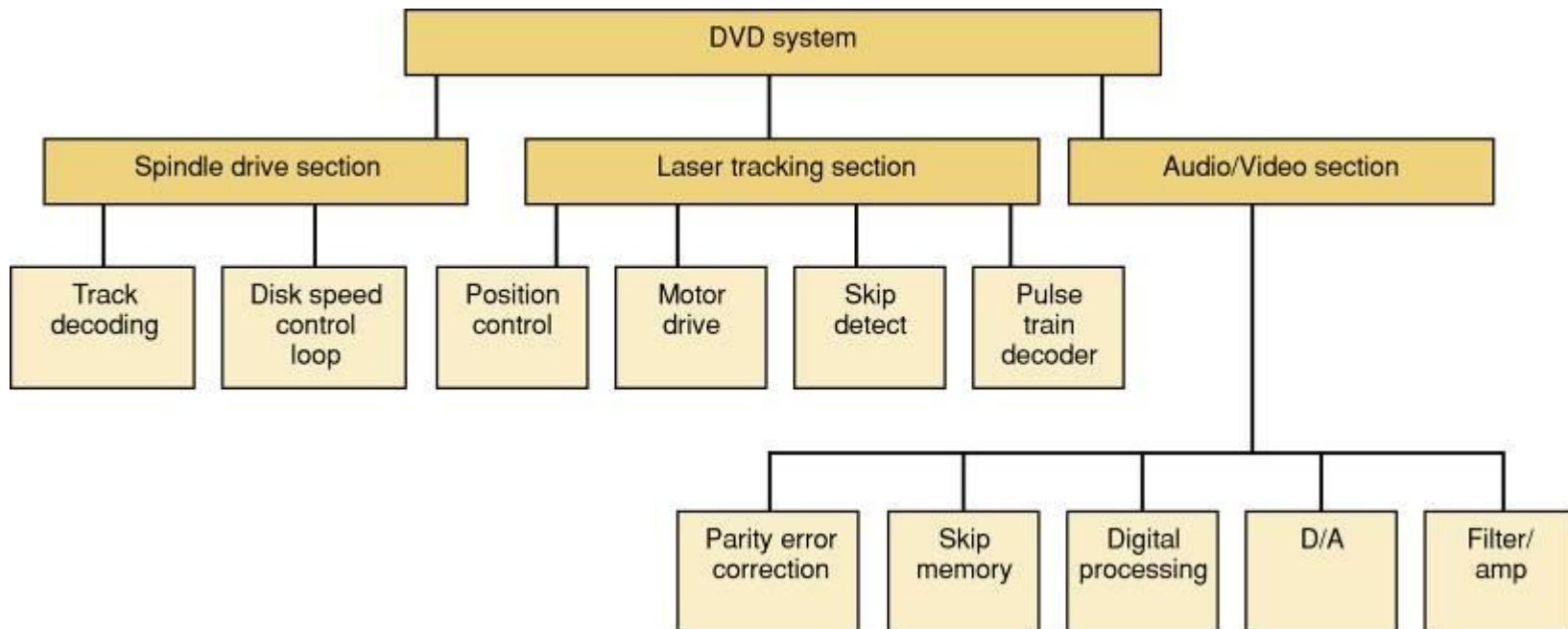
- Hierarchical design—small logic circuits are defined and combined with other circuits to form a large section of a project.
 - Large sections can be combined and connected to form a system.

Combining blocks developed using different description methods.



FPGAs: Design Flow

- Top-down design—requires the definition of subsections that will make up the system.
 - And definition of the individual circuits that will make up each sub section.
- Each level can be designed and tested individually.

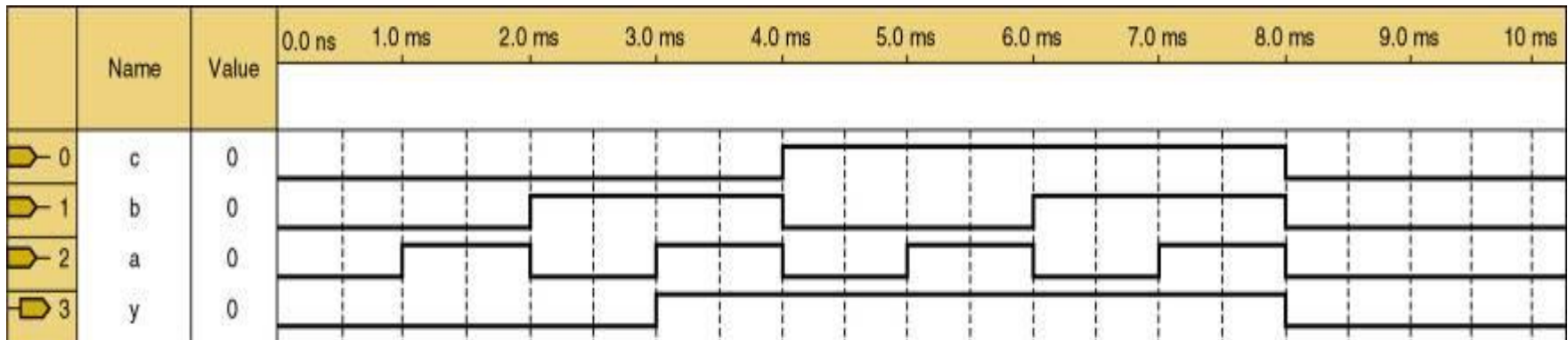


FPGAs: Design Flow

- A system is built from bottom up.
- Each block is described by a design file.
- After testing it is compiled using development software.
- The compiled block is tested using a simulator for verify correct operation.
- A PLD is programmed to verify correct operation.

FPGAs: Design Flow

- Timing simulation of a circuit described in HDL.

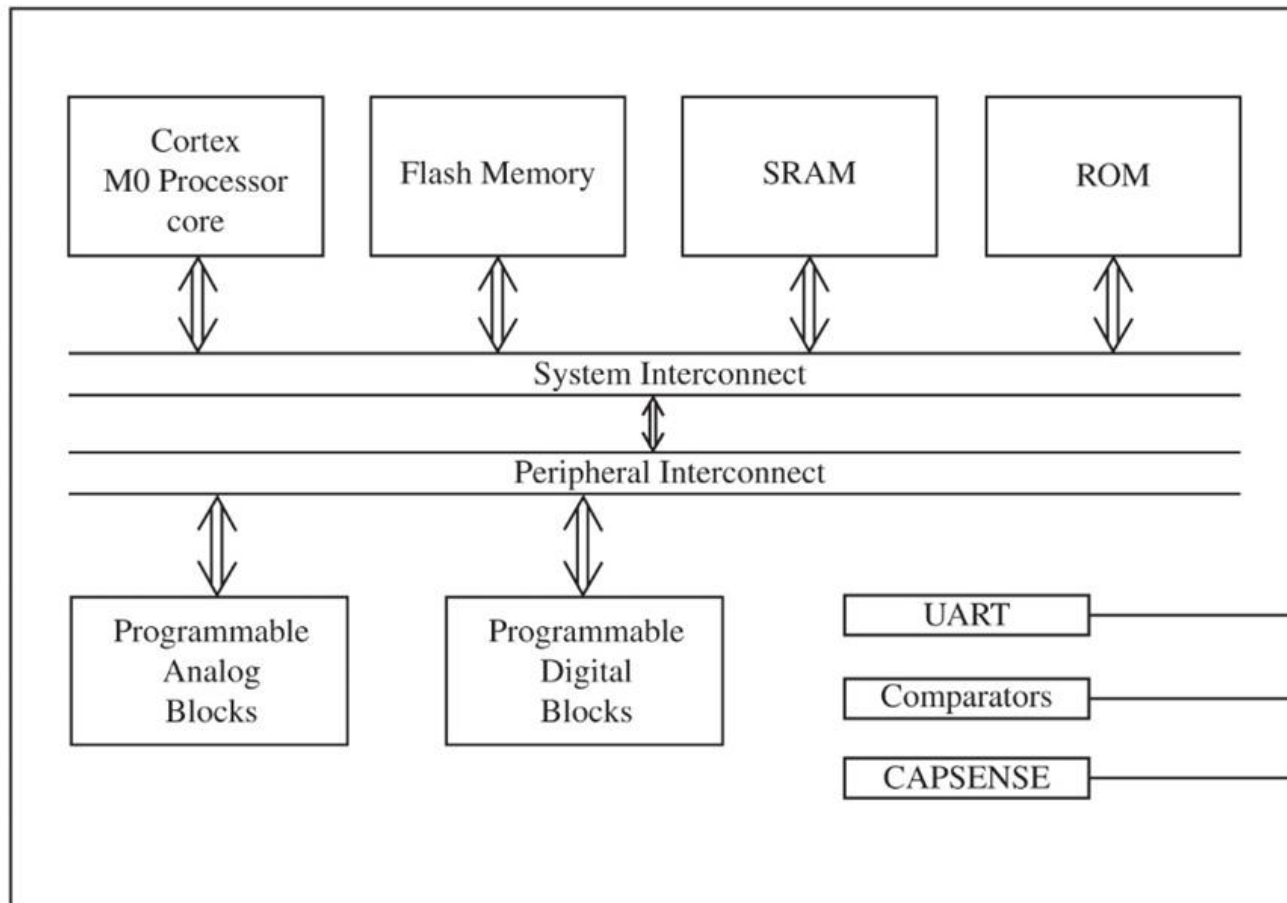


Programmable SoCs (PSOC)

- A **system on a chip (SoC)** is an IC that integrates all parts of a system into one chip.
- Intellectual property (IP) cores can be bought and electronically integrated.
- A PSOC is a field-programmable SoC chip.

Programmable SoCs (PSOC)

- Overview of a programmable SoC:

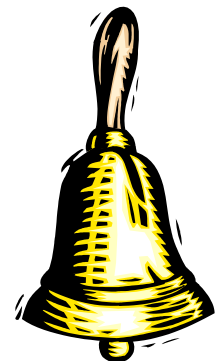


Zynq® UltraScale+™ MPSoCs: CG Devices

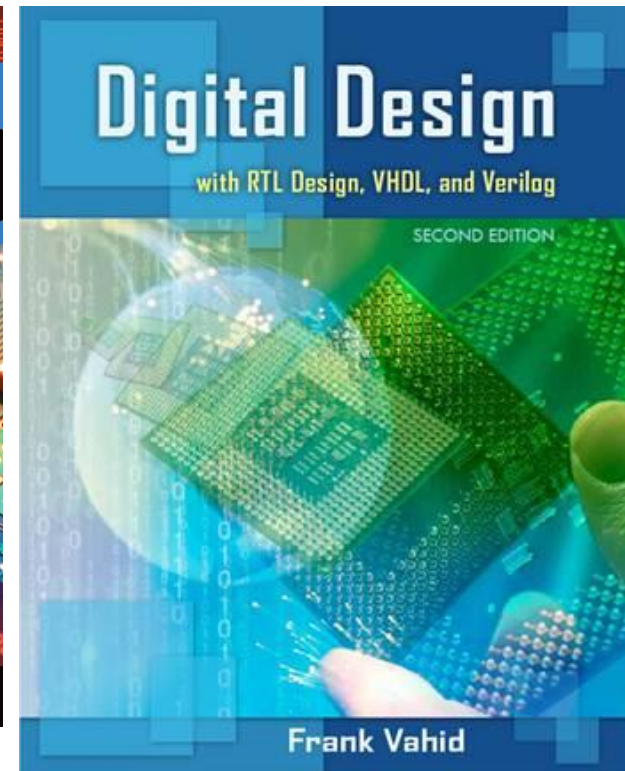
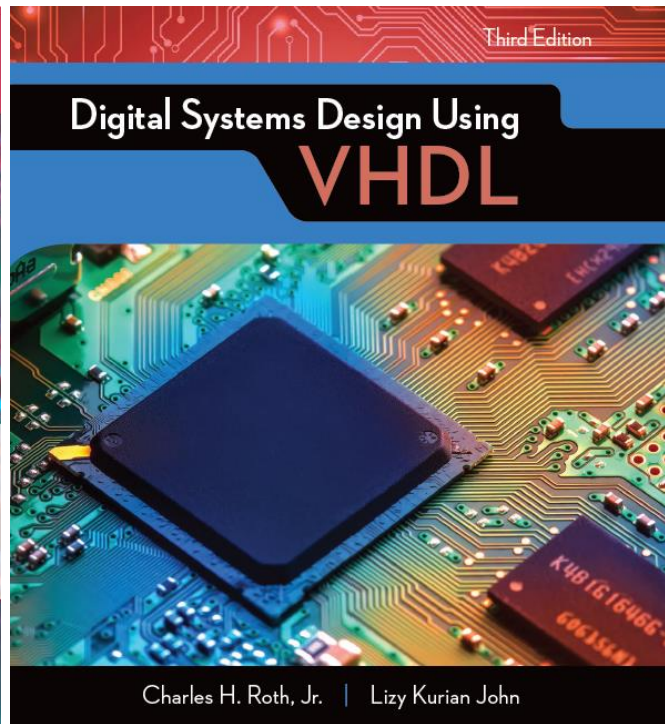
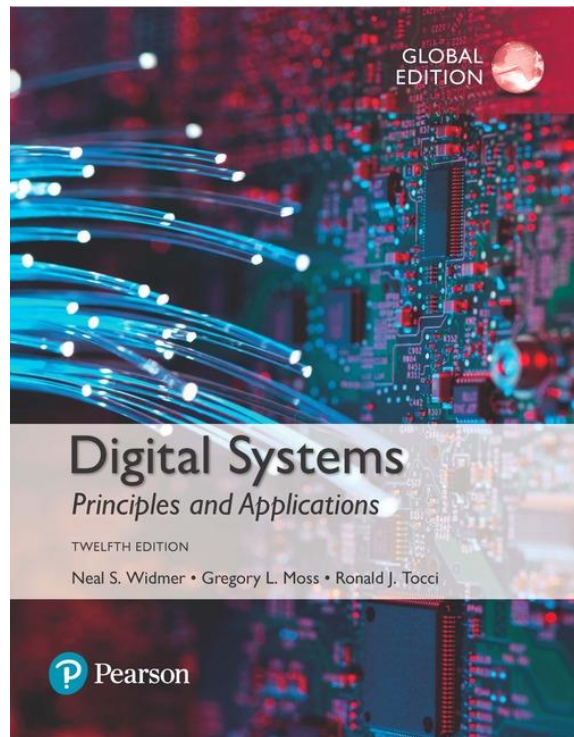
		Device Name ⁽¹⁾	ZU2CG	ZU3CG	ZU4CG	ZU5CG	ZU6CG	ZU7CG	ZU9CG
Processing System (PS)	Application	Processor Core	Dual-core ARM® Cortex™-A53 MPCore™ up to 1.3GHz						
	Processor Unit	Memory w/ECC	L1 Cache 32KB I / D per core, L2 Cache 1MB, on-chip Memory 256KB						
	Real-Time	Processor Core	Dual-core ARM Cortex-R5 MPCore up to 533MHz						
	Processor Unit	Memory w/ECC	L1 Cache 32KB I / D per core, Tightly Coupled Memory 128KB per core						
	External Memory	Dynamic Memory Interface	x32/x64: DDR4, LPDDR4, DDR3, DDR3L, LPDDR3 with ECC						
		Static Memory Interfaces	NAND, 2x Quad-SPI						
	Connectivity	High-Speed Connectivity	PCIe® Gen2 x4, 2x USB3.0, SATA 3.1, DisplayPort, 4x Tri-mode Gigabit Ethernet						
		General Connectivity	2xUSB 2.0, 2x SD/SDIO, 2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO						
	Integrated Block Functionality	Power Management	Full / Low / PL / Battery Power Domains						
		Security	RSA, AES, and SHA						
AMS - System Monitor		10-bit, 1MSPS – Temperature and Voltage Monitor							
PS to PL Interface		12 x 32/64/128b AXI Ports							
Programmable Logic (PL)	Programmable Functionality	System Logic Cells (K)	103	154	192	256	469	504	600
		CLB Flip-Flops (K)	94	141	176	234	429	461	548
		CLB LUTs (K)	47	71	88	117	215	230	274
	Memory	Max. Distributed RAM (Mb)	1.2	1.8	2.6	3.5	6.9	6.2	8.8
		Total Block RAM (Mb)	5.3	7.6	4.5	5.1	25.1	11.0	32.1
		UltraRAM (Mb)	-	-	13.5	18.0	-	27.0	-
	Clocking	Clock Management Tiles (CMTs)	3	3	4	4	4	8	4
	Integrated IP	DSP Slices	240	360	728	1,248	1,973	1,728	2,520
		PCI Express® Gen 3x16 / Gen4x8	-	-	2	2	-	2	-
		150G Interlaken	-	-	-	-	-	-	-
		100G Ethernet MAC/PCS w/RS-FEC	-	-	-	-	-	-	-
		AMS - System Monitor	1	1	1	1	1	1	1
	Transceivers	GTH 16.3Gb/s Transceivers	-	-	16	16	24	24	24
GTY 32.75Gb/s Transceivers		-	-	-	-	-	-	-	
Speed Grades	Extended ⁽²⁾	-1 -2 -2L							
	Industrial	-1 -1L -2							

Summary

- Technology underlying early programmable logic devices, such as ROMs, PALs, and PLAs was presented.
- Simple PLDs and GALs were presented.
- CPLDs and FPGAs were presented.
- FPGA topics discussed include an overview of the general technology, general organization of FPGAs, general structure of logic blocks, and typical programming techniques.
- PSOC designs were presented.



Reference Books



Resources

- <https://www.xilinx.com/support/university.html>
- <https://www.xilinx.com/support/university/vivado/vivado-workshops.html>
- You can now install a FREE Xilinx Webpack
- <https://www.xilinx.com/products/design-tools/vivado/vivado-webkit.html>
- <https://www.xilinx.com/products/design-tools/ise-design-suite/ise-webkit.html>