Chapter 1
VLSI Design Methods

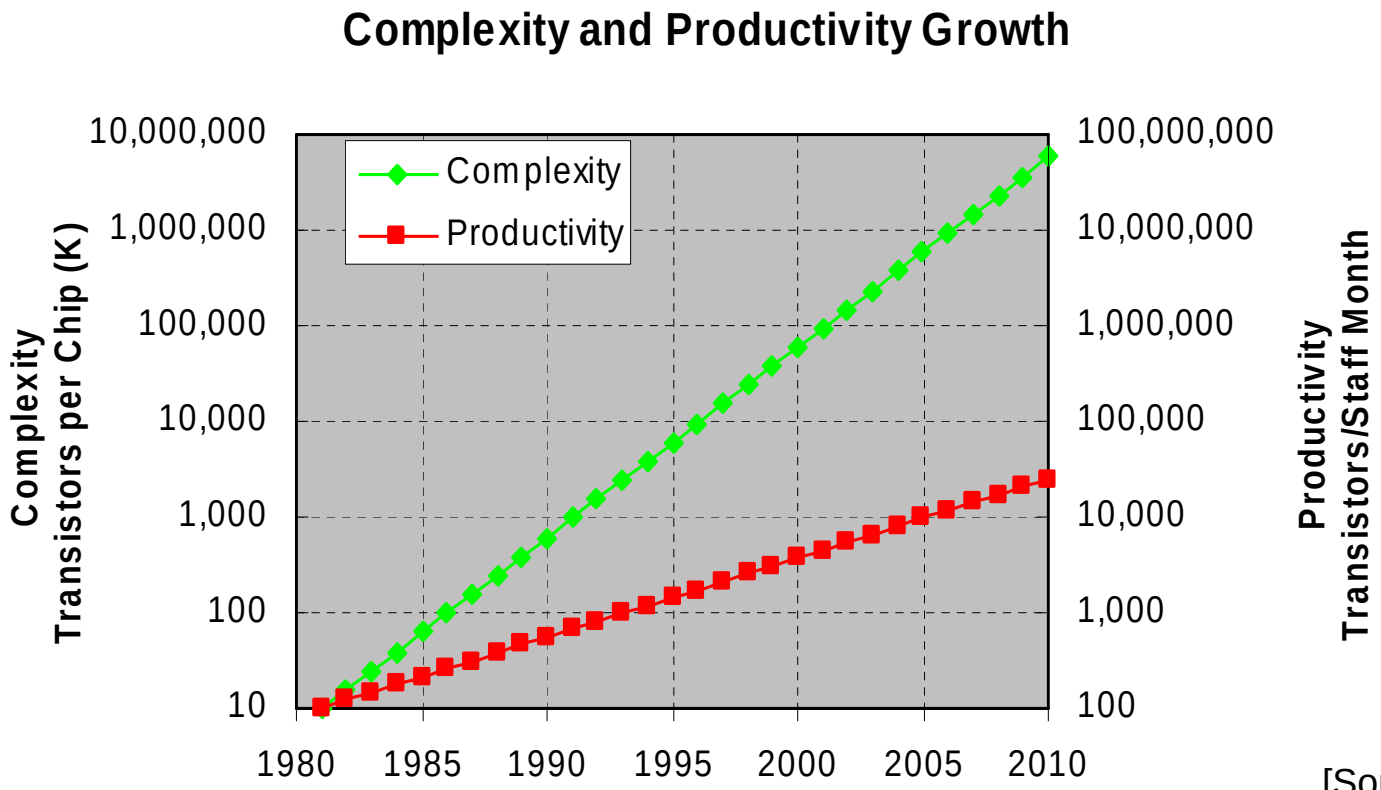
Jin-Fu Li
Advanced Reliable Systems (ARES) Laboratory
Department of Electrical Engineering
National Central University
Jhongli, Taiwan

Outline

- Introduction
- VLSI Design Flows & Design Verification
- VLSI Design Styles
- System-on-Chip Design Methodology

Complexity & Productivity Growth of ICs

- Complexity grows 58%/yr (doubles every 18 mos)
- Productivity grows 21%/yr (doubles every $3^1/_2$ yrs) unless methodology is updated

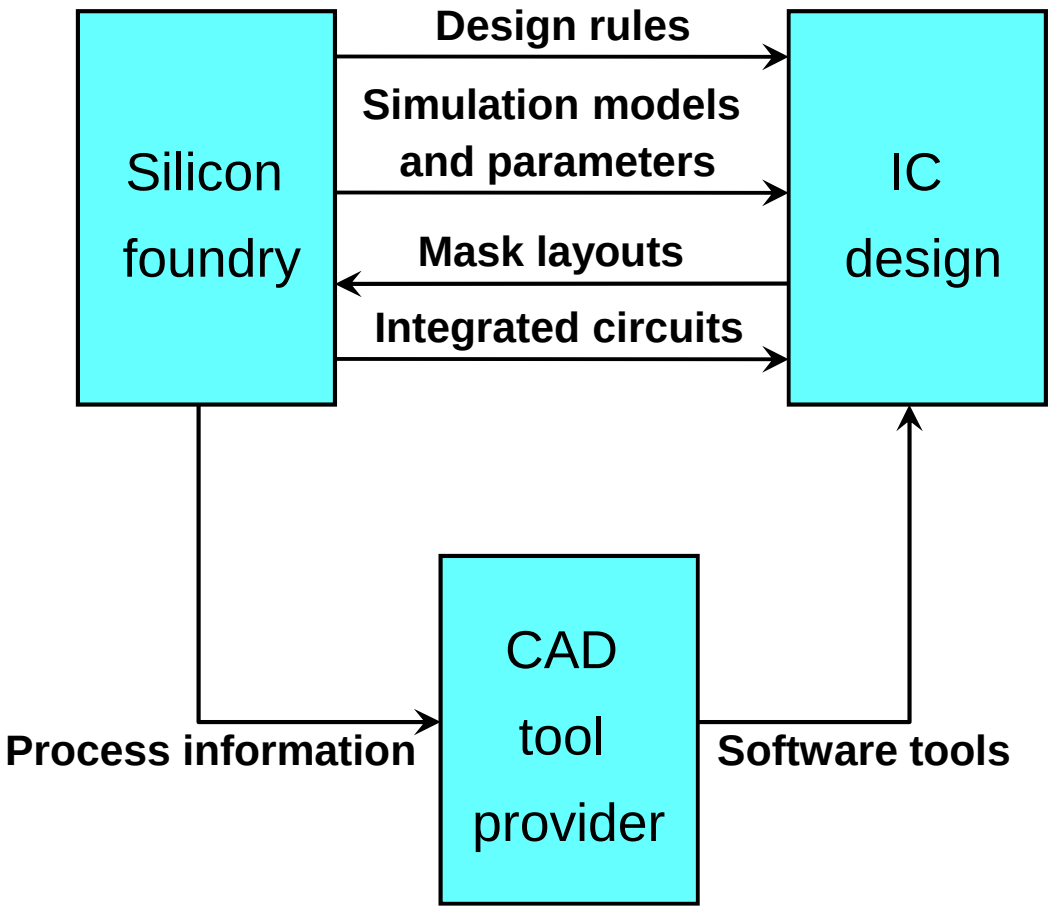


[Source: MITRE]

VLSI Design Methodologies

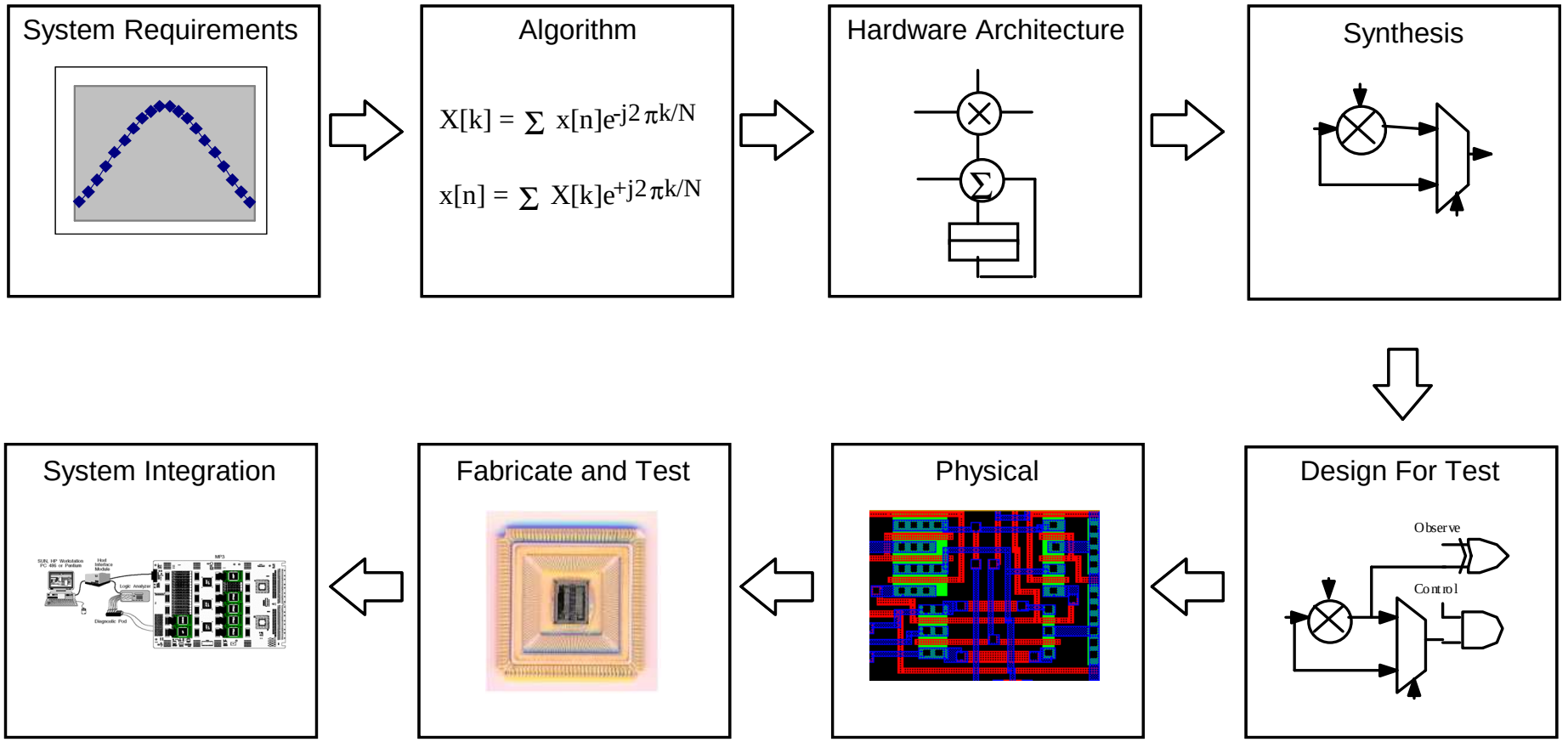
- Design methodology
  - Process for creating a design
- Methodology goals
  - Design cycle
  - Complexity
  - Performance
  - Reuse
  - Reliability

IC Community



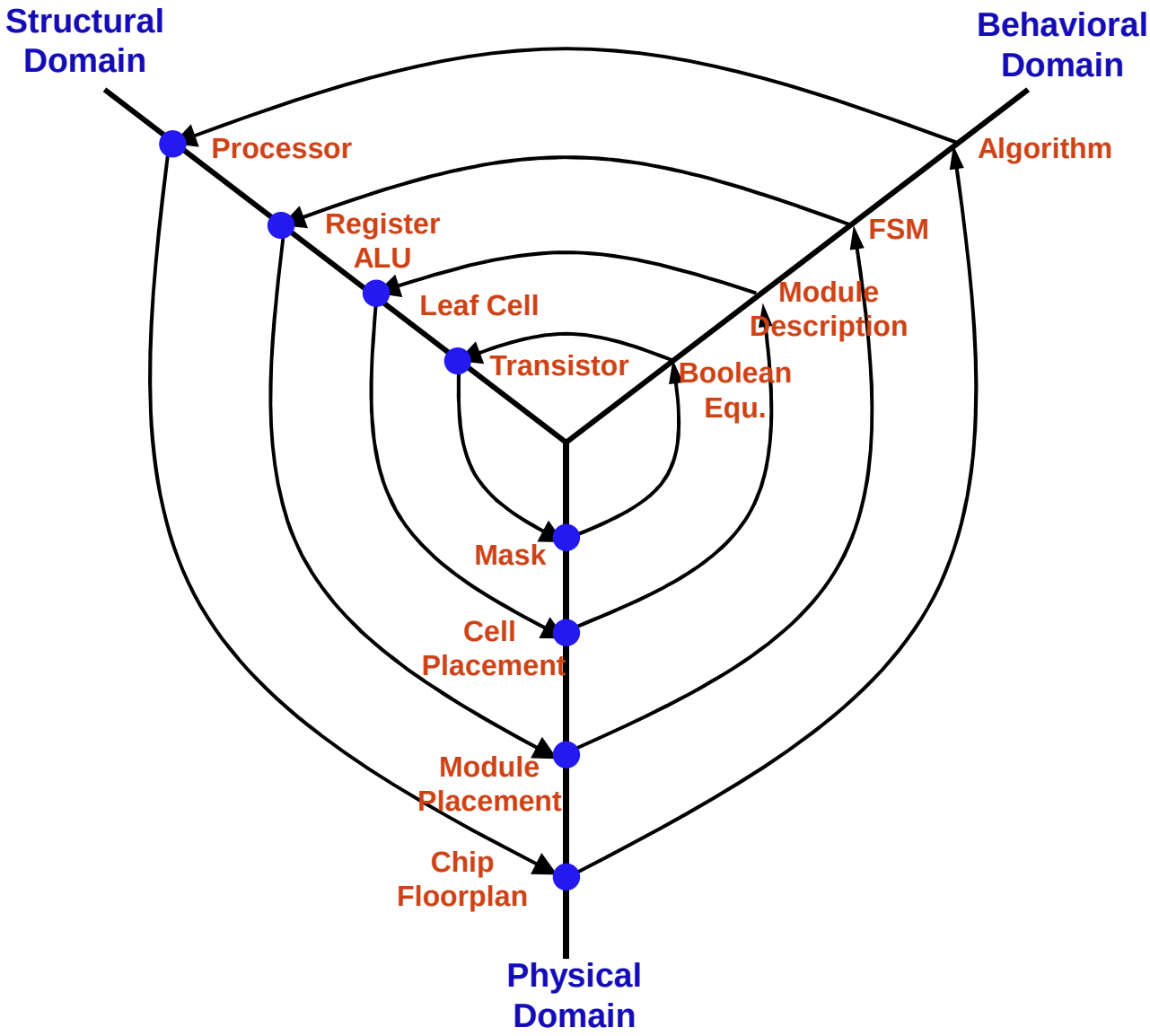
[M. M. Vai, VLSI design]

System to Silicon Design

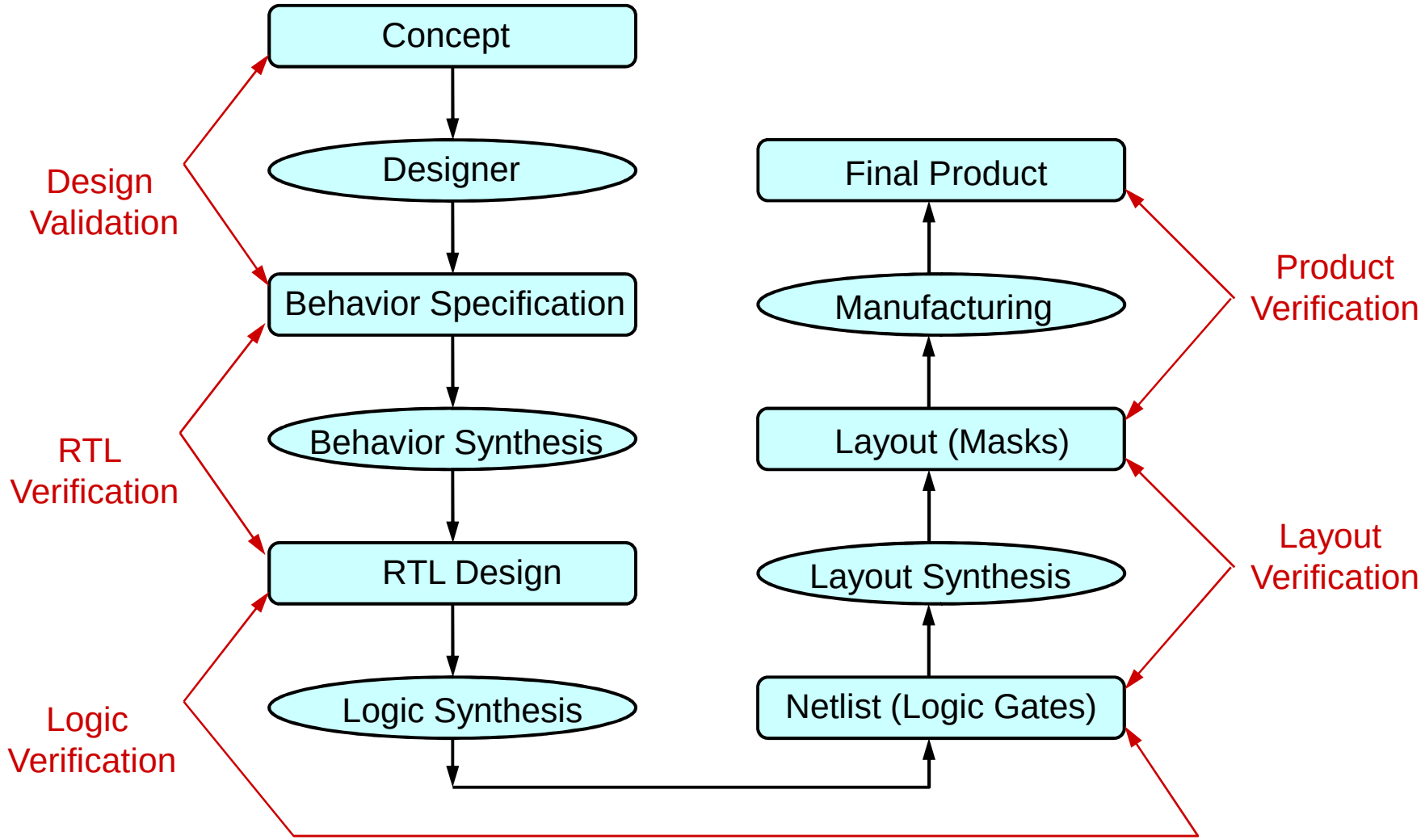


[Source: MITRE]

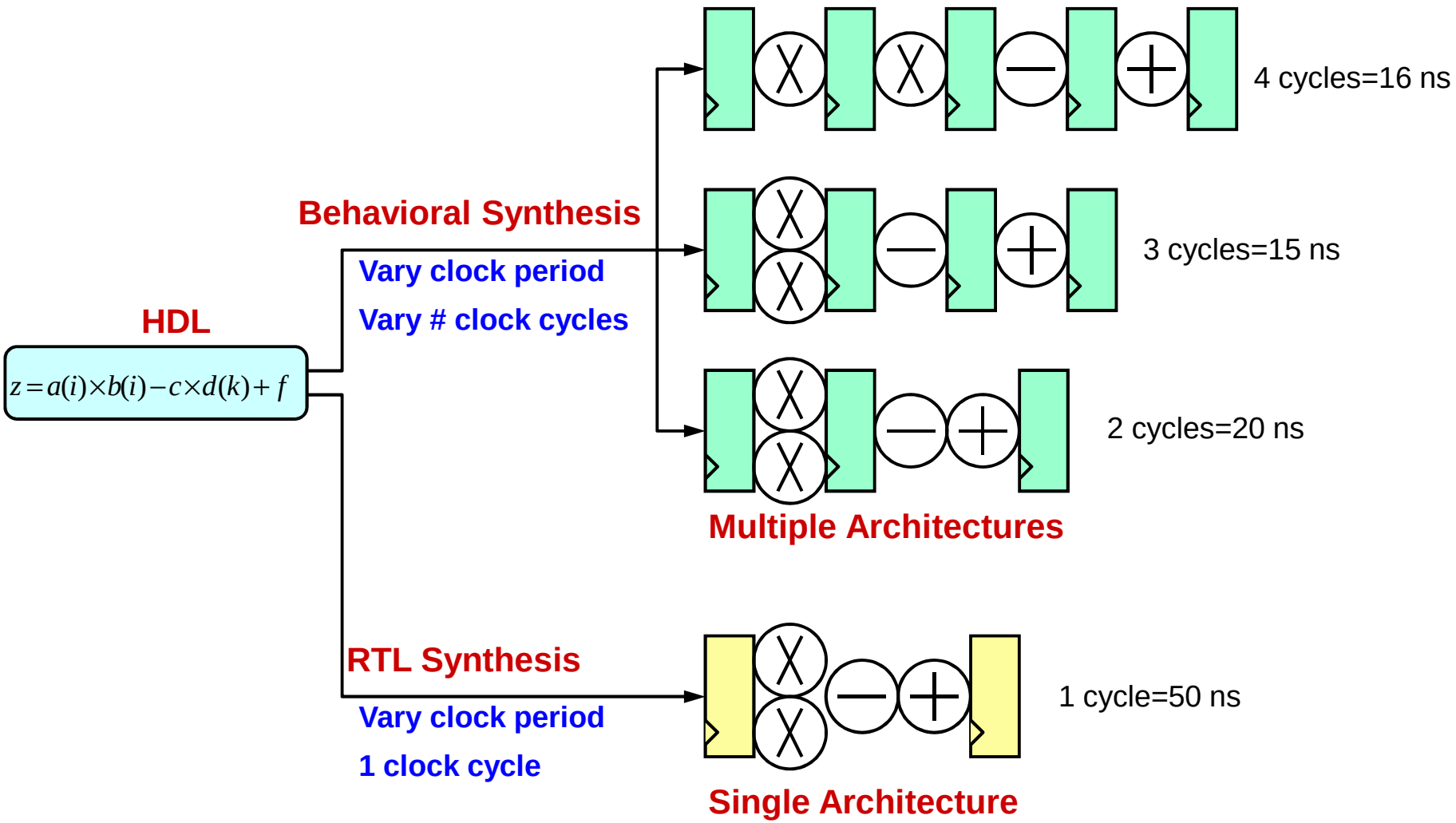
Y-Chart


Structural Domain
Behavioral Domain
Processor
Algorithm
Register ALU
FSM
Leaf Cell
Module Description
Transistor
Boolean Equ.
Mask
Cell Placement
Module Placement
Chip Floorplan
Physical Domain

VLSI Design Flow



Behavioral Synthesis & RTL Synthesis



Source: Synopsys

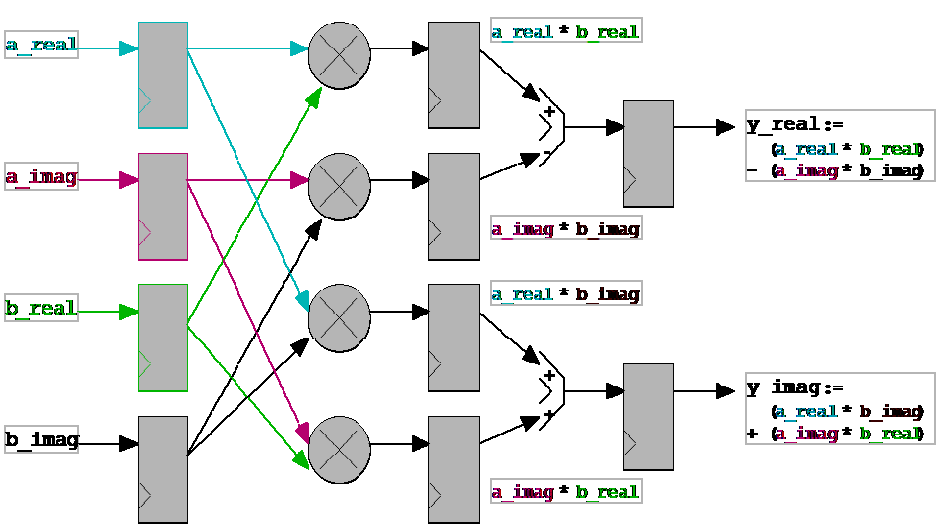
Behavioral Synthesis (Resource Allocation)

- Source code defines the functionality

  y_real := a_real * b_real - a_imag * b_imag

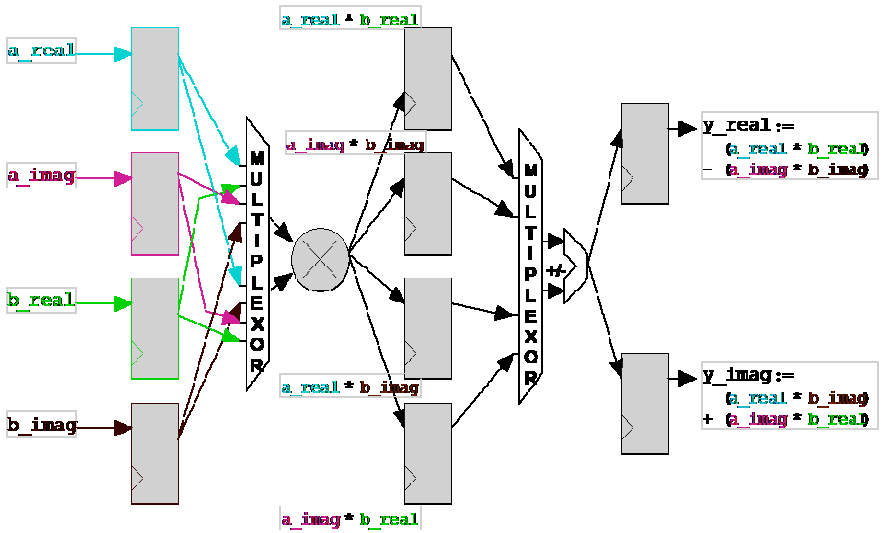
  y_imag := a_real * b_imag + a_imag * b_real

- Constraints allow designer to explore different architectures, trading off speed vs. area
- Two possible implementations for a complex multiplier:



Fast, but large

(4 mult, 1 add, 1 sub)



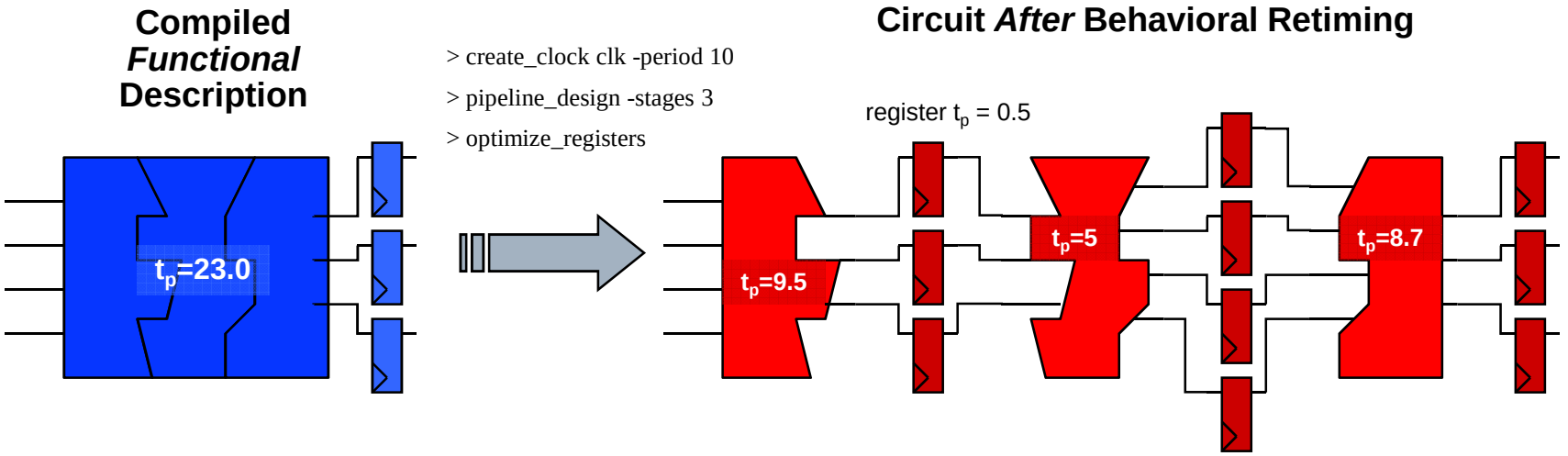
Small, but slow

(1 mult, 1 add/sub)

[Source: MITRE]

Behavioral Synthesis (Retiming)

- Allows designer to trade off latency for throughput by adding and moving registers in order to meet timing constraints
- Specification needs to include registers at *functional* boundaries, without regard to register-to-register timing: software takes care of optimizing register placement



Max. Speed= $^1/_{23.0\ nSec}$ = 43 MHz

Latency = 1 clock cycle

Max. Speed= $^1/_{10\ nSec}$ = 100 MHz

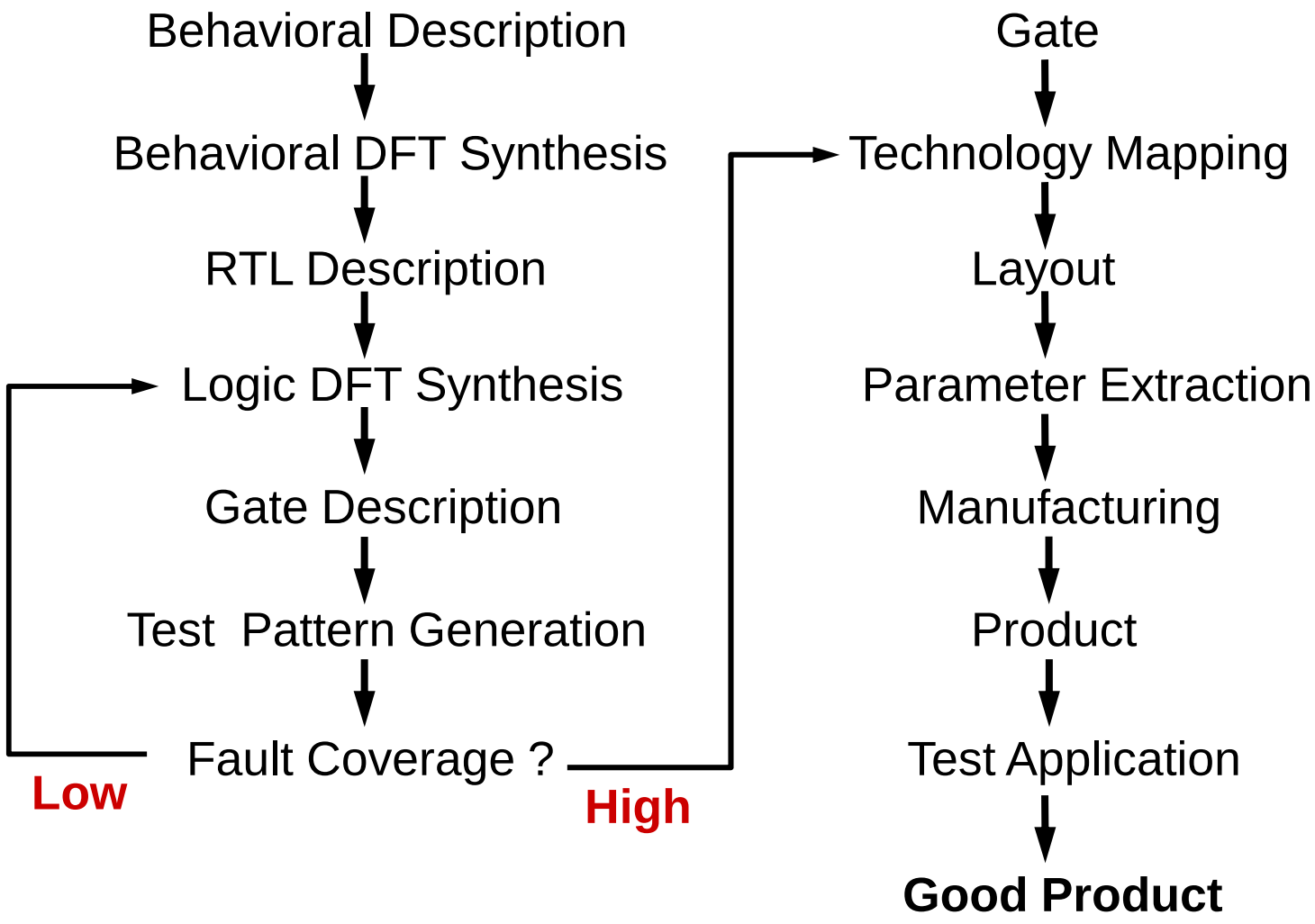
Latency = 3 clock cycles

[Source: MITRE]

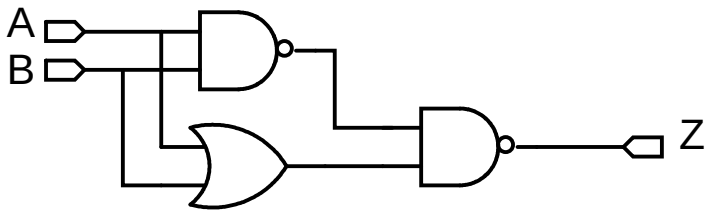
Verification

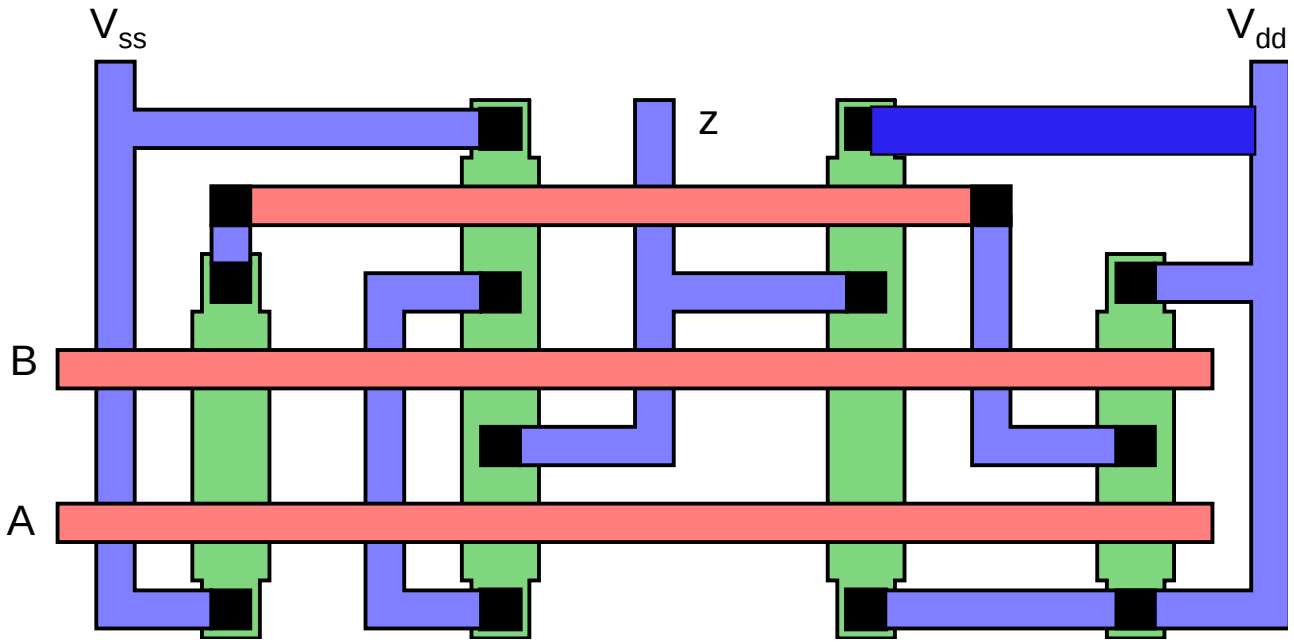
- The four representations of the design
  - Behavioral, RTL, gate level, and layout
- In mapping the design from one phase to another, it is likely that some errors are produced
  - Caused by the CAD tools or human mishandling of the tools
- Usually, *simulation* is used for verification, although more recently, *formal verification* has been gaining in importance
- Two types of simulations are used to verify the design
  - *Functional simulation* & *timing simulation*

DFT Flow

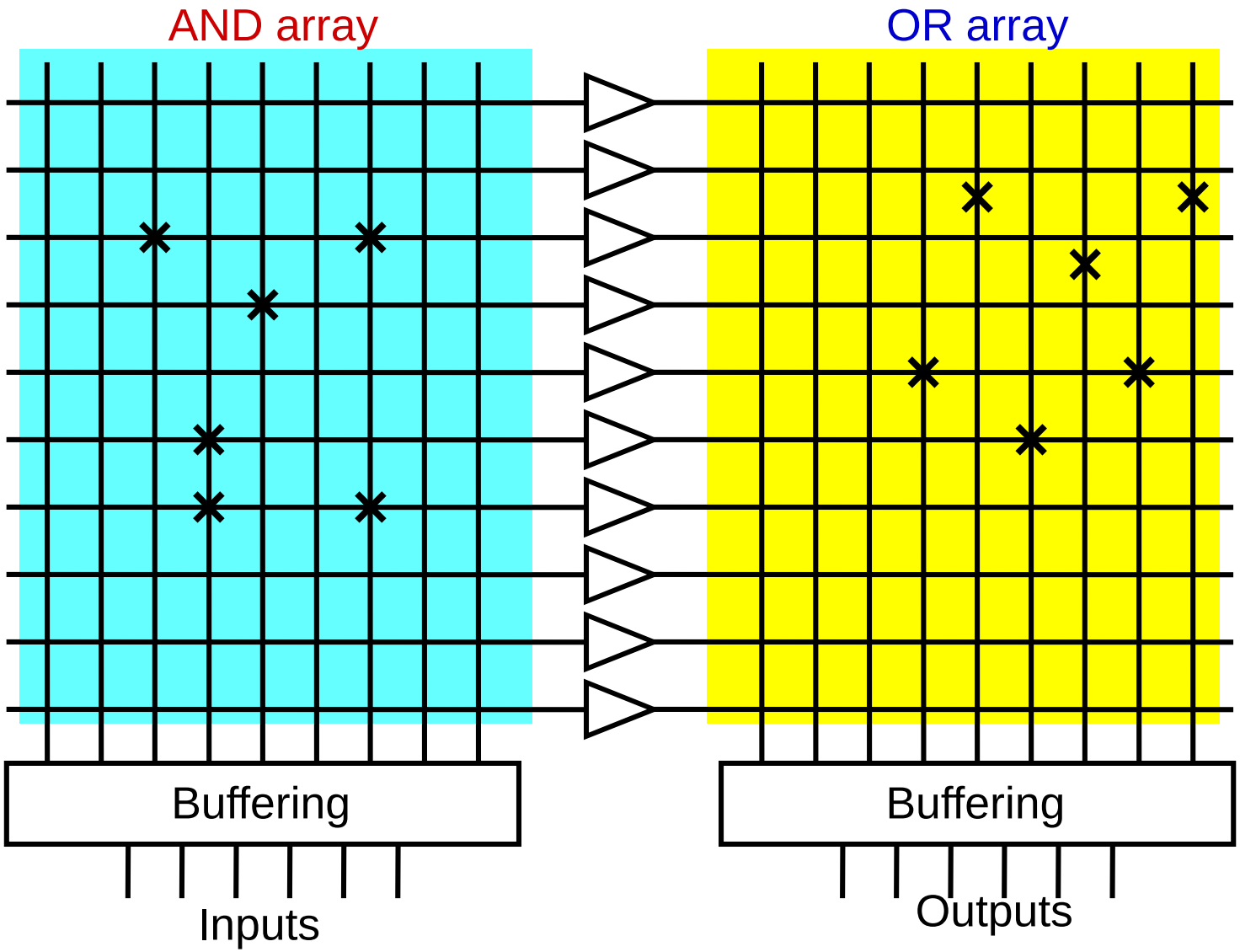

Behavioral Description
Behavioral DFT Synthesis
RTL Description
Logic DFT Synthesis
Gate Description
Test Pattern Generation
Fault Coverage ?
Low
High
Gate
Technology Mapping
Layout
Parameter Extraction
Manufacturing
Product
Test Application
Good Product

Design Styles – *Full Custom*

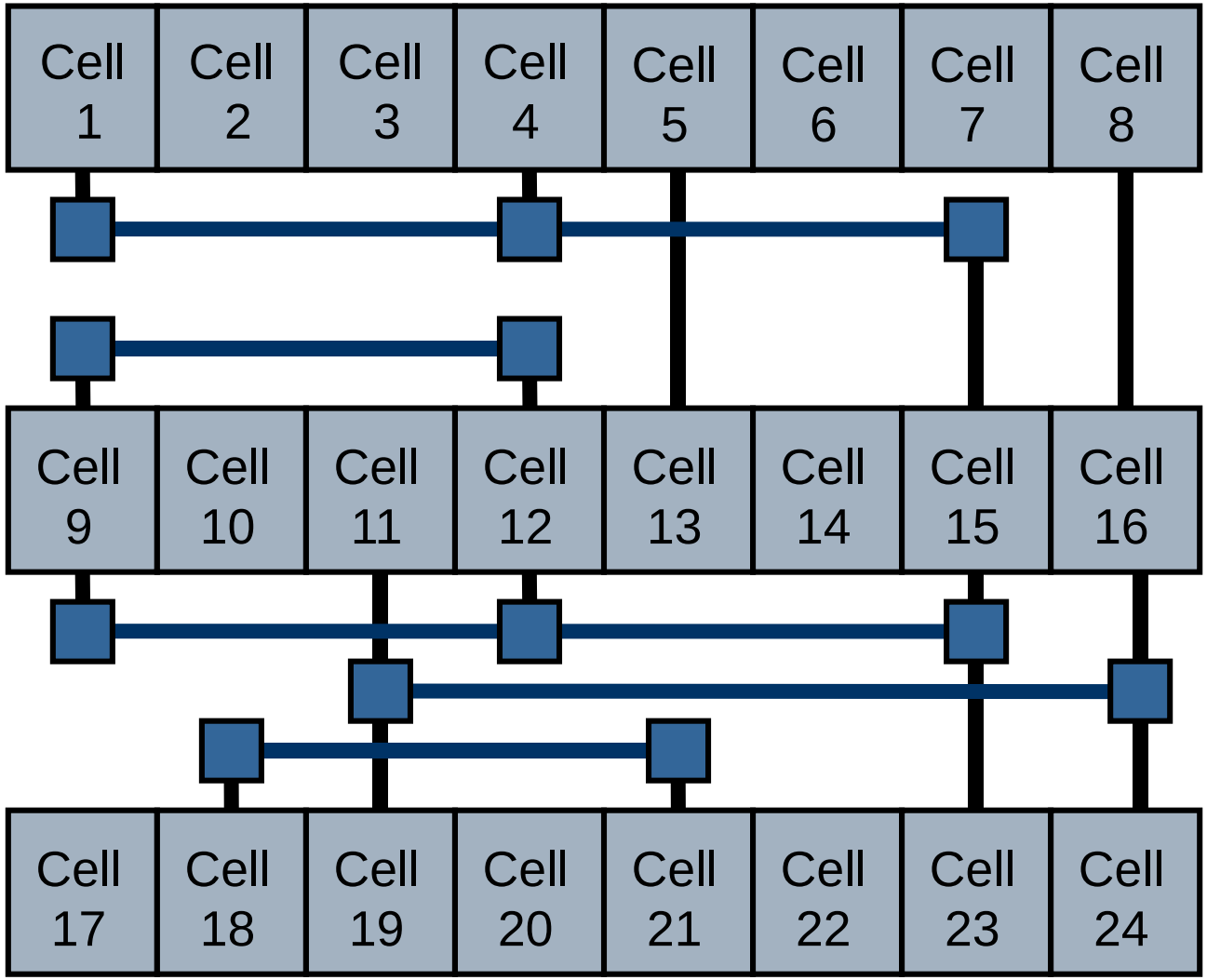




Design Styles – *Programmable Logic Array*

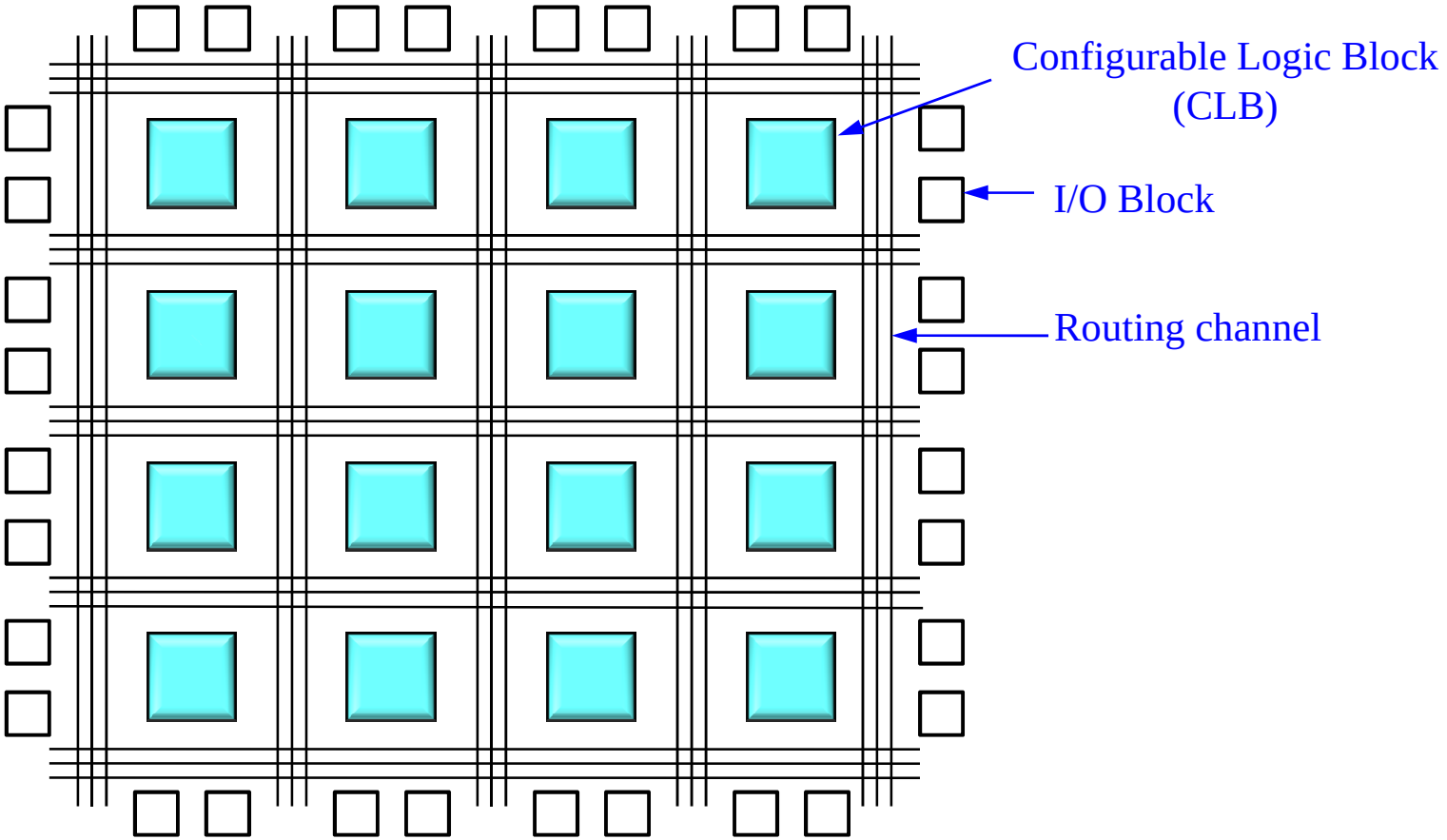


Design Styles – *Gate Array*



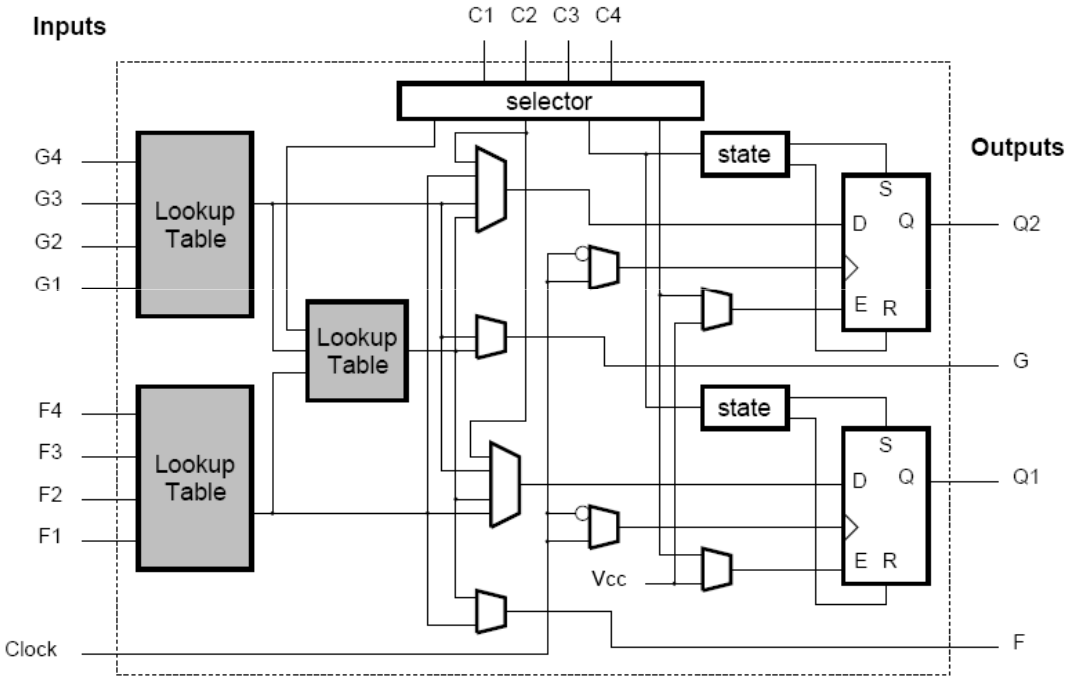
Design Styles – *Field Programmable Gate Array (FPGA)*

- Xilinx SRAM-based FPGA



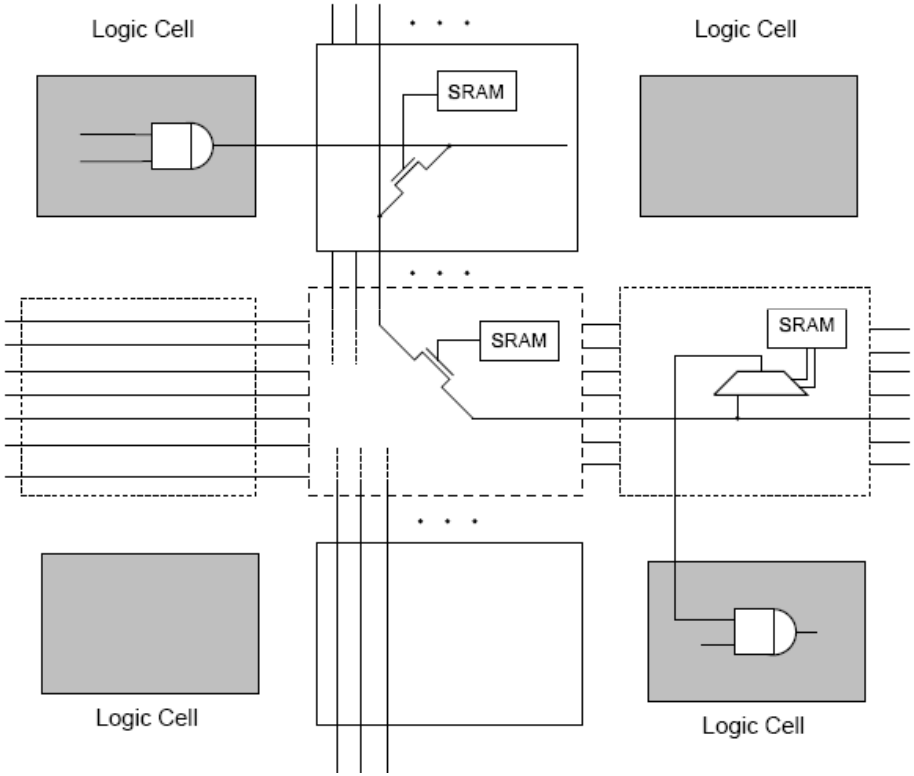
Design Styles – *Field Programmable Gate Array (FPGA)*

- Xilinx XC4000 Configurable logic block



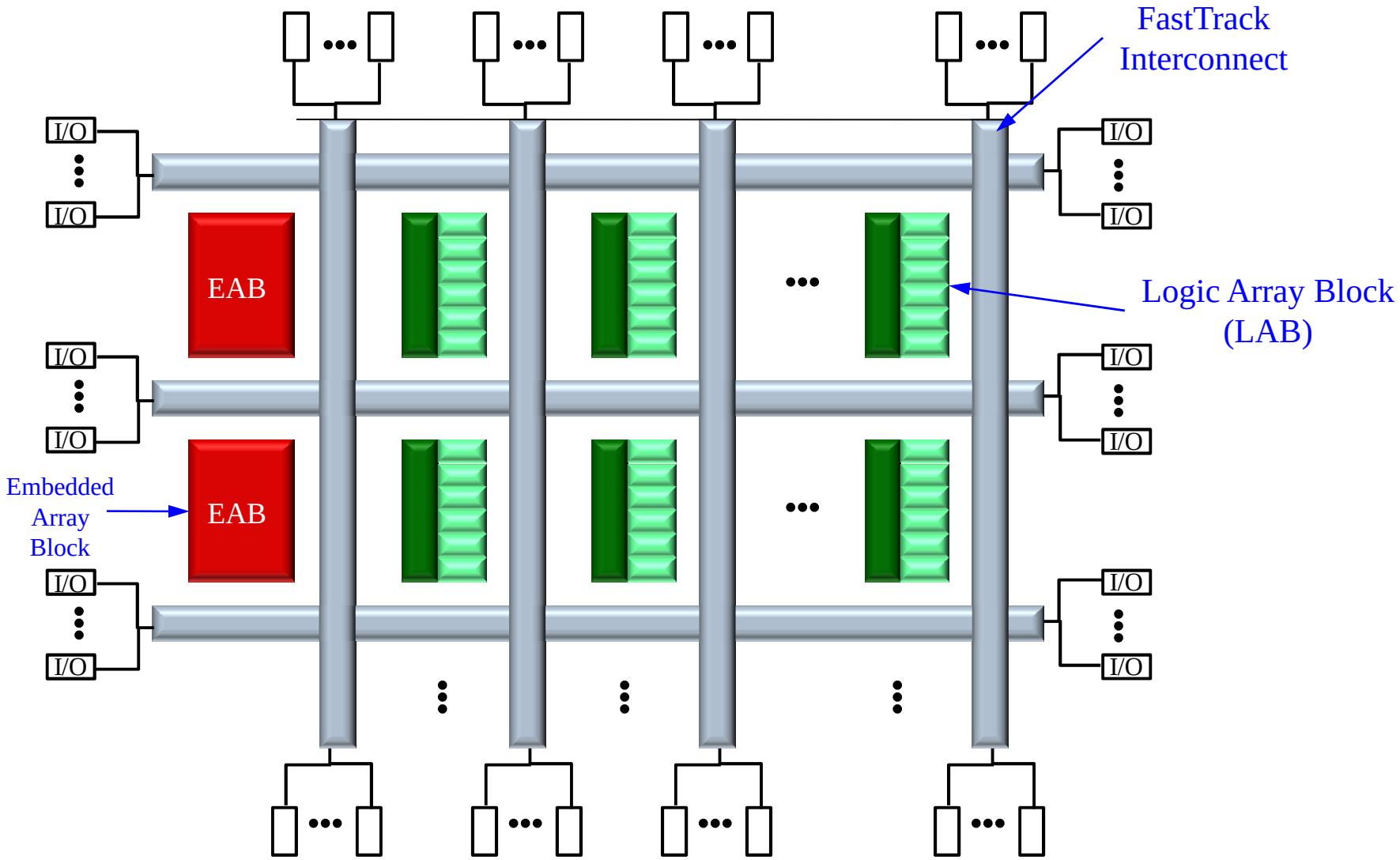
Design Styles – *Field Programmable Gate Array (FPGA)*

- SRAM-based programmable switch



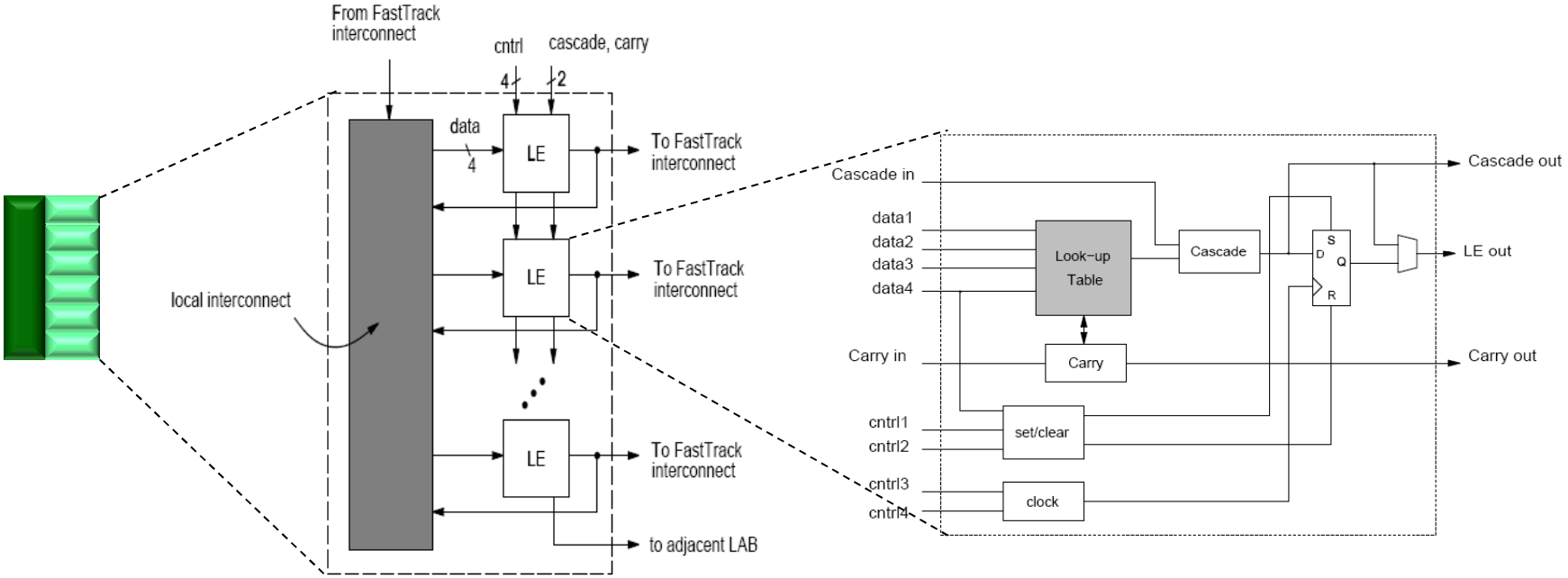
Design Styles – *Field Programmable Gate Array (FPGA)*

- Architecture of Altera FLEX 10K FPGA

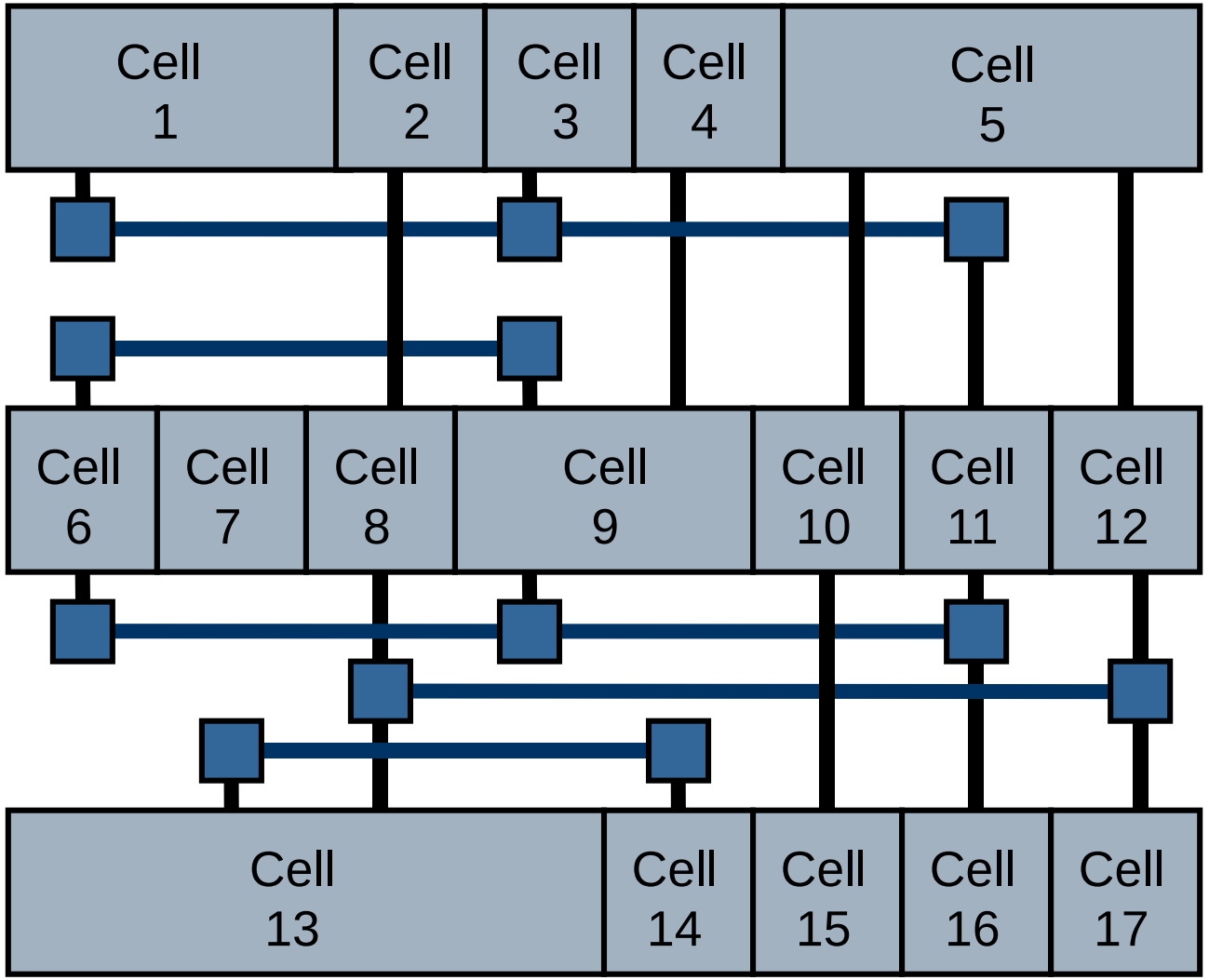


Design Styles – *Field Programmable Gate Array (FPGA)*

- Logic array block



Standard-Cell Design Styles

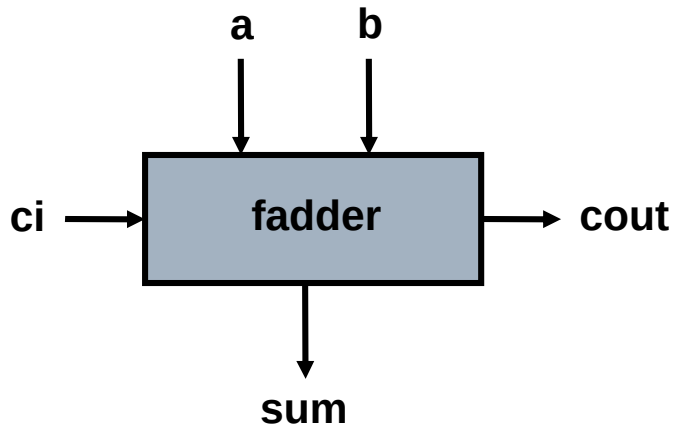


Standard-Cell Design Styles

- Design entry
  - Enter the design into an ASIC design system, either using a *hardware description language* (HDL) or *schematic entry*
- An example of Verilog HDL

```
module fadder(sum,cout,a,b,ci);
output sum, cout;
input a, b, ci;
reg sum, cout;

always @(a or b or ci) begin
   sum = a^b^ci;
   cout = (a&b)|(b&ci)|(ci&a);
end
endmodule
```

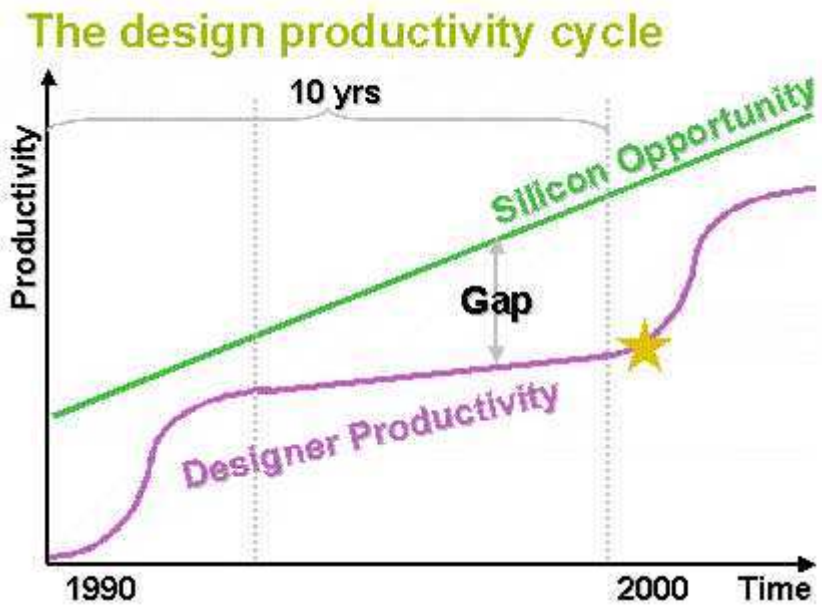


Design Styles – *Comparison*

Design Styles	Advantages	Disadvantages
Full-custom	- Compact designs; - Improved electrical characteristics;	- Very time consuming; - More error prone;
Semi-custom	-Well-tested standard cells which can be shared between users; -Good for bottom-up design;	-Can be time consuming to built-up standard cells; -Expensive in the short term but cheaper in long-term costs;
FPGA	-Fast implementation; -Easy updates;	-Can be wasteful of space and pin connections; -Relatively expensive in large volumes;

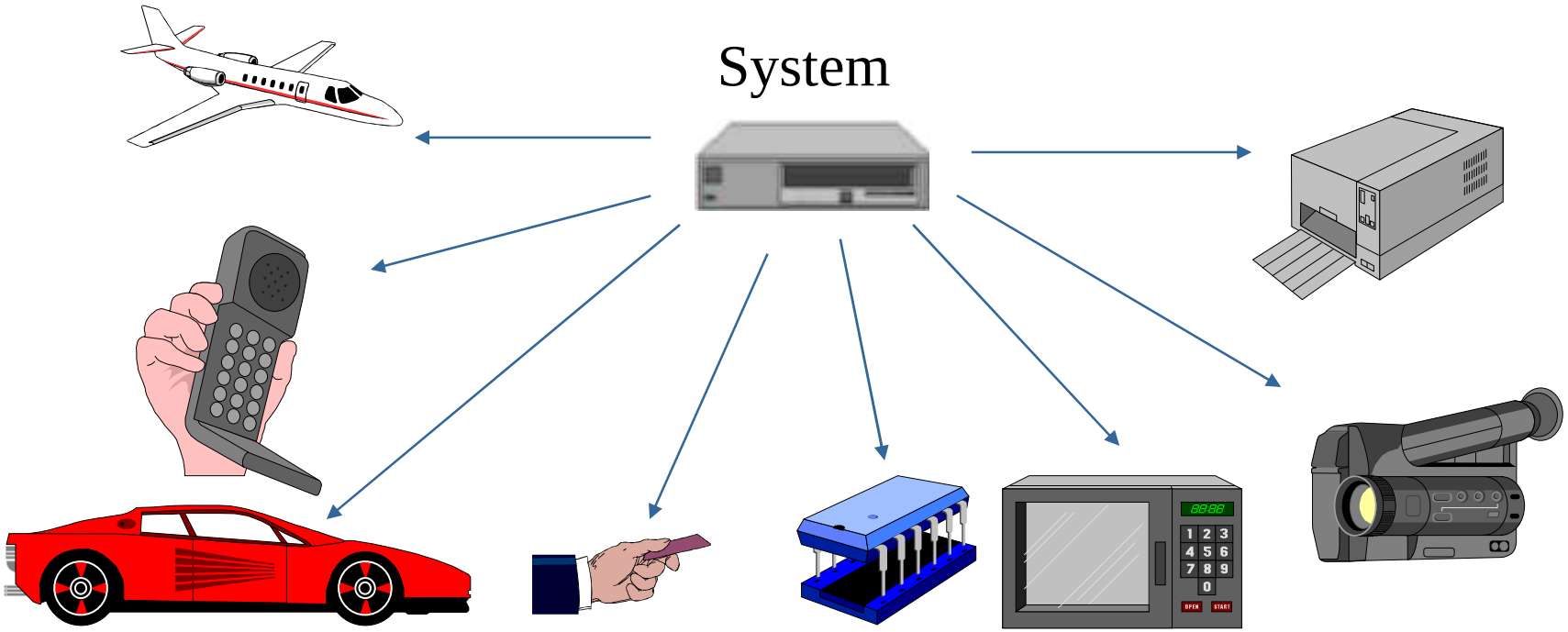
Emergence of SOC Idea

- Motivation:
  - Transistor density
  - Moor's law
- Integration with analog parts
  - AMS specification, synthesis, simulation
- SoC Design Methodology and Tools
  - Filling the Gap through
    - Reuse
    - Design Automation
    - System Specification Methodology



Source: Synopsys

What's a System?



[Source: M. Gudarzi]

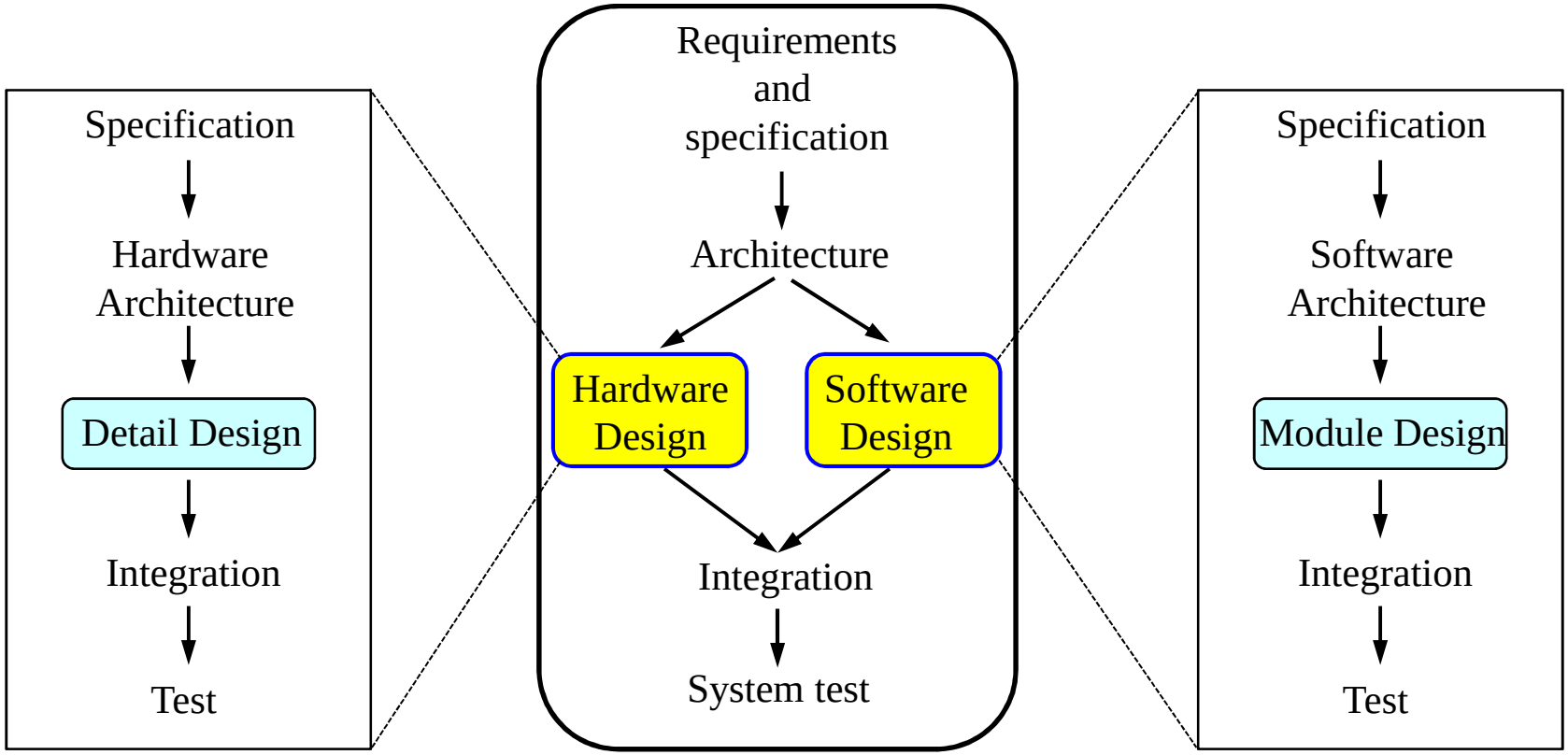
What's a System?

- Customer's view:
  System = User/Customer-specified ***functionality*** + ***requirements*** in terms of: Cost, Speed, Power, Dimensions, Weight, …

- Designer's view:
  System = ***HW*** components +***SW*** modules

[Source: M. Gudarzi]

Hierarchical Design Flow for an System Chip



HDL's & SDL's: Requirements

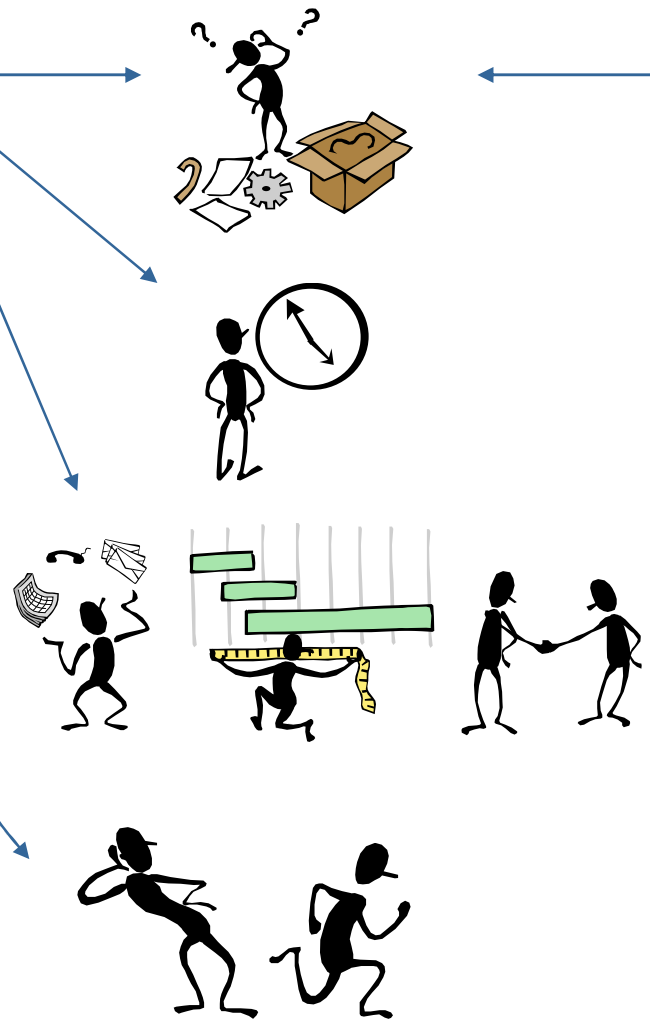
- HDL's
- HardwareC
- Verilog
- AHDL
- VHDL

SDL's

C

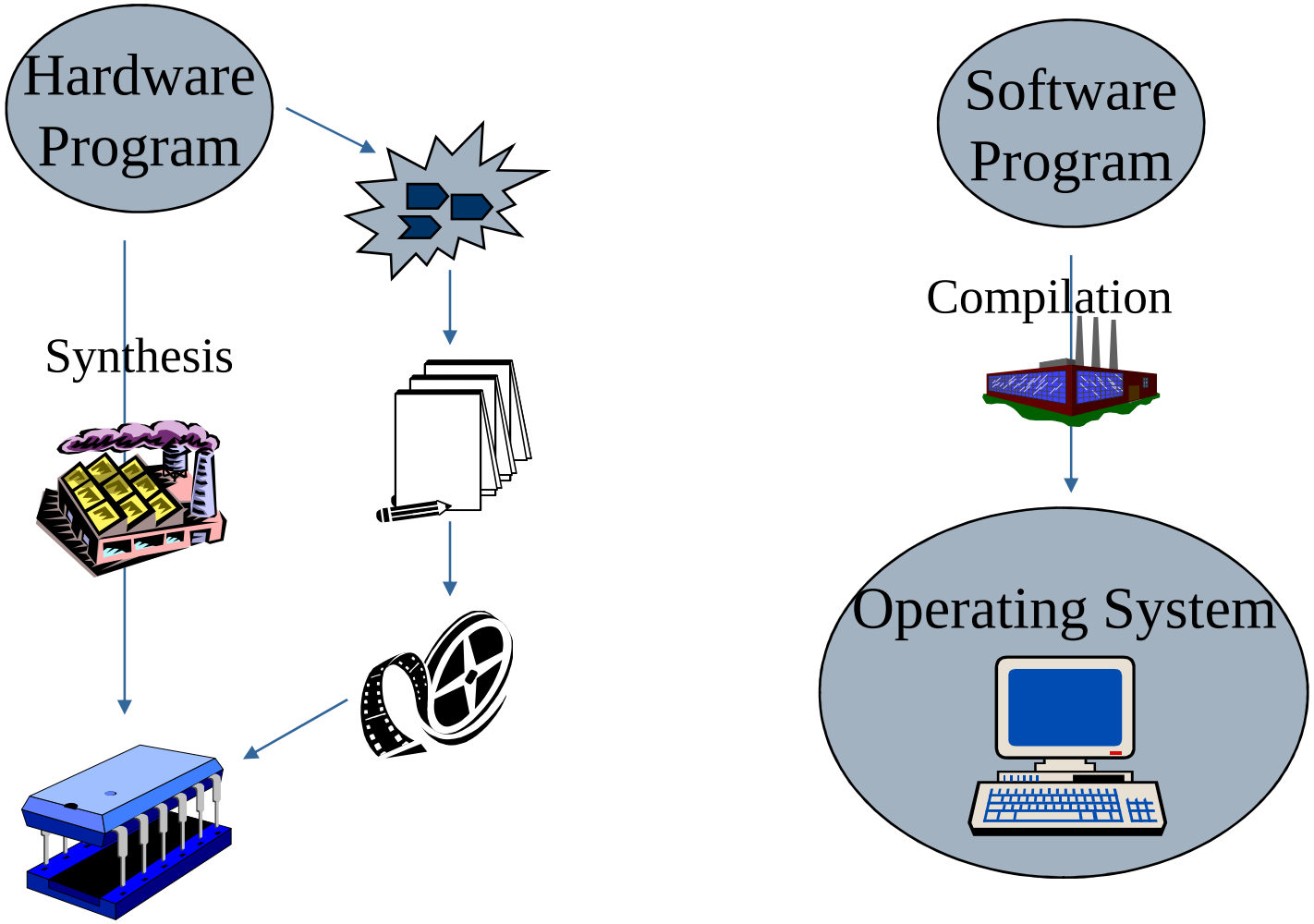
Pascal

ADA



[Source: M. Gudarzi]

HDL's & SDL's: Realization



[Source: M. Gudarzi]

HDL's & SDL's: Features

Any ***SW-realizable algorithm*** is ***HW-realizable*** as well.

- Hardware Realization
  - Speed
  - Energy Efficiency
  - Cost Efficiency (in high volumes)
- Software Realization
  - Flexibility
  - Ease of Development
  - Ease of Test and Debug
  - Cost = SW + Processor

[Source: M. Gudarzi]

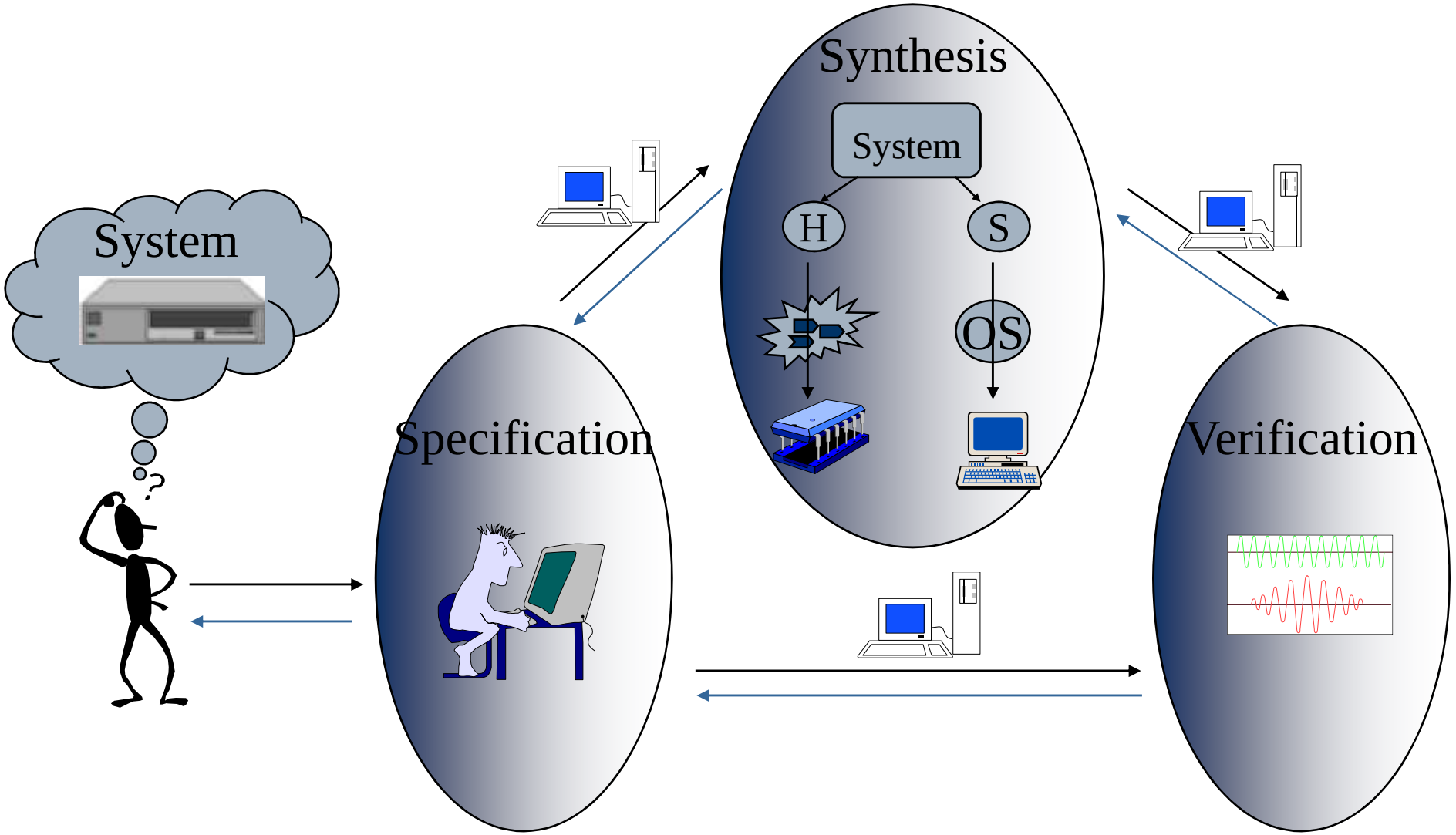
HW-SW Co-design

- How much SW + how much HW?
- Objectives:
  - Power
  - Speed
  - Area
  - Memory space
  - Time-to-market
- Implementation platform:
  - Collection of chips on a board (MCM)
  - …

HW/SW Co-Design Methodology

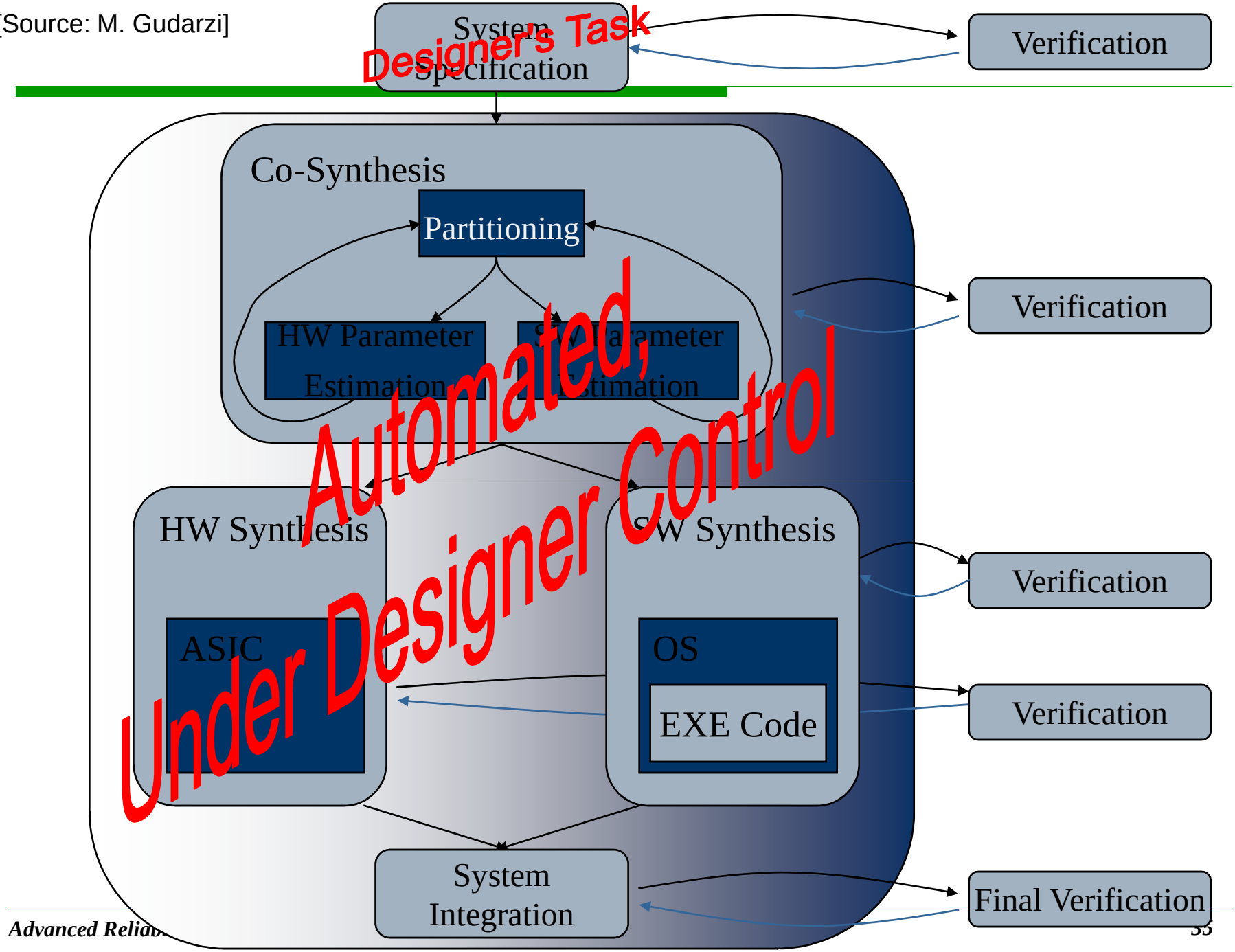
- Must architect hardware and software together:
  - provide sufficient resources;
  - avoid software bottlenecks.
- Can build pieces somewhat independently, but integration is major step.
- Also requires bottom-up feedback

HW/SW Co-design Main Topics



[Source: M. Gudarzi]

[Source: M. Gudarzi]



SOC Design Essentials

- Realization strategy:
  - Automated HW-SW Co-design + Reusable Cores
  - Intellectual Property: IP Cores
- IP Core Examples:
  - Processors: PowerPC, 680x0, ARM, …
  - Controllers: PCI, …
  - DSP Processors: TI
  - ...
- IP Core Categories:
  - Soft Cores: HDL, SW/HW Cores
  - Firm Cores: Synthesized HDL
  - Hard Cores: Layout for a specific fabrication process

Trends and Challenges of SOC Designs

- Technology trends
  - 3D technology + System-in-package
- Architecture trends
  - Regular architectures, e.g., multi-core architecture
  - Network-on-chip communication
- Challenges
  - Power
  - Reliability
  - Yield
  - Design-for-manufacturability
  - …