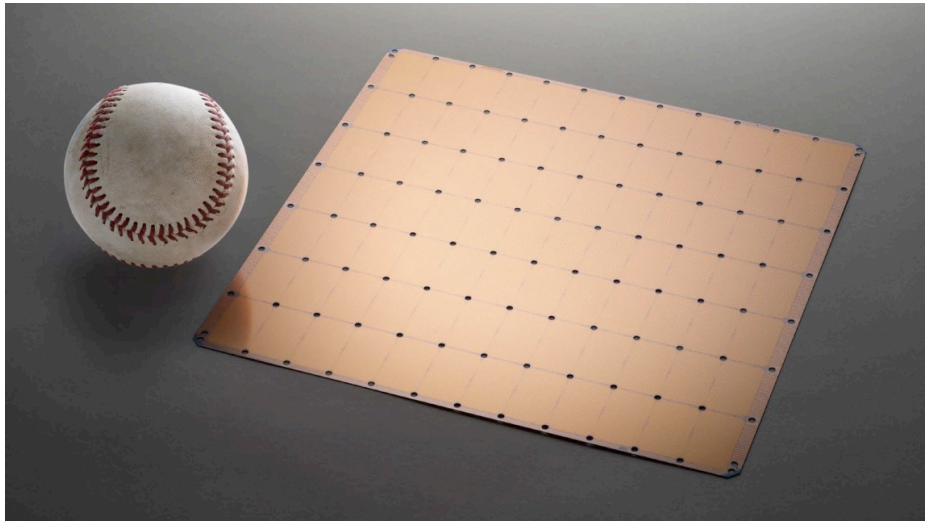
EECS151/251A
Introduction to Digital Design and ICs

Lecture 2 – Design Abstraction

Sophia Shao

"Largest Chip Ever Built!"

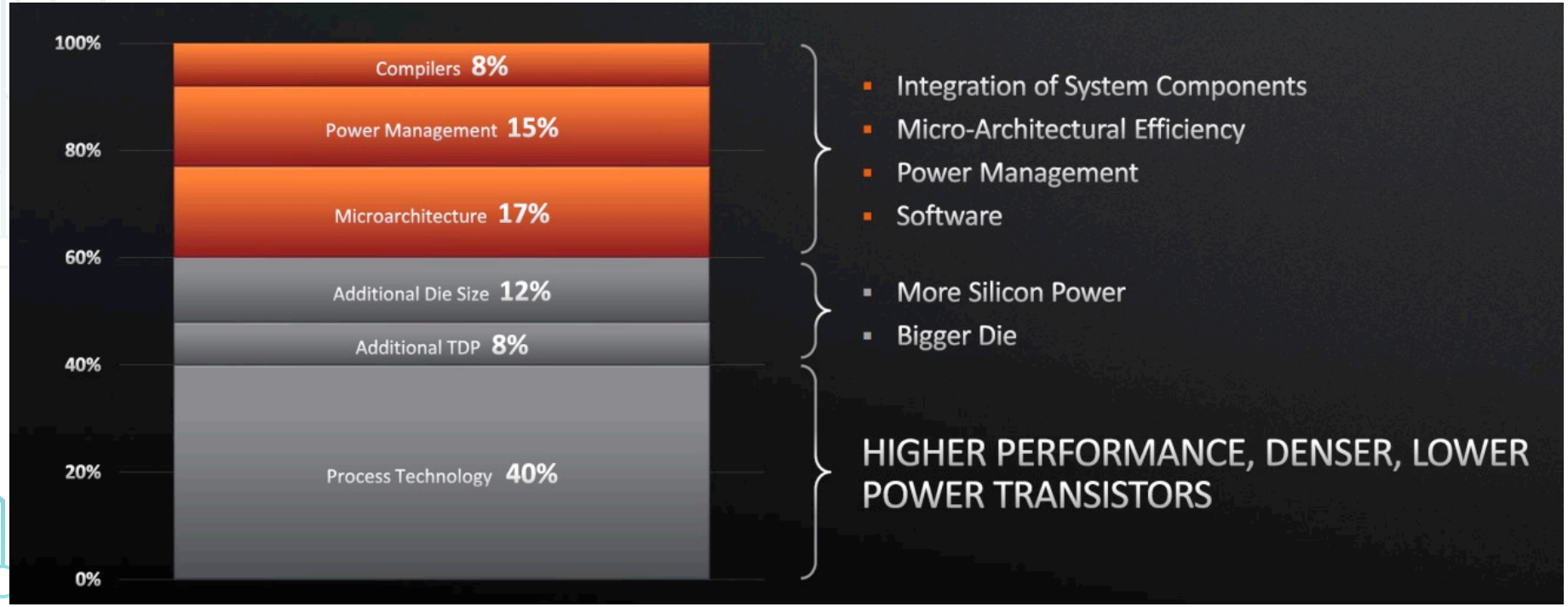
At HotChips'19 Cerebras announced the largest wafer-scale chip in the world at 46,225 mm^2 with 1.2 trillion transistors, and 15KW of power, aimed for training of deep-learning neural networks

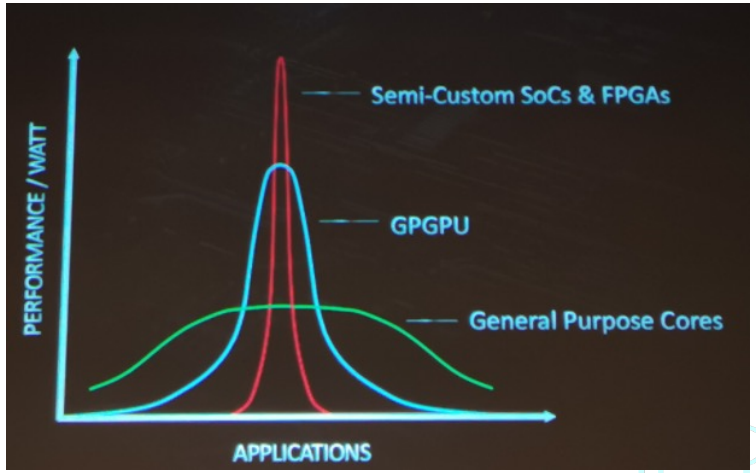


Review

- Moore's law is slowing down
  - There are continued improvements in technology, but at a slower pace
- Dennard's scaling has ended a decade ago
  - All designs are now power limited
- Specialization and customization provides added performance
  - Under power constraints and stagnant technology
- Design costs are high
  - Methodology and better reuse to rescue!

Putting it in Perspective



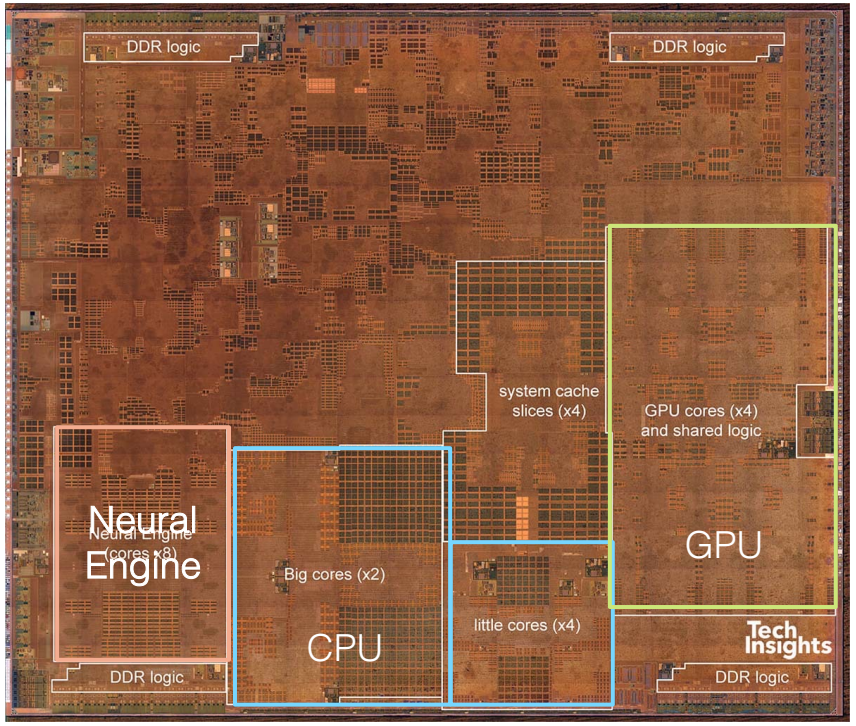


Lisa Su, HotChips'19 keynote

- Design Abstraction
- Implementing Digital System w/ Logic Gates

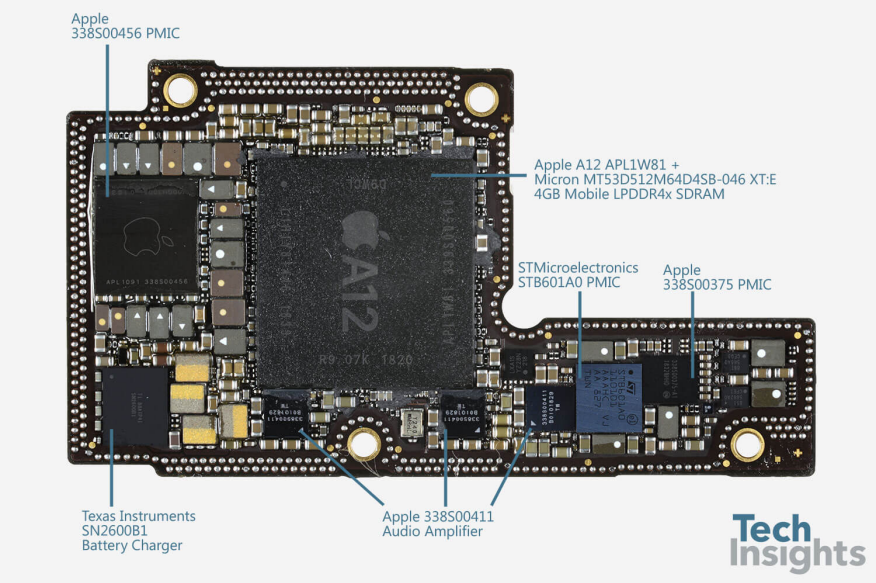
Modern System-On-Chip (SoC)

- Apple A12



TechInsights

- 2x Large CPUs
- 4x Small CPUs
- GPUs
- Neural processing unit (NPU)
- Lots of memory
- DDR memory interfaces

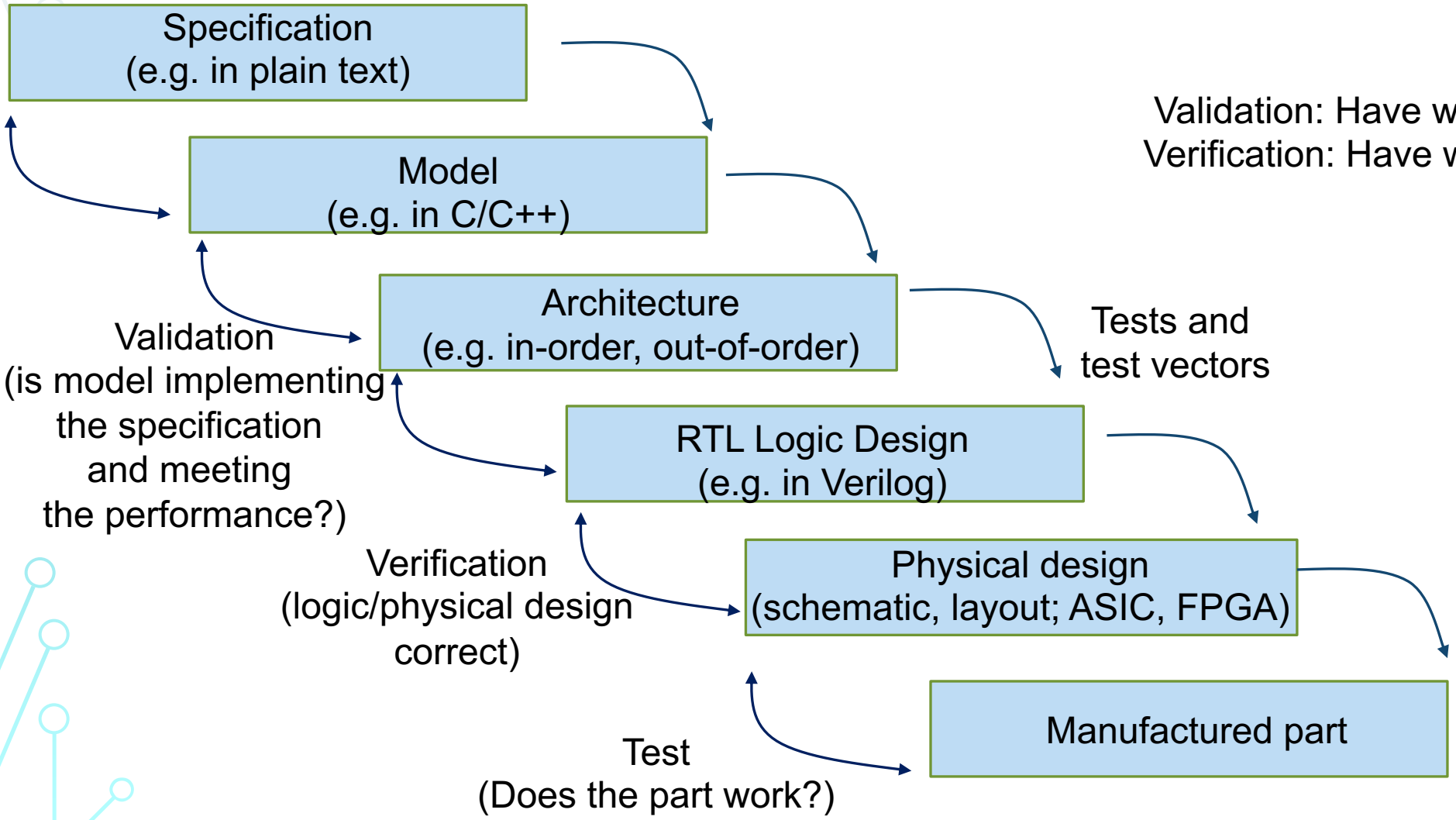


- 7nm CMOS
- Up to 2.49GHz

iPhone XS, 2018

Design Abstractions

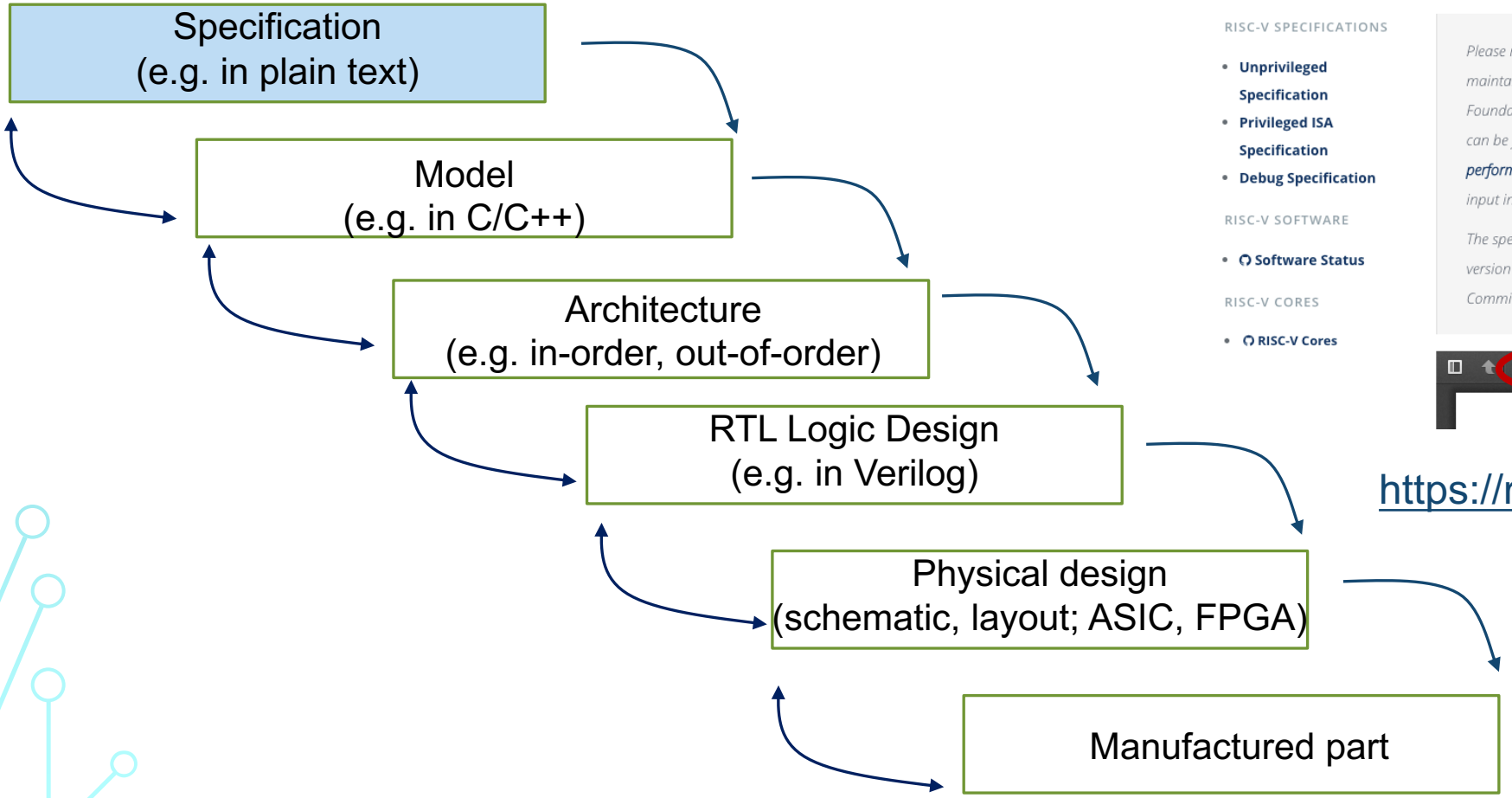
- Design through layers of abstractions

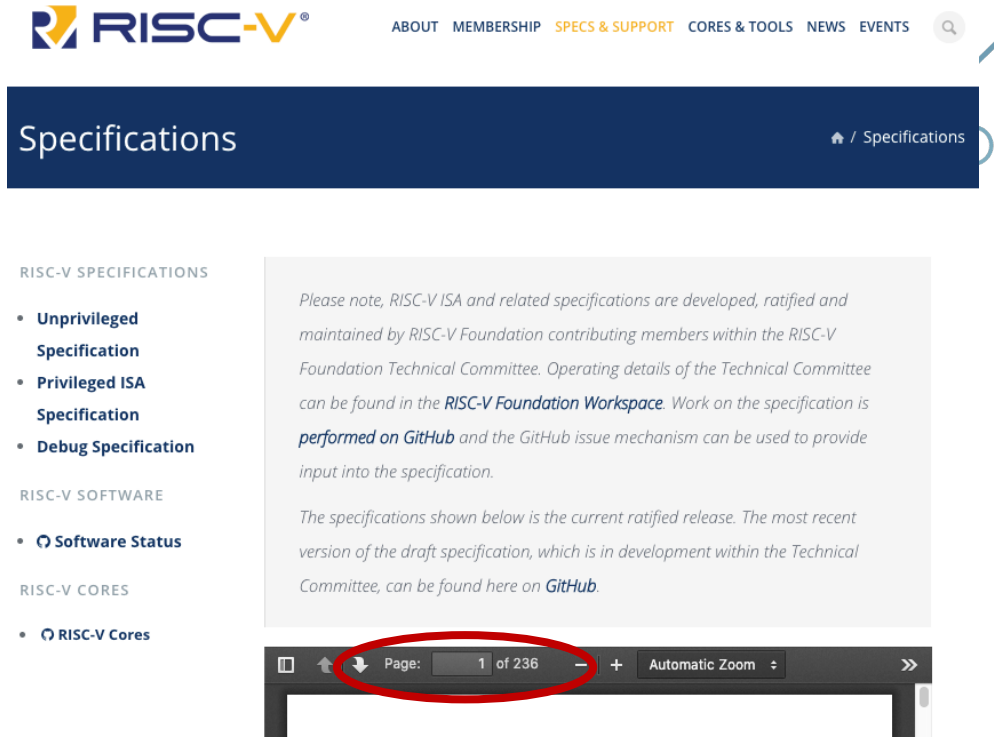


Validation: Have we built the right thing?
Verification: Have we built the thing right?

Example: RISC-V Design Process

- Design through layers of abstractions

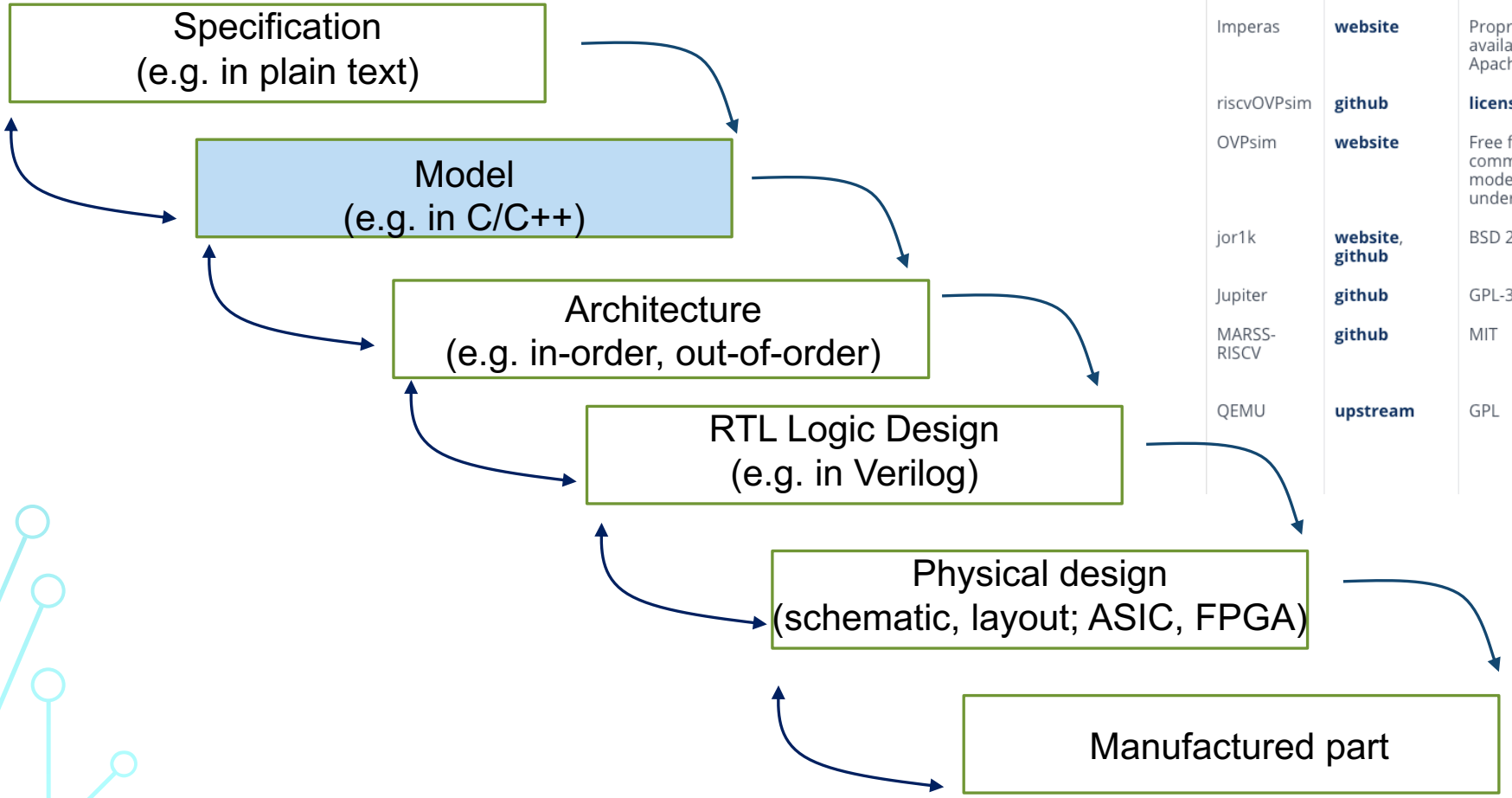




https://riscv.org/specifications/

Example: RISC-V Design Process

- Design through layers of abstractions



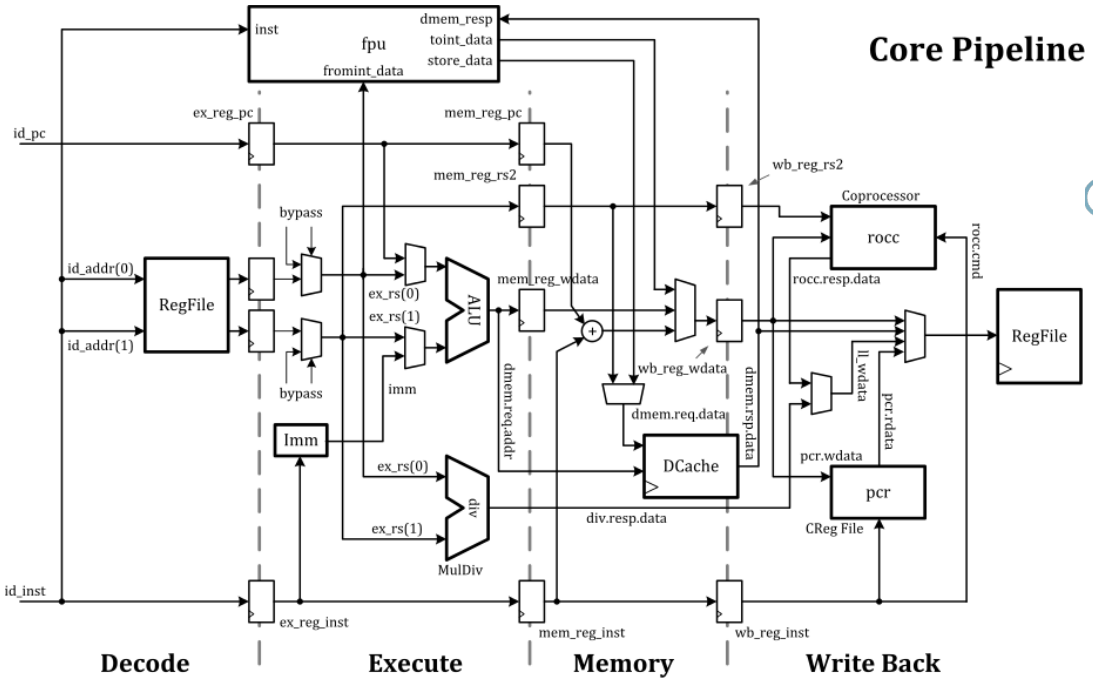
Simulators

Name	Links	License	Maintainers
DBT-RISE-RISCV	**github**	BSD-3-Clause	**MINRES Technologies**
FireSim	**website**, **mailing list**, **github**, **ISCA 2018 Paper**	BSD	Sagar Karandikar, Howard Mao, Donggyu Kim, David Biancolin, Alon Amid, **Berkeley Architecture Research**
gem5	**SW-dev thread**, **repository**	BSD-style	Alec Roelke (University of Virginia)
Imperas	**website**	Proprietary, models available under Apache 2.0	**Imperas**
riscvOVPsim	**github**	**license**	**Imperas**
OVPsim	**website**	Free for non commercial use, models available under Apache 2.0	**Imperas**
jor1k	**website**, **github**	BSD 2-Clause	Sebastian Macke
Jupiter	**github**	GPL-3.0	Andrés Castellanos
MARSS-RISCV	**github**	MIT	Gaurav N Kothari, Parikshit P Sarnaik, Gokturk Yuksek (State University of New York at Binghamton)
QEMU	**upstream**	GPL	Sagar Karandikar (University of California, Berkeley), Bastian Koppelmann (University of Paderborn), Alex Suykov, Stefan O'Rear and Michael Clark (SiFive)

https://riscv.org/software-status/#simulators

Example: RISC-V Design Process

- Design through layers of abstractions



Specification
(e.g. in plain text)

Model
(e.g. in C/C++)

Architecture
(e.g. in-order, out-of-order)

RTL Logic Design
(e.g. in Verilog)

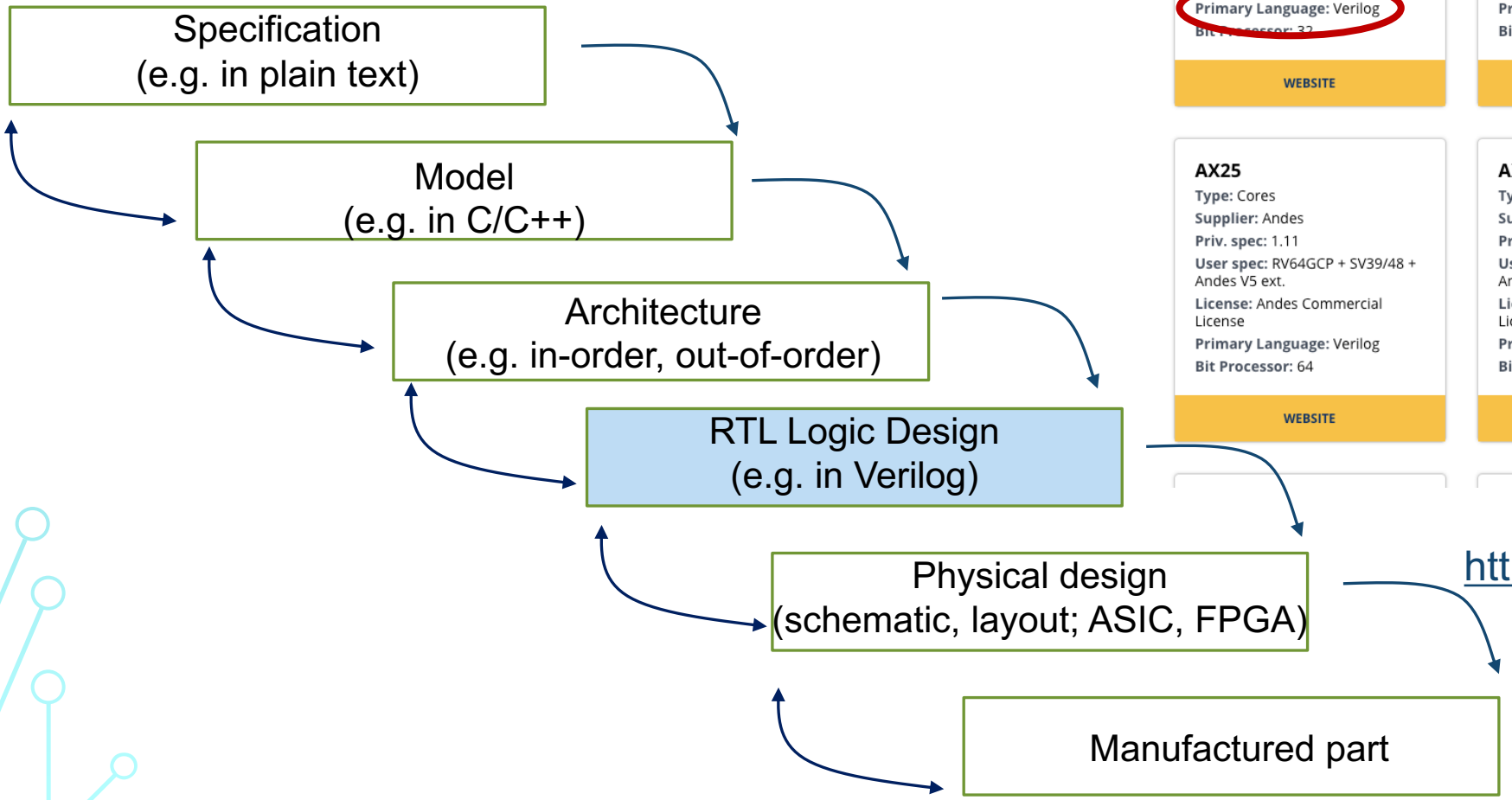
Physical design
(schematic, layout; ASIC, FPGA)

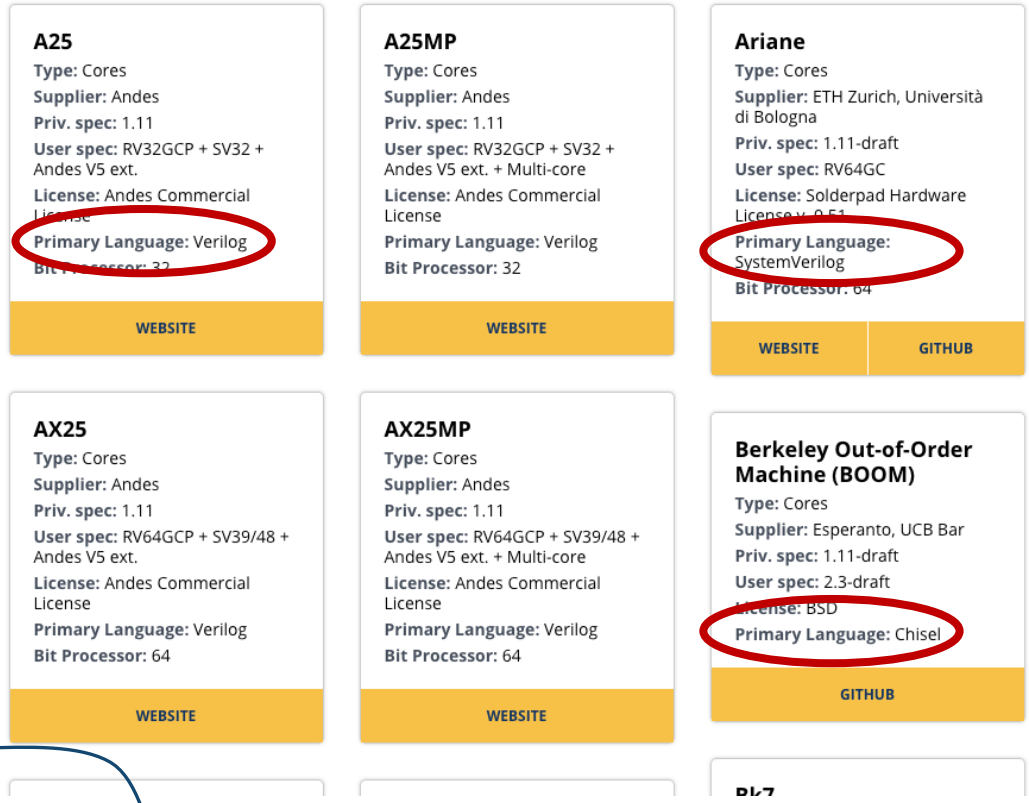
Manufactured part

https://www.lowrisc.org/docs/tagged-memory-v0.1/rocket-core/
…and CS152

Example: RISC-V Design Process

- Design through layers of abstractions

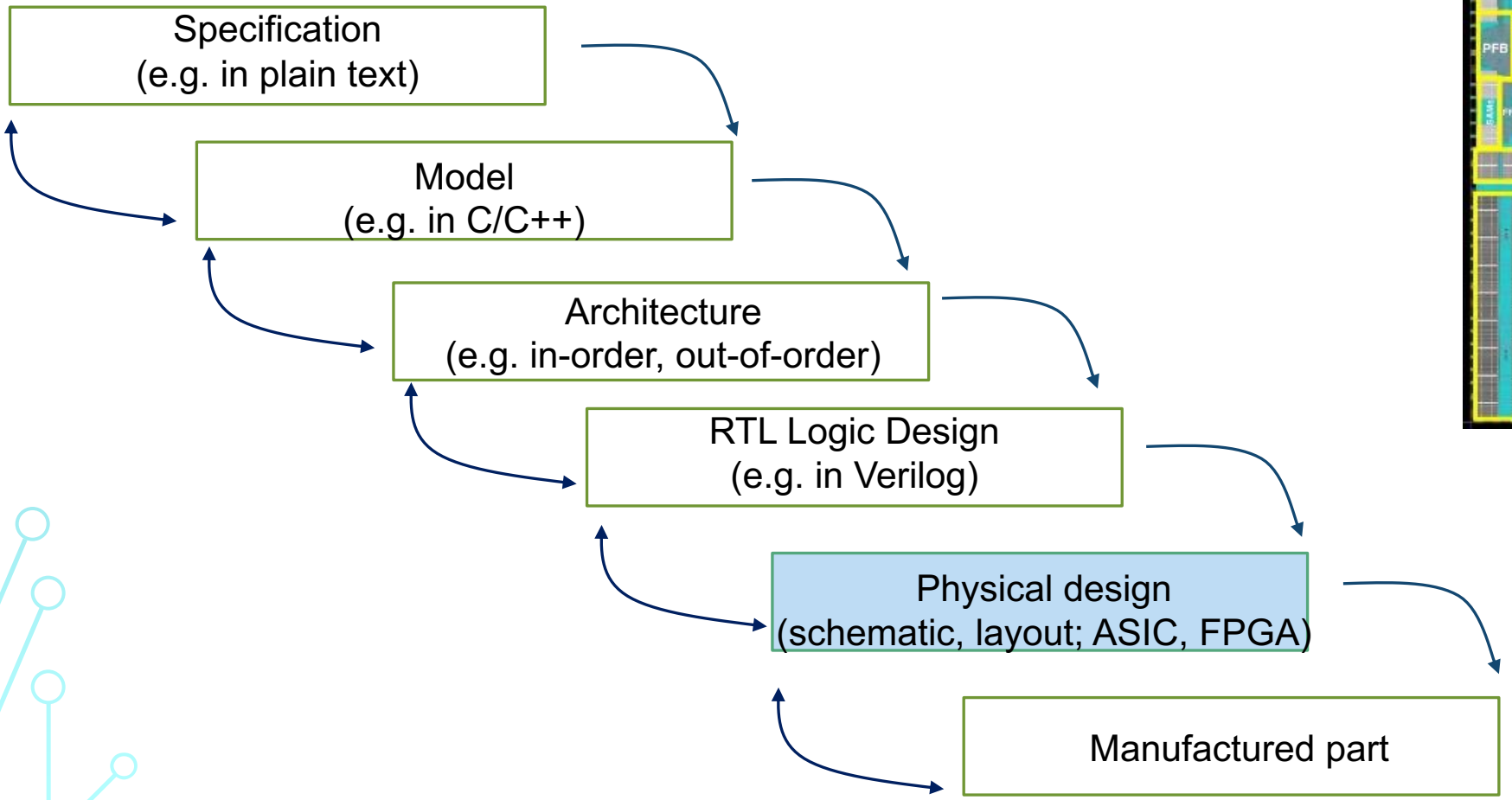


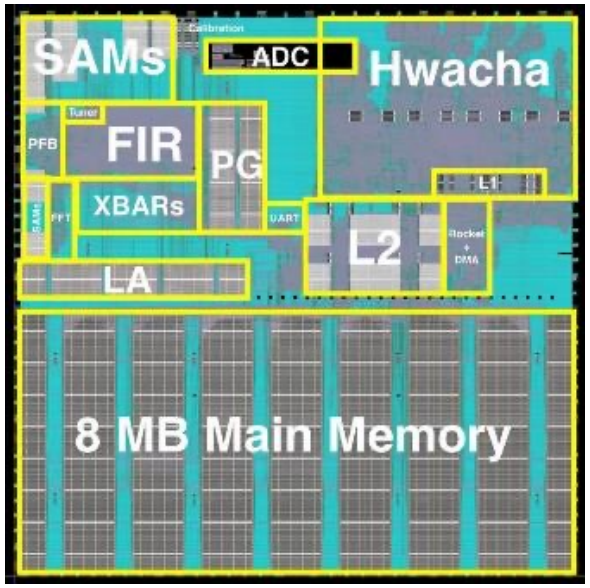


https://riscv.org/risc-v-cores/

Example: RISC-V Design Process

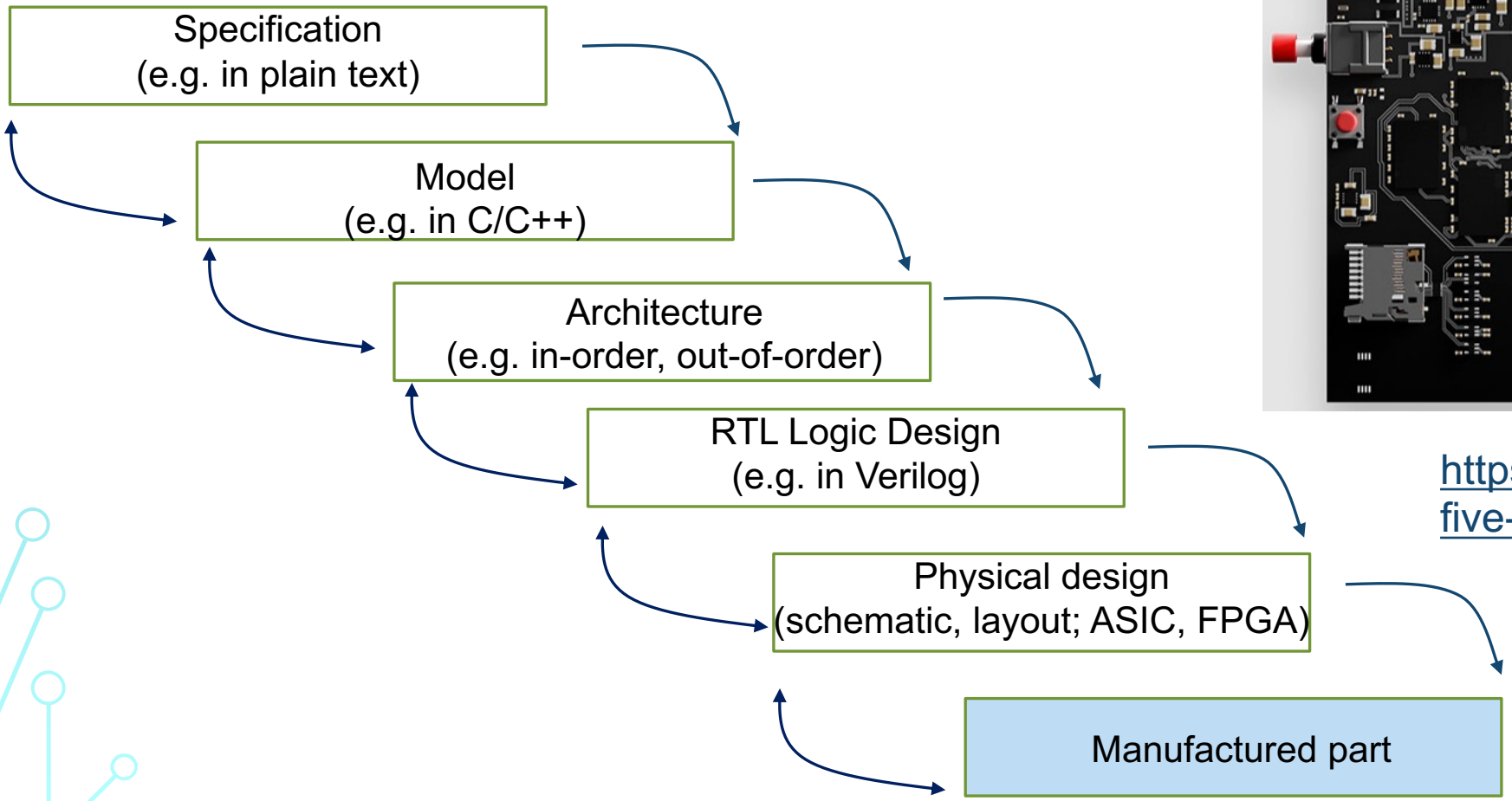
- Design through layers of abstractions

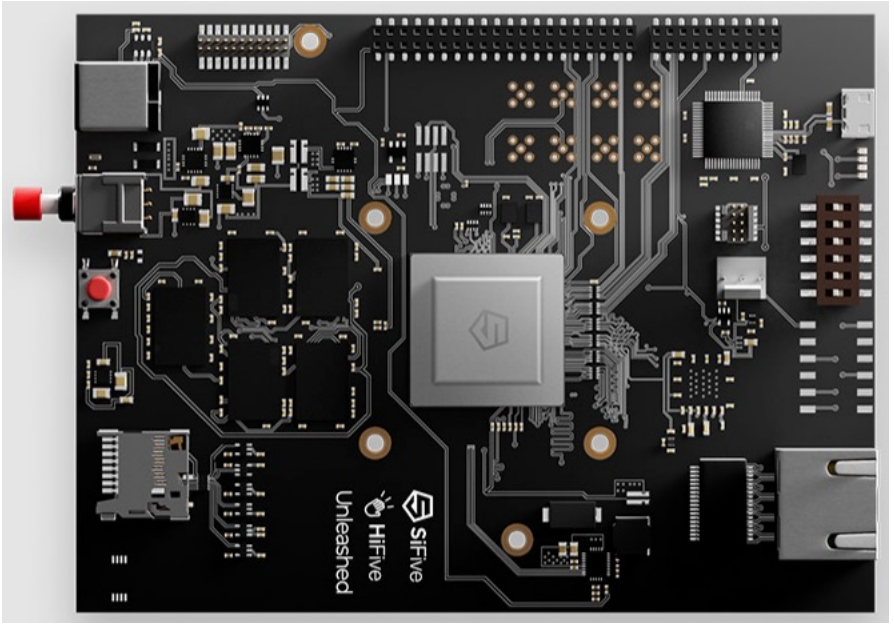




Example: RISC-V Design Process

- Design through layers of abstractions

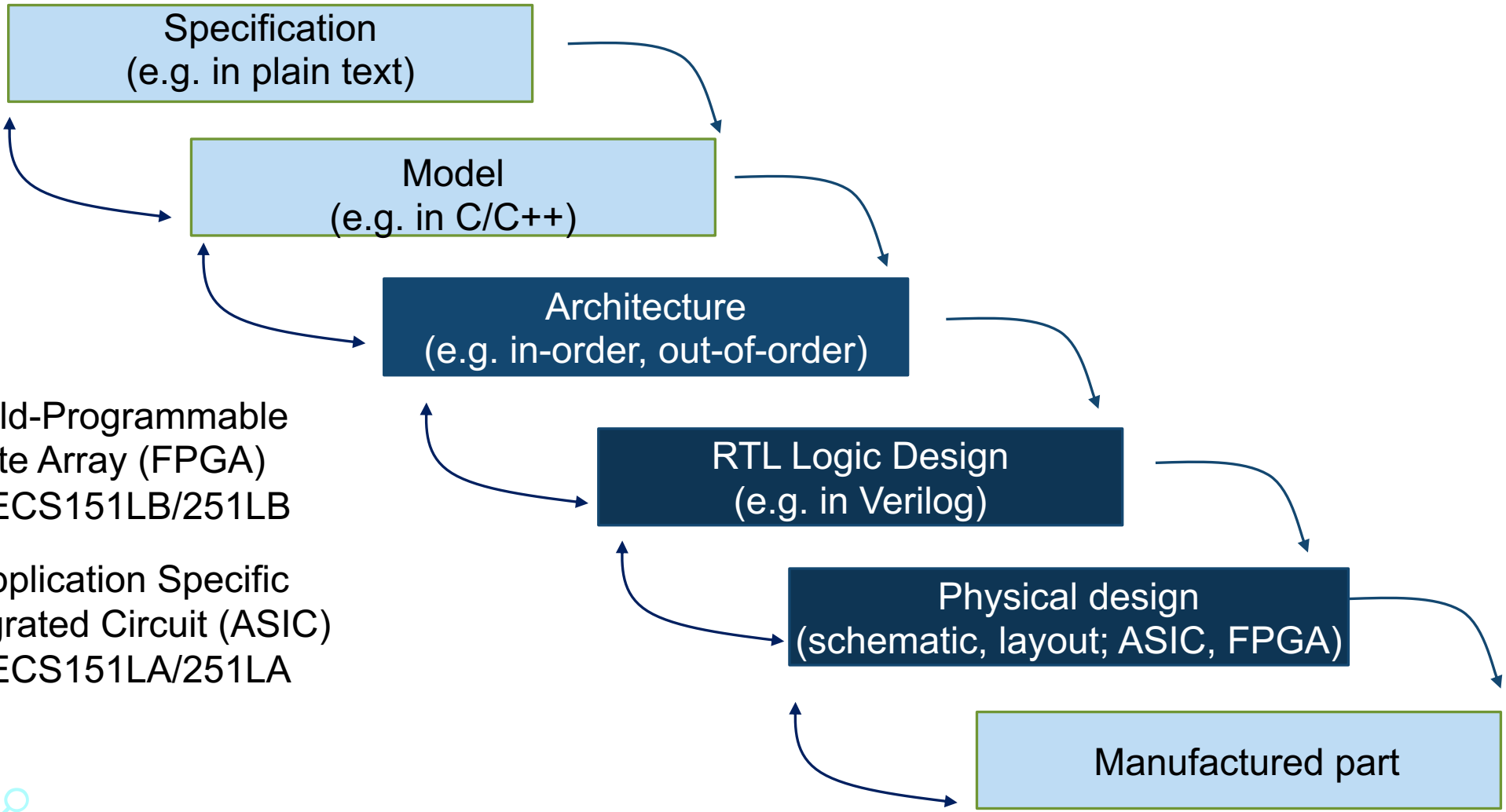




https://www.sifive.com/boards/hifive-unleashed

Design Abstractions in EECS151/251A

- Design through layers of abstractions



Field-Programmable Gate Array (FPGA)
EECS151LB/251LB

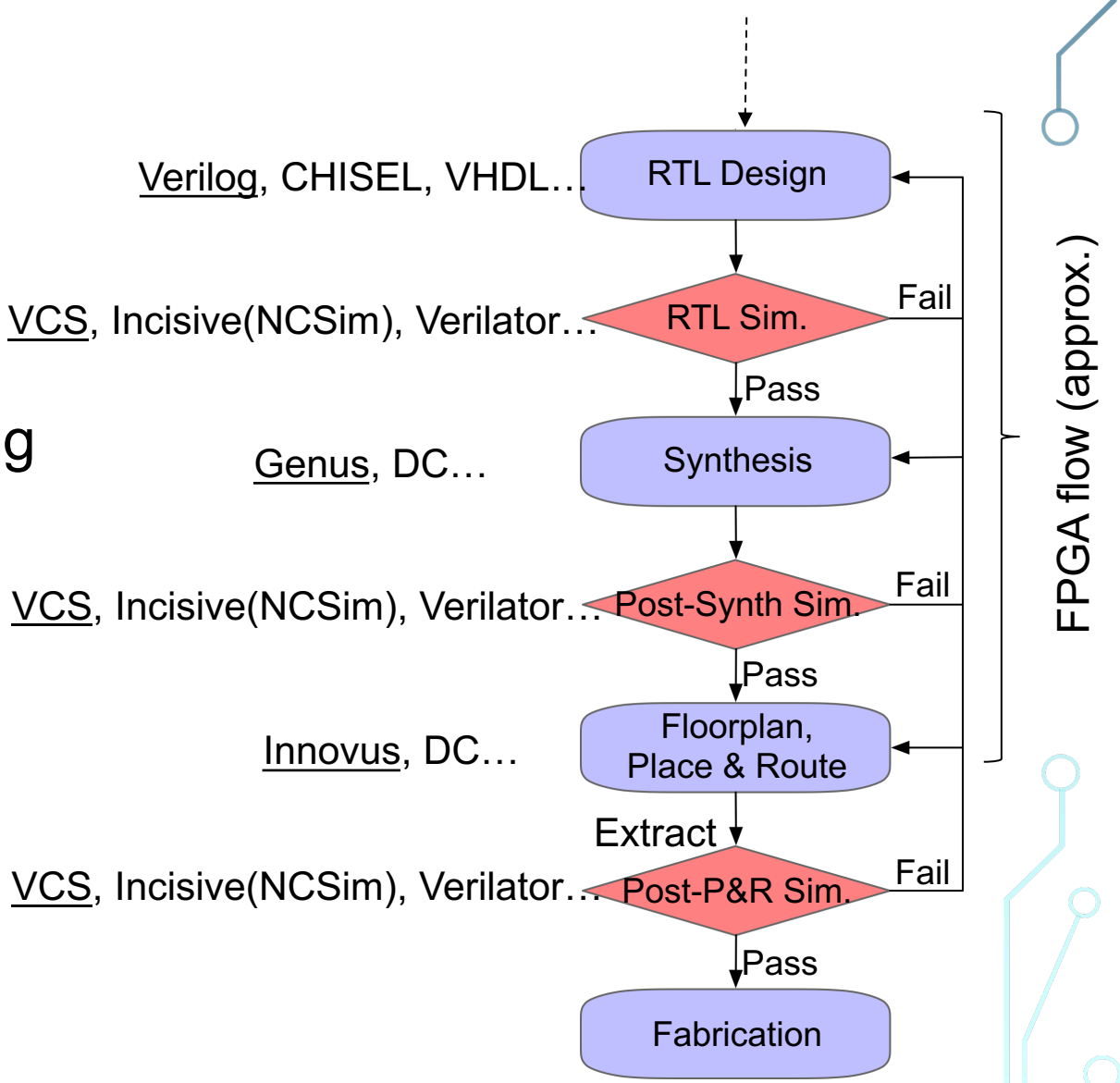
Application Specific Integrated Circuit (ASIC)
EECS151LA/251LA

RTL → Physical Design

RTL Logic Design
(e.g. in Verilog)

Physical design
(schematic, layout; ASIC, FPGA)

- Labs focus on a process of translating RTL to physical ASIC or FPGA by using industry-standard tools.
- Explores the entire design stack.



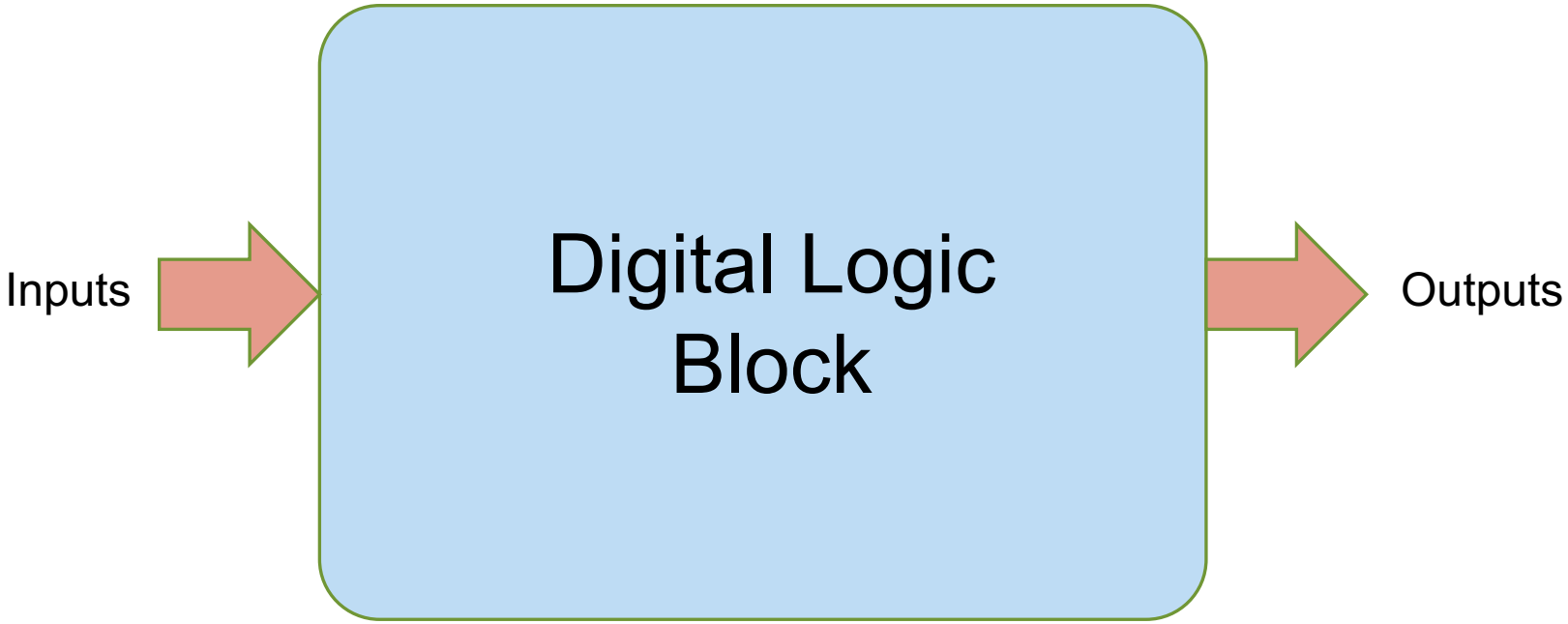
Announcement

- Labs start this week.
  - Check Ed on lab logistics.
  - Fill out the FPGA lab enrollment form.
- Discussions start this Friday.
  - Check course website for details.

- Design Abstraction
- Implementing Digital System w/ Logic Gates

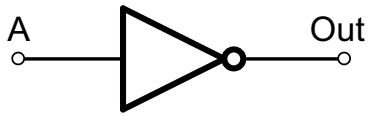
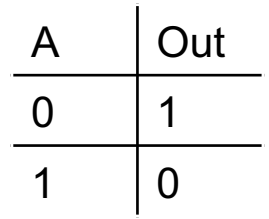
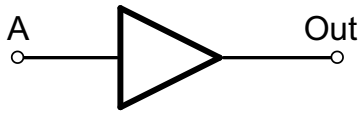
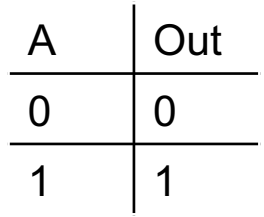
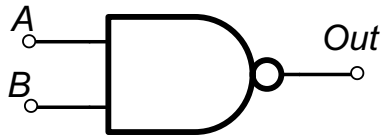
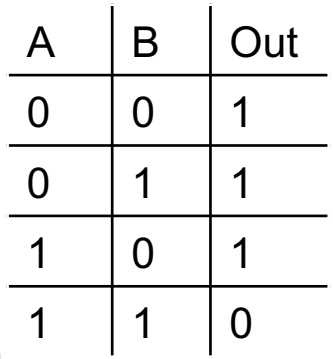
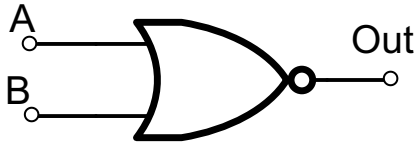
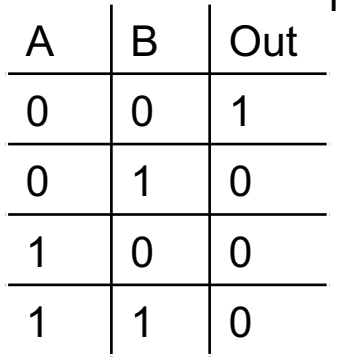
Implementing Digital Systems

- Digital systems implement a set of Boolean equations



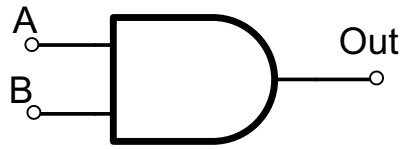
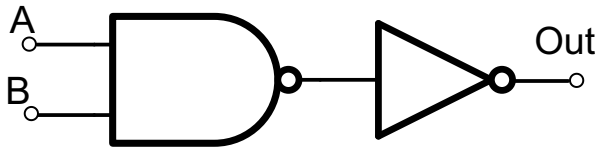
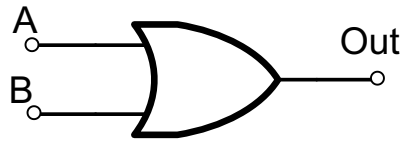
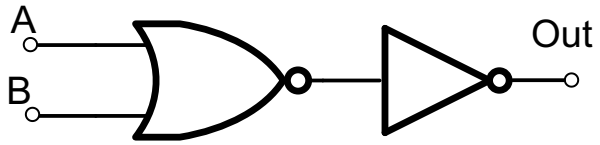
Logic Gates (From CS61C/EE16B)

- Logic gates

	Name	Boolean equation	Symbol	Truth table
Single input	NOT or Inverter	$Out = \overline{A}$		
	Buffer	$Out = A$		
	NAND	$Out = \overline{A \cdot B}$		
	NOR	$Out = \overline{A + B}$		

- CMOS gates are always inverting.

Logic Gates

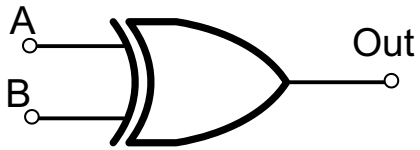
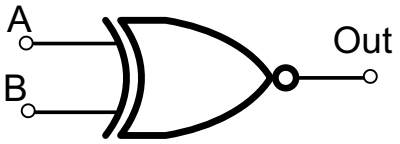
Name	Boolean equation	Symbol
AND	$Out = A \cdot B$	
In CMOS		
OR	$Out = A + B$	
In CMOS		

Truth table

A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

More Logic Gates

Name	Boolean equation	Symbol
Exclusive OR XOR	$Out = A \oplus B$	
Exclusive NOR XNOR	$Out = \overline{A \oplus B}$	

Truth table

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

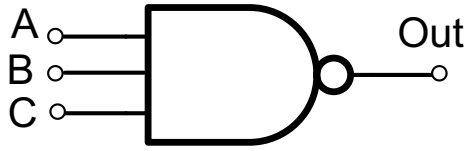
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

Multi-Input Gates

3-Input NAND

NAND3 Boolean equation

$$Out = \overline{A \cdot B \cdot C}$$

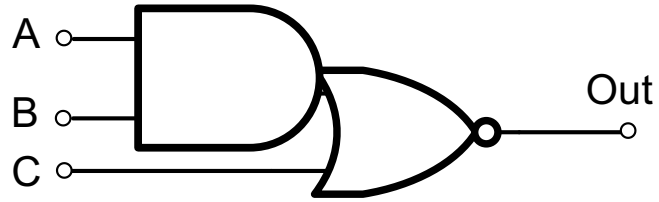


A	B	C	Out
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

And-Or-Invert

AOI21 Boolean equation

$$Out = \overline{A \cdot B + C}$$

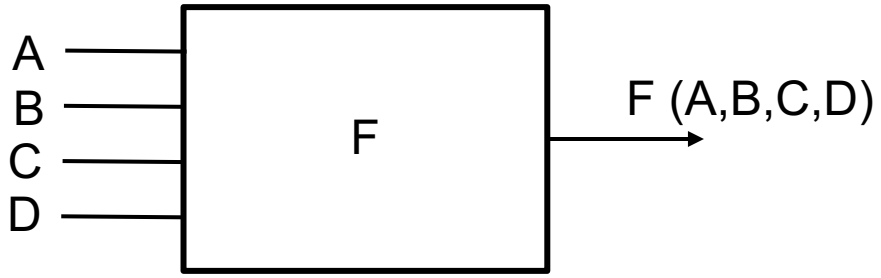


A	B	C	Out
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

- Single gate in modern CMOS usually doesn't have more than 3-4 inputs

Combinational Logic

- Output a function only of the current inputs (no history).
- Truth-table representation of function. Output is explicitly specified for each input combination.
- In general, CL blocks have more than one output signal, in which case, the truth-table will have multiple output columns.

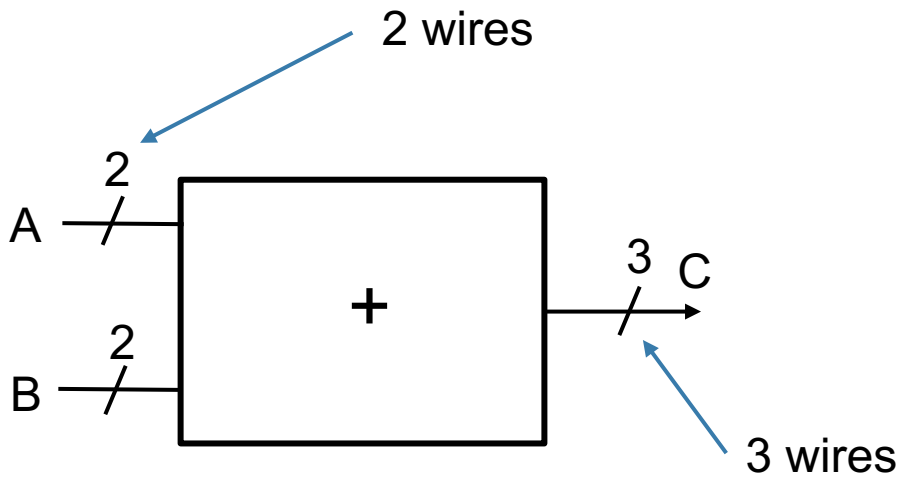


Truth Table

A	B	C	D	Out
0	0	0	0	F(0,0,0,0)
0	0	0	1	F(0,0,0,1)
0	0	1	0	F(0,0,1,0)
0	0	1	1	F(0,0,1,1)
0	1	0	0	F(0,1,0,0)
0	1	0	1	F(0,1,0,1)
0	1	1	0	F(0,1,1,0)
0	1	1	1	F(0,1,1,1)
1	0	0	0	F(1,0,0,0)
1	0	0	1	F(1,0,0,1)
1	0	1	0	F(1,0,1,0)
1	0	1	1	F(1,0,1,1)
1	1	0	0	F(1,1,0,0)
1	1	0	1	F(1,1,0,1)
1	1	1	0	F(1,1,1,0)
1	1	1	1	F(1,1,1,1)

Combinational Logic

- 2-bit adder. Takes two 2-bit integers and produces 3-bit result.



A1	A0	B1	B0	C2 C1 C0
0	0	0	0	0 0 0
0	0	0	1	0 0 1
0	0	1	0	0 1 0
0	0	1	1	0 1 1
0	1	0	0	0 0 1
0	1	0	1	0 1 0
0	1	1	0	0 1 1
0	1	1	1	1 0 0
1	0	0	0	0 1 0
1	0	0	1	0 1 1
1	0	1	0	1 0 0
1	0	1	1	1 0 1
1	1	0	0	0 1 1
1	1	0	1	1 0 0
1	1	1	0	1 0 1
1	1	1	1	1 1 0

- Think about truth table for 32-bit adder. It's possible to write out, but it might take a while!

Peer Instruction

Total number of possible **truth tables** with 4 inputs and 1 output is:

a) 4

b) 16

c) 256

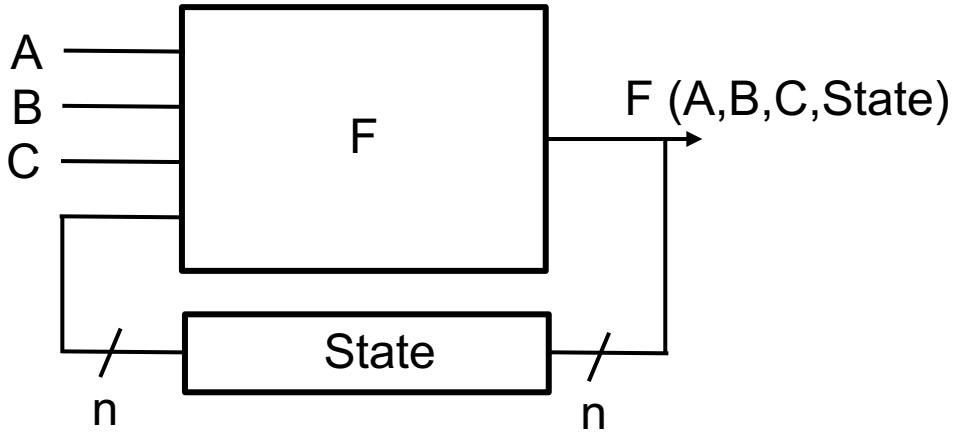
d) 16,384

e) 65,536

f) None of the above

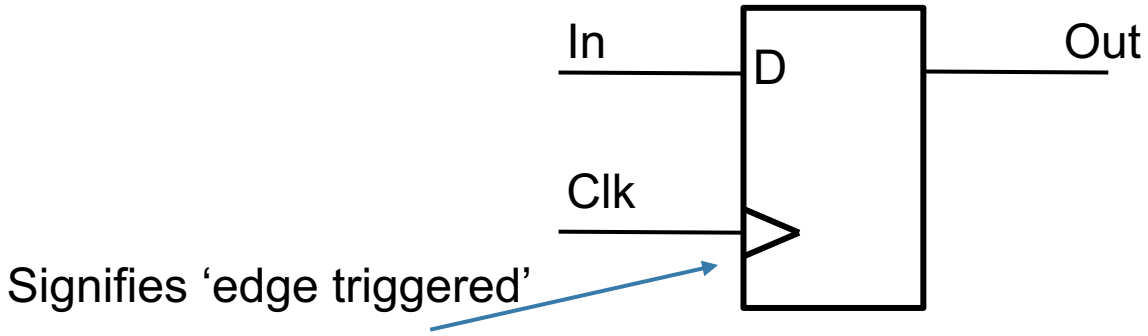
Sequential Logic

- Output is a function of both the current inputs and the state.
- State represents the memory.
- State is a function of previous inputs.
- In synchronous digital systems, state is updated on each clock tick.

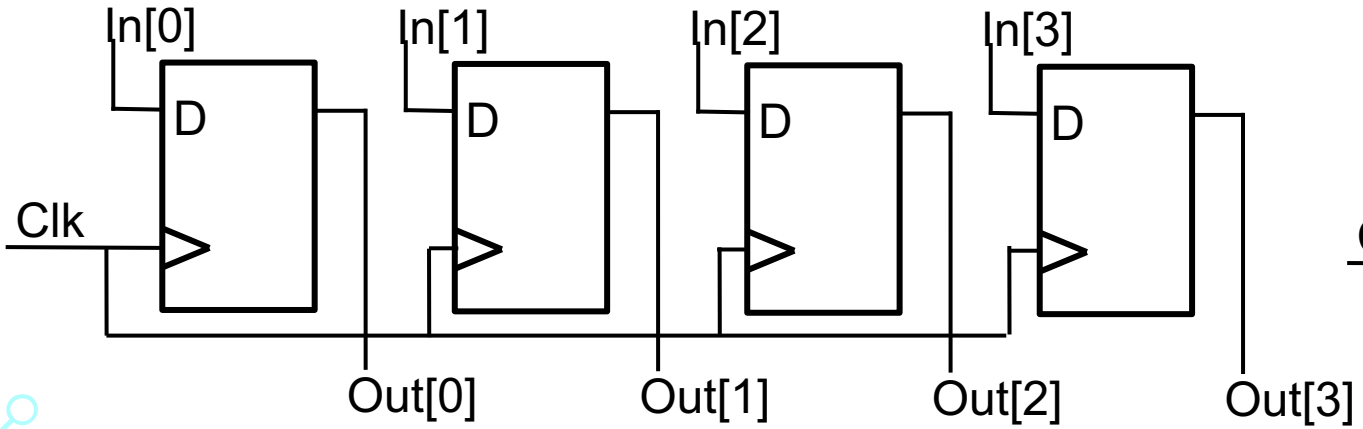


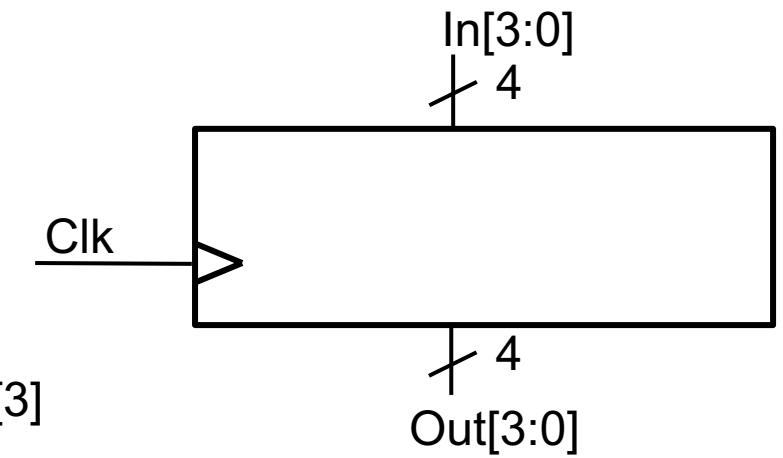
Sequential Logic Example: Flip-Flop

- Synchronous state element transfers its input to the output on a rising (or, rarely, falling) clock edge
- Flip-flop
  - Rising edge



- 4-bit register

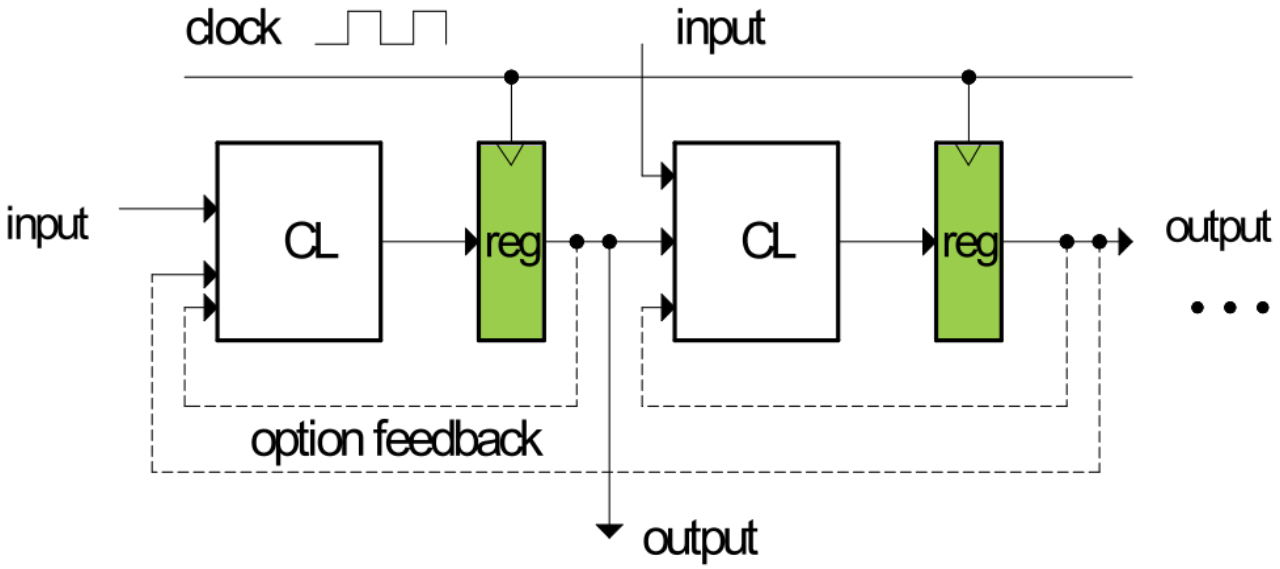




Register Transfer Level Abstraction (RTL)

Any synchronous digital circuit can be represented with:

- Combinational Logic Blocks (CL), plus
- State Elements (registers or memories)
- Clock orchestrates *sequencing* of CL operations



- State elements are combined with CL blocks to control the flow of data.

Summary

- The design process involves traversing the layers of specification, modeling, architecture, RTL design and physical implementation
- Digital systems implement a set of Boolean equations