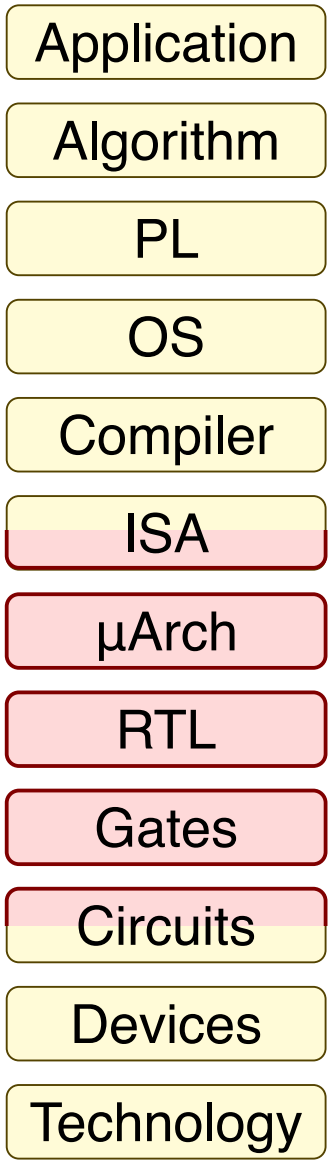
ECE 2300 Digital Logic and Computer Organization Course Overview

Christopher Batten

School of Electrical and Computer Engineering
Cornell University

`http://www.csl.cornell.edu/courses/ece2300`



Agenda

What is Digital Logic and Computer Organization?

Trend 1: A New Era of Hardware Specialization

Trend 2: Hardware for AI and AI for Hardware

Course Logistics

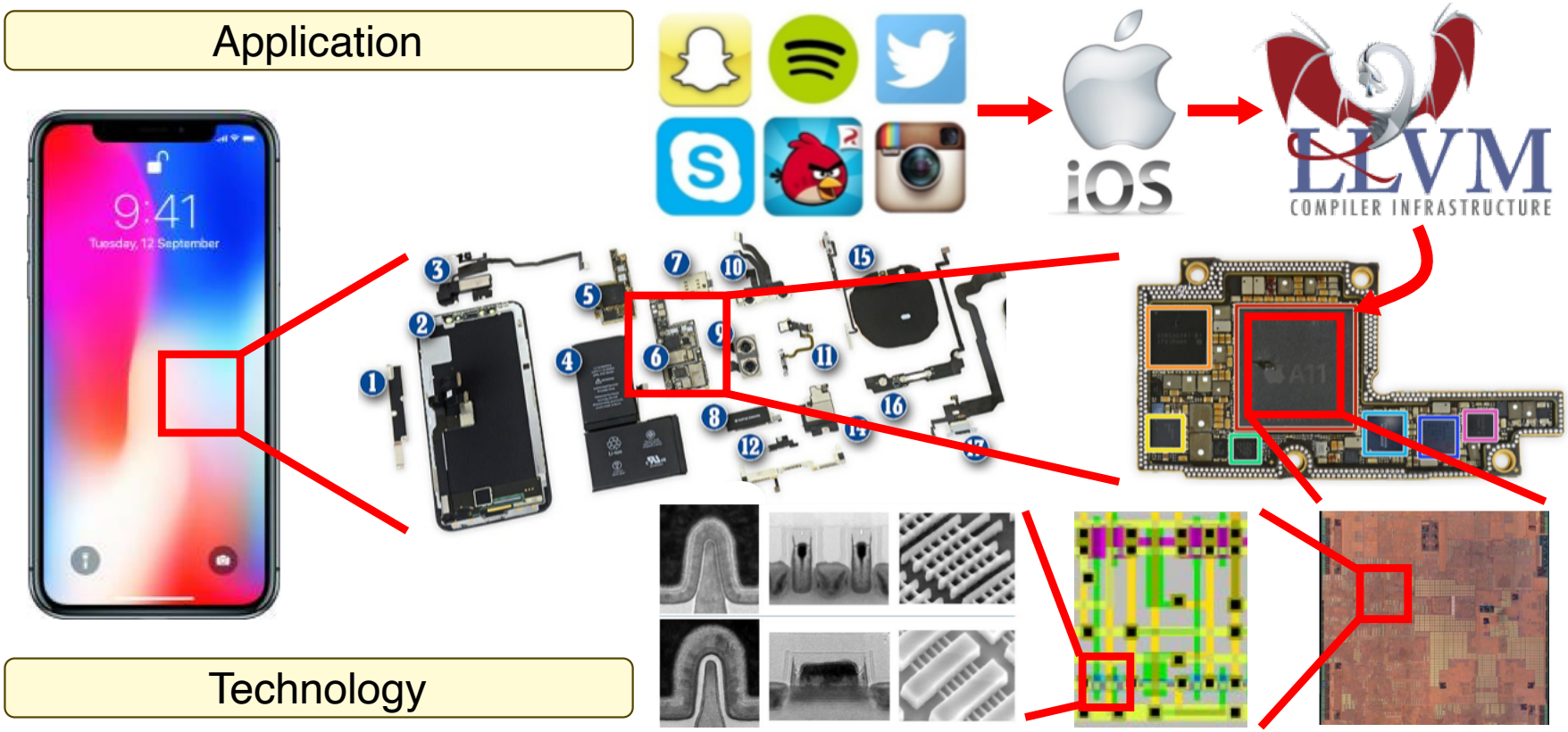
The Complexity of Modern Computer Systems

Application

Trivial systems might be able to bridge gap in one step

Technology

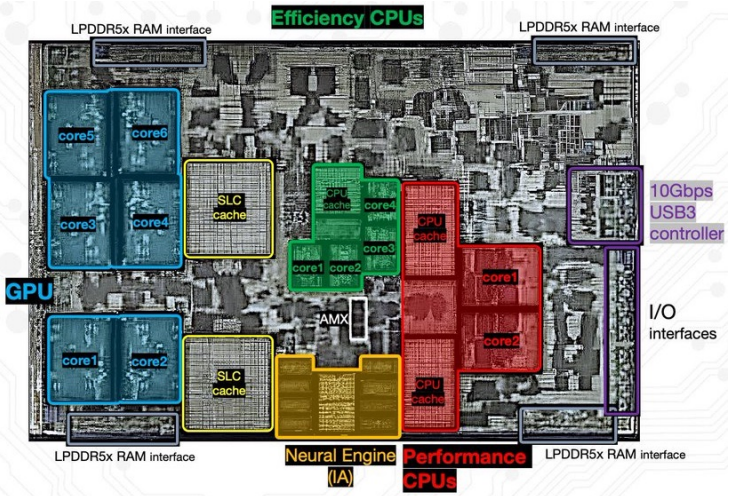
The Complexity of Modern Computer Systems



The Complexity of Modern Computer Systems

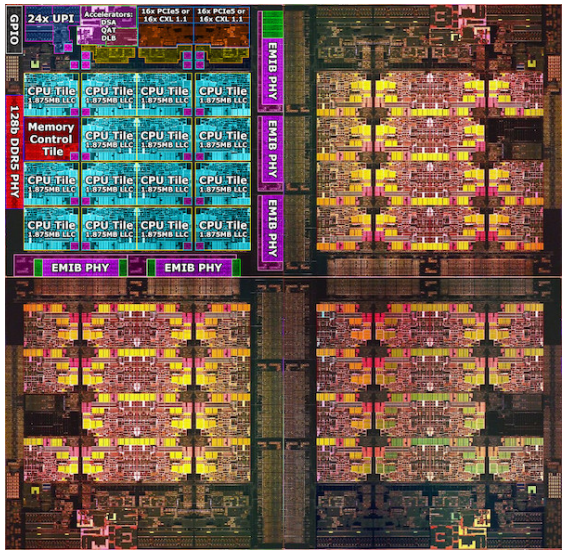
Application

Technology

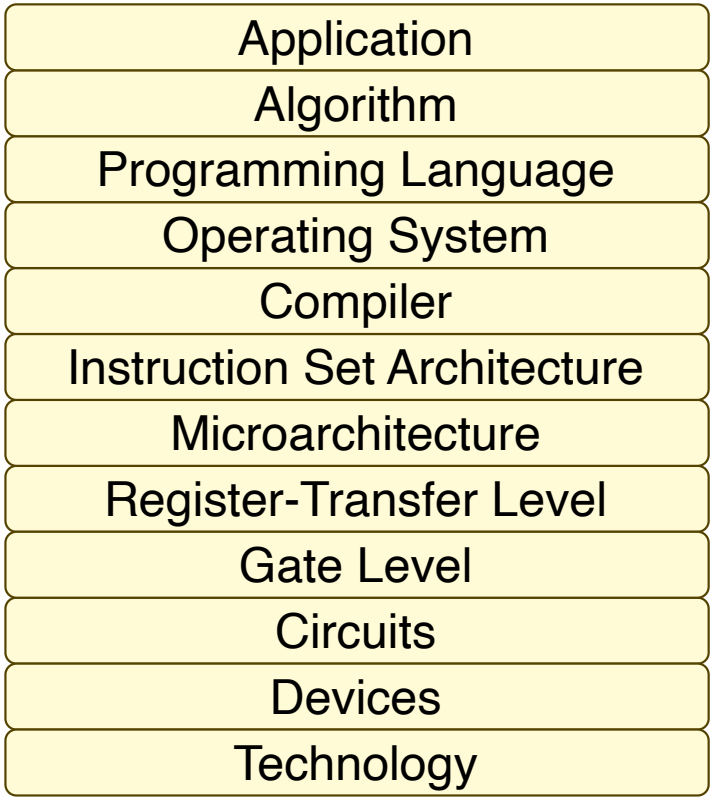


Apple A17 Pro (2023)
19B transistors
10s of unique subsystems

Intel Saphire Rapids (2023)
~30B transistors
60 cores
4 chiplets

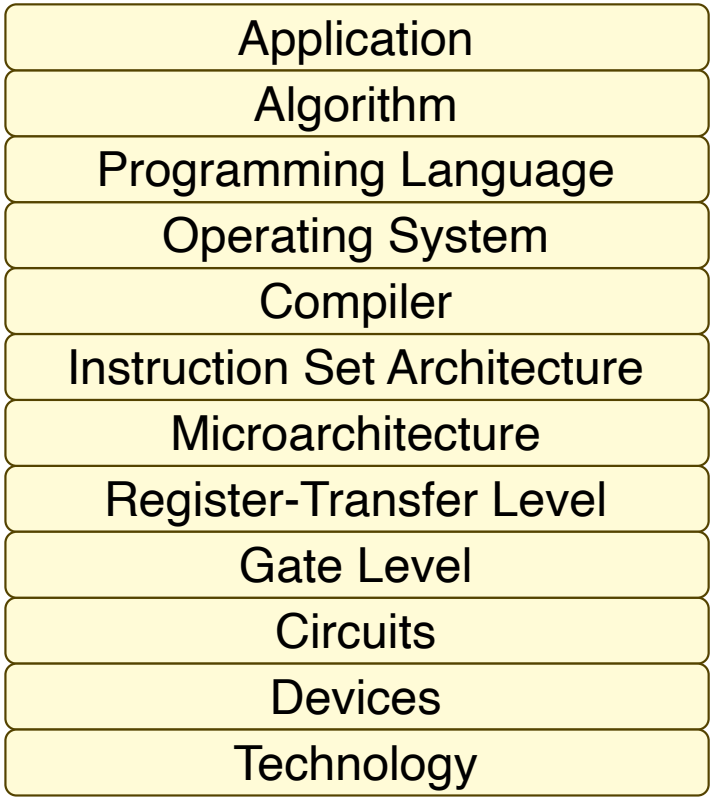


The Computer Systems Stack



In its broadest definition, computer engineering is the development of the abstraction layers that allow us to execute information processing applications efficiently using available manufacturing technologies

The Computer Systems Stack



Sort an array of numbers

2,6,3,8,4,5 -> 2,3,4,5,6,8

Out-of-place selection sort algorithm

1. Find minimum number in array
2. Move minimum number into output array
3. Repeat steps 1 and 2 until finished

C implementation of selection sort

```
void sort( int b[], int a[], int n ) {
  for ( int idx, k = 0; k < n; k++ ) {
    int min = 100;
    for ( int i = 0; i < n; i++ ) {
      if ( a[i] < min ) {
        min = a[i];
        idx = i;
      }
    }
    b[k]   = min;
    a[idx] = 100;
  }
}
```

The Computer Systems Stack

- Application
- Algorithm
- Programming Language
- Operating System
- Compiler
- Instruction Set Architecture
- Microarchitecture
- Register-Transfer Level
- Gate Level
- Circuits
- Devices
- Technology

Mac OS X, Windows, Linux

Handles low-level hardware management

C Compiler

Transform programs into assembly

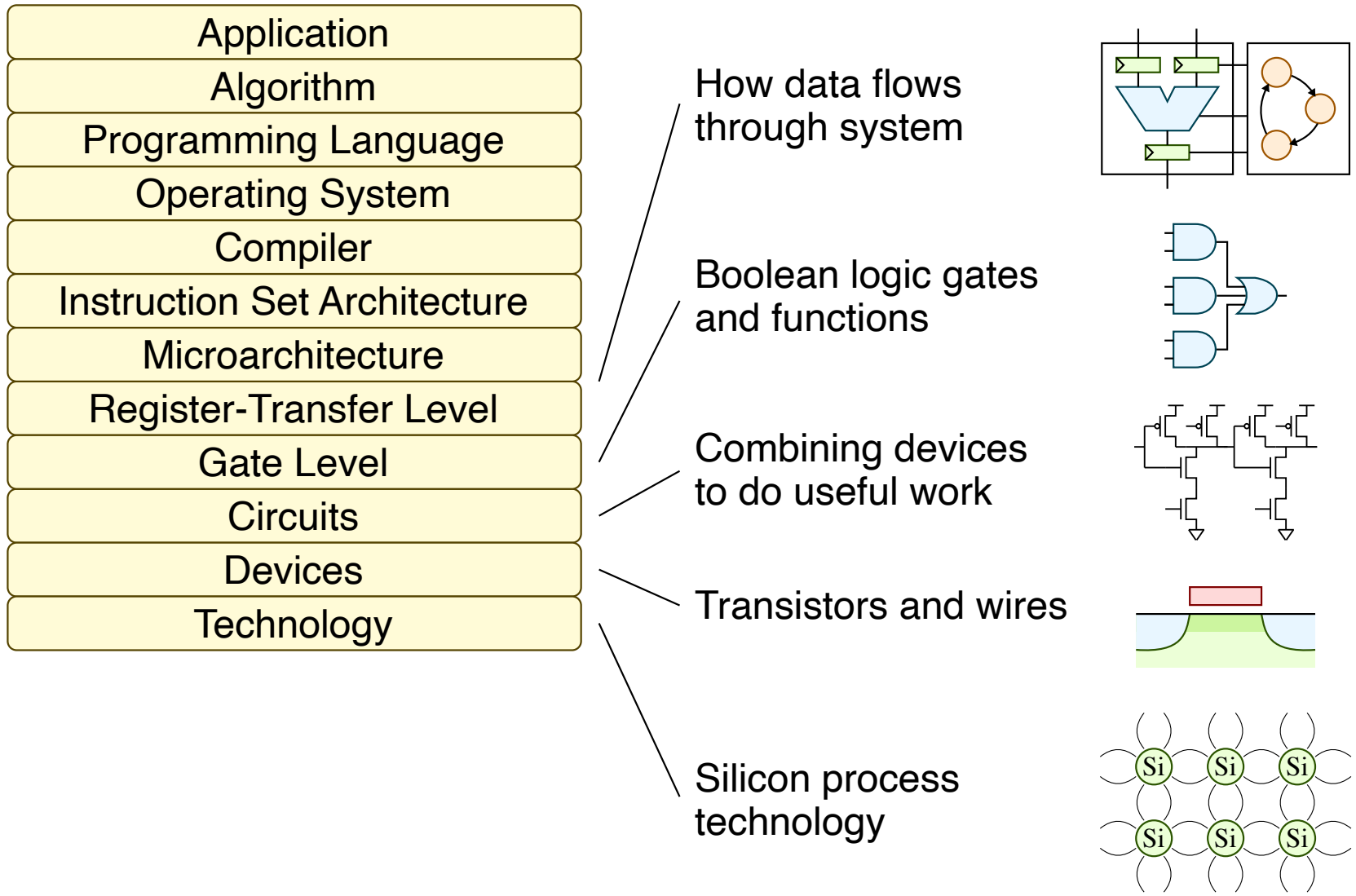
```
int a = b + c;          add $t0, $t1, $t2
A[i] = a;               sw  $t0, 0($t3)
```

RISC-V Instruction Set

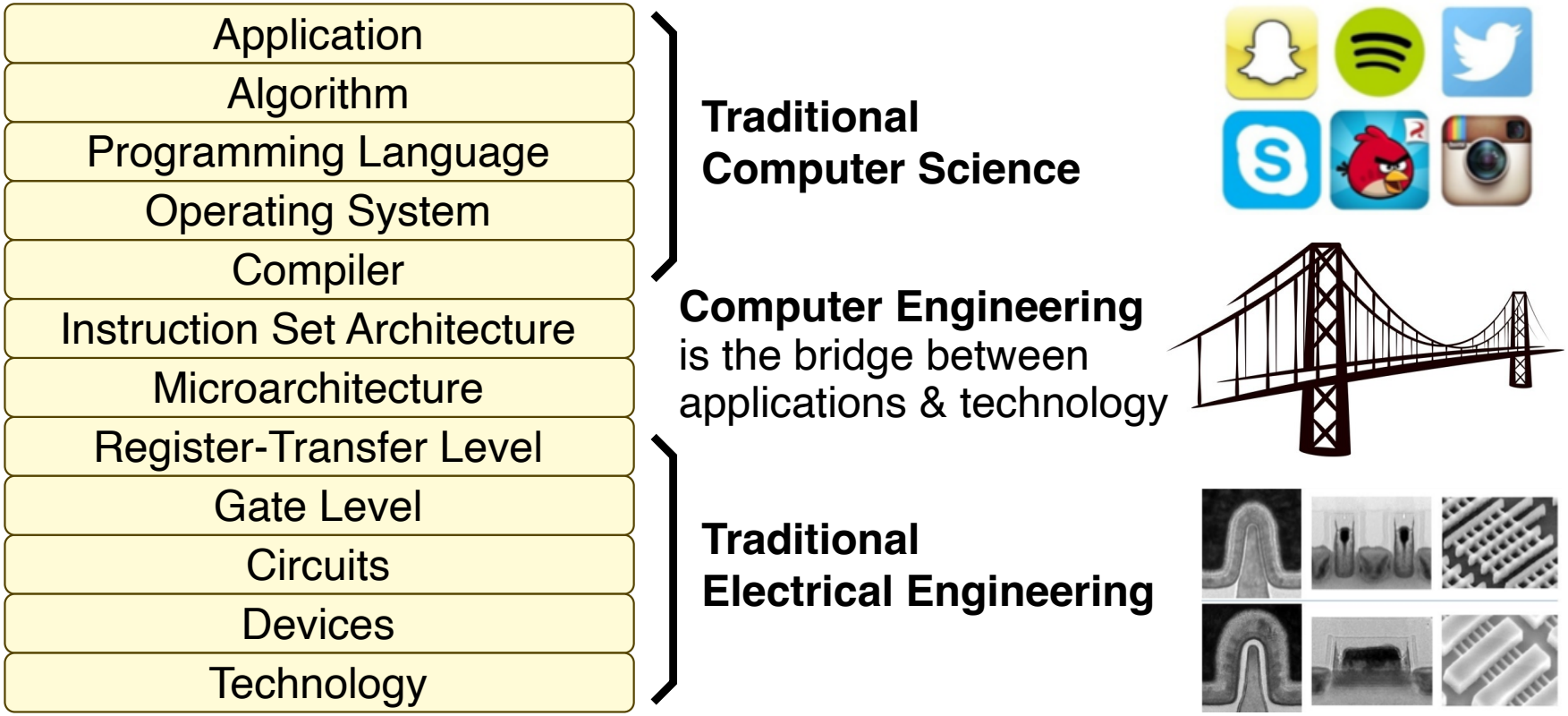
Instructions that machine executes

```
blez $a2, done
move $a7, $zero
li   $t4, 99
move $a4, $a1
li   $a3, 99
lw   $a5, 0($a4)
```

The Computer Systems Stack

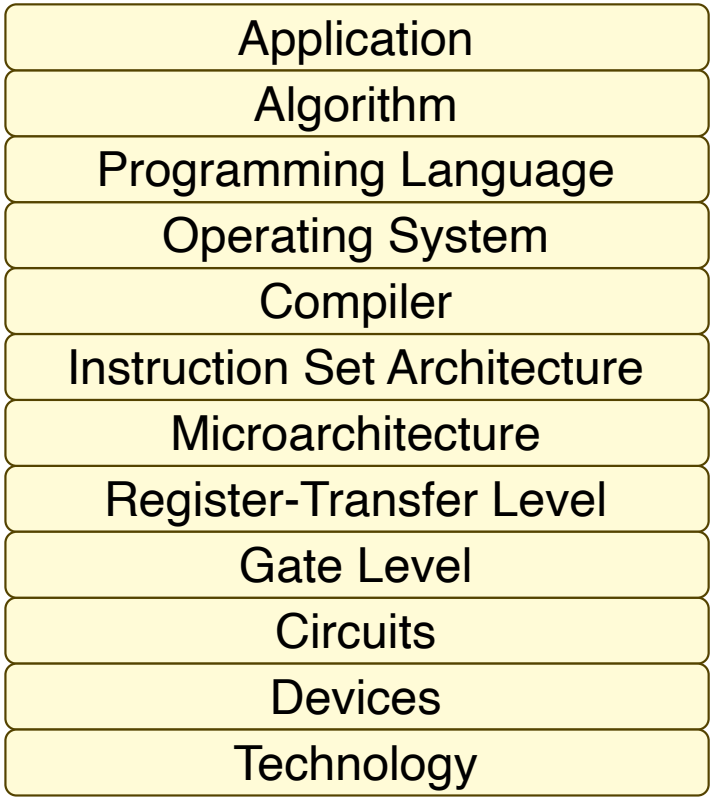


Abstraction in EE, CS, and CE



- **Modularity:** well defined behavior and interfaces simplify composition
- **Hierarchy:** recursively divide system into smaller and smaller modules
- **Regularity:** reuse common modules to reduce number of distinct modules

ECE 2300 in the Computer Systems Stack



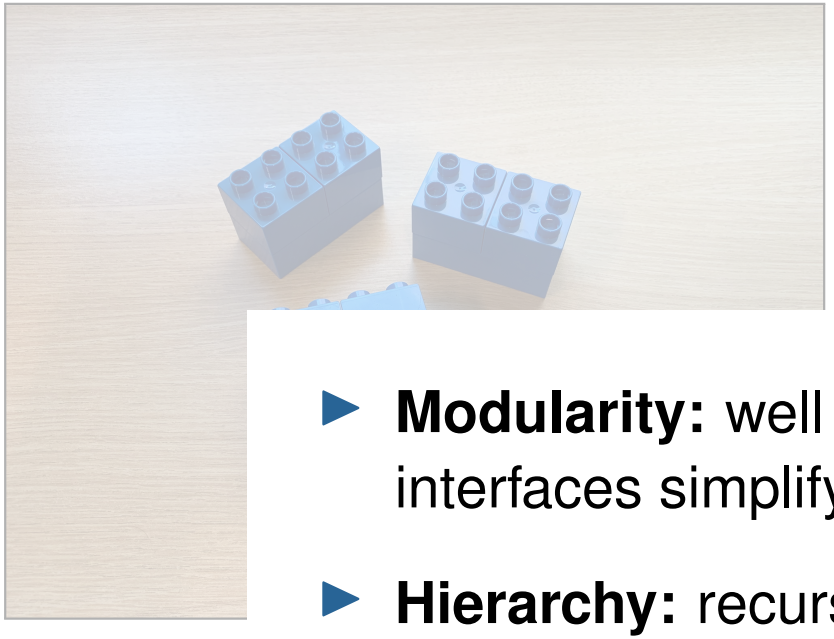
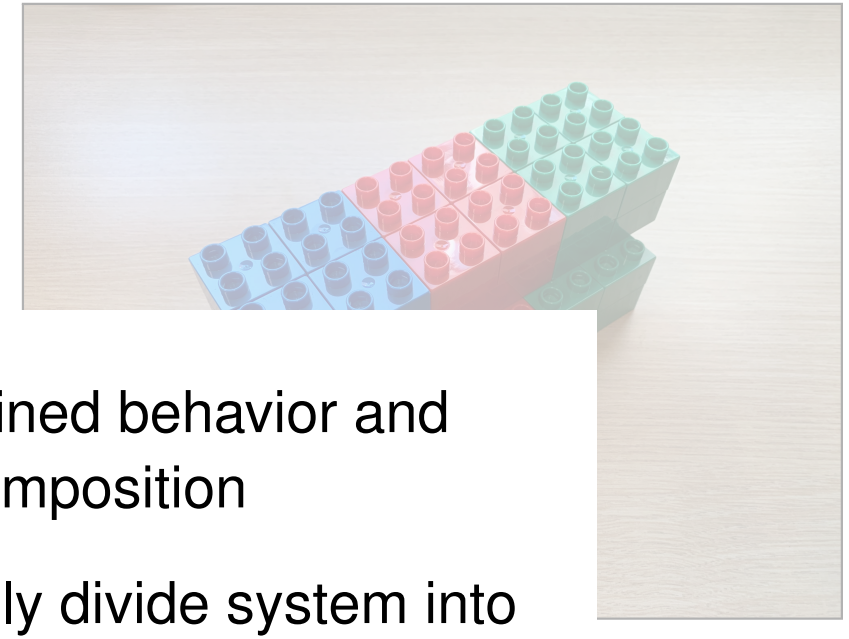
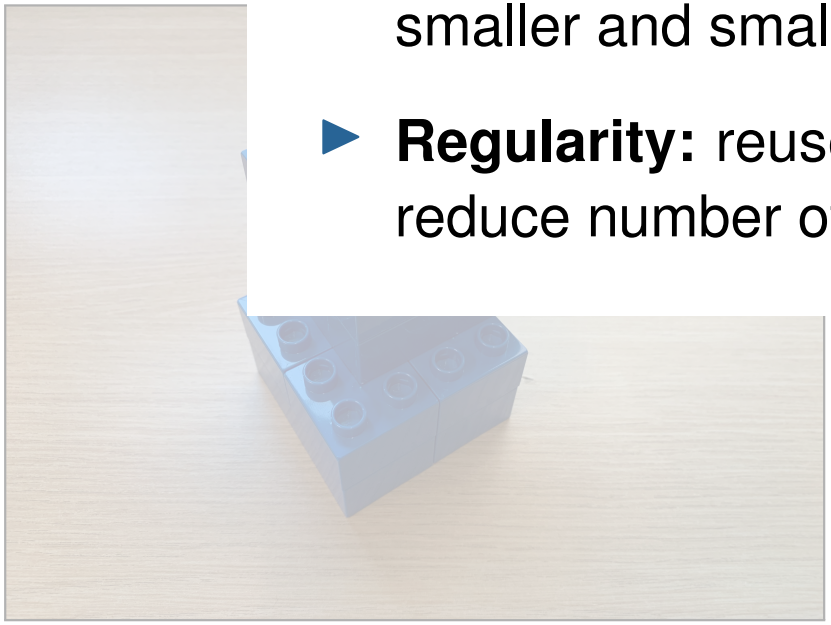
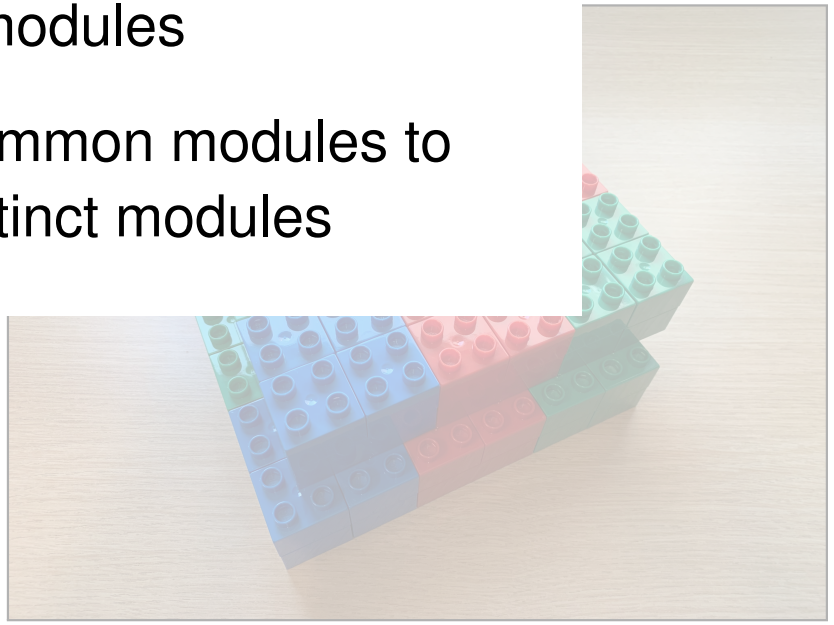
CS 4410 Operating Systems
CS 4420 Compilers
ECE 2400 Computer Systems Programming
ECE 3140 Embedded Systems
ECE 4760 Design with Microcontrollers
ECE 4750 Computer Architecture

ECE 2300 Digital Logic & Computer Org
ECE 4740 Digital VLSI Design

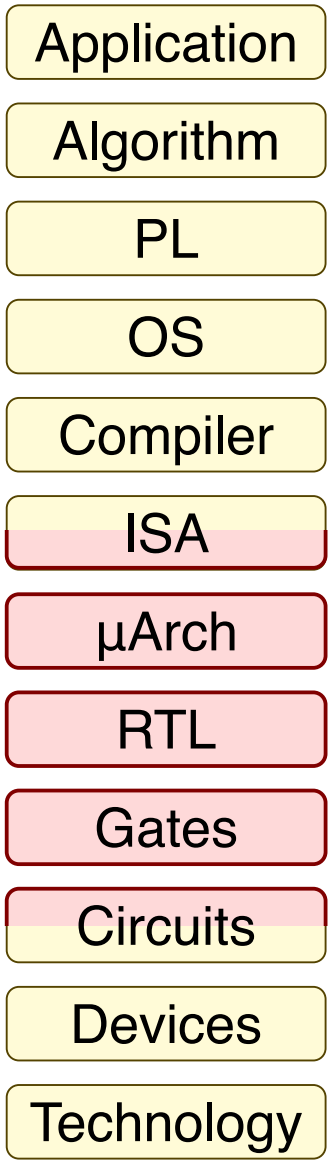
Related Graduate Courses

- ECE 5760 Advanced Microcontroller Design
- ECE 5750 Advanced Computer Architecture
- ECE 5745 Complex Digital ASIC Design
- ECE 5775 High-Level Design Automation

Activity #1: Exploring Abstraction

- **Modularity:** well defined behavior and interfaces simplify composition
- **Hierarchy:** recursively divide system into smaller and smaller modules
- **Regularity:** reuse common modules to reduce number of distinct modules



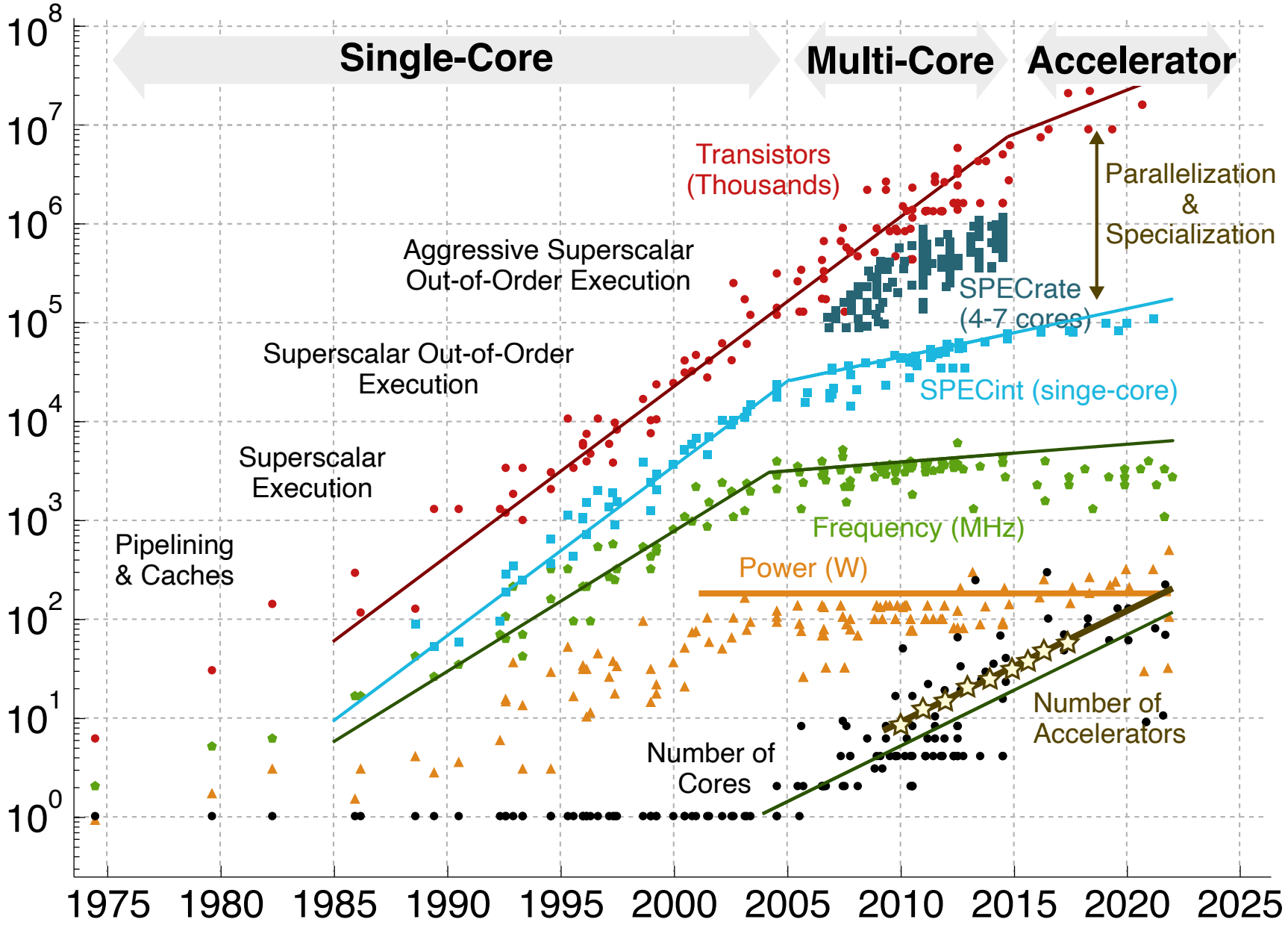
Agenda

What is Digital Logic and Computer Organization?

Trend 1: A New Era of Hardware Specialization

Trend 2: Hardware for AI and AI for Hardware

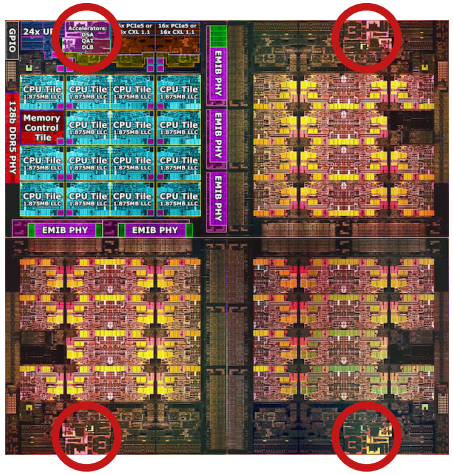
Course Logistics

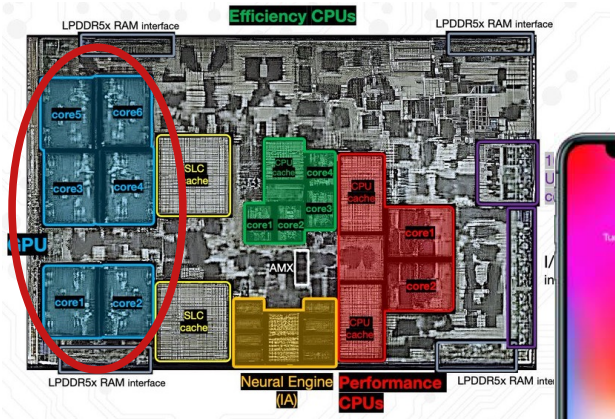


C. Batten, M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, K. Rupp & [Y. Shao, IEEE Micro'15] & [C. Leiserson, Science'20]

Specialization Across Cloud and Edge

Cloud Computing





Edge Computing

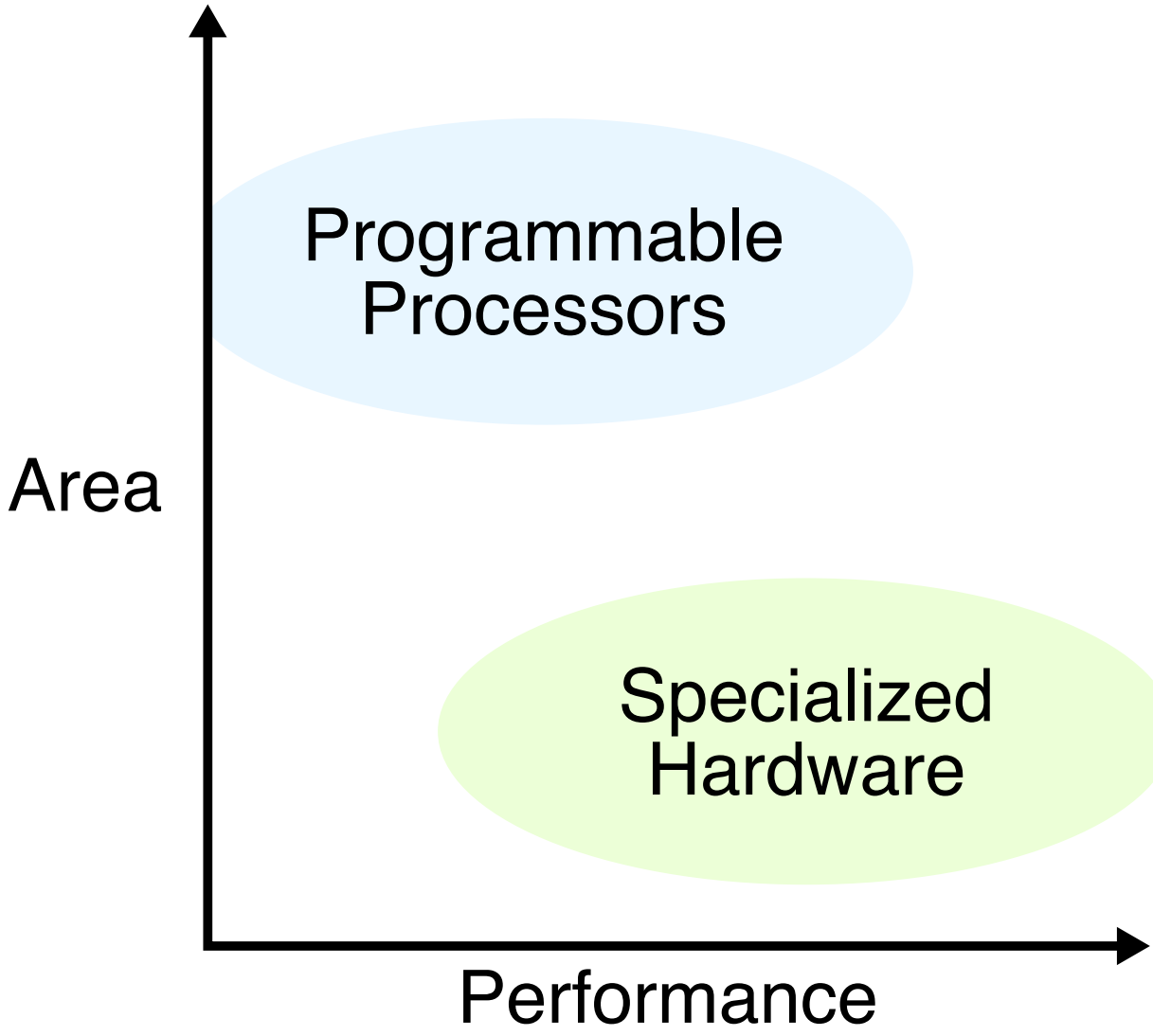
Intel Saphire Rapids

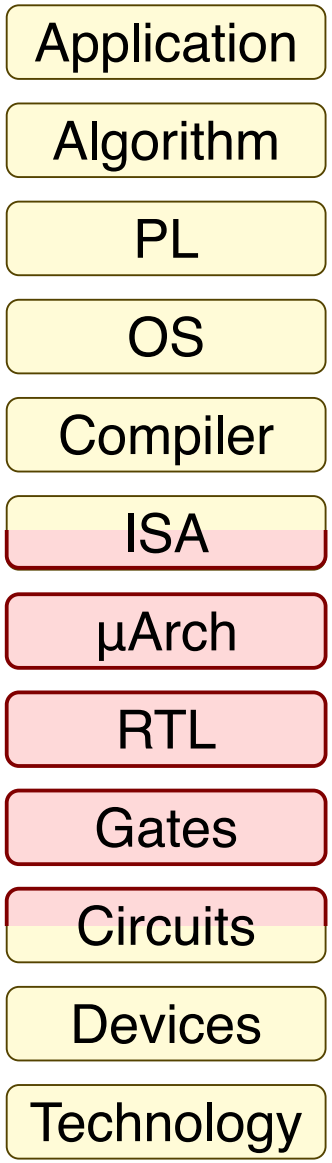
- Accelerators for cryptography
- Accelerators for compression
- Accelerators for security
- Accelerators for networking
- Accelerators for wireless communication
- Accelerators for storage

Apple A17 Pro

- Accelerators for video decoding
- Accelerators for raytracing

Activity #2: Specialization vs. Programmability





Agenda

What is Digital Logic and Computer Organization?

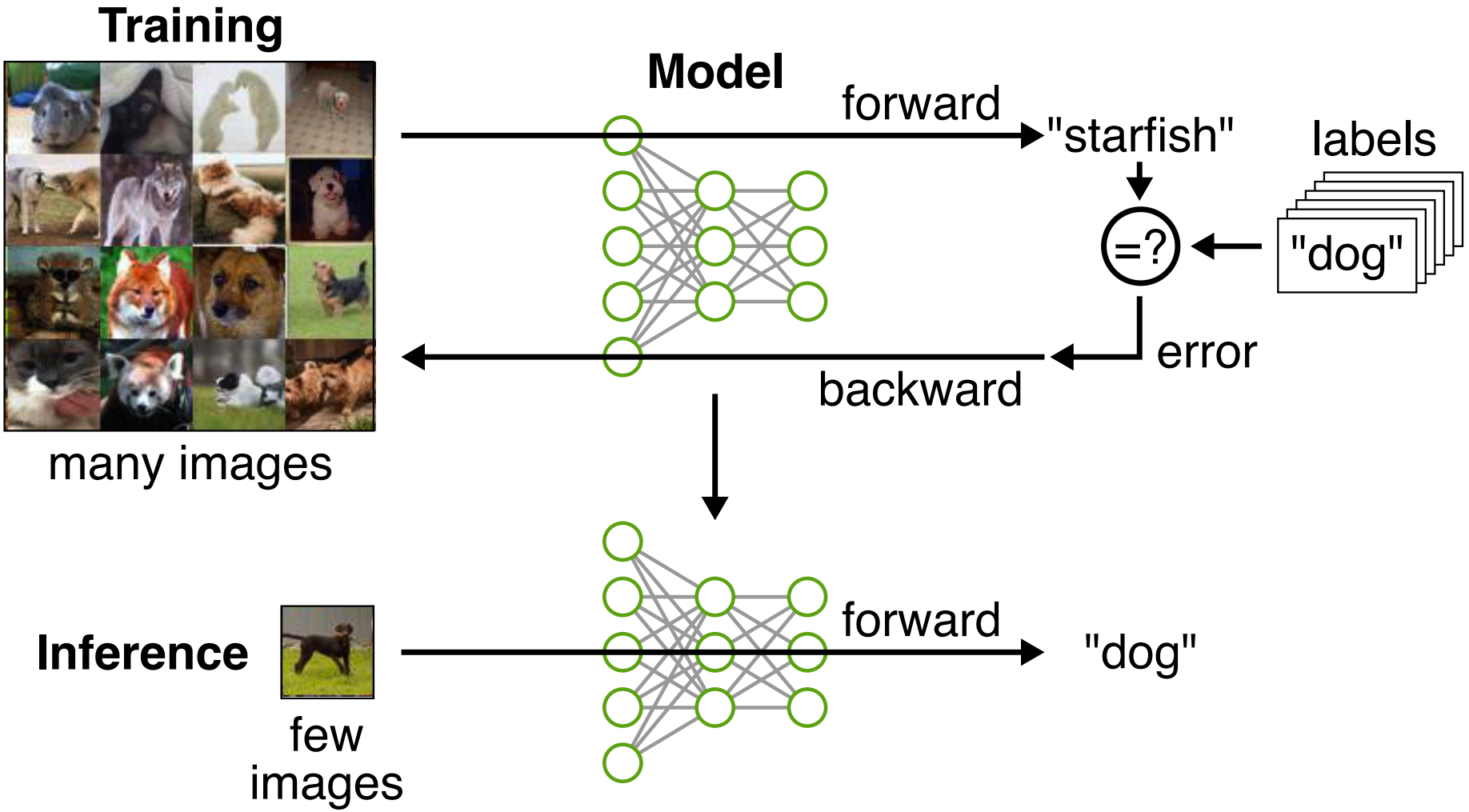
Trend 1: A New Era of Hardware Specialization

Trend 2: Hardware for AI and AI for Hardware

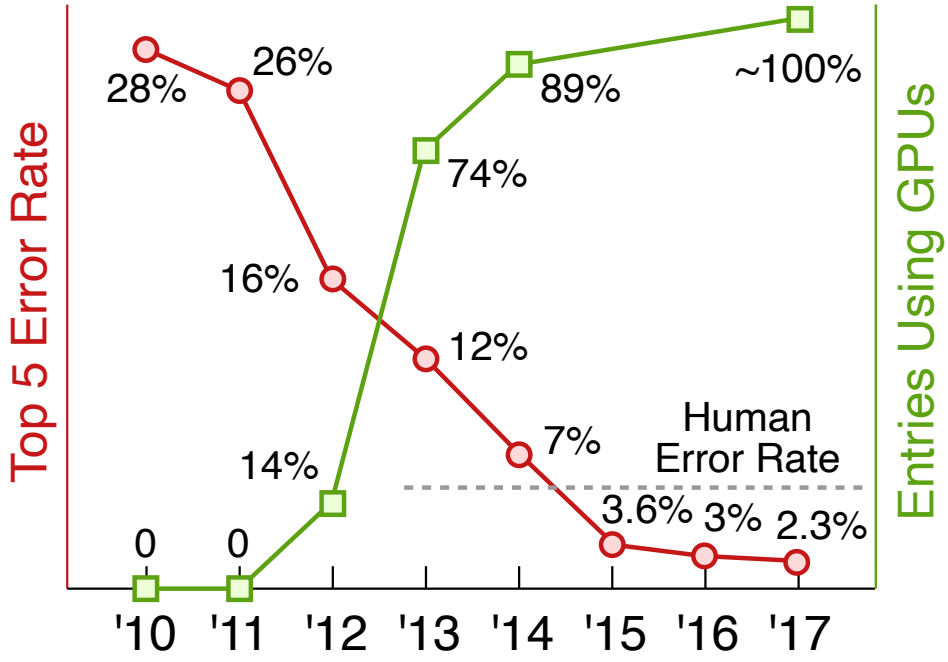
Course Logistics

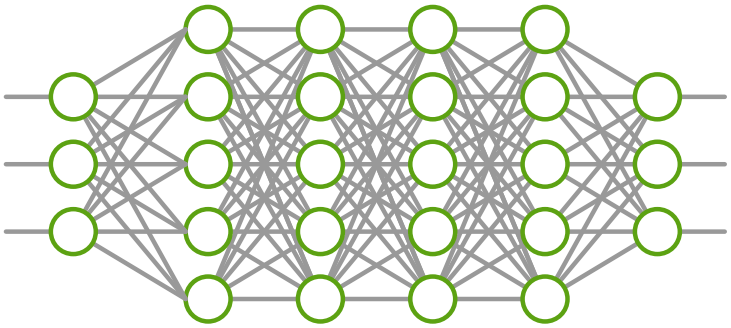
Image Recognition

Machine Learning (ML): Training vs. Inference



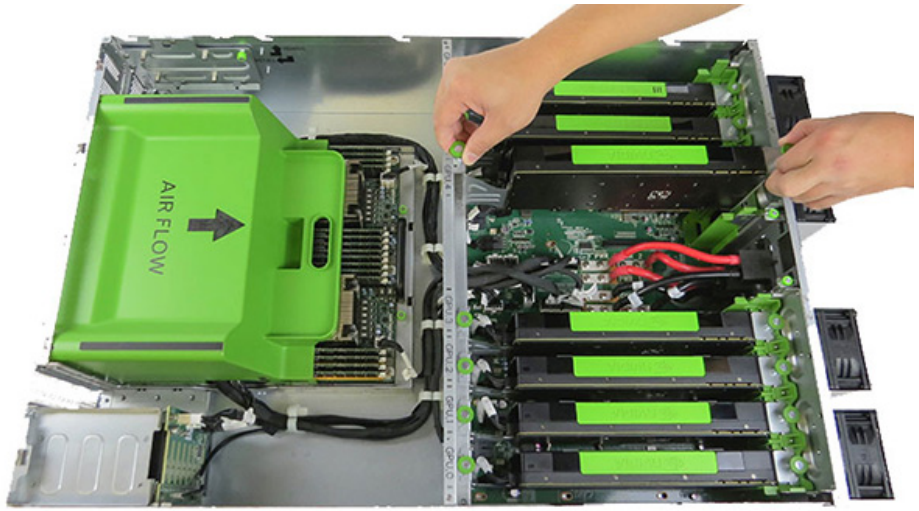
ImageNet Large-Scale Visual Recognition Challenge





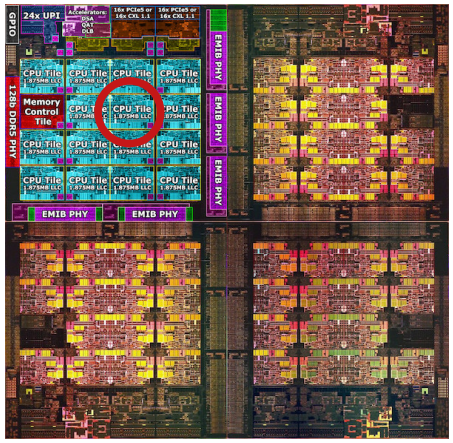
Software: Deep Neural Network

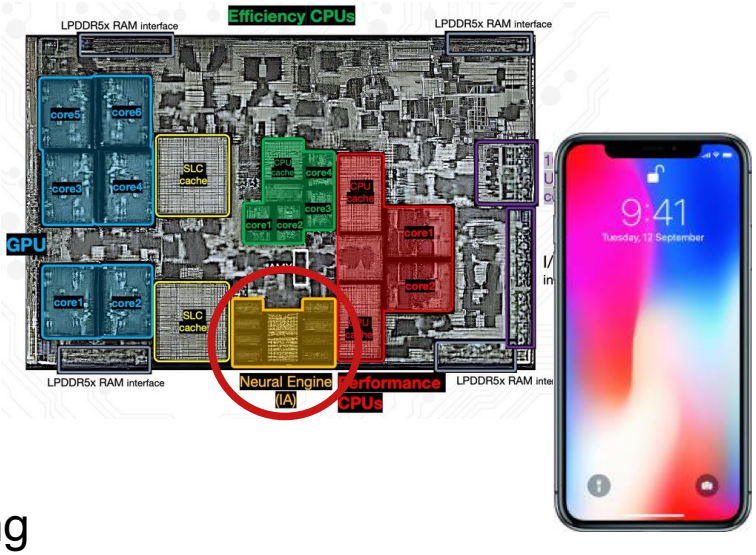
Hardware: Graphics Processing Units



Hardware for AI Across Cloud and Edge

Cloud Computing





Edge Computing

Intel Saphire Rapids

- Special instructions specifically designed for dense matrix multiplication in AI workloads

Apple A17 Pro

- Custom “Neural Engine” specifically designed to accelerate AI workloads

Top-five software companies are all building AI accelerators

- **Facebook:** MTIA
- **Amazon:** Echo, Inferentia, Trainium
- **Microsoft:** Maia
- **Google:** Edge and cloud TPUs
- **Apple:** SoCs for phones and laptops

Chip startup ecosystem for AI accelerators is thriving!

- **Graphcore**
- **Nervana**
- **Cerebras**
- **Wave Computing**
- **Horizon Robotics**
- **Cambricon**
- **DeePhi**
- **Esperanto**
- **SambaNova**
- **Eyeriss**
- **Tenstorrent**
- **Mythic**
- **ThinkForce**
- **Groq**
- **Lightmatter**

Rapidly Emerging Foundation Models

Can you explain what digital logic and computer organization are at a level suitable for a sophomore college student?

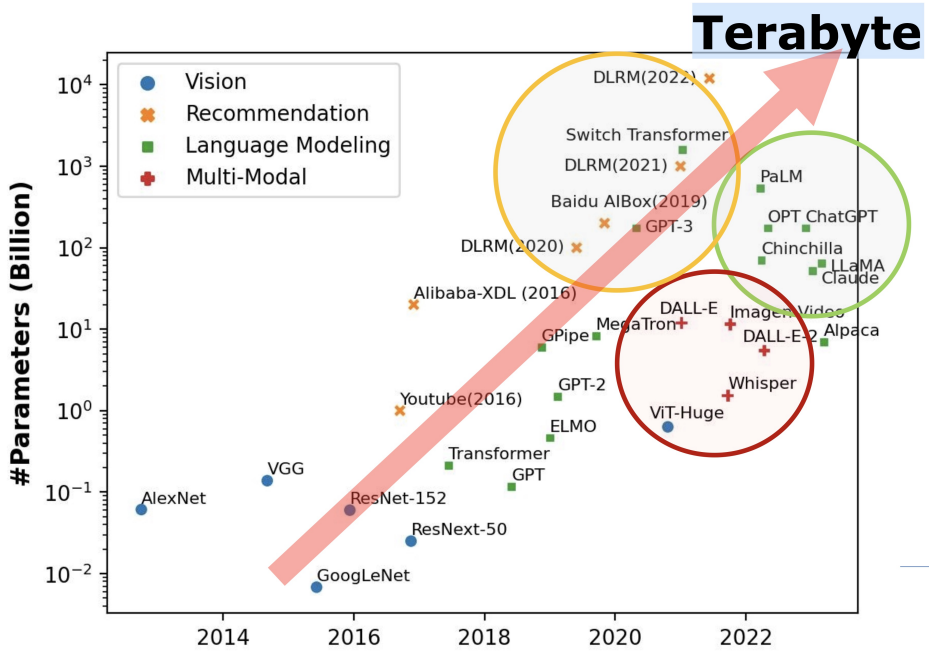
Certainly!

Digital Logic

Digital logic is the foundation of all digital systems, including computers. It involves the study of circuits that process binary information—essentially 1s and 0s. These binary values correspond to two states: 'on' and 'off,' which can represent anything from simple true/false conditions to complex data.

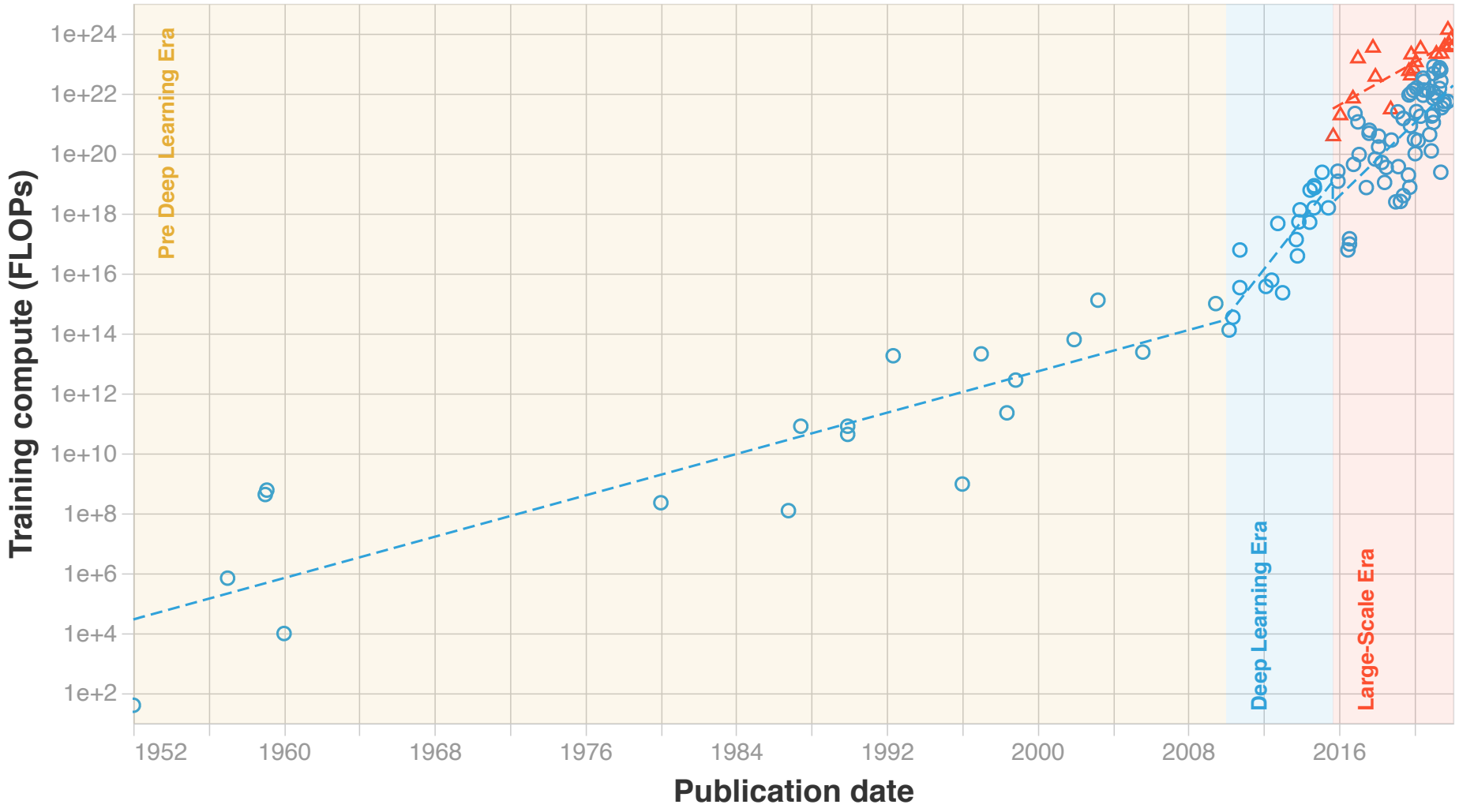
Key concepts in digital logic include:

1. **Boolean Algebra**: The mathematical framework for working with binary variables and logic gates. It involves operations like AND, OR, and NOT, which are the building blocks of digital circuits.
2. **Logic Gates**: The basic components that perform operations on one or more binary inputs to produce a binary output. The main types are:
   - **AND Gate**: Outputs 1 only if all inputs are 1.
   - **OR Gate**: Outputs 1 if at least one input is 1.
   - **NOT Gate**: Inverts the input (i.e., 1 becomes 0 and vice versa).
3. **Combinational Logic**: Circuits where the output is determined solely by the current inputs. Examples include adders, multiplexers, and decoders.
4. **Sequential Logic**: Circuits where the output depends on both current inputs and previous states. This introduces the concept of memory and timing. Examples include flip-flops, registers, and counters.



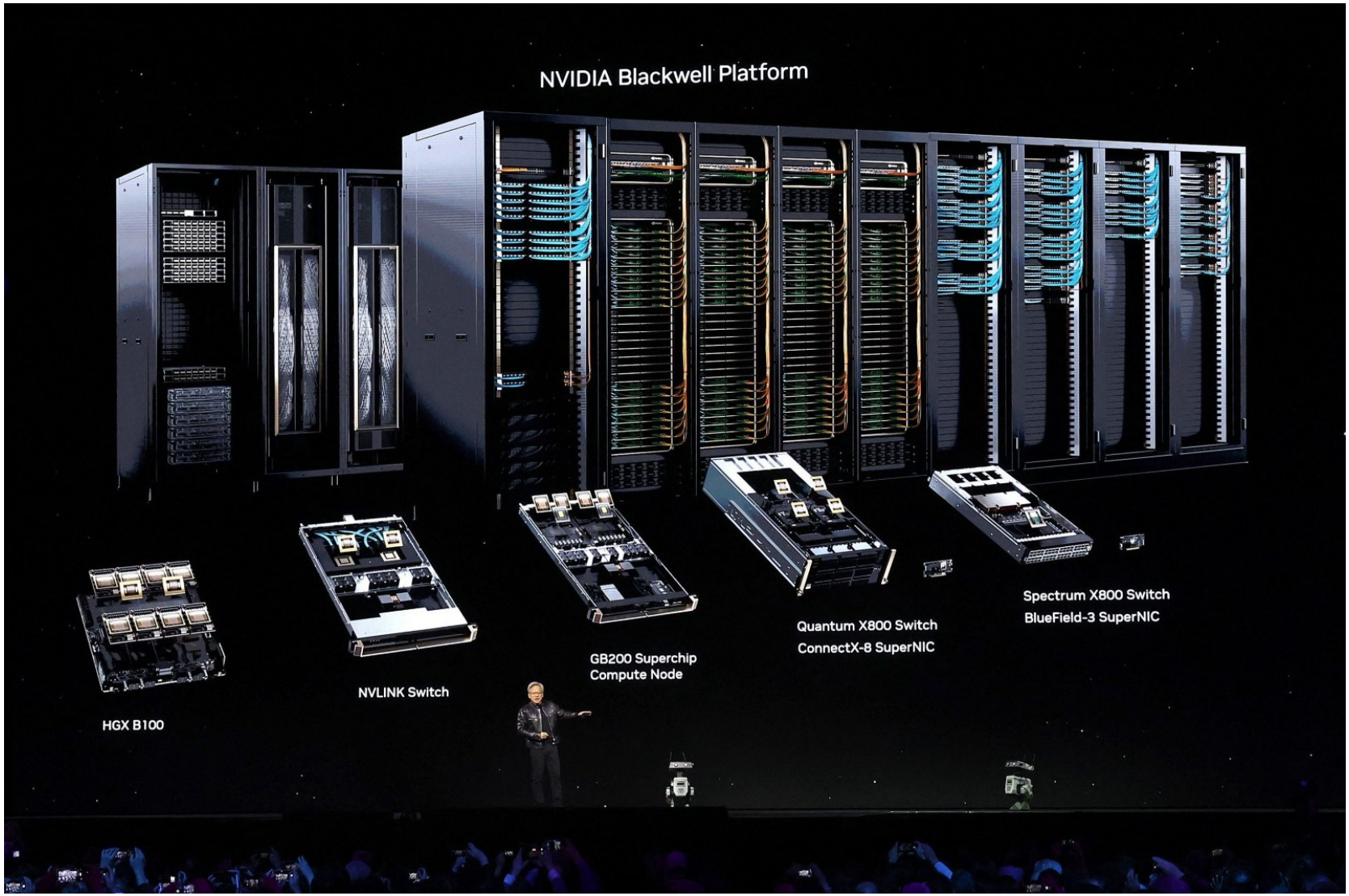
Foundation models now use 100s of billions of parameters placing tremendous demands on computing systems for inference

Compute Required for Training Foundation Models



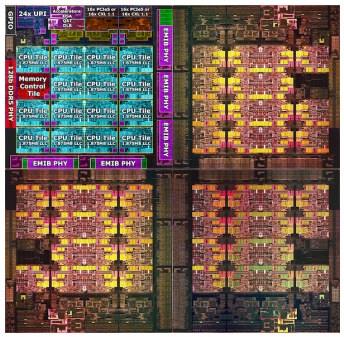
Adapted from [J. Sevilla et al., "Compute Trends Across Three Eras of ML," IJCNN'22.]

NVIDIA Blackwell AI Supercomputers



AI ↔ Hardware

Using AI to Accelerate Hardware Design



Designing Hardware to Accelerate AI

Please implement a one-bit full adder in Verilog.

Here's an implementation of a one-bit full adder in Verilog:

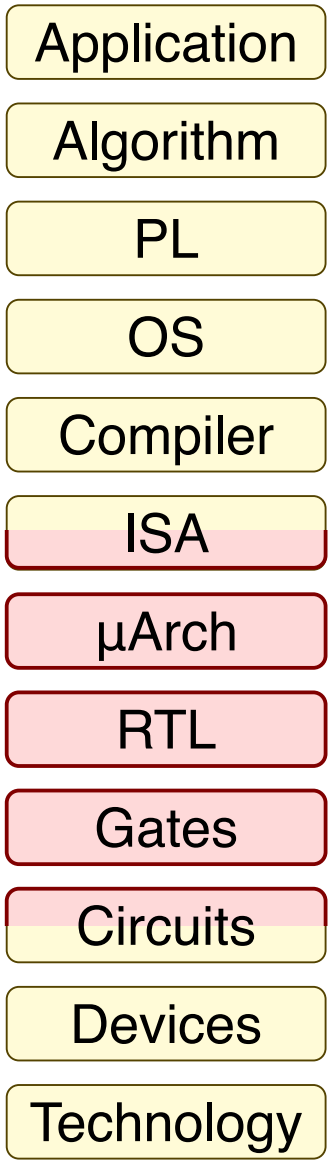
verilog — Copy code

```verilog
module full_adder (
    input A,       // Input bit A
    input B,       // Input bit B
    input Cin,     // Carry-in bit
    output Sum,    // Sum output
    output Cout    // Carry-out bit
);
    // Sum is calculated as A XOR B XOR Cin
    assign Sum = A ^ B ^ Cin;

    // Carry-out is calculated as (A AND B) OR (Cin AND (A XOR B))
    assign Cout = (A & B) | (Cin & (A ^ B));

endmodule
```

Explanation:

- **Sum:** The sum of the bits A, B, and the carry-in Cin is computed using the XOR operation.
- **Cout:** The carry-out is computed by summing the carry from the inputs A and B and the carry propagated by Cin.

This simple Verilog module can be instantiated in a larger design to perform multi-bit additions. If you need any more details or have additional questions, feel free to ask!



Agenda

What is Digital Logic and Computer Organization?

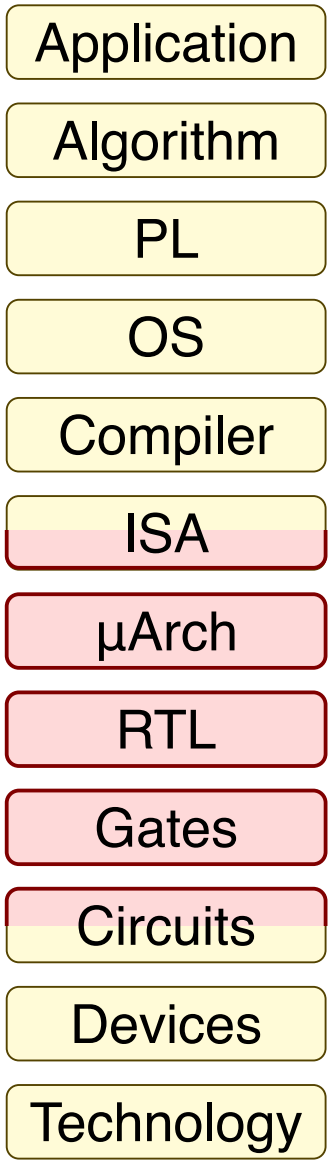
Trend 1: A New Era of Hardware Specialization

Trend 2: Hardware for AI and AI for Hardware

Course Logistics

Course Staff

- **Head TA:** Derin Ozturk
- **Graduate TAs:** Mahathi Andavolu, Vesal Bakhtazad
  Aidan McNay, Bolong Tan Klora Wang, Zichao Yue
- **Undergraduate TAs:** Mohammad Al-Labadi, Anjelica Bian
  Dyllan Hofflich, Zach Jessup Zarif Karim, Nita Kattimani
  Amy Le, Nicole Li, Kevin Rodriguez Paige Shelton, Max Trager
  Justin Wong, Steven Yu, Wei Zheng



Course Structure

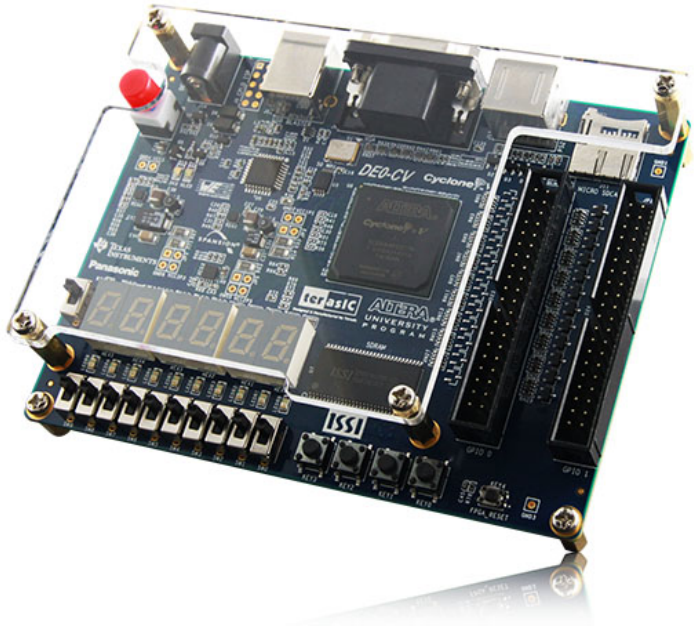
- **Part 1: Combinational Digital Logic**
  - transistors; logic gates; Boolean algebra; logic minization; decoders; multiplexors; arithmetic units
- **Part 2: Sequential Digital Logic**
  - latches and flip-flops; Moore and Mealy finite-state machines; counters; shift registers; memory arrays
- **Part 3: Computer Processor Organization**
  - instruction set architecture; arithmetic, memory, control instructions; single-cycle processor; FSM multi-cycle processor; pipelined processor
- **Part 4: Computer Memory Organization**
  - main memory; virtual memory; caches

Course Format and Procedures

- **Lectures:** students expected to attend all lectures; we will start promptly at 11:40am; use of cellular phones and laptops during lecutre is not allowed
- **Quizzes:** five minute pencil-and-paper quizzes at beginning of some lectures; no make-up quizzes; lowest quiz score is dropped; instructor goes through solution in lecture
- **Discussion Section:** three optional sections on Fridays in 225 Upson Hall; relatively informal focusing on faciliting students ability to complete lab assignments; students can sign up for a seat for each section on Canvas
- **Readings:** Harris and Harris, "Digital Design and Computer Architecture: RISC-V Edition," 2021; available through CAMP
- **Practice Problems:** no graded homework; practice problems distributed throughout the semester to help students put the concepts learned in lecture and reading into practice; solutions will not be provided; students can discuss their solutions with other students or instructors during office hours
- **Exams:** two prelims, final exam

Programming Assignments

- **Labs 1–3: Digital Logic**
  - Lab 1: 5-bit Seven-Segment Display
  - Lab 2: Addition/Multiplication Calculator
  - Lab 3: Music Player
- **Labs 4–5: Computer Organization**
  - Lab 4: Single-Cycle and FSM Processor
  - Lab 5: RISC-V Assembly Programming

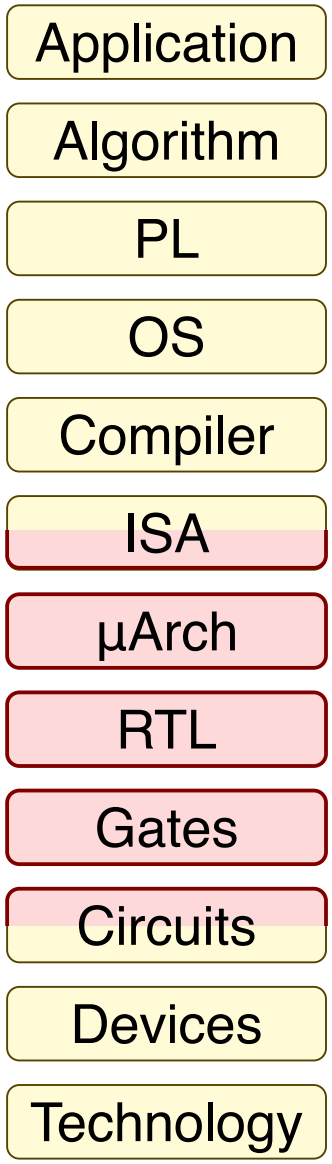


- **Labs involve two parts**
  - *Simulation part* involves students writing and testing their designs in Verilog using open-source simulators and is meant to be completed using the ecelinux servers (students still go to lab to work on Verilog design!)
  - *FPGA part* involves students synthesizing their designs using commercial tools and is meant to be completed during their assigned lab session

Course Schedule

			Monday	Tuesday	Wednesday	Thursday	Friday
11:00 AM	--	11:15 AM					
11:15 AM	--	11:30 AM					
11:30 AM	--	11:45 AM	**Lab**				
11:45 AM	--	12:00 PM	**Session 1**				
12:00 PM	--	12:15 PM	11:15-2:15pm	**Lecture**		**Lecture**	
12:15 PM	--	12:30 PM	238 Phillips	11:40-12:55pm		11:40-12:55pm	
12:30 PM	--	12:45 PM		155 Olin		155 Olin	
12:45 PM	--	1:00 PM					
1:00 PM	--	1:15 PM					
1:15 PM	--	1:30 PM					**Optional**
1:30 PM	--	1:45 PM					**Section 1**
1:45 PM	--	2:00 PM		**Lab**			1:15-2:15pm
2:00 PM	--	2:15 PM		**Session 3**			255 Upson
2:15 PM	--	2:30 PM		1:25-4:25pm			**Optional**
2:30 PM	--	2:45 PM		238 Phillips			**Section 2**
2:45 PM	--	3:00 PM					1:15-2:15pm
3:00 PM	--	3:15 PM					255 Upson
3:15 PM	--	3:30 PM					**Optional**
3:30 PM	--	3:45 PM					**Section 3**
3:45 PM	--	4:00 PM					1:15-2:15pm
4:00 PM	--	4:15 PM					255 Upson
4:15 PM	--	4:30 PM					
4:30 PM	--	4:45 PM					**Prof Batten**
4:45 PM	--	5:00 PM					**Office Hours**
5:00 PM	--	5:15 PM					4:30-5:30pm
5:15 PM	--	5:30 PM	**Office**	**Office**		**Office**	323 Rhodes
5:30 PM	--	5:45 PM	**Hours 1**	**Hours 2**	**Office**	**Hours 5**	
5:45 PM	--	6:00 PM	5:00-7:00pm	5:00-7:00pm	**Hours 4**	5:00-7:00pm	
6:00 PM	--	6:15 PM	203 Phillips	203 Phillips	5:30-7:30pm	203 Phillips	
6:15 PM	--	6:30 PM			307 Phillips		
6:30 PM	--	6:45 PM					
6:45 PM	--	7:00 PM					
7:00 PM	--	7:15 PM					
7:15 PM	--	7:30 PM					
7:30 PM	--	7:45 PM					
7:45 PM	--	8:00 PM	**Lab**	**Office**	**Lab**		
8:00 PM	--	8:15 PM	**Session 2**	**Hours 3**	**Session 4**		
8:15 PM	--	8:30 PM	7:30-10:30pm	7:30-9:30pm	7:30-10:30pm		
8:30 PM	--	8:45 PM	238 Phillips	203 Phillips	238 Phillips		
8:45 PM	--	9:00 PM		or			
9:00 PM	--	9:15 PM		307 Phillips			

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
26	27 Overview	28	29 **T01** Transistors	30 **S01** Linux Verilog	31	**Sep 1**
2 *Labor Day*	3 **T02** Logic Gates	4	5 **T03** Boolean Algebra	6 **S02** Verilog GL Design	7	8
9	10 **T03** Boolean Algebra	11 → **Lab 1.1** Display	12 **T04** Comb Blocks	13 **S03** Verilog Testing	14	15
16	17 **T05** Arithmetic Units	18 → **Lab 1.2** Display	19 **T05** Arithmetic Units	20 **S04** Lab 2 Head Start	21	22
23	24 **T06** Latches Flip-Flops	25 → **Lab 2.1** Calculator	26 **T06** Latches Flip-Flops	27 **S05** PBL	28	29
30	**Oct 1** **T07** FSMs	2 → **Lab 2.2** Calculator	3 **T07** FSMs	4 **S06** Verilog Latches/FFs	5	6
7 → **Prelim #1**	8 **T08** Sequential Blocks	9	10 **T09** Memory Arrays	11 *No Section*	12	13
14 *Fall Break*	15 *Fall Break*	16 → **Lab 3.1** Timer	17 **T10** ISA	18 **S07** Lab 4 Head Start	19	20
21 *Drop Deadline*	22 **T11** Single Cycle Proc	23 → **Lab 3.2** Timer	24 **T11** Single Cycle Proc	25 **S08** Prelim Review	26	27



Take-Away Points

- Digital logic transforms low-level circuits into hardware blocks dedicated to processing, storing, and moving digital data; and computer organization transforms these hardware blocks into programmable computing systems capable of executing high-level software.
- We are now in the accelerator era which requires carefully managing the tension between specialization and programmability; and we are seeing a sea change at the intersection of AI and hardware design.
- This course will serve as a foundation for more advanced courses in embedded systems and computer architecture so that students can eventually contribute to this new era!