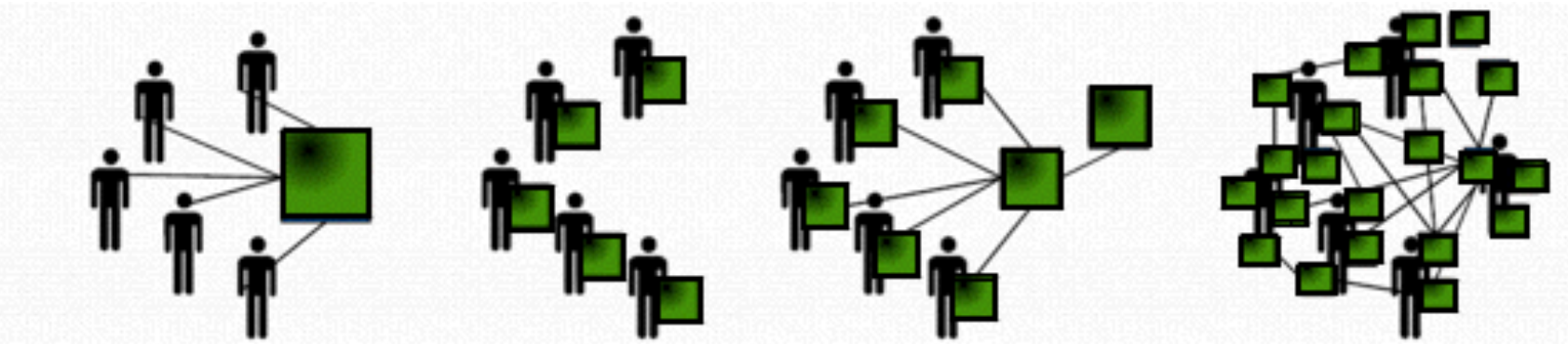
Disciplina
Sistemas de Computação

Aula 03



Hardware & Software: What is in-between?

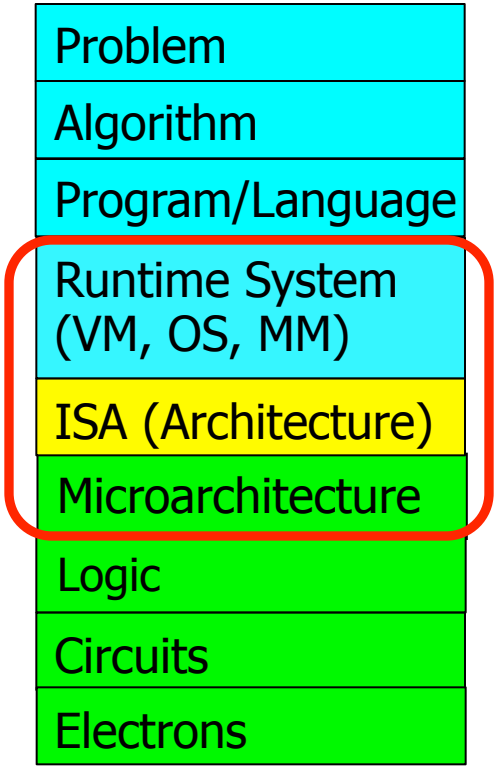
Programmer's view of a computer system works

- How does an assembly program end up executing as digital logic?
- What happens in-between?
- How is a computer designed using logic gates and wires to satisfy specific goals?

Architect/microarchitect's view: How to design a computer that meets system design goals. Choices critically affect both the SW programmer and the HW designer

HW designer's view of a computer system works

Levels of Transformations



The Power of Abstraction

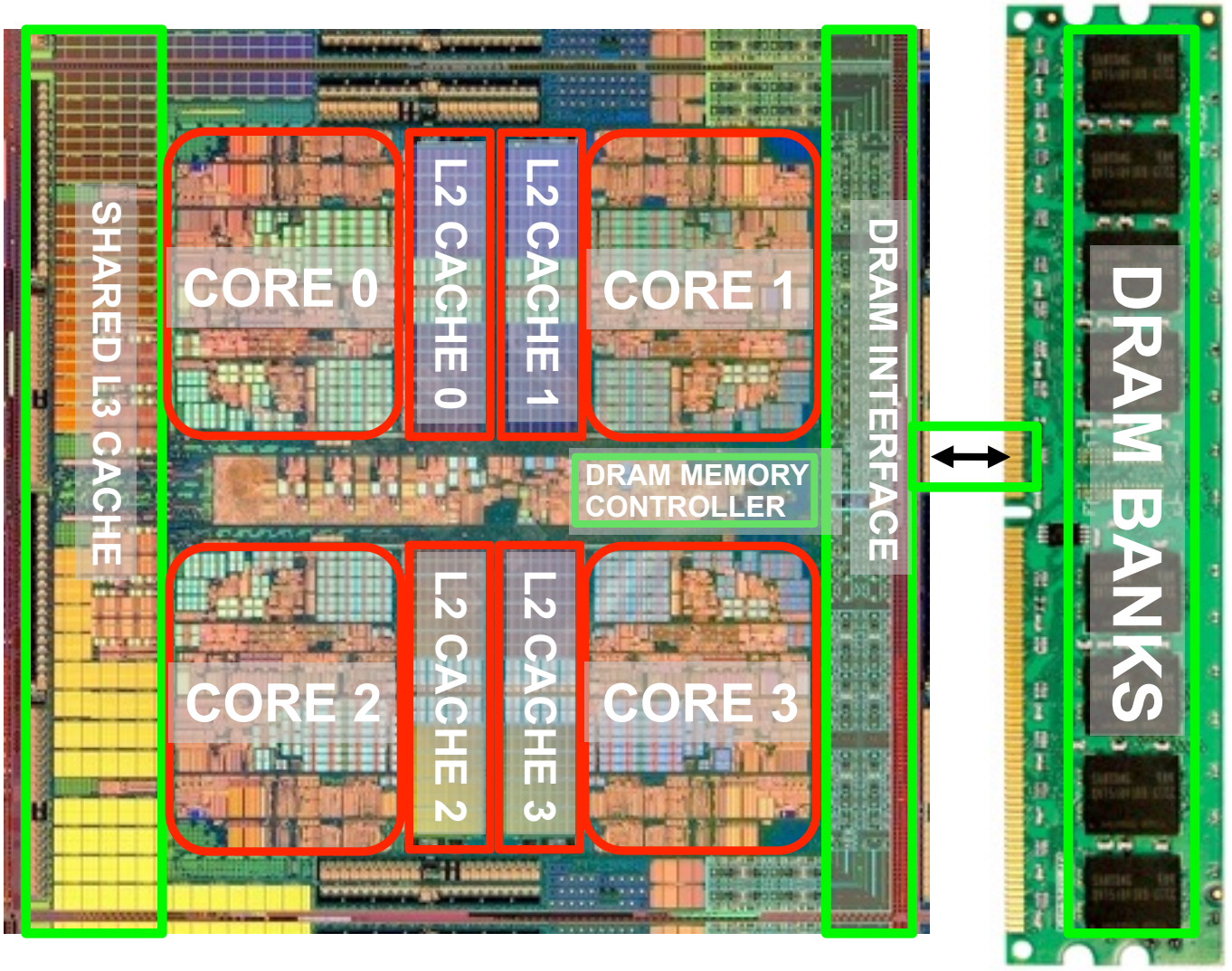
- Levels of transformation create abstractions
  - Abstraction: A higher level only needs to know about the interface to the lower level, not how the lower level is implemented
  - E.g., high-level language programmer does not really need to know what the ISA is and how a computer executes instructions
- Abstraction improves productivity
  - No need to worry about decisions made in underlying levels
  - E.g., programming in Java vs. C vs. assembly vs. binary vs. by specifying control signals of each transistor every cycle
- Then, why would you want to know what goes on underneath or above?

Crossing the Abstraction Layers

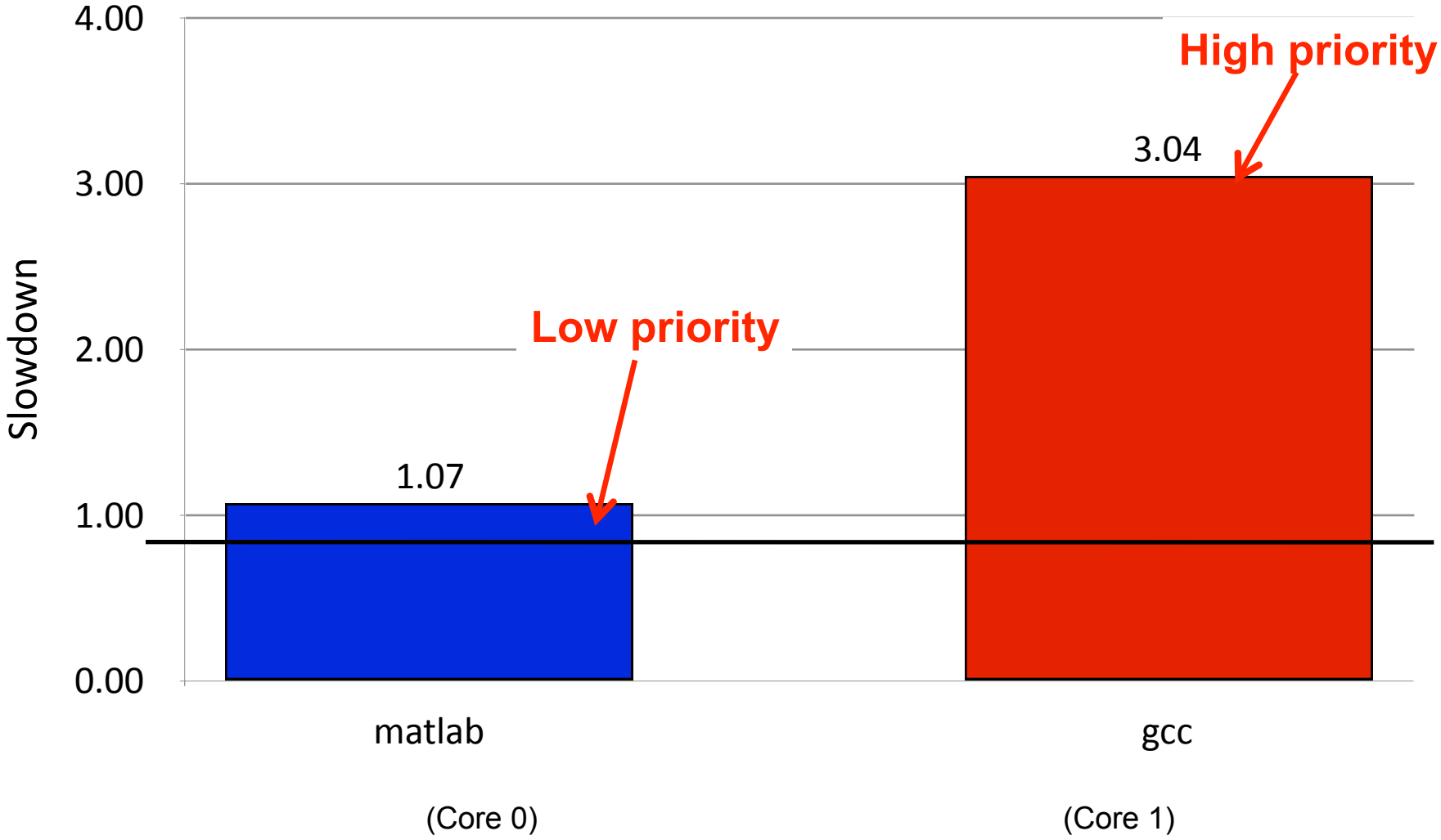
- As long as everything goes well, not knowing what happens in the underlying level (or above) is not a problem.
- What if
  - The program you wrote is running slow?
  - The program you wrote does not run correctly?
  - The program you wrote consumes too much energy?
- What if
  - The hardware you designed is too hard to program?
  - The hardware you designed is too slow because it does not provide the right primitives to the software?
- To all understand all of those What if's, it is important to understand how a processor works underneath the software layer and how decisions made in hardware affect the software/programmer

An Example: Multi-Core Systems

Multi-Core Chip



Unexpected Slowdowns in Multi-Core



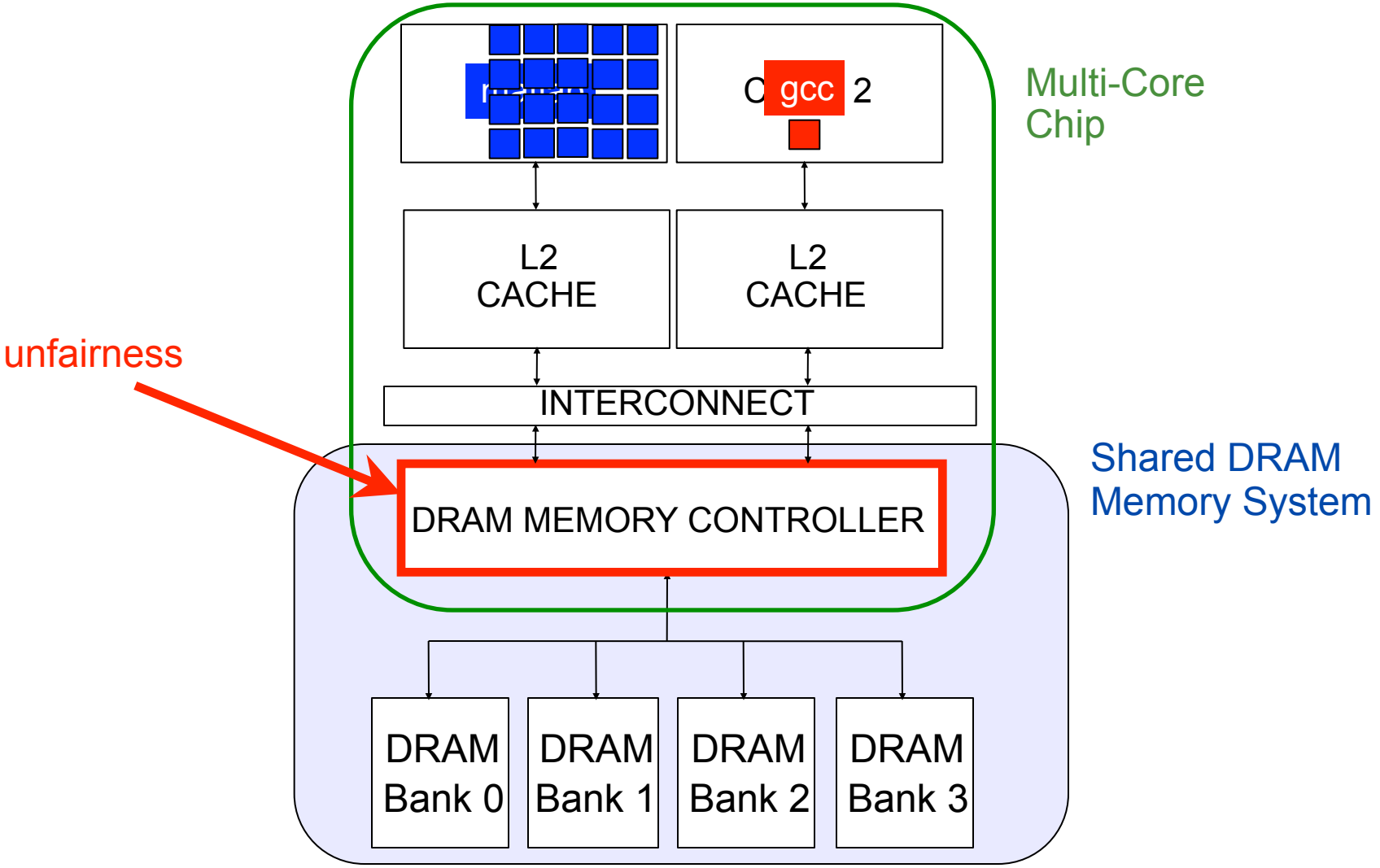
Moscibroda and Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," USENIX Security 2007.

A Question or Two

- Can you figure out why there is a disparity in slowdowns if you do not know how the processor executes the programs?

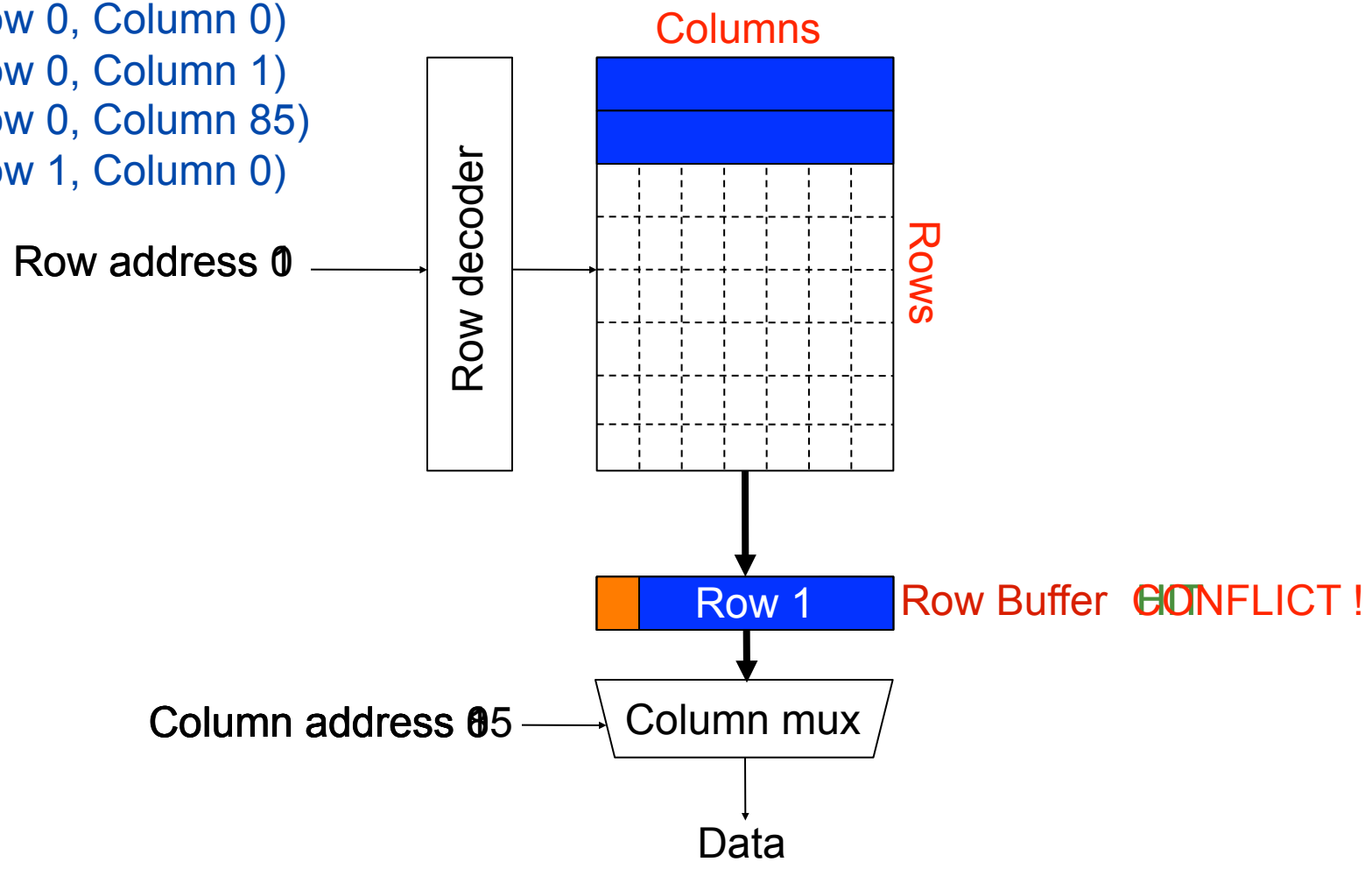
- Can you fix the problem without knowing what is happening “underneath”?

Why the Disparity in Slowdowns?



DRAM Bank Operation

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



DRAM Controllers

- A row-conflict memory access takes significantly longer than a row-hit access
- Current controllers take advantage of the row buffer
- Commonly used scheduling policy (FR-FCFS)

  (1) Row-hit first: Service row-hit memory accesses first

  (2) Oldest-first: Then service older accesses first
- This scheduling policy aims to maximize DRAM throughput

The Problem

- Multiple threads share the DRAM controller
- DRAM controllers designed to maximize DRAM throughput

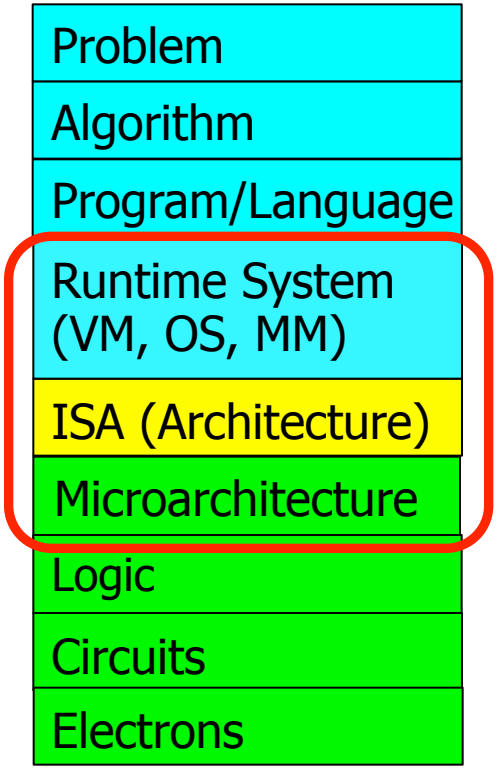
- DRAM scheduling policies are thread-unfair
  - Row-hit first: unfairly prioritizes threads with high row buffer locality
    - Threads that keep on accessing the same row
  - Oldest-first: unfairly prioritizes memory-intensive threads

- DRAM controller vulnerable to denial of service attacks

Now That We Know What Happens Underneath

- How would you solve the problem?

- What is the right place to solve the problem?
  - Programmer?
  - Compiler?
  - Hardware (Memory controller)?
  - Hardware (DRAM)?
  - Circuits?



What is Computer Architecture?

- The science and art of designing, selecting, and interconnecting hardware components and designing the hardware/software interface to create a computing system that meets functional, performance, energy consumption, cost, and other specific goals.

Why Study Computer Architecture?

- Enable better systems: make computers faster, cheaper, smaller, more reliable, …
  - By exploiting advances and changes in underlying technology/circuits
- Enable new applications
  - Life-like 3D visualization 20 years ago?
  - Virtual reality?
  - Personal genomics?
- Enable better solutions to problems
  - Software innovation is built into trends and changes in computer architecture

Computer Architecture Today

- Industry is in a large paradigm shift (to multi-core)
- Many problems motivating and caused by the shift
  - Power/energy constraints
  - Complexity of design → multi-core
  - Technology scaling → new technologies
  - Memory wall/gap
  - Reliability wall/issues
  - Programmability wall/problem
- You can revolutionize the way computers are built, if you understand both the hardware and the software (and change each accordingly)

Aviso

- Leiam:

  Moscibroda and Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," USENIX Security 2007.

- Nosso priemiro PrS será baseado nele