15-740/18-740
Computer Architecture
Lecture 1: Intro, Principles, Tradeoffs

Prof. Onur Mutlu
Carnegie Mellon University

Agenda

- Announcements
- Homework and reading for next time
- Projects
- Some fundamental concepts
  - Computer architecture
  - Levels of transformation
  - ISA vs. microarchitecture
  - Design point
  - Tradeoffs
    - ISA, microarchitecture, system/task
  - Von Neumann model
  - Performance equation and Amdahl's Law

Last Time …

- Course logistics, info, requirements
  - See slides for Lecture 0 and syllabus online

- Homework 0

- Readings for first week
  - G. M. Amdahl "Validity of the single processor approach to achieving large scale computing capabilities," AFIPS Conference, April 1967.
  - G. E. Moore, "Cramming more components onto integrated circuits," Electronics, April 1965.
  - Ronen et al., "Coming Challenges in Microarchitecture and Architecture," Proceedings of the IEEE, vol. 89, no. 11, 2001.
  - Y. N. Patt, "Requirements, bottlenecks, and good fortune: agents for microprocessor evolution," Proceedings of the IEEE, vol. 89, no. 11, 2001.

Teaching Assistants and Emails

- Teaching Assistants
  - Vivek Seshadri
    - GHC 7517
    - vseshadr@cs.cmu.edu
  - Lavanya Subramanian
    - HH 2nd floor
    - lsubrama@andrew.cmu.edu
  - Evangelos Vlachos
    - HH A312
    - evlachos@ece.cmu.edu

- 740-official@ece.cmu.edu
  - Email for me and the TAs

Summary

- Homework 0 – Part 1
  - Due Today

- Homework 0 – Part 2
  - Due September 10 (Fri), 11:59pm

- First readings
  - Reviews due September 10, 11:59pm

- Project ideas and groups
  - Read, think, and brainstorm
  - Project statement document online
  - Sample project topics document online
  - Proposal due September 27

Research Project

- Your chance to explore in depth a computer architecture topic that interests you
- Your chance to publish your innovation in a top computer architecture/systems conference.

- Start thinking about your project topic from now!
- Interact with me and Evangelos, Lavanya, Vivek

- Groups of 3
- Proposal due: Sep 27

- https://www.ece.cmu.edu/~ece740/wiki/lib/exe/fetch.php?media=projects.pdf
- https://www.ece.cmu.edu/~ece740/wiki/lib/exe/fetch.php?media=project-topics.doc

Readings Referenced Today

- On-chip networks
  - Dally and Towles, “Route Packets, Not Wires: On-Chip Interconnection Networks,” DAC 2001.
  - Wentzlaff et al., “On-Chip Interconnection Architecture of the Tile Processor,” IEEE Micro 2007.
  - Grot et al., “Preemptive Virtual Clock: A Flexible, Efficient, and Cost-effective QOS Scheme for Networks-on-Chip,” MICRO 2009.
- Main memory controllers
  - Moscibroda and Mutlu, “Memory performance attacks: Denial of memory service in multi-core systems,” USENIX Security 2007.
  - Rixner et al., “Memory Access Scheduling,” ISCA 2000.
- Architecture reference manuals
  - Digital Equipment Corp., “VAX11 780 Architecture Handbook,” 1977-78.
  - Intel Corp. “Intel 64 and IA-32 Architectures Software Developer’s Manual”
- ISA and Compilers
  - Colwell et al., “Instruction Sets and Beyond: Computers, Complexity, and Controversy,” IEEE Computer 1985.
  - Wulf, “Compilers and Computer Architecture,” IEEE Computer 1981.

Papers for Review

- Colwell et al., “Instruction Sets and Beyond: Computers, Complexity, and Controversy,” IEEE Computer 1985.
- Due September 17

Comp Arch @ Carnegie Mellon

- Computer Architecture Lab at Carnegie Mellon (CALCM) @ www.ece.cmu.edu/CALCM
- Send mail to calcm-list-request@ece
  - body: subscribe calcm-list

- Seminars
  - CALCM weekly seminar
  - SDI weekly seminar

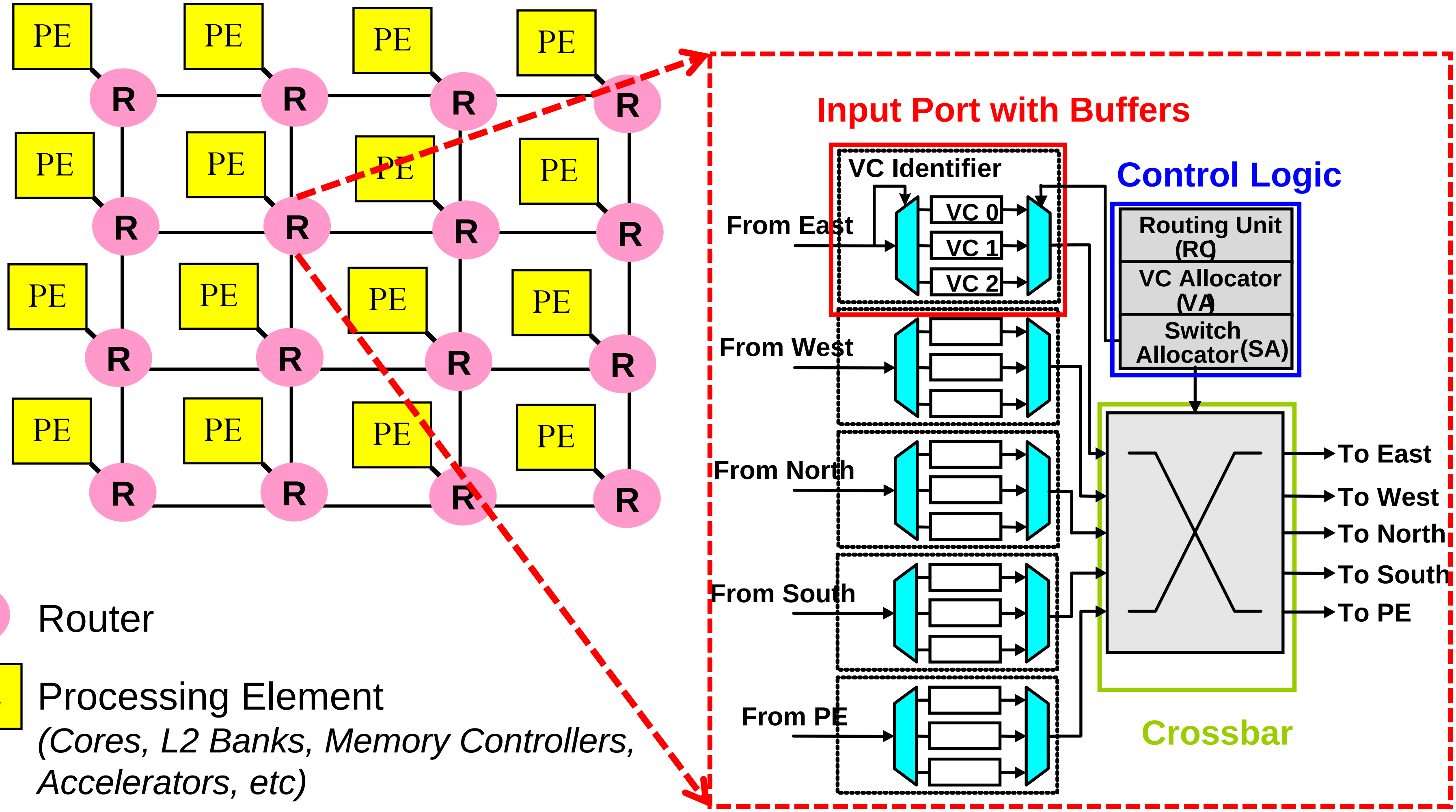
CALCM Seminar Tomorrow

- “Service Guarantees in Networks-on-a-Chip”
- Boris Grot, UT-Austin
- 1-2 pm, September 9, Thursday
- HH-D210

- Attend and optionally provide a review online

On-Chip Network Based Multi-Core Systems

- A scalable multi-core is a distributed system on a chip



R Router

PE Processing Element
(Cores, L2 Banks, Memory Controllers, Accelerators, etc)

Idea of On-Chip Networks

- Problem: Connecting many cores with a single bus is not scalable
  - Single point of connection limits communication bandwidth
    - What if multiple core pairs want to communicate with each other at the same time?
  - Electrical loading on the single bus limits bus frequency

- Idea: Use a network to connect cores
  - Connect neighboring cores via short links
  - Communicate between cores by routing packets over the network

- Dally and Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," DAC 2001.

Advantages/Disadvantages of NoCs

- Advantages compared to bus
  + More links → more bandwidth → multiple core-to-core transactions can occur in parallel in the system (no single point of contention) → higher performance

  + Links are short and less loaded → high frequency

  + More scalable system → more components/cores can be supported on the network than on a single bus

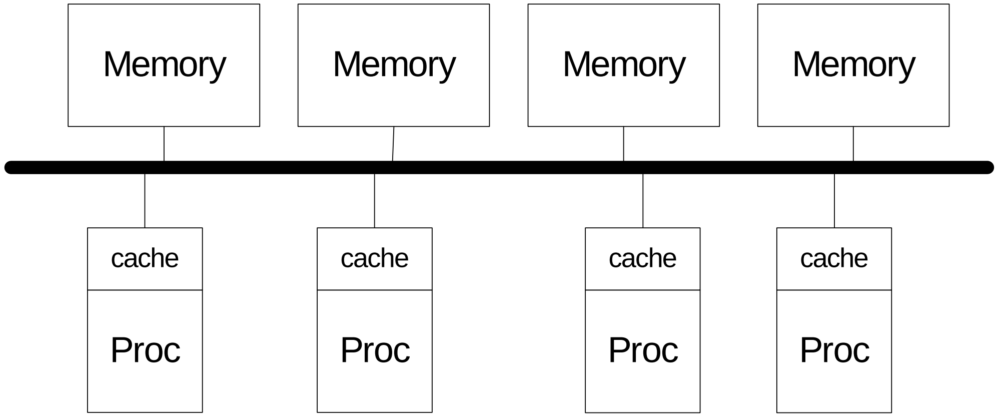
  + Eliminates single point of failure

- Disadvantages
  - Requires routers that can route data/control packets → costs area, power, complexity

  - Maintaining cache coherence is more complex

Bus

+ Simple
+ Cost effective for a small number of nodes
+ Easy to implement coherence (snooping)
- Not scalable to large number of nodes (limited bandwidth, electrical loading → reduced frequency)
- High contention



Crossbar

- Every node connected to every other
- Good for small number of nodes

+ Least contention in the network: high bandwidth

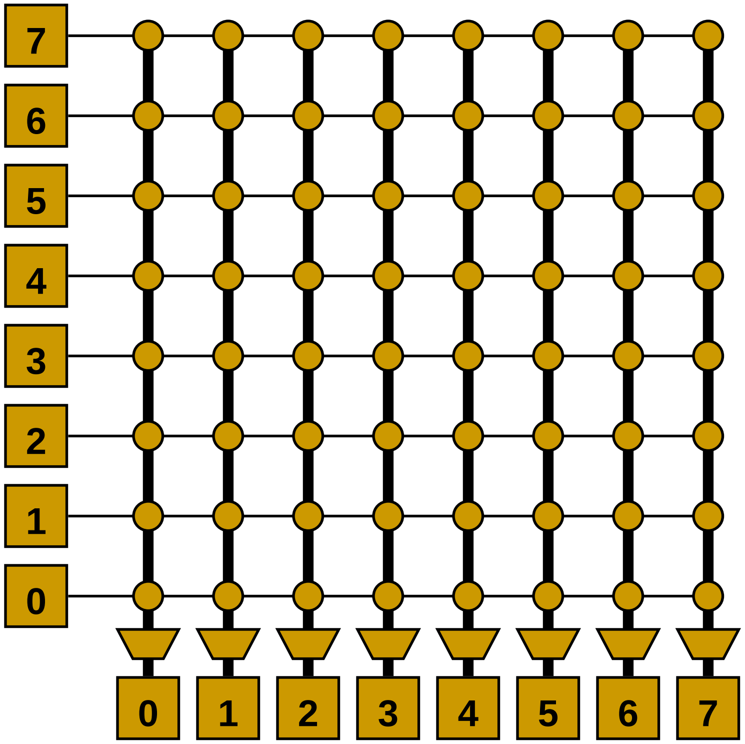
- Expensive

- Not scalable due to quadratic cost

Used in core-to-cache-bank networks in

- IBM POWER5

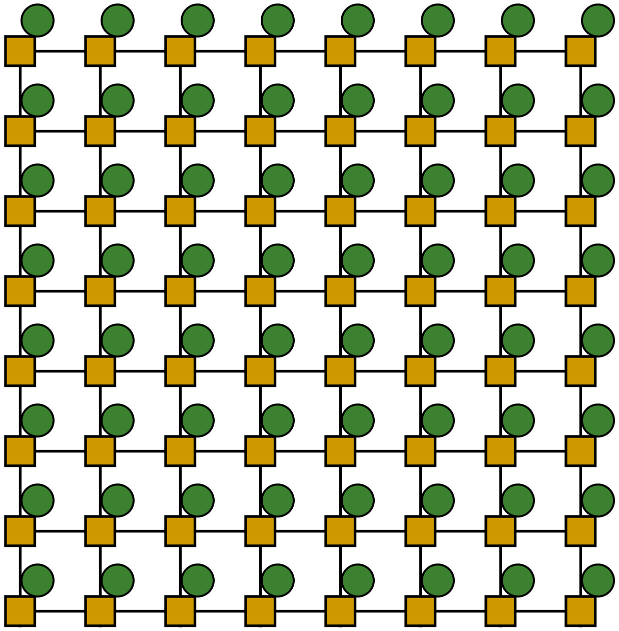
- Sun Niagara I/II



Mesh

- O(N) cost
- Average latency: O(sqrt(N))
- Easy to layout on-chip: regular and equal-length links
- Path diversity: many ways to get from one node to another

- Used in Tilera 100-core
- And many on-chip network prototypes



Torus

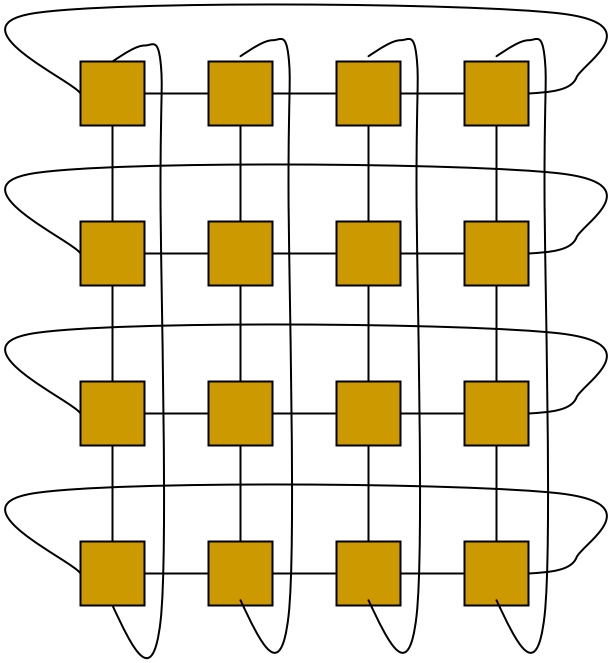
- Mesh is not symmetric on edges: performance very sensitive to placement of task on edge vs. middle
- Torus avoids this problem

+ Higher path diversity than mesh

- Higher cost

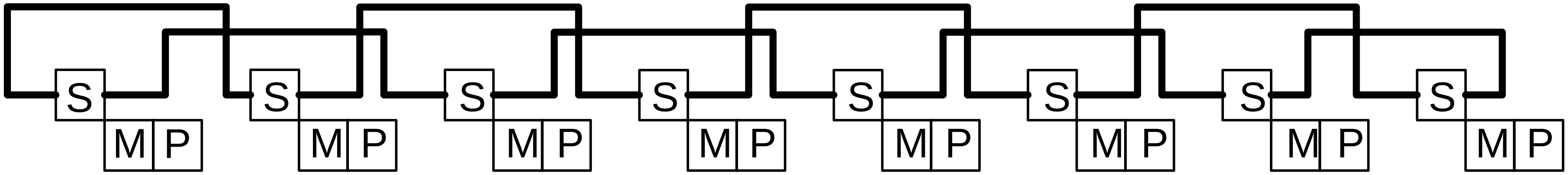
- Harder to lay out on-chip

  - Unequal link lengths



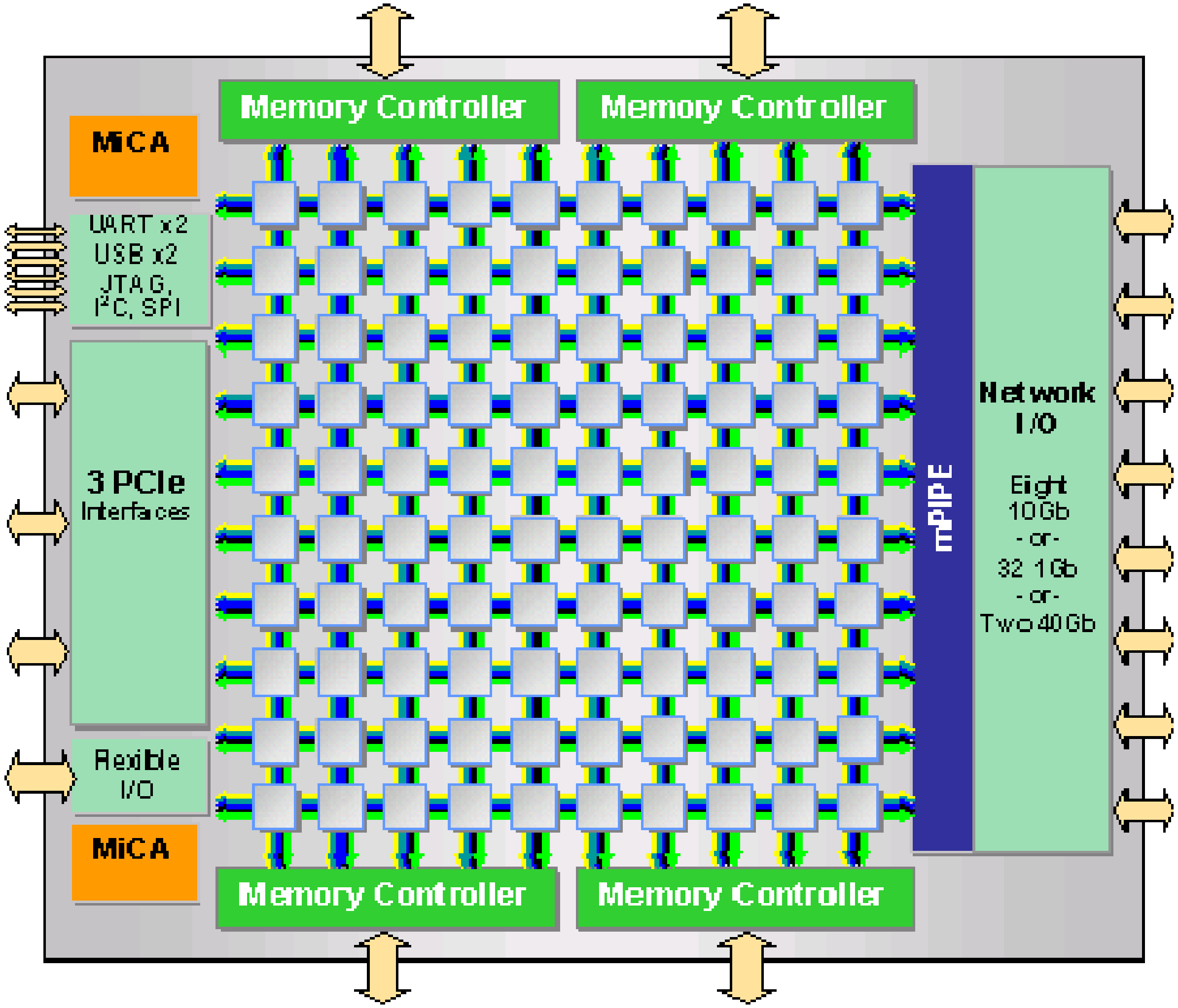
Torus, continued

- Weave nodes to make inter-node latencies ~constant



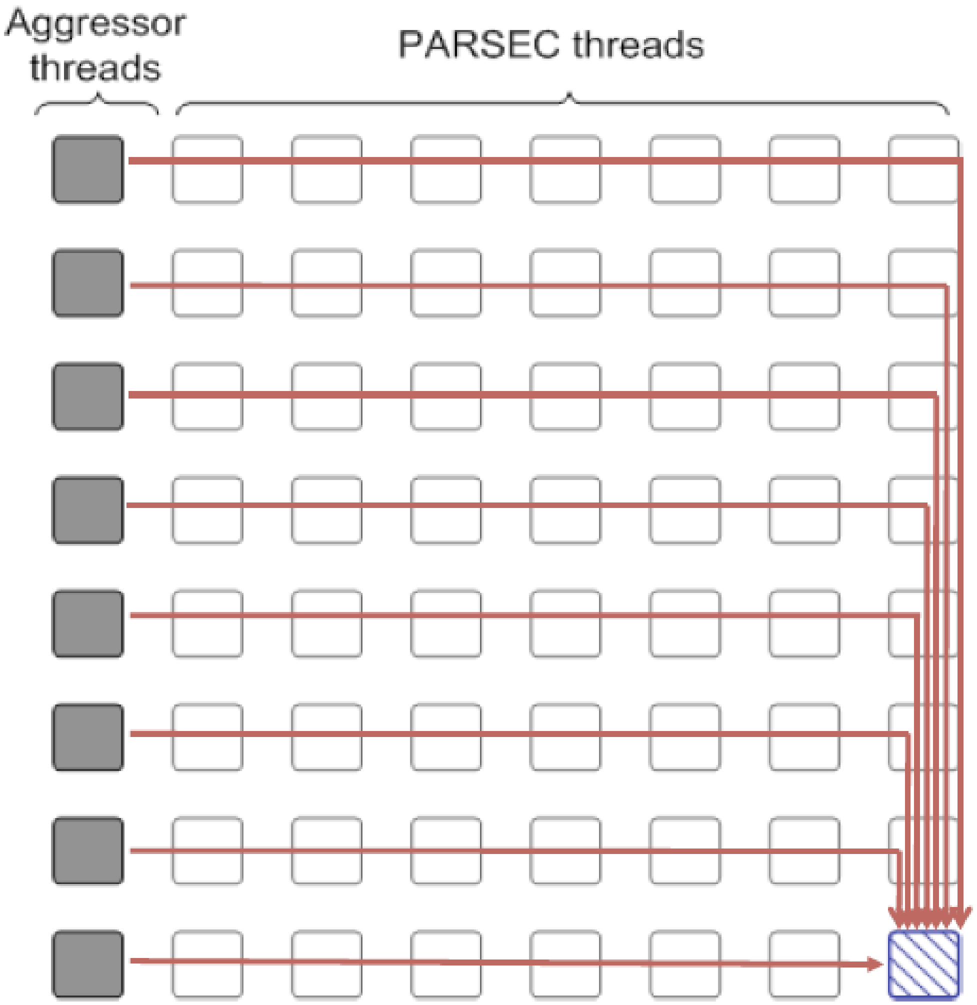
Example NoC: 100-core Tilera Processor

- Wentzlaff et al., “On-Chip Interconnection Architecture of the Tile Processor,” IEEE Micro 2007.



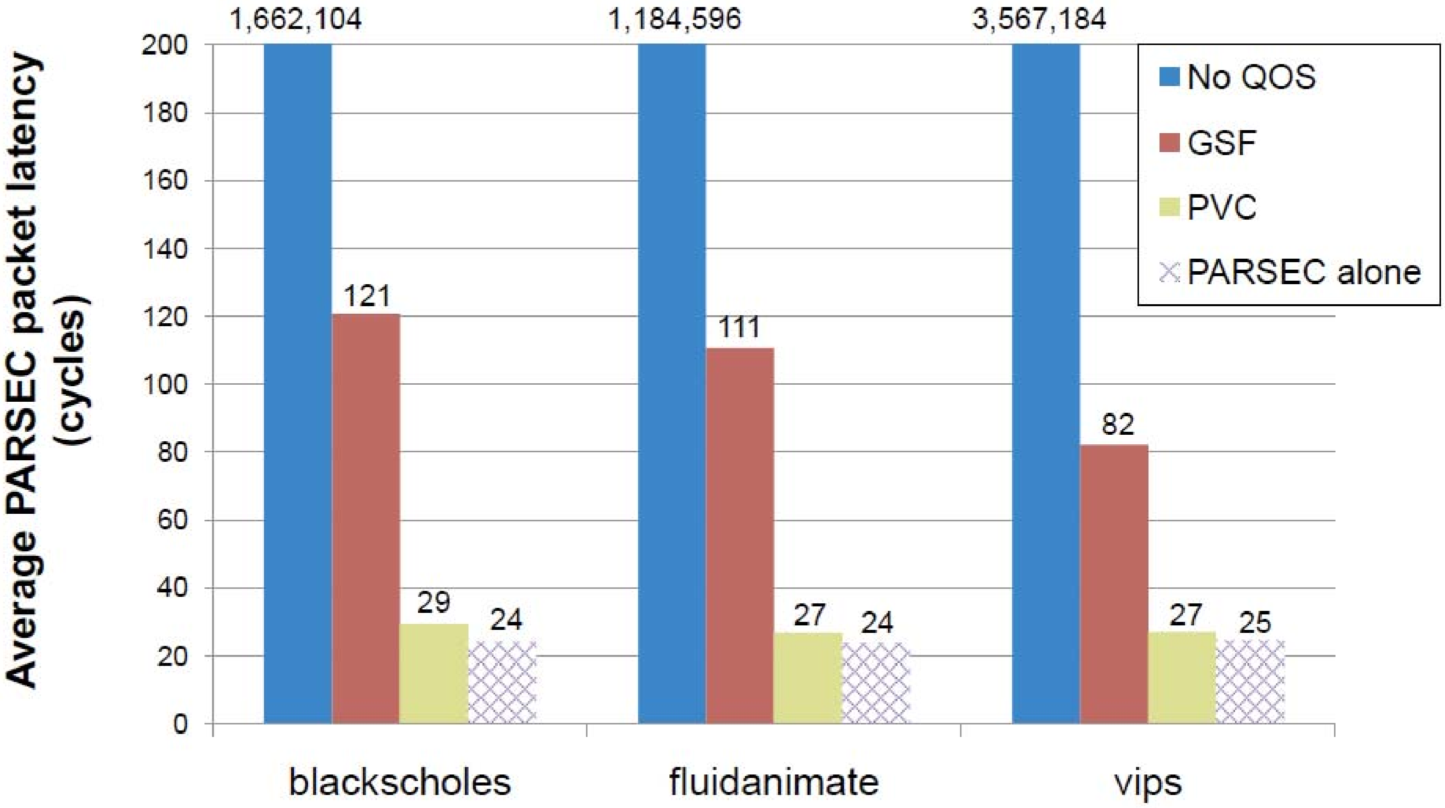
The Need for QoS in the On-Chip Network

- One can create malicious applications that continuously access the same resource → deny service to less aggressive applications



The Need for QoS in the On-Chip Network

- Need to provide packet scheduling mechanisms that ensure applications' service requirements (bandwidth/latency) are satisfied



- Grot et al., "Preemptive Virtual Clock: A Flexible, Efficient, and Cost-effective QOS Scheme for Networks-on-Chip," MICRO 2009.

On Chip Networks: Some Questions

- Is mesh/torus the best topology?
- How do you design the router?
  - High frequency, energy efficient, low latency
  - What is the routing algorithm? Is it adaptive or deterministic?
- How does the router prioritize between different threads'/applications' packets?
  - How does the OS/application communicate the importance of applications to the routers?
  - How does the router provide bandwidth/latency guarantees to applications that need them?
- Where do you place different resources? (e.g., memory controllers)
- How do you maintain cache coherence?
- How does the OS scheduler place tasks?
- How is data placed in distributed caches?

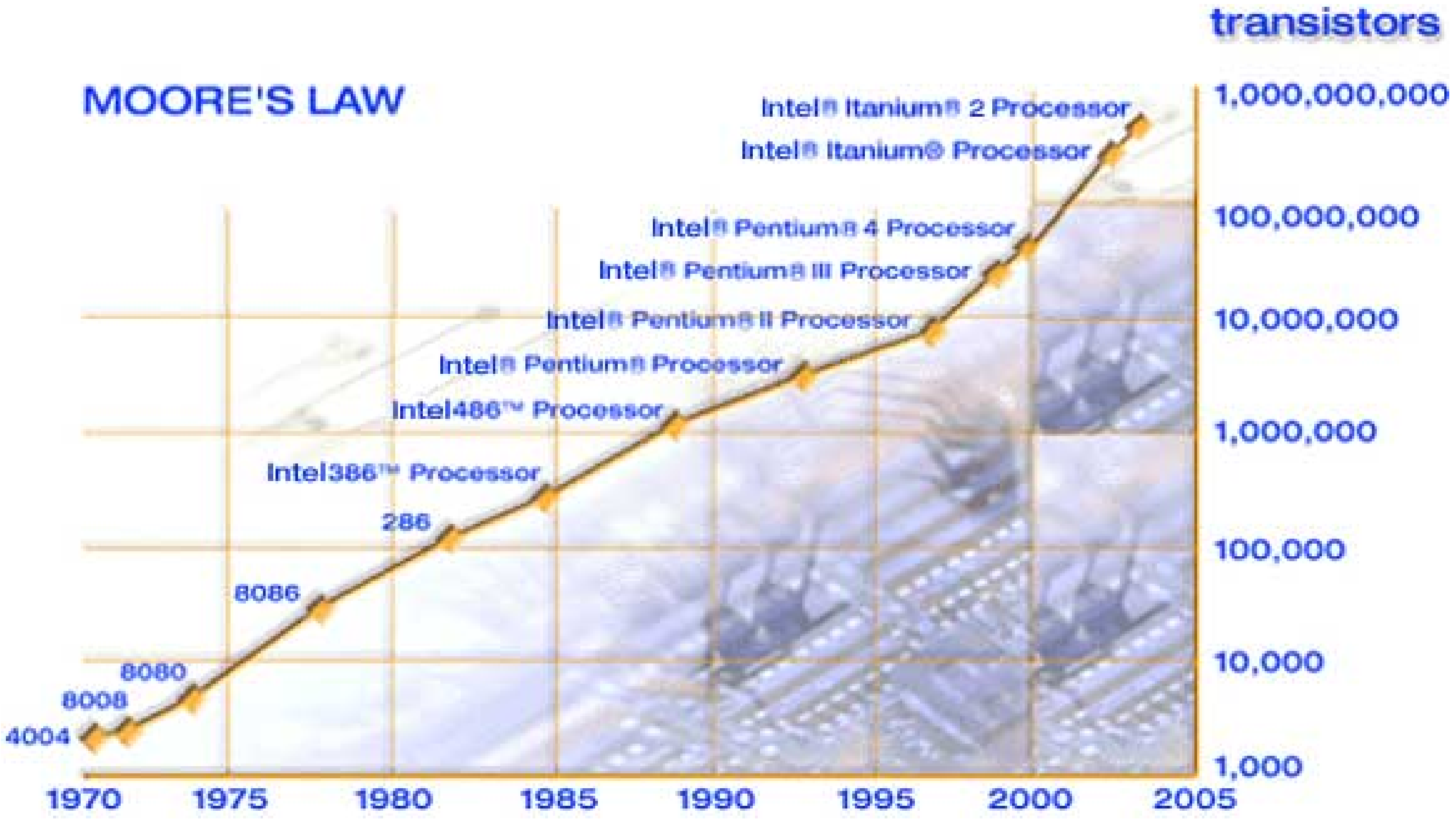
What is Computer Architecture?

- 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.

- We will soon distinguish between the terms *architecture*, *microarchitecture*, and *implementation*.

Why Study Computer Architecture?

Moore's Law



Moore, "Cramming more components onto integrated circuits," Electronics Magazine, 1965.

Why Study Computer Architecture?

- Make computers faster, cheaper, smaller, more reliable
  - By exploiting advances and changes in underlying technology/circuits

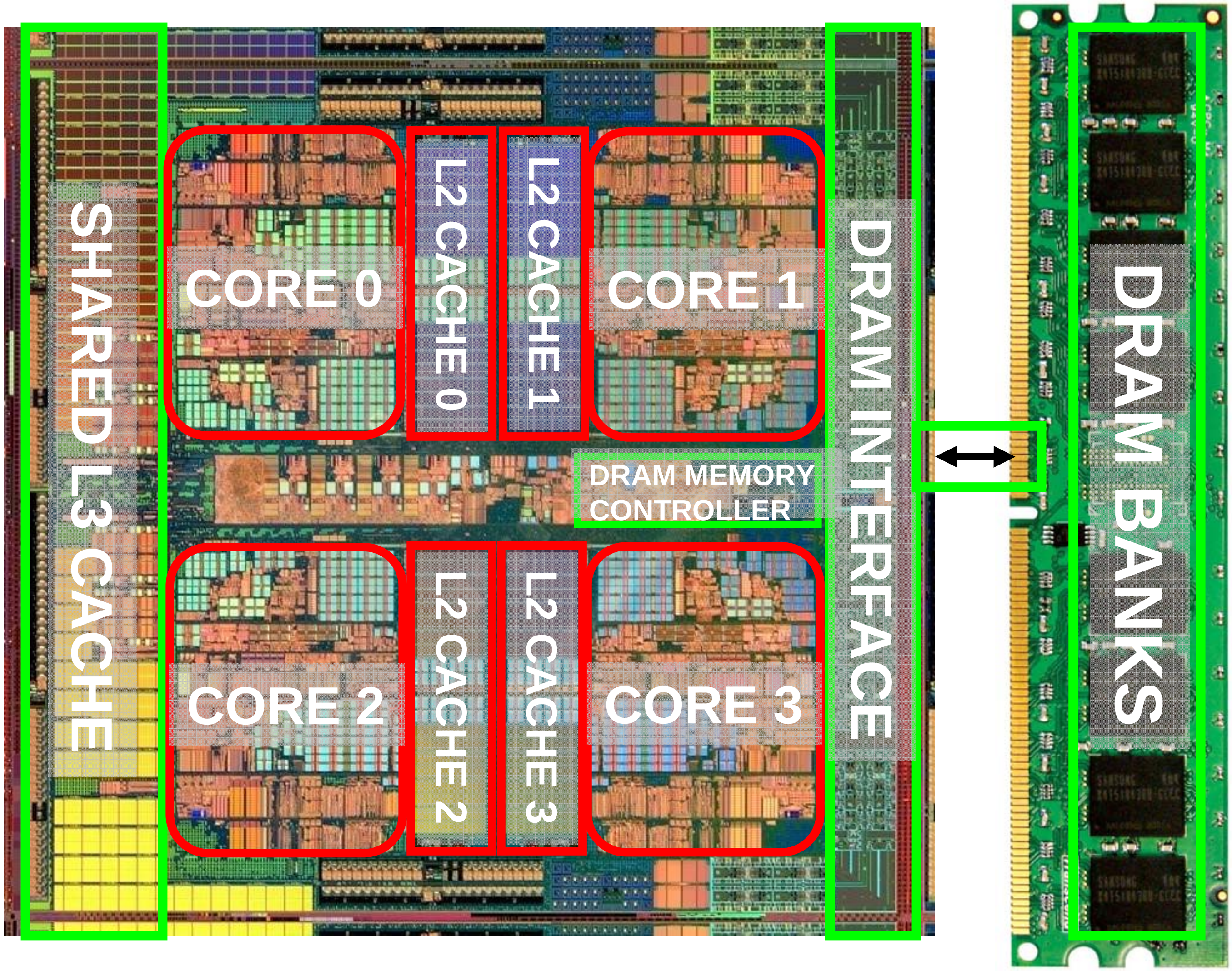
- Enable new applications
  - Life-like 3D visualization 20 years ago?
  - Virtual reality?
  - Personal genomics?

- Adapt the computing stack to technology trends
  - Innovation in software is built into trends and changes in computer architecture
    - > 50% performance improvement per year

- Understand why computers work the way they do

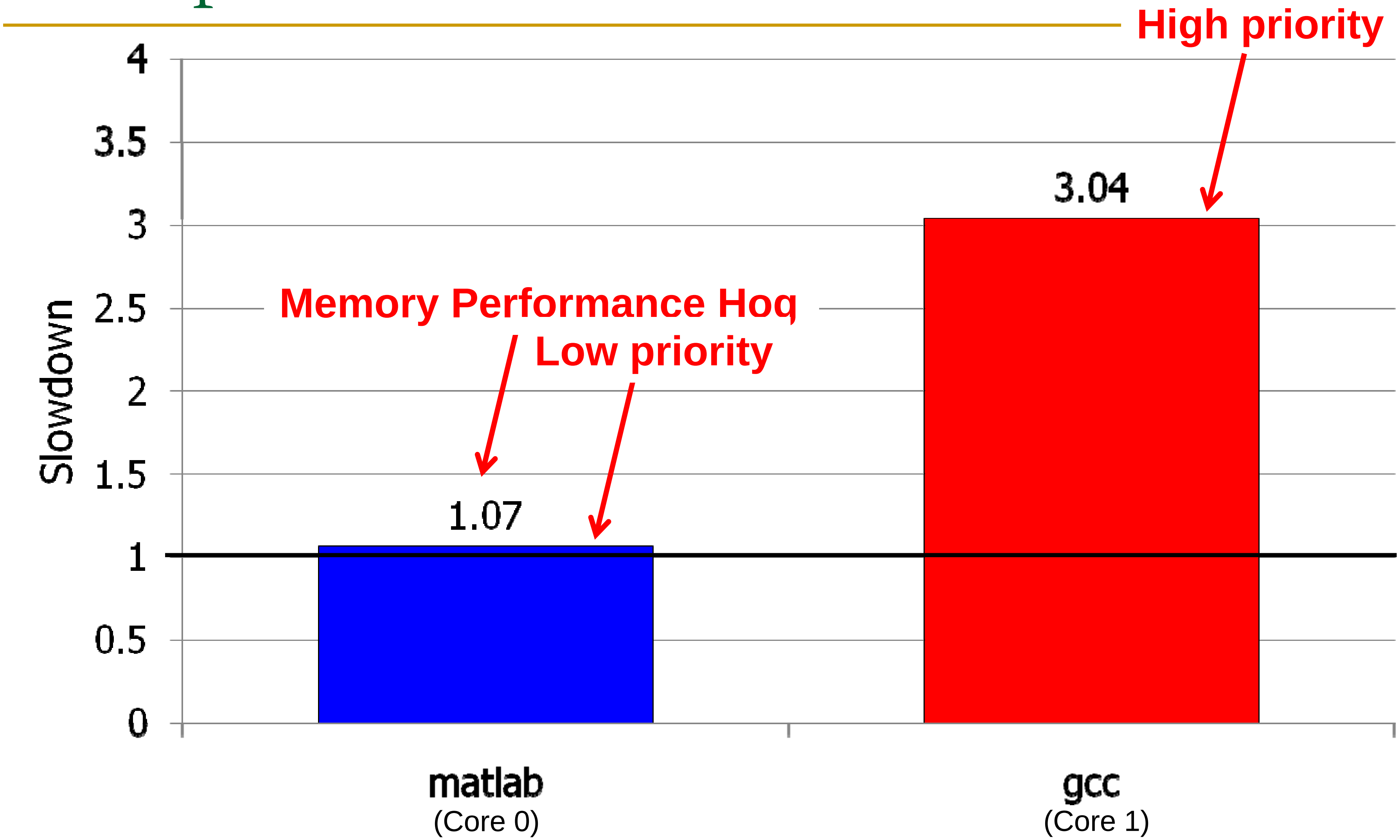
An Example: Multi-Core Systems

Multi-Core Chip



*Die photo credit: AMD Barcelona

Unexpected Slowdowns in Multi-Core



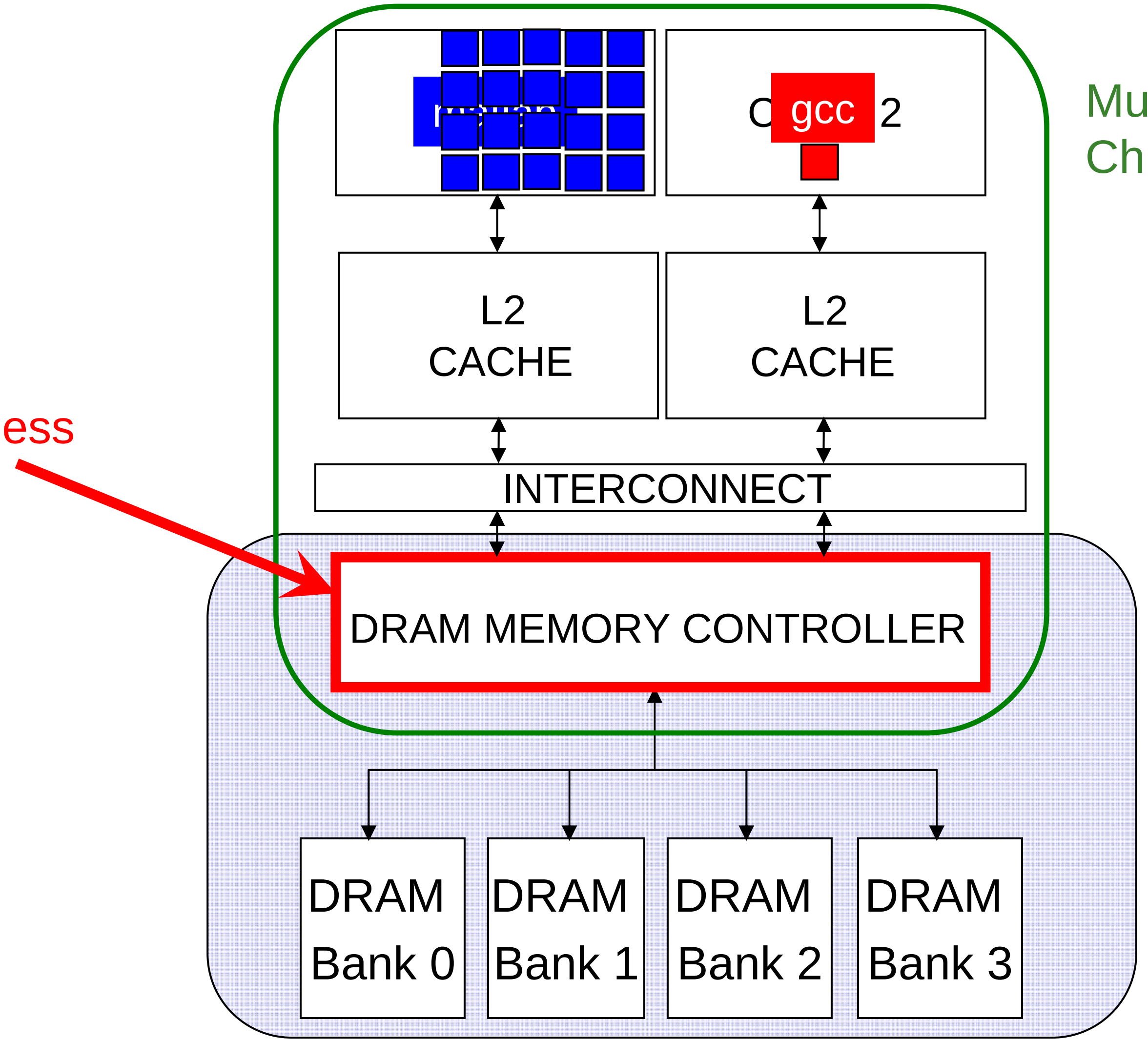
High priority

Memory Performance Hog

Low priority

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

Why the Disparity in Slowdowns?



C gcc 2

L2 CACHE

L2 CACHE

INTERCONNECT

DRAM MEMORY CONTROLLER

DRAM Bank 0

DRAM Bank 1

DRAM Bank 2

DRAM Bank 3

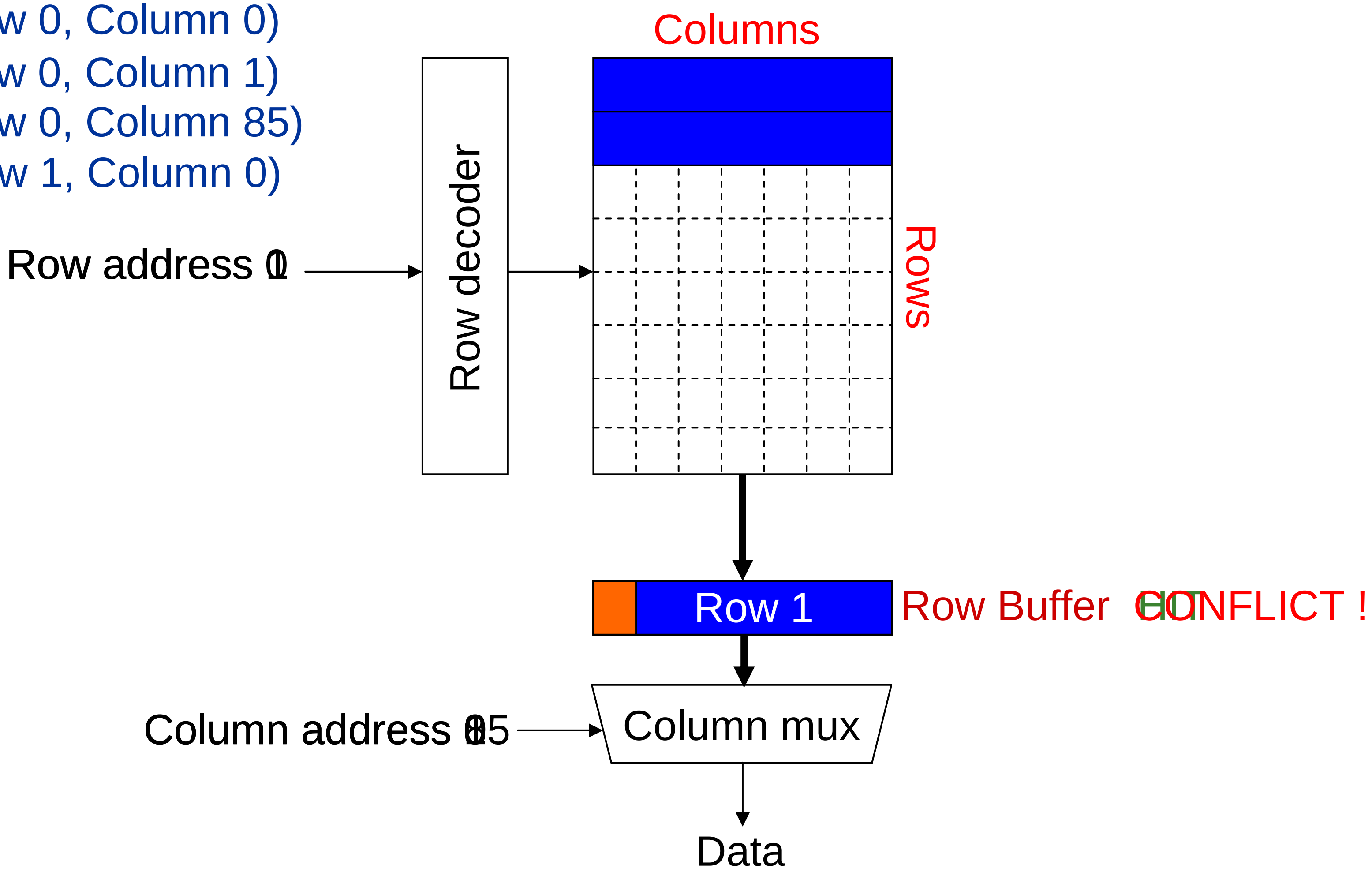
unfairness

Multi-Core Chip

Shared DRAM Memory System

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) [Rixner 2000]*

  (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

*Rixner et al., “Memory Access Scheduling,” ISCA 2000.
*Zuravleff and Robinson, “Controller for a synchronous DRAM …,” US Patent 5,630,096, May 1997.

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
  - Can write programs to exploit unfairness

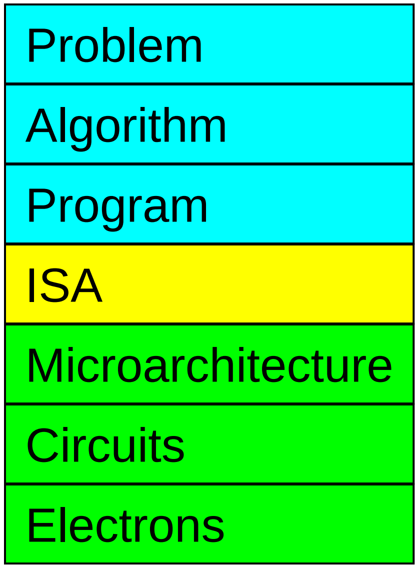
Fundamental Concepts

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.

- **Traditional definition:** “The term *architecture* is used here to describe the attributes of a system as seen by the programmer, i.e., the conceptual structure and functional behavior as distinct from the organization of the dataflow and controls, the logic design, and the physical implementation.” *Gene Amdahl*, IBM Journal of R&D, April 1964

Levels of Transformation



Problem

Algorithm

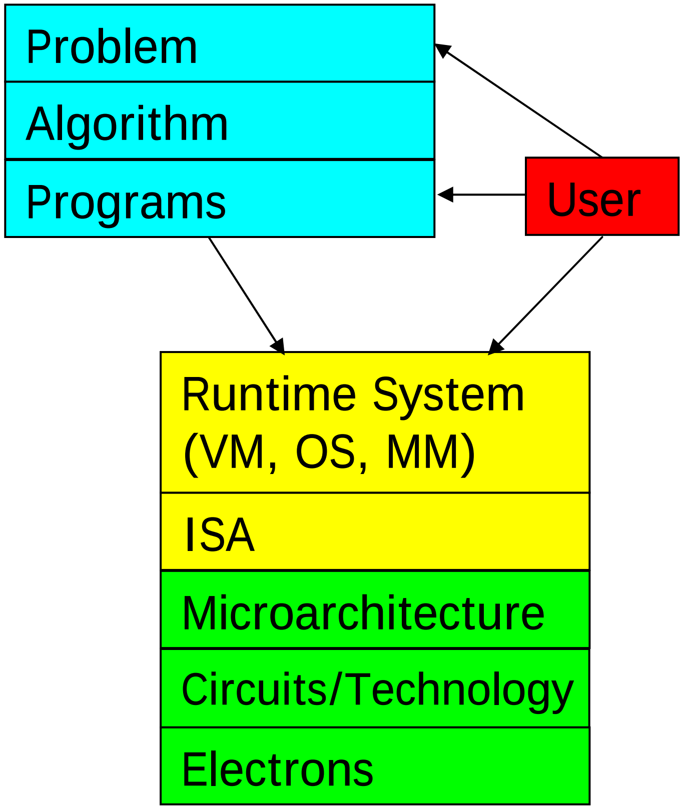
Program

ISA

Microarchitecture

Circuits

Electrons



Problem

Algorithm

Programs

User

Runtime System (VM, OS, MM)

ISA

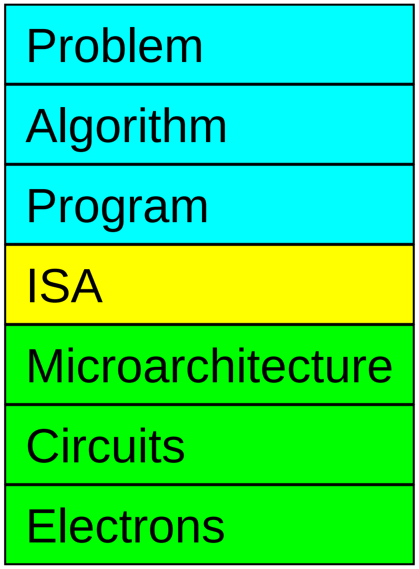
Microarchitecture

Circuits/Technology

Electrons

Levels of Transformation

- ISA
  - Agreed upon interface between software and hardware
    - SW/compiler assumes, HW promises
  - What the software writer needs to know to write system/user programs
- Microarchitecture
  - Specific implementation of an ISA
  - Not visible to the software
- Microprocessor
  - **ISA, uarch**, circuits
  - "Architecture" = ISA + microarchitecture



Problem

Algorithm

Program

ISA

Microarchitecture

Circuits

Electrons

ISA vs. Microarchitecture

- What is part of ISA vs. Uarch?
  - Gas pedal: interface for “acceleration”
  - Internals of the engine: implements “acceleration”
  - Add instruction vs. Adder implementation

- Implementation (uarch) can be various as long as it satisfies the specification (ISA)
  - Bit serial, ripple carry, carry lookahead adders
  - x86 ISA has many implementations: 286, 386, 486, Pentium, Pentium Pro, …

- Uarch usually changes faster than ISA
  - Few ISAs (x86, SPARC, MIPS, Alpha) but many uarchs
  - *Why?*