COMP 522 Multicore Computing: An Introduction

John Mellor-Crummey

Department of Computer Science
Rice University

johnmc@rice.edu

Logistics

Instructor: John Mellor-Crummey

- —email: johnmc@rice.edu
- —phone: x5179
- —office: DH 3082
- —office hours: by appointment

Teaching Assistant: Keren Zhou

- —email: kz21@rice.edu
- —office: DH 2069
- —office hours: by appointment

Meeting time

- —scheduled T/Th 1:00 - 2:15

Class Location: DH 1075

Web site: http://www.cs.rice.edu/~johnmc/comp522

Multicore Computing

How are modern microprocessors organized and why?

How do threads share cores for efficient execution?

How do threads share data?

What are memory models and why do you need them?

How does one synchronize data accesses and work?

How does one write highly concurrent data structures?

How do parallel programming models work?

How can a runtime schedule work efficiently on multiple cores?

How can you automatically detect errors in parallel programs?

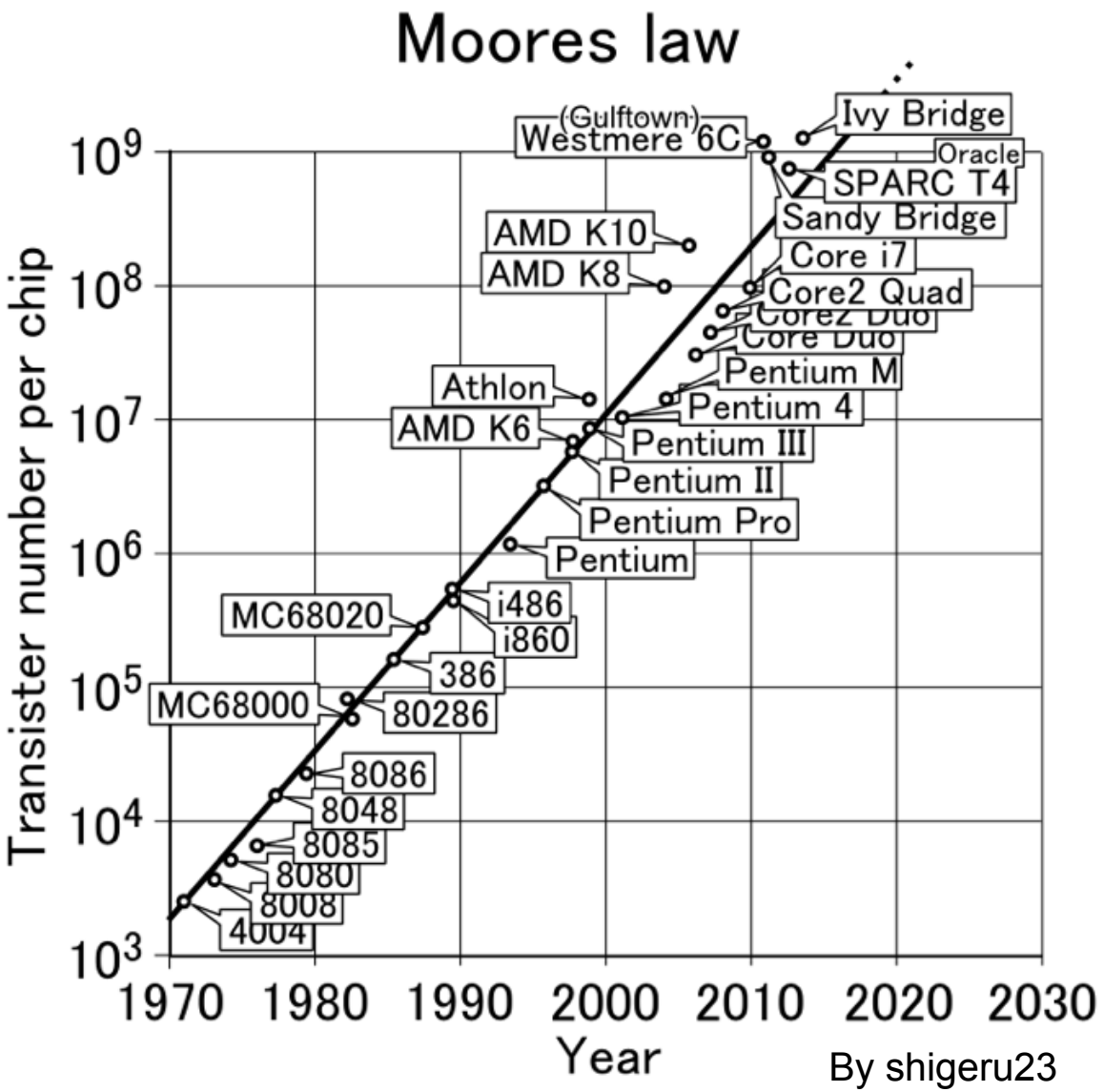
How can you tell if you are using microprocessors efficiently?

The Shift to Multicore Microprocessors

Advance of Semiconductors: “Moore’s Law”

Gordon Moore, Founder of Intel

- **1965: since the integrated circuit was invented, the number of transistors in these circuits roughly doubled every year; this trend would continue for the foreseeable future**
- **1975: revised - circuit complexity doubles every two years**



By shigeru23
CC BY-SA 3.0,
via Wikimedia
Commons

Increasing problems with power consumption and heat as clock speeds increase

25 Years of Microprocessors

- Performance increased over 1000x
- Transistor density gains from Moore's Law have driven
  - —increases in transistor speed from higher clock rates
  - —energy scaling
  - —microarchitecture advances from additional transistors

Why Multicore Microprocessors?

- Rising transistor count enables more functionality
- Why not make single cores more sophisticated?
  - —let's take a look at a few microprocessors for some intuition …

Pentium 4 (Super-Pipelined, Super-Scalar)

Stages 1-2
- — Trace cache next instruction pointer

Stages 3-4
- — Trace cache fetch

Stage 5
- — Drive (wire delay)

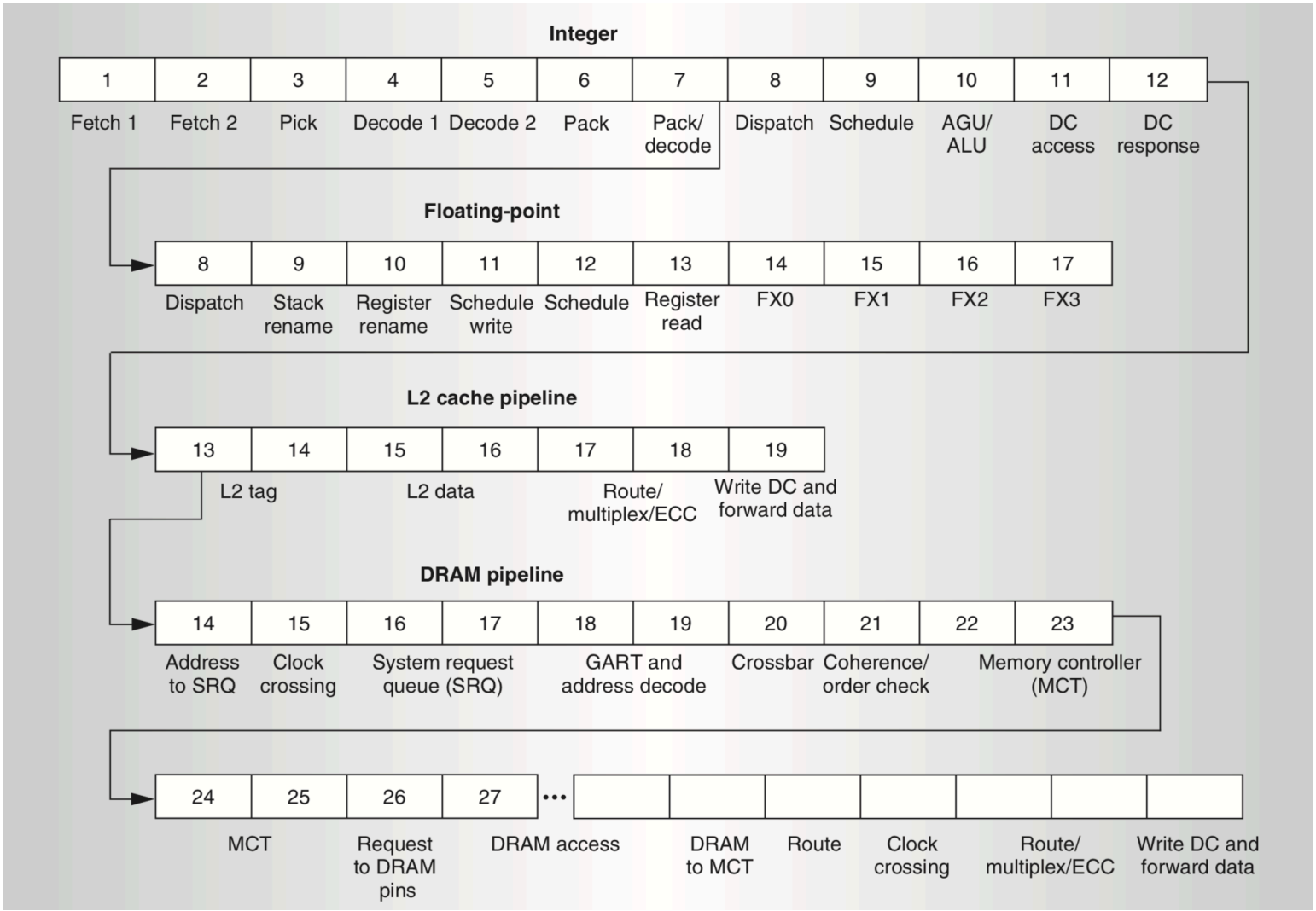
Stages 6-8
- — Allocate and rename

Stages 10-12
- — Schedule instructions
  - – memory/fast ALU/slow ALU & general FP/simple FP

- **Stages 13-14**
  - — Dispatch
- **Stages 15-16**
  - — Register access
- **Stage 17**
  - — Execute
- **Stage 18**
  - — Set flags
- **Stage 19**
  - — Check branches
- **Stage 20**
  - — Drive (more wire delay)

5 operations issued per clock
1 load, 1 store unit
2 simple/fast, 1 complex/slower integer units
1 FP exe, 1 FP move unit
Up to 126 instructions in flight: 48 loads, 24 stores, …

Opteron Pipeline (Super-Pipelined, Super-Scalar)



Fetch/decode 3 inst/cycle, 3 integer units, 3 address units
3 FPU/multimedia units, 2 load/stores to D-cache/cycle

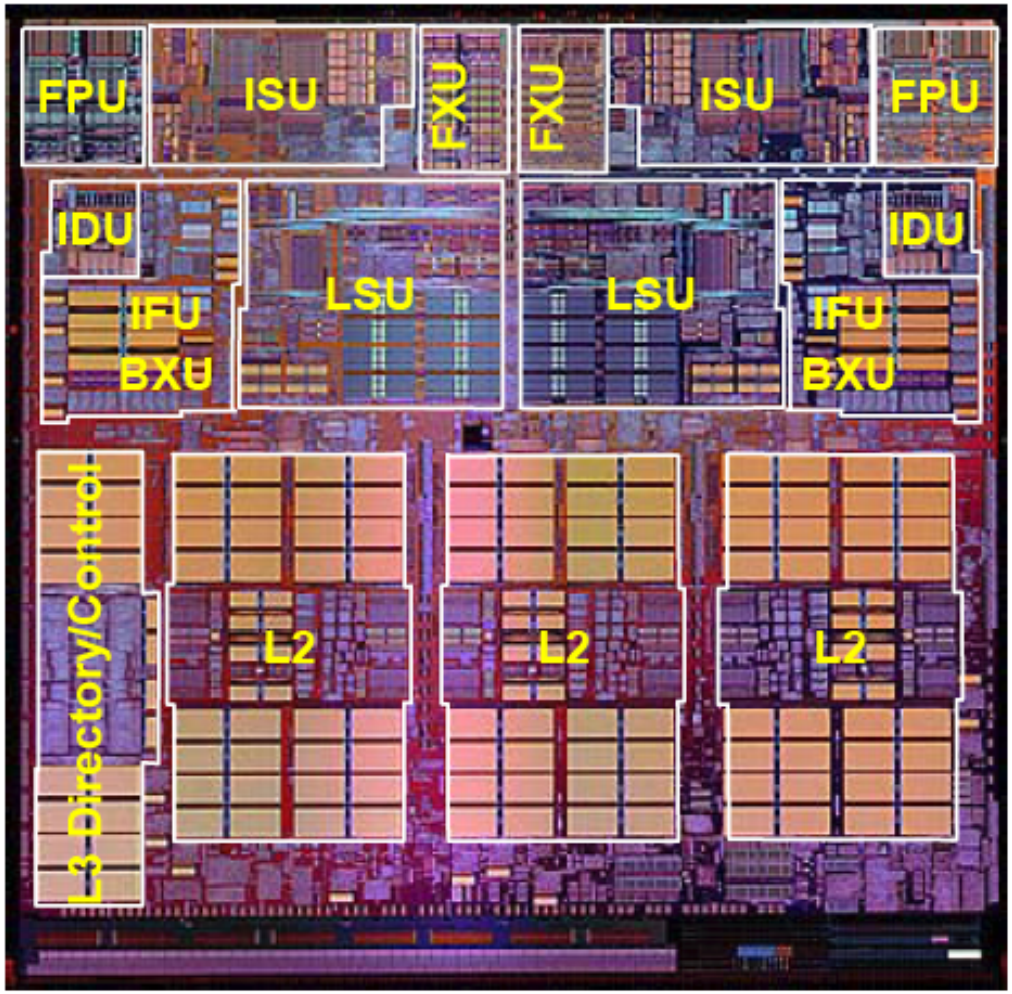
C. N. Keltcher, K. J. McGrath, A. Ahmed, and P. Conway. 2003. The AMD Opteron Processor for Multiprocessor Servers. IEEE Micro 23, 2 (March 2003), 66-76. http://dx.doi.org/10.1109/MM.2003.1196116

Multicore Microprocessors

- Why not make single cores more sophisticated?
  - —limits to instruction-level parallelism available in programs
    - especially for codes with difficult-to-predict branches
- A new approach
  - —use available chip real estate to support thread-level parallelism
- Strategies
  - —multiple threads per core
    - Simultaneous multithreading: maximizing on-chip parallelism, Dean Tullsen, Susan Eggers, and Henry Levy, ISCA, 1995.
  - —multiple cores
    - A single-chip multiprocessor, Lance Hammond, Basem Nayfeh, Kunle Olukotun. Computer 30(9):79-85, September 1997.

IBM Power4 - December 2001

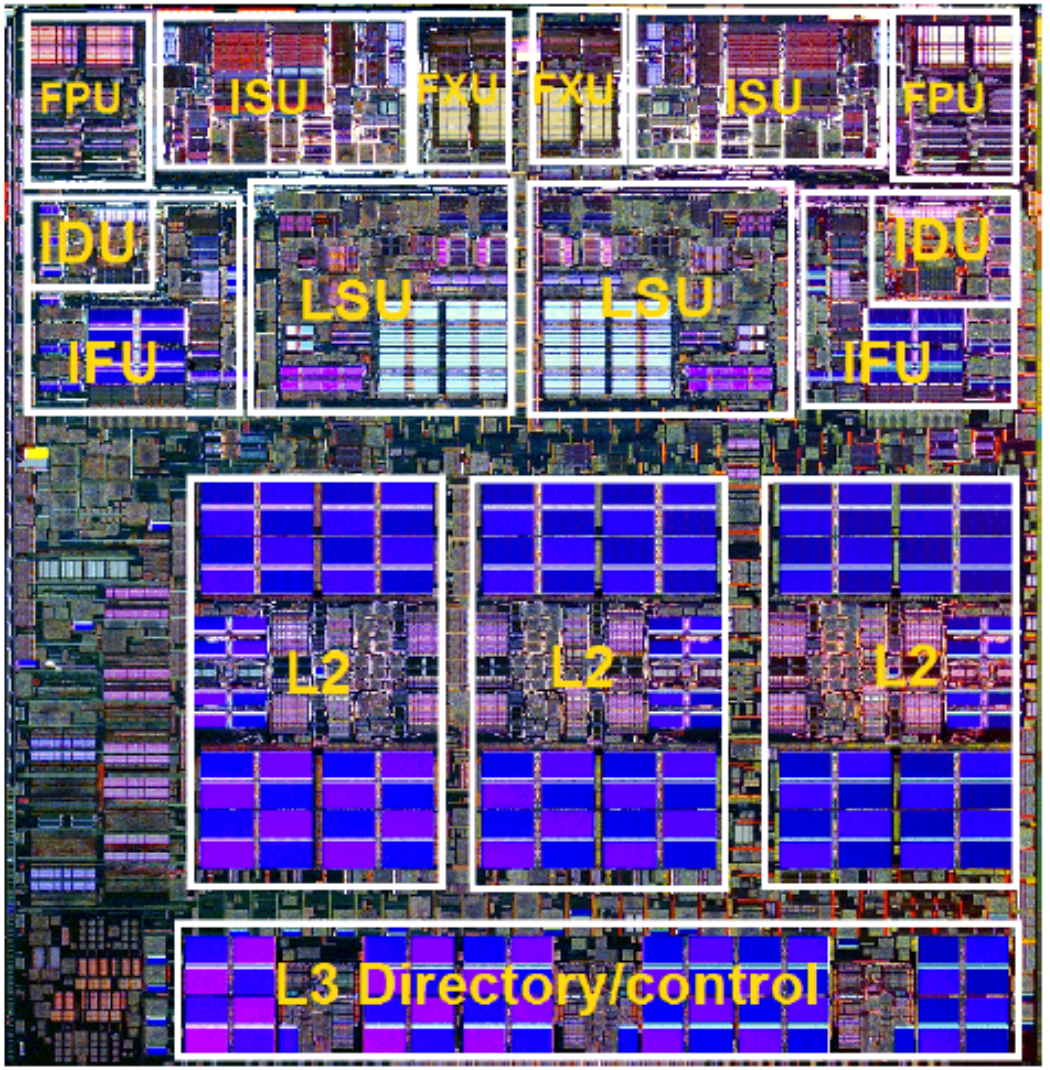
- Technology: 180nm lithography, Cu, SOI
  - POWER4+ shipping in 130nm today
- Dual processor core
- 8-way superscalar
  - Out of Order execution
  - 2 Load / Store units
  - 2 Fixed Point units
  - 2 Floating Point units
  - Logical operations on Condition Register
  - Branch Execution unit
- > 200 instructions in flight
- Hardware instruction and data prefetch



From Hot Chips 15, August 2003 *POWER5: IBM's Next Generation POWER Microprocessor*, Ron Kalla (IBM)

IBM Power 5 - August 2003

- Technology: 130nm lithography, Cu, SOI
- Dual processor core
- 8-way superscalar
- Simultaneous multithreaded (SMT) core
  - Up to 2 virtual processors per real processor
  - 24% area growth per core for SMT
  - Natural extension to POWER4 design



From Hot Chips 15, August 2003 *POWER5: IBM's Next Generation POWER Microprocessor*, Ron Kalla (IBM)

Intel x86 Development Changes Course …

May 17, 2004 … Intel, the world's largest chip maker, publicly acknowledged that it had hit a "thermal wall" on its microprocessor line. As a result, the company is changing its product strategy and disbanding one of its most advanced design groups. Intel also said that it would abandon two advanced chip development projects …

Now, Intel is embarked on a course already adopted by some of its major rivals: obtaining more computing power by stamping multiple processors on a single chip rather than straining to increase the speed of a single processor … Intel's decision to change course and embrace a "dual core" processor structure shows the challenge of overcoming the effects of heat generated by the constant on-off movement of tiny switches in modern computers … some analysts and former Intel designers said that Intel was coming to terms with escalating heat problems so severe they threatened to cause its chips to fracture at extreme temperatures…

New York Times, May 17, 2004

Dual-core AMD Opteron Announced

August 31, 2004 - Advanced Micro Devices plans to demonstrate its version of a new approach to processor design on Tuesday, with a chip that is expected to offer faster computing and relatively less power consumption. The chip, which is called Opteron and has two processing units …

The shift to multiple processing units, or cores, embedded in the same chip has recently become a significant technological approach for IBM, Sun Microsystems and Intel as well as Advanced Micro, as computer designers hunt for new ways to increase processing power … Advanced Micro said on Monday that it would make the chips available commercially for servers in mid-2005 and in its 64-bit Athlon chips for desktop computers before the end of next year. Intel has not yet set dates for its dual-core X86 processors.

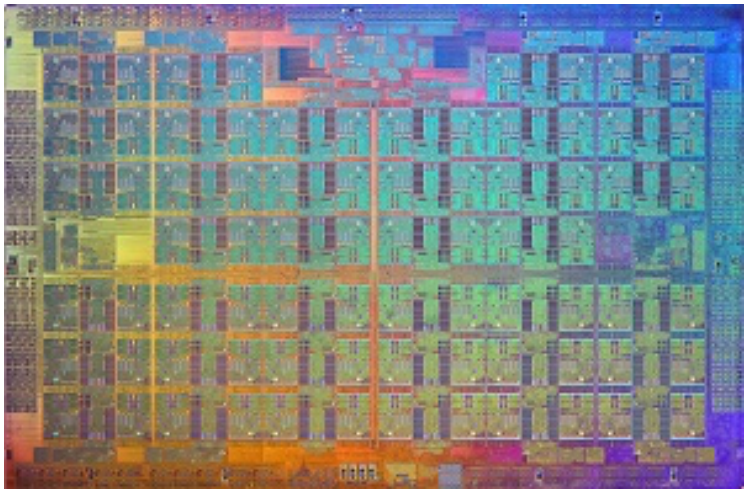
New York Times, August 31, 2004

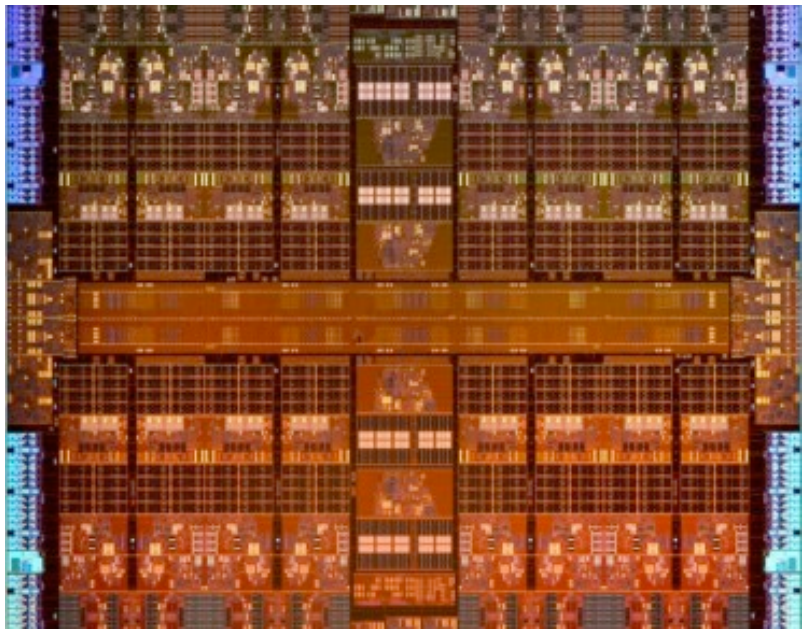
Sun Niagara (Ultrasparc T1) Announced

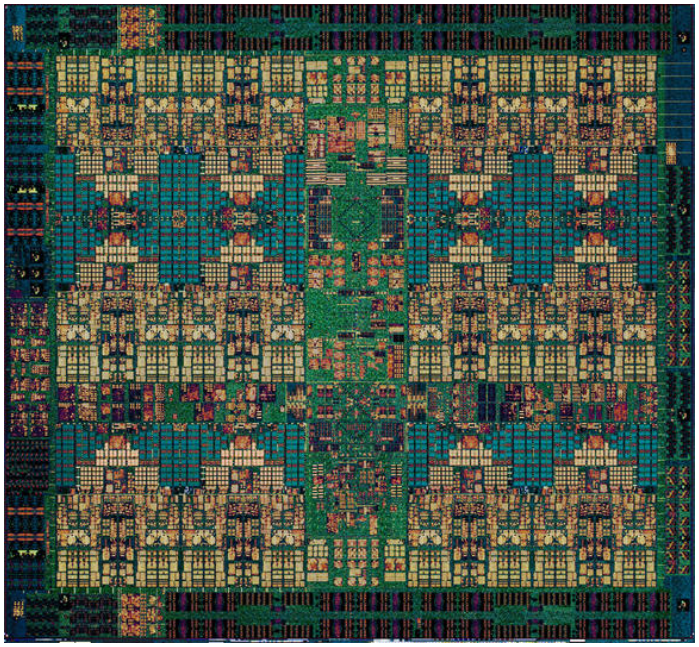
November 14, 2005 Sun Microsystems is set to announce the first of a new generation of processors for computer servers on Monday that the company says offers faster performance with far less energy use … The chip, code-named Niagara while in development, is designed for a specific niche of the server market: high-volume Web service operations, ... The UltraSparc T1, following a trend in the semiconductor industry, adds new features that conserve energy significantly … The UltraSparc T1 has eight processing cores, each able to execute four instruction sequences, called threads …

New York Times, November 15, 2005

Recent Multicore Designs





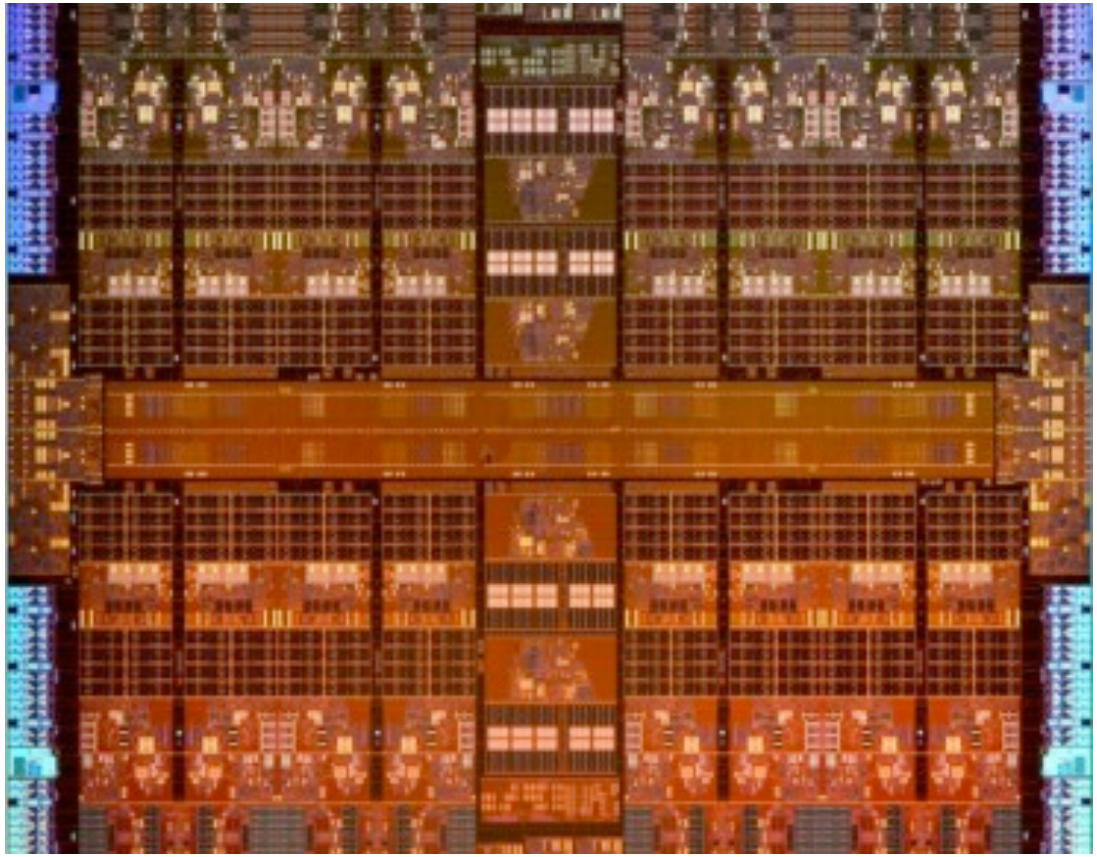


Intel Xeon Platinum 8180 Processor (2017)

- 2.5GHz (Turbo 3.8GHz)
- 28 cores
  - 2 way SMT
  - 32KB L1I cache
  - 32KB L1D cache
  - 1MB L2 cache
- 56 threads per chip
- 38.5MB L3 cache
- 119.21 GB/s BW

Latency-optimized cores

Oracle SPARC M7 (2015)



- 10B transistors
- 4.13GHz
- 32 cores
  - —8 threads per core
  - —16 KB L1 I-cache
  - —16 KB of L1 D-cache
- 256 KB L2 I-cache/4-cores
- 256 KB L2 writeback data cache per 2-cores
  - —total L2 BW > 8 TB/sec
- 64 MB L3 cache

dual-issue, 2-way SMT

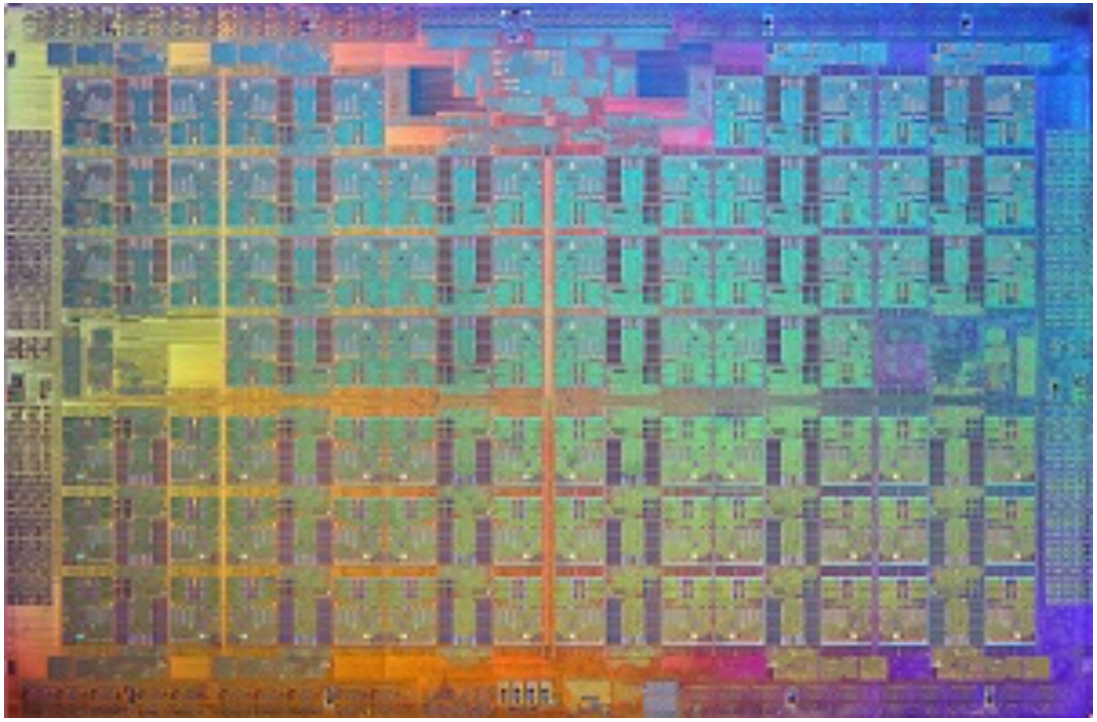
http://www.enterprisetech.com/2014/08/13/oracle-cranks-cores-32-sparc-m7-chip/

Throughput-optimized cores

Intel Knight's Landing 7290F (2016)

- > 8 billion transistors
- 1.3GHz (1.5GHz Turbo)
- 72 cores
  - 4-way SMT per core
  - 288 threads per chip
- Up to 384GB of DDR4-2400 main memory
  - 115GB/s max mem BW
- Up to 16GB of MCDRAM on-package (3D stacked)
  - 400GB/s max mem BW

2nd Generation Xeon Phi

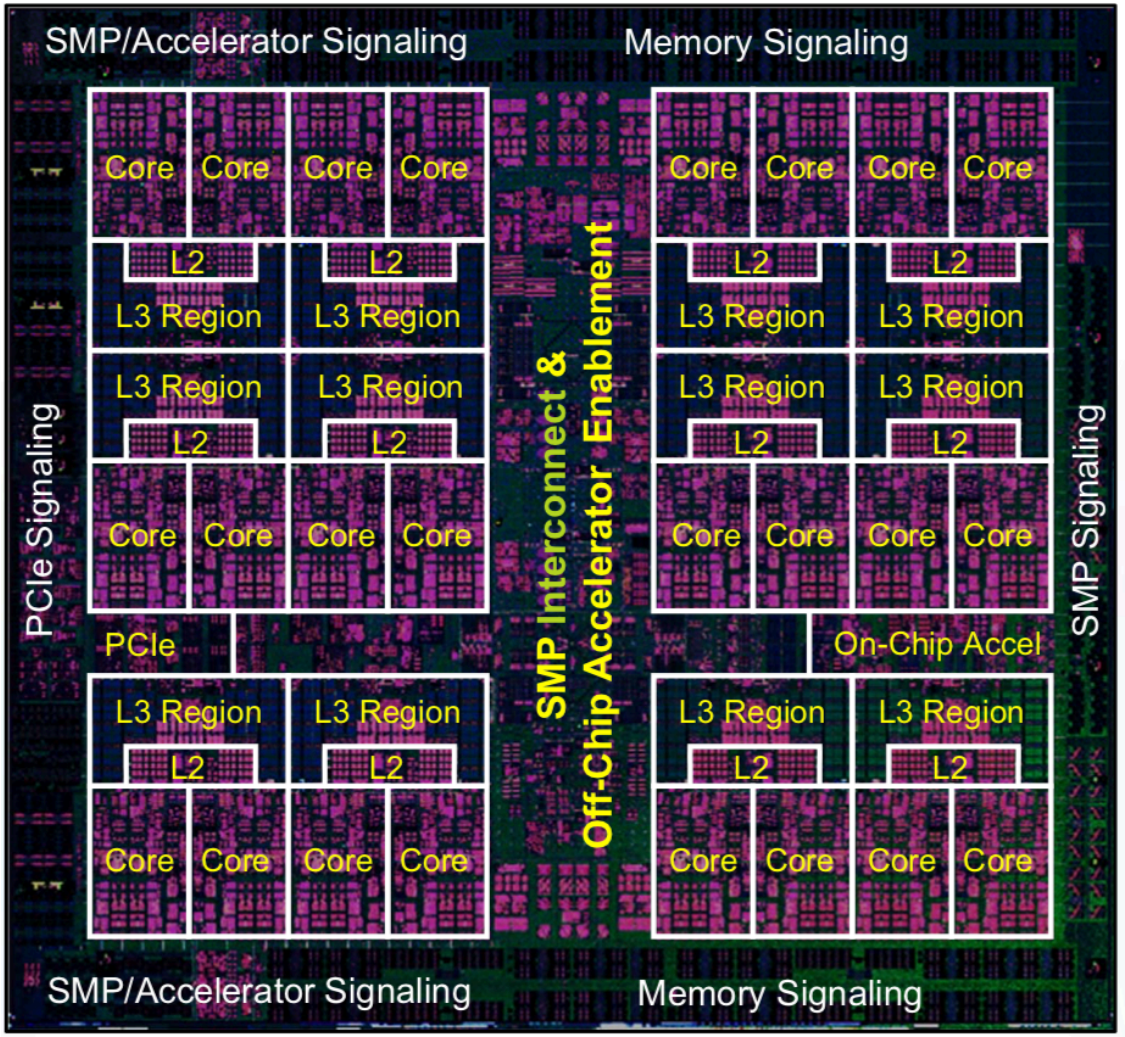


http://ark.intel.com/products/95831/Intel-Xeon-Phi-Processor-7290F-16GB-1_50-GHz-72-core

Throughput-optimized cores

IBM Power9 (2017)

- 8B transistors
- 24-core processor
  - 16 execution units
  - 4-way SMT per core
  - 96 threads per chip
  - 512KB L2 per core
  - 12-stage pipeline
- 120MB on-chip shared L3
- 7TB/s on-chip BW
- 120GB/s memory BW
- OpenCAPI
  - coherent accelerator processor interface



Brian Thompto, Power9: Processor for the Cognitive Era, Hot Chips 28, August 2016. (link)

Latency-optimized cores

Using Multicore Processors

The processor core is only part of the picture …

Data sharing

- —cache hierarchy designs and implications
  - shared cache vs. private cache: costs vs. benefits?

Synchronization

- —what is possible?
- —what hardware primitives exist?
- —how should they be used?
- —what are the implications for programming models?
- —what alternative hardware primitives should be explored?

Isn’t Multicore just More of the Same?

No! The emergence of multicore processors marks a watershed for software

Processor clock frequency increases can no longer compensate for increasing software bloat

Application speed won’t track processor enhancements unless software is highly concurrent

What about the Software?

With concurrency essential for performance …

Languages and programming systems must embrace concurrency to survive

- —semantics: what operations are atomic? what ordering of operations is guaranteed? when are memory effects of operations visible?
- —expressiveness: embarrassingly parallel, data-parallel, task parallel
- —simplicity: make parallelism easy to use
- —efficient

Challenge: concurrent programming is much more difficult!

- —software development is much harder
  - – lack of standardized & effective development tools, programming models, and environments
- —algorithm development is harder
  - – complexity of specifying and coordinating concurrent activities
- —concurrent data structures are extremely difficult to design
- —managing memory layout and data locality is critical for performance

What Makes Concurrent Programming So Hard?

The problem of shared state

Application threads generally need to share state

Data race

- **—two threads access a shared variable**
- **—at least one access is a write**
- **—no ordering guaranteed by synchronization**

Data races can yield unpredictable or corrupted values

Race conditions are easy to write and hard to pinpoint

Data races must be avoided for correct execution!

Avoiding Data Races

Conventional approach: mutual exclusion via locks

- —each thread must acquire a lock for shared data before using it

Problems with locks

- —not robust: if lock holder delayed, progress stalls
- —relationship between lock and data is implicit
  - preserved only through programmer discipline
  - association between lock and data is a global property
    - convention must be observed by all code accessing the data
- —hard to use
  - coarse-grain locks shackle parallelism
  - fine-grain locks admit possibility of deadlock
  - lack of composability when layering software
    - calling into code you don't control is a recipe for deadlock
      - locks must be acquired in a fixed global order to avoid deadlock
    - extensible frameworks often call virtual functions while holding lock

Java's Synchronized Methods

Works as long as

- —methods properly annotated with synchronized declarations
- —data accessed only by methods

Problems

- —calling virtual function while holding a lock admits deadlock
- —locking for synchronized methods adds overhead even when no concurrency
- —doesn't guarantee atomicity across multiple method calls
  - example: acount1.debit(amount); account2.credit(amount);

    object locking preserves atomicity of method, but not sequence

    additional explicit synchronization required

Alternatives to Locks

Lock free programming of concurrent data structures

- —requires deep knowledge of processor's memory model
- —difficult and fragile
- —implementations of new data structures are publishable results!
  - e.g. Maged M. Michael, Michael L. Scott: Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms. PODC, 1996, 267-275.

    doubly-ended queue, two-lock and non-blocking versions

Transactional memory

- —operations that appear to execute indivisibly
  - concurrent operations see state before or after operation
- —area of active research
- —recent hardware implementations: IBM Power8, Intel Haswell

Course Outline

Objectives

Immersion in research related to multicore processors

- —issues shaping the design of multicore processors
- —difficulties of programming multicore systems
- —emerging programming models for multicore
- —emerging technologies to simplify multicore programming
  - synchronization, debugging, concurrent data structures

Hone your ability to analyze research papers

- —paper evaluations
- —class discussions

Develop skill preparing and delivering presentations

Explore a topic of interest in greater depth

- —final project or term paper

Topics

- **Microprocessors**
  - explore the design space, threading, flavors of multicore designs
- **Memory hierarchy**
  - cache organizations and their implications for performance
- **Memory models**
  - hardware memory models, Java and C++ memory models
- **Programming models**
  - languages (Cilk/Cilk++/CilkPlus), directives (OpenMP), libraries (TBB)
- **Performance analysis of multithreaded code**
- **Scheduling**
  - strategies for distributing parallel work among processors
- **Debugging**
  - techniques for uncovering and pinpointing data races in parallel code
- **Synchronization**
  - theoretical underpinnings, a range of approaches h/w and s/w approaches
- **Concurrent data structures**
- **Transactional memory**

Award Winning Papers ...

Simultaneous multithreading: maximizing on-chip parallelism, Dean Tullsen, Susan Eggers, and Henry Levy, *ISCA,* 1995. **25 Years of ISCA: Retrospectives and Reprints, 1998.**

The Implementation of the Cilk-5 Multithreaded Language, Matteo Frigo, Charles E. Leiserson, and Keith H. Randall. 1998 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), Montreal, Canada, June 1998. **PLDI most influential paper award, 2008.**

Wait-free synchronization, Maurice Herlihy, ACM Trans. Program. Lang. Syst. 13, 1 (Jan. 1991), 124-149. **2003 Edsger Dijkstra Prize.**

Transactional memory: architectural support for lock-free synchronization synchronization, Maurice Herlihy and J. Elliot Moss, ISCA 1993, 289-300. **ISCA most influential paper award, 2008.**

Memory model papers by Sarita Adve**, 2008 Maurice Wilkes Award.**

Algorithms for scalable synchronization on shared-memory multiprocessors, John Mellor-Crummey and Michael L. Scott, ACM Trans. Comput. Syst. 9, 1 (Feb. 1991), 21-65. **2006 Edsger Dijkstra Prize**

Recommended Prerequisites

Understanding of computer systems (COMP 320)

Understanding of machine architecture (COMP 425)

See me if you have concerns!

Course Format

Paper Evaluations - 20%

Students must evaluate a paper for each class session. A paper evaluation consists of

- **—your name**
- **—the paper name**
- **—paper summary (~5 sentences)**
- **—most important strengths (no more than 3; 1 sentence each)**
- **—most important weaknesses (no more than 3; 1 sentence each)**
- **—one problem or issue left open (no more than 3 sentences)**

Strengths and weaknesses should be technical issues.

Evaluations will be a completion grade. Unsatisfactory evaluations will not receive credit.

Submit evaluations of papers to the instructor before noon on the day before the papers will be discussed.

Late or incomplete evaluations will receive no credit.

Presentations - 20%

Analyze one or more papers + supporting work

Prepare two or three (depends upon class size) presentations

- **—the motivation for the work**
- **—the key techniques, insights, and/or results**
- **—a critical evaluation of the work**
- **—open issues**

Lead class discussion

Presenter(s) advised, but not required, to show presentations to the instructor in advance

Provide the instructor with electronic version of the presentation suitable for posting on the class WWW page

Class Participation - 20%

Research papers

- —not always well written
- —sometimes make misleading claims
- —occasionally contain errors

Students are expected to contribute to the discussion of the papers in class

- —subject the papers to critical analysis
- —ask questions
- —offer observations

Presentation Evaluations - 10%

Submit brief written evaluations presentations by others

Why?

- **—if writing reviews of presentations, you will pay closer attention**
- **—it will encourage you to think about presentation issues**
- **—if you know that your presentation is being evaluated by your classmates, you may try harder to make it engaging**

A presentation evaluation

- **—vision: how well did the presenters explain why the area matters?**
- **—style: mumbling, fail to make eye contact, too quickly or slowly?**
- **—exposition: were the slides too busy, too ugly, or just right?**
- **—question and answer: how well did the presenters seem to know the material? were they honest about admitting when they didn't know something?**
- **—comments: any additional information that you would like to add**

Final Project - 30%

Explore a topic of interest in greater depth

Requirement: one of the following

- **—a term paper (must be done individually)**
  - **may focus on the same topic as one of your presentations**
  - **you should study different papers**
- **—a final project (which may be a group), written project report**
  - **group projects will submit a single writeup**
  - **writeup will include a description of each member's contribution to the project**

References

John Markoff. Advanced Micro Plans to Show New Chip Ahead of Intel. Technology, New York Times, August 31, 2004.

John Markoff. Intel's Big Shift After Hitting Technical Wall. Technology, New York Times, May 17, 2004.

Laurie J. Flynn. Sun Microsystems Will Offer New Generation of Processors, New York Times, November 14, 2005.

Ron Kalla. POWER5: IBM’s Next Generation POWER Microprocessor, Hot Chips 15, August 2003.

Herb Sutter and James Larus. Software and the Concurrency Revolution, ACM Queue Special Issue on Multiprocessors, 3(7), September, 2005.

Maurice Herlihy. The Transactional Manifesto. Keynote address at PLDI 2005.