ADVANCED PARALLEL COMPUTING 2017
LECTURE 01 - INTRODUCTION

Holger Fröning
holger.froening@ziti.uni-heidelberg.de
Institute of Computer Engineering
Ruprecht-Karls University of Heidelberg

ABOUT ME

JProf. Dr. Holger Fröning

- PI of Computer Engineering Group @ ZITI
- http://www.ziti.uni-heidelberg.de/compeng

Research: Application-specific computing under hard power and energy constraints (HW/SW)

- GPU computing (heterogeneity & massive concurrency)
- High-performance analytics (machine learning, scalable graph computations)
- High-performance computing (traditional and emerging)
- Resource-constrained devices (digital design, high-level synthesis)
- Emerging technologies (approximate computing and stacked memory)

OBJECTIVES & PREREQUISITES

Objectives: The students …

- understand advanced concepts and principles of parallel architectures
- will be able to develop optimized programs for parallel architectures
- can use the learned structures to develop new architectures

Methodology

- Lectures focus on current architectures, trends, technology constraints
- Exercises: Practical hands-on, reading
- Excessive programming & paper reviewing

Prerequisites

- Required: Solid knowledge about computer architecture and C language
- Recommended: Course „Introduction to HPC“ & GPU Computing“

ORGANIZATION

Lectures – 2 hours/week - holger.froening@ziti.uni-heidelberg.de

- Tuesday, 14:00 ct

Exercises – 2 hours/week - d.zosgornik@stud.uni-heidelberg.de

- Tuesday, 16:00 ct
- Groups of up to two students allowed – individual work must be visible
- Mixture of reading/exercises/programming/experiments
- Second half of term will be project work

One final oral or written exam

- Prerequisite: min. 50% of all exercise points

6 CPs

ASSIGNMENTS

Practical exercises: usually coding & experiments

Reading & feedback based on paper review

- Ideal review here is 3 sentences for each of the following:
  1. Primary contribution
  2. Key insight of the contribution
  3. Your opinion/reaction to the content
- Review: rating relative to all other papers (of this venue)
  - Strong reject, weak reject, weak accept, accept
- Old papers: optionally include some comments on how right this paper was

Provide review (summary) using Moodle until next Tuesday

- Discussion round as part of the exercise

ADDITIONAL READING

Books

Culler et al., Parallel Computer Architecture: A Hardware/Software Approach, Morgan Kaufmann

Hennessy/Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufmann

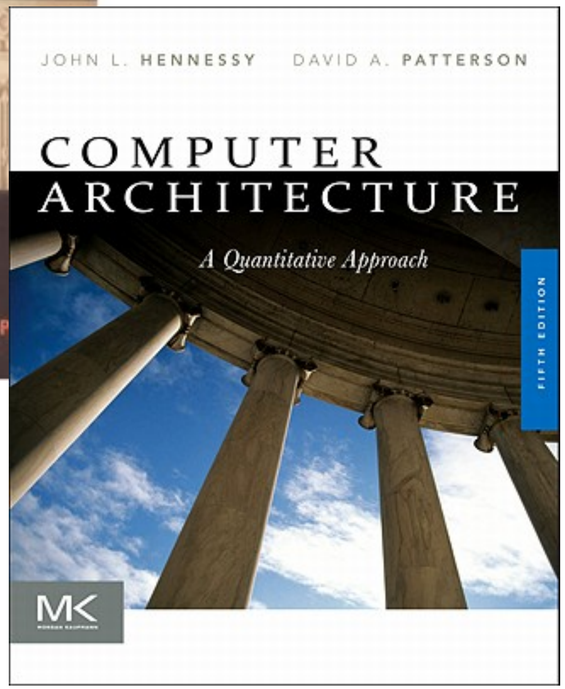
(Keckler et al., Multicore Processors and Systems, Springer)

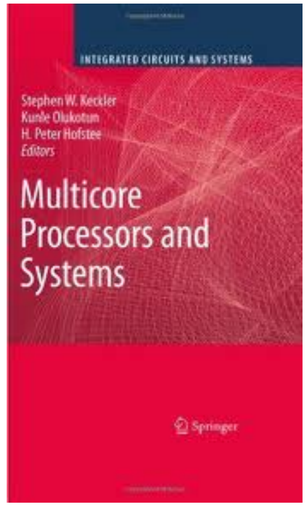
Publications/Conferences

ISCA, HDCA, ASPLOS, PACT, IPDPS, ICPP, ISPASS, HPDC, ...

See ACM/IEEE websites (or author‘s web site for limited copies)

Google Scholar and similar





QUESTIONNAIRE

Amdahl’s law

Pthread & OpenMP

Difference between coherence and consistency

Intel MIC

Exponential back-off

Dennard scaling

Scratchpad memory

Victim cache

Data-race free

Transactional memory

CURRENT COMPUTING TRENDS

PARALLEL COMPUTING - DEFINITION

A collection of processing elements that communicate and cooperate to solve large problems fast.” - Almasi & Gottlieb, 1989

Issues

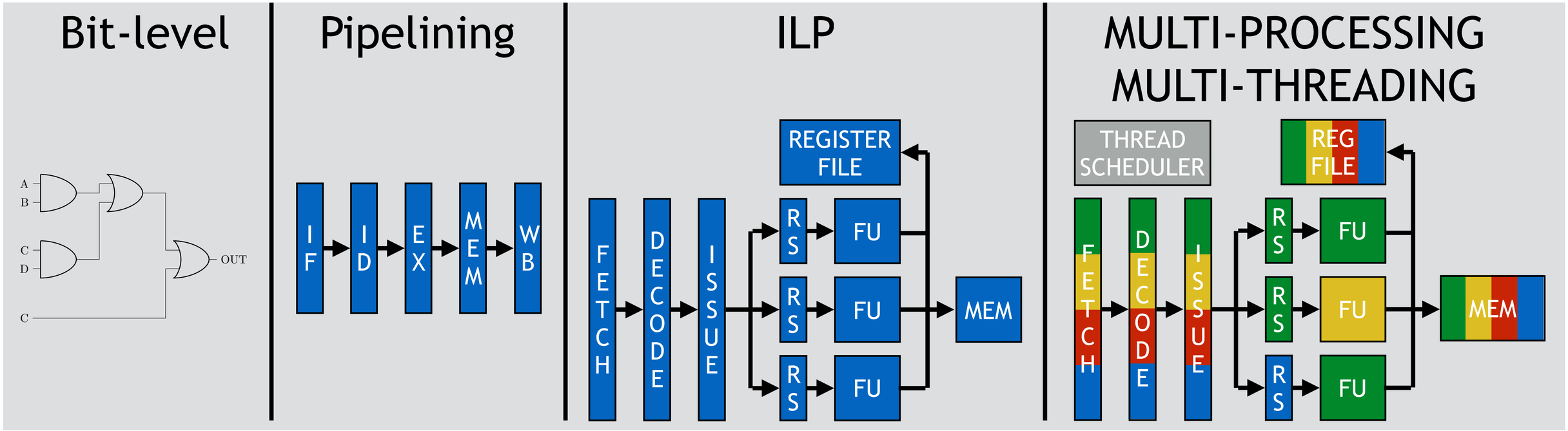
- How large/powerful/scalable?
- Methods for communication/synchronization?
- Interconnection network and operations?
- Primitive abstractions for the programmer?

Note that this definition gears to increase performance

- Other reasons: availability (not covered in this course)

Keys: concurrency and parallelism

DIFFERENT LEVELS OF PARALLELISM



Main motivation: performance!

Limited amounts of parallelism

Coarse grain parallelism scales better

DIFFERENT LEVELS OF PARALLELISM

Instruction-Level Parallelism (ILP) ==> Pipelining, Superscalar

- Parallelism within one instruction stream
- Limited parallelism
- Huge amount of dependencies and branches

Thread-Level Parallelism (TLP) ==> SMT, Multi-Core, Cluster

- Parallelism in multiple independent instruction streams
- Less amount of dependencies, no limitations due to branches
- Limited by maximal concurrently executable I-streams

Data-Level Parallelism (DLP) ==> GPUs

- Vectorization techniques
- Applying one operation on multiple elements of a data structure
- Parallelism dependent on data structure

Request-Level Parallelism (RLP) ==> Datacenters & WSC

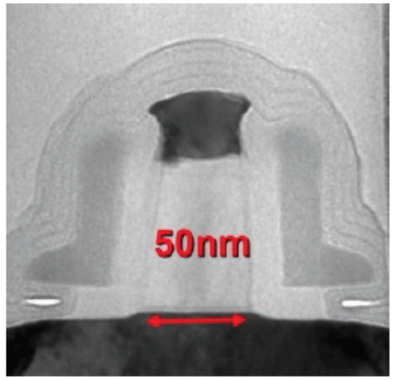
- Huge amount of concurrent requests
- Unstructured, randomized patterns

Basics of computer architecture

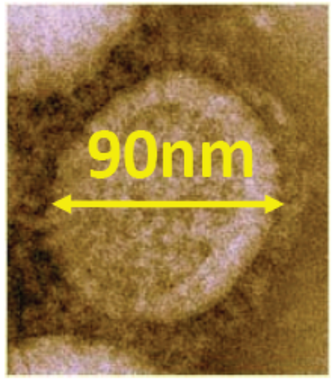
Course “Parallel Computer Architecture”

Course “GPU Computing”

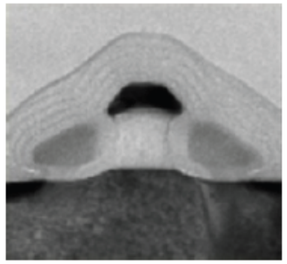
OBSERVATION: TRANSISTOR COUNT



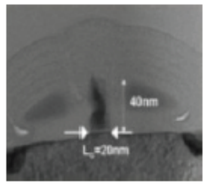
90nm transistor
(Intel, 2005)



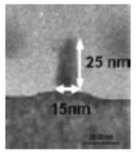
Swine Flu A/H1N1
(CDC)



65nm
2007

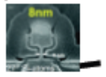


45nm
2010

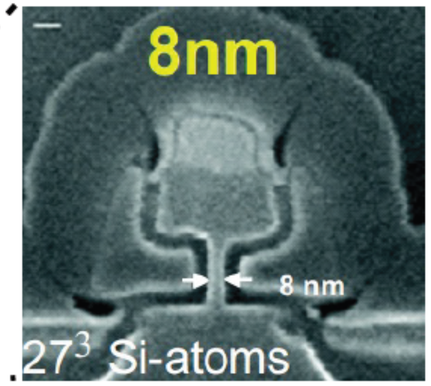


32nm
2013

22nm
2016



16nm
2019



T. Wenisch

Moore's law will continue for another decade

OBSERVATION: ILP WALL

Applications don‘t contain enough ILP

IPC for 6-issue is higher, but not 3x in comparison to 2-issue

Main limitations: Stalls due to cache misses and dependencies

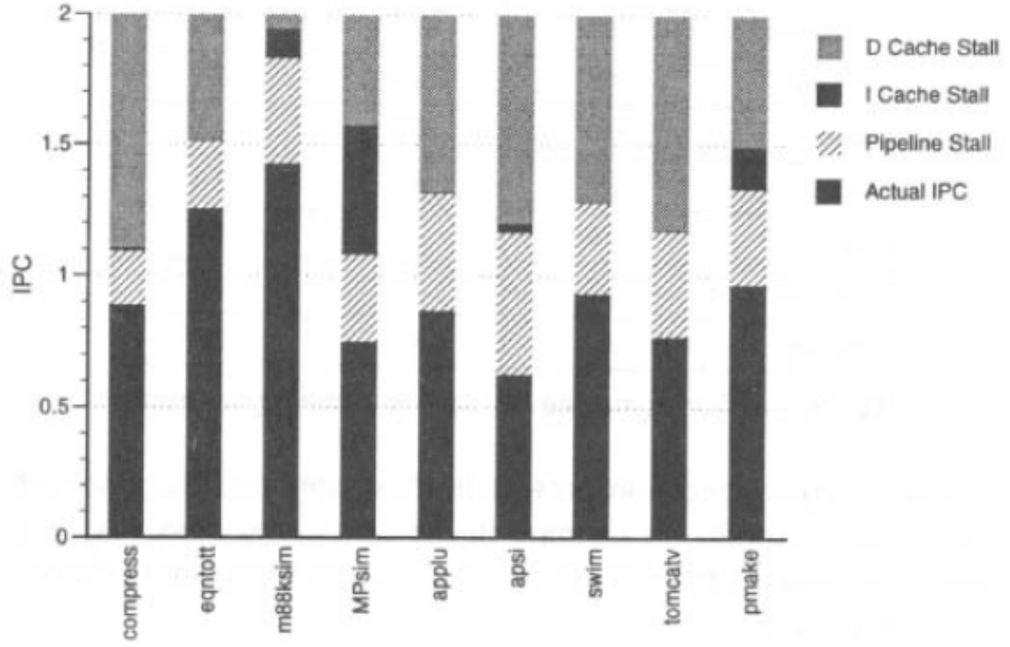


Figure 4. IPC Breakdown for a single 2-issue processor.

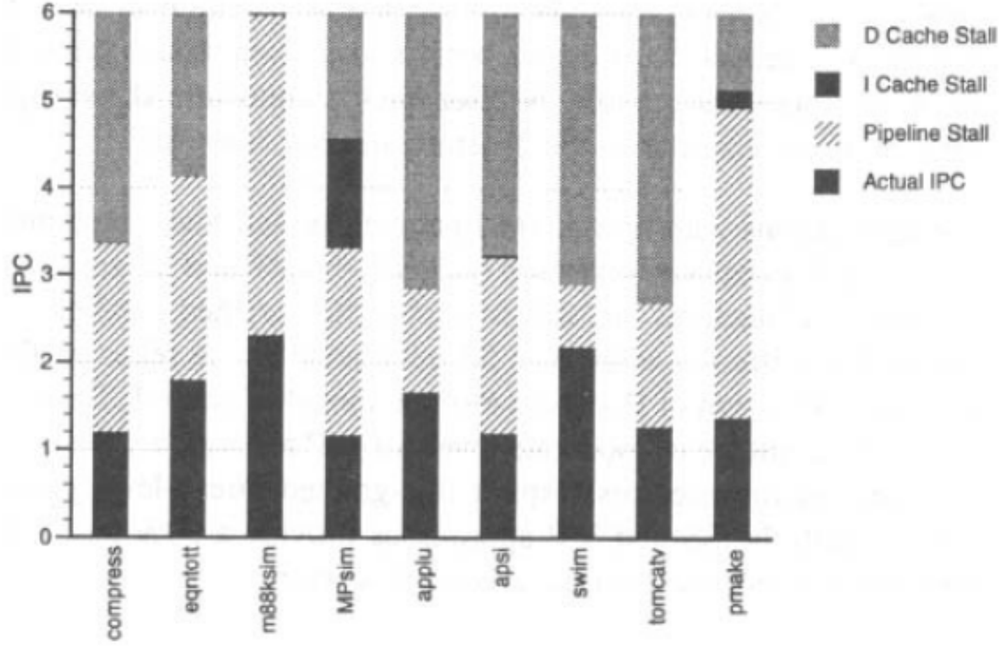


Figure 5. IPC Breakdown for the 6-issue processor.

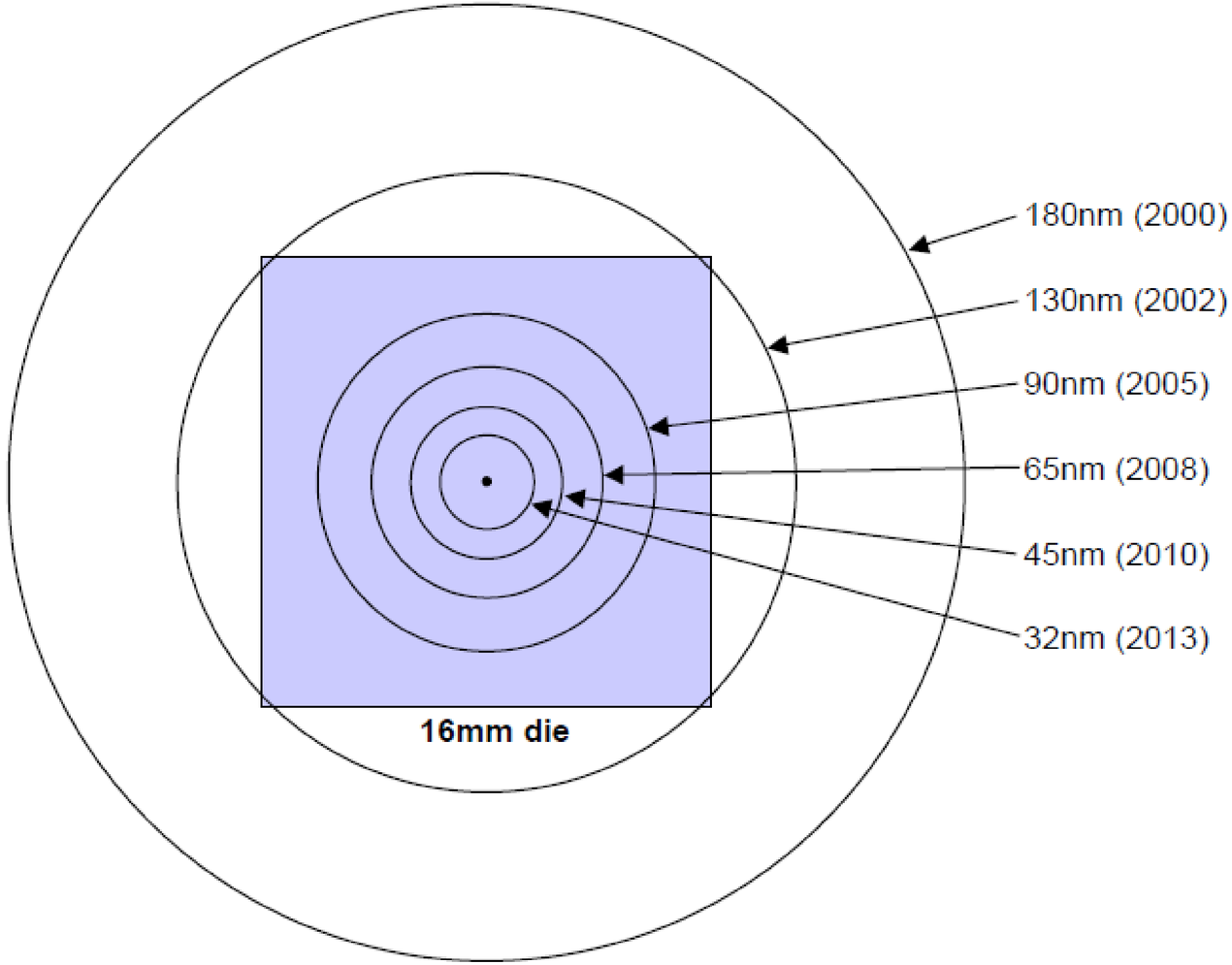
Olukotun et al. The Case for a Single-Chip Multiprocessor. ASPLOS1996

OBSERVATION: LIMITED SIGNAL REACH

Propagation delay increases, frequency increases/increased

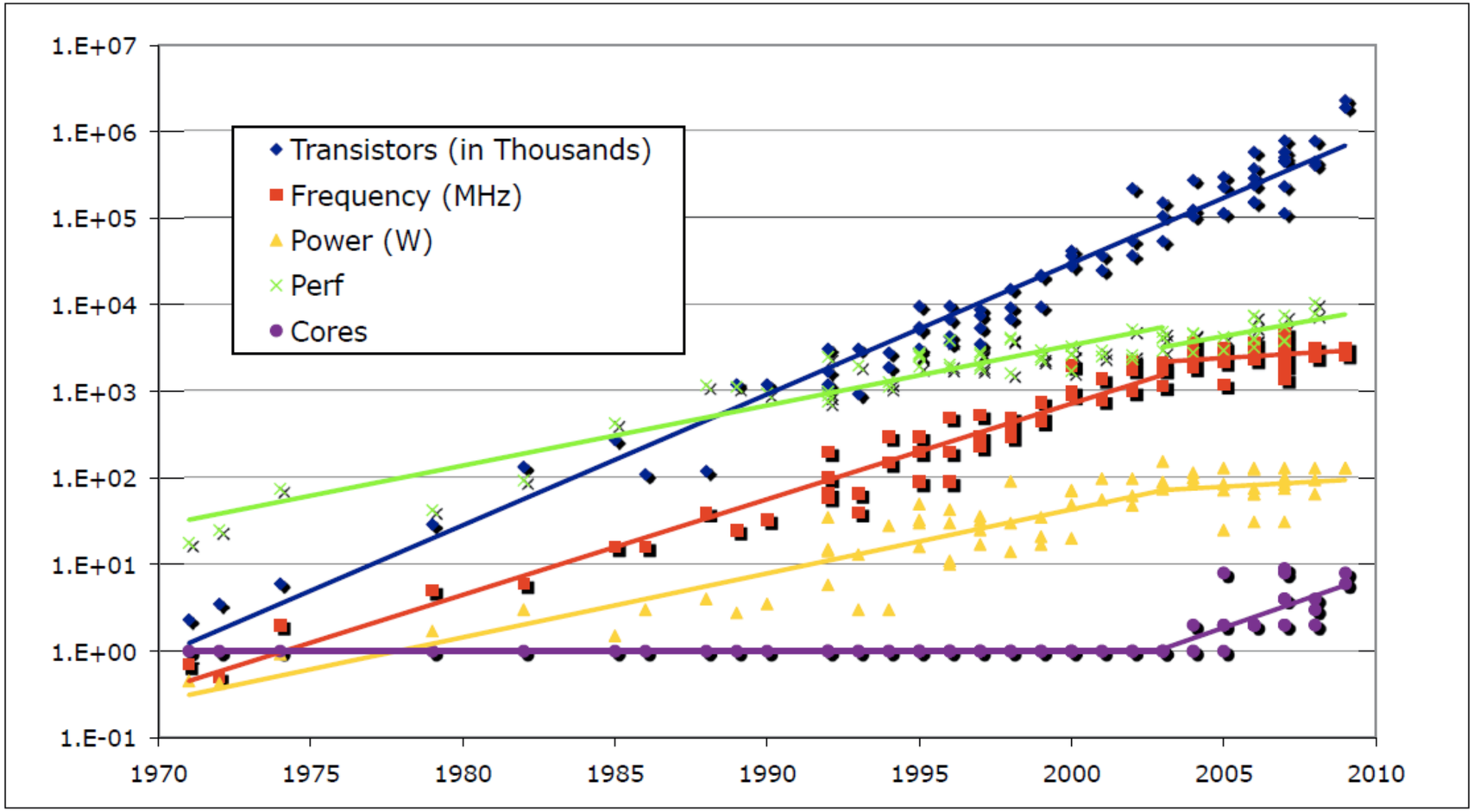
=> Signal speed decreases

=> Limited signal reach



Signal reach in one clock cycle (8 FO4) *S. Scott, 2005*

OBSERVATION: POWER WALL



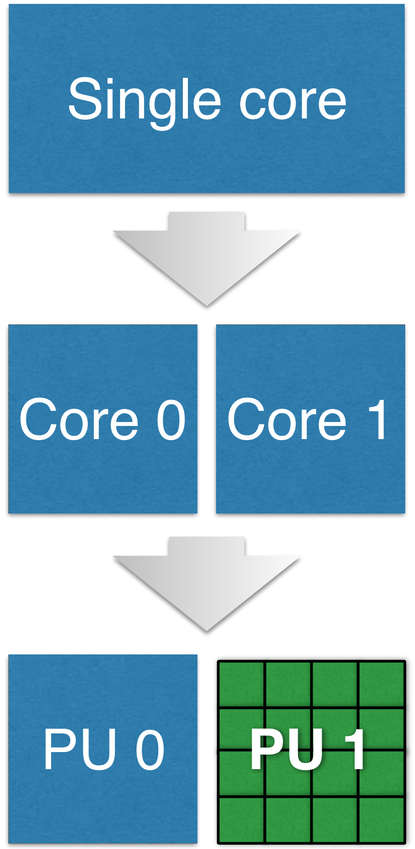
K. Yelick

Inflection point in 2005 - end of Dennard scaling

THE FUNDAMENTAL TRANSITION TO MULTI-CORE

$$P = afCV^2 + VI_{leakage}$$

- Transition to multi-/many-core
  - Power = aCV²f
    - a constant, f ~ V
  - Given a single-core design
    - Reduce clock to 80%
    - Power ~ f³ => now at 51.2%
  - Add second core
    - Assume that single-core performance ~ f
  - Same power budget (envelope), but 1.6x performance
- Transition to asymmetry/heterogeneity
  - Specialization by massive parallelization



OVERVIEW OF PARALLEL ARCHITECTURES

PARALLEL PROGRAMMING MODELS

Programming Model: language & libraries that define an abstract view of a machine

Control – expose parallelism

- How parallelism is created
- How are dependencies solved?

Data – communication, placement and partitioning

- Shared/Private
- Accessibility

Concurrency control - synchronization

- How to coordinate/orchestrate?
- Atomic operations to ensure mutual exclusion

ARCHITECTURE FOUNDATIONS

Naming

How is data communicated?

How are control flows addressed?

Operations

What operations are allowed on named data?

Ordering

How can control flows coordinate their activities?

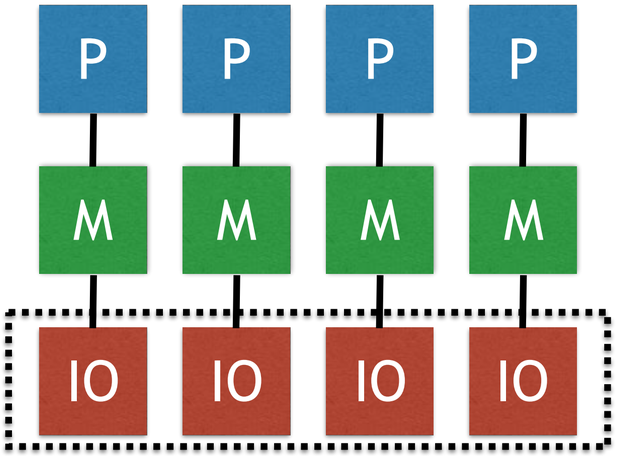
Performance

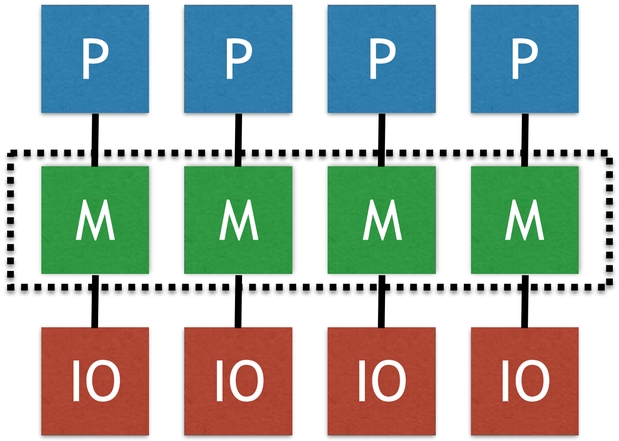
Latency and Bandwidth

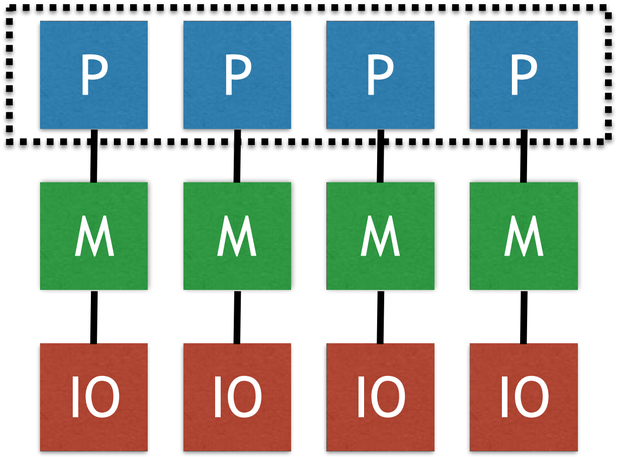
Data races / Race condition

A race condition or race hazard is a flaw in an electronic system or process whereby the output or result of the process is unexpectedly and critically dependent on the sequence or timing of other events. The term originates with the idea of two signals racing each other to influence the output first. (wikipedia.org)

ARCHITECTURE OVERVIEW





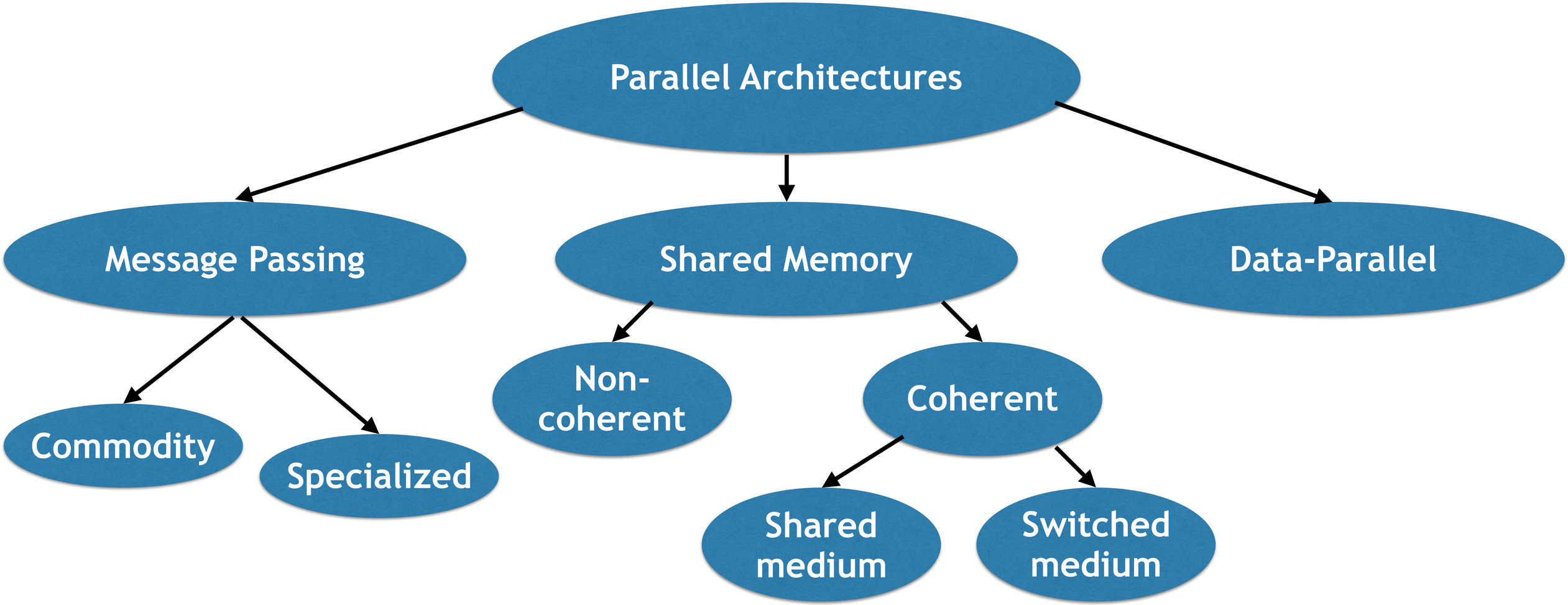


IO: Message passing

Memory: Shared Memory

Processor: SIMD, VLIW, CUDA, dataflow architectures

ARCHITECTURE TAXONOMY



MULTI-/MANY-CORE TREND

Moore‘s law gives us plenty of transistors, question is how to use them

- Memory ruled out as DRAM technology is different
- ILP is too limited, no big uniprocessor
- Big caches suffer from increased hit times, increasing average memory access time
- Thus: use them for caches or for processing cores?

Chip Multi-Processors (CMP, aka multi-core)

- Foundation of today‘s computing arena, making parallel programming pervasive

Data-Parallel Architectures (vector processors, GPUs, MICs)

- Specialized styles of computing for improved performance & energy efficiency

REST OF THIS COURSE

Shared Memory Architectures: programming models, fundamental design issues, coherence vs consistency

Snooping Coherence

Synchronization: basics, locks, barriers, optimizations

Transactional Memory: hardware/software

Scalable Coherence: basics, probe filter, COMA, token coherence

Memory Consistency: strong & weak models

Advanced concepts: shared virtual memory, multi-threading architectures

Future: NUCA, MIC, trends

SUMMARY

This course

Post-Dennard I: concurrency & specialization in compute (and memory)

- Strong NUMA effects, effective tolerance techniques
- Implications have yet only marginally been addressed

Post-Dennard II: communication-centric systems

- Communication more expensive than compute
- Energy is fundamental, no hiding possible
- Strong and even dynamic NUMA effects

Post-CMOS: specialization & complexity