ADVANCED PARALLEL COMPUTING LECTURE 01 - INTRODUCTION

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

ABOUT ME & THIS COURSE

Professor at Heidelberg University, Germany

- Currently advising 9 PhDs, 4 being external (3 on ML)
- PhD from Mannheim University
- Post-Doc, TU Valencia & Heidelberg University (FPGAs, ASICs)
- Visiting professor, TU Graz (2015 - first contact with ML)
- Visiting scientist, NVIDIA Research (Santa Clara, CA) (2016)

Performance, energy efficiency & programmability for future & emerging technologies

- Computer engineers/architects with a focus on low-level software layers
- High-performance computing, machine learning & data analytics
- Bipolar computing landscape: super small & super big

This course

- Basics and advanced methods for scalable multi-core computing systems
- Research-driven & forward-looking

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 - bernhard.klein@ziti.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: see exercise announcement

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

Project assignment for second half of the term (tba)

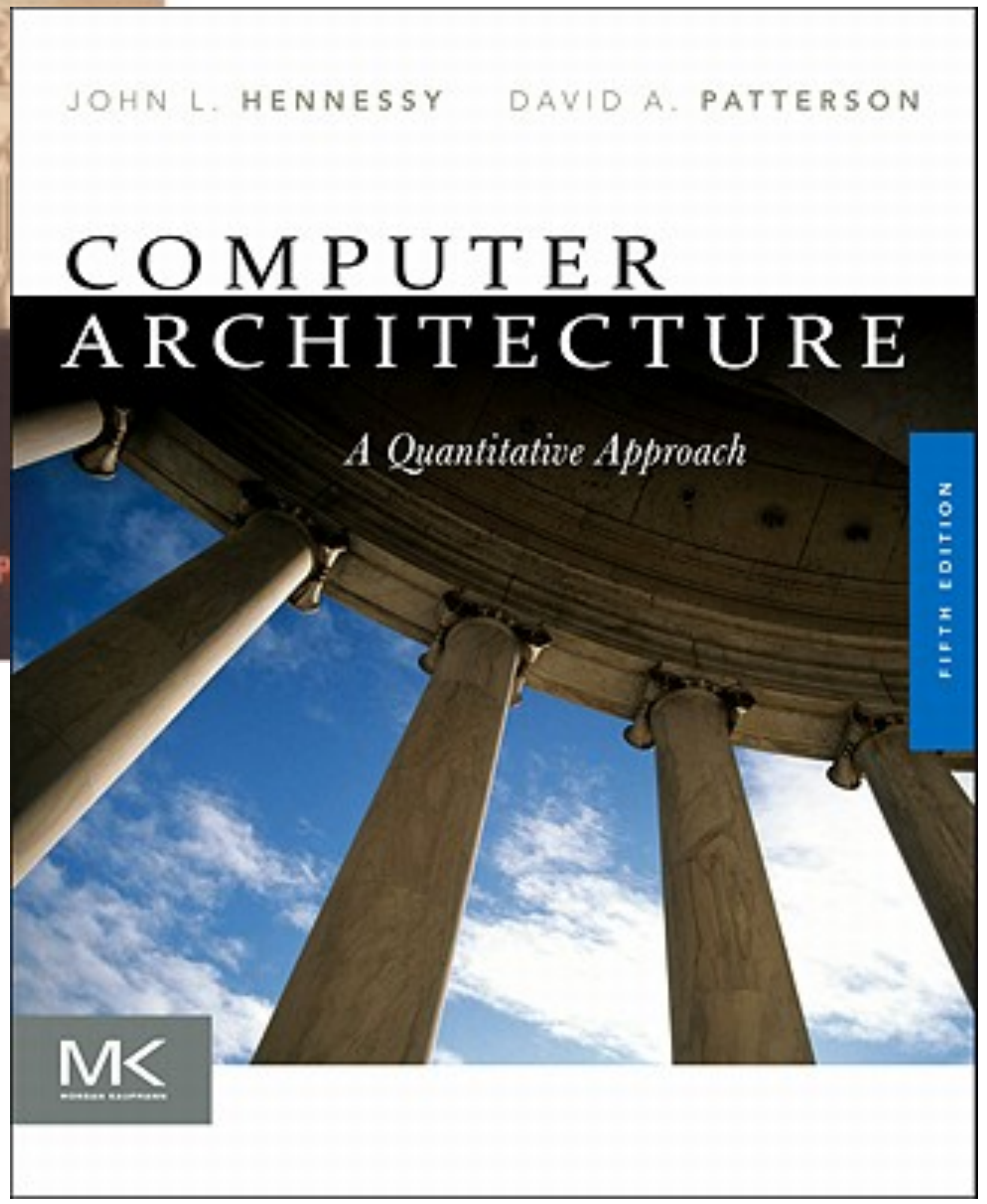
ADDITIONAL READING

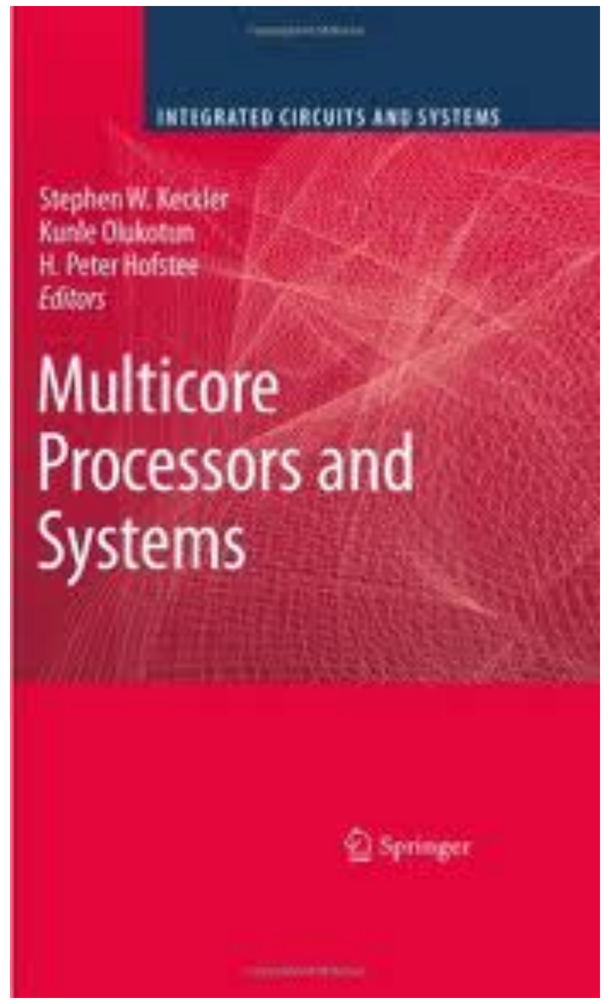
Books

- 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

Pthread & OpenMP

Amdahl's law

Difference between coherence and consistency

Intel MIC

Exponential back-off

Dennard scaling

Victim cache

Data-race free

Cache conflict

CURRENT COMPUTING TRENDS

PARALLEL COMPUTING - DEFINITION

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

Issues

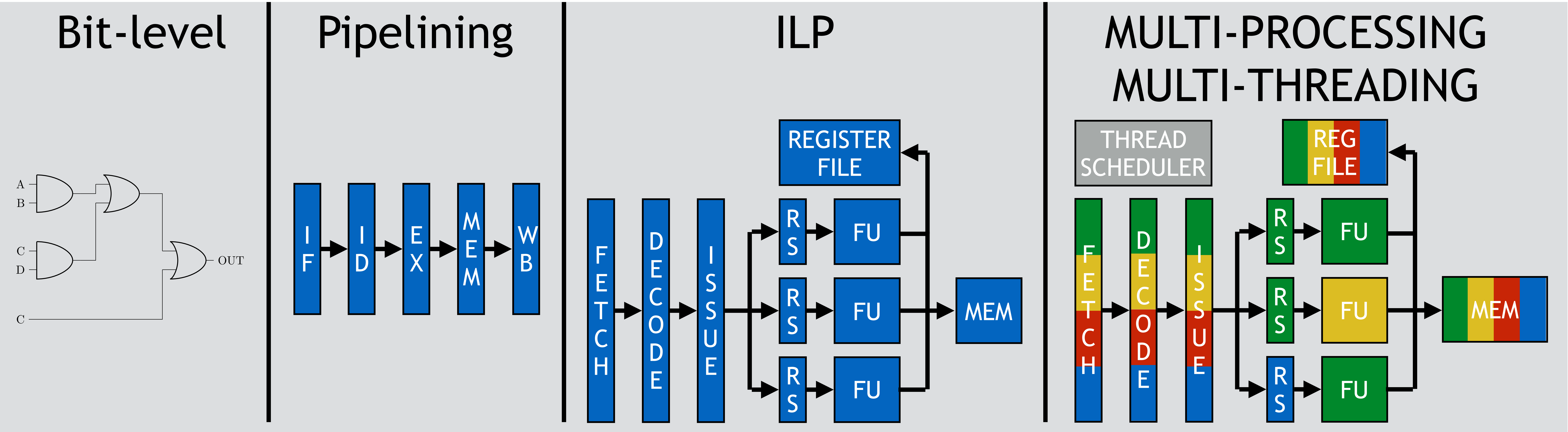
- How large/powerful/scalable?
- Methods for communication/synchronization?
- Interconnection network and operations?
- Primitive abstractions for the programmer?

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

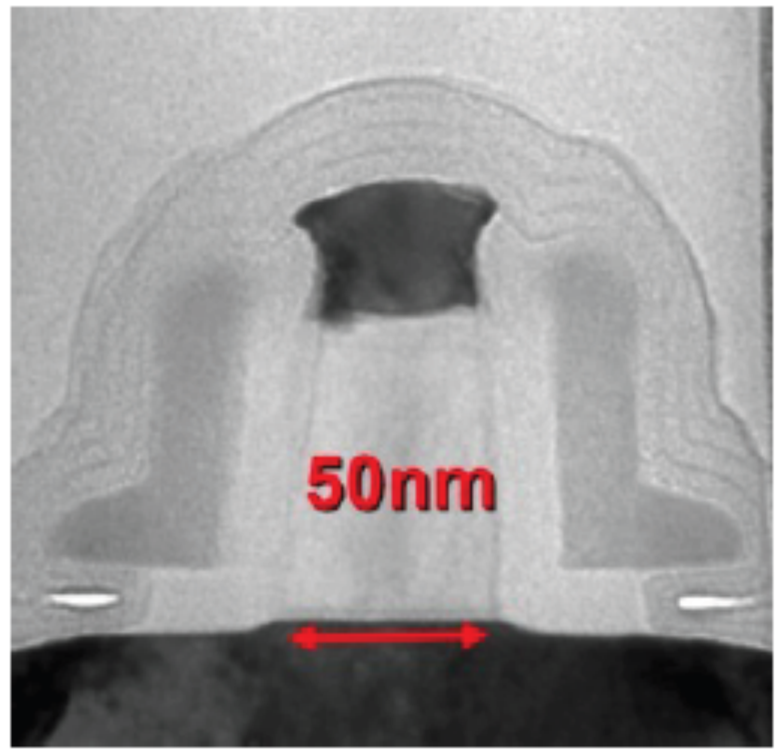
- Huge amount of concurrent requests
- Unstructured, randomized patterns

Basics of computer architecture

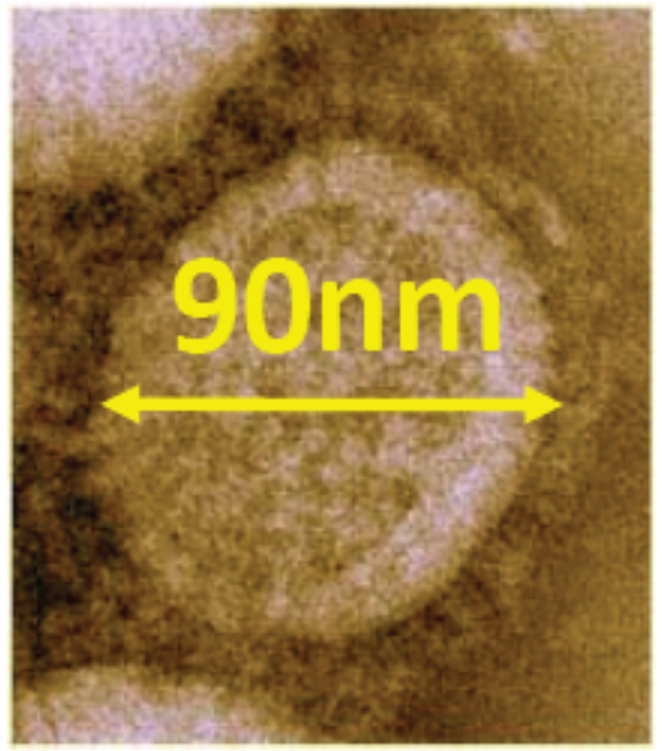
Course “Parallel Computer Architecture”, this course

Course “GPU Computing”

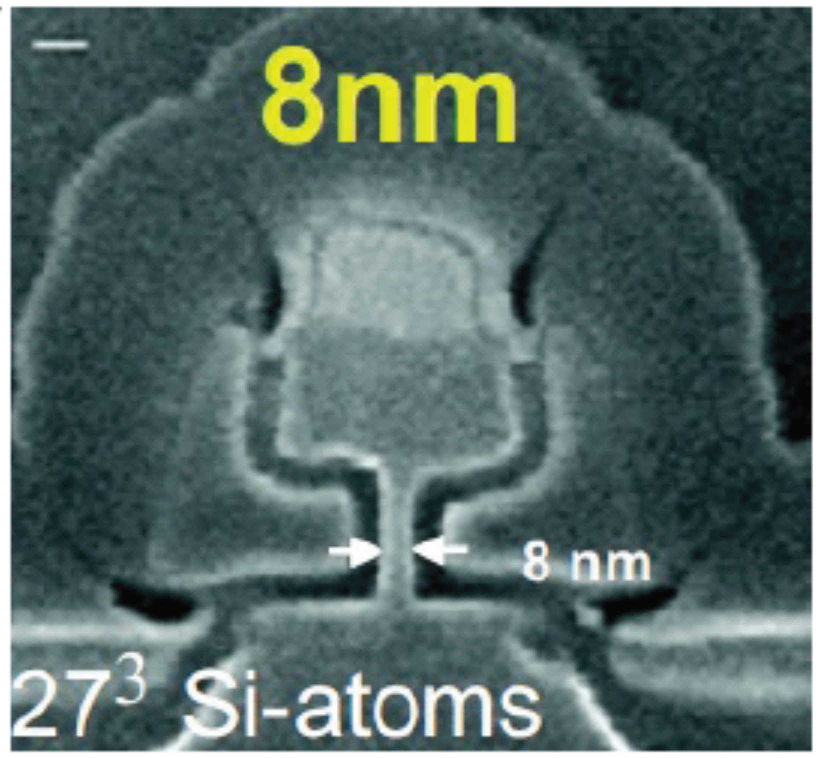
OBSERVATION: TRANSISTOR COUNT

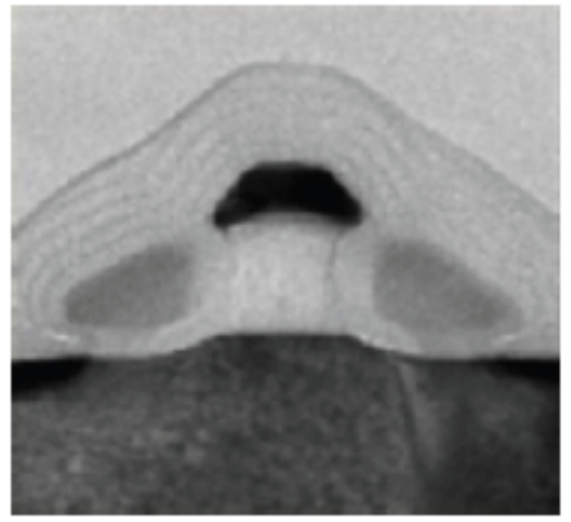


90nm transistor
(Intel, 2005)

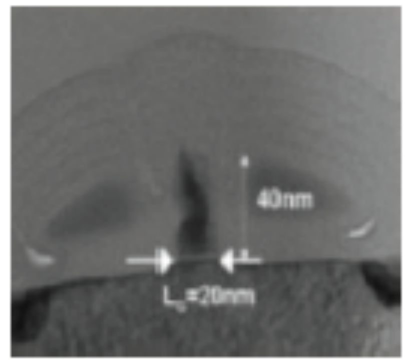


Swine Flu A/H1N1
(CDC)

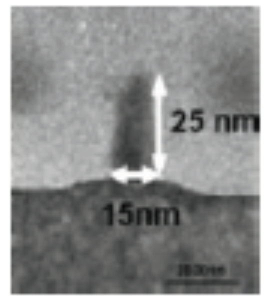




65nm
2007

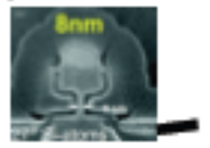


45nm
2010



32nm
2013

22nm
2016



16nm
2019

T. Wenisch

Moore's law will probably 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

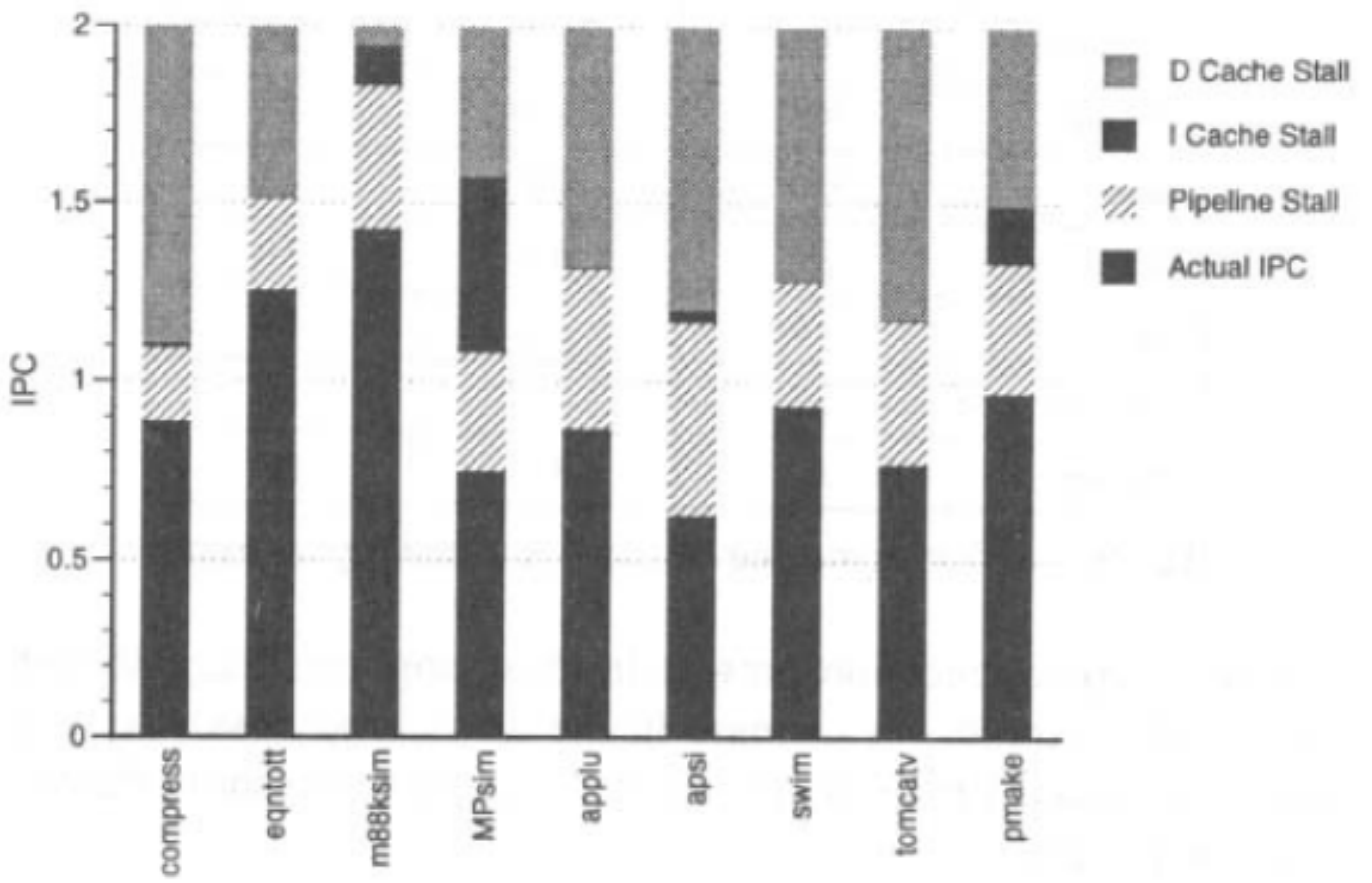


Figure 4. IPC Breakdown for a single 2-issue processor.

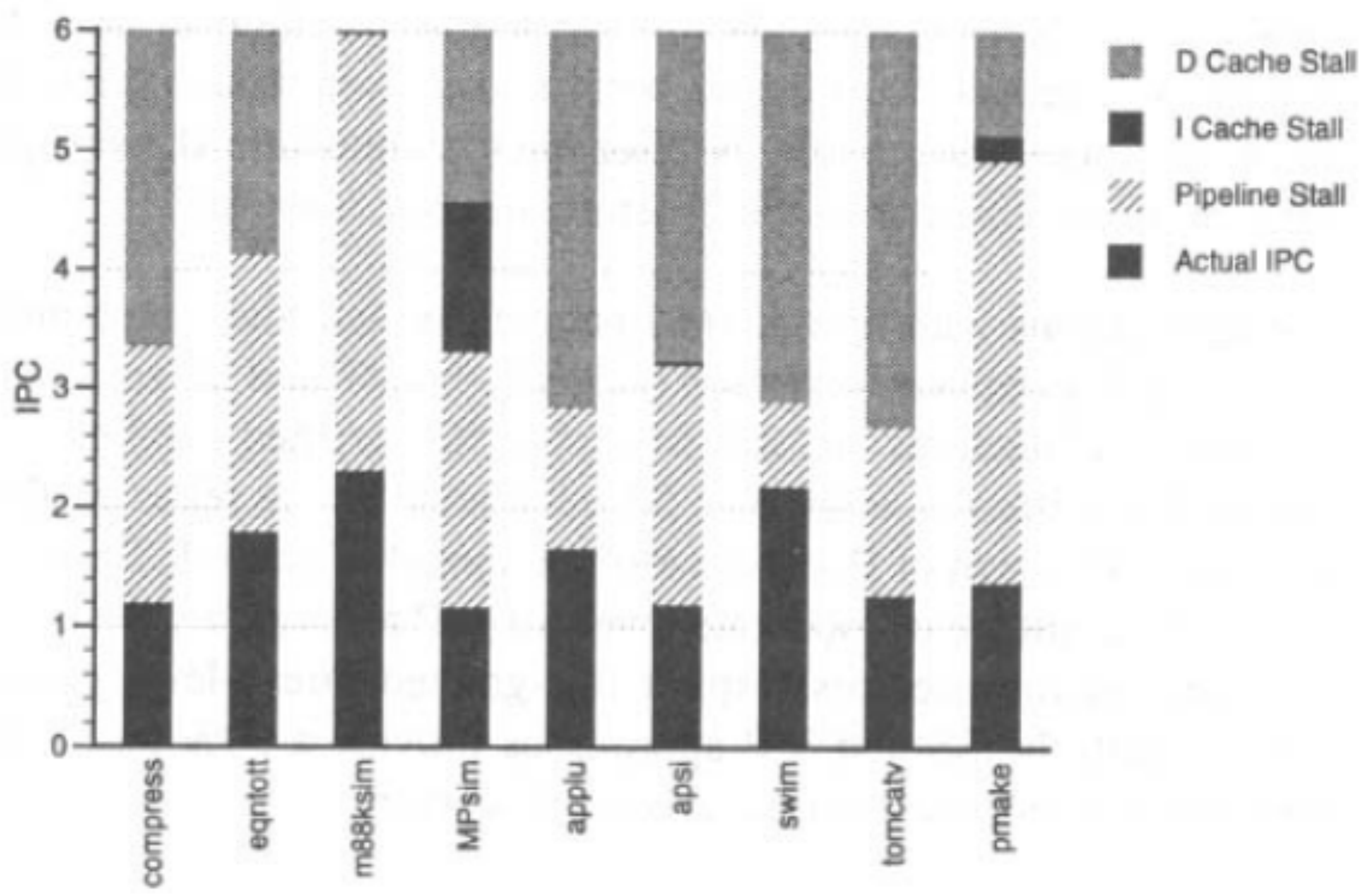


Figure 5. IPC Breakdown for the 6-issue processor.

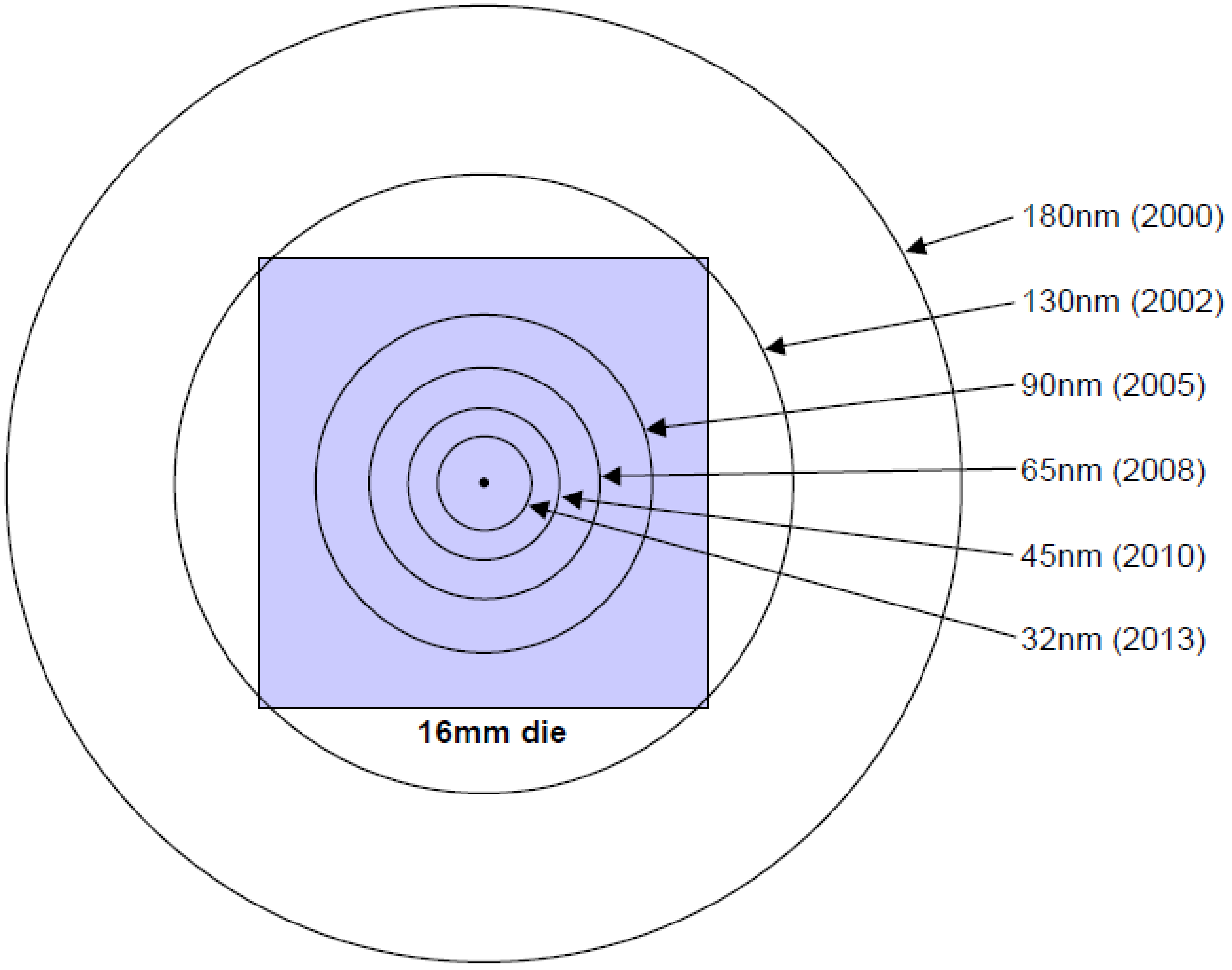
Olukotun et al. The Case for a Single-Chip Multiprocessor. ASPLOS1996

OBSERVATION: LIMITED SIGNAL REACH

Propagation delay increases, frequency increases/increased

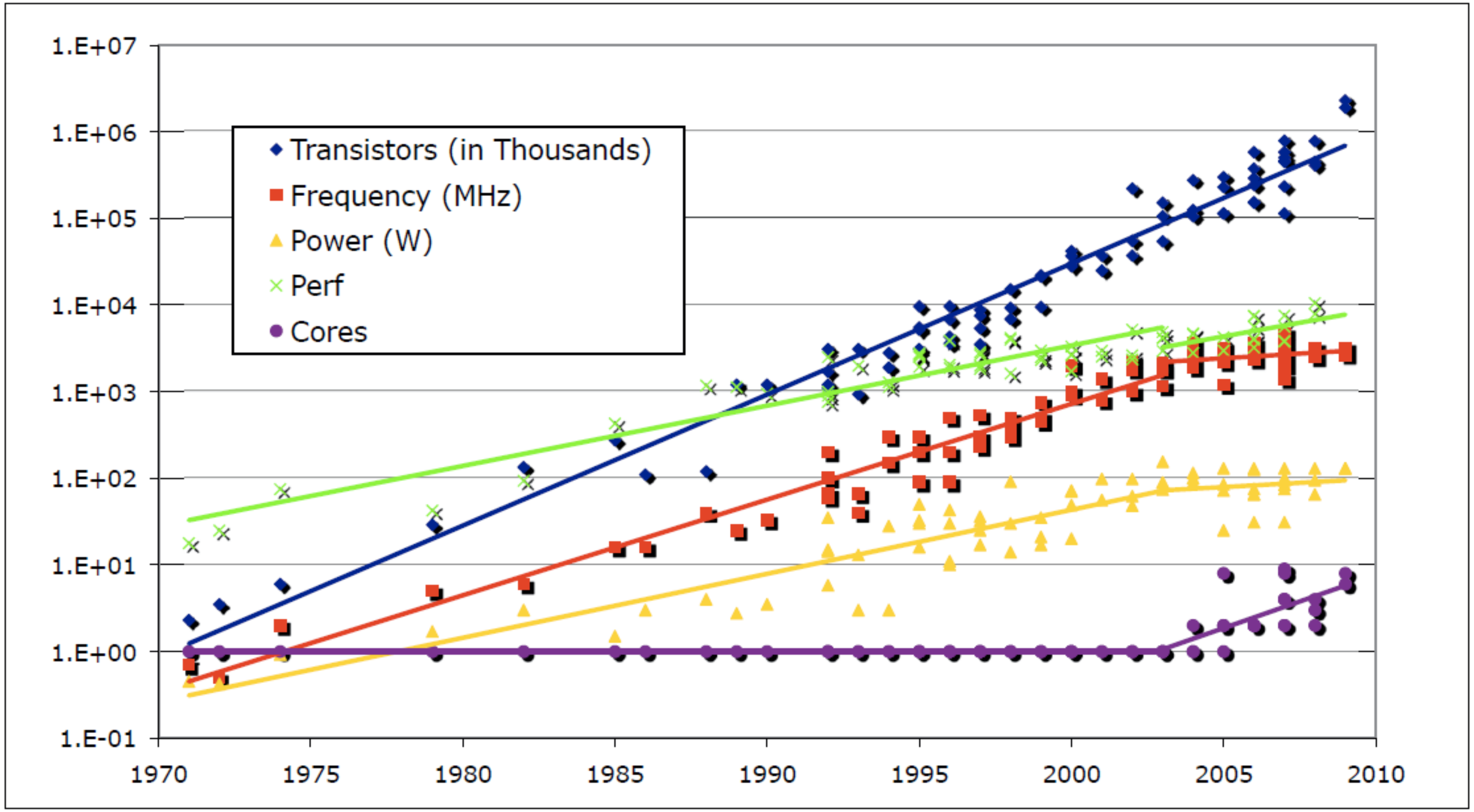
=> Signal speed decreases

=> 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

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

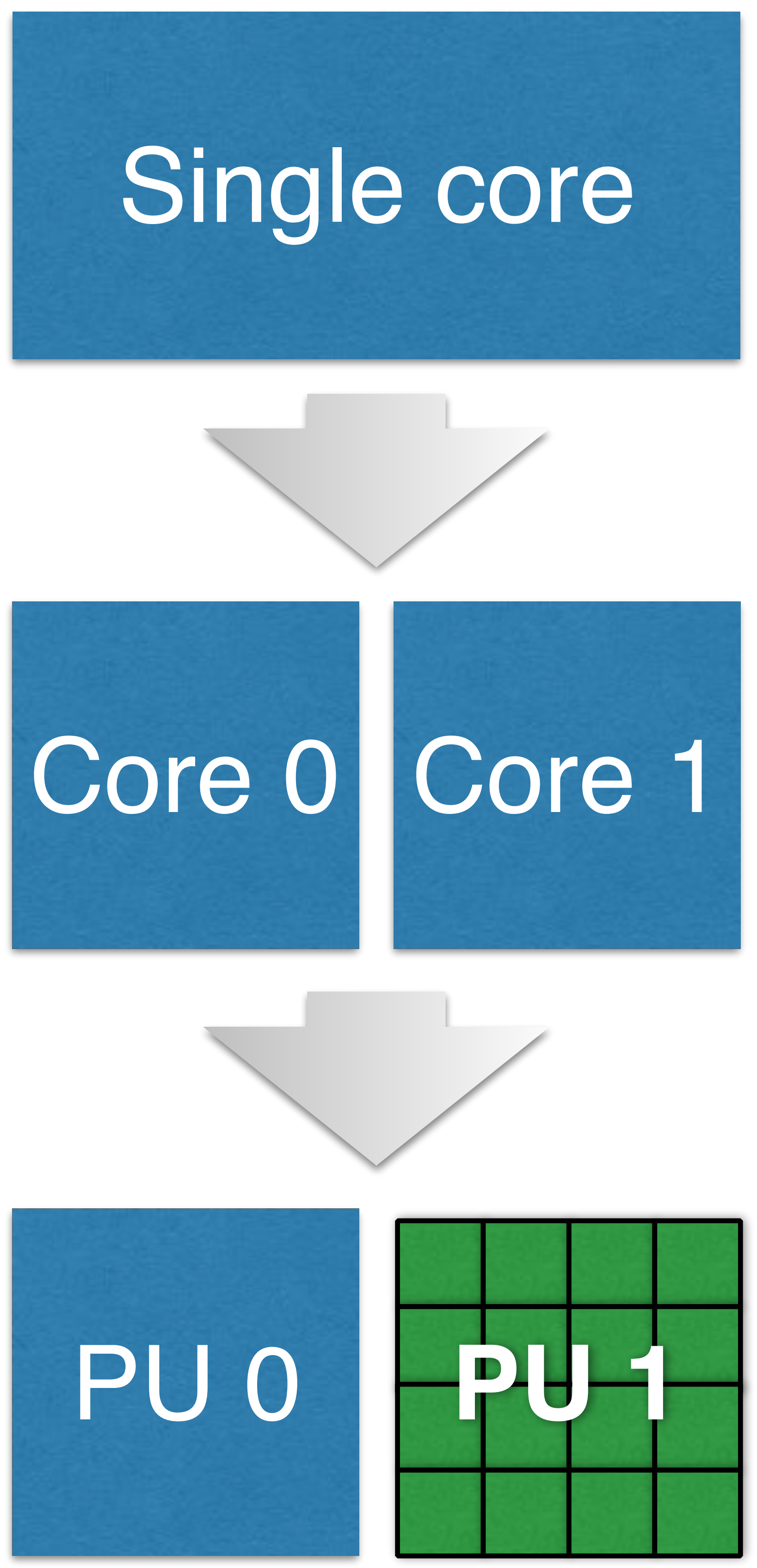
a constant, f ~ V

Transition to multi-/many-core

- Given a single-core design
  - Reduce clock to 80%
  - Power ~ f^3 => 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?

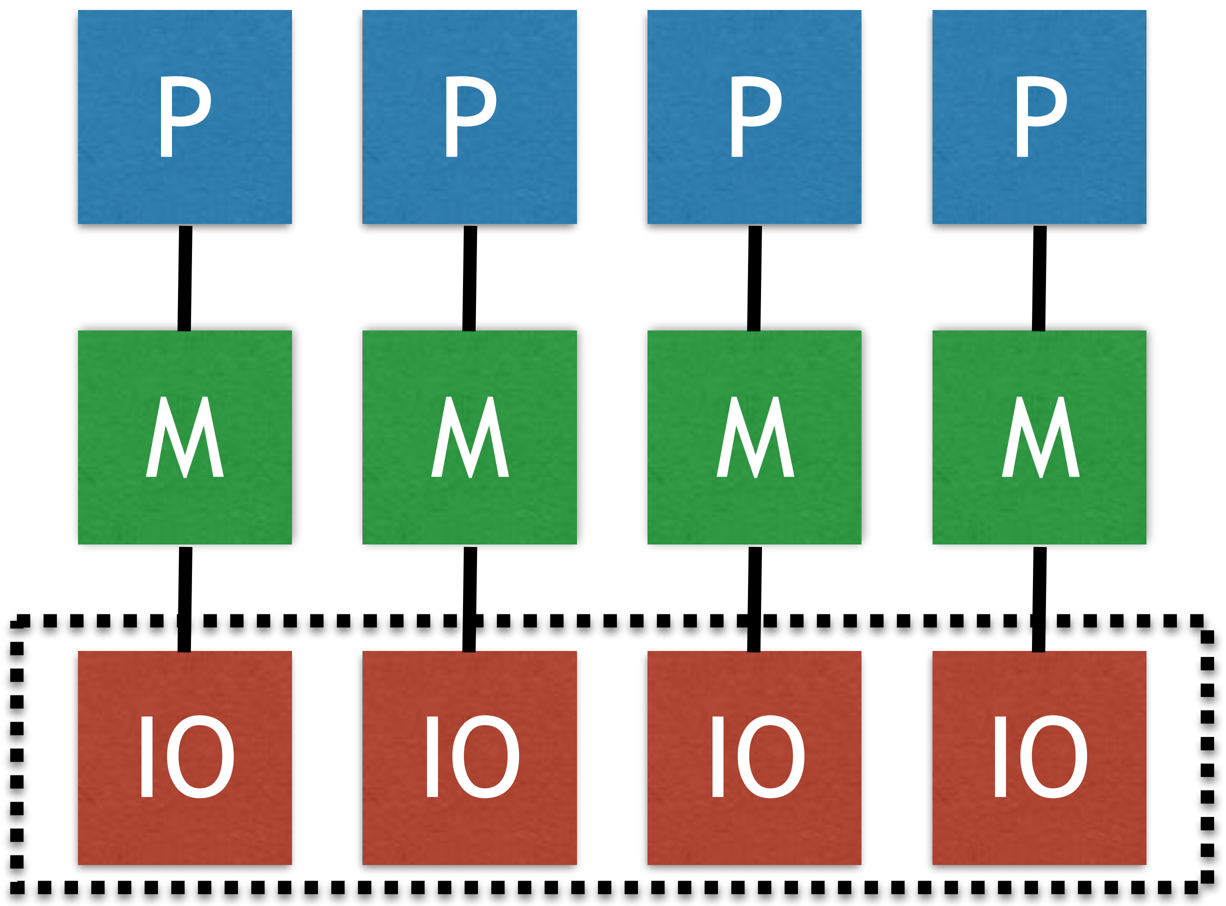
Performance

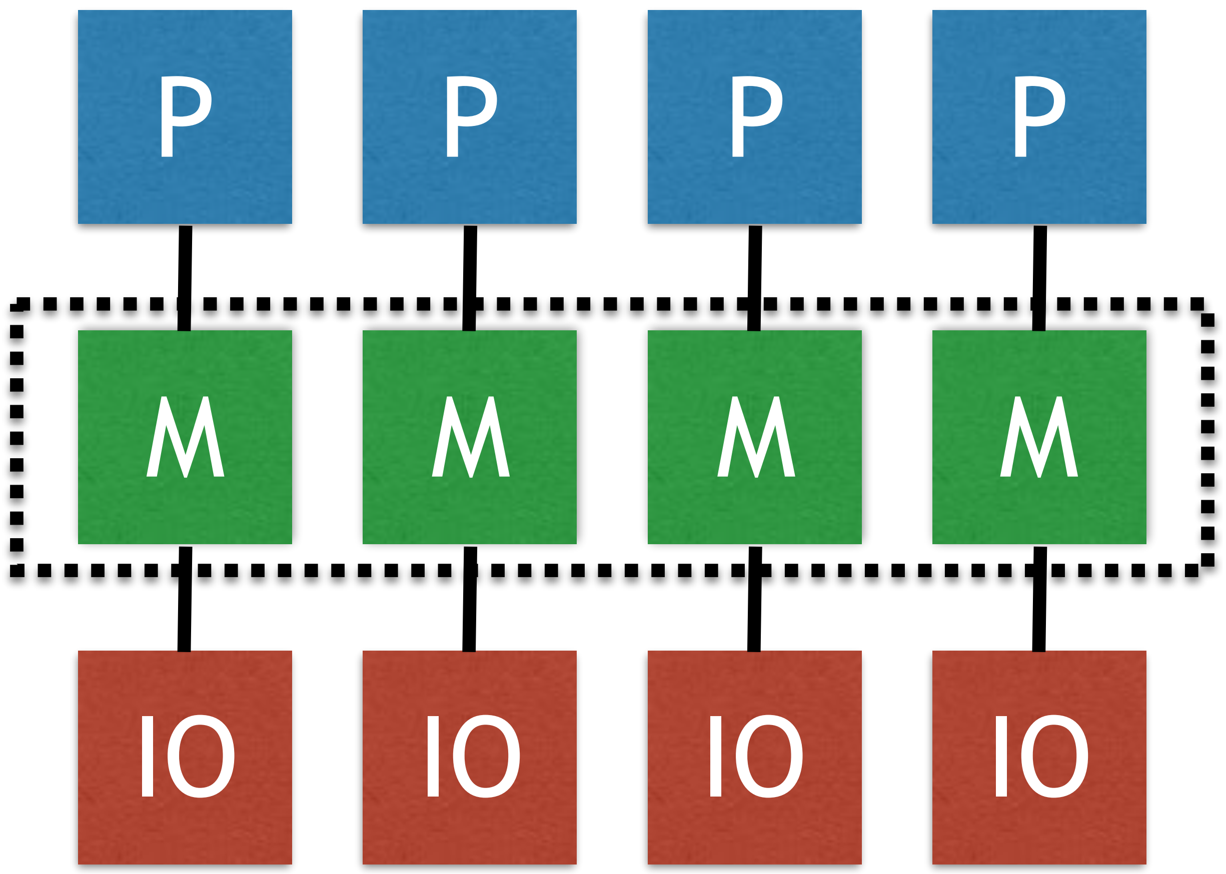
- Latency and Bandwidth

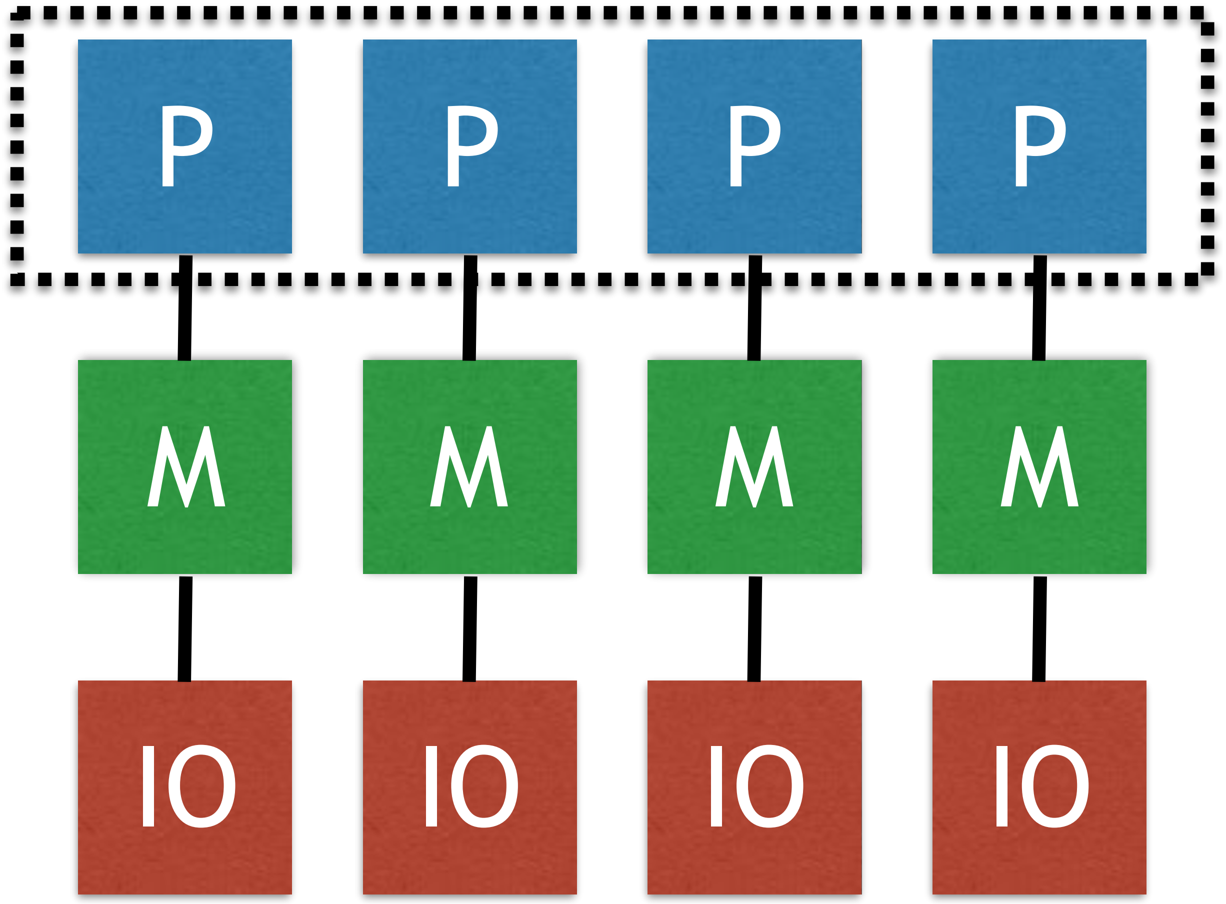
Data races / Race condition

- A race condition is a flaw in a computing system as the result of an unexpected and critically dependent sequence of events.

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

SCHEDULE

Date	Lecture	Exercise
23.04.19	Intro	Reading
30.04.19	Shared Memory Architectures	Pointer Chasing
07.05.19	Snooping coherence	Shared counter (manual lock), atomics
14.05.19	Synchronization I: locks	MCS lock
21.05.19	Synchronization II: barriers	Barrier
28.05.19	Transactional memory	Parallel Prefix Scan
04.06.19	Scalable coherence	Project work
11.06.19	Advanced coherence techniques	-
18.06.19	Memory consistency	-
25.06.19	Applied consistency	-
02.07.19	ML introduction & optimizations	-
09.07.19	Trends & Future	-
16.07.19	<place holder>	Final presentation
23.07.19	Exam	-

SUMMARY

This course

<u>Post-Dennard I: concurrency & specialization in compute (and memory)</u>

- 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