CS4/MSc Parallel Architectures

Dr. Vijay Nagarajan

Institute for Computing Systems Architecture

Parallel Architectures

- How to build computers that execute tasks concurrently
  - Tasks can be instructions, methods, threads, programs etc.

- How to provide support for coordination and communication
  - interconnection networks, coherence protocols, memory consistency model, synchronisation instructions, transactional memory etc.

Parallel Architectures: Why?

- Be a good (systems) programmer
  - Most computers today are parallel (supercomputers, datacentres, even mobile phones), need to understand them if you want to program them well!

- Research future computer architectures and systems
  - A job in Intel, ARM, platform/ infrastructure job at Google, Microsoft, Amazon etc.
  - Academic researcher.

- Appreciate other related courses
  - PPLS, Extreme computing etc.

General Information

- TA: Arpit Joshi (arpit.joshi@ed.ac.uk)
- Pre-requisites: CS3 Computer Architecture
- Assignments: Assignment 1 – out 23-01-18; due 2-02-18
  Assignment 2 – out 13-02-18; due 6-03-18
- Recommended Books:
  - Culler & Singh - Parallel Computer Architecture: A Hardware/Software Approach – Morgan Kaufmann
  - Hennessy & Patterson - Computer Architecture: A Quantitative Approach – Morgan Kaufmann – 5th edition
- Lecture slides (no lecture notes)
- More info: www.inf.ed.ac.uk/teaching/courses/pa/
- Please interrupt with questions at any time

What is a Parallel Architecture?

"A collection of processing elements that cooperate to solve large problems fast"

Almasi and Gottlieb, 1989

Examples: Parallel Architectures

- ARM11
  - 8 stage pipeline
  - Upto 8 instructions in-flight

- Intel Pentium 4
  - 31 stage pipeline superscalar
  - Upto 124 instructions in-flight

- Intel Skylake
  - Quad core
  - 2 threads per core (SMT)
  - GPU

Examples: Parallel Architectures

- Sunway Taihulight
  - 40,960 SW26010 256 core CPUs
  - Upto 125 petaflops

- Tianhe-2
  - 32000 Intel Xeon 12 cores CPUs
  - 48000 Intel Xeon Phi 57 cores CPUs
  - Upto 54 petaflops

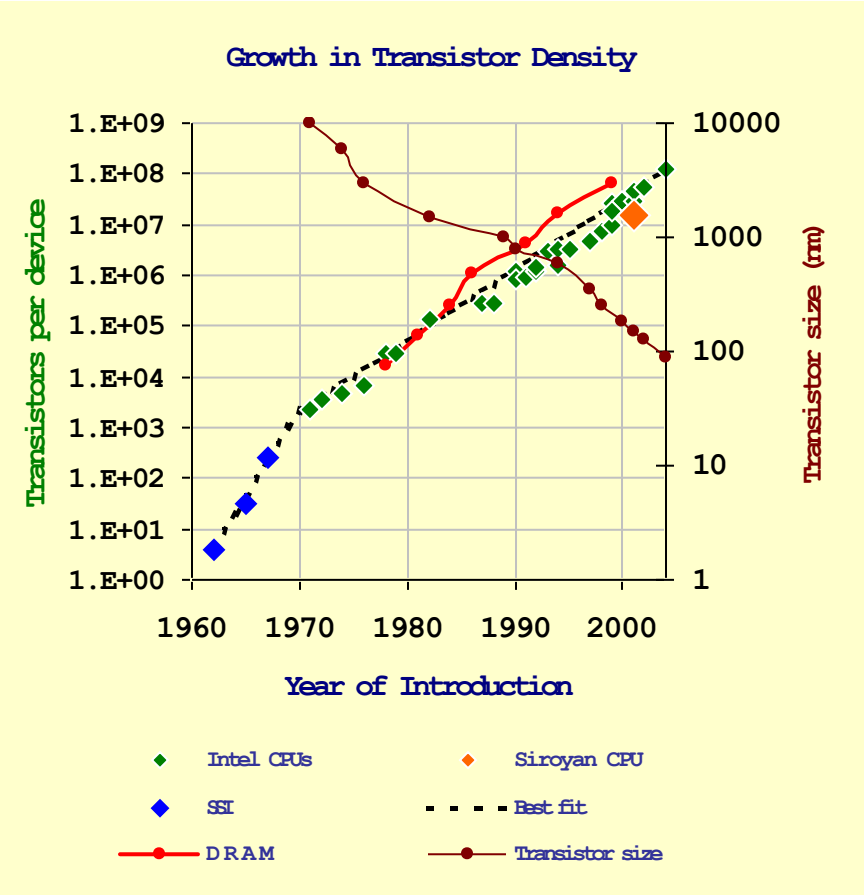
- Google Network
  - ??? linux machines
  - Several connected cluster farms

Why Parallel Architectures?

- Performance of sequential architecture is limited
  - Computation/data flow through logic gates, memory devices
  - At all of these there is a non-zero delay (at least delay of speed of light)
  - Thus, the speed of light and the minimum physical feature sizes impose a hard limit on the speed of any sequential computation
- Important applications that require performance (pull)
  - Nuclear reactor simulation; Predicting future climate etc.
  - Cloud/Big data:
    - Google: 40K searches per second
    - Facebook about a billion active users per day.
- Technological reasons (push)
  - What to do with all those transistors?

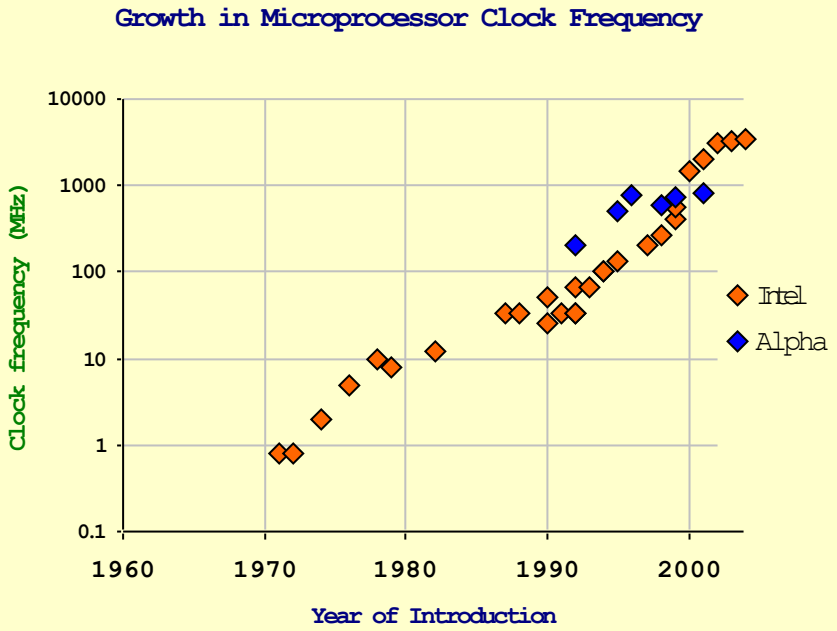
Technological Trends: Moore's Law

Moore's law

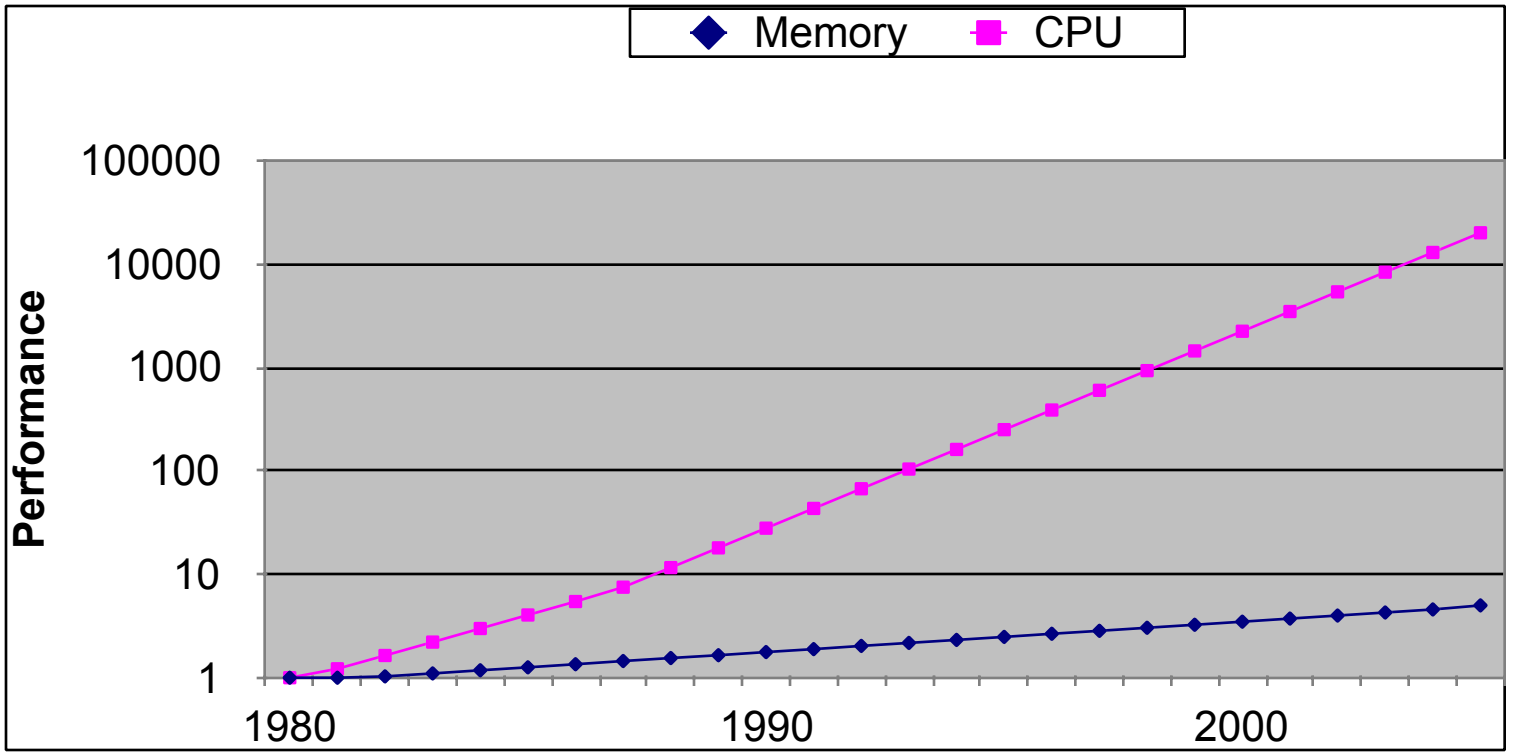


- 1965 – Gordon Moore's "Law"
  - Densities double every year (2x)
- 1975 – Moore's Law revised
  - Densities double every 2 years (1.42x)
- Actually 5x every 5 years (1.35x)

Dennard Scaling



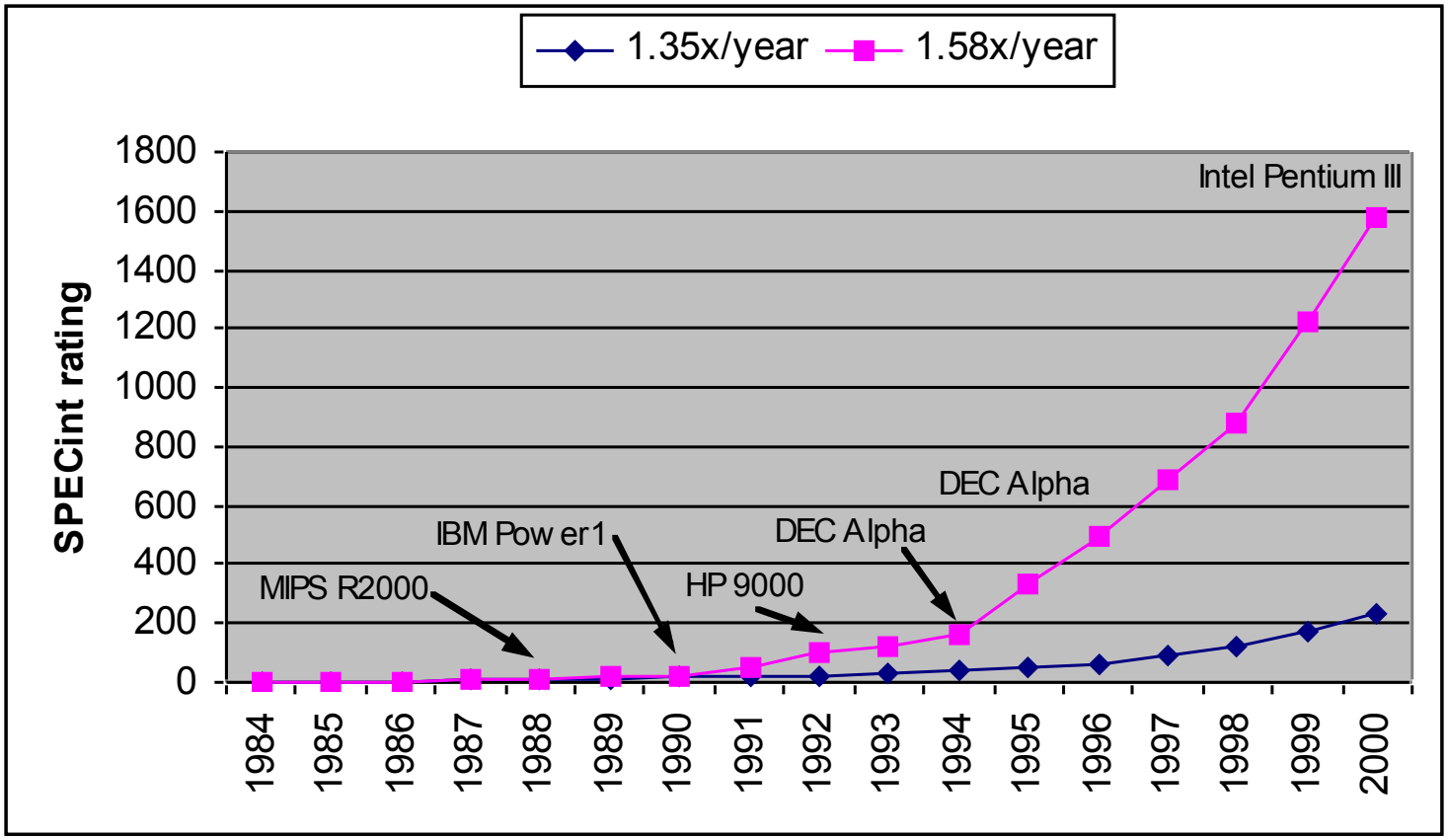
Technological Trend: Memory Wall



H&P Fig. 5.2

- **Bottom-line: memory access is increasingly expensive and CA must devise new ways of hiding this cost**

Tracking Technology: The role of CA

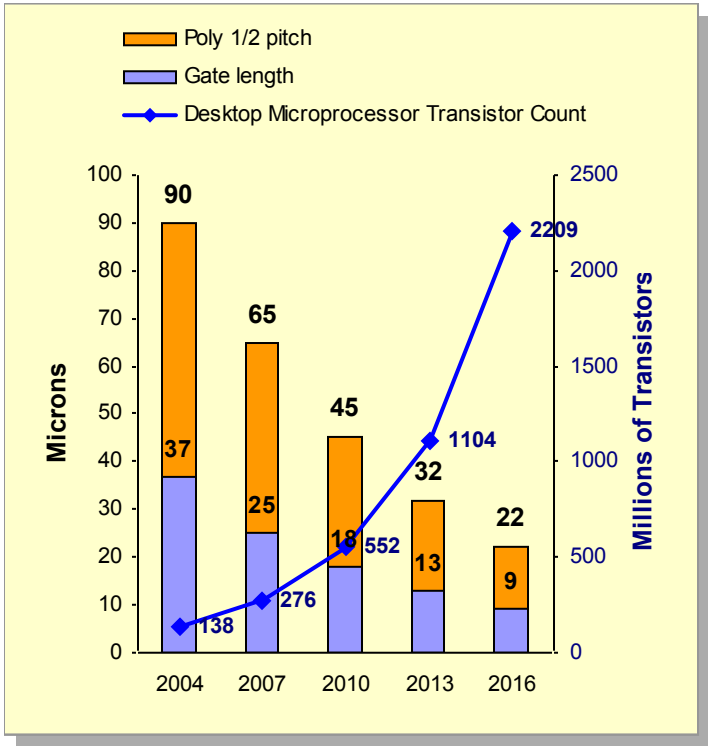


H&P Fig. 1.1

- **Bottom-line: architectural innovation complement technological improvements (ILP + Cache)**

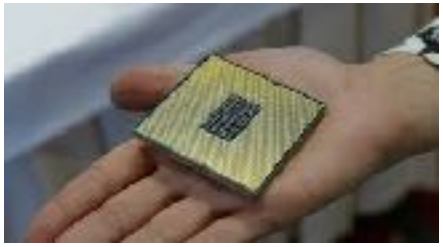
Future Technology Predictions (2003)

- Moore's Law will continue to ~2016
- Procs. will have 2.2 billion transistors
- DRAM capacity to reach 128 Gbit
- Procs. clocks should reach 40 GHz



Source: International Technology Roadmap for Semiconductors, 2003

State-of-the-art - January 2018



Qualcomm Centriq 2400

- 48cores
- 18B transistors, 398mm2
- ~120W @ 2.2GHz

Apple A11

- 6 CPU cores + 3 GPU cores + "Neural Engine"
- 4.3B transistors, 89mm^2
- ~2W @ ~2.3GHz

AMD Ryzen

- 16 cores, 2 threads per core
- 9.6transistors, 192mm^2
- ~180W @ ~3.4GHz

End of the Uniprocessor?

- Frequency has stopped scaling : Power Wall
  - End of Dennard scaling
- Memory wall
  - Instructions and data must be fetched
  - Memory becomes the bottleneck
- ILP Wall
  - Dependencies between instruction limit ILP

End of performance scaling for uniprocessors has forced industry to turn to chip-multiprocessors (Multicores)

Multicores

- Use transistors for adding cores
  - *(Note: Moore's law projected to end soon, now for real!)*
  - *(According to ITRS 2015, by 2021 will not be viable to shrink transistor any further!)*

- But, software must be parallel!
  – Remember Amdahl's law

- Lot of effort on making it easier to write parallel programs
  – For e.g Transactional memory

Amdahl's Law

- Let: F → fraction of problem that can be optimized
  S_{opt} → speedup obtained on optimized fraction

$$\therefore\ S_{overall} = \frac{1}{(1 - F) + \frac{F}{S_{opt}}}$$

- e.g.: F = 0.5 (50%), S_{opt} = 10 Sopt = ∞

$$S_{overall} = \frac{1}{(1 - 0.5) + \frac{0.5}{10}} = 1.8$$

$$S_{overall} = \frac{1}{(1 - 0.5) + 0} = 2$$

- **Bottom-line: performance improvements must be balanced**

Amdahl's Law and Efficiency

- Let: F → fraction of problem that can be parallelized
  S_{par} → speedup obtained on parallelized fraction
  P → number of processors

$$S_{overall} = \frac{1}{(1 - F) + \frac{F}{S_{par}}} \qquad E = \frac{S_{overall}}{P}$$

- e.g.: 16 processors ($S_{par} = 16$), F = 0.9 (90%),

$$S_{overall} = \frac{1}{(1 - 0.9) + \frac{0.9}{16}} = 6.4 \qquad E = \frac{6.4}{16} = 0.4\ (40\%)$$

For good scalability: E>50%; when resources are "free" then lower efficiencies are acceptable

Topics

- Fundamental concepts
  - Introduction
  - Types of parallelism
- Uniprocessor parallelism
  - Pipelining, Superscalars
- Shared memory multiprocessors
  - Cache coherence and memory consistency
  - Synchronization and Transactional Memory
- Hardware Multithreading
- Vector, SIMD Processors, GPUs
- Supercomputers and datacenter architecture (if time permits)