Chapter 1

Fundamentals of Quantitative Design and Analysis

Current Trends in Architecture

- Cannot continue to leverage Instruction-Level parallelism (ILP)
  - Single processor performance improvement ended in 2003
- New models for performance:
  - Data-level parallelism (DLP)
  - Thread-level parallelism (TLP)
  - Request-level parallelism (RLP)
- These require explicit restructuring of the application

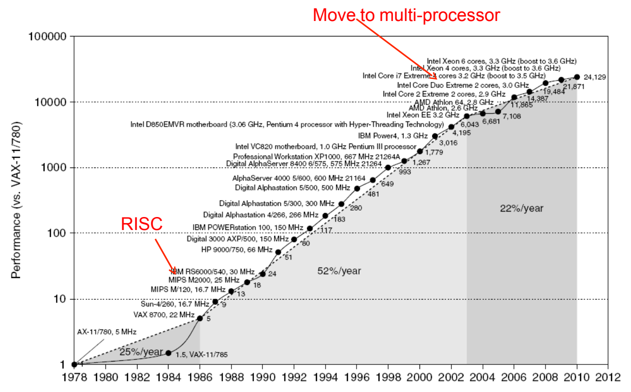
Computer Technology

- Performance improvements:
  - Improvements in semiconductor technology
    - Feature size, clock speed
  - Improvements in computer architectures
    - Enabled by HLL compilers, UNIX
    - Lead to RISC architectures
  - Together have enabled:
    - Lightweight computers
    - Productivity-based managed/interpreted programming languages

Classes of Computers

- Personal Mobile Device (PMD)
  - e.g. start phones, tablet computers
  - Emphasis on energy efficiency and real-time
- Desktop Computing
  - Emphasis on price-performance
- Servers
  - Emphasis on availability, scalability, throughput
- Clusters / Warehouse Scale Computers
  - Used for “Software as a Service (SaaS)”
  - Emphasis on availability and price-performance
  - Sub-class: Supercomputers, emphasis:
    - floating-point performance and fast internal networks
- Embedded Computers
  - Emphasis: price

Single Processor Performance



Parallelism

- Classes of parallelism in applications:
  - Data-Level Parallelism (DLP)
  - Task-Level Parallelism (TLP)
- Classes of architectural parallelism:
  - Instruction-Level Parallelism (ILP)
  - Vector architectures/Graphic Processor Units (GPUs)
  - Thread-Level Parallelism
  - Request-Level Parallelism

Flynn's Taxonomy

- Single instruction stream, single data stream (SISD)
- Single instruction stream, multiple data streams (SIMD)
  - Vector architectures
  - Multimedia extensions
  - Graphics processor units
- Multiple instruction streams, single data stream (MISD)
  - No commercial implementation
- Multiple instruction streams, multiple data streams (MIMD)
  - Tightly-coupled MIMD
  - Loosely-coupled MIMD
- Compare with...
  - CUDA's SIMT
  - Modern NUMA server with multiple multicore processor and accelerators

Trends in Technology

- Integrated circuit technology
  - Transistor density: 35%/year
  - Die size: 10-20%/year
  - Integration overall: 40-55%/year
  - Laws
    - Moore, Dennard
- DRAM capacity: 25-40%/year (slowing)
- Flash capacity: 50-60%/year
  - 15-20X cheaper/bit than DRAM
- Magnetic disk technology: 40%/year
  - 15-25X cheaper/bit then Flash
  - 300-500X cheaper/bit than DRAM

Defining Computer Architecture

- "Old" view of computer architecture:
  - Instruction Set Architecture (ISA) design
  - i.e. decisions regarding:
    - registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding
- "Real" computer architecture:
  - Specific requirements of the target machine
  - Design to maximize performance within constraints: cost, power, and availability
  - Includes ISA, microarchitecture, hardware

Math Sidebar: Compound Interest

- Suppose performance improves 50% per year
- How long does it take for performance to quadruple (factor of 4)?
- Does it take 8 years?
  - $8 \times 0.5 = 4$
- After 1 year: perf x $(1 + 0.5)$
- After 2 years: perf x $(1 + 0.5)(1+0.5)$ = perf x $(1+0.5)^2$
- After k years: perf x $(1 + 0.5)^k$
- Answer:
  $(1+0.5)^k = 4 \quad \Rightarrow \quad k = 3.42$ years

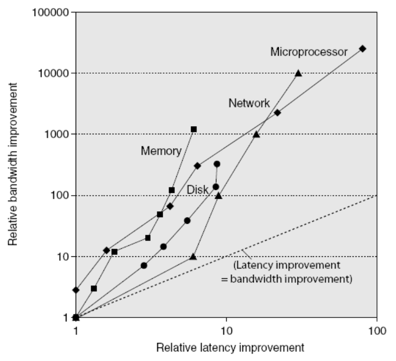
Instruction Set: Debate Won?

- Common instruction sets
  - CISC
    - x86 (but not the micro-code)
  - RISC
    - MIPS, HP PA, IBM Power, Sun SPARC, ARM
  - VLIW
    - Itanium, some GPUs (internally)
  - Vector
    - Cray, NEC, ... (mostly gone)
- Hybrids?
  - Intel Xeon Phi
    - x86 CISC
    - RISC-like microcode?
    - 512-bit vector floating-point

Bandwidth and Latency

- Bandwidth or throughput
  - Total work done in a given time
  - 10,000-25,000X improvement for processors
  - 300-1200X improvement for memory and disks
  - Units
    - flop/s, B/s, b/s
- Latency or response time
  - Time between start and completion of an event
  - 30-80X improvement for processors
  - 6-8X improvement for memory and disks
  - Units
    - CPU, memory: nano-second
    - Network: micro-seconds
    - Disk: milli-seconds

Bandwidth and Latency



Transistors and Wires

- Feature size
  - Minimum size of transistor or wire in x or y dimension
  - 10 microns in 1971 to .032 microns in 2011
  - Transistor performance scales linearly
    - Wire delay does not improve with feature size!
  - Integration density scales quadratically
- Law's of silicon chip manufacturing
  - Moore
  - Dennard

Power and Energy

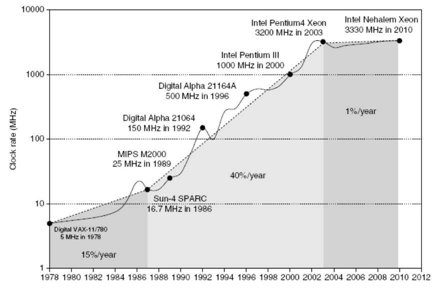
- Problem: Get power in, get power out
- Thermal Design Power (TDP)
  - Characterizes sustained power consumption
  - Used as target for power supply and cooling system
  - Lower than peak power, higher than average power consumption
- Clock rate can be reduced dynamically to limit power consumption
- Energy per task is often a better measurement

Dynamic Energy and Power

- Dynamic energy
  - Transistor switch from 0 -> 1 or 1 -> 0
  - ½ x Capacitive load x Voltage^2
- Dynamic power
  - ½ x Capacitive load x Voltage^2 x Frequency switched
- Reducing clock rate (frequency) reduces power, not energy
  - To reduce energy, lower the frequency of under-utilized or idle units

Power

- Intel 80386 consumed ~ 2 W
- 3.3 GHz Intel Core i7 consumes 130 W
- Heat must be dissipated from 1.5 x 1.5 cm chip
- This is the limit of what can be cooled by air



Reducing Power

- Techniques for reducing power:
  - Do nothing well
    - Idle state power
    - C-states, P-states
      - Cost of switching between them
  - Dynamic Voltage-Frequency Scaling
    - Implementations: silicon, OS-level, user-level
  - Low power state for DRAM, disks, interconnect
  - Overclocking, turning off cores
    - Race to halt
    - Number of power planes in a single chip

Static Power

- Static power consumption
  - $\text{Current}_{\text{static}}$ x Voltage
  - Scales with number of transistors
    - Another limit on Moore's law
  - Reduction achieved through power gating

Integrated Circuit Cost

- Formulas for integrated circuit

$$\text{Cost of integrated circuit} = \frac{\text{Cost of die} + \text{Cost of testing} + \text{Cost of packaging and final test}}{\text{Final test yield}}$$

$$\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{Die yield}}$$

$$\text{Dies per wafer} = \frac{\pi \times \left(\frac{1}{2} \times \text{Wafer diameter}\right)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2 \times \text{Die area}}}$$

- Bose-Einstein formula (empirical) [Sydow 2006]:

$$\text{Die yield} = \frac{\text{Wafer yield}}{(1 + \text{Defects per unit area} \times \text{Die area})^N}$$

  - Defects per unit area = 0.016-0.057 defects per square cm (2010)
  - N = process-complexity factor = 11.5-15.5 (40 nm, 2010)

Thoughts on Scaling Limits

- Feature size
  - Silicon mesh size (quantum effects)
  - Litography limits (wavelength)
  - Wire cross-talk
- Frequency
  - Dynamic power dissipation
- Voltage
  - Reliability of switching when moving from 5V down to 0.7V
  - Near-threshold circuits
- Core count
  - On-chip interconnect wiring and messaging

Dependability

- Commercial offerings (PaaS, SaaS, …)
  - Service Level Agreements (SLAs) or SLObjectives
  - Service accomplishment vs. interruption
    - Transitions: failures and restorations
- Module reliability
  - Mean time to failure (MTTF)
    - 1/MTTF = Failure In Time (FIT)
  - Mean time to repair (MTTR)
  - Mean time between failures (MTBF) = MTTF + MTTR
  - Availability = MTTF / MTBF = MTTF / (MTTF+MTTR)

Trends in Cost

- Cost driven down by learning curve
  - How much we've learned about the manufacturing process
  - Yield varies at various price-points
    - High-end vs. low-end parts:
      - IBM Cell and PS3
      - Intel Xeon Phi and Tianhe-2's Xeon Phi
- DRAM: price closely tracks cost
  - Standards, competition, patents
- Microprocessors: price depends on volume
  - 10% less for each doubling of volume

Measuring Performance

- Typical performance metrics:
  - Response time
  - Throughput
- Speedup of X relative to Y
  - Execution time_Y / Execution time_X
  - Geometric average is best suitable for combining relative values

$$\textit{Geometric average} = \sqrt[n]{\prod_{i=1\ldots n} a_i} = \sqrt[n]{a_1 \times a_2 \times \ldots \times a_n}$$

- Execution time
  - Wall clock time: includes all system overheads
  - CPU time: only computation time
- Benchmarks
  - Kernels (e.g. matrix multiply)
  - Toy programs (e.g. sorting)
  - Synthetic benchmarks (e.g. Dhrystone)
  - Benchmark suites (e.g. SPEC06fp, TPC-C)

Principles of Computer Design

- Take Advantage of Parallelism
  - e.g. multiple processors, disks, memory banks, pipelining, multiple functional units
- Principle of Locality
  - Reuse of data and instructions
  - Rules of thumb: 10/90, 20/80
  - Temporal vs. Spatial
- Focus on the Common Case
  - Amdahl's Law

$$\text{Execution time}_{\text{new}} = \text{Execution time}_{\text{old}} \times \left((1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}} \right)$$

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Principles of Computer Design

- The Processor Performance Equation

$$\text{CPU time} = \text{CPU clock cycles for a program} \times \text{Clock cycle time}$$

$$\text{CPU time} = \frac{\text{CPU clock cycles for a program}}{\text{Clock rate}}$$

$$\text{CPI} = \frac{\text{CPU clock cycles for a program}}{\text{Instruction count}}$$

$$\text{CPU time} = \text{Instruction count} \times \text{Cycles per instruction} \times \text{Clock cycle time}$$

$$\underbrace{\frac{\text{Instructions}}{\text{Program}}}_{\text{Algorithm}} \times \underbrace{\frac{\text{Clock cycles}}{\text{Instruction}}}_{\text{CPI}} \times \underbrace{\frac{\text{Seconds}}{\text{Clock cycle}}}_{\text{Cycle time}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU time}$$

Observations on Amdahl's Law

$$\lim_{f \to 0} S_{\text{overall}} = \lim_{f \to 0} \frac{1}{(1-f) + f/S_e} = 1$$

Too small of a portion parallelized

$$\lim_{f \to 1} S_{\text{overall}} = \lim_{f \to 1} \frac{1}{(1-f) + f/S_e} = S_e$$

Everything is parallelized

$$\lim_{S_e \to \infty} S_{\text{overall}} = \lim_{S_e \to \infty} \frac{1}{(1-f) + f/S_e} = \frac{1}{1-f}$$

Perfect parallelism

Also see Gustafson's Law: Reevaluating of Amdahl's Law

Principles of Computer Design

- Different instruction types might have different CPIs

$$\text{CPU clock cycles} = \sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i = (\text{IC}_{\text{load}} \times \text{CPI}_{\text{load}}) + (\text{IC}_{\text{store}} \times \text{CPI}_{\text{store}}) + \ldots$$

$$\text{CPU time} = \left(\sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i \right) \times \text{Clock cycle time}$$

Remember Amdahl: focus on common case

Amdahl's Law Example

- A database server spends 60% of time in I/O transactions and 40% of time in computing
- After an upgrade, the processor is 10x faster
- What is the speedup after the upgrade?

$$\text{Speedup}_{\text{overall}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}} = \frac{1}{(1-0.4) + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$$

Optimize CPI or IPC?

- 1980s
  - RISC era
  - Minimize CPI
- 1990s
  - Superscalar RISC
  - Maximize IPC
- 2000s+
  - x86 ISA
  - Optimize both: x86 ISA and microcode