Chapter 1

Fundamentals of Quantitative Design and Analysis

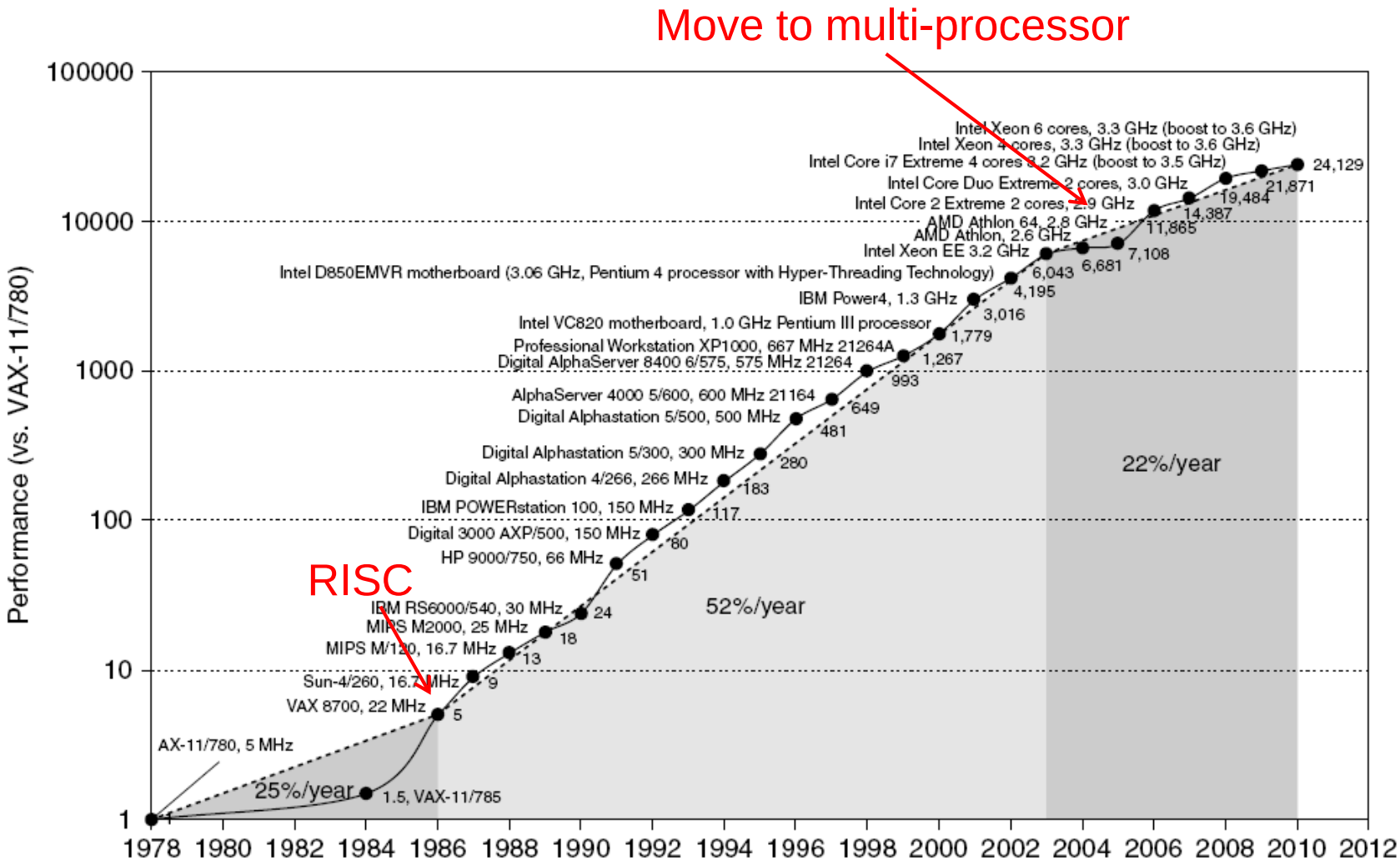
Outline

- Introduction
- Classes of Computers
- Design Computer Architecture
- Trends in Technology
- Trends in Power and Energy in ICs
- Trends in Cost
- Dependability
- Measuring, Reporting, and Summarizing Performance
- Quantitative Principles of Design

Computer Technology

- Performance improvements:
  - Improvements in semiconductor technology (材料、技術)
    - Feature size, clock speed
  - Improvements in computer architectures (軟硬體介面)
    - Enabled by HLL compilers, UNIX
      - Elimination of assembly language
      - Creation of standardized, vendor-independent operating systems
- The two changes lead to RISC architectures
  - Focus on
    - instruction-level parallelism and
      - Pipeline and multiple instruction issue
    - the use of caches
  - Together have enabled:
    - Lightweight computers
    - Productivity-based managed/interpreted programming languages
- Intel rose to the challenge, primarily by translating 80x86 instructions into RISC-like instruction internally. (macro-instruction, micro-instruction)

Single Processor Performance



Current Trends in Architecture

- Cannot continue to leverage Instruction-Level parallelism (ILP) (3rd)
  - Single processor performance improvement ended in 2003
  - In 2004, Intel canceled its high-performance uniprocessor projects.
- New models for performance:
  - Data-level parallelism (DLP) (4th) (6th)
  - Thread-level parallelism (TLP) (5th)
  - Request-level parallelism (RLP) (6th)
- Whereas the compiler and hardware conspire to exploit ILP implicitly without programmer's attention, DLP, TLP, and RLP are explicitly parallel, requiring the restructuring of the application.

Outline

- Introduction
- Classes of Computers
- Design Computer Architecture
- Trends in Technology
- Trends in Power and Energy in ICs
- Trends in Cost
- Dependability
- Measuring, Reporting, and Summarizing Performance
- Quantitative Principles of Design

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

Feature	Personal mobile device (PMD)	Desktop	Server	Clusters/warehouse-scale computer	Embedded
Price of system	$100–$1000	$300–$2500	$5000–$10,000,000	$100,000–$200,000,000	$10–$100,000
Price of micro-processor	$10–$100	$50–$500	$200–$2000	$50–$250	$0.01–$100
Critical system design issues	Cost, energy, media performance, responsiveness	Price-performance, energy, graphics performance	Throughput, availability, scalability, energy	Price-performance, throughput, energy proportionality	Price, energy, application-specific performance

Parallelism

- Classes of parallelism in applications:
  - Data-Level Parallelism (DLP)
    - There are many data items that can be operated on at the same time.
  - Task-Level Parallelism (TLP)
    - Tasks of work are created that can operate independently and largely in parallel.
- Classes of architectural parallelism:
  - Instruction-Level Parallelism (ILP)
    - Exploits DLP
      - at modest levels with compiler help using ideas like pipeline and
      - at medium levels using ideas like speculative execution.
  - Vector architectures/Graphic Processor Units (GPUs)
    - Exploits DLP by applying a single instruction to a collection of data in parallel
  - Thread-Level Parallelism
    - Exploits DLP or TLP in a tightly coupled hardware model that allows for interaction among parallel threads.
  - Request-Level Parallelism
    - Exploits DLP or TLP among largely decoupled tasks specified by the programmers or the operating system.

Flynn's Taxonomy

- Single instruction stream, single data stream (SISD)
  - Uniprocessor
  - Exploit ILP
  - Chapter 3 cover SISD architectures that use ILP techniques such as superscalar and speculative execution.
- Single instruction stream, multiple data streams (SIMD)
  - The same instruction is executed by multiple processors using different data streams.
  - Chapter 4 cover SIMD and three different architectures that exploits DLP:
    - Vector architectures
    - Multimedia extensions
    - Graphics processor units
- Multiple instruction streams, single data stream (MISD)
  - No commercial implementation
- Multiple instruction streams, multiple data streams (MIMD)
  - Each processor fetches its own instructions and operates on its own data, and it targets TLP.
  - Can also exploit DLP, although the overhead is likely to be higher.
  - Chapter 5 covers tightly-coupled MIMD which exploits TLP since multiple cooperating threads operate in parallel.
  - Chapter 6 covers loosely-coupled MIMD which exploits RLP, where many independent tasks can proceed in parallel naturally with little need for communication or synchronization.

Outline

- Introduction
- Classes of Computers
- Design Computer Architecture
- Trends in Technology
- Trends in Power and Energy in ICs
- Trends in Cost
- Dependability
- Measuring, Reporting, and Summarizing Performance
- Quantitative Principles of Design

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 includes ISA, microarchitecture (organization), hardware
  - Design a computer to meet functional requirements as well as cost, power, performance, and availability goals

Computer Architecture不只包含抽象化的ISA，乃至於實作時的組織架構與底層硬體設計都包含在內。

Genuine (實質) Computer Architecture

- The implementation of a computer has two components: organization and hardware.
  - Organization includes the high-level aspects of a computer's design, such as the memory system, the memory interconnect, and the design of the internal processor.
    - AMD Opteron and Intel Core i7 have the same ISA but different organization (different pipeline and cache organization).
  - Hardware refers to the specifics of a computer, including the detailed logic design and the packing technology of the computer.

Discussion

- The Intel Core i7 and the Intel Xeon 7650 are nearly identical but offer different clock rates and different memory system.
  - Making the Xeon 7650 more effective for server computers.

Outline

- Introduction
- Classes of Computers
- Design Computer Architecture
- Trends in Technology
- Trends in Power and Energy in ICs
- Trends in Cost
- Dependability
- Measuring, Reporting, and Summarizing Performance
- Quantitative Principles of Design

Trends in Technology

- Integrated circuit technology
  - Transistor density: 35%/year
  - Die size: 10-20%/year
  - Integration overall: 40-55%/year
    - Doubling every 18 to 24 months (Moore's law)
- 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

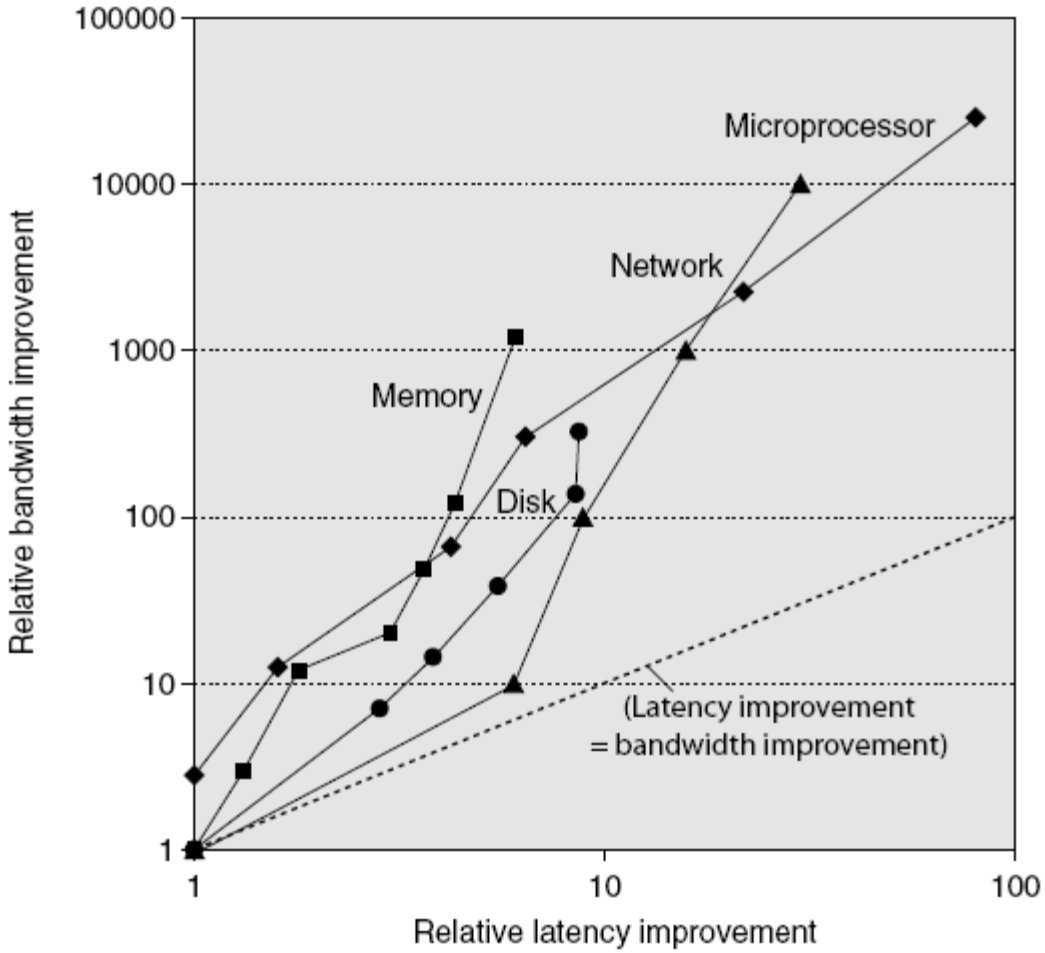
Bandwidth and Latency

- Bandwidth or throughput
  - Total amount of work done in a given time
    - Megabytes per second for a disk transfer
  - 10,000-25,000X improvement for processors
  - 300-1200X improvement for memory and disks

為何兩者間有差異?

- Latency or response time
  - Time between start and completion of an event
    - Milliseconds for a disk access
  - 30-80X improvement for processors
  - 6-8X improvement for memory and disks

Bandwidth and Latency



Log-log plot of bandwidth and latency milestones

Transistors and Wires

- Feature size
  - Minimum size of transistor or wire in x or y dimension
  - 10 microns in 1971 to .032 microns (32 nano) in 2011, 10 nano in 2016。2018年台積電 5 奈米廠的動工，將於2020年量產。
  - Transistor performance scales linearly
    - Wire delay does not improve with feature size!
  - Integration density scales quadratically

Outline

- Introduction
- Classes of Computers
- Design Computer Architecture
- Trends in Technology
- Trends in Power and Energy in ICs
- Trends in Cost
- Dependability
- Measuring, Reporting, and Summarizing Performance
- Quantitative Principles of Design

Power and Energy

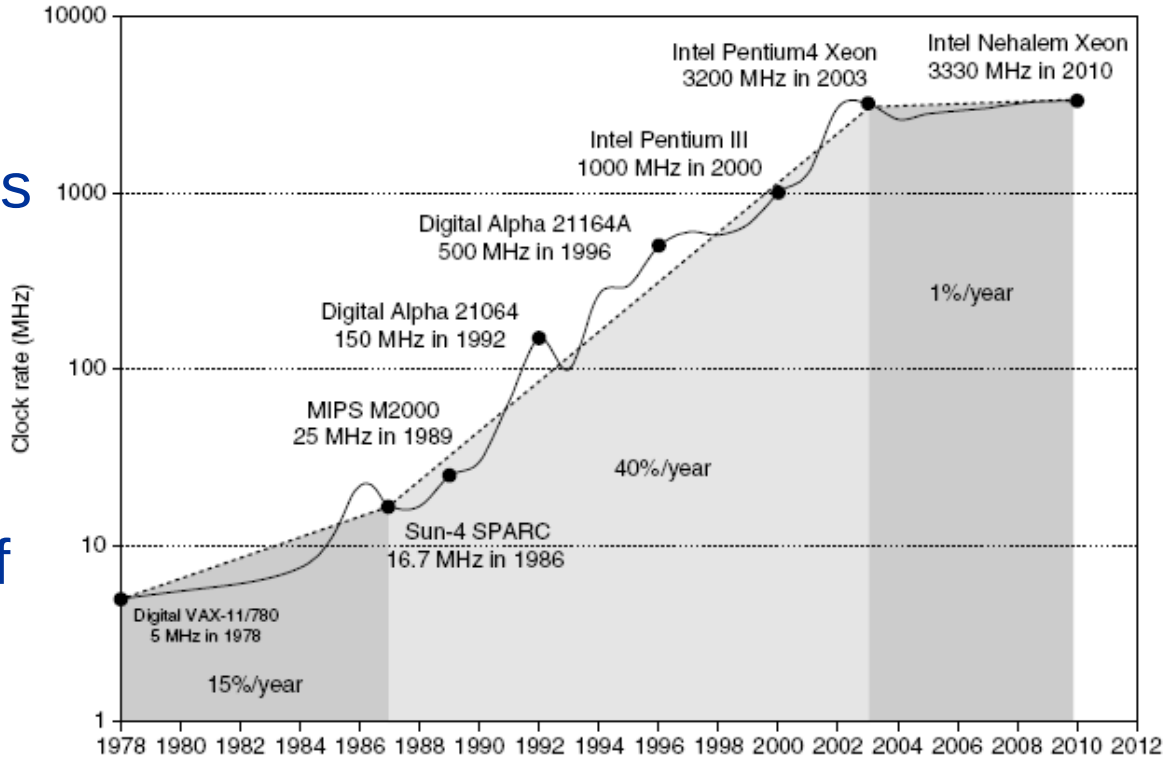
- Problem: Get power in, get power out
- Thermal Design Power (TDP:散熱設計功率)
  - Characterizes sustained power consumption
  - Used as target for power supply and cooling system
  - Lower than peak power, higher than average power consumption
- Clock rate can be reduced dynamically to limit power consumption
- 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 reduces power, not energy

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
    - Turn off the clock of inactive modules
  - Dynamic Voltage-Frequency Scaling
    - Offer a few clock frequencies and voltages
  - Low power state for DRAM, disks
    - Offer low power modes
  - Overclocking,
    - For single threaded code, these micorporcessors can turning off cores but one and run it

Static Power

- Static power consumption
  - $Current_{static}$ x Voltage
  - Scales with number of transistors
  - To reduce: power gating
    - A technique used in integrated circuit design to reduce power consumption, by shutting of the flow of current to blocks of the circuit that are not currently in use.

Outline

- Introduction
- Classes of Computers
- Design Computer Architecture
- Trends in Technology
- Trends in Power and Energy in ICs
- Trends in Cost
- Dependability
- Measuring, Reporting, and Summarizing Performance
- Quantitative Principles of Design

Trends in Cost

- Cost driven down by learning curve (學習曲線)
  - Manufacturing costs decrease over time (技術純熟後，成本降低)
  - The learning curve is best measured by change in yield
    - Yield: the percentage of manufactured devices that survives the testing procedure
- DRAM: price closely tracks cost
- Microprocessors: price depends on volume
  - 10% less for each doubling of volume

Integrated Circuit Cost

- Integrated circuit

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

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

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

- Bose-Einstein formula:

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

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

Outline

- Introduction
- Classes of Computers
- Design Computer Architecture
- Trends in Technology
- Trends in Power and Energy in ICs
- Trends in Cost
- Dependability
- Measuring, Reporting, and Summarizing Performance
- Quantitative Principles of Design

Dependability

- Module reliability=measure of continuous service accomplishment (or time to failure).
  - Mean time to failure (MTTF)
  - Mean time to repair (MTTR)
  - Mean time between failures (MTBF) = MTTF + MTTR
  - Module Availability = MTTF / MTBF

Example calculating reliability

- If modules have *exponentially distributed lifetimes* (age of module does not affect probability of failure), overall failure rate is the sum of failure rates of the modules
- Calculate MTTF for 10 disks (1M hour MTTF per disk), 1 ATA controller (0.5M hour MTTF), 1 power supply (0.2M hour MTTF), 1 fan (0.2M hour MTTF), and 1 ATA cable (1M hour MTTF):

$$
\begin{aligned}
FailureRate &= 10 \times (1/1{,}000{,}000) + 1/500{,}000 + 1/200{,}000 + 1/200{,}000 + 1/1{,}000{,}000 \\
&= 23{,}000/1{,}000{,}000{,}000 \\
MTTF &= 1{,}000{,}000{,}000/23{,}000 \\
&\approx 43{,}500 hours
\end{aligned}
$$

Outline

- Introduction
- Classes of Computers
- Design Computer Architecture
- Trends in Technology
- Trends in Power and Energy in ICs
- Trends in Cost
- Dependability
- Measuring, Reporting, and Summarizing Performance
- Quantitative Principles of Design

Measuring Performance

- Typical performance metrics:
  - Response time
  - Throughput
- Speedup of X relative to Y
  - Execution time$_Y$ / Execution time$_X$
- 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)

How Summarize Suite Performance

- Arithmetic average of execution time of all pgms?
  - But they vary by 4X in speed, so some would be more important than others in arithmetic average
- Could add a weights per program, but how pick weight?
  - Different companies want different weights for their products
- SPECRatio: Normalize execution times to reference computer, yielding a ratio proportional to performance =

$$\frac{\text{time on reference computer}}{\text{time on computer being rated}}$$

How Summarize Suite Performance

- If program SPECRatio on Computer A is 1.25 times bigger than Computer B, then

$$1.25 = \frac{SPECRatio_A}{SPECRatio_B} = \frac{\frac{ExecutionTime_{reference}}{ExecutionTime_A}}{\frac{ExecutionTime_{reference}}{ExecutionTime_B}}$$

$$= \frac{ExecutionTime_B}{ExecutionTime_A} = \frac{Performance_A}{Performance_B}$$

  - Note that when comparing 2 computers as a ratio, execution times on the reference computer drop out, so choice of reference computer is irrelevant

How Summarize Suite Performance

- Since ratios, proper mean is geometric mean (SPECRatio unitless, so arithmetic mean meaningless)

$$GeometricMean = \sqrt[n]{\prod_{i=1}^{n} SPECRatio_i}$$

1. Geometric mean of the ratios is the same as the ratio of the geometric means
2. Ratio of geometric means
   = Geometric mean of performance ratios
   $\Rightarrow$ choice of reference computer is irrelevant!

- These two points make geometric mean of ratios attractive to summarize performance

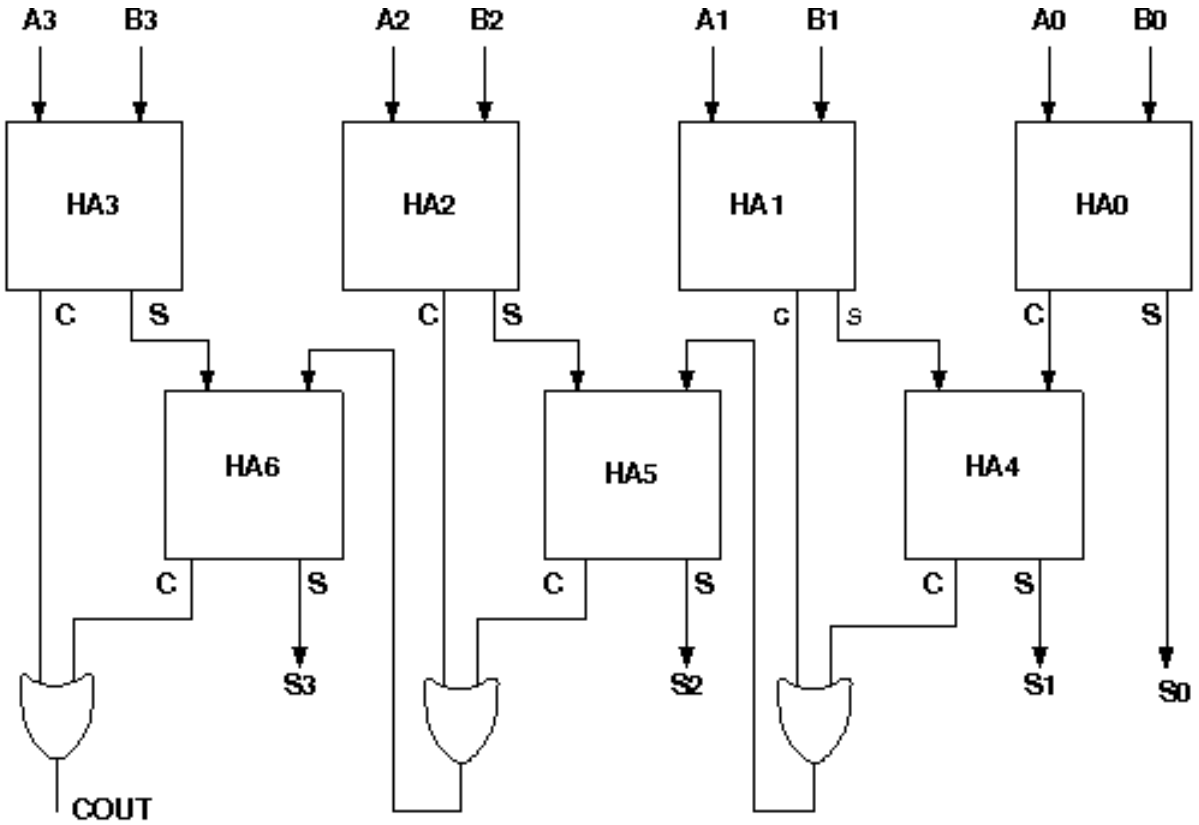
Outline

- Introduction
- Classes of Computers
- Design Computer Architecture
- Trends in Technology
- Trends in Power and Energy in ICs
- Trends in Cost
- Dependability
- Measuring, Reporting, and Summarizing Performance
- Quantitative Principles of Design

Principles of Computer Design

- Take Advantage of Parallelism
  - Increasing throughput of computer via multiple processors, disks, memory banks, pipelining, multiple functional units
  - Detailed HW design
    - Carry lookahead adders uses parallelism to speed up computing sums from linear to logarithmic in number of bits per operand
    - Multiple memory banks searched in parallel in set-associative caches
  - Pipeline: overlap instruction execution to reduce the total time to complete an instruction sequence
    - Not every instruction depends on its immediate predecessor, so executing the instructions completely or partially in parallel.
    - Classic 5-stage pipeline:
      1) Instruction Fetch (Ifetch),
      2) Register Read (Reg),
      3) Execute (ALU),
      4) Data Memory Access (Dmem),
      5) Register Write (Reg)

Carry lookahead adders



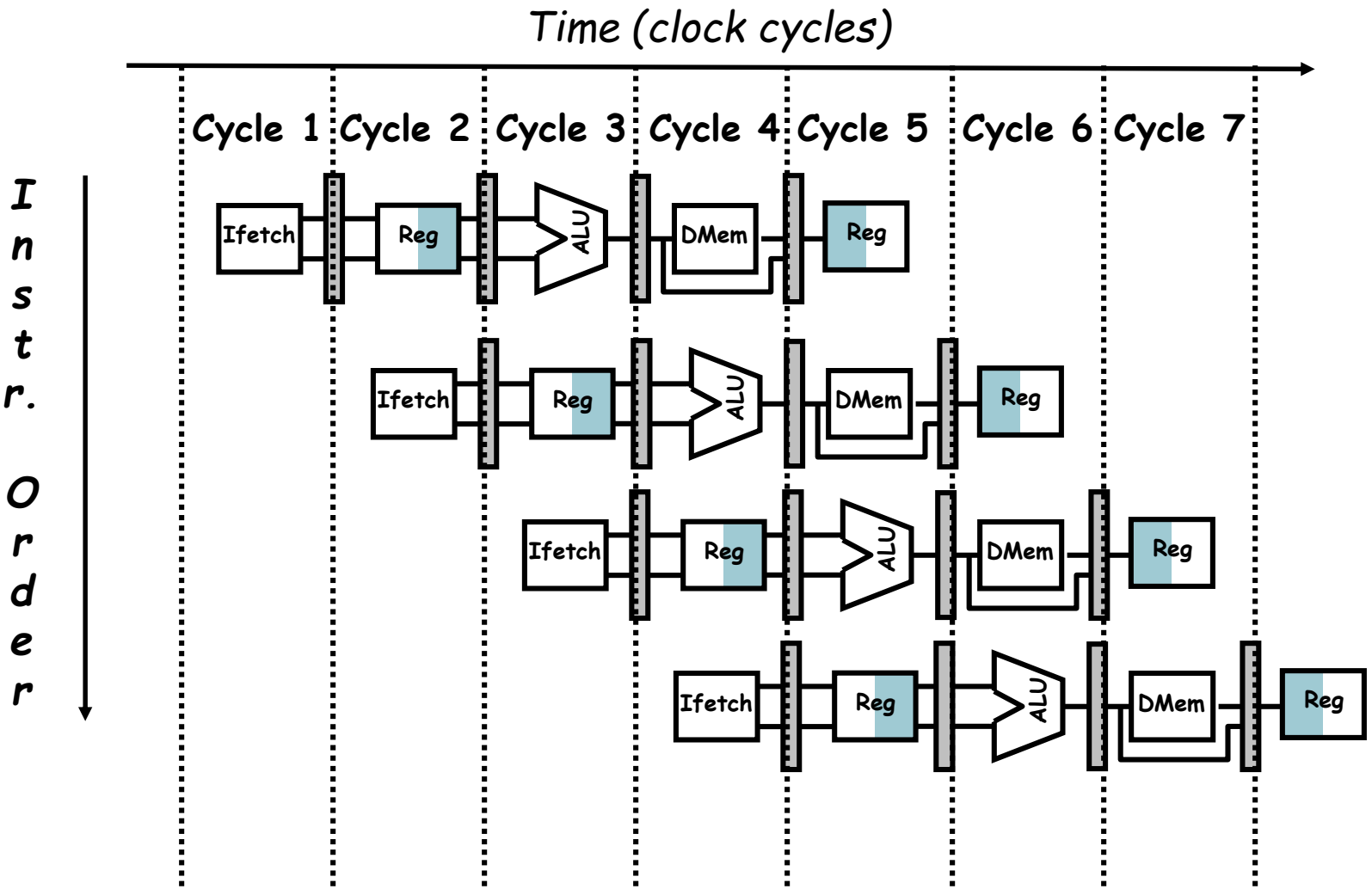
Principles of Computer Design

- Principle of Locality
  - Reuse of data and instructions
- Focus on the Common Case
  - Amdahl's Law

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

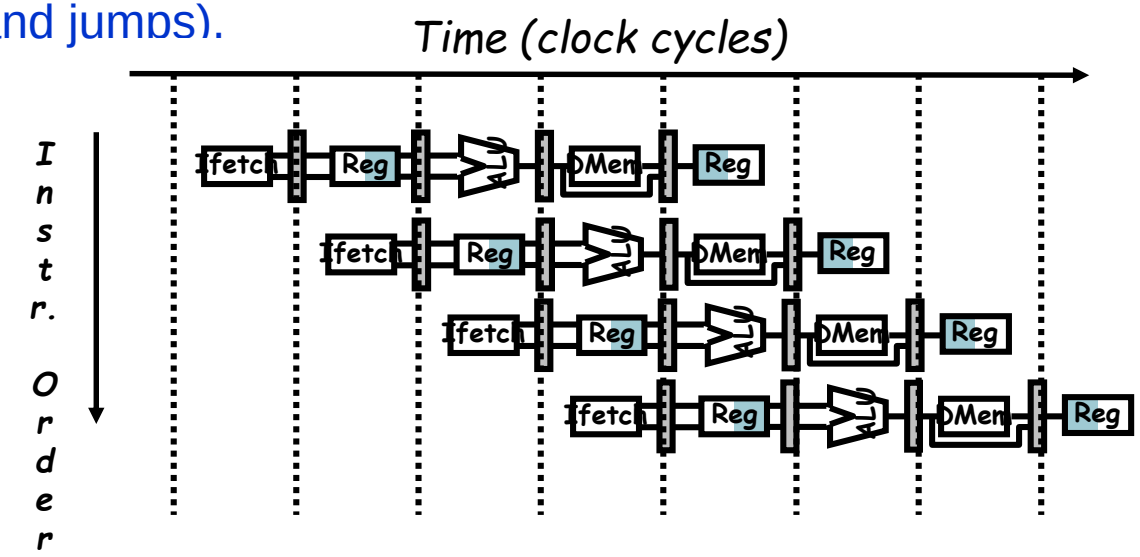
$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

Pipelined Instruction Execution



Limits to pipelining

- Hazards prevent next instruction from executing during its designated clock cycle
  - Structural hazards: attempt to use the same hardware to do two different things at once
  - Data hazards: Instruction depends on result of prior instruction still in the pipeline
  - Control hazards: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches and jumps).



Principles of Computer Design

- Principle of Locality
  - Reuse of data and instructions
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon
    (e.g., straight-line code, array access)

Principles of Computer Design

- Focus on the Common Case
  - Amdahl's Law

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

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

Amdahl's Law example

- New CPU 10X faster
- I/O bound server, so 60% time waiting for I/O

$$\text{Speedup}_{\text{overall}} = \frac{1}{\left(1 - \text{Fraction}_{\text{enhanced}}\right) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$= \frac{1}{\left(1 - 0.4\right) + \frac{0.4}{10}} = \frac{1}{0.64} = 1.56$$

- Apparently, its human nature to be attracted by 10X faster, vs. keeping in perspective its just 1.6X faster

Principles of Computer Design

- The Processor Performance Equation

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

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

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

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

$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU time}$$

Principles of Computer Design

- Different instruction types having different CPIs

$$\text{CPU clock cycles} = \sum_{i=1}^{n} \text{IC}_i \times \text{CPI}_i$$

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

Outline

- Introduction
- Classes of Computers
- Design Computer Architecture
- Trends in Technology
- Trends in Power and Energy in ICs
- Trends in Cost
- Dependability
- Measuring, Reporting, and Summarizing Performance
- Quantitative Principles of Design
- Fallacies and Pitfalls

Pitfall and Fallacy

- Pitfalls are generalizations of principles that are true in a limited context
  - 以偏概全的描述，描述的正確性僅限於在某些特例之下，不是通則。
- Fallacies
  - 錯誤的觀念，只以一個反例即可證明觀念的錯誤性

Fallacies and Pitfalls

- Fallacies
  - Multiprocessors are a silver bullet (高招).
    - There was no other option due to the ILP walls and power falls.
    - It is possible to continue to improve performance by replacing a high-clock-rate, inefficient core.
  - Hardware enhancements that increase performance improve energy efficiency or are at most energy neutral.
  - Benchmarks remain valid indefinitely.
  - Peak performance tracks observed performance.
  - The rated mean time to failure of disks is 1,200,000 hours or almost 140 years, so disks practically never fail.