Computer Organization and Architecture

Faisal Syafar

Introduction

What is "Computer Architecture"?

Term coined by Fred Brooks and colleagues at IBM:

"…the structure of a computer that a machine language programmer must understand to write a correct (timing independent) program for that machine."

Amdahl, Blaauw, and Brooks, 1964
"Architecture of the IBM System 360",
IBM Journal of Research and Development

Do you know about System 360 family?

What is "Computer Architecture"?

Term used differently by Hennessy and Patterson (our textbook)

- includes much implementation

Several years ago, the term computer architecture often referred only to instruction set design. Other aspects of computer design were called implementation, often insinuating that implementation is uninteresting or less challenging.

We believe this view is incorrect. The architect's or designer's job is much more than instruction set design, and the technical hurdles in the other aspects of the project are likely more challenging than those encountered in instruction set design.

- Patterson & Hennessy

Outline

- Course Information
  - Logistics
  - Grading
  - Syllabus
  - Course Overview
- Technology Trends
  - Moore's Law

Course Information (1)

Time and Place

- Thursday 13:30-14:20, Lab. Komputer

Instructor

- Faisal Syafar
- Faisal.syafar@unm.ac.id

Course Web Page

- Linked from mine: https://faisalsyafar.wixsite.com/myweb

Course Information (2)

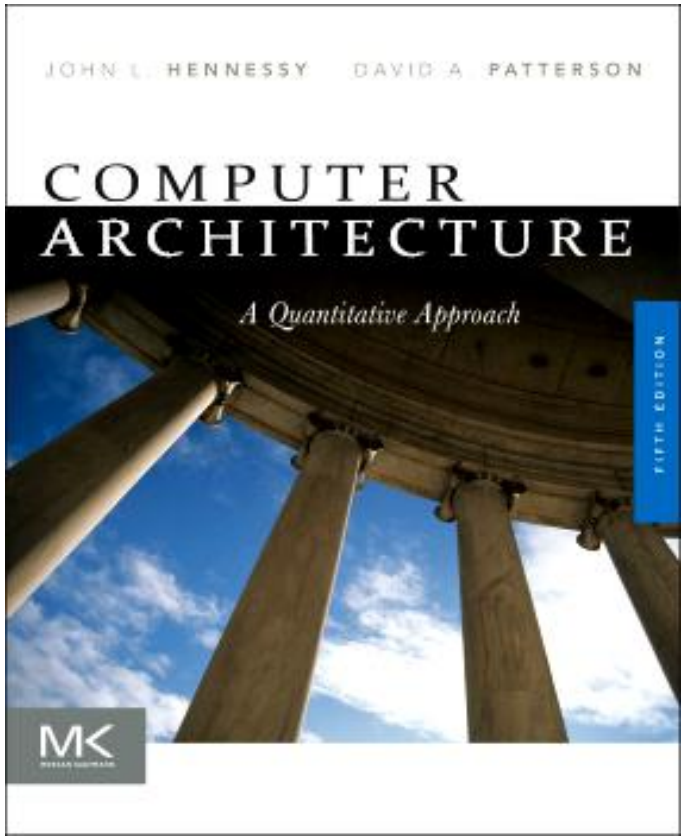
Prerequisites

- Databases and Digital electronics
- I assume you know the following topics
  - CPU: ALU, control unit, registers, buses, memory management
  - Control Unit: register transfer language, implementation, hardwired and microprogrammed control
  - Memory: address space, memory capacity
  - I/O: CPU-controlled (polling, interrupt), autonomous (DMA)
  - Pipelining (at least an overview)
- Representative books (available in Brauer Library)
  - Patterson & Hennessy: Computer Organization and Design: The Hardware/Software Interface. Morgan Kaufmann Publishers.
  - Harris & Harris: Digital Design and Computer Architecture, 2nd ed. (July 2012), Morgan Kaufmann Publishers.

Course Information (3)

Textbook

- Hennessy & Patterson: Computer Architecture: A Quantitative Approach *(5th edition)*, Morgan Kaufmann Publishers, Sep 2011
  - available at amazon.com, bn.com…
- Quite different from earlier eds.: more on multiprocessing (multicore)



Course Information (4)

Textbook (contd.)

- We will cover the following material:
  - Fundamentals of Computer Design (Chapter 1)
  - Instruction Set Principles and Examples (App A & K)
  - Memory-Hierarchy Design (App B & Chapter 2)
  - Pipelining: Basic and Intermediate Concepts (App C)
  - Instruction-Level Parallelism (Chapter 3)
  - Data-Level Parallelism (Chapter 4)
  - Thread-Level Parallelism (Chapter 5)
  - Storage Systems (App D)
  - On-Chip Networks (selected readings)
  - Emerging Technologies of Computation (selected readings)

Additional readings/papers may be handed out

- e.g., case studies on the last two topics above

Course Information (5)

Grading

- 10% Individual assignment (I)
- 10% Quizzes (Homeworks)
- 30% Midterm exam
- 40% Final exam
- 10% Project (Team) Assignment (II)
  - e.g., present a survey or case study on multicore
  - e.g., present a paper on Computer Organization or Architecture technologies

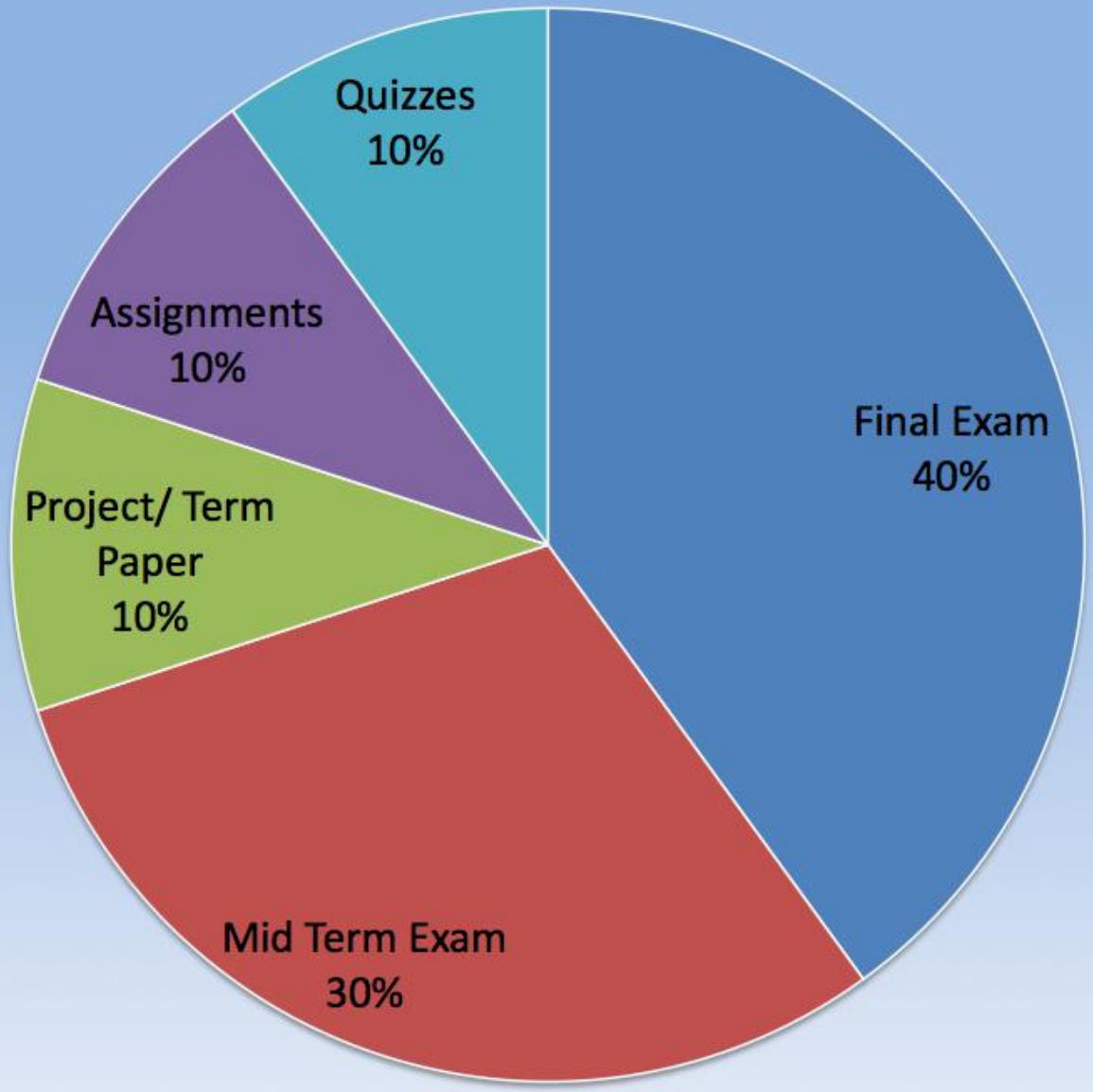
Assignments are due at beginning of class on due date

- Late assignments: penalty=10%/day or part thereof

Honor Code is in effect: for all homework/exams/projects

- encouraged to discuss ideas/concepts with others
- work handed in must be your own

Evaluation breakdown



Policies/ Administrivia

- Class discipline
- Mobile phones and laptops
- Learning disabilities/ Special needs
- Announcements on course website
- Attendance policy
- Assignment policy
- Quizzes (announced/ unannounced)

Tata Tertib Perkuliahan

- Keterlambatan maksimal Tahun 2019: **9 menit** (2018: 11 Menit; 2017: 13 Menit);
- Untuk dapat lulus MK ini, HARUS hadir kuliah minimal 80%;
- Berpakaian yang rapi dan sopan (tidak berkaos oblong) dan bersepatu;
- Rambut disisir rapi (Laki-laki TIDAK menyerupai wanita, dan wanita TIDAK menyerupai Laki-Laki);
- Mahasiswa yang tidak bisa mengikuti UTS/UAS, harus ijin sebelum pelaksanaan Ujian, dan susulan akan diberikan maksimal 3 hari setelah Ujian;
- Selama kuliah HP dimatikan atau di "*silent*";
- Menjaga kesopanan dalam berkomunikasi dengan dosen baik secara lansung maupun lewat E-Mail.
- Tidak ada komunikasi melalui SMS/Call, KECUALI dengan Ketua Kelas;
- Mahasiswa diperkenankan membawa minuman ke dalam ruang kelas akan tetapi DILARANG meninggalkan bekas minuman di dalam ruang kuliah

Deskripsi Singkat Mata Kuliah

Mata kuliah Organisasi dan Arsitektur Komputer ini merupakan mata kuliah yang bertujuan membekali mahasiswa untuk dapat mengetahui konsep organisasi dan arsitektur komputer, representasi data dengan benar dan memahami perkembangan sistem komputer; menganalisis dengan menggunakan skema tentang komponen utama suatu sistem komputer dan interkoneksi antar sistem maupun interkoneksi dengan perangkat luarnya; menganalisis arsitektur dan organisasi internal prosesor; menganalisis arsitektur dan organisasi dari control unit sebuah sistem komputer; serta pemahaman organisasi paralel dan multiprosessing dari sebuah sistem komputer.

What is in 16B51C408?

Understand modern computer architecture so you can:

- Write better programs
  - Understand the performance implications of algorithms, data structures, and programming language choices
- Write better compilers
  - Modern computers need better optimizing compilers and better programming languages
- Write better operating systems
  - Need to re-evaluate the current assumptions and tradeoffs
  - Example: fully exploit multicore/manycore architectures
- Design better computer architectures
  - There are still many challenges left
  - Example: how to design efficient multicore architectures
- Satisfy the Distribution Requirement

Content


1
Organisasi dan Arsitektur
2
Struktur dan Fungsi
3
Komponen Utama Komputer

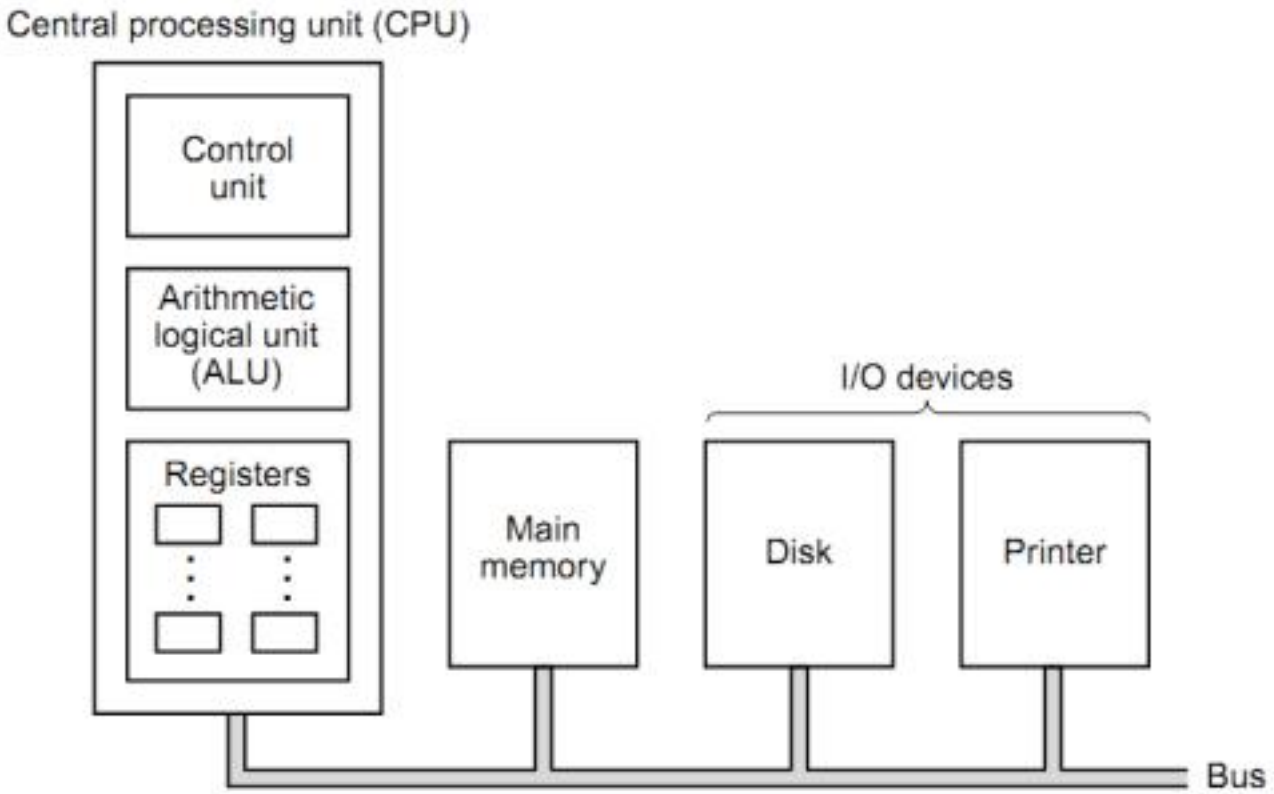
- What is Computer Organization?
- What is Computer Architecture?
- What is Computer?

What is Computer Organization?

Organisasi komputer mengacu pada unit operasional dan interkoneksi mereka yang merealisasikan spesifikasi arsitektur.

Contoh Atribut organisasi termasuk detail-detail hardware transparan untuk programmer, seperti sinyal kontrol, interface antara computer dan peripheral, dan teknologi memori yang digunakan.

Organisasi Komputer Sederhana



What is Computer Architecture?

Arsitektur komputer mengacu pada atribut-atribut dari sistem terlihat seorang programmer atau dengan kata lain, atribut-atribut yang memiliki dampak langsung pada eksekusi logis dari sebuah program

Contoh atribut arsitektur termasuk set instruksi, jumlah bit yang digunakan untuk wakili berbagai jenis data (misalnya, angka, karakter), I / O mekanisme, dan teknik untuk mengatasi memori.

Contoh lain misalkan seorang programmer ketika akan coding, dia perlu mengetahui apakah pada komputer yang sedang digunakan memiliki instruksi perkalian atau tidak? Bagaimana pengalamatan data di memori? Bagaimana data direpresentasikan? Bagaimana bilangan direpresentasikan?

Pertanyaan-pertanyaan ini merupakan pertanyaan domain arsitektur komputer. Jika dipersempit, maka fokus dari pembelajaran di bidang arsitektur komputer adalah "bagaimana **software** bekerja pada **hardware**".

What is Computer?

Kata komputer berasal dari bahasa Latin yaitu Computare yang artinya menghitung. Dalam bahasa Inggris disebut to compute.

Definisi komputer: sekumpulan alat elektronik yang saling bekerja sama, dapat menerima data (input), mengolah data (proses) dan memberikan informasi (output) serta terkoordinasi menurut seperangkat instruksi dibawah kontrol program yang tersimpan di memorinya.

Semua komputer memiliki 4 fungsi:
- Pengolahan data - Data processing
- Penyimpanan data - Data storage
- Pemindahan data - Data movement
- Kendali - Control

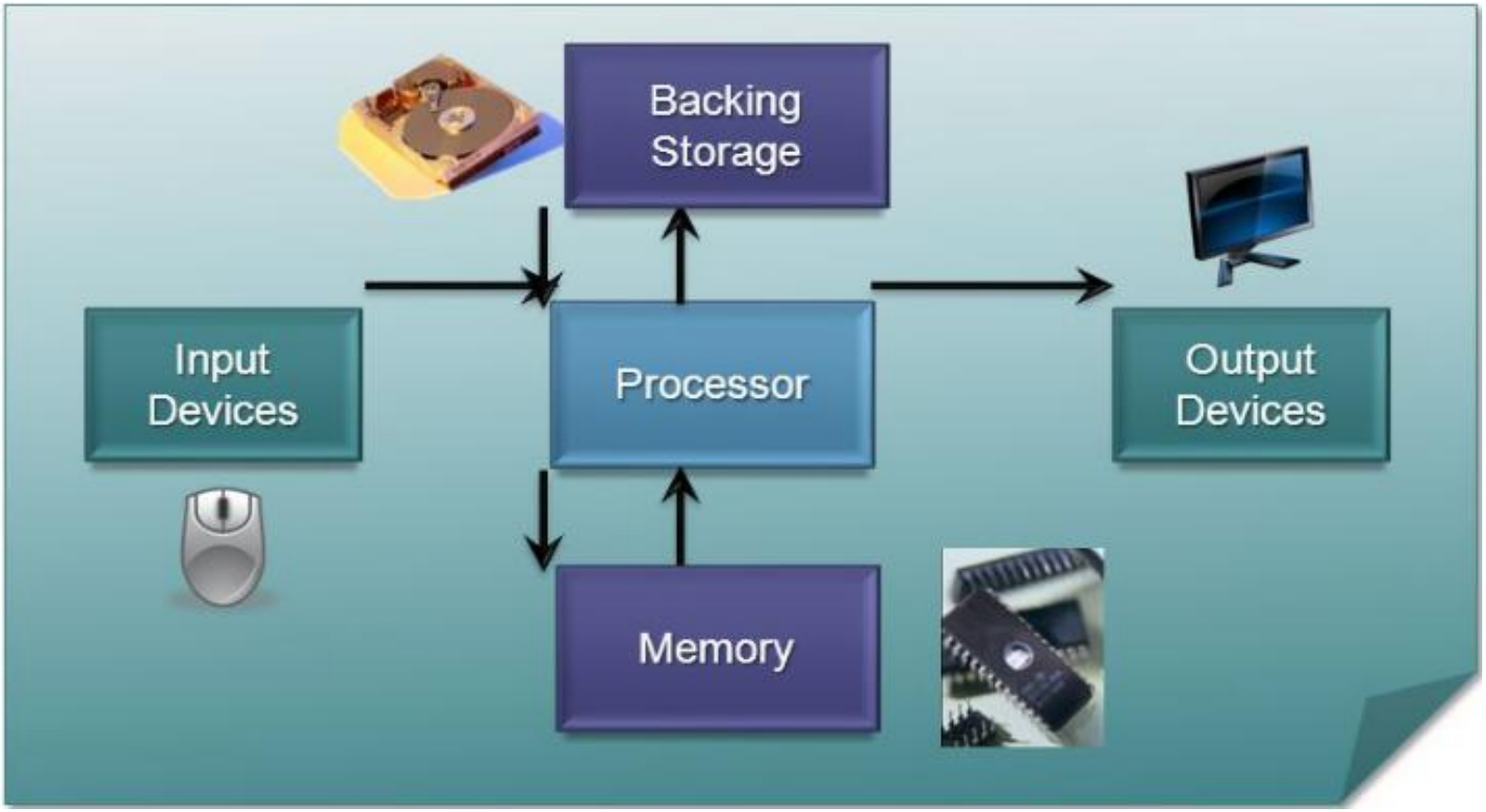
Why You Need to study This Course?

Students need to understand computer architecture in order to structure a program so that it runs more efficiently on a real machine. In selecting a system to use, they should be able to understand the tradeoff among various components, such as CPU clock speed vs. memory size, for example.

Architecture extends upward into computer software because a processor's architecture must cooperate with the operating system and system software. It is difficult to design an operating system well without knowledge of the underlying architecture.

Concepts used in computer architecture find application in other courses. In particular, the way in which the computer provides architectural support for programming languages and operating system facilities reinforces concepts from those areas.

What is Computer Function and Structure?



Structure

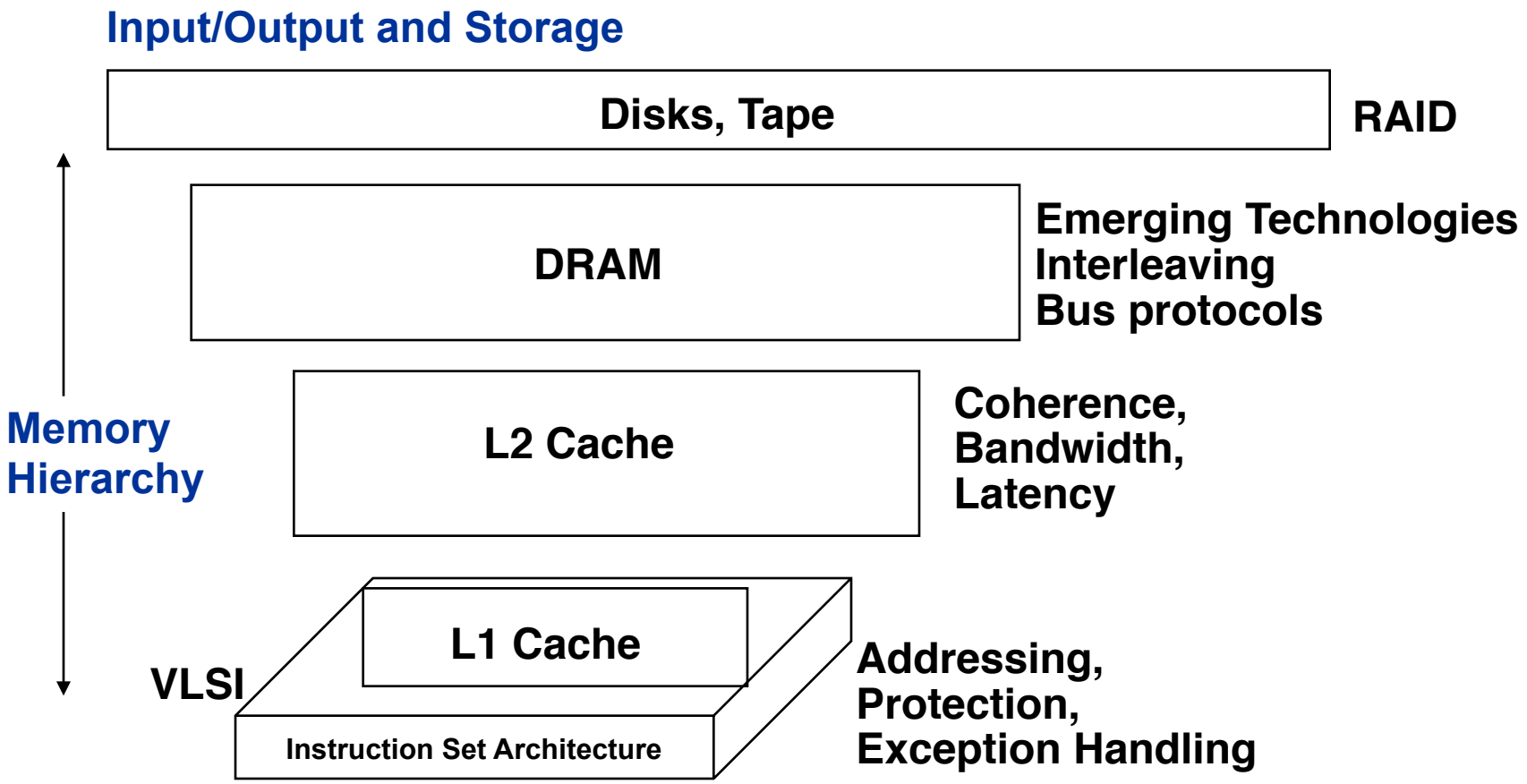
Is the way how each component/unit of computer communicates to each other.

Function

Refers to the operation of each component which include in a structure.

- Detail…Next Week

Computer Architecture Topics



Trends of 2015-2025

- Technology
  - Very large dynamic RAM: 256 Mbits to 4Gb and beyond
  - Large fast static RAM: 16 MB, 5ns
- Complete systems on a chip
  - 100+ million to 1+ billion transistors
- Parallelism
  - Superscalar, Superpipelined, Vector, Multiprocessors?
  - Processor Arrays?
  - Multicore/manycore!
- Special-Purpose Architectures
  - GPU's, mp3 players, nanocomputers …
- Reconfigurable Computers?
  - Wearable computers

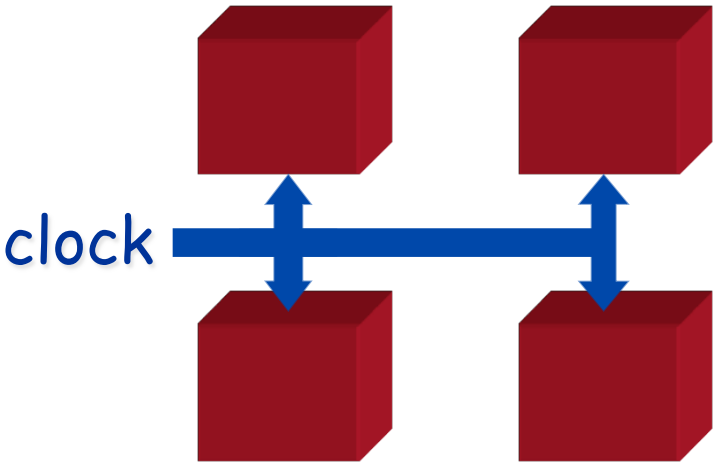
Trends of 2015-2020

- Low Power
  - Over 50% of computing devices portable now (!)
  - Hand held communicators
  - Performance per watt, battery life
  - Transmeta
  - Asynchronous (clockless) design 
- Communication (I/O)
  - Many applications I/O limited, not computation
  - Computation scaling, but memory, I/O bandwidth not keeping pace
- Multimedia
  - New interface technologies
  - Video, speech, handwriting, virtual reality, …

Diversion: Clocked Digital Design

Most current digital systems are *synchronous:*

- *Clock:* a global signal that paces operation of all components



Benefit of clocking: *enables discrete-time representation*

- all components operate exactly once per clock tick
- component outputs need to be ready by next clock tick
  - allows "glitchy" or incorrect outputs between clock ticks

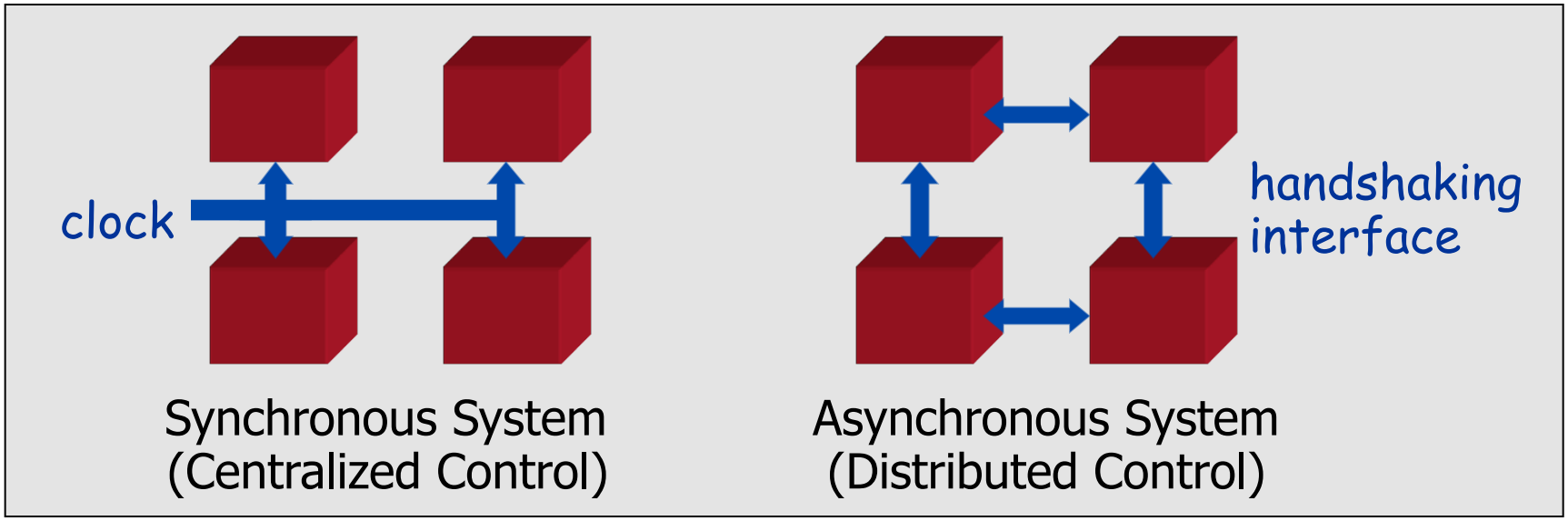
Microelectronics Trends

Current and Future Trends: Significant Challenges

- Large-Scale "Systems-on-a-Chip" (SoC)
  - 100 Million ~ 1 Billion transistors/chip
- Very High Speeds
  - multiple GigaHertz clock rates
- Explosive Growth in Consumer Electronics
  - demand for ever-increasing functionality …
  - … with very low power consumption (limited battery life)
- Higher Portability/Modularity/Reusability
  - "plug 'n play" components, robust interfaces

Alternative Paradigm: Asynchronous Design

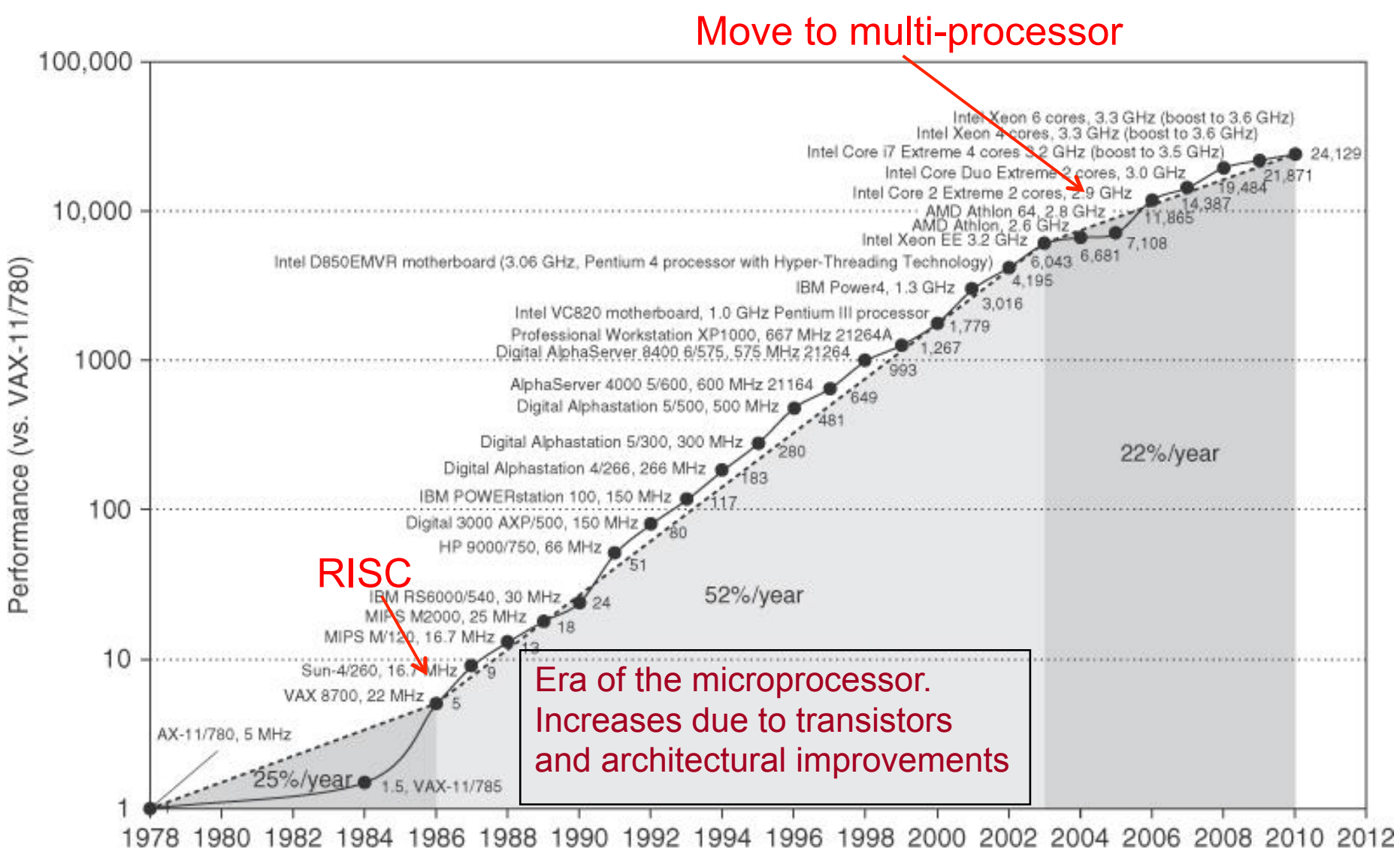
- Digital design with no centralized clock
- Synchronization using local *"handshaking"*



Asynchronous Benefits:

- Higher Performance: not limited by slowest component
- Lower Power: zero clock power; inactive parts consume little power
- Reduced Electromagnetic Noise: no clock spikes [e.g., Philips pagers]
- Greater Modularity: variable-speed interfaces; reusable components

Trends: Performance



Trends: Clock speed

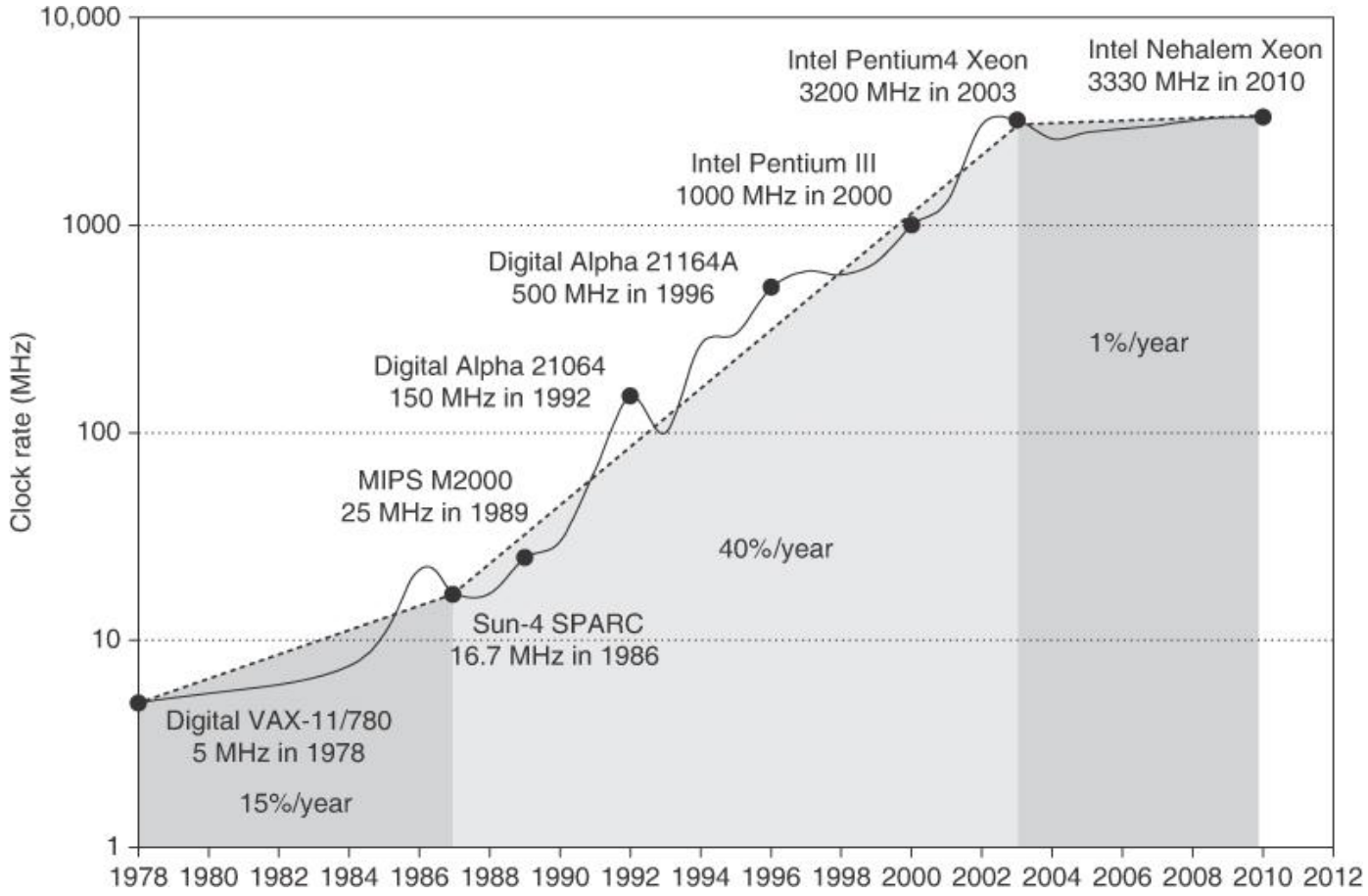


Figure 1.11 Growth in clock rate of microprocessors in Figure 1.1. Between 1978 and 1986, the clock rate improved less than 15% per year while performance improved by 25% per year. During the "renaissance period" of 52% performance improvement per year between 1986 and 2003, clock rates shot up almost 40% per year. Since then, the clock rate has been nearly flat, growing at less than 1% per year, while single processor performance improved at less than 22% per year.

Performance

- Increase by 2002 was much faster than would have been due to fabrication tech (e.g. 0.13 micron) alone
- What has slowed the trend?
  - Note what is really being built
    - A commodity device!
    - So cost is very important
  - Problems
    - Amount of heat that can be removed economically
    - Limits to instruction level parallelism
    - Memory latency

Moore's Law

- What is Moore's Law?

- Originally: Number of transistors on a chip
  - at the lowest cost/component
- It's not quite clear what it really is ☺
  - Moore's original paper, doubling yearly
    - Didn't make it in 1975
  - Often quoted as doubling every 18 months
  - Sometimes as doubling every two years
- Moore's article worth reading
  - http://download.intel.com/research/silicon/moorespaper.pdf

Quantitative Principles of Computer Design

$$\left[\frac{\text{work / results / program / instructions / bits}}{\text{time}}\right]$$

Performance
Rate of producing results
Throughput
Bandwidth

$$P = \frac{1}{T}$$

Execution time
Response time
Latency

$$\left[\frac{\text{time}}{\text{work / result / program / instruction / bit}}\right]$$

Quick Look: Classes of Computers

- Used to be
  - mainframe,
  - mini, and
  - micro
- Now
  - Desktop
    - Price/performance, single app, graphics
  - Server
    - Reliability, scalability, throughput
  - Embedded
    - Not only “toasters”, but also cell phones, etc.
    - Cost, power, real-time performance
  - Mobile
    - smartphones, tablets, fitness devices, etc.

Chip Performance

- Based on a number of factors
  - Feature size (or "technology" or "process")
    - Determines transistor & wire density
    - Used to be measured in microns, now nanometers
    - Currently: 90 nm, 65 nm, 45 nm, even 22 nm
  - Die size
  - Device speed

- Note section on wires in textbook
  - Thin wires → more resistance and capacitance
  - Wire delay scales poorly
    - simply making transistors faster is not sufficient anymore
    - managing connectivity is the big challenge now

ITRS

International Technology Roadmap for Semiconductors

- http://www.itrs2.net/
- An industry consortium
- predicts trends
- take a look at the yearly report on their website
  - take a few minutes to skim the "executive summary"

ITRS Predictions (2012 update)

Table B *ITRS Table Structure—Key Lithography-related Characteristics by Product*

Near-term Years

Year of Production	2011	2012	2013	2014	2015	2016	2017	2018
Flash ½ Pitch (nm) (un-contacted Poly)(f)[2]	22	20	18	17	15	14.2	13.0	11.9
DRAM ½ Pitch (nm) (contacted)[1,2]	36	32	28	25	23	20.0	17.9	15.9
MPU/ASIC Metal 1 (M1) ½ Pitch (nm)[1,2]	38	32	27	24	21	18.9	16.9	15.0
MPU High-Performance Printed Gate Length (GLpr) (nm) ††[1]	35	31	28	25	22	19.8	17.7	15.7
MPU High-Performance Physical Gate Length (GLph) (nm)[1]	24	22	20	18	17	15.3	14.0	12.8
ASIC/Low Operating Power Printed Gate Length (nm) ††[1]	41	35	31	25	22	19.8	17.7	15.7
ASIC/Low Operating Power Physical Gate Length (nm)[1]	26	24	21	19.4	17.6	16.0	14.5	13.1
ASIC/Low Standby Power Physical Gate Length (nm)[1]	30	27	24	22	20	17.5	15.7	14.1
MPU High-Performance Etch Ratio GLpr/GLph [1]	1.4589	1.4239	1.3898	1.3564	1.3239	1.2921	1.2611	1.2309
MPU Low Operating Power Etch Ratio GLpr/GLph [1]	1.5599	1.4972	1.4706	1.2869	1.2640	1.2416	1.2196	1.1979

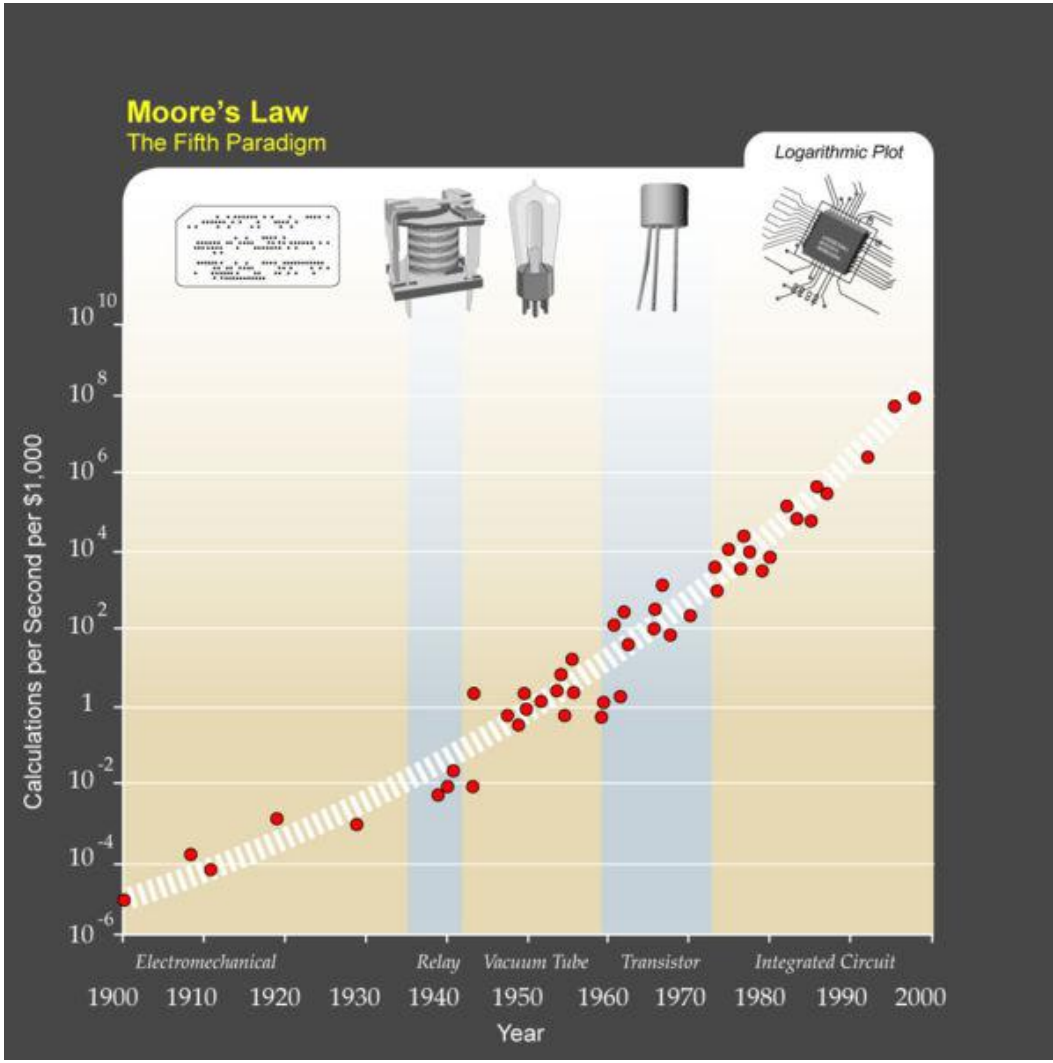
Long-term Years

Year of Production	2019	2020	2021	2022	2023	2024	2025	2026
Flash ½ Pitch (nm) (un-contacted Poly)(f)[2]	10.9	10.0	8.9	8.0	8.0	8.0	8.0	8.0
DRAM ½ Pitch (nm) (contacted)[1,2]	14.2	12.6	11.3	10.0	8.9	8.0	7.1	6.3
MPU/ASIC Metal 1 (M1) ½ Pitch (nm)[1,2]	13.4	11.9	10.6	9.5	8.4	7.5	6.7	6.0
MPU High-Performance Printed Gate Length (GLpr) (nm) ††[1]	14.0	12.5	11.1	9.9	8.8	7.9	6.79	5.87
MPU High-Performance Physical Gate Length (GLph) (nm)[1]	11.7	10.6	9.7	8.9	8.1	7.4	6.6	5.9
ASIC/Low Operating Power Printed Gate Length (nm) ††[1]	14.0	12.5	11.1	9.9	8.8	7.9	6.8	5.8
ASIC/Low Operating Power Physical Gate Length (nm)[1]	11.9	10.8	9.8	8.9	8.1	7.3	6.5	5.8
ASIC/Low Standby Power Physical Gate Length (nm)[1]	12.7	11.4	10.2	9.2	8.2	7.4	6.6	5.9
MPU High-Performance Etch Ratio GLpr/GLph [1]	1.2013	1.1725	1.1444	1.1169	1.0901	1.0640	1.0315	1.0000
MPU Low Operating Power Etch Ratio GLpr/GLph [1]	1.1766	1.1558	1.1352	1.1151	1.0953	1.0759	1.0372	1.0000

Aside: Ray Kurzweil

- Kurzweil: futurist, author
  - Book in 2005: "The Singularity is Near"
    - Movie in 2010
    - Predicts singularity around 2045

[From Wikipedia]
The technological singularity, or simply the singularity, is a hypothetical moment in time when artificial intelligence will have progressed to the point of a greater-than-human intelligence, radically changing civilization, and perhaps human nature. Since the capabilities of such an intelligence may be difficult for a human to comprehend, the technological singularity is often seen as an occurrence (akin to a gravitational singularity) beyond which the future course of human history is unpredictable or even unfathomable.



Trends

- Now let's look at trends in
  - Bandwidth (Throughput) vs. Latency
  - Power
  - Cost
  - Dependability
  - Performance

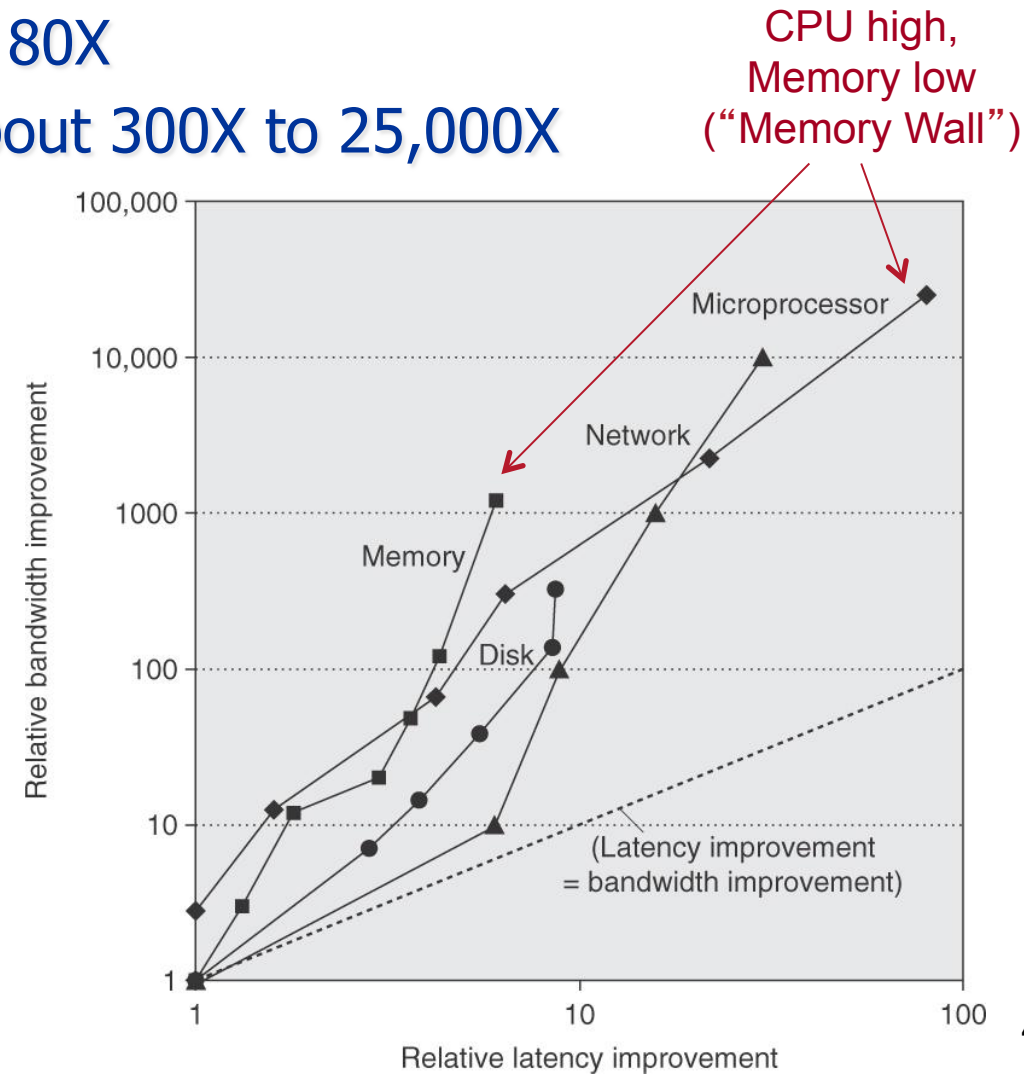
Bandwidth vs. Latency

- What is "bandwidth"?
  - or "throughput"
  - total amount of work done per time
  - e.g.: MB/sec for disk or network transfer
  - e.g.: billions of instructions/sec executed by CPU

- What is "latency"?
  - or "response time"
  - time between start and completion
  - e.g.: milliseconds for a disk access
  - e.g.: time lag for video game to respond to user input

Bandwidth vs. Latency

- These metrics apply to various components
  - microprocessor, memory, disk, network
  - latency improved 6X to 80X
  - bandwidth improved about 300X to 25,000X



Textbook Figure 1.9 Log–log plot of bandwidth and latency milestones.
Note that latency improved 6X to 80X while bandwidth improved about 300X to 25,000X.

Bandwidth vs. Latency

- Bandwidth improves much faster than latency
  - In the time that bandwidth doubles, latency improves by no more than a factor of 1.2 to 1.4
  - (and capacity improves faster than bandwidth)
- Or:
  - bandwidth improves by more than the square of the improvement in latency
  - what could be the reason for bandwidth to improve faster than latency?
    - easier to exploit parallelism of more transistors to improve bandwidth
    - improvements to latency limited by delays, sequentiality, etc.

Why Less Improvement?

- Moore's Law helps bandwidth
  - Longer distance for signal to travel, so longer latency
  - Which offsets faster transistors
- Distance limits latency
  - Speed of light lower bound
- Bandwidth sells
  - Capacity, processor "speed" and benchmark scores
- Latency can help bandwidth
  - Often bandwidth is increased by adding latency
- OS introduces latency

Techniques to Ameliorate

- Caching
  - Use capacity ("bandwidth") to reduce average latency
- Replication
  - Again, leverage capacity
- Prediction
  - Use extra processing transistors to pre-fetch
  - Maybe also to recompute instead of fetch

Trends

- Now let's look at trends in
  - Bandwidth vs. Latency
  - Power
  - Cost
  - Dependability
  - Performance

Power

- For CMOS chips, traditional dominant energy consumption has been in switching transistors, called dynamic power

$$Power_{dynamic} = \frac{1}{2} CapacitiveLoad \times Voltage^2 \times FrequencySwitched$$

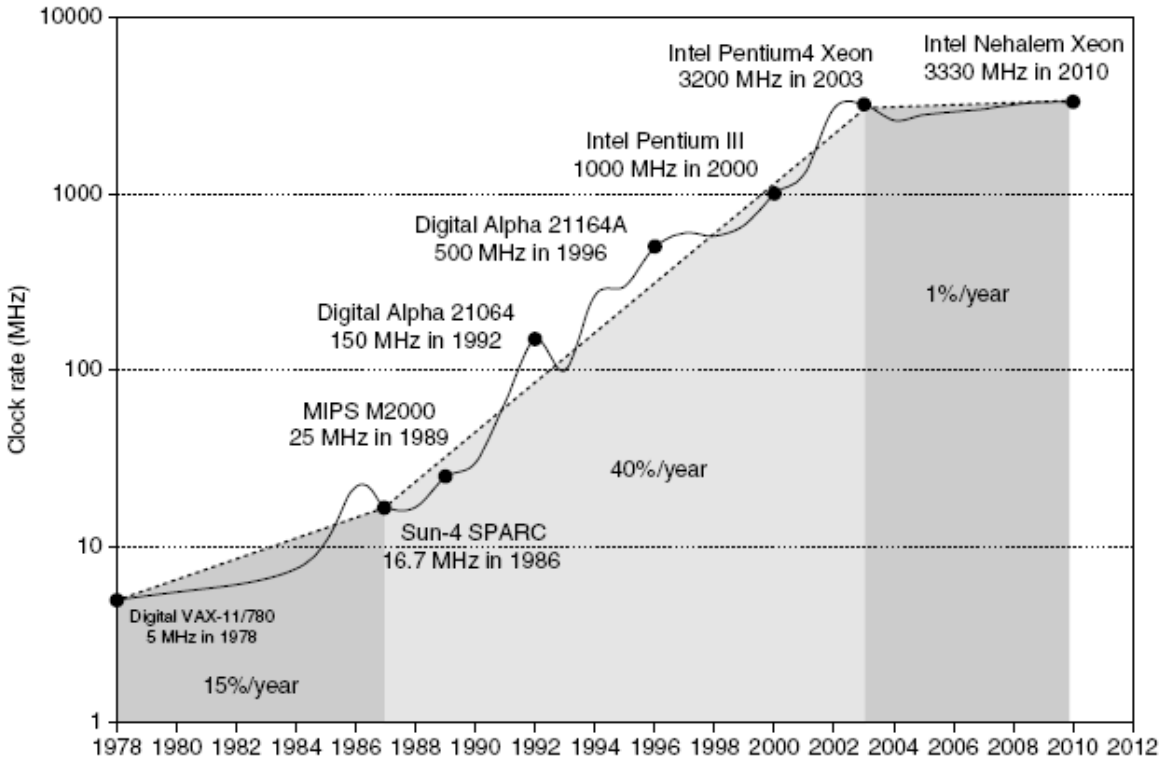
- For mobile devices, energy is better metric:

$$Energy_{dynamic} = \frac{1}{2} CapacitiveLoad \times Voltage^2$$

- For fixed task, slowing clock rate reduces power, not energy
- Capacitive load a function of number of transistors connected to output and of technology, which determines capacitance of wires and transistors
- Dropping voltage helps both, moved from 5V to 1V
- Clock gating

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



How to reduce power?

- Several techniques
  - Design circuits to be energy efficient
    - some clever techniques avoid wasteful energy consumption
    - other techniques adversely affect performance
  - Reduce clock rate
    - frequency scaling
  - Reduce voltage and clock rate
    - voltage-frequency scaling
    - done dynamically, in response to demand
  - Clock gating and voltage gating
    - turn off parts of the system
    - e.g., turn off unneeded cores, put memory/disks in idle

Voltage and Frequency Scaling

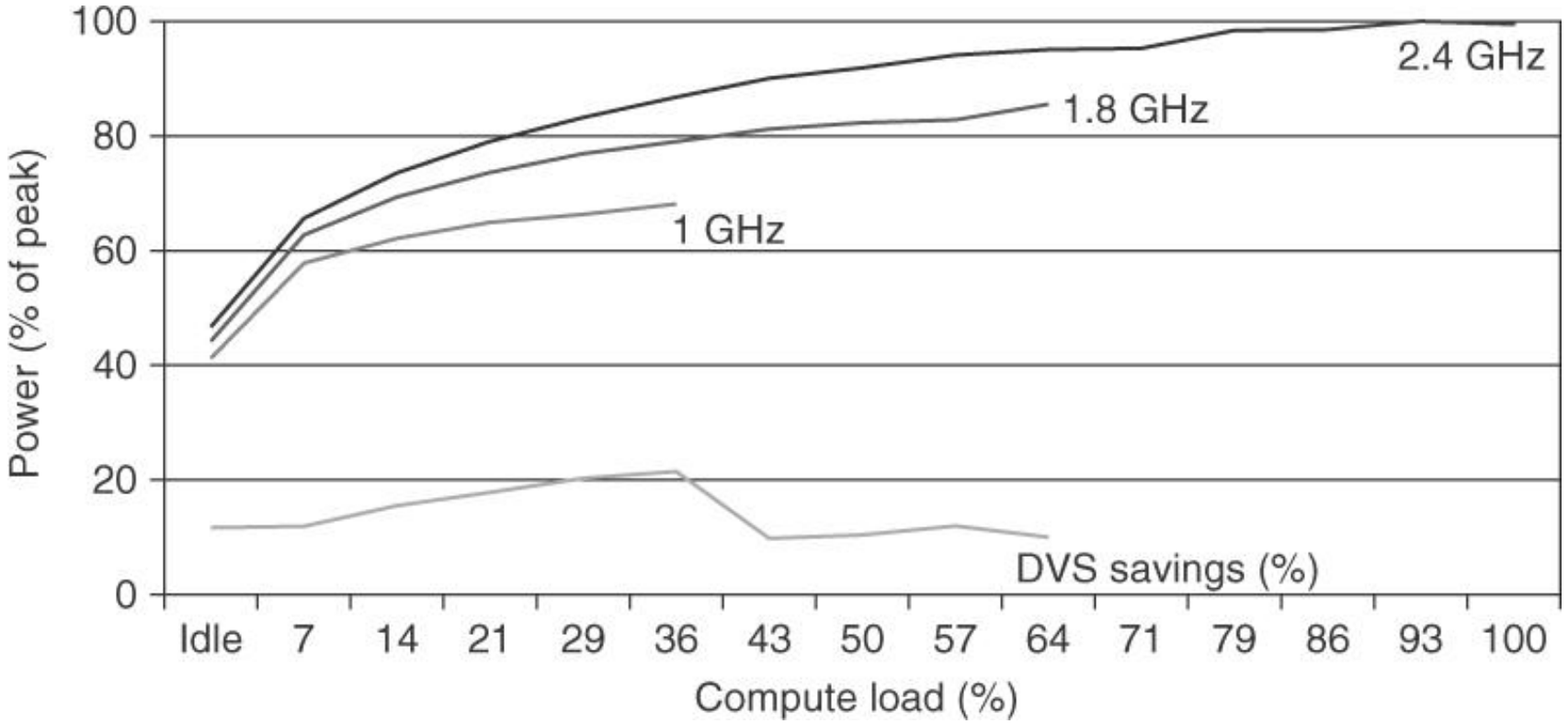


Figure 1.12 Energy savings for a server using an AMD Opteron microprocessor, 8 GB of DRAM, and one ATA disk. At 1.8 GHz, the server can only handle up to two-thirds of the workload without causing service level violations, and, at 1.0 GHz, it can only safely handle one-third of the workload. (Figure 5.11 in Barroso and Hölzle [2009].)

Example: Reducing Power

- Suppose 15% reduction in voltage results in a 15% reduction in frequency. What is impact on dynamic power?

$$
\begin{aligned}
Power_{dynamic} &= 1/2 \times CapacitiveLoad \times Voltage^2 \times FrequencySwitched \\
&= 1/2 \times .85 \times CapacitiveLoad \times (.85 \times Voltage)^2 \times FrequencySwitched \\
&= (.85)^3 \times OldPower_{dynamic} \\
&\approx 0.6 \times OldPower_{dynamic}
\end{aligned}
$$

Trends in Power

- Because leakage current flows even when a transistor is off, now *static power* important too

$$Power_{static} = Current_{static} \times Voltage$$

  - Leakage current increases in processors with smaller transistor sizes
  - Increasing the number of transistors increases power even if they are turned off
  - In 2006, goal for leakage was 25% of total power consumption; high performance designs at 40%
  - Very low power systems even gate voltage to inactive modules to control loss due to leakage

Trends

- Now let's look at trends in
  - Bandwidth vs. Latency
  - Power
  - Cost
  - Dependability
  - Performance

Cost of Integrated Circuits

$$\text{Cost of IC} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging}}{\text{Final test yield}}$$

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

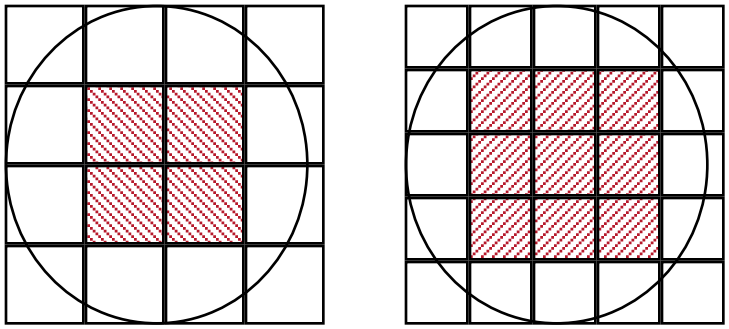
Dingwall's Equation

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

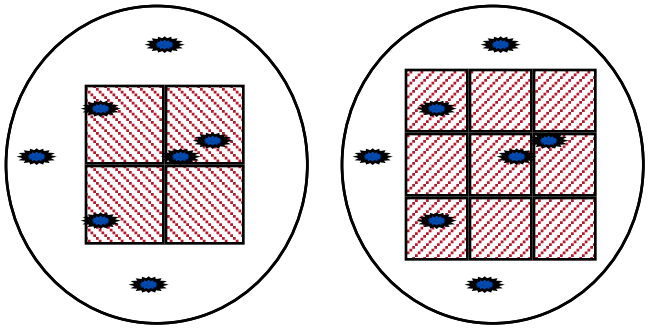
$$\text{Dies per wafer} = \frac{\pi\left(\frac{\text{Wafer diameter}}{2}\right)^2}{\text{Die area}} - \frac{\pi \times \text{Wafer diameter}}{\sqrt{2 \times \text{Die area}}} - \text{Test dies per wafer}$$

$$\text{Cost of testing die} = \frac{\text{Cost of testing per hour} \times \text{Average die test time}}{\text{Die yield}}$$

Explanations



Second term in "Dies per wafer" corrects for the rectangular dies near the periphery of round wafers



"Die yield" assumes a simple empirical model: defects are randomly distributed over the wafer, and yield is inversely proportional to the complexity of the fabrication process (indicated by α)

α=3 for modern processes implies that cost of die is proportional to (Die area)^4

Real World Examples

"Revised Model Reduces Cost Estimates", Linley Gwennap, Microprocessor Report 10(4), 25 Mar 1996

	Intel Pentium	AMD 5K86	Cyrix 6x86	MIPS R5000	PowerPC 603e	PowerPC 604	Pentium Pro	Sun UltraSparc	Hitachi SH7604
Process	BiCMOS	CMOS	CMOS	CMOS	CMOS	CMOS	BiCMOS	CMOS	CMOS
Line width (microns)	0.35	0.35	0.44	0.35	0.64	0.44	0.35	0.47	0.8
Metal layers	4	3	5	3	4	4	4	4	2
Wafer size (mm)	200	200	200	200	200	200	200	200	150
Wafer cost	$2,700	$2,200	$2,400	$2,600	$2,500	$2,300	$2,700	$2,200	$500
Die area (sq mm)	91	181	204	84	98	196	196	315	82
Effective area	85%	75%	85%	48%	65%	72%	85%	68%	75%
Dice/wafer	297	159	122	325	275	128	128	74	177
Defects/sq cm	0.6	0.8	0.7	0.8	0.5	0.8	0.6	0.8	0.5
Yield	65%	40%	36%	74%	74%	38%	42%	26%	75%
Die cost	$14	$40	$55	$11	$9	$47	$50	$116	$4
Package size (pins)	296	296	296	272	240	304	387	521	144
Package type	PGA	PGA	PGA	PBGA	CQFP	CQFP	MCM	PGA	PQFP
Package cost	$18	$21	$21	$11	$14	$21	$40	$45	$3
Test & assembly cost	$8	$10	$10	$6	$6	$12	$21	$28	$1
Total mfg cost	$40	$71	$86	$28	$29	$80	$144	$189	$8

Trends

- Now let's look at trends in
  - Bandwidth vs. Latency
  - Power
  - Cost
  - Dependability
  - Performance

Dependability

- When is a system operating properly?
- Infrastructure providers now offer Service Level Agreements (SLA) to guarantee that their networking or power service would be dependable
- Systems alternate between 2 states of service with respect to an SLA:
  - Service accomplishment, where the service is delivered as specified in SLA
  - Service interruption, where the delivered service is different from the SLA
- Failure = transition from state 1 to state 2
- Restoration = transition from state 2 to state 1

Definitions

Module reliability = measure of continuous service accomplishment (or time to failure)

- Two key metrics:
  - Mean Time To Failure (MTTF) measures Reliability
  - Failures In Time (FIT) = 1/MTTF, the rate of failures
    - Traditionally reported as failures per billion hours of operation
- Derived metrics:
  - Mean Time To Repair (MTTR) measures Service Interruption
    - Mean Time Between Failures (MTBF) = MTTF+MTTR
  - Module availability measures service as alternate between the 2 states of accomplishment and interruption (number between 0 and 1, e.g. 0.9)
    - Module availability = MTTF / (MTTF + MTTR)

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 FIT and MTTF for 10 disks (1M hour MTTF per disk), 1 disk controller (0.5M hour MTTF), and 1 power supply (0.2M hour MTTF):

$$FailureRate =$$

1M hours = 114 years!

0.2M hours = 22 years!

$$MTTF =$$

Solution next

Solution

- 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 FIT and MTTF for 10 disks (1M hour MTTF per disk), 1 disk controller (0.5M hour MTTF), and 1 power supply (0.2M hour MTTF):

$$
\begin{aligned}
FailureRate &= 10 \times (1/1{,}000{,}000) + 1/500{,}000 + 1/200{,}000 \\
&= 10 + 2 + 5/1{,}000{,}000 \\
&= 17/1{,}000{,}000 \\
&= 17{,}000 FIT \\
MTTF &= 1{,}000{,}000{,}000/17{,}000 \\
&\approx 59{,}000 hours
\end{aligned}
$$

less than 7 years!

Trends

- Now let's look at trends in
  - Bandwidth vs. Latency
  - Power
  - Cost
  - Dependability
  - Performance

First, What is Performance?

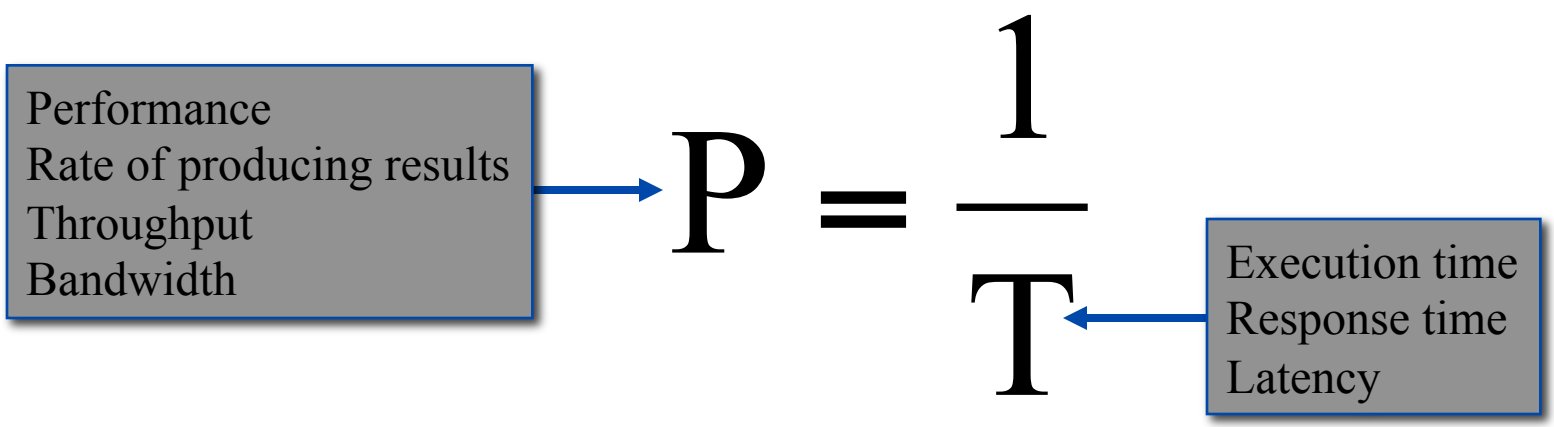
- The starting point is universally accepted
  - "The time required to perform a specified amount of computation is the ultimate measure of computer performance"
- How should we summarize (reduce to a single number) the measured execution times (or measured performance values) of several benchmark programs?
  - Two properties
    - A single-number performance measure for a set of benchmarks expressed in units of time should be directly proportional to the total (weighted) time consumed by the benchmarks.
    - A single-number performance measure for a set of benchmarks expressed as a rate should be inversely proportional to the total (weighted) time consumed by the benchmarks.

from "Characterizing Computer Performance with a Single Number", J. E. Smith, CACM, October 1988, pp. 1202-1206

Quantitative Principles of Computer Design

- Performance is in units of things per second
  - So bigger is better
- What if we are primarily concerned with response time?

$$\left[\frac{\text{work / results / program / instructions / bits}}{\text{time}}\right]$$



$$\left[\frac{\text{time}}{\text{work / result / program / instruction / bit}}\right]$$

Performance: What to measure?

- What about just MIPS and MFLOPS?
- Usually rely on benchmarks vs. real workloads
- Older measures were
  - Kernels or
  - Small programs designed to mimic real workloads
- Whetstone, Dhrystone
- http://www.netlib.org/benchmark
- Note LINPACK and Top500

MIPS

$$\text{CPU time} = \frac{\text{CPI} \times \text{Instruction count}}{\text{Clockrate}}$$

$$\frac{\text{Clockrate}}{\text{CPI}} = \frac{\text{Instruction count}}{\text{CPU time}}$$

$$\frac{\text{Clockrate}}{\text{CPI} \times 10^6} = \frac{\text{Instruction count}}{\text{CPU time} \times 10^6} = \text{MIPS}$$

- Machines with different instruction sets?
- Programs with different instruction mixes?
- Uncorrelated with performance
  - Marketing metric
- "Meaningless Indicator of Processor Speed"

MFLOP/s

$$\text{MFLOP/s} = \frac{\text{Number of FP operations}}{\text{CPU time} \times 10^6}$$

- Popular in supercomputing community
- Often not where time is spent
- Not all FP operations are equal
  - "Normalized" MFLOP/s
- Can magnify performance differences
  - A better algorithm (e.g., with better data reuse) can run faster even with higher FLOP count

Peak Performance

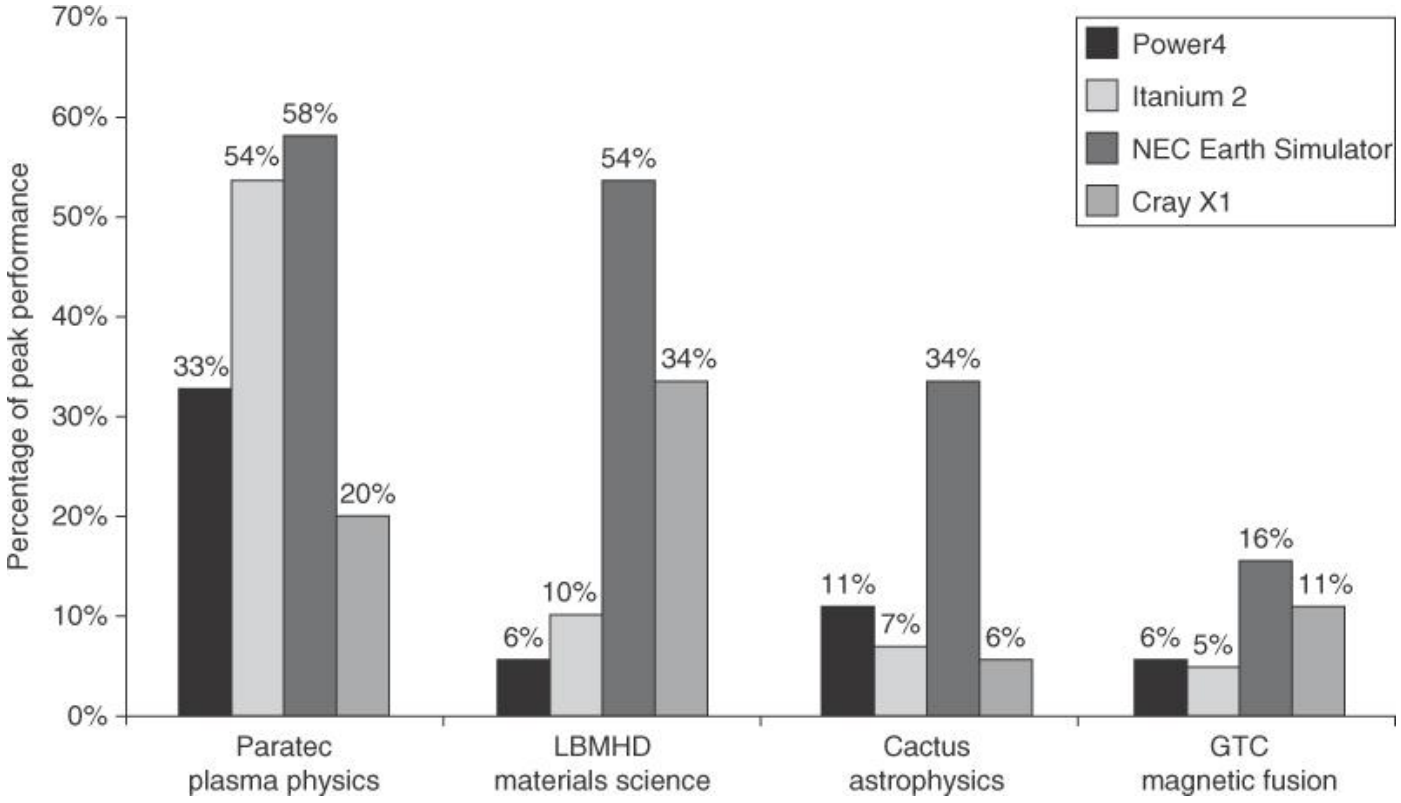


Figure 1.20 Percentage of peak performance for four programs on four multiprocessors scaled to 64 processors. The Earth Simulator and X1 are vector processors (see Chapter 4 and Appendix G). Not only did they deliver a higher fraction of peak performance, but they also had the highest peak performance and the lowest clock rates. Except for the Paratec program, the Power 4 and Itanium 2 systems delivered between 5% and 10% of their peak. From Oliker et al. [2004].

Benchmarks

- To increase predictability, collections of benchmark applications, called *benchmark suites*, are popular
- SPECCPU: popular desktop benchmark suite
  - CPU only, split between integer and floating point programs
  - SPECint2000 has 12 integer, SPECfp2000 has 14 integer pgms
  - SPECCPU2006 was announced Spring 2006
  - SPECSFS (NFS file server) and SPECWeb (WebServer) added as server benchmarks
  - www.spec.org
- Transaction Processing Council measures server performance and cost-performance for databases
  - TPC-C Complex query for Online Transaction Processing
  - TPC-H models ad hoc decision support
  - TPC-W a transactional web benchmark
  - TPC-App application server and web services benchmark

SPEC2006 Programs

SPEC2006 benchmark description	Benchmark name by SPEC generation				
	SPEC2006	SPEC2000	SPEC95	SPEC92	SPEC89
GNU C compiler					gcc
Interpreted string processing			perl		espresso
Combinatorial optimization		mcf			li
Block-sorting compression		bzip2		compress	eqntott
Go game (AI)	go	vortex	go	sc	
Video compression	h264avc	gzip	ijpeg		
Games/path finding	astar	eon	m88ksim		
Search gene sequence	hmmer	twolf			
Quantum computer simulation	libquantum	vortex			
Discrete event simulation library	omnetpp	vpr			
Chess game (AI)	sjeng	crafty			
XML parsing	xalancbmk	parser			
CFD/blast waves	bwaves				fpppp
Numerical relativity	cactusADM				tomcatv
Finite element code	calculix				doduc
Differential equation solver framework	dealII				nasa7
Quantum chemistry	gamess				spice
EM solver (freq/time domain)	GemsFDTD			swim	matrix300
Scalable molecular dynamics (~NAMD)	gromacs		apsi	hydro2d	
Lattice Boltzman method (fluid/air flow)	lbm		mgrid	su2cor	
Large eddie simulation/turbulent CFD	LESlie3d	wupwise	applu	wave5	
Lattice quantum chromodynamics	milc	apply	turb3d		
Molecular dynamics	namd	galgel			
Image ray tracing	povray	mesa			
Spare linear algebra	soplex	art			
Speech recognition	sphinx3	equake			
Quantum chemistry/object oriented	tonto	facerec			
Weather research and forecasting	wrf	ammp			
Magneto hydrodynamics (astrophysics)	zeusmp	lucas			
		fma3d			
		sixtrack			

How to Summarize Performance?

- Arithmetic average of execution times??
  - But they vary in basic speed, so some would be more important than others in arithmetic average
- Could add weights per program, but how to pick weight?
  - Different companies want different weights for their products
- SPECRatio: Normalize execution times to reference computer, yielding a ratio proportional to performance =
  - time on reference computer / time on computer being rated
  - Spec uses an older Sun machine as reference

Ratios

- 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

Review: Different means

Let $\mathbf{r} = (r_1, \ldots, r_n)$ be an n-tuple of positive numbers, $n > 1$.

$$\text{Quadratic mean } Q(\mathbf{r}) = \sqrt{\frac{r_1^2 + \cdots + r_n^2}{n}} = \sqrt{\frac{\sum_i r_i^2}{n}}$$

$$\text{Arithmetic mean } A(\mathbf{r}) = \frac{r_1 + \cdots + r_n}{n} = \frac{\sum_i r_i}{n}$$

$$\text{Geometric mean } G(\mathbf{r}) = \left(r_1 \times \cdots \times r_n\right)^{1/n} = \sqrt[n]{\prod_i r_i}$$

$$\text{Harmonic mean } H(\mathbf{r}) = \frac{1}{\left(\frac{\frac{1}{r_1} + \cdots + \frac{1}{r_n}}{n}\right)} = \left(\frac{\sum_i r_i^{-1}}{n}\right)^{-1}$$

Geometric Mean

- 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

Different Take

- Smith (CACM 1988, see references) takes a different view on means
- First let's look at an example

TABLE I. Performance of Three Computers on Two Benchmarks

Benchmark	Millions of floating pt. ops.	Computer 1 time (secs.)	Computer 2 time (secs.)	Computer 3 time (secs.)
Program 1	100	1	10	20
Program 2	100	1000	100	20
Total Time		1001	110	40

Rates

- Change to MFLOPS and also look at different means

TABLE II. Performance of Benchmarks in Mflops

Benchmark	Computer 1	Computer 2	Computer 3
Program 1	100.0 mflops	10.0 mflops	5.0 mflops
Program 2	.1 mflops	1.0 mflops	5.0 mflops
Arith. Mean	50.1 mflops	5.5 mflops	5.0 mflops
Geom. Mean	3.2 mflops	3.2 mflops	5.0 mflops
Harm. Mean	.2 mflops	1.8 mflops	5.0 mflops

TABLE I. Performance of Three Computers on Two Benchmarks

Benchmark	Millions of floating pt. ops.	Computer 1 time (secs.)	Computer 2 time (secs.)	Computer 3 time (secs.)
Program 1	100	1	10	20
Program 2	100	1000	100	20
Total Time		1001	110	40

Avoid the Geometric Mean?

- If benchmark execution times are normalized to some reference machine, and means of normalized execution times are computed, only the geometric mean gives consistent results no matter what the reference machine is
  - This has led to declaring the geometric mean as the preferred method of summarizing execution time (e.g., SPEC)
- Smith's comments
  - "The geometric mean does provide a consistent measure in this context, but it is consistently wrong."
  - "If performance is to be normalized with respect to a specific machine, an aggregate performance measure such as total time or harmonic mean rate should be calculated before any normalizing is done. That is, benchmarks should not be individually normalized first."
  - He advocates using time, or normalizing after taking mean

Variability

- Does a single mean summarize performance of programs in benchmark suite?
- Can decide if good predictor by characterizing variability of distribution using standard deviation
- Like geometric mean, geometric standard deviation is multiplicative rather than arithmetic
- Can simply take the logarithm of SPECRatios, compute the standard mean and standard deviation, and then take the exponent to convert back:

$$GeometricMean = \exp\left(\frac{1}{n} \times \sum_{i=1}^{n} \ln\left(SPECRatio_i\right)\right)$$

$$GeometricStDev = \exp\left(StDev\left(\ln\left(SPECRatio_i\right)\right)\right)$$

Form of Standard Deviation

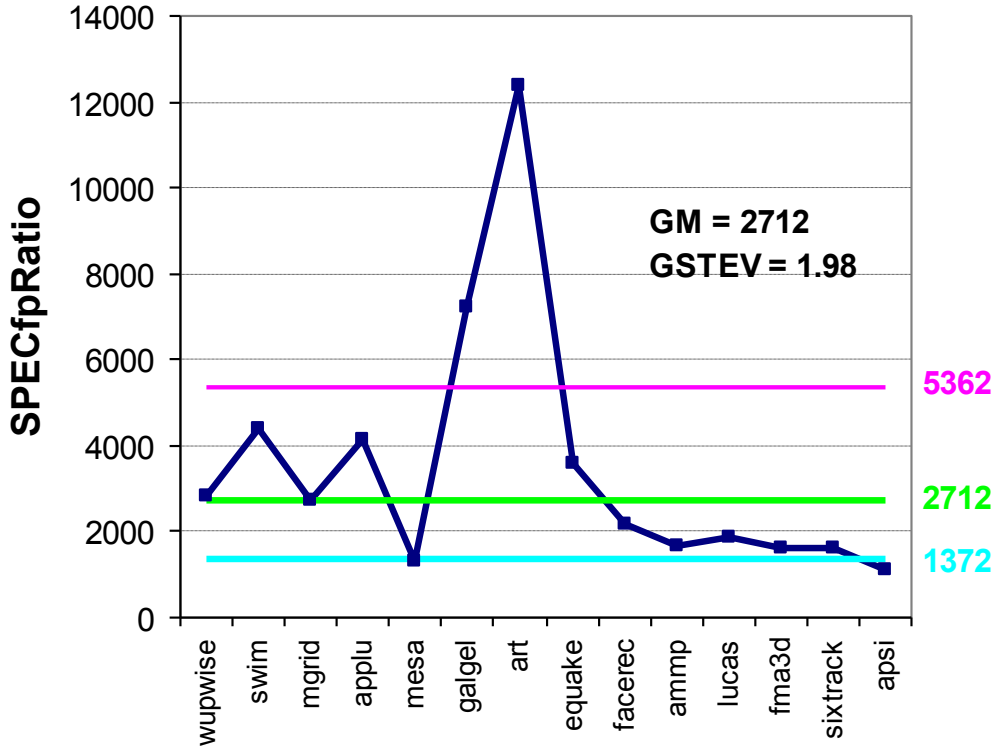
- Standard deviation is more informative if we know distribution has a standard form
  - *bell-shaped normal distribution*, whose data are symmetric around mean
  - *lognormal distribution*, where logarithms of data--not data itself--are normally distributed (symmetric) on a logarithmic scale
- For a lognormal distribution, we expect that

68% of samples fall in range $\left[mean / gstdev, mean \times gstdev\right]$

95% of samples fall in range $\left[mean / gstdev^2, mean \times gstdev^2\right]$

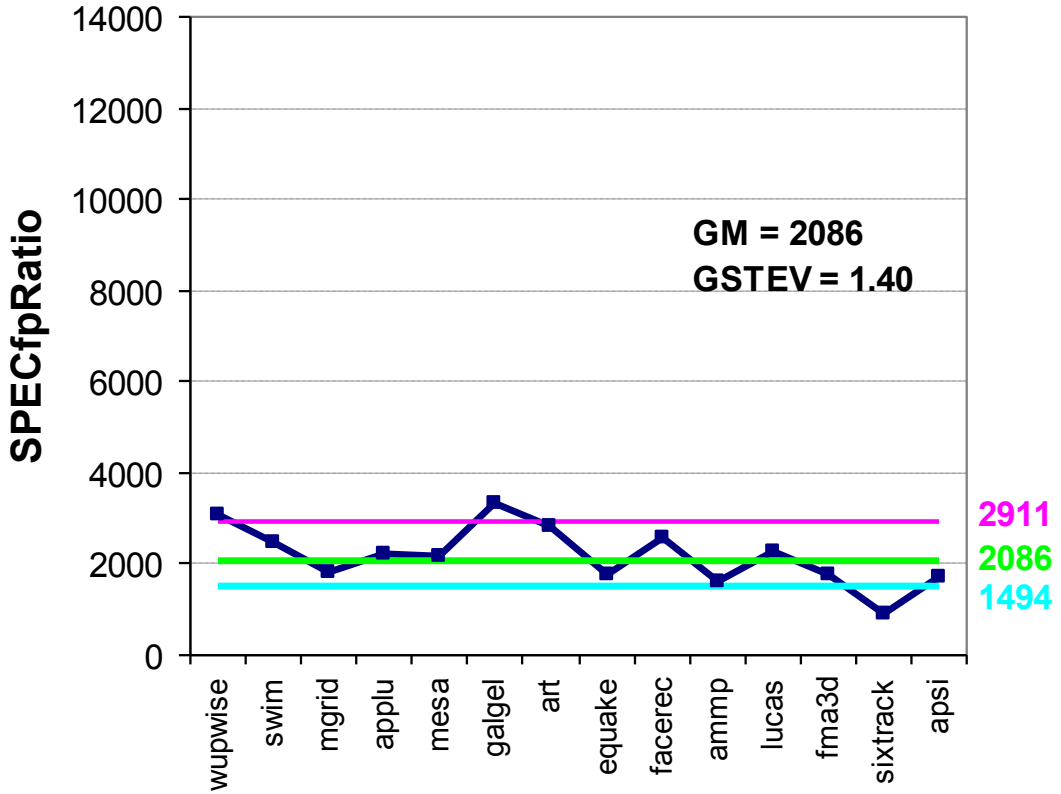
Example (1/2)

- GM and multiplicative StDev of SPECfp2000 for Itanium 2



Example (2/2)

- GM and multiplicative StDev of SPECfp2000 for AMD Athlon



Comments

- Standard deviation of 1.98 for Itanium 2 is much higher-- vs. 1.40--so results will differ more widely from the mean, and therefore are likely less predictable
- Falling within one standard deviation:
  - 10 of 14 benchmarks (71%) for Itanium 2
  - 11 of 14 benchmarks (78%) for Athlon
- Thus, the results are quite compatible with a lognormal distribution (expect 68%)

Next Lecture

- Principles of Computer Design
- Amdahl's Law

Readings/References

- Gordon Moore's paper
  - http://www.intel.com/pressroom/kits/events/moores_law_40th/index.htm
  - http://download.intel.com/museum/Moores_Law/Articles-Press_Releases/Gordon_Moore_1965_Article.pdf
- Paper on which latency section is based
  - Patterson, D. A. 2004. Latency lags bandwidth. *Commun. ACM* 47, 10 (Oct. 2004), 71-75.
- "Characterizing Computer Performance with a Single Number", J. E. Smith, CACM, October 1988, pp. 1202-1206