Comp4611 Tutorial 1

Computer Processor History
Die Cost Calculation
Performance Measuring & Evaluation

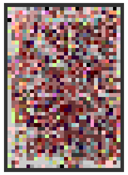
Sept. 9, 2013

Overview

- **Computer Processor History**
  - **A Brief Summary of Intel Microprocessors**
- Die Cost Calculation
- Performance Measuring & Evaluation
- Appendix: Amdahl's Law & Example

Advances Come from Design
- A Brief History of Intel Microprocessors

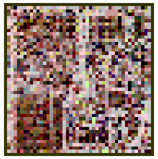
- 4 bits processor



4004 (1971)
- Intel's first microprocessor

- 8 bits processors

8008 (1972)
- twice as powerful as the 4004



8080 (1974)
- brains of the first personal computer
- ~US$ 400

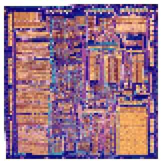
- 16 bits processor

8086 – 8088 (1978)
- brains of IBM's new hit product -- the IBM PC
- The 8088's success propelled Intel into the ranks of the Fortune 500, and Fortune magazine named the company one of the "Business Triumphs of the Seventies."

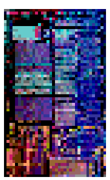
80286 (1982)
- first Intel processor that could run all the software written for its predecessor
- Within 6 years of its release, an estimated 15 million 286-based personal computers were installed around the world.

Advances Come from Design - in the 80's

- 32 bits processors

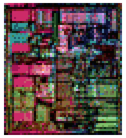


80386 (1985)
- 275,000 transistors--more than 100times as many as the original 4004
- 32-bit chip
- "multi tasking"

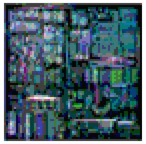


80486 (1989)
- 32 bit chip
- built-in math coprocessor
- packaged together with cache memory chip
- command-level computer → point-and-click computing
- color computer

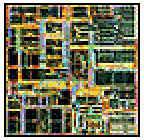
Advances Come from Design - in the 90's



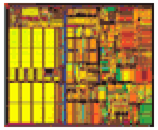
Pentium (1993)
- incorporate "real world" data such as speech, sound, handwriting and photographic images



Pentium Pro (1995)
- 5.5 million transistors
- packaged together with a second speed-enhancing cache memory chip,
- pipelining
- enabling fast computer-aided design, mechanical engineering and scientific computation

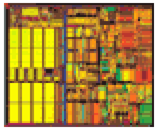


Pentium II (1997)
- 7.5 million-transistor
- MMX technology, designed specifically to process video, audio and graphics data efficiently
- high-speed cache memory chip



Celeron (1999)
- excellent performance in gaming

Advances Come from Design - in the Millenniums'



Pentium III (1999)
- 9.5 million transistors, 0.25-micron technology
- 70 new SSE (Streaming SIMD Extension) instructions
- dramatically enhance the performance of advanced imaging, 3-D, streaming audio, video and speech recognition applications, Internet experiences

Pentium 4 (2000)
- 42 million transistors and circuit lines of 0.18 microns
- 1.5 gigahertz (4004 ran at 108 kilohertz)
- SSE2 instructions, more pipeline stages, higher successful prediction rate
- can create professional-quality movies; deliver TV-like video via the Internet; communicate with real-time video and voice; render 3D graphics in real time; quickly encode music for MP3 players; and simultaneously run several multimedia applications while connected to the Internet.

Advances Come from Design - in last decade

- 64 bits processors

Pentium D (2005)

- Dual-core processing technology
  → high-end entertainment: multimedia entertainment, digital photo editing, multiple users and multitasking

Pentium Dual-Core (2007)

- High-value performance for multitasking (CPU executes more instructions in less time)
- Smart Cache: smarter, more efficient cache and bus design
  → enhanced performance, responsiveness and power savings

Core 2 Duo (2006)

- A dual-core CPU
- A new microarchitecture to replace Netburst
- Memory Hierarchy System
- Low power consumption

Core™2 Quad (2008)

- Four execution cores
- More intensive entertainment and more media multitasking

Advances Come from Design - Nowadays

Core™ i series (2008)

- aims at
  - Reducing idle power
  - Boosting performance by increasing processor frequency
  - Hyper-threading

Core™ i3
- A dual-core CPU

Core™ i5
- A quad-core CPU

Core™ i7
- A quad-core CPU (up to 8 cores)

Advances Come from Technology - A Comparison

Processor	Intel® Pentium® D Processor	Intel® Pentium® Dual-Core Processor	Intel® Core™2 Duo Processor	Intel® Core™2 Quad Processor	Intel® Core™i-series Processor
Process Technology	65 nm - 90 nm	65 nm	65 nm	65 nm	45nm
L2 Cache	1MB - 2MB for each core	1MB	2M - 4M	8M	3M – 8M
Clock Speed	2.80 - 3.60 GHz	1.6 - 2 GHz	1.86 – 3.0 GHz	2.4 - 2.66 GHz	1.6 GHz – 2.9 GHz
Chipset	Intel® 945P, 945G, 955X, 975X chipsets	N/A	Intel® Q965, Q963, G965, P965, 975X	Intel® P965, 975X	Intel® Z87, H87, H81 chipsets

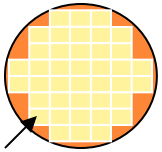
***Increase* in *processor performance* due to the *growth in CPU* Transistor Count**

Overview

- Computer Processor History
  - A Brief Summary of Intel Microprocessors
- **Die Cost Calculation**
- Performance Measuring & Evaluation
- Appendix: Amdahl’s Law & Example

Cost of an Integrated Circuit

$$\text{IC Cost} = \frac{\text{Die Cost} + \text{Testing Cost} + \text{Packaging Cost}}{\text{Final Test Yield}}$$



$$\text{Die Cost} = \frac{\text{Wafer Cost}}{(\text{Dies/Wafer}) \times \text{Die Yield}}$$

$$\text{Dies per Wafer} = \frac{\pi(\text{Wafer diameter}/2)^2}{\text{Die Area}} - \frac{\pi(\text{Wafer diameter})}{(2 \times \text{Die Area})^{1/2}}$$

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

Where **α** is a parameter inversely proportional to the number of mask Levels, which is a measure of the manufacturing complexity.
For today's CMOS process, good estimate is **α** = 3.0 – 4.0

Yield: the percentage of manufactured devices that survives the testing procedure

Example: Die Cost

Given:

wafer 30cm, die 1cm,
defect density 0.6 per cm^2 ,
$\alpha=4.0$
30-cm-diameter wafer with 3-4 metal layers : $3500
wafer yield is 100%

Calculate:

die cost

To calculate the die cost,

$$\text{Die Cost} = \frac{\text{Wafer Cost}}{\left(\text{Dies}/\text{Wafer}\right) \times \text{Die Yield}}$$

- Given wafer cost
- Dies/Wafer?
- Die Yield?

Example: Die Cost

Given:

wafer 30cm, die 1cm,
defect density 0.6 per cm^2 ,
α=4.0
30-cm-diameter wafer with 3-4 metal layers : $3500
wafer yield is 100%

Calculate:

die cost

Step 1: dies per wafer

$$\text{Dies per Wafer} = \frac{\pi(\text{Wafer diameter}/2)^2}{\text{Die Area}} - \frac{\pi(\text{Wafer diameter})}{(2 \times \text{Die Area})^{1/2}}$$

$$= \frac{\pi(30/2)^2}{1\times 1} - \frac{\pi \times 30}{\sqrt{2\times 1\times 1}} = 640$$

Example: Die Cost

Given:

wafer 30cm, die 1cm,
defect density 0.6 per cm^2 ,
$\alpha=4.0$
30-cm-diameter wafer with 3-4 metal layers : \$3500
wafer yield is 100%

Calculate:

die cost

Step 2: die yield

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

$$= 1 \times \left(1+\frac{0.6 \times 1}{4.0}\right)^{-4} = 0.57$$

Example: Die Cost

Given:
wafer 30cm, die 1cm,
defect density 0.6 per cm^2,
α=4.0
30-cm-diameter wafer with 3-4 metal layers : $3500
wafer yield is 100%

Calculate:
die cost

Step 3: die cost

$$\text{Die Cost} = \frac{\text{Wafer Cost}}{(\text{Dies/Wafer}) \times \text{Die Yield}}$$

$$= \frac{3500}{640 \times 0.57} = 9.59$$

Overview

- Computer Processor History
  - A Brief Summary of Intel Microprocessors
- Die Cost Calculation
- **Performance Measuring & Evaluation**
- Appendix: Amdahl's Law & Example

How to Measure Performance ?

- Performance Rating
  - CPU Time
  - Benchmark programs
    - Integer programs and floating point programs
      - Compression
      - Compiler
      - Artificial Intelligence
      - Physics / Quantum Computing
      - Video Compression
      - Path-finding Algorithms
  - Comparing & Summarizing Performance

Measuring Performance - CPU Execution Time

- Performance = 1 / Execution Time

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

Instruction count

CPI

Clock cycle = 1 / Clock rate

Measuring CPU Time – Example 1

A SPEC CPU2006 integer benchmark (464.h264ref, a video compression program written in C) is run on a Pentium D processor:

Total instruction count:	3731 billion
Average CPI for the program:	2.5 cycles/instruction.
CPU clock rate:	2.1 GHz

$$\boxed{\text{CPU time} = \text{Instruction count} \times \text{CPI} \times (1/\text{clock rate})}$$

$$\begin{aligned}\text{CPU time} &= 3731 \times 10^9 \times 2.5 / (2.1 \times 10^9) \\ &= 4442 \text{ seconds}\end{aligned}$$

Source: Analysis of Redundancy and Application Balance in the SPEC CPU2006 Benchmark Suite

Comparing Performance – Example 2

Suppose SPEC CPU2006 integer benchmark (464.h264ref) is run on a faster processor, with a new compiler:

New total instruction count:	2000 billion
New average CPI for the program:	4 cycles/instruction.
New CPU clock rate:	3.6 GHz

$$\textbf{Speedup} = \frac{\textbf{Old Execution Time}}{\textbf{New Execution Time}} = \frac{I_{old} \times CPI_{old} \times \text{Clock cycle}_{old}}{I_{new} \times CPI_{new} \times \text{Clock Cycle}_{new}}$$

$$\text{New CPU time} = 2000 \times 10^9 \times 4 / (3.6 \times 10^9) = 2222 \text{ seconds}$$

$$\text{Speedup} = \frac{\text{Old CPU time}}{\text{New CPU time}} = \frac{4442}{2222} = 1.999$$

Comparing Performance – Example 3

$$\text{Speedup} = \frac{\text{Old Execution Time}}{\text{New Execution Time}} = \frac{I_{old} \times CPI_{old} \times \text{Clock cycle}_{old}}{I_{new} \times CPI_{new} \times \text{Clock Cycle}_{new}}$$

Question: Should the new design be implemented?

Instruction Class	Frequency	Old CPI	New CPI
ALU	40%	3	2
Load	20%	1	2
Store	20%	1	2
Branch	20%	2	3

Old CPI

$= 0.40 \times 3 + 0.2 \times 1 + 0.2 \times 1 + 0.2 \times 2$

$= 2$

New CPI

$= 0.4 \times 2 + 0.2 \times 2 + 0.2 \times 2 + 0.2 \times 3$

$= 2.2$

Speedup

$$= \frac{\cancel{I_{old}} \times 2 \times \cancel{\text{Clock cycle}_{old}}}{\cancel{I_{new}} \times 2.2 \times \cancel{\text{Clock cycle}_{new}}}$$

$= 0.91$

Answer: Shouldn't be implemented

Metrics for Performance

CPU time: most accurate and fair measure

$$\text{CPU Time} = \underline{\text{Instruction Count} \times \text{CPI}} \times \text{Clock Cycle Time}$$

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

$$CPI = \sum_{i=1}^{n} (\mathrm{CPI}_i \times F_i)$$

a priori frequency of the instruction set

Measuring Performance - Example 4

- Suppose we have made the following measurements:
  Frequency of FP operations (other than FPSQR) = 23%
  Average CPI of FP operations (other than FPSQR) = 4.0
  Frequency of FPSQR = 2%, CPI of FPSQR = 20
  Average CPI of other instructions = 1.33
- Assume that the two design alternatives
  - decrease the CPI of FPSQR to 3
  - decrease the average CPI of FP operations (other than FPSQR) to 2.
- Compare these two design alternatives using the CPU performance equation.

Solution

Step 1: Original CPI without enhancement:

$CPI_{original} = 4\times23\% + 20x2\% + 1.33\times75\% = 2.3175$

Step 2: compute the CPI for the enhanced FPSQR by subtracting the cycles saved from the original CPI:

$CPI_{with\ new\ FPSQR} = CPI_{original} - 2\%\times(CPI_{old\ FPSQR} - CPI_{new\ FPSQR\ only})$

$= 2.3175 - 0.02x(20-3) = 1.9775$

Step 3: compute the CPI for the enhancement of all FP instructions:

$CPI_{with\ new\ FP} = CPI_{original} - 23\%\times(CPI_{old\ FP} - CPI_{new\ FP})$

$= 2.3175 - 0.23x(4-2) = 1.8575$

Step 4: the speedup for the FP enhancement over FPSQR enhancement is:

$Speedup = CPU\ time_{with\ new\ FPSQR} / CPU\ time_{with\ new\ FP}$

$= (I \times CPI_{with\ new\ FPSQR} \times C) / (I \times CPI_{with\ new\ FP} \times C)$

$= CPI_{with\ new\ FPSQR} / CPI_{with\ new\ FP}$

$= 1.9775 / 1.8575 = 1.065$

Comparing & Summarizing Performance
- By total execution time

- Total execution time

	Machine A	Machine B
Program 1	1	10
Program 2	1000	100
Total	1001	110

How much faster is Machine B than Machine A?

9.1 times?

Machine A is faster in running Program 1.
Machine B is faster in running Program 2.

UNCLEAR

Comparing & Summarizing Performance

- by arithmetic mean of execution time

Arithmetic Mean: $$\frac{1}{n}\sum_{i=1}^{n} \text{Execution Time}_i$$

	Machine A	Machine B
Program 1	1	10
Program 2	1000	100
Total	1001	110
AM	500.5	55

	Machine A	Machine B
Program 1	200	400
Program 2	250	400
Program 3	450	100
AM	300	300

Can be misleading

Valid only if programs run equally

Comparing & Summarizing Performance
- by weighted arithmetic mean of execution time

Weighted Arithmetic Mean: $\sum_{i=1}^{n} \text{Weight}_i \times \text{Execution Time}_i$

For the 1st of Weights:

	Machine A	Machine B	W (1)
Program 1	200	400	0.4
Program 2	250	500	0.4
Program 3	550	100	0.2
AM	300	300	
WAM (1)	200 x 0.4 + 250 x 0.4 + 550 x 0.2 = 290	400 x 0.4 + 500 x 0.4 + 100 x 0.2 = 380	

Machine A is better

Comparing & Summarizing Performance
- by weighted arithmetic mean of execution time

Weighted Arithmetic Mean: $\sum_{i=1}^{n} \text{Weight}_i \times \text{Execution Time}_i$

For the 2nd set of Weights:

	Machine A	Machine B	W (2)
Program 1	200	400	0.2
Program 2	250	500	0.2
Program 3	550	100	0.6
AM	300	300	
WAM (2)	200 x 0.2 + 250 x 0.2 + 550 x 0.6 = 420	400 x 0.2 + 500 x 0.2 + 100 x 0.6 = 240	

Machine B is better

It depends very much on how to weigh each testing item

Comparing & Summarizing Performance

- by geometric mean of execution time

Geometric Mean: $\sqrt[n]{\prod_{I=1}^{n} \text{Execution time ratio}_i}$

	Machine A	Machine B
Program 1	1	10
Program 2	1000	100
Total	1001	110

Same GM
≠
same execution time or
same performance

Normalized Execution Time *to a reference machine*

	Normalized to A	
	Machine A	Machine B
Program 1	1	10
Program 2	1	0.1
GM	1	1

$$\frac{GeometricMean_A}{GeometricMean_B} = \frac{\sqrt[n]{\prod_{i=1}^{n} SPECRatioA_i}}{\sqrt[n]{\prod_{i=1}^{n} SPECRatioB_i}} = \sqrt[n]{\prod_{i=1}^{n} \frac{SPECRatioA_i}{SPECRatioB_i}}$$

$$= \sqrt[n]{\prod_{i=1}^{n} \frac{ExecutionTimeB_i}{ExecutionTimeA_i}} = \sqrt[n]{\prod_{i=1}^{n} \frac{PerformanceA_i}{PerformanceB_i}}$$

Overview

- Computer Processor History
  - A Brief Summary of Intel Microprocessors
- Die Cost Calculation
- Performance Measuring & Evaluation
- **Appendix: Amdahl's Law & Example**

Amdahl's Law

- Amdahl's Law – law of diminishing returns
- In general case, assume several enhancements has been taken for the system, the speedup for whole system is:

$$\text{Speedup}_{\text{overall}} = \frac{\text{Execution Time Without Enhancement}}{\text{Execution Time With Enhancement}} = \frac{1}{(1 - F1 - F2 \ldots) + \frac{F1}{S1} + \frac{F2}{S2} + \ldots}$$

where **F*i*** is the fraction of enhancement ***i*** and **S*i*** is the speedup of the corresponding enhancement

- The new execution time is

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

Amdahl's Law
- An Example

- Float instruction:
  - Fraction: 50%
  - Speedup: 2.0x
- Integer instruction:
  - Fraction: 30%
  - Speedup: 3.0x
- Others keep the same.

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

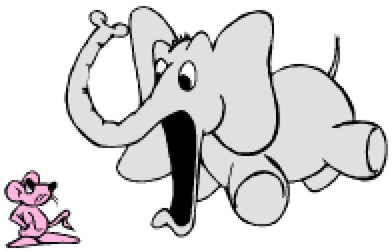
$$= 1 / ((1-0.5-0.3)+(0.5/2+0.3/3))=1.818$$

Amdahl's Law
- Intuition: "Make the common case faster"

- I have two processors, which can help accelerate one of the below parts by parallel processing. Two parts occupy the total time percentage of 95% and 5%.

- $\text{Fraction}_{\text{enhanced}} = 95\%$, $\text{Speedup}_{\text{enhanced}} = 2.0\text{x}$
  $\text{Speedup}_{\text{overall}} = 1/((1\text{-}0.95)+0.95/2) = 1.905$

- $\text{Fraction}_{\text{enhanced}} = 5\%$, $\text{Speedup}_{\text{enhanced}} = 2.0\text{x}$
  $\text{Speedup}_{\text{overall}} = 1/((1\text{-}0.05)+0.05/2) = 1.026$



1.905 vs. 1.026

Make the common case faster!!

- $\text{Fraction}_{\text{enhanced}} = 5\%$, $\text{Speedup}_{\text{enhanced}}$ -> infinity
  $\text{Speedup}_{\text{overall}} = 1/(1\text{-}0.05) = 1.052$

1.052 is still much smaller than 1.905.

A Common Confusion: CPI vs. Amdahl's Law

- Assume a program consists of three classes of instructions A,B and C, as shown below.
- An enhancement is made by doubling the speed of instruction class A
- Assume instruction count for the program and CPU clock cycle is not influenced
- What is the overall speedup achieved for the program

Instruction Class	Frequency	Old CPI	New CPI
A	20%	2	1
B	20%	3	3
C	60%	4	4

Method 1: CPI

$$CPI_{old} = 0.2\times2+0.2\times3+0.6\times4=3.4$$

$$CPI_{new} = 0.2\times1+0.2\times3+0.6\times4=3.2$$

$$Speedup_{overall} = \frac{\text{Execution time}_{\text{old}}}{\text{Execution time}_{\text{new}}} = \frac{\text{CPI}_{\text{old}}\times\text{IC}\times\text{Clock Cycle}}{\text{CPI}_{\text{new}}\times\text{IC}\times\text{Clock Cycle}} = 1.0625$$

~~Method 2: Amdahl's Law~~

$$\cancel{Speedup_{overall} = \frac{1}{(1-20\%)+\frac{20\%}{2}} = 1.111}$$

The fraction in Amdahl's law is **time fraction**

Method 2: Amdahl's Law

$$\text{Time fraction of A} = \frac{2\times20\%}{2\times20\%+3\times20\%+4\times60\%} = 0.11764$$

$$Speedup_{overall} = \frac{1}{(1-0.11764)+\frac{0.11764}{2}} = 1.0625$$

References:

- John L. Hennessy and David A. Patterson. Computer Architecture: A Quantitative Approach. Morgan Kaufman Publishers, 5th Edition, 2011
- Intel
  - http://www.intel.com
  - http://www.intel.com/content/www/us/en/history/museum-story-of-intel-4004.html
  - http://www.intel.com/pressroom/kits/quickrefyr.htm
- http://en.wikipedia.org/wiki/Microprocessor
- Timeline of Microprocessor http://m.theinquirer.net/inquirer/feature/2124781/microprocessor-development