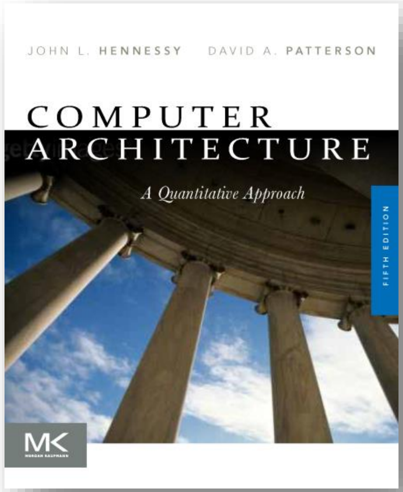


Chapter 1

Fundamentals of Quantitative Design and Analysis

Based on and extended from lecture note provided by publisher

Soontae Kim

Dept. of Computer Science

KAIST

Power and Energy

power 조절이 microprocessor 성장에 있어서 중요하다 -> generate heat -> generate temperature

- Problem: Get power in, get power out
- Max or peak power = maximum power consumed by microprocessor
  - More power than power supply provides=>voltage drop and malfunction (decrease reliability)
    modern processors provide voltage indexing methods that allow the processor to slow down and regulate voltage within a wider margin
- Thermal Design Power (TDP)
  - Sustained (long-term average) power consumption
  - Used as target for power supply and cooling system
    exceed the TDP match or exceed TDP
  - Lower than peak power, higher than average power consumption
    if current power>TDP (thermal temperature approaches the junction temperature limit) - cooling system can't handle the power -> clock rate reduced to reduced power consumption + second thermal overload trip is activated to power down the chip
- 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
  - Similar to distance

    capacitive load : a function of the number of transistors connected to an output and the technology, which determines the capacitance of the wires and the transistors.

- Dynamic power energy consumption per second
  power(watt) - 전력 = V*I 단위시간당 energy 의미
  - ½ x Capacitive load x Voltage^2 x Frequency switched
  - Similar to speed

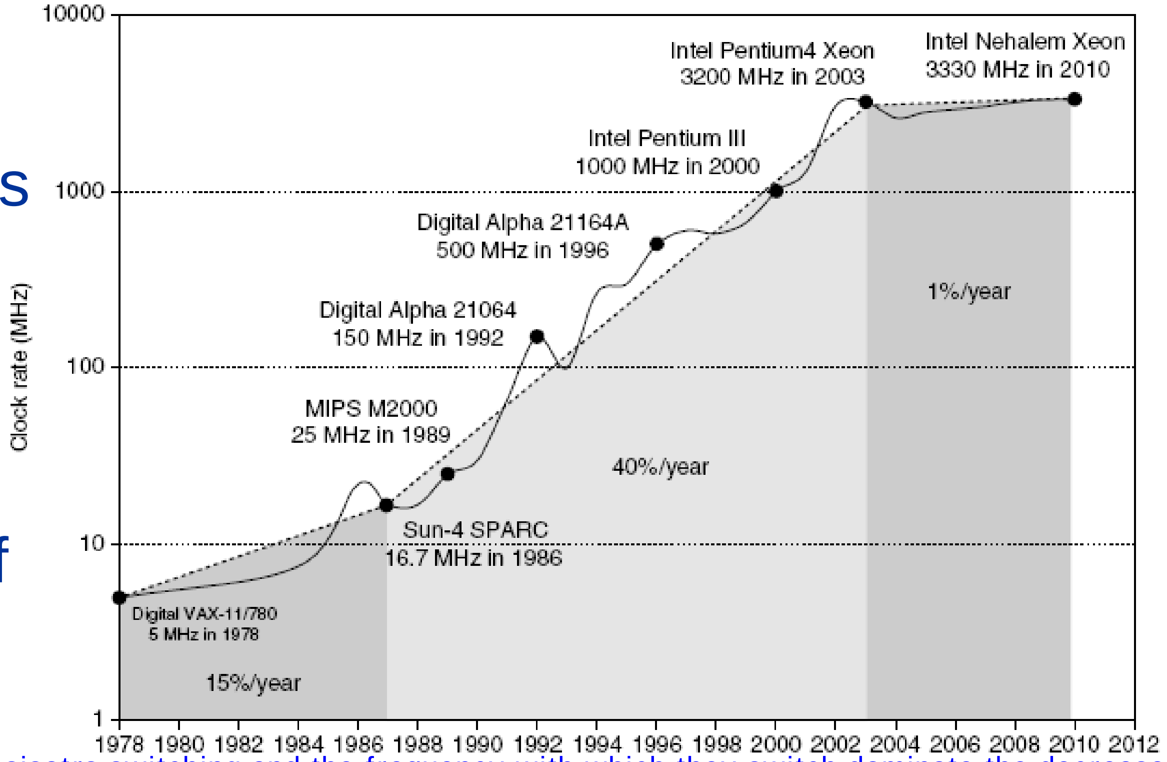
  reduce voltage -> transistor slow -> frequency low

- Reducing clock rate reduces power, not energy

  pitfall : 만약 clock rate를 낮추면 dynamic power가 반으로 감소하지만, 주어진 work를 처리할 때 걸리는 시간의 2배 정도 걸릴 수 있으므로 전체적인 dynamic energygy가 감소하지 않는다.

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



the increase in the number of transisotrs switching and the frequency with which they switch dominate the decrease in load capacitance and voltage, leading to an overall growth in power consumption and energy

Reducing Power

- Techniques for reducing power:
  - Do nothing well
    - Stop clock of inactive modules
  - Dynamic Voltage-Frequency Scaling (DVFS)
  - Low power state for DRAM, disks
  - Overclocking, turning off cores 일 한곳으로 몰아주기!

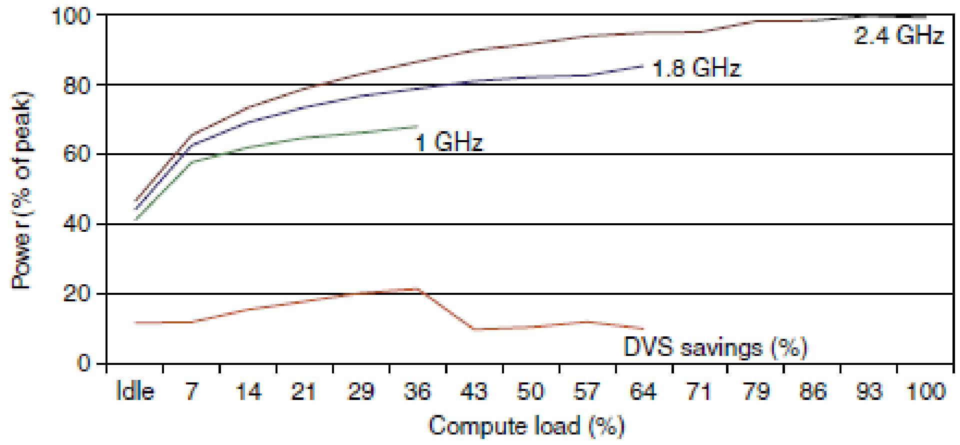


Figure 1.12 Energy savings for a server using an AMD Opteron microprocessor, 8 GB of DRAM, and one ATA disk. At 1.8 GHz, the server can only handle up to two-thirds of the workload without causing service level violations, and, at 1.0 GHz, it can only safely handle one-third of the workload. (Figure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

Static Power

- Static power consumption
  - $\text{Current}_{\text{static}}$ x Voltage
  - Scales with number of transistors
  - To reduce: power gating

leakage current flows even when a transistor is off
increasing the number of transistors increases power even if they are idle, and leakage current increase in processors with smaller transistor sizes.

Trends in Cost

- Cost driven down by learning curve
  - Manufacturing costs decrease over time by change in yield
- Yield
  - Percentage of manufactured devices that survive testing procedure
- DRAM: price closely tracks cost
- Microprocessors: price depends on volume
  - 10% less for each doubling of volume

Wafer

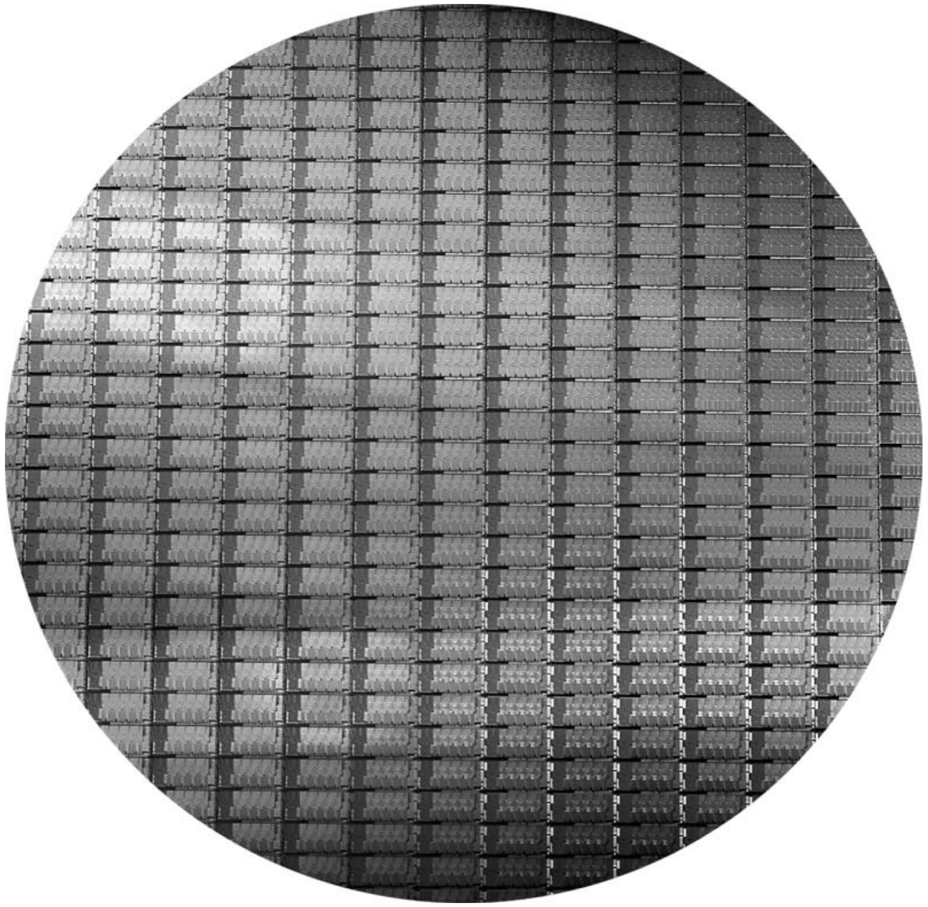


Figure 1.15 This 300 mm wafer contains 280 full Sandy Bridge dies, each 20.7 by 10.5 mm in a 32 nm process. (Sandy Bridge is Intel's successor to Nehalem used in the Core i7.) At 216 mm2, the formula for dies per wafer estimates 282. (Courtesy Intel.)

Microprocessor Die

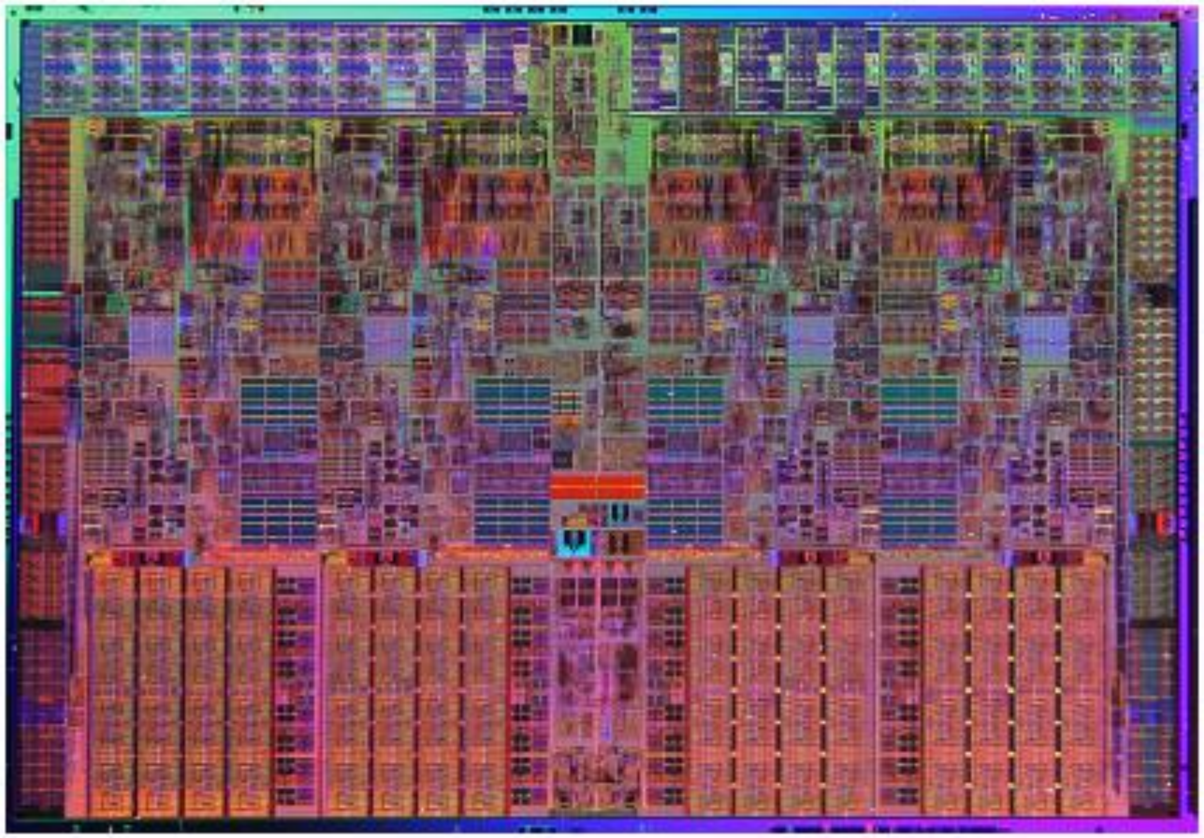


Figure 1.13 **Photograph of an Intel Core i7 microprocessor die, which is evaluated in Chapters 2 through 5.** The dimensions are 18.9 mm by 13.6 mm (257 mm^2) in a 45 nm process. (Courtesy Intel.)

Floorplan

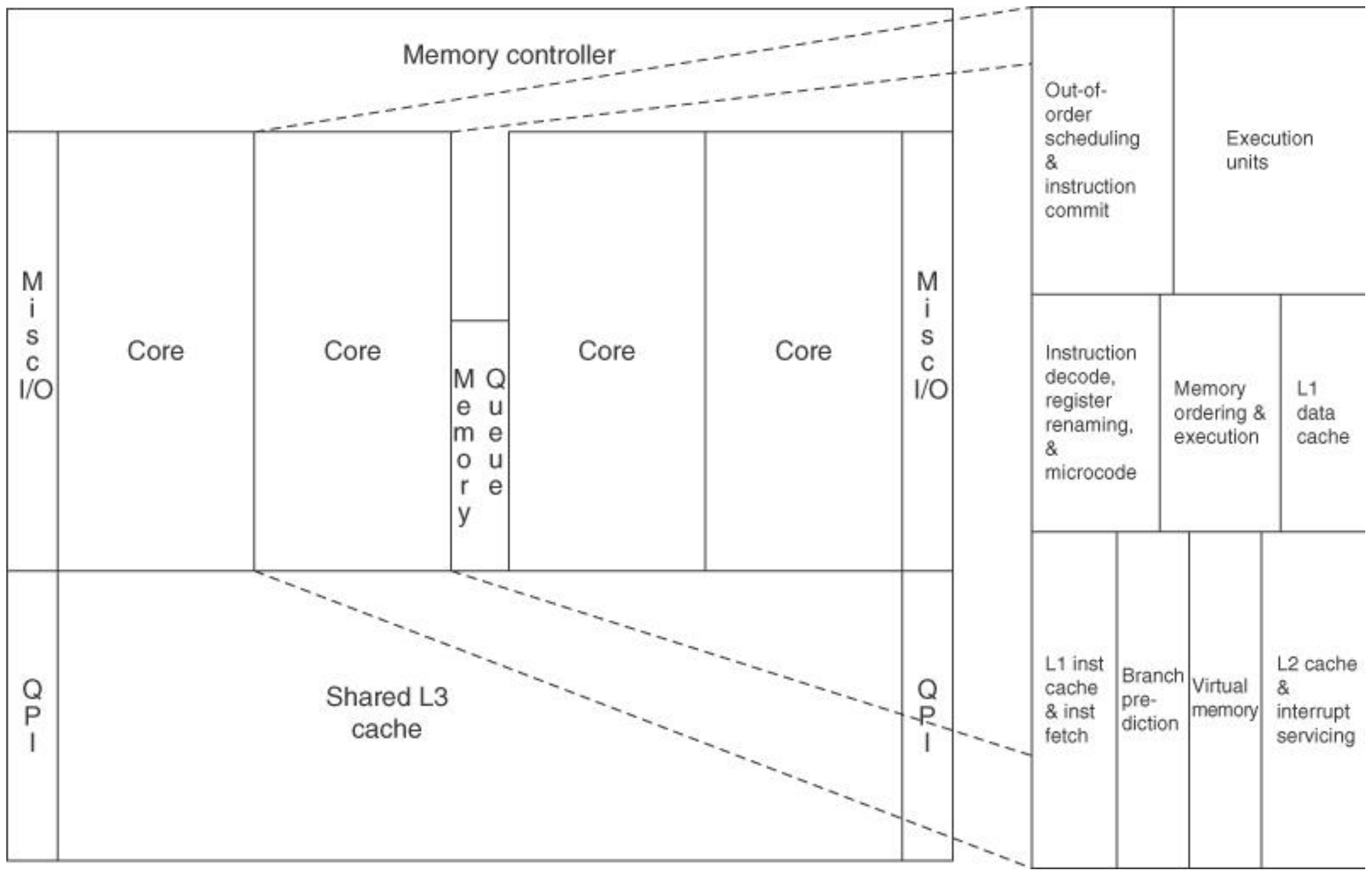

Memory controller
Misc I/O
Core
Core
Memory
Queue
Core
Core
Misc I/O
QPI
Shared L3 cache
QPI
Out-of-order scheduling & instruction commit
Execution units
Instruction decode, register renaming, & microcode
Memory ordering & execution
L1 data cache
L1 inst cache & inst fetch
Branch pre-diction
Virtual memory
L2 cache & interrupt servicing

Figure 1.14 Floorplan of Core i7 die in Figure 1.13 on left with close-up of floorplan of second core on right.

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)

Dependability

- Chips(ICs) have been reliable
  - However, transient and permanent faults are becoming common now
- Failures
  - Cause transitions b/t service accomplishment and interruption
- Module reliability
  - A measure of continuous service accomplishment of the time from a reference initial instant to failure
  - MTTF (mean time to failure) is used as a measure
- Failure rates
  - 1/MTTF
  - Use FIT (failures per billion hours of operation

Dependability (cont'd)

- Service interruption
  - MTTR (mean time to repair)
- MTBF (mean time b/t failure) = MTTF + MTTR
- Module availability
  - MTTF/(MTTF+MTTR)
- Coping with failures
  - Primarily use redundancy
  - Time redundancy 동일한 기계에 여러번 계산하고서 결과 비교
    sometimes can't determine some error in this case
    - Repeat operation to see if it is still erroneous
  - Resource redundancy 다른 기계에서 계산하고 계산 결과 비교
    - Have other components to take over from the one that failed

Measuring Performance

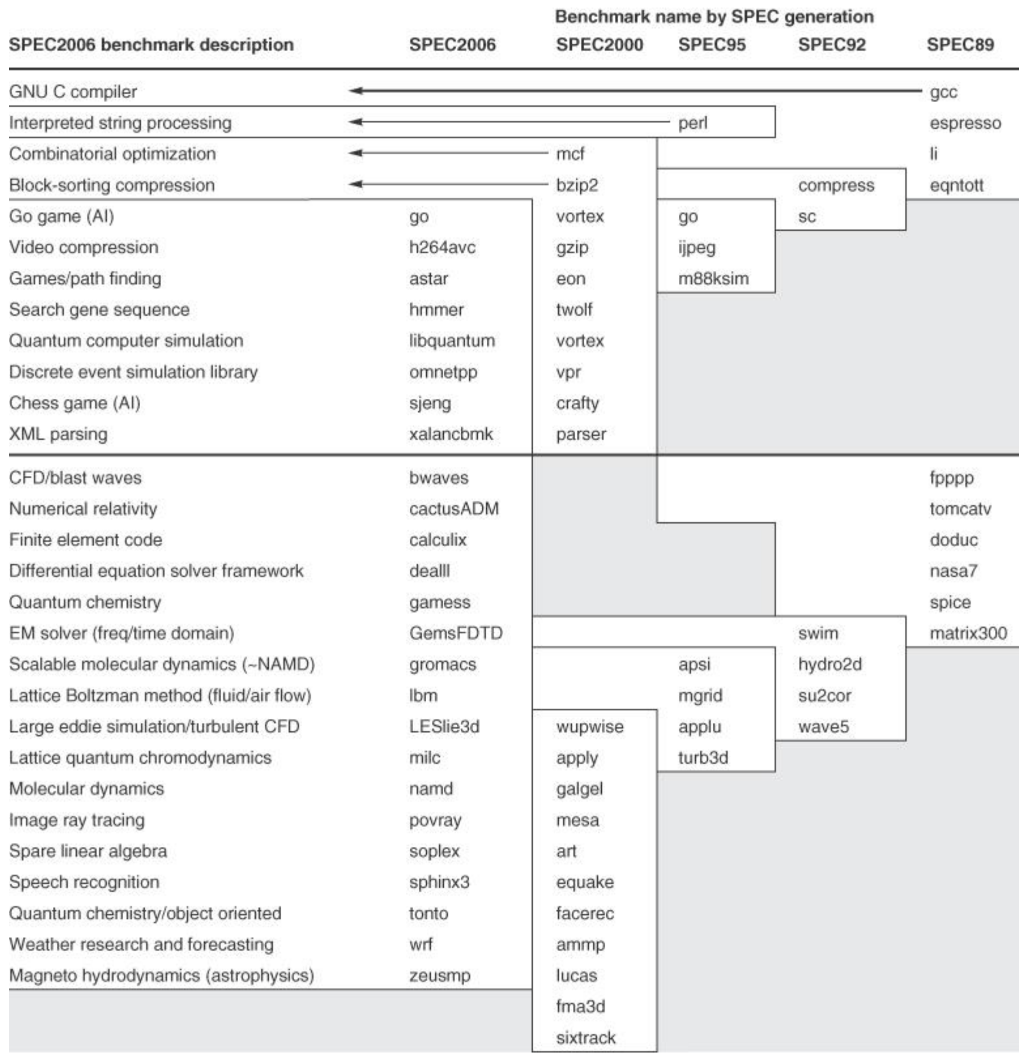
- Typical performance metrics:
  - Response time the time between the start and the completion of an event (=execution time)
  - Throughput the total amount of work done in a given time

- Speedup of X relative to Y
  - Execution time$_Y$ / Execution time$_X$

- Execution time
  - Wall clock time: includes all system overheads
    =response time = elapsed time
  - CPU time: only computation time exclude the time waiting for I/O or running other programs

- Benchmarks
  - Kernels (e.g. matrix multiply)
  - Toy programs (e.g. sorting)
  - Synthetic benchmarks (e.g. Dhrystone)
  - Benchmark suites (e.g. SPEC06fp, TPC-C)

SPEC2006 benchmark suite



	Benchmark name by SPEC generation				
SPEC2006 benchmark description	**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	dealll				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			

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
- Focus on the Common Case
  - Amdahl's Law

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

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

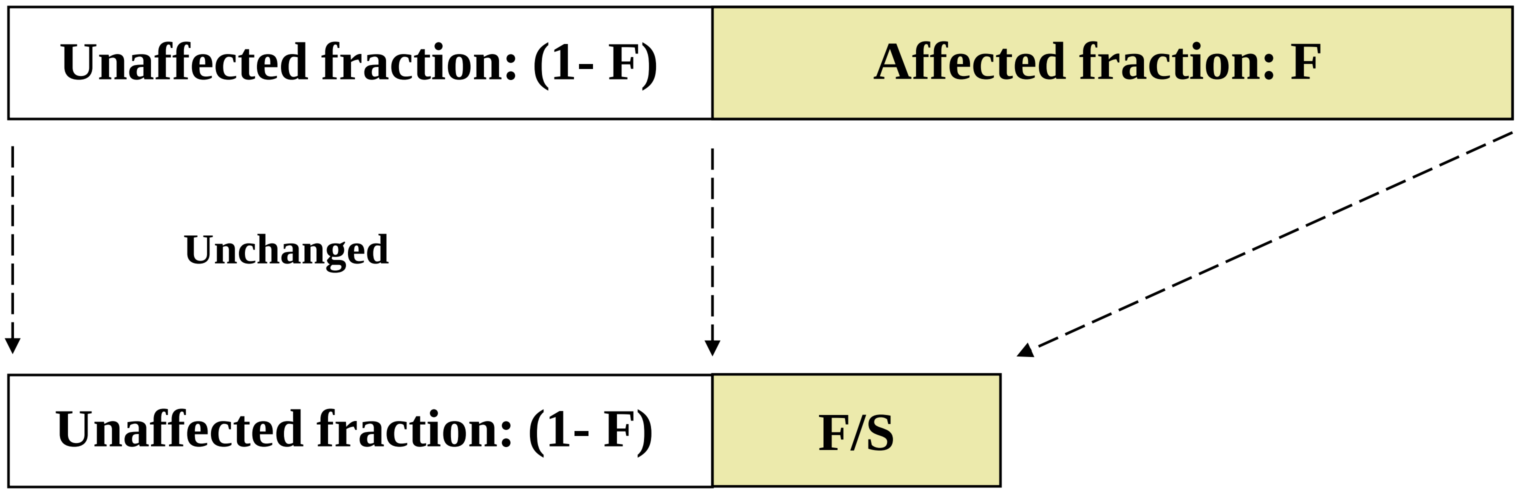
Pictorial Depiction of Amdahl's Law

Enhancement E accelerates fraction F of original execution time by a factor of S

Before:

Execution Time without enhancement E: (Before enhancement is applied)

- **shown normalized to 1 = (1-F) + F =1**



After:

Execution Time with enhancement E:

$$\text{Speedup(E)} = \frac{\text{Execution Time without enhancement E}}{\text{Execution Time with enhancement E}} = \frac{1}{(1 - F) + F/S}$$

Performance Enhancement Example

- A program runs in 100 seconds on a machine with multiply operations responsible for 80 seconds of this time. By how much must the speed of multiplication be improved to make the program four times faster?

$$\text{Desired speedup} = 4 = \frac{100}{\text{Execution Time with enhancement}}$$

$\rightarrow$ Execution time with enhancement = 25 seconds

25 seconds = (100 - 80 seconds) + 80 seconds / n

25 seconds = 20 seconds + 80 seconds / n

$\rightarrow$ 5 = 80 seconds / n

$\rightarrow$ n = 80/5 = 16

Hence multiplication should be 16 times faster to get speedup of 4.

Performance Enhancement Example II

- For the previous example with a program running in 100 seconds on a machine with multiply operations responsible for 80 seconds of this time. By how much must the speed of multiplication be improved to make the program five times faster?

$$\text{Desired speedup} = 5 = \frac{100}{\text{Execution Time with enhancement}}$$

$\rightarrow$ Execution time with enhancement = 20 seconds

20 seconds = (100 - 80 seconds) + 80 seconds / n
20 seconds = 20 seconds + 80 seconds / n

$\rightarrow$ 0 = 80 seconds / n

No amount of multiplication speed improvement can achieve this.

Principles of Computer Design

- The Processor Performance Equation

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

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

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

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

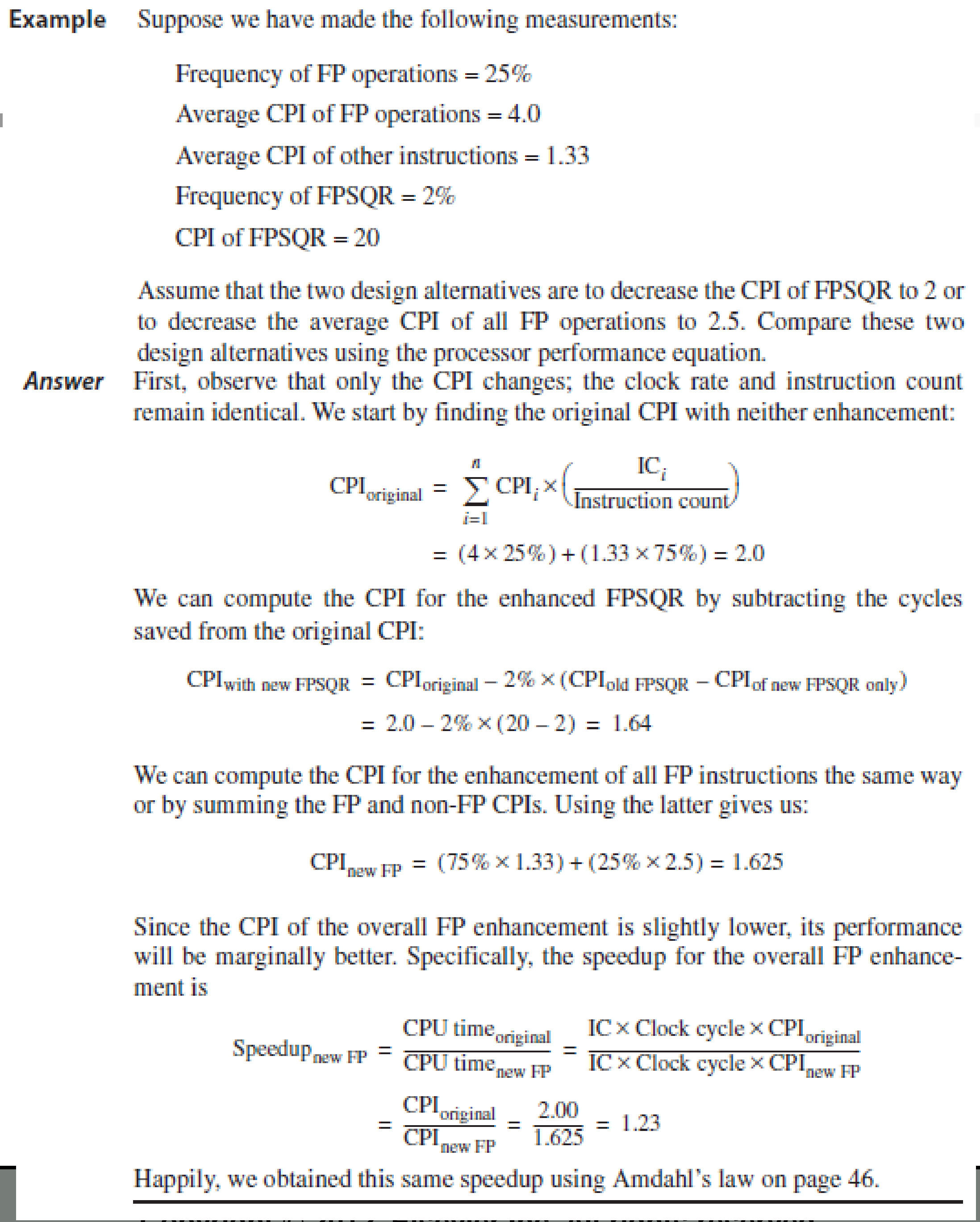
$$\frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Clock cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clock cycle}} = \frac{\text{Seconds}}{\text{Program}} = \text{CPU time}$$

Principles of Computer Design

- Different instruction types having different CPIs

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

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



Example Suppose we have made the following measurements:

Frequency of FP operations = 25%

Average CPI of FP operations = 4.0

Average CPI of other instructions = 1.33

Frequency of FPSQR = 2%

CPI of FPSQR = 20

Assume that the two design alternatives are to decrease the CPI of FPSQR to 2 or to decrease the average CPI of all FP operations to 2.5. Compare these two design alternatives using the processor performance equation.

Answer First, observe that only the CPI changes; the clock rate and instruction count remain identical. We start by finding the original CPI with neither enhancement:

$$\text{CPI}_{\text{original}} = \sum_{i=1}^{n} \text{CPI}_i \times \left(\frac{\text{IC}_i}{\text{Instruction count}}\right)$$

$$= (4 \times 25\%) + (1.33 \times 75\%) = 2.0$$

We can compute the CPI for the enhanced FPSQR by subtracting the cycles saved from the original CPI:

$$\text{CPI}_{\text{with new FPSQR}} = \text{CPI}_{\text{original}} - 2\% \times (\text{CPI}_{\text{old FPSQR}} - \text{CPI}_{\text{of new FPSQR only}})$$

$$= 2.0 - 2\% \times (20 - 2) = 1.64$$

We can compute the CPI for the enhancement of all FP instructions the same way or by summing the FP and non-FP CPIs. Using the latter gives us:

$$\text{CPI}_{\text{new FP}} = (75\% \times 1.33) + (25\% \times 2.5) = 1.625$$

Since the CPI of the overall FP enhancement is slightly lower, its performance will be marginally better. Specifically, the speedup for the overall FP enhancement is

$$\text{Speedup}_{\text{new FP}} = \frac{\text{CPU time}_{\text{original}}}{\text{CPU time}_{\text{new FP}}} = \frac{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{original}}}{\text{IC} \times \text{Clock cycle} \times \text{CPI}_{\text{new FP}}}$$

$$= \frac{\text{CPI}_{\text{original}}}{\text{CPI}_{\text{new FP}}} = \frac{2.00}{1.625} = 1.23$$

Happily, we obtained this same speedup using Amdahl's law on page 46.