Chapter 1: Fundamentals of Computer Design (Part 2)

What is computer architecture?

Why study computer architecture?

Common principles

- Performance
  - What is performance: latency, throughput
  - The performance equation
  - Measuring performance
  - Improving performance: parallelism, locality, Amdahl's law
- Power
- Cost
- Reliability

What is Performance?

Two Metrics

- Latency (or response time or execution time)

- Throughput (or bandwidth)

Performance (Cont.)

Assume an optimization results in machine X running faster than machine Y

Definition: X is n% faster than Y (or has n% higher performance than Y, assuming performance = 1/execution time) if

$$\frac{Execution\ Time_Y}{Execution\ Time_X} = 1 + \frac{n}{100}$$ [or (Y-X)/X = n/100]

Example: X = 1 minute, Y = 2 minutes: X is 100% faster than Y

Definition: X shows n% reduction (improvement) in execution time over Y if

$$\frac{Execution\ Time_X}{Execution\ Time_Y} = 1 - \frac{n}{100}$$ [or (Y-X)/Y = n/100]

Example: X = 1 minute, Y = 2 minutes: X shows 50% reduction in execution time over Y

Key Performance Equation

$$CPU_{time} = \frac{instructions}{program} \; X \; \frac{cycles}{instruction} \; X \; \frac{time}{cycle}$$

Instructions per program (path length)

- ISA and compiler

Cycles per instruction (CPI)

- ISA and organization (e.g., cache misses)

Time per cycle (clock time, cycle time)

- Organization and hardware

Measuring Performance

- MIPS, MFLOPS don't mean much
- Benchmarks
  - Real programs
    - Representative of real workload
    - Only way to characterize performance
    - SPEC89 → SPEC92 → SPEC95 → SPEC CPU2000 → CPU2006 → CPU2017 …
    - SPECFS, SPECWeb, SPECjbb, SPECvirt_Sc2010, TPC
  - Kernels
    - ``Representative'' program fragments
    - Often not representative of full applications
    - EEMBC for embedded systems
  - Toy benchmarks and synthetic benchmarks
    - Don't mean much

Improving Performance – Basic Principles

Parallelism

Locality

Focus on common case – Amdahl’s law

Amdahl's Law

(Or why the common case matters most)

Let

$$Speedup = \frac{new\ rate}{old\ rate} = \frac{old\ latency}{new\ latency}$$

Consider an enhancement x that speeds up fraction f_x of a task by S_x

$$Speedup_{overall} = \frac{old\ latency}{new\ latency}$$

$$= \frac{\{(1 - f_x) + (f_x)\} \times old\ latency}{(1 - f_x) \times old\ latency + f_x / S_x \times old\ latency}$$

Amdahl's law gives

$$Speedup_{overall} = \frac{1}{(1 - f_x) + f_x / S_x}$$

Amdahl's Law, cont.

Example: f_x = 95% and S_x = 1.10

$$Speedup_{overall} = \frac{1}{(1 - 0.95) + (0.95/1.10)} = 1.094$$

Example: f_x = 5% and S_x = 10

$$Speedup_{overall} = \frac{1}{(1 - 0.05) + (0.05/10)} = 1.047$$

Example: f_x = 5% and S_x = ∞

$$Speedup_{overall} = \frac{1}{(1 - 0.05) + (0.05/\infty)} = 1.052$$

Amdahl's Law Corollary

Since $S_x \rightarrow \infty$ implies

$$Speedup_{overall} = \frac{1}{(1 - f_x) + (f_x / \infty)}$$

For all real speedups:

$$Speedup_{overall} < \frac{1}{1 - f_x}$$

Example

f_x	1/(1-f_x)
1%	1.01
2%	1.02
5%	1.05
10%	1.11
20%	1.25
50%	2.00

Or *make the common case fast*

An application?

Power

Power

Energy

Temperature

Power and Energy

Power = Dynamic power + Static power

Energy = Power * Time

Dynamic Power $\propto$ Capacitance * Voltage^2 * Frequency

Static power = Static current * Voltage

Growth in Clock Rate

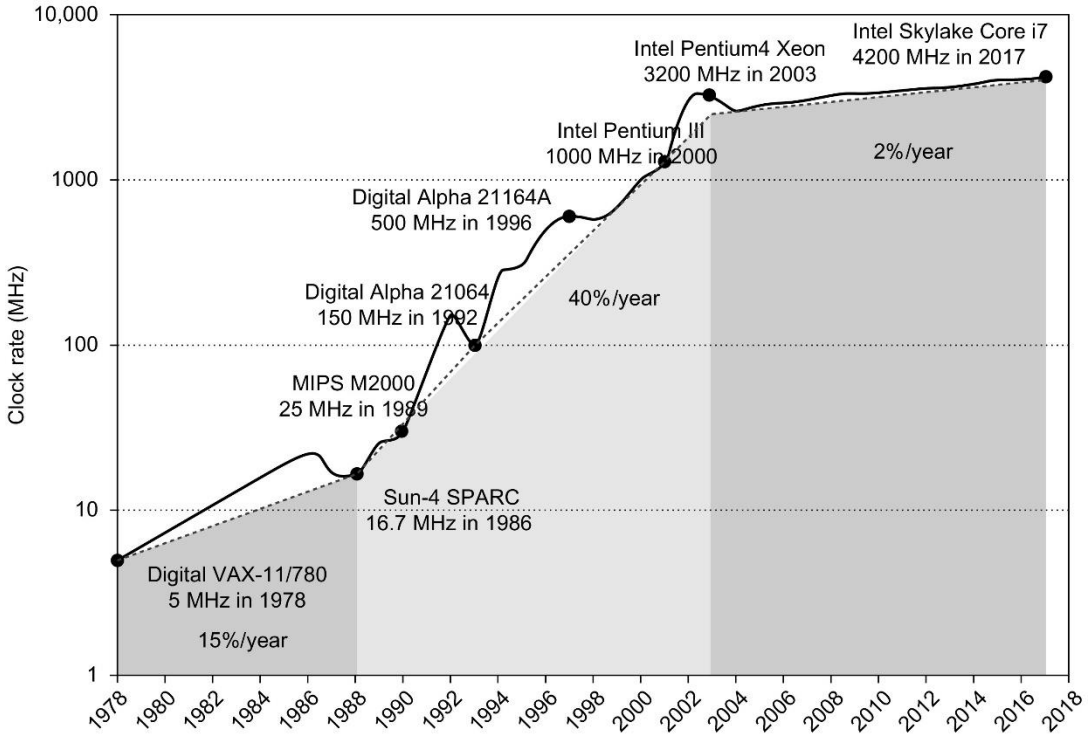

Figure 1.11 Growth in clock rate of microprocessors in Figure 1.1. Between 1978 and 1986, the clock rate improved less than 15% per year while performance improved by 22% per year. During the “renaissance period” of 52% performance improvement per year between 1986 and 2003, clock rates shot up almost 40% per year. Since then, the clock rate has been nearly flat, growing at less than 2% per year, while single processor performance improved recently at just 3.5% per year.

Cost

Cost is very important in most real designs

- But usually hard to quantify for the architect

Costs change over time

- Learning curve lowers manufacturing costs
- Technology improvements lower costs

Focus on IC costs next

A Wafer

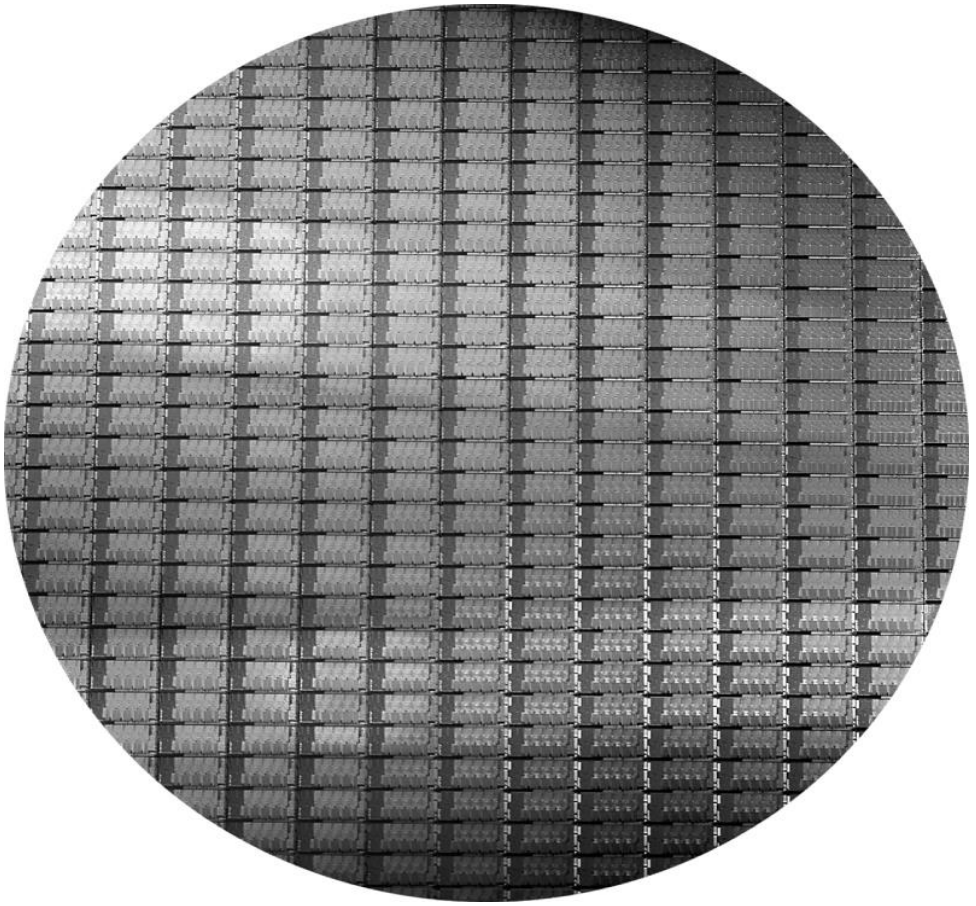

Figure 1.15 This 300 mm wafer contains 280 full Sandy Bridge dies, each 20.7 by 10.5 mm in a 32 nm process. (Sandy Bridge is Intel's successor to Nehalem used in the Core i7.) At 216 mm2, the formula for dies per wafer estimates 282. (Courtesy Intel.)

A Die

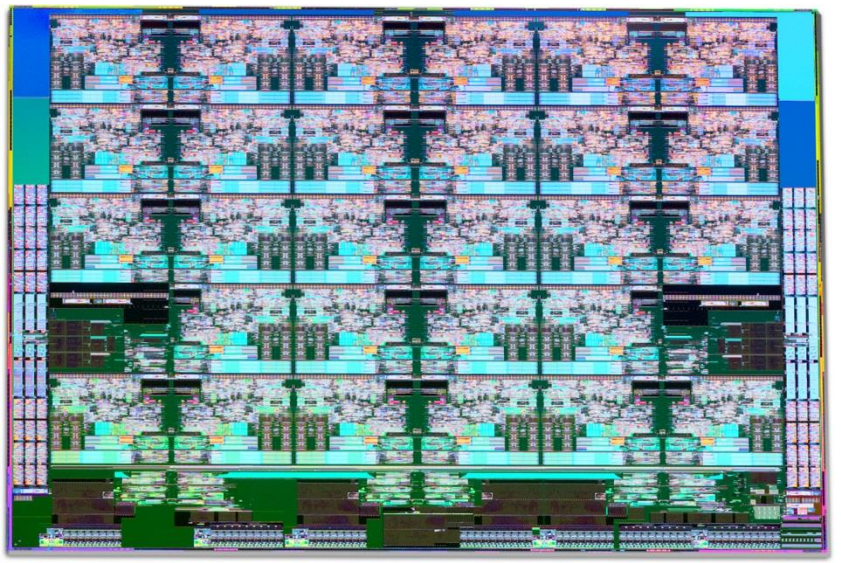

Figure 1.14 Photograph of an Intel Skylake microprocessor die, which is evaluated in Chapter 4.

A Die with Parts Labeled

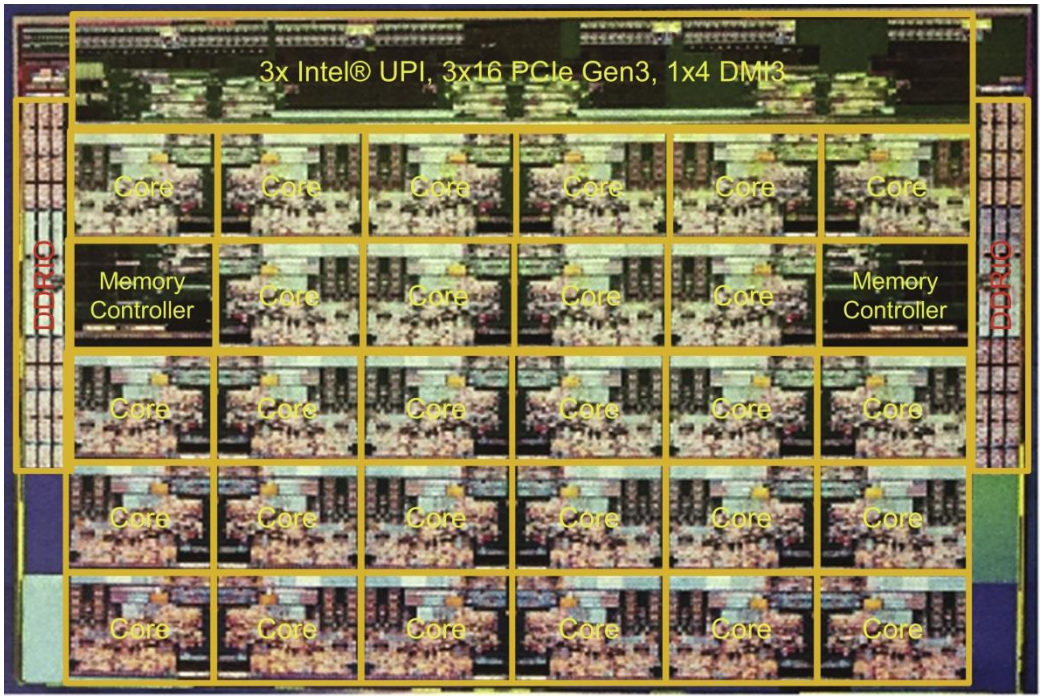

Figure 1.15 The components of the microprocessor die in Figure 1.14 are labeled with their functions.

Integrated Circuit Cost

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

$$Cost\ of\ Die = \frac{Cost\ of\ Wafer}{Dies\ per\ Wafer \times Die\ Yield}$$

$$Dies\ per\ Wafer = \left(\frac{\pi \times (Wafer\ Diameter/2)^2}{Die\ Area}\right) - (Correction\ factor\ for\ Edge\ Effects)$$

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

α= 10 to 14 for 16nm in 2017

Bottom line: Cost per die grows roughly as the square of the die area

Cost different from price; cost of manufacturing different from cost of operation

Reliability

Many sources of unreliability

- Soft errors due to radiation, hard errors due to wearout, …

Common metrics

- Mean time to failure – MTTF
- For exponentially distributed time to failure
  - Define failures in time or FITs
    - FIT = failures in a billion hours
    - FIT α 1/MTTF
    - FIT of system = Sum of FITs of components

Common solution