Computer Architecture

A Quantitative Approach, Fifth Edition

Chapter 1

Fundamentals of Quantitative Design and Analysis Part II

These slides are based on the slides provided by the publisher.

The slides will be modified, annotated, explained on the board, and sometimes corrected in the class

Trends in Cost

- Cost driven down by learning curve
  - Yield
- DRAM: price closely tracks cost
- Microprocessors: price depends on volume
  - 10% less for each doubling of volume

Integrated Circuit Cost

Trends in Cost

- Integrated circuit

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

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

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

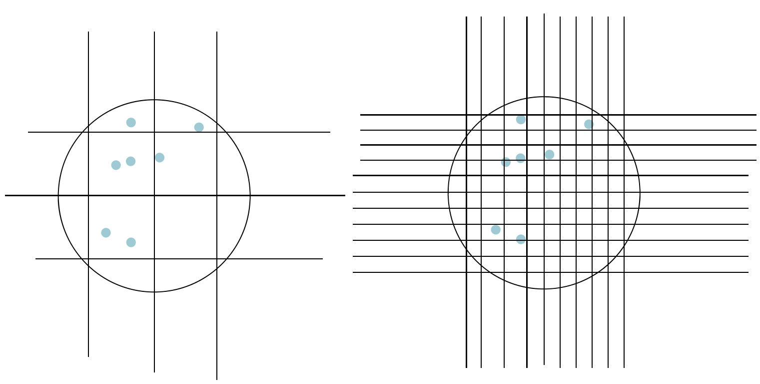
- Bose-Einstein formula:

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

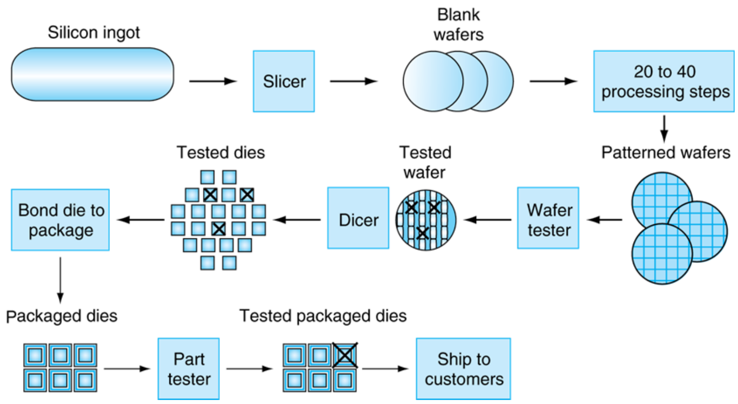
- Defects per unit area = 0.016-0.057 defects per square cm (2010)
- N = process-complexity factor = 11.5-15.5 (40 nm, 2010)

Integrated Circuit Cost

Wafer Yield



Manufacturing IC's



 40

Dependability

Dependability

- Service Level Agreement (SLA) guarantees a certain level of dependability.
- Module reliability
  - Mean time to failure (MTTF)
  - Mean time to repair (MTTR)
  - Mean time between failures (MTBF) = MTTF + MTTR
  - Availability = MTTF / (MTTF+MTTR)
- Cost of failure: varies hugely depending on applications

 41

- Example
  - 10 disks 1,000,000-hour MTTF
  - 1 ATA controller 500,000-hour MTTF
  - 1 Power supply 200,000-hour MTTF
  - 1 Fan 200,000-hour MTTF
  - 1 ATA cable 1,000,000-hour MTTF
- Assume lifetimes are exponentially distributed and failures are independent
- Calculate MTTF

- What if we added one extra power supply

Measuring Performance

- You drive at 100Km/h for 1 km
- Then drive at 200Km/h for 1 km
- What is your average speed

 44

How to measure performance

- Instructions/second
- Clock rate
- How long to complete a program
- How many jobs/second you can complete
- Which one is a better indication of performance?

 45

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

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

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

Speedup

X is n time faster than Y

$$\frac{\text{Execution Time}_Y}{\text{Execution Time}_X} = n$$

Throughtput of X is n time thet of Y

$$\frac{\text{tasks per unit time}_X}{\text{tasks per unit time}_Y} = n$$

Example

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)

benchmarks

- Embedded Microprocessor Benchmark Consortium
  - www.eembc.org
  - 41 kernels
- SPEC: Standard Performance Evaluation Corporation
  - www.spec.org
  - Covers many application classes (desktop, SPEC Web, SPECFS)
- TPC: Transaction Processing Council
  - www.tpc.org
  - Database transactions

Reporting Performance

- Many programs, how can we capture performance using a single number?

	P1	P2	P3
Machine-A	10	8	25
Machine-B	12	9	20
Machine-C	8	8	30

- Sum of execution time
- Sum of weighted execution time
- Geometric mean of execution time

Reporting perofrmance

- Arithmetic
  - Usually used with time (or anything proportional to time).
- Harmonic
  - Usually with inversely proportional to time
  - Throughput N/SUM(1/throughput_i)
- Geometric
  - Usually for unit-less quantities (speedup, or ratios).

Reporting Performance

- Many programs, how can we capture performance using a single number?

	P1	P2	P3
Machine-A	10	8	25
Machine-B	12	9	20
Machine-C	8	8	30

- Sum of execution time
- Sum of weighted execution time
- Geometric mean of execution time

Reporting Performance

- Many programs, how can we capture performance using a single number?

	P1	P2	P3
Machine-A	10	8	25
Machine-B	12	9	20
Machine-C	8	8	30

- Sum of execution time
- Sum of weighted execution time
- Geometric mean of execution time

Reporting Performance

	machine_A	M/C_B	M/C_C
P1	1sec	10sec	20sec
P2	1000sec	100sec	20sec

Reporting Performance

- Time = TC × CPI × IC
- Must be reproducible
- Complete description of the computer and compiler flags.
- Usually, compared to a standard machine execution time SPECRatioA = T_{ref}/T_A.
- Geometric mean

CINT2006 for Opteron X4 2356

Name	Description	IC×10⁹	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,118	0.75	0.40	637	9,777	15.3
bzip2	Block-sorting compression	2,389	0.85	0.40	817	9,650	11.8
gcc	GNU C Compiler	1,050	1.72	0.47	24	8,050	11.1
mcf	Combinatorial optimization	336	10.00	0.40	1,345	9,120	6.8
go	Go game (AI)	1,658	1.09	0.40	721	10,490	14.6
hmmer	Search gene sequence	2,783	0.80	0.40	890	9,330	10.5
sjeng	Chess game (AI)	2,176	0.96	0.48	37	12,100	14.5
libquantum	Quantum computer simulation	1,623	1.61	0.40	1,047	20,720	19.8
h264avc	Video compression	3,102	0.80	0.40	993	22,130	22.3
omnetpp	Discrete event simulation	587	2.94	0.40	690	6,250	9.1
astar	Games/path finding	1,082	1.79	0.40	773	7,020	9.1
xalancbmk	XML parsing	1,058	2.70	0.40	1,143	6,900	6.0
Geometric mean							11.7

High cache miss rates

CINT2006 for 2.66 GHz i7 920

Name	Description	IC×10⁹	CPI	Tc (ns)	Exec time	Ref time	SPECratio
perl	Interpreted string processing	2,252	0.60	0.376	508	9,770	19.2
bzip2	Block-sorting compression	2,390	0.70	0.376	629	9,650	15.4
gcc	GNU C Compiler	794	1.20	0.376	358	8,050	22.5
mcf	Combinatorial optimization	221	2.66	0.376	221	9,120	41.2
go	Go game (AI)	1,274	1.10	0.376	527	10,490	19.9
Hmmer	Search gene sequence	2,616	0.60	0.376	590	9,330	15.8
sjeng	Chess game (AI)	1,948	0.80	0.376	586	12,100	20.7
libquantum	Quantum computer simulation	659	0.44	0.376	109	20,720	190.0
h264avc	Video compression	3,793	0.50	0.376	713	22,130	31.0
omnetpp	Discrete event simulation	367	2.10	0.376	290	6,250	21.5
astar	Games/path finding	1,250	1.00	0.376	470	7,020	14.9
xalancbmk	XML parsing	1,045	0.70	0.376	275	6,900	25.1
Geometric mean							25.7

SPEC Power Benchmark

- Power consumption of server at different workload levels
  - Performance: ssj_ops/sec
  - Power: Watts (Joules/sec)

$$\text{Overall ssj\_ops per Watt} = \left(\sum_{i=0}^{10} \text{ssj\_ops}_i\right) \Big/ \left(\sum_{i=0}^{10} \text{power}_i\right)$$

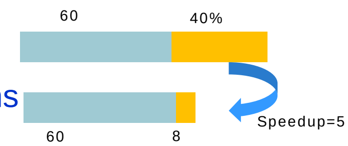
SPECpower_ssj2008 for X4

Target Load %	Performance (ssj_ops/sec)	Average Power (Watts)
100%	231,867	295
90%	211,282	286
80%	185,803	275
70%	163,427	265
60%	140,160	256
50%	118,324	246
40%	920,35	233
30%	70,500	222
20%	47,126	206
10%	23,066	180
0%	0	141
Overall sum	1,283,590	2,605
∑ssj_ops/ ∑power		493

Principles of Computer Design

Principles

- 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

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

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

Fallacies and Pitfalls

Fallacies

- Multiprocessors are a silver bullet
- H/W enhancements improve energy consumption or at least energy neutral
- Misreading MTTF
- Peak performance tracks observed performance

Pitfalls

- Falling prey to Amdahl's law
- A single point of failure
- Fault detection can lower availability