Lecture 2: System Metrics and Pipelining

- Today's topics: (Sections 1.5 – 1.10)
  - Power/Energy examples
  - Performance summaries
  - Measuring cost and dependability
- Class mailing list sign up
- Class notes
- Assignment 1 will be posted over weekend; due in 12 days

Reducing Power and Energy

- Can gate off transistors that are inactive (reduces leakage)
- Design for typical case and throttle down when activity exceeds a threshold
- DFS: Dynamic frequency scaling -- only reduces frequency and dynamic power, but hurts energy
- DVFS: Dynamic voltage and frequency scaling – can reduce voltage and frequency by (say) 10%; can slow a program by (say) 8%, but reduce dynamic power by 27%, reduce total power by (say) 23%, reduce total energy by 17%

  (Note: voltage drop → slow transistor → freq drop)

DVFS Example

Other Technology Trends

- DRAM density increases by 40-60% per year, latency has reduced by 33% in 10 years (the memory wall!), bandwidth improves twice as fast as latency decreases
- Disk density improves by 100% every year, latency improvement similar to DRAM
- Emergence of NVRAM technologies that can provide a bridge between DRAM and hard disk drives

Measuring Performance

- Two primary metrics: wall clock time (response time for a program) and throughput (jobs performed in unit time)

- To optimize throughput, must ensure that there is minimal waste of resources

- Performance is measured with benchmark suites: a collection of programs that are likely relevant to the user
  - SPEC CPU 2006: cpu-oriented programs (for desktops)
  - SPECweb, TPC: throughput-oriented (for servers)
  - EEMBC: for embedded processors/workloads

Summarizing Performance

- Consider 25 programs from a benchmark set – how do we capture the behavior of all 25 programs with a single number?

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

  - Total (average) execution time
  - Total (average) weighted execution time
    or Average of normalized execution times
  - Geometric mean of normalized execution times

AM Example

AM Example

- We fixed a reference machine X and ran 4 programs A, B, C, D on it such that each program ran for 1 second
- The exact same workload (the four programs execute the same number of instructions that they did on machine X) is run on a new machine Y and the execution times for each program are 0.8, 1.1, 0.5, 2
- With AM of normalized execution times, we can conclude that Y is 1.1 times slower than X – perhaps, not for all workloads, but definitely for one specific workload (where all programs run on the ref-machine for an equal #cycles)
- With GM, you may find inconsistencies

GM Example

	Computer-A	Computer-B	Computer-C
P1	1 sec	10 secs	20 secs
P2	1000 secs	100 secs	20 secs

Conclusion with GMs: (i) A=B
(ii) C is ~1.6 times faster

- For (i) to be true, P1 must occur 100 times for every occurrence of P2

- With the above assumption, (ii) is no longer true

Hence, GM can lead to inconsistencies

Summarizing Performance

- GM: does not require a reference machine, but does not predict performance very well
  - So we multiplied execution times and determined that sys-A is 1.2x faster…but on what workload?
- AM: does predict performance for a specific workload, but that workload was determined by executing programs on a reference machine
  - Every year or so, the reference machine will have to be updated

Normalized Execution Times

- Advantage of GM: no reference machine required
- Disadvantage of GM: does not represent any “real entity” and may not accurately predict performance
- Disadvantage of AM of normalized: need weights (which may change over time)
- Advantage: can represent a real workload

CPU Performance Equation

- Clock cycle time = 1 / clock speed
- CPU time = clock cycle time x cycles per instruction x number of instructions
- Influencing factors for each:
  - clock cycle time: technology and pipeline
  - CPI: architecture and instruction set design
  - instruction count: instruction set design and compiler
- CPI (cycles per instruction) or IPC (instructions per cycle) can not be accurately estimated analytically

Measuring System CPI

- Assume that an architectural innovation only affects CPI
- For 3 programs, base CPIs: 1.2, 1.8, 2.5
  CPIs for proposed model: 1.4, 1.9, 2.3
- What is the best way to summarize performance with a single number? AM, HM, or GM of CPIs?

Example

- AM of CPI for base case = $\frac{1.2 \text{ cyc}}{\text{instr}} + \frac{1.8 \text{ cyc}}{\text{instr}} + \frac{2.5 \text{ cyc}}{\text{instr}}$ /3

  5.5 cycles is execution time if each program ran for one instruction – therefore, AM of CPI defines a workload where every program runs for an equal #instrs

- HM of CPI = 1 / AM of IPC ; defines a workload where every program runs for an equal number of cycles

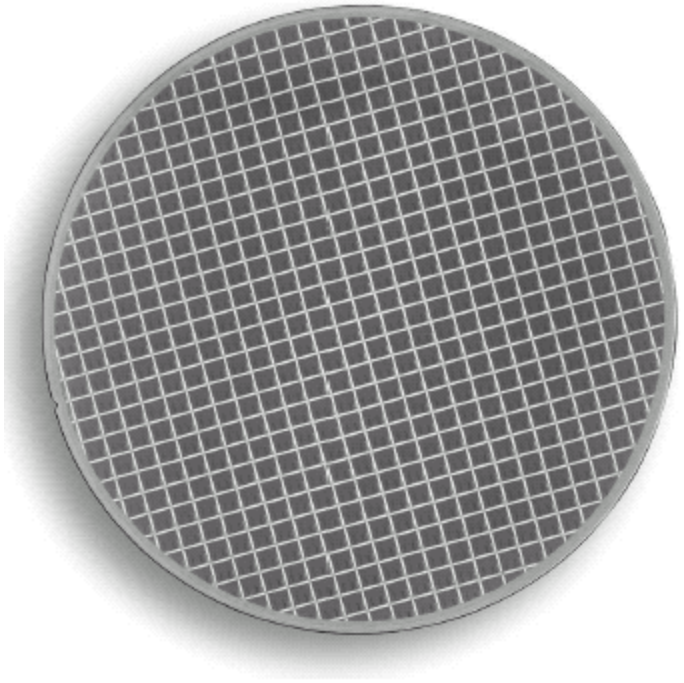
- GM of CPI: warm fuzzy number, not necessarily representing any workload

Speedup Vs. Percentage

- “Speedup” is a ratio
- “Improvement”, “Increase”, “Decrease” usually refer to percentage relative to the baseline
- A program ran in 100 seconds on my old laptop and in 70 seconds on my new laptop
  - What is the speedup?
  - What is the percentage increase in performance?
  - What is the reduction in execution time?

Wafers and Dies

An entire wafer is produced and chopped into dies that undergo testing and packaging



Integrated Circuit Cost

- Cost of an integrated circuit =
  (cost of die + cost of packaging and testing) / final test yield
- Cost of die = cost of wafer / (dies per wafer x die yield)
- Dies/wafer = wafer area / die area - π wafer diam / die diag
- Die yield = wafer yield x (1 + (defect rate x die area) / α) $^{-\alpha}$
- Thus, die yield depends on die area and complexity arising from multiple manufacturing steps (α ~ 4.0)

Integrated Circuit Cost Examples

- Bottomline: cost decreases dramatically if the chip area is smaller, if the chip has fewer manufacturing steps (less complex), if the chip is produced in high volume (10% lower cost if volume doubles)

- A 30 cm diameter wafer cost $5-6K in 2001

- Such a wafer yields about 366 good 1 cm^2 dies and 1014 good 0.49 cm^2 dies (note the effect of area and yield)

- Die sizes: Alpha 21264 1.15 cm^2 , Itanium 3.0 cm^2 , embedded processors are between 0.1 – 0.25 cm^2

Contribution of IC Costs to Total System Cost

Subsystem	Fraction of total cost
Cabinet: sheet metal, plastic, power supply, fans, cables, nuts, bolts, manuals, shipping box	6%
Processor	22%
DRAM (128 MB)	5%
Video card	5%
Motherboard	5%
Processor board subtotal	37%
Keyboard and mouse	3%
Monitor	19%
Hard disk (20 GB)	9%
DVD drive	6%
I/O devices subtotal	37%
Software (OS + Office)	20%

Defining Fault, Error, and Failure

- A *fault* produces a *latent error*; it becomes *effective* when activated; it leads to *failure* when the observed actual behavior deviates from the ideal specified behavior

- Example I : a programming mistake is a fault; the buggy code is the latent error; when the code runs, it is effective; if the buggy code influences program output/behavior, a failure occurs

- Example II : an alpha particle strikes DRAM (fault); if it changes the memory bit, it produces a latent error; when the value is read, the error becomes effective; if program output deviates, failure occurs

Defining Reliability and Availability

- A system toggles between
  - Service accomplishment: service matches specifications
  - Service interruption: services deviates from specs
- The toggle is caused by *failures* and *restorations*
- Reliability measures continuous service accomplishment and is usually expressed as mean time to failure (MTTF)
- Availability measures fraction of time that service matches specifications, expressed as MTTF / (MTTF + MTTR)

Title

- Bullet