CMSC 132: Computer Architecture

Asst. Prof. Reginald Neil C. Recario
rncrecario@gmail.com
Institute of Computer Science
University of the Philippines Los Baños

Cost of an Integrated Circuit

- Redundancy is included by designers because of tremendous price pressures on DRAM and SRAM.
  - Raise yield
- DRAMs have included redundant memory cells to accommodate flaws.
- SRAMS use similar approaches for cache arrays within microprocessors.

Cost of an Integrated Circuit

- What should a computer designer remember about chip costs?
- Sole control of the designer is the die area.
- Manufacturing process dictates the
  - Wafer cost
  - Wafer yield
  - Defects per unit area

Cost of an Integrated Circuit

- In practice, because the number of defects per unit area is small, the number of good dies per wafer, and hence the cost per die, grows roughly as the square of the die area.

Cost of an Integrated Circuit

- The computer designer affects die size
  - Thus affecting cost!
- How?
  - Functions included or excluded
  - Number of I/O pins

Cost of an Integrated Circuit

- Dies are tested, packaged and tested again before ending up in our computers.
  - These steps obviously add up to the cost.
- General factors for high volume production.

Cost of an Integrated Circuit

- Mask set is an important factor on fixed cost low volume productions.
  - Significantly affect cost.
- Each step in the IC process requires a mask.
  - Expensive.

Cost vs Price

- Commoditization of computers led to the shrinking of price-cost margin
  - Cost to the manufacture of a product
  - Price the product sells
  - Margins are used for R&D, marketing, sales, manufacturing equipment maintenance, building rental, cost of financing, pre-tax profits and taxes.
- Most companies spend only 4-12% of profit for R&D.

Dependability

- ICs were one of the most reliable components of the computer.
  - Low error rate in the chip
- This conventional wisdom changes in smaller ICs
  - Transient and permanent faults are common place

Dependability

- Computers are constructed in different levels of abstraction.
- Some faults are widespread but many can be limited within a module
  - Regarded as component error

Dependability

- When is a system operating properly?
  - Became concrete with the popularity of Internet services.
- Infrastructure providers started offering Service Level Agreements (SLA) or Service Level Objectives (SLO) to guarantee that their networking or power service would be dependable.

Dependability

- Systems alternate between two states of service with respect to an SLA:
  - Service accomplishment, where the service is delivered as specified
  - Service interruption, where the delivered service is different from the SLA

Dependability

- Transitions between these two states are caused by
  - failures (from state 1 to state 2)
  - restorations (2 to 1).

Dependability

- Two main measures:
  - Module reliability
  - Module availability

Module Reliability

- Module reliability is a measure of the continuous service accomplishment (or, equivalently, of the time to failure) from a reference initial instant.
- Mean time to failure (MTTF) is a reliability measure.

Module Reliability

- The reciprocal of MTTF is a rate of failures, generally reported as failures per billion hours of operation, or FIT (for failures in time).

- Example
- An MTTF of 1,000,000 hours would be equal to 1,000 FIT.

Module Reliability

- Service interruption is measured as mean time to repair (MTTR).
- Mean time between failures (MTBF) is simply the sum of MTTF + MTTR.

Module Reliability

- If a collection of modules have exponentially distributed lifetimes—meaning that the age of a module is not important in probability of failure—the overall failure rate of the collection is the sum of the failure rates of the modules.

Module Availability

- Module availability is a measure of the service accomplishment with respect to the alternation between the two states of accomplishment and interruption.

Module Availability

- For nonredundant systems with repair, module availability is

$$\text{Module Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

Dependability

- Example
- Assume a disk subsystem with the following components and MTTF:
  - 10 disks, each rated at 1,000,000-hour MTTF
  - 1 SCSI controller, 500,000-hour MTTF
  - 1 power supply, 200,000-hour MTTF
  - 1 fan, 200,000-hour MTTF
  - 1 SCSI cable, 1,000,000-hour MTTF

Dependability

- Using the simplifying assumptions that the lifetimes are exponentially distributed and that failures are independent, compute the MTTF of the system as a whole.

Dependability

- Solution:

$$\text{Failure}_{\text{system}} = 10 * \frac{1}{1{,}000{,}000} + \frac{1}{500{,}000} + \frac{1}{200{,}000} + \frac{1}{200{,}000} + \frac{1}{1{,}000{,}000}$$

Dependability

- Simplify:

$$\text{Failure}_{\text{system}} = \frac{10 + 2 + 5 + 5 + 1}{1{,}000{,}000}$$

$$\text{Failure}_{\text{system}} = \frac{23}{1{,}000{,}000}$$

Dependability

- Take note, however, that FIT is a reciprocal in terms of billion hours

$$\text{Failure}_{\text{system}} = \frac{23{,}000}{1{,}000{,}000{,}000}$$

This can be interpreted as 23,000 FIT.

Dependability

- To get the MTTF of the whole system, we have:

$$\text{MTTF}_{\text{system}} = \frac{1}{\text{Failure}_{\text{system}}}$$

$$= \frac{1{,}000{,}000{,}000}{23{,}000}$$

- 43,478.26 hours or around 4.96 years.

Dependability

- The primary way to cope with failure is redundancy,
- Redundancy in either in time (repeat the operation to see if it still is erroneous) or in resources (have other components to take over from the one that failed).

Dependability

- Once the component is replaced and the system fully repaired, the dependability of the system is assumed to be as good as new.

Dependability

- Example:
- Disk subsystems often have redundant power supplies to improve dependability. Using the components and MTTFs from above, calculate the reliability of a redundant power supply. Assume one power supply is sufficient to run the disk subsystem and that we are adding one redundant power supply.

Dependability

- Solution:
  - We need a formula to show what to expect when we can tolerate a failure and still provide service.
  - We assume that the lifetimes of the components are exponentially distributed and that there is no dependency between the component failures.

Dependability

- Solution:
  - MTTF for our redundant power supplies is the mean time until one power supply fails divided by the chance that the other will fail before the first one is replaced.
  - Thus, if the chance of a second failure before repair is small, then MTTF of the pair is large.

Dependability

- Solution:
  - Since we have two power supplies and independent failures, the mean time until one disk fails is $MTTF_{\text{power supply}} / 2$.
  - A good approximation of the probability of a second failure is MTTR over the mean time until the other power supply fails.

Dependability

Set up the equation:

$$MTTF_{power\ supply\ pair} = \frac{MTTF_{power\ supply}/2}{\frac{MTTR_{power\ supply}}{MTTF_{power\ supply}}}$$

$$MTTF_{power\ supply\ pair} = \frac{MTTF^2_{power\ supply}}{2 * MTTR_{power\ supply}}$$

Dependability

- If we assume it takes on average 24 hours for a human operator to notice that a power supply has failed and replace it, the reliability of the fault tolerant pair of power supplies is

Dependability

- MTTF is equal to

  $=200{,}000^2/(2*24) = 830{,}000{,}000$

- The pair is about 4,150 times more reliable than a single power supply.

Measuring, Reporting, and Summarizing Performance

- When we say one computer is faster than another is, what do we mean?
- Do we mean *faster* because…
  - Computer A runs a program in less time than Computer B
  - Computer C completes more transaction in a given period compared to Computer D

Measuring, Reporting, and Summarizing Performance

- The computer user is interested in reducing response time
- Response time is the time between the start and the completion of an event also referred to as execution time.

Measuring, Reporting, and Summarizing Performance

- One might be interested in increasing the throughput
- Throughput is the total amount of work done in a given time.

Measuring, Reporting, and Summarizing Performance

- In comparing design alternatives, we often want to relate the performance of two different computers, say, X and Y.
- The phrase "X is faster than Y" is used here to mean that the response time or execution time is lower on X than on Y for the given task.

Measuring, Reporting, and Summarizing Performance

- In particular, "X is n times faster than Y" will mean

$$n = \frac{\text{Execution time}_y}{\text{Execution time}_x}$$

Measuring, Reporting, and Summarizing Performance

=

- Since execution time is the reciprocal of performance, the following relationship holds:

$$n = \frac{\text{Execution time}_y}{\text{Execution time}_x}$$

$$\frac{\frac{1}{\text{Performance}_y}}{\frac{1}{\text{Performance}_x}} = \frac{\text{Performance}_x}{\text{Performance}_y}$$

Measuring, Reporting, and Summarizing Performance

- Example:
- The phrase "the throughput of A is 1.6 times higher than B" signifies here that the number of tasks completed per unit time on computer A is 1.6 times the number completed on B

Measuring, Reporting, and Summarizing Performance

- Time is not always the metric quoted in comparing the performance of computers.
  - Even execution time can be defined in different ways depending on what we count.

Measuring, Reporting, and Summarizing Performance

- The most straightforward definition of time is called wall-clock time, response time, or elapsed time
  - The latency to complete a task, including disk accesses, memory accesses, I/O activities and operating system overhead.

Measuring, Reporting, and Summarizing Performance

- In a multiprogramming environment
  - The processor works on another program while waiting for I/O .
  - May not necessarily minimize the elapsed time of one program.
- Time considered must not include waiting time!

Measuring, Reporting, and Summarizing Performance

- Benchmarks
- What is a benchmark?

Measuring, Reporting, and Summarizing Performance

- Benchmark
  - Relative performance assessment
  - Set of programs and tests/trials

Measuring, Reporting, and Summarizing Performance

- The best choice of benchmarks to measure performance are real applications, such as a compiler.

Measuring, Reporting, and Summarizing Performance

- Attempts at running programs that are much simpler than a real application have led to performance pitfalls. Examples include
  - kernels, which are small, key pieces of real applications;
  - toy programs, which are 100-line programs from beginning programming assignments, such as quicksort; and
  - synthetic benchmarks, which are fake programs invented to try to match the profile and behavior of real applications.

Measuring, Reporting, and Summarizing Performance

- Examples of synthetic benchmarks are dhrystone and whetstone.

Measuring, Reporting, and Summarizing Performance

- Issues
  - Conspiracy between compiler writer and architect to make computer appear faster.
  - Conditions under which the benchmark is run.
    - Improve using specific flags
    - Countermeasure: require a standard set of flags under a common language (C or FORTRAN)
  - Source code modifications (allowed or not?)

Measuring, Reporting, and Summarizing Performance

- Approaches in addressing code modification
  - No source code modifications are allowed.
  - Source code modifications are allowed, but are essentially impossible.
  - Source modifications are allowed, as long as the modified version produces the same output.

Measuring, Reporting, and Summarizing Performance

- The key issue that benchmark designers face in deciding to allow modification of the source is whether such modifications will reflect real practice and provide useful insight to users, or whether such modifications simply reduce the accuracy of the benchmarks as predictors of real performance.

Measuring, Reporting, and Summarizing Performance

- Solution:
- Collections of benchmark applications called benchmark suites are used
  - Popular measure of performance of processors with a variety of applications.

Measuring, Reporting, and Summarizing Performance

- Key advantage of such suites is that the weakness of any one benchmark is lessened by the presence of the other benchmarks.

- There are a variety of benchmarks which include:
  - Integer benchmarks
  - Floating-Point benchmarks

Measuring, Reporting, and Summarizing Performance

- The goal of a benchmark suite is that it will characterize the relative performance of two computers, particularly for programs not in the suite that customers are likely to run.

Measuring, Reporting, and Summarizing Performance

- Two broad groups of benchmarks
  - Desktop Benchmarks
  - Server Benchmarks

Measuring, Reporting, and Summarizing Performance

- Desktop Benchmarks
- Desktop Benchmarks divide into two broad classes:
  - processor-intensive benchmarks
  - graphics-intensive benchmarks
    - Although many graphics benchmarks include intensive processor activity

Measuring, Reporting, and Summarizing Performance

- Server Benchmarks
- Servers, having multiple functions, have multiple types of benchmarks.
  - Processor Throughput-Oriented benchmark is the simplest server benchmark
  - Transaction Processing benchmark

Measuring, Reporting, and Summarizing Performance

- Transaction-processing (TP) benchmarks measure the ability of a system to handle transactions, which consist of database accesses and updates.
- Used by
  - Airline reservation systems
  - bank ATM systems
- more sophisticated TP systems involve complex databases and decision-making.

Measuring, Reporting, and Summarizing Performance

- The TPC benchmarks are described at www.tpc.org.
- The first TPC benchmark, TPC-A, was published in 1985 and has since been replaced and enhanced by several different benchmarks.

Measuring, Reporting, and Summarizing Performance

- TPC-W is a transactional Web benchmark.
  - Workload is performed in a controlled Internet commerce environment that simulates the activities of a business-oriented transactional Web server.

Measuring, Reporting, and Summarizing Performance

- The most recent is TPC-App, an application server and Web services benchmark.
  - The workload simulates the activities of a business-to-business transactional application server operating in a 24x7 environment.

Measuring, Reporting, and Summarizing Performance

- All the TPC benchmarks measure performance in transactions per second.
- In addition, they include
  - a response time requirement, so that throughput performance is measured only when the response time limit is met.
  - system cost for a benchmark system must also be included, allowing accurate comparisons of cost-performance.

Measuring, Reporting, and Summarizing Performance

- Reporting Performance Results
- Guiding principle of reporting performance measurements should be reproducibility
  - List everything so that the another experimenter can duplicate the results
- These reports are excellent sources for finding the real cost of computing systems, since manufacturers compete on high performance and cost-performance.

Reference(s):

- **Hennessy, J.L., Patterson, D.A**. Computer Architecture: A Quantitative Approach (4th Ed)