EE 156: Advanced Topics in Computer Architecture

Spring 2023
Tufts University

Instructor: Prof. Mark Hempstead
mark.hempstead@tufts.edu
Lecture 1: Bandwidth, latency, power and Performance Metrics
[Chapter 1]

Lecture Outline

- Technology Trends
- Bandwidth vs. Latency
- Performance Metrics
  - Averaging
  - Amdahl's Law
  - CPU performance equation
- Benchmarks and Quantitative Analysis

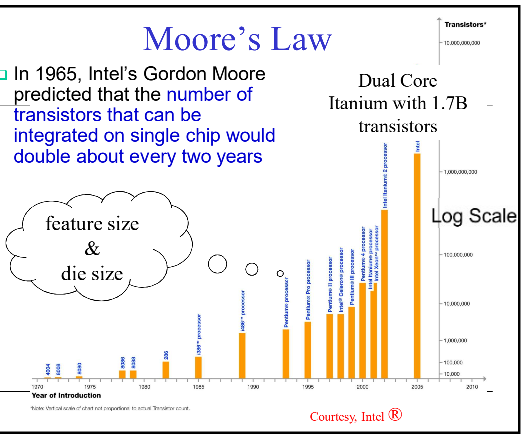
Homework: **Read Chapter 1** for more detail

Technology scaling

- Everyone has heard of Moore's Law. It's probably been mentioned in most newspapers at some point. But what does it really mean?
- Basic premise: every generation (1-2 years) all dimensions scale by 0.7.
  - $.7^2=0.5$, so devices/unit area doubles
  - This is less of a natural law, and more of a prediction of human progress. Actually making this happen has huge consequences

Moore's Law

- In 1965, Intel's Gordon Moore predicted that the number of transistors that can be integrated on single chip would double about every two years



Courtesy, Intel ®

Technology Scaling Road Map (ITRS no IRDS)

Year	2004	2006	2008	2010	2012	2014	2017	2019	2021	2024
Feature size (nm)	90	65	45	32	22	14	"10"	"7"	"5"	"3"
Intg. Capacity (BT)	2	4	8	16	33	83	162	332	651	1808

- Fun facts about 45nm transistors
  - 30 million can fit on the head of a pin
  - You could fit more than 2,000 across the width of a human hair
  - If car prices had fallen at the same rate as the price of a single transistor has since 1968, a new car today would cost about 1 cent

5

We've done clever things

- Architectural Innovations
  - Massive pipelining (good and bad!)
  - Branch Prediction, Register Renaming, OOO-issue/execution, Speculation
  - Hierarchical caches and prefetching
  - Virtual memory acceleration (TLBs)
- Circuit/Logic Innovations
  - New logic circuit families (dynamic logic)
  - Better CAD tools
  - Advanced computer arithmetic

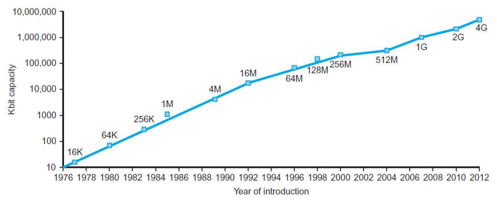
EE194/CS140 Mark Hempstead 6

How have we used these transistors?

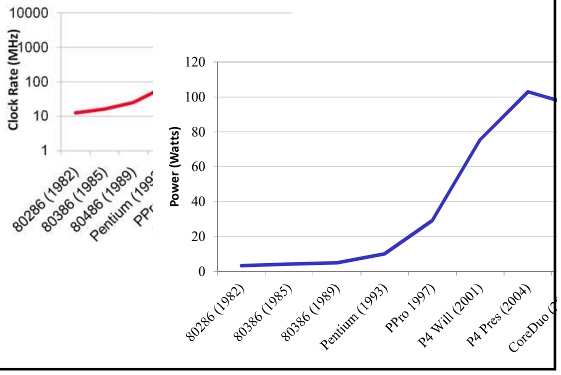
- More functionality on one chip
  - Early 1980s – 32-bit microprocessors
  - Late 1980s – On Chip Level 1 Caches
  - Early/Mid 1990s – 64-bit microprocessors, superscalar (ILP)
  - Late 1990s – On Chip Level 2 Caches
  - Early 2000s – Chip Multiprocessors, On Chip Level 3 Caches
  - Early 2010s – Many-Core, SoC integration, specialized hardware
  - Late 2010s – Many-Core Mobile, Domain Specific Processors (e.g. Google TPU), Support for Virtualization
- What is next?
  - How much more cache can we put on a chip? (Itanium2)
  - How many more cores can we put on a chip? (Niagara, etc)
  - What else can we put on chips? (Apple and Mobile SoCs)
  - What can we build custom chips for? (Machine Learning (TPU))

Another Example of Moore's Law Impact

DRAM capacity growth over 3 decades

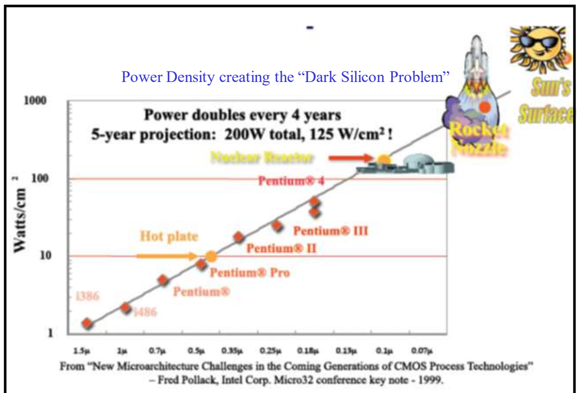


But What Happened to Clock Rates and Why?



-

Power Density creating the "Dark Silicon Problem"

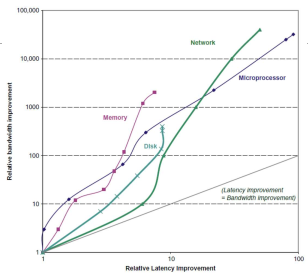


[Taylor, DAC and DaSi 2012]

Bandwidth and Latency

- What is the difference between bandwidth & latency?
- Not every performance measure has benefited equally from scaling
- Bandwidth (or throughput)
  - Total work done in a given time
  - 10,000-25,000X improvement for processors
  - 300-1200X improvement for memory and disks
- Latency (or response time)
  - Time between start and completion of an event
  - 30-80X improvement for processors
  - 6-8X improvement for memory and disks

Bandwidth and Latency



Log-log plot of bandwidth and latency milestones
[New Version from H&P 2018]

Rule of Thumb for Latency Lagging BW

- In the time that bandwidth doubles, latency improves by no more than a factor of 1.2 to 1.4
- Stated alternatively:
  Bandwidth improves by more than the square of the improvement in Latency

EE194/CS140 Mark Hempstead 13

Why has latency lagged?

- When we halve delay (and double frequency), we halve latency and double bandwidth. Both benefit. The same is true for most things that benefit latency (e.g., faster disks).
- Adding more transistors rarely benefits both equally:
  - Widen datapaths, add more cores or execution units, etc. all directly benefit bandwidth but not latency
  - Using the transistors to build a faster multiplier (e.g., a Wallace tree) or adder (e.g., better carry lookahead) would help latency; but we are far into diminishing returns in these areas.
- Most algorithms we will talk about (more pipelining, branch prediction, etc) help bandwidth and slightly *hurt* latency.
  - We'll keep discussing algorithms' effects on latency, bandwidth & power.

EE194/CS140 Mark Hempstead 14

Metrics

- We've seen that power, frequency, latency and bandwidth do not improve at the same rate.
- So what do we measure when we benchmark architectures against each other?
  - Most common answer: just measure execution time.
  - Execution time may measure latency (if we're only measuring the time for one action) or throughput (if we're measuring the time for 10K tasks and the program overlaps them).
  - Anyone can munge the results to, e.g., look only at computers in the same power class.

EE194/CS140 Mark Hempstead 15

Performance Metrics

- How do we decide the tasks?
- Some benchmarks
  - What the customer cares about, real applications
  - Representative programs (SPEC, SYSMARK, etc)
  - Kernels: Code fragments from real programs (Linpack)
  - Toy Programs: Quicksort
  - Synthetic Programs: Just a representative instruction mix (Whetsone, Dhrystone)

Better

EE194/CS140 Mark Hempstead 16

Averaging over a suite of benchmarks

- We can average total execution time:

$$\frac{1}{n}\sum_{i=1}^{n} \text{Time}_i$$

- This is the *arithmetic* mean
  - It should be used when measuring performance in execution *times*

EE194/CS140 Mark Hempstead 17

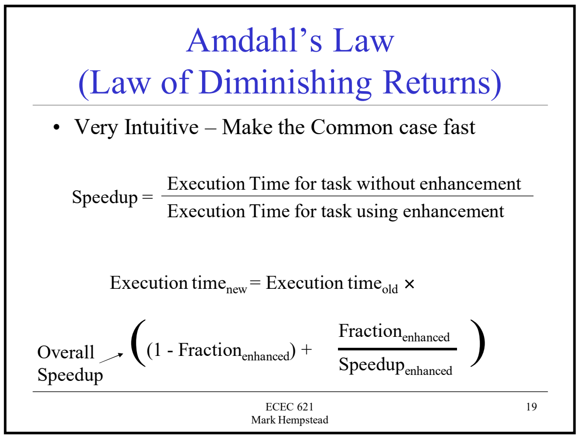
Averaging over a suite of benchmarks

- We can average the "normalized execution time," i.e., time / time-on-a-reference-machine
- Can only use **geometric mean** (arithmetic mean can vary depending on the reference machine)

$$\sqrt[n]{\prod_{i=1}^{n} ExecutionTimeRatio_i}$$

- This is what SPEC does
- Problem: Ratio not Execution Time is the result

EE194/CS140 Mark Hempstead 18



Amdahl's Law (Law of Diminishing Returns)

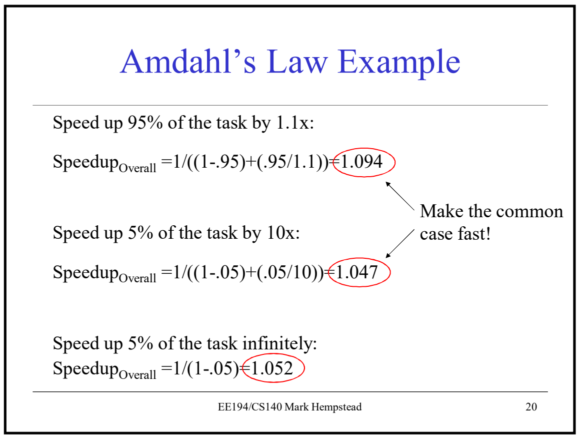
- Very Intuitive – Make the Common case fast

$$\text{Speedup} = \frac{\text{Execution Time for task without enhancement}}{\text{Execution Time for task using enhancement}}$$

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

Overall Speedup

ECEC 621
Mark Hempstead 19



Amdahl's Law Example

Speed up 95% of the task by 1.1x:

$\text{Speedup}_{Overall} = 1/((1-.95)+(.95/1.1)) = 1.094$

Make the common case fast!

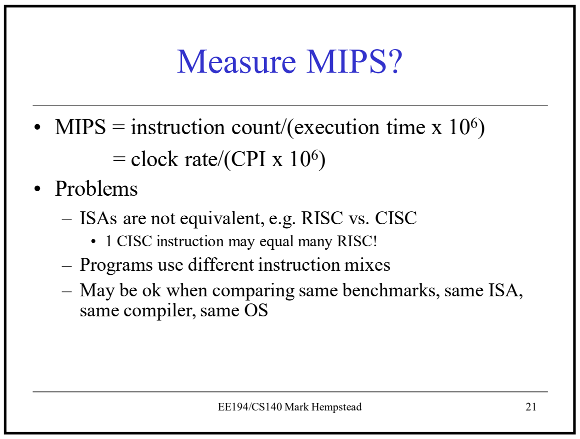
Speed up 5% of the task by 10x:

$\text{Speedup}_{Overall} = 1/((1-.05)+(.05/10)) = 1.047$

Speed up 5% of the task infinitely:

$\text{Speedup}_{Overall} = 1/(1-.05) = 1.052$

EE194/CS140 Mark Hempstead 20



Measure MIPS?

- MIPS = instruction count/(execution time x 10^6)
  = clock rate/(CPI x 10^6)
- Problems
  – ISAs are not equivalent, e.g. RISC vs. CISC
    - 1 CISC instruction may equal many RISC!
  – Programs use different instruction mixes
  – May be ok when comparing same benchmarks, same ISA, same compiler, same OS

EE194/CS140 Mark Hempstead 21

Measure MFLOPS

- Same as MIPS, just FP ops
- Not useful either
  - FP-intensive apps needed
  - Traditionally, FP ops were slow, INT can be ignored
  - BUT, now memory ops can be the slowest!
- "Peak MFLOPS" is a common marketing fallacy
  - Basically, it just says #FP-pipes X Clock Rate

Measure GHz?

- Is this a metric? Maybe as good as the others…
- One number, no benchmarks, what can be better?
- Many designs *are* frequency driven

Processor	Clock Rate	SPEC FP2000
IBM POWER3	450 MHz	434
Intel PIII	1.4 GHz	456
Intel Pentium 4	2.4 GHz	833
Itanium-2	1.0 GHz	1356

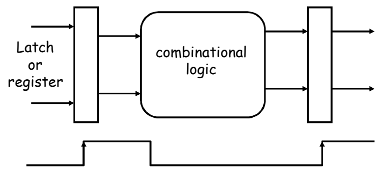
Performance: What to measure

- To increase predictability, collections of benchmark applications, called *benchmark suites*, are popular
- SPECCPU: popular desktop benchmark suite
  - SPECint2000 has 12 integer, SPECfp2000 has 14 integer pgms
  - SPECCPU2006 announced Spring 2006
  - SPECSFS (NFS file server) and SPECWeb (WebServer) added as server benchmarks
- Transaction Processing Council measures server performance and cost-performance for databases
  - TPC-C Complex query for Online Transaction Processing
  - TPC-H models ad hoc decision support
  - TPC-W a transactional web benchmark
  - TPC-App application server and web services benchmark
- **Splash2** Parallel Benchmark Suite (use in lab)
- New Emerging suites: PARSEC, RODINA, BIO-BENCH
- Embedded industry has its own benchmarks: Dhrystone, EEMBC, SPECjvm

New Emerging Benchmarks

- Significant updates over the few years with new application domains
- SPEC2017
  - Designed to provide performance measurements that can be used to compare compute-intensive workloads on different computer systems
  - 43 benchmarks organized into four suites: SPECspeed 2017 Integer, SPECspeed 2017 FP, SPECrate 2017 Integer, and SPECrate 2017 FP
- MLPerf https://mlperf.org/
  - A broad machine learning benchmark suite for measuring performance of ML software frameworks, ML hardware accelerators, and ML cloud platforms.
  - Image classification, Object Detection, Translation, Recommendation, Reinforcement Learning

What's a Clock Cycle?



- Old days: 10 levels of gates (e.g. 10 FO4)
- Today: determined by numerous time-of-flight issues + gate delays
  - clock propagation, wire lengths, drivers

THE Performance Equation

CPU time = Instruction_count x CPI x clock_cycle

or

$$\text{CPU time} = \frac{\text{Instruction\_count} \times \text{CPI}}{\text{clock\_rate}}$$

- These equations separate the three key factors that affect performance
  - Can measure the CPU execution time by running the program
  - The clock rate is usually given
  - Can measure overall instruction count by using profilers/ simulators without knowing all of the implementation details
  - CPI varies by instruction type and ISA implementation for which we must know the implementation details

CPU Performance Equation

- Execution Time = seconds/program

$$\frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

Program	Compiler (Scheduling)	Technology
Architecture (ISA)	Organization (uArch)	Physical Design
Compiler	Microarchitects	Circuit Designers

Alternately put…

- The equation again for total time:

$$\frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

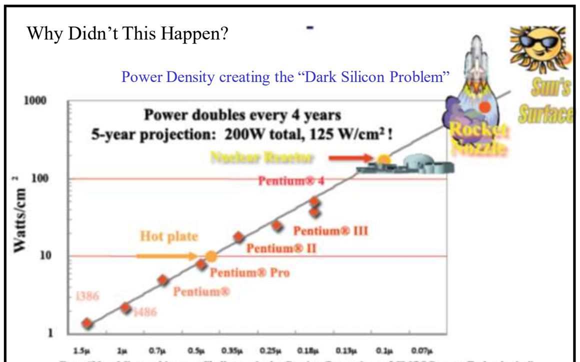
Can be re-phrased as
total time = (total instructions)*CPI/freq.

More CPI

- CPI can be useful even when a program is a mix of instruction types, each with their own CPI:
- Total CPU cycles $=\sum_i IC_i CPI_i$, where the summation is over each different instruction type.
- This can tell us where to focus our efforts:
  - But always remember Amdahl's Law
  - And remember that improving one instruction may hurt another, or even hurt your cycle time. Many published studies do in fact forget this.

Why Didn't This Happen?

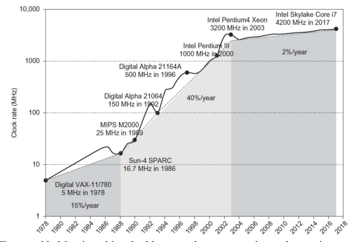
Power Density creating the "Dark Silicon Problem"



From "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies" – Fred Pollack, Intel Corp. Micro32 conference key note - 1999.

[Taylor, DAC and DaSi 2012]

Why Power stopped increasing

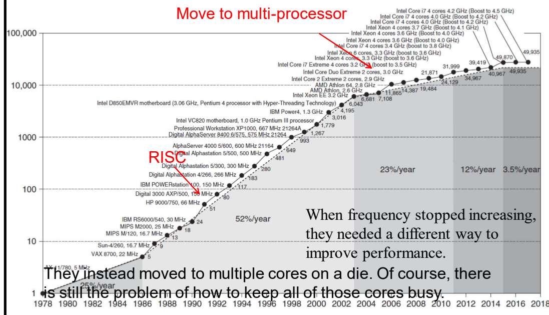


Because frequency stopped increasing!

- They avoided having chips double as nuclear reactors by no longer increasing frequency. But shouldn't that hurt performance?
- From H&P 6th Ed new version 2018

Crossroads: Uniprocessor Performance

From Hennessy and Patterson, *Computer Architecture: A Quantitative Approach*, 6th edition, 2018



When frequency stopped increasing, they needed a different way to improve performance.

They instead moved to multiple cores on a die. Of course, there is still the problem of how to keep all of those cores busy.

Chapter 1 Summary

- Technology Trends have a significant impact on design
  - Scaling of wires, transistors, power
  - Latency has been lagging BW
  - Designers have plenty of transistors? Just don't know what to do with them.
- Performance should be measured quantitatively
  - However there are different benchmarks and metrics available

H&P Turing Lecture

Backup

- List of Benchmarks
- Additional material hidden in this PPT. It is discussed in chapter 1, but we will cover it later in the class.

SPEC CPU2006: Integer Benchmarks

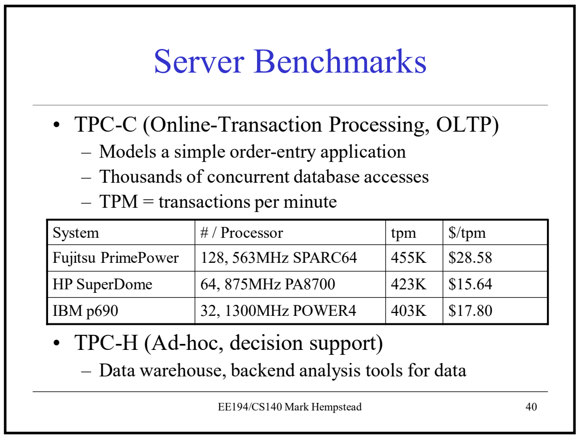
400.perlbenc	C	Perl Programming Language
401.bzip2	C	Bzip compression
403.gcc	C	GCC Compiler
429.mcf	C	Optimization. Vehicle scheduling for public transit
445.gobmk	C	AI the game of GO
456.hmmer	C	Protein sequence analysis using profile hidden
458.sjeng	C	AI Chess program
462.libquant	C	Simulates a quantum computer, running Shor's
464.h264ref	C	Video Compression, H.264/AVC standard
471.omnetp	C++	Uses the OMNet++ discrete event simulator to
473.astar	C++	Pathfinding library for 2D maps, including the well
483.xalancb	C++	XML Processing

SPEC CPU2006: Floating Point Benchmarks

410.bw aves	Fortran	Fluid Dynamics
416.gamess	Fortran	Quantum Chemistry
433.milc	C	Physics / Quantum Chromodynamics
434.zeusmp	Fortran	Physics / computational fluid dynamics
435.gromacs	C, Fortran	equations of motion for hundreds to millions
436.cactusADM	C, Fortran	Fortran Physics / General Relativity
437.leslie3d	Fortran	Large-Eddy Simulations
444.namd	C++	Biology / Molecular Dynamics
447.dealII	C++	Finite Element Analysis
450.soplex	C++	Linear Programming, Optimization
453.povray	C++	Image Ray-tracing
454.calculix	C, Fortran	Structural Mechanics
459.GemsFDTD	Fortran	Computational Electromagnetics
465.tonto	Fortran	Quantum Chemistry
470.lbm	C	Fluid Dynamics
481.w rf	C, Fortran	Weather modeling
482.sphinx3	C	Speech recognition

Parsec Benchmarks

- SPEC CPU does not include multithreaded and emerging workloads
- Other suites such as PARSEC, SPLASH-2, Rodina (CUDA) include such workloads
- List of PARSEC benchmarks:
  - **blackscholes** - Option pricing with Black-Scholes Partial Differential Equation (PDE)
  - **bodytrack** - Body tracking of a person
  - **canneal** - Simulated cache-aware annealing to optimize routing cost of a chip design
  - **dedup** - Next-generation compression with data deduplication
  - **facesim** - Simulates the motions of a human face
  - **ferret** - Content similarity search server
  - **fluidanimate** - Fluid dynamics for animation purposes with Smoothed Particle Hydrodynamics (SPH) method
  - **freqmine** - Frequent itemset mining
  - **raytrace** - Real-time raytracing
  - **streamcluster** - Online clustering of an input stream
  - **swaptions** - Pricing of a portfolio of swaptions
  - **vips** - Image processing (Project Website)
  - **x264** - H.264 video encoding



Server Benchmarks

- TPC-C (Online-Transaction Processing, OLTP)
  - Models a simple order-entry application
  - Thousands of concurrent database accesses
  - TPM = transactions per minute

System	# / Processor	tpm	$/tpm
Fujitsu PrimePower	128, 563MHz SPARC64	455K	$28.58
HP SuperDome	64, 875MHz PA8700	423K	$15.64
IBM p690	32, 1300MHz POWER4	403K	$17.80

- TPC-H (Ad-hoc, decision support)
  - Data warehouse, backend analysis tools for data