CS2410: Computer Architecture

Technology, software, performance, and cost issues

Sangyeun Cho

Computer Science Department
University of Pittsburgh

Welcome to CS2410!

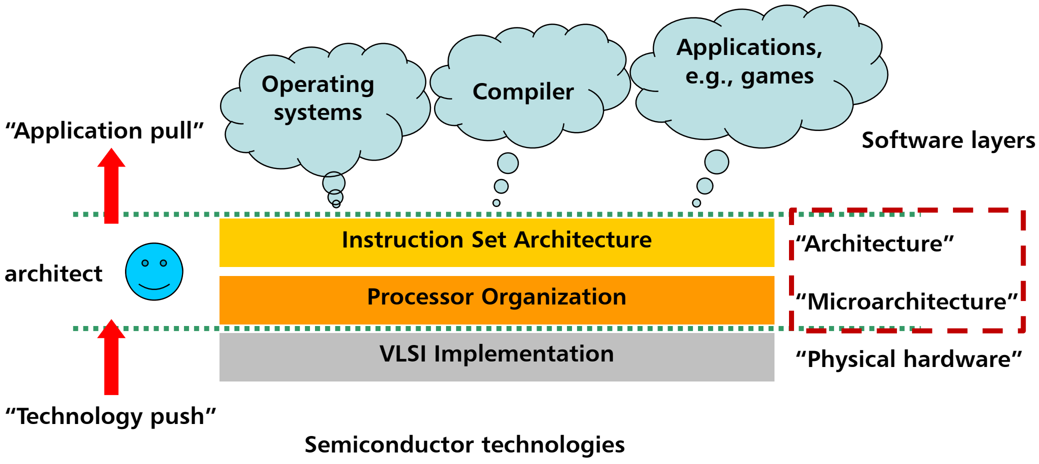
- This is a grad-level introduction to Computer Architecture
- Let's take a look at the course info. Sheet
- Schedule

Computer architecture?

- A Computer Science discipline that explores:
  - Principles and practices to exploit characteristics of hardware & software artifacts relevant for computer systems hardware design;
  - Computer hardware design itself; and
  - Changing interaction between hardware and software

- Goals
  - Sustain the historic computer performance (what is performance?) improvement rate and expand a computer's capabilities
  - Keep the cost down

Computer architecture?



Operating systems

Compiler

Applications, e.g., games

Software layers

"Application pull"

Instruction Set Architecture

"Architecture"

architect

Processor Organization

"Microarchitecture"

VLSI Implementation

"Physical hardware"

"Technology push"

Semiconductor technologies

Uniprocessor performance

- Performance = 1 / time
- Time = IC × CPI × CCT

- Instructions/program
  - Also called "instruction count" (IC above)
  - Represents how many (dynamic) instructions are required to finish the program
  - Highly depends on "architecture"

- Clocks/instruction
  - Also called CPI (Clocks Per Instruction)
  - Depends on pipelining and "microarchitecture" implementation

- Time/clocks
  - Also called clock cycle time (inverse of frequency)
  - Highly depends on circuit & VLSI chip realization

Today's topics

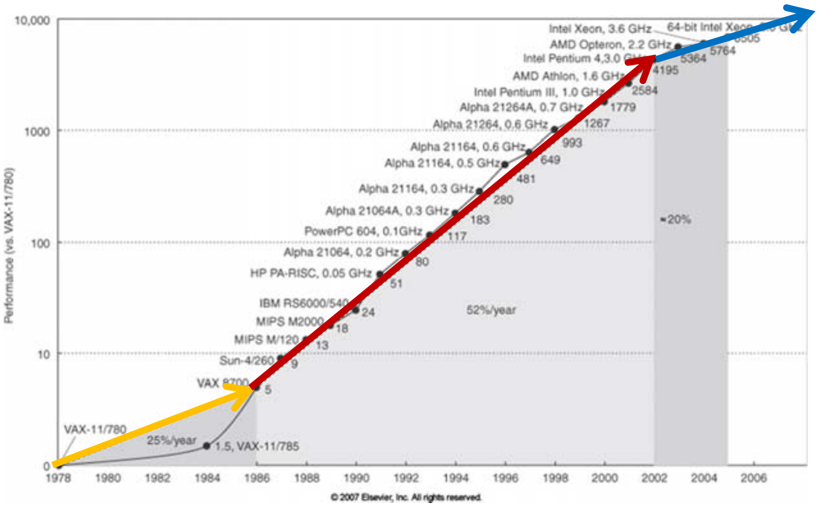
- Technology trends
  - "Switches"
  - Impact of CMOS scaling

- Cost
  - IC chip cost

- Performance
  - Benchmarks
  - Summarizing performance measurements
  - Quantitative approach to computer design

- Application trends

Uniprocessor performance trend



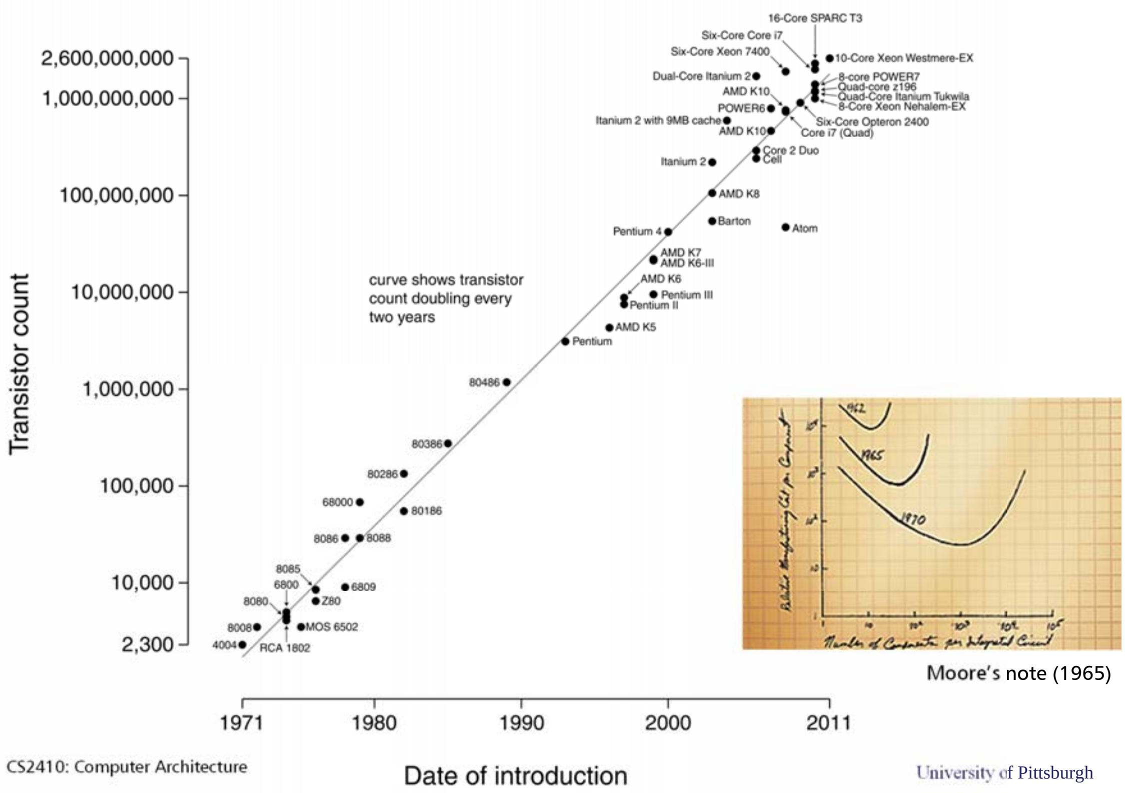
© 2007 Elsevier, Inc. All rights reserved.

Uniprocessor performance hurdles

- Maximum power dissipation
  - 100W ~ 150W
- Little instruction-level parallelism left
- Little-changing memory latency

- “We are dedicating all of our future product development to multicore designs. … This is a sea change in computing.”
  - Paul Otellini, President, Intel (2004)

Microprocessor Transistor Counts 1971-2011 & Moore's Law



Moore's note (1965)

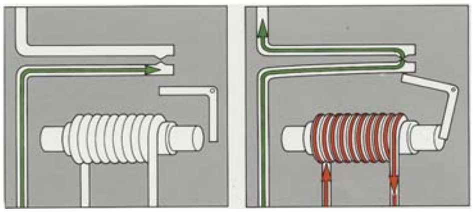
How does technology scaling help?

- Time = (inst. count)×(clocks per inst.)×(clock cycle time)
- Faster circuit
  - Scaling makes transistors not only smaller but also faster
  - Faster clock ⇒ smaller clock cycle time
- More transistors
  - Larger L2 caches (relatively simple design change)
  - Smaller CPI
- Design changes enabled by scaling
  - Deep pipeline using more pipeline registers
  - Superscalar pipeline using more functional units
  - Larger, more sophisticated branch predictors
  - …
  - Multicores

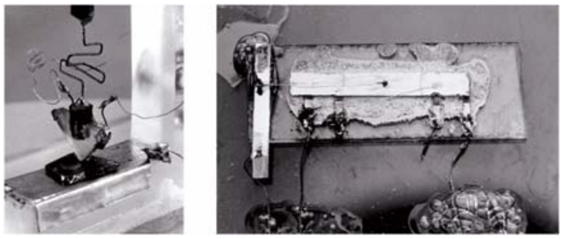
Switches

- Building block for digital logic
  - NAND, NOR, NOT, …
- Technology advances have provided designers with switches that are
  - Faster;
  - Lower power;
  - More reliable (e.g., vacuum tube vs. transistor); and
  - Smaller.
- Nano-scale technologies will not continue promising the same good properties

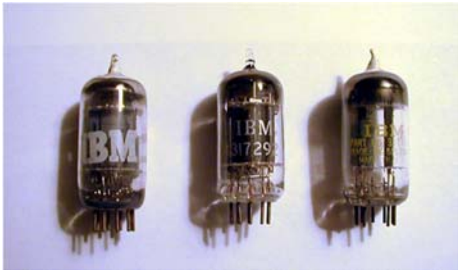
History of switches



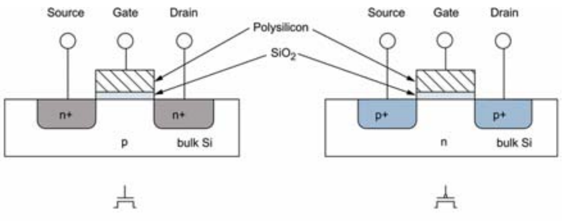
Called "relay"; Mark I (1944)



Bell lab. (1947); Kilby's first IC (1957)



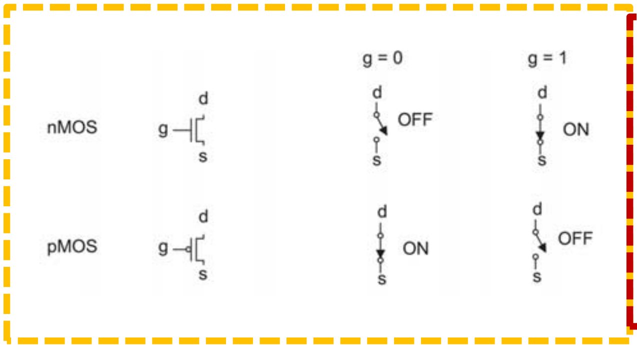
Vacuum tubes; ENIAC (1946, 18k tubes)

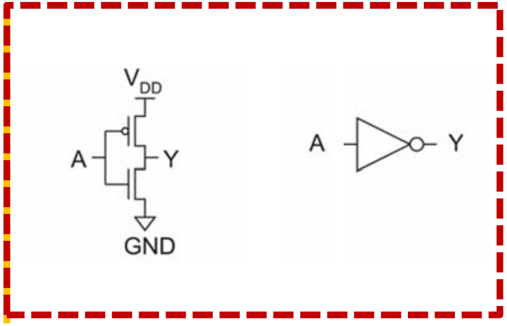


Solid-state MOS devices

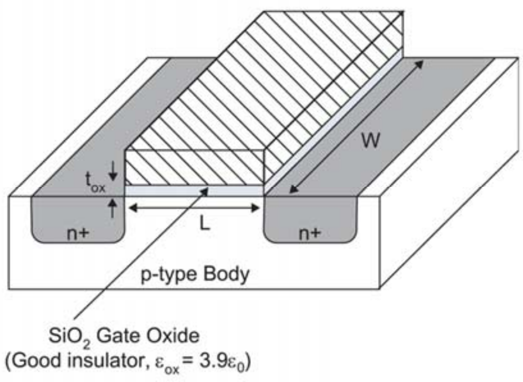
MOS transistors

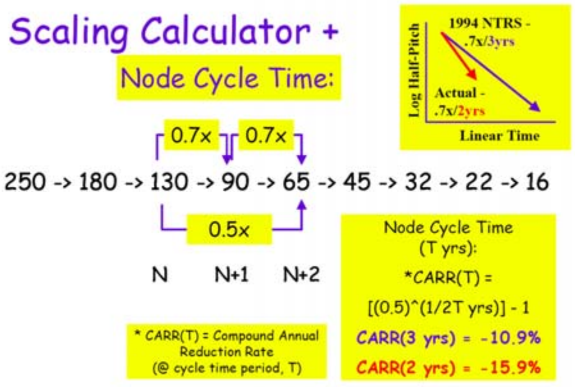
- Today's chips heavily depend on CMOS (complementary MOS)-style logic design





MOS transistor scaling



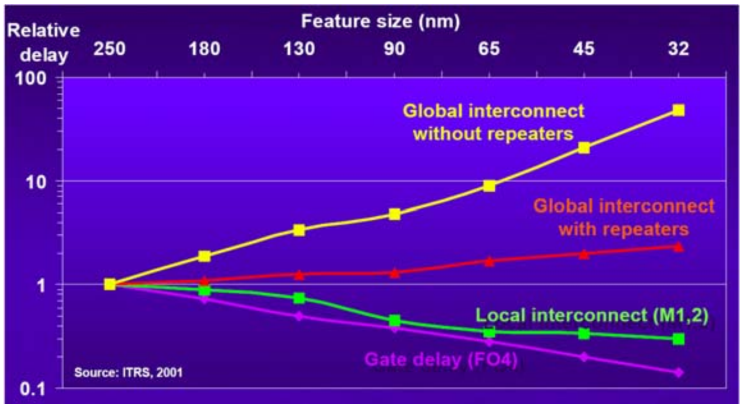


Impact of MOS transistor scaling

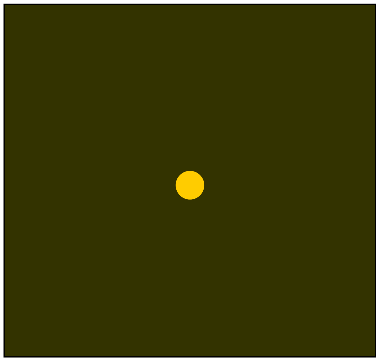
- In general
  - Smaller transistors (i.e., density doubling with each new generation)
  - Faster transistors (latency ∝ L)
  - Roughly constant wire delay (⇒ relatively slow wires!)
  - Lower supply voltage (⇒ lower dynamic power)

- Downside
  - Increased global wire delay
  - Increased power density (W/cm^2)
  - Increased leakage power
  - Increased susceptibility to noise and transient errors
  - On-chip variation
  - Cost of manufacturing

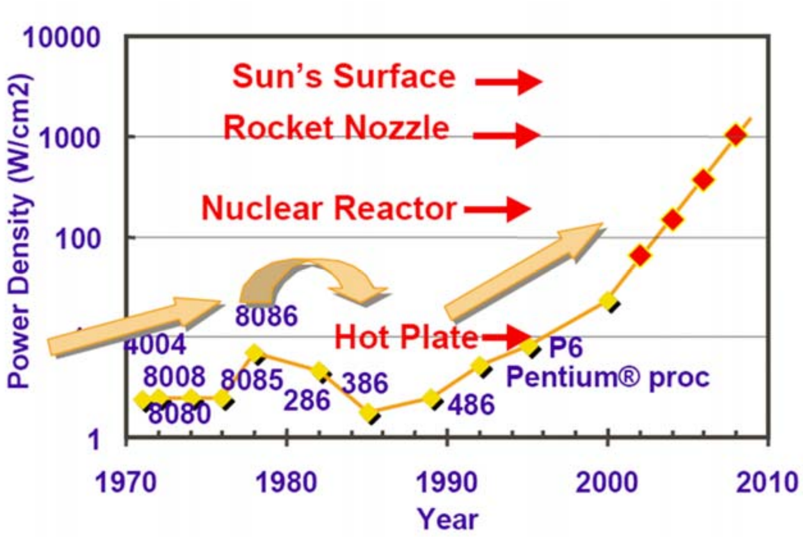
Global wire delay



Source: ITRS, 2001

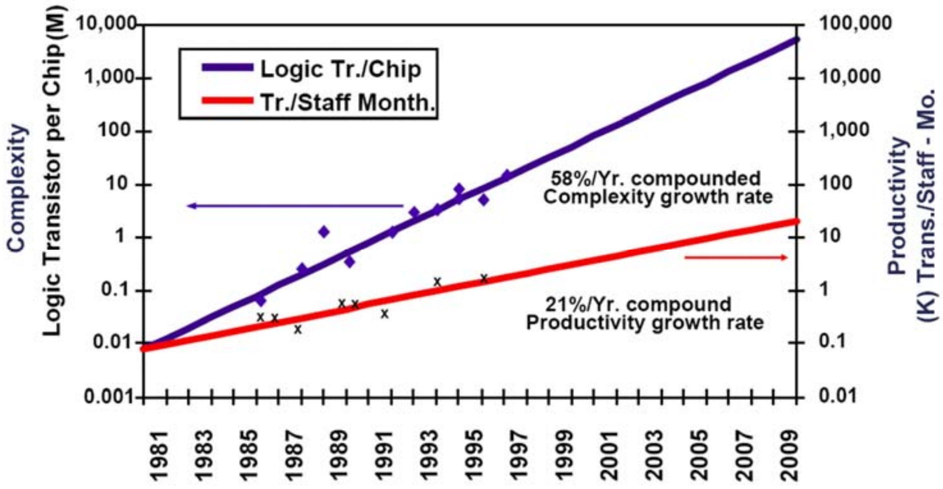


Power density



S. Borkar

Productivity



Source: Sematech

Component-level performance trend

- Four key components in a computer system
  - Disks
  - Memory
  - Network
  - Processors

- Compare ~1980 Archaic (or “Nostalgic”) vs. ~2000 Modern (or “Newfangled”)
  - (Patterson)

- Metric
  - Bandwidth: # operations or events per unit time
  - Latency: elapsed time for a single operation or event

Disk: Archaic vs. Modern

CDC Wren I, 1983

- 3,600 RPM
- 0.03 GB
- Tracks/inch: 800
- Bits/inch: 9,550
- Three 5.25” platters

- Bandwidth: 0.6 MB/s
- Latency: 48.3 ms
- Cache: none

Seagate 373453, 2003

- 15,000 RPM (4x)
- 73.4 GB (2,500x)
- Tracks/inch: 64,000 (80x)
- Bits/inch: 533,000 (60x)
- Four 2.5” platters

- Bandwidth: 86 MB/s (140x)
- Latency: 5.7 ms (8x)
- Cache: 8MB

Memory: Archaic vs. Modern

- 1980 DRAM (asynchronous)
- 0.06 Mbits/chip
- 64,000 xtors, 35 mm^2
- 16-bit data bus per module, 16 pins/chip
- 13 Mbytes/sec
- Latency: 225 ns
- (no block transfer)

- 2000 Double Data Rate Synchr. (clocked) DRAM
- 256.00 Mbits/chip (4000X)
- 256,000,000 xtors, 204 mm^2
- 64-bit data bus per DIMM, 66 pins/chip (4X)
- 1600 Mbytes/sec (120X)
- Latency: 52 ns (4X)
- Block transfers (page mode)

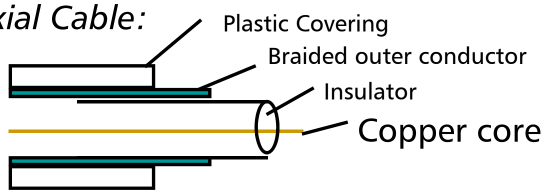
LANs: Archaic vs. Modern

- Ethernet 802.3
- Year of Standard: 1978
- 10 Mbits/s link speed
- Latency: 3000 μsec
- Shared media
- Coaxial cable

- **Ethernet 802.3ae**
- **Year of Standard: 2003**
- **10,000 Mbits/s (1000X) link speed**
- **Latency: 190 μsec (15X)**
- **Switched media**
- **Category 5 copper wire**

"Cat 5" is 4 twisted pairs in bundle

Coaxial Cable:

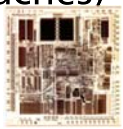


Twisted Pair:

Copper, 1mm thick,
twisted to avoid antenna effect

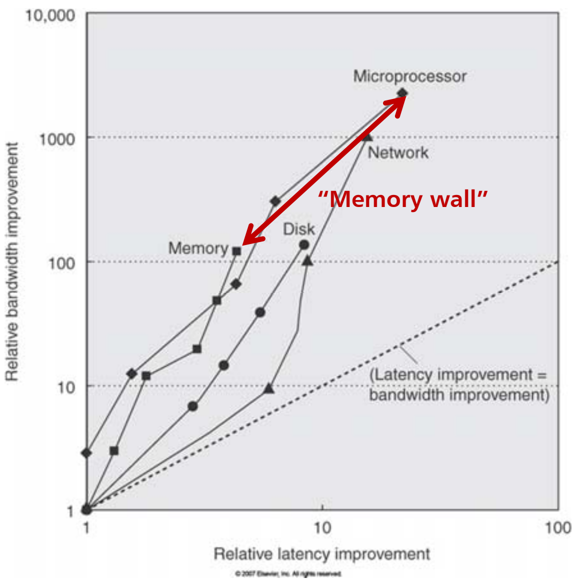
CPUs: Archaic vs. Modern

- 1982 Intel 80286
- 12.5 MHz
- 2 MIPS (peak)
- Latency 320 ns
- 134,000 xtors, 47 mm^2
- 16-bit data bus, 68 pins
- Microcode interpreter, separate FPU chip
- (no caches)



- 2001 Intel Pentium 4
- 1500 MHz (120X)
- 4500 MIPS (peak) (2250X)
- Latency 15 ns (20X)
- 42,000,000 xtors, 217 mm^2
- 64-bit data bus, 423 pins
- 3-way superscalar, Dynamic translation to RISC, Superpipelined (22 stage), Out-of-Order execution
- On-chip 8KB Data caches, 96KB Instr. Trace cache, 256KB L2 cache

Latency lags bandwidth (last ~20 years)

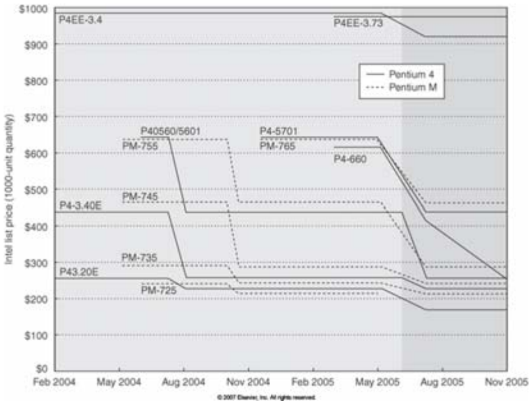


- CPU
  - 21x vs. 2250x
- Ethernet
  - 16x vs. 1000x
- Memory module
  - 4x vs. 120x
- Disk
  - 8x vs. 143x

Rule of thumbs: latency lagging BW

- In the time that bandwidth doubles, latency improves by no more than a factor of 1.2 to 1.4
  - (Capacity improves faster than bandwidth)
- In other words, bandwidth improves by more than the square of the improvement in latency

Cost trend



- Time
  - Learning curve
  - Change in yield
- Volume
  - Decreases cost, increases efficiency
  - “Shrinking” by deploying next-generation technology (without changing the design itself)
- Commoditization
  - Standards push this
  - Multiple vendors compete

IC (Integrated Circuit) cost

- Cost of IC = (cost of production) / (final test yield)

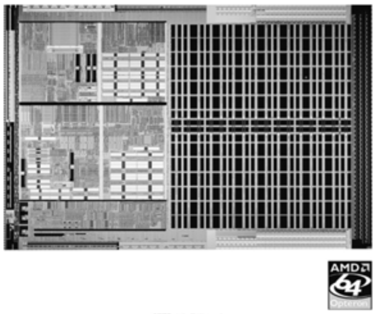
- Cost of production
  - Cost of die
  - Cost of testing die
  - Cost of packaging and final test

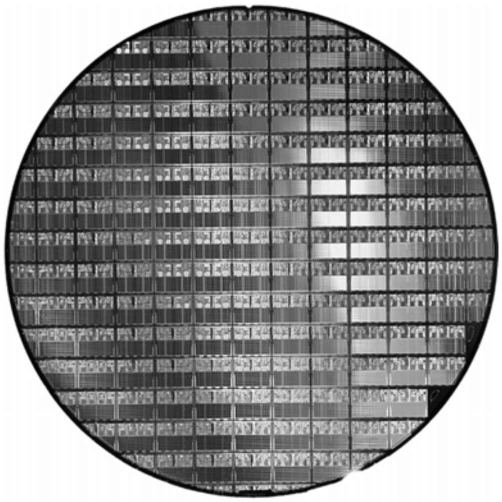
- Cost of production at time line
  - NRE (Non-Recurring Engineering) cost
    - R & D
    - Mask
  - Chip production
    - “Front end”
    - “Back end” – packaging, etc.
  - Test cost

- Cost of die = (cost of wafer) / ((dies per wafer) × (die yield))

IC (Integrated Circuit) cost

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





IC (Integrated Circuit) cost

$$\text{Die yield} = \text{wafer yield} \times \left(1 + \frac{\text{defect density} \times \text{die area}}{\alpha}\right)^{-\alpha}$$

- defect density = # defects in unit area
- defect density × die area will be then average # of defects per die

- α: manufacturing complexity
  - 2006 CMOS process: $\alpha = 4.0$

Performance analysis

- Which computer is faster for what you want to do?
  - Time matters
  - Workload matters

- Throughput (jobs/sec) vs. latency (sec/job)
  - Single processor vs. multiprocessor
  - Pentium4 @2GHz vs. Pentium4 @4GHz

- Commonly used techniques
  - Direct measurement
  - Simulation
  - Analytical modeling

Performance analysis

- Combination of
  - Measurement
  - Interpretation
  - Communication
- Overall performance vs. specific aspects
  - Choice of metric
- Considerations in performance analysis
  - Perturbation
  - Accuracy
  - Reproducibility
  - …

Performance report

- Reproducibility
  - Provide all necessary details so that others can reproduce the same result
  - Machine configuration, compiler flags, …
- Single number is attractive, but
  - It does not show how a new feature affects different programs
  - It may in fact mislead; a technique good for a program may be bad for others

Performance analysis techniques

- Direct measurement
  - Can provide the best result – no simplifying assumptions
  - Not flexible (difficult to change parameters)
  - Prone to perturbation (if instrumented)
  - Made much easier these days by using performance counters
- Simulation
  - Very flexible
  - Time consuming
  - Difficult to model details and validate
- Analytical modeling
  - Quick insight for overall behaviors
  - Limited applicability
  - Used to confine simulation scope, validate simulations, etc.

Performance metrics

- (Preferably) single number that essentially extracts a desired characteristic
  - Cache hit rate
  - AMAT (Average Memory Access Time)
  - IPC (Instructions Per Cycle)
  - Time (or delay)
  - Energy-delay product
  - …

Comparing two

$$\frac{\text{Execution time}_{\text{Y}}}{\text{Execution time}_{\text{X}}} = n$$

"X is *n* times faster than Y"

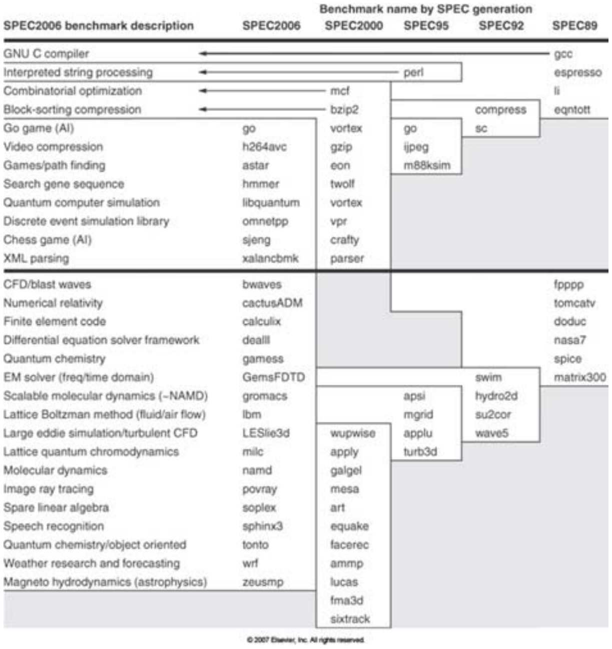
- Two different machines
- Two different options (e.g., memory sizes) on a machine
- …

$$n = \frac{\text{Execution time}_{\text{Y}}}{\text{Execution time}_{\text{X}}} = \frac{\text{Performance}_{\text{X}}}{\text{Performance}_{\text{Y}}}$$

Benchmarks

- Real programs
- Benchmark suites: a set of real applications
  - SPEC CPU 2006 (desktop and servers)
  - EEMBC, SPECjvm (embedded)
  - TPC-C, TPC-H, SPECjbb, ECperf (servers)
  - …
- Kernels: important pieces of codes from real applications
  - Livermore loops, …
- Toy programs: small programs that we easily understand
  - Quicksort
  - Sieves of Eratosthenes, …
- Synthetic program: to mimic a program behavior "uniformly"
  - Dhrystone
  - Whetstone, …

SPEC CPU2006



SPEC2006 benchmark description	SPEC2006	SPEC2000	SPEC95	SPEC92	SPEC89
		Benchmark name by SPEC generation			
GNU C compiler					gcc
Interpreted string processing			perl		espresso
Combinatorial optimization		mcf			li
Block-sorting compression		bzip2		compress	eqntott
Go game (AI)	go	vortex	go	sc	
Video compression	h264avc	gzip	ijpeg		
Games/path finding	astar	eon	m88ksim		
Search gene sequence	hmmer	twolf			
Quantum computer simulation	libquantum	vortex			
Discrete event simulation library	omnetpp	vpr			
Chess game (AI)	sjeng	crafty			
XML parsing	xalancbmk	parser			
CFD/blast waves	bwaves				fpppp
Numerical relativity	cactusADM				tomcatv
Finite element code	calculix				doduc
Differential equation solver framework	dealII				nasa7
Quantum chemistry	gamess				spice
EM solver (freq/time domain)	GemsFDTD			swim	matrix300
Scalable molecular dynamics (~NAMD)	gromacs		apsi	hydro2d	
Lattice Boltzman method (fluid/air flow)	lbm		mgrid	su2cor	
Large eddie simulation/turbulent CFD	LESlie3d	wupwise	applu	wave5	
Lattice quantum chromodynamics	milc	apply	turb3d		
Molecular dynamics	namd	galgel			
Image ray tracing	povray	mesa			
Spare linear algebra	soplex	art			
Speech recognition	sphinx3	equake			
Quantum chemistry/object oriented	tonto	facerec			
Weather research and forecasting	wrf	ammp			
Magneto hydrodynamics (astrophysics)	zeusmp	lucas			
		fma3d			
		sixtrack			

- 12 integer programs
  - 9 use C
  - 3 use C++
- 17 floating-point programs
  - 3 use C
  - 4 use C++
  - 6 use Fortran
  - 4 use a mixture of C and Fortran
- Package available at /afs/cs.pitt.edu/projects/spec-cpu2006

Summarizing performance results

- Arithmetic mean
  - When dealing with times
- Weighted arithmetic mean
- Geometric mean
  - When dealing with ratios
  - SPEC CPU uses this method

$$\text{Geometric mean} = \sqrt[n]{\prod_{i=1}^{n} \text{sample}_i}$$

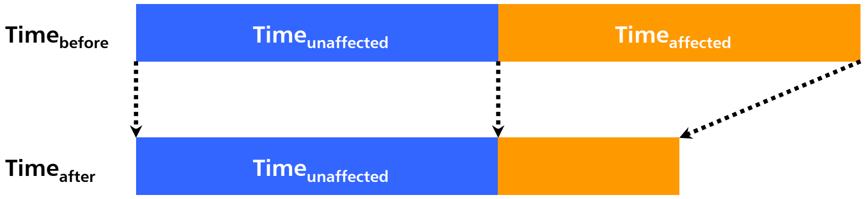
- In the case of SPEC, sample$_i$ is the SPECRatio for program *i*

SPEC2k scoring method

- Get execution time of each benchmark
- Get a ratio for each benchmark by dividing the time with that of the reference machine
  - Sun Ultra 5_10, 300MHz SPARC, 256MB memory
  - Its score is 100
- Get a geometric mean of all the computed ratios

Amdahl's law

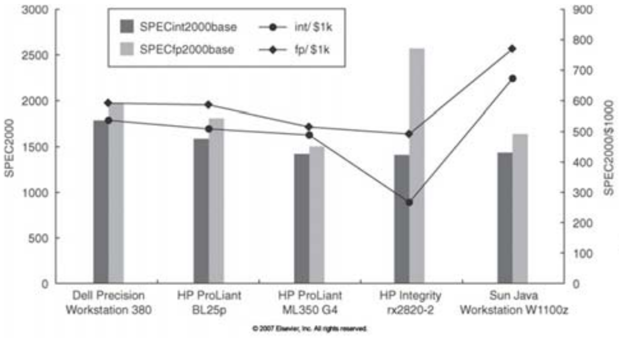
- Optimization or parallelization usually applies to a portion
  - Places "limitation" of the scope of an optimization
  - Leads us to focus on "common cases"
  - "Make common case fast and rare case accurate"

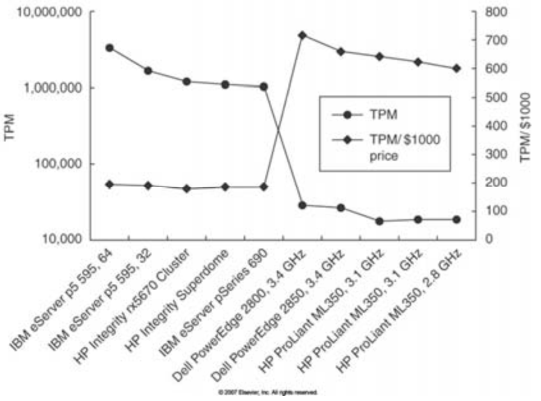

$\text{Time}_{\text{before}}$
$\text{Time}_{\text{unaffected}}$
$\text{Time}_{\text{affected}}$
$\text{Time}_{\text{after}}$
$\text{Time}_{\text{unaffected}}$

Principle of locality

- Locality found in memory access instructions
  - Temporal locality: if an item is referenced, it will tend to be referenced again soon
  - Spatial locality: if an item is referenced, items whose addresses are close by tend to be referenced soon
  - …
- 90/10 locality rule
  - A program executes about 90% of its instructions in 10% of its code
- We will look at how this principle is exploited in various microarchitecture techniques

Performance vs. performance-price



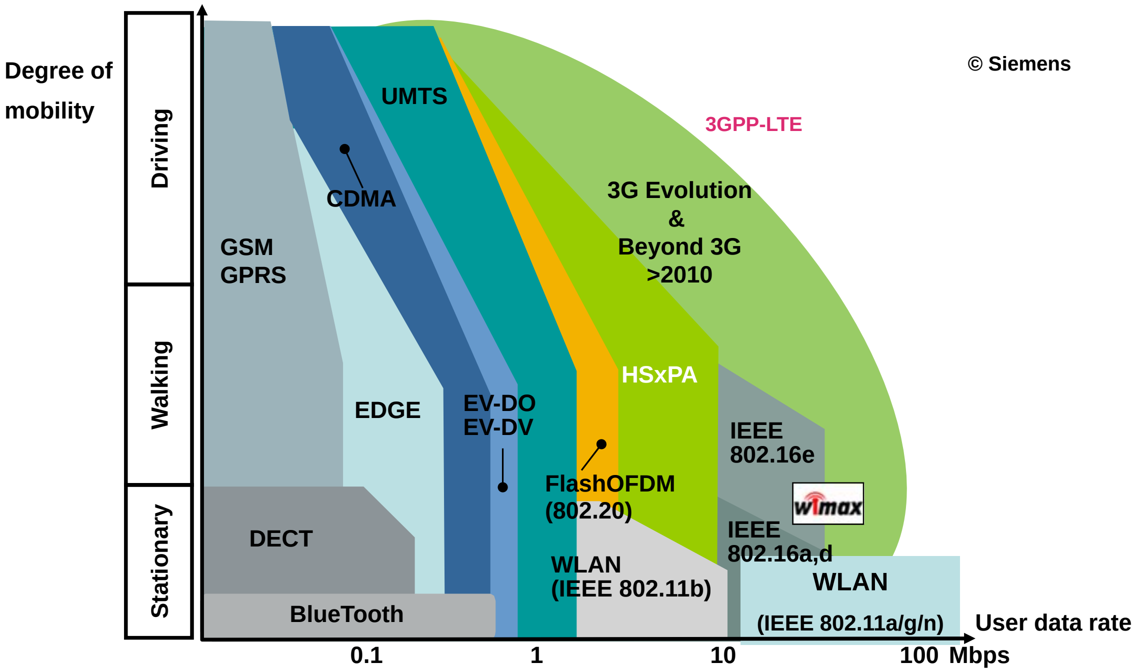


Killer apps?

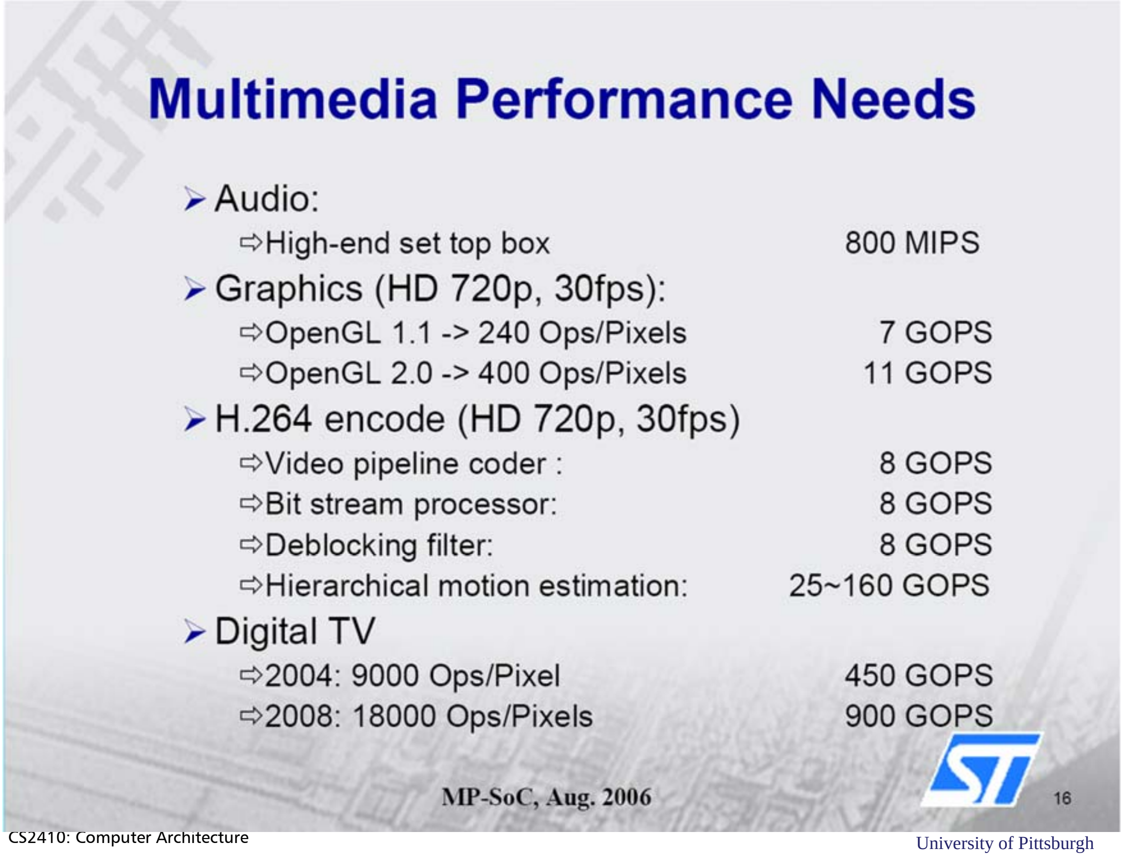
- Multimedia applications
- Games
  - 3D graphics
  - Physics simulation
- Virtual reality
- RMS (Recognition, Mining, and Synthesis)
  - Speech recognition
  - Video mining
  - Voice synthesis
  - …

- (Cf.) Software defined radio and other mobile applications

Software defined radio



© Siemens



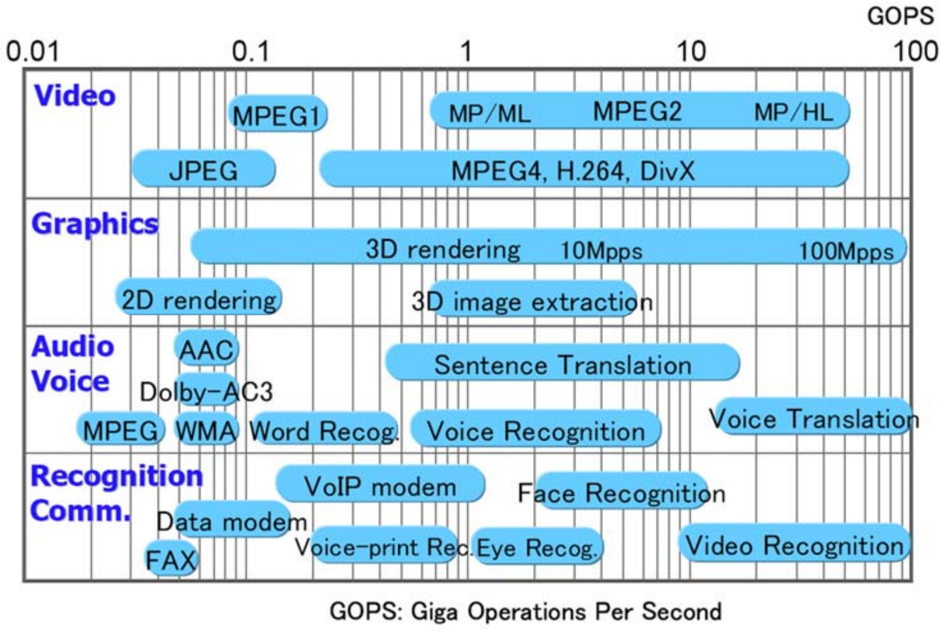
Multimedia Performance Needs

- Audio:
  - ⇨High-end set top box — 800 MIPS
- Graphics (HD 720p, 30fps):
  - ⇨OpenGL 1.1 -> 240 Ops/Pixels — 7 GOPS
  - ⇨OpenGL 2.0 -> 400 Ops/Pixels — 11 GOPS
- H.264 encode (HD 720p, 30fps)
  - ⇨Video pipeline coder : — 8 GOPS
  - ⇨Bit stream processor: — 8 GOPS
  - ⇨Deblocking filter: — 8 GOPS
  - ⇨Hierarchical motion estimation: — 25~160 GOPS
- Digital TV
  - ⇨2004: 9000 Ops/Pixel — 450 GOPS
  - ⇨2008: 18000 Ops/Pixels — 900 GOPS

MP-SoC, Aug. 2006

16

Multimedia performance needs



(K. Uchiyama, ACSAC '07)