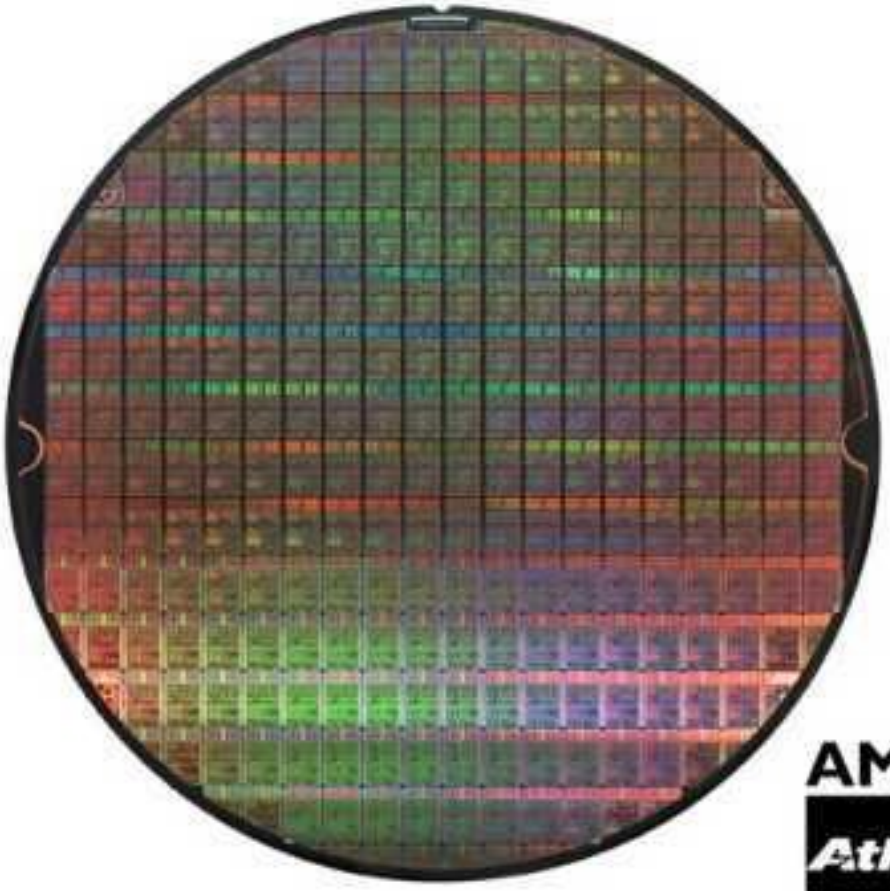
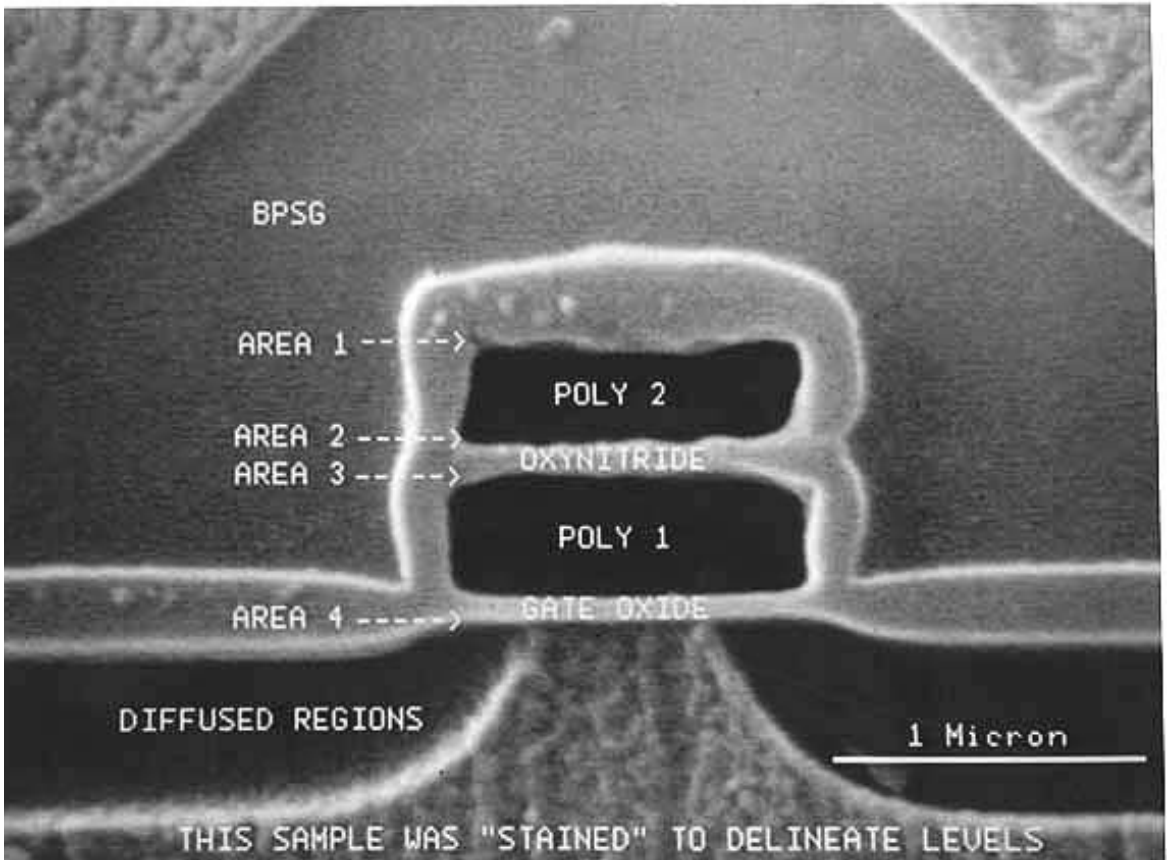
Technology

Giorgio Richelli

What Comes out of the Fab

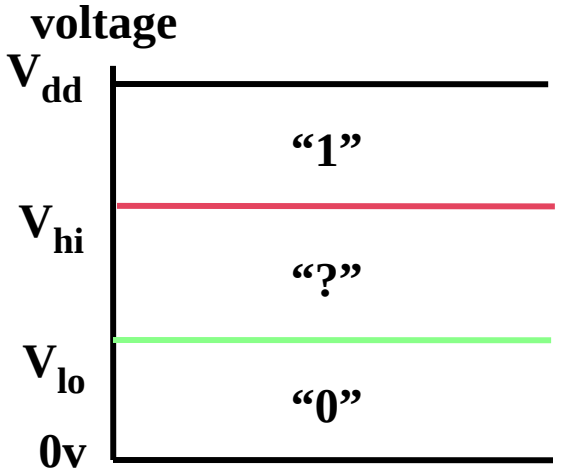


Transistor


BPSG
AREA 1
POLY 2
AREA 2
AREA 3
OXYNITRIDE
POLY 1
AREA 4
GATE OXIDE
DIFFUSED REGIONS
1 Micron
THIS SAMPLE WAS "STAINED" TO DELINEATE LEVELS

Abstractions in Logic Design

- In physical world
  - Voltages, Currents
  - Electron flow
- In logical world - abstraction
  - $V < V_{lo} \Rightarrow$ “0” = FALSE
  - $V > V_{hi} \Rightarrow$ “1” = TRUE
  - In between - forbidden
- Simplify design problem



The Ugly Truth

- Transistors are not ideal switches!
  - Gate Capacitance (C_g)
  - S-D resistance (R)
  - Drain capacitance
- Issues
  - Delay - actually takes real time to turn transistors on and off
  - Power/Energy (static versus dynamic power)
  - Noise (from transistors, power rails)

- Reducing transistor size
  - Decrease C_g

Define and quantify power (1 / 2)

- For CMOS chips, traditional dominant energy consumption has been in switching transistors, called *dynamic power*

$$Power_{dynamic} = 1/2 \times CapacitiveLoad \times Voltage^2 \times FrequencySwitched$$

- *CapacitiveLoad:*
  - a function of number of transistors connected to output and technology, which determines capacitance of wires and transistors
- Slowing clock rate reduces power, but not energy
- Dropping voltage helps both
  - Thus now is approx. 1V (from 5V)
- To save power, most CPUs now slow down or turn off clock to inactive modules.

Define and quantify power (2 / 2)

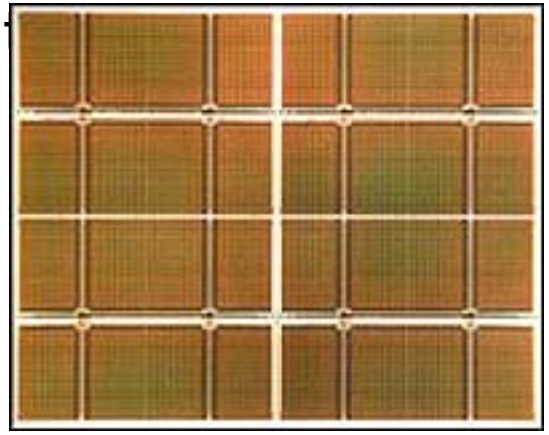
- Because leakage current flows even when a transistor is off, now *static power* is getting important

$$Power_{static} = Current_{static} \times Voltage$$

- Leakage current increases in processors with smaller transistor sizes
- Increasing number of transistors increases power even if they are turned off
- In 2006, goal for leakage was 25% of total power consumption; high performance designs at 40%
- Very-low-power systems gate voltage to inactive modules to control loss due to leakage

Memory

- Moves information in time (wires move it in space)
- Provides state
- Requires energy to change state
  - Feedback circuit - SRAM
  - Capacitors – DRAM
  - Magnetic media - disk
- Required for memories
  - Storage medium
  - Write mechanism
  - Read mechanism



4Gb DRAM Die

Tracking Performance Trends

- Drill down into 4 technologies:
  - Disks,
  - Memory,
  - Network,
  - Processors
- Compare ~1980 Archaic vs. ~2000 Modern
  - Performance milestones in each technology
- Compare for bandwidth vs. latency improvements in performance over time
- Bandwidth: number of events per unit time
  - E.g., M bits / second over network, M bytes / second from disk
- Latency: elapsed time for a single event
  - E.g., one-way network delay in microseconds, average disk access time in milliseconds

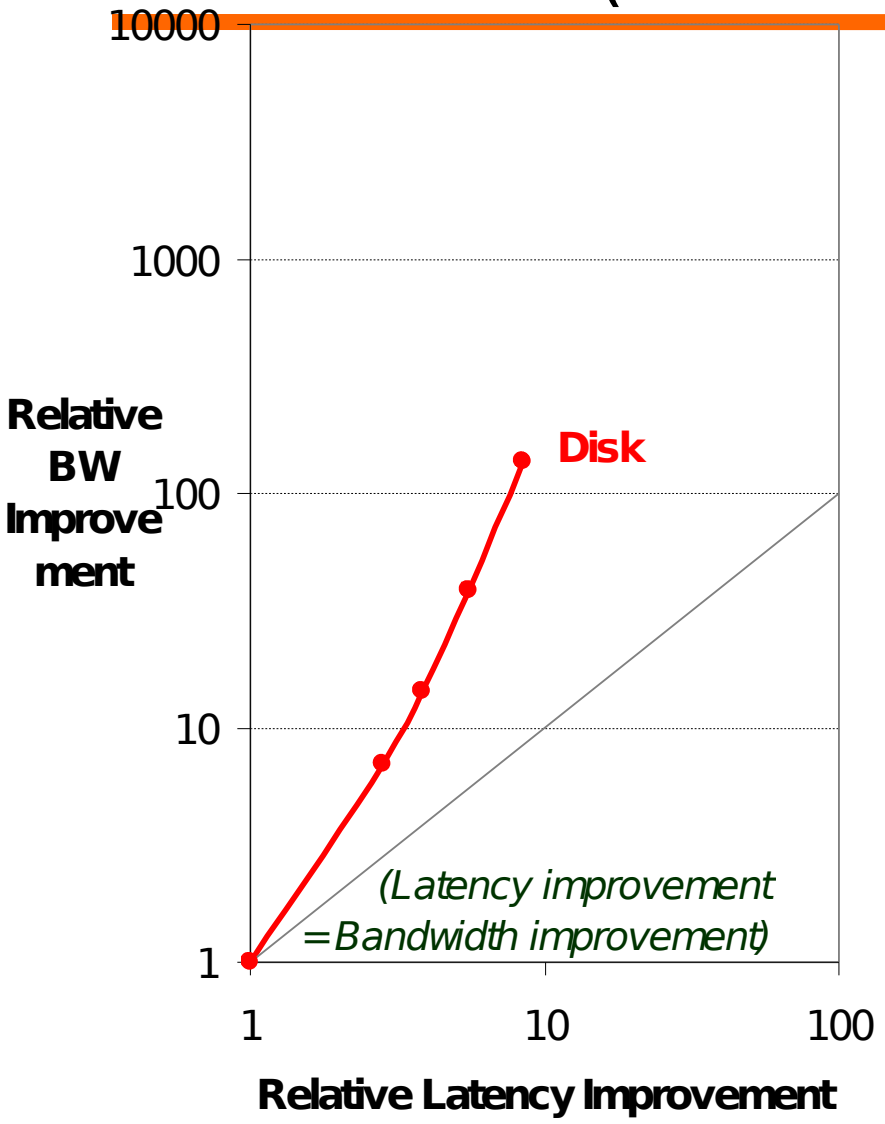
Disks: Archaic vs. Modern

- CDC Wren I, 1983
- 3600 RPM
- 0.03 GBytes capacity
- Tracks/Inch: 800
- Bits/Inch: 9550
- Three 5.25” platters

- Bandwidth:
  0.6 MBytes/sec
- Latency: 48.3 ms
- Cache: none

- Seagate 373453, 2003
- 15000 RPM (4X)
- 73.4 GBytes (2500X)
- TPI: 64000 (80X)
- BPI: 533,000 (60X)
- Four 2.5” platters
  (in 3.5” form factor)
- Bandwidth:
  86 MBytes/sec (140X)
- Latency: 5.7 ms (8X)
- Cache: 8 MBytes

Latency Lags Bandwidth (for last ~20 years)



- Performance Milestones

- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

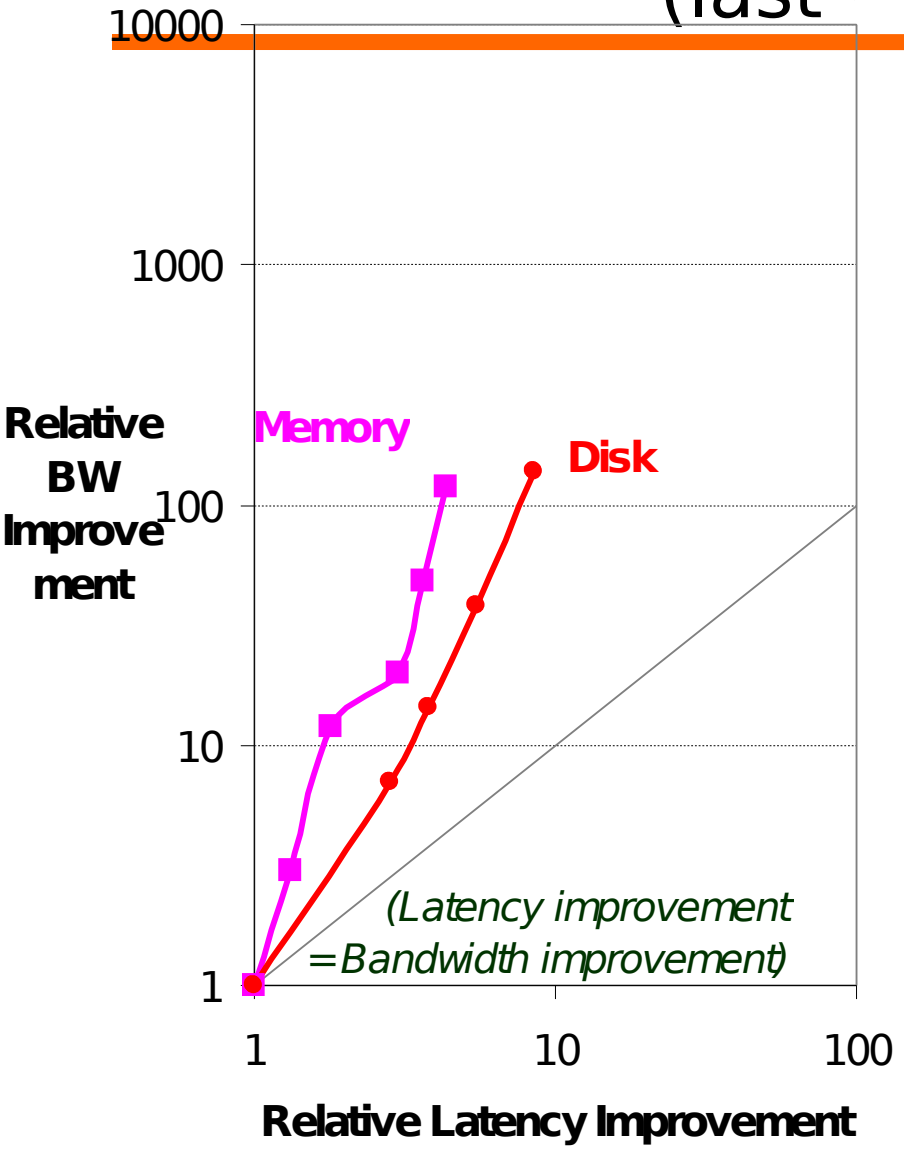
(latency = simple operation w/o contention
BW = best-case)

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)

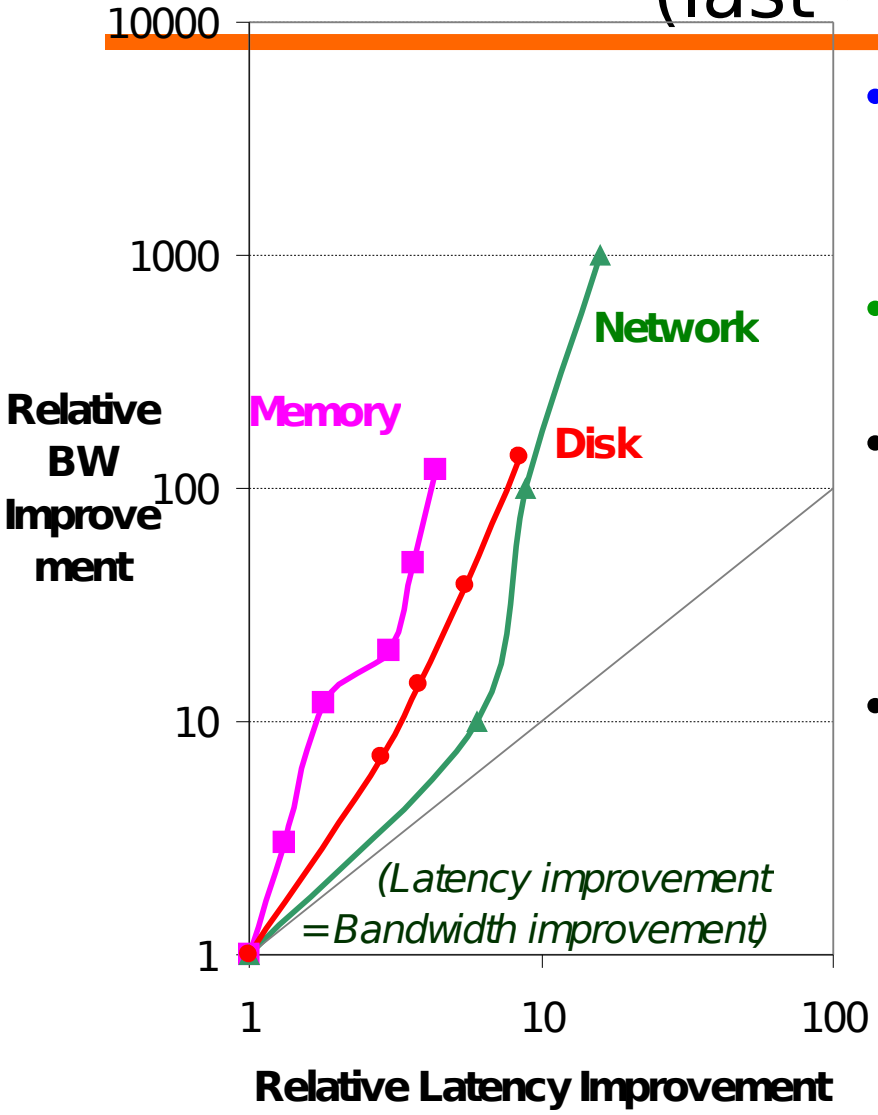
Latency Lags Bandwidth (last ~20 years)



- Performance Milestones
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM (4x,120x)
- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

(latency = simple operation w/o contention
BW = best-case)

Latency Lags Bandwidth (last ~20 years)



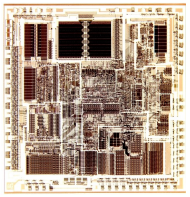
- Performance Milestones

- Ethernet: 10Mb, 100Mb, 1000Mb, 10000 Mb/s (15x,1000x)
- Memory Module: 16bit plain DRAM, Page Mode DRAM, 32b, 64b, SDRAM, DDR SDRAM (4x,120x)
- Disk: 3600, 5400, 7200, 10000, 15000 RPM (8x, 143x)

(latency = simple operation w/o contention
BW = best-case)

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 translate to RISC, Superpipelined (22 stage), Out-of-Order execution
- On-chip 96KB Data cache, 8KB Instr. Trace cache, 256KB L2 cache

6 Reasons Latency Lags Bandwidth

1. Moore's Law helps BW more than latency
   - Faster transistors, more transistors, more pins help bandwidth
     - MPU Transistors: 0.130 vs. 42 M xtors (300X)
     - DRAM Transistors: 0.064 vs. 256 M xtors (4000X)
     - MPU Pins: 68 vs. 423 pins (6X)
     - DRAM Pins: 16 vs. 66 pins (4X)
   - Smaller, faster transistors but communicate over (relatively) longer lines: limits latency
     - Feature size: 1.5 to 3 vs. 0.18 micron (8X,17X)
     - MPU Die Size: 35 vs. 204 mm2 (ratio sqrt ⇒ 2X)
     - DRAM Die Size: 47 vs. 217 mm2 (ratio sqrt ⇒ 2X)

6 Reasons Latency Lags Bandwidth (cont'd)

2. Distance limits latency

- Size of DRAM block → long bit and word lines → most of DRAM access time
- Speed of light and computers on network

3. Bandwidth easier to sell ("bigger=better")

- E.g., 10 Gbits/s Ethernet ("10 Gig") vs. 10 msec latency Ethernet
- 4400 MB/s DIMM ("PC4400") vs. 50 ns latency
- Even if just marketing, customers now trained
- Since bandwidth sells, more resources thrown at bandwidth, which further tips the balance

6 Reasons Latency Lags Bandwidth (cont'd)

4. Latency helps BW, but not vice versa

- Spinning disk faster improves both bandwidth and rotational latency
  - 3600 RPM → 15000 RPM = 4.2X
  - Average rotational latency: 8.3 ms ⇒ 2.0 ms
    - Things being equal, also helps BW
- Lower DRAM latency → more accesses/second (higher bandwidth)
- Higher linear density helps disk BW (and capacity), but not disk latency
  - 9,550 BPI → 533,000 BPI → 60X in BW

6 Reasons Latency Lags Bandwidth (cont'd)

5. Bandwidth hurts latency
   - Queues help bandwidth, hurt latency (Queuing Theory)
   - Adding chips to widen a memory module increases bandwidth but higher fan-out on address lines may increase latency
6. Operating System overhead hurts latency more than bandwidth
   - Long messages amortize overhead; overhead bigger part of short messages

Summary of Technology Trends

- For disk, LAN, memory, and microprocessor, bandwidth improves by square of latency improvement
  - In time that bandwidth doubles, latency improves by no more than 1.2X to 1.4X

- Lag probably even larger in real systems, as bandwidth gains multiplied by replicated components
  - Multiple processors in cluster or even in chip
  - Multiple disks in disk array
  - Multiple memory modules in large memory
  - Simultaneous communication in switched LAN

Technology Scaling Trends

- CPU Transistor density – 60% per year
- CPU Transistor speed – 15% per year
- DRAM density – 60% per year
- DRAM speed – 3% per year

- On-chip wire speed – decreasing relative to transistors
- Off-chip pin bandwidth – increasing, but slowly
- Power – approaching costs limits
  - $P = CV^2f + I_{leak}V$

- All of these factors affect the end system architecture

Crossroads: Conventional Wisdom

- Old conventional wisdom:
  - Power is free
  - Transistors are expensive
- New conventional wisdom: "Power wall"
  - Power is expensive
  - Transistors are "free"
    (Can put more on chip than can afford to turn on)

Conventional Wisdom (cont'd)

- Old conventional wisdom:
  - Instruction-level parallelism gives performance advances
    - Compilers
    - Innovation
      - Out-of-order execution
      - Speculation
      - Very long instruction words (VLIW)
- New conventional wisdom: "ILP wall"
  - Law of diminishing returns on more HW for ILP

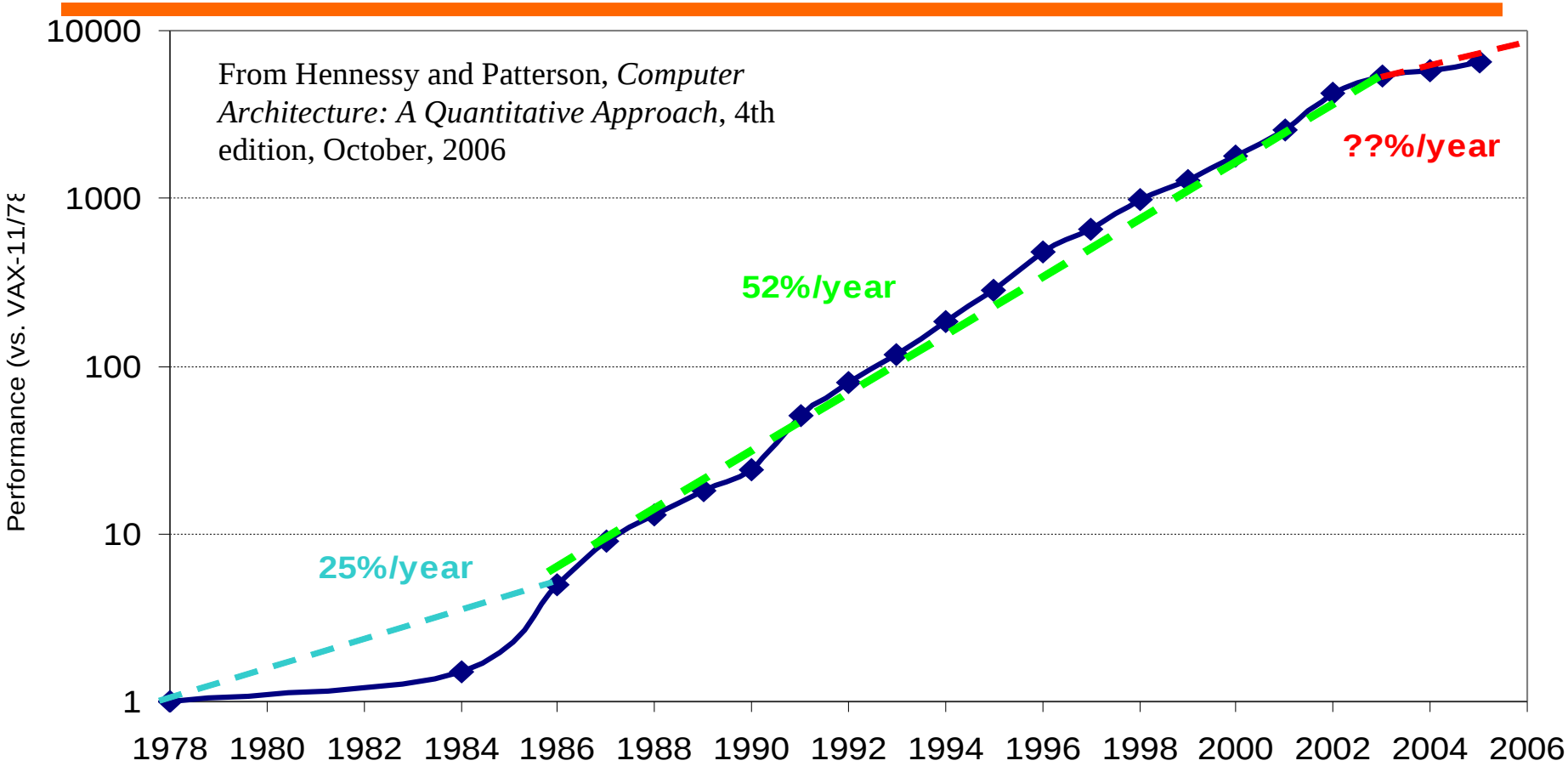
Conventional Wisdom (cont'd)

- Old conventional wisdom:
  - Multiplies are slow
  - Memory access is fast
- New conventional wisdom: "Memory wall"
  - Memory slow
    (200+ clock cycles to DRAM memory)
  - Multiplies are fast
    (4 clocks, pipelined)

Conventional Wisdom (cont'd)

- Old conventional wisdom:
  - Uniprocessor performance doubles every 1.5 yrs
- New conventional wisdom:
  - Power Wall + ILP Wall + Memory Wall = Brick Wall

Crossroads: Uniprocessor Performance



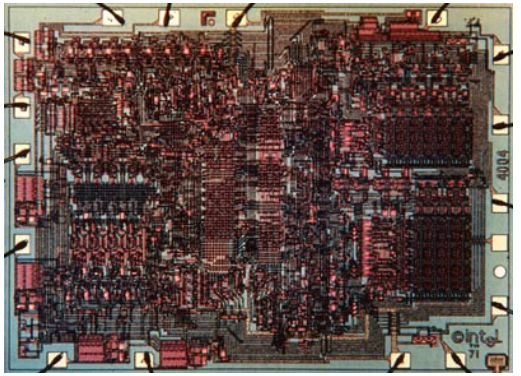
The End of Conventional Wisdom

- Moore Law' is slowing down
  - Uniprocessor performance now doubles every 5(+?) yrs
- Radical change in chip design: multiple "cores"

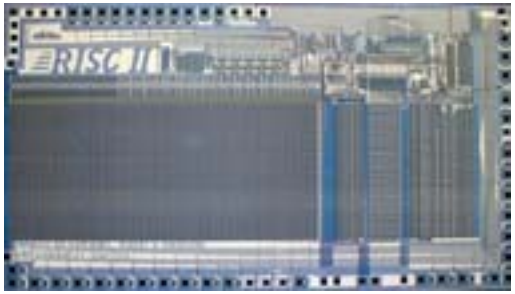
  (2X processors per chip every ~2 years)
- More but simpler processors
  - More power efficient

Change in Chip Design

- Intel 4004 (1971): 4-bit processor, 2312 transistors, 0.4 MHz, 10 micron PMOS, 11 mm^2 chip



- RISC II (1983): 32-bit, 5 stage pipeline, 40,760 transistors, 3 MHz, 3 micron NMOS, 60 mm^2 chip



- 125 mm^2 chip, 0.065 micron CMOS = 2312 RISC II+FPU+Icache+Dcache
  - RISC II shrinks to ~ 0.02 mm^2 at 65 nm
  - Caches via DRAM or SRAM
  - Proximity Communication via capacitive coupling at > 1 TB/s (Ivan Sutherland @ Sun / Berkeley)

→ **Core could be the new transistor**

Problems with Change

- Algorithms, Programming Languages, Compilers, Operating Systems, Architectures, Libraries, … not ready to supply thread-level or data-level parallelism for 1000 CPUs/chip (or even tens)

- Architectures are just not ready (yet) for 1000 CPUs/chip

What Computer Architecture Brings to Table

- Quantitative Principles of Design
  - Take Advantage of Parallelism
  - Principle of Locality
  - Focus on the Common Case
  - Amdahl’s Law
  - The Processor Performance Equation

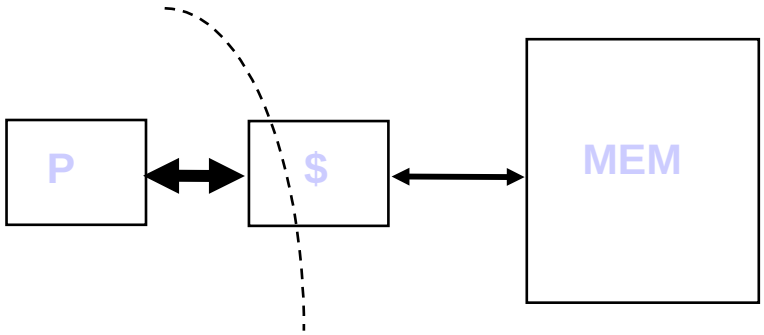
1) Taking Advantage of Parallelism

- Increasing throughput of server computer via multiple processors or multiple disks
- Detailed HW design
  - Carry-lookahead adders use parallelism to speed up computing sums from linear to logarithmic in number of bits per operand
  - Multiple memory banks searched in parallel in set-associative caches
- Pipelining: overlap instruction execution to reduce the total time to complete an instruction sequence.
  - Not every instruction depends on immediate predecessor ⇒ executing instructions completely/partially in parallel possible
  - Classic 5-stage pipeline:
    1) Instruction Fetch (Ifetch),
    2) Register Read (Reg),
    3) Execute (ALU),
    4) Data Memory Access (Dmem),
    5) Register Write (Reg)

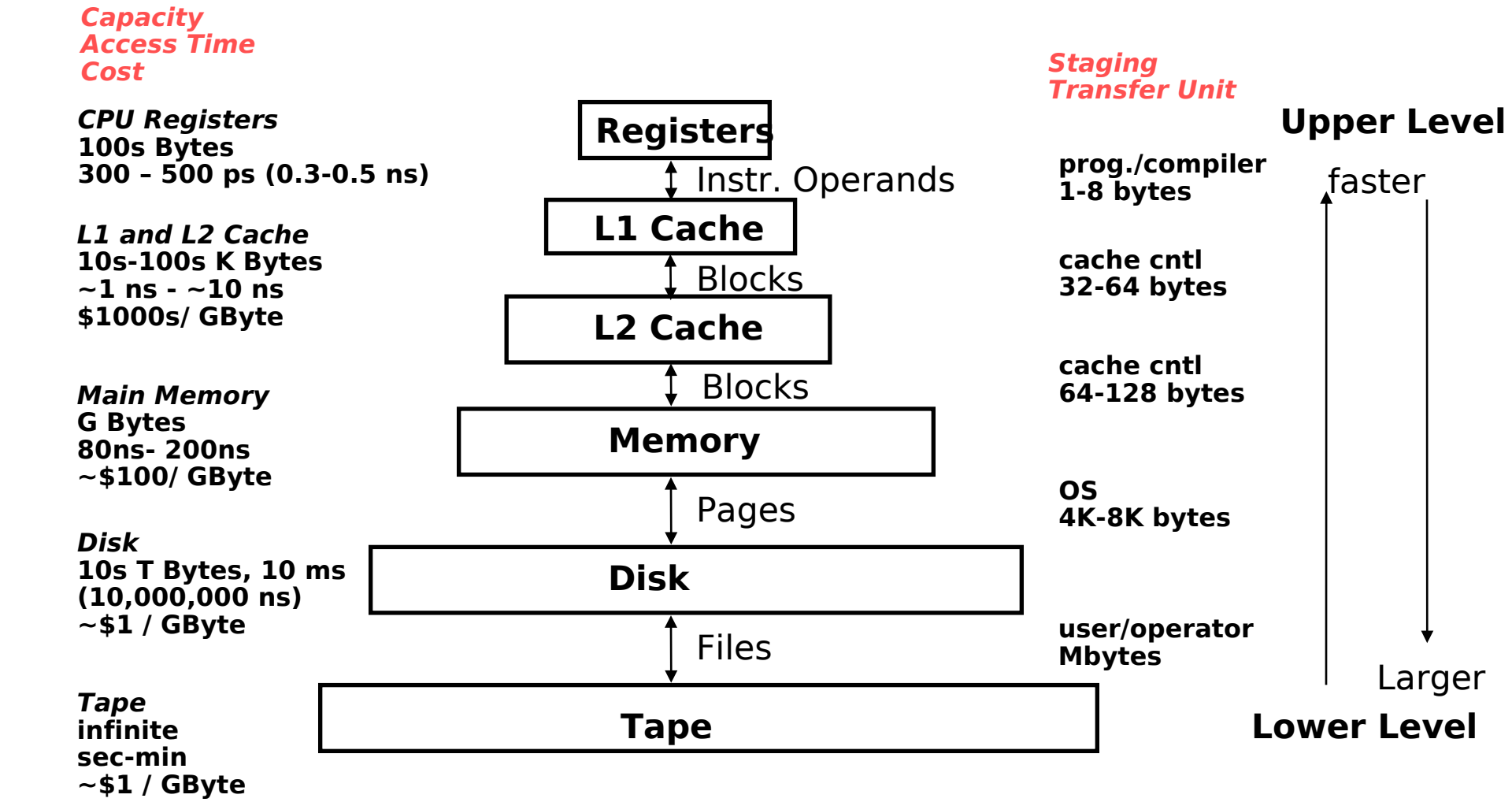
2) The Principle of Locality

- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.
- Two Different Types of Locality:
  - Temporal Locality (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)
  - Spatial Locality (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon

    (e.g., straight-line code, array access)
- Last 30 years, HW relied on locality for memory perf.



Levels of the Memory Hierarchy



3) Focus on the Common Case

- Common sense guides computer design
  - Since it's engineering, common sense is valuable
- In making a design trade-off, favor the frequent case over the infrequent case
  - E.g., Instruction fetch and decode unit used more frequently than multiplier, so optimize it 1st
  - E.g., If database server has 50 disks / processor, storage dependability dominates system dependability, so optimize it 1st
- Frequent case is often simpler and can be done faster than the infrequent case
  - E.g., overflow is rare when adding 2 numbers, so improve performance by optimizing more common case of no overflow
  - May slow down overflow, but overall performance improved by optimizing for the normal case
- What is frequent case and how much performance improved by making case faster => Amdahl's Law

4) Amdahl's Law

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

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

Best you could ever hope to do:

$$\text{Speedup}_{maximum} = \frac{1}{\left(1 - \text{Fraction}_{enhanced}\right)}$$

Amdahl's Law Example

- New CPU 10X faster
- 30% time waiting for I/O

$$\text{Speedup}_{\text{overall}} = \frac{1}{\left(1 - \text{Fraction}_{\text{enhanced}}\right) + \frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

$$= \frac{1}{\left(1 - 0.7\right) + \frac{0.7}{10}} = \frac{1}{0.37} = 2.7$$

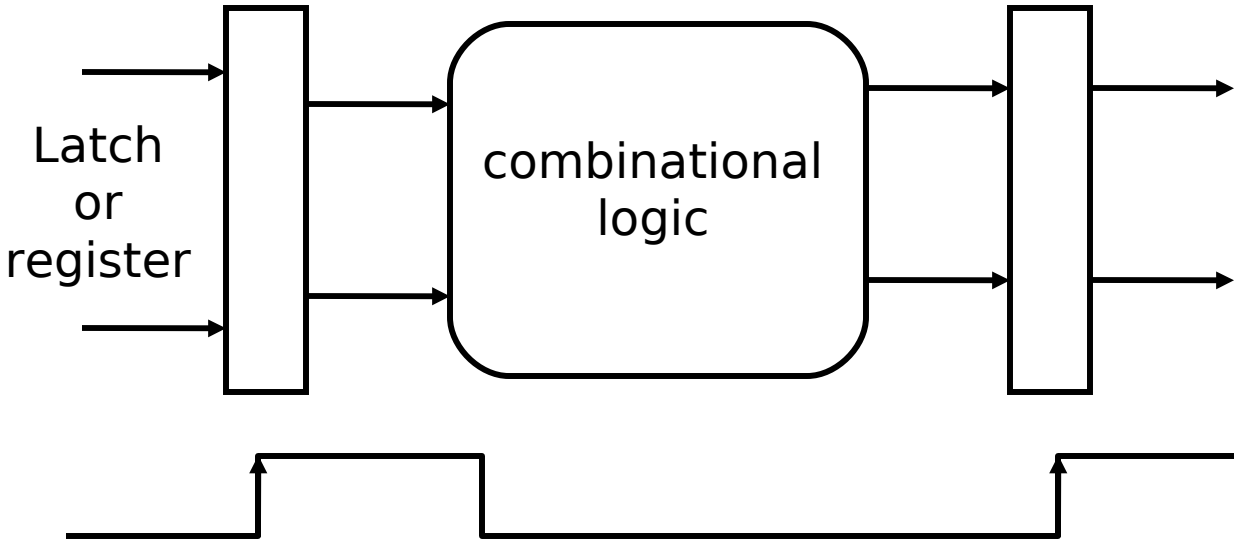
- Apparently, it's human nature to be attracted by 10X faster, vs. keeping in perspective it's just 2.7X faster

Amdahl’s Law in Reality

- John Ousterhout: “Why Aren’t Systems Getting Faster as Fast as Hardware?”, Usenix Summer Conference, 1990

  – we’re I/O-bound

What's a Clock Cycle?



- Old days: 10 levels of gates
- Today: determined by numerous time-of-flight issues + gate delays
  - Clock propagation, wire lengths, drivers

And in conclusion ...

- Computer Architecture >> instruction sets
- Computer Architecture skill sets are different
  - 5 Quantitative principles of design
  - Quantitative approach to design
  - Solid interfaces that really work
  - Technology tracking and anticipation
- Computer Science at the crossroads from sequential to parallel computing
  - Salvation requires innovation in many fields, including computer architecture