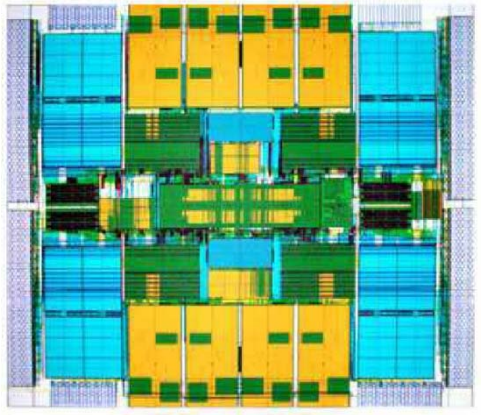
Performance



Performance

- What & How to Measure
- Why So?

Why is some hardware better than others for different programs?

What factors of system performance are hardware related? (e.g., Do we need a new machine, or a new operating system?)

How does the machine's instruction set affect performance?

Performance

- Measure, Report, and Summarize
- Make intelligent choices
- See through the marketing hype
- Key to understanding underlying organizational motivation

Computer Performance: TIME, TIME, TIME

- **Response Time (latency): How fast**
  - — How long does it take for my job to run?
  - — How long does it take to execute a job?
  - — How long must I wait for the database query?
- **Throughput: How much**
  - — How many jobs can the machine run at once?
  - — What is the average execution rate?
  - — How much work is getting done?

Time and Rate

Example

Plane	Wash. DC to Paris	Latency (Speed)	Passengers	Throughput (pmph)
Boeing 747	6.5 hours	610 mph	470	286,700
BAD/Sud Concodre	3 hours	1350 mph	132	178,200

Execution Time

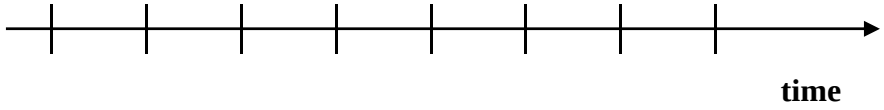
- Elapsed Time
  - counts everything *(disk and memory accesses, I/O,etc.)*
  - a useful number, but often not good for comparison purposes
- CPU time
  - doesn't count I/O or time spent running other programs
  - can be broken up into system time, and user time
- **Our focus: user CPU time**
  - time spent executing the lines of code that are "in" our program

Clock Cycles

- Instead of reporting execution time in seconds, we often use cycles

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

Clock “ticks” indicate when to start activities (one abstraction):



- cycle time = time between ticks = seconds per cycle
- clock rate (frequency) = cycles per second (1 Hz. = 1 cycle/sec)
  A 200 Mhz. clock has a cycle time

$$\frac{1}{200 \times 10^6} \times 10^9 = 5 \text{ nanoseconds}$$

Other Measures:

- MIPS
  - MIPS = instructions/10^{-6} sec.
- MFLOPS
  - MFLOPS = FP operations/10^{-6} sec
- CPU cycle time
- Memory/cache size

Too Simplistic to Measure Complex Interactions

Another Metric: Cycles Per Instruction

- **Cycles per Instruction (CPI)**

CPI = no. of cycles for an instruction execution on average

  - CPI is an average of all the instructions executed in a program
  - CPI is useful in comparing two different implementations of the same architecture
  - CPI is less dependent on circuit/device technology

IPC (instructions per cycle) = 1/CPI

Another Metric: Cycles per Instruction

- Example … a benchmark has 100 instructions:

  45 instructions are loads/stores (each takes 2 cycles)
  50 instructions are adds (each takes 1 cycle)
  5 instructions are square root (each takes 100 cycles)

  CPI = ((45 * 2) + (50 * 1) + (5 * 100)) / 100 = 640 / 100 = 6.4

- Can different instructions take different amounts of time?
  - Multiply vs. add
  - Memory accesses and cache misses
  - Multi-cycle implementation

Processor Performance

- **The "Iron Law" of processor performance:**

$$\text{Processor Performance} = \frac{\text{Time}}{\text{Program}}$$

$$= \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Time}}{\text{Cycle}}$$

(code size) **(CPI)** **(cycle time)**

	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X		
Inst. Set.	X	X	
Organization		X	X
Technology			X

Processor Performance

- To Improve Performance
  - Smaller Code Size (= Path Length)
  - Small CPI (Cycles Per Instruction)
  - Faster Cycle Time (Clock)

- Given the same ISA,
  - Code Size will remain (more or less) the same
    - Architecture/Organization will change CPI
    - Implementation will change Cycle TIme

Example

- Our favorite program runs in 10 seconds on computer A, which has a 400 Mhz. clock. We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program. What clock rate should we tell the designer to target?"

400x(10/6)x(12/10) = 800

A: n-cycles x (1/400) x 10-6sec
B: 1.2n-cycles x (1/C) x 10-6sec
A/B= 10/6; 10/6 = C/(1.2x400); C=800

CPI Example

- Suppose we have two implementations of the same instruction set architecture (ISA).
- For some program,
  Machine A has a clock cycle time of 10 ns. and a CPI of 2.0
  Machine B has a clock cycle time of 20 ns. and a CPI of 1.2

  What machine is faster for this program, and by how much?

  10x2.0=20 vs. 20x1.2=24

- *If two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will always be identical?*

  #of instructions (or None)

of Instructions Example

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).

  The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
  The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

  Which sequence will be faster? How much?
  What is the CPI for each sequence?

  I: 2x1 + 1x2 + 2x3 = 10cycles; cpi = 10/5 = 2
  II: 4x1 + 1x2 + 1x1 = 7cycles; cpi = 7/6 = 1.16

MIPS example

- Two different compilers are being tested for a 100 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.
  The first compiler's code uses 5 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.
  The second compiler's code uses 10 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.
- Which sequence will be faster according to MIPS?
- Which sequence will be faster according to execution time?

I: 5Mx1 + 1Mx2 + 1Mx3 = 10Mcycles for 7M instr; 0.7instr/cycle; 70MIPS
II: 10Mx1 + 1Mx2 + 1Mx3 = 15Mcycles for 12M instr; 0.8instr.cycle: 80MIPS

Q: how to measure CPI?

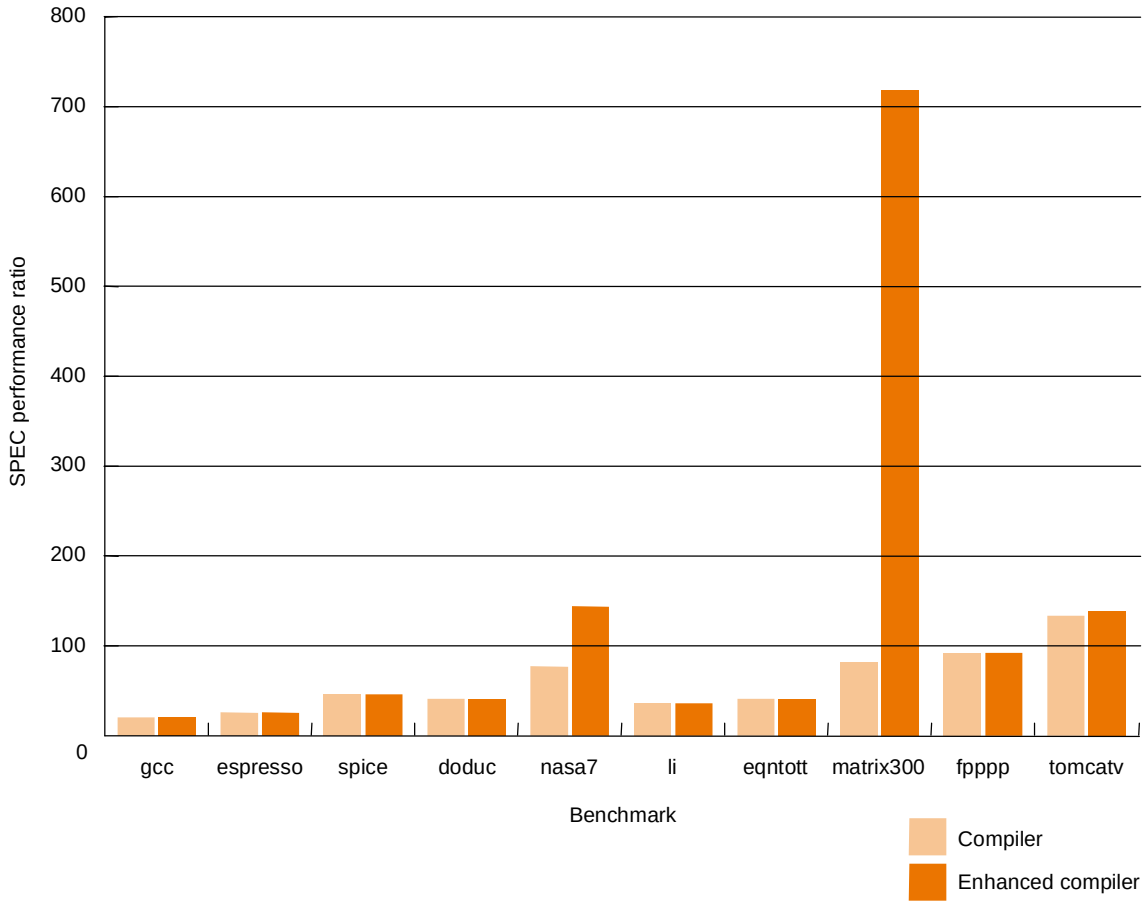
So, assuming CPI is a good metric to measure processor performance, how do we obtain CPI from programs?

Benchmarks

- Performance best determined by running a real application
  - Use programs typical of expected workload
  - Or, typical of expected class of applications
    e.g., compilers/editors, scientific applications, graphics, etc.
- Small benchmarks
  - nice for architects and designers
  - easy to standardize
  - can be abused
- SPEC (System Performance Evaluation Cooperative)
  - companies have agreed on a set of real program and inputs
  - can still be abused
  - valuable indicator of performance (and compiler technology)

SPEC ‘89

- Compiler “enhancements” and performance



SPEC Benchmarks

NOTE Measuring Relative Performance

Normalized performance is handy for scaling performance

e.g., *X* times faster than yours vs X times faster than reference

- Geometric mean: instead of adding, multiply the execution times.

$$\sqrt[N]{\prod_{i=1}^{N} T_i}$$

SPEC Benchmarks

NOTE Measuring Relative Performance

- Geometric Mean of execution time wrt. base machine
  - Relative performance
  - Less impact from exception case

$$\frac{G(X_i)}{G(Y_i)} = G\left(\frac{X_i}{Y_i}\right)$$

SPEC Benchmarks

NOTE Measuring Relative Performance

Example:
Suppose a machine X and VAX 11/780 run four programs, A, B,C, D, with execution times shown below

	A	B	C	D
VAX	2	2	1	1
X	.2	.4	.01	2

SPEC Benchmarks

NOTE Measuring Relative Performance

Example(cont.):

	A	B	C	D	Arith	G
VAX	2	2	1	1	1.5	1.42
X	.2	.4	.01	2	0.65	0.2
VAX/X	10	5	100	0.5	28	7

then SPEC mark for the machine X would be $(10*5*100*0.5)^{1/4} \sim 7.07$

If used arithmetic mean,
(10+5+100+0.5)/4 = 28.875

g1 taking relative performance with average arithmetic mean is 2.3, while with geometric mean is ~7. Note that aritmetic mean of relative performance for each case is quite different, 28.875.

ghlee, 2007-09-03

SPEC Benchmarks

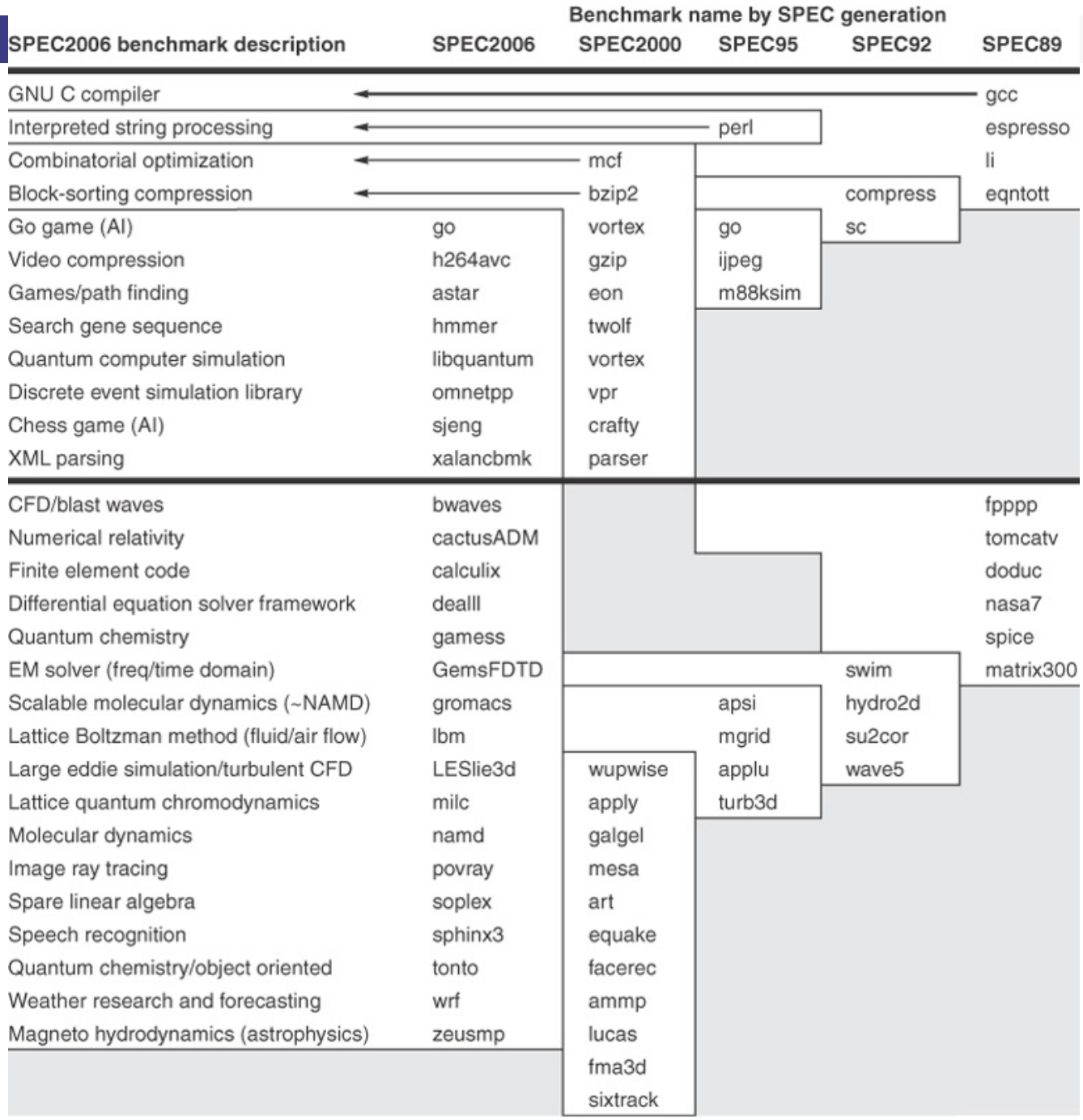
NOTE Measuring Relative Performance

- **Base machine** :
  - VAX 11/780 for SPEC'92
  - SUN Sparc10 for SPEC'95

SPEC CPU 2000

Cint: 11 C & 1C++, Cfp: 14)

- Gzip — compression using Lempel-Ziv
- Vpr — FPGA placement and routing
- Gcc — GNU C compiler
- Mcf — public transit scheduling optimization
- Crafty — Chess
- Parser — Syntactic English parser
- Eon (c++) — Graphics with probabilistic ray tracing
- Perlmbk — perl with four input scripts
- Gap — math (group theory) package
- Vortex — OO database system
- bzip2 — block compression
- twolf — Timberwolf: VLSI placement



SPEC2006 benchmark description	Benchmark name by SPEC generation				
	SPEC2006	SPEC2000	SPEC95	SPEC92	SPEC89
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			

SPEC CPU 2000

NOTE:

- More comprehensive report on system parameters and execution time (input data size).
- For SpecRate, Reference machine is
  Sun Ultra5_10 with a 300MHz processor
  (at least 256MB Memory and about two days to run).
- SPEC CPU 2006 is the newest**.**

Very popular for Microprocessor Performance but always behind the machine design practice. Visit **http://www.spec.org/cpu2006/** for FAQs on SPEC CPU benchmarks

Amdahl's Law

Execution Time After Improvement =

Execution Time Unaffected +(Execution Time Affected / Amount of Improvement)

- Example:

  "Suppose a program runs in 100 seconds on a machine, with multiply responsible for 80 seconds of this time. How much do we have to improve the speed of multiplication if we want the program to run 4 times faster?"

  How about making it 5 times faster?

  80 to 5; 16 times for 4 times faster;80 to 0 for 5 times faster

- ***Principle: Make the common case fast***
  - ***What to do after making the common cast fast?***

Some Quantitative Principles

Amdahl's Law

Let T_1 and T_p be execution time with single processor and *p* processors, respectively, for a program. Then,

$$\textbf{\textit{Speedup}}\ \ S_p = T_1 / T_p$$

Suppose a program has two parts, one part that can be improved and the other part that can't be improved. Let *f* be the fraction of the part that can be improved with speedup of S_{fp}

$$T_p = (1-f)T_1 + f\,T_1/S_{fp} \text{ and } S_p = 1/((1-f) + f/S_{fp})$$

Some Quantitative Principles (Cont.)

Amdahl's Law (Cont.)

Suppose f=90% and $S_{fp} = p$, Then

P	S_p
2	1.8
10	5.3
100	9.2
1000	9.9
10000	9.99

Example

- Suppose we enhance a machine making all floating-point instructions run five times faster. If the execution time of some benchmark before the floating-point enhancement is 10 seconds, what will the speedup be if half of the 10 seconds is spent executing floating-point instructions?

  5 + 5(1/5) = 6sec; Sp=10/6

- We are looking for a benchmark to show off the new floating-point unit described above, and want the overall benchmark to show a speedup of 3. One benchmark we are considering runs for 100 seconds with the old floating-point hardware. How much of the execution time would floating-point instructions have to account for in this program in order to yield our desired speedup on this benchmark?

  1-x + x/5 = 1/3; (4/5)x=2/3; x=5/6

- What to run for measurement
  - Benchmark programs
    - For CPU, SPEC is popular

- Other Issues
  - Power
  - Cost

- Architectural Guide
  - Any design principles to follow
    - Amdahl's Law
    - Locality

    Read Text Sec. 1.2 p.11~12 Eight Great Ideas (?)

Power Consumption

$Pd \sim CV^2f$ for switching 1-0-1 or 0-1-0

Where

C	is a load capacitance
V	is a supply voltage
f	is a clock frequency

- **Dynamic**

$\sim 1/2 CV^2af$

a is a fraction between 0 and 1 indicating how often clock ticks lead to switching activity on average.

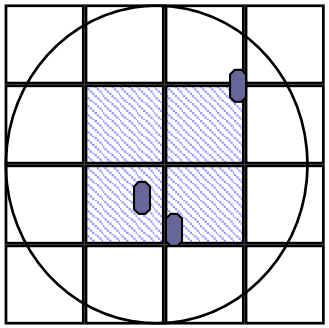
Lower and lower driving voltage
Less switching
Less leakage

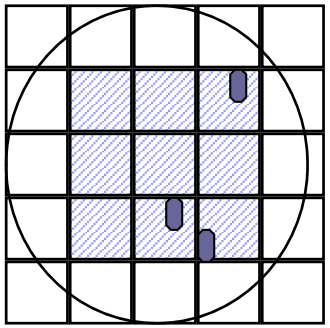
COST

IC cost: (Die cost + Testing cost + Packaging)*
Final test yield

Die cost: Wafer cost/(Dies per Wafer*Die yield)

1/4





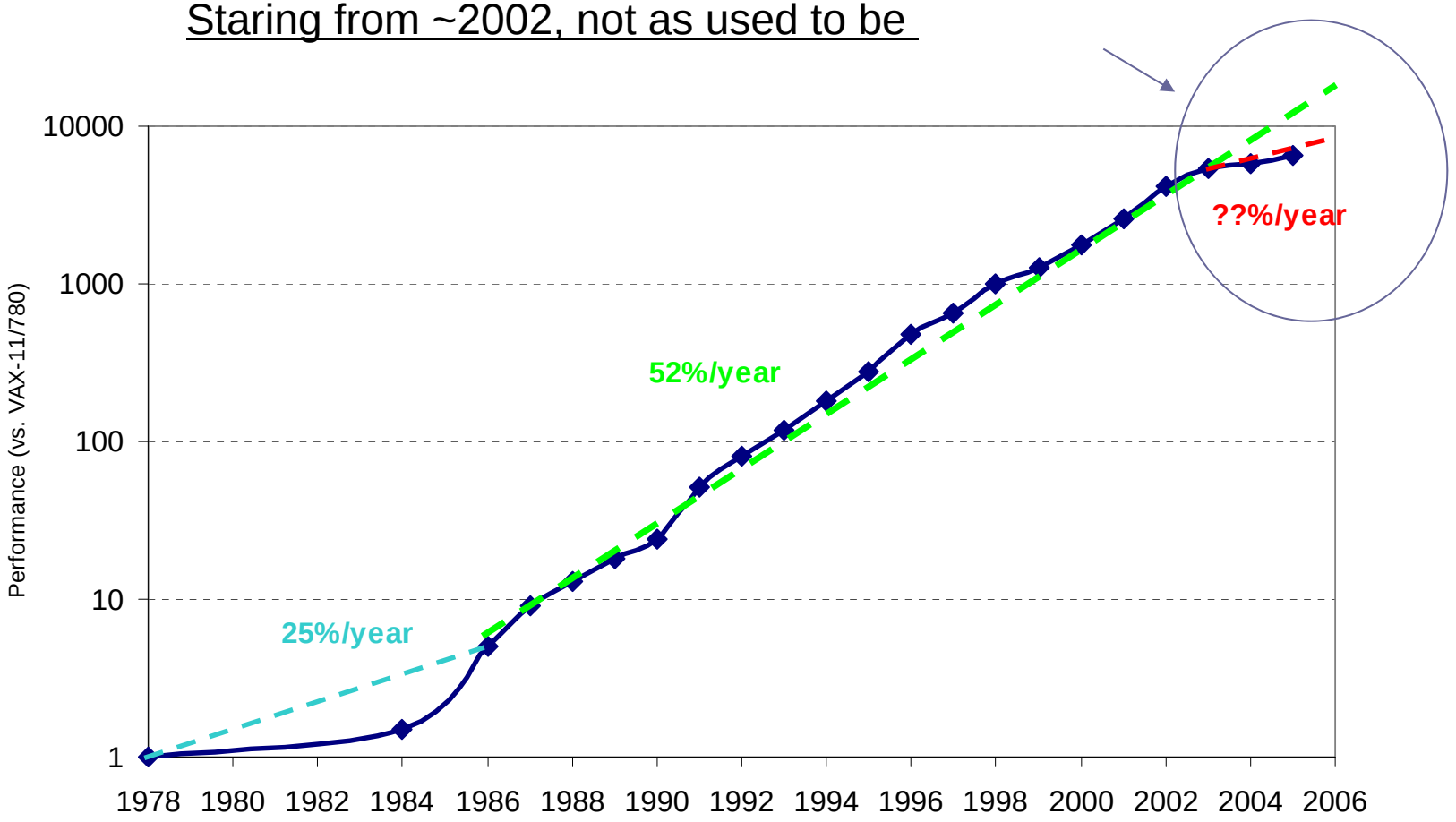
6/9

COST "Estimating IC Manufacturing Costs," by Linley Gwennap, *Microprocessor Report*, August 2, 1993.

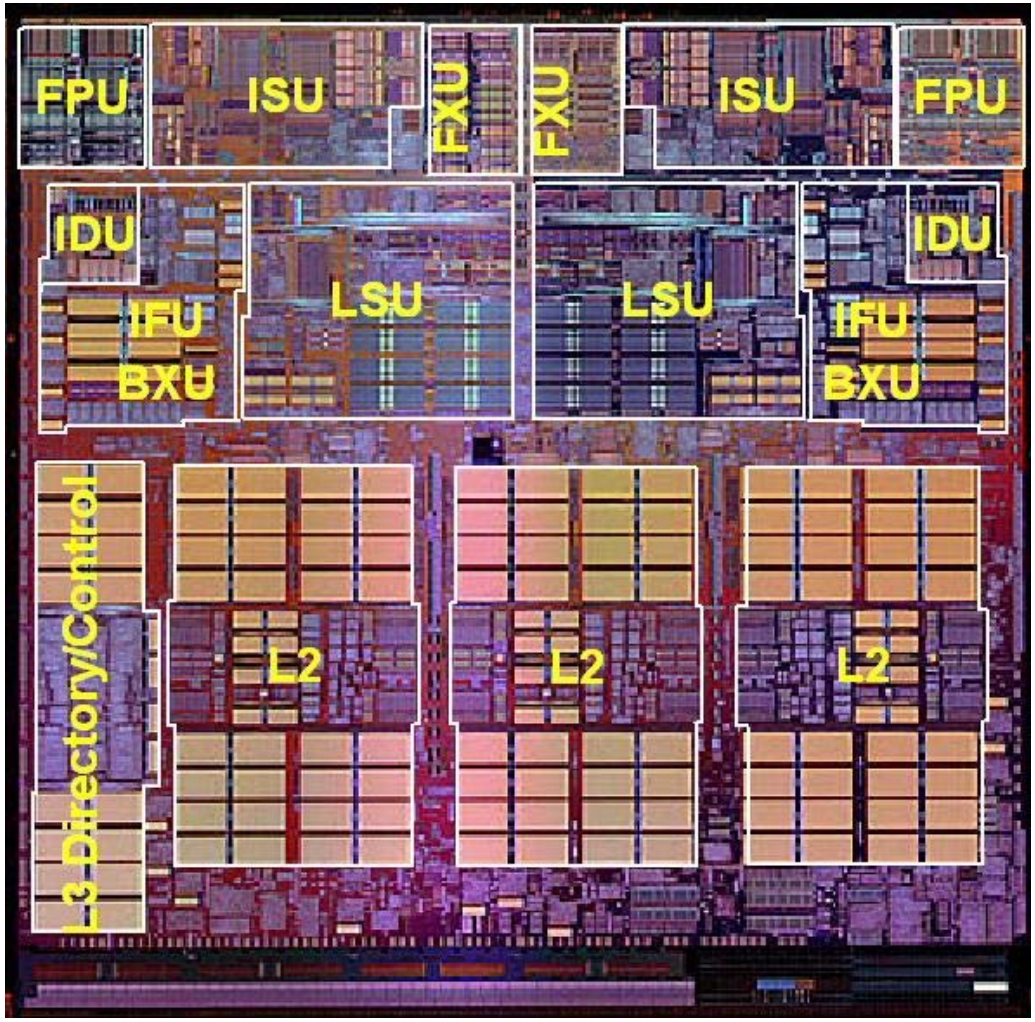
Chip	Metal layers	Line width	Wafer cost	Defect /cm^2	Area mm^2	Dies/ wafer	Yield	Die Cost
386DX	2	0.90	$900	1.0	43	360	71%	$4
486DX	3	0.80	$1200	1.0	81	181	54%	$12
Power PC601	4	0.80	$1700	1.3	121	115	28%	$53
HP PA7100	3	0.80	$1300	1.0	196	66	27%	$73
DEC Alpha	3	0.70	$1500	1.2	234	53	19%	$149
Super SPARC	3	0.70	$1700	1.6	256	48	13%	$272
Pentium	3	0.80	$1500	1.5	296	40	9%	$417

Processor performance improvement

Staring from ~2002, not as used to be

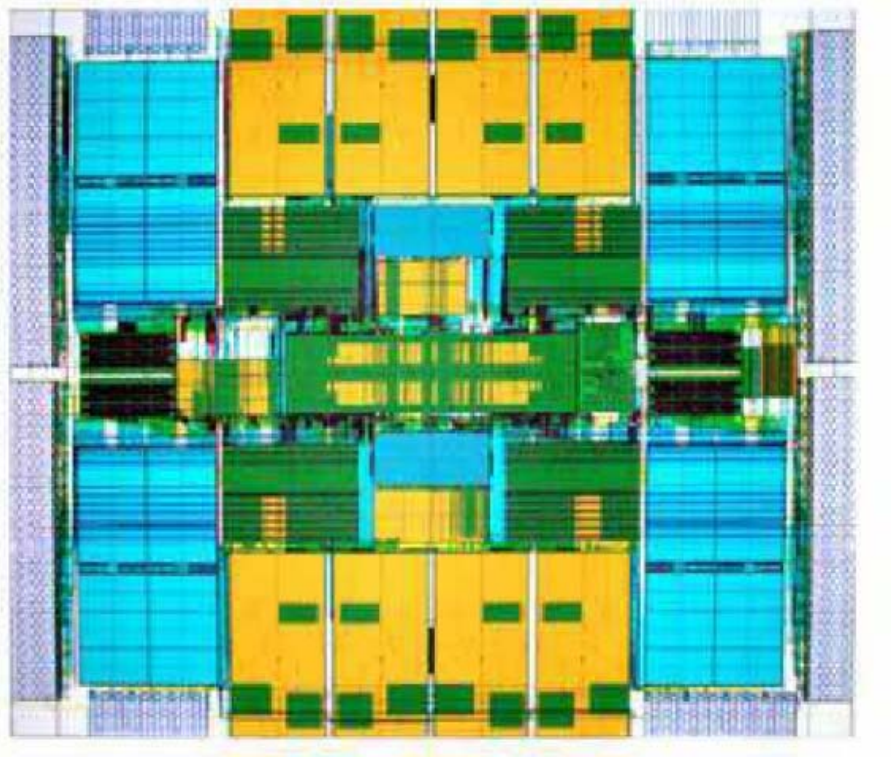


IBM dual core Power 4



1.5MB L-2 cache
170M Trs with
30M Trs/processor core

Sun's Niagara



Some Qualitative Principles

Locality of Reference

- Memory reference
  - Temporal
  - Spatial

cf. Communication Locality

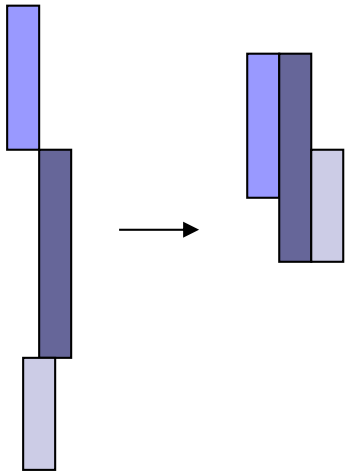
Some Qualitative Principles

Parallelism

Space-time trade-off

Shorter Critical Path

More Resources



for high throughput,

Pipelining has been the choice at less overhead

Remember

- For a given architecture performance increases come from:
  - increases in clock rate (without adverse CPI affects)
  - improvements in processor organization that lower CPI
  - compiler enhancements that lower CPI and/or instruction count
- Pitfall: expecting improvement in one aspect of a machine's performance to affect the total performance
  - Improvement in one factor may lead to degradation in others

CPU Time = I x CPI x cycle time
I: instruction count or path length
CPI: cycle per instruction

Walls

- Objectives
  - High Performance – Supercomputing
  - Dependability – Trusted Computing
- Technical stumbling block
  - Memory wall
  - ILP Wall
  - Power Wall