Tradeoffs between cost and performance

- You can’t do a design without knowing the cost and performance targets.
- Some designs emphasize performance:
  - High performance, low-volume designs
  - Supercomputers from Cray
- Some designs emphasize low cost:
  - High-volume, low general-purpose performance
  - Nintendo, gadgets, set-top boxes
- Balanced cost/performance design involve the most difficult tradeoffs: workstations, PCs
- CMOS VLSI microprocessors have closed the performance gap between the low end and the high end:
  - RISC, Superscalar
How to measure performance

- Program execution time is the only valid measure: seconds for a particular program
- Be careful defining execution time:
  - "wall clock": elapsed time as seen by the user
    - includes disk, OS overhead, competition with other jobs
  - Total CPU time: processor time for your job, excluding I/O.
    - User CPU time, plus
    - OS CPU time
  - User CPU time

Analyzing CPU performance

\[
\text{CPUtime} = \text{CPU clockcycles per program} \times \text{clockcycle time}
\]

\[
\text{CPI} = \frac{\text{CPU clockcycles per program}}{\text{Instruction count}}
\]

\[
\text{CPUtime} = \text{instruction count} \times \text{CPI} \times \text{clockcycle time}
\]

\[
\text{CPUtime} = \frac{\text{Seconds}}{\text{program}} \times \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Clockcycle}}
\]
Performance based on instruction types

- Another way to compute CPU time, based on \( n \) different instruction types with different execution times:

\[
\text{CPU time} = \sum_{i=1}^{n} (\text{CPI}_i \times \text{IC}_i) \times \text{Clock cycle time}
\]

CPI is now:

\[
\text{CPI} = \sum_{i=1}^{n} \left( \text{CPI}_i \times \frac{\text{IC}_i}{\text{Instruction count}_i} \right)
\]

where

\[
\text{Frequency of Instruction}_i = \frac{\text{IC}_i}{\text{Instruction count}_i}
\]

CPU Performance Example

- Assume a simple load/store machine with the following instruction frequencies:

<table>
<thead>
<tr>
<th>Instruction type</th>
<th>Frequency</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>loads</td>
<td>25%</td>
<td>2</td>
</tr>
<tr>
<td>stores</td>
<td>15%</td>
<td>2</td>
</tr>
<tr>
<td>branches</td>
<td>20%</td>
<td>2</td>
</tr>
<tr>
<td>ALU</td>
<td>40%</td>
<td>1</td>
</tr>
</tbody>
</table>

- Conditional branches currently use simple test against zero.
  (BEQZ Rn,loc; BNEZ Rn,loc)
- Should we add complex comparison/branch combination?
  (BEQ Rn,Rm,loc; BNE Rn,Rm,loc)
  » 25% of branches can use the complex scheme and save the preceding ALU instruction
  » The CPU cycle time of the new machine has to be 10% longer
  » Will this increase CPU performance?
CPU Performance Example:
Solution #1

- Old CPU performance
  \[ \text{CPI}_{\text{old}} = 0.25 \times x_2 + 0.15 \times x_2 + 0.20 \times x_2 + 0.40 \times x_1 = 1.6 \]
  \[ \text{CPUtime}_{\text{old}} = 1.6 \times \text{IC}_{\text{old}} \times \text{CCT}_{\text{old}} \]

- New CPU performance
  \[ \text{CPI}_{\text{new}} = \frac{0.25 \times x_2 + 0.15 \times x_2 + 0.20 \times x_2 + (0.40 - 0.25 \times 0.2) \times 1}{1 - 0.25 \times 0.2} = 1.63 \]
  \[ \text{IC}_{\text{new}} = 0.95 \times \text{IC}_{\text{old}} \]
  \[ \text{CCT}_{\text{new}} = 1.1 \times \text{CCT}_{\text{old}} \]
  \[ \text{CPUtime}_{\text{new}} = \text{IC}_{\text{new}} \times \text{CPI}_{\text{new}} \times \text{CCT}_{\text{new}} = 1.63 \times (0.95 \times \text{IC}_{\text{old}}) \times (1.1 \times \text{CCT}_{\text{old}}) = 1.71 \times \text{IC}_{\text{old}} \times \text{CCT}_{\text{old}} \]

- Answer: Don’t do it!

CPU Performance Example:
Solution #2

- If you don’t need to know \( \text{CPI}_{\text{new}} \):

- Old CPU performance
  \[ \text{CPI}_{\text{old}} = 0.25 \times x_2 + 0.15 \times x_2 + 0.20 \times x_2 + 0.40 \times x_1 = 1.6 \]
  \[ \text{CPUtime}_{\text{old}} = 1.6 \times \text{IC}_{\text{old}} \times \text{CCT}_{\text{old}} \]

- New cycles needed for same mix:
  \[ \text{CPUtime}_{\text{new}} = \text{CPI}_{\text{new}} \times \text{IC}_{\text{old}} \times \text{CCT}_{\text{new}} = 1.55 \times \text{IC}_{\text{old}} \times (1.1 \times \text{CCT}_{\text{old}}) = 1.71 \times \text{IC}_{\text{old}} \times \text{CCT}_{\text{old}} \]
Megaflops
-- Floating point intensive applications

- MFLOPS = \( \frac{\text{FP op count}}{\text{Execution time} \times 10^6} \)

- Applies only to floating-point (non-integer) problems
- Assumes that the same number and type of floating-point operations are executed on all machines
- Livermore Loop benchmarks use “normalized” MFLOPS based on operations in the source code weighted as follows:

<table>
<thead>
<tr>
<th>Original FP Operation</th>
<th>Normalized FP ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>add, sub, cmp, mult</td>
<td>1</td>
</tr>
<tr>
<td>divide, sqrt</td>
<td>4</td>
</tr>
<tr>
<td>exp, sin, etc.</td>
<td>8</td>
</tr>
</tbody>
</table>

Evaluating Performance:
What Workload should We Use?

- Use real programs
  - CAD, text processing, business aps, scientific aps
  - Specify input and output
  - Hard to do, but the most realistic
- Kernels
  - Small key code pieces where programs spend most of their time
  - Examples: Livermore loops, LINPACK
  - Can be analyzed manually
- Micro Benchmarks
  - Measure one performance dimension
    - cache bandwidth
    - main memory bandwidth
    - procedure call overhead
    - FP performance
- Synthetic Benchmarks
  - Match the average frequency of operations and operands of a large set of real programs
  - Examples: Whetstone, Dhrystone
Problems with Benchmarking

- Toy and synthetic benchmarks don’t load the memory system realistically. (Cache effects abound!)
- Once benchmarks become standardized they are susceptible to specialized compiler tricks and hardware trends
  - Benchmarks are useful for about three years
  - SPEC89, SPEC92, SPEC95, SPEC98
- Hard to evaluate real benchmarks:
  - Machine not built yet, simulators too slow
  - Benchmarks not ported
  - Compilers not ready
- Benchmark performance is a composition of hardware and software (program, input, compiler, OS) performance, which must all be specified
- Benchmark performance for a representative sample of tasks must be mathematically combined. How?

Summarizing Performance

- For times, use the weighted arithmetic mean
  \[
  \text{AVG\_TIME} = \frac{\sum_{i=1}^{n} \text{time}_i \times \text{weight}_i}{\sum_{i=1}^{n} \text{weight}_i}
  \]
- For rates (MIPS, MFLOPS), use the weighted harmonic mean
  \[
  \text{AVG\_RATE} = \frac{1}{\sum_{i=1}^{n} \frac{\text{weight}_i}{\text{rate}_i}}
  \]
- For ratios (normalized measures, like relative MIPS), use the geometric mean
  \[
  \text{AVG\_RATIO} = \sqrt[n]{\prod_{i=1}^{n} \text{ratio}_i}
  \]
- Does not follow principle of performance measurement!
### Performance Summary Example

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>MFLOP</th>
<th>Computer A (seconds)</th>
<th>Computer B (seconds)</th>
<th>Computer C (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>100</td>
<td>1.0</td>
<td>10.0</td>
<td>20.0</td>
</tr>
<tr>
<td>Program 2</td>
<td>100</td>
<td>1000.0</td>
<td>100.0</td>
<td>20.0</td>
</tr>
<tr>
<td>Total Time</td>
<td>100</td>
<td>1001.0</td>
<td>110.0</td>
<td>40.0</td>
</tr>
<tr>
<td>Arith. Mean</td>
<td></td>
<td>500.5</td>
<td>55.0</td>
<td>20.0</td>
</tr>
<tr>
<td>Geom. Mean (A)</td>
<td></td>
<td>1.0</td>
<td>1.0</td>
<td>0.6</td>
</tr>
</tbody>
</table>

### Performance Summary Example contd.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>MFLOP</th>
<th>Computer A (MFLOPS)</th>
<th>Computer B (MFLOPS)</th>
<th>Computer C (MFLOPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program 1</td>
<td>100</td>
<td>100.0</td>
<td>10.0</td>
<td>5.0</td>
</tr>
<tr>
<td>Program 2</td>
<td>100</td>
<td>0.1</td>
<td>1.0</td>
<td>5.0</td>
</tr>
<tr>
<td>Arith. Mean</td>
<td></td>
<td>50.1</td>
<td>5.5</td>
<td>5.0</td>
</tr>
<tr>
<td>Geom. Mean (A)</td>
<td></td>
<td>3.2</td>
<td>3.2</td>
<td>5.0</td>
</tr>
<tr>
<td>Harm. Mean</td>
<td></td>
<td>0.2</td>
<td>1.8</td>
<td>5.0</td>
</tr>
</tbody>
</table>
Estimating Performance

- **Statistics**
  - E.g. Instruction mix

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVE</td>
<td>39%</td>
</tr>
<tr>
<td>BR</td>
<td>20%</td>
</tr>
<tr>
<td>LOAD</td>
<td>20%</td>
</tr>
<tr>
<td>STORE</td>
<td>10%</td>
</tr>
<tr>
<td>ALU</td>
<td>11%</td>
</tr>
</tbody>
</table>

- **Trace-driven simulation**
  - Replay recorded accesses
    - Cache, branch, register

- **Execution-driven simulation at many levels**
  - ISA, cycle accurate, RTL, gate, circuit
  - Trade accuracy for simulation rate

- **Analysis**
  - Closed form equations
  - E.g., queuing theory

Cost

-- Just as important as performance

- You will rarely be able to ignore cost in your design
- Some important cost issues:
  - Base decisions on projected costs at the time the project will ship
  - Consider cost changes due to the production learning curve
  - Larger volumes lead to lower costs: 10% reduction for doubling in volume
  - Design cost counts too: dollars as well as time
    - Verification of complex designs takes a lot of time
    - Minimize complexity to save both!
- **Technology trends**
  - Higher density
  - Larger dies
  - Implies: IC cost is an increasing fraction of the controllable cost for low-cost designs
Designing for future costs:

IC Chip Cost

\[
\text{ICCost} = \frac{\text{Cost of die} + \text{Cost of testing} + \text{Cost of packaging}}{\text{Final test yield}}
\]

\[
\text{Cost of die} = \frac{\text{Cost of wafer}}{\text{Dies per wafer} \times \text{die yield}}
\]

\[
\text{Dies per wafer} = \frac{\pi(D/2)^2}{A} - \frac{\pi D}{\sqrt{2A}} - \text{test dies per wafer}
\]

where \( A = \) die area, \( D = \) wafer diameter

- Some typical numbers:
  - A 20 cm (8") wafer costs $2500 and holds
    - 269 1 cm\(^2\) dies (Pentium)
    - 79 3 cm\(^2\) dies (Original Pentium Pro)
IC Yield and Die Cost

- Yield = fraction of dies that work and meet specs

\[
\text{Die yield} = \frac{\text{Wafer yield} \times (1 + \text{Defects per unit area} \times A)}{}
\]

- Moral: Die cost is nonlinear function of the die size!

<table>
<thead>
<tr>
<th>Wafer Cost</th>
<th>2,500.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wafer Diameter</td>
<td>200 mm</td>
</tr>
<tr>
<td>Wafer Area</td>
<td>31416 mm²</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chip Size</th>
<th>Die Area</th>
<th>Die/Wafer Yield</th>
<th>Good Die</th>
<th>Cost/Die</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30971</td>
<td>1.00</td>
<td>30971</td>
<td>$0.08</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>1.00</td>
<td>1145</td>
<td>$2.18</td>
</tr>
<tr>
<td>2.5</td>
<td>56.25</td>
<td>0.83</td>
<td>414</td>
<td>$6.04</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>0.63</td>
<td>170</td>
<td>$14.71</td>
</tr>
<tr>
<td>7.5</td>
<td>225</td>
<td>0.36</td>
<td>39</td>
<td>$64.10</td>
</tr>
<tr>
<td>15</td>
<td>306.25</td>
<td>0.26</td>
<td>21</td>
<td>$119.05</td>
</tr>
</tbody>
</table>

- Other Chip Costs and Real Examples

- IC Packaging and testing cost
  - \( \frac{\text{cost/die}}{\text{yield}} = \frac{\text{cost/hour} \times \text{test time}}{} \)
  - could be $10-$20 or more for complex chips

- IC Packaging: depends on die size and number of pins

- Chip Costs
### Microprocessor Cost & Performance

<table>
<thead>
<tr>
<th></th>
<th>Digital 2164</th>
<th>IBM P2SC</th>
<th>PowerPC 750</th>
<th>PowerPC 604e</th>
<th>Sun Ultra-2</th>
<th>Sun Ultra-2i</th>
<th>HP PA-8200</th>
<th>HP PA-7300LC</th>
<th>MIPS R10000</th>
<th>Intel Pentium II</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clock rate</strong></td>
<td>600 MHz</td>
<td>160 MHz</td>
<td>300 MHz</td>
<td>350 MHz</td>
<td>360 MHz</td>
<td>300 MHz</td>
<td>236 MHz</td>
<td>160 MHz</td>
<td>250 MHz</td>
<td>400 MHz</td>
</tr>
<tr>
<td><strong>Cache size</strong></td>
<td>8K/8K/16K</td>
<td>32K/64K/128K</td>
<td>32K/128K</td>
<td>32K/128K</td>
<td>16K/16K</td>
<td>16K/16K</td>
<td>None</td>
<td>None</td>
<td>32K/32K</td>
<td>16K/16K</td>
</tr>
<tr>
<td><strong>Pipeline stages</strong></td>
<td>4 stages</td>
<td>6 stages</td>
<td>3 stages</td>
<td>4 stages</td>
<td>4 stages</td>
<td>4 stages</td>
<td>4 stages</td>
<td>4 stages</td>
<td>4 stages</td>
<td>3 stages</td>
</tr>
<tr>
<td><strong>Out of order renaming</strong></td>
<td>6 loads</td>
<td>5 intrs</td>
<td>5 intrs</td>
<td>4 intrs</td>
<td>4 intrs</td>
<td>4 intrs</td>
<td>4 intrs</td>
<td>4 intrs</td>
<td>4 intrs</td>
<td>3 x86 intrs</td>
</tr>
<tr>
<td><strong>BHT entries</strong></td>
<td>2K = 2-bit</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td><strong>TLB entries</strong></td>
<td>48/1/64 D</td>
<td>64/64 D</td>
<td>64/64 D</td>
<td>64/64 D</td>
<td>64/64 D</td>
<td>64/64 D</td>
<td>64/64 D</td>
<td>64/64 D</td>
<td>64/64 D</td>
<td>64/64 D</td>
</tr>
<tr>
<td><strong>Memory size/Package</strong></td>
<td>400 MB(*)</td>
<td>256 MB(*)</td>
<td>256 MB(*)</td>
<td>256 MB(*)</td>
<td>256 MB(*)</td>
<td>256 MB(*)</td>
<td>256 MB(*)</td>
<td>256 MB(*)</td>
<td>256 MB(*)</td>
<td>256 MB(*)</td>
</tr>
<tr>
<td><strong>IC process size</strong></td>
<td>0.35µm/4M</td>
<td>0.25µm/5M</td>
<td>0.25µm/5M</td>
<td>0.25µm/5M</td>
<td>0.29µm/4M</td>
<td>0.29µm/4M</td>
<td>0.29µm/4M</td>
<td>0.29µm/4M</td>
<td>0.29µm/4M</td>
<td>0.25µm/4M</td>
</tr>
<tr>
<td><strong>Die size</strong></td>
<td>209 mm²/255mm²</td>
<td>70 mm²/47 mm²</td>
<td>126 mm²/76 mm²</td>
<td>150 mm²/100 mm²</td>
<td>345 mm²/299 mm²</td>
<td>197 mm²/131 mm²</td>
<td>345 mm²/299 mm²</td>
<td>197 mm²/131 mm²</td>
<td>345 mm²/299 mm²</td>
<td>197 mm²/131 mm²</td>
</tr>
<tr>
<td><strong>Transistors</strong></td>
<td>9.3 million</td>
<td>16 million</td>
<td>6.4 million</td>
<td>5.1 million</td>
<td>3.8 million</td>
<td>4.1 million</td>
<td>3.9 million</td>
<td>2.9 million</td>
<td>6.8 million</td>
<td>7.5 million</td>
</tr>
<tr>
<td><strong>Est. mfg. cost</strong></td>
<td>$125</td>
<td>$380</td>
<td>$40</td>
<td>$50</td>
<td>$50</td>
<td>$50</td>
<td>$95</td>
<td>$190</td>
<td>$195</td>
<td>$225</td>
</tr>
<tr>
<td><strong>Power (max)</strong></td>
<td>19.2/26.6</td>
<td>30 W</td>
<td>5 W</td>
<td>5 W</td>
<td>7 W</td>
<td>20 W</td>
<td>38 W</td>
<td>&gt;40 W</td>
<td>15 W</td>
<td>16 W</td>
</tr>
<tr>
<td><strong>Availability</strong></td>
<td>2Q97</td>
<td>3Q96</td>
<td>3Q97</td>
<td>3Q97</td>
<td>3Q97</td>
<td>3Q97</td>
<td>1Q98</td>
<td>3Q98</td>
<td>3Q98</td>
<td>1Q98</td>
</tr>
<tr>
<td><strong>SRAM price</strong></td>
<td>Net public</td>
<td>Net public</td>
<td>$495</td>
<td>$495</td>
<td>$495</td>
<td>$3,695</td>
<td>$470</td>
<td>Not public</td>
<td>Net public</td>
<td>$722</td>
</tr>
</tbody>
</table>

Microprocessor Report, June 22, 1998