2050, Earth...
Using Simulator to Analyze Power Consumption

Song Li
Zizhen Yao
Outline

- Introduction to Power dissipation problems and solutions.
- Power Modeling and Simulation (Previous research)
- Our work
- Future plan
What you need for a cool chip
Cooling techniques
Source of Power Dissipation
Power Saving Techniques

- Voltage Scaling: ultimate limitation
  - The voltage to tell data from noise
  - The threshold of a P-N node: (Si ~ 0.7V, Ge ~ 0.3V)

- Logic:
  - Logic layout optimization – put functional units closer to each other if they need to communicate a lot

- Conditional clocking
  - Certain functional units are idle when not in use
  - i.e. FPU
  - Wakeup time is critical in performance impact
Power Saving Examples

- Alpha:
  - Lower VDD (2.2V for 21264 vs. 3.3 for 21064 and 21164)
  - Conditional clocking
  - Low power bus
Power Saving Examples Cont’d

- PIII vs. Crusoe
Power Saving Examples Cont’d

- Crusoe:
  - Scalable VDD
  - The power of code morphing
    - No architectural backward-compatibility requirement
    - So that instructions can be optimized to achieve better performance / power ratio

- Die size:
  - Crusoe: TM5600: 88mm\(^2\) TM5800: 55mm\(^2\)
  - Alpha: 21164: 299mm\(^2\) 21264: 314mm\(^2\)
Power Saving Examples Cont’d
Power Analysis Tool

- Circuit level
  - PowerMill, QuickPower
  - Can be applied only at the late stage of architecture design.
  - Accurate, but expensive.

- Architecture level
  - Wattch
  - Can analyze high level architecture design
  - Tradeoff accuracy for efficiency.
Power Model

- Computation of power
  \[ P = CV^2 f \]

- Computation of \( C \) (capacitance)
  - Diffusion capacitance (transient state).
  - Gate capacitance (gate charge and discharge).
  - Wire capacitance (wire to substrate, wire to wire)
Figure 1 The Basic 21264 Pipeline
Types of logic Structure

- Array Structure (RAM)
  - Data and instruction caches
  - Register file
  - Branch predictor
  - Renaming table
- Fully Associative Content-Addressable Memories (CAM)
  - Wakeup logic, TLB
  - Dependency check
    - Renaming
    - Load store queue:
- Combinatorial Logic (ALU) and Clock
Capacitance Modeling (RAM)

Decoder, wordline drive, bitline discharge, and output drive.
Wordline and Bitline

\[ C_{\text{wordline}} = C_{\text{diff}}(\text{WordlineDriver}) + C_{\text{gate}}(\text{CellAccess}) \times \text{NumBitlines} + C_{\text{metal}} \times \text{WordlineLength} \]

\[ C_{\text{bitline}} = C_{\text{diff}}(\text{Precharge}) + C_{\text{gate}}(\text{CellAccess}) \times \text{NumWordlines} + C_{\text{metal}} \times \text{BitlineLength} \]
CAM

\[ C_{\text{tagline}} = C_{\text{gate}}(\text{CompareEn}) \times \text{NumberTags} + C_{\text{diff}}(\text{CompareDriver}) + C_{\text{metal}} \times \text{TaglineLength} \]

\[ C_{\text{matchline}} = 2 \times C_{\text{diff}}(\text{CompareEn}) \times \text{TagSize} + C_{\text{diff}}(\text{MatchPreCharge}) + C_{\text{diff}}(\text{MatchOR}) + C_{\text{metal}} \times \text{MatchlineLength} \]
Wattch

- Using Power model to compute the power consumption at each access of each device
- Extend execution-driven simulator to keep the access counter of each device during execution
- Conditional Clocking is modeled
  1. All or nothing clock gating
  2. Linear clock gating
  3. Linear scale with port or unit usage, plus unused units dissipates 10% of their maximum power.
Verification of Wattch

<table>
<thead>
<tr>
<th>Hardware Structure</th>
<th>Intel Data</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Fetch</td>
<td>22.2%</td>
<td>21.0%</td>
</tr>
<tr>
<td>Register Alias Table</td>
<td>6.3%</td>
<td>4.9%</td>
</tr>
<tr>
<td>Reservation Stations</td>
<td>7.9%</td>
<td>8.9%</td>
</tr>
<tr>
<td>Reorder Buffer</td>
<td>11.1%</td>
<td>11.9%</td>
</tr>
<tr>
<td>Integer Exec. Unit</td>
<td>14.3%</td>
<td>14.6%</td>
</tr>
<tr>
<td>Data Cache Unit</td>
<td>11.1%</td>
<td>11.5%</td>
</tr>
<tr>
<td>Memory Order Buffer</td>
<td>6.3%</td>
<td>4.7%</td>
</tr>
<tr>
<td>Floating Point Exec. Unit</td>
<td>7.9%</td>
<td>8.0%</td>
</tr>
<tr>
<td>Global Clock</td>
<td>7.9%</td>
<td>10.5%</td>
</tr>
<tr>
<td>Branch Target Buffer</td>
<td>4.7%</td>
<td>3.8%</td>
</tr>
</tbody>
</table>

Table 4: Comparison between Modeled and Reported Power Breakdowns for the Pentium Pro®.

<table>
<thead>
<tr>
<th>Hardware Structure</th>
<th>Alpha 21264</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caches</td>
<td>16.1%</td>
<td>15.3%</td>
</tr>
<tr>
<td>Out-of-Order Issue Logic</td>
<td>19.3%</td>
<td>20.6%</td>
</tr>
<tr>
<td>Memory Management Unit</td>
<td>8.6%</td>
<td>11.7%</td>
</tr>
<tr>
<td>Floating Point Exec. Unit</td>
<td>10.8%</td>
<td>11.0%</td>
</tr>
<tr>
<td>Integer Exec. Unit</td>
<td>10.8%</td>
<td>11.0%</td>
</tr>
<tr>
<td>Total Clock Power</td>
<td>34.4%</td>
<td>30.4%</td>
</tr>
</tbody>
</table>

Table 5: Comparison between Modeled and Reported Power Breakdowns for the Alpha 21264.
Our work

- Port Wattch to Sim-Alpha
- Use Wattch-Alpha to analyze power consumption of architecture designs
Diffence between SimpleScalar and Sim-Alpha (1)

Sim-Alpha  (MIPS R10000, DEC 21264)

SimpleScalar  (Intel Pentium Pro, PowerPC)
Diffence between SimpleScalar and Sim-Alpha (2)

- There is no slot stage in SimpleScalar
- Instruction fetch
  - Sim-alpha: fetches by block
  - SimpleScalar: fetches each instruction.
- Line and set prediction.
  - Each fetch block of four instructions includes a line and set prediction, which indicates where is the next block of instructions.
  - Combines the speed advantage of a direct-mapped cache with the lower miss ratio of a two-way set-associative cache.
  - Only simulated by Sim-alpha simulates this design.
- Memory system
  - Bus and MSHR are not simulated in SimpleScalar.
Our Work: different from Wattch

- Icache access:
  - In Wattch Icache access is per instruction
  - In SimAlpha Icache access is per block

- ALU access
  - More instructions in Alpha’s ISA
  - i.e. ITOF: IALU
  - i.e. Addressing mode: DISP, RR

- Rename access
  - Wattch only considers the power for reservation station allocation.
  - Rename table-lookup for operands is neglected.
Experiments to perform

- How is the associativity of CAM device effect the performance and power consumption
  - Issue Queue
  - Set associative cache
- Microarchitecture that reduce the associativity
  - Fifo-based design for issue queue
Fifo-based microarchitecture

- Motivation
  - Decrease associativity of logic device without performance degradation.
  - Exploit the natural dependences among the instructions.
- The issue window is replaced by several fifo queues. Each queue
  - Contains dependent instructions
  - Inorder issue
- Selection instructions from head of queues.
- Result broadcast to head of queues
Fifo-based microarchitecture

Further Reduction of associativity by clustering