Energy Efficient Hybrid Wakeup Logic

Michael Huang, Jose Renau*, and Josep Torrellas*
University of Rochester,
University of Illinois at Urbana-Champaign*
Introduction

- Instruction window: major energy consumer
  - Wakeup logic checks many entries

- Schemes to reduce unnecessary operations:
  - Gating empty entries*
  - Gating ready entries*
  Comparisons reduced to 25%

- Exploiting a different fact:
  - Dependence tree is narrow (small fan-out)
  Comparisons reduced to 1%

[*] Folegnani and Gonzalez, ISCA 2001
Rationale: Number of Dependents

Many instructions have 1 or less dependent
- Not counting instructions without destination register (branch, store)
- Close-by dependent is what matters – even fewer
# Experimental Setup

## Processor

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1GHz 6-issue OOO core</strong></td>
<td><strong>BR: 1 unit (8 cycle min penalty)</strong></td>
</tr>
<tr>
<td><strong>I-window: 96 entries</strong></td>
<td><strong>RAS: 32 entries</strong></td>
</tr>
<tr>
<td><strong>Int/FP/LS units: 5/4/2</strong></td>
<td><strong>BTB: 4 way 2048 entries</strong></td>
</tr>
<tr>
<td><strong>LSQ: 16/16</strong></td>
<td><strong>Predictor: GAp (10,8)</strong></td>
</tr>
</tbody>
</table>

## Caches

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L1: 32KB 2-way LRU 32B</strong></td>
<td><strong>FSB: 128-bit 333MHz</strong></td>
</tr>
<tr>
<td></td>
<td><strong>DRAM: 2-channel Rambus</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Bandwidth: 3.2GB/s</strong></td>
</tr>
<tr>
<td><strong>1,3ns (occupancy/latency)</strong></td>
<td><strong>Memory RT: 108ns</strong></td>
</tr>
</tbody>
</table>

## Bus & Memory

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>L2: 512MB 8-way PLRU 64B</strong></td>
<td><strong>Bandwidth: 3.2GB/s</strong></td>
</tr>
<tr>
<td><strong>4,12ns (occupancy/latency)</strong></td>
<td><strong>Memory RT: 108ns</strong></td>
</tr>
<tr>
<td><strong>I-Cache: 32KB 2-way LRU</strong></td>
<td></td>
</tr>
</tbody>
</table>
Number of Close-by Dependents

- Distant dependents do not need wake-up
Tracking the Dependents

- Renaming logic already does some of the job
Slowdown of Indexing-Only Wakeup

Stalling due to lack of indexing pointer results in large slowdown
Waking Up More Than One Instruction

- Revert to broadcast – a hybrid approach
Reduction in Tag Comparison

- **Hybrid-Plain:** \( \frac{1}{N_{\text{entry}}} \cdot P_{\text{single}} + 1 \cdot P_{\text{more}} \)
- **Hybrid-Snoop:** \( \frac{1}{N_{\text{entry}}} \cdot P_{\text{single}} + \frac{N_{\text{snoop}}}{N_{\text{entry}}} \cdot P_{\text{more}} \)

- \( N_{\text{entry}} \): # of I-win entries
- \( N_{\text{snoop}} \): # of entries with snoop bit set
- \( P_{\text{single}} \): Prob(instr. having 1 CBD)
- \( P_{\text{more}} \): Prob(instr. having more CBDs)
Other Issues in Paper

- Handling branch misprediction
  - Checkpoint the PIE field
  - Tolerating dangling pointers in the DIE field

- Applicability to other window organizations
  - Distributed windows: attach window ID
  - Compacting windows: not compatible

- Waking up two dependents: a special case

- Stalling several cycles before reverting to broadcast
  e.g. Stalling 1 cycle when DIE is not empty
  - reduces 45% broadcasts
  - 2.1% performance degradation
Summary

Reducing wakeup comparisons

- By exploiting small “fan-out” of producer instructions:
  - 91.3% instructions have 0 or 1 close-by dependent
- Using indexing to handle common case
- Large reductions:
  - 8.9 comparisons/instruction (hybrid-plain)
  - 0.8 comparisons/instruction (hybrid-snoop)
  - compare to 23.7 if gating ready and empty entries
Energy Efficient Hybrid Wakeup Logic

Michael Huang, Jose Renau*, and Josep Torrellas*
University of Rochester,
University of Illinois at Urbana-Champaign*
Michael.Huang@ece.rochester.edu