Energy Characterization of Hardware Data Prefetching

Y. Guo, S. Chheda, I. Koren, M. Krishna, C. Andras Moritz
Electrical & Computer Engineering
University of Massachusetts, Amherst
Jan 30th, 2004

Motivation

- Data Prefetching has been successful in hiding memory access latency.
- Different techniques have been proposed
  - Software: Mowry '84, Lipasti et al.'95, Luk & Mowry '96
  - Hardware: Smith '78, Baar '91, Roth et al.'98, Cooksey et al.'02.
- Power and energy consumption becomes more and more important in recent years.

- How does prefetching affect on-chip energy consumption?
  - The scope of my presentation today
  - On longer term we are interested in developing new energy-aware prefetching solutions.

How Does Prefetching Work?

- The Prefetch Engine decides which data (address) to be prefetched.
- No prefetching if data is already in L1 Cache.

Sources of Prefetching Energy

- Extra Tag-checks in L1 cache
  - When a prefetch hits in L1.
- Extra memory accesses to L2 Cache
  - Due to useless prefetches from L2 to L1.
- Extra off-chip memory accesses
  - When data cannot be found in the L2 Cache.
- Prefetching hardware: data (history table) and control logic.
Prefetching Techniques Used

- Prefetching-on-miss (POM) - basic technique
- Tagged Prefetching - A variation of POM.
- Stride Prefetching [Baer & Chen] – Effective on array accesses with regular strides
- Dependence-based Prefetching [Roth & Sohl] – Focuses on pointer-chasing relations
- Combined Stride and Pointer Prefetching [new] – Applied on general-purpose programs

Experimental Setup

- SimpleScalar
  - Implementation of prefetching techniques
  - Gather statistics which will be used for energy estimation.
- Energy Estimation for L1 & L2 cache accesses
  - Spice simulation with 100-nm BPTM technology
- Benchmark Suites
  - SPEC2000 – Array-intensive benchmarks
  - Olden – Pointer-intensive benchmarks

Cache Configuration & Power

<table>
<thead>
<tr>
<th>Parameter</th>
<th>L1</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>32KB</td>
<td>256KB</td>
</tr>
<tr>
<td>tag array</td>
<td>CAM-based</td>
<td>RAM-based</td>
</tr>
<tr>
<td>associativity</td>
<td>32-way</td>
<td>4-way</td>
</tr>
<tr>
<td>bank size</td>
<td>2KB</td>
<td>4KB</td>
</tr>
<tr>
<td># of banks</td>
<td>16</td>
<td>64</td>
</tr>
<tr>
<td>cache line</td>
<td>32B</td>
<td>64B</td>
</tr>
<tr>
<td>Power (mW)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_tag</td>
<td>6.5</td>
<td>6.27</td>
</tr>
<tr>
<td>P_read</td>
<td>9.5</td>
<td>100.52</td>
</tr>
<tr>
<td>P_write</td>
<td>10.3</td>
<td>118.62</td>
</tr>
<tr>
<td>P_leakage</td>
<td>3.1</td>
<td>23.0</td>
</tr>
<tr>
<td>P_reduced_leakage</td>
<td>0.62</td>
<td>1.15</td>
</tr>
</tbody>
</table>

Performance Speedup
Memory Traffic Increase

Dynamic Energy Consumption - L1
- L1 Cache Hit Energy increased significantly due to extra tag lookups.

Dynamic Energy Consumption – L2
- L2 Cache Hit Energy increased slightly in most situations.

Adding Leakage Energy for L1
- Leakage dominates the total energy consumption without leakage optimization.
Leakage Reduction Techniques

- Many leakage optimizations proposed: body biasing, dual $V_t$, VTCMOS, MTCMOS, asymmetric cells, etc
- E.g., leakage can be reduced by 7X for writes and 40X for reads in cells [Azizi et al ISLPED 2002, evaluated for 130-nm]
- We assume that cache leakage could/will be reduced by 80% with circuit techniques.

Energy-Delay Product

- Energy-delay product improves with prefetching in most cases.

Off-Chip Memory Access Energy

- Off-chip = 32X L1 Access Energy
- Off-chip = 128X L1 Access Energy
- Off-chip = 1024X L1 Access Energy

Conclusion

- Prefetching can be considered as an energy reduction technique as well, esp. in deep submicron tech. where leakage becomes dominate.
- Aggressive prefetching techniques increase L1 access energy significantly due to extra tag-checks.
  - We are working on a new technique to improve it.
- Effective prefetching techniques consistently improve energy-delay products (EDP) due to performance improvements.