Energy Efficient Data Access Techniques

Alen Bardizbanyan, Daniel Moreau, Magnus Själander*,
David Whalley**, Per Larsson-Edefors

Chalmers University of Technology
*Norwegian University of Science and Technology
**Florida State University

Energy Efficient Processor Design

- Need for energy efficient processors.
  - Need to extend battery life for mobile systems.
  - Reduce generated heat for general-purpose processors.
  - Electricity cost for computing is increasing.
- Should also not negatively impact performance.
- Architecture features need to be reexamined with respect to
  energy efficiency, while retaining performance.

Embedded Processor Energy Breakdown


Techniques for Improving Data Access Energy Usage

- speculative tag access (STA)
- early load data dependence detection (ELD^3)
- speculative halt-tag access (SHA)
Evaluation Framework Used for These Studies

- Simulated an in-order processor with a 1 or 2 pipeline stages for level-one data cache (L1 DC) accesses.
- L1 DC is 16kB, 4-way set-associative, and has a 32B line size.
- Data translation lookaside buffer (DTLB) has 16 entries and is fully associative.
- RTL implementation synthesized (Synopsys Design Compiler) and placed and routed (Cadence Encounter) to obtain energy values for various events.

## Energy Values for Various Events

<table>
<thead>
<tr>
<th>Component</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTLB</td>
<td>17.5 pJ</td>
</tr>
<tr>
<td>L1 DC Read Tags - All Ways</td>
<td>57.3 pJ</td>
</tr>
<tr>
<td>L1 DC Read Data - All Ways</td>
<td>112.7 pJ</td>
</tr>
<tr>
<td>L1 DC Write Data</td>
<td>33.9 pJ</td>
</tr>
<tr>
<td>L1 DC Read Data</td>
<td>28.2 pJ</td>
</tr>
</tbody>
</table>

- Used the SimpleScalar simulator to count events and estimate total energy usage.

Benchmarks

- 20 benchmarks simulated from the MiBench benchmark suite.

<table>
<thead>
<tr>
<th>Category</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automotive</td>
<td>Basicmath, Bitcount, Qsort, Susan</td>
</tr>
<tr>
<td>Consumer</td>
<td>JPEG, Lame, TIFF</td>
</tr>
<tr>
<td>Network</td>
<td>Dijkstra, Patricia</td>
</tr>
<tr>
<td>Office</td>
<td>Ispell, Rsynth, Stringsearch</td>
</tr>
<tr>
<td>Security</td>
<td>Blowfish, Rijndael, SHA, PGP</td>
</tr>
<tr>
<td>Telecom</td>
<td>ADPCM, CRC32, FFT, GSM</td>
</tr>
</tbody>
</table>

Set-Associative Data Caches

- Most L1 data caches use a set associative organization to decrease the miss rate.
- L1 DC data arrays require more power to access than L1 DC tag arrays since the data arrays are much larger.

For stores all L1 DC tag ways are checked first and then a single data way is updated on the following cycle.

For loads all L1 DC tag and data ways are accessed in parallel since the loaded value may be used in subsequent instructions.

The requested data can at most reside in one of the n ways!

- We found an 8% execution time overhead on average when the L1 DC tag and data memories are sequentially accessed.

10% 30% 60% Contribution to overall L1 load access energy
Address Generation and L1 DC Access

- The address generation unit (AGU) calculates the effective address in a stage before the L1 DC is accessed.
- The AGU takes as input:
  - base address from a register value
  - displacement from an immediate value in the instruction
- The figure below assumes a virtually-indexed, physically-tagged (VIPT) organization.

Data Memory Address Calculation

If the displacement is small, then it is possible that the tag and set index portions of the memory address will be the same as these fields in the base address.

Speculative Tag Access

- Use the set index of the base address to speculatively access the L1 DC tags.
- Use the virtual page number (primarily the tag) of the base address to speculatively access the DTLB.
- One L1 DC data way is accessed after a successful speculation.

Speculative Address Calculation and Failure Detection

The carry-out signals from the line offset and set index are used to detect if there is a speculation failure due to an invalid set index to the L1 DC tag memory or due to an invalid virtual page number to the DTLB, respectively.
Speculative Access Benefits and Costs

- When the speculative access is successful:
  - On an L1 DC hit, the read energy of accessing \( n-1 \) data arrays of the \( n \)-way associative L1 DC is avoided.
  - On an L1 DC miss, the read energy of accessing all \( n \) data arrays of the L1 DC is avoided and the next level of the memory hierarchy is accessed one cycle earlier.

- When the speculative access is unsuccessful:
  - A speculation failure for the L1 DC index field requires the extra energy cost of unnecessarily accessing all the L1 DC tag arrays.
  - A speculation failure for the L1 DC tag field requires the extra energy cost of unnecessarily accessing the DTLB.

Load Speculation Success Rate

- 71.9\% of all loads successfully access L1 DC tags early.
- 1.9\% of loads cause speculation failures affecting tag access and 0.2\% cause failures affecting DTLB access.

L1 DC and DTLB Energy Results

- Idle, store, and miss energy are unaffected.
- Energy due to loads decreased from 77.2\% to 53.3\%, resulting in a 23.9\% L1 DC and DTLB energy savings.
Speculative Tag Access Contributions

- Speculative tag access reduces energy dissipated in an set-associative L1 DC with no execution time penalty.
- Benefits should increase as L1 DC line size increases.
  - Size of tag arrays compared to data arrays become smaller.
  - Fewer carry outs into the line index as line offset is larger.
- Benefits may increase as L1 DC associativity increases since the portion of L1 DC data being accessed on loads will decrease.

Nonspeculative Sequential Access to L1 DC Tag and Data

- Would be beneficial to sequentially access L1 DC tag and data memories for loads when the use of loaded value will not cause a stall.
  - Access all tag arrays during the typical L1 DC pipeline stage.
  - Access a single L1 DC data array in the following cycle.

L1 DC Access Spanning Two Pipeline Stages

Often the L1 DC access spans multiple pipeline stages.

L1 DC Access Spanning Three Pipeline Stages

Sequentially accessing the L1 DC tag and data memories can degrade performance.
**Dependences of Following Instructions in a Pipeline**

Cannot determine if instructions \( b \) and \( c \) have dependences with the load before the load accesses the L1 DC.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>load</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>DC1</td>
<td>DC2</td>
<td>DC3</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inst a</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>DC1</td>
<td>DC2</td>
<td>DC3</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inst b</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>DC1</td>
<td>DC2</td>
<td>DC3</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inst c</td>
<td>IF</td>
<td>ID</td>
<td>EX</td>
<td>DC1</td>
<td>DC2</td>
<td>DC3</td>
<td>WB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Early Load Data Dependence Detection (ELD³)**

- Associate a bit with each L1 IC instruction that is placed in a separate data dependence bit (DDB) memory.
- Each associated bit is set when the L1 IC line is filled.
- When committing a load, clear the associated DDB bit if there was no dependence detected between the load and the following three instructions.
- During the address generation stage for a load, read the associated bit in the DDB memory to determine if the L1 DC tag and data memories are to be sequentially accessed.

---

**Loads Accessing Only One L1 DC Data Way Using ELD³**

- Simulated an in-order processor with 2 pipeline stages for L1 DC accesses as the baseline.
- About 49% of the loads on average are able to sequentially access the L1 DC tag and data arrays using ELD³.

---

**L1 DC Energy Dissipation Using ELD³**

Reduces L1 DC energy usage by about 13%.
We also only applied ELD³ during address generation when the offset is too large to apply STA.

Combining STA and ELD³ reduces L1 DC energy usage by an additional 5% as compared to using the STA technique alone.

ELD³ is an effective technique for reducing L1 DC energy dissipation by accessing only a single L1 DC data array a cycle later when there is no dependency with a load instruction in the pipeline.

DDB memory is only accessed in the address generation pipeline stage when a load is detected during instruction decode.

Showed the ELD³ technique can be combined with the STA technique to further reduce L1 DC energy usage.

Way halting has been proposed to reduce energy usage.

L1 DC tags are split into two parts.

- A few low-order bits called the halt tag.
- The remaining higher-order bits.

Halt tags are checked first and only the ways of the halt tags that match the corresponding bits in the address are accessed for the remaining bits of the tag and the data.

The insight is that the low-order bits are the ones most likely to differ.

Conventional SRAMs are synchronous and can only be controlled at the start of a clock cycle.

The halt tag approach is impractical as it would either require an extra access cycle or a custom SRAM implementation, which would be costly to implement.

We propose to speculatively check halt tags in the address generation stage when the displacement is small.

The speculative halt tag access succeeds when the index and tag fields do not change after adding the displacement.

For large displacements the L1 DC is conventionally accessed without a halt tag comparison.

We use the virtual address to avoid speculatively accessing the DTLB and the OS performs page coloring so that the halt tag bits for the virtual and physical address are identical.
Speculative Halt Tag Access (SHA) (cont.)

- Only when a halt tag matches in the address generation stage are the corresponding L1 DC tag and data ways enabled in the SRAM access stage.

Pipeline with Speculative Halt-Tag Arrays and N-Way Cache

Energy Usage for Different Halt Tag Widths

- STA stands for speculative tag access, where the regular L1 DC tags are speculatively accessed.
- Only had access to a 32-bit wide SRAM macro for the halt tag array where we could store up to four 8-bit halt tags.

Halt Distribution Cases

- # ways means only # L1 DC tag and data arrays are enabled.
- No speculation means the displacement was too large.
- Over 55% of the accesses enable one L1 DC tag array and one L1 DC data array.

Fraction of L1 DC Data Arrays Accessed per Benchmark

- The average fraction of L1 DC data arrays accessed on successful speculations is slightly less than 0.25.
The average energy usage reduction is 25.6% compared to an L1 DC baseline with no halt tags.

### Average Energy Distribution

- The speculative halt tag access (SHA) reduces energy usage by 6% more than the speculative tag access (STA) approach.

### SHA Contributions

- The SHA design is simpler than the speculative tag access (STA) approach since halt tags are accessed in the address generation stage and the DTLB, L1 DC tags, and L1 DC data are accessed in the SRAM access stage.
- Unlike previously proposed way halting techniques, the SHA approach is practical since it can use conventional SRAM chips to implement the L1 DC.

### Related Work

Related Work (cont.)


Conclusions

- The speculative tag access (STA) approach reduces L1 DC and DTLB energy usage by avoiding unnecessary load accesses to L1 DC data arrays via speculatively accessing the L1 DC tags a cycle early.
- The early load data dependence detection (ELD³) approach reduces L1 DC and DTLB energy usage by avoiding unnecessary load accesses to L1 DC data arrays via sequentially accessing the L1 DC tag and data arrays when delaying the load by a cycle will not cause a stall.
- The speculative halt-tag access (SHA) approach reduces L1 DC and DTLB energy usage by an additional 6% over the speculative tag access (STA) approach.

Publications

- Reducing Set-Associative L1 Data Cache Energy by Early Load Data Dependence Detection (ELD³) by A. Bardizbanyan, M. Sjalander, D. Whalley, P. Larsson-Edefors in Design Automation & Test in Europe Conference (DATE), March 2014.