Improving Data Access Efficiency by Using Context-Aware Loads and Stores

Alen Bardizbanyan, Magnus Själander†, David Whalley‡, Per Larsson-Edefors

Chalmers University of Technology
†Uppsala University
‡Florida State University
Conventional L1 DC Access

Execution Units

Register File
Other Forwarding

ADDR-GEN

Base Address
Offset

AGU

SRAM-ACCESS

DTLB
TAG-0
TAG-N
DATA-0
DATA-N

Forwarding
DATA-FORMATTING

Way Select Format Forward Data

Writeback
Energy Usage of a 4-way L1 Data Cache

Address: Virtual Page Number (VPN) Set index Line offset

DTLB

VPN

Physical Page Number (PPN)

Way Select Logic

Way Select-0 Data-0

Way Select-1 Data-1

Way Select-2 Data-2

Way Select-3 Data-3

Data Out

10% 30% 60% Contribution to overall L1 load access energy

60% of the energy is due to reading the data memories in parallel.
Energy Usage of a 4-way L1 Data Cache

Address: Virtual Page Number (VPN) | Set index | Line offset

DTLB

Physical Page Number (PPN)

VPN

set index

line offset

set index

line offset

set index

line offset

set index

line offset

VPN

DTLB

Tag Array-0

Data Array-0

Tag Array-1

Data Array-1

Tag Array-2

Data Array-2

Tag Array-3

Data Array-3

Way Select Logic

Data Out

10%  30%  60% Contribution to overall L1 load access energy

60% of the energy is due to reading the data memories in parallel
Phased L1 DC Access
Average performance overhead of 8%.


\[ r[2] = M[r[4]] \]
\[ r[2] = M[r[4]] \]
Context Aware Loads — Case2

Context Aware Loads — Case2


\(<3 \text{ or more insts}>\)

Context Aware Loads — Case3

\[ <3 \text{ or more insts}> \]

\(<3 \text{ or more insts}>\)

**Context Aware Loads — Cases 0-3**

**Case 0:**
Normal Access

**Case 1:**
Avoids Stalls

**Case 2:**
1x Data Array Access

**Case 3:**
1x Data Array Access
No Tag Speculation

\[
\begin{align*}
\text{Case 1:} & \quad r[2] = M[r[4]] \\
\text{Case 3:} & \quad r[2] = M[r[4]+4] <3 \text{ or more insts}> \\
\end{align*}
\]
Context Aware Loads — Pipeline
L3: \( \text{r}[2] = \text{M}[\text{r}[4]]; \)
\[
\ldots
\]
\( \text{r}[4] = \text{r}[4] + 4; \)
\( \text{PC} = \text{r}[4] \neq \text{r}[5], L3; \)

\[
\ldots
\]
\( \text{r}[22] = \text{M}[\text{r}[sp] + 100]; \)
\( \text{r}[21] = \text{M}[\text{r}[sp] + 96]; \)
\( \text{r}[20] = \text{M}[\text{r}[sp] + 92]; \)
\[
\ldots
\]
### Strided Accesses — Strided Access Structure

\[ \text{L3: } r[2] = M[r[4]]; \]
\[ \ldots \]
\[ \text{PC} = r[4] != r[5], \text{L3}; \]
\[ \ldots \]
\[ r[22] = M[r[sp] + 100]; \]
\[ r[21] = M[r[sp] + 96]; \]
\[ r[20] = M[r[sp] + 92]; \]
\[ \ldots \]

<table>
<thead>
<tr>
<th>( V )</th>
<th>L1 DC tag</th>
<th>L1 DC index</th>
<th>L1 DC way</th>
<th>PP</th>
<th>DV</th>
<th>word offset</th>
<th>strided data (SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 2^n - 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Context Aware Loads — Case4

\[ r[2] = M[r[4]] \]
\[ r[2] = M[r[4]] \]

Context Aware Loads — Cases 4-7

**Case 4:**
Avoid 1 Stall
No DTLB Access
No Tag Checks
1x Data Array Access

**Case 5:**
No DTLB Access
No Tag Checks
1x Data Array Access

**Case 6:**
Avoid 2 Stalls
No DTLB Access
No Tag Checks
No Data Array Access

**Case 7:**
Avoid 1 Stall
No DTLB Access
No Tag Checks
No Data Array Access
Context Aware Loads — Pipeline
### Instruction Format

<table>
<thead>
<tr>
<th>6 bits</th>
<th>5 bits</th>
<th>5 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>rs</td>
<td>rt</td>
<td>immediate</td>
</tr>
</tbody>
</table>

(a) MIPS Instruction I Format

<table>
<thead>
<tr>
<th>6 bits</th>
<th>5 bits</th>
<th>5 bits</th>
<th>2+n bits</th>
<th>14–n bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>opcode</td>
<td>rs</td>
<td>rt</td>
<td>meminfo</td>
<td>immediate</td>
</tr>
</tbody>
</table>

(b) Enhanced Load and Store Instruction Format
Evaluation Framework

- VPO compiler
- MiBench
- Simple Scalar
- L1 DC: 16KiB, 4-way, 32-byte line
- DTLB: 16 entries, fully associative
- Energy: P&R netlist in 65-nm technology
<table>
<thead>
<tr>
<th>SAS Entries</th>
<th>Relative Data Access Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.45</td>
</tr>
<tr>
<td>3</td>
<td>0.50</td>
</tr>
<tr>
<td>7</td>
<td>0.55</td>
</tr>
</tbody>
</table>

![Graph showing Relative Data Access Energy for different SAS Entries](graph.png)

- **L1 DC**
- **DTLB**
- **SAS**
Classification of Loads and Stores

<table>
<thead>
<tr>
<th>Memory Access Classifications</th>
<th>Relative All Loads or Stores</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0-S0</td>
<td>0.44</td>
</tr>
<tr>
<td>C5</td>
<td>0.28</td>
</tr>
<tr>
<td>Original</td>
<td>0.32</td>
</tr>
</tbody>
</table>
Per Case

Memory Access Classifications

Data Access Energy Savings (%)

L1 DC  DTLB

Execution Time Savings (%)

C1  C4  C6  C7

0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5

C2  C3  C4  C5  C6  C7  S2  S3

Memory Access Classifications

Data Access Energy Savings (%)
Per Case

Memory Access Classifications

Data Access Energy Savings (%)

Execution Time Savings (%)

Classifications

C2 C3 C4 C5 C6 C7 S2 S3

L1 DC DTLB

C1 C4 C6 C7
On average 43% energy usage reduction.
On average 43% energy usage reduction
Execution Time Improvements

On average 6% execution time improvement
On average 6% execution time improvement
• Conventional associative L1 caches are power hungry
• Context aware data accesses reduce L1 data cache power
• Speculative and early tag access improves performance
Summary

- Conventional associative L1 caches are power hungry
- Context aware data accesses reduce L1 data cache power
- Speculative and early tag access improves performance

- 43% reduction in L1 data cache and DTLB energy usage
- 6% performance improvement