Towards General Purpose Tagged Memory

Wei Song, Alex Bradbury, and Robert Mullins
Computer Laboratory, University of Cambridge

2\textsuperscript{nd} RISC-V workshop, 30/06/2015

Code release and tutorial:
http://www.lowrisc.org/docs/tutorial/
lowRISC Project

• **lowRISC**
  – Open source SoC provider
  – 64-bit RISC-V ISA, Rocket core
  – Key features: tagged memory and “minion” cores

• **Tagged memory**
  – Adding tag bits for each 64-bit word.
  – [Permission] Protection against control-flow hijacking
  – [Lock] Memory synchronization
  – [Type] Garbage collection
  – [Breakpoint] Debugging
Rocket Chip:
Rocket Tile: Rocket core, private I$ and D$
Crossbar between L1 and banked L2
Banked L2 (coherence manager)
Single memory port, format converter
Memory
Support Tagged Memory (1)

Augment each 64-bit word with tag bits

Augmented cache line is transparent to coherence control.
Support Tagged Memory (1)

Augment each 64-bit word with tag bits

Augmented cache line is transparent to coherence control.

Memory is partitioned into data and tag regions.

Every memory access needs a data access and an extra tag access.

To reduce the number of tag access, a tag cache is added.
Support Tagged Memory (2)

New instructions for load/store tag:

LTAG rd, imm(rs1)   # load tag @ rs1 + imm to rd

STAG rs2, imm(s1)   # store tag rs2 @ rs1 + imm

Adding a new memory op type M_T in D$. No change in core pipeline.
New instructions for load/store tag:

LTAG rd, imm(rs1)    # load tag @ rs1 + imm to rd

STAG rs2, imm(s1)   # store tag rs2 @ rs1 + imm

Adding a new memory op type M_T in D$.
No change in core pipeline.

Multiple trackers (transaction handlers) to serve multiple memory access in parallel.

Non-intrusive to current Rocket chip.
Easy to implement.
But not efficient.
Function and Performance Tests

• Function tests:
  – Inline assembly to use LTAG/STAG
  – Bare metal assembly test cases (riscv-tests)
  – Larger C programs (proxy kernel) to memory sweep with tags
  – Boot Linux with separated tag region
  – SystemVerilog unit test for the tag cache (random package generation)

• Performance tests:
  – SPECInt 2006 benchmark (9 out of 12)
## SPECInt 2006 Result (1)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>perlbench</td>
<td>20</td>
<td>5</td>
<td>&lt;1</td>
<td>2</td>
<td>&lt;1</td>
<td>1.289</td>
<td>&lt;1</td>
<td>1.089</td>
<td>&lt;1</td>
<td>1.025</td>
<td>&lt;1</td>
<td>1.011</td>
</tr>
<tr>
<td>bzip2</td>
<td>&lt;1</td>
<td>14</td>
<td>10</td>
<td>16</td>
<td>10</td>
<td>1.941</td>
<td>7</td>
<td>1.688</td>
<td>3</td>
<td>1.281</td>
<td>&lt;1</td>
<td>1.007</td>
</tr>
<tr>
<td>gcc</td>
<td>15</td>
<td>11</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>1.497</td>
<td>&lt;1</td>
<td>1.240</td>
<td>&lt;1</td>
<td>1.072</td>
<td>&lt;1</td>
<td>1.023</td>
</tr>
<tr>
<td>mcf</td>
<td>&lt;1</td>
<td>168</td>
<td>104</td>
<td>136</td>
<td>67</td>
<td>1.651</td>
<td>40</td>
<td>1.409</td>
<td>11</td>
<td>1.128</td>
<td>3</td>
<td>1.040</td>
</tr>
<tr>
<td>gobmk</td>
<td>24</td>
<td>8</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>1.368</td>
<td>&lt;1</td>
<td>1.146</td>
<td>&lt;1</td>
<td>1.073</td>
<td>&lt;1</td>
<td>1.046</td>
</tr>
<tr>
<td>sjeng</td>
<td>11</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1.673</td>
<td>&lt;1</td>
<td>1.482</td>
<td>&lt;1</td>
<td>1.383</td>
<td>&lt;1</td>
<td>1.316</td>
</tr>
<tr>
<td>h264ref</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>&lt;1</td>
<td>1.480</td>
<td>&lt;1</td>
<td>1.265</td>
<td>&lt;1</td>
<td>1.109</td>
<td>&lt;1</td>
<td>1.028</td>
</tr>
<tr>
<td>omnetpp</td>
<td>40</td>
<td>5</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>1.653</td>
<td>&lt;1</td>
<td>1.415</td>
<td>&lt;1</td>
<td>1.190</td>
<td>&lt;1</td>
<td>1.042</td>
</tr>
<tr>
<td>astar</td>
<td>&lt;1</td>
<td>21</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>1.750</td>
<td>2</td>
<td>1.471</td>
<td>&lt;1</td>
<td>1.173</td>
<td>&lt;1</td>
<td>1.009</td>
</tr>
<tr>
<td>average</td>
<td>12</td>
<td>27</td>
<td>14</td>
<td>20</td>
<td>10</td>
<td>1.589</td>
<td>6</td>
<td>1.356</td>
<td>2</td>
<td>1.159</td>
<td>&lt;1</td>
<td>1.058</td>
</tr>
</tbody>
</table>

MPKI: misses per 1000 instructions  
TPKI: transactions per 1000 instructions
SPECInt 2006 Result (2)

Memory traffic ratio
Tagged/no tag

Memory Traffic (MPKI)
Future Work (1)

• Optimizing the tag cache
  – Identify non-tag pages
  – Automatic check for non-tag (zero tags) cache line
  – Write back only revised tags

• Further ISA support
  – Atomic data + tag load/store
  – I$, trap instructions lack of permissions in tags
  – Set/reset tags for a whole cache line
Future Work (2)

• General tag interpretation
  – Propagate tags through core pipeline
  – General tag interpretation/check/operation
  – Rule configuration
    • Static table
    • Controlled by minion cores
Conclusion

• Add tagged memory in Rocket chip
  – Augment cache lines with tag bits
  – Add a tag cache to reduce memory traffic
  – New instructions for tag load/store
  – More optimization in the future

For more information and get in touch:
1. Talk to Alex or me.
3. E-Mail to lowrisc-dev@lists.lowrisc.org