A Scorchingly Fast FPGA-Based Precise L1 LRU Cache Simulator

Josef Schneider, Jorgen Peddersen, Sri Parameswaran
Why simulate a cache?

To find optimal configuration for a specific application.

*max. Hit do not correlate with min. execution time / min. energy consumption*
Why simulate a cache?

- For many modern processors, the cache configuration can be chosen at design time (ARM, Xtensa, NIOS II, etc.)
- Optimal cache configuration can be found through cache simulation.
- Current cache simulator solutions require lengthy simulation times.
Contributions

- FPGA-based Multiple Cache Simulator exhibiting:
  - 100MHz memory trace consumption rate.
  - Simulates 44 caches concurrently.
  - Ability to implement in-system for real-time cache simulation.
Presentation Contents

- What is a cache?
- Existing simulator shortcomings
- Cache simulator design
- Performance comparison
- Future work
What is a cache?

CPU

Cache

1 – 2 CC

10+ CC

Main Memory
The Accountant’s Desk Analogy

A cache is like a desk.

Picking up a document from the desk takes very little time. Getting it from the filing cabinet takes a lot longer. When you get a document from the filing cabinet, you are likely to reuse it, so you put it on the desk for faster access. Getting it from the filing cabinet takes a lot longer.
Desk Considerations

The bigger the desk is, the more documents it can hold, but:

• It will cost more
• It will take longer to find this document on the desk
Three Factors, Many Combinations

1) Line Length
2) Set size
3) Associativity

Hundreds of Combinations

0
1
2
3
4
5
6
7
8
9

4 bytes
Lowest bits give cache index. For example, a 16 line cache will store: 0x2003
Notable Existing LRU Cache Sims

- DineroIV (J. Edler and M. Hill) for LRU, FIFO and Random cache replacement policies.
- Cheetah (Sugumar et al.).
- “Finding optimal L1 cache configuration for embedded systems” (A. Janapsatya et al.) for LRU policy.
- CRCB Algorithm (Tojo et al.).
- SuSeSim (M. Haque et al.).
Existing Cache Sim. Shortcomings

• All are implemented in software and require lengthy computations for advanced applications
  (Using Cheetah: simulating an MPEG2 encoder for 24 video frames took over 65 minutes for 44 cache configurations.)

• Simulations are *usually* based on static traces which take a very long time to generate
  (On the Xtensa processor simulator: creating the trace for the MPEG2 encoding application took over 70 hours.)

• Application inputs are hard-coded
  (Simulated inputs are contrived and may not represent realistic application input.)
Our Solution: Cache Sim. In Hardware

Our hardware-based cache simulator can be configured on an FPGA. By exploiting two LRU cache inclusion properties, many caches can be simulated concurrently and efficiently.

In software, inclusion properties allow for faster simulation (fewer searches). In hardware, the same properties are used to minimise hardware utilisation.
Inclusion Property 1

An LRU cache of set size $s$, line length $l$ and associativity $a$ is a subset of an LRU cache of set size $s$, line length $l$ and associativity larger than $a$.

(Mattson et al. 1970)
LRU Replacement Example

Access Address: 0x9004

Hit!
The same behaviour is exhibited by an LRU shift register.
Thanks to Inclusion Property 1, cache sets of associativity \( a \) and smaller can be simulated using the same hardware.
Top Level Cache Sim. Design

Enables the simulation of caches of set size 4 and associativities 4 and smaller.
A cache of line length $l$, associativity $a$ and set size $s$ is always a subset of a cache of line length $l$, associativity $a$ and set size larger than $s$. (Mattson et al. 1970)
Making use of Inclusion Property 2

This can be used to our advantage

Top level
Set Size = 4

Set Size = 2

‘Sub Set’

<table>
<thead>
<tr>
<th>Addr</th>
<th>1) 0b101000</th>
<th>2) 0b101010</th>
<th>3) 0b011000</th>
<th>4) 0b101000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid Location</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>
Combining Top Level with Sub Sets

Enables the simulation of caches of set sizes 16 to 4, and associativities 4 and smaller. Total caches simulated: 12
Simulating Multiple Line Sizes

Different line lengths = shift line index in address

Set index for a set of size 16 lines, line length = 1 byte

Address = 0b00010110100

Set index for a set of size 16 lines, line length = 2 bytes

Disadvantage: need to re-run simulation for each different line size.
## Implementation in Altera Stratix IV FPGA

<table>
<thead>
<tr>
<th>addr width</th>
<th>max assoc.</th>
<th>32</th>
<th>256</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LUTs</td>
<td>Registers</td>
<td>LUTs</td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>4932</td>
<td>3702</td>
<td>25085</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>10529</td>
<td>7163</td>
<td>56859</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>21434</td>
<td>14084</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>4</td>
<td>5517</td>
<td>4342</td>
<td>26541</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>11644</td>
<td>8443</td>
<td>60321</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>23672</td>
<td>16644</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>4</td>
<td>5997</td>
<td>4982</td>
<td>31990</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>12655</td>
<td>9723</td>
<td>70216</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>26203</td>
<td>19204</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>6410</td>
<td>5622</td>
<td>35313</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>13500</td>
<td>11003</td>
<td>77732</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>27829</td>
<td>21764</td>
<td></td>
</tr>
</tbody>
</table>
Simulator Characteristics on Altera Stratix IV FPGA

- Capable of simulating 308 cache configurations, of which it can simulate 44 concurrently.
- Address consumption rate of 100MHz.
- Fully written in VHDL and easily parameterisable.
- Two buses – one for trace input, another for control.
At the time of writing the paper, no direct performance comparison was possible. Simulator throughput was compared with:

- DineroIV
- Cheetah
- SuSeSim
Example Use: Static Trace Simulation

PCIe

Cache Sim
Example Use: In-System Implementation

Directly connected to soft-core processor
- Real time, real-input cache simulation
- Executes as fast as the system
Future Work

Multi-processor cache simulator.
Conclusions

• First hardware-based multiple cache simulator.
• Reduces hardware usage by utilising cache inclusion properties.
• Allows for real-time, in-system cache simulation.
• Throughput up to 53x faster than one of the fastest software-based cache simulators.
Thank You