Improving Performance of Software Implemented Floating Point Addition

Hindborg, Andreas Erik; Passas, Stavros; Karlsson, Sven

Publication date:
2011

Document Version
Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Citation (APA):
Improving Performance of Software Implemented Floating Point Addition
Andreas Erik Hindborg, Stavros Passas and Sven Karlsson
Technical University of Denmark

Motivation
- Multicore processors and systems are often constrained on power and hardware resources – It matters how resources are spent
- Dedicated hardware for floating point (FP) operations requires valuable hardware resources and consumes power
- Accelerators consume valuable chip area and may lead to an overall reduction of the number of cores
- Achieving acceleration of FP operations without spending valuable silicon area on big accelerators is desirable

Contributions
- We propose simple hardware extensions to an integer processor pipeline that enables acceleration of IEEE 754 [4] FP addition operations
- We simulate five core configurations with support for our extensions to evaluate their performance behavior

Methodology
- We propose twelve instructions to efficiently implement FP addition
  - When executed in sequence they realize FP addition
  - The instructions can be implemented by reusing many of the logic blocks found in modern processor cores
  - We estimate that a low amount of additional logic is needed

Simulation Setup
- We use the cycle accurate SimpleScalar ARM sim-outorder simulator [1]
- For each core configuration we execute the 470.1 benchmark SPEC2006 [3]
- We simulate the following base configuration:

  Configuration | Super scalar | Memory subsystem
  --------------|--------------|------------------
  Config. 1     | Yes          | Real

- And the following extended configurations:

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config. 2</td>
<td>Config. 1 with dedicated addressing unit</td>
</tr>
<tr>
<td>Config. 3</td>
<td>Config. 1 with four extended integer units</td>
</tr>
<tr>
<td>Config. 4</td>
<td>Config. 2 with four integer units (one extended)</td>
</tr>
<tr>
<td>Config. 5</td>
<td>Config. 2 with four extended integer units</td>
</tr>
</tbody>
</table>

Simulation Accuracy
- Utilizes twelve simple instructions to implement a single FP addition
- Simulates these instructions as a single instruction that occupies the integer pipeline for twelve cycles
- Captures the effects of resource allocation of the functional unit
- Effects of fetch, decode and commit are not simulated

Results

![Figure: Simulation results for our core configurations. Results shown are relative slowdowns compared to full hardware support.](image)

- **Configuration 1**: Super-scalar baseline processor with real memories
- **Configuration 2**: The addition of dedicated addressing unit increase the performance by 3.1 %
- **Configuration 3**: The use of four extended integer units improves the performance by 13.7 %

- **Configuration 4**: The addition of four integer units where only one is extended increases the performance by 0.5 %
- **Configuration 5**: The use of four extended integer units improves the performance by 9.9 %

Conclusions
- Our benchmark exhibits a relative slowdown of 3.38 to 15.15 when compared to dedicated hardware acceleration
- Pure software implementation leads to relative slowdowns of up to 45.33
- For processors with extra dedicated integer or addressing units performance improves by up to 13.7 % over our base configuration

Future Work
- Develop actual hardware model for the proposed methods using HDL
- Validate the cost of the proposed instructions with respect to area and power resources
- Extend the current work to include other operations such as division and multiplication

Related Work
- Chong et. al. [2] states that when using a pure software FP implementation, 90 % of the instructions are FP computations
- Rodolfo et al. [5] shows a speedup of 22 when using hardware instead of software FP operations

References