Hardware Managed Scratchpad for Embedded Systems

Introduction

• Major concern
  – Energy consumption, execution time
• Embedded Systems
  – Profiling advantage
• What is a Scratchpad Memory (SPM)
  – Array of SRAM cells
  – No extra bits or tags
• How can it be utilised?

Approach

black = data, orange = address, green = control
**Approach**

- SimpleScalar simulator, with syscall modification

- Accuracy within 0.05% of hardware simulations for large programs (> 1 million cycles)

**Software Modification**

```assembly
li $4,0x3b9aca00
move $16,$0
...
 ...
li $4,0x3b9aca00
move $16,$0
...
 ...

$L295:
addu $16,$16,1
...
 ...

$L294:
addu $16,$16,2
...
 ...

$end12:
bltz $7,$L284
...
 ...
```

**Results**

- 4 test cases: mem, cache, spm, optimised

**CPI for benchmarks**

```text
0.0
1.0
2.0
3.0
4.0
5.0
6.0

cpg     dpkg     marcaso    rawaudio     pegwit     pegwitenc
 rawca     rawda     pegwit

cpg     dpkg     marcaso    rawaudio     pegwit     pegwitenc
 rawca     rawda     pegwit
g721enc
g721dec
avg

<table>
<thead>
<tr>
<th>memory type</th>
<th>CPI</th>
<th>cache</th>
<th>spm</th>
<th>optimised</th>
<th>mem only</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional units</td>
<td>6 stage, statically scheduled, single instruction pipeline</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional unit latencies</td>
<td>All single cycle</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction cache</td>
<td>2 Kb (256 entries), direct mapped, 1 word block (64 bit), 1 cycle hit latency, 2 cycle miss latency (plus mem delay)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction SPM</td>
<td>8 Kb (1024 entries), 1 cycle hit latency</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction main memory</td>
<td>8 Mb SDRAM (10ns), simplified burst mode 10-1-1-1*, 4 word line size</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data cache</td>
<td>2 Kb (121 entries), direct mapped, 1 word block (32 bit), 1 cycle hit latency, 2 cycle miss latency (plus mem delay)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data main memory</td>
<td>8 Mb SDRAM (10ns), simplified burst mode 10-1-1-1*, 4 word line size</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results

Speedup from mem only

Problems

- Overhead
  - SPM fill vs cache miss
  - SPM fill vs cache hit (common case)
  - SPM hit vs cache hit (optimised case)
- Loop size
  - Three extra instructions (&mi, &jump, &jump)
  - Number of iterations
- Loop structure
  - if statements
  - loop in loop
- Optimised case
  - Saves on SPM fill overhead
  - Still 2 instructions

Limitations

- Library functions
  - Can't be copied at the moment
  - Account for 30% of execution time (adpcm)
- Hand maintenance
  - Instruction insertion
  - Update Controller
  - &jump addresses
- Size of SPM
  - Optimised case only

Saviour

2 Kb cache → 1.3 ns (760 MHz)  8 Kb SPM → 1.05 ns (950 MHz)
Future Work

- Software compiler
  - Automatically insert *smi* instructions (13 → 70)
  - Automatically update the Controller
  - Automatically update *jump* addresses
- Better procedure to locate blocks and loops of interest
- Optimisation mark II
  - Modify the optimised *smi* placement scheme
  - Use an extra SPM register
  - Allows > 1024 to be stored in the SPM

Conclusion

- Not overly promising so far
- Potential room for improvement through automation
- Next step:
  - Calculate energy consumption
  - Energy profile of the hardware model