Scheduling Reusable Instructions for Power Reduction

J. S. Hu, N. Vijaykrishnan, S. Kim, M. Kandemir, and M. J. Irwin

Microsystems Design Lab
The Pennsylvania State University
Power: StrongARM SA-110

- Power dissipation
  - ICache 27%
  - IBox 18%
  - EBox 8%
  - IMMU 9%
  - DCache 16%
  - DMMU 8%
  - Clock 10%
  - Write Buffer 2%
  - Bus Ctrl 2%
  - PLL < 1%

Die of DEC StrongARM
Related Work

- Stage-skip pipeline
  - A small decoded instruction buffer [1][2]
- Loop caches
  - Dynamic/preloaded/hybrid loop caches [3][4][5]
- Filter cache
  - Filter dcache [6], decode filter cache [7]
Related Work


Our Proposed Approach

- Scheduling reusable loop instructions within the issue queue
  - No need of an additional instruction buffer
  - Utilize the existing issue queue resources
  - Be able to gate the front-end of pipeline
  - Automatically unroll loops in the issue queue
  - No ISA modification
Embedded Processor based on MIPS Core

(a)

(b)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch</td>
<td>Decode</td>
<td>Rename Queue</td>
<td>Issue</td>
<td>Reg Read</td>
<td>Execute DcacheAcc</td>
</tr>
</tbody>
</table>

Reorder Buffer (ROB)

FP Function Units

Int Function Units

Add calc

Data Cache
Schedule Reusable Instructions

- Array-intensive embedded applications
- Utilizing issue queue
- Reusable instructions – innermost loops
- Self-steaming issue queue
- Gate front-end of the datapath
Loop Detection

Add check logic for conditional branch/jump instructions

Two checks:
(a) Backward branch/jump;
(b) Static distance <= issue queue size

This check is performed at decode stage instead of commit stage

NBLT is used to store current non-bufferable loops

Outer Loop (non-bufferable)

Innermost Loop (bufferable)

```assembly
addu r22, r0, r0
addiu r20, r0, 499
beq r20, r0, 0x4002e8
addu r4, r0, r6
addu r3, r0, r5
addu r2, r24, r22
sw r2, 0(r3)
subu r2, r24, 422
sw r2, 0(r4)
addiu r4, r4, 4
addiu r3, r3, 4
addiu r22, r22, 1
slti r2, r22, 499
bne r2, r0, 0x4002a0
addiu r6, r6, 2000
addiu r5, r5, 2000
addiu r24, r24, 1
slti r2, r24, 499
bne r2, r0, 0x400278
```
Buffering Reusable Instructions

- Extended issue queue microarchitecture
- Reusable instructions are marked, logical register numbers are stored in LRA
- Buffering integer number of loops

<table>
<thead>
<tr>
<th>63</th>
<th>62</th>
<th>---</th>
<th>---</th>
<th>---</th>
<th>---</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Index</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>63</th>
<th>62</th>
<th>---</th>
<th>---</th>
<th>---</th>
<th>---</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

Original Issue Queue

<table>
<thead>
<tr>
<th>63</th>
<th>62</th>
<th>---</th>
<th>---</th>
<th>---</th>
<th>---</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 1</td>
</tr>
</tbody>
</table>

Classification Bit

<table>
<thead>
<tr>
<th>63</th>
<th>62</th>
<th>---</th>
<th>---</th>
<th>---</th>
<th>---</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 0 0</td>
</tr>
</tbody>
</table>

Issue State Bit

<table>
<thead>
<tr>
<th>63</th>
<th>62</th>
<th>---</th>
<th>---</th>
<th>---</th>
<th>---</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

Logical Register List

<table>
<thead>
<tr>
<th>63</th>
<th>62</th>
<th>---</th>
<th>---</th>
<th>---</th>
<th>---</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 1</td>
</tr>
</tbody>
</table>

R_loophead

R_loopptail

4/4/2004
Reusing Buffered Instructions

<table>
<thead>
<tr>
<th>Logical Register List (15 bits)</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>63 62</td>
</tr>
<tr>
<td></td>
<td>2 1 0</td>
</tr>
</tbody>
</table>

---

**Original Issue Queue**

<table>
<thead>
<tr>
<th>Classification Bit</th>
<th>Issue State Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>1 0 0</td>
</tr>
</tbody>
</table>

---

**Logical Register List**

- Logical register numbers are sent for renaming.
- Check the first `m` instructions. If the first `n` instructions with both CB and ISB set, reuse those instructions. Reuse pointer advances by `n`.
- Reuse pointer is reset to `R_{loophead}` when it reaches `R_{looptail}`.
- ISB bit is set after the instruction is issued. Inst. With CB bit set will not be removed after its issue if ISB bit is set after the inst. is issued.
The New Datapath

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Rename Queue</th>
<th>Issue</th>
<th>Reg Read</th>
<th>Execute DcacheAcc</th>
<th>WriteBack</th>
<th>Commit</th>
</tr>
</thead>
</table>

(a)
- Inst. Cache
- Inst. Decoder
- Register Map Resource
- Gate-Signal
- Gate-Signal
- Register Rename
- Reuse Control
- Loop Detected
- Register #

(b)
- Reorder Buffer (ROB)
- FP Function Units
- Int Function Units
- Add calc
- Data Cache

4/4/2004  MDL@PSU
## Experiment Setup

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue Queue</td>
<td>64 entries</td>
</tr>
<tr>
<td>Load/Store Queue</td>
<td>32 entries</td>
</tr>
<tr>
<td>ROB</td>
<td>64 entries</td>
</tr>
<tr>
<td>Fetch Queue</td>
<td>4 entries</td>
</tr>
<tr>
<td>Fetch/Decode Width</td>
<td>4 inst. per cycle</td>
</tr>
<tr>
<td>Issue/Commit Width</td>
<td>4 inst. per cycle</td>
</tr>
<tr>
<td>Function Units</td>
<td>4 IALU, 1 IMULT, 4 FPALU, 1 FPMULT</td>
</tr>
<tr>
<td>Branch Predictor</td>
<td>bimod, 2048 entries, RAS 8 entries</td>
</tr>
<tr>
<td></td>
<td>BTB 512 set 4 way assoc.</td>
</tr>
<tr>
<td>L1 ICache</td>
<td>32KB, 2 way, 1 cycle</td>
</tr>
<tr>
<td>L1 DCache</td>
<td>32KB, 4 way, 1 cycle</td>
</tr>
<tr>
<td>L2 UCache</td>
<td>256KB, 4 way, 8 cycles</td>
</tr>
<tr>
<td>TLB</td>
<td>ITLB: 16 set 4 way, DTLB: 32 set 4 way</td>
</tr>
<tr>
<td>Memory</td>
<td>4KB page size, 30 cycle penalty</td>
</tr>
<tr>
<td></td>
<td>80 cycles for first chunk, 8 cycles the rest</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bench</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adi</td>
<td>Livermore</td>
</tr>
<tr>
<td>Aps</td>
<td>Perfect Club</td>
</tr>
<tr>
<td>Btrix</td>
<td>Spec92/NASA</td>
</tr>
<tr>
<td>Eflux</td>
<td>Perfect Club</td>
</tr>
<tr>
<td>Tomcat</td>
<td>Spec95</td>
</tr>
<tr>
<td>Tsf</td>
<td>Perfect Club</td>
</tr>
<tr>
<td>Vpenta</td>
<td>Spec92/NASA</td>
</tr>
<tr>
<td>wss</td>
<td>Perfect Club</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Simulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arch.</td>
<td>SimpleScalar 3.0</td>
</tr>
<tr>
<td>Power</td>
<td>Wattch</td>
</tr>
</tbody>
</table>
On the average, the pipeline front-end gated rate increase from 42% to 82% as the issue queue size increase.
Power Savings in Front-end

On the average, a reduction of 35% - 72% in ICache, 19% - 33% in branch predictor, 46% - 83% in Inst. decoder, and 12% - 21% in issue queue as the issue queue size increase, with < 2% overhead.
Impact of Compiler Optimizations

Optimized code increases power savings from 8% to 13% with issue queue size of 64 entries.
Conclusions

- Proposed a new issue queue architecture
  - Detect capturable loop code
  - Buffer loop code in the issue queue
  - Schedule the reusable loop inst. buffered

- Significant power reduction in pipeline front-end components while gated

- Compiler optimizations can further improve the power savings
Thank You!
Power: A Design Limiter?

- A major constraint in embedded systems design
- Superscalar architecture is more likely used for performance
- Front-end: a power-hungry component
- Seek to optimize the front-end of datapath in embedded processors
No impact on exception handling

Non-successful buffering will be revoked

Changing control flow in a buffering loop will cause buffering to be revoked

Branch prediction is disabled & switched to static prediction during Code_Reuse state

Misprediction due to normally existing a loop will restore issue queue to Normal state
Overall Power Reduction

Overall Power (per Cycle) Savings

IQ−32
IQ−64
IQ−128
IQ−256

adi
aps
trix
eflux
tomcat
tsf
vpenta
wss
avg
Performance Loss

Average performance loss ranges from 0.2% to 4% due to buffering integer number of loops.