Reducing Power with A Dynamically Reconfigurable Issue Queue

Yu Bai and Iris Bahar
Brown University
Division of Engineering

Motivation

- Performance and power trends
- Many complex architectural features are included
- These features consume power regardless of usage
- Adjustable datapath resources to match the application’s needs
- Focus on issue logic since it consumes a large portion of overall power dissipation
- For instance, it was projected that the 21464 issue logic would account for 46% of the total power

The FIFO Approach

- Proposed by Palacharla & Smith [ISCA 97]
- Fixed number and size of FIFOs
- Combined in-order & out-of-order issuing
- Dependent instructions are inserted into a single FIFO
- Instructions are issued from FIFOs in parallel
- Only the instruction at the head of each FIFO is visible to the arbitration logic

Fixed-sized FIFOs

- Proposed by Palacharla & Smith [ISCA 97]
- Fixed number and size of FIFOs
- Combined in-order & out-of-order issuing
- Dependent instructions are inserted into a single FIFO
- Instructions are issued from FIFOs in parallel
- Only the instruction at the head of each FIFO is visible to the arbitration logic

The FIFO Approach

- Proposed by Palacharla & Smith [ISCA 97]
- Fixed number and size of FIFOs
- Combined in-order & out-of-order issuing
- Dependent instructions are inserted into a single FIFO
- Instructions are issued from FIFOs in parallel
- Only the instruction at the head of each FIFO is visible to the arbitration logic

Fixed-sized FIFOs

- Proposed by Palacharla & Smith [ISCA 97]
- Fixed number and size of FIFOs
- Combined in-order & out-of-order issuing
- Dependent instructions are inserted into a single FIFO
- Instructions are issued from FIFOs in parallel
- Only the instruction at the head of each FIFO is visible to the arbitration logic
Limitations of Fixed FIFO Scheme

- A single configuration works well for some benchmarks, but not for others
- High ILP: use more, or smaller FIFOs
- Low ILP: use few FIFOs
- Change number and size of FIFOs dynamically according to program needs

Scheme 1: variable number of FIFOs

<table>
<thead>
<tr>
<th>FIFO 1</th>
<th>FIFO 2</th>
<th>FIFO 3</th>
<th>FIFO 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>req 1</td>
<td>req 2</td>
<td>req 3</td>
<td>req 4</td>
</tr>
<tr>
<td>gnt 1</td>
<td>gnt 2</td>
<td>gnt 3</td>
<td>gnt 4</td>
</tr>
</tbody>
</table>

4 Active 2-entry FIFOs (LPM)

Scheme 1: variable number of FIFOs

<table>
<thead>
<tr>
<th>FIFO 1</th>
<th>FIFO 2</th>
<th>FIFO 3</th>
<th>FIFO 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>req 1</td>
<td>req 2</td>
<td>req 3</td>
<td>req 4</td>
</tr>
<tr>
<td>gnt 1</td>
<td>gnt 2</td>
<td>gnt 3</td>
<td>gnt 4</td>
</tr>
</tbody>
</table>

2 Active 2-entry FIFOs (LPM 2)
Scheme 2: variable sized FIFOs

Arbitration Logic

8 1-entry FIFOs (FPM)

req 1

gnt 1

FIFO 1

req 2

gnt 2

FIFO 2

req 8

gnt 8

FIFO 8

8 instructions bid for issue slot

Scheme 2: variable sized FIFOs

Arbitration Logic

4 2-entry FIFOs (LPM 1)

req 1

gnt 1

FIFO 1

req 2

gnt 2

FIFO 2

req 3

gnt 3

FIFO 3

req 4

gnt 4

FIFO 4

4 instructions bid for issue slot

Scheme 2: variable sized FIFOs

Arbitration Logic

2 4-entry FIFOs (LPM 2)

req 1

gnt 1

FIFO 1

req 2

gnt 2

FIFO 2

req 3

gnt 3

FIFO 3

req 4

gnt 4

FIFO 4

2 instructions bid for issue slot

Scheme 1 vs. Scheme 2

Scheme 1
- Easier to implement
- Save more power
- Larger performance loss

Scheme 2
- Harder to implement
- Save less power
- Smaller performance loss

Combine Scheme 1 and Scheme 2
Hybrid Scheme: Scheme 1 + Scheme 2

Start from a traditional issue queue as Scheme 2

Apply Scheme 2 until reaching MAX_FIFO_SIZE

Then apply Scheme 1

How Do We Decide When to Switch?

Assumption: short term past behavior is a good indicator of behavior in the near future

How do we keep track of “program needs”? Keep track of statistics while a program is running

Help decide the optimal configuration

We use an array of monitors
**Experimental Results**

- Performance (Scheme 1)
- Performance (Scheme 2)
- Performance (Hybrid Scheme)
- Power savings (Scheme 1)
- Power savings (Scheme 2)
- Power savings (Hybrid Scheme)

**Conclusions**

- Issue queue is a major contributor to power
- Flexible schemes so we do not hamper performance
- Dynamically reconfigurable, FIFO-structured issue queue can save power with negligible performance impact