Recovery Mechanism for Latency Misprediction

Enric Morancho, José María Llabería and Àngel Olivé
Departament d'Arquitectura de Computadors
Universitat Politècnica de Catalunya - Spain

Work supported by the Ministry of Education and Science of Spain (TIC98-0511-C02-01)
Motivation

- High-performance processors demand back-to-back execution of dependent instructions

  \[
  \begin{array}{c|c|c|c|c|c}
  \text{R1} & \ldots & \text{IQ} & \text{R} & \text{exe} & \text{W} \\
  \ldots & \leftarrow & \text{R1} & \ldots & \text{IQ} & \text{R} & \text{exe} & \text{W} \\
  \end{array}
  \]

- Source-instruct. latency must be known on its issue cycle

- Load instructions have unknown latency

  - Delaying the issue of depend. instructions degrades IPC:
    - Hit latency 3 cycles: 6\% (+1), 11\% (+2), 16\% (+3)

  - Back-to-back execution is achieved by:
    - Latency prediction (hit)
    - Speculative scheduling of dependent instructions
    - Recovery mechanism on latency mispredictions
Outline

• Terminology
• Processor Model
• Recovery Mechanisms
  ❑ Issue-Queue Mechanism
  ❑ Recovery-Buffer Mechanism
• Methodology and Results
• Conclusions
Terminology

- **Independent Window (IW)**: interval where issued instructions are independent on the latency-predicted load instruction

- **Speculative Window (SW)**: interval between issuing the first instr. potentially dependent on the load and tag-checking (TC)

- **Verification Delay**: duration of the Speculative Window

  - Constant value
Tasks of a Recovery Mechanism

- **Nullify** some instructions issued during the Speculative Window
  - Remain marked as uncompleted in the Reorder Buffer
  - Nullification policy:
    - ✓ Non-selective: all instructions
    - ✓ Selective: only dependent instructions
- **Sleep** instrs. dependent on mispredicted and nullified instrs.
- **Re-issue** nullified instructions
  - Independent instructions: on next cycles
  - Dependent instructions: on data availability
- **Keep** speculatively issued instructions in a storage structure
Ex.: Non-selective Recovery Mechanism

- Issued instructions: 1 2 3 4 5 6 7
- a, v: IQ R @ M M/TC ...
- w: IQ R exe W
- x: IQ R exe W
- b, y: IQ R x
- c, z: IQ x
- y: IQ R ...
- z: IQ ...

a mispredicted
b, y nullified
c, z no issued
y re-issued
b, c, z slept

Lost Cycles
Processor Model (1/2)

- Pipeline stages:

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode/Rename</th>
<th>Issue Queue</th>
<th>Register Read</th>
<th>Execute</th>
<th>Write</th>
<th>Commit</th>
</tr>
</thead>
</table>

- Instructions are extracted from IQ after issuing them
- Issue-Queue capacity < Reorder-Buffer capacity

- Latency predictor:
  - Always predicts cache-hit latency
Processor Model (2/2)

• Structure of the Issue Queue

- **Rows:** related to queued instructions
- **Columns:** related to physical registers, mark data availability. Columns are set by latency counters
- **Ready bits are evaluated every cycle**
Storage for speculative instructions

• Evaluated structures
  ❑ Issue Queue
  ❑ Recovery Buffer
Recovery Mechanism for Latency Misprediction

**Issue-Queue Mechanism**

- Issued instructions are made non-visible to the select logic
  - non-request bits are added to the IQ entries
- On mispredictions, unsets columns / makes instructions visible
- non-visible instructions are extracted after Verification Delay cycles
Recovery Mechanism for Latency Misprediction

Recovery-Buffer Mechanism (1/3)

- Keeping speculatively-issued instructions in the Issue Queue reduces its ability to look-ahead for independent instructions
- RB keeps issued instructs while they can be nullified
  - As soon as an instruction is issued, it is extracted from IQ
  - A RB entry contains all the instructions issued concurr.
  - Instructions are ordered in issue-cycle order
Recovery Buffer Mechanism (2/3)

- On a misprediction:
  - Nullifies issued instructions dependent on the mispredict.
  - Sleeps instructions dependent on the misprediction
    - Same operations than in IQ mechanism
  - Nullified instructions are kept in the RB
    - An entry range is related to every misprediction
  - The scheduling recorded in RB is valid
    - Re-issue does not need to account latencies
Recovery Mechanism for Latency Misprediction

Recovery-Buffer Mechanism (3/3)

- Re-issue performed from the RB
  - Checks a RB entry per cycle
  - Also issues instructions from IQ in the free issue slots
  - Wakes-up dependent instructions recorded in IQ

![Diagram of Recovery-Buffer Mechanism](image-url)
Example: RB with selective nullification

Recovery Buffer: (b,o,c) (b,-,c)
Methodology

• Cycle-by-cycle simulation of SPEC-95 benchmarks

• Simulations performed:
  ❑ 4-way processor
    ✓ first-level cache latency: 2 cycles
  ❑ Issue-Queue size:
    ✓ 15, 20 and 25-entry integer IQ's
    ✓ 10, 15 and 20-entry floating-point IQ's
  ❑ Verification Delay: 2, 3 and 4 cycles
## Evaluated mechanisms

<table>
<thead>
<tr>
<th>Nullification policy</th>
<th>Storage Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non selective</td>
<td>Issue Queue</td>
</tr>
<tr>
<td></td>
<td>IQNS</td>
</tr>
<tr>
<td>Selective</td>
<td>Recovery Buffer</td>
</tr>
<tr>
<td></td>
<td>RBNS</td>
</tr>
<tr>
<td></td>
<td>IQS</td>
</tr>
<tr>
<td></td>
<td>RBS</td>
</tr>
</tbody>
</table>

- After issuing an instruction, extracting it from IQ is delayed:
  - IQNS: Verification-Delay cycles
  - IQS: 1 cycle (to decide dependencies)
  - RBNS and RBS: 0 cycles
Results: Integer benchmarks (1/2)

- Sensitivity to the verification delay
  - IQNS and IQS: function of Issue-Queue size
    - significant for 15-entry Issue Queue
  - RBNS and RBS: almost independent

- For 25-entry Issue Queues, IQS & RBNS are almost equivalent
Results: Integer benchmarks (2/2)

- Sensitivity to Issue-Queue size

  RBNS and RBS allow Issue-Queue size reductions respect IQNS and IQS around 25%
Results: Floating-Point benchmarks (1/2)

- Present different behaviour than integer benchmarks

  Latencies forbid the existence of a chain of dependent instructions larger than one in the Speculative Window

Fr1 ← load... 

Fr2 ← float (Fr1, ...) 

Fr3 ← float (Fr2, ...)
Results: Floating-Point benchmarks (2/2)

- Sensitivity to the verification delay
  - Non-selective mechanisms present degradation
    - Most nullified instructions are independent on the misprediction:
      85% (floating-point) versus 53% (integer)
  - Selective mechanisms are almost independent
Conclusions

• The Recovery Buffer increases the capacity of the scheduler to look-ahead for independent instructions

• Results depend on the dominating instruction latency
  - Integer benchmarks: (1-cycle latency)
    ✓ Recovery-Buffer mechanisms are less sensitive to the Verification Delay than IQ mechanisms
    ✓ Recovery-Buffer mechanisms allows a reduction in the Issue-Queue size around 25%
    ✓ The Recovery-Buffer mechanism with non-selective nullification policy is an attractive alternative
  - Floating-point benchmarks: (4-cycle latency)
    ✓ Non-selective nullification degrades performance
# Comparison of prediction types

<table>
<thead>
<tr>
<th></th>
<th>Branch prediction</th>
<th>Memory dependence prediction</th>
<th>Latency prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which instructions are predicted?</td>
<td>branches</td>
<td>loads</td>
<td>loads</td>
</tr>
<tr>
<td>Can the speculative instructions be issued before issuing the predicted instruction?</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Which instruction performs the verification of the prediction?</td>
<td>branch</td>
<td>previous store</td>
<td>load</td>
</tr>
<tr>
<td>Speculative-Window duration?</td>
<td>Variable</td>
<td>Variable</td>
<td>Fixed</td>
</tr>
<tr>
<td>Which instructions must be re-executed?</td>
<td>New path</td>
<td>The nullified ones</td>
<td>The nullified ones</td>
</tr>
</tbody>
</table>
## Verification-Delay values

### Decoupled TC

<table>
<thead>
<tr>
<th>IQ</th>
<th>R</th>
<th>@</th>
<th>M</th>
<th>M/TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>IQ</td>
<td>R</td>
<td>exe</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>IQ</td>
<td>R</td>
<td>exe</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ</td>
<td>R</td>
<td>exe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2 Read Stages, Decoupled TC

<table>
<thead>
<tr>
<th>IQ</th>
<th>R</th>
<th>@</th>
<th>M</th>
<th>M</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>IQ</td>
<td>R</td>
<td>R</td>
<td>exe</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>IQ</td>
<td>R</td>
<td>R</td>
<td>exe</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>IQ</td>
<td>R</td>
<td>R</td>
<td>exe</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Pipelined scheduling logic, Decoupled TC

<table>
<thead>
<tr>
<th>WU</th>
<th>S</th>
<th>R</th>
<th>@</th>
<th>M</th>
<th>M</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WU</td>
<td>S</td>
<td>R</td>
<td>exe</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WU</td>
<td>S</td>
<td>R</td>
<td>exe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WU</td>
<td>S</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2 Read Stages, Decoupled TC

<table>
<thead>
<tr>
<th>IQ</th>
<th>R</th>
<th>@</th>
<th>M</th>
<th>M</th>
<th>TC</th>
</tr>
</thead>
<tbody>
<tr>
<td>IQ</td>
<td>R</td>
<td>exe</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ</td>
<td>R</td>
<td>exe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IQ</td>
<td>R</td>
<td>exe</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3

<table>
<thead>
<tr>
<th>IQ</th>
<th>R</th>
<th>exe</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>IQ</td>
<td>R</td>
<td>exe</td>
<td></td>
</tr>
<tr>
<td>IQ</td>
<td>R</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 4

<table>
<thead>
<tr>
<th>IQ</th>
<th>R</th>
<th>exe</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>IQ</td>
<td>R</td>
<td>exe</td>
<td></td>
</tr>
<tr>
<td>IQ</td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>IQ</td>
<td>R</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WU</th>
<th>S</th>
<th>R</th>
<th>exe</th>
</tr>
</thead>
<tbody>
<tr>
<td>WU</td>
<td>S</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>WU</td>
<td>S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WU</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| WU |        |
|---|---
| WU |
Recovery Buff. & Branch Mispredictions

• On a branch misprediction, some instructions that belong to a wrong path can be recorded in the Recovery Buffer
  - These instructions must not be re-issued
    ✓ A "structure" contains the instruction-identifier ranges related to wrong-path instructions to filter-out them
  - Recovery Buffer maintains locally the status of the physical registers:
    ✓ Set: on issue and re-issue
    ✓ Unset: on nullifications
• These actions are performed concurrently with the normal operations of the Recovery Buffer