CHERRY: CHECKPOINTED EARLY RESOURCE RECYCLING

José F. Martínez¹, Jose Renau²
Michael C. Huang³, Milos Prvulovic², and Josep Torrellas²
**Problem**: Limited processor resources

**Goal**: More efficient use by aggressive recycling

**Opportunity**: Resources reserved until retirement

**Solution**: Decouple recycling from retirement
Current Processors

LDQ Entry

STQ Entry

Register

Conventional ROB

Tail (newest)

Head (oldest)
CURRENT PROCESSORS

LDQ Entry
STQ Entry
Register

Conventional ROB

Tail (newest)  Head (oldest)

LD
CURRENT PROCESSORS

LDQ Entry
STQ Entry
Register

Conventional ROB

Tail (newest)

Head (oldest)

Add LD

LD Add
CURRENT PROCESSORS

LDQ Entry
STQ Entry
Register

Conventional ROB
Tail (newest)

Head (oldest)

LD
Add
BR
ST
Current Processors

LDQ Entry

STQ Entry

Register

Conventional ROB

Tail (newest)

Head (oldest)
Current Processors

LDQ Entry

STQ Entry

Register

Conventional ROB

Tail (newest)

Head (oldest)

LD  Add  BR  ST  BR  Add  LD
CURRENT PROCESSORS

LDQ Entry

STQ Entry

Register

Conventional ROB

Tail (newest)

Head (oldest)
CURRENT PROCESSEORS

LDQ Entry
STQ Entry
Register

Conventional ROB

Tail (newest)

Head (oldest)

LD Add
BR ST
LD ST
CURRENT PROCESSORS

LDQ Entry
STQ Entry
Register

Conventional ROB

Tail (newest)

Head (oldest)
CURRENT PROCESSORS

LDQ Entry
STQ Entry
Register

Conventional ROB
Tail (newest)

Head (oldest)

LD Add
BR ST
BR LD
ST
CURRENT PROCESSORS

LDQ Entry
STQ Entry
Register

Conventional ROB
Tail (newest)
Head (oldest)
PROPOSAL: EARLY RECYCLING

- Decouple resource recycling from instruction retirement
  - Recycle when no longer needed

- Targeted resources:
  - Load queue entries
  - Store queue entries
  - Physical registers

- Potential when targeted resources are unlimited:
  - 1.12 speedup in SPECint2000
  - 1.32 speedup in SPECfp2000
Early Recycling

- Decouple resource recycling from retirement

LDQ Entry
STQ Entry
Register

Conventional ROB
Tail (newest)

Head (oldest)
Decouple resource recycling from retirement.
Decouple resource recycling from retirement

LDQ Entry

STQ Entry

Register

Conventional ROB

Head (oldest)

Tail (newest)
Decouple resource recycling from retirement

LDQ Entry
STQ Entry
Register

Conventional ROB
Tail (newest)
Head (oldest)
Events that require precise state:
- Branch mispredictions
- Memory replay traps
- Exceptions
- Interrupts

Early recycling makes this difficult
- E.g. recycled registers no longer available
- E.g. memory may be overwritten prematurely
PROPOSAL: POINT OF NO RETURN

- Classify events according to frequency:
  - Frequent: branch mispredictions, memory replays
  - Infrequent: exceptions

- Split ROB into two logical sets of instructions
  - Irreversible set
    - Subject to infrequent events only
    - Checkpoint recovery
  - Reversible set:
    - Subject to frequent and infrequent events
    - Conventional ROB recovery mechanisms

- Boundary: **Point of No Return** (PNR)
**CONVENTIONAL VS CHERRY ROB**

- Reversible
- Conventional ROB
- Head (oldest)
CONVENTIONAL VS CHERRY ROB

Conventional ROB

Cherry ROB

Reversible

Irreversible

Point of No Return

Head (oldest)
CONVENTIONAL VS CHERRY ROB

Conventional ROB:
- Reversible
- Head (oldest)

Cherry ROB:
- Reversible
- Irreversible
- Point of No Return
- Head (oldest)
Conventional ROB
Checkpointed Early Resource Recycling in Out-of-order Microprocessors

Cherry

Conventional ROB | Cherry | Cherry

Timeline

Checkpoint  Checkpoint  Checkpoint
Checkpoint

Checkpoint

Checkpoint

Conventional ROB

Cherry

Cherry

Rollback

Exception

Timeline
Checkpoint

Checkpoint

Checkpoint

Checkpoint

Checkpoint

Rollback

Re-exec

Exception

Conventional ROB

Cherry

Cherry

Cherry

Cherry

Conventional ROB

Timeline
- Backup architectural registers
  - Can be done in a handful of cycles
- Allow PNR to race ahead of ROB head
- Updates to cache hierarchy set Volatile bit
Done regularly to bound re-execution overhead

Stop early recycling to create checkpoint (collapse)
- Freeze PNR (stop recycling)

Once ROB head catches up with PNR:
- Clear Volatile bits from cache
- Backup architectural registers
Exception in Irreversible set:
- Roll back to checkpoint
- Temporarily disable recycling (for some time)
- Re-execute in conventional OOO mode (w/o Cherry)
- Allow exception to re-occur
Exception in Irreversible set:
- Roll back to checkpoint
- Temporarily disable recycling (for some time)
- Re-execute in conventional OOO mode (w/o Cherry)
- Allow exception to re-occur
Exception in Irreversible set:
- Roll back to checkpoint
- Temporarily disable recycling (for some time)
- Re-execute in conventional OOO mode (w/o Cherry)
- Allow exception to re-occur
Exception in Irreversible set:
- Roll back to checkpoint
- Temporarily disable recycling (for some time)
- Re-execute in conventional OOO mode (w/o Cherry)
- Allow exception to re-occur
Exception in Irreversible set:
- Roll back to checkpoint
- Temporarily disable recycling (for some time)
- Re-execute in conventional OOO mode (w/o Cherry)
- Allow exception to re-occur
**Precise Exception Handling**

- Exception in Irreversible set:
  - Roll back to checkpoint
  - Temporarily disable recycling (for some time)
  - Re-execute in conventional OOO mode (w/o Cherry)
  - Allow exception to re-occur
Exception in Reversible set:
- Disable resource recycling (freeze PNR)
- Process exception
- Take new checkpoint; re-enter Cherry mode
Exception in Reversible set:
- Disable resource recycling (freeze PNR)
- Process exception
- Take new checkpoint; re-enter Cherry mode
Exception in Reversible set:
- Disable resource recycling (freeze PNR)
- Process exception
- Take new checkpoint; re-enter Cherry mode
Exception in Reversible set:
- Disable resource recycling (freeze PNR)
- Process exception
- Take new checkpoint; re-enter Cherry mode
Interrupts are asynchronous: delay handling
- Disable resource recycling (freeze PNR)
- Process interrupt
- Take new checkpoint and re-enter Cherry mode
Interrupts are asynchronous: delay handling

- Disable resource recycling (freeze PNR)
- Process interrupt
- Take new checkpoint and re-enter Cherry mode
Interrupt Handling

- Interrupts are asynchronous: delay handling
  - Disable resource recycling (freeze PNR)
  - Process interrupt
  - Take new checkpoint and re-enter Cherry mode
Interrupts are asynchronous: delay handling
- Disable resource recycling (freeze PNR)
- Process interrupt
- Take new checkpoint and re-enter Cherry mode

**Diagram:**
- Timeline showing the process of checkpointing and handling interrupts.
  - Initial checkpoint labeled as "Checkpoint 1.
  - Process interrupt indicated by the red arrow labeled "Process Interrupt.
  - A new checkpoint labeled "Checkpoint 2 is taken.
  - The process enters another Cherry mode segment labeled "Cherry 2."
Interrupts are asynchronous: delay handling
- Disable resource recycling (freeze PNR)
- Process interrupt
- Take new checkpoint and re-enter Cherry mode
Motivation

Overview

Implementation
  - Checkpointing
  - Exception and interrupt handling
  - Early resource recycling

Results
Each resource defines its own PNR

Based on three ROB pointers:

- $U_b$: oldest unresolved branch
- $U_s$: oldest store with unresolved address
- $U_l$: oldest load with unresolved address
Early LDQ Entry Recycling

LQQ
  L2
  L1

Us

Head (oldest)

ST-LD replay check
Constraint: Must detect memory replay traps
- older($U_s$): ST-LD replay trap
- Not affected by branches

\[ \text{PNR}_{LDQ} = U_s \]
Constraint: Must not contain potentially useful data
- older($U_s$): Free of ST-LD replay trap
- older($U_b$): Free of branch misprediction

$$\text{PNR}_{\text{Reg}} = \text{oldest}(U_s, U_B)$$

- Of course register must be dead (paper)
**EARLY RECYCLING SUMMARY**

<table>
<thead>
<tr>
<th>Resource</th>
<th>PNR Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDQ entries</td>
<td>$U_S$</td>
</tr>
<tr>
<td>STQ entries</td>
<td>$\text{oldest}(U_L, U_S, U_B)$</td>
</tr>
<tr>
<td>Registers</td>
<td>$\text{oldest}(U_S, U_B)$</td>
</tr>
</tbody>
</table>

**Diagram:**

- $\text{PNR}_{LDQ}$
- $\text{PNR}_{STQ}$
- $\text{PNR}_{Reg}$

**Labels:**

- B (newest)
- S
- L
- $U_b$
- $U_s$
- $U_l$
- Tail (newest)
- Head (oldest)
Execution-driven simulation
SPEC 2000 applications
SGI MIPSPro compiler
Compare Cherry against:
  - Processor with unlimited resources
  - Three aggressive base processor designs (paper)
**Processor Model**

<table>
<thead>
<tr>
<th>Frequency: 3.2GHz</th>
<th>Cherry checkpoint frequency: 5μs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch/issue/commit width: 8/8/12</td>
<td>Up to 1 taken branch/cycle</td>
</tr>
<tr>
<td>I. window/ROB size: 128/384</td>
<td>RAS: 32 entries</td>
</tr>
<tr>
<td>Int/FP registers : 192/128</td>
<td>BTB: 4K entries, 4-way assoc.</td>
</tr>
<tr>
<td>Ld/St units: 2/2</td>
<td>Branch predictor:</td>
</tr>
<tr>
<td>Int/FP/branch units: 7/5/3</td>
<td>Hybrid with speculative update</td>
</tr>
<tr>
<td>Ld/St queue entries: 32/32</td>
<td>Bimodal size: 8K entries</td>
</tr>
<tr>
<td>MSHRs: 24</td>
<td>Two-level size: 64K entries</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cache</th>
<th>L1</th>
<th>L2</th>
<th>Bus &amp; Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size:</td>
<td>32KB</td>
<td>512KB</td>
<td>FSB frequency: 400MHz</td>
</tr>
<tr>
<td>RT:</td>
<td>2 cycles</td>
<td>10 cycles</td>
<td>FSB width: 128bit</td>
</tr>
<tr>
<td>Assoc:</td>
<td>4-way</td>
<td>8-way</td>
<td>Memory: 4-channel Rambus</td>
</tr>
<tr>
<td>Line size:</td>
<td>64B</td>
<td>128B</td>
<td>DRAM bandwidth: 6.4GB/s</td>
</tr>
<tr>
<td>Ports:</td>
<td>4</td>
<td>1</td>
<td>Memory RT: 120ns</td>
</tr>
</tbody>
</table>
OVERALL RESULTS

<table>
<thead>
<tr>
<th>Application</th>
<th>Cherry</th>
<th>Unlimited</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip2</td>
<td>1.06</td>
<td></td>
</tr>
<tr>
<td>crafty</td>
<td>1.06</td>
<td></td>
</tr>
<tr>
<td>gcc</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>gzip</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>mcf</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>parser</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>perlbench</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>vortex</td>
<td>1.36</td>
<td></td>
</tr>
<tr>
<td>vpr</td>
<td>1.06</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>1.06</td>
<td></td>
</tr>
</tbody>
</table>
OVERALL RESULTS

Speedup

1.60
1.54
1.48
1.42
1.36
1.30
1.24
1.18
1.12
1.06
1.00

Cherry
Unlimited

1.00
1.06
1.12
1.18
1.24
1.30
1.36

Speedup

1.26
1.06

bzip2  crafty  gcc  gzip  mcf  parser  perlbench  vortex  vpr  Average

applu  apsi  art  equake  mesa  mgrid  swim  wupwise  Average

Cherry: Checkpointed Early Resource Recycling in Out-of-order Microprocessors
OVERALL RESULTS

Cherry: Checkpointed Early Resource Recycling in Out-of-order Microprocessors
PNR advance as a % of ROB use
PNR advance as a % of ROB use

LDQ Entry

77.4%

SPECint

Tail (newest)  Head (oldest)
PNR advance as a % of ROB use

- LDQ Entry: 77.4% SPECint
- STQ Entry: 19.7% SPECint
PNR advance as a % of ROB use

- LDQ Entry: 77.4%
- STQ Entry: 19.7%
- Register: 28.7%

SPECint
PNR advance as a % of ROB use

- **LDQ Entry**
  - 77.4% SPECint
  - 78.3% SPECfp

- **STQ Entry**
  - 63.4% SPECint
  - 19.7% SPECfp

- **Register**
  - 67.7% SPECint
  - 28.7% SPECfp

---

**Cherry:** Checkpointed Early Resource Recycling in Out-of-order Microprocessors
SUMMARY

- Decoupling resource recycling from retirement
  - Split ROB into two logical sets
  - Use of ROB+checkpointing simultaneously
- Early recycling mechanism: LDQ, STQ, Registers
- Integration with speculative multithreading (paper)
- Speedup over baseline
  - 1.06 for SPECint2000
  - 1.26 for SPECfp2000
José F. Martínez¹
Jose Renau²
Michael C. Huang³
Milos Prvulovic²
Josep Torrellas²
BACKUP SLIDES
<table>
<thead>
<tr>
<th>Resource</th>
<th>PNR Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDQ entries (uniprocessor)</td>
<td>$U_S$</td>
</tr>
<tr>
<td>LDQ entries (multiprocessor)</td>
<td>oldest($U_L, U_S$)</td>
</tr>
<tr>
<td>STQ entries</td>
<td>oldest($U_L, U_S, U_B$)</td>
</tr>
<tr>
<td>Registers (uniprocessor)</td>
<td>oldest($U_S, U_B$)</td>
</tr>
<tr>
<td>Registers (multiprocessor)</td>
<td>oldest($U_L, U_S, U_B$)</td>
</tr>
</tbody>
</table>
Constraint: Must not overwrite potentially useful data
- $\text{older}(U_l)$: Unresolved LD may need old data
- $\text{older}(U_s)$: No older LD is replayable
- $\text{older}(U_b)$: Free of branch misprediction

$$\text{PNR}_{STQ} = \text{oldest}(U_L, U_S, U_B)$$
\[ T_o = c_k + p_e c_e \]  \hspace{1cm} (1)

\[ p_e = \frac{T_c}{T_e} \]  \hspace{1cm} (2)

\[ c_e = s \frac{T_c}{2} + (s - 1) \]  \hspace{1cm} (3)

\[ \frac{T_o}{T_c} = \frac{c_k}{T_c} + \left( s - \frac{1}{2} \right) \frac{T_c}{T_e} \]  \hspace{1cm} (4)

\[ T_c = \sqrt{\frac{c_k T_e}{s - \frac{1}{2}}} \]  \hspace{1cm} (5)
Checkpoint Selection

\[ \frac{T_0}{T_c} \] vs. \[ T_c (\mu s) \]

- \( T_e = 200 \mu s \)
- \( 400 \mu s \)
- \( 600 \mu s \)
- \( 800 \mu s \)
- \( 1000 \mu s \)
### SPECint2000 Parameters

<table>
<thead>
<tr>
<th>SPECint</th>
<th>Apps</th>
<th>Used ROB (%)</th>
<th>Irreversible Set (% of Used ROB)</th>
<th>Collapse Step (Cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Reg</td>
<td>LQ</td>
<td>SQ</td>
</tr>
<tr>
<td>bzip2</td>
<td>29.9</td>
<td>24.3</td>
<td>55.8</td>
<td>19.5</td>
</tr>
<tr>
<td>crafty</td>
<td>28.8</td>
<td>33.4</td>
<td>97.6</td>
<td>28.6</td>
</tr>
<tr>
<td>gcc</td>
<td>19.1</td>
<td>19.0</td>
<td>82.3</td>
<td>17.8</td>
</tr>
<tr>
<td>gzip</td>
<td>28.5</td>
<td>65.5</td>
<td>81.7</td>
<td>8.5</td>
</tr>
<tr>
<td>mcf</td>
<td>30.1</td>
<td>14.6</td>
<td>37.7</td>
<td>13.8</td>
</tr>
<tr>
<td>parser</td>
<td>30.7</td>
<td>26.1</td>
<td>80.7</td>
<td>21.8</td>
</tr>
<tr>
<td>perlbmk</td>
<td>12.2</td>
<td>24.6</td>
<td>89.9</td>
<td>20.5</td>
</tr>
<tr>
<td>vortex</td>
<td>39.3</td>
<td>26.3</td>
<td>87.1</td>
<td>24.9</td>
</tr>
<tr>
<td>vpr</td>
<td>32.9</td>
<td>25.2</td>
<td>83.6</td>
<td>21.5</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>27.9</strong></td>
<td><strong>28.7</strong></td>
<td><strong>77.4</strong></td>
<td><strong>19.7</strong></td>
</tr>
<tr>
<td>Apps</td>
<td>Used ROB (%)</td>
<td>Irreversible Set (% of Used ROB)</td>
<td>Collapse Step (Cycles)</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------------</td>
<td>-----------------------------------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reg</td>
<td>LQ</td>
<td>SQ</td>
</tr>
<tr>
<td>applu</td>
<td>62.2</td>
<td>61.6</td>
<td>62.4</td>
<td>60.7</td>
</tr>
<tr>
<td>apsi</td>
<td>76.8</td>
<td>82.3</td>
<td>83.1</td>
<td>81.6</td>
</tr>
<tr>
<td>art</td>
<td>88.0</td>
<td>54.3</td>
<td>62.6</td>
<td>29.2</td>
</tr>
<tr>
<td>equake</td>
<td>41.6</td>
<td>61.6</td>
<td>69.1</td>
<td>57.3</td>
</tr>
<tr>
<td>mesa</td>
<td>29.8</td>
<td>35.1</td>
<td>44.6</td>
<td>34.6</td>
</tr>
<tr>
<td>mgrid</td>
<td>65.1</td>
<td>91.5</td>
<td>93.5</td>
<td>91.3</td>
</tr>
<tr>
<td>swim</td>
<td>59.4</td>
<td>64.8</td>
<td>65.4</td>
<td>64.7</td>
</tr>
<tr>
<td>wupwise</td>
<td>71.9</td>
<td>90.3</td>
<td>71.2</td>
<td>87.9</td>
</tr>
<tr>
<td>Average</td>
<td>61.9</td>
<td>67.7</td>
<td>78.3</td>
<td>63.4</td>
</tr>
</tbody>
</table>
Register Access Time

Access Time (ps)

Number of Registers
OTHER CONSIDERATIONS (PAPER)

- Exception on instruction older than youngest PNR:
  - Roll back to checkpoint
- Each PNR recycles at its own pace
- Not all PNRs critical, e.g. $\text{PNR}_{\text{LDQ}}$
- Early register recycling:
  - Must count in-flight consumers (Moudgill et al 1993)
- Optimal interval between checkpoints
Cherry Parameters

- Cherry checkpoint frequency: $5\mu s$
- Checkpoint overhead: 60 cycles (18.75\text{ns})
- Avg time between exceptions: $800\mu s$ (not modeled)
Incorporating $U_b$

### Graph:

- **Y-axis**: Speedup
- **X-axis**: Benchmark Applications
- **Legend**:
  - Cherry
  - Unlimited
  - Pass Branches

- **Applications**:
  - bzip2
  - crafty
  - gcc
  - gzip
  - mcf
  - parser
  - perlbench
  - vortex
  - vpr
  - Average

- **Speedup Values**:
  - bzip2: 1.00 - 1.06
  - crafty: 1.12
  - gcc: 1.09
  - gzip: 1.18
  - mcf: 1.24
  - parser: 1.30
  - perlbench: 1.36
  - vortex: 1.00 - 1.12
  - vpr: 1.00 - 1.09
  - Average: 1.00 - 1.12
PERFECT BRANCH PREDICTION

![Graph showing speedup of various benchmarks with Cherry, Unlimited, and Pass Branches categories.]

- Cherry
- Unlimited
- Pass Branches

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Cherry</th>
<th>Unlimited</th>
<th>Pass Branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip2</td>
<td>1.12</td>
<td>1.30</td>
<td>1.18</td>
</tr>
<tr>
<td>crafty</td>
<td>1.26</td>
<td>1.36</td>
<td>1.18</td>
</tr>
<tr>
<td>gcc</td>
<td>1.18</td>
<td>1.30</td>
<td>1.18</td>
</tr>
<tr>
<td>gzip</td>
<td>1.12</td>
<td>1.30</td>
<td>1.18</td>
</tr>
<tr>
<td>mcf</td>
<td>1.26</td>
<td>1.36</td>
<td>1.18</td>
</tr>
<tr>
<td>parser</td>
<td>1.18</td>
<td>1.30</td>
<td>1.18</td>
</tr>
<tr>
<td>perlbench</td>
<td>1.12</td>
<td>1.30</td>
<td>1.18</td>
</tr>
<tr>
<td>vortex</td>
<td>1.12</td>
<td>1.30</td>
<td>1.18</td>
</tr>
<tr>
<td>vpr</td>
<td>1.12</td>
<td>1.30</td>
<td>1.18</td>
</tr>
<tr>
<td>Average</td>
<td>1.12</td>
<td>1.30</td>
<td>1.18</td>
</tr>
</tbody>
</table>
INTERRUPT COLLAPSING

Point of No Return

Head (oldest)
Interrupt

Point of No Return

Head (oldest)
Interrupt

Point of No Return & Head
## EQUIVALENT SYSTEM

<table>
<thead>
<tr>
<th>Resource</th>
<th>Baseline</th>
<th>SPECint2000</th>
<th>SPECfp2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDQ entries</td>
<td>32</td>
<td>80</td>
<td>128</td>
</tr>
<tr>
<td>STQ entries</td>
<td>32</td>
<td>64</td>
<td>112</td>
</tr>
<tr>
<td>Int Registers</td>
<td>192</td>
<td>240</td>
<td>240</td>
</tr>
<tr>
<td>FP Registers</td>
<td>128</td>
<td>128</td>
<td>192</td>
</tr>
</tbody>
</table>