CacheRewinder: Revoking Speculative Cache Updates Exploiting Write-Back Buffer

Jongmin Lee*, Junyeon Lee†, Taewon Suh*, Gunjae Koo*

*Korea University
†Samsung Electronics
Outline

- Backgrounds & Motivation
- Architecture of CacheRewinder
- Evaluation
- Conclusion
Attacks Exploiting Speculative Executions

Before Speculation

In Speculation

After Rolling-Back

Vulnerable!!
# Inefficiency of Previous Mitigations

<table>
<thead>
<tr>
<th>Software Approaches</th>
<th>Hardware Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Static</td>
<td>Delaying</td>
</tr>
<tr>
<td>• By a user</td>
<td>• Performance</td>
</tr>
<tr>
<td>• By a compiler</td>
<td>degradation by</td>
</tr>
<tr>
<td>• Significant</td>
<td>correct speculations</td>
</tr>
<tr>
<td>performance drop</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Additional buffer</td>
</tr>
<tr>
<td></td>
<td>for delayed data</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Restoring</td>
</tr>
<tr>
<td></td>
<td>• Performance</td>
</tr>
<tr>
<td></td>
<td>degradation by</td>
</tr>
<tr>
<td></td>
<td>restorations of</td>
</tr>
<tr>
<td></td>
<td>cache states</td>
</tr>
<tr>
<td></td>
<td>• Additional buffer</td>
</tr>
<tr>
<td></td>
<td>for data that may</td>
</tr>
<tr>
<td></td>
<td>be restored</td>
</tr>
</tbody>
</table>
The Solution

Restoring from an Existing Module Nearby the target

Write-Back Buffer

Storage

Performance
Write-Back Buffer

Response to the core

Response for load

Written-back to the lower-level cache
Underutilized Write-Back Buffer

- SPEC2006
- Sampled at every L1 D-Cache miss
- Average Occupancy : 0.4
- Only 5% are utilized for 8-entries-WBB

Average occupied entries

Underutilized
CacheRewinder

- **For Core**
  - Newly added STB (Speculation Tracking Buffer)
  - Modified LQ
- **R/C Doorbell**
- **For L1 D-Cache**
  - Modified WBB (as a RB)
  - Modified MSHR
  - Modified Tag Memory
Restore Operation against Spectre

L1 Cache

Probing

1 Hit!!!
2 Hit!!!
3 Hit!!!

SAFE

Restoring

MSHR

New

Response for speculative load

Restore Doorbell

sets

Primed

Hit!!!
Commit Operation

Commit Doorbell

L1 Cache

MSHR
Old
New

Response for speculative load

Written-back to the lower cache (or discarded)
Speculation Tracking Buffer (STB)

- **Valid Bit**
  - 1
  - 1

- **ROB ID**
  - 0x05
  - 0x0D

---

**1st Speculation Block**

**2nd Speculation Block**

---

**Order**

**Pointing**

---

**STB**

- **LQ**
  - **SPL**
    - 0
    - 0
    - 1
    - 1
  - **STB ID**
    - 0xXX
    - 0x00
  - **Status**
    - 0xABCC
    - 0xED00
    - 0x5678
    - 0x1234
    - 0x9000
  - **Addr**
    - 0x01
    - 0x03
    - 0x07
    - 0x0A
    - 0x0F
  - **ROB ID**
    - 0x01
    - 0x03
    - 0x07
    - 0x0A
    - 0x0F

---

**ROB**

- **0x00 ADD...**
- **0x01 LW ...**
- **0x02 ADDI ...**
- **0x03 LW ...**
- **0x04 AND ...**
- **0x05 BNE ...**
- **0x06 ADDI ...**
- **0x07 LW ...**
- **0x08 ADDI ...**
- **0x09 ORI ...**
- **0x0A LW ...**
- **0x0B ADDI ...**
- **0x0C AND ...**
- **0x0D BEQ ...**
- **0x0E AND ...**
- **0x0F LW ...**
Modification of L1 D-Cache

- **SPI**: installed by a speculative load
- **SPE**: evicted by speculative load
- **SPL**: checking speculative load miss
- **SQ**: canceling the in-flight case
- **Requested Tag**: the address of the replacer
- **Status Flags including dirty bit**: data to be restored

Diagram:

- Table with columns **Tags** and **Data**
- **SPI**: Status, Dirty, Tag, Data
- **SPE**: Status, Dirty, Requested Tag, Tag, Data
- **SPL** and **SQ** columns for MSHR

14/25
Speculative Load of Modified Cache
Restore Operation of Modified Cache

Restore Doorbell

<table>
<thead>
<tr>
<th>SPI</th>
<th>Status</th>
<th>Dirty</th>
<th>Tag</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Status</td>
<td>Dirty</td>
<td>0x1234</td>
<td>Data</td>
</tr>
<tr>
<td>1</td>
<td>Status</td>
<td>0</td>
<td>0x9000</td>
<td>Data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPE</th>
<th>Status</th>
<th>Dirty</th>
<th>Requested Tag</th>
<th>Tag</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Status</td>
<td>Dirty</td>
<td>0x9000</td>
<td>0x5678</td>
<td>Data</td>
</tr>
</tbody>
</table>

MSHR

<table>
<thead>
<tr>
<th>SPL</th>
<th>SQ</th>
<th>Addr</th>
</tr>
</thead>
</table>
Commit Operation of Modified Cache

Commit Doorbell

<table>
<thead>
<tr>
<th>C</th>
<th>Flags</th>
<th>Addr:0x9000</th>
</tr>
</thead>
</table>

Tags

<table>
<thead>
<tr>
<th>SPI</th>
<th>Status</th>
<th>Dirty</th>
<th>Tag</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Status</td>
<td>Dirty</td>
<td>0x1234</td>
<td>Data</td>
</tr>
<tr>
<td>0</td>
<td>Status</td>
<td>0</td>
<td>0x9000</td>
<td>Data</td>
</tr>
</tbody>
</table>

WBB

<table>
<thead>
<tr>
<th>SPE</th>
<th>Status</th>
<th>Dirty</th>
<th>Requested Tag</th>
<th>Tag</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Status</td>
<td>Dirty</td>
<td>0x9000</td>
<td>0x5678</td>
<td>Data</td>
</tr>
</tbody>
</table>

MSHR

<table>
<thead>
<tr>
<th>SPL</th>
<th>SQ</th>
<th>Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Evaluation

- **Gem5 with SE mode**
- **SPEC2006 benchmarks and modified Spectre PoC code**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>x86 ISA, out-of-order, no SMT, 64 IQ entries, 192 ROB entries, 32 LQ entries, 32 SQ entries, 32 STB entries</td>
</tr>
<tr>
<td>Branch Predictor</td>
<td>L-TAGE</td>
</tr>
<tr>
<td>L1-I cache</td>
<td>32 KB, 64B line, 8-way</td>
</tr>
<tr>
<td>L1-D cache</td>
<td>32 KB, 64B line, 8-way, 8 MSHR entries</td>
</tr>
<tr>
<td>L2 cache</td>
<td>256 KB, 64B line, 8-way</td>
</tr>
<tr>
<td>Write-back buffer</td>
<td>8 entries</td>
</tr>
<tr>
<td>Coherence protocol</td>
<td>MESI</td>
</tr>
</tbody>
</table>
Defense Evaluation

Average Latency (Cycles)

Array Index

secret:0x48

Unprotected

Protected by CacheRewinder

SAFE
Performance Comparison

- The performance overhead is negligible
- Exploiting nearby resources minimizes the restoration time
Performance w.r.t the Length of WBB

- Buffer-full of WBB stalls the execution speculative loads
  - To prevent new side-channels*
- It is only 0.2% of the difference overhead between 4-entries and infinite-entries-WBB

Utilization of WBB

- Average Occupancy: 1.4
- Increased by about 0.9 entries

Average occupied entries

Baseline Occp.  Increased Occp.
Storage Overhead

- Comparison for additional storage
- Focus on only Core and L1-D Cache, except L2 Cache
- Total overhead of CacheRewinder is minimum, on the same setting.

<table>
<thead>
<tr>
<th>Solutions</th>
<th>Core</th>
<th>L1-D Cache</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvisiSpec</td>
<td>28 B</td>
<td>2072 B</td>
<td>2100 B</td>
</tr>
<tr>
<td>CleanupSpec</td>
<td>224 B</td>
<td>56 B</td>
<td>280 B</td>
</tr>
<tr>
<td>ReViCe</td>
<td>24 B</td>
<td>2500 B</td>
<td>2524 B</td>
</tr>
<tr>
<td>CacheRewinder</td>
<td>60 B</td>
<td>131 B</td>
<td>191 B</td>
</tr>
</tbody>
</table>
Conclusion

• **CacheRewinder** – an efficient architectural defense solution
  • Exploiting WBB (write-back buffer) as a restore buffer
  • Implementing STB (speculation tracking buffer) for precise restore/commit operations

• **Extremely low performance/cost overhead**
  • Negligible performance overhead (0.6%)
  • Low storage overhead (191 Bytes)
Thank you

CacheRewinder: Revoking Speculative Cache Updates Exploiting Write-Back Buffer
Jongmin Lee, Junyeon Lee, Taeweon Suh, Gunjae Koo

✉ flackekd@korea.ac.kr