Timing Anomalies Reloaded

Gernot Gebhard
AbsInt Angewandte Informatik GmbH

Brussels, 6\textsuperscript{th} July, 2010
Timing Anomalies

• Intuition:

  *Local speed-up leads to global speed-down*

• Local worst-case need not imply global worst-case

• One of the main challenges of static timing analysis

• Triggered by interaction between several (average-case) performance-increasing hardware features
Timing Anomalies – Example

- Speculation-triggered timing anomaly:

```
Cache Hit: A -> Prefetch B - Cache Miss C
```

```
Cache Miss: A -> C
```

Branch condition evaluated
Timing Anomalies – Example cont'

- **Variable-execution-time-triggered** timing anomaly:
Classification of Timing Anomalies

- Reineke et al. discern three classes:
  - Scheduling timing anomaly
    \(\text{A faster task execution might lead to a globally longer schedule}\)
  - Speculation timing anomaly
    \(\text{An initial cache hit leads to more subsequent cache misses}\)
  - Cache timing anomaly
    \(\text{Caused by some non-LRU cache replacement policies (such as MRU)}\)
Timing Anomaly Types

• We discern two types of timing anomalies

  – **k-bounded** timing anomalies
    the effect on timing eventually stabilizes
    *(bounded by a constant k)*

  – **unbounded** timing anomalies (domino effects)
    the effect on timing never stabilizes
    *(further complicates static timing analysis)*
Coping with Timing Anomalies

• Depends on timing anomaly:
  – none present
    *Analysis may always assume local worst-case*
  – $k$-bounded timing anomaly
    *Trade precision against computational resources:*
    *Analysis may assume local worst-case and add constant $k$ to computed bound*
  – unbounded timing anomaly
    *Analysis has to explore all possible execution paths to provide safe bounds*
MRU Replacement Policy

- MRU = Most Recently Used
- Evicts the most recently used item upon a miss
- Useful when older data is likely to be reused
- Suffers from a domino effect
MRU Cache Set

- MRU cache set:

  2-way MRU cache set

  ![Diagram of a 2-way MRU cache set with two cache lines labeled a and b, one labeled most-recently used cache line and the other labeled least-recently used cache line.]
MRU Domino Effect

- Empty initial set, access sequence: \( ab(ab)^+ \)
MRU Domino Effect cont'

- Non-empty initial set, access sequence: $ab(ab)^+ \allowbreak$

![Diagram showing the MRU Domino Effect with non-empty initial set and access sequence $ab(ab)^+$]
MRU Domino Effect cont'

- Initially empty cache set:
  2 misses

- Initially non-empty cache set:
  $1 + 2n$ misses

- Difference in timing cannot be bounded by a constant $k$

  $\Rightarrow$ **MRU is prone to domino effects**
LEON2 Architecture

- Simple 5-stage pipeline
- In-order execution and completion
- No branch prediction mechanism
- Disjoint caches with LRU replacement policy
LEON2 Architecture cont'

- LEON2
- 5-Stage Integer Unit
- I-Cache
- D-Cache
- Memory Interface
I-Cache Line Filling Mechanism

- Cache line comprises 8 words

- Valid bit for each word in a cache line

- Cache line filling starts at critical word and stops and end of cache line (no wrap-around)

- Valid tags (words) are re-fetched

- Cache line filling might be interrupted
LEON2 Timing Anomaly

- Example triggering a speculation timing anomaly:

```
A: 0x00: ba 0x10
0x04: nop

B: 0x08: call <exit>
0x0c: nop

C: 0x10: nop
0x14: nop
0x18: ba 0x08
0x1c: nop
```

Legend:
- Basic block
- Cache line
LEON2 Timing Anomaly cont'

- Initial cache miss (basic block A)
  - Whole cache line is burst-filled
  - Code fetches in basic blocks B and C are hits
  - 8 memory accesses in total

- Access pattern:
LEON2 Timing Anomaly cont'

- Initial cache hit (basic block A)
  - Cache is potentially only partly filled
  - Code fetches in blocks B and C might be misses
  - 10 memory accesses in total

- Access pattern:
LEON2 Timing Anomaly Impact

Test I
Test II
Test III
Test IV
Test V
Test VI

WCET Bound (anomaly possible)
WCET Bound (no anomaly)
Analysis States (anomaly possible)
Analysis States (no anomaly)

Percent (%)

0
10
20
30
40
50
60
70
80
90
100
Conclusion

- Even seemingly simple architectures may suffer from timing anomalies
- The effect on timing is not to be neglected
- The list of non-LRU cache replacement policies featuring timing anomalies is completed
- More research required to fully understand timing anomalies