On-the-Fly Load Data Value Tracing in Multicores

Mounika Ponugoti, Amrish K. Tewar, and Aleksandar Milenković
ECE Department, The University of Alabama in Huntsville
Email: mp0046@uah.edu
Web: http://lacasa.uah.edu
Outline

- Background and Motivation
- Proposed Trace Module – mlvCFiat
- Experimental Environment
- Results
- Conclusions
Trends in Embedded Systems

- Society increasingly relies on embedded systems: communication, transportation, medicine, military, ...

- Forces shaping embedded systems
  - Technology trends: Integration ↑, Miniaturization ↑, Cost ↓
  - Application trends: Functionality ↑, Complexity ↑, Mobility ↑
  - Market trends: Proliferation ↑, Diversification ↑, Time-to-market ↓

- Implications
  - SW development cost exceeds 80% of the total cost
  - Developers spend 50%-75% of time in debugging; increases as we transition to multicores
  - Estimated cost of SW bugs and glitches: $20-$60 billion annually

- Need for better tools to help SW/HW developers find bugs faster
Tracing and Debugging Challenges

- What is my system doing now?
- Limited visibility of internal signals
  - High operating frequencies
  - High system complexity
  - Limited bandwidth for debugging
- Traditional approach to debugging
  - Typically done through a JTAG port:
    - stop the processor & examine or change the system state
  - Slow and expensive
  - May change the sequence of events
- Include dedicated on-chip trace and debug infrastructure
Tracing and Debugging in Multicores

- IEEE Nexus 5001 standard
- Class 1: Run-control debugging: run, stop, single-step, examine memory or register contents, set values
- Class 2: Captures control-flow traces in near real time
- Class 3: Captures data traces in near real time
- Class 4: Emulating memory and I/O through a trace port
Load Data Value Tracing

- Software debugger can replay program offline using:
  - Instruction set simulator
  - Program binary
  - Initial state of GPRs/SPRs
  - Exception traces and load data value traces
- For each memory read emit a trace message that includes: [Timestamp, Core ID, Load Data Value]
- Evaluate the average trace port bandwidth [TPB]: the number of bits traced through the trace port
- Metrics: the average bpi and bpc
  - Average bpi = \frac{\text{Total trace size in bits}}{\text{Total number of executed instructions}}
  - Average bpc = \frac{\text{Total trace size in bits}}{\text{Execution time measured in clock cycles}}
Data Tracing Challenges

(a) Trace Port Bandwidth [bpi, bits per instruction]

(b) Trace Port Bandwidth [bpc, bits per clock cycle]

Total TPB [bpi]

- N = 1: 12.34
- N = 2: 12.63
- N = 4: 12.89
- N = 8: 13.17

Total TPB [bpc]

- N = 1: 4.92
- N = 2: 8.76
- N = 4: 15.61
- N = 8: 25.64
mlvC Fiat

- **mlvC Fiat** – multicore load value cache first access tracking
  - Hardware/software mechanism for filtering load data value traces
- Target platform: L1 data caches are augmented to include first-access tracking bits (FA bits)
- Software debugger maintains software copies of L1 data caches
  - Same updating policies, same cache organization as in hardware
  - Instruction set simulator requires load data values from the target platform only for reads that cannot be inferred
mlvCFiat Hardware Structures

- First-access tracking bits track whether a particular sub-block is accessed for the first time
  - Previously reported sub-blocks are not reported again
  - Hardware overhead depends on granularity
- fahCnt: First Access Hit Counter
- Logic for encoding time stamps
Operations On Target Platform

- Ti: Memory Read
  - Ti: Cache Lookup
    - HIT?
      - Y
        - Corresponding FA Bits Set?
          - Y
            - Replace Cache Block
              - Replace Cache Block
                - Clear FA Bits
                  - Ti.fahCnt++
                    - Emit Trace Msg.
                      - [Ti.dCC, Ti, Ti.fahCnt, Ti.LV]
                        - Set Corresponding FA Bits
                          - Ti.fahCnt = 0
                          - END
                      - END
                    - Ti.fahCnt++
                      - Emit Trace Msg.
                        - [Ti.dCC, Ti, Ti.fahCnt, Ti.LV]
                          - Set Corresponding FA Bits
                            - Ti.fahCnt = 0
                            - END
                  - N
                    - Clear FA Bits
                      - Replace Cache Block
                        - Clear FA Bits
                          - Ti.fahCnt = 0
                          - END
                    - END
                - N
                  - Replace Cache Block
                    - Clear FA Bits
                      - Ti.fahCnt = 0
                      - END
                - END
            - N
              - Corresponding FA Bits Set?
                - Y
                  - Replace Cache Block
                    - Clear FA Bits
                      - Ti.fahCnt = 0
                      - END
                - N
                  - Replace Cache Block
                    - Clear FA Bits
                      - Ti.fahCnt = 0
                      - END
          - N
            - Corresponding FA Bits Set?
              - Y
                - Replace Cache Block
                  - Clear FA Bits
                    - Ti.fahCnt = 0
                    - END
              - N
                - Replace Cache Block
                  - Clear FA Bits
                    - Ti.fahCnt = 0
                    - END
      - N
        - Ti: Memory Read
          - Ti: Cache Lookup
            - Hit?
              - Y
                - Ti: Memory Write
                  - Ti: Cache Lookup
                    - Hit?
                      - Y
                        - Replace Cache Block
                          - Clear FA Bits
                            - Acquire Ownership
                              - Update The Cache
                                - Set Corresponding FA
                                  - END
                      - N
                        - Replace Cache Block
                          - Clear FA Bits
                            - Acquire Ownership
                              - Update The Cache
                                - Set Corresponding FA
                                  - END
                      - Ti: Memory Write
                        - Ti: Cache Lookup
                          - Hit?
                            - Y
                              - Replace Cache Block
                                - Clear FA Bits
                                  - Acquire Ownership
                                    - Update The Cache
                                      - Set Corresponding FA
                                        - END
                            - N
                              - Replace Cache Block
                                - Clear FA Bits
                                  - Acquire Ownership
                                    - Update The Cache
                                      - Set Corresponding FA
                                        - END
                      - Ti: Memory Write
                        - Ti: Cache Lookup
                          - Hit?
                            - N
                              - Replace Cache Block
                                - Clear FA Bits
                                  - Acquire Ownership
                                    - Update The Cache
                                      - Set Corresponding FA
                                        - END
                      - Ti: Memory Write
                        - Ti: Cache Lookup
                          - Hit?
                            - Y
                              - Replace Cache Block
                                - Clear FA Bits
                                  - Acquire Ownership
                                    - Update The Cache
                                      - Set Corresponding FA
                                        - END
                            - N
                              - Replace Cache Block
                                - Clear FA Bits
                                  - Acquire Ownership
                                    - Update The Cache
                                      - Set Corresponding FA
                                        - END
                          - N
                            - Replace Cache Block
                              - Clear FA Bits
                                - Acquire Ownership
                                  - Update The Cache
                                    - Set Corresponding FA
                                      - END
                          - Ti: Memory Write
                            - Ti: Cache Lookup
                              - Hit?
                                - Y
                                  - Replace Cache Block
                                    - Clear FA Bits
                                      - Acquire Ownership
                                        - Update The Cache
                                          - Set Corresponding FA
                                            - END
                                  - Replace Cache Block
                                    - Clear FA Bits
                                      - Acquire Ownership
                                        - Update The Cache
                                          - Set Corresponding FA
                                            - END
                                - N
                                  - Replace Cache Block
                                    - Clear FA Bits
                                      - Acquire Ownership
                                        - Update The Cache
                                          - Set Corresponding FA
                                            - END
                          - Ti: Memory Write
                            - Ti: Cache Lookup
                              - Hit?
                                - N
                                  - Replace Cache Block
                                    - Clear FA Bits
                                      - Acquire Ownership
                                        - Update The Cache
                                          - Set Corresponding FA
                                            - END
                          - Ti: Memory Write
                            - Ti: Cache Lookup
                              - Hit?
                                - Y
                                  - Replace Cache Block
                                    - Clear FA Bits
                                      - Acquire Ownership
                                        - Update The Cache
                                          - Set Corresponding FA
                                            - END
                - N
                  - Replace Cache Block
                    - Clear FA Bits
                      - Acquire Ownership
                        - Update The Cache
                          - Set Corresponding FA
                            - END
            - N
              - Replace Cache Block
                - Clear FA Bits
                  - Acquire Ownership
                    - Update The Cache
                      - Set Corresponding FA
                        - END
Operations in Software Debugger

- **Ti: Memory Read**
  - Ti.fahCnt--
  - Ti.fahCnt > 0?
    - Y: Read n Bytes From Trace Msg. Update SW Cache Get New Trace Msg. Load Ti.fahCnt
    - N: END
  - END

- **Ti: Memory Write**
  - Ti: Cache Lookup
  - HIT?
    - Y: Acquire Ownership Update The SW Cache Set Corresponding FA Bits
    - N: Clear FA Bits
  - END

- **External Invalidation**
  - Invalidate The Cache Block Clear FA Flags
  - END

---

**Background & Motivation**

- **Experimental Environment**
- **Results**
- **Conclusions**

**Ti**: Cache Lookup

**Ti**: Memory Write

**HIT?**

**N**: Clear FA Bits

**Y**: Acquire Ownership Update The SW Cache Set Corresponding FA Bits

**END**

**Ti.fahCnt > 0?**

**N**: Read n Bytes From Trace Msg. Update SW Cache Get New Trace Msg. Load Ti.fahCnt

**Y**: Lookup SW Cache Get Data From SW Cache

**END**
Encoding of Trace Messages

- In mlvCFiat length of LV field depends on granularity
- Encoding parameters \((h_0, h_1) = (4, 2), (i_0, i_1) = (2, 2)\)

(a) Nexus-like encoding (NX_b)

Legend:
- \(dCC\) Clock Cycle (differential enc.)
- \(Ti\) Thread/Core ID \(- \lceil \log_2 N \rceil \) bits
- \(LV\) Load Value
- \(fahCnt\) First Access Hit Counter
- \(h_0, h_1\) Chunk Sizes for CC
- \(i_0, i_1\) Chunk Sizes for fahCnt

(b) mlvCFiat baseline encoding (CF_b)

(c) mlvCFiat variable encoding (CF_e)
Experimental Environment

TmTrace: Software Timed Trace Generator

- Performance Statistics
- Application Output

Multi2Sim
- Multi2Sim Configuration Files
- TmTrace Flags

32 bit Target Application
- Application Input
- Number Of Threads

TmTrace
- mlvCFiat Simulator
- mlvCFiat Configuration

Hardware traces
- Trace Filtering
  - Fixed Encoding
  - Variable Encoding
- Fixed Encoding
- Variable Encoding

mlvCFiat Simulator
- mlvCFiat Trace

Background&Motivation | mlvCFiat | Experimental Environment | Results | Conclusions
Multicore Model

CS16:
- L1D/L1I cache size: 16 KB
- L2 cache size: N*64 KB

CS32:
- L1D/L1I cache size: 32 KB
- L2 cache size: N*128 KB

L1D/L1I hit time: 4 cc
- L1D/L1I associativity: 4-way
- L2 hit time: 12 cc
- L2 associativity: 16-way
- Cache block size: 32 B
- First-access granularity: 4 B
- Memory latency: 100 cc
Trace Port Bandwidth [bpi]

- **NX_b (CS16):**
  - 12.34 bpi (N=1)
  - 13.17 bpi (N=8)

- **mlvCFiat:**
  - CF_b(CS16) – 0.88 bpi (N=1), 1.0 bpi (N=8)
  - CF_b(CS32) – 0.40 bpi (N=1), 0.71 bpi (N=8)
  - CF_e(CS16) – 0.80 bpi (N=1), 0.92 bpi (N=8)
  - CF_e(CS32) – 0.37 bpi (N=1), 0.66 bpi (N=8)
Where Do Bits Go?

- LV field requires 68 – 78%, dCC requires 13 – 16%, fahCnt 8 – 15%, Ti 0-5% of total trace port bandwidth
### Trace Port Bandwidth [bpc]

#### NX_b (CS16)
- 4.92 bpc (N=1)
- 25.64 bpc (N=8)

#### mlvCFiat:
- CF_b(CS16) – 0.35 bpc (N=1), 1.95 bpc (N=8)
- CF_b(CS32) – 0.17 bpc (N=1), 1.40 bpc (N=8)
- CF_e(CS16) – 0.32 bpc (N=1), 1.79 bpc (N=8)
- CF_e(CS32) – 0.16 bpc (N=1), 1.29 bpc (N=8)
## Speedup (Trace Reduction Ratio)

<table>
<thead>
<tr>
<th># Cores</th>
<th>N=1</th>
<th>N=2</th>
<th>N=4</th>
<th>N=8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mech.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NX_b.gz</td>
<td>CF_e</td>
<td>NX_b.gz</td>
<td>CF_e</td>
<td>NX_b.gz</td>
</tr>
<tr>
<td>Config</td>
<td>Unif</td>
<td>Split</td>
<td>CS16</td>
<td>CS32</td>
</tr>
<tr>
<td>barnes</td>
<td>1.4</td>
<td>2.1</td>
<td>6.8</td>
<td>19.1</td>
</tr>
<tr>
<td>cholesky</td>
<td>1.7</td>
<td>6.7</td>
<td>8.2</td>
<td>20.4</td>
</tr>
<tr>
<td>fft</td>
<td>1.4</td>
<td>1.9</td>
<td>4.2</td>
<td>7.2</td>
</tr>
<tr>
<td>fmm</td>
<td>1.9</td>
<td>4.9</td>
<td>24.2</td>
<td>37.8</td>
</tr>
<tr>
<td>lu</td>
<td>1.6</td>
<td>5.9</td>
<td>20.5</td>
<td>20.7</td>
</tr>
<tr>
<td>radiosity</td>
<td>1.6</td>
<td>3.9</td>
<td>47.7</td>
<td>128.8</td>
</tr>
<tr>
<td>radix</td>
<td>2.0</td>
<td>4.2</td>
<td>17.9</td>
<td>24.8</td>
</tr>
<tr>
<td>raytrace</td>
<td>1.5</td>
<td>3.9</td>
<td>14.3</td>
<td>44.2</td>
</tr>
<tr>
<td>water-ns</td>
<td>1.4</td>
<td>2.7</td>
<td>21.7</td>
<td>47.4</td>
</tr>
<tr>
<td>water-sp</td>
<td>1.4</td>
<td>3.0</td>
<td>168.1</td>
<td>210.3</td>
</tr>
<tr>
<td>Total</td>
<td>1.5</td>
<td>3.3</td>
<td>15.3</td>
<td>33.4</td>
</tr>
</tbody>
</table>

- Unified – NX_b as input to gzip
- Split – Two streams (dCC, Ti), (LV) compressed separately
Dynamic Trace Port Bandwidth Analysis

**raytrace: Trace port bandwidth in bpc as a function of time**

<table>
<thead>
<tr>
<th></th>
<th>NX_b(CS16)</th>
<th>CF_e(CS32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg (bpc)</td>
<td>42.7</td>
<td>2.6</td>
</tr>
<tr>
<td>Peak (bpc)</td>
<td>61.5</td>
<td>4.9</td>
</tr>
</tbody>
</table>

**water-ns: Trace port bandwidth in bpc as a function of time**

<table>
<thead>
<tr>
<th></th>
<th>NX_b(CS16)</th>
<th>CF_e(CS32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg (bpc)</td>
<td>43.5</td>
<td>1.60</td>
</tr>
<tr>
<td>Peak (bpc)</td>
<td>56.4</td>
<td>6.0</td>
</tr>
</tbody>
</table>
Conclusions

- Need for trace modules that can guarantee
  - Unobtrusive program tracing in real-time
  - High compression (low trace port bandwidth)
  - Low complexity: narrow trace ports and small trace buffers

- mlvCFiat - multicore load value Cache First-access tracking mechanism for filtering load data values
  - Trace out only data cache read misses or data cache read hits that occur for the first-time
  - Relatively low-complexity: only storage for first-access bits
  - Significant reduction in load data value trace: from 15 to 33 times when N = 1 and from 14 to 20 times when N = 8

- Variable encoding mechanism
  - Reduces the trace port bandwidth 8-9% relative to base encoding