CADRE: Cycle-Accurate Deterministic Replay for Hardware Debugging

Smruti R. Sarangi
Brian L. Greskamp
Josep Torrellas

University of Illinois Urbana Champaign

http://iacoma.cs.uiuc.edu
Motivation

- 50-70% effort spent on verification
- 1-2 year verification times
- More features on-chip
- Verification speed not keeping up with complexity
- Some bugs inevitably slip through
  - Pentium fdiv bug
  - Prefetching bug in Pentium 4 Xeon
  - IBM G-3 frequency bug
Error Rate vs Time

- Reduced time to debug → Vital ingredient of profitability
Outline

- Problems in Debugging
  - Sources of Non-Determinism
  - Handling Non-Determinism in Buses
- CADRE Architecture
- Evaluation
  - Space Overhead
  - Performance Overhead
Design Bugs

- An example of a design bug in IBM G3
  - Power manager shuts down L1
  - AND L1 is waiting for data
  - AND L2 is being invalidated
  - ALL the L2 lines might not be invalidated

- Two Features
  - Infrequent conditions
  - Large detection latency — Data corruption bugs are detected only after an observable event: program crash, HW hang, wrong output, etc.
Problems in Debugging Hardware

- Infrequent conditions
  - RTL simulators are very slow, roughly 30 cyc/s
  - Need to test as many paths as possible
  - At breakpoints transfer state to RTL simulator
  - Probably want to put it on the field and send bug reports back

- Large detection latency
  - Large debugging window & modest storage

- Some bugs are NOT reproducible
Existing Debugging Frameworks

- Pentium-M debugging framework – Golan
  - Log all the signals at the pins
  - Replay them

- Disadvantages
  - Expensive pin snooping electronics
  - For very high speed buses
    - Hard to snoop anymore
    - Log signals inside the processor
    - Extra pins required to send data to stable storage
Agent: CMP, MCH, … . Each agent has its own clock.

An input/output is *deterministic* if it is always observed at the same clock cycle w.r.t the agent.

An agent is *deterministic* if deterministic inputs imply deterministic outputs.
Ideal Hardware Debugger

- High speed execution till the “buggy point”
- Minimal storage & large debugging window
- Executions are completely reproducible

<table>
<thead>
<tr>
<th></th>
<th>RTL Simulator</th>
<th>Golan</th>
<th>CADRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Very Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Storage</td>
<td>Very High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Debugging Window</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Reproducibility</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>
Outline

- Problems in Debugging
  - Sources of Non-Determinism
  - Handling Non-Determinism in Buses
- CADRE Architecture
- Evaluation
  - Space Overhead
  - Performance Overhead
Sources of Non-Determinism

- DIMMs
- CMP chip
- Memory Controller (MCH)
- IO Controller (ICH)

Non-deterministic message delay

- To IO devices
- IO and Interrupts

- Power/Thermal Events
- Voltage Freq. Scaling

- Soft Errors
- Refresh/Scrubbing

Smruti R. Sarangi
Non-Determinism in Buses

The probability of non-determinism is very high for buses in the future.

Smruti R. Sarangi
Enforcing Determinism

- **CPU**
  - Design Deterministically
  - Properly initialize all elements
  - DETRST instruction (to set a deterministic state)
  - Log all exceptions, power/thermal events

- **Memory**
  - Start every checkpoint interval with a refresh
  - Checkpoint scrub register

- **IO**
  - Log all the data along with cycle counts
Enforcing Determinism in Buses

Transmitter

Receiver

\[ x_T \]

\[ y_R \]

Deterministic Processing

Non-deterministic Processing

Uncertainty Interval

\[ W = \theta_2 - \theta_1 + 1 \]

\[ \log_2(W) \]

Optimal

\[ z_R = x_T + \theta_2 \]
Schemes to Enforce Determinism

- Assume $f_{\text{Transmitter}} = f_{\text{Receiver}}$
- Receiver needs to compute: $z_R = x_T + \theta_2$
- Trivial Solution: Send $x_T$ along with every message.
- Better Solution: After first message is delivered deterministically
  - Transmitter sends a message every cycle
  - Receiver then processes the messages at the rate of transmission
- Disadvantages
  - This scheme requires a line between all pairs of nodes
  - A receiver has to be aware of all the transmitters
Offset Scheme

\[ \begin{align*}
\min(x_T) & \quad \max(x_T) \\
\theta_2 - \theta_1 & \quad \theta_1 & \quad \theta_2 \\
\text{Disjoint} & \quad \text{Case 1} & \quad \text{Case 2}
\end{align*} \]

- \((y_R - \theta_1 - \rho)\) is in the same window as \(x_T\)
- \(x_T = \left\lfloor (y_R - \theta_1 - \rho)/W \right\rfloor \times W + \rho\)
Implementation of Offset Scheme

\[ z_R = \left\lfloor \frac{y_R - \theta_1 - \rho}{W} \right\rfloor W + \rho + \theta_2 \]
Architecture

- CMP
  - CPU
  - Synchronizer
- Memory Log
  - Reg. Ckpt
- Synchronizer
- Memory Controller
- CADRE Controller
- IO Log
- IO Controller Hub
- PCI devices

- Synchronous Bus
- Source Sync. Bus
- Asynchronous Bus

- Baseline
- HW for determinism
- HW for checkpointing

Smruti R. Sarangi
Outline

- Problems in Debugging
  - Sources of Non-Determinism
  - Handling Non-Determinism in Buses
- CADRE Architecture
- Evaluation
  - Space Overhead
  - Performance Overhead
Evaluation – CADRE

- Configuration
  - 2.8 GHz dual proc. Pentium-4 Xeon server with hyper-threading
  - Intel E7525 chipset with 800 MHz FSB
  - We estimate the overheads of a 4-processor CMP
  - Benchmarks – Spec: Int, FP, JBB, OMP and Web

- To estimate overheads
  - Reconfigure MCH and ICH
  - Use memory mapped IO to access PCIX registers
  - See our paper in WARP ’06 (workshop along with ISCA ’06)
Space Overhead

- Space overhead of mem. checkpointing
  - SafetyNet: 50MB/s/proc
  - Revive: 38-120 MB/s/proc
  - Average IO bandwidth – 100 kB/s

- Design Point
  - 1 sec checkpoint interval
  - 50 MB * 4 = 200 MB for memory ckpt.
  - 4 MB log for IO traffic
Performance Overhead

- Periodic cache flushing overhead for proc. checkpointing.
  - negligible — once per second
- For every message assume worst case delay → increase bus latency
  - assume 1-4 (MCH) cycles of non-determinism
  - add an extra 1-4 cycles of bus latency
  - Increase programmable read pointer delay, clock guard band, RAS-CAS delay
With a 1 sec.ckpt. interval, CADRE has a 1% slowdown and requires 200 MB of storage.

As compared to 64 ms for Golan for same overheads.
Using CADRE

- **Hardware debugging**
  - Record and replay executions
  - Transfer state to an RTL simulator
  - Use scan chains to observe certain latches
  - Use CADRE in deployed systems in the field
    - Send a hardware core dump back to the vendor

- **Lock-stepped execution in TMR systems**

- **Software debugging – CADRE hardware guarantees determinism**
CADRE: Cycle-Accurate Deterministic Replay for Hardware Debugging

Questions?

http://iacoma.cs.uiuc.edu