Bridging Pre- and Post-silicon Debugging with BiPeD

Andrew DeOrio
Jialin Li and Valeria Bertacco

University of Michigan
Verification Opportunities

Pre-Silicon
- Low speed
+ High observability
+ Reproducible bugs

Post-Silicon
+ High speed
- Poor observability
- Intermittent bugs

little information sharing
Verification Opportunities

Pre-Silicon

Post-Silicon

High observability →
learn correct behavior

High speed →
enforce correct behavior

Shared correctness model
Contributions

- High speed
- High observability, detailed debugging info
- No need for bug reproduction

Shared correctness model
BiPeD Overview

Pre-silicon
- Run correct tests
- Monitor interfaces
- Learn correct protocols

Post-silicon
- Transfer debug data off-chip
- Extract debugging information

Protocol extraction
- Run correct tests
- Monitor interfaces
- Learn correct protocols

Protocol detection
- Run many unknown tests
- HW detects protocols
- Detect errors in protocols

Transaction extraction
- Transfer debug data off-chip
- Extract debugging information
BiPeD Overview

1. Pre-silicon protocol extraction
2. Post-silicon protocol detection
3. Offline transaction extraction
Pre-silicon Protocol Extraction

Pre-silicon Tests

Design Under Test

Simulation

protocol extraction

select interface signals to analyze

Protocol diagram:
Describes interface behavior

"INFERNO: Streamlining Verification with Inferred Semantics", DeOrio, et. al, 2009
Pre-silicon Protocol Extraction

- protect
- thread sync
- TLB bypass
- ASI reload
- flush

protocol diagram

01100

transition

00000

event

00010

00100

00101
TLU Protocol Example

bit 0: protect
bit 1: thread sync
bit 2: TLB bypass
bit 3: ASI reload
bit 4: flush
1. Pre-silicon protocol extraction
2. Post-silicon protocol detection
3. Offline transaction extraction
Post-silicon Protocol Detection

load protocols into programmable HW

post-si tests

run high-coverage post-silicon tests

Error!

only stop when error is detected
Post-silicon protocol detection

detect multiple protocols simultaneously

monitored interface

protocol detector

circular buffer

valid event
valid transition
error out

valid event

history out

event history

current event

previous event

detect multiple protocols simultaneously

test platform

...
Protocol detector hardware

- Programmable
- Circular history buffer

Diagram:

- Event history check
- Transition history check
- Valid event
- Valid transition
- Error out
- History out
- Monitored interface
- Priority encoder
- Event CAM
- Transition CAM
Area overhead

- **0.7%** of OpenSPARC T2 for 10 detectors
  - 15.3KB storage each, for biggest OST2 protocol

10 protocols

33 bits x 62 events

622 transitions

1,024 events
TLU Protocol Example

• Injected bug in OpenSPARC TLU/LSU interface
  – Cycle 10,000
• Programmed TLU/LSU protocol into detector
• Ran test
• BiPeD HW detected bug at cycle 10,017
1. Pre-silicon protocol extraction
2. Post-silicon protocol detection
3. Offline transaction extraction
Off-line transaction extraction

Test platform

Post-si Tests

protocol detector

circular buffer

transfer off-chip

transaction extraction

module testbench

initial

begin

clock = 0;
#5 clock = 1;
end
Transaction extraction

• Leverage transaction extraction similar to Inferno [DeOrio, et. al, 2009]

• Input: circular event buffer

• Output: intuitive, high-level transactions

thread sync

TLB bypass

burst TLB bypass w/ thread sync
TLU Protocol Example

bit 0: protect
bit 1: thread sync
bit 2: TLB bypass
bit 3: ASI reload
bit 4: flush

SPARC core
TLU
interface
LSU

- 01100: burst TLB bypass w/sync
- 00010: address reload
- 00000: TLB bypass
- 00100: TLB bypass w/flush
- 00101:
Transaction extraction example

Extracted transaction history

thread sync
burst TLB bypass
W/B bypass
TLB bypass
thread sync

3,694-3,732
4,492-4,531
4,539-4,543
4,545-4,602
cycle

4,609 – 10,017
Transaction extraction example

• **Time:** cycle 10,017  
• **Interface:** TLU  
• **Signals:** protect, thread sync, TLB bypass, ASI reload, flush  
• **Preceding activity:** thread sync, burst TLB bypass w/thread sync, TLB bypass, TLB bypass  
• **Event:** 10100  
  **Transition:** 00100 -> 10100  
• **Transaction:**

!!buggy transition!!
Limitations

• False negatives
  – May miss bugs that only affect data signals
  – Interface signal selection important
    • Control signals work well in practice

• False positives
  – High pre-silicon coverage → fewer false positives
  – If f.p. is encountered, update the database
Experimental setup

1,000 passing runs

100 random seeds: variable memory delay, crossbar random traffic

100 buggy runs

10 testcases

10 bugs: e.g., functional bug in PCX, fetch thread ID

10 interfaces

BiPeD HW

BiPeD SW

detected transactions
# Signal Localization

<table>
<thead>
<tr>
<th>Interfaces</th>
<th>branch</th>
<th>EX valid inst.</th>
<th>cache-proc</th>
<th>MEM rd ack</th>
<th>FPU execept.</th>
<th>fetch thread</th>
<th>LSU access</th>
<th>table walk</th>
<th>PCX stall</th>
<th>CCX/PCX req</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPX</td>
<td>1,719</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>branch</td>
<td>242</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCX</td>
<td>16k</td>
<td>39</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>742</td>
</tr>
<tr>
<td>memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>execute</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPU</td>
<td>f.p.</td>
<td>22k</td>
<td>48k</td>
<td>739</td>
<td>48k</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fetch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>perf.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Green** indicates the first interface to find a bug.
- **Red** indicates a false positive (f.p.).
- **Orange** indicates a false negative (f.n.).
Protocol Extraction

Testcase and total number of test executions
Transaction Extraction

Number of transactions vs. Circular buffer size (entries)

- Total transactions
- Unique transactions

- 0.1 KB
- 4 KB
Leave-one-out Cross Validation

False positives (percent)

Omitted testcase

- blimp_rand
- fp_addsub
- fp_muldiv
- isa2_basic
- isa3_asr_pr
- isa3_window
- ldst_sync
- mpgen_smc
- n2_lsu_asl
- tlu_rand
Related Work

• Invariant detection
  [Ammons 2002, Ernst 2008]
  – Detect invariants
  – Check tests against invariants

Pre-silicon verification
  – Inferno: verification with transactions [DeOrio 2009]
  – Data mining high-level specifications [Li 2010]

Post-silicon validation
  – Manual debugging [Abramovici 2006]
  – Automated debugging of specific components [Park 2011]
  – Manual, hardcoded txn checkers [Singerman 2011]
Conclusions and Future Work

- BiPeD bridges pre-silicon protocol extraction with post-silicon detection
- Automatically detects bugs
- Provides intuitive debugging information

- Future applications for flexible hardware
  - Coverage metrics
  - Runtime verification