Bridging Pre- and Post-silicon Debugging with BiPeD

Andrew DeOrio

Jialin Li and Valeria Bertacco

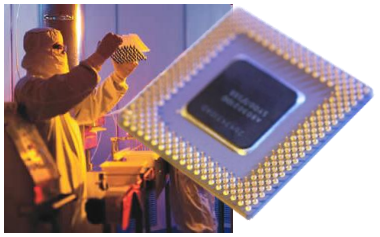
University of Michigan

Verification Opportunities

Pre-Silicon

- Low speed
+ High observability
+ Reproducible bugs

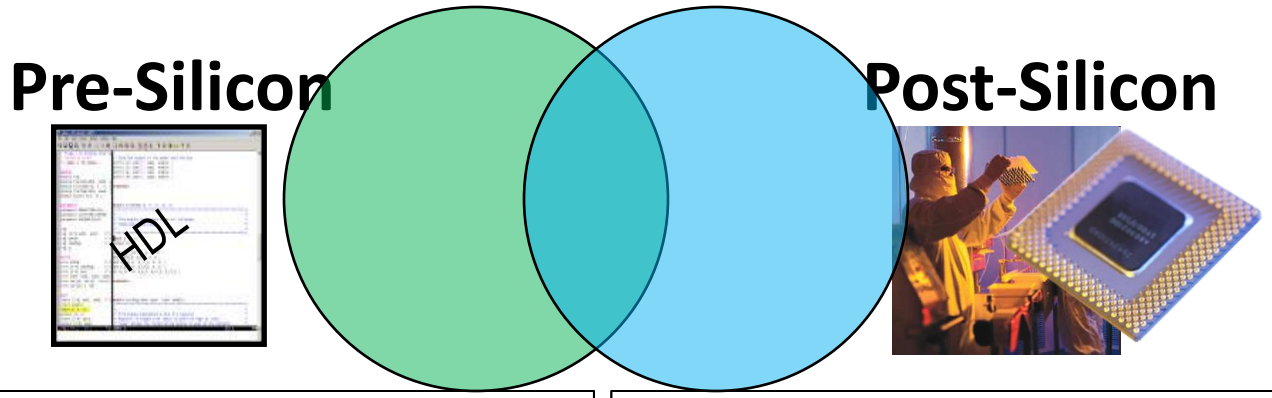
Post-Silicon



+ High speed
- Poor observability
- Intermittent bugs

little information sharing

Verification Opportunities

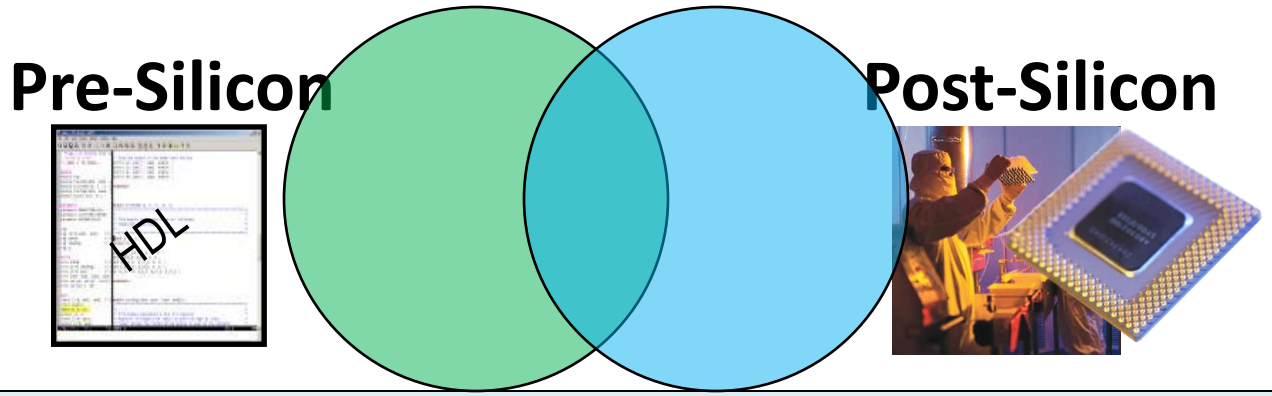


High observability → **learn correct behavior**

High speed → **enforce correct behavior**

Shared correctness model

Contributions




High speed


High observability, detailed debugging info


No need for bug reproduction

Shared correctness model

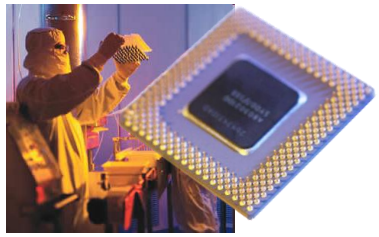
BiPeD Overview

Pre-silicon

Post-silicon

on-line

off-line



Protocol extraction

- Run correct tests
- Monitor interfaces
- Learn correct protocols

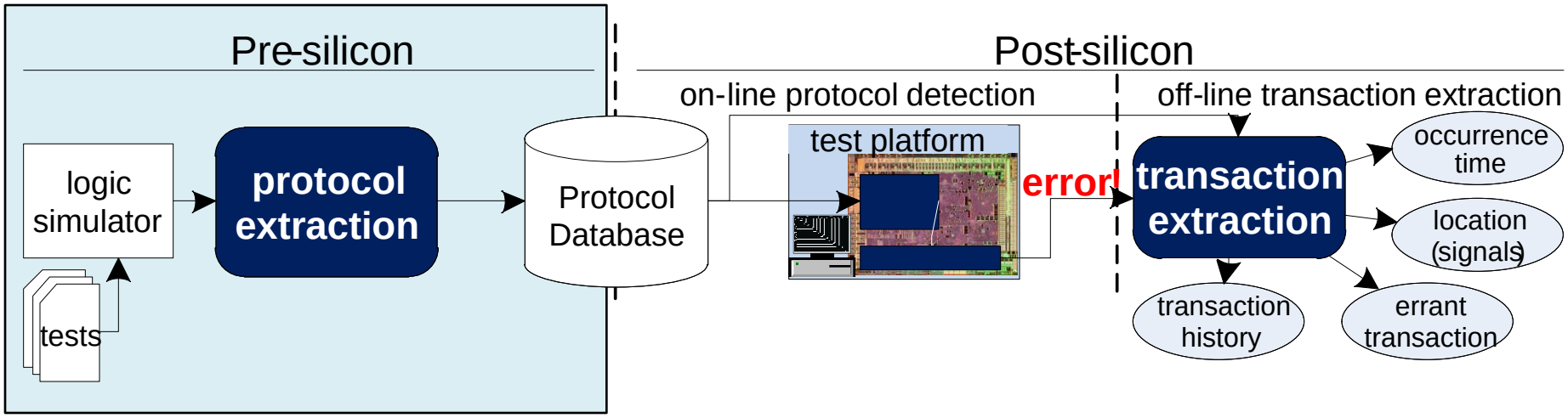
Protocol detection

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

Transaction extraction

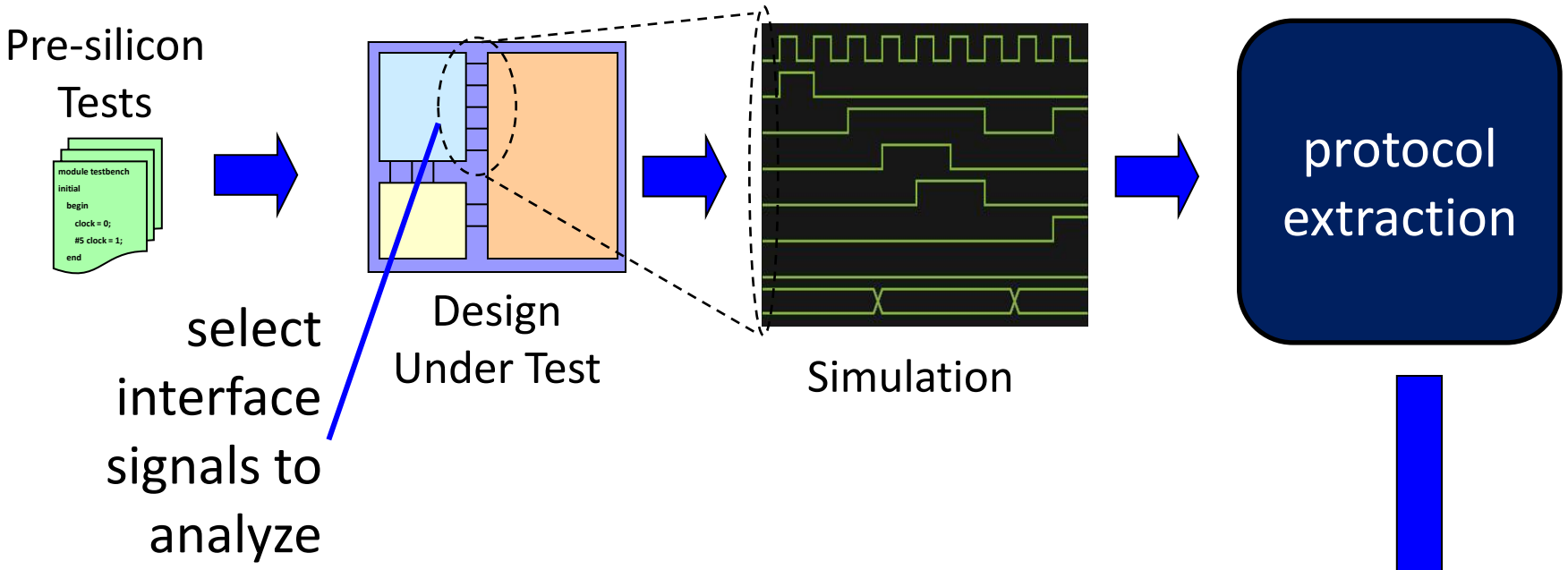
- Transfer debug data off-chip
- Extract debugging information

BiPeD Overview

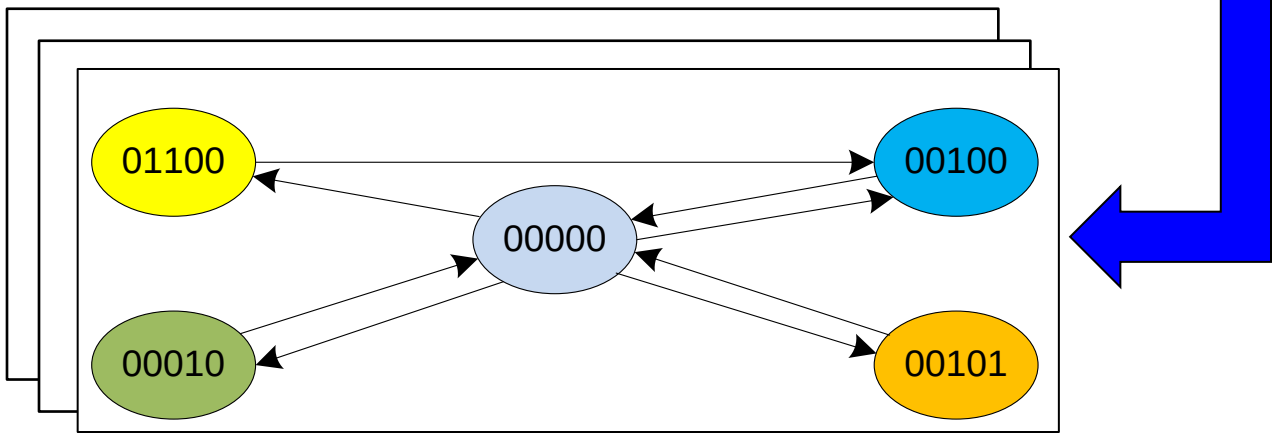


1. Pre-silicon protocol extraction
2. Post-silicon protocol detection
3. Offline transaction extraction

Pre-silicon Protocol Extraction

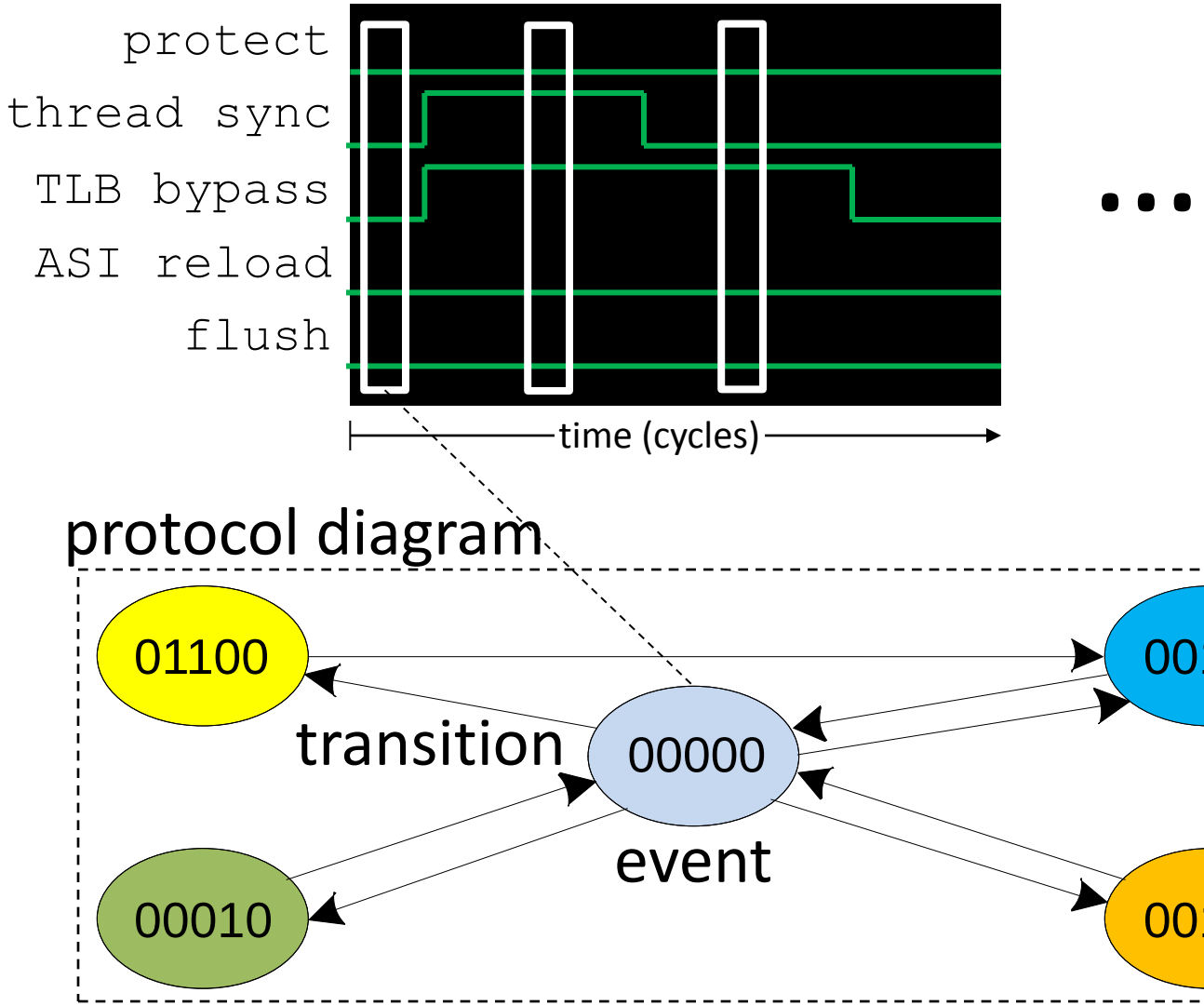


protocol diagram: describes interface behavior

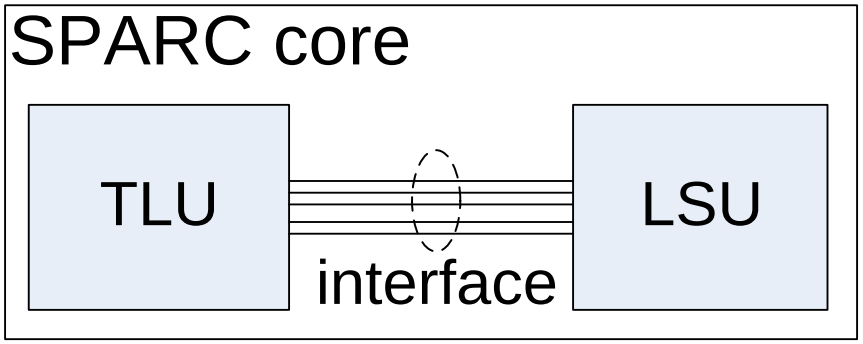


"INFERNO: Streamlining Verification with Inferred Semantics", DeOrio, et. al, 2009

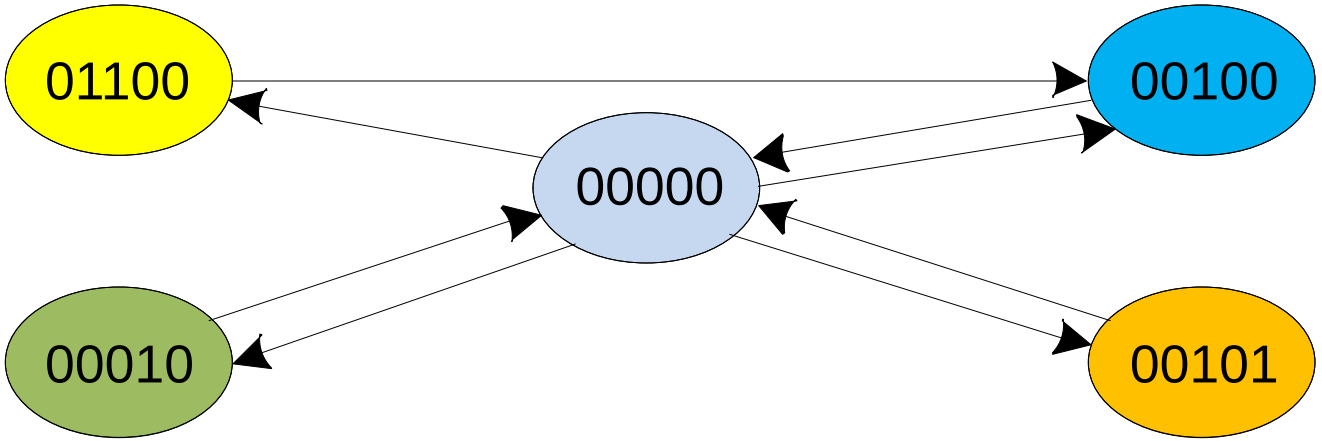
Pre-silicon Protocol Extraction



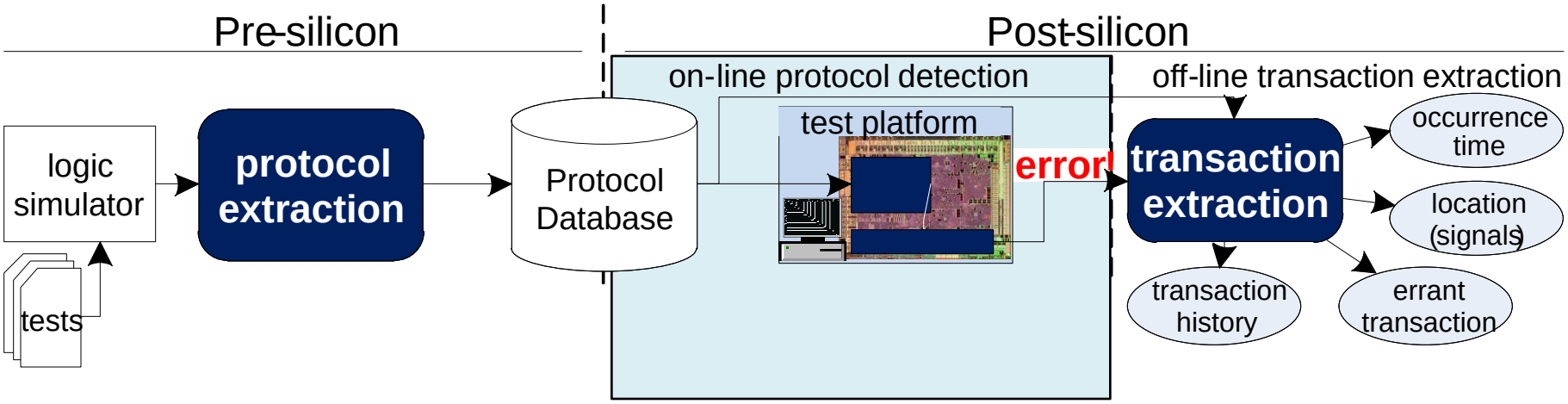
TLU Protocol Example




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

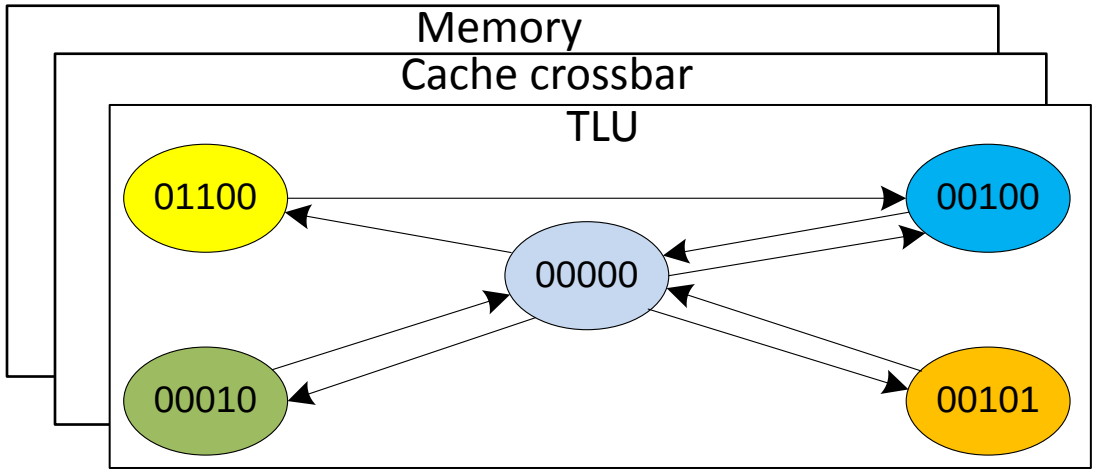


Outline



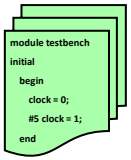
1. Pre-silicon protocol extraction
2. Post-silicon protocol detection
3. Offline transaction extraction

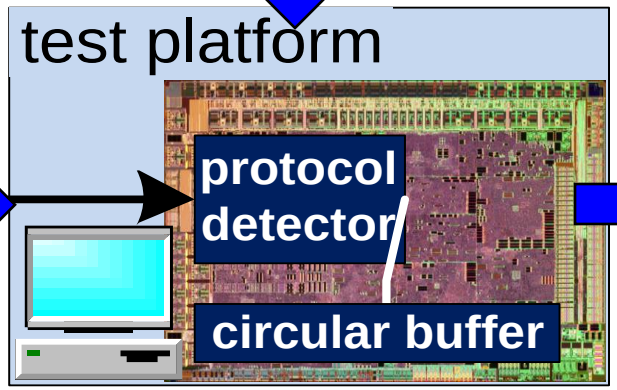
Post-silicon Protocol Detection



load protocols into programmable HW

post-si tests



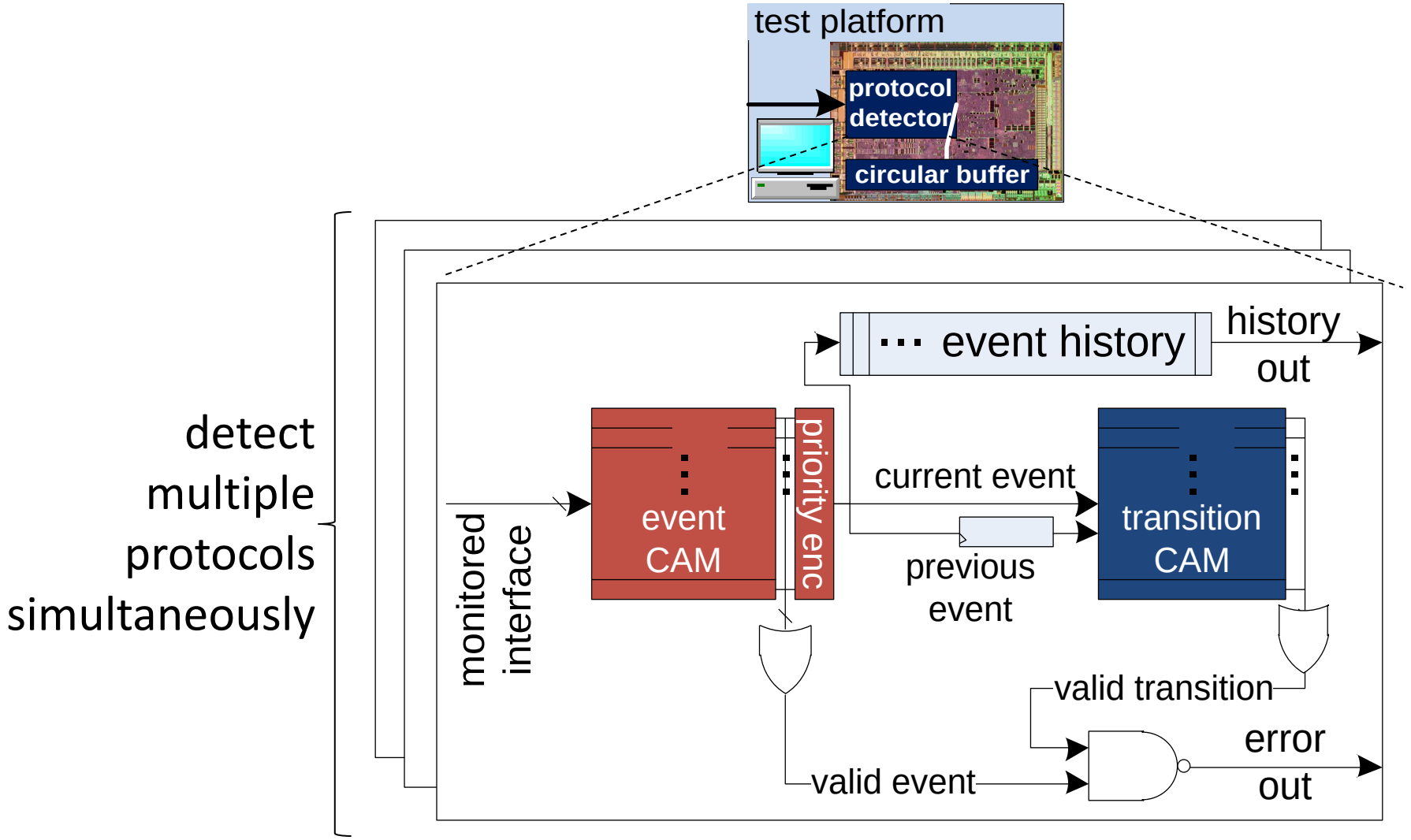


Error!

run high-coverage post-silicon tests

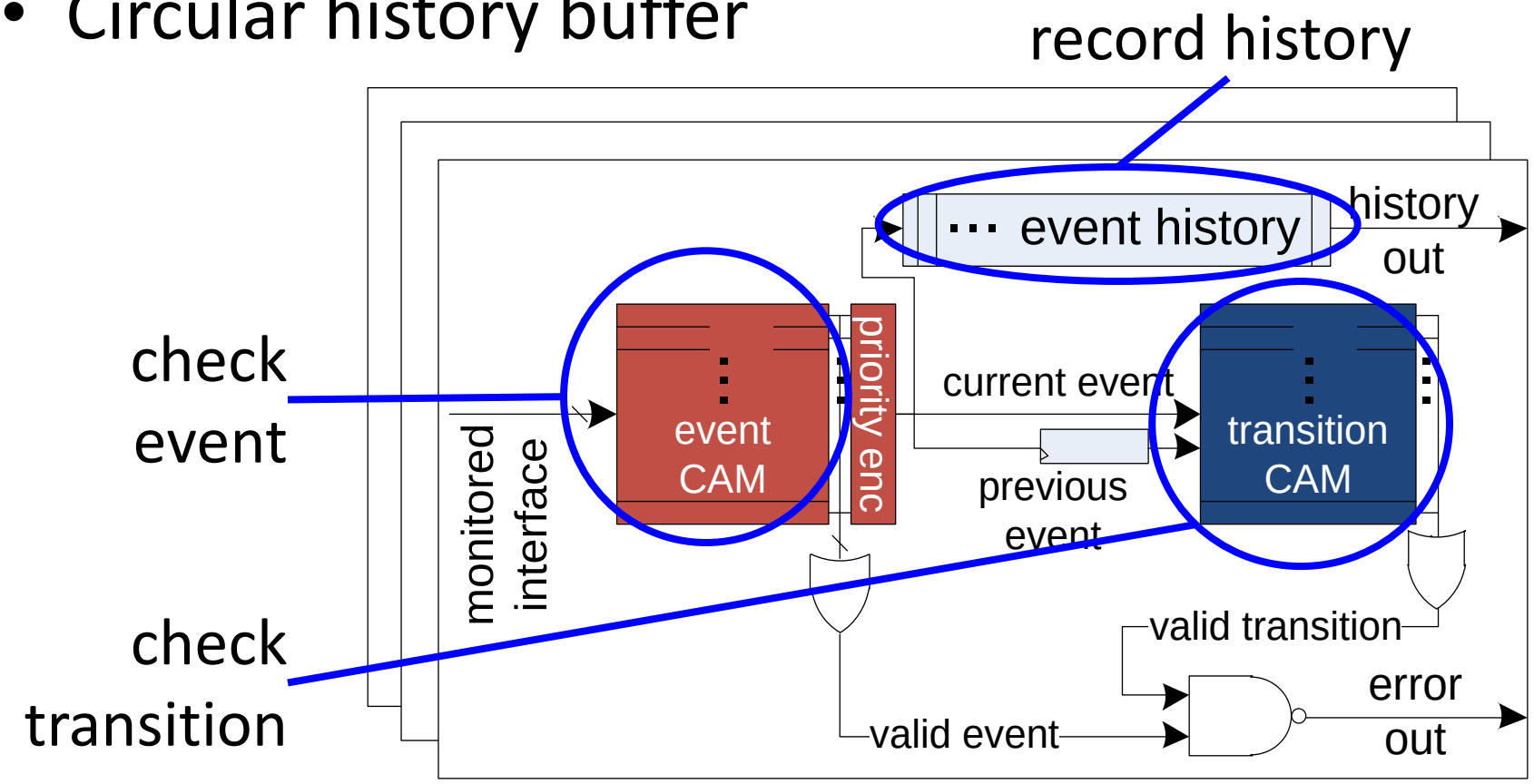
only stop when error is detected

Post-silicon protocol detection



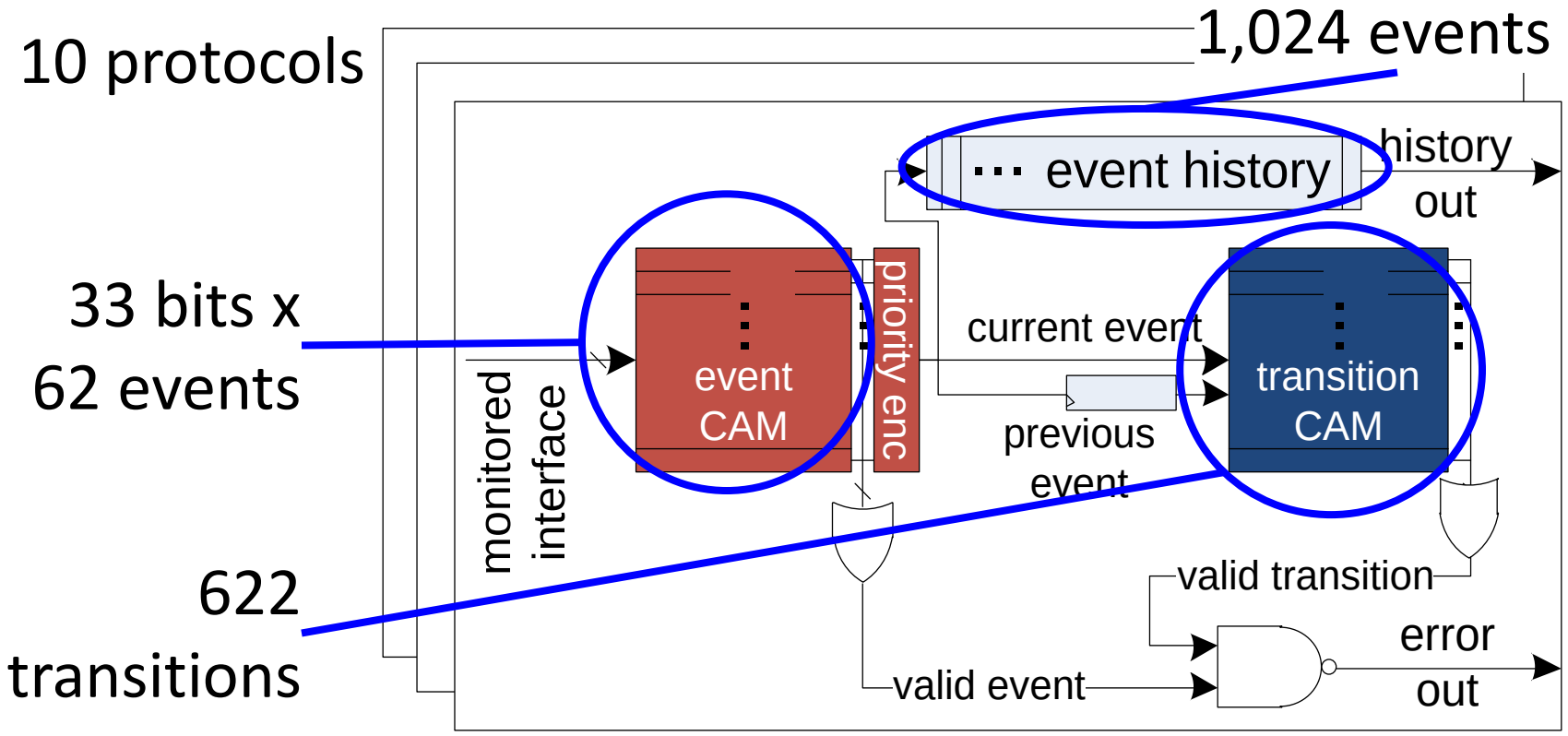
Protocol detector hardware

- Programmable
- Circular history buffer



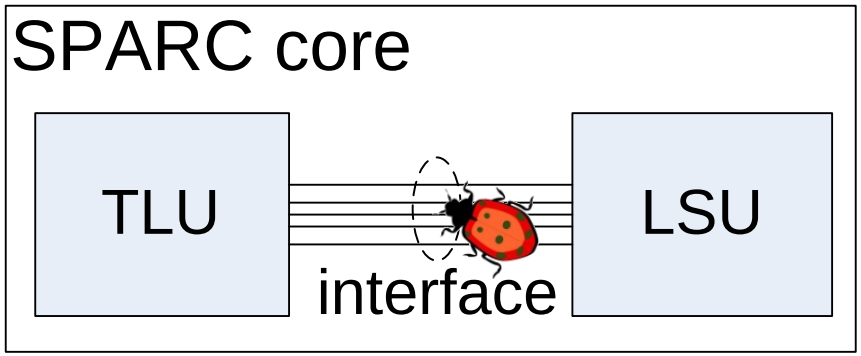
Area overhead

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

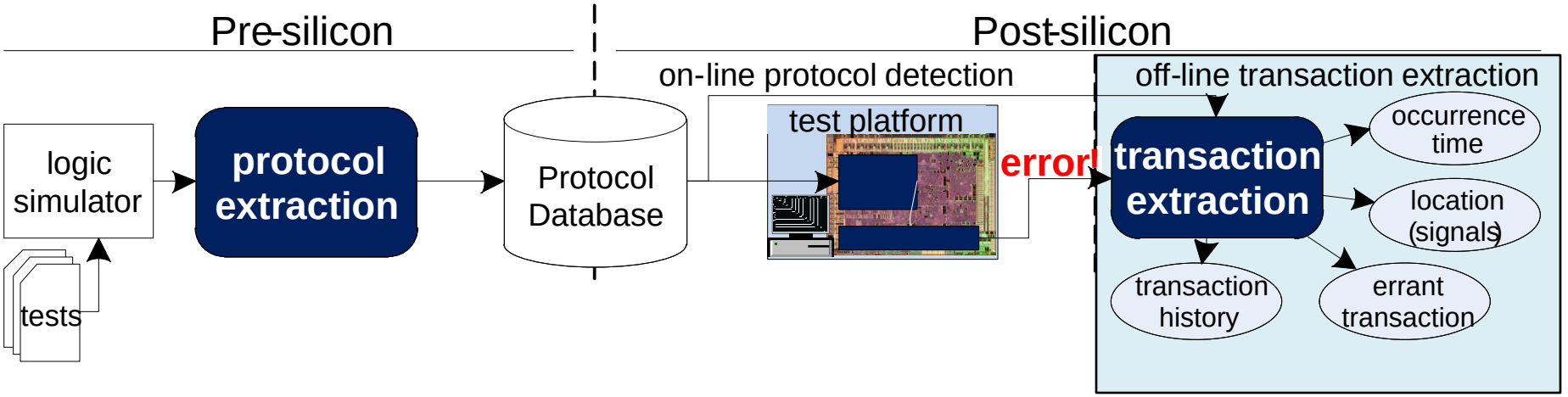


TLU Protocol Example

- 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

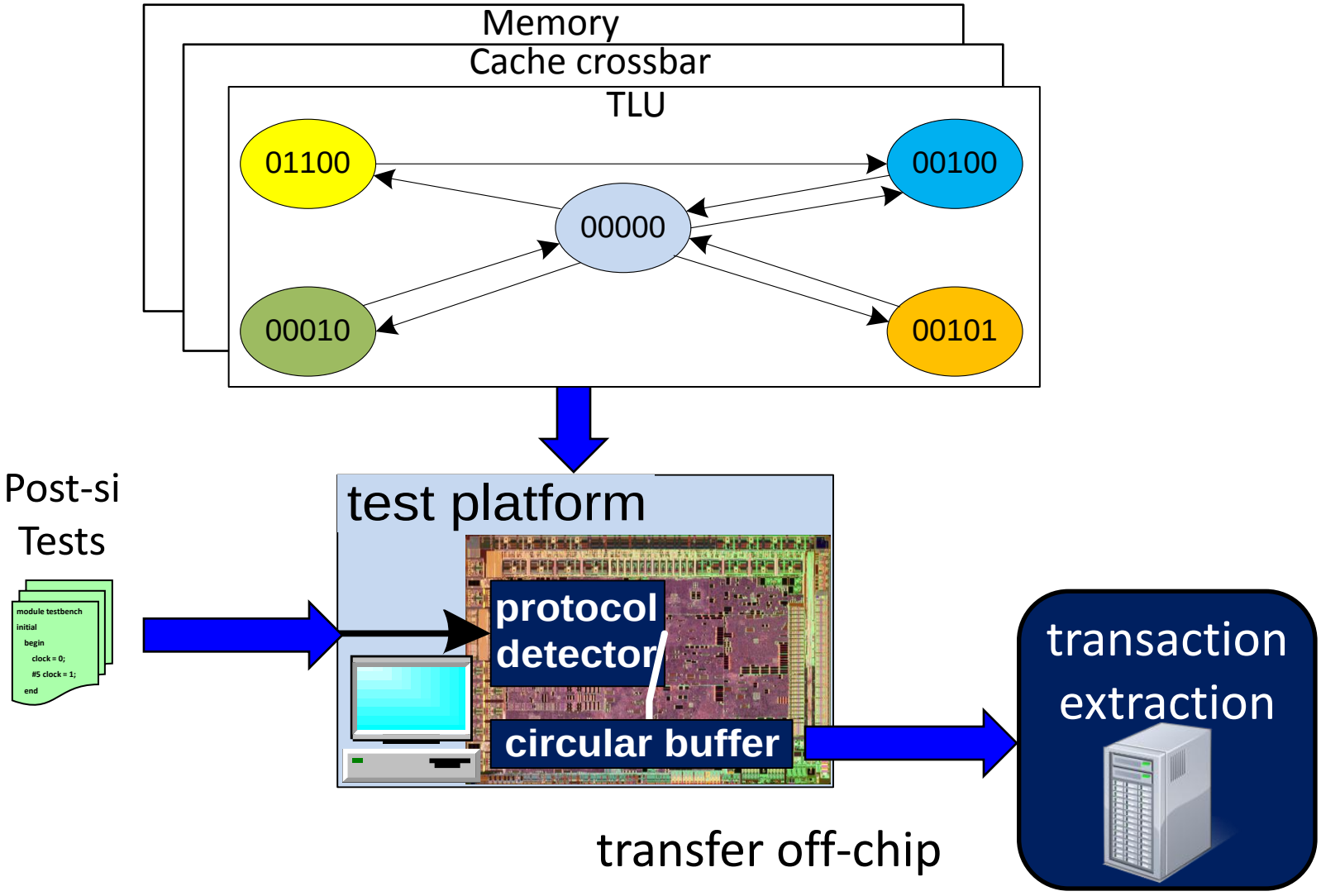


Outline



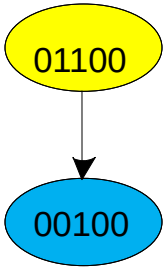
1. Pre-silicon protocol extraction
2. Post-silicon protocol detection
3. Offline transaction extraction

Off-line transaction extraction

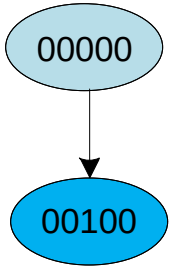


Transaction extraction

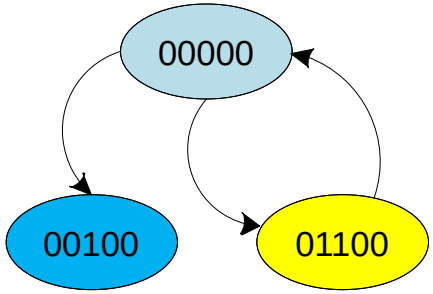
- Leverage transaction extraction similar to Inferno [DeOrio, et. al, 2009]
- Input: circular event buffer
- Output: intuitive, high-level transactions



thread sync

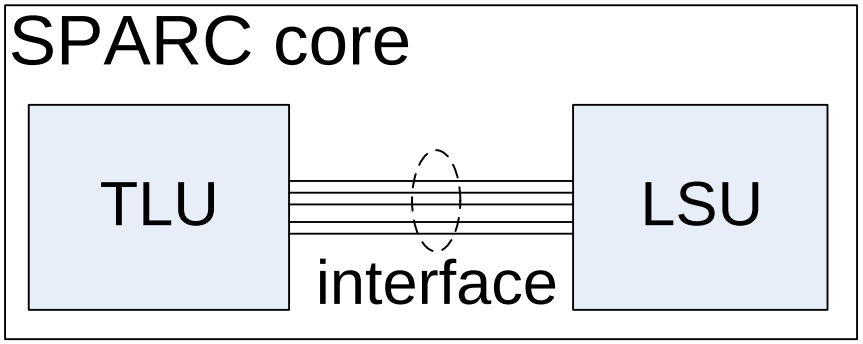


TLB bypass

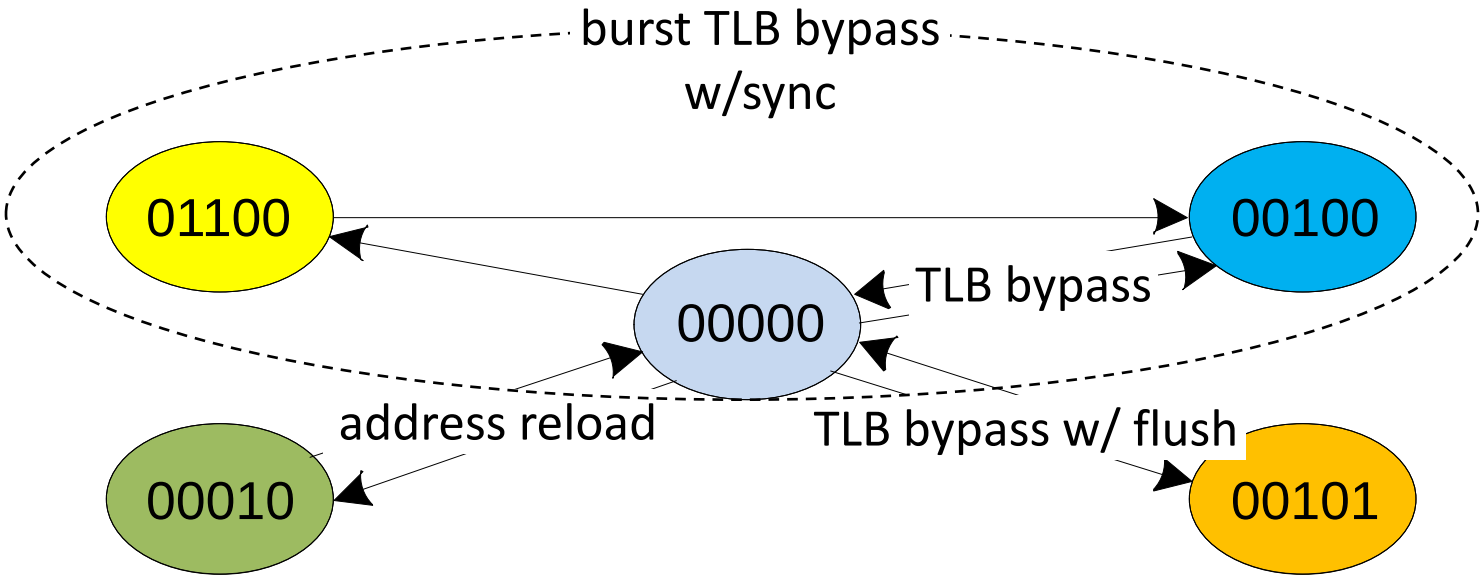


burst TLB bypass w/ thread sync

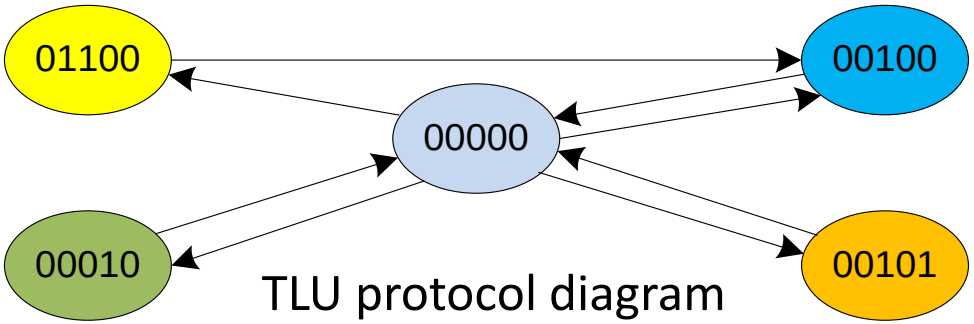
TLU Protocol Example




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

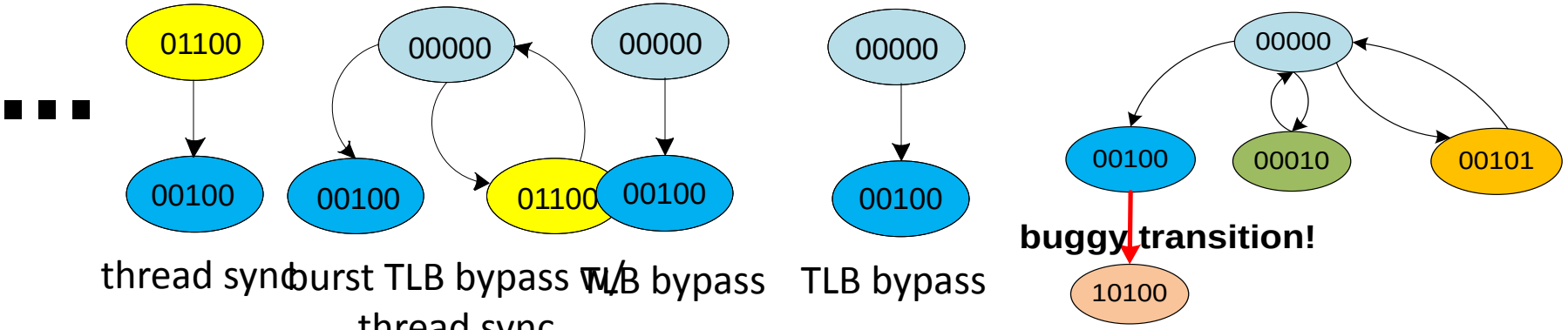


Transaction extraction example



TLU protocol diagram
(subset)

Extracted transaction history



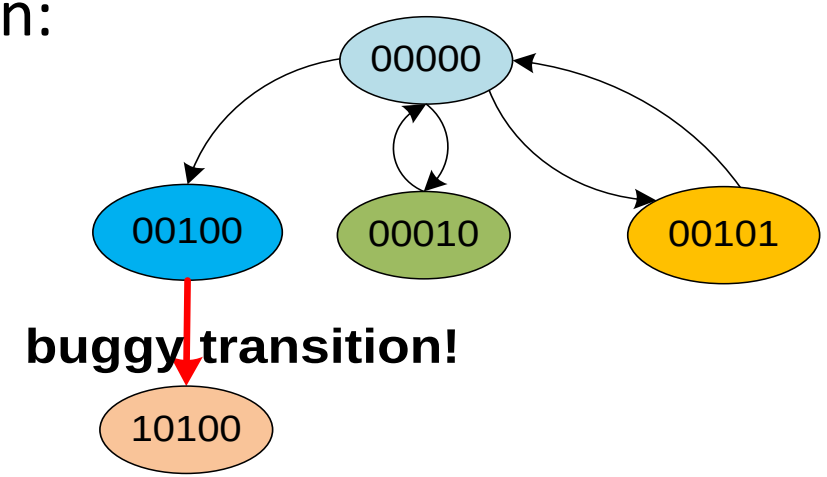
thread sync | burst TLB bypass w/ thread sync | TLB bypass | TLB bypass

3,694-3,732 — 4,492-4,531 — 4,539-4,543 — 4,545-4,602 — ... — 4,609 – 10,017

cycle

Transaction extraction example

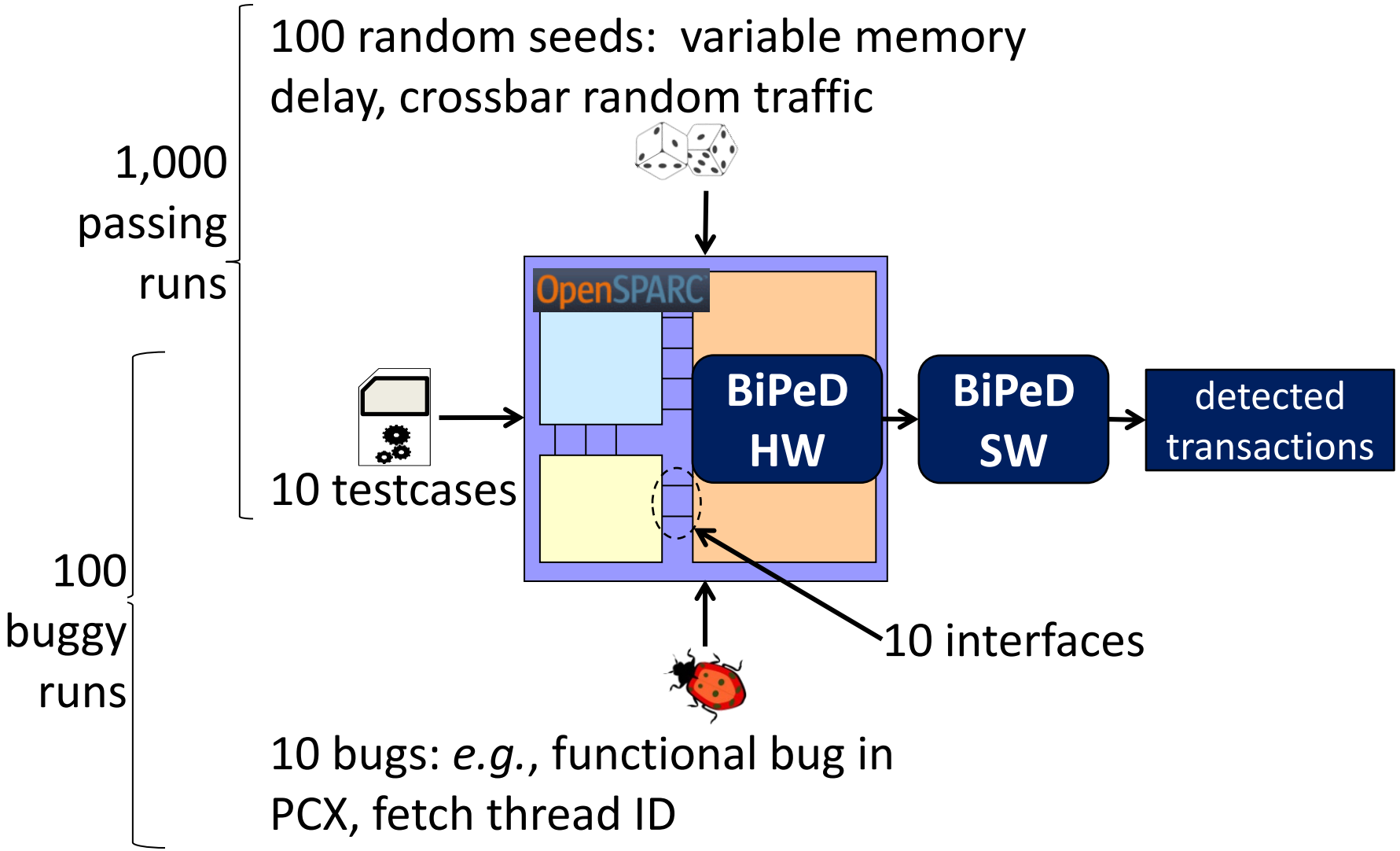
- Time: `cycle 10,017`
- Interface: `TLU`
- Signals: `protect, thread sync, TLB bypass, ASI reload, flush`
- Preceding activity: `thread sync, burst TLB bypass w/thread sync, TLB bypass, TLB bypass`
- Event: `10100` Transition: `00100` -> `10100`
- Transaction:



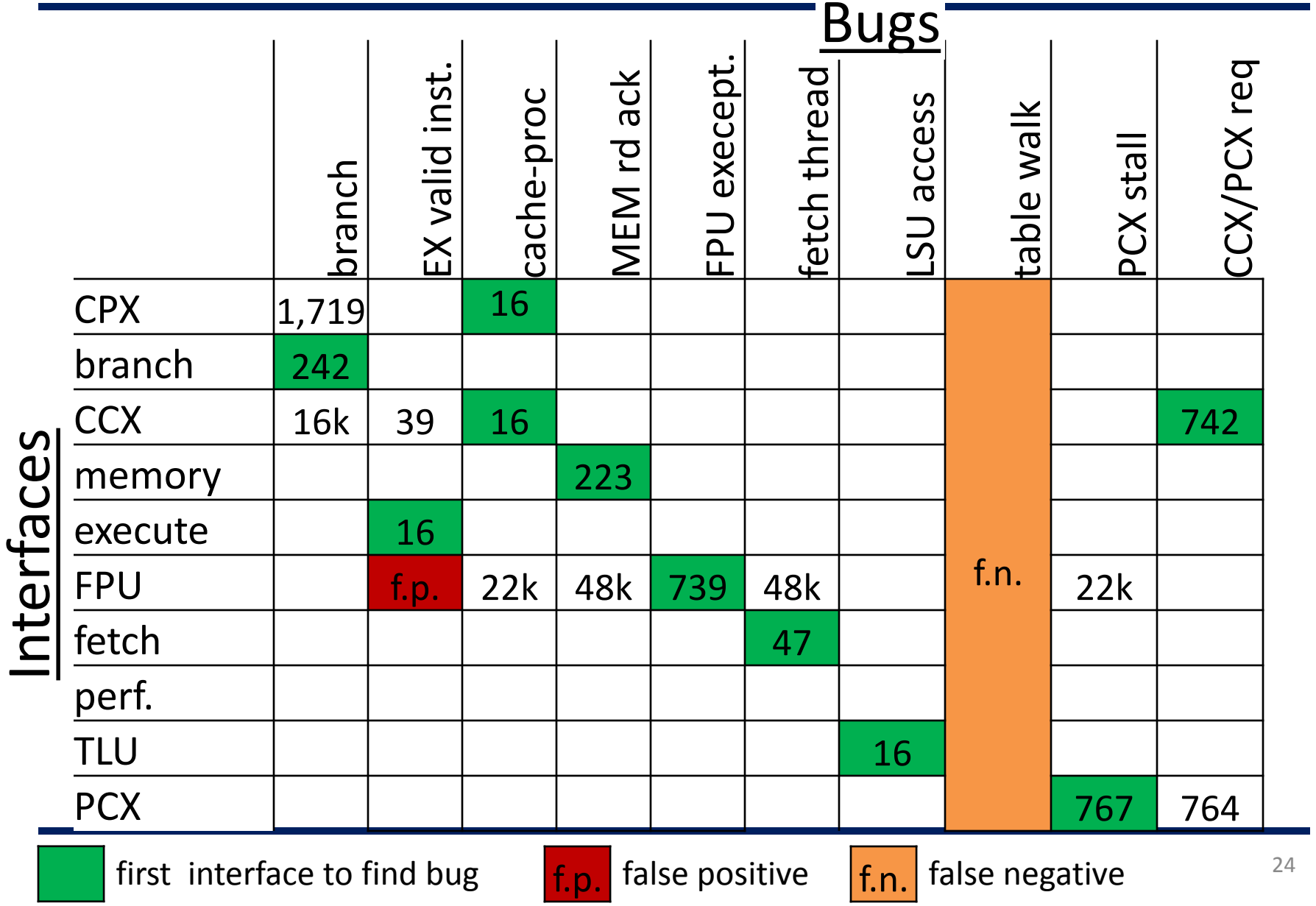
Limitations

- 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

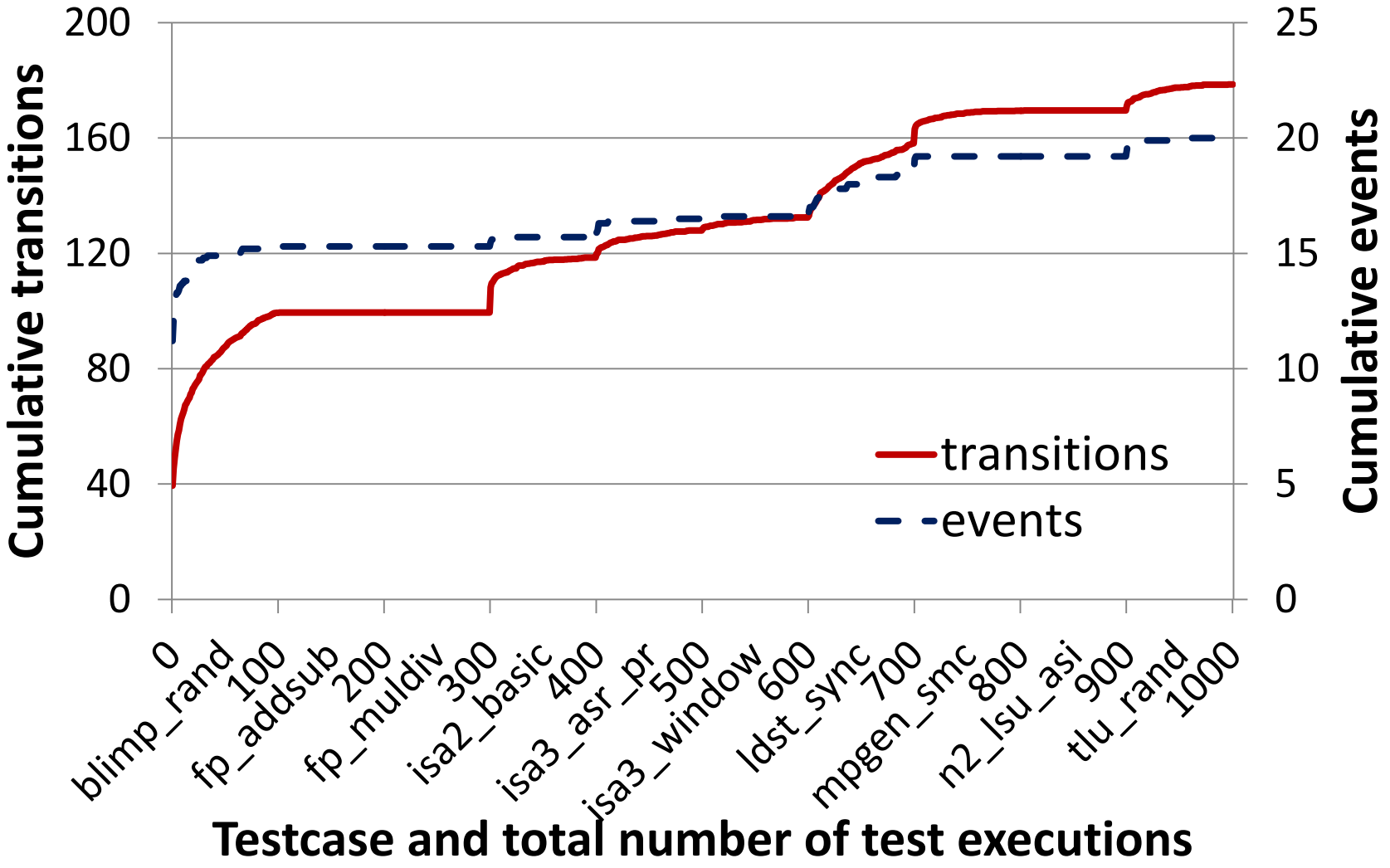
Experimental setup



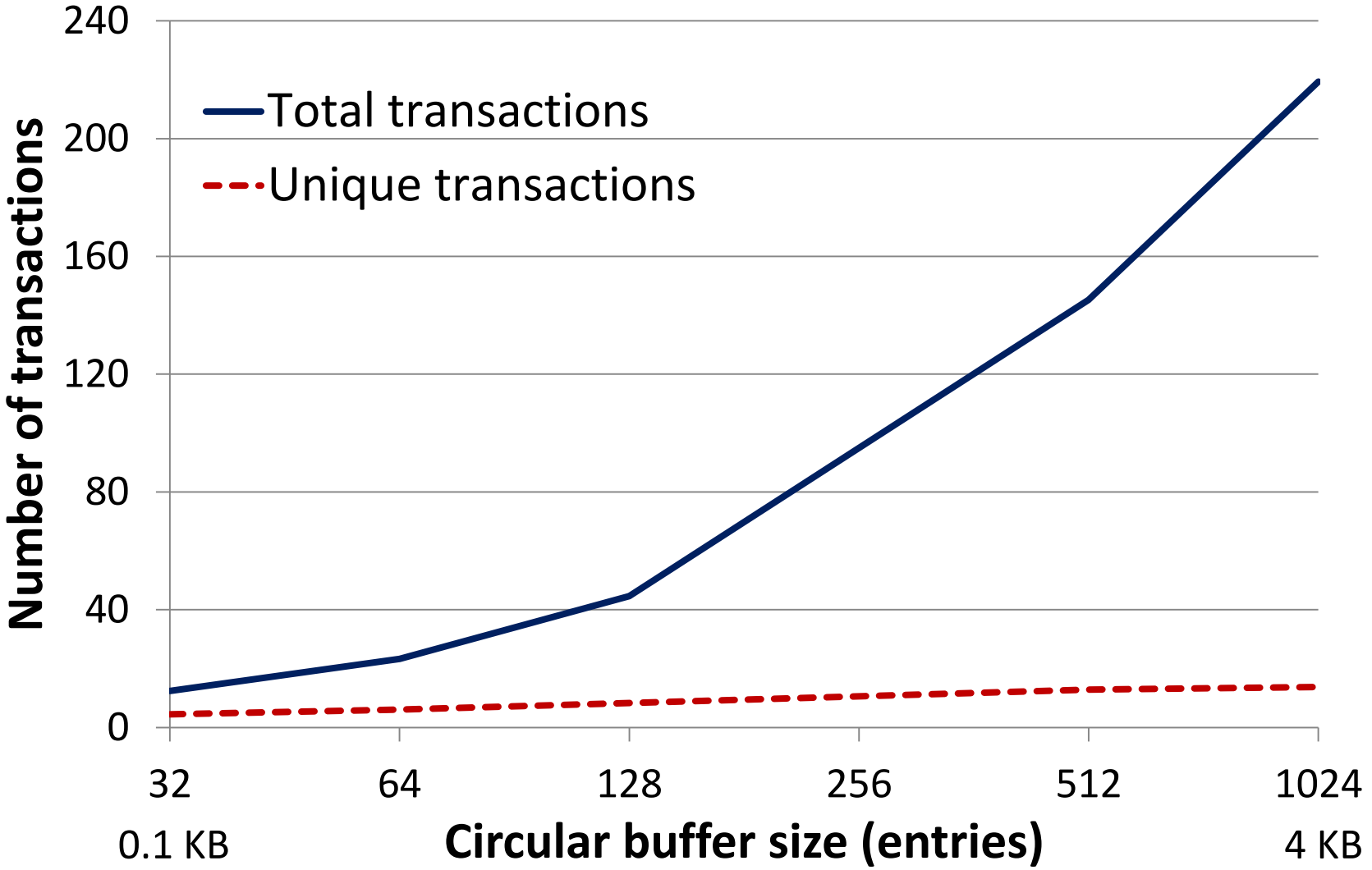
Signal Localization



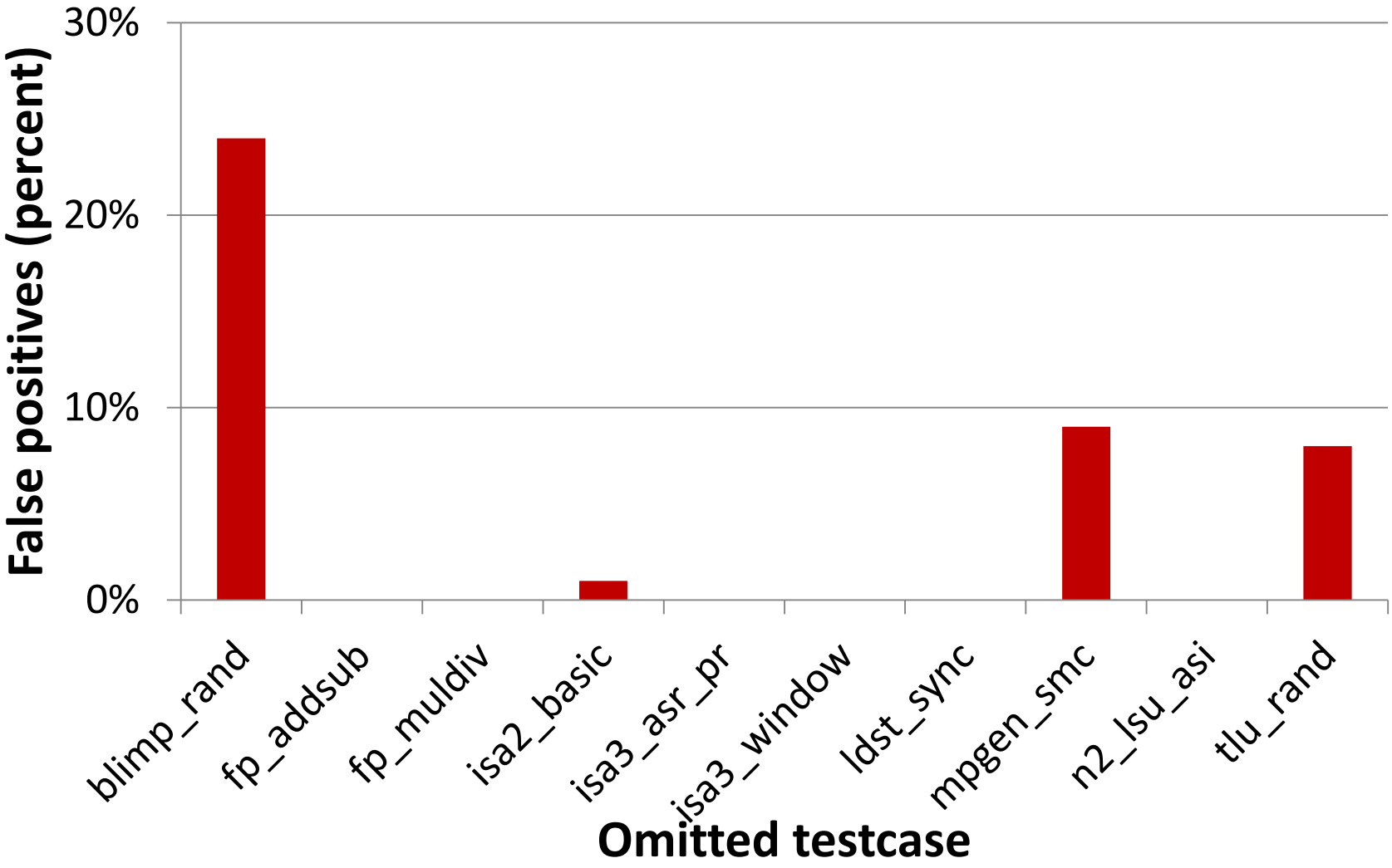
Protocol Extraction



Transaction Extraction



Leave-one-out Cross Validation



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