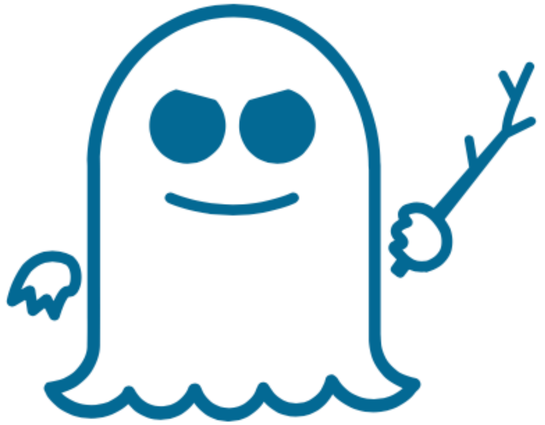
RISC-V Sim State Space Enumeration

By Griffin Knipe, Derek Rodriguez, David Kaeli and Yunsi Fei

Cost of complexity

Why is verification important?

- Unverified speculative execution leaks data, even with current hardware and software protections in place.
- Post-silicon designs are too costly to be verified at gate-level.
- Hardware vulnerabilities break software security guarantees.



Yori Goals

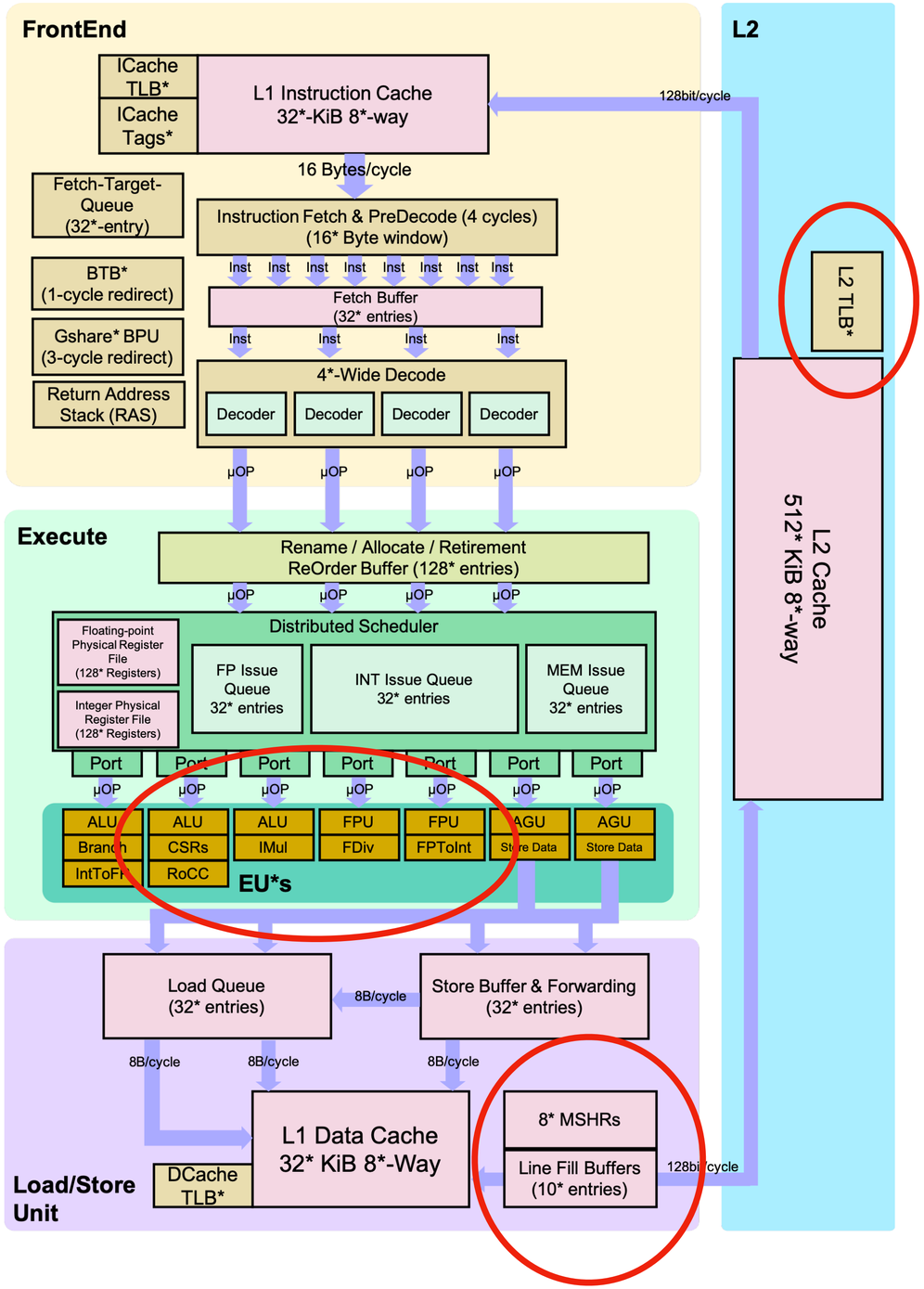
Verifying communication at the block level

- Catch speculative attacks at start of arch design.
- Collect performance estimates and run litmus tests from a single implementation.
- Provide a verification platform for design exploration that supports block-level performance simulation.

Do speculated loads bypass ASID checks in L2?

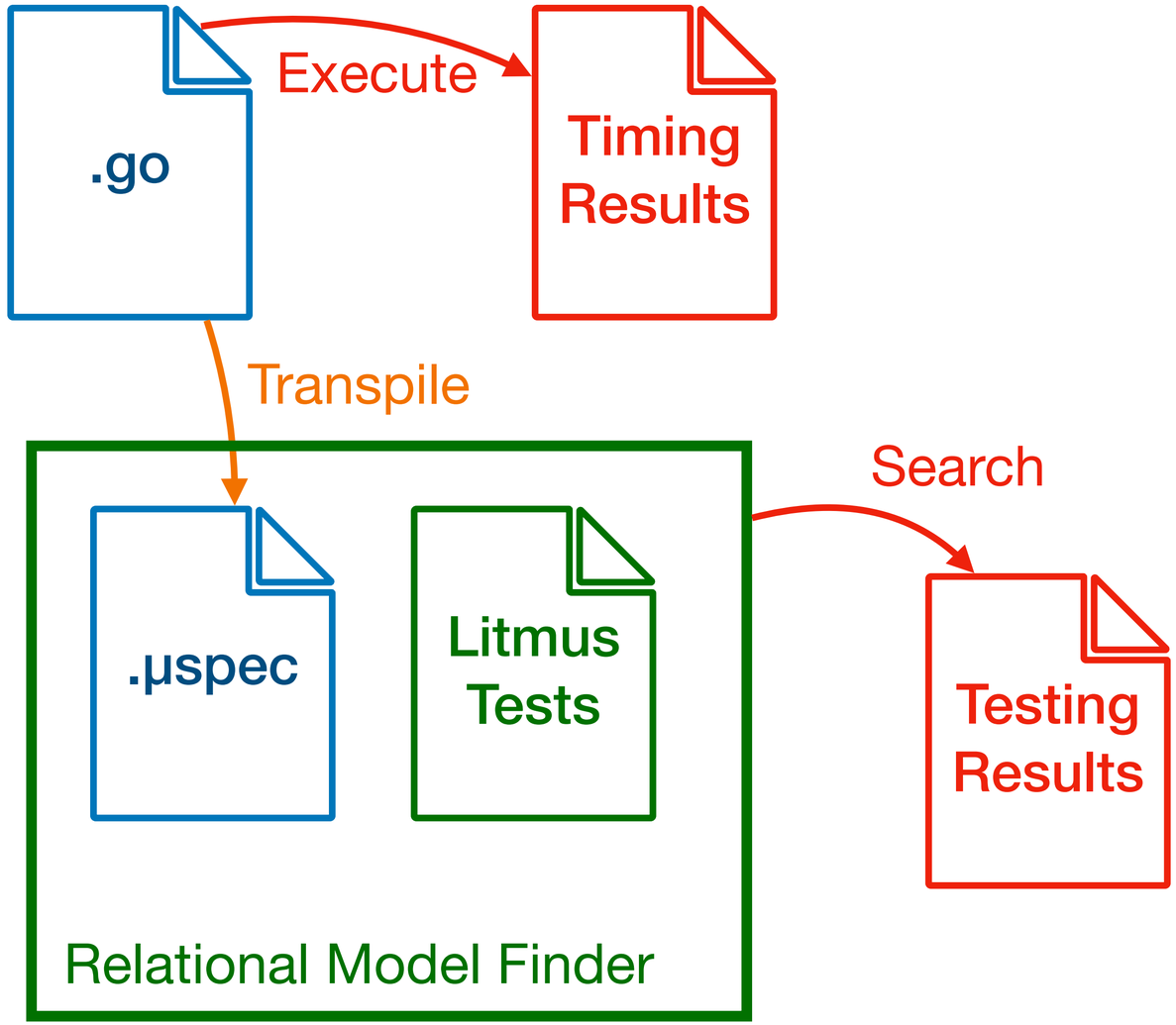
Is there a data race during FPU fault handling?

Is the line fill buffer safe from timing side-channels?



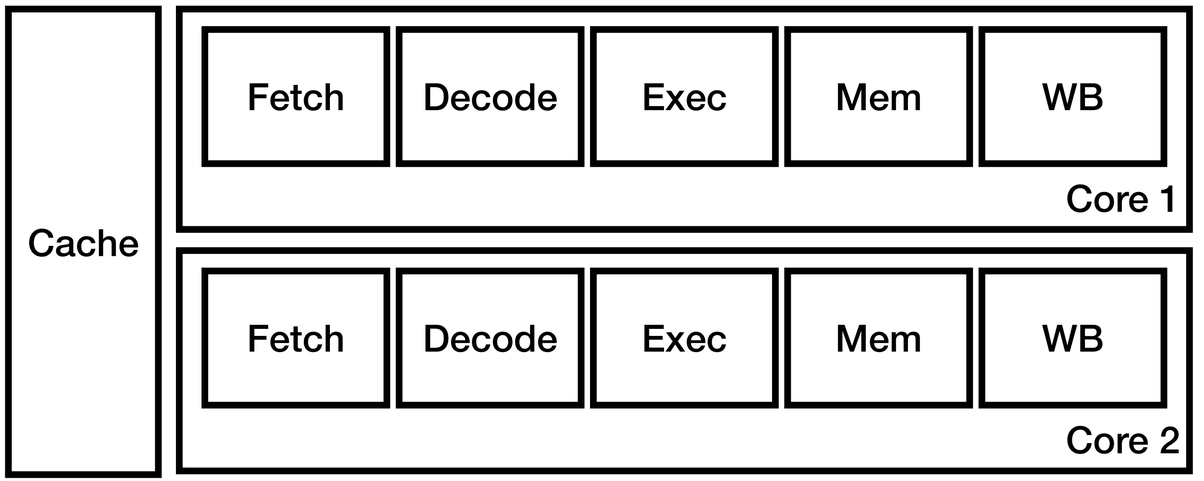
Bridging between Akita and CheckMate

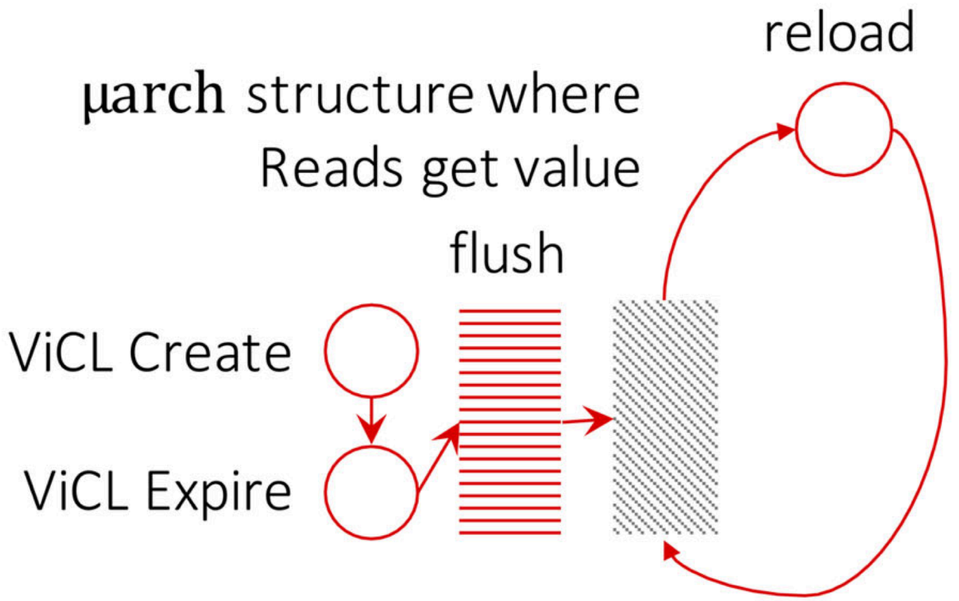
- Both Akita and CheckMate rely on **events** and **time** to describe a μarch.
- Akita is imperative (Go), while CheckMate is declarative (μspec).
- Akita flexibly interfaces with different levels of accuracy.
- Treat Akita as a Go DSL for state machines.



CheckMate for Relational Modeling

- A relational model is a directed graph, where edges indicate interactions.
- Relational models can be verified against a set of constraints, e.g., the RISC-V WMO **memory consistency model.**
- Proof by Counterexample for both **MCM** litmus tests and **security** litmus tests = subgraph matching problems using RMF.

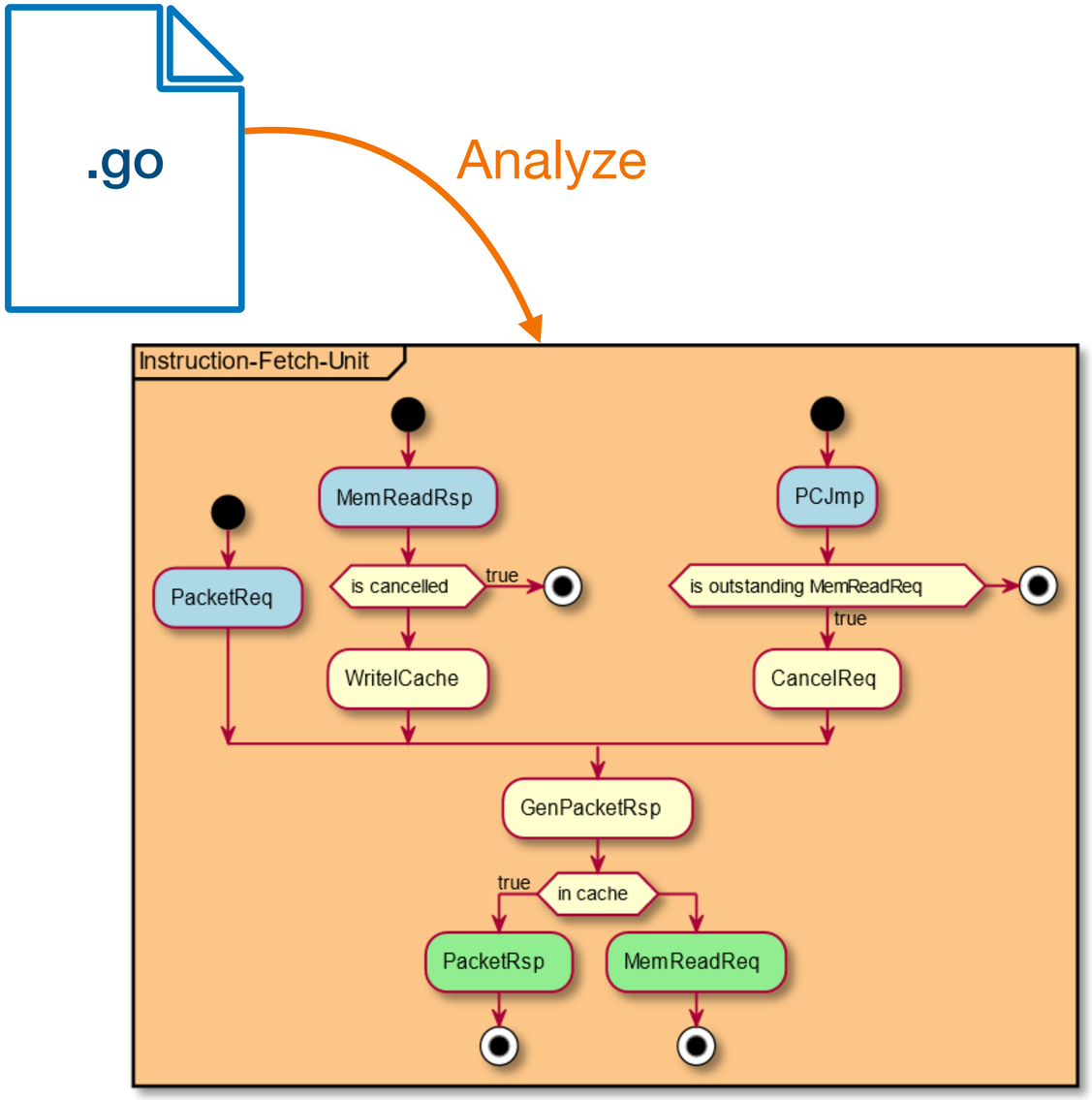




Source: Trippel, Caroline et al. *Security Verification via Automatic Hardware-Aware Exploit Synthesis: The CheckMate Approach*. IEEE MICRO 2019

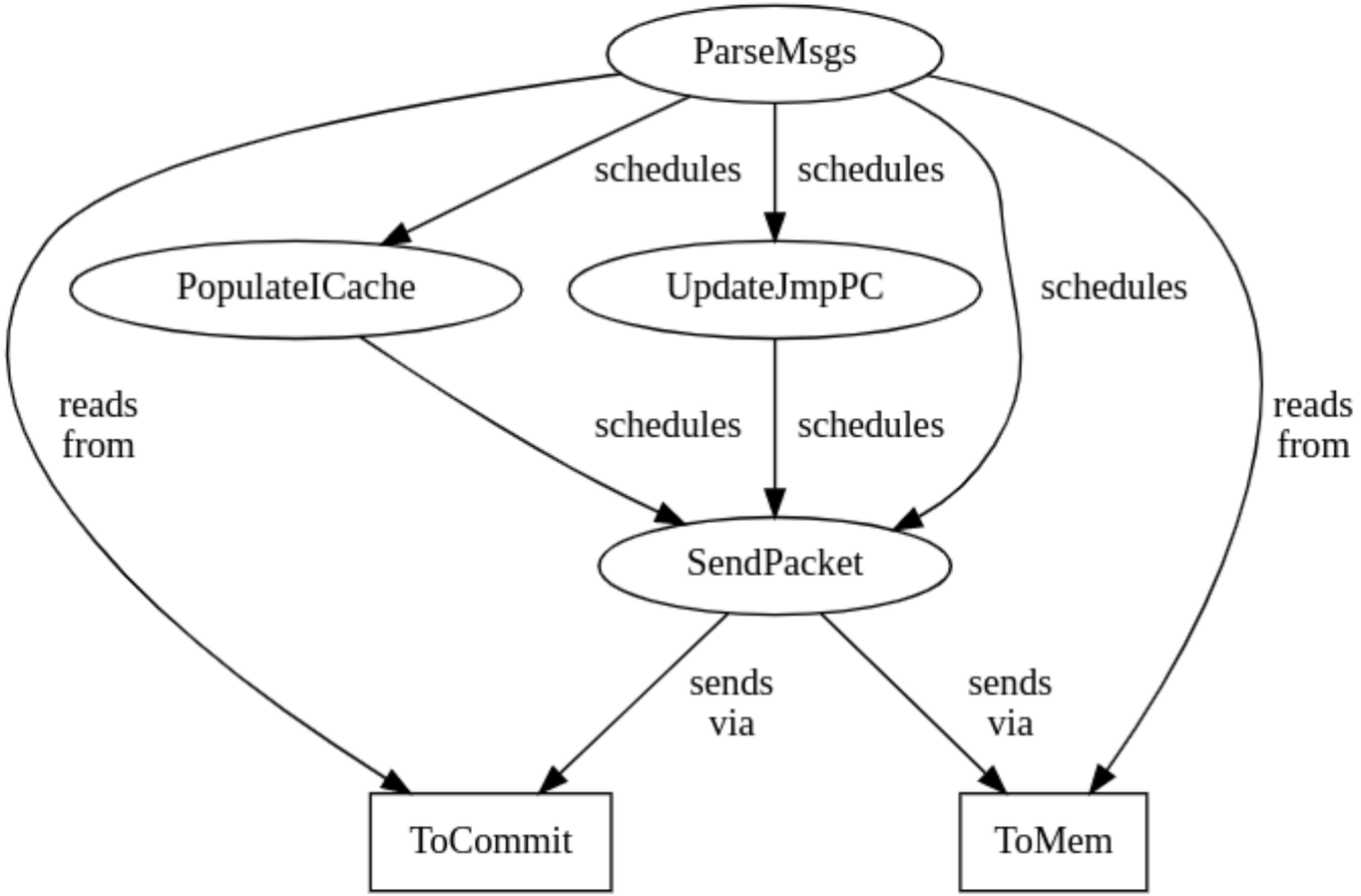
Imperative → Declarative: Static Analysis

1. Parse Akita component definitions for state-holding members
2. Determine component data in-flow
3. Map data in-flow to out-flow
4. Map flow paths to Akita events
5. (In Progress) Convert event+flow to μspec
6. (Future Work) Chain event sequences across components
7. (Future Work) Map event sequences to instructions



Example: Fetch Unit

State Machine represented in GraphViz



Future Work

- Coalesce mappings into ISA behavior. For example, what events correspond to a load instruction?
- Extend relational model extraction to all components: leverage Akita as a domain specific language.
- Demonstrate complete CheckMate integration by demonstrating automatic vulnerability detection.

Conclusion

- Yori combines **architectural simulation** with **bounded verification** for **ISA correctness** and **side channel detection**.
- We illustrate complete enumeration of the state space for a simple instruction fetch unit via a multi-pass, static analysis of Akita.
- We propose a path towards generating relational models from enumerated state spaces, which can be used to generate the CheckMate DSL.

This work was supported in part by the National Science Foundation under Grant CNS-1916762 with the industry support from the Center for Hardware and Embedded Systems Security and Trust (CHEST) and Draper Laboratory.