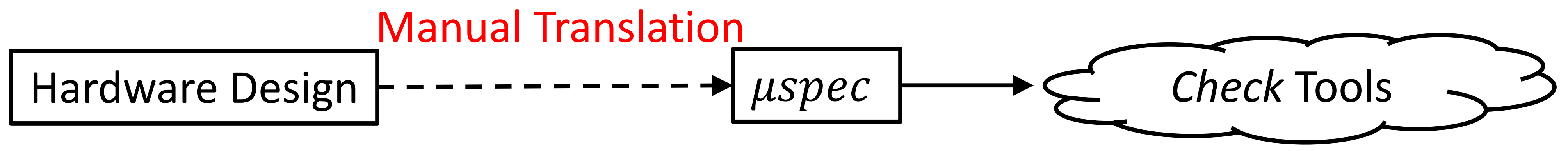
Synthesizing Formal Models of Hardware from RTL for Efficient Verification of Memory Model Implementations

Stanford | ENGINEERING

Yao Hsiao, Dominic P. Mulligan*, Nikos Nikoleris*, Gustavo Petri*, Caroline Trippel
Stanford University, *ARM Research

Introduction & Background

- **Motivation:** Check tools[1] enable hardware memory consistency model (MCM) verification, but require manually-constructed formal microarchitecture specifications (μ*spec* models) as input



- **Goal:** Efficient Check-based verification of hardware MCM implementations rooted in RTL
- **Key Challenges:** 1) Discerning μ*spec* model completeness 2) Resolving the gap between *operational* RTL and *axiomatic* μ*spec* models

A Taxonomy for Constructing Complete μ*spec* Models

- ***μspec* models** specify the space of all possible *micro-architectural happens-before* graphs (μ*hb* graphs) using first-order logic axioms that, given a program and a microarchitecture, instantiate
  - μ*hb* nodes: Microarchitectural events, i.e., (instruction, location) pairs
  - μ*hb* edges **:** Happens-before (HB) relationships between nodes
- **Verilog:** Temporal description of a hardware design (Fig. 1a)
- ***μspec* model taxonomy:** Happens-Before Invariants (HBIs) preserved by an operational Verilog design
  - **Intra-instruction HBIs**: How each instruction flows through the design (Fig. 1b, ➞ in Fig. 2c)
  - **Inter-instruction HBIs**: Interactions between instructions during program executions
    - Structural dependencies: Serialized access to shared state elements (Fig. 1c, ➞ in Fig. 2c
    - Dataflow dependencies: Share data through common resources (Fig. 1d, ➞/ ➞ in Fig. 2c)

Core 0	Core 1
(i0) lw x, 0 (i1) sw y, 1	(i2) lw y 1

Fig. 1e: Program input. Observable execution in μ*hb* graph of Fig. 2c

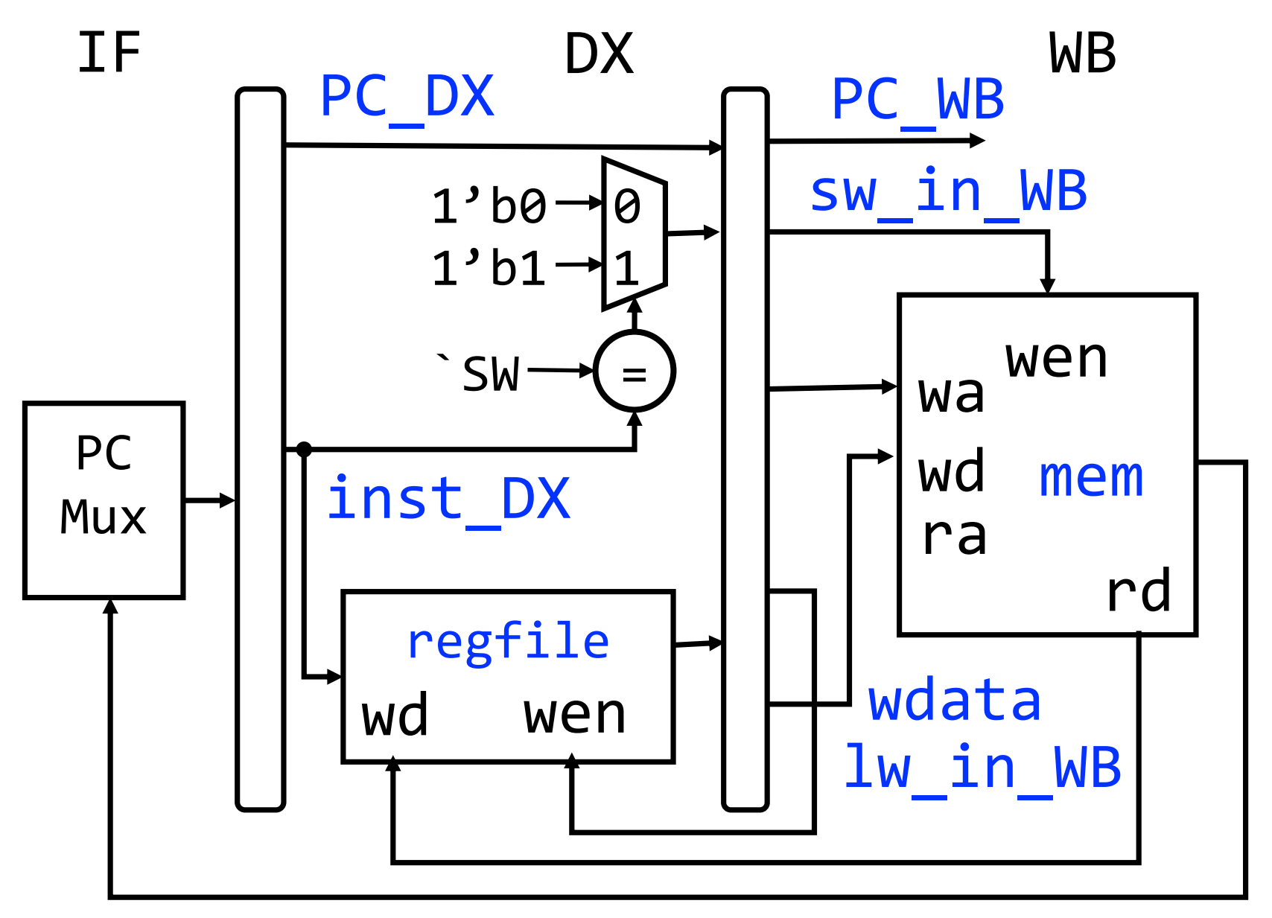


Fig. 1a: Multi-V-Scale

```
Axiom W_path: forall microops i1,
IsAnyWrite i1 ⇒ AddEdges [
((i1, inst_DX), (i2, sw_in_WB));
(i1, inst_DX), (i1, lw_in_WB));
(i1, sw_in_WB), (i1, mem))].
```

Fig. 1b: Intra-inst. HBIs

```
Axiom Structural_Temporal:
forall microops i1, i2,
IsAnyWrite i1 ⇒ IsAnyRead i2 ⇒
ProgramOrder i1 i2 ⇒ AddEdge(
(i1, mem), (i2, regfile)).
```

Fig. 1c: Inter-inst. Structural HBIs

```
Axiom Dataflow:
forall microops i1, i2,
IsAnyWrite i1 ⇒ IsAnyRead i2 ⇒
SamePA i1 i2 ⇒ SameData i1 i2 ⇒
NoWritesInBetween i1 i2 ⇒
AddEdge((i1, mem), (i2, regfile)).
```

Fig. 1d: Inter-inst. Dataflow HBIs

Synthesizing a μ*spec* Model from RTL with rtl2μ*spec*

- **A Verilog netlist** contains a subset of the targeted HBIs. **A full-design data-flow graph (DFG)** represents over-approximation of data-flow that can be induced by instruction execution.
  - **DFG nodes:** instruction-dependent registers
  - **DFG edges:** over-approximation of information flow one register to another (an HB relation)
- **Intra-instruction HBI synthesis:**
  - Perform DFS on full-design DFG to generate **intra-instruction HBI hypotheses** as SVAs
  - **SVA evaluation** outputs full set of intra-instruction HBIs and instruction specific DFGs
- **Inter-instruction HBI synthesis:**
  - Pairwise compare instruction-specific DFGs to identify each type of dependency
  - Formulate dependencies as **HBI hypotheses** and instantiate them as SVAs
  - SVA evaluation outputs a full set of inter-instruction HBIs

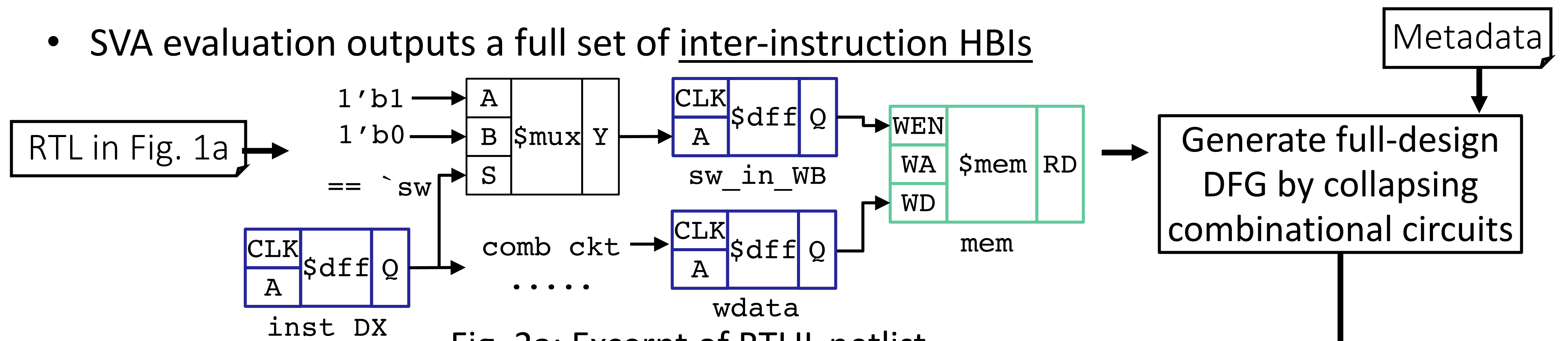


Fig. 2a: Excerpt of RTLIL netlist

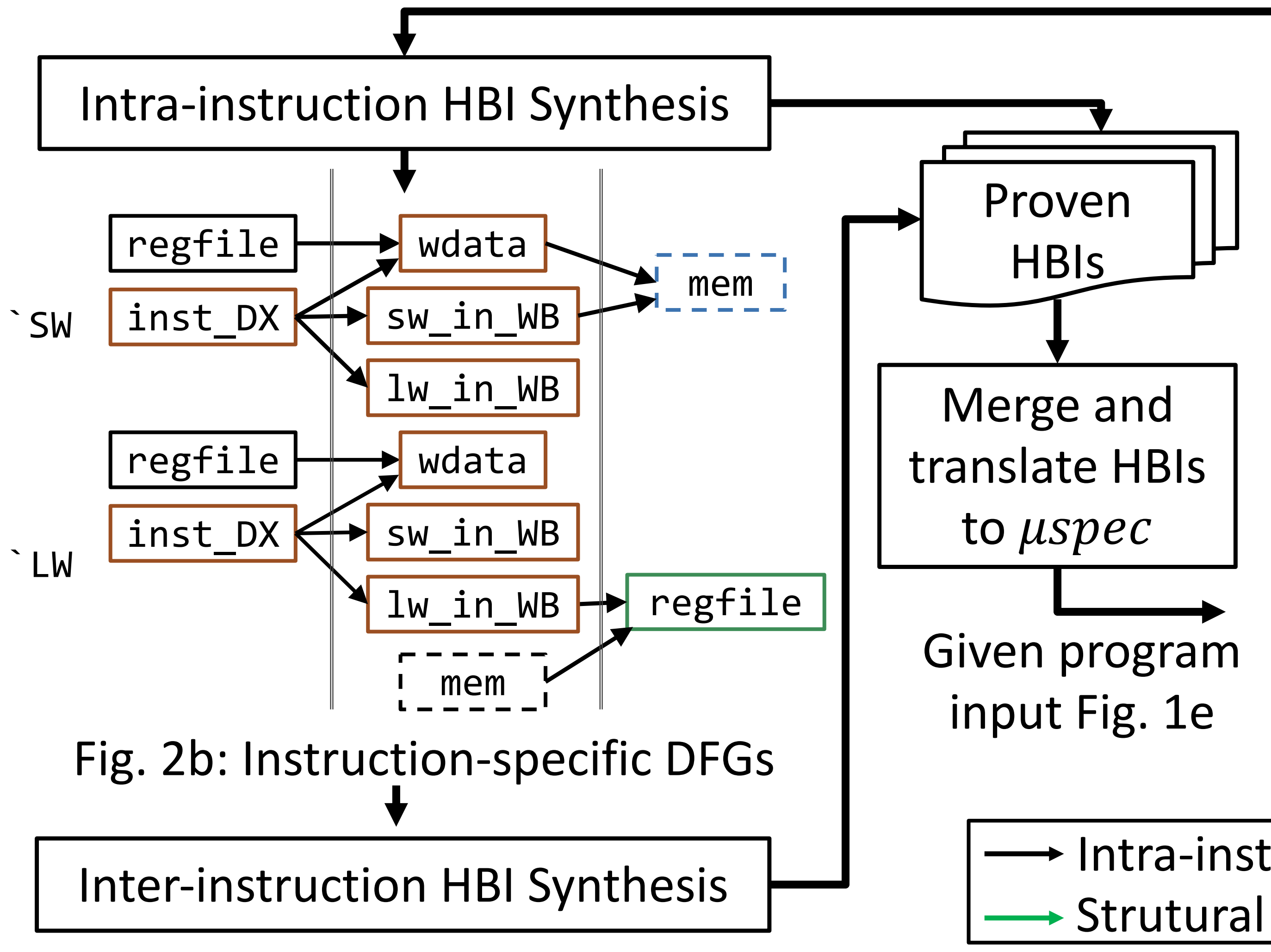

Intra-instruction HBI Synthesis
`SW
regfile
inst_DX
wdata
sw_in_WB
lw_in_WB
mem
`LW
regfile
inst_DX
wdata
sw_in_WB
lw_in_WB
regfile
mem
Proven HBIs
Merge and translate HBIs to μspec
Given program input Fig. 1e
Inter-instruction HBI Synthesis

Fig. 2b: Instruction-specific DFGs

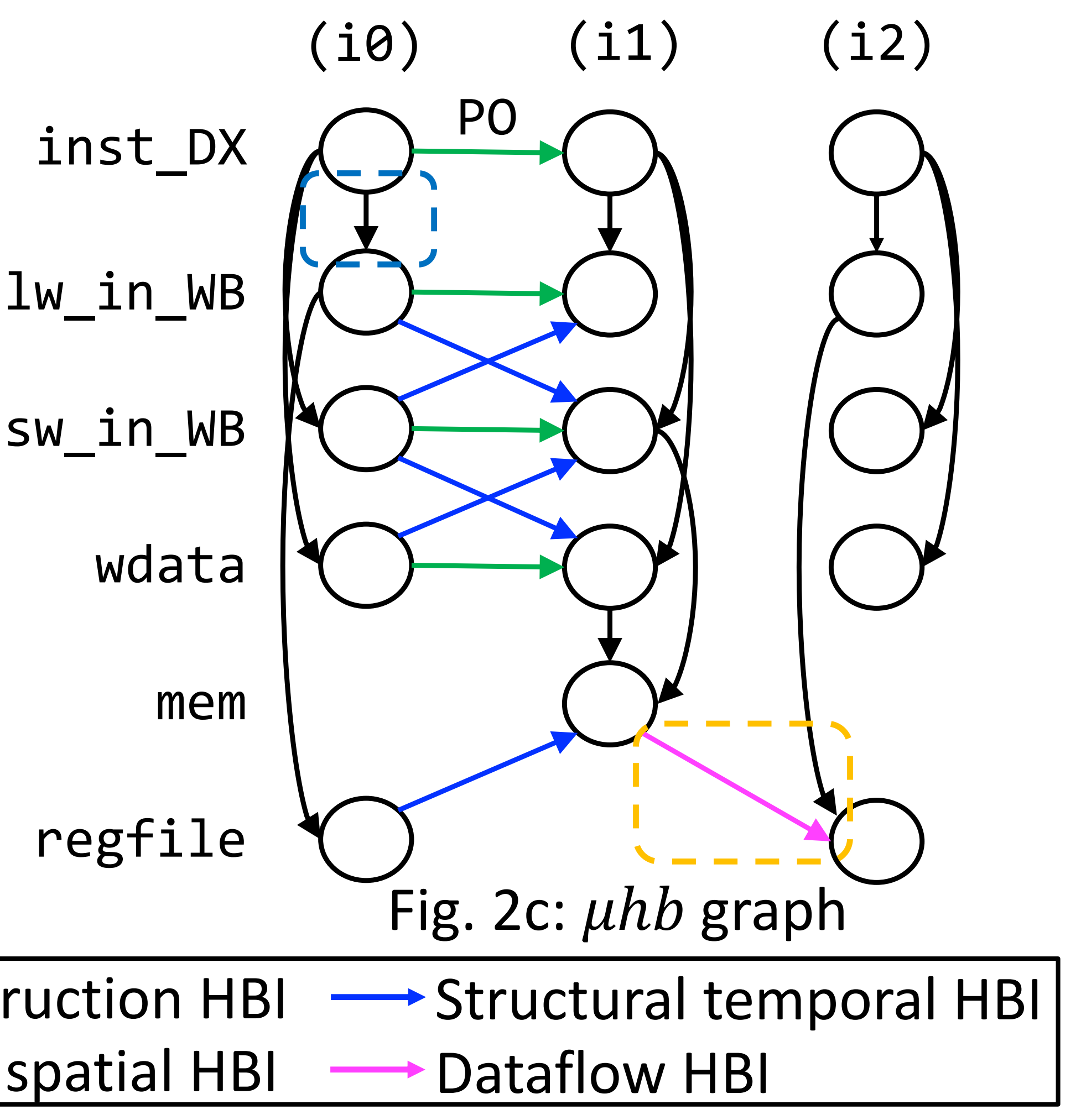
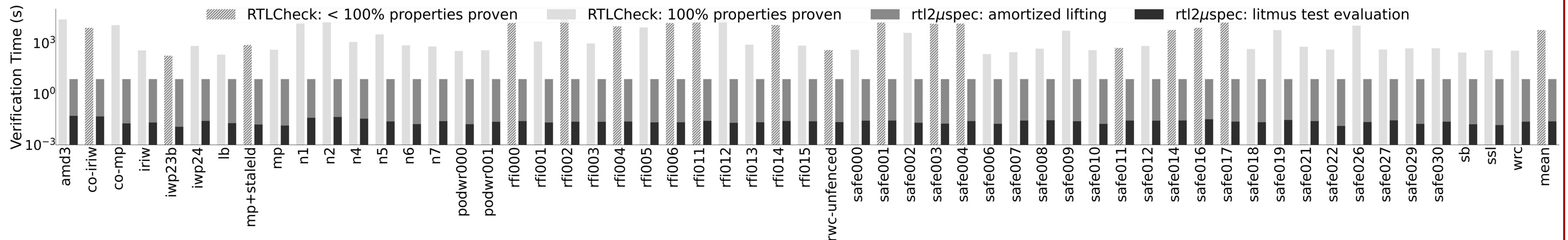


Fig. 2c: μ*hb* graph

Case Study Result



Performance comparison of rtl2μspec-assisted versus RTLCheck [1]-based verification of hardware MCMs (avg: 7.36 sec vs 25 min)

Conclusion & Contribution

- Define what constitutes a complete μ*spec* model for an RTL design
- **rtl2*μspec* tool** for synthesizing complete, and proven correct by construction, μ*spec* models from RTL with minimal user-intervention
- **Verification** of the RISC-V multi-V-Scale MCM implementation: rtl2μspec synthesizes a μ*spec* model in 6.84 mins, during which a new bug is found. Subsequent Check-based verification takes less than 1 second per litmus test.

[1] https://check.cs.princeton.edu/

[1] Manerkar, Yatin A., et al. RTLCheck: Verifying the memory consistency of RTL designs." Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture. 2017.