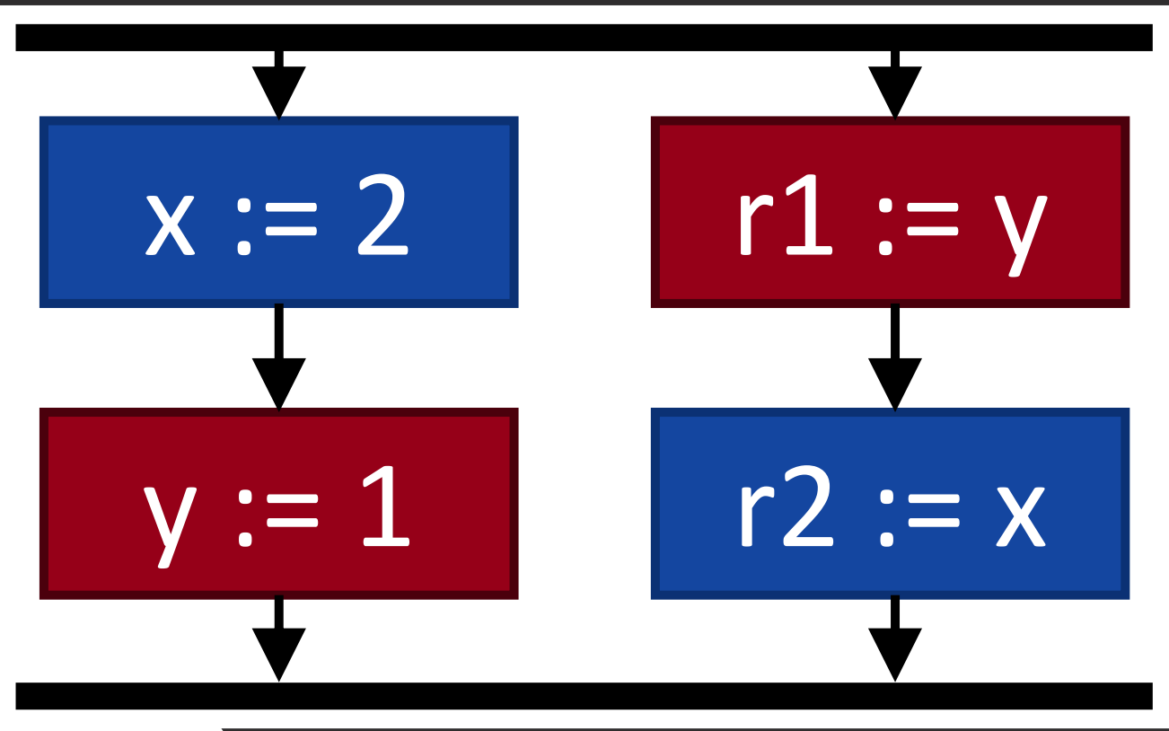
Will My Program Break on This Faulty Processor?

ftsrg

Multi-core Memory Models

What do we expect?

Naive approach:
Strong sequentiality

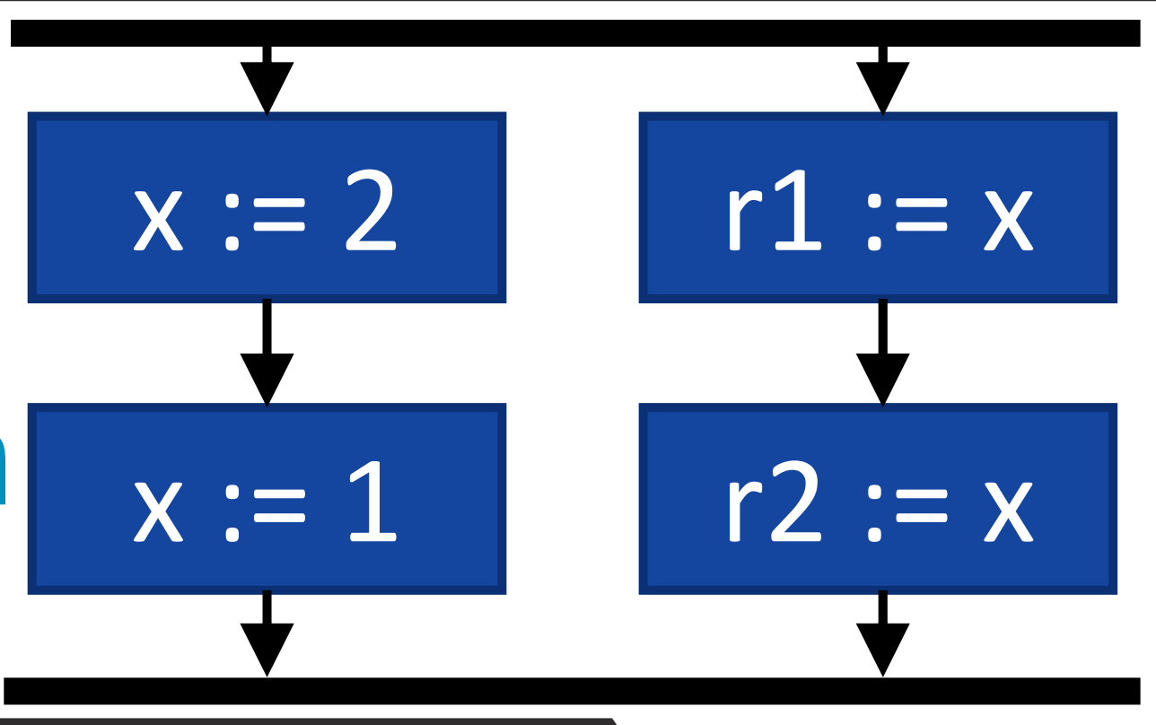


$r1 == 1 \;\Rightarrow\; r2 == 2$

- **Program order** is always intact
- Reordering **not possible**
- Not **optimal**, but **simple**

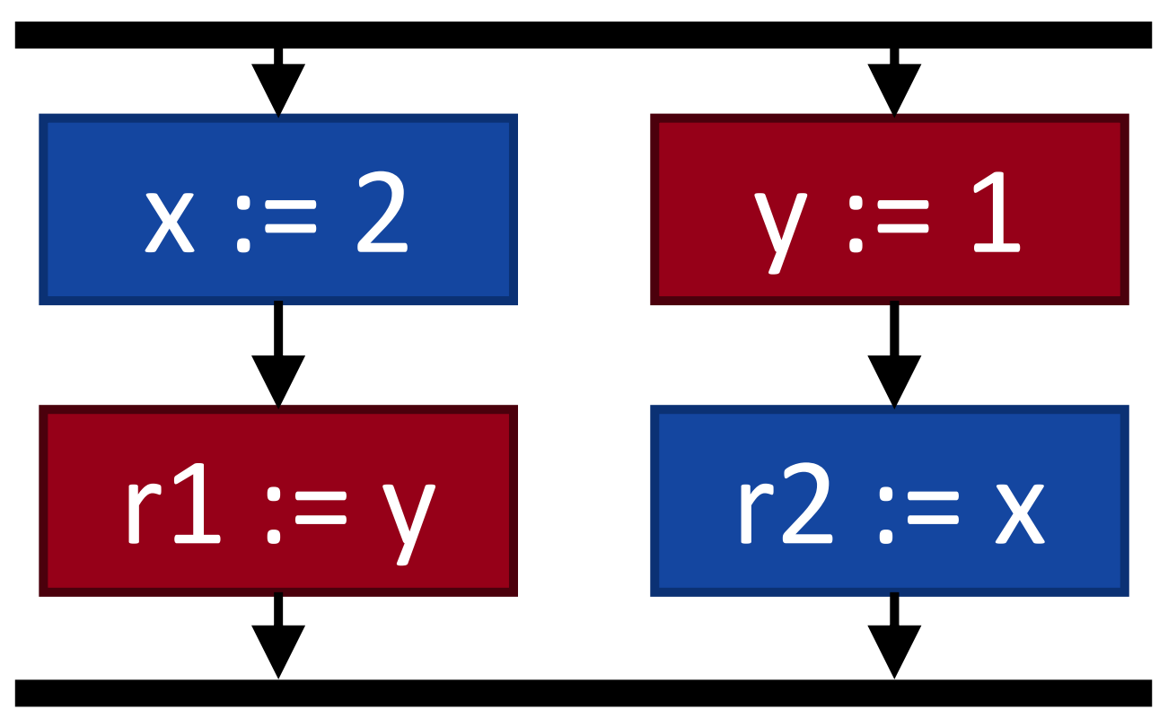
Speed-optimized approach:
Weak memory

$r1 == 1 \;\Rightarrow\; r2 == 2$



- **Same address accesses sequential**
- **Reordering distinct variable accesses**
- **Complex, but optimal**

Middle ground:
Total store order



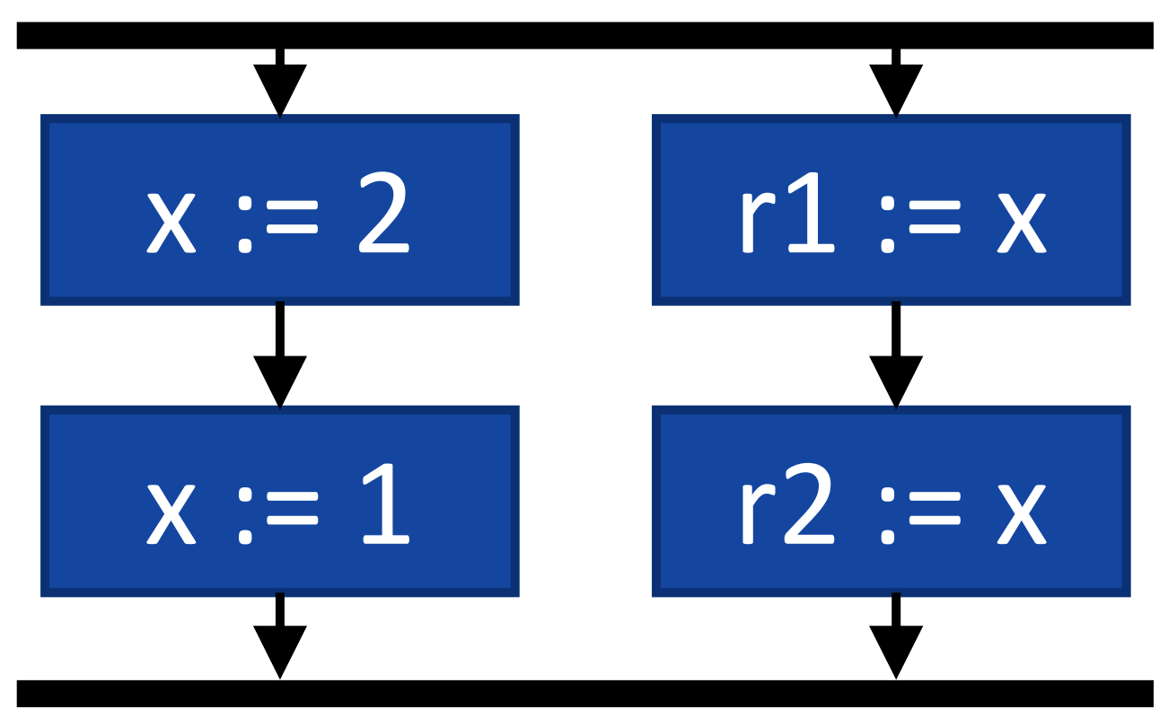
$r1 == 1 \;\not\Rightarrow\; r2 == 2$

- **Stores also sequential**
- **Reordering loads before stores**
- **Tradeoff: complexity vs. effectiveness**

Memory Consistency Problems

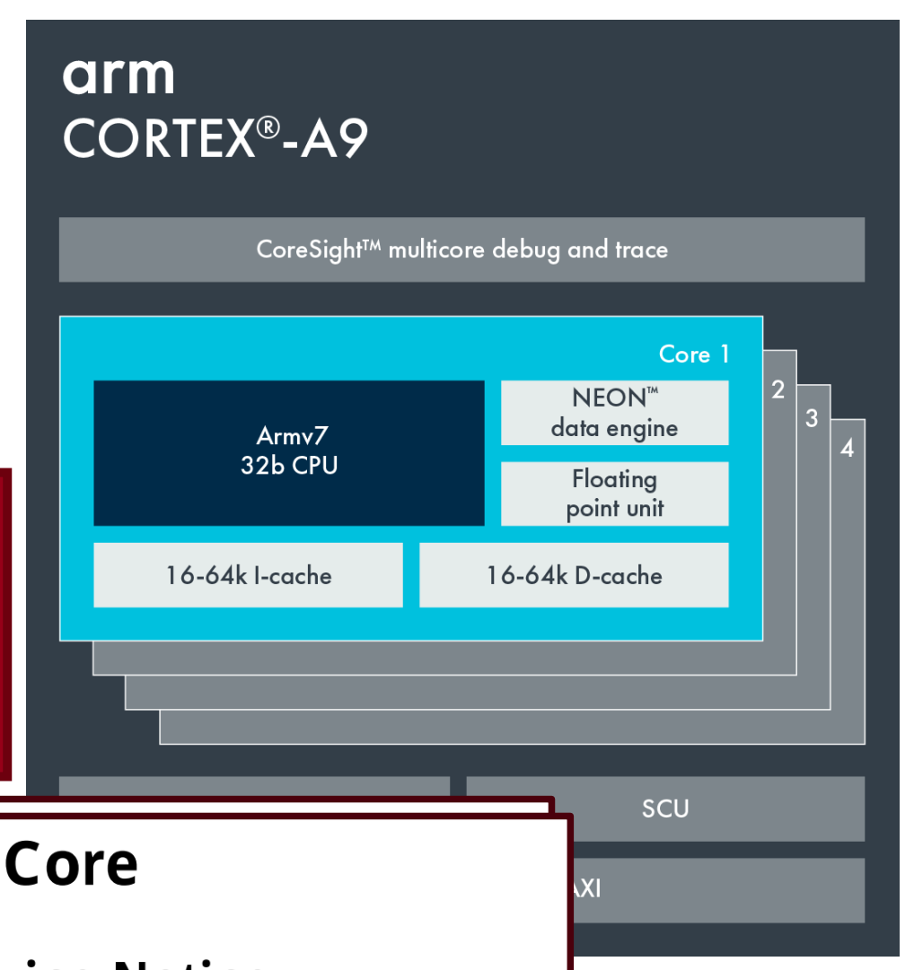
What can we find?

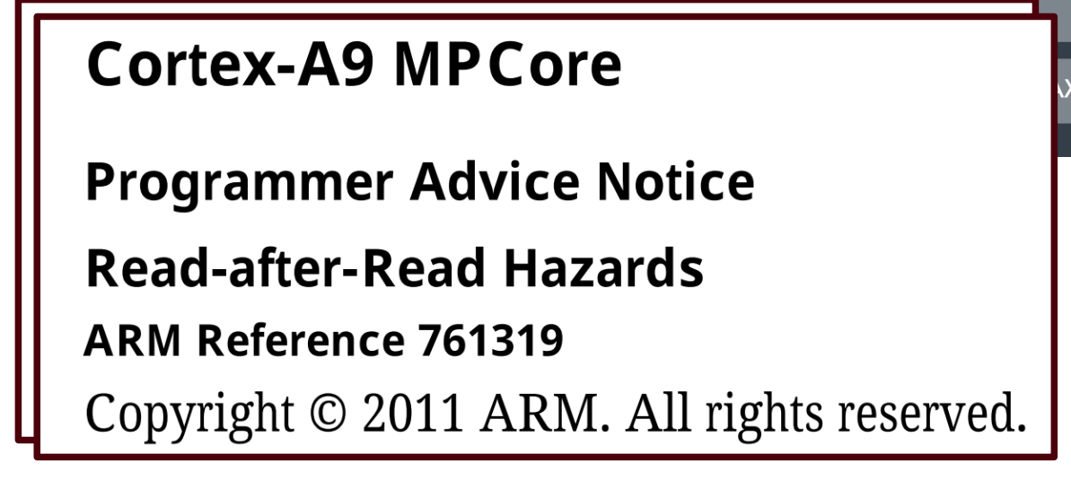
The frightening reality:
Fault latent for **3 years**



$(r1, r2) = (1, 0)$

Observable, but forbidden!





Cortex-A9 MPCore

Programmer Advice Notice

Read-after-Read Hazards

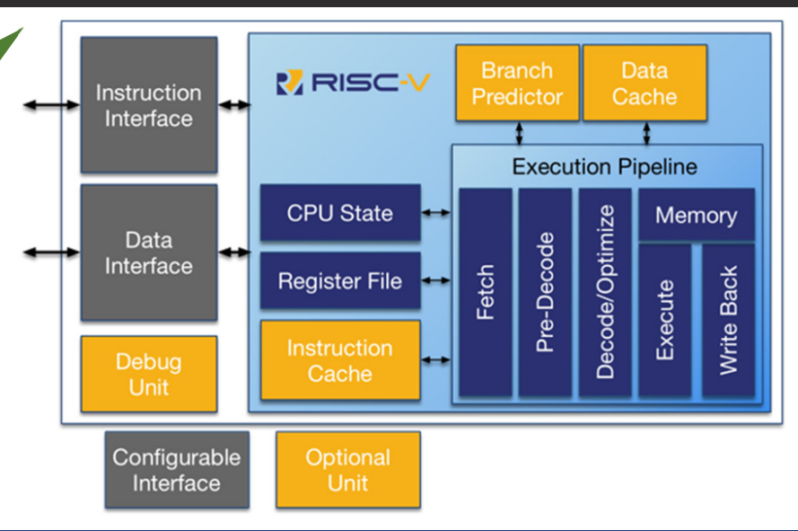
ARM Reference 761319

Copyright © 2011 ARM. All rights reserved.

- First released on **31 March 2008**
- Fault acknowledged in **2011**

Oversimplified Architecture:
RISC-V Case Study*

"a RISC-V-compliant microarchitecture allows 144 outcomes ***forbidden*** *by C11"*



Lack of Cumulative Lightweight Fences
Lack of Cumulative Heavyweight Fences
Reordering Loads to the Same Address

"our recommended solution to this problem is to ***modify the ISA****"*

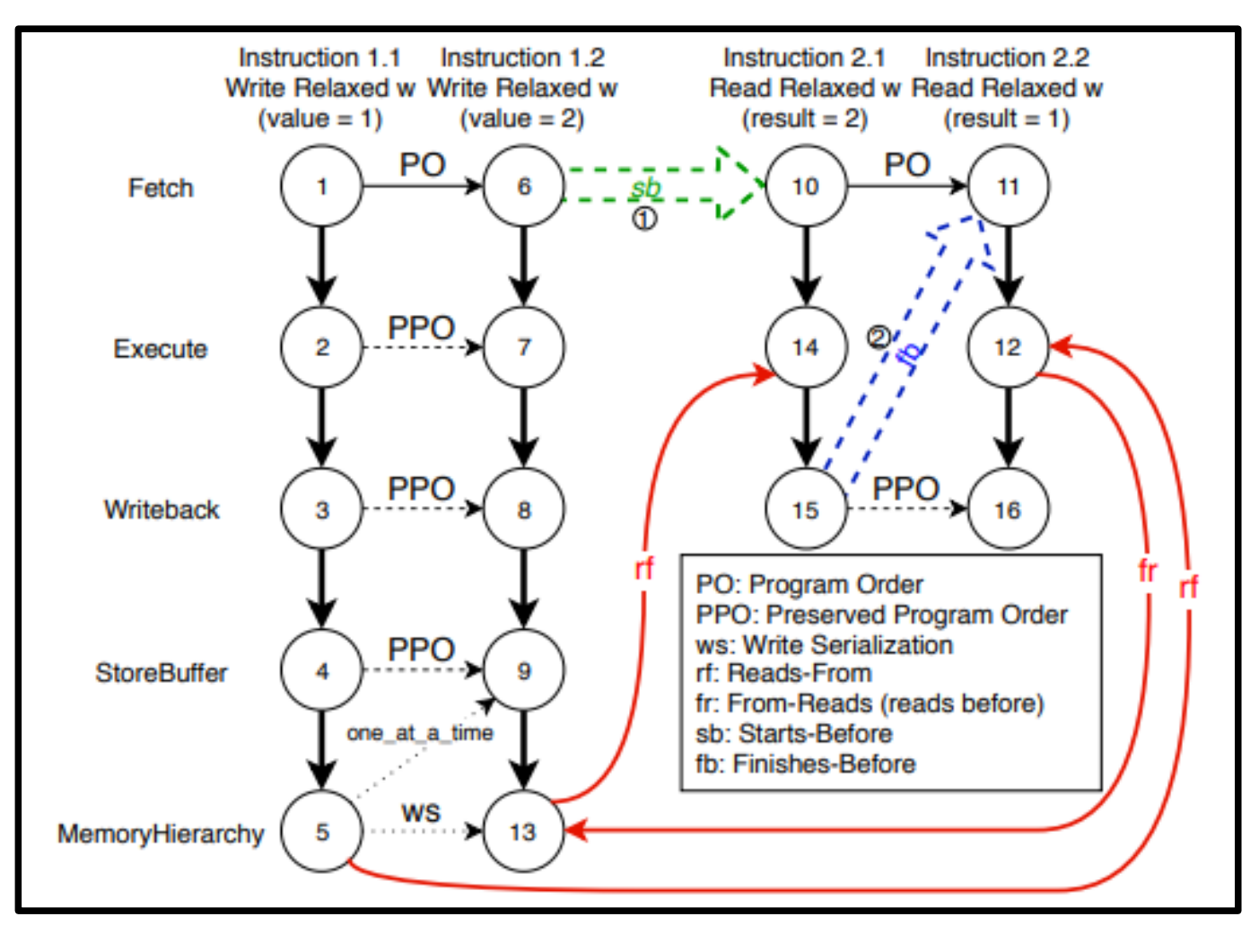
*: C. Trippel et al: *TriCheck: Memory Model Verification at the Trisection of Software, Hardware, and ISA*, ASPLOS 2017

Bridging the Gap between Hardware Design and Formal Verification

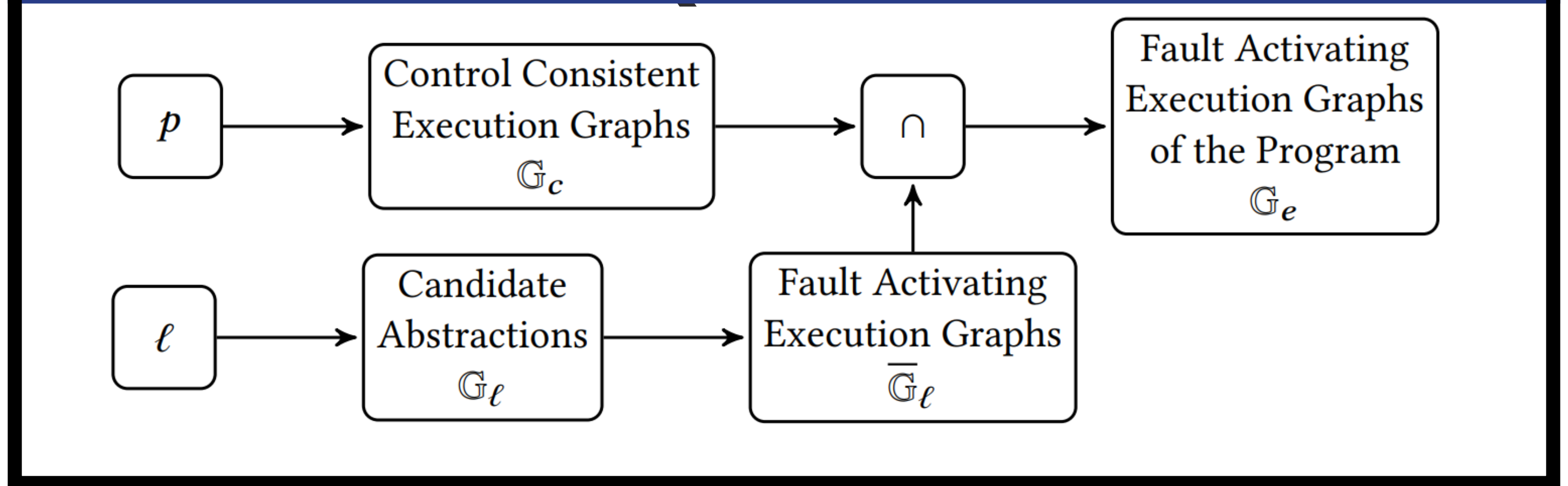
Hardware Design
Ordering along the Pipeline

Most model checkers
- **Sequential** ordering →
  incomplete state space discovery

Detailed view
- ***Happens-before*** edges
- Intra-core **dependencies**
- Inter-core **synchronization**

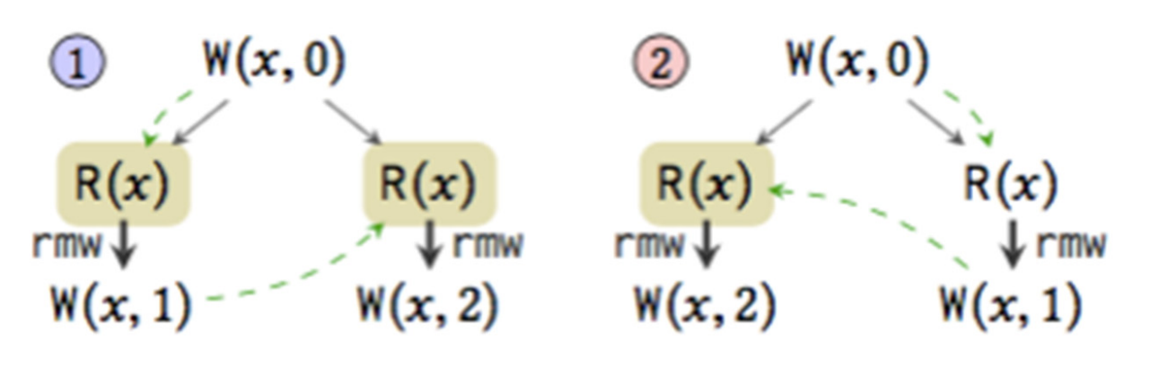

Instruction 1.1 Write Relaxed w (value = 1)
Instruction 1.2 Write Relaxed w (value = 2)
Instruction 2.1 Read Relaxed w (result = 2)
Instruction 2.2 Read Relaxed w (result = 1)
Fetch
Execute
Writeback
StoreBuffer
MemoryHierarchy
PO
PPO
WS
PO: Program Order
PPO: Preserved Program Order
ws: Write Serialization
rf: Reads-From
fr: From-Reads (reads before)
sb: Starts-Before
fb: Finishes-Before

Goal: ***Formal Analysis of Hardware Fault Activations in Concurrent Embedded Software***


p
Control Consistent Execution Graphs $\mathbb{G}_c$
∩
Fault Activating Execution Graphs of the Program $\mathbb{G}_e$
ℓ
Candidate Abstractions $\mathbb{G}_\ell$
Fault Activating Execution Graphs $\mathbb{G}_\ell$

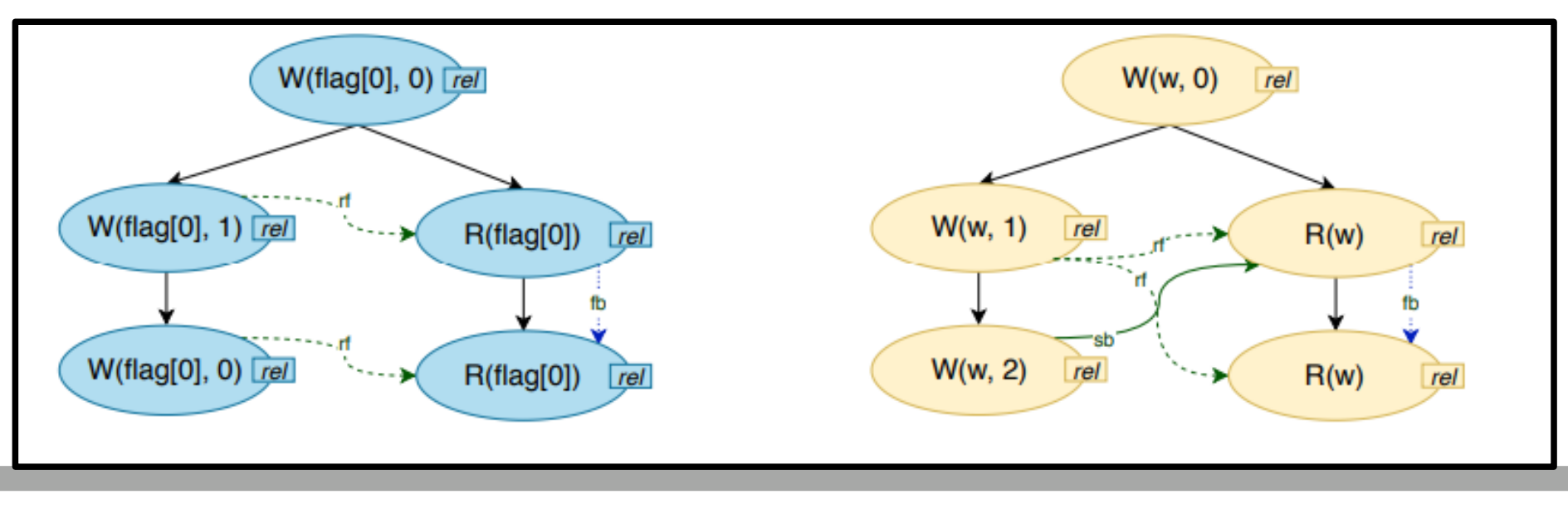
False negatives

Novel model checkers
- **Hardcoded weak** ordering →
  implies flawless processor



Source: M. Kokologiannakis et al: *Effective stateless model checking for C/C++ concurrency*. POPL 2017

System view
vs
Programmer's view



Abstract view
- **Execution graphs**
- ***Read-from*** edges

Formal Verification
Implied Memory Ordering

MŰEGYETEM 1782

ftsrg research group | ftsrg.mit.bme.hu