SAND2022-6612C

Eris: Fault Injection and Tracking Framework for Reliability Analysis of Open-Source Hardware

Shubham Nema, Justin Kirschner, Debpratim Adak, Sapan Agarwal, Ben Feinberg, Arun F. Rodrigues, Matthew Marinella, Amro Awad

snema@ncsu.edu

05/24/2022

NC STATE UNIVERSITY

Department of Electrical and Computer Engineering, North Carolina State University

ISPASS 2022

2022 IEEE International Symposium on Performance Analysis and System Software

ISPASS-22, National University of

Outline

- Introduction and Background
- Motivation
- Design
- Methodology
- Results
- Conclusion

Need for Reliability

Reliability in:

- Safety-Critical Systems
- Mission-Critical systems
- Server systems

is of utmost importance!!

Autonomous Self-Driving Cars

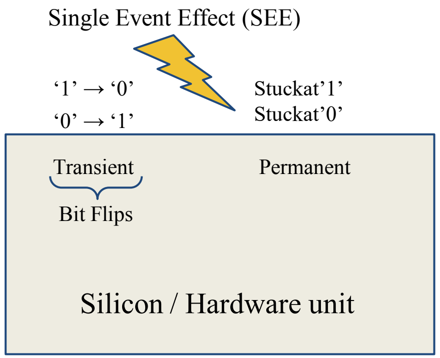
In-Flight Control System

QoS and data integrity and security in server/data centers

And Many More….

Introduction

- A ***fault*** is an undesirable change in the architectural state which may result in an error. ***Single event effects*** are responsible for these faults.
- ***Fault Injection*** (FI) is an empirical methodology to analyze system behavior during faults to assess reliability. FI can be carried out in:
- Vulnerability depends on cell characteristics and cross section layout
- Erroneous outcome can be:
  - Detectable-correctable
  - Detectable-unrecoverable (DUEs leading to *e.g.,* crash or hangs)
  - Non-detectable (silent data corruptions)



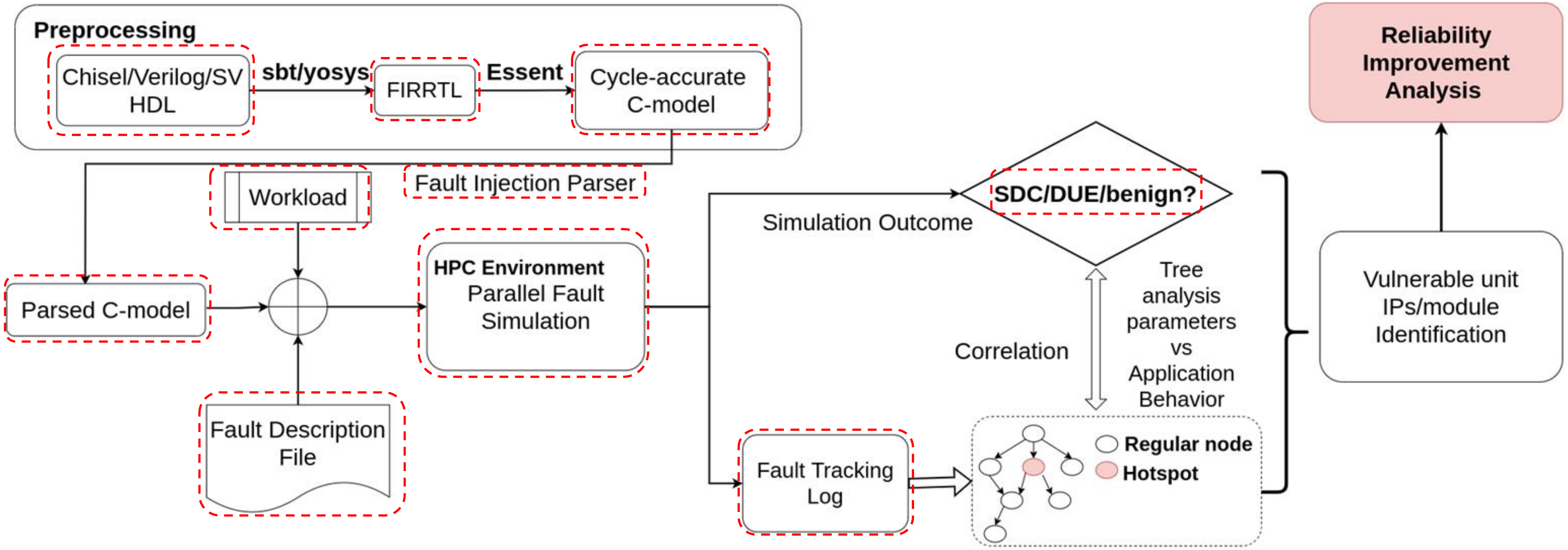
Prior Works

FI Techniques	
Hardware (*e.g.*, FIST, MARS) • Fault injection in fabricated component + Any ASIC Hardware - Limited controllability and repeatability - High cost - Post-fabrication (late in design cycle)	**Software (*e.g.*, LLFI, Xception)** • Fault injection using software running on design under test (DUT) + Controllable and repeatable + Low cost - Post-fabrication (late in design cycle)
Emulation (*e.g.*, Chiffre) • Fault injection in emulation hardware (*e.g.*, FPGA) + Any ASIC Hardware + Pre-fabrication (Early in design cycle) + Controllable and repeatable with hardware support - Requires additional logic in the emulated hardware - High cost	**Simulation (*e.g.*, GemFI)** • Fault injection into a simulated hardware model + Controllable and repeatable + Low cost + Pre-fabrication (Early in design cycle) + **Supports visibility into fault propagation** - **Often limited to abstract models** - **Slower than other techniques**

Eris Features

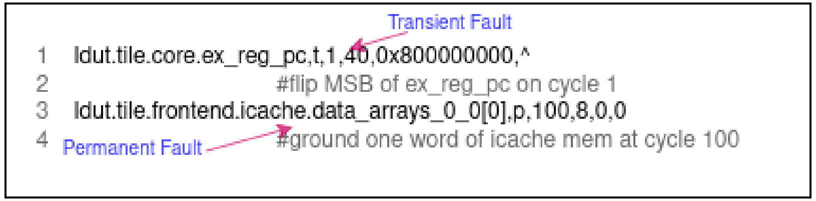
- Supports designs written in hardware description languages convertible to FIRRTL (Chisel, Verilog, VHDL)
- Novel fault tracking analysis enables identification of vulnerable components without directly injecting faults
- Supports targeted fault injection based on physical device characteristics and application profiling
- Supports control flow deviation detection

Eris Tool Flow



Eris Design

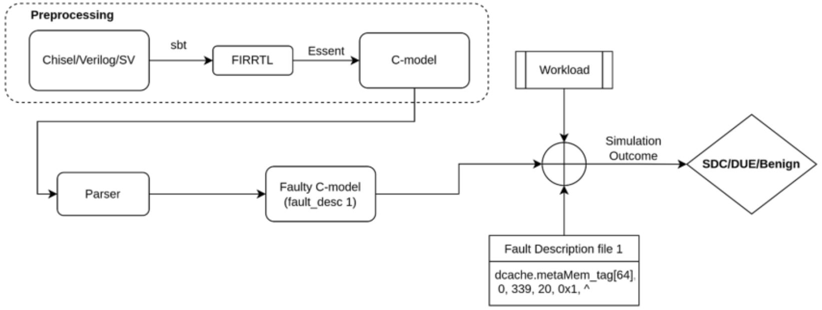
- The *parser* instruments the cycle-accurate C-model to enable fault injection
- The Essent or FIRRTL headers are modified to support fault injection and tracking
- The operators of each Essent data type are overloaded with the following fault metadata:
    - Unique index of faulted register
    - Source index
    - Cycle of propagation
    - Original non-corrupted data
    - Current fault status of register



```
ldut.tile.core.ex_reg_pc,t,1,40,0x800000000,^
                        #flip MSB of ex_reg_pc on cycle 1
ldut.tile.frontend.icache.data_arrays_0_0[0],p,100,8,0,0
                        #ground one word of icache mem at cycle 100
```

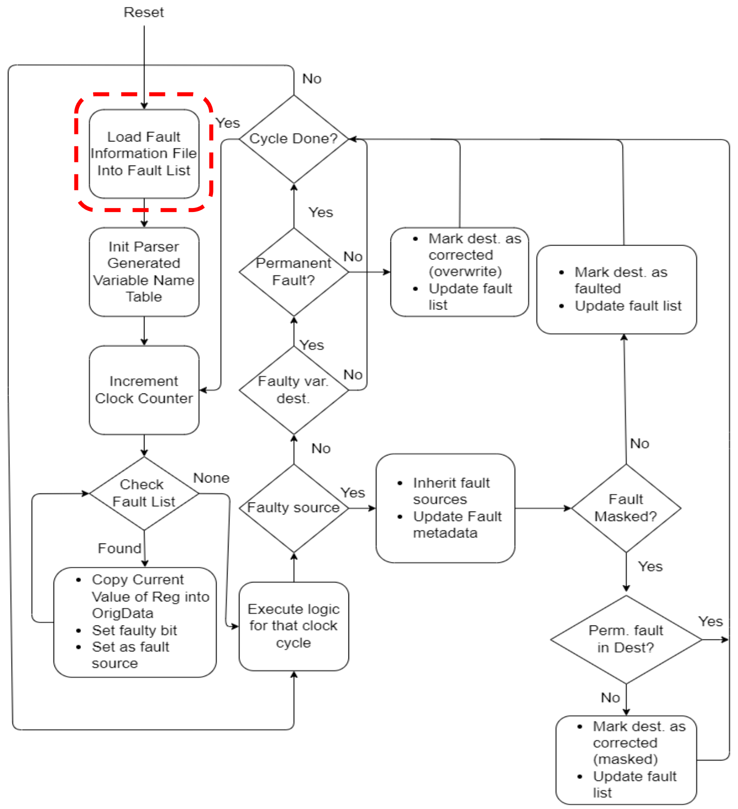
Transient Fault
Permanent Fault

Fault Information file



Eris: Single fault simulation flow

Fault Injection and Tracking Flow



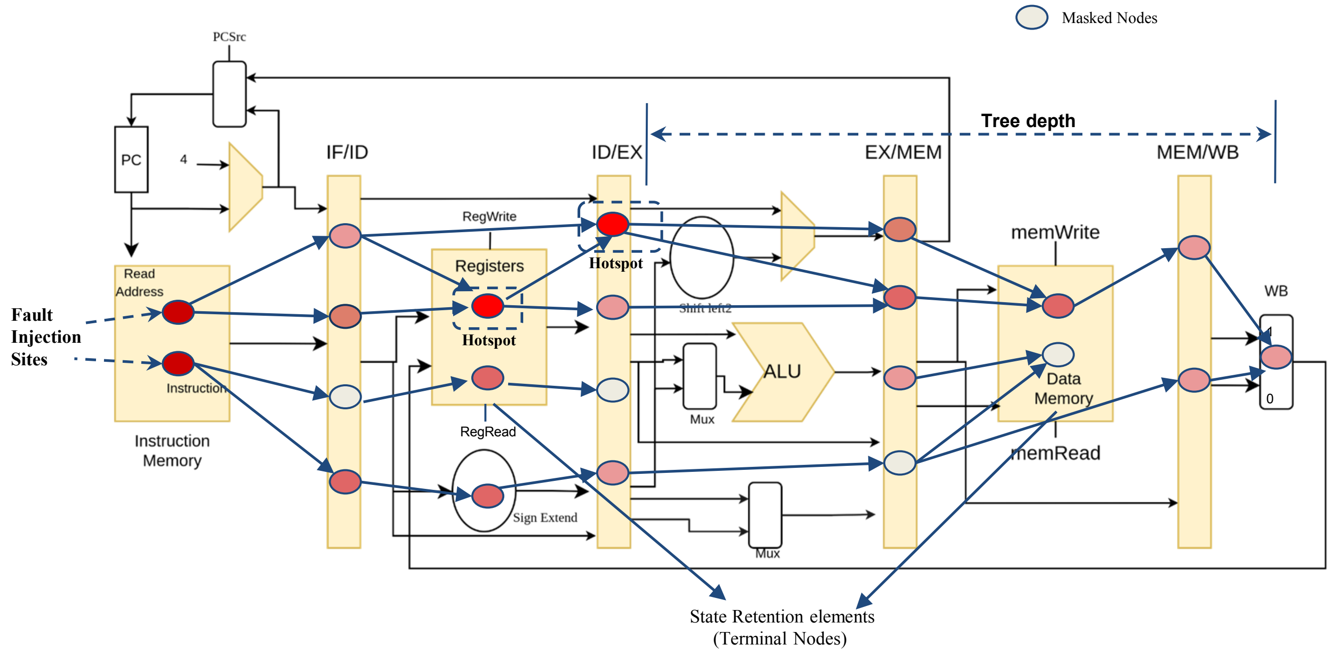
ISPASS 2022

Evaluation

Methodology:

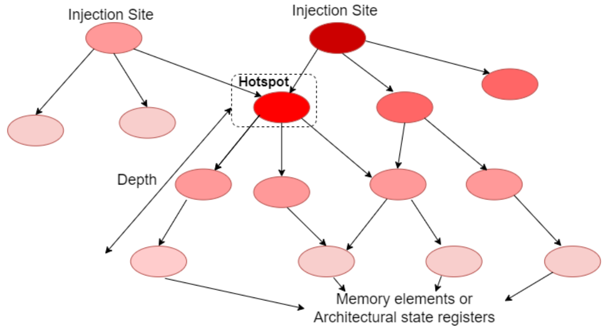
Evaluation Parameters	
Fault Simulation Design	Rocketchip SoC
Processor	Dual Rocketcore with private L1I & L1D and private TLB
Main Memory	256MB
Data Cache	16 KB
System Bus	Tile Link
Types of fault	Transient, Stuckat’0’ & Stuckat’1’
Benchmarks	Quick sort, Radix sort, Multithreaded matrix multiplication, Vector multiply
Fault Outcome	SDC, Crash, Hang, Benign, Garbled output

Building Fault Tree



Propagation Factor

- Propagation Factor (P.F) represents the contribution of an individual node to an SDC or DUE in the final program result.
- P.F is used to determine ***Hotspots***. A hotspot is an intermediate node that has an outsized contribution to error compared to its neighboring nodes.



$\sum P.F_{\mathbf{child}}$ / $\mathbf{C_{Self\ occurence}}$ / $\mathbf{C_{Parent}}$

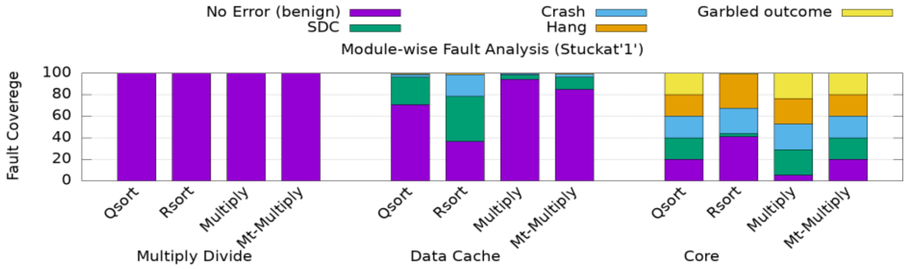
- Sum of the P.F's of the children nodes / Count of faults propagated from parents to itself. / Count of faults propagated to children.

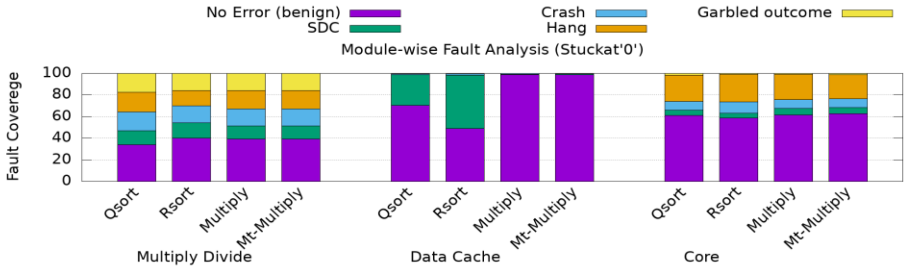
$$P.F_{\mathbf{node}} = \sum P.F_{\mathbf{child}} + C_{\mathbf{Self\ occurence}} + 2 * C_{\mathbf{Parent}}$$

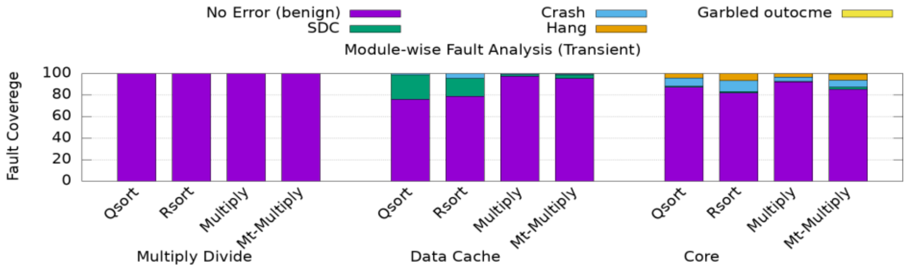
where,
$P.F_{node}$: Propagation factor of node under examination
$P.F_{child}$: Propagation factor of child node
$C_{self\_occurence}$: Number of times a fault propagates into the node
C_{parent} : Number of times a node propagates fault to child nodes

Transient and Permanent Fault Analysis

- Results are from 2500 FI simulations for each targeted module
- **SDC** → Checksum mismatch of final program data output
- **Garbled outcome** → Random data outcome rather than incorrect checksum
- **More results in the paper**



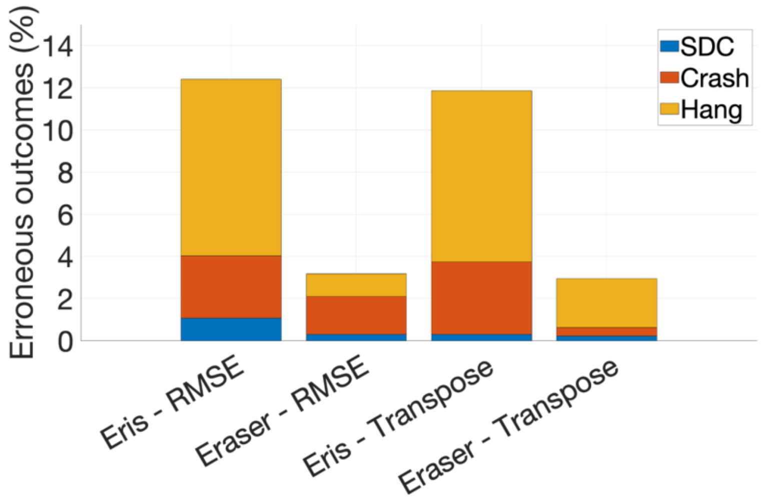




ISPASS 2022

Targeted Injection Metrics

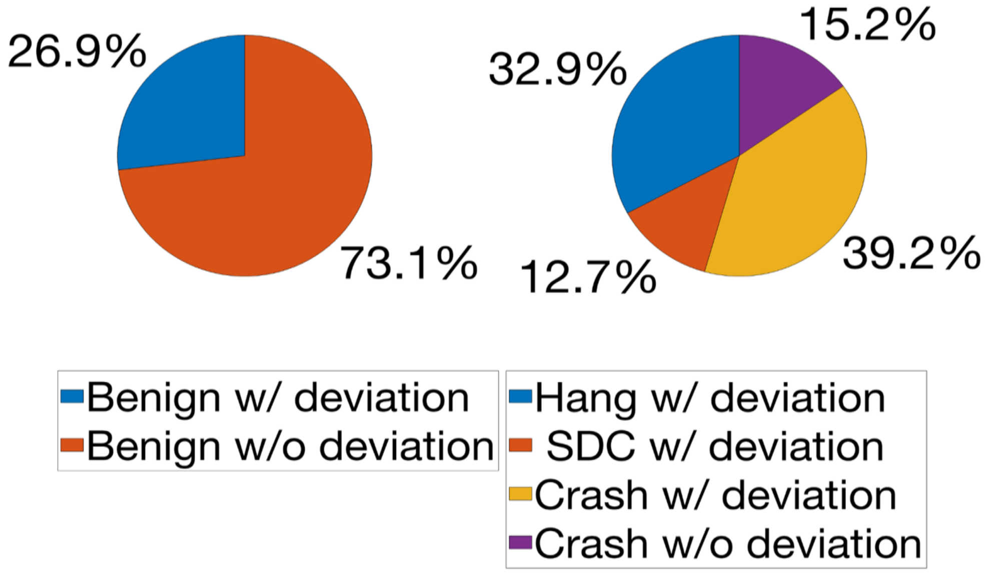
- ***Eris*** → Target registers for FI based on the register accesses count for a simulated application
- ERASER → Target registers for FI based number of cycles where the data is resident in each register
- ***Eris*** shows more erroneous outcomes compared to ERASER for same number of FI simulations



ERASER: Early-Stage Reliability and Security Evaluation for RISCV **[DSN ’21]**

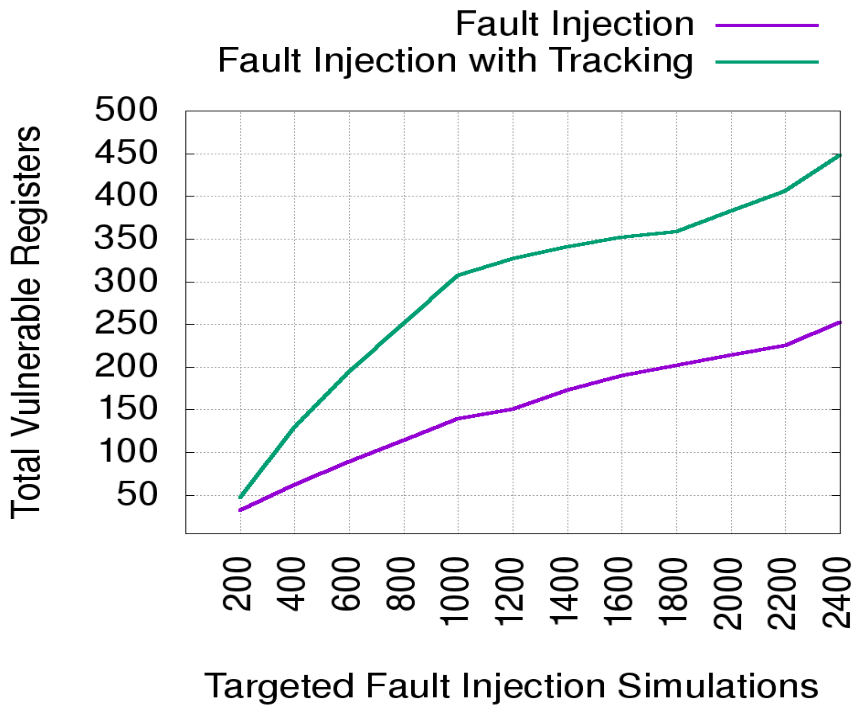
Analysis of Control Flow Deviation

- Control flow deviation is detected as change in the register access pattern
- Not all control flow deviations may result in error

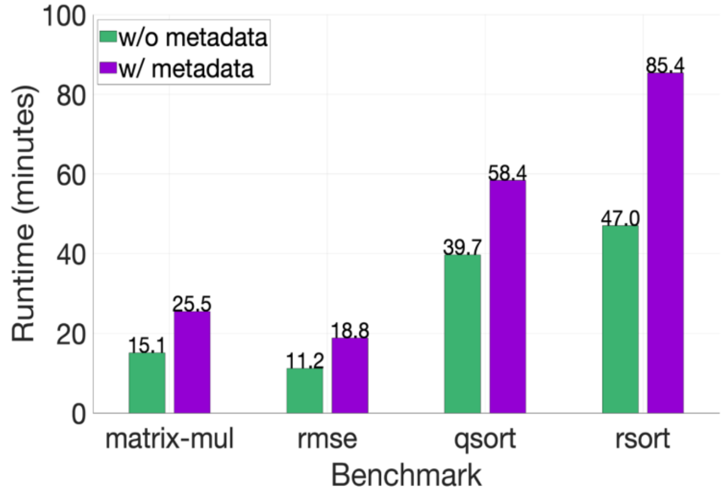


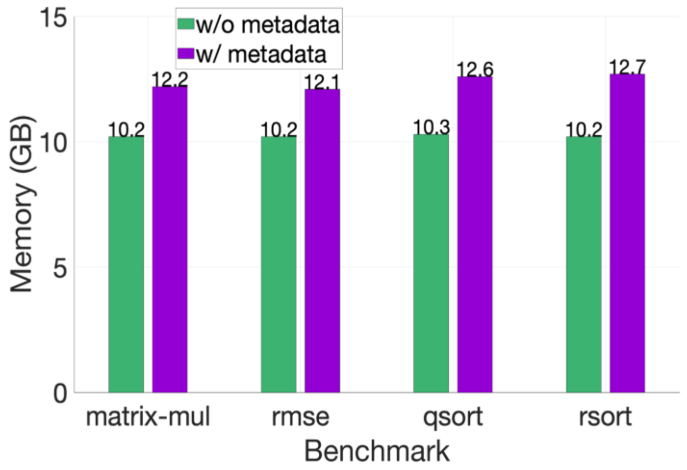
Fault Tracking Efficacy

- Fault tracking using P.F finds **78% more** vulnerable registers
- Without fault tracking, only registers that are directly injected with faults can be determined as vulnerable



Tracking Overhead





- **82% increase** in simulation time due to duplicate computation
- **18.6% increase** in memory overhead due to tracking metadata

Conclusion

- ***Eris*** enables early-stage reliability analysis of RTL designs (Chisel, Verilog, VHDL).
- ***Eris*** supports random or targeted injection of both transient or permanent faults. Targeting is based on application profiling.
- ***Eris*** can identify control flow deviation due to injected faults
- Novel fault tracking capabilities identifies **78% more vulnerable registers** in the same number of FI simulations.

https://github.com/amroawad2/Eris