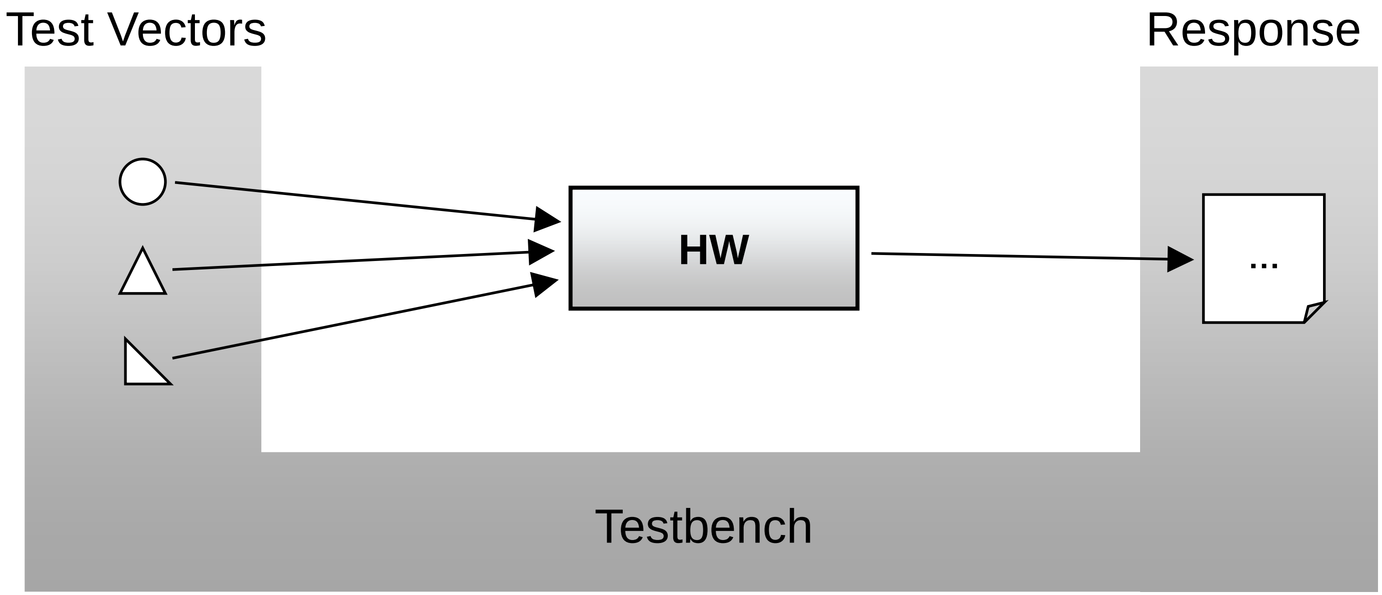
HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

QEMU Based Fault Effect Analysis for RISC-V

Peer Adelt, Bastian Koppelmann, Wolfgang Müller, Christoph Scheytt

Heinz Nixdorf Institute – Paderborn, Germany

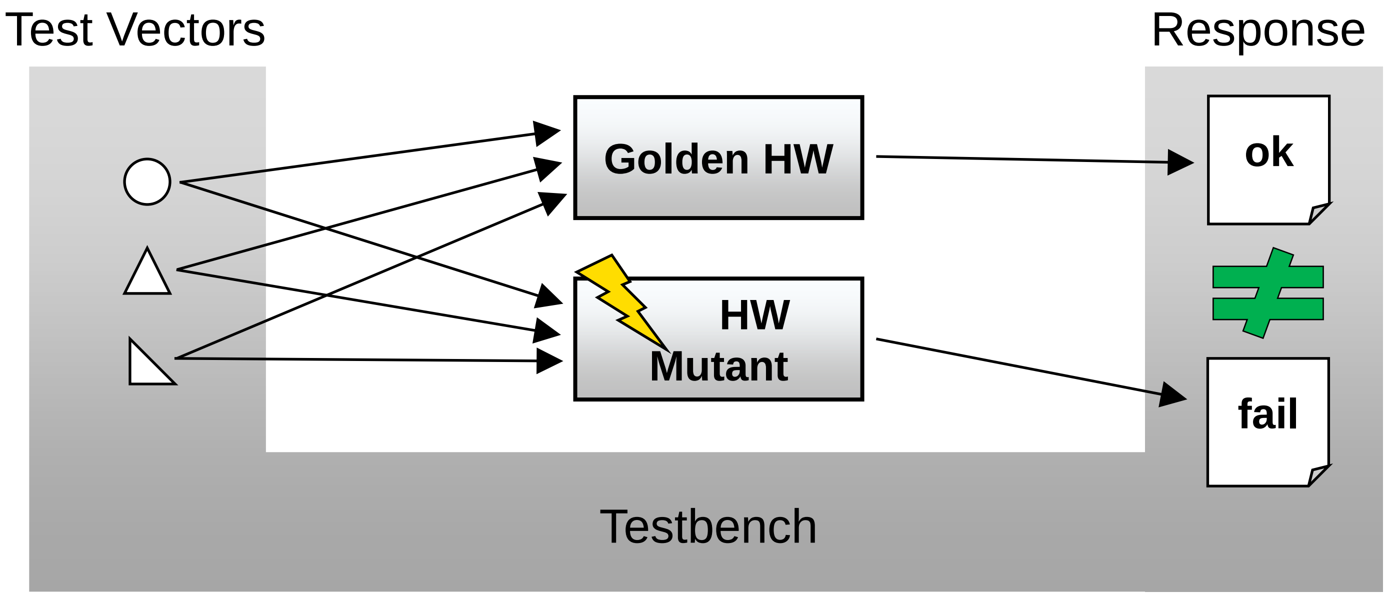
Hardware Simulation



- **HW:** Model of the HW under test
- **Stimuli:** Test vectors for the HW
- **Response:** Any externally observable HW reaction during simulation

➢ Response is compared with expected results after simulation to validate correct behaviour of the HW

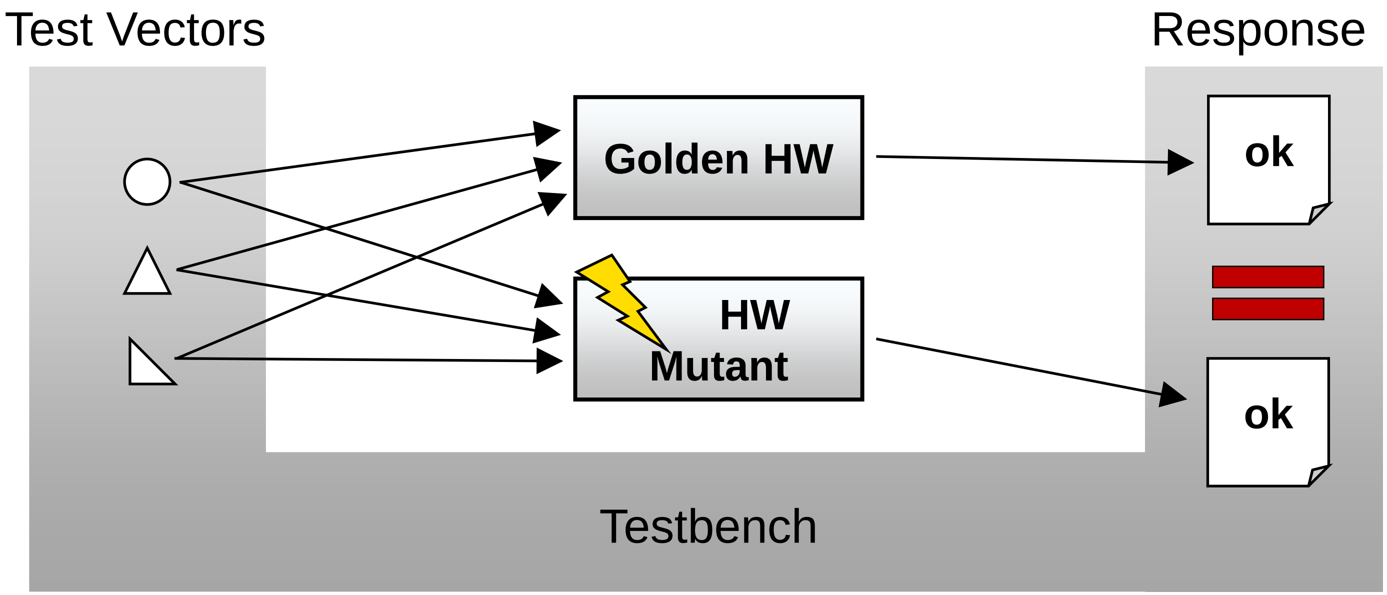
HW Fault Simulation



Mutant Killed: Response differs from golden response

- test vector can detect the fault → useful test vector → keep it

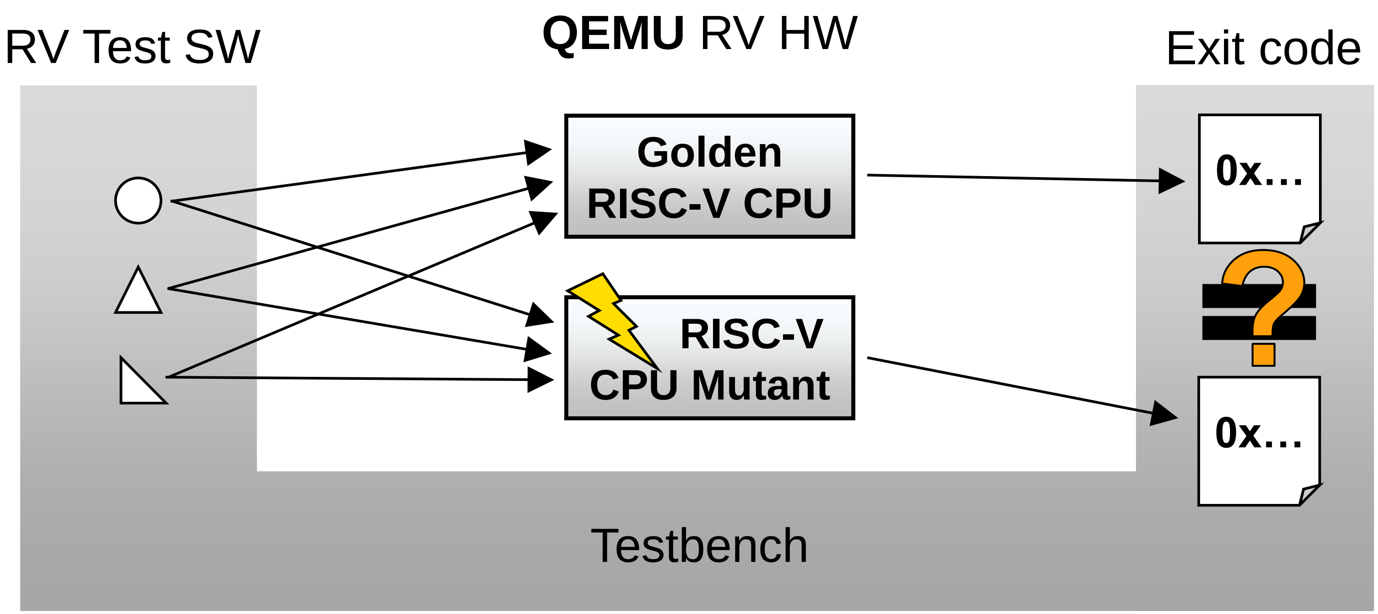
HW Fault Simulation



Mutant Not Killed: Same output as golden response

- test vector cannot detect the fault → useless test vector → discard

RISC-V QEMU Fault Simulation



- **Golden RISC-V CPU**
- **RISC-V CPU Mutant:**

  Copy of the Golden RISC-V CPU model an inject fault
- **Exit code:** different signature, exception, …

Our Fault Model: Bit Faults in Microprocessors

Faults in RISC-V CPU registers (GPR, CSR, Instructions) and Memory

Bit-level fault analysis for:

- **Instructions** (Opcode, Operands)
  - Pipeline-related faults
  - Instruction cache faults
  - Instruction memory faults
- **General Purpose Registers (GPR)**
- **Control and Status Registers (CSR)**
  - Core CSRs
  - MM CSRs (Device registers; memory-mapped)
- **Memory (incl. MMIO)**

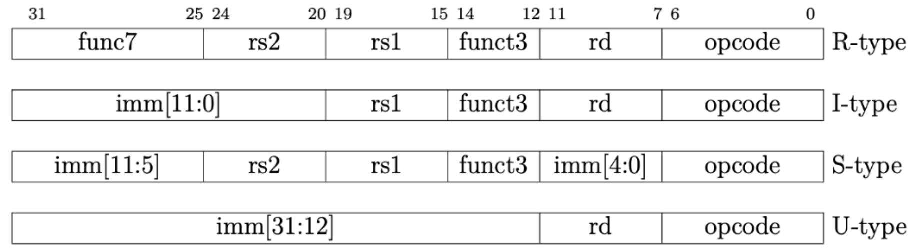
Permanent Stuck-Ats-0/1 in RISC-V 32 Bit Microprocessors

Fault Models

Permanent faults: Stuck-at-1/0 and Bit-Flip

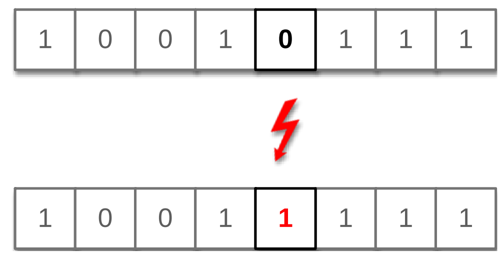
Instruction Faults (Opcode and Operand):

- 32 Bit Instructions
- #Permanent 1bit Faults: **32 * <#Instruction Instances>**



GPR Faults:

- 31 Registers á 32 Bit
- #Permanent 1bit Faults: **32 * <#GPRs> = 992**



CSR Faults (Core CSRs, MM CSRs):

- Register Count is implementation specific, Word Length: 32 Bit
- #Permanent 1bit Faults: **32 * <#CSRs>**

Memory Faults (incl. MMIO):

- Memory divided into Bytes
- #Permanent 1bit Faults: **8 * <#Memory Bytes>**

Based on static memory content initialization

Transient Bit-Flips in RISC-V 32 Bit Microprocessors

Fault Models

Transient faults: Bit-Flip

Instruction Faults (Opcode and Operand):

- 32 Bit Instructions
- #Transient 1bit Faults: **32 * <#Instruction Executions>** → Loops, Jumps, Branches, etc.

GPR Faults:

- 31 Registers á 32 Bit
- #Transient 1bit Faults: **32 * <#GPR Executions>**

CSR Faults (Core CSRs, MM CSRs):

- Register Count is implementation specific, Word Length: 32 Bit
- #Transient 1bit Faults: **32 * <#CSR Executions>**

Memory Faults (incl. MMIO):

- Memory divided into Bytes
- #Transient 1bit Faults: **8 * <#Memory Bytes * #Executions>**

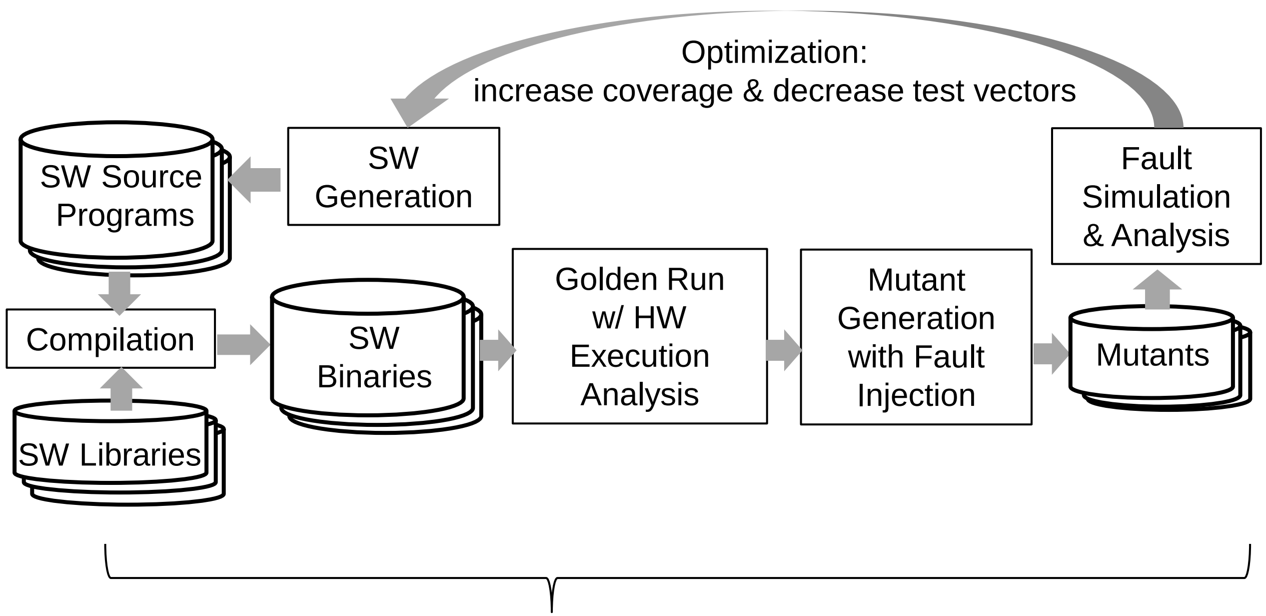
Issue: #Transient Faults explode, because each execution creates a separate mutant

Scalable Fault Effect Analysis for RISC-V with QEMU

Framework Overview

Phases of Fault Effect Analysis for RISC-V (FEAR-V):

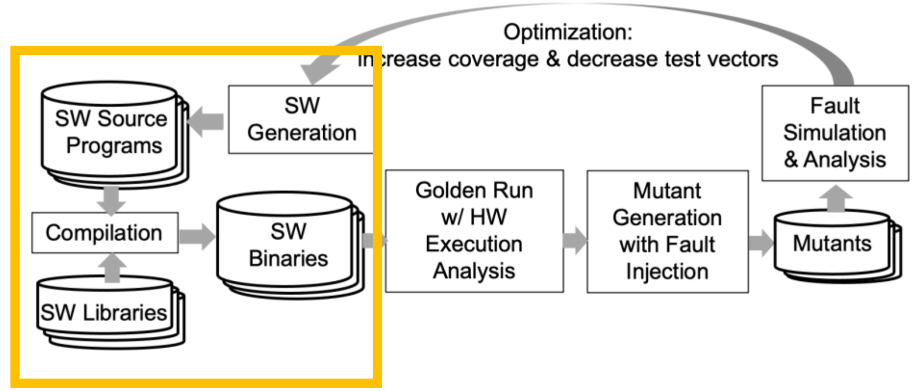
1. Software Generation & Compilation
2. Golden Run w/ HW Execution Analysis
3. Mutant Generation with Fault Injection
4. Fault Simulation & Analysis & Optimization



Configurable: RV ISA subset, iterations, n-bit faults, transient/permantent, fault model, …

Test Software Generation

For different RISC-V ISA Subsets



1. **Software Generation & Compilation:**

SW Generators start with basic parameters

- Csmith SW Generator * (University of Utah)
- Torture SW Generators * (University of California, Berkeley)

SW Test Libraries

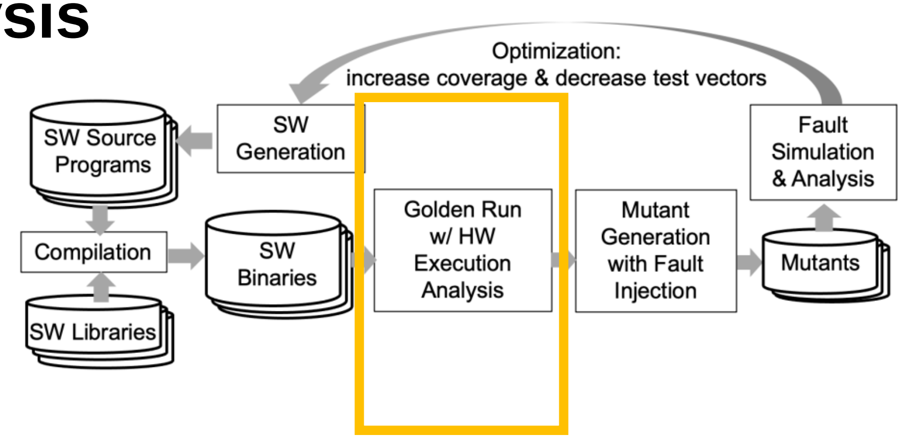
- RISC-V – Tests (University of California, Berkeley)
- RISC-V Architecture Tests * (RISC-V Foundation Architecture Test SIG)
- UPB CSR Tests (Paderborn University)

Compiler: RISC-V GCC.

* with signatures

ISA and Register Execution Analysis

For different RISC-V ISA Subsets



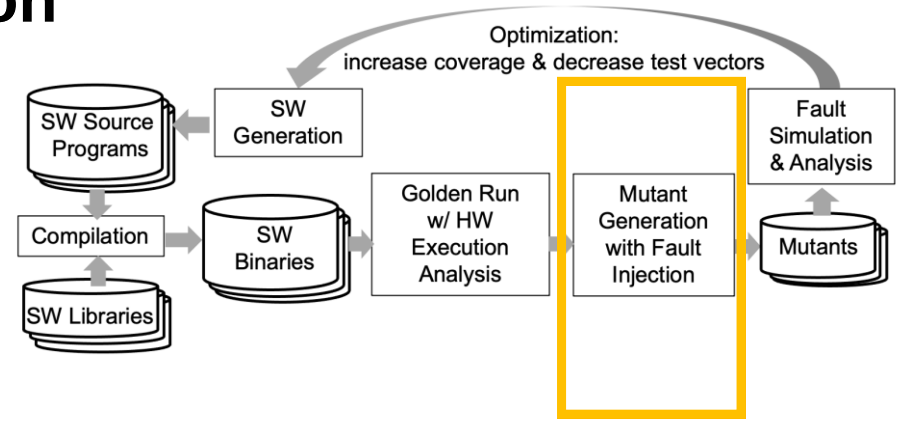
2. **Golden Run w/ HW Execution Analysis:**
   - Which and how many (%) Instructions are executed?
   - Which and how many (%) GPRs and CSRs are executed?
   - Additional properties/statistics:
     - #Instructions (types, executions)
     - Lines of code
     - Program Execution time
     - Memory Execution
     - …

Example:
- ISA: RV32GC
- 7 Csmith Programs

Analysis/ SWProgram	P01	P02	P03	P04	P05	P06	P07
Instr. Type (#)	59	59	58	57	59	62	58
Instr. Type (%)	38,1	38,1	37,4	36,8	38,1	40	37,4
GPR Cov (#)	30	30	30	27	27	30	27
GPR Cov (%)	96,8	96,8	96,8	87,1	87,1	96,8	87,1
CSR Cov (#)	5	5	5	5	5	5	5
Total LoC (#)	4071	3385	2455	2436	2428	3575	3735

Mutant Generation & Fault Injection

With automatic Mutant Reduction

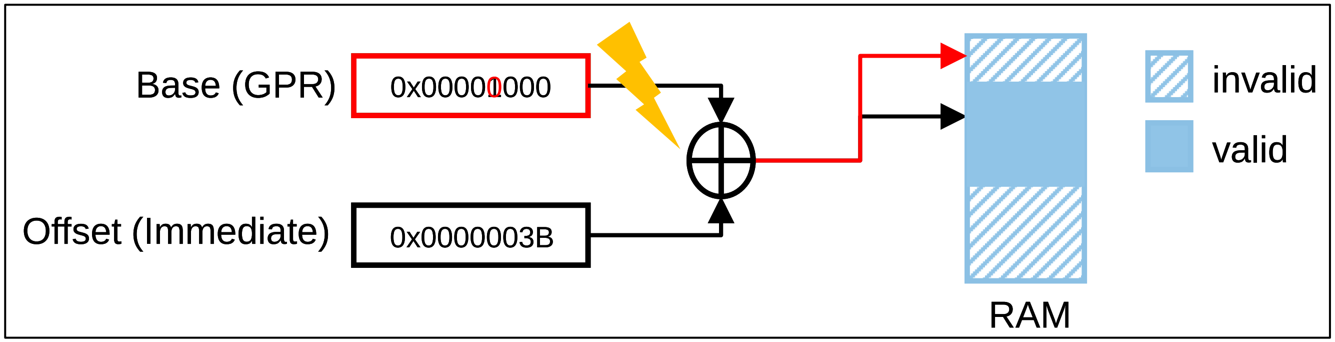


3. **Mutant Generation w/ Fault Injection:**
   - Estimate fault simulation time
   - Inject permanent / transient n-bit faults:

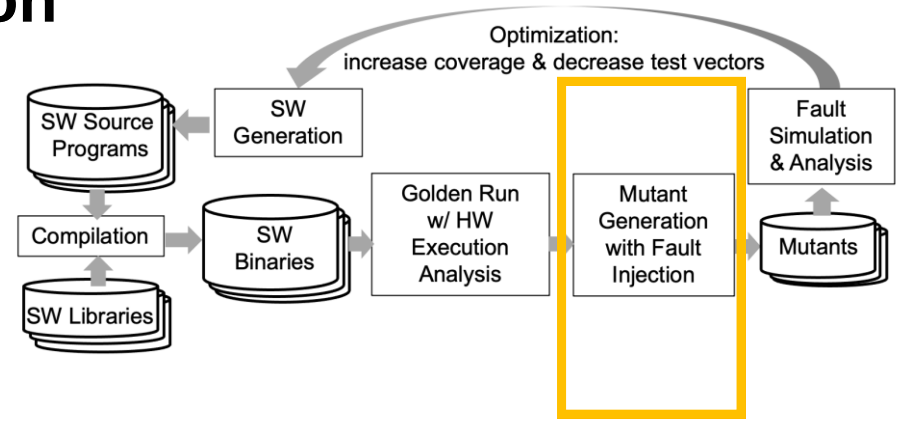
     Instructions (Opcode & Operands), GPRs, CSRs, Memory (incl. MMIO)

Mutant reduction → reduce #simulation runs/time

- Discard mutants for non-executed Opcodes & Registers
- Discard mutants accessing invalid memory when memory protection (MMU/MPU) is available



Mutant Generation & Fault Injection
With automatic Mutant Reduction



Mutant Reduction

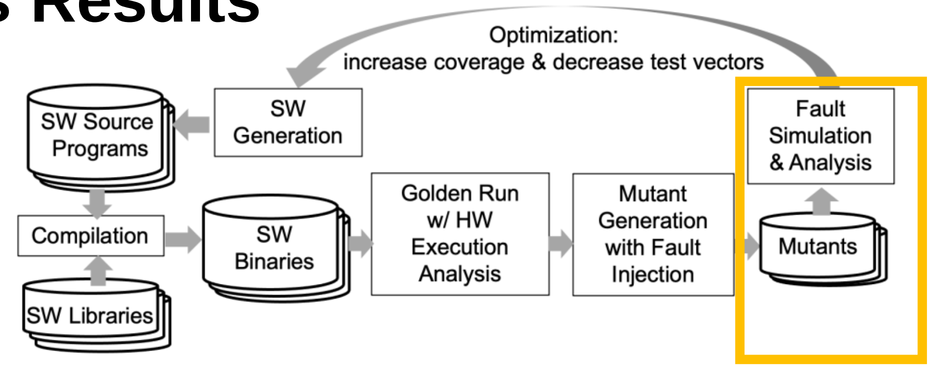
- Discard mutants for non-executed Opcodes & Registers
  - <u>Example:</u> Only 30 of 31 GPRs are executed,
    → 960 instead of 992 permanent GPR mutants
- Discard mutants executing invalid memory when memory protection is available
  - <u>Example:</u> SW has 1.943Mio GPR executions, from which 1.607M are Load/Store instructions for invalid memory.
    → Only 0.335Mio GPR mutants (~17%) need to be simulated.

Mutants/ SWProg.		P01
Permanent	IFR (#)	32
	GPR (#)	960
	CSR (#)	160
Transient	Instr (#)	99824
	GPR (#)	1943104 (335936)
	CSR (#)	21792
Total (#)		2065872 (458704)

Fault Effect Simulation & Analysis Results

Mutant Execution

4. **Fault Simulation & Analysis:**
   - Simulate the reduced set of mutants from previous step
   - exit code for each mutant



Optimization based on min setcover of killed/timeout mutants
optional weights: simulation time, instruction executions or program size

Exit codes:

- **Timeout:** Endless loops, etc. → Watchdog
- **Killed:**
  - RISC-V Standard Exceptions
    (unaligned memory access, illegal instruction, misaligned instruction, access fault, load misaligned, store address misaligned, etc.)
  - Different Signature
    (Golden output != mutant output)
- **Not Killed:** no observable faulty behavior
  - to be discarded by setcover optimization

  → Improve HW/SW safety measures

Termination/ SWProgram	P01
Not Killed (#)	238568
Timeout (#)	6412
Killed (#)	213724

Not Killed/ Fault Type		P01
Permanent	IFR (#)	0
	GPR (#)	468
	CSR (#)	126
Transient	IFR (#)	54024
	GPR (#)	162819
	CSR (#)	21131

Fault Effect Analysis for RISC-V (FEAR)

Example: Analysis of 24 Test Programs from RISCV-Torture for Freedom E300

Golden Run/Programs	P1	P2	P3	P4	P5		P24
Memory Executions (#)	344366	342190	343317	343433	343936	…	343565
GPR Cov (#)	31	31	31	31	31	…	31
GPR Cov (%)	100.00	100.00	100.00	100.00	100.00	…	100.00
GPR Exe (#)	1509760	1506573	1507902	1508039	1506600	…	1503688
CSR Cov (#)	9	9	9	9	9	…	9
Instr. Type Cov (#)	76	75	76	75	77	…	76
Instr. Type Cov (%)	84.4%	83.3%	84.4%	83.3%	85.5%	…	84.4%
Instr. Type Total (#)	90	90	90	90	90	…	90
Instr. LoC (#)	1526	1479	1493	1500	1480	…	1480
Instr. Executions (#)	807061	802448	804558	804508	804066	…	802807
Prg Execution Time (us)	8168	6399	6863	10375	7312	...	6577

Exit Codes/Programs	P1	P2	P3	P4	P5		P24
Not Kill. (#)	19676	18729	19574	19505	18168	…	19101
Timeout (#)	594	772	799	414	25694	…	1120
Killed (#)	25948	25421	24901	25995	26317	…	24877
Total (#)	46218	44922	45274	45914	45162	…	45098

Total #Instr. Executions: 19,279,825

Total #Mutants: 1,081,200

→ Fast Fault Simulation: 3776.54 MIPS (on 24 Threads *)

(Total Sim. Time: 229 seconds)

* AMD Ryzen Threadripper PRO 3945WX System

Summary & Next Step

Designed and implemented a mutation testing framework for Fault Effect Analysis for RISC-V (FEAR-V):

- Based on QEMU
- SW Library and Generation frontend
- Highly Configurable with YAML
  (ISA subsets, 1/2/..n-bit faults, bit-flip, SA-1/0, transient/permanent, …)
- Fast Fault Simulation Time (> 3000 MIPS)
  fast enough for up to 2-bit faults for RISC-V processors
  fast enough for transient faults
  fast enough to run multiple iterations for test vector optimization

Next Step:

- **Chip Layout dependent fault generation**
  Based on observation:
  1:1 relationship between QEMU bit variable and flipflop in chip layout

Thank you for your attention

Peer Adelt, Bastian Koppelmann, Wolfgang Müller, Christoph Scheytt

Heinz Nixdorf Institute – Paderborn, Germany

Backup: TCG with Fault Injection

QEMU Fault Injection