Enhancing Privilege Architecture Support in RISC-V ISAC (RISCOF)

Muhammad Hammad Bashir[1,3], Umer Shahid[1,3], Allen Baum[2] , Pawan Kumar Sanjaya[4]

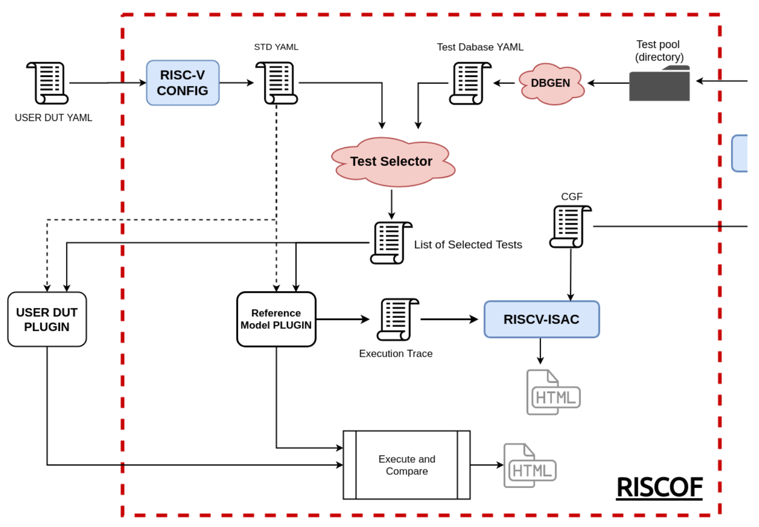
[1]10xEngineers
[2]Esperanto Technologies
[3]Department of Electrical Engineering, U.E.T Lahore
[4]Department of Computer Science, University of Toronto

Abstract

RISCOF, a Python-based framework, ensures RISC-V processor implementations comply with instruction set simulators like Spike and Sail. It supports both manual and automated test suite generation via RISC-V CTG, with coverage analysis performed through RISC-V ISAC. However, the coverage analysis of privilege architectural tests has been limited due to incomplete support in RISC-V ISAC. To address this, we have introduced new features in RISC-V ISAC specifically for privileged architecture, along with a more efficient method for writing coverpoints. These enhancements aim to improve compliance testing comprehensively.

Overview of RISCOF and RISC-V ISAC

- **RISCOF** - The RISC-V Compatibility Framework is a Python-based tool for testing RISC-V targets (hardware and software) against a standard RISC-V reference model using architectural assembly tests.
- **RISC-V ISAC**, part of the RISCOF Framework, is an open-source tool that verifies the thoroughness and quality of these test suites. It defines and checks ISA-level coverpoints, generating comprehensive coverage and data propagation reports for accurate testing.



Enhanced Features

The necessary track variables and functions for the Privileged Architecture were added in the RISC-V ISAC. The features like the macros support and other useful features for the Virtual Memory Support including SV32, SV39, SV48 and SV59 are mentioned below:

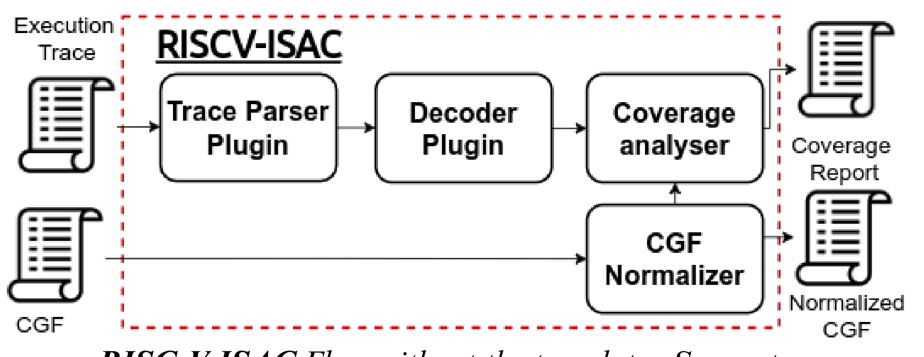
Feature	Coverpoint Format	Support
mode flag	mode == 'M'	New
get_addr()	get_addr('label_name')	New
mem_val()	mem_val(address in hex/dec, number_of_bytes_to_fetch)	New
mode_change	mode_change == "S to M"	New
rd_val	rd_val == "value in hex/dec"	New
PTE Property Check	get_pte_prop(prop_name, pte_addr, pgtb_addr)	New
Virtual and Physical Address Variables	iepa, ieva, depa, deva	New
Page Table Walk Variables	iptw0a ... iptw4a, iptw0cont ... iptw4cont	New

Translator support in RISC-V ISAC

- Translator support introduces a more concise format to define coverpoints, utilizing a variety of rules and operations, including ranges, macros, placeholders, loops, enumeration with operations, and advanced range based enumeration to reduce redundancy in user-defined coverpoints.
- The first coverpoint under the label *csr_comb* label will be evaluated to total of 8 coverpoints since the maxium range is 8. The *${custom_macro}* can be used across multiple coverpoints to reduce redundancy.

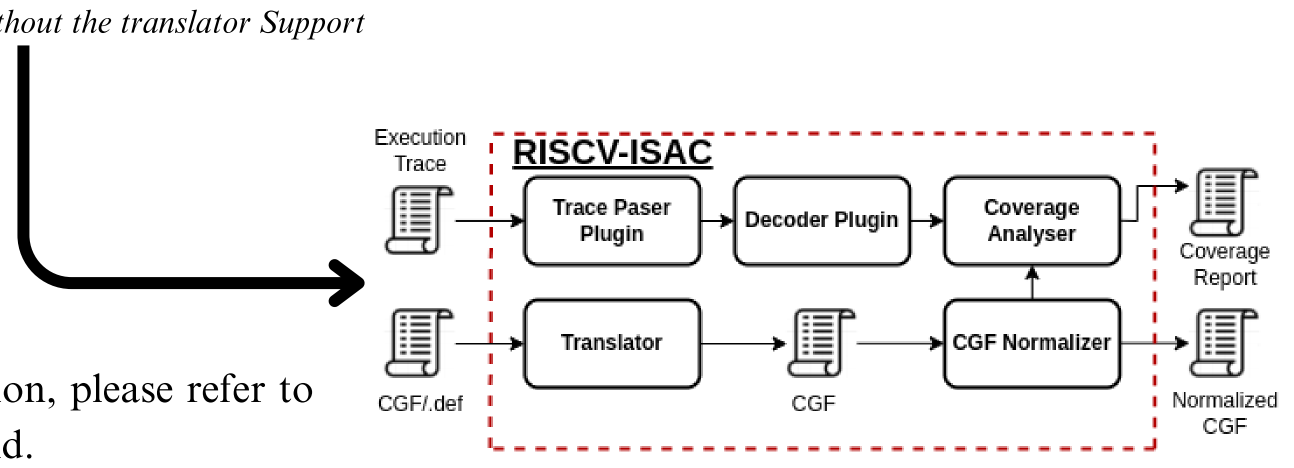
```
pmp_cover:
  config:
    - check ISA:=regex(.*I.*Zicsr.*)
  mnemonics:
    "{lw,sw,${custom_cov},csrrw}": 0
  csr_comb:
    # Braces and placeholder features
    (pmpcfg{{0 ... 7}>>2} & 0x80 == 0x80) and (old(
        "pmpaddr$1")) ^ (pmpaddr$2) == 0x00:0
    # List and more braces features
        # Example for understanding purposes
    (pmpcfg{0 ... 3} & 0x80 == 0x80) and pmpcfg{4
        ... 7}[$1] ^ (pmpaddr$2) == 0x00:0
```

- *Coverpoint format example for the* ***Translator***



- ***RISC-V ISAC*** *Flow without the translator Support*

- *$<number>* refers to the placeholder that points to the previous value being used in the {}/curly braces. {}*[]* is a feature used for a list whose index will be a number written in *[]* and will be pointing to some previous brace.



- *Updated* ***RISC-V ISAC*** *flow*

- For further information, please refer to the QR code at the end.

Results

- The updated features for the privileged architecture were utilized to write coverpoints for both Physical Memory Protection and Virtual Memory
- By using the Translator, we achieved up to 2x reduction in the size of coverpoints for Physical Memory Protection.

References

1. Pawan Kumar S et al. “Automating Generation and Maintenance of a High-Quality Architectural Test Suite for RISC-V”. In: *Proceedings of the Sixth Workshop on Computer Architecture Research with RISC-V, Co-located with ISCA (2022)*. (url:*https:// carrv.github.io/2022/papers/CARRV2022_paper_2_Kumar.pdf*)
2. RISC-V. Sail RISC-V model. (*https://github.com/riscv/sail-riscv*). 2024
3. RISC-V International. Spike, a RISC-V ISA Simulator. (*https://github.com/riscv/riscv-isa-sim*). 2024
4. RISC-V International. RISC-V Architectural Tests. (*https://github.com/riscv/riscv-arch-test*). 2024
5. RISC-V Software Source. RISC-V CTG, Compliance Test Generator. (*https://github.com/riscv- software- src/riscv-ctg*). 2024.
6. RISC-V Software Source. RISC-V ISAC, a coverage analyser. (*https://github.com/riscv- software- src/riscv-isac*). 2024