Verifying Enhanced PMP Behavior in Ibex

Marno van der Maas[1]*, Andreas Kurth[1], Harry Callahan[1], Greg Chadwick[1]

[1]lowRISC CIC, Cambridge, UK

Abstract

To uphold the principle of least privilege, RISC-V can further limit M mode's access to memory through PMP enhancements in the Smepmp *extension. Ibex is an open-source processor, and it uses these enhancements to be a compelling choice for security-critical applications. Because Ibex is thoroughly verified and production quality, the verification requirements are high for adding this extension. In this proposal, we design a coverage plan for Smepmp, generate sensible constrained-random Smepmp configurations and add directed tests to maximize coverage of corner cases. Smepmp poses a particular challenge because many transitions and configurations lead to immediate faults, such as when MML is enabled and the PC is not in an executable region. Through this work, we improve the overall PMP coverage with Smepmp from 63 % to 98 %.*

Introduction

Physical memory protection (PMP) is RISC-V's answer to the need for a memory protection unit [1]. It is a technique to enforce read, write and execute privileges on specified memory regions. The extension, Smepmp, improves the original PMP specification by further limiting memory access of M mode (machine mode) [2]. It does this in three ways:

- Machine mode lockdown (MML): allows regions to have different permissions in M mode and in supervisor/user modes.
- Machine mode whitelist policy (MMWP): prevents M mode from accessing regions outside of the PMP.
- Rule lock bypass (RLB): defines whether M mode is allowed to bypass PMP lock bits or not.

MML in particular is an invasive feature where the four permission bits take on alternate behavior. For example, if the lock (L), execute (X), write (W) and read (R) bits are all set, this means a region is shared read-only. MMWP changes the default behavior of the PMP by disallowing memory accesse that do not match any region, and RLB allows locked PMP regions to be edited.

Ibex is a production-quality embedded RISC-V CPU that supports Smepmp for security applications [3]. The project uses RISCV-DV [4] to generate random instructions for verification which stimulates PMP very well. However, before this work the coverage collected for Smepmp was lacking. To gain confidence that Ibex properly implements Smepmp, we define new coverpoints and add new constraints to the verification environment.

Functional Coverage

To give confidence in the functional coverage of Smepmp, we add coverpoints and crosses for relevant behavior. Coverpoints are a way to collect signals to ensure certain functionality is tested. For example, we can add a coverpoint for the MML bit being enabled in `mseccfg`. Crosses are useful to track whether multiple relevant behaviors are seen at the same time, for example that a data access occurs while MML is enabled.

In this work, we add a global covergroup that looks for behavior across all PMP regions, and a regional covergroup to track behavior for each individual region. The coverpoints and crosses for the global covergroup are shown in Table 1, and those for the regional covergroup are shown in Table 2.

Table 1: *Global covergroup for Smepmp across all PMP regions.*

Name	Description
Code & data	Unmatched requests with MMWP and MML.
`mseccfg`	RLB, MMWP and MML on and off.
Stickiness	Smepmp modes cannot be unset.
Write `mseccfg`	Write to each mode.

Table 2: *Regional Smepmp covergroup for each individual PMP region.*

Name	Description
Region mode	Off, TOR, NA4 and NAPOT.
NAPOT address	All possible region sizes.
Permissions	MML, L, X, W, R.
Code & data	Each entry with each mode and permissions.
Legend:	
TOR	*Top of range*
NA4	*Four-byte aligned*
NAPOT	*Power-of-two aligned*

*Corresponding author: `mvdmaas@lowrisc.org`

Basic Smepmp testing

Using the default RISCV-DV test for PMP, our global and regional coverage are 68.3 % and 62.7 %, respectively. To start testing Smepmp, we add directed static PMP configurations for MMWP, RLB and MML while still randomly generating the test body. These tests improve our coverage to 85.8 % globally and 86.1 % regionally.

Random Smepmp configurations

Most of the remaining coverage holes are because of missing Smepmp configurations. It would be infeasible to write specific tests for all of these; a more optimal approach is to use constrained-random testing to systematically explore the problem space. However, the problem with random configurations of PMP and Smepmp enhancements is that most of them gather no useful coverage by falling into faulting loops. For example, when MML is enabled without having the trap handler and test code set as executable, the processor ends up in an infinite trap loop. It is important to verify this behavior, but we also need tests to make forward progress and collect coverage of other relevant behavior. To do so, we constrain the random PMP test to place the kernel stack in a readable/writable region and the test code and trap handlers in an executable region. The kernel stack is needed for the trap handlers to behave properly.

Another modification is making the trap handler skip loads and stores that fail because of a locked PMP region or because of MMWP being enabled. To skip loads and stores, code regions must be readable so that the trap handler can check whether the faulting instruction is compressed or not and increment the program counter accordingly. This has the benefit that we do not incur the startup cost after a single faulting load or store and can continue to stimulate the current PMP configuration further.

We also add random writes to `mseccfg` to ensure that the bits in this register are sticky. For the verification environment, we add an infinite trap detection mechanism to exit early when the processor enters a trap loop. After making all of these changes our coverage increases to 96.3 % and 93.1 % for global and regional covergroups, respectively.

Directed testing

The constrained-random PMP test gets us very close to full coverage, but returns diminish as runtime increases and there are relevant behaviors that are unlikely to be hit. An example of such behavior is trying to add execution permission to a PMP region when MML is enabled, which is illegal according to the Smepmp specification. Another undertested behavior is trying to execute instructions from regions with each permutation of the MML permissions. This is because such accesses cause unrecoverable faults and cannot be effectively skipped like faulting loads and stores can. Finally, we need to cover an instruction access where the instruction straddles two PMP regions and both cause a fault.

To hit these cases, while keeping the runtime reasonable, we introduce directed Smepmp tests that deliberately put the PMP in an otherwise undertested configuration and do the appropriate access or modification to hit the coverpoints. After adding these tests, our coverage is 97.0 % for the global covergroup and 98.1 % for the regional covergroup.

Conclusion

Smepmp is a valuable extension to the base RISC-V PMP feature and it is paramount that Ibex's implementation is thoroughly verified. This work explains which coverpoints are important for Smepmp and goes through the steps of augmenting the existing verification environment to improve the coverage. Our work improves the global coverage from 68 % to 97 % and the regional coverage from 63 % to 98 %. All our work has been upstreamed to the Ibex [3] and RISCV-DV [4] repositories.

In the future, formal analysis could be performed to gain further confidence in the hardware, similar to approaches taken for regular PMP [5]. Additionally, verification can be done against the official specification of RISC-V in Sail [6]. In general, our work improves the state of verification of Smepmp, which we consider to be an essential RISC-V security extension. This further strengthens Ibex's position as a production-ready RISC-V core for security-sensitive applications.

References

[1] Andrew Waterman, Krste Asanović, and John Hauser. *The RISC-V instruction set manual, volume II: Privileged architecture, document version 20211203*. Tech. rep. RISC-V Foundation, 2021.

[2] Nick Kossifidis et al. *PMP Enhancements for memory access and execution prevention on Machine mode (Smepmp)*. Tech. rep. RISC-V Foundation, 2021.

[3] lowRISC contributors. *Ibex RISC-V Core*. `https://github.com/lowRISC/ibex`. 2023.

[4] Tao Liu, Anil Sharman, and Puneet Goel. *RISCV-DV: Random instruction generator for RISC-V processor verification*. `https://github.com/chipsalliance/riscv-dv`. 2023.

[5] Kevin Cheang et al. "Verifying RISC-V Physical Memory Protection". In: *CoRR* abs/2211.02179 (2022). arXiv: `2211.02179`. URL: `https://doi.org/10.48550/arXiv.2211.02179`.

[6] Prashanth Mundku et al. *RISCV Sail Model*. `https://github.com/riscv/sail-riscv`. 2023.