Open-source SPMP-based CVA6 Virtualization

Manuel Rodríguez*, José Martins, Bruno Sá and Sandro Pinto

Centro ALGORITMI/LASI - Uminho, OSYX Technologies

Abstract

This work presents the first open-source design and implementation of the RISC-V SPMP for hypervisor. We integrated the dual-stage SPMP implementation into an MMU-less 64-bit CVA6 core with the Hypervisor extension enabled. The modified core was functionally validated using a software stack built around the Bao hypervisor. Preliminary results regarding FPGA resource utilization are also provided. Future work includes benchmarking the SPMP-based platform and comparing its performance with conventional MMU-based virtualization. Additionally, we plan to contribute to the RISC-V community by developing a QEMU implementation of the dual-stage SPMP.

Introduction

Virtualization is a key enabling technology in various domains, from cloud and high-performance computing to mobile, embedded, and Internet of Things (IoT) systems. In order to aid hypervisors and ease virtualization, RISC-V introduced the Hypervisor extension into its privileged architecture. However, even with these hardware extensions, virtualization adds performance overhead and nondeterminism, primarily due to operations inherent to implicit memory accesses in two-stage address translation. To address these challenges, modern MMU-less architectures (e.g., Arm Cortex-R52 and Infineon TC4x) have adopted MPU-based virtualization. RISC-V has followed suit, extending its Physical Memory Protection (PMP) mechanism with supervisor-mode PMP (SPMP) for task isolation in RTOS environments and later introducing a dual-stage SPMP for virtualization, compliant with the Hypervisor extension. In this work, we present the first open-source, virtualization-capable, SPMP-based RISC-V platform. Our design builds on a 64-bit CVA6 core with the Hypervisor extension, replacing the MMU in the Load-Store Unit (LSU) with our implementation of the dual-stage SPMP to enable MPU-based virtualization. The modified design was functionally validated using a virtualization-based software environment built around the Bao hypervisor [1].

RISC-V SPMP for Hypervisor

In RISC-V, the PMP mechanism was initially available only in machine mode to enforce isolation for firmware and OS resources. To support Real-time Operating Systems (RTOS) running in supervisor mode with hardware-enforced isolation between tasks, the RISC-V community introduced the supervisor-mode PMP (SPMP). The SPMP follows a design similar to the original PMP, featuring up to 64 entries that define region boundaries, permissions, and a mode bit to differentiate between supervisor and user access.

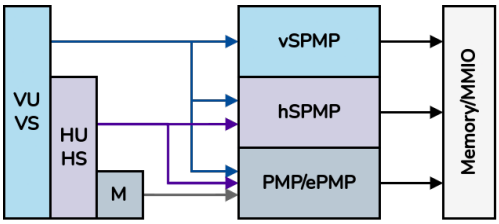

Figure 1: *Unified model of the dual-stage SPMP.*

The SPMP is being extended to support virtualization by introducing a dual-stage SPMP. In this design, the first stage, referred to as *virtual SPMP (vSPMP)*, is controlled by guest OSes running in VS-mode, while the second stage is managed by the hypervisor (HS-mode) to enforce isolation between Virtual Machines (VMs). Although the current proposal [2] defines two separate SPMPs controlled by the hypervisor (baseline SPMP and hgPMP) and one controlled by the VMs (vSPMP), we follow the alternative model described in [3], which encompasses a single *unified hSPMP* controlled by HS-mode and the vSPMP, controlled by VS-mode. In this model, VM memory accesses are interpreted as user-mode accesses by the unified SPMP, while hypervisor accesses are checked against supervisor permissions. This approach avoids the potential waste of unused entries when virtualization is not used.

SPMP-based CVA6

Dual-stage SPMP. We implemented the proposed unified model of the dual-stage SPMP for hypervisor in a CVA6 core. The SPMP module includes address matching and permission enforcement logic replicated according to the number of SPMP entries defined for the hart. The module follows a hybrid design, enabling it to function as either the vSPMP or hSPMP based on a configurable design parameter. For both cases, all accesses are internally classified as *Smode* or *Umode*, but the effective privilege is determined based on the access privilege and the virtualization (V) bit.

*Corresponding author: id11674@alunos.uminho.pt

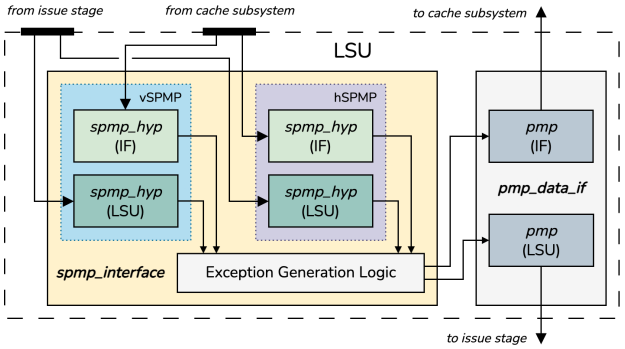

Figure 2: *SPMP integration into the CVA6 LSU.*

Integration into CVA6. To integrate the SPMP into the CVA6 core, we started with the *cv64a6_imafdch_sv39* configuration, disabling the MMU by clearing the *CVA6ConfigMmuPresent* parameter while keeping the Hypervisor extension enabled (*CVA6ConfigHExtEn*). We introduced two design parameters to control the SPMP: *CVA6ConfigSpmpPresent* for enabling it and *CVA6ConfigNrSPMPEntries* to define the number of entries. Additionally, we incorporated the hSPMP and vSPMP CSRs into the register file and propagated their values across the design to the LSU. As shown in Figure 2, we instantiate the *spmp_interface* module within the LSU to house all SPMP logic for memory accesses generated by the core. The design includes dedicated dual-stage SPMPs for instruction fetch (IF) and load/store (LS) requests. To enable parallel vSPMP and hSPMP checks, the SPMP module is instantiated twice to represent each dual-stage SPMP. Each request is processed with a fixed latency of one clock cycle, after which the SPMP signals completion and outputs exception data, if an exception occurs.

Preliminary Evaluation

Functional Validation. We adapted the virtualization software environment from [3] to validate the SPMP-based CVA6. The modifications primarily involved adding support for the CVA6 platform in both the Bao hypervisor and the guest image, as well as configuring Bao for a single-core VM setup. Validation experiments were conducted on FPGA technology using a *Genesys2* development board.

Hardware Utilization After functional validation, we analyzed FPGA resource utilization across four CVA6 LSU configurations: (1) with neither MMU nor SPMP (*Bare CVA6*), (2) with a dual-stage SPMP featuring 32 entries, (3) with a dual-stage SPMP featuring 64 entries, and (4) with MMU but no SPMP (*MMU-based CVA6*). All configurations include the RISC-V Hypervisor extension. Table 1 shows the Lookup Table (LUT) and Flip-Flop (FF) utilization for each configuration, with configurations (2), (3), and (4) showing

Configuration	Resource	Utilization
(1) Bare CVA6	LUT	3633/203800
	FF	1951/407600
(2) SPMP-based CVA6 w/ 32 entries	LUT	8671 (+139%)
	FF	2085 (+6.9%)
(3) SPMP-based CVA6 w/ 64 entries	LUT	12324 (+239%)
	FF	2085 (+6.9%)
(4) MMU-based CVA6	LUT	11370 (+213%)
	FF	7646 (+292%)

Table 1: *Hardware resources used by the CVA6 LSU. The percentage values represent the increase of the LSU resource utilization relative to the Bare CVA6 (1).*

the percentage increase in LSU resource utilization relative to the Bare CVA6 configuration (1). Our analysis revealed that the SPMP-based CVA6 uses more LUTs than the MMU-based configuration when featuring the maximum number of entries defined by the standard, but the increase in FF utilization is negligible. This is expected as the address matching and permission checking logic within the SPMP is entirely done using combinatorial logic. A few registers are used within the *spmp_interface* module to avoid timing issues.

Roadmap

As for our next steps, we plan to benchmark the SPMP-based implementation, comparing the results with the conventional MMU-based virtualization approach. We also intend to make the SPMP integration more flexible in order to allow hybrid configurations between the dual-stage SPMP and the MMU. In a more advanced phase, we plan to aid in the standardization of the dual-stage SPMP by implementing support in QEMU.

Conclusion

In this paper, we described the design and implementation of the RISC-V SPMP for hypervisor in the CVA6 core. We followed a novel approach for the dual-stage SPMP design, based on an alternative unified model. We intend to open-source our implementation to promote collaboration with the RISC-V community.

Acknowledgments

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the R&D Unit Project of ALGORITMI Centre.

References

[1] J. Martins et al. “Bao: A Lightweight Static Partitioning Hypervisor for Modern Multi-Core Embedded Systems”. In: *Workshop on NG-RES*. 2020.

[2] Dong Du and Sandro Pinto. *RISC-V S-mode Physical Memory Protection for Hypervisor*. 2023.

[3] Sandro Pinto and Matjaz Breskvar. “A Novel Trusted Execution Environment for Next-Generation RISC-V MCUs”. In: *EmbeddedWorld*. 2024.