Proposal of sPMP: S-mode PMP for RISC-V

Dong Du, Xu Lu, Wenhao Li, Yubin Xia, Shanghai Jiao Tong University

1. Motivation

We propose sPMP (S-mode Physical Memory Protection) for the following two reasons: better isolation and scalability.

First, RISC-V based processors recently stimulate great interest in the emerging internet of things (IoT). However, as the page-based virtual memory (MMU) is usually not available on IoT devices, it is hard to isolate the S-mode OSes (e.g., RTOS) and user-mode applications. To support secure processing and isolate faults of U-mode software, it is desirable to enable S-mode OS to limit the physical addresses accessible by U-mode software on a hart.

Second, several existing TEE/enclave solutions of RISC-V (e.g., HexFive) are based on PMP. However, since the number of PMP registers is limited, current TEE/enclave systems usually rely on software based PMP virtualization/scheduling mechanisms to support large number of enclaves, which will lead to larger TCB (Trusted Computing Base) in the machine mode, as shown in Figure 1-a.

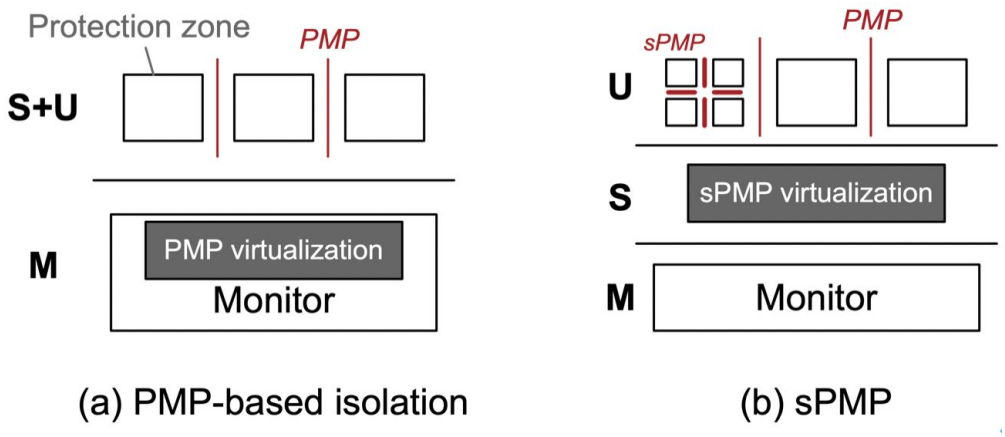

Figure 1: Comparison of PMP and sPMP. sPMP reduces the TCB in the machine mode and unlocks more optimization opportunities.

2. S-mode Physical Memory Protection (sPMP)

An optional S-mode Physical Memory Protection (sPMP) unit provides per-hart supervisor-mode control registers to allow physical memory access privileges (read, write, execute) to be

specified for each physical memory region. The sPMP values are checked after the physical address to be accessed pass both the PMA checks and PMP checks described in the privileged spec.

Like PMP, the granularity of sPMP access control settings are platform-specific and within a platform may vary by physical memory region, but the standard sPMP encoding should support regions as small as four bytes. Certain regions' privileges can be hardwired——for example, memory regions where code and data of S-mode OS reside on can only ever be visible in supervisor and machine mode but in no lower-privilege layers.

sPMP checks are applied to all accesses when the hart is running in U modes, and for loads and stores when the MPRV bit is set in the mstatus register and the MPP field in the mstatus register contains U. Optionally, sPMP checks can also apply to S-mode accesses, in which case the sPMP values are locked to S-mode software, so that S-mode cannot change their values. Unlike PMP registers, sPMP registers can always be modified by M-mode software even when they are locked. sPMP registers can grant permissions to U-mode, which has none by default, and revoke permissions from S-mode, which has full permissions by default.

2.1. Supervisor-mode Physical Memory Protection Keys

Like PMP, sPMP entries are described by an 8-bit configuration register and one XLEN-bit address register. Some sPMP settings additionally use the address register associated with the preceding sPMP entry. The number of sPMP entries can vary by implementation, and up to 16 sPMP entries are supported in standard.

The sPMP configuration registers are packed into CSRs in the same way as PMP does. For RV32, four CSRs, spmpcfg0-spmpcfg3, hold the configurations $spmp_0cfg$-$spmp_{15}cfg$ for the 16 sPMP entries, as shown in Figure 2. For RV64, spmpcfg0 and spmpcfg2 hold the configurations for the 16 sPMP entries, as shown in Figure 3; spmpcfg1 and spmpcfg3 are illegal.

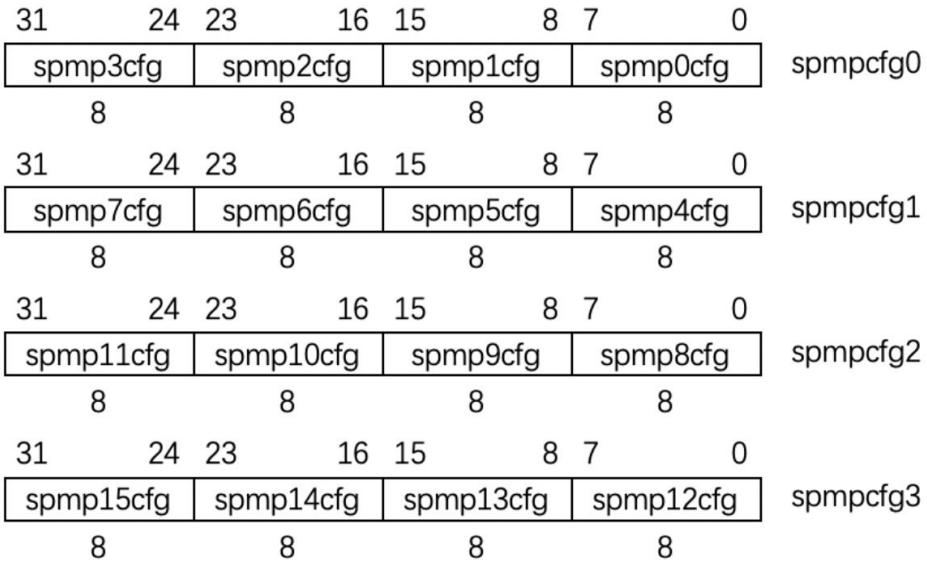

Figure 2: RV32 sPMP configuration CSR layout

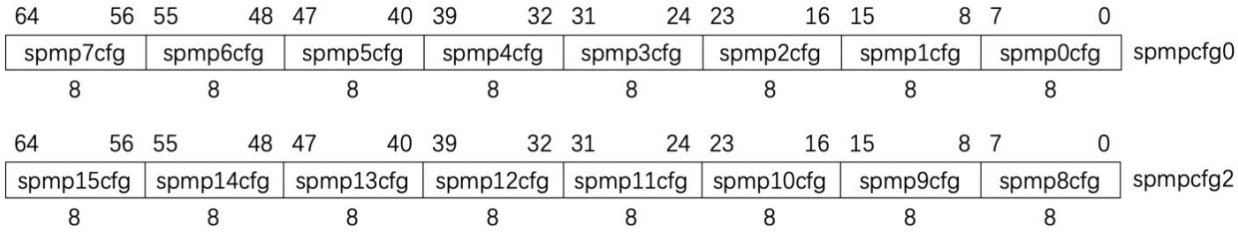

Figure 3: RV64 sPMP configuration CSR layout

The sPMP address registers are CSRs named $spmpaddr_0$-$spmpaddr_{15}$. Each sPMP address register encodes bits 33-2 of 34-bit physical address for RV32, as shown in Figure 4. For RV64, each sPMP address encodes bits 55–2 of a 56-bit physical address, as shown in Figure 5. Not all physical address bits may be implemented , and so the spmpaddr registers are WARL.

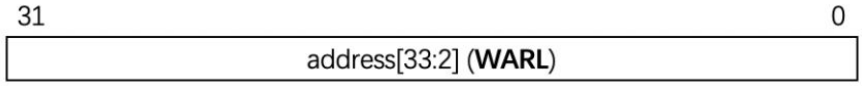

Figure 4: sPMP address register format, RV32

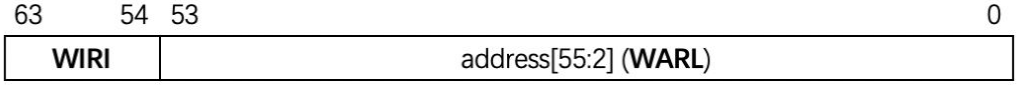

Figure 5: sPMP address register format, RV64

The layout of sPMP configuration registers is the same as PMP configuration registers, as is shown in Figure 6. The R, W, and X bits, when set, indicate that the sPMP entry permits read, write and instruction execution, respectively. When one of these bits is clear, the corresponding

access type is denied. The U bit represent that the sPMP entry is for user mode, and will be used to enforce SMAP and SMEP (described in Section 2.5). The remaining two fields, A and L will be described in the following sections.

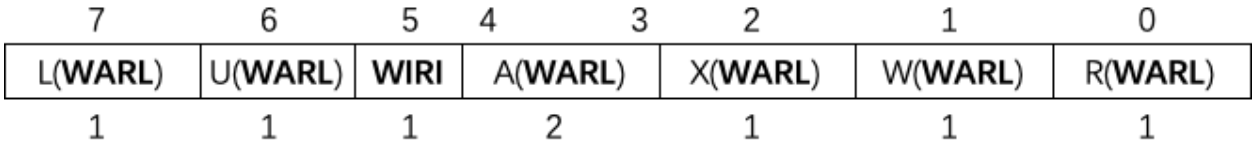

Figure 6: sPMP configuration register format

The number of sPMP entries: The proposal advocates 16 sPMP entries, which can provide 16 isolated regions concurrently. To provide more isolation regions, the software in S-mode (usually an OS) can virtualize more isolated regions and schedule them by switching the values in sPMP entries. Moreover, we can combine sPMP with PMP to provide more isolated regions, e.g., with 16 PMP entries, there are 256 (16x16) isolated regions concurrently.

2.2. Address Matching

Like PMP's design, the A field in an sPMP entry's configuration register encodes the address-matching mode of the associated sPMP address register. The encoding of A field is the same as PMP's, as shown in Table 1. When A=0, this sPMP entry is disabled and matches no address. Two other address-matching modes are supported: naturally aligned power-of-2 regions (NAPOT), including the special case of naturally aligned four-byte regions (NA4); and the top boundary of an arbitrary range (TOR). These modes support four-byte granularity.

A	Name	Description
0	OFF	Null Region (turned off)
1	TOR	Top of Range
2	NA4	Naturally aligned 4-byte region
3	NAPOT	Naturally aligned power-of-two region, >= 8 bytes

Table 1: Encoding of A field in sPMP configuration registers

NAPOT ranges make use of the low-order bits of the associated address register to encode the size of the range, as shown in Table 2.

spmpaddr	spmpcfg.A	Match type and size
aaaa···aaaa	NA4	4-byte NAPOT range
aaaa···aaa0	NAPOT	8-byte NAPOT range
aaaa···aa01	NAPOT	16-byte NAPOT range
aaaa···a011	NAPOT	32-byte NAPOT range
...	...	...
aa01···1111	NAPOT	2^{XLEN}-byte NAPOT range
a011···1111	NAPOT	2^{XLEN+1}-byte NAPOT range
0111···1111	NAPOT	2^{XLEN+2}-byte NAPOT range

Table 2: NAPOT range encoding in sPMP address and configuration registers

If TOR is selected, the associated address register forms the top of the address range, and the preceding sPMP register forms the bottom of the address range. If sPMP entry i's A field is set to TOR, the entry matches any address such that $spmpaddr_{i-1}$ <= a < $spmpaddr_i$. If sPMP entry 0's A field is set to TOR, zero is used for the lower bound, and so it matches any address a < $spmpaddr_0$.

2.3. Locking and Privilege Mode

The L bit indicates that the sPMP is locked to S-mode, i.e., S-mode writes to the configuration register and associated address registers are ignored. Locked sPMP entries can only be unlocked by M-mode or by a system reset. If sPMP entry i is locked, writes to the $spmp_icfg$ and $pmpaddr_i$ are ignored. Additionally, if $spmp_icfg$.A is set to TOR, writes to $pmpaddr_{i-1}$ are ignored.

In addition to locking the sPMP entry, the L bit indicates whether the R/W/X permissions are enforced on S-mode accesses. When the L bit is set, these permissions are enforced for both user and supervisor modes (M-mode accesses are not affected). When the L bit is clear, any S-mode access matching the sPMP entry will succeed; the R/W/X permissions apply only to U modes.

2.4. Priority and Matching Logic

The sPMP checks only take effect after the memory access passes the PMP permission checks. An M-mode access will not be checked by sPMP property.

Like PMP entries, sPMP entries are also statically prioritized. The lowest-numbered sPMP entry that matches any byte of an access determines whether that access succeeds or fails. The matching sPMP entry must match all bytes of an access, or the access fails, irrespective of the L, R, W, and X bits.

If an sPMP entry matches all bytes of an access, then the L, R, W and X bits determine whether the access succeeds or fails. If the privilege mode of the access is M, the access succeeds. Otherwise, if the L bit is set or the privilege mode of the access is U, then the access succeeds only if the R, W, or X bit corresponding to the access type is set.

If no sPMP entry matches an S-mode access (i.e., there is no such an sPMP entry whose L bit is set and region contains the memory access), the access succeeds, otherwise the access is checked according to the permission bits in sPMP entry. If no sPMP entry matches an U-mode access, but at least one sPMP entry is implemented, the access fails.

Failed accesses generate a page fault exception. Note that a single instruction may generate multiple accesses, which may not be mutually atomic.

Interaction with paging (satp): The sPMP can be used together with paging. The sPMP checks are performed using physical address translated by the page table when virtual addressing is enabled. The only way to disable sPMP in S-mode is setting 0 (i.e., matching off) in the A field in sPMP configuration registers. We do not rely on satp to control sPMP as there are no remaining bits in RV32's satp.

2.5. SMAP and SMEP with sPMP

We follow strategies of SMAP and SMEP on current S-mode.

For SMAP, we leverage the SUM (permit Supervisor User Memory access) bit in the status register to indicate the privilege with which S-mode loads, stores, and instruction fetches access physical memory.

- When SUM=0, S-mode physical memory accesses to memory that are accessible by U-mode (U=1 in Figure 6) will fault.
- When SUM=1, these accesses are permitted.

The SUM can take effect even when page-based virtual memory is not in effect.

For SMEP, we do not allow the S-mode to execute codes in physical memory that are for U-mode (U=1 in Figure 6). This is compatible with the existing design that the supervisor may not execute code on pages with U=1. Violations will trigger page faults.

2.6. Delegation

Unlike PMP which uses access faults for violations, sPMP uses page fault for violations. In such case, the violation code is treated as "S-mode memory protection faults" rather than "page faults." The benefit of using page fault is that we can delegate the violations caused by sPMP to S-mode, while the access violations caused by PMP can still be handled by machine mode.

2.7. Interaction with Hypervisor mode

We have seen three kinds of combinations that sPMP can be used with the H extension.

H-mode paging	S-mode paging	Description
✓	x	The S-mode can use sPMP to provides isolation (based on GPA). And the H-mode uses paging to isolate guests by translating their GPA to different HPA.
✓	✓	The software in S-mode can use sPMP, or paging or even both. Using both can further enhance the isolation, e.g., isolate a physical region which should never be mapped to U-mode.
x	x	Relying on PMP for isolation.
x	✓	Rare cases

Table 3: Combination of sPMP and H-mode

Case-1: H-mode supports paging, but S-mode doesn't. In this case, the S-mode can use sPMP to provide isolation, which is based on GPA but not HPA. And the H-mode uses paging to isolate guests by translating their GPA to different HPA. For access in the U-mode in such case, it should first pass sPMP permission check, then H-mode's PT, and then PMP.

Case-2: Both H-mode and S-mode support paging. In such case, the software in S-mode (usually an OS) can use sPMP, or paging, or even both. Using sPMP only in S-mode is the same as the first case. Using both sPMP and MMU in S-mode can further enhance the isolation, e.g., isolate a physical region which should never be mapped to user mode when paging is enabled. For access in the U-mode in such case, it should first pass guest PT check, then sPMP permission check, then h-mode's PT, and then PMP.

Case-3: Both H-mode and S-mode do not support paging. In such case, we may need an h-mode physical memory protection mechanism or relies on the machine mode to use PMP to provide isolation between H-mode and S/U modes.

The case that “H-mode do not support paging but S-mode can support” is very rare.