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

Dong Du, Xu Lu, Bicheng Yang, Wenhao Li, Yubin Xia, Shanghai Jiao Tong University and RISC-V TEE Task Group

Contributors: Nick Kossifidis, Joe Xie, Paul Ku, Bill Huffman, Jonathan Behrens, Allen Baum, Robin Zheng, Zeyu Mi, Members of the TEE Task Groups

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.

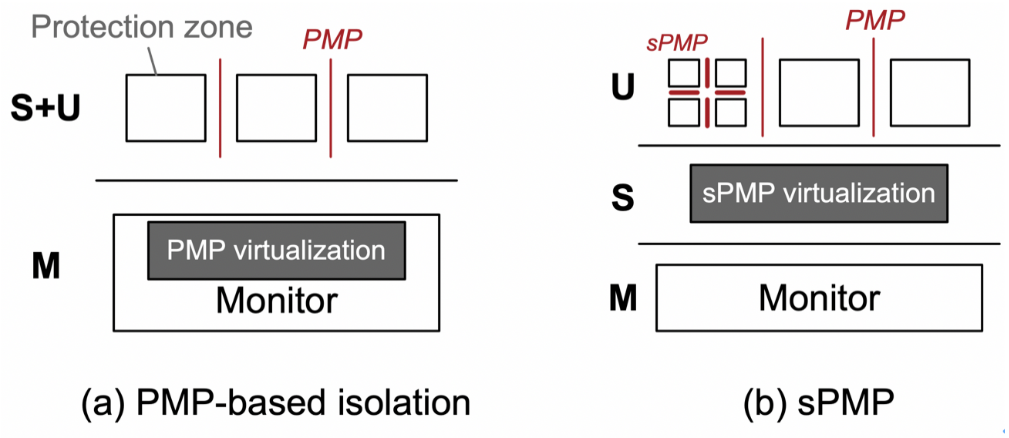


Figure 1: Comparison of PMP and sPMP. sPMP reduces the TCB in the machine mode and unlocks more optimization opportunities.

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.

sPMP checks could be applied to all accesses for both U mode and S mode, depending on the values in the configuration registers. sPMP registers can always be modified by M-mode and S-mode software. 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 $spmp_0$cfg-$spmp_{15}$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.

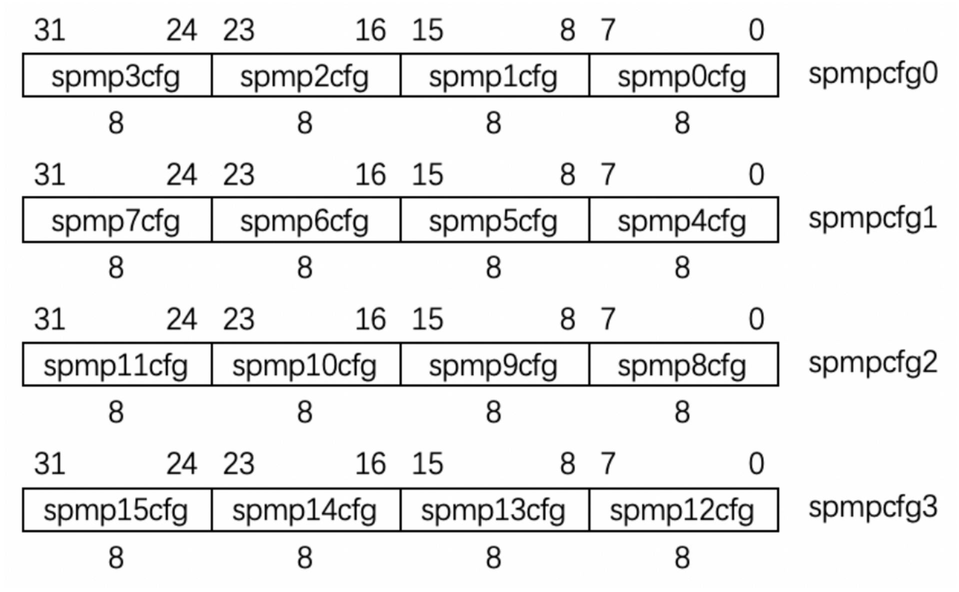

31 24 23 16 15 8 7 0
spmp3cfg spmp2cfg spmp1cfg spmp0cfg spmpcfg0
8 8 8 8
spmp7cfg spmp6cfg spmp5cfg spmp4cfg spmpcfg1
spmp11cfg spmp10cfg spmp9cfg spmp8cfg spmpcfg2
spmp15cfg spmp14cfg spmp13cfg spmp12cfg spmpcfg3

Figure 2: RV32 sPMP configuration CSR layout

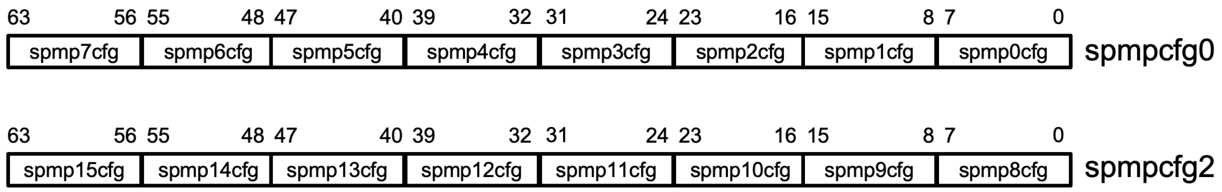

63 56 55 48 47 40 39 32 31 24 23 16 15 8 7 0
spmp7cfg spmp6cfg spmp5cfg spmp4cfg spmp3cfg spmp2cfg spmp1cfg spmp0cfg spmpcfg0
spmp15cfg spmp14cfg spmp13cfg spmp12cfg spmp11cfg spmp10cfg spmp9cfg spmp8cfg spmpcfg2

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.

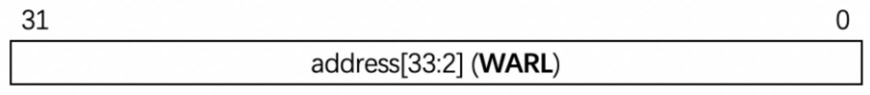


Figure 4: sPMP address register format, RV32

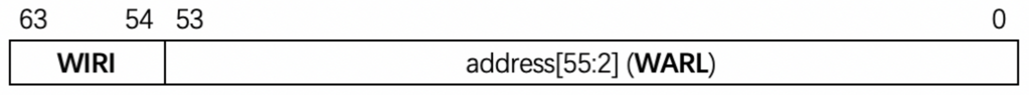


Figure 5: sPMP address register format, RV64

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 S bit represents whether an sPMP entry is for S-mode. When an sPMP entry is for U-mode （S = 0), the permission will be enforced in U-mode and restrict the access by S-mode (for SMAP and SMEP); otherwise（S = 1), the sPMP entry is for S-mode and the permission checking is enforced in S-mode. sPMP also introduces shared data and code regions. The encoding for permission is shown in section 2.3, which is the same as ePMP (except sPMP does not need a lockdown bit).

The remaining field, A bit, will be described in the following sections (2.2).

7	6 5	4 3	2	1	0
S **(WARL)**	**WIRI**	A **(WARL)**	X **(WARL)**	W **(WARL)**	R **(WARL)**
1	2	2	1	1	1

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.

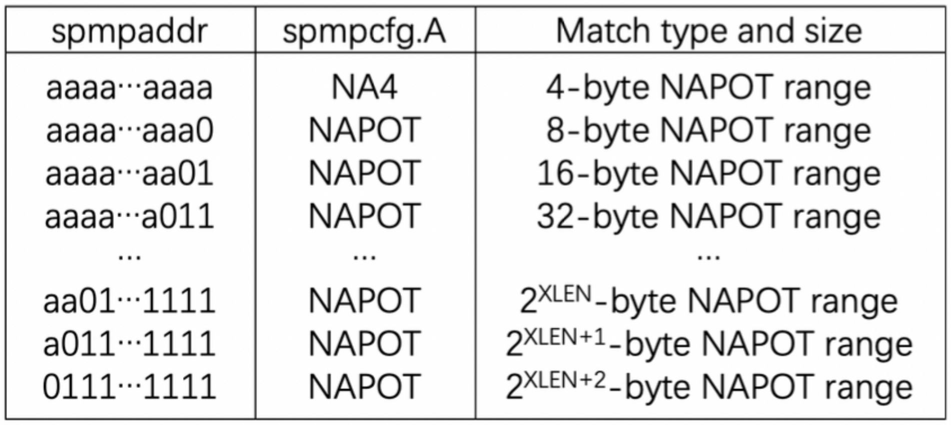
2.2. Address Matching

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. Encoding of Permissions

sPMP has three kinds of regions: U-mode region, S-mode region, and shared regions. Normally, when the S bit is set, the region could be either a S-mode region or a shared region. When the S bit is clear, the region could be either a U-mode region or a shared region.

S-mode regions indicate that the R/W/X permissions are enforced on S-mode accesses. M-mode accesses are not affected and U-mode access will trigger faults.
For U-mode regions, any S-mode access matching the sPMP entry will trigger faults; the R/W/X permissions apply only to U modes.
For shared regions, the permissions are enforced for both U and S modes, according to the table below.

S-mode can always modify the value in S bit.

The encoding and results are shown in the table (SUM=0):

Bits on *spmpcfg* register				Result	
S	R	W	X	S Mode	U Mode
0	0	0	0	Inaccessible region (Access Exception)	

0	0	0	1	Access Exception	Execute-only region
0	0	1	0	Shared data region: Read/write on S mode, read-only on U mode	
0	0	1	1	Shared data region: Read/write for both S and U modes	
0	1	0	0	Access Exception	Read-only region
0	1	0	1	Access Exception	Read/Execute region
0	1	1	0	Access Exception	Read/Write region
0	1	1	1	Access Exception	Read/Write/Execute region
1	0	0	0	Shared region: read/write/execute for both S and U modes	
1	0	0	1	Execute-only region	Access Exception
1	0	1	0	Shared code region: Execute only on both S and U mode.	
1	0	1	1	Shared code region: Execute only on U mode, read/execute on S mode.	
1	1	0	0	Read-only region	Access Exception
1	1	0	1	Read/Execute region	Access Exception
1	1	1	0	Read/Write region	Access Exception
1	1	1	1	Shared data region: Read only on both S and U mode.	

SUM bit: We re-use the SUM (permit Supervisor User Memory access) bit in sstatus to modify the privilege with which S-mode loads and stores access physical memory. When SUM=0, S-mode memory accesses to U-mode regions will trigger fault, as shown in the table. When SUM=1, these accesses are permitted. The semantics of SUM in sPMP is consistent with it in paging.
SUM only affects S-mode behaviors on U-mode regions. It does not affect S-mode regions and shared regions, as shown in the following tables.

Note: sPMP does not permit supervisor mode to execute instructions from U-mode regions, regardless of the SUM setting.

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 S, R, W, and X bits.

If an sPMP entry matches all bytes of an access, then the S, 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, the access succeeds only if it satisfies the permission checking with the S, R, W, or X bit corresponding to the access type.

If no sPMP entry matches an S-mode 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.

2.5. Exceptions

Failed accesses generate an exception. sPMP follows the strategy that uses different exception codes for different cases, i.e., load/store/instruction sPMP faults for memory load, memory store and instruction fetch respectively. The sPMP will add three new exception codes for clarity.
Note that a single instruction may generate multiple accesses, which may not be mutually atomic.

Table of new exception codes:

Interrupt	Exception Code	Description
0	16	Instruction sPMP fault
0	17	Load sPMP fault
0	18	Store/AMO sPMP fault

Note: You can refer to the Table 3.6 in riscv-privileged spec.

Delegation: Unlike PMP which uses access faults for violations, sPMP uses new sPMP faults for violations. The benefit of using new exception codes 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.

2.6. Enabling and Disabling sPMP

sPMP is enabled by default when it exists. This is consistent with PMP.

When a device has both sPMP and MMU (or paging), and the S-mode software wants to disable sPMP to only use paging, it can configure one sPMP entry covering all addresses to grant all permissions to both S-mode and U-mode.

2.7. sPMP and Paging

Like PMP, the sPMP mechanism is designed to compose with the page-based virtual memory systems. When paging is enabled, instructions that access virtual memory may result in multiple physical-memory accesses, including implicit references to the page tables. The sPMP checks apply to all of these accesses. The effective privilege mode for implicit page-table accesses is S.

Implementations with virtual memory are permitted to perform address translations speculatively and earlier than required by an explicit virtual-memory access. The sPMP settings for the resulting physical address may be checked at any point between the address translation and the explicit virtual-memory access. Hence, when the sPMP settings are modified in a manner that affects either the physical memory that holds the page tables or the physical memory to which the page tables point, S-mode software must synchronize the sPMP settings with the virtual memory system. This is accomplished by executing an SFENCE.VMA instruction with rs1=x0 and rs2=x0, after the sPMP CSRs are written.

If page-based virtual memory is not implemented, or when it is disabled, memory accesses check the sPMP settings synchronously, so no fence is needed.

3. Summary of Hardware Changes

CSR for sPMP address registers:	16 new CSR registers
CSR for sPMP configuration registers:	4 new CSR registers for RV32 and 2 for RV64
New exception codes:	Instruction sPMP fault
	Load sPMP fault
	Store/AMO sPMP fault

4. Interaction with other proposals

This section discusses how sPMP interacts with other (ongoing) proposals.

RISC-V PMP enhancements: sPMP is compatible with ePMP proposal, and uses almost the same encoding as ePMP.

J-extension pointer masking proposal: When both PM and sPMP are used, sPMP checking should be performed using the actual addresses generated by PM (pointer masking).

Hypervisor extension: To support both sPMP and hypervisor extension, there are some further changes.

- First, a set of sPMP CSRs for the VS-mode are required, including 16 vspmp address registers and 4 configuration registers. When V=1, vspmp CSR substitutes for the usual spmp CSR, so instructions that normally read or modify spmp CSR actually access vspmp CSR instead. This is consistent with the paging in VS-mode (i.e., vsatp).
- Second, for HLV, HLVX, and HSV instructions, the hardware should perform vsPMP checking between VS-level address translation and G-stage address translation.
- Third, the result of G-stage translation should be checked through HS-level SPMP.
- Last, the vsPMP checking is performed in the guest physical addresses, between VS-level address translation and G-stage address translation.