RISC-V Supervisor Domains Access Protection (SmMTT)

RISC-V SmMTT Task Group

Version 1.0, 7/2023: This document is in development. Assume everything can change. See http://riscv.org/spec-state for details.

Table of Contents

Preamble

This document is in the Development state

Assume everything can change. This draft specification will change before being accepted as standard, so implementations made to this draft specification will likely not conform to the future standard.

Copyright and license information

Contributors

This RISC-V specification has been contributed to directly or indirectly by (in alphabetical order): Andy Dellow, Dean Liberty, Deepak Gupta, Guerney Hunt, Krste Asanovic, Mark Hill, Nick Wood, Osman Koyuncu, Paul Elliott, Ravi Sahita (Editor), Samuel Ortiz, Ved Shanbhogue, Wojchiech Ozga

Chapter 1. Supervisor Domain Access Protection (SmMTT)

v0.51

The following is a proposal being submitted for the purposes of discussion and collaboration to extend the RISC-V privileged ISA. This proposal has inputs from AP-TEE TG, Security HC discussions and the RVI Security F2F on April 25, 2023. At this time, this is an initial proposal and is not an official draft of a RISC-V specification.

1.1. Introduction and Motivation

This document describes Supervisor Domain Access Protection - RISC-V privileged architecture extension to support physical address space (memory and devices) isolation for more than one supervisor domain use cases for RISC-V platforms. Use cases may use supervisor domains to reduce the supervisor Trusted Computing Base (TCB), with differential access to memory and other platform resources e.g. Confidential Computing (CoVE), Security Services, Secure Devices etc.

Tenant (application or VM) workloads on multi-tenant platforms rely on hardware-based isolation primitives that are managed by the host/privileged software. Traditional host software (operating systems and virtual machine monitors) have unfettered access to system memory, devices and hardware isolation mechanisms such as the memory management unit (MMU) or physical memory protection (PMP) registers. Typical multi-tenant platforms also have a very large Trusted Computing Base (TCB) and have many threat actors that influence the TCB. For example, in a multi-tenant data-center environment, the host software, devices and device drivers, cloud operators, orchestration services, other tenant workloads etc. are all considered threat vectors. Additional threats originate due to deployment models, for example, an embedded platform deployed in the field is exposed to physical threat vectors. Hence, in many scenarios isolated supervisor domains are desired to be able to express differentiated trust models and secure access to platform resources. A supervisor domain has resources (such as memory/IO regions, processing elements, devices and interrupts) to perform their function.

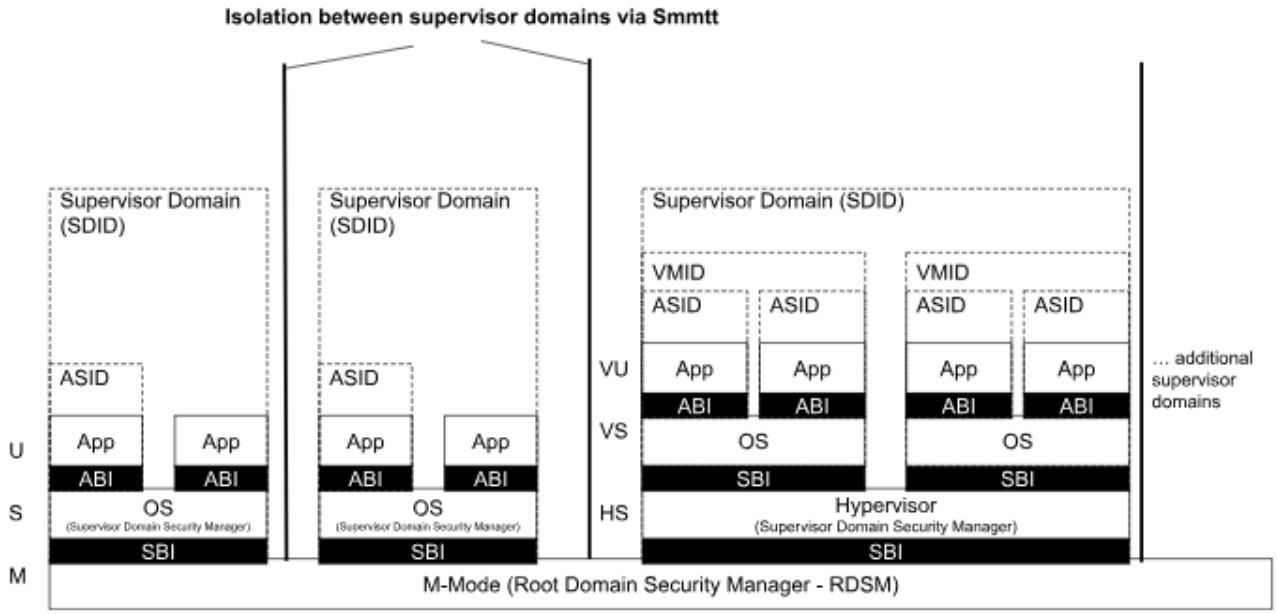

Figure 1: Supervisor Domains

A supervisor domain is associated with a set of physical address regions that are isolated from other supervisor domains on the same platform, with only the Root Domain Security Manager (RDSM) with access to all of the physical address space. A supervisor domain identifier (SDID) is associated with the supervisor domain to facilitate physical address protection fences on a per supervisor domain basis. Supervisor domains must rely on a TCB which consists of the RDSM (software) and hardware (hart, SoC, Root-of-trust) that enforces the isolation properties for the supervisor domain. Isolation of the workloads within a supervisor domain is the responsibility of the OS/hypervisor managing the supervisor domain, here referred to as the Supervisor Domain Security Manager (SDSM).

A key goal of using multiple domains is to be able to reduce the common TCB across domains, and should enable the attestation [2] of each domain independently from other domains. Sensitive data may be entrusted to a particular domain after verifying the trust properties statically (via boot) or dynamically (via attestation). These trust properties are established as part of the hardware and software supply chain, system configuration and may be additionally evaluated using attestation mechanisms. The security certification of the RDSM is desirable but out of scope of this specification.

Use cases for supervisor domain isolation range from embedded to application/server-class platforms. Some examples where supervisor domains can be used are:

- A trusted execution environment domain that isolates security services/applications.
- A confidential computing domain which enforces confidentiality and integrity for workload data-in-use from the host/untrusted hypervisor, along with attestation of the TCB.
- A host (operator) domain that manages resources on the platform, and may assign resources to other domains.
- A service-provider domain that has exclusive access to some devices.

In order to avoid re-factoring of deployed host software, workloads and applications, new hardware primitives are required to support memory isolation for domains. A second key

requirement the new hardware primitives must address is the performance and scalability of physical memory isolation at a page-level to support rich-OS memory management models. This specification describes the architecture primitives to support the requirements of a multi-supervisor domain physical address isolation model via a Supervisor Domain Access Protection (Smmtt) extension for RISC-V processor-based platforms.

Chapter 2. Semantics and Security Properties

2.1. Supervisor Domain Access Protection extension (Smmtt)

The Supervisor Domain Access Protection extension (Smmtt) specifies if a physically addressed memory or device mapped region is accessible (read, write) by a particular supervisor domain (or not). Associating a memory region with a supervisor domain via Smmtt implies that the physical addressable region is accessible only from software or hardware access occurring in the context of the owner supervisor domain. Hence, software or hardware accesses that originate from other supervisor domains outside the owner supervisor domain are prevented by hardware. The RDSM has access to all supervisor domains.

In order to enforce these properties, the following architecture interfaces are required:

- An interface to signal the active supervisor domain under which a hart is operating. This is a dynamic control state on the hart that can be held in an M-mode CSR and modifiable by the RDSM via the CSR r/w instructions - herewith called the supervisor domain identifier assigned to the hart. Domains are orthogonal to privilege levels within the supervisor domain and since Smmtt enables physical memory isolation, there is one CSR (per hart) managed by the M-mode. Similarly for devices, a supervisor domain identifier may be assigned to the device to signify assignment - A cross-reference to the IO supervisor domain will be added in the future; Isolation of data within a device is out of scope of this specification.
- An interface to set the access permissions for a memory region or page associated with a supervisor domain. This interface allows dynamic changes of association (which may require appropriate flushing of any state cached in harts). The association mapping is programmed via an Memory Tracking Table (MTT) structure, accessed via per-hart M-mode CSRs and which may be backed by additional in-memory structures. The M-mode CSR interface is expected to program the root physical page (MTTPPN) - for when the MTT is a memory-based structure, the MTTPPN would hold the physical address of the root page of the MTT structure in memory - the MTT is expected to be memory resident at time of access. Write access to MTT structures must be restricted by and to the RDSM (except for when explicitly allowed by the RDSM). Privilege levels may affect changes in the MTT under purview of the Supervisor Domain Security Manager (SDSM) either through an SBI interface into M-mode (or may have the ability to edit MTT structures by virtue of how the MTT structure in memory is accessible to lower privilege levels).
- MTT checker - this functional block looks up the MTT using the physical address of the access as an index to retrieve the access permissions for the supervisor domain. This checker thus enforces that for a load initiated by the hart, the physical address is readable, and for a store initiated by the hart, the physical address is also writable, else reports a fault. An access violation is reported as a trap to the M-mode Root domain security manager, and the access is disallowed with no data divulged. This checker may be implemented as an MMU extension in the hart. The MTT checker is designed to work together with the page-based virtual memory (MMU) systems and Physical Memory Protection (PMP) mechanism. Read and Write

permissions for memory are derived from the page table, the PMP and the MTT - an access is allowed only when all protection mechanisms allow the access. When paging is enabled, instructions that access virtual memory may result in multiple physical-memory accesses, including (implicit S-mode) accesses to the page tables. MTT checks also apply to these implicit accesses - those accesses will be treated as reads for translation and as writes when A/D bits are updated in page table entries for Svadu is implemented).

- Physical address metadata selector (Svpams) - To support IO/memory sharing, a hart/device may perform accesses to memory exclusively accessible to its supervisor domain, and to memory shared globally with all supervisor domains or other specific supervisor domains. Memory sharing between supervisor domains is achieved by simply making the physical memory region accessible to the supervisor domains via the MTT structure. Access to physical addresses initiated from a hart or a device that is initialized with an MTT in non-`Bare` mode and valid supervisor domain identifier may be denied by virtue of the permissions in the MTT lookup - such disallowed accesses cause a trap to the RDSM to report a fault. When access to shared memory is allowed by the MTT, it may need to be additionally qualified to enforce downstream security-controls. To achieve this, a physical address metadata selector may be provided as part of the access. The possible metadata are specified in N(16?) of `MXLEN` bit CSRs per hart. The metadata that is associated with the accessed physical address is selected via a 4-bit physical address metadata selector field programmed into the S-mode or G-stage page table leaf entry traversed as part of the address translation. The domain workload is expected to manage the S-mode page table selectors, and the Supervisor domain security manager (operating in `HS-mode` is expected to manage the G-stage page table entry selectors. Additional usage information about Svpams is described below.

Additional protection/isolation for memory associated with a supervisor domain is orthogonal (and usage-specific). Such additional protection for memory may be derived by the use of cryptography and/or access-control mechanisms. The mechanisms chosen for these additional protection methods are independent of Smmtt and may be platform-specific, though they may utilize the physical address metadata selected during the access. The TCB of a particular supervisor domain may be independently evaluated via attestation of the HW and SW TCB by a relying party using standard Public-Key Infrastructure-based mechanisms.

Memory regions may be accessed by harts or by other devices on the platform. When harts and devices are assigned to a supervisor domain, the hart/device is said to perform memory accesses in the context of that supervisor domain. For all accesses using a physical address, the SDID is the supervisor domain identifier programmed into a CSR. This CSR is programmed on the hart by the Root Domain Security Manager (RDSM). The assignment of the hart/device to a supervisor domain may be static (e.g. device assignment to a VM) or dynamic (e.g. scheduling a VM virtual cpu within a domain). The MTT for the supervisor domain active on the hart is programmed on the hart along with the supervisor domain identifier. The MTT does not perform any address translation; it simply provides access permissions for the physically addressed region/page (post any S-mode and/or G-stage address translation) to enforce the isolation properties per the use case requirements (see Figure 2.1).

The assignment of devices and IOMMUs to supervisor domains is also expected to be under the purview of the RDSM. The specifics of device assignment are expected to be device/bus-specific and are out of scope of this specification.

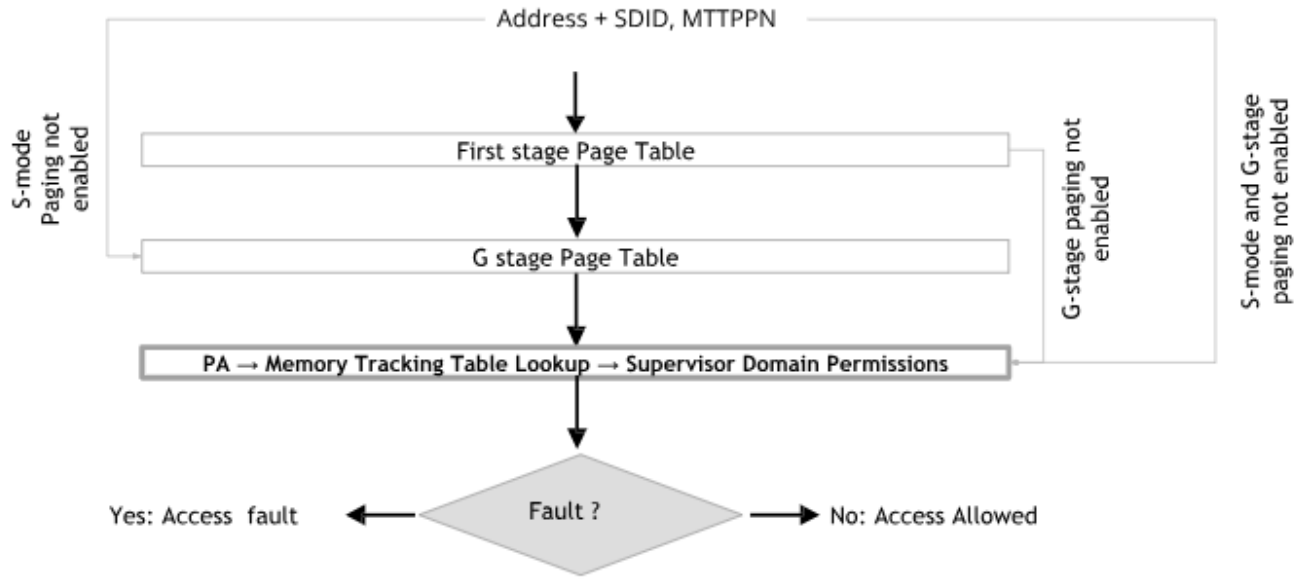

Figure 2: MTT lookup for Supervisor Domain Access

2.2. Supervisor domain physical address metadata selector (Svpams)

As described in the section above, M-mode software uses the MTT to create isolated physical memory regions associated with specific supervisor domains to enforce the required software access-control policies. Typically a supervisor domain has **domain exclusive** access to the memory region/pages of a region. Supervisor domains may also negotiate/grant access to a common/shared pool of memory regions/pages between supervisor domains - these memory regions may be accessible to all supervisor domains i.e. **domain global-shared** memory, or a specific subset of supervisor domains i.e., **domain restricted-shared** memory. Note here that the MTT enforces access-control based isolation only.

Some supervisor domain usages and threat models may also require cryptographic protection for memory (against forms of physical access attacks). In these cases, meta-data must be supplied to the platform as part of the physical memory accesses to identify the cryptographic contextto be used for those physical memory accesses - this metadata may be identified via a function that combines the SDID and additional execution context fields e.g. VMID, ASID, to identify a unique cryptographic context. The scope of such cryptographic context identifiers are typically the SoC platform but may vary by implementations.

When accessing shared memory from a supervisor domain workload, it is required for the workload and/or the supervisor domain security manager to be in control of specifying the cryptographic contexts to be used for memory accesses - this context can be selected via the **physical address metadata selector (Svpams).** In the minimal case, globally-shared memory setup for all supervisor domains may require the selection of at least two cryptographic contexts for memory - **domain exclusive** and **globally-shared** memory. For **domain restricted-shared** cases, other n-lateral setup cryptographic contexts can be selected via the Svpams capability. Other meta-data values associated with physical addresses may also be selected via the Physical Address Metadata Selector.

The Svpams mechanism is independent of the Smmtt mechanism since Svpams enables additional properties to be enforced for memory accesses beyond the access-control/isolation mechanism that Smmtt enables. Smmtt is covered in chapter 3 and 4. Svpams is covered in Section 5.

Chapter 3. Supervisor Domain Identifier and Protection Register (mttp)

The mttp register is an XLEN-bit read/write register, formatted as shown in Figure 3.1 for XLEN=32 and Figure 3.2 for XLEN=64, which controls physical address protection for supervisor domains. This register holds the physical page number (MTTPPN) of the root page of the memory tracking table (MTT), a supervisor domain identifier (SDID), which facilitates address protection fences on a per-supervisor-domain basis; and the MODE field, which selects the address protection scheme (MTT Mode to be enforced) for physical addresses. The MTT is a structure that holds the access permissions for a physical address and is looked up per the programmed MODE.

Attempts to read or write mttp while executing in U, S or HS-mode will raise an illegal instruction exception.

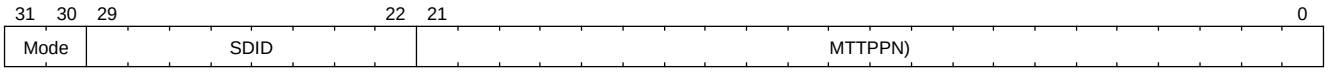

Figure 1. M-mode MTTP register (mttp) when XLEN=32 for MODE values Bare, Smmtt34. All sub-fields are WARL.

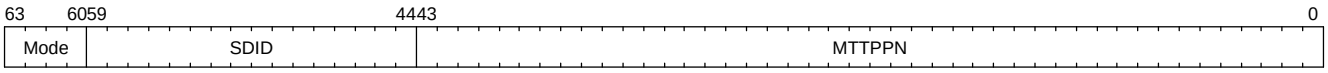

Figure 2. M-mode MTTP register (mttp) when XLEN=64 for MODE values Bare, Smmtt46, Smmtt56. All sub-fields are WARL.

Table 3.1 shows the encodings of the MODE field when XLEN=32 and XLEN=64. When mttp MODE=Bare, supervisor physical addresses have no MTT-based protection across supervisor domains beyond the physical memory protection scheme described in Section 3.7 of the RISC-V privileged architecture specification [1]. In this case, the remaining fields (SDID, MTTPPN) in mttp must be set to zeros, else generate a fault. When XLEN=32, the other valid settings for MODE are Smmtt34 and Smmtt34rw, to support allow/disallow and read-write access permissions for 34-bit system physical addresses.

When XLEN=64, other than BARE, the other valid settings for MODE are Smmtt[46, 56][rw] to support read-write/access permissions for 56-bit system physical addresses.

The remaining MODE settings when XLEN=64 are reserved for future use and may define different interpretations of the other fields in mttp.

Table 3.2: Encoding of mttp MODE field for XLEN=64.

Value	Name	Description
0	Bare	No inter-supervisor domain protection
1	Smmtt34	Page-based supervisor domain protection for 34 bit physical addresses with access allowed/disallowed per page
2	Smmtt34rw	Page-based supervisor domain protection for 34 bit physical addresses with RW permissions per page

Table 3.1: Encoding of mttp MODE field for XLEN=32.

Value	Name	Description
0	Bare	No inter-supervisor domain protection
1	Smmtt46	Page-based supervisor domain protection for 46 bit physical addresses
2	Smmtt46rw	Page-based supervisor domain protection for 46 bit physical addresses with RW permissions per page
3	Smmtt56	Page-based supervisor domain protection for 56 bit physical addresses
4	Smmtt56rw	Page-based supervisor domain protection for 46 bit physical addresses with RW permissions per page
5-15	-	Reserved

Implementations are not required to support all defined MODE settings when XLEN=64. A write to mttp with an unsupported MODE value is not ignored. Instead, the fields of mttp are WARL in the normal way, when so indicated.

The MTTPPN refers to an MTTL3 table or an MTTL2 table based on physical address width (PAW). For 56⇐PAW<46, MTTL3 table must be of size 2^(PAW-43) bytes and naturally aligned to that sized byte boundary. For 46⇐PAW<32 the MTTL2 table must be of size 2^(PAW-23) or 2^(PAW-22) bytes (depending on the Smmtt MODE selected) and must be naturally aligned to that sized byte boundary. In these modes, the lowest two bits of the physical page number (MTTPPN) in mttp always read as zeros.

The number of SDID bits is UNSPECIFIED and may be zero. The number of implemented SDID bits, termed SDIDLEN, may be determined by writing one to every bit position in the SDID field, then reading back the value in mttp to see which bit positions in the SDID field hold a one. The least-significant bits of SDID are implemented first: that is, if SDIDLEN > 0, SDID[SDIDLEN-1:0] is writable. The maximal value of SDIDLEN, termed SDIDMAX, is 8 for Smmtt34[rw] or 16 for Smmtt46[rw], Smmtt56[rw].

The mttp register is considered active for the purposes of the physical address protection algorithm unless the effective privilege mode is M.

Physical accesses that began while mttp was active are not required to complete or terminate when mttp is no longer active, unless an FENCE.MTT instruction matches the SDID (and optionally, PA) is executed. The FENCE.MTT instruction must be used to ensure that updates to the MTT data structures are observed by subsequent implicit reads to those structures by a hart.

Note that writing mttp does not imply any ordering constraints between S-mode and G-stage page-table updates and subsequent address translations. If a supervisor domain's MTT structure has been modified, or if a SDID is reused, it may be necessary to execute a FENCE.MTT instruction before or after writing mttp.

3.1. Smmtt: Page-based Physical Address Protection for Supervisor Domains

3.1.1. Smmtt[34, 46, 56]rw

The MTTPPN rooted structure for the MTT is shown below. The structure below shows a 56 bit

physical address lookup; for lower physical address widths e.g. 46 bits, the MTTL3 table is not applicable. In this mode of the Smmtt[34, 46, 56]rw, each page is associated with a read and a write access permission (2 bits). Figure 4.1.1 shows the Physical address (PA) being used to index into the MTT structure in memory to lookup access permissions for the supervisor domain specified in the MTTL1 entry. Intermediate MTTL3 and MTTL2 entries are used to allow this structure to be sparsely populated.

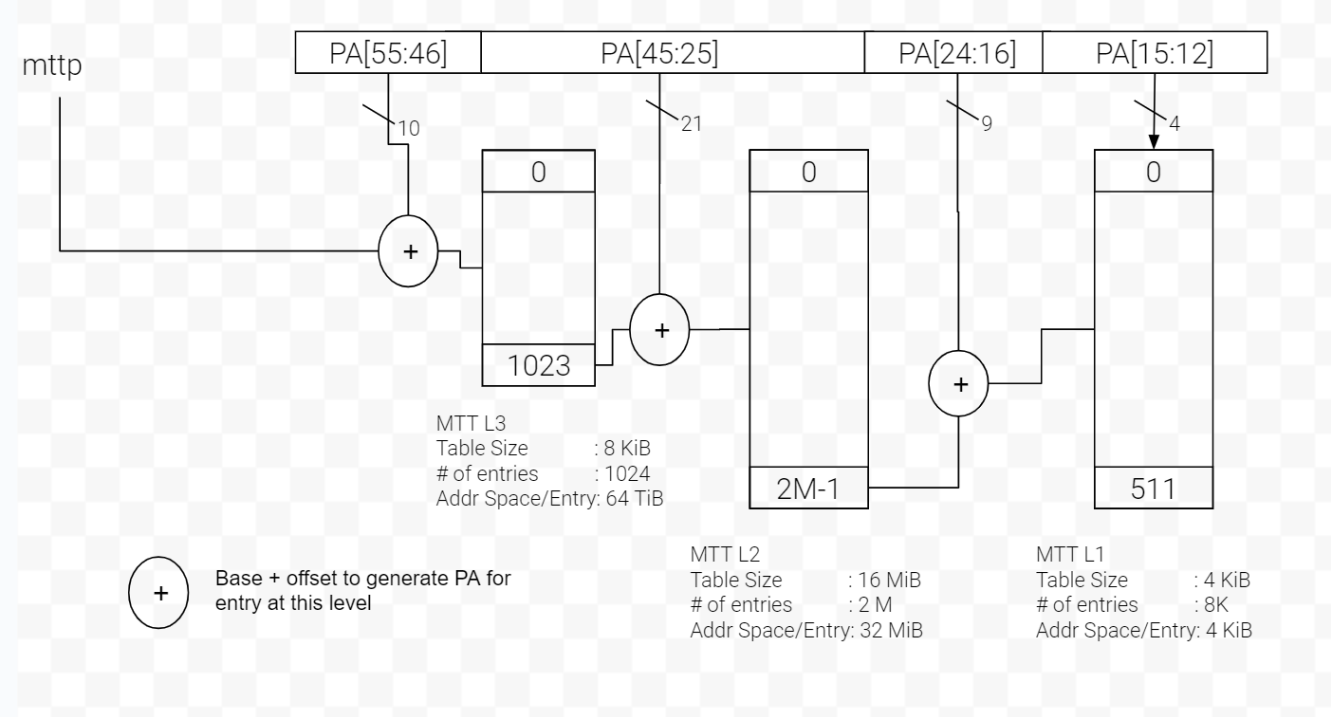

Figure 3: MTT Structures (overview)

Figure 3. MTTL3 Entry register

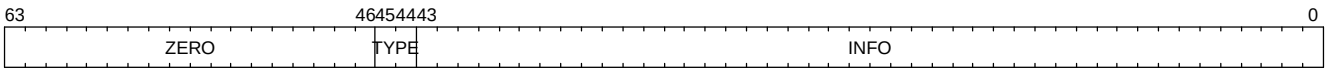

Figure 4. MTTL2 Entry register

The ZERO field must always be 0.

The TYPE field determines the interpretation of the MTTL2 entry. The TYPE field encoding is as follows:

- 0000b - 1G_disallow - read and write access is not allowed for the 1G range for this supervisor domain
- 0001b - 1G_allow_r - read access for the 1G range is allowed for the supervisor domain
- 0011b - 1G_allow_rw - read and write access for the 1G range is allowed for the supervisor domain
- 0100b - MTT_L1_DIR - The 32M range is composed of 8192x4K pages
- 0111b - 2M_PAGES - The 32M range is composed of 16x2M pages
- All other encodings are reserved and cause an access violation

The INFO field depends on the TYPE field and is formatted as per the following table:

Table 4.1.1: MTTL2 Entry Type for Smmtt

MTTL2 Entry TYPE	Description, INFO and TYPE field encoding
1G_disallow	*The 1G range of address is not allowed for the domain.* The INFO field must be 0. When configuring 1G ranges, RDSM ensures that 16 MTTL2 entries, each corresponding to 32M of address space, have identical TYPE field values.
1G_allow_r	*The 1G range of address is allowed (read only) for the domain.* The INFO field must be 0. When configuring 1G ranges, RDSM ensures that 16 MTTL2 entries, each corresponding to 32M of address space, have identical TYPE field values. The INFO field must be 0.
1G_allow_rw	*The 1G range of address is allowed (read/write) for the domain.* The INFO field must be 0. When configuring 1G ranges, RDSM ensures that 16 MTTL2 entries, each corresponding to 32M of address space, have identical TYPE field values.
MTT_L1_DIR	The INFO field provides the PPN of the MTTL1 page. Entries of the MTTL1 page hold a 4-bit PERM (permissions) field to indicate the access for the supervisor domain (described in the MTTL1 entry - see figure 4.1.4).
2M_PAGES	*The 32M range of address space is partitioned into 2M pages where each page has read/write access allowed/not.* The INFO field 31:0 holds 2 PERM bits per 2M address range to indicate access_disallowed (00b), read_allowed (01b), read_write_allowed (11b); (10b is reserved). Bits 32:43 are reserved and must be zero.

MTTL1 table is populated if protection granularity of the 4KiB page is desired for the supervisor domain.

MTTL1 entry is indexed using PA[24:12] and each entry is a 4-bit PERM field in the referenced page. The bits 24:16 are used to select a 64-bit field in the page and bits 15:12 are used to select a 4 bit PERM field in those 64 bits. Thus, there are 4 PERM bits for each 4 KiB page. The encoding of PERM is as follows:

- 0000b - the 4K page specifies access is **not allowed** for the domain
- 0001b - the 4K page specifies **read** access is allowed for the domain
- 0011b - the 4K page specifies **read** and **write** access is allowed for the domain
- Remaining encodings are **reserved** and cause an access violation.

Figure 5. MTTL1 Entry

3.1.2. Smmtt[34, 46, 56]

The MTTPPN rooted structure for the MTT is shown below. The structure below shows a 56 bit physical address lookup; for lower physical address widths e.g. 46 bits, the L3 table is not applicable. In this mode of the SmmttX, each page is associated with an access allowed/disallowed

permission (1 bit) to allow for efficient caching.

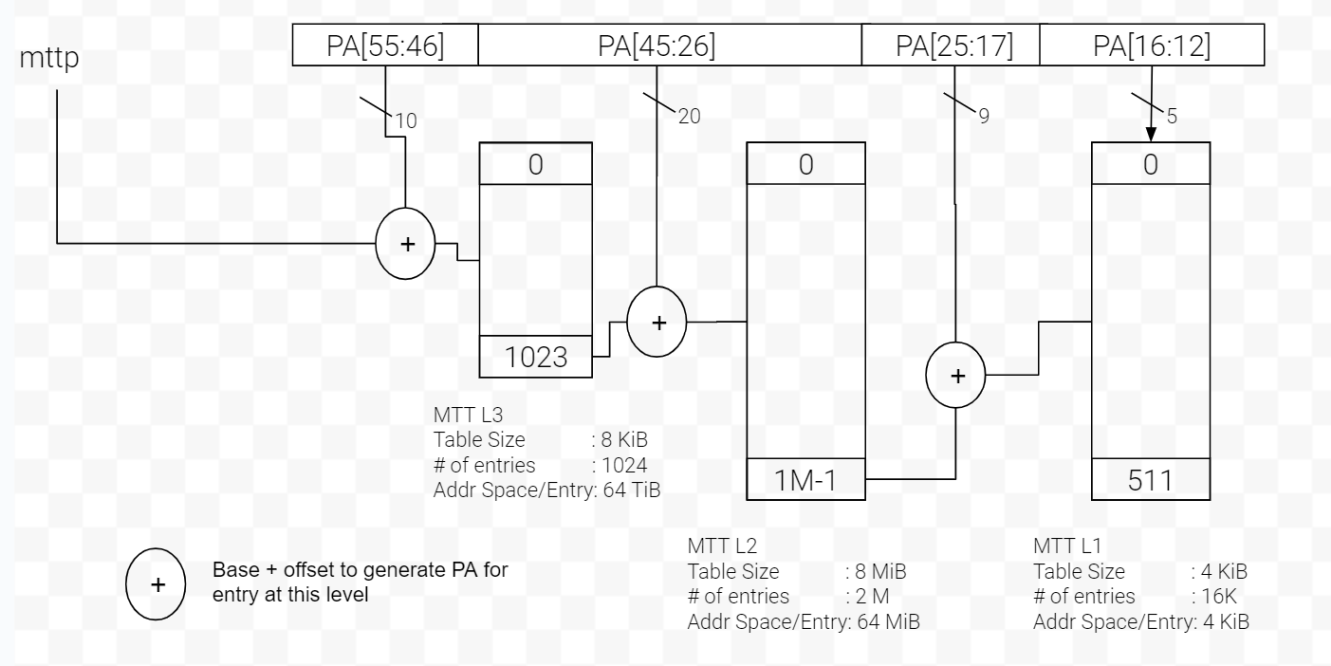

Figure 4: MTT Structures (overview)

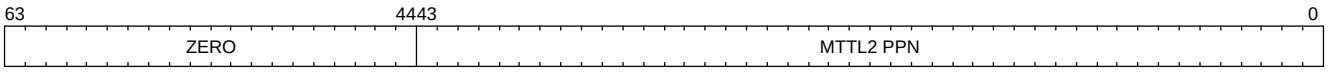

Figure 6. `MTTL3` Entry register

Figure 7. `MTTL2` Entry register

The `ZERO` field must always be 0.

The `TYPE` field determines the interpretation of the `MTTL2` entry. The `TYPE` field encoding is as follows:

- 00b - `1G_disallow` - access to the 1G range is disallowed
- 01b - `1G_allow` - access to the 1G range is allowed
- 10b - `MTT_L1_DIR` - The 64M range is composed of 16384x4K pages
- 11b - `2M_PAGES` - The 64M range is composed of 32x2M pages

The `INFO` field depends on the `TYPE` field and is formatted as per the table:

Table 4.2.1: `MTTL2` Entry for Smmtt[34, 46, 56]

`MTTL2` Entry Type	Description, `INFO` and `TYPE` field encoding
`1G_allow`	*The 1G range of address is allowed for the domain.* The `INFO` field must be 0. When configuring 1G ranges, RDSM ensures that 16 `MTTL2` entries, each corresponding to 64M of address space, have identical `TYPE` field values.

1G_disallow	*The 1G range of address is not allowed for the domain.* The INFO field must be 0. When configuring 1G ranges, RDSM ensures that 16 MTTL2 entries, each corresponding to 64M of address space, have identical TYPE field values.
MTT_L1_DIR	The INFO field provides the PPN of the MTTL1 page. Entries of the MTTL1 page hold a 2-bit PERM field to indicate the access for the supervisor domain (described in the MTTL1 entry - see figure 4.2.4).
2M_PAGES	*The 64M range of address space is partitioned into 2M pages where each page has access allowed/not.* The INFO field bits 31:0 holds 1 bit per 2M address range to indicate access disallowed(0b) or allowed (1b). INFO field bits 43:32 are reserved (must be zero).

MTTL1 table is populated if 4KiB page confidential pages are required for the supervisor domain.

MTTL1 entry is indexed using PA[25:12] and each entry is a 2-bit PERM field in the referenced page. The bits 25:16 are used to select a 64-bit field in the page and bits 16:12 are used to select a 2 bit PERM field in those 64 bits. Thus, there are 2 PERM bits for each 4 KiB page. The encoding is as follows:

- 00b - the 4K page specifies access is **not allowed** for the domain
- 01b - the 4K page specifies access is **allowed** for the domain
- 1xb - **reserved** (access causes access violation).

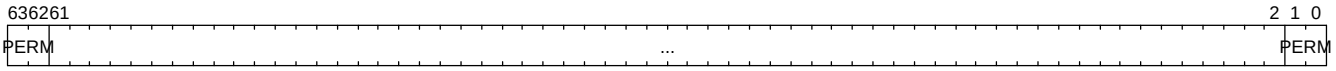

Figure 8. MTTL1 Entry

Caching

Implementations with virtual memory are permitted to cache translations and permissions in address translation cache structures. Similarly, access permissions from the MTT lookup may be cached. The PMP and MTT settings for the resulting physical address may be checked (and possibly cached) at any point between the address translation and the explicit memory access. If caching is occuring, when the MTT settings are modified, M-mode software must synchronize the cached MTT state with the virtual memory system and any PMP, MTT or address-translation caches. This is accomplished by executing an SFENCE.VMA instruction with rs1=x0 and rs2=x0, or HFENCE.GVMA as needed, after the MTT is modified. If page-based virtual memory is not implemented, memory accesses check the PMP settings synchronously, but may check MTT settings that are cached, so a MTT invalidation (MTTINVAL) instruction is needed. When Svinval is implemented, MTTINVAL is only ordered against SFENCE.W.INVAL and SFENCE.INVAL.IR instructions. As part of the MTT update, the RDSM must ensure that it uses SFENCE.W.INVAL to guarantee that any previous stores to MTT are made visible before invoking the MTTINVAL. The RDSM must then use SFENCE.INVAL.IR to guarantee that all subsequent implicit references to MTT are ordered to be after the MTT cache invalidation.

[TBD - register interface for flushing all MTT cached entries, vs specific physical address at page size granularity].

Data and instructions cached for one supervisor domain shall not be accessible to another supervisor domain, so the `SDID` (along with the other execution context identifiers such as `ASID`, `VMID`, shall factor into the data and instruction caches tags to enforce isolation.

Chapter 4. Svpams: Physical Address Metadata Selector

Svpams may be enabled via the menvcfg (and henvcfg) registers. Once enabled, the 4 bit PA metadata is derived as part of the address translation and is used to select 1 of 16 possible metadata qualifiers that should be applied to accesses that use the translated address. Any caching structures that cache the address translation may also cache the PA meta-data as part of the cached translation.

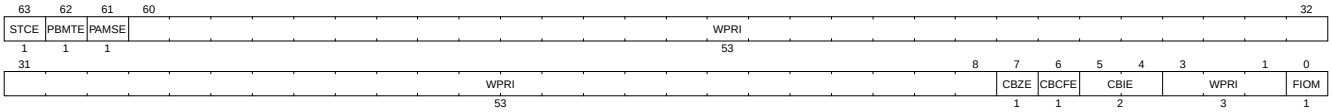

Figure 9. Enabling Svpams via PAMSE bit in menvcfg and menvcfgh

Figure 10. Enabling Svpams via PAMSE bit in henvcfg and henvcfgh

Figure 11. Page table entry for Sv57, Sv48, Sv39 with PAMS control field

Page table entry for Sv32 (TBD): Only option is to use bits 8:9 in the pte for a 2-bit PAMS

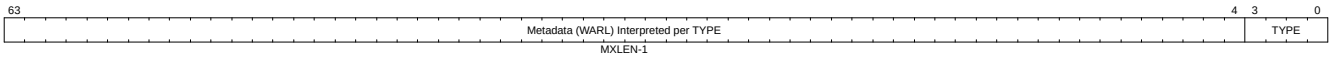

Figure 12. Svpams metadata CSR0 - CSR15

16 such CSRs are available from CSR0 - CSR15.

PAMS Types: 0 - None - Metadata is Reserved (must be zero). 1 - Encryption context identifier - Metadata is an encryption context identfier (WARL).

Other metadata types may be defined in the future. Should the same CSRs be indexable from PMP also?

When two-stage translation is not enabled in a supervisor domain, and satp.MODE is not zero, S-stage PAMS selects the physical address metadata. When two-stage address translation is enabled using the H-extension in a supervisor domain, If the hgatp.MODE is not zero, G-stage PAMS is used to select the metadata and VS-stage PAMS is ignored (VS-stage PAMS becomes reserved). This model is enforced to enforce the policy per the highest privilege supervisor entity in the supervisor domain.

Chapter 5. Static and Runtime Configuration

MTT must support both static and run-time configurability. A memory region (consisting of one or more pages) may be (re)assigned from one domain to another at run-time e.g. this is done by revoking the permission for one domain and assigning permissions to another domain. Run-time configuration may be performed via M-mode CSRs and/or in-memory structures. The in-memory structures used for MTT must themselves be access-limited to the RDSM by use of the MTT structures to disallow any supervisor domain from accessing the structures unless explicitly delegated by the Root Domain Security Manager (RDSM) to a particular domain (per use case policies). To support MTT dynamic reconfiguration, an interface is expected to be provided to set the attributes by passing requests to a trusted driver (in the RDSM) that can reconfigure the memory region assignment. Converting memory regions assignment from one domain to another might involve platform-specific operations based on the enforcement mechanism, such as TLB/cache flushes, that must be enforced by the RDSM and hardware. The RDSM is expected to change the settings and flush caches if necessary, so the system is only incoherent during the transition between domain assignment settings. This transitory state should not be visible to lower privilege levels (i.e. supervisor domains).

5.1. Platform considerations

MTT may be used to provide permissions for physical memory addresses that hold regular main memory or IO memory. Memory may be assigned to the RDSM to bootstrap the subsequent run-time lookup structures for MTT. All memory should be covered by the MTT, though some memory may not be eligible to be qualified for assignment to a specific supervisor domain. This limitation may arise due to platform configuration and security policies - for example, if the platform security policy requires memory for a domain to be encrypted and some memory access paths are not enforced via an inline memory encryption engine. It is expected that the RDSM can use trusted platform-specific methods to enumerate which regions can be designated as access-controlled via the MTT.

Chapter 6. MTT Checker Implementation considerations

MTTs are checked by the MTT checker for all accesses to eligible physical memory, including accesses that have undergone virtual to physical memory translation, but excluding MTT structure accesses. The MTT checker indexes the MTT using the physical address of the access to retrieve the access permissions, and checks that the hart is allowed to access the physical memory accessed. A mismatch of the access type and the access permissions specified in the MTT entry that applies to the accessed region is reported as a trap to the RDSM and the access is disallowed with no data divulged. As described above, to support architectural virtual address page sizes, the MTT allows configuration at those supported architectural page sizes. MTT violations manifest as instruction, load, or store access-fault exceptions. The exception conditions for MTT are checked when the access to memory is performed.

The intra-domain isolation of memory between two harts/devices belonging to the same supervisor domain, but different tenant workloads, is achieved via the use of MMU, (S)PMP, IOMMU and IOPMP depending on the type of platform and the type of access. To successfully achieve this isolation, the page table structures for a domain's workloads must be managed by the Supervisor Domain Security Manager (SDSM) and the paging structures must be located in memory exclusively-accessible only to the Supervisor Domain. Additional security properties may be enforced based on type (data fetch, instruction fetch, etc.) and locality (hart supervisor domain identifier) of memory accesses as required for the security policy specific to usages. An example policy may be to require certain accesses to target only exclusively-owned domain memory. The MTT checker may utilize the SDID or the Svpams derived metadata to enforce such policies. The description of Supervisor Domain policies is outside the scope of this document.

Different types of trust models are possible to build with domain isolation. For example, with mutually distrusting domains, the RDSM will always enforce the MTT check on all accesses. But if one domain is considered at a higher trust level than the other, then the RDSM may choose to program the MTT checker to bypass MTT checks for the higher trust domain but not the lower trust one to optimize performance for the higher trust domain workloads. To support such an option, the hart MTT CSR mode field must allow the RDSM to explicitly disable the MTT lookup for RDSM-selected SDIDs. The CSR allows this by setting the MTT to Bare mode for such cases.

Chapter 7. Extensibility

In addition to the access permissions looked up via the MTT structure, additional access qualifiers and metadata may be maintained and/or carried/associated on the SoC fabric to support other capabilities. MTTs must apply to domain-related hart and non-hart accesses. This enables other hardware agents (for example devices connected over interconnects, IOMMU etc.) to be able to access domain memory in a consistent manner. Devices assigned to a domain may generate memory access requests with the appropriate domain identifier for evaluation by the IO-side access checks. IO-side access checks enforce the domain isolation of physical memory by using the MTT checker similar to the hart-side MTT checker described above. Trust establishment of devices attached to an exposed interconnect is outside the scope of this specification and should be performed via a fabric specific mechanism (for example, see DMTF SPDM and PCIe Trusted Device Interface Security Protocol). See Figure 10.1 below for an example use of supervisor domains in a SoC.

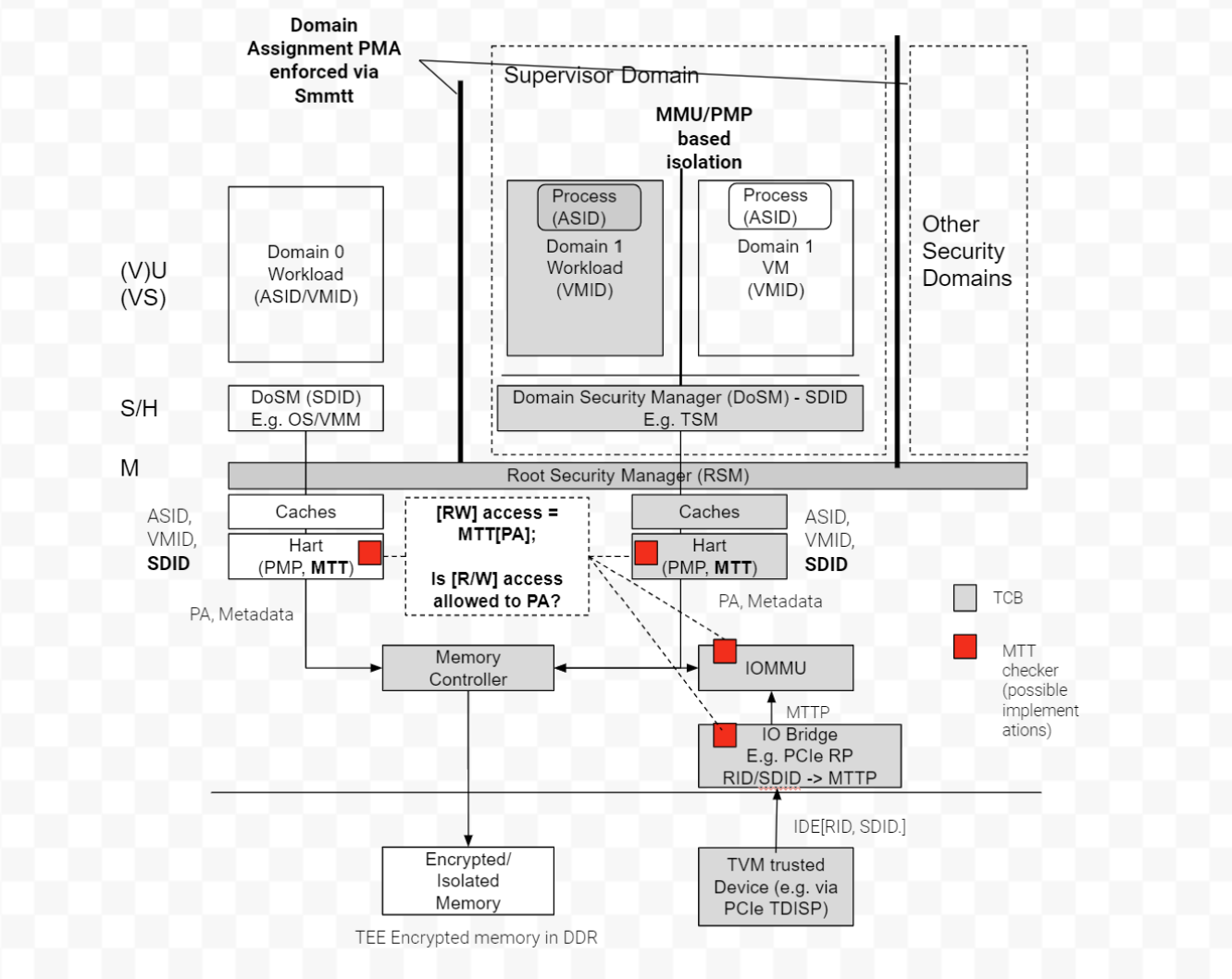

Figure 5: Example of MTT Checker in a SoC