Physical Memory Protection

Design Document

Contents

List of Figures

List of Tables

Revision Table

1. Introduction

Physical Memory protection (PMP) controls memory access permissions for specific regions of the physical memory. It can grant special memory access to lesser privileged modes such as U (user) and S (Supervisor) mode. It allows certain memory regions to have read(R),write(W) and execute(X) permissions using a set of M-mode controlled CSR's.

1.1. PMP unit features :

1. RISC-V PMP limits the physical addresses accessible by software running on a hart (hardware thread).
2. Can grant R/W/X permissions on >= 4 byte granularity.
3. Up to 16 PMP entries are supported.
4. PMP entries are described by an 8-bit configuration register and one 32 (or 64) bit address register.
5. PMP's can be locked.
6. PMP can grant permissions to S and U modes, which by default have none, and can revoke permissions from M-mode, which by default has full permissions.

1.2. Terminology

1. **PMP** : Physical Memory protection
2. **U** : User Mode
3. **S** : Supervisor Mode
4. **M** : Machine Mode
5. **R** : Read
6. **W** : Write
7. **X** : Execute
8. **L** : Locked
9. **A** : Address matching mode
10. **NAPOT** :Naturally aligned power-of-2 regions
11. **TOR** : Top boundary of an arbitrary range
12. **CSR** : Control and Status Register
13. **HW** : Hardware
14. **SW** : Software

15. **R/W** : Read and write
16. **MRW** : Machine Read Write
17. **WLRL** : Write/Read Only Legal Values

2. Privilege Levels

Privilege levels are used to provide protection between different components of the software stack, and attempts to perform operations not permitted by the current privilege mode will cause an exception to be raised.

At any time, a RISC-V hardware thread (hart) is running at some privilege level encoded as a mode in one or more CSRs (control and status registers).

Three RISC-V privilege levels are currently defined as shown in Table 1.

Level	Encoding	Name	Abbreviation
0	00	User/Application	U
1	01	Supervisor	S
2	10	Reserved	
3	11	Machine	M

Table 1: RISC-V privilege levels.

- The machine level has the highest privileges and is the only mandatory privilege level for a RISC-V hardware platform.
- Code run in machine-mode (M-mode) is usually inherently trusted, as it has low-level access to the machine implementation.
- M-mode can be used to manage secure execution environments on RISC-V.
- User-mode (U-mode) and supervisor-mode (S-mode) are intended for conventional application and operating system usage respectively.

Number of levels	Supported Modes	Intended Usage
1	M	Simple embedded systems
2	M, U	Secure embedded systems
3	M, S, U	Systems running Unix-like operating systems

Table 2: Supported combinations of privilege modes.

All hardware implementations must provide M-mode, as this is the only mode that has unfettered access to the whole machine. The simplest RISC-V implementations may provide only M-mode, though this will provide no protection against incorrect or malicious application code.

Many RISC-V implementations will also support at least user mode (U-mode) to protect the rest of the system from application code.

A hart normally runs application code in U-mode until some trap (e.g., a supervisor call or a timer interrupt) forces a switch to a trap handler, which usually runs in a more privileged mode. The hart will then execute the trap handler, which will eventually resume execution at or after the original trapped instruction in U-mode.

3. Physical Memory Protection

To support secure processing and contain faults, it is desirable to limit the physical addresses accessible by software running on a hart. An optional physical memory protection (PMP) unit provides per-hart machine-mode control registers to allow physical memory access privileges (read, write, execute) to be specified for each physical memory region.

PMP unit consists of two blocks

1. PMP Checks
2. PMA Checks

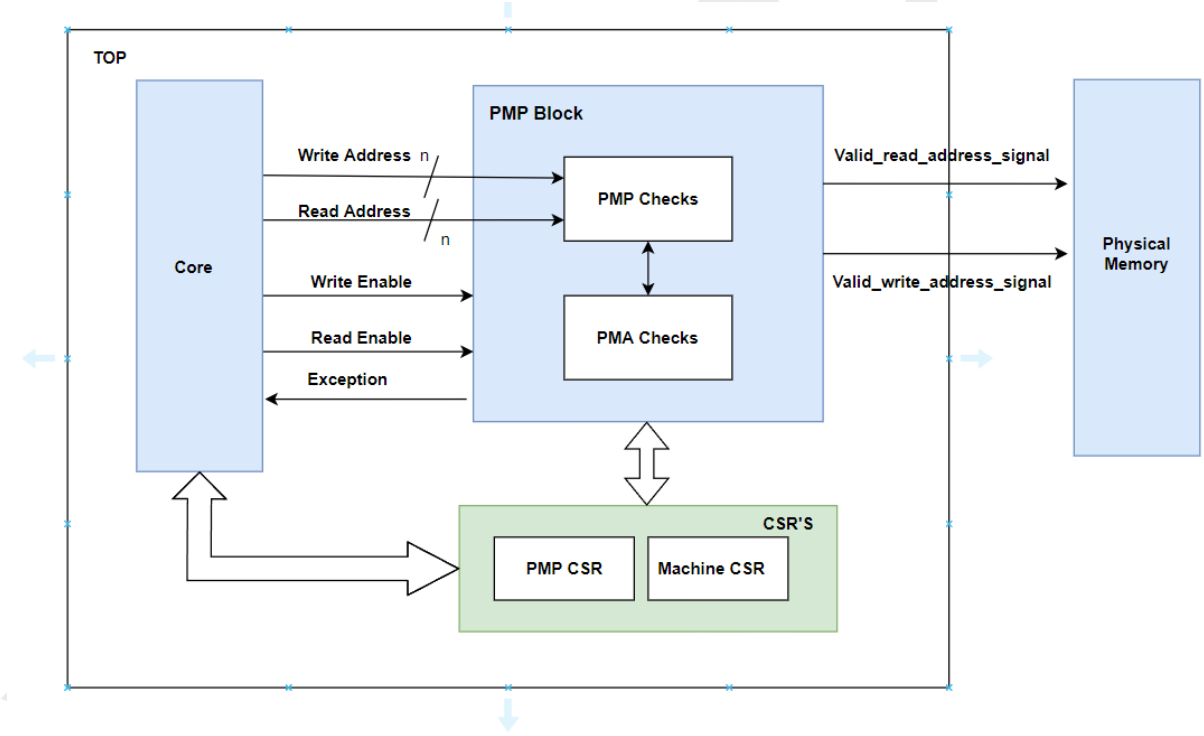


Figure 1 : Block diagram of Physical Memory Protection unit

3.1. PMA Checks (Physical Memory Attribute Checks)

- PMA checks block is a combinational logic with comparators which checks the regions readability,writability and executability for a particular address range.
- The physical memory map for a complete system includes various address ranges, some corresponding to memory regions, some to memory-mapped control registers, and some to empty holes in the address space.

- Some memory regions might not support reads, writes, or execution; some might not support subword or subblock accesses; some might not support atomic operations; and some might not support cache coherence or might have different memory models.
- Similarly, memory mapped control registers vary in their supported access widths, support for atomic operations, and whether read and write accesses have associated side effects.
- In RISC-V systems, these properties and capabilities of each region of the machine's physical address space are termed physical memory attributes (PMAs).

3.2. PMP Checks(Physical Memory Protection Checks)

- PMP checks block is a combinational logic with comparators which compares the generated address for each load and store instruction in reference with pmpcsr's.
- The PMP values are checked in parallel with the PMA checks.
- PMP checks are applied to all accesses when the hart is running in S or 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 S or U.
- Optionally, PMP checks may additionally apply to M-mode accesses, in which case the PMP registers themselves are locked, so that even M-mode software cannot change them without a system reset.
- PMP violations are always trapped precisely at the processor.

4. PMP Work-Flow

The PMP unit will work as mentioned in the below steps,

Step 1: PMP unit checks load or store instruction trying to access the physical memory.

Step 2: checks the MPP and MPRV bits of mstatus register to check whether it is in user mode or machine mode.

Step 3: PMP unit checks whether all pmpcsr and regions are configured.

Step 4: Matching PMP address access with pmp entry.

- PMP entries are statically prioritized. The lowest-numbered PMP entry that matches any byte of an access determines whether that access succeeds or fails.
- The matching PMP entry must match all bytes of an access, or the access fails, irrespective of the L, R, W, and X bits.
- For example, if a PMP entry is configured to match the four-byte range 0xC–0xF, then an 8-byte access to the range 0x8–0xF will fail, assuming that PMP entry is the highest-priority entry that matches those addresses.

Step 5: PMA Checks

- If a PMP entry matches all bytes of an access, then the L, R, W, and X bits determine whether the access succeeds or fails.
- If the L bit is clear and 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 S or U, then the access succeeds only if the R, W, or X bit corresponding to the access type is set.

Step 6:Exception Generation

- Failed accesses generate a load, store, or instruction access exception.
- Attempting to fetch an instruction from a PMP region that does not have execute permissions raises an instruction access-fault exception.
- Attempting to execute a load or load-reserved instruction which accesses a physical address within a PMP region without read permissions raises a load access-fault exception.
- Attempting to execute a store, store-conditional, or AMO instruction which accesses a physical address within a PMP region without write permissions raises a store access-fault exception.
- Note that a single instruction may generate multiple accesses, which may not be mutually atomic.
- An access exception is generated if at least one access generated by an instruction fails, though other accesses generated by that instruction may succeed with visible side effects.

5. CSR Field Specifications

Reserved Writes Ignored, Reads Ignore Values (WIRI)

Some read-only and read/write registers have read-only fields reserved for future use. These reserved read-only fields should be ignored on a read. Writes to these fields have no effect, unless the whole CSR is read-only, in which case writes might raise an illegal instruction exception. These fields are labeled WIRI in the register descriptions.

Reserved Writes Preserve Values, Reads Ignore Values (WPRI)

Some whole read/write fields are reserved for future use. Software should ignore the values read from these fields, and should preserve the values held in these fields when writing values to other fields of the same register. These fields are labeled WPRI in the register descriptions.

Write/Read Only Legal Values (WLRL)

Software should not write anything other than legal values to such a field, and should not assume a read will return a legal value unless the last write was of a legal value

Write Any Values, Reads Legal Values (WARL)

Allow any value to be written while guaranteeing to return a legal value whenever read.

6. PMP CSRs Memory Map

Address	Privilege	Name	Description
0x300	MRW	mstatus	Machine status register.
0x3A0	MRW	pmpcfg0	Physical memory protection configuration.
0x3A1	MRW	pmpcfg1	Physical memory protection configuration, RV32 only.
0x3A2	MRW	pmpcfg2	Physical memory protection configuration.
0x3A3	MRW	pmpcfg3	Physical memory protection configuration, RV32 only.
0x3B0	MRW	pmpaddr0	Physical memory protection address register.
0x3B1	MRW	pmpaddr1	Physical memory protection address register.
0x3B2	MRW	pmpaddr2	Physical memory protection address register.
0x3B3	MRW	pmpaddr3	Physical memory protection address register.
0x3B4	MRW	pmpaddr4	Physical memory protection address register.
0x3B5	MRW	pmpaddr5	Physical memory protection address register.
0x3B6	MRW	pmpaddr6	Physical memory protection address register.
….	…	..	….
0x3BF	MRW	pmpaddr15	Physical memory protection address register.

Table 3 : PMP CSRs Memory Map

7. Machine Status Register(mstatus)

The mstatus register keeps track of and controls the hart's current operating state.

The mstatus register is an XLEN-bit read/write register formatted as shown in Figure 3.6 for RV32 and Figure 3.7 for RV64 and RV128.

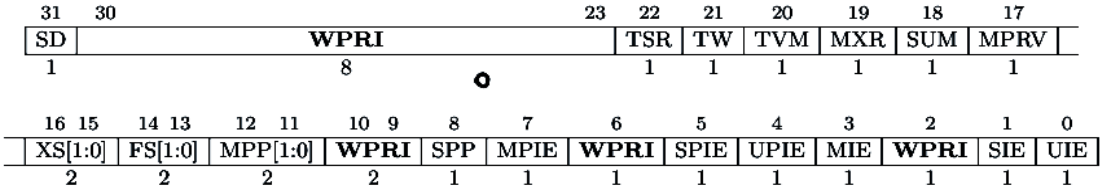


Figure 2 : Machine-mode status register (mstatus) for RV32.

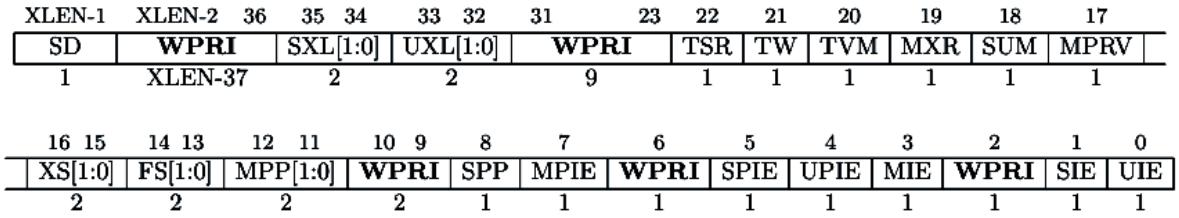


Figure 3 : Machine-mode status register (mstatus) for RV64 and RV128.

Bit	Field	Description	Access	Preset
31:18		Reserved, hardwired to 0		
17	MPRV	The **MPRV (Modify PRiVilege)** bit modifies the privilege level at which loads and stores execute in all privilege modes. When MPRV=0 hart will be in machine mode. When MPRV=1,user mode or supervisor mode	WARL	
16:13		Unimplemented, hardwired to 0		
12:11	MPP	**MPP : Machine Previous Privilege** level holds the previous privilege mode Mpp = 00 : user mode 01 :supervisor mode 10 : reserved	WLRL	

		11 : machine mode		
10:8		Unimplemented, hardwired to 0		
7	MPIE	**Previous Machine Interrupt Enable:** When an exception is encountered, MPIE will be set to MIE. When the mret instruction is executed, the value of MPIE will be stored to MIE.	WARL	
6:5		Unimplemented, hardwired to 0		
4	UPIE	**Previous User Interrupt Enable:** If user mode is enabled, when an exception is encountered, UPIE will be set to UIE. When the uret instruction is executed, the value of UPIE will be stored to UIE.	WARL	
3	MIE	**Machine Interrupt Enable:** If you want to enable interrupt handling in your exception handler, set the Interrupt Enable MIE to 1 inside your handler code.	WARL	
2:1		Unimplemented, hardwired to 0		
0	UIE	**User Interrupt Enable:** If you want to enable user level interrupt handling in your exception handler, set the Interrupt Enable UIE to 1 inside your handler code.	WARL	

Table 4 : Machine-mode status register (mstatus) bit field.

7.1. Memory Privilege in mstatus Register

- The MPRV (Modify PRiVilege) bit modifies the privilege level at which loads and stores execute in all privilege modes.

- When MPRV=0, translation and protection behave as normal.
- When MPRV=1, load and store memory addresses are translated and protected as though the current privilege mode were set to MPP.
- Instruction address-translation and protection are unaffected. MPRV is hardwired to 0 if U-mode is not supported.

7.2. Privilege and Global Interrupt-Enable Stack in mstatus register

Interrupt-enable bits, MIE, SIE, and UIE, are provided for each privilege mode. These bits are primarily used to guarantee atomicity with respect to interrupt handlers at the current privilege level. When a hart is executing in privilege mode x, interrupts are enabled when x IE=1. Interrupts for lower privilege modes are always disabled, whereas interrupts for higher privilege modes are always enabled. Higher-privilege-level code can use separate per-interrupt enable bits to disable selected interrupts before ceding control to a lower privilege level.

To support nested traps, each privilege mode x has a two-level stack of interrupt-enable bits and privilege modes. xPIE holds the value of the interrupt-enable bit active prior to the trap, and xPP holds the previous User-level interrupts are an optional extension and have been allocated the ISA extension letter N. If user-level interrupts are omitted, the UIE and UPIE bits are hardwired to zero. For all other supported privilege modes x, the x IE and xPIE must not be hardwired.

User-level interrupts are primarily intended to support secure embedded systems with only M mode and U-mode present, but can also be supported in systems running Unix-like operating systems to support user-level trap handling. privilege mode. The xPP fields can only hold privilege modes up to x, so MPP is two bits wide, SPP is one bit wide, and UPP is implicitly zero. When a trap is taken from privilege mode y into privilege mode x, xPIE is set to the value of x IE; x IE is set to 0; and xPP is set to y.

The MRET, SRET, or URET instructions are used to return from traps in M-mode, S-mode, or U-mode respectively. When executing an xRET instruction, supposing xPP holds the value y, x IE is set to xPIE; the privilege mode is changed to y; xPIE is set to 1; and xPP is set to U (or M if user-mode is not supported).

xPP fields are WLRL fields that need only be able to store supported privilege modes, including x and any implemented privilege mode lower than x.

8. PMP CSR'S

- PMP controls the access permissions to a specified physical memory region, by using a set of control status registers (CSR's).
- PMP entries are described by an 8-bit configuration register and one address register.
- Up to 16 PMP entries are supported.
- If any PMP entries are implemented, then all PMP CSRs must be implemented, but all PMP CSR fields are WARL and may be hardwired to zero.
- PMP CSRs are only accessible to M-mode.
- The PMP configuration registers are densely packed into CSRs to minimize context-switch time.

8.1. PMP configuration CSR

- For RV32, four CSRs, pmpcfg0–pmpcfg3, hold the configurations pmp0cfg–pmp15cfg for the 16 PMP entries, as shown in Figure

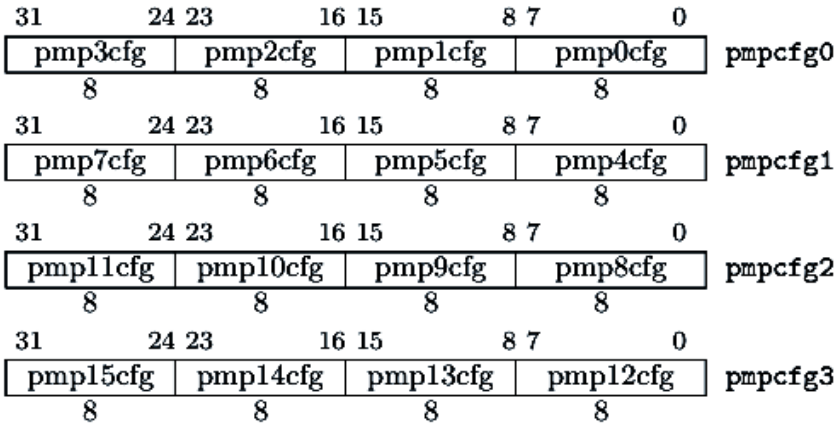


Figure 4 : RV32 PMP configuration CSR layout.

- For RV64, pmpcfg0 and pmpcfg2 hold the configurations for the 16 PMP entries, as shown in Figure;
- pmpcfg1 and pmpcfg3 are illegal for RV64..
- RV64 systems use pmpcfg2, rather than pmpcfg1, to hold configurations for PMP entries 8–15. This design reduces the cost of supporting multiple M-XLEN values, since the configurations for PMP entries 8–11 appear in pmpcfg2[31:0] for both RV32 and RV64.

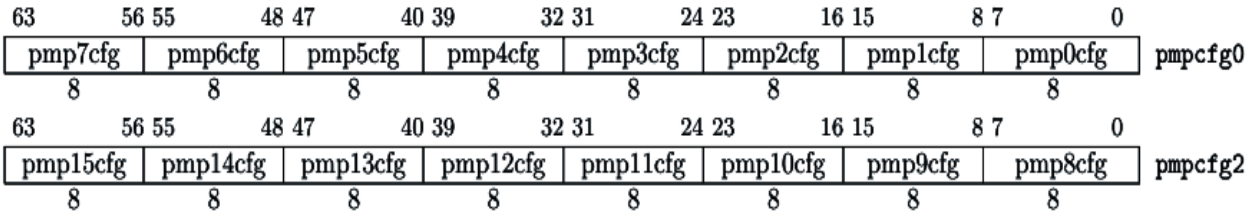


Figure 5: RV64 PMP configuration CSR layout.

8.2. PMP address registers

The PMP address registers are CSRs named pmpaddr0–pmpaddr15. Each PMP address register encodes bits 33–2 of a 34-bit physical address for RV32, as shown in Figure.

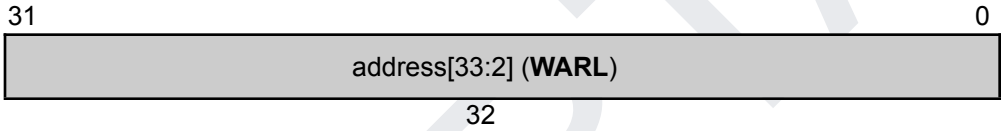


Figure 6 : PMP address register format, RV32.

For RV64, each PMP address register encodes bits 55–2 of a 56-bit physical address, as shown in Figure 3.26. Not all physical address bits may be implemented, and so the pmpaddr registers are WARL.

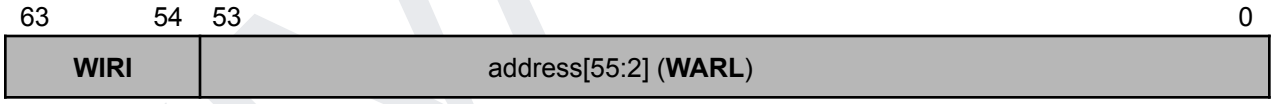


Figure 7 : PMP address register format, RV64

8.3. PMP configuration register

Figure shows the layout of a PMP configuration register. The R, W, and X bits, when set, indicate that the PMP entry permits read, write, and instruction execution, respectively. When one of these bits is clear, the corresponding access type is denied. The remaining two fields, A and L, are described in the following sections.

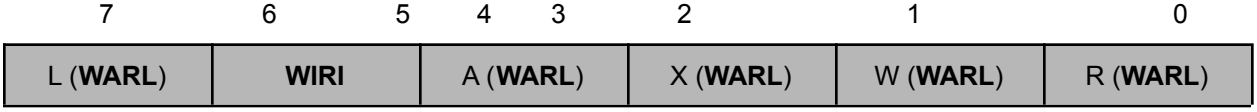


Figure 8 : PMP configuration register format

Field	Description	Access	Preset
R	R = 1, indicate that the PMP entry permits read R = 0, indicate that the PMP entry does not permit read	WARL	
W	W = 1, indicate that the PMP entry permits write W = 0, indicate that the PMP entry does not permit write	WARL	
X	X = 1, indicate that the PMP entry permits instruction execution X = 0, indicate that the PMP entry does not permit instruction execution	WARL	
A	Address-matching mode of the associated PMP address register. 00 : OFF 01 : TOR 10 : NA4 11 : NAPOT	WARL	
L	The L bit indicates that the PMP entry is locked.	WARL	0

Table 5 : PMP configuration register format bit fields

Note :The R, W, and X fields form a collective WARL field for which the combinations with R=0 and W=1 are reserved.

8.4. Combinations preferred for R/W/X

R	W	X	Comment
0	0	0	Inaccessible
0	0	1	Read and Write denied,execute permitted only for S/U mode
0	1	0	Reserved
0	1	1	
1	0	0	Execute and Write denied,Read permitted only for S/U

			mode
1	0	1	Write denied,Read and Execute permitted only for S/U mode
1	1	0	Execute denied,Read and Write permitted only for S/U mode
1	1	1	Read,write and Execute permitted for S/U mode

Table 6 : Read , Write , Execute combinations

9. Address Matching

The A field in a PMP entry's configuration register encodes the address-matching mode of the associated PMP address register. The encoding of this field is shown in Table 3.8.

- When A=0, this PMP entry is disabled and matches no addresses.
- 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 (disabled)
1	TOR	Top of range
2	NA4	Naturally aligned four-byte region
3	NAPOT	Naturally aligned power-of-two region, ≥ 8 bytes

Table 7 : Encoding of A field in PMP configuration registers.

9.1. Top Of Range (TOR)

- If TOR is selected, the associated address register forms the top of the address range, and the preceding PMP address register forms the bottom of the address range.
- If PMP entry i's A field is set to TOR, the entry matches any address a such that pmpaddri−1 ≤ a < pmpaddri .
- If PMP entry 0's A field is set to TOR, zero is used for the lower bound, and so it matches any address a < pmpaddr0 .

9.2. Encoding of region size using NAPOT

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 .

pmpaddr	pmpcfg.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 8 : NAPOT range encoding in PMP address and configuration registers.

- NAPOT ranges make use of the low-order bits of the associated address register to encode the size of the range.
- The NAPOT region size is 2^(zerobit position + 3) and the region base is the MSB above first zero.

10. Locking and Privilege Mode

- The L bit indicates that the PMP entry is locked, i.e., writes to the configuration register and associated address registers are ignored.
- Locked PMP entries may only be unlocked with a system reset.
- If PMP entry i is locked, writes to pmpicfg and pmpaddri are ignored.
- Additionally, if pmpicfg.A is set to TOR, writes to pmpaddri-1 are ignored.
- In addition to locking the PMP entry, the L bit indicates whether the R/W/X permissions are enforced on M-mode accesses.
- When the L bit is set, these permissions are enforced for all privilege modes.
- When the L bit is clear, any M-mode access matching the PMP entry will succeed; the R/W/X permissions apply only to S and U modes.

11. Regions of physical memory

PMP encoding supports regions as small as four bytes. Certain regions' privileges can be hardwired—for example, some regions might only ever be visible in machine mode but in no lower-privilege layers. Below figure shows the regions configuration in physical memory

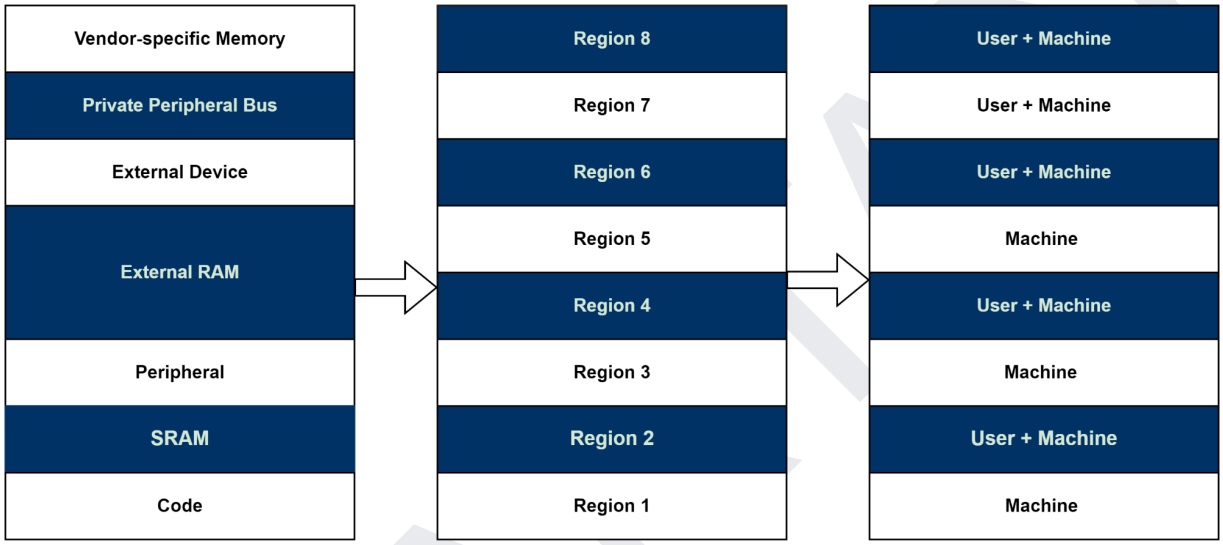


Figure 9 : Regions Configuration in Physical Memory

11.1. Region information

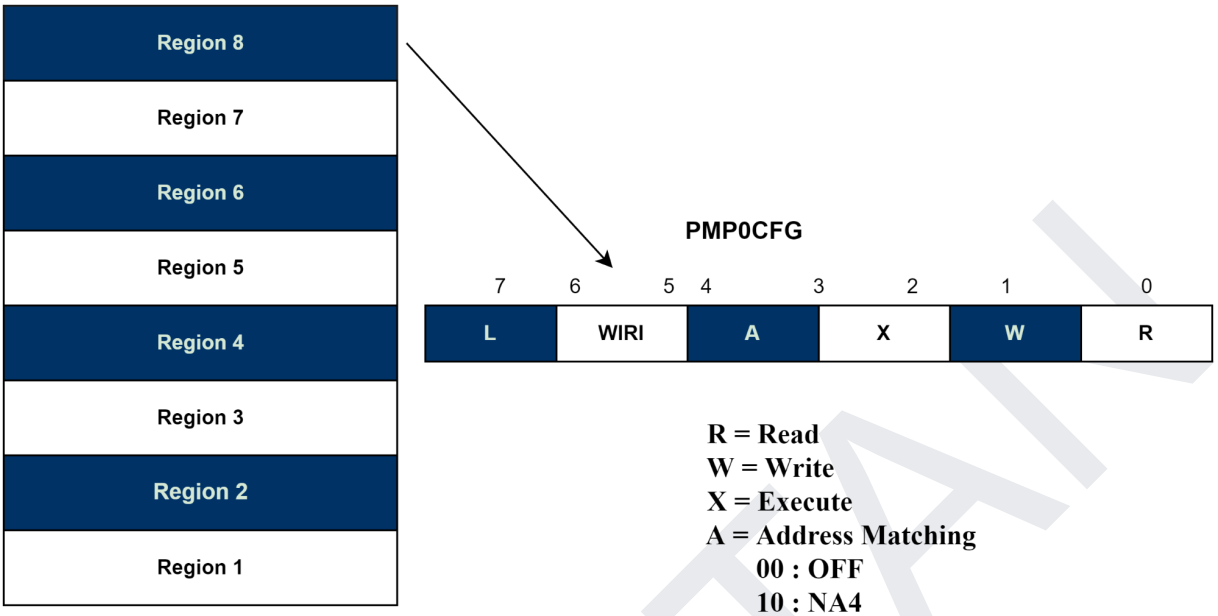
- Smallest region size is 4 Bytes.
- Maximum size of a region is 32 GB.
- Region granularity is configurable (2^n+2 Bytes).
- Hybrid privileged and unprivileged settings.
- Supported memory attributes are R/W/X .
- Maximum number of supported memory regions 16.

11.2. Region Access Using pmp configuration registers

Every region is configured by one distinct pmpicfg register.

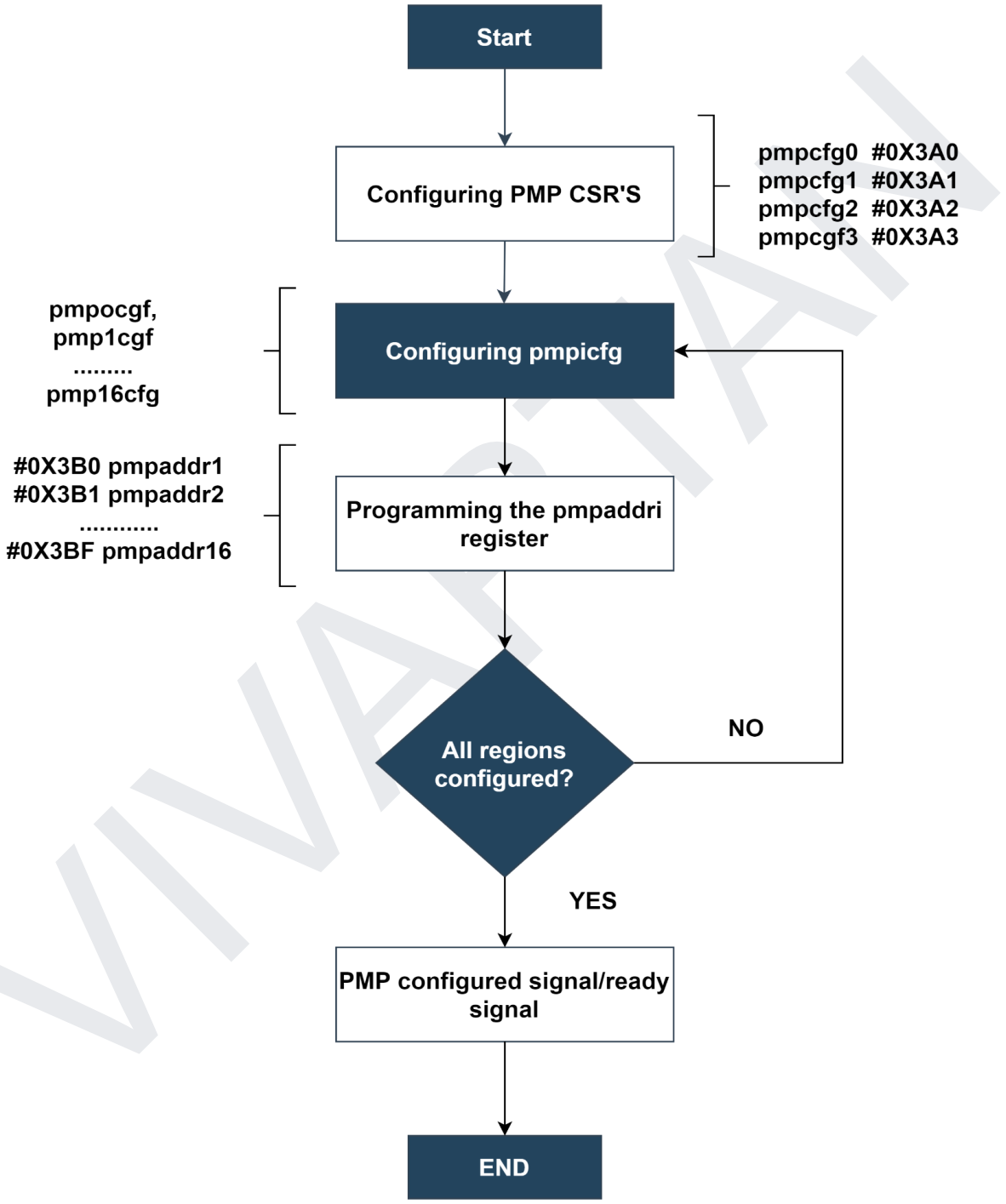
For every pmpicfg register there will be distinct pmpaddri associated with that register

From the below diagram we can say that 8 pmp

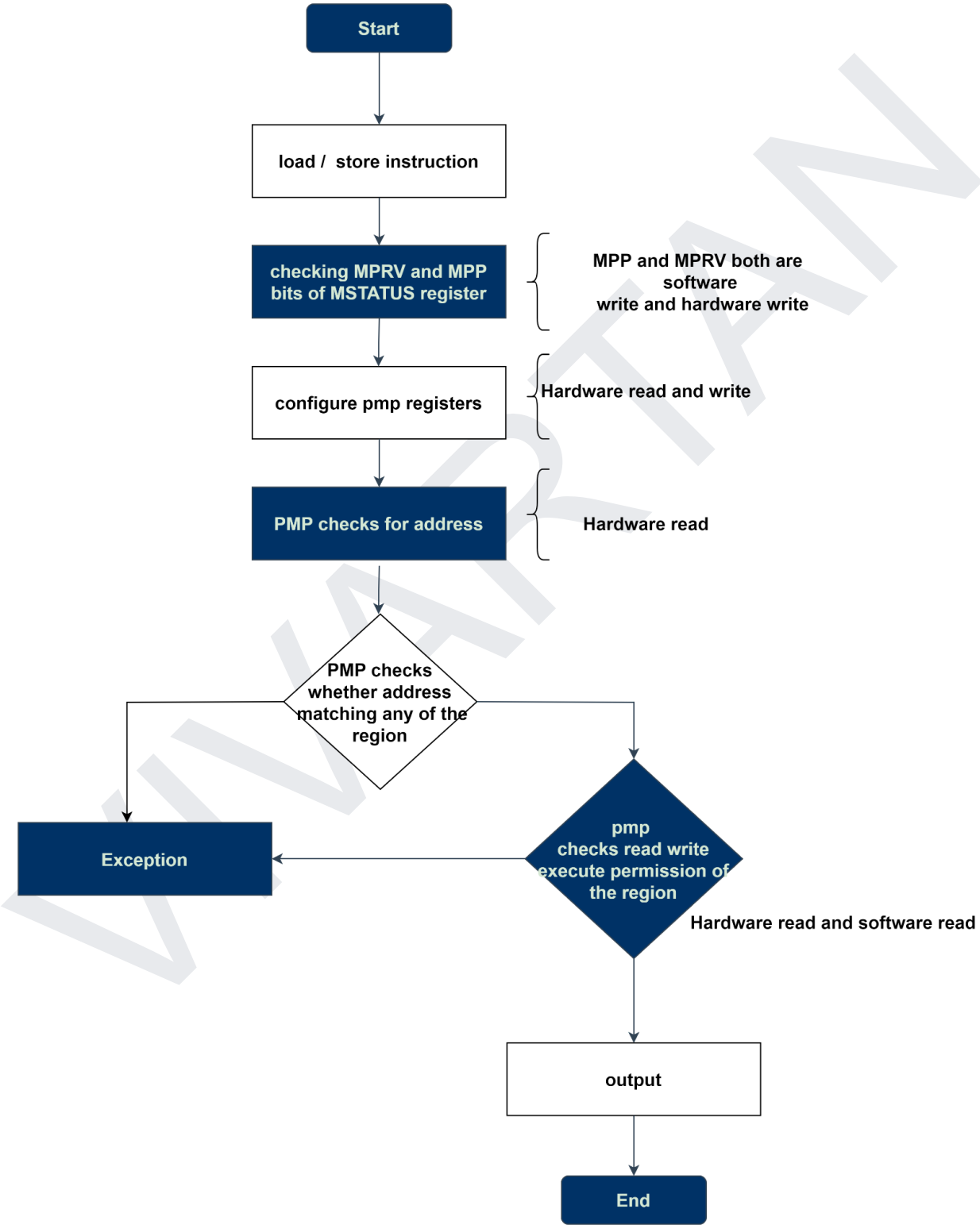


R = Read
W = Write
X = Execute
A = Address Matching
00 : OFF
10 : NA4
11 : NAPOT
01 : TOR
L : Locked region

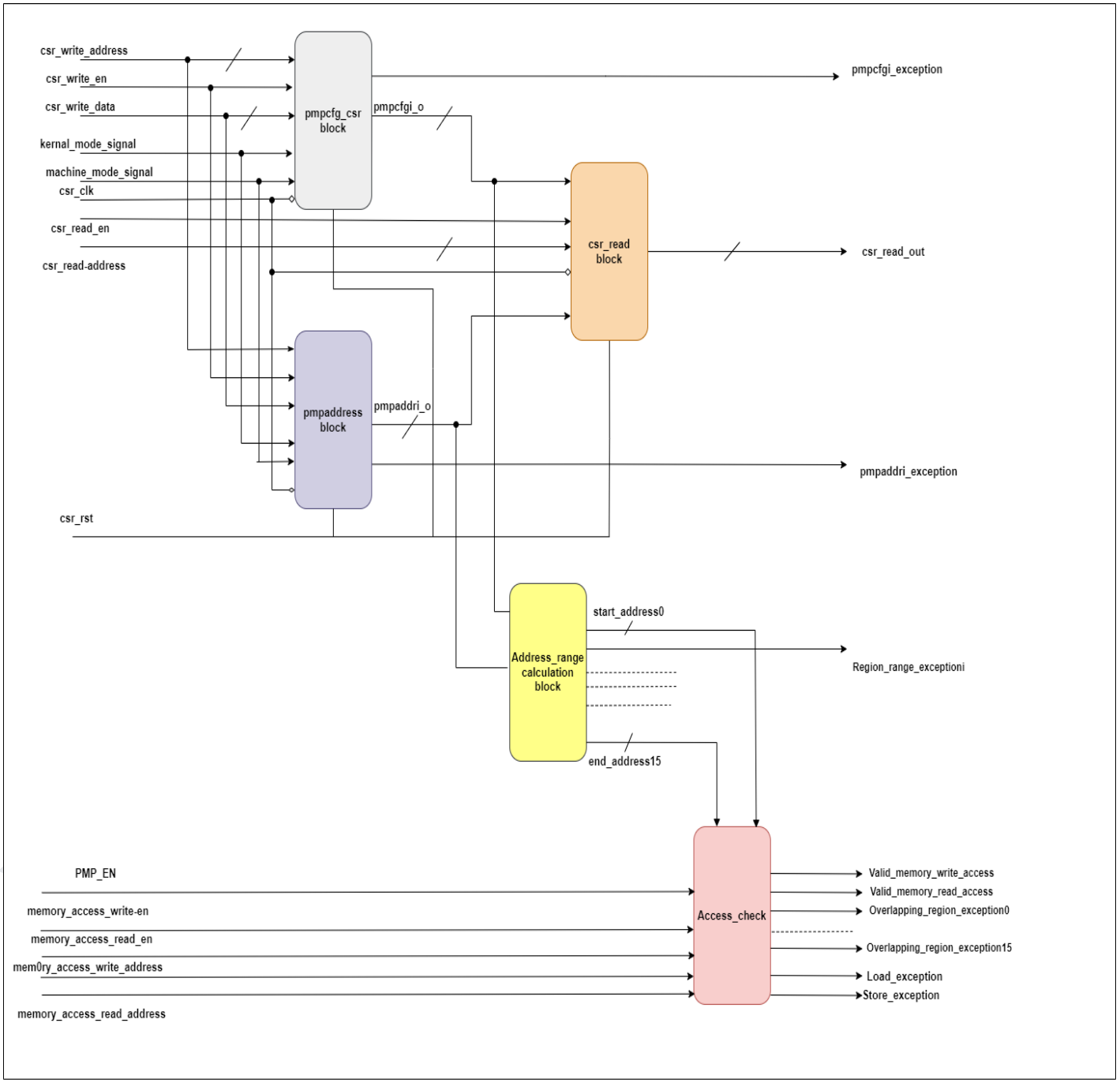
12. PMP Region Configuration Flowchart



13. PMP working flowchart



14.Implemented PMPCSR's module



- Here ,pmpaddri_o are multiple outputs where i = 0 to 15. That is pmpaddr1,pmpaddr2……..upto pmpaddr15
- Similarly pmpcfgi_o are multiple outputs where i =0 to 3. That is pmpcfg0,pmpcgf1,pmpcfg2 and pmpcfg3.

- Pmpcfgi_exception are multiple outputs where i = 0 to 3.
- Pmpaddri_exception are multiple outputs where i = 0 to 15.
- Kernel _mode_signal is signal from user trying to access pmp.
- machine _mode_signal is machine mode signal which is trying to access pmp.
- When valid_memory_write_access is high then write to the memory location is permitted.
- When valid_memory_read_access is high then read from the memory location is permitted

15. Verification Goals

NOTE :

- pmpcfgi csr registers few bits are hardwired to zero(refer spec)and we need to write the registers by keeping in mind . pmpaddri registers are 34 bits but last 2 lsb bits are hardwired to zero (as per spec) which we use it for calculations.
- Permitting access to **read and write for memory regions** has been implemented.
- **pmp_en** signal must be logic high after configuring regions.
- All the regions are configured for both machine and user mode.
- For TOR, Non contiguous memory allocation has been implemented. I.e when using TOR instead of the previous pmp address now we will get previous pmpaddr + 1.
- **Non Overlapping functionality has been added**
    - If you are not configuring the regions then dont configure pmpcfgi registers orelse overlapping may happen.
- Don't test **region_range_exception** for start_address = 1 and end_address = 0 when TOR is configured in pmpcfg and respective addresses are not assigned.
- Locked mode for the machine is not implemented.

Conditions to test pmp block (active low reset)

Conditions to write in any pmp register

1. csr_write_en signal should be high.
2. machine_mode_signal should be high.
3. kernal_mode_signal must be low.(as we are making it only hardware write for now).
4. csr_rst signal should be high.
5. csr_write_addr must be valid.

Conditions to read from pmp register(Involves complete combo logic)

1. csr_read_en should be high.
2. csr_read_addr should be valid.
3. kernal_mode_signal and machine_mode_signal can be high or low as csr
4. read does not depend on these signals because we made it software read.

Conditions where exception can be raised

1. when kernal_mode_signal is high and trying to write in pmpcsr registers.(we allow only hardware write)
2. When start_address > end_address region_range_exceptions will be ranged
3. When regions overlap will happen then there will be overlap exception

Test Cases

1. writing into all the registers by giving proper valid csr address
2. reading from each register by giving particular csr address
3. checked csr_read_en signal working properly or not.
4. checked whether csr write is happening when machine_mode_signal is high and kernal_mode_signal is low.
5. checked csr_write_en signal working correctly or not.
6. checked whether the pmpcfgi registers are hardwired as per spec
7. checked for the exception when an attempt of writing into csr is happening when kernal_mode_signal is high.