PTW Specifications

Spec version	**1.12 Privileged**
Date	**11/12/2023**
Author(s)	**Javier Salamero** **Xavier Carril**
Code version	
Modules/files involved	
ptw.sv **ptw_arb.sv** **pseudoLRU.sv** **mmu_pkg.sv**	

Change log

VERSION	**DESCRIPTION**
1.0	Specifications of the PTW release.
Related documentation	TLB doc

Design Intention

A Page Table Walker (PTW) is part of the memory management unit (MMU), responsible for translating virtual addresses generated by the CPU into physical addresses to access the main memory hierarchy system.

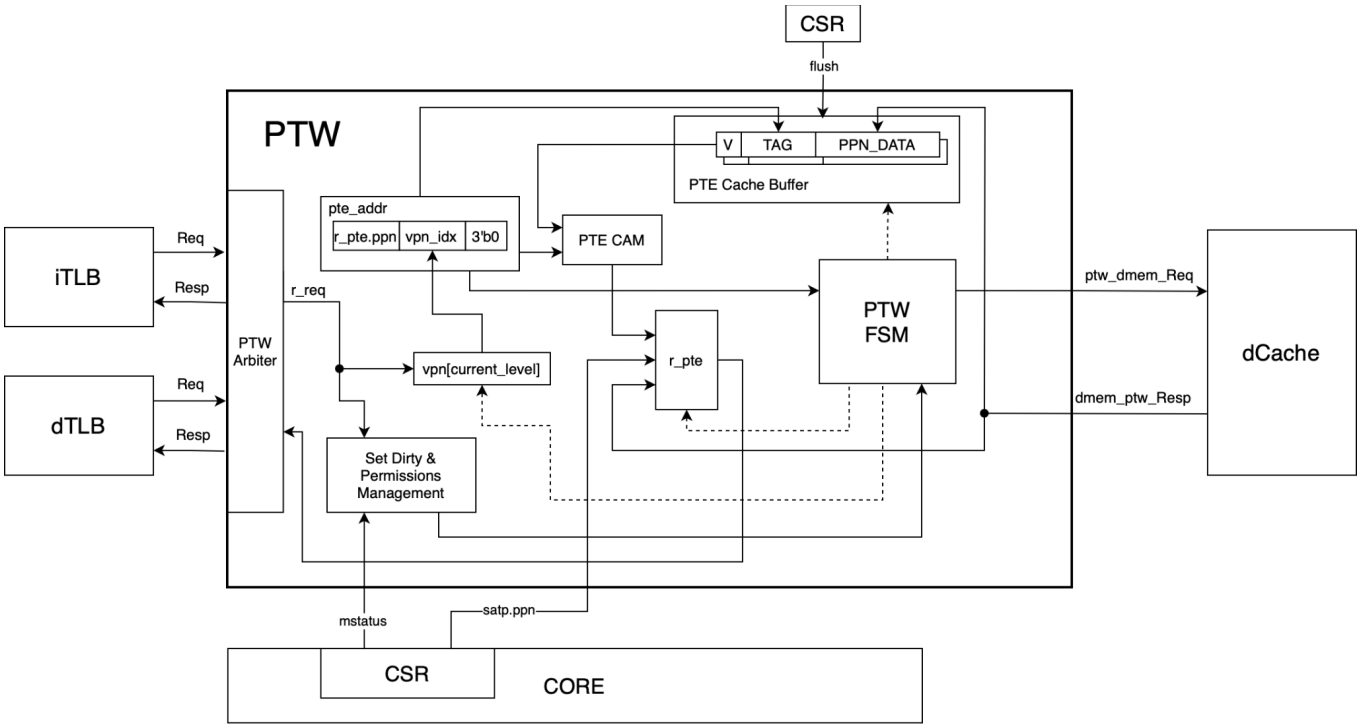
The Page Table Walker performs the translation by following a hierarchical structure called a page table. The page table organizes the mapping between virtual addresses and physical addresses. When the CPU core generates a virtual address, the Page Table Walker traverses the page table to locate the corresponding entry for that address. It follows a multi-level lookup process, where each level of the page table provides more specific details about the physical address corresponding to the virtual address.

The functionality of a Page Table Walker involves the following steps:

1. **Address Translation:** When the CPU core generates a virtual address, and after a TLB miss, the Page Table Walker starts the translation process by breaking down the virtual address into multiple parts (such as page number and offset).
2. **Page Table Lookup:** Using the hierarchical page table structure, the walker starts from the root of the page table (stored in *satp* register) and navigates through multiple levels, accessing intermediate page table entries (a.k.a PTE Table) to find the final mapping for the last page table entry with the virtual address (a.k.a PTE Leaf).
3. **The Page Table Entry Leaf (PTE Leaf):** The walker retrieves the corresponding PTE Leaf containing information like the physical page number (PPN), permissions bits (Read, Write, eXecute), status bits (Dirty, Access, Global mapping), and privilege status bit (User).
4. **Memory Access:** Finally, the walker provides the PTE Leaf with the actual physical address to the correspondent TLB, allowing the CPU to access the required memory location in dCache/Main Memory.

Our current PTW design supports **SV39** (Virtual addresses of 39 bits), with **3 Levels** of hierarchy. So, following the RISC-V privileged specs for SV39, the **VPN** has a size of 27 bits, and the **PPN** has a size of 44 bits. At the same time, the PTW supports Giga (1GB), Mega (2MB), and Kilo (4KB) pages. PTE Leafs received at Level 0 gives a Giga-page, at Level 1 a Mega-page, and at Level 2 a Kilo-Page.

Module Functionality



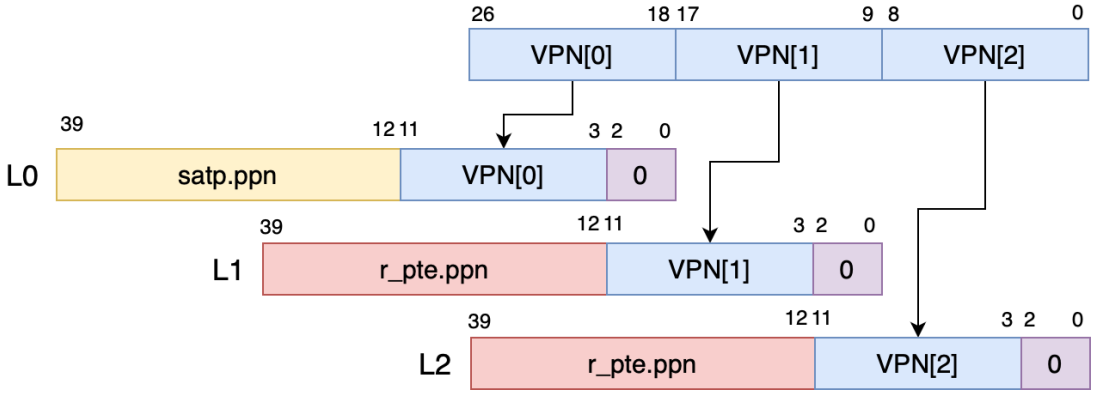
This PTW follows the RISC-V Instruction Set Manual version 1.12.

The PTW includes an arbiter responsible for selecting a request when both the iTLB and dTLB send requests simultaneously. Priority is always given to dTLB requests over iTLB requests.

The PTW receives a Virtual Page Number (VPN) from the TLB, which is segmented into three parts, each corresponding to one of the three hierarchy levels. The FSM (Finite State Machine) detailed in the subsequent Flow Diagram section manages the traversal through these levels.

The formation of the address for each Page Table Entry (PTE) at every level (pte_addr) is as follows:

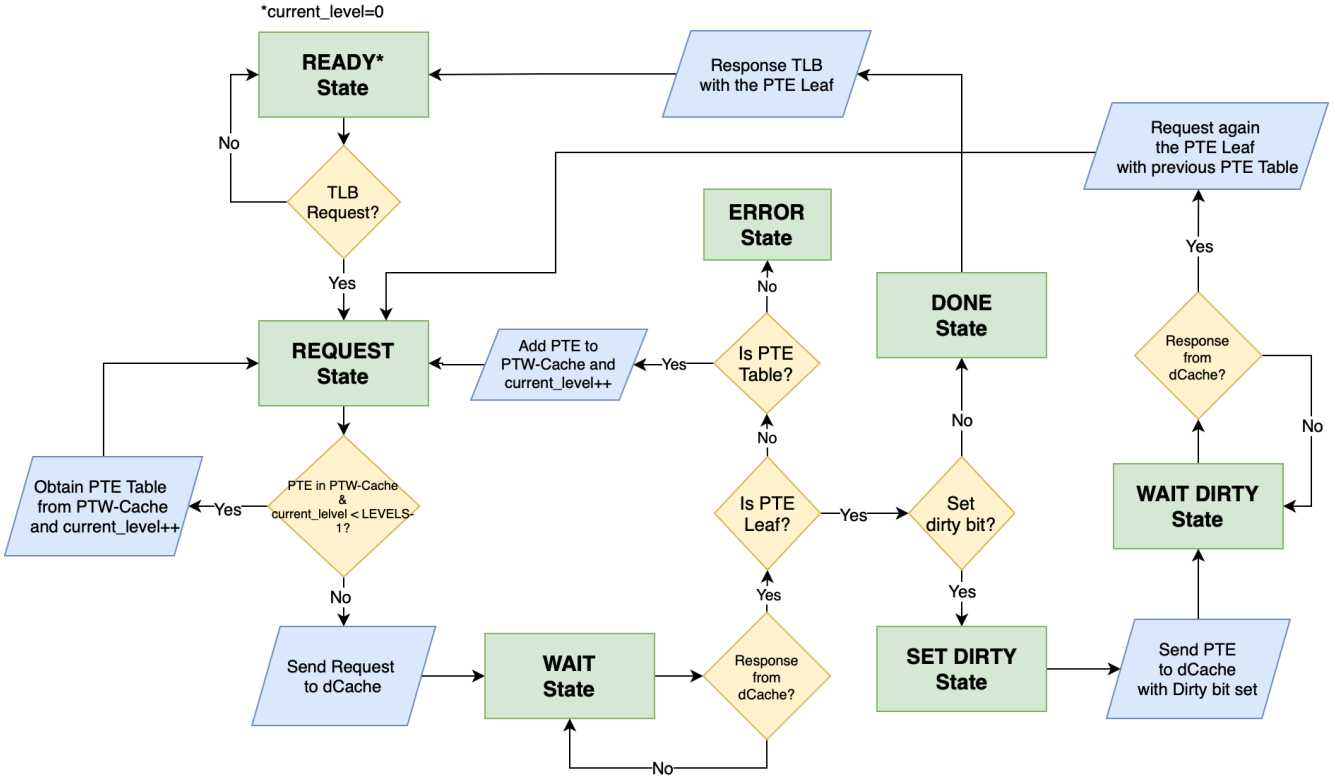
- For the initial level (L0), the PPN (Physical Page Number) field from the satp register sets the root of the main page table, with VPN[0] serving as an offset. Subsequently, the address is aligned to 8 bits before being sent to memory. The resulting address fetches the PTE of the subsequent level, stored in r_pte. If this PTE is a leaf, we stop the walk and we obtain a Giga-Page.
- Upon reaching L1, and having a PTE Table, a request is made for the next level by combining the PPN of the current PTE (stored in r_pte) with VPN[1]. If the giving PTE is a leaf, we stop the walk and we obtain a Mega-Page.
- The final PTE level (L2) must be a leaf node, representing a Kilo-Page. If it isn’t, the PTW triggers an error signal to the TLB.



Once getting the PTE leaf, if all the privilege permission passes (comparing mstatus SUM and MXR bits with the request operation, fetch or store, and PTE bits), the PTE leaf proceeds to the Set Dirty Management. If the A bit (access) is not set, or the request is a store and the D bit (dirty) is not set, the FSM proceeds to write to memory setting the D and A bits. Once all this management logic concludes, the PTE, containing the corresponding Physical Page Number (PPN), is forwarded to the respective TLB.

In addition, to optimize the walking steps, the PTW contains a PTE Cache buffer. This buffer retains each received PTE Table from memory, using a pseudoLRU replacement strategy. Before initiating a request for a PTE from memory, the PTW checks this buffer cache using a Content-Addressable Memory (CAM). In the case where the TAG comparison registers a hit, the PTW can swiftly retrieve the PTE Table's PPN without the need for a memory request.

FSM Flow Diagram



Interface Description

To see the struct definitions, please read the next section.

SIGNAL NAME	TYPE	WIDTH	DIRECTION	DESCRIPTION
clk_i	logic	1	Clock > PTW	System operational clock.
rstn_i	logic	1	Reset > PTW	System reset signal (active low).
iTLB request-response				
itlb_ptw_comm_i	tlb_ptw_comm_t		iTLB > PTW	Communication from iTLB to PTW for requests.
ptw_itlb_comm_o	ptw_tlb_comm_t		iTLB < PTW	Communication from PTW to iTLB for responses.
Coverage Notes				
itlb_ptw_comm_i.req.valid can only be set to 1 if and only if ***ptw_itlb_comm_o.ptw_ready*** is 1. ***itlb_ptw_comm_i.req.fetch*** and ***itlb_ptw_comm_i.req.store*** should be mutually exclusive. If one of them is set to 1, the other cannot be also 1. However, both signals can be 0 to indicate the access is a load. ***ptw_itlb_comm_o.resp.error*** has relevance only and only if ***ptw_itlb_comm_o.resp.valid*** is set to 1. ***ptw_itlb_comm_o.ptw_status*** and ***ptw_itlb_comm_o.invalidate_tlb*** are functionally independent of the ***ptw_itlb_comm_o.resp*** values.				
dTLB request-response				
dtlb_ptw_comm_i	tlb_ptw_comm_t		dTLB > PTW	Communication from dTLB to PTW for requests.
ptw_dtlb_comm_o	ptw_tlb_comm_t		dTLB < PTW	Communication from PTW to dTLB for responses.
Coverage Notes				
dtlb_ptw_comm_i.req.valid can only be set to 1 if and only if ***ptw_dtlb_comm_o.ptw_ready*** is 1. ***dtlb_ptw_comm_i.req.fetch*** and ***dtlb_ptw_comm_i.req.store*** should be mutually exclusive. If one of them is set to 1, the other cannot be also 1. However, both signals can be 0 to indicate the access is a load. ***ptw_dtlb_comm_o.resp.error*** has relevance only and only if ***ptw_dtlb_comm_o.resp.valid*** is set to 1. ***ptw_dtlb_comm_o.ptw_status*** and ***ptw_dtlb_comm_o.invalidate_tlb*** are functionally independent of the ***ptw_dtlb_comm_o.resp*** values.				
dMem request-response				
dmem_ptw_comm_i	dmem_ptw_comm_t		dMem > PTW	Communication from dMem to PTW for requests.
ptw_dmem_comm_o	ptw_dmem_comm_t		dMem < PTW	Communication from PTW to dMem for responses.
Coverage Notes				
ptw_dmem_comm_o.req.valid can be set to 1 and ***dmem_ptw_comm_i.dmem_ready*** be 0. ***ptw_dmem_comm_o.req.valid*** will pull down to 0 once ***dmem_ptw_comm_i.dmem_ready*** sets to 1. ***ptw_dmem_comm_o.req.typ*** is always 4'b0011 (MT_D), being the block size request (64 bits). ***ptw_dmem_comm_o.req.cmd*** need to be an Atomic Memory Operation OR (5'b01010), in order to write the PTE on ***ptw_dmem_comm_o.req.data*** The PTW module only uses the following inputs from dMem: ***dmem_ptw_comm_i.dmem_ready***, ***dmem_ptw_comm_i.resp.data***, ***dmem_ptw_comm_i.resp.valid*** and ***dmem_ptw_comm_i.resp.nack***				

CSR interface				
csr_ptw_comm_i	csr_ptw_comm_t		CSR > PMU	Input values of satp, flush and mstatus config. registers.
Coverage Notes				
No notes.				
PMU counter events				
pmu_ptw_hit_o	logic	1	PTW > PMU	Hit on PTE Cache Buffer
pmu_ptw_miss_o	logic	1	PTW > PMU	Miss on PTE Cache Buffer
Coverage Notes				
These signals are raised up each time there is a hit on the PTE Cache Buffer (corresponding to ***pmu_ptw_hit_o***) and a miss (corresponding to ***pmu_ptw_miss_o***). No valid signals are needed. At the same time, ***pmu_ptw_hit_o*** and ***pmu_ptw_miss_o*** should be mutually exclusive. If we have a hit, we cannot have a miss, and vice versa. However, both signals can be 0 to indicate there is no access.				

Struct definitions

pte_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DESCRIPTION**
ppn	logic	PPN_SIZE	Physical Page Number.
rfs	logic	2	Reserved field for supervisor software (RSW).
d	logic	1	Dirty Bit.
a	logic	1	Access Bit.
g	logic	1	Global Mapping Bit.
u	logic	1	User Page bit.
x	logic	1	Executable Page.
w	logic	1	Writable Page.
r	logic	1	Readable Page.
v	logic	1	Indicates valid PTE.

tlb_ptw_req_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DESCRIPTION**
valid	logic	1	Translation request valid.
vpn	logic	VPN_SIZE	Virtual page number.
asid	logic	ASID_SIZE	Address space identifier.
prv	logic	2	Privilege level of the translation: 2'b00 (User), 2'b01 (Supervisor), 2'b11 (Machine).
store	logic	1	Store operation.
fetch	logic	1	Fetch operation.

tlb_ptw_comm_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DESCRIPTION**
req	tlb_ptw_req_t	1	Translation request of the TLB to the PTW.

ptw_tlb_resp_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DESCRIPTION**
valid	logic	1	Translation response valid.
error	logic	1	An error has ocurred with the translation request. Only check if the response is valid.

pte	pte_t	1	Page table entry.
level	logic	2	Page entry size: 2'b00 (1 GiB Page), 2'b01 (2 MiB Page), 2'b10 (4 KiB Page).

ptw_tlb_comm_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DESCRIPTION**
resp	ptw_tlb_resp_t	1	PTW response to TLB translation request.
ptw_ready	logic	1	PTW is ready to receive a translation request.
ptw_status	csr_mstatus_t	1	mstatus csr register value, sent through the ptw.
invalidate_tlb	logic	1	Signal to flush all entries in TLB and don't allocate in-progress transactions with the PTW.

ptw_dmem_req_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DESCRIPTION**
valid	logic	1	Request valid.
addr	logic	SIZE_VADDR +1	The address extracted from pte_addr to the memory hierarchy. +1 bit extra is for sign bit extension.
cmd	logic	5	Operation memory commands: 5’b0 (Load), 5’b01010 (AMO_OR)
typ	logic	4	Bit-packet Size: 4’b0011 (64 bits)
data	logic	64	PTE to write

ptw_dmem_comm_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DESCRIPTION**
req	ptw_dmem_req_t		PTW request of the to the memory hierarchy.

dmem_ptw_resp_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DESCRIPTION**
valid	logic	1	Request valid.
data	logic	64	Incoming PTE from memory.
nack	logic	8	Invalid acknowledgment.

dmem_ptw_comm_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DESCRIPTION**
dmem_ready	logic	1	dMem is ready to receive a new request.
resp	ptw_dmem_resp_t		PTW response from the to the memory hierarchy.

csr_ptw_comm_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DESCRIPTION**
satp	logic	64	satp CSR value
flush	logic	1	Flush the PTW Cache Buffer
mstatus	csr_mstatus_t		mstatus CSR value