TLB Specifications

Spec version	**1.12 Privileged**
Date	**11/12/2023**
Author(s)	**Javier Salamero** **Xavier Carril**
Modules/files involved	**tlb.sv** **tlb_wrapper.sv** **psuedoLRU.sv**

CHANGE LOG

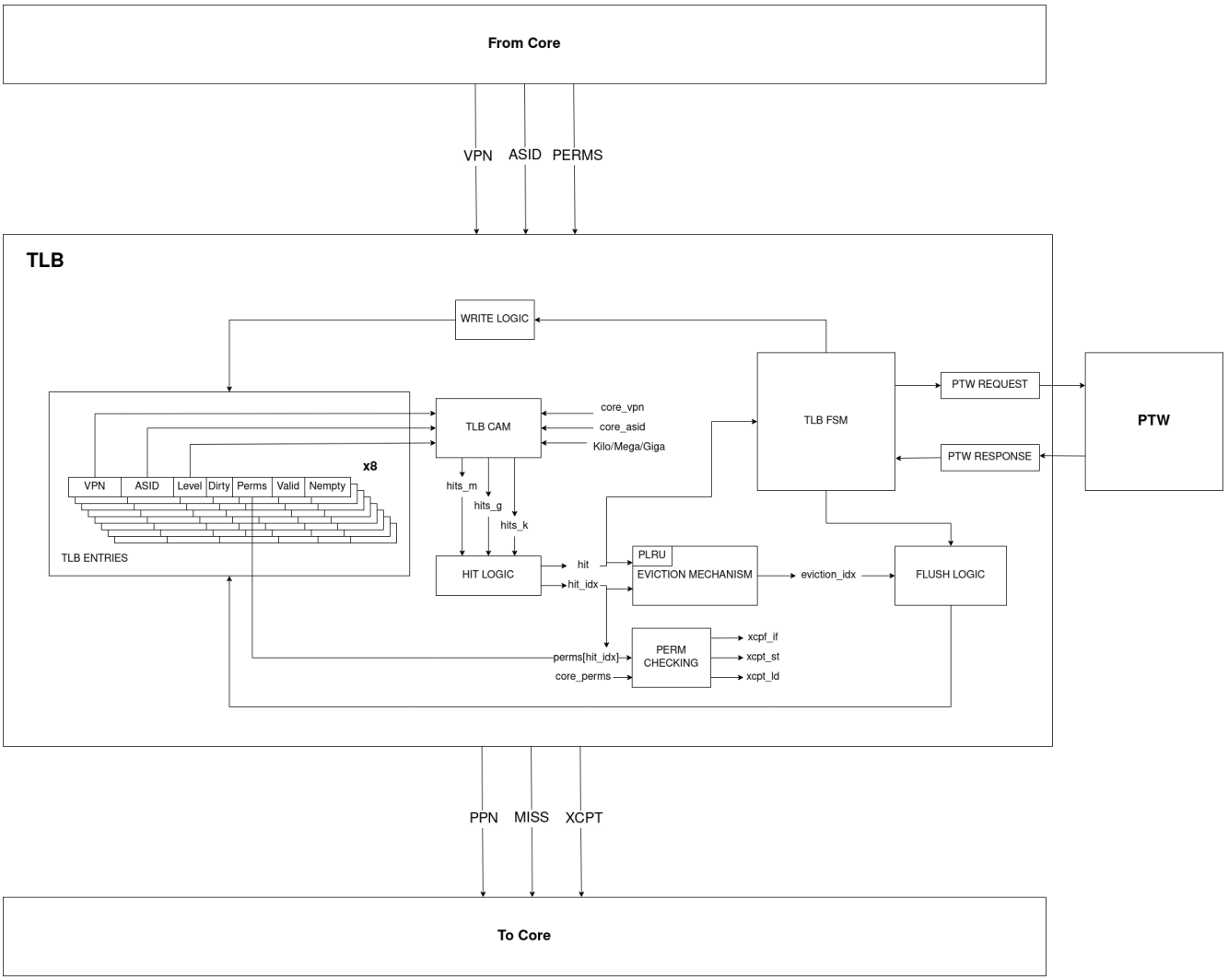
VERSION	**DESCRIPTION**
1.0	Specifications of the TLB release.

Design Intention

The Translation Lookaside Buffer (TLB) serves as a cache for virtual-to-physical address translations, a key function in managing virtual memory. By storing recently used mappings, the TLB accelerates the translation process, minimizing the time required to access data in RAM. Acting as a bridge between virtual and physical addresses, it helps circumvent the need for a complete lookup in page tables for each memory access. TLB hits, where the translation is found in the TLB, result in faster access to physical addresses. In the event of a TLB miss, requiring a full translation, the result is stored for future use, maintaining efficiency.

In addition, this TLB also checks that the access rights to a page, raising a page fault in the case of a memory access violating them.

Module Functionality



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

Every time there is an access to a memory page, the TLB CAM is consulted. If the virtual to physical memory translation was requested recently in the past, there is a good chance that it is still cached in the TLB. If this is the case, it is a TLB hit, and the permissions of the access are checked, raising an exception along with the TLB hit.

In the case of a translation request not being stored in the TLB, we have a TLB miss, and it is requested to the page-table walker (PTW). The TLB blocks until the response is obtained, and the requested information should be kept stable until this response arrives. When the page-table walkthrough finalizes, the page-table entry (PTE) is stored in the TLB, using a pseudo-least recently used (PLRU) replacement algorithm. In the next cycle, the request hits and after checking the access permissions, it is sent to the requestor. The flush logic and eviction mechanisms ensure that empty and/or invalid PTEs that could be stored in the TLB are prioritized as the next slot to be written. The process of requesting a PTE to the PTW is implemented with a finite-state machine.

Interface Description

This module's interface relies upon the usage of structs. The fields of each struct are disclosed in the next chapter.

SIGNAL NAME	TYPE	WIDTH	DIRECTION	DESCRIPTION
clk_i	logic	1	Clock > TLB	System operational clock.
rstn_i	logic	1		System reset signal (active low).
TLB request-response				
cache_tlb_comm_i	cache_tlb_comm_t	1	Translation Requester > TLB	Communication from translation requester to TLB.
tlb_cache_comm_o	tlb_cache_comm_t	1	TLB > Translation Requester	Communication from TLB to translation requester.
Coverage Notes				
A TLB request can only be initiated if TLB is ready, ***cache_tlb_comm_i.req.valid*** of a new request can only be set to 1 if and only ***tlb_cache_comm_o.tlb_ready*** is 1. Since the TLB is blocking, once a request is sent, its information cannot be withdrawn until the page is received. In other words, keep ***cache_tlb_comm_i.req.valid***=1 and the rest of the request information until the response has ***cache_tlb_comm_i.resp.miss=0***. If it is a TLB hit, this will be given combinationally. ***cache_tlb_comm_i.req.instruction*** and ***cache_tlb_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. ***tlb_cache_comm_o.resp.xcpt.load***, ***tlb_cache_comm_o.resp.xcpt.store*** and ***tlb_cache_comm_o.resp.xcpt.fetch*** are **NOT** mutually exclusive. These signals only check there are enough permissions to execute a load, store or fetch respectively. Therefore, a translation request to the instruction TLB can set ***tlb_cache_comm_o.resp.xcpt.store*** to 1 but should be ignored. Only check the exception that can be raised by the access. A write translation access that hits in the tlb but does not have the dirty bit set behaves as a tlb miss, requesting to the PTW a write translation that sets the dirty bit in the page table and returns the PTE, now with the dirty bit correctly set.				
PTW request-response				
ptw_tlb_comm_i			TLB < PTW	Communication from TLB to PTW.
tlb_ptw_comm_o	ctrl	2	TLB > PTW	Communication from to PTW to TLB.
Coverage Notes				
tlb_ptw_comm_o.req.fetch and ***tlb_ptw_comm_o.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.				
PMU counter events				
pmu_tlb_access_o	logic	1	TLB > PMU	Accepted Translation request to TLB.
pmu_tlb_miss_o	logic	1	TLB > PMU	Accepted Translation request to TLB misses.
Coverage Notes				
These signals can only be 1 when the translation request has been accepted. This is when ***cache_tlb_comm_i.req.valid*** and ***tlb_cache_comm_o.tlb_ready*** are 1s.				

Struct definitions

cache_tlb_comm_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DIRECTION**
req	cache_tlb_req_t	1	Translation request.
priv_lvl	logic	1	Privilege level of the translation: 2'b00 (User), 2'b01 (Supervisor), 2'b11 (Machine).
vm_enable	logic	1	Memory virtualization is active.

cache_tlb_req_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DIRECTION**
valid	logic	1	Translation request.
asid	logic	ASID_SIZE	Privilege level of the translation: 2'b00 (User), 2'b01 (Supervisor), 2'b11 (Machine).
vpn	logic	VPN_SIZE	Memory virtualization is active.
passthrough	logic	1	Virtual address directly corresponds to physical address, for direct assignment between a virtual machine and the physical device.
instruction	logic	1	The translation request is for a instruction fetch address.
store	logic	1	The translation request is for a store address.

tlb_ex_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DIRECTION**
load	logic	1	Load operation.
store	logic	1	Store operation.
fetch	logic	1	Fetch operation.

tlb_cache_comm_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DIRECTION**
tlb_ready	logic	1	The tlb is ready to accept a translation request. If 0 it shouldn't receive any translation request.
resp	tlb_cache_resp_t	1	Translation response.

tlb_cache_resp_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DIRECTION**

miss	cache_tlb_req_t	1	If the translation request missed set to 1. Otherwise, the rest of the signals have valid information of the response.
ppn	logic	1	Physical page number.
xcpt	tlb_ex_t	1	Exceptions produced by the requests
hit_idx	logic	TLB_IDX_SIZE	CAM hit index of the translation request.

tlb_cache_comm_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DIRECTION**
tlb_ready	logic	1	The tlb is ready to accept a translation request. If 0, it shouldn't receive any translation requests.
resp	tlb_cache_resp_t	1	Translation response.

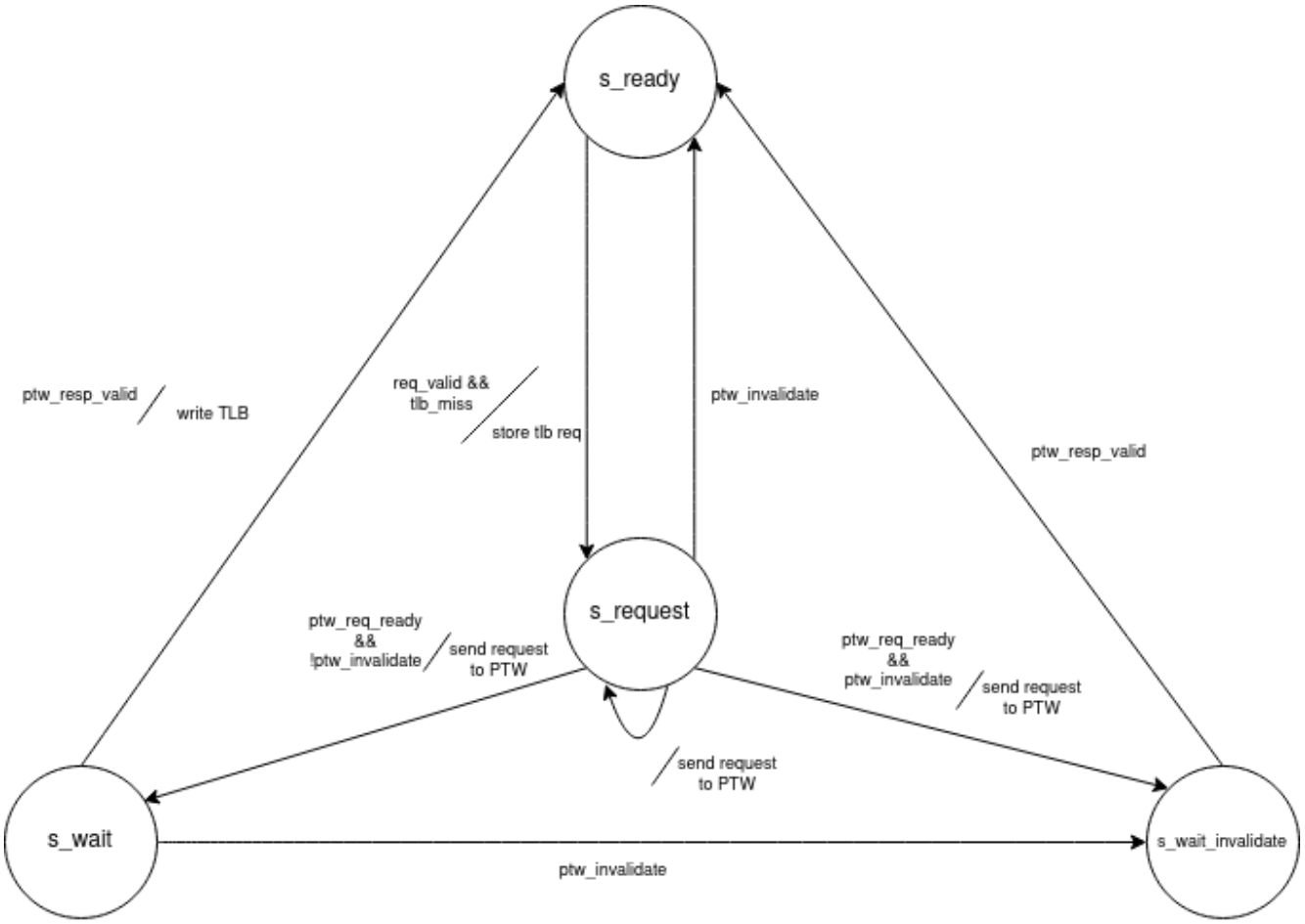
tlb_ptw_req_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DIRECTION**
valid	logic	1	Translation request valid.
vpn	logic	VPN_SIZE	Virtual page number.
asid	logic	ASID_SIZE	Address space identifier.
prv	logic	2	Privilege level of the translation: 2'b00 (User), 2'b01 (Supervisor), 2'b11 (Machine).
store	logic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

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

ptw_tlb_resp_t			
SIGNAL NAME	**TYPE**	**WIDTH**	**DIRECTION**
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**	**DIRECTION**
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.

TLB Finite-State Machine



This finite-state machine manages the communication with the PTW after a TLB miss. After reset, the initial state is *s_ready.*

s_ready: The TLB is ready to service any new requests, it will only change state if there is a TLB miss. If this happens, the request is stored, and it transitions to *s_request.*

s_request: It requests the saved address-translation request to the PTW and will keep requesting until the PTW accepts it. If the TLB gets invalidated in the cycle when the PTW accepts the petition, it transitions to *s_wait_invalidate.* Otherwise, transitions to *s_wait.*

s_wait: In this state, the TLB waits for the PTW's response. When it arrives, it writes its content in the position indicated by the eviction mechanism and the transaction finishes, moving to *s_ready.* If an invalidation happens while waiting, the FSM moves to the *s_wait_invalidate.* If a response happens at the same time as the invalidation, the transaction is finished and moves to *s_ready,* since the invalidation is prioritized over the tlb write.

s_wait_invalidate: In this state, it is known that the response it is waiting for will be invalidated. But it still needs to collect it before attending to any new requests to avoid any possible aliasing between requests and responses to PTW.