

TLB for 64-bit processor

# rfTLB

Robert Finch

---

## Goals / Requirements

- Parameterized
- Hardware or Software managed
- Eight-byte PTEs
- 64-bit virtual address space
- 48 (or more) bit physical address space
- LRU or random replacement policy (parameter)
- 4-way associative (parameter)
- 8kB / 8MB / 8GB page sizes
- Low resource usage

### Hardware or Software Managed

Many systems frugal with resources will use a software managed TLB, while high performance systems will use a hardware managed TLB. A software managed TLB requires an interface for a control processor (CPU). A hardware managed TLB requires an interface to the page table walker (PTW).

### Eight-byte PTEs

PTEs must fit evenly into a page which is a power-of-two in size. This is necessary otherwise computing the address of the PTE for a corresponding virtual address becomes very challenging.

The PTE must support at least at 48-bit physical address space. That means the PTE requires at least 35 bits just to represent the address.

### 64-bit Virtual Address Space

Address space requirements continue to grow. 64-bit machines require a large address space.

### 48-bit Physical Address Space

### LRU Replacement Policy

LRU (least recently used) replacement policy replacement the least recently used TLB entry with a new mapping. This is one of the better options for simplicity and performance.

8kB / 8MB / 8GB page sizes

## Design

1024 entries fit into an 8kB page, therefore 10 address bits from the virtual address are absorbed for each page table level. The size of a page is set by the LOG\_PAGESIZE parameter which specifies the number of address bits required for a page.

Translations are disabled while the TLB is updated. This allows the use of single ported memory to access the TLB.

The core has a WISHBONE bus interface to access the TLB programmatically.

Since a TLB entry is 128-bits and the bus size is only 64-bits, the TLB entries are accessible in an indirect fashion so that the entire TLB entry may be updated or read in the TLB as a single unit. The TLB entry is set up in a holding register, the entry number and way to update are specified in another register. If bit 31 is set while writing to the entry number register the TLB entry in the holding register is transferred to the TLB.

To read a TLB entry specify the entry number in register 28h with bit 30 set then read the value from the TLBE registers.

Note that for LRU operation the highest way should be used to update an entry in the TLB. LRU operates by shifting data from next higher way into the current way. Data in the lowest way is lost. The second highest way may be used if it is desired to keep the translations in the highest way.

## Operation

### Reset

On reset TLB entries are set up to be able to access the system boot ROM / RAM and I/O. This initialization is done in the `tlb_reset_machine.sv` file. The last 64-pages of virtual memory are set to map 1:1 to the last 64 pages of physical memory. This file may need to be modified to suit the system.

Once `rst_busy` is no longer active, addresses may be translated.

## Registers

Register		Description
00h	TLBE[63:0]	Low order 64-bits of TLBE (PTE)
08h	TLBE[127:64]	High order 64-bits of TLBE (VPN, ASID, COUNT)
10h		reserved
18h		reserved
20h	ENTRY_NO	TLB entry number and way to update. Bit 0 to 15 specify the entry number Bit 16 to 23 specify the way. Bit 31 indicates to update the TLB with the TLBE Bit 30 indicates to read the TLBE from the TLB
28h	LOCK	Bitmap of locked entries.

### Entry\_no

Specifying the way is useful only for reading the TLB entry. The way is automatically chosen for updates. The way chosen will either be one at random for a random update policy or the highest way for an LRU policy.

### Lock

Locking a range of TLB entries applies only to the highest associated way. Entries in other ways are not affected by a lock. For instance, if the TLB is five-way associative then only the entries in the fifth way may be locked.

If LRU operation is selected, entries shift down from the highest way to the lowest. Adding an entry will automatically place it in the highest way. To add an entry, the entry must first be unlocked; however, then relocked immediately after the update.

Each bit of the bitmap identifies a range of  $1/64^{\text{th}}$  of the TLB entries. For a TLB with 512 entries and five ways each bit represents a group of eight entries. Bit zero would represent entries 0 to 7 in the fifth way. If there are fewer than 64 entries in the TLB, then each bit of the lock map represents one entry. There may be unused bits in the lock map in that case.

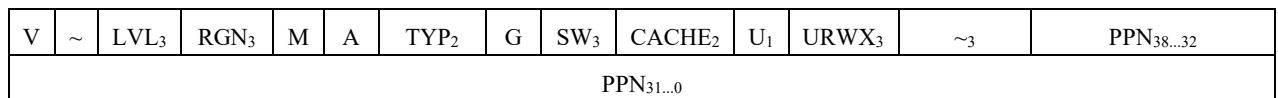
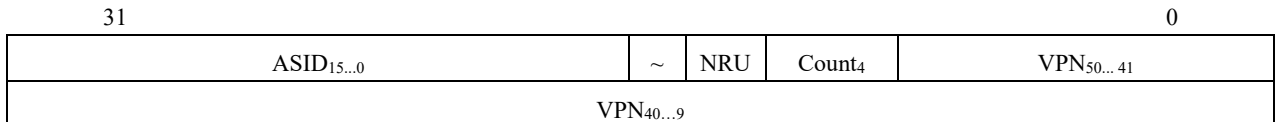
If an LRU update takes place to an entry that is locked, the next lower way is updated. The locked entry remains locked.

If a random update takes place to an entry that is locked, the way is altered to a non-locked way (the LSB of the way selected is flipped).



## TLB Entry Format

The low order 64-bits of the TLBE are a copy of the PTE. The high order 64-bits are used for matching the virtual address with the physical address.



Field	Size	Purpose
PPN	39	Physical page number
URWX	3	User read-write-execute
U	1	1=user page
CACHE	2	Cache location bits
SW	3	OS software usage
G	1	1=global translation
TYP	2	0 = PTE, 1 = PTP, 2 = Shortcut PTP
A	1	1=accessed/used
M	1	1=modified
RGN	3	Memory region select
LVL	3	0 to 7 indicates level of page table
~	1	Reserved bit
V	1	1 if entry is valid, otherwise 0

Field	Size	Purpose
VPN	42	Virtual page number
Count	4	Invalidation count
NRU	1	Not recently used indicator
~	1	Reserved bit
ASID	16	Address Space Identifier

## PPN

The physical page number of 39 bits is sufficient to specify a 52-bit physical address. The low order 13 bits of the address pass through the page table walker unaltered.

For a shortcut page the low order 23 bits of the address pass through unaltered. The PPN specifies the shortcut page number (an 8MB page number). The high order 10 bits of the PPN are not used and should be zero.

## URWX

User Read, Write and eXecute attributes for the page. These attributes are output as part of the selected TLB entry and require external logic for use. Generally these bits are ignored for system level memory access.

## U

Indicates a user / app level page indicating that the URWX bits are valid.

## SW

These three bits are available for use by software.

## G

Global page indicator. If a page is global then the ASID is not included when testing for an address match. This allows pages to be included in all address spaces.

## A

Whenever the page is accessed (any read, write or execute), this bit is set. It may be cleared only when the TLBE is updated, usually by software.

## M

Whenever a write cycle is done to the page the modified bit is set. It may be cleared only when the TLBE is updated, usually by software. The TLB detects write cycles using the 'store\_i' input signal when a valid virtual address is present.

## RGN

RGN is used to select a region from the region table in the MMU. The region table describes characteristics of the selected region including type and starting and end addresses. More information on the region table is available in the MMU documentation.

## LVL

LVL specifies the level of the page table. The page table level affects the interpretation of the shortcut page number. Page table entries in a level zero page table may not be shortcut. Shortcutting a level one entry results in an 8MB page. Shortcutting a level two entry results in an 8GB page.

## VPN

Virtual Page Number. The low order bits of the VPN are not stored in the TLB as they are used in selecting the TLB entry. There is already an implicit match on the low order bits. How many low order bits are not stored depends on the number of TLB entries. Forty-two bits are stored, when combined with the un-stored bits and the bits required to represent a page, a full 64-bit virtual address may be used in the default configuration.

## Count

The count field of the TLBE is used in invalidating the entire TLB. The count field must match the invalidate count (iv\_count) input for the TLBE to be considered valid. To invalidate the entire TLB increment the input iv\_count field and use the new value for new entries. Existing translations will no longer match. It is still necessary to step through the TLB invalidating old entries, but does not need to be done immediately.

## NRU

This bit is reserved to indicate ‘not recently used’. It is not currently in use.

## ASID

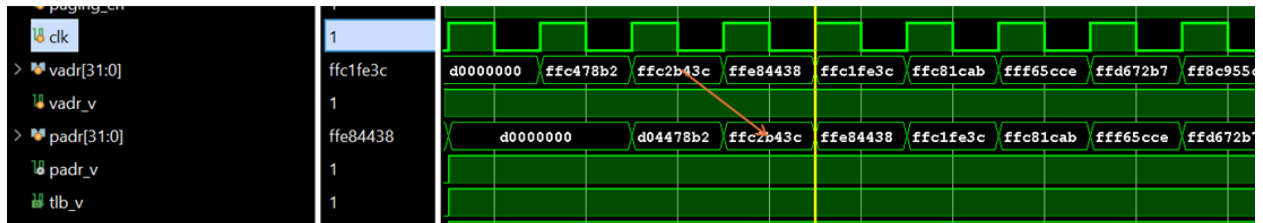
The ASID must match the ASID supplied with the virtual address for a match unless the global ‘G’ bit is set in the PTE in which case the ASID is ignored.



## Performance

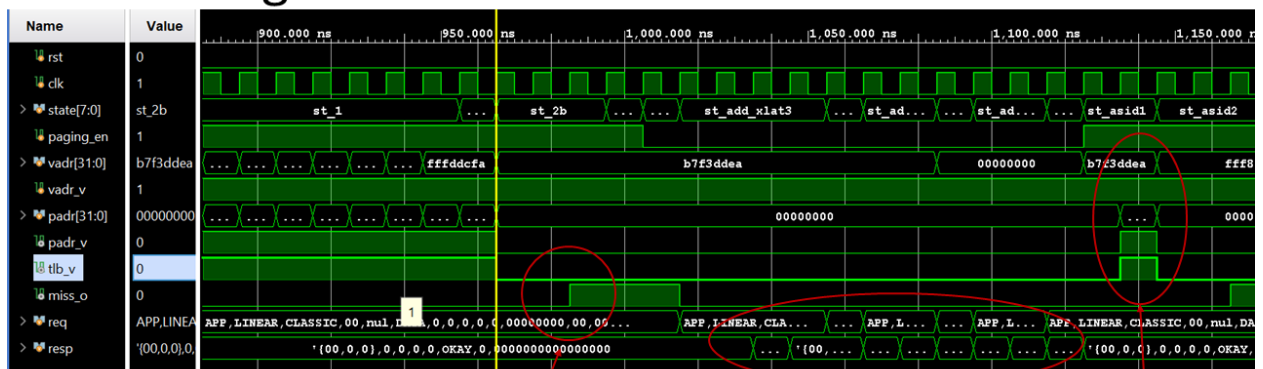
For a TLB hit the address is translated in one clock cycle. There may be an additional clock cycle(s) required if the page changes. If addressing stays on the same page timing is single cycle. On a miss, timing depends on how fast the TLB can be updated.

# Address Translation Timing



Address is translated after one clock cycle as shown above.  
Address translations shown above are for a 1:1 mapping.

## Miss Timing



Miss recognized

### Test state machine updates TLB

Translation valid

## Approximate Resource Usage

(default configuration – AMD / Xilinx Vivado toolset, runtime optimized flow)

3100 LUTs

500 FFs

8 BRAMs

## Timing

200 MHz (-2 Kintex device)

## Parameters

Parameter	Default Value	Description
TLB_ENTRIES	512	Number of entries in the TLB, should be a power of two with a maximum of 65536.
TLB_ASSOC	4	Associativity of the TLB. Maximum of 256.
LOG_PAGESIZE	13	Number of address bits required to index a page.
LRU	1	Least Recently Used – replacement policy, if LRU is zero replacement is random.

## Signal Description

Signal	Width	Dir.	
clk	1	I	Core clock
bus		I/O	WISHBONE bus interface (includes reset)
idle	1	I	Indicates that an external machine is idling waiting for misses
paging_en	1	I	Enables paging to take place, the TLB valid signal will not be active unless paging is enabled.
cs_tlb	1	I	Circuit select for the TLB registers
iv_count	4	I	Invalidation count, used when invalidating the entire TLB
store_i	1	I	Indicates that a memory store operation is taking place. Used to update the modified flag. This should be supplied with the virtual address.
id	8	I	A generic identifier, propagated to the miss_id on a miss, could be a buffer index.
asid	16	I	The current ASID for the system
vadr	32+	I	Virtual address to translate
vadr_v	1	I	Indicates a valid virtual address is present
padr	32+	O	The physical address for the corresponding virtual one.
padr_v	1	O	Indicates the physical address is valid. Note this signal does not indicate translation is complete. See the tlb_v signal.
tlb_v	1	O	Indicates that the translation is complete and valid.
missack	1	I	Indicates that the miss has registered externally
miss_adr_o	32+	O	The miss address (pipelined version of vadr)
miss_asid_o	16	O	The asid for the corresponding miss
miss_id_o	8	O	From the id field
tlb_entry	128	O	The TLB entry selected for the translation
rst_busy	1	O	Indicates the TLB is resetting.
empty		O	Indicates that the currently selected set contains empty TLBEs

paging\_en should be zero to disable paging while the TLB is updated or read.

## Limits

64k Entries

256 ways

Since the TLB may be used in a hash table system, it is allowed a large number of entries.

A realistic limit is probably four to eight ways. Each way uses more memory and reads in parallel with other ways.

## WISHBONE Compatibility Datasheet

WISHBONE Datasheet		
WISHBONE SoC Architecture Specification, Revision B.3		
Description:	Specifications:	
General Description:	Translation Lookaside Buffer (TLB core)	
Supported Cycles:	SLAVE, READ / WRITE SLAVE, READ-MODIFY-WRITE	
Data port, size:	64 bit	
Data port, granularity:	64 bit	
Data port, maximum operand size:	64 bit	
Data transfer ordering:	Little Endian	
Data transfer sequencing	any (undefined)	
Clock frequency constraints:	none	
Supported signal list and cross reference to equivalent WISHBONE signals	Signal Name:	WISHBONE Equiv.
	Bus.resp.ack_i	ACK_I
	Bus.req.adr_o(31:0)	ADR_O()
	clk	CLK_I
	bus.resp.dat(63:0)	DAT_I()
	bus.req.dat(63:0)	DAT_O()
	bus.req.cyc	CYC_O
	bus.req.stb	STB_O
	bus.req.we	WE_O
	bus.req.sel(7:0)	SEL_O
Special Requirements:		