

VME64x User Guide

25/07/2012

1 Introduction

This document provides an overview of the features VME64x slave supports. All implemented functionalities conform to the standards defined by ANSI/VITA VME64x Standard [1] [2][3][6], but not all are implemented. It also provides an overview of the default power-up configuration and configuration procedure.

The implementation follows the design rules set by Design rules for custom VME modules in CMS [4].

This core implements a VME64 slave on one side and a WishBone master on the other without FIFOs in-between.

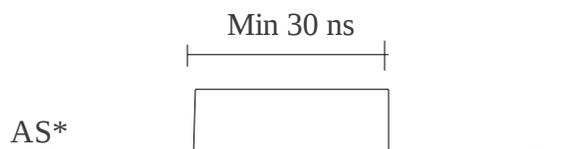
The WB **pipelined single** read/write implemented functionalities conform the Wishbone B4 standard [5].

The core supports SINGLE, BLT (D32), MBLT (D64), transfers in A16, A24, A32 and A64 address modes and D08 (OE), D16, D32, D64 data transfers. The core can be configured via implemented CR/CSR configuration space. A ROACK type IRQ controller with one interrupt input and a programmable interrupt level and Status/ID register is provided.

2 System clock

Due to the follow timing constraint a 50 MHz clock or more is suggested:

Fig. 25 , pag. 107 “VME bus specification” ANSI/IEEE STD1014-1987



As show in the figure, to be sure that the slave detects the rising edge and the following falling edge on the AS* signal the clk_i's period must be maximum 30 ns.

The minimum time between two consecutive cycle observed with the

VMETRO is 45 ns, but a 50 MHz clock or more is suggested.

The VME Bus is asynchronous so each board plugged into the crate can work with a different frequency.

The clock in the vme64x core must be the same as the clock in the WB Slave application.

3 VME64x features

This chapter lists and explains features that VME64x slave implements.

3.1 CR/CSR space

For the sake of a “plug and play” capability CR/CSR space is implemented as defined by ANSI/VITA Standards for VME64 Extensions [2].

In order to provide “plug and play” capability VME64x provides a mechanism very similar to PCI. A dedicated “Configuration ROM / Control & Status Register” (CR/CSR) address space has been introduced. It consists of ROM and RAM regions with a set of well defined registers. It is addressed with the address modifier 0x2F in the A24 address space.

Every VME module occupies a 512 kB page in this address space. The location of this page in the A24 space is defined by geographical address lines on the backplane: each slot is provided with a unique geographical five bit address at the J1 connector (row d). From these bits A23...A19 of the CR/CSR page are derived.

If the geographical address is not correct (GA parity bit does not match), the base address is set to 0x00, which indicates a faulty condition. An **odd** parity is used!

If the board is plugged into an old crate that doesn't provide the GA lines, all the GA lines are asserted to '1' by the PU resistors on the boards and the BAR register is set to 0x00; it is not set by hand with some switches on the board so in this condition the CR/CSR space can't be accessed! The vme64x core provides an internal flag s_BARerror that detects this error condition. You can find this flag in the VME_CR_CSR_Space.vhd component and you can use this flag to drive a led on the board.

The CR/CSR space can be accessed with the data width D08(E0), D16 byte(2-3) and D32. Please note that in compliance with the CR/CSR definition, only every fourth location in the CR/CSR space is used. If the master tries to write another location the write will not take effect and if the master reads the byte(0) or byte(1) or byte(2) locations the value returned is 0.

User CSR space and User CR space are not implemented. The CRAM space is implemented from 0x001000 to 0x07fbff.

The location of the user defined CR and CSR regions as well as the CRAM region are programmable. For each of these, six bytes defining the start and the end address (with respect to the start of the configuration space) are reserved in the CR region. Designers are free to use these regions for module specific purposes.

If a user defined region is not implemented the start and end address must be set to 0x000000.

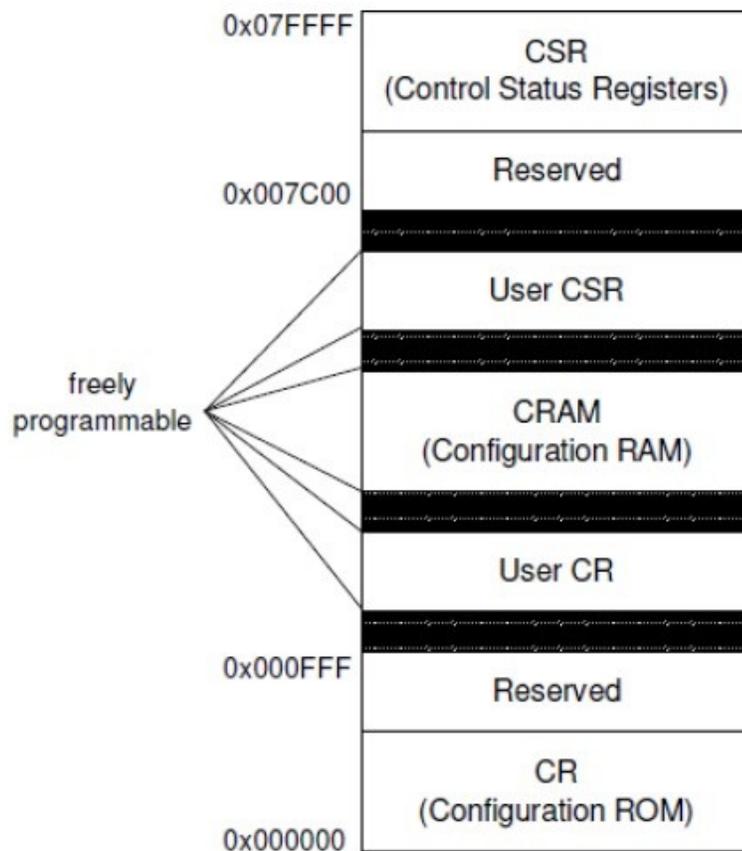


Figure 1: Configuration space organization in VME64x

The CR space is implemented in the file VME_CR_pack.vhd.

The CSR space is implemented in the file VME_CSR_pack.vhd.

All the registers in the CSR space have been implemented as defined by the VME64 Extensions [2].

The following registers have been added in the Reserved CSR:

IRQ_Vector : loc. 0x07FF5F

IRQ_level : loc. 0x07FF5B

MBLT_Endian : loc. 0x07FF53

TIME0 : loc. 0x07FF4F

TIME1 : loc. 0x07FF4b

TIME2 : loc. 0x07FF47

TIME3 : loc. 0x07FF43

TIME4 : loc. 0x07FF3F

BYTES0 : loc. 0x07FF3b

BYTES0 : loc. 0x07FF37

WB32or64 : loc. 0x07FF33

The VME Master can set the IRQ level and the Status/ID (IRQ Vector) of each slave board.

With the MBLT_Endian register the Master can select the data swap type:

0x00 NO swap.

0x01 Byte swap.

0x02 Word swap.

0x03 Word + Byte swap.

0x04 Dword + Word + Byte swap.

All the TIME_x and BYTE_x registers can be used to calculate the data transfer rate; these registers storage the time in ns and the number of bytes transferred.

The WB32or64 tells the slave if the WB slave is 32 or 64 bit:

0x00 : WB 64 bit

0x01 : WB 32 bit

If the data is stored in more than 1 location of memory (eg. The ADER registers are 4 byte width) the BIG ENDIAN order is used!

3.2 Data types

This VME64x core supports D08(OE), D16, D32, D64 data transfers. The implementation assumes that the target data memory is 8-bit wide.

Each byte in the memory has one unique address.

The vme64x core supports the byte access so the WB memory should be 8 bit granularity.

Upon D16 access, only every other byte is addressed indeed the D16 byte(1-2) access is not supported, upon D32 every fourth and upon D64 data access only every eighth byte is addressed.

3.3 Addressing types

The address width can be 16 bit, 24 bit, 32 bit and 64 bit to address more than 1 TB of memory.

The list of the addressing type supported with their address modifier codes can be find here:

http://www.ohwr.org/projects/vme64x-core/repository/raw/trunk/documentation/user_guides/VME_access_modes.pdf

3.4 2e transfers

Two edge transfers **have not been implemented yet.**

The 2eVME transfers don't need FIFO memory between the VME and WB bus; the WB Master inside the vme64x core can work in the pipelined single write/read mode. The 2eSST mode is not an handshake protocol so the FIFO memory between the VME and WB bus will be necessary.

With 2e transfers, master supplies the address along with some additional data in three address phases as shown in Table 2.

Signal Line	Address Phase 1	Address Phase 2	Address Phase 3	Data Phase
AM[5:0]	0x20	0x20	0x20	0x20
A[7:0]	XAM Code	A[3:0]= 0 Device address A[7:0]	Reserved	D[39:32]
A[15:8]	Device address A[15:8]	Beat/Cycle count	Reserved	D[47:40]
A[23:16]	Device address A[23:16]	A[23:21]= 0 A[20:16]= GA of Master	Reserved	D[55:48]
A[31:24]	Device address A[31:24]	Subunit Number in Master	Reserved	D[63:56]
D[31:0]	Device address A[63:32]	D[3:0] = Transfer rate 2eSST D[4] = Odd bit for 2eSST D[31:5] = Reserved	Reserved	D[31:0]

note: signal LWORD is regarded as address bit 0 .

Table3 shows the supported XAMs:

XAM	Address/Data Mode
0x01	A32/D64 2eVME
0x02	A64/D64 2eVME
0x11	A32/D64 2eSST
0x12	A32/D64 2eSST

3.5 Signals

This section focuses on functionality of certain VME bus signals.

3.5.1 RESET

RESET resets the entire core to the default configuration.

The reset signals are: VME_RST_n_i, software reset (BIT_SET_REG[7]), and we suggest to add a hand reset on the board.

3.5.2 BERR

BERR signal is used to signal a bus error. A transfer cycle is terminated with assertion of this signal if the VME64x slave does not recognize the data or addressing type used in the transfer cycle, if master attempts to write to a read-only memory (CR) or if error is received from the module which is addressed or if master attempts to access in BLT mode with D08 or D16, or if the master attempts to access in MBLT mode and the WB bus is 32 bit. The vme64x asserts this signal in the DTACK_LOW state, and not as soon it detects one error condition, to avoid that a temporary error condition causes the BERR assertion.

3.5.3 RETRY

RETRY signal terminates the transfer cycle if VME64x slave receives a retry request from the addressed module (via the WishBone bus), signaling that the read/write request cannot be completed at this time.

3.6 Interrupts

Interrupt controller is a ROACK type controller.

Upon synchronously detecting a pulse on the interrupt request signal input on the WB bus, the VME64x core drives the IRQ request line on the VME bus low thus issuing an interrupt request. VME master acknowledges the interrupt in a form of an IACK cycle.

During the IACK cycle the vme64x core sends the IRQ_Vector to the master. After the interrupt is acknowledged, the VME IRQ line is released.

Before issuing a new interrupt request the vme64x core waits that the master reads the Interrupt Counter register.

There are seven VME IRQ lines but only one interrupt request input. For the purpose of configuring which of the seven IRQ lines the VME64x core will drive (in response to a rising edge on the IRQ input), an IRQ Level register has been implemented in the user CSR space. The value of this register corresponds to the number of the IRQ line on the VME bus that is to be used (note that on the VME master side priorities are taken into an account, IRQ7

having the highest priority and IRQ1 the lowest). If the IRQ level register is set to 0x00 or values above 0x07, interrupts are disabled.

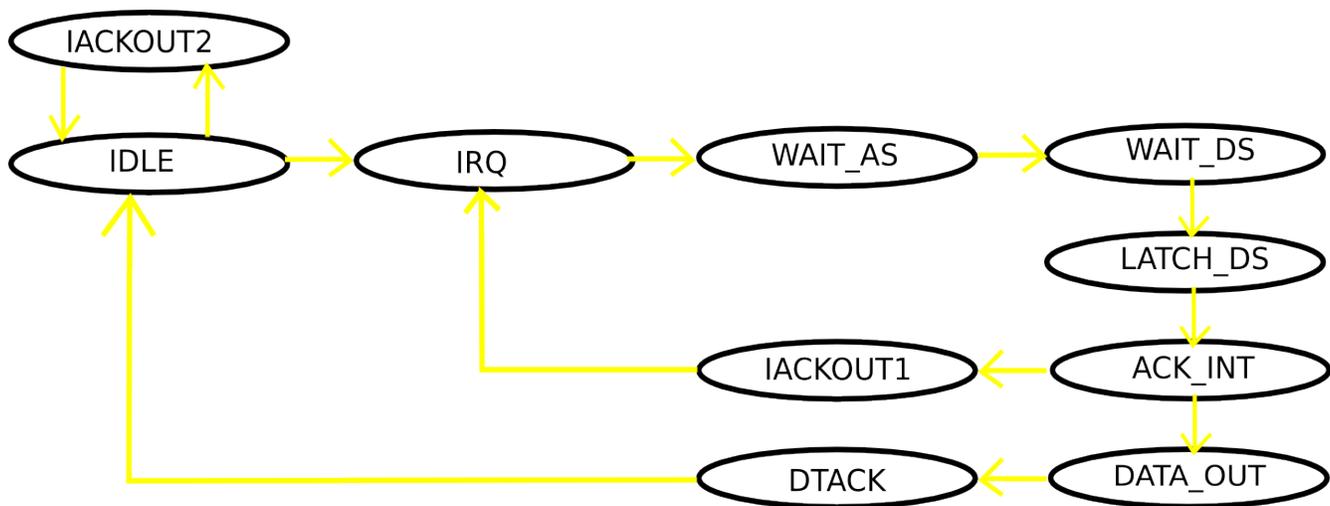
In the default power-up and reset configuration the interrupts are disabled. To enable the interrupt the Master must write a non-zero value in the Interrupt rate register.

The table 4 shows the Interrupts related registers:

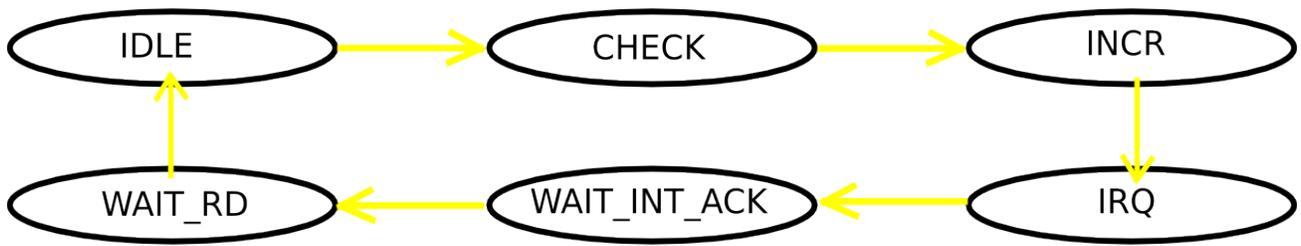
Register	Where	Address
IRQ Vector	CR/CSR Space	0x07ff5f
IRQ Level	CR/CSR Space	0x07ff5b
INT_RATE	Application memory	0x04
INT_COUNT	Application memory	0x00

To generate the Interrupt request pulse the IRQ_Generator must be insert in the Application (WB Slave) as shows the Interconnection Diagram, chapter 8.

3.6.1 IRQ Controller FSM



3.6.2 IRQ Generator FSM

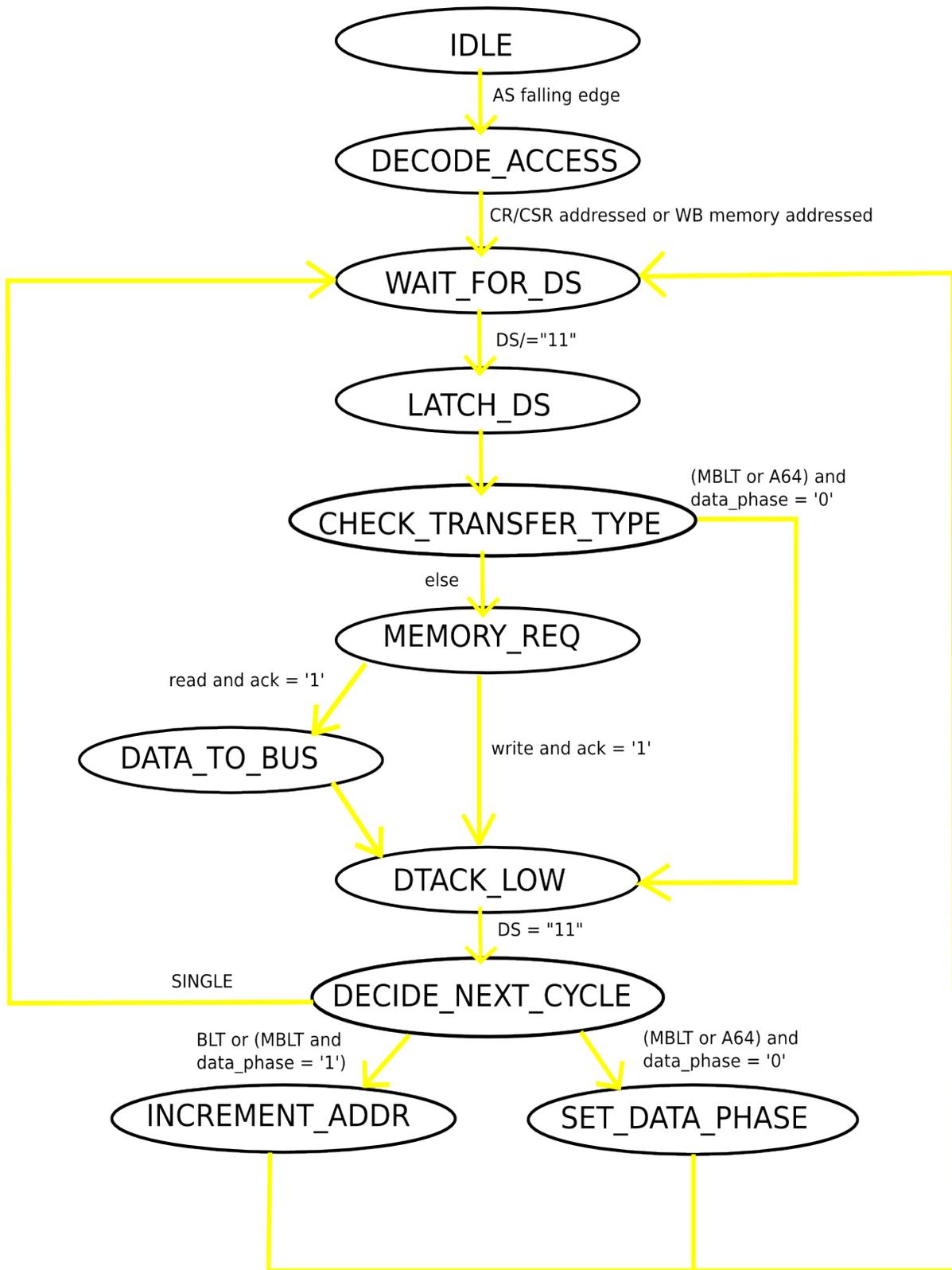


3.7 MAIN FSM

The Main finite state machine is located in the VME_bus.vhd component. This FSM does not support the 2e transfers.

If the board is not addressed the FSM is in the DECODE_ACCESS state until the rising edge on AS signal. Indeed the FSM is resetted by a rising edge on AS signal.

If the board is addressed, it will acknowledge the cycle in the DTACK_LOW state. If same error are detected, in this state the BERR* line is asserted.



4 Configuration

Upon power-up or reset, the module is disabled and only its CR/CSR space can be accessed. Software must then first map the module memory in the 64-bit address space by setting Address Decoder Compare (ADER) registers in CSR, which, together with Address Decoder Mask (ADEM) registers in CR relocate the module memory to the desired address range.

ADER for each function also contains an AM or XAM code to which it responds.

After the module has been placed in the desired address space, it can be enabled by writing to Bit Set Register in the CSR and thus setting the correct enable bit:

BIT_SET_REG = 0x10

Upon power-up or reset, the vme64x core supports WB data bus 32 bit. The vme64x core provides 64 bit data by writing into WB32or64 register in the CSR space:

loc 0x7ff33 WB32or64 = 0x00 ← WB 64 bit.

Function0 reserved for A32, A32_BLT and A32_MBLT access mode.

Function1 reserved for A24, A24_BLT and A24_MBLT access mode.

Function2 reserved for A16 access mode.

Function3 and Function4 reserved for A64, A64_BLT and A64_MBLT access mode.

Function5 and Function6 reserved for 2e transfers.

For more information about the decode phase see the VME_Access_Decompose.vhd component.

5 VME bus transceivers

The VME64x slave core also includes output signals that drive external hardware transceivers. These signals are DTACK OE, DATA DIR, DATA OE, ADDR DIR and ADDR OE, RETRY_OE.

Direction (DIR) signals specify the direction of data and address. These signals should be used as select line in the multiplexer on bidirectional signals.

Output enable (OE) signals are used to disable the transceivers so that the

buses are effectively isolated.

OEAB	OEBy _n	OUTPUT
L	H	Z
H	H	A to B
L	L	B to A
H	L	A to B and B to Y

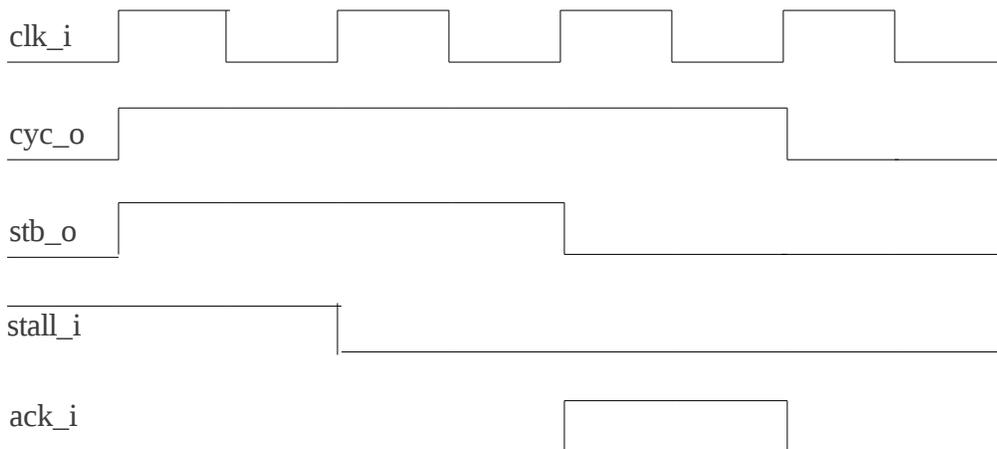
OEn	DIR	OUTPUT
H	X	Z
L	H	A to B
L	L	B to A

6 WishBone master

This section describes the functional operation of the WishBone Master component VME_Wb_master.vhd.

When the core is addressed with SINGLE, BLT, MBLT, 2eVME (not yet supported) transfers the WB master operates in accordance with official WB specifications document [5].

In particular it works in pipelined single write/read mode:



If the WB slave doesn't drive the stall signal, it can set the stall signal to 0. Err and rty signals are supported by the vme64x core.

The Wb bus can be 32 bit or 64 bit .

7 I/O ports

Clock:

clk_i : in std_logic;

VME signals:

VME_AS_n_i : in std_logic;
VME_RST_n_i : in std_logic;
VME_WRITE_n_i : in std_logic;
VME_AM_i : in std_logic_vector(5 downto 0);
VME_DS_n_i : in std_logic_vector(1 downto 0);
VME_GA_i : in std_logic_vector(5 downto 0);
VME_BERR_o : out std_logic;
VME_DTACK_n_o : out std_logic;
VME_RETRY_n_o : out std_logic;
VME_LWORD_n_b_i : in std_logic;
VME_LWORD_n_b_o : out std_logic;
VME_ADDR_b_i : in std_logic_vector(31 downto 1);
VME_ADDR_b_o : out std_logic_vector(31 downto 1);
VME_DATA_b_i : in std_logic_vector(31 downto 0);
VME_DATA_b_o : out std_logic_vector(31 downto 0);
VME_IRQ_n_o : out std_logic_vector(6 downto 0);
VME_IACKIN_n_i : in std_logic;
VME_IACK_n_i : in std_logic;
VME_IACKOUT_n_o : out std_logic;

VME bus transceivers :

VME_DTACK_OE_o : out std_logic;
VME_DATA_DIR_o : out std_logic;
VME_DATA_OE_N_o : out std_logic;
VME_ADDR_DIR_o : out std_logic;
VME_ADDR_OE_N_o : out std_logic;
VME_RETRY_OE_o : out std_logic;

WishBone signals:

DAT_i : in std_logic_vector(63 downto 0);
DAT_o : out std_logic_vector(63 downto 0);
ADR_o : out std_logic_vector(63 downto 0);
CYC_o : out std_logic;
ERR_i : in std_logic;
RTY_i : in std_logic;
SEL_o : out std_logic_vector(7 downto 0);
STB_o : out std_logic;
ACK_i : in std_logic;
WE_o : out std_logic;
STALL_i : in std_logic;

IRQ Generator signals:

INT_ack : out std_logic;
IRQ_i : in std_logic;

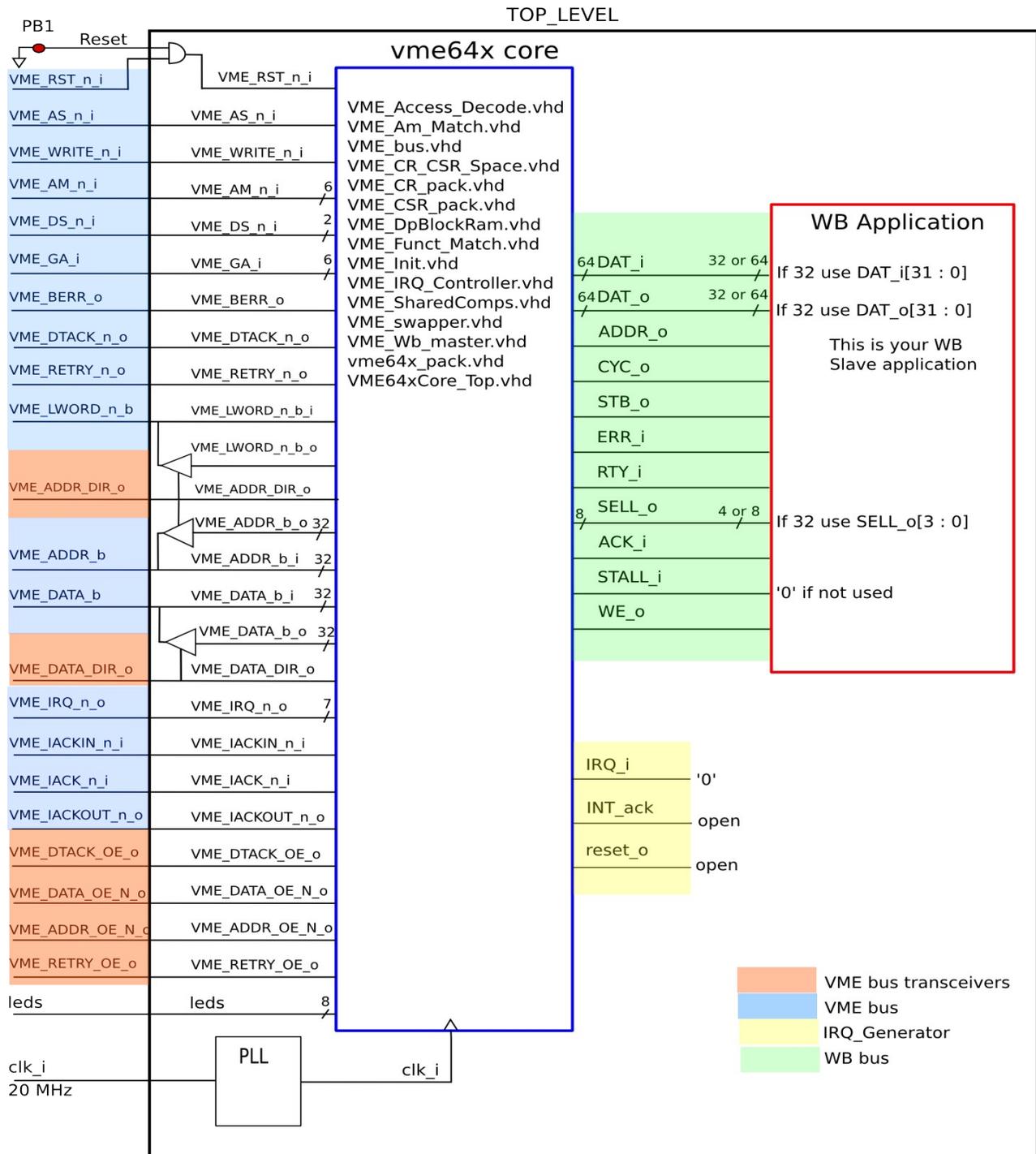
```

reset_o      : out  std_logic;    -- asserted when '1'
Debug (8 leds on the board):
leds        : out  std_logic_vector(7 downto 0);

```

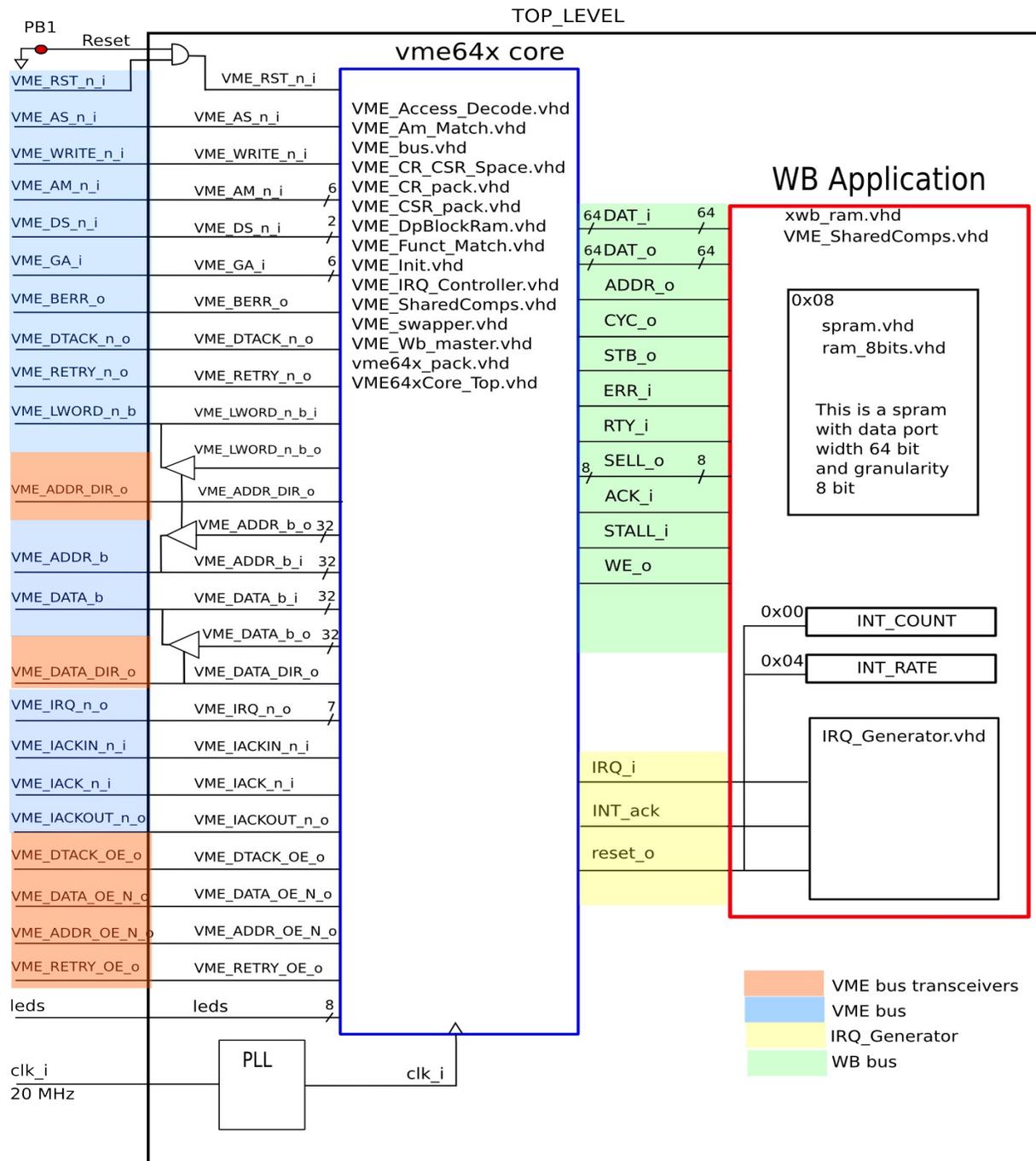
8 Interconnection Diagram

This diagram would be an example showing you how you can connect the vme64x core to your WB Slave application without interrupt generator:



The WB slave Data Bus can be 32 bit or 64 bit; if 32 bit remember to set the following bit to '1':
 loc: 0x7ff33, reg. WB32or64 : 0x01 (default configuration).

The next Interconnection Diagram shows the WB Slave that is used to debug the vme64x core and to develop the Interrupter. If you need the Interrupter you should use these components and you can't use your WB application yet.



9 TEST

Here some guidelines to use the vme64x core:

http://www.ohwr.org/projects/vme64x-core/repository/raw/trunk/documentation/user_guides/python_test.pdf

10 Transfer Rate

Here some transfer rates detected during the tests:

50 MHz		80 MHz	
Single access	MBLT access	Single access	MBLT access
About 10 MB/s	About 18 MB/s	About 15 MB/s	About 23 MB/s

11 References

- [1] The VMEbus Specification ANSI/VITA October 1987
- [2] VME64 Extensions ANSI/VITA 1.1 1997
- [3] VME64 ANSI/VITA 1 1994
- [4] CMS Internal Note
Design rules for custom VME modules in CMS, CMS IN 2004/005
- [5] Wishbone System-on-chip (SoC) Interconnection Architecture for
Portable IP Cores, Revision B4
- [6] ANSI/VITA 1.5-2003 2eSST