VME64x to WB core User Guide

This core implements a VME64x slave - WB master bridge.

The vme64x core conform to the standards defined by ANSI/VITA VME64 and VME64x Standards. In particular this core has been provided of the "plug and play" capability; it means that in the vme64x core you can find a CR/CSR space whose base address is setted automatically with the geographical address lines and has not to be set by jumpers or switches on the board. Indeed this operation is error prone. Software must map the module memory in the address space by writing the CSR space as explained later.

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

Since the vme64x core acts as a WB master in the WB side, the WB pipelined single read/write transfer has been provided to the core. This functionality conform the Wishbone B4 standard.

Features

VME interface

This section lists the VME features supported by the core.

Supported:

- D08(EO), D16, D32
- Addressing mode: A16, A24, A32
- BLT, MBLT
- D08(O), I(7-1), ROAK interrupts
- (compatible with MEN A25 master board)
- CR/CSR space with extensions from VME64x
- Geographic Address (GA), dynamic configuration (ader). (TODO: support noga/usega for svec ?)
- Interrupt

Not supported:

- 2eVME
- 2eSST (no hardware)
- Dynamic size (DFSR, DFS)
- Fixed address (FAF)
- Extra Function Mask (EFM)
- XAM (no 2e)
- A40, A64 (not supported by MEN A25)

- MD32 (multiplexed data cycle, only for A40)
- LCK (bus lock)
- UAT (unaligned accesses)
- RMW cycles (but should work)
- ADO, ADOH (address only cycle)
- D08, D16 for BLT (??)
- RETRY (cf rule 2.91 - incompatibility with WB)

WB interface

This section corresponds to the datasheet required by the WB specification.

1. Compliant to Wishbone B4 specifications
2. Slave
3. Signals name follows the specification
4. err_i is forwarded to VME as BERR*
5. rty_i is not supported
6. no TAGs
7. Port size is 32 bit
8. Port granularity is 8 bit
9. Maximum operand size is 8 bit (TBC)
10. Data transfer ordering is BIG ENDIAN
11. Sequence of data transfer is defined by the VME side
12. No CLK_I signal, clock is provided separately.

- Non pipeline behaviour (but compatible with pipeline). The core doesn't take any advantage of the pipeline behaviour, as the WB bus is much faster than the VME bus.

CR/CSR space

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

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 pull-up 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 CR/CSR space can be accessed with the data width D08(EO), 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.

As you can see in the table below, not all this space of memory is defined yet, and the digital designer can add additional CR and CSR spaces called User Csr and User CR which are not implemented in the vme64x core.

The location of the User CR and User 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.

By default the size of the CRAM space is 0, which means it is disabled and doesn't use any resources. User can define the address range of CRAM in order to generate a programmable area.

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

Start Address	End Address	Content
0x7ff60	0x7ffff	CSR (Control Status Registers)
0x7fc00	0x7ff5f	Reserved for CSR
BEG_USER_CSR	END_USER_CSR	User defined CSR (option)
BEG_CRAM	END_CRAM	User defined CRAM (option)
BEG_USER_CR	END_USER_CR	User defined CR (option)
0x00000	0x00fff	CR (Configuration ROM)

In addition to the standard registers in the CSR space, the VME64x defines by default a user CSR space (within the CSR space reserved by the VME64x standard) with the following registers:

Address	Content	Reset value
0x7ff5f	IRQ vector	0x00
0x7ff5b	IRQ level	0x00

This is for compatibility with existing drivers for previous version of the core.

Interrupt controller

The interrupt controller implemented is a ROAK (Release On Acknowledge) type controller. It means that the Interrupter releases the interrupt request lines when it acknowledges the interrupt cycle. Upon synchronously detecting a rising edge 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. The 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.

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 account, IRQ7 having the highest priority and IRQ1 the lowest). If the IRQ level register is set to 0x00, interrupts are disabled. In the default power-up and reset configuration the interrupts are disabled.

There is a non-standard mechanism to retrigger unhandled interrupts. Once an interrupt is asserted by the WB slave, the interrupt is marked as pending and the interrupt request is relayed on the VME bus. The OS and the driver has to acknowledge the interrupt and to act on the hardware so that the WB slave doesn't request anymore OS attention. If the OS acknowledge the interrupt but doesn't quiet the request, the VME64x Core will relay again the interrupt on the VME bus after 1ms.

References

The specifications used for this core are:

- VME64 ANSI/VITA 1-1994 (Stabilized Maintenance 2011)
- VME64 Extensions ANSI/VITA 1.1-1997 (Stabilized Maintenance 2011)
- Wishbone System-on-chip (SoC) Interconnection Architecture for Portable IP Cores, Revision B4

VME64x Core Instantiation

There are two top-level entities. The `xvme64x_core` is the main one, where records are used for the `g_DECODER` generic, VME and WB buses in order to simplify the connections. The `vme64x_core` has exactely the same features but all generics and ports are unwrapper to allow interfacing with verilog code.

Generics

The first generic `g_CLOCK_PERIOD` defines the clock in ns. This generic must be set by user and is used for synchronization of the VME DS signal.

Generic `g_DECODE_AM` can be set to false for compatibility with the previous version of this core. When false, the AM field of ADER is not used when decoding address, so the core will recognize any access allowed by the corresponding AMCAP.

Generic `g_USER_CSR_EXT` can be set to true if a user defined CSR is provided. The interface with the user CSR is a very simple synchronous SRAM (signals `user_csr_addr_o`, `user_csr_data_i`, `user_csr_data_o` and `user_csr_we_o`). In addition, the ports `irq_level_i` and `irq_vector_i` are used by the interrupt controller to define the irq level and vector (otherwise they are read from the default user CSR registers).

The other generics define values in the CSR. The package `vme64x_pkg` defines some constants for these values, and you can refer to VME64x specification for details about these values:

- `g_MANUFACTURER_ID` for the manufacturer ID,
- `g_BOARD_ID` for the board ID,
- `g_REVISION_ID` for the revision ID,
- `g_PROGRAM_ID` for the type of code in CR,
- `g_ASCII_PTR` for the pointer to the user defined ASCII string in CR,
- `g_BEG_USER_CR` and `g_END_USER_CR` for the range of the user defined CR area. If the range is not null, ports `user_cr_addr_o` and `user_cr_data_i` must be connected to a ROM.
- `g_BEG_CRAM` and `g_END_CRAM` for the range of user CRAM. If the range is not null, the core instantiates an SRAM.
- `g_BEG_USER_CSR` and `g_END_USER_CSR` for the range of the user defined CSR. See above the description of `g_USER_CSR_EXT`.
- `g_BEG_SN` and `g_END_SN` for the area in CR of the serial number.
- `g_DECODER` describes the 8 function decoder. Each decoder is described by the following bits (see VME64x specification for details):
    - `adem` bits 8 to 31: address mask
    - `adem` bits 0 to 7: must be set to 0
    - `amcap`: address modifier supported by the decoder. Only bits 0x08 to 0x0f and 0x38 to 0x3f can be set to 1.
    - `dawpr`: data access width (ignored by the decoder). Note that setting `adem` to 0 disable the decoder. If decoders N to 7 are disabled, they don't use any hardware resources.

Ports

- `clk_i` is the clock signal, and the clock of the WB bus. Note that the `g_CLOCK_PERIOD` generic must be set according to the `clk_i` frequency to get correct timing for the VME `DS` signals. The VME `DTACK` and `BERR` signals are supposed to be released at most 30ns once `DS` is released; as the design needs 4 closk to release them (due to synchronizer), this means the minimal frequency is supposed to be 133Mhz. In practice, VME masters are much more tolerant.
- `rst_n_i` is the reset signal. It could be considered as a power-on reset.
- `rst_n_o` is the reset signal to the wishbone core. It is asserted in case of reset on the VME bus, or if the module reset bit is set in the CSR, or if the `rst_n_i` signal is asserted.
- `vme_i` and `vme_o` are signals for the VME bus. Refer to the VME64 standard for details.
- `wb_i` and `wb_o` are the signals for the WB bus. Refer to the WB specification for details. Note that the WB `rty` (retry) signal cannot be used, as the VME BLT transactions can only be retried during the address phase and this restriction is not exposed to the WB side. The WB err signal is forwarded to the VME bus as BERR. The address on the WB bus corresponds to the lower bits of the address on the VME bus (bits used to decode the address are cleared on the WB bus).
- `irq_ack_o` signal is asserted during one cycle when the VME64x Core acknowledge the interrupt on the VME bus. This signal could be used by the slace interrupt controller.

The following signals are used only when the `g_USER_CSR_EXT` generic is set to true:

- `irq_level_i` defines which VME IRQ signal is asserted by the VME64x Core to send an interrupt to

the VME bus. If set to 0, interrupts are never sent. The level corresponds to the interrupt priority on the VME bus, 7 is the highest priority and 1 the lowest. The value shouldn't change while an interrupt is pending.
- `irq_vector_i` is the vector sent on the VME bus by the core during an acknowledge cycle.
- `user_csr_addr_o`, `user_csr_data_i`, `user_csr_data_o`, `user_csr_we_o` define an interface to an external SRAM containing the user CSR values. For read cycles, the data value must be stable on the next cycle.

The following signals are used when a user CR area is defined:

- `user_cr_addr_o`, `user_cr_data_i` define an interface to an external ROM containing the user CR values. Data must be stable on the next cycle.

Note that the `vme` ports are designed to be connected to VME bus transceivers like SN74VMEH2250. In the particular case of the CERN SVEC card, the signals `berr` and `irq` are inverted by the transceivers, so a `not` gate must be inserted in the FPGA. You can refer to the `svec_vmecore_test_top.vhd` file in the svec repository for an example.

Programing the VME64x Core

After power-up or reset, the VME core is disabled (as the `module_enable` bit is cleared) and therefore only the CS/CSR space can be accessed. Software must then first map the module memory in the address space by setting the Address Decoder Compare (ADER) registers in CSR, which, together with Address Decoder Mask (ADEM) registers in the CR relocate the module memory to the desired address range. ADER for each function must also contains athe AM code to which it responds. After the module has been placed in the desired address space, it can be enabled by writing 0x10 (`module_enable`) to the Bit Set Register i the CSR.

If the master needs to access to the slave using different address space (e.g. both A32 and A24), or different transaction (e.g. both single and BLT), several function decoders must be used.

The base address is of the CR/CSR space is setd by the geographical lines. If the core is used in an old VME system without GA lines, the core should be provided with a logic that detects if GA = "11111", which is invalid.

It is possible to reset the card in software by setting the `reset` bit in the Bit Set Register. Contrary to the VME64 specification (and for backward compatibility with drivers and previous versions of the core), this bit clears itself during the next CSR write access.

Performances

The performances were measured with the `test_vme` program, available in the svec repository. In these measures, the master is the A20 board.

A24 SCT DMA:

- Read Rate: 15.761240 MB/sec
- Write Rate: 17.172745 MB/sec

A24 BLT DMA:

- Read Rate: 12.472540 MB/sec
- Write Rate: 12.932751 MB/sec

A24 MBLT DMA:

- Read Rate: 25.906049 MB/sec
- Write Rate: 26.259754 MB/sec

According to the simulation, the bad performances of BLT transfer is due to the master.

Changes in V2 (compared to previous version)

- Core is smaller (< 1000 slices)
- No retry
- No endianess convertion
- WB data bus is 32 bit
- Internal component declarations removed.
- Async inputs registered with gc_sync_register.

Internals

xvme64x_core.vhd

The top module `xvme64x_core` instantiates the sub-modules, and also synchronize the asynchronous VME signals (that need to be) to avoid metastability problems.

This module also handles the `g_USER_CSR_EXT` generic and instantiate a default user CSR if the generic is set to false.

vme_bus.vhd

This is the main module. It implements an FSM to handle the VME protocol, and acts as the interface between the VME bus and either the WB bus or the CR/CSR memory.

The module also handles the interrupt acknowledge. If IACK is asserted on a falling edge of AS, the cycle is considered as a acknowledge cycle. The FSM then waits until IACKIN is asserted (or until AS is deasserted). If an interrupt was pending at the right level when IACKIN is asserted, the VME64x Core responds to the acknowledge cycle with the interrupt vector; otherwise it asserts IACKOUT.

vme_cr_csr_space.vhd

This module implements the CR and CSR spaces. It builds (during elaboration) the CR memory from the generics value, handle accesses from the VME bus to these memories, interfaces with CRAM (if present), user CR (if present) and user CSR (if present).

vme_func_match.vhd

This module checks if the VME address+AM has to be handled by this VME slave according to ADER and decoder values. Gives back the corresponding WB address.

vme_user_csr.vhd

This module implements a default user CSR with the irq_level and irq_vector registers.

vme_irq_controller.vhd

This module implements the interrupt controller. The interrupt cycle is:

1. The wishbone slave generates an pulse on the `int` line when it has to interrupts the master
2. If no interrupt is pending and the retry mechanism is not started, this module asserts (to 0) the corresponding VME IRQ line (as defined by `irq_level`).
3. When ack'ed, the interrupt is marked as not pending anymore.
4. If the interrupt request stays active for more than 1 cycle (therefore it isn't a pulse), a retry mechanism is started. The interrupt will be re-sent on the VME bus every 1ms as long as it is active.

VME64 VITA-1 rules compliance

Note: Master/D64/A64 means N/A as the rule doesn't concern this core.

- 2.1a: Master
- 2.69: Master
- 2.2: Followed
- 2.3: Followed, excluded from c_AMCAP_ALLOWED. [no TB]
- 2.70: D64
- 2.7: Followed
- 2.8: Followed
- 2.9: Followed
- 2.71: A64
- 2.10: Master
- 2.11: Followed
- 2.72: A64
- 2.73: A40
- 2.74: Followed (A32, A24, A16 supported)
- 2.75: Followed (likewise)
- 2.76: Followed (D16 and D08(EO) supported)
- 2.77: Followed (likewise)
- 2.4: Followed (D32 supported)
- 2.5: Followed (D16 supported)

- 2.12a: Master
- 2.78: Master
- 2.66: Followed
- 2.79: Master
- 2.80: Master
- 2.6: Followed [no TB]
- 2.68: Followed [no TB]
- 2.81: Followed (LOCK not accepted)
- 2.82: Followed (likewise)
- 2.83: Master
- 2.84: Followed (lock)
- 2.85: Followed (CR/CSR layout)
- 2.86: Followed
- 2.87: Followed (D08(O) is the data access supported)
- 2.93: Master
- 2.18: Followed (A[] and LWORD lines are registered)
- 2.19: Master
- 2.20: Master
- 2.21: Master
- 2.22: Master
- 2.23: Master
- 2.24: Master
- 2.25: Followed (DATA lines are all driven for read, MBLT not supported)
- 2.26: Followed (Likewise)
- 2.27: Master
- 2.28: Master
- 2.29: Master
- 2.30: Master
- 2.31: Master
- 2.32: Master
- 2.33a: Master
- 2.34a: Master
- 2.35: Master
- 2.36: Master
- 2.37: Master
- 2.38: Master
- 2.39: Master
- 2.40: Master
- 2.41: Master
- 2.42: Master
- 2.43: Master
- 2.44a: Master
- 2.94: Master

- 2.45: Master
- 2.46a: Master
- 2.47a: Master
- 2.48a: Master
- 2.49: Master
- 2.50: Master
- 2.51: Master
- 2.52: Master
- 2.95: Master
- 2.96: Master
- 2.53a: Followed (VME_DATA_DIR is set only once DSA goes low)
- 2.54a: Followed
- 2.55: Followed (number of states in the main FSM + synch FF)
- 2.28a: Followed (likewise)
- 2.56a: Followed
- 2.57: Followed
- 2.98: TBC
- 2.58a: Followed (released at the same time)
- 2.99: TBC (retry)
- 2.100: TBC (retry)
- 2.101: TBC (retry)
- 2.102: TBC (retry)
- 2.103: TBC (retry)
- 2.104: TBC (retry)
- 2.105: TBC (retry)
- 2.59: Bus timer
- 2.60: Bus timer
- 3.x: Arbitration
- 4.1: Backplace
- 4.50: Followed (slave can generate interrupt)
- 4.2 Followed