VMEbus Slave Core Component

User's Guide

Version 2.0
December 12, 2002

TABLE OF CONTENTS

1. License Information

This document is intended to describe the features and use of the Millogic LTD MDSVME-S VHDL Core Component of the VMEbus Slave. This VHDL core is distributed in source form on a single site right to use basis. Any redistribution of this core, for use at other sites or for resale is prohibited without the written consent of a representative of Millogic LTD. Please refer to the Millogic IP License for further details.

2. Introduction

2.1. Core Definition

The VMEbus Slave Core Component is a synthesizable VHDL core used to incorporate the VMEbus slave function into an FPGA or ASIC. It is highly configurable and includes the following features:

· A16, A24, and/or A32 address response
· D32 and/or D16 data bus support
· user configurable address modifier code response
· user-defined address decode with programmable wait states for each region
· read and write strobes for user registers
· asynchronous and synchronous ready input lines
· latched address generation

The core allows for all of the features to be configured in terms of bus size and offset, and enabled or disabled through VHDL constants in a single header file. The result is a core which can be easily integrated into a target design and synthesized into an efficient core.

2.2. VHDL Environment

The core is written in IEEE-1076-87 VHDL using the data types defined in IEEE-1164 (STD_LOGIC) and will be compatible with any simulator/synthesizer which meets these standards. For use with noncompliant tools, please contact Millogic. The package "millsubs.vhd" contains subtype declarations for STD_LOGIC and

STD_LOGIC_VECTOR types which are used throughout the design. STD_LOGIC is cast as SLB and STD_LOGIC_VECTOR is cast as SLV. SLV may be followed by the width which is always of the form *descending range downto 0*. For example, SLV32 is equivalent to STD_LOGIC_VECTOR(31DOWNTO 0).

2.3. Overview of Use

The VMEbus Slave Core Component consists of the following three modules, each of which is an entity/architecture pair: vme_addr.vhd, vme_data.vhd, and vme_slave_core.vhd. An overview of each of these modules is presented in this section. There are also two packages provided for use with the core:

vme_slave_header.vhd and
millsubs.vhd

The vme_slave_header package is used to define all configurable parameters for the core as well as arrays used throughout and component declarations. The millsubs package is discussed above under *VHDL Environment*. All core files reside in *[vmes/behv].*

2.3.1. The VMEbus Address Module

The VMEbus Address module (vme_addr.vhd) implements the address portion of the VMEbus slave cycle. It can be configured to respond to A16, A24, and/or A32 cycles, and with separate address modifier codes for each. Each of the three addressing modes may be enabled or disabled through constants. When an addressing mode is disabled it is excluded from the synthesized output netlist. For each addressing mode, the user defines the specific range of address bits, and the patterns on those address bits that the slave will respond to.

The address modifier codes are defined individually for each addressing mode as an unconstrained array of AM codes. In this way the user defines both the specific AM codes, as well as the quantity of codes which the addressing mode will respond to.

The core provides a latched address, if desired. The specific address bits for the latched address can be defined, as well as the function enabled/disabled via constants. The vme_addr module generates a signal called "address match" (ADDR_MATCH) which is sent to the VMEbus Data module to qualify the cycle.

2.3.2. The VMEbus Data Module

The VMEbus Data module (vme_data.vhd) implements the data portion of the VMEbus slave cycle. It contains a synchronous state machine, wait state counter, mux, and upper and lower strobe decoders. A cycle is initiated in the VMEbus Data module by the assertion of both data strobes when the ADDR_MATCH signal from the VMEbus Address module is active.

There are two separate upper address decoders, one decoder for reads and one decoder for writes provided by the module (signals VMEREAD_SPACE and VMEWRITE_SPACE). These are intended to define separate address spaces for different regions of the slave, such as DRAM, device registers, PROM, etc. They are driven active in the beginning of the data portion of the slave cycle and remain active until it is finished. The number of strobes and specific address bits used in the upper decode are defined as constants in the header file, as are enable constants to disable them for synthesis.

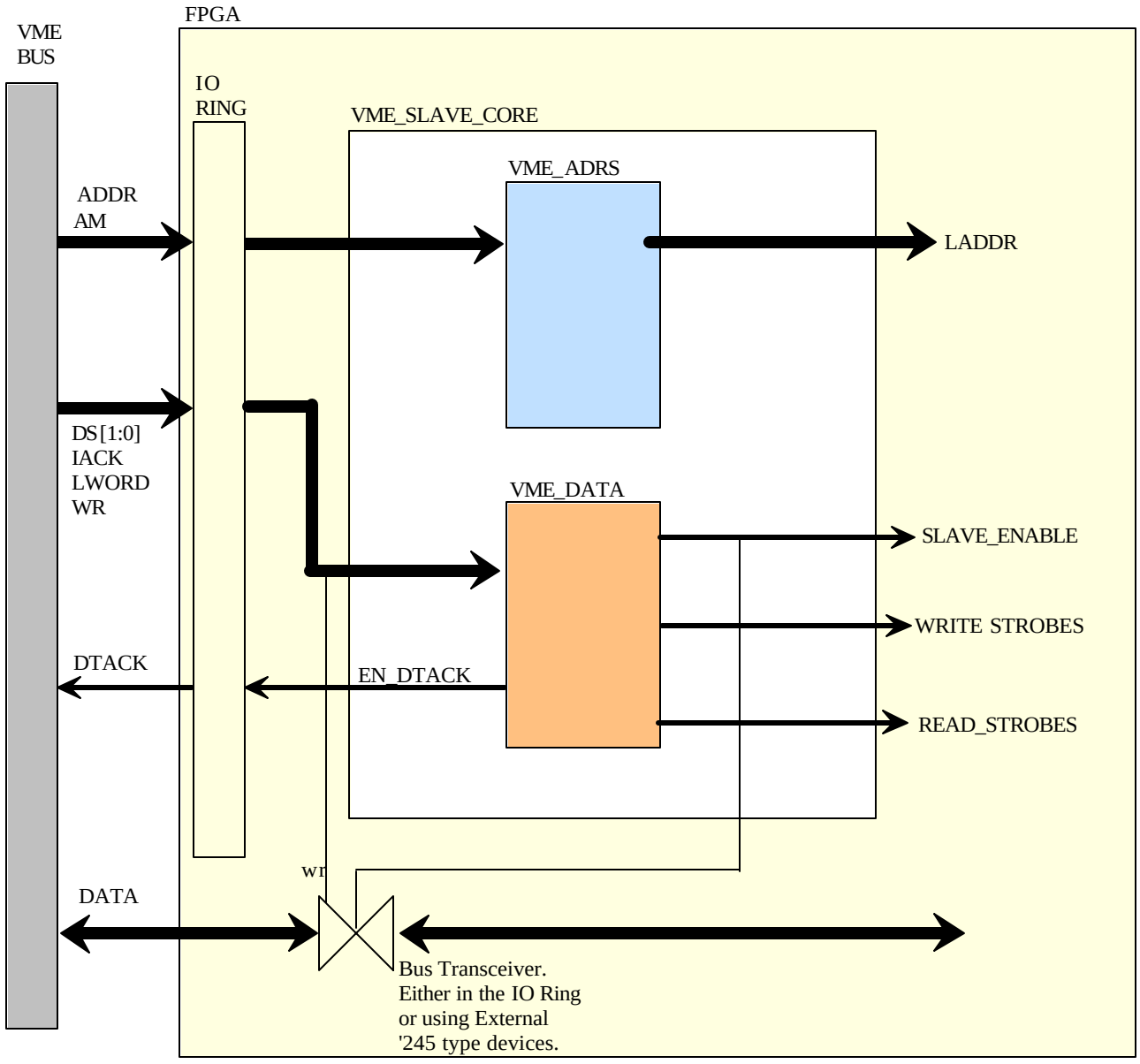
Wait states for slave operations are defined in an array of integers. The specific wait state value is selected in a multiplexer by the WAITREGION_SELECT vector. The intention is that the upper order address decode is fed back into the WAITREGION_SELECT vector to select the wait state value for a particular address space. This method allows for a separate number of wait states for read and write cycles, as well as for each address space. Wait state generation may also be disabled for synthesis through an enable constant.

There are four separate lower address decoders - for read operations and write operations (the signals RD_STROBES and WR_STROBES) and for D16 support (LOW_RD_STROBES and LOW_WR_STROBES). The write strobes are intended to be used as "clock enables" or clocks for device registers. They are activated for one clock pulse at the end of all write wait states. The read strobes are activated in the beginning of the read cycle and can be used to gate data to the VMEbus. The number of strobes and the specific address bits used for decode can be defined via constants. Also, the read and write strobes can be disabled for synthesis via an enable constant.

The VMEbus Data module also generates the signal SLAVE_ENABLEN which is a low active signal used to enable external data transceivers (’245 style) and the internal bidirect pads. The VMEbus acknowledge signal DTACK is driven by an open-collector (or open drain) output. The VMEbus Data module generates the enable signal for this driver (EN_DTACK). This signal enables when low and can be used with either

internal open-drain drivers (if high sink current drivers are available in the target technology) or an external driver.

2.3.3. The VMEbus Slave Core Module



The VMEbus Slave Core module is the top level of the core component. It contains one instantiation of the VMEbus Address module and one instantiation of the VMEbus Data module. The constants for each of the instantiated modules are brought up to the top level of the VMEbus Slave Core module to allow for

integration of the slave by instantiation of a single component. A configuration declaration at the bottom of the vme_slave_core.vhd module binds the two instantiated modules from the library WORK.

2.3.4. The VMEbus Testbench

Also on the disk is a testbench [*vmes/tbench/*] for the core consisting of the following files (listed in dependency order):

Std_logic_textio	STD text IO file used in the vme_model.vhd file.
vme_model.vhd	This is a number of modules thatexecute cycles defined in the vme_slave_cycs.vhd file.
vme_slave_cycs.vhd	This is a definition of VME cycles to execute targeted at the core.
Vme_timer.vhd	A bus timeout module to help in simulation.
vme_slave_tb.vhd	Top level testbench. This file instantiates the core and VME behavioral models.

These files allow the slave core to be demonstrated immediately.

3. Installation

The installation disk contains three categories of files:
1. The VHDL VMEbus Slave Core Component files
vme_slave_core.vhd - the top level of the core
vme_addr.vhd - the address cycle logic
vme_data.vhd - the data cycle logic
vme_slave_header.vhd - a package of type definitions
millsubs.vhd - a Papillon package of common subtype definitions and subroutines

2. The VHDL testbench (see above).

3. This manual in electronic form:
vme_slave_core_man.pdf - a pdf version of this manual

3.1. Installation Directories

The user must relocate the files into the appropriate directories within their project directory. The base directory is name vmes. Under that is “behv” which contains the core code. The directory “tbench” contains the testbench. The directory “docs” contains documentation.

3.2. Compilation and Use with the VMEbus Behavioral Master Model

If the MDL-VME-M VMEbus Behavioral Master Model has been purchased with the slave core, the testbench and cycle file can be compiled with the model and immediately demonstrated. The testbench instantiates theVMEbus Behavioral Master Model (design unit *vme_model.vhd*), the slave core (design unit *vme_slave_core*), and a bus timer to generate the buserror signal (design unit *vme_timer*).

Assuming that the installation of both the master model and the slave core has been completed including compilation, the following three files should be compiled in order:
1. vme_slave_cycs.vhd
2. vme_model.vhd
3. vme_slave_tb.vhd

The vme_model module must be compiled since it imports vme_slave_cyc.vhd. Upon completion of this, the master and slave can be immediately simulated together.

4. Core Integration into the target application.

This section describes the details of integrating the VMEbus Slave Core Component into the target application.

4.1. Instantiation

Instantiation of the VMEbus Slave Core component is accomplished by instantiating the top level - vme_slave_core.vhd and modifying the constants to customize it for a particular application. The vme_slave_core module contains instantiations of the vme_addr and vme_data components.

4.1.1. Port Description

The following is a detailed description of each port:

vme_slave_core.vhd port description			
PORT	**CLASS**	**MODE**	**DESCRIPTION**
CLK	SYS	IN	Used for the synchronous logic in vme_data. The frequency of operation is technology dependent.
RESET	SYS	IN	Asynchronous reset is used to reset all flip-flops and (programmably) the latched address latches. The polarity of the reset level for the flip-flops for target technology is controlled by the RESET_LEVEL constant. RESET must be manually inverted in the device if the board-level reset differs from the value of the RESET_LEVEL constant.
ADDR	VME	IN	Bits 31 down to 1 of the VME address are input to the module for address decode and latched address generation.
AM	VME	IN	The six address modifier code bits from the VMEbus are input to the module for decode. The desired codes to compare against are defined in the EXT_AM_ARRAY, STD_AM_ARRAY, and SHORT_AM_ARRAY constants. AS IN Address Strobe from the VMEbus.
IACK	VME	IN	Interrupt Acknowledge is used in the decode in vme_addr. Cycles in which IACK is active are ignored.
LWORD	VME	IN	Longword is used in the decode in vme_addr.
DS0,DS1	VME	IN	The data strobes are synchronized in vme_data and used to start the data portion of the slave cycle. They are also used to deassert DTACK at the termination of a cycle
WR	VME	IN	The VMEbus WRITE signal is used to distinguish between a read and a write in vme_data.

SYNC_READY	Enable	IN	The synchronous ready signal is used to gate the assertion of DTACK. It is logically ANDed with the WAITDONE output from the wait state logic such that the later of SYNC_READY and the wait state counter expiration will initiate DTACK. SYNC_READY is not synchronized and must be setup to CLK.
ASYNC_READY	Enable	IN	The asynchronous ready signal is used to gate the assertion of DTACK. It is logically ANDed with the WAITDONE output from the wait state logic such that the later of ASYNC_READY and the wait state counter being expired will initiate DTACK. ASYNC_READY is synchronized in vme_data.
RD_STROBE_DEC	Enable	IN	This input is used to qualify the generation of the RD_STROBES. When this signal is high, and the RD_STROBES are enabled, the RD_STROBES will be generated for read cycles. This input may be connected to the desired VMEREAD_SPACE output.
WR_STROBE_DEC	Enable	IN	This input is used to qualify the generation of the WR_STROBES. When this signal is high, and the WR_STROBES are enabled, the WR_STROBES will be generated for write cycles. This input may be connected to the desired VMEWRITE_SPACE output.
WAITREGION_ SELECT	Enable	IN	This input is used to select the particular wait state from the WAIT_ARRAY constant. This signal is automatically sized by the number of values entered into this constant. This signal is usually driven by the VMEWRITE_SPACE and VMEREAD_SPACE address decode outputs
SLAVE_ENABLEN	Enable	OUT	An active low signal, SLAVE_ENABLEN is used to enable both the device data output pins (when WR is high) and the external data transceivers. WR_STROBES OUT The WR_STROBES are outputs from the lower address decoder. They are asserted high when the wait states and/or READY lines have been asserted (at the end of the write cycle) for one clock period. The number of strobes is determined by the NO_OF_WR_STROBES constant. The address bits

			used for the decode are determined by the WR_REG_HIGH and WR_REG_LOW constants. The WR_STROBES are used for D32 accesses and D16 accesses of bytes 2 and 3.
LOW_WR_STROBES	Enable	OUT	The LOW_WR_STROBES are outputs from the lower address decoder. They are asserted high when the wait states and/or READY lines have been asserted (at the end of the write cycle) for one clock period. The number of strobes is determined by the NO_OF_WR_STROBES constant. The address bits used for the decode are determined by the WR_REG_HIGH and WR_REG_LOW constants. These strobes are only used when D16 access is enabled with the EN_D16 constant and are used for D32 or D16 accesses of bytes 0 and 1.
VMEWRITE_SPACE	Enable	OUT	The VMEWRITE_SPACE signals are the outputs from the upper address decode for write operations. The number of strobes is determined by the NO_OF_WR_SPACE_STROBES constant. The specific address bits used in the decode are determined by the WR_SPACE_HIGH and WR_SPACE_LOW constants.
RD_STROBES	Enable	OUT	The RD_STROBES are outputs from the lower address decoder. They are asserted high at the beginning of the read cycle. The number of strobes is determined by the NO_OF_RD_STROBES constant and the address bits used for the decode are determined by the RD_REG_HIGH and RD_REG_LOW Constants. The RD_STROBES are used for D32 accesses and D16 accesses of bytes 2 and 3.
LOW_RD_STROBES	Enable	OUT	The LOW_RD_STROBES are outputs from the lower address decoder. They are asserted high at the beginning of the read cycle. The number of strobes is determined by the NO_OF_RD_STROBES constant. The address bits used for decode are determined by the RD_REG_HIGH and RD_REG_LOW constants. These strobes are only used when D16 access is enabled with the EN_D16 constant and are used for D32 or D16 accesses of bytes 0 and 1.

VMEREAD_SPACE	Enable	OUT	The VMEREAD_SPACE signals are the outputs from the upper address decode for read operations. The number of strobes is determined by the NO_OF_RD_SPACE_STROBES constant. The specific address bits used in the decode are determined by the RD_SPACE_HIGH and RD_SPACE_LOW constants.
EN_DTACK	VME Enable	OUT	Enable DTACK is a low-active signal used to enable the open-collector DTACK driver. The input to the driver should be tied to ground. Note that the DTACK driver needs to be a high-current driver to be VME compliant (an external driver may be required depending upon the target technology).
LADDR	Local	OUT	The Latched Address outputs are simply latches on the ADDR lines. The latch is transparent when AS is high and latched when AS is low. The constants LA_HIGH and LA_LOW are used to define the address range of the latched address. If latches with asynchronous reset are available in the target technology, the constant RESET_LATCHES should be set to TRUE. MUXDATA16 OUT Used to mux bytes 0 and 1 onto byte lanes 2 and 3 for D16 accesses.

4.1.2. Constant Descriptions

This section describes all of the constants in detail. All constants used in the core are in the file “vme_slave_header.vhd”. The following is a listing of the complete constant set, followed by detailed descriptions of the constants categorized by function:

4.1.2.1. Address Decode Constants (vme_addr module)

The vme_addr module provides three separate decodes - one for extended address space, one for standard address space, and one for short address space. Each of the three decoders has five constants to control its response.

The first is an enable constant which turns on or off that specific decoder for synthesis. The decoder will not be included in the target netlist if the enable bit is set to FALSE.

There are three constants for each decoder to specifically control the address being compared. The ADDR_HIGH and ADDR_LOW constants specify the address range for the comparison and the ADDR_MATCH constant specifies the pattern. For example, if one wants the core to respond to address bits 31downto 24 equal to all 1's in extended address space, the EXT_ADDR_HIGH constant would be set to 31, the EXT_ADDR_LOW constant would be set to 24, and the EXT_ADDR_MATCH constant would be set to "11111111".

The last constant is the AM_ARRAY constant which is declared to be an array of STD_LOGIC_VECTOR(5 DOWNTO 0), or an array of address modifier codes. This array is declared with the AM codes which the slave should respond to (for a particular address space). For example, if the Extended address decoder is to respond to AM codes hex 0E and hex 0D, the constant would be declared in the following fashion:

```
EXT_AM_ARRAY : AM_ARRAY_TYPE := ("001110", -- 0X0E
                                 "001101"); -- 0X0D
```

The array is unconstrained; the size is determined by the amount of values assigned in the constant. Note that for only one entry, some VHDL tools prefer the value declared with a named association (otherwise they get confused by the bit string literal) such as:

```
EXT_AM_ARRAY : AM_ARRAY_TYPE := (0 => "001110"); -- 0X0E
```

The following table lists all the address decode constants with a detailed description of each:

CONSTANT	DATA TYPE	DESCRIPTION
EN_EXTENDED	BOOLEAN	Disables the address decode for Extended address space when FALSE. Enables the decode when

		TRUE.
EXT_ADDR_HIGH	INTEGER	The high address bit of the range to compare against in Extended address space.
EXT_ADDR_LOW	INTEGER	The low address bit of the range to compare against in Extended address space.
EXT_ADDR_MATCH	SLV	The actual pattern to match against for Extended address space. This value is compared against the address bits defined by EXT_ADDR_HIGH and EXT_ADDR_LOW and must match that size.
EXT_AM_ARRAY	array of SLV6	The address modifier codes for Extended address space that the slave is to respond to are entered in this array. Any amount of 6-bit vectors may be entered.
EN_STANDARD	BOOLEAN	Disables the address decode for Standard address space when FALSE. Enables the decode when TRUE.
STD_ADDR_HIGH	INTEGER	The high address bit of the range to compare against in Standard address space.
STD_ADDR_LOW	INTEGER	The low address bit of the range to compare against in Standard address space.
STD_ADDR_MATCH	SLV	The actual pattern to match against for Standard address space. This value is compared against the address bits defined by STD_ADDR_HIGH and STD_ADDR_LOW and must match that size.
STD_AM_ARRAY	array of SLV6	The address modifier codes for Standard address space that the slave is to respond to are entered in this array. Any amount of 6-bit vectors may be entered.

EN_SHORT	BOOLEAN	Disables the address decode for Short address space when FALSE. Enables the decode when TRUE. SHORT_ADDR_HIGH INTEGER The high address bit of the range to compare against in Short address space.
SHORT_ADDR_LOW	INTEGER	The low address bit of the range to compare against in Short address space.
SHORT_ADDR_MATCH	SLV	The actual pattern to match against for Short address space. This value is compared against the address bits defined by SHORT_ADDR_HIGH and SHORT_ADDR_LOW and must match that size.
SHORT_AM_ARRAY	array of SLV6	The address modifier codes for Short address space that the slave is to respond to are entered in this array. Any amount of 6-bit vectors may be entered.

4.1.2.2. The Reset Level Constant (all modules)

The RESET_LEVEL constant defines the polarity of asynchronous reset for the target technology. If the flip-flops in the target technology have a low active reset the value should be '0'. If the flip-flops have a high-active reset the value should be '1'. Note that it will be necessary to invert the polarity of the reset pin if the board-level reset polarity does not match the value of the RESET_LEVEL constant.

CONSTANT	DATA TYPE	DESCRIPTION
RESET_LEVEL	SLB	If the flip-flops in the target technology have a low active reset the value should be '0'. If the flip flops have a high-active reset the value should be '1'.

4.1.2.3. Latched Address Constants (vme_addr module)

The VMEbus Address module provides a latched address for interface to peripherals and memory devices. The code is written to synthesize into transparent latches, if available in the target technology, otherwise it will synthesize into a discrete gate implementation. The latches are transparent when the VMEbus address strobe (AS) signal is high and latched when low.

If the target technology contains latches with asynchronous reset, the constant RESET_LATCHES should be set to TRUE. This will tie the asynchronous reset into the latches using the polarity defined by the RESET_LEVEL constant.

The address range for the latched address is defined by the LA_HIGH and LA_LOW constants. Note that one can generally define a large address range and rely upon the synthesizer and vendor specific tools to "eat back" unused latches. The latching function can be disabled completely by setting the EN_ADDR_LATCH boolean to FALSE.

The following table describes the constants for the address latch:

CONSTANT	DATA TYPE	DESCRIPTION
EN_ADDR_LATCH	BOOLEAN	Disables the address latch when FALSE. Enables the address latch when TRUE.
RESET_LATCHES	BOOLEAN	Synthesizes asynchronous reset latches when TRUE, no asynchronous reset when FALSE.
LA_HIGH	INTEGER	Defines the high address range to latch.
LA_LOW	INTEGER	Defines the low address range to latch.

4.1.2.4. Wait State Generation (vme_data module)

The wait state generation logic consists of a loadable down counter, a mux used to select the number of wait states for a particular address space, and an equal-to-zero comparator. The size of the counter, number of wait state regions, and number of wait states for each region may be defined through constants.

The size of the counter, in bits, is defined in the WAITSTATE_COUNTER_SIZE constant. The wait state values are defined in the unconstrained array of type NATURAL called WAIT_ARRAY.

For example, if we want to define four different wait state regions with 3, 7, 2, and 1 wait states for the four regions respectively, the constant declaration would be:

WAIT_ARRAY : WAIT_ARRAY_TYPE := (3, 7, 2, 1);

The select lines for these four regions (un-encoded) are the WAITREGION_SELECT signal. In defining four entries in the WAIT_ARRAY constant the WAITREGION_SELECT signal is sized to be STD_LOGIC_VECTOR(3 DOWNTO 0) automatically. The WAITREGION_SELECT signal is generally driven by the VMEWRITE_SPACE and VMEREAD_SPACE outputs thus allowing for separate wait states for reads and writes, as well as for different address spaces.

Note that some VHDL tools will require named association for a single entry in the WAIT_ARRAY, such as:

WAIT_ARRAY : WAIT_ARRAY_TYPE := (0 => 3);

The wait state generation logic can be completely disabled by setting the EN_WAITSTATES boolean to FALSE.

The following table describes the constants for the wait state generation logic:

CONSTANT	DATA TYPE	DESCRIPTION
EN_WAITSTATES	BOOLEAN	Disables wait state generation when FALSE. Enables wait state generation when TRUE.
WAITSTATE_COUNTER_SIZE	NATURAL	Defines the size of the wait state counter (in bits).
WAIT_ARRAY	array of NATURAL	Defines the wait states for different regions.

4.1.2.5. Write Strobe Generation (vme_data module)

There are two different address decoders for write cycles in the VMEbus Data module - one for the upper address and one for the lower address. The intention of the upper address decoder is to define address spaces for the slave, such as

memory, I/O, device registers, etc. The intention of the lower decoder is to generate clock enable (or clock) signals for register writes.

The upper decoders (the VMEWRITE_SPACE signal) are asserted (high) in the beginning of the data portion of a write cycle and stay valid for the entire data cycle. The lower decode strobes (the WR_STROBES and LOW_WR_STROBES signals) are active for only one clock period at the end of the write cycle.

The WR_STROBES are used to perform writes for the entire 32-bit word when D16 operation is disabled (EN_D16 constant equals FALSE). In this case the LOW_WR_STROBES signal is not used. When D16 operation is enabled, the WR_STROBES are used to select bytes 2 and 3 and the LOW_WR_STROBES are used to select bytes 0 and 1. Both sets of strobes will go active for a D32 operation in this case.

There is a separate decode input for generating the WR_STROBES and LOW_WR_STROBES. The WR_STROBE_DEC signal must be active (high) for these signals to be asserted. It is intended that the decoded address space (one bit from VMEWRITE_SPACE) be used to drive the WR STROBE_DEC signal.

For example, if VMEWRITE_SPACE(0) is defined to be register address space, this signal will drive the WR_STROBE_DEC port to enable the write strobes for register writes. For both decoders, the number of strobes and the address range to be decoded are defined in constants.

There is also a separate enable constant for each set of strobes to disable it for synthesis.

The following table describes the constants for the write strobes:

CONSTANT	DATA TYPE	DESCRIPTION
EN_WR_SPACE_STROBES	BOOLEAN	Disables VMEWRITE_SPACE strobe generation when FALSE. Enables VMEWRITE_SPACE generation when TRUE.
NO_OF_WR_SPACE_STROBES	NATURAL	Defines the number of the VMEWRITE_SPACE strobes.
WR_SPACE_HIGH	NATURAL	Defines the upper address bit for the decode of VMEWRITE_SPACE.
WR_SPACE_LOW	NATURAL	Defines the lower address bit for the decode of VMEWRITE_SPACE.
EN_WR_STROBES	BOOLEAN	Disables WR_STROBES and

		LOW_WR_STROBES generation when FALSE. Enables write strobe generation when TRUE.
NO_OF_WR_STROBES	NATURAL	Defines the number of the WR_STROBES and LOW_WR_STROBES.
WR_REG_HIGH	NATURAL	Defines the upper address bit for the decode of the write strobes.
WR_REG_LOW	NATURAL	Defines the lower address bit for the decode of the write strobes.

4.1.2.6. Read Strobe Generation (vme_data module)

There are two different address decoders for read cycles in the VMEbus Data module - one for the upper address and one for the lower address. The intention of the upper address decoder is to define address spaces for the slave, such as memory, I/O, device registers, etc. The intention of the lower decoder is to generate enable signals for register read operations.

The upper decoders (the VMEREAD_SPACE signal) are asserted (high) in the beginning of the data portion of a read cycle and stay valid for the entire data cycle.

The lower decode strobes (the RD_STROBES and LOW_RD_STROBES signals) are also active for the entire read cycle. The RD_STROBES are used to perform reads of the entire 32-bit word when D16 operation is disabled (EN_D16 constant equals FALSE). In this case the LOW_RD_STROBES signal is not used. When D16 operation is enabled, the RD_STROBES are used to select bytes 2 and 3 and the LOW_RD_STROBES are used to select bytes 0 and 1. Both sets of strobes will go active for a D32 operation in this case.

There is a separate decode input for generating the read strobes. The RD_STROBE_DEC signal must be active (high) for the RD_STROBES and LOW_RD_STROBES to be asserted. It is intended that the decoded address space (one bit from VMEREAD_SPACE) be used to drive the RD_STROBE_DEC signal.

For example, if VMEREAD_SPACE(0) is defined to be register address space, this signal will drive the RD_STROBE_DEC port to enable the read strobes for register read operations. For both decoders, the number of strobes and the address range

to be decoded are defined in constants. There is also a separate enable constant for each set of strobes to disable it for synthesis.

The following table describes the constants for the read strobes:

CONSTANT	DATA TYPE	DESCRIPTION
EN_RD_SPACE_STROBES	BOOLEAN	Disables VMEREAD_SPACE strobe generation when FALSE. Enables VMEREAD_SPACE generation when TRUE.
NO_OF_RD_SPACE_STROBES	NATURAL	Defines the number of the VMEREAD_SPACE strobes.
RD_SPACE_HIGH	NATURAL	Defines the upper address bit for the decode of VMEREAD_SPACE. RD_SPACE_LOW NATURAL Defines the lower address bit for the decode of VMEREAD_SPACE.
EN_RD_STROBES	BOOLEAN	Disables RD_STROBES generation when FALSE. Enables RD_STROBES generation when TRUE.
NO_OF_RD_STROBES	NATURAL	Defines the number of the RD_STROBES and LOW_RD_STROBES.
RD_REG_HIGH	NATURAL	Defines the upper address bit for the decode of the read strobes.
RD_REG_LOW	NATURAL	Defines the lower address bit for the decode of the read strobes.

4.1.2.7. D16 Operation

The VMEbus Slave Core Component supports D16 operation. When the EN_D16 constant is set to TRUE, the slave will respond to D16 cycles. When the EN_D16 constant is set to FALSE, the core will not respond thus generating a bus error via the bus timer module.

When D16 operation is enabled, the LOW_WR_STROBES and LOW_RD_STROBES signals are used to select bytes 0 and 1. These strobes will go active for either a

D16 access to these bytes, or a D32 access. For the latter (D32 access), the WR_STROBES and RD_STROBES will also be activated to select bytes 2 and 3. The WR_STROBES and RD_STROBES will also be activated to

When D16 operation is disabled, LOW_WR_STROBES and LOW_RD_STROBES are not used; all accesses are performed with WR_STROBES and RD_STROBES. The MUXDATA16 signal is used to control a multiplexor for D16 operation. When N_D16 is set to TRUE, the slave will respond to D16 operations. In this case, all data transfers take place over byte lanes 2 and 3, and it is necessary to mux bytes 0 and 1 onto these byte lanes. MUXDATA16 goes active (high) for these cases.

4.2. Core Timing

This section describes some of the timing issues that can be encountered when instantiating the VMEbus Core Component.

4.2.1. The SLAVE_ENABLEN Signal

The SLAVE_ENABLEN signal is used to enable both the device data outputs, as well as the board-level transceivers (if required). The board-level transceivers should be of the 245 type with the VMEbus WRITE signal used as the direction. SLAVE_ENABLEN is generated by a purely combinatorial path which consists of the ADDR_MATCH signal from the vme_addr module and the VMEbus data strobes. This can be a very large combinatorial decode dependent upon the constants configured in the vme_addr module. In the vme_data module, SLAVE_ENABLEN is passed through a synchronizing flip-flop and is used to start the synchronous state machine in vme_data (the signal slave_enablen_s).

4.2.2. The WAITDONE Signal

I

The WAITDONE signal is generated by a comparison of the waitstate counter output to zero. There are two critical paths associated with this signal, and these are somewhat dependent upon the size of the counter and comparator. The first is when the waitstate counter is loaded. The LOADWAIT signal is asserted in the LOADWAITS state which means that the wait state counter actually gets loaded at the end of the WAITPROP state (on the next rising edge). There must be sufficient time for the waitstate counter to load, and then the loaded value to propagate through the WAITDONE comparator (WAITDONE will go inactive).

The first test for WAITDONE is performed in the following state, state WAITS. If higher frequency operation of the synchronous logic is desired, a second WAITPROP state can be added to allow an extra state for the counter load and compare. The second critical path is the decrement of the counter propagating through the compare-to-zero, and being set up as the WAITDONE signal in the WAITS state.

This critical path can be minimized with an added pipeline stage on the WAITDONE signal test in the WAITS state. Note that this will add an additional wait state.

4.2.3. The SYNC_READY Signal

The SYNC_READY signal is used to terminate the read or write cycle on a synchronous basis. It is directly tested in the state machine and must meet the setup and hold requirements.

4.2.4. The VMEWRITE_SPACE, VMEREAD_SPACE, WR_STROBES, RD_STROBES Signals

These strobe signals are decoded from the specified address bits and then registered in the state machine. Since the address should be stable before the start of the cycle, these signal are not very critical in terms of setup time.

4.2.5. The VMEbus WRITE Signal

The VMEbus WRITE signal, referred to as WR in the VMEbus Core Component, is tested directly in the state machine. This should not be time critical since the state machine is not initiated until a synchronized version of the data strobes is tested active (slave_enablen_s).