VME Slave Controller

A24-D32

Version 2.1

Abstract:

This VME slave controller is designed for custom integration using standard FPGA and ASIC technologies. It is fully compliant to the VME specification supporting A24 address mode, D8/D16/D32 data modes (read/write/read-modify-write) as well as interrupt acknowledge cycles. VMEbus timing is guaranteed by using a system clock of 40 Mhz or higher. A synchronous design approach is used to simplify interfacing to the asynchronous VMEbus. The user side interface is full synchronous. Data access is either single cycle or multi-cycle controlled through user wait states.
Deliverables of the core include RTL code, a self verifying tes bench, synthesis information and a user guide.

1 Overview

This VME slave controller core provides an easy-to-use, synchronous parallel interface towards custom logic (called user side), including full interrupt support. VME compatible external drivers can be directly connected to the VME. Examples are shown in the application section.

1.1 Features

This enhanced VME slave controller is based on our 16-bit iniVME core. Therefore it has the same easy-to-use and full synchronous user side interface, custom user address decode module and interrupt controller. This version supports the A24 address modes as well as D32 data modes. Additional enhancements include selectable rescinding DTACK, and read-modify-write cycles.

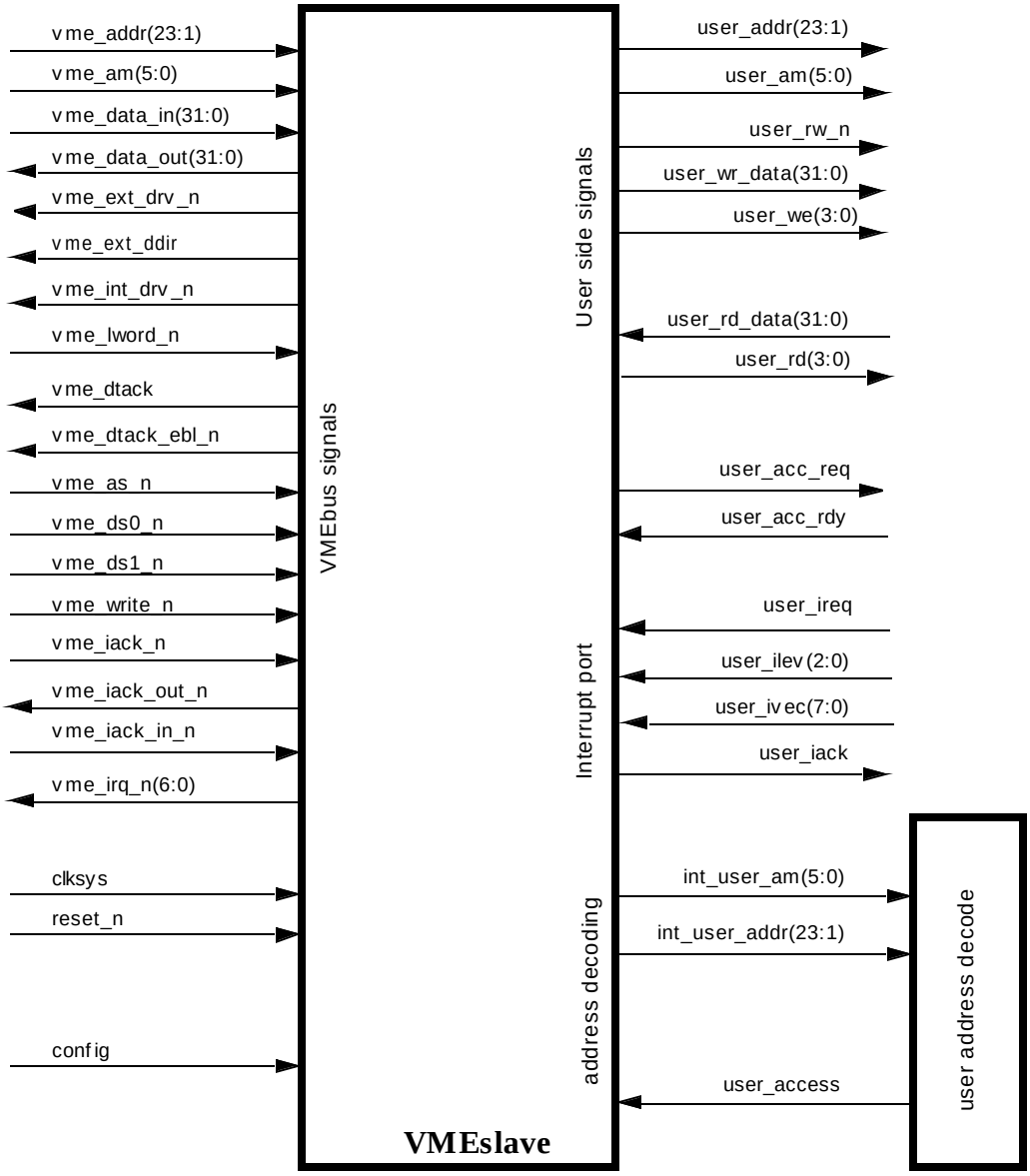
List of features:

- Data modes: D8, D16, D32
- Address modes: A16, A24
- Read, write, read-modify-write cycles
- Selectable rescinding DTACK
- D8, D16 and D32 interrupter
- Full synchronous user side interface for registers, peripherals and memories
- User selectable wait states

The VME slave core provides an easy to use, synchronous parallel interface towards custom logic (called user side), including full interrupt support. VME compatible external drivers can be directly connected to the VME. Examples are shown in the application section.

1.2 Inputs - Outputs

The following pictures shows all inputs and outputs:



2 IO description

The following part lists the input and output ports of the VME core and provides an description of their functionality.

2.1 General inputs

These pins are used to clock and initialize the whole VME core. To guarantee VME compliance, the falling edge of vme_as_n is used to latch the vme_addr and the vme_am signals. All other registers use the rising edge of 'clk' as the system clock. For proper operation of the VME interface, it is recommended that the system clock is 40 Mhz or higher.

pin name	type	description
clk	in	System clock
reset_n	in	Asynchronous system reset, active low

2.2 VME Bus

These pins are used to control data transfer through the VME interface.

pin name	type	description
vme_addr[23:1]	in	VME address bus input
vme_am[5:0]	in	VME address modifier input
vme_data_in[31:0]	in	VME data bus input (from bus driver)
vme_data_out [31:0]	out	VME data bus output (goes to bus driver)
vme_ext_drv_n	in	Active low drive enable signal for external bidirectional data bus drivers.
vme_int_drv_n	in	Active low drive enable signal for internal bidirectional data bus drivers.
vme_ext_ddir	in	Direction control signal for external bidirectional data bus drivers: '1' to VME bus '0' from VME bus
vme_lword_n	in	VME long word access indicator, low active
vme_dtack	out	Data transfer acknowledge. Used to inidcate whether the DTACK is drive low or high (for rescinding)
vme_dtack_ebl_n	out	Data transfer acknowledge driver output, active low. This is the enable signal of the external DTACK driver.
vme_as_n	in	VME address strobe: clocks with falling edge the internal synchronisation signals like vme_addr and vme_am. vme_as_n is also used as data signal for access start detection.
vme_ds0_n	in	Data strobes 0, active low
vme_ds1_n	in	Data strobes 1, active low
vme_write_n	in	Read/write signal, active low
vme_iack_n	in	Interrupt acknowledge, active low
vme_iack_in_n	in	Interrupt acknowledge daisy chain, active low

pin name	type	description
vme_iack_out_n	out	Interrupt acknowledge daisy chain, active low
vme_irq_n[6:0]	out	Interrupt, active low. Have to be connected to open collector driver.

2.3 Configuration

All configuration signals are static. They can be either tied to GND/VCC or driven by a configuration register.

pin name	type	description
config.rsc_dtack	in	Rescinding DTACK enable.

2.4 User Side Interface

This paragraph describes the user side interface, which is designed for easy register and memory access.

2.4.1 Signal description

The entire user side interface is part of the 40 Mhz clock domain. The application dependent address modifier logic is not integrated in the core. This allows the user to build his own address decoding logic without changing the code of the VME. For convenience, more signals are provided than a minimal approach would require. This simplifies the design of the user side logic. Timing information can be found in paragraph *3 Timing.*

pin name	type	description
int_user_addr(23:1)	out	Registered VME address bus (by falling edge of vme_as_n)
int_user_am(5:0)	out	Registered VME address bus modifier (by falling edge of vme_as_n)
user_access	in	User access signal. The user has 50 ns time to decode the address and asserting the user_access signal when addressed.
user_acc_req	out	User access request: Active high until user_acc_rdy acknowledges the request (or VME bus error occurs)
user_acc_rdy	in	User side acknowledgement signal. Active one event which finishes user side access.
user_addr(23:1)	out	Registered VME address bus
user_am(5:0)	out	Registered VME address bus modifier
user_wr_data(31:0)	out	Write data bus.
user_rd_data(31:0)	in	Read data bus. Must be valid when user_acc_rdy is 1.
user_rw_n	out	Data read/write_not signal. ‘1’ : read data ‘0’ : write data

pin name	type	description
user_we	out	User data write byte enable signal. Indicates which byte of the user_wr_data bus is valid. Bit (0): user_wr_data(7:0) is valid Bit (1): user_wr_data(15:8) is valid Bit (2): user_wr_data(23:16) is valid Bit (3): user_wr_data(31:24) is valid
user_rd	out	User data read byte enable signal. An active one indicates which byte the VME-read requests. Bit (0): user_rd_data(7:0) is requested Bit (1): user_rd_data(15:8) is requested Bit (2): user_rd_data(23:16) is requested Bit (3): user_rd_data(31:24) is requested
user_ireq	in	Interrupt request. Active one indicates that an interrupt is pending and a VME interrupt will be generated. Must return to zero with user_iack = 1.
user_iack	out	Interrupt acknowledgement. An active one event indicates the end of a valid interrupt acknowledge cycle.
user_ilev(2:0)	in	Interrupt level
user_ivec(7:0)	in	Interrupt vector

2.4.2 User address decoding

To decide if an access is for the actual card or not, a decode logic has to be added externally. The signals int_user_addr and int_user_am allow a standard memory mapped address decoding scheme. The address modifiers may be used to differentiate between user / system access, io or memory etc. It is not mandatory to use the address modifiers.

The int_user_addr and int_user_am signals are latched with the falling edge of vme_as_n and are immediately valid. The user address decode logic has to be designed for a delay time shorter than 50ns (2 cycle path). This gives more flexibility for the user and lowers the constraints for this part of logic.

3 Timing

This paragraph describes the timing of the iniVME core.

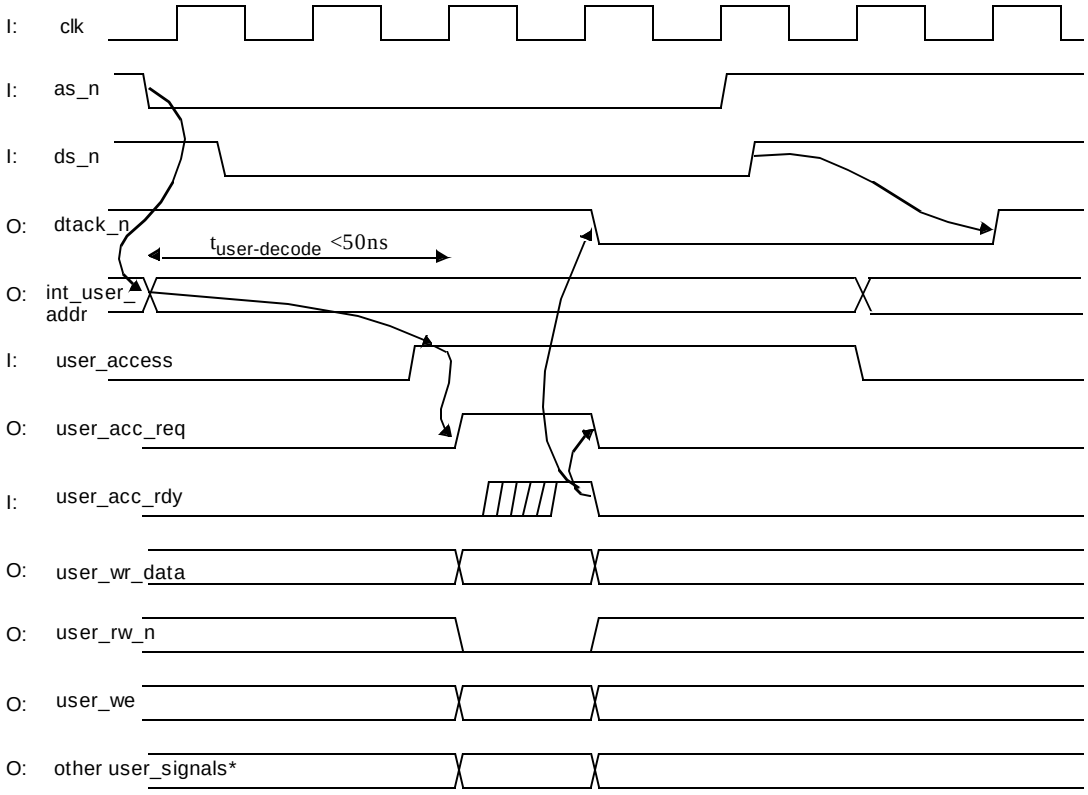
3.1 General Information

All VMEbus inputs are synchronized to the local clock domain. A frequency of 40 MHz is required to detect all level changes on the VME bus correctly.

Outputs are all synchronous to the VME clock domain. Only int_user_addr and int_user_am are registered by the falling edge of vme_as_n signal.

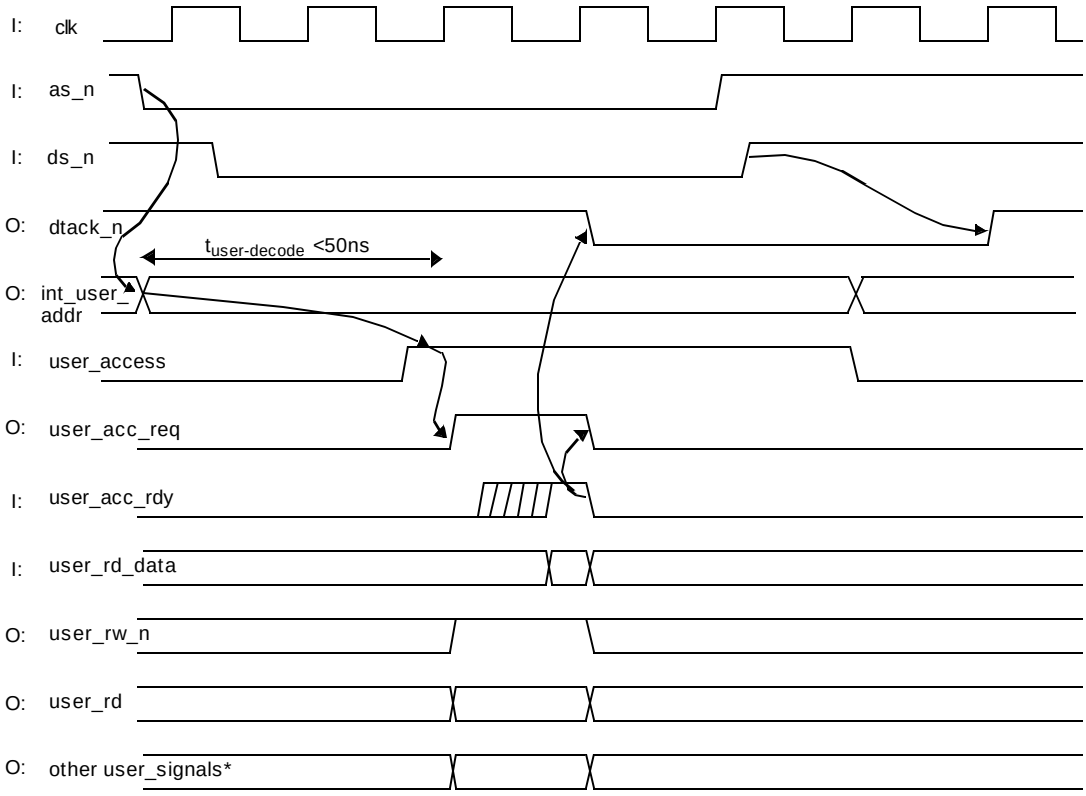
3.2 User access timing

3.2.1 Write access, no user waitstates



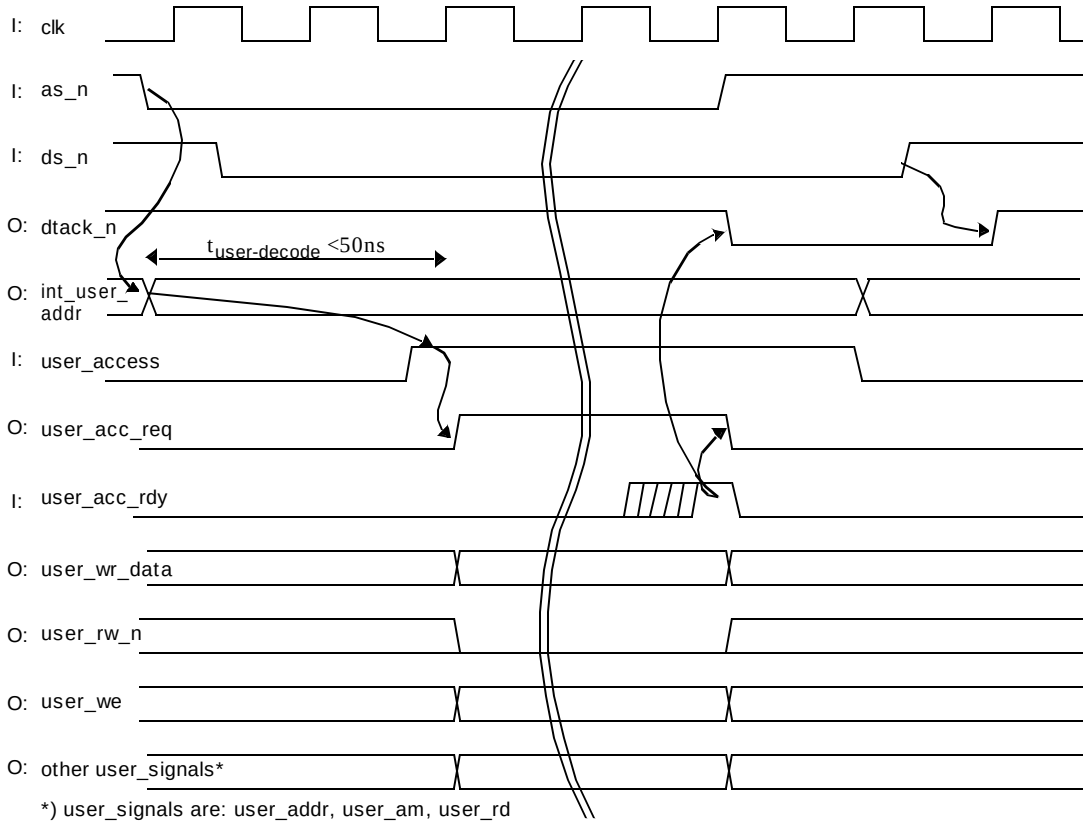
*) user_signals are: user_addr, user_am, user_rd

3.2.2 Read access, no user waitstates



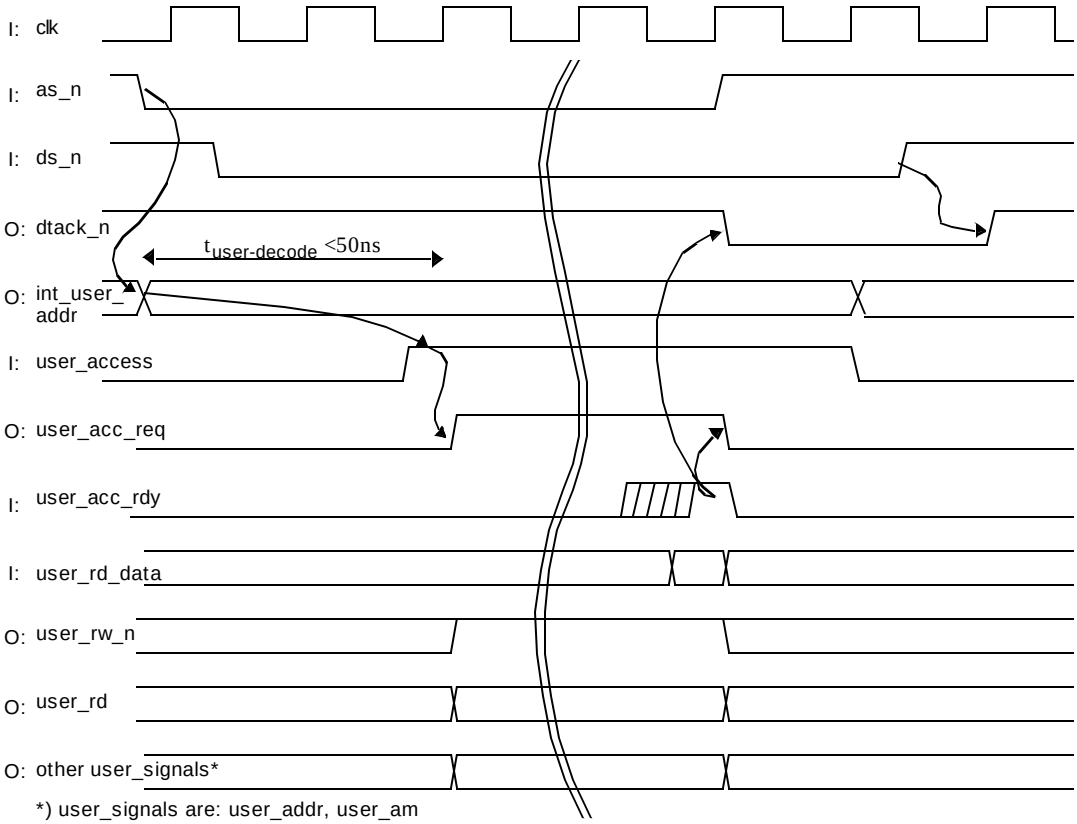
*) user_signals are: user_addr, user_am, user_we

3.2.3 Write access, user waitstates



*) user_signals are: user_addr, user_am, user_rd

3.2.4 Read access, user waitstates



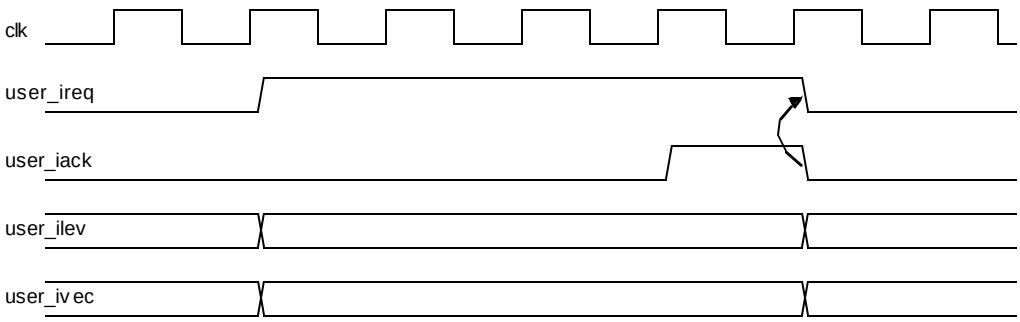
3.2.5 Interrupt Timing

Interrupt requests to the VME bus are signalled by the active high user_ireq. Depending on the user_ilev (interrupt level), the vme_irq_n(x) will be asserted. As soon as the interrupt is acknowledged by the VME bus, the user_iack event is asserted. If the interrupter uses the ROAK (Release On AcKnowledge) scheme, then the user_ireq has to be released immediately. If it uses the RORA (Release on Register Access) scheme then the user_ireq has to be released when the interrupt source is acknowledged.

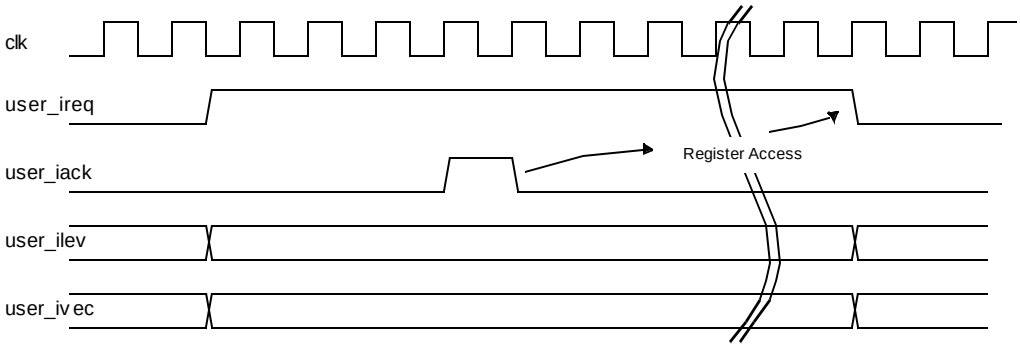
The interrupt vector is 8-bit. It responds to D08(O), D16 and D32 interrupt cycles.

user_ilev and user_ivec have to be stable for the whole time period where user_ireq is high.

Timing using ROAK scheme:



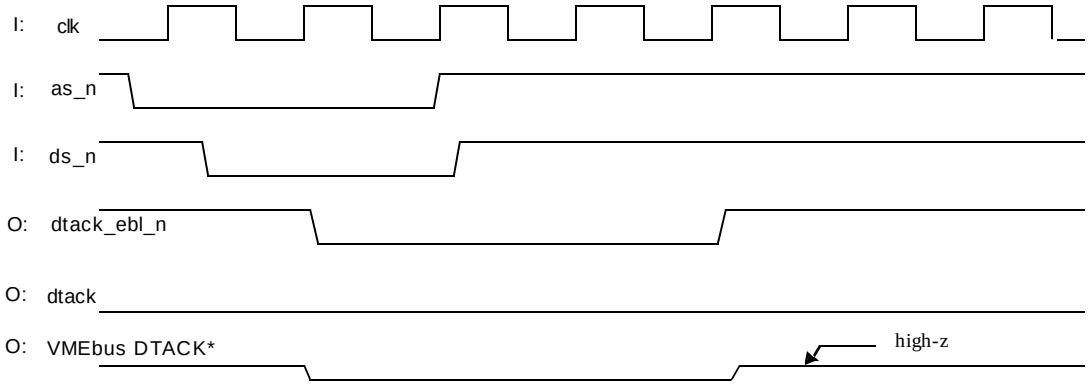
Timing using RORA scheme:



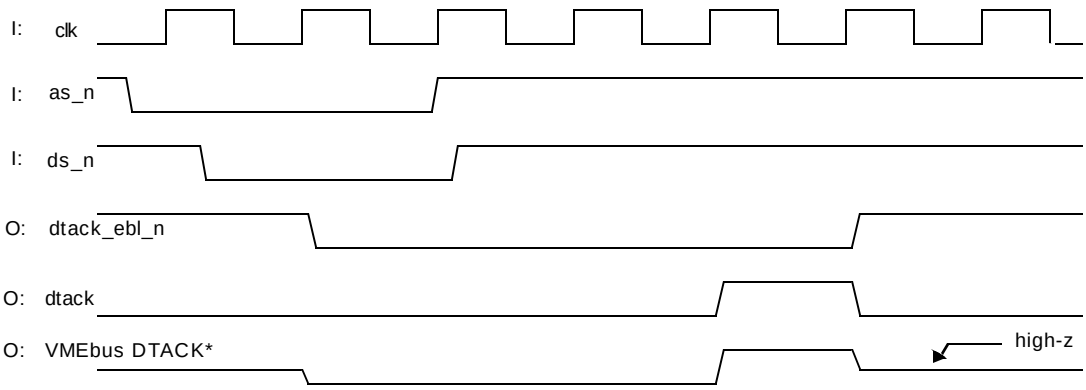
3.2.6 Rescinding DTACK

The VME64 specification allows DTACK to be operated as a rescinding signal instead of an open-collector class signal. This results in an accelerated bus cycle. This feature can be selected through config.rsc_dtack = ‘1’.

Timing diagram with open-collector DTACK:



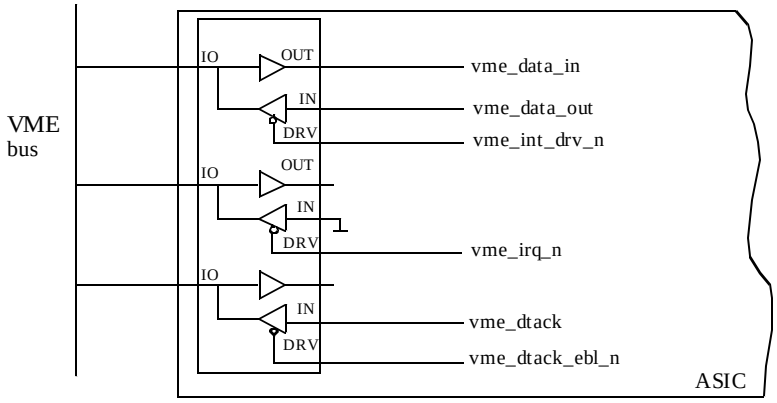
Timing diagram with rescinding DTACK:



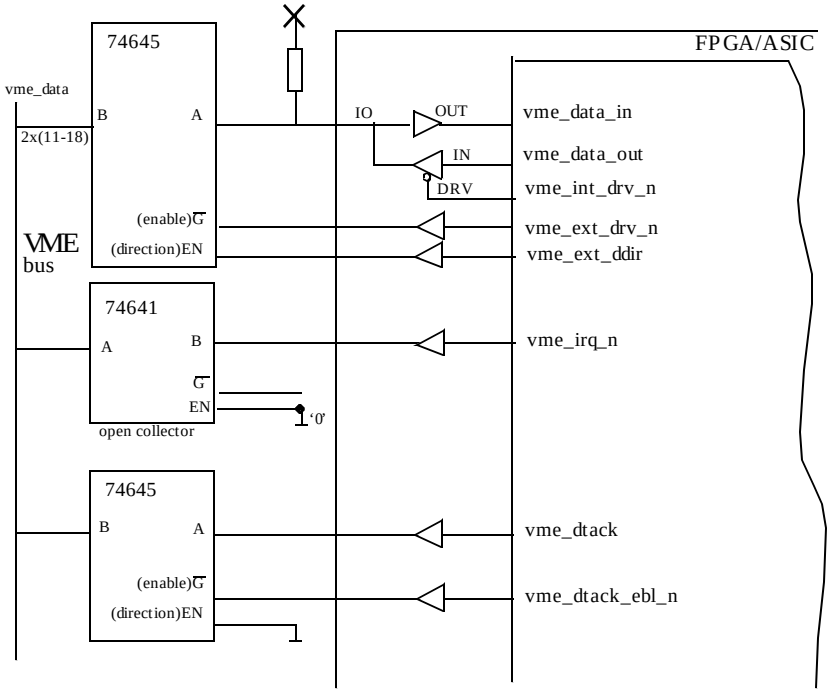
4 Applications

The VME specification requires 64 mA bus drivers. The bus drivers are not integrated in the VME core. Since the FPGA are not able to drive this load, external drivers might be used. The driver should have the following functions:

This diagram shows how internal buffers can be used to directly interface with the VME bus. These drivers have to fulfil VME specifications.



This diagram shows how external standard parts can be used to interface with the VME bus. The used 74xx parts have to fulfill VME specifications. Please note that outputs like vme_irq_n etc. are push-pull outputs and require and external open collector driver like the 74641.



If rescinding DTACK is not required than DTACK can be driven by an open collector driver as is vme_irq_n.