GEB-VME
FIRMWARE SPECIFICATION
VPC-QP192-BS
EDITION

DOCUMENT REVISION HISTORY

Rev.	Date	Description
A	07/02/2017	First Release

Emission Date	07/02/2017	Resp. PRG		Resp. RSQ	

INDEX

INDEX of FIGUREs

INDEX of TABLEs

1.Introduction

VPC-QP192-BS is VME compliant board has been designed to be hosted in test system that uses in the Virginia Panel interconnection system. The board is equipped with with Virginia Panel QP192 connector, an Intel (Altera) Fpga, 1xRS232 Interfaces, Local DRAM, 1149.1 Jtag TAP. The onboard firmware makes available a set of simple command, such as VmeRead or VmeWrite with many addresses and data modes, or complex commando such as memory tests.

1.1. Features

The FPGA VME logic includes the following Capabilities:

- VME Master interface A32-A24-A16, D32-D16-D8E-D8O
- VME system controller
- VME slave interface, A32-A24-A16, D32-D16-D8E-D8O
- VME Interrupter
- VME Interrupt handler
- Standard and Special (Custom) configuration space master and salve capability
- DRAM Controller
- NIOS CPU with User Monitor allows test commands from RS232/USB
  - Simple Read and Write cycles, including interrupt acknowledge cycles
  - Complex Macro command such as Memory tests.

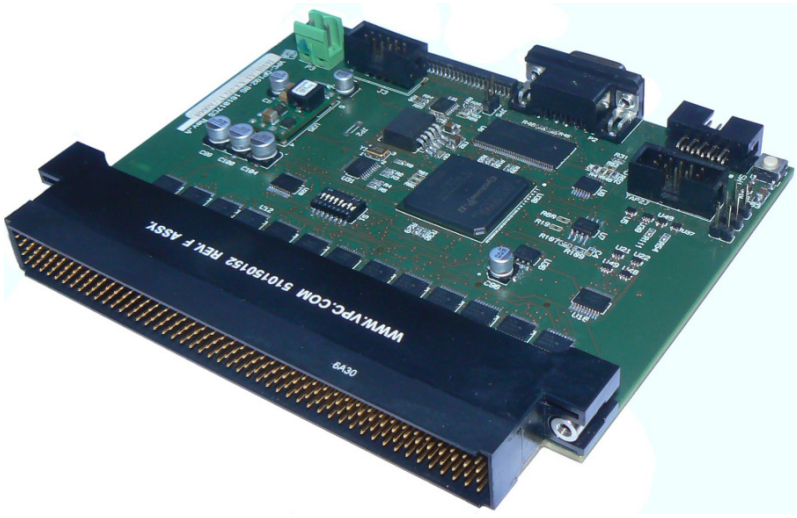

Figure 1 VPC-QP192-BS Board

1.2. Application, Functional and BSCAN Testing

The VPC-QP192-BS board can operate in a test environment can operate both in functional both in BSCAN mode.

In functional test mode, command and response can be managed by RS232 or USB interfaces. The on board MPU will decode the command, will perform the actions, and will send back the results. Commands can be oriented to do simple operation, such as VmeReadm VmeWrite or VmeCheck, or complex, such as Memory tests. Both of them can be performed at 8, 16 o 32 bits. Some low level command are dedicated to interrupt handling, they can be used to generate interrupts, check interrupt status, and perform interrupt ack cycles at any level. API to interface VPC-QP192-BS commands to common used tests enviroments are available. Special cycles, with user defined address modifyers, are allowed to perform to initalization of VME modules configuration space.

BSCAN tools and Altera VJTAG IP can be used to access to all resources, such as registers or memories, onboard and on VME Bus. This function allows BSCAN test operations at test speed using BSCAN test tools, or generate single VME cycle at full speed, using Jtag Functional Test tools, typically written in Python Language. The interface between the BSCAN tools could be done through a standard 1149.1 TAP connector using a high performance external BSCAN controller or through an USB port using an onboard, embedded, medium performances BSCAN controller.

1.3. Board and Fpga architecture

The GEB-VME board hosts the control logic in an Fpga; it makes available the modular VME IP, granting large flexbility and growtly if this will be neededmto in the user test application. The board hosts the vme buffers and receiver fully compatible with vme driving capability, being the board to one ends of the bus the default configuration host the VME termination

The board hosts a Sdram that can be used how target in vme slave cycles, the slave address translation block (TLB), windows base address and size registers (BARS) allow allocation of local Sdram everywhere in Vme addressing space.

Test Master is a bus cycles generator that allows an indirect access, through the VME Master, to everywhere in VME Addressing space. Consequently, due to a test commands, the vme bus parameters will be stored in some registers before start the cycle.

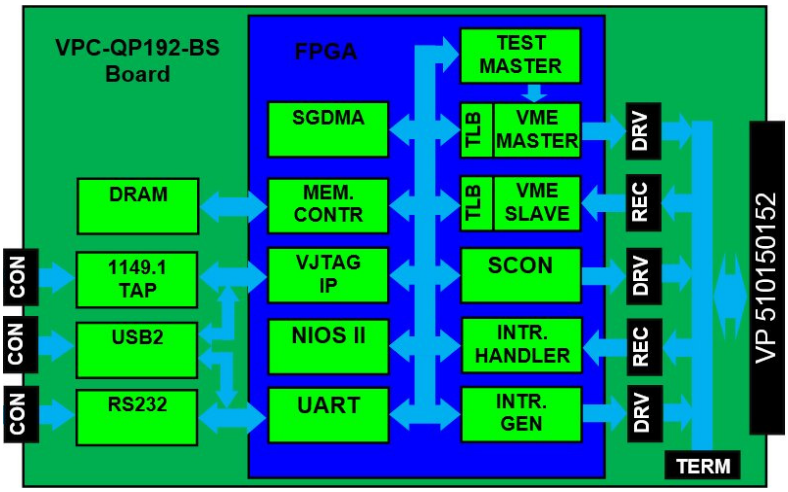

Figure 2 VPC-QP192-BS Block Diagram

Areas in Vme addressing space can be locally mapped through the Master TLB, consequently the NIOS Cpu can access directly in a VME memory area, without reprogramming any registers, at its full speed, gaining the capability of perform memory tests at high speed.

The interrupter function can activate any one of the seven available interrupt lines and can accept interrupt acknowledge at correspondent IPL, giving back the interrupt vector. The vector can be programmed in a register.

The interrupt handler function is able to read the status of the interrupt lines, making it available to Interrupt read status commands. The interrupt handler can also perform interrupt acknowledge cycles at any IPL level to reset interrupt source and get its vectors.

The VPC-QP192-BS board supports non-standard VME cycles also, it, using some user defined Address Modifiers, is able to address non-configuration registers implemented in some special functiobs board. This metohd is an ancestor of CSR whose definition began in the VME64 specifications, it's used sometime in some VME derived bus, such as the avionic MCS bus.

2.Resident Standard Firmware

This section describes the command that you can execute with the firmware, programmed on the board GEB-VME.

2.1. Serial Port Setting

The following table reports the setting for the correct communication with the GEB-VME Firmware.

speed	57600
Nbit	8
parity	none
bit stop	1
flow control	XON/XOFF

Table 1 Serial Port Settings

Emulation	VT100
local echo	Off
new-line Tx/Rx	CR

Table 2 Terminal Settings

2.2. Usage Notes Command Console

In this section are reported the general rules to using correctly the command console:

- ALL the parameters are interpreted, unless otherwise indicated, as numbers in hexadecimal format (HEX), even if you do not enter the "0x" initial.
- The parameters separator is the “blank” character.
- At the end of each command the cursor “>” is printed.
- You cannot enter a new command before the end of the previous.
- Where provided, at the end of the commands, a "status" string is printed.
  This string report the state of VME bus for the operation performed.
  The possible states are:
  - STATUS DONE: The VME Bus cycle was successful
  - WRONG ADD: The address entered is invalid
  - BUS ERR: The VME Bus cycle is terminated with error
- The command that perform the data check, report:
  - The VME Bus Status string
  - The result of verify, with the data read on Bus and the data using for check
- The commands with equal root in the name, perform the same function with different data size, defined by the final letter of the command, according to the following table:
  - “B”: Byte (8 Bit)
  - “W”: Word (16 Bit)
  - “D”: Double word (32 Bit)

2.3. Generic Command List

The following table reported the list of the command that you can run, with the firmware:

Command	Parameter	Description
h	----	Print the command list
lst	----	Display the history commands
echo	<0|1>	Enable (1) or Disable (0) the echo of terminal for the command and prompt
vrst	----	Perform a reset of VME Bus
vint	<AM> <SD> <IS> <IM> <TV> <BS>	Perform the initialization of the BusTestStsReg: AM: Arbitration Mode; SD: Swap Enable/Disable IS: Interrupter Line Selection; IM: Interrupter Mode; TV: Timeout value; BR: Bus Request Level. At the end of command is reported the value of the BusTestStsReg reared Example: BusTestStsReg ReRead: 0x80020003
vam	<AM>	Set the address modifier of VME bus
vmb vmw vmd	<Add> <Data>	Write the data <*Data*> at address <*Add*>. At the end of command is reported the status string. Example: VM: Add 0x0000057E Data 0x55 --> STATUS DONE.
vdb vdw, vdd	<Add> <Len>	Dump an area of a <*Len*> length of the memory from the address <*Add*>. The data are formatted according to <*Len*> parameter: <*Len*> = 1: report a single string with the read data and the bus status string. <*Len*> > 1: report a table with the read data without the bus status string for each data read
vfb vfw vfd	<Add> <Data> <Len>	Fill an area of a <*Len*> length of the memory from the address <*Add*>, with the <*Data*> data. For each data written is reported the status bus string
vcb vcw vcd	<Add> <Data> <Mask>	Read and check of data <Data> at address <Add> with a mask <Mask>. At the end of command is reported the result of operation and the bus status string.

Command	Parameter	Description
tvab taw	<Add> <Len>	Execute the address bus test. The test performs the algorithm of 1 walking for the address bus: do(i=0 to <*Len*>) <*Add*> = <*Add*> xor (1<<I) The data written at first location is 0x55 if byte, or is 0x5555 if word. In the next locations is written the value of "i".
tdb tdw	<Add>	Perform the test of 1 walking algorithm to data bus at the address <*Add*>
tsb tsw	<Add> <Len>	Execute a memory stress test. Have performed a <*Len*> length number of write and read from the address <*Add*> Note: initialization of the registers that permitted the memory access is the user's responsibility.
vpe	----	Print Amount of errors that occurred in tests.
vce	----	Reset the amount errors variable.
sbcr	<0-3>	Print the selected Slave BAR configuration.
sbcw	<0-3> <Add> <AM> <AMNo> <AdrMode> <AdrSize> <LAR>	Write the selected Slave BAR configuration: <Add>: base address <AM>: address modifier <AMNo>: address modifier index <AdrMode>: addressing mode <AdrSize>: size of address space ($2^{AdrSize}$= size in byte) <LAR>: local base address for slave.
mbcr	<0-3>	Print the selected Master BAR configuration.
mbcw	<0-3> <Add> <AM> <AMNo> <AdrMode> <AdrSize> <LAR>	Write the selected Master BAR configuration: <Add>: base address <AM>: address modifier <AMNo>: address modifier index <AdrMode>: addressing mode <AdrSize>: size of address space ($2^{AdrSize}$= size in byte) <LAR>: local base address for slave.
mvmb mvmw mvmd	<Add> <Data>	The command are the same as [vmb, vmw, vmd], but are performed using the VME Master core instead of TestBus Core.
mvdb mvdw mvdd	<Add> <Len>	The command are the same as [vdb, vdw, vdd], but are performed using the VME Master core instead of TestBus Core.
mvfb mvfw mvfd	<Add> <Data> <Len>	The command are the same as [vfb, vfw, vfd], but are performed using the VME Master core instead of TestBus Core.

Command	Parameter	Description
mvcb mvcw mvcd	<Add> <Data> <Mask>	The command are the same as [vcb, vcw, vcd], but are performed using the VME Master core instead of TestBus Core.
swap	----	Enable the byte swapping for the commands “vmw”, “vmd”, “vfw”, “vfd”, “vdw” and “vdd”. Return the message “Swap Enabled”
noswap	----	Disable the byte swapping
stsswap	----	Print the swap status
Commands for the optional IOBUS IP		
Command	**Parameter**	**Description**
iocsrw	<Data>	Init with Data of IOBUSs CSR Register
iocsrr	----	Dump of IOBUSs CSG Register
iomw	<Data>	Write the Data at the address Add
iomr	----	Read the data in address Add

Table 3 Generic Command List

3.Firmware commands example

The Firmware loaded in the GEB-VME Board permits to access to the VME-IP by using the commands explained in the Table 3.

We have connected a GEB-VME board as a Master VME to another GEB-VME Board as a SLAVEVME by using an apposite Motherboard. Both the Boards are programmed with the SYS-N2-VME-64-IOBUS system

Below the commands sequence (for the vme registers, refer to the document "130911UM_Rev.A.3_(VMEIP User Manual)"):

//***** TEST WITH TWO GEB VME, ONE USED AS MASTER AND THE OTHER USED AS SLAVE ******

//GA MASTER: 0x1; Geografic address forced by the dedicated dip-switche on the GEB-VME MASTER

//GA SLAVE: 0x2; Geografic address forced by the dedicated dip-switche on the GEB-VME SLAVE

//set CSR Mode

vam 2f

//*****MASTER ACCESS*******

// the base address to access to the Master is 0x80000 (80000 because A23:A19 represents the GA, that for the MASTER //is 0x1)

//Read IP Version Register Expected Value: 0x4417

vdd 803FC 1

//Read GA MASTER (bits 24:20)

vdd 80310 1

//*****SLAVE ACCESS******

// the base address to access to the Master is 0x100000 (100000 because A23:A19 represents the GA, that for the SLAVE is 0x2)

//Read GA SLAVE (bits 24:20)

vdd 100310 1

//Write BAR0Addr (BAR programming) to set the Slave Base Address

vmd 100100 12000000

//Reread the BAR value previously written

vdd 100100 1

//Program A32 supervisory data access and A32 non privileged (user) data access

vmd 100104 8d89

```
//Reread
vdd 100104 1
//Programming of the BAR to access to the on chip memory in the system
//Write ADRMode&BARSize (MSb of the on chip ram)
vmd 100108 6C  //0x6C => D[7:5] = 0x3 (A32 Address Mode); D[4:0] = 0xC (12 bits to obtain 16 Mbits)
//Reread
vdd 100108 1
//Write Local Address Register (LAR) of the on chip ram in the sopc system (0x02000)
vmd 10010C 2000
//Reread
vdd 10010C 1
//Try to reread all the previous registers written
vdd 100100 4
//
//
//set User Mode to access to the local resources of the system (on chip memory)
vam 0D
//Write/Read from Slave addrees programmed
vmd 12000000 0 //first valid address
vmd 12000ffc 0 //last valid address (16Mbyte offset)
vmd 12001000 0 //first invalid address (BUS ERROR)
//Memory test (on chip ram)
tsw 12000000 1000 1  //(size=0x1000 (4096))
//
//**** IOBUS  ***
//Set the Register in CSR Mode to access to the IOBUS in the system
//!vam 2f
//Programming of the BAR to access to the IOBUS in the system
//Write ADRMode&BARSize (MSb of the IOBUS)
vmd 100108 72  //0x72 => D[7:5] = 0x3 (A32 Address Mode); D[4:0] = 0x12 (18 bits to obtain 262144 bits (7FFFF-40000))
//Write Local Address Register (LAR) of the IOBUS mem in the sopc system (0x40000)
//!vmd 10010C 40000
```

```
//Try to reread all the previous registers written
//!vdd 100100 4
//set User Mode to access to the local resources of the system (IOBUS)
//!vam 0D
//Perform an IOBUS write cycle
//!vmw 12000000 0
//Perform an IOBUS read cycle
//!vdw 12000000 1
```

NOTE The charapters "//" are seen by the firmware as the beginning of a comment