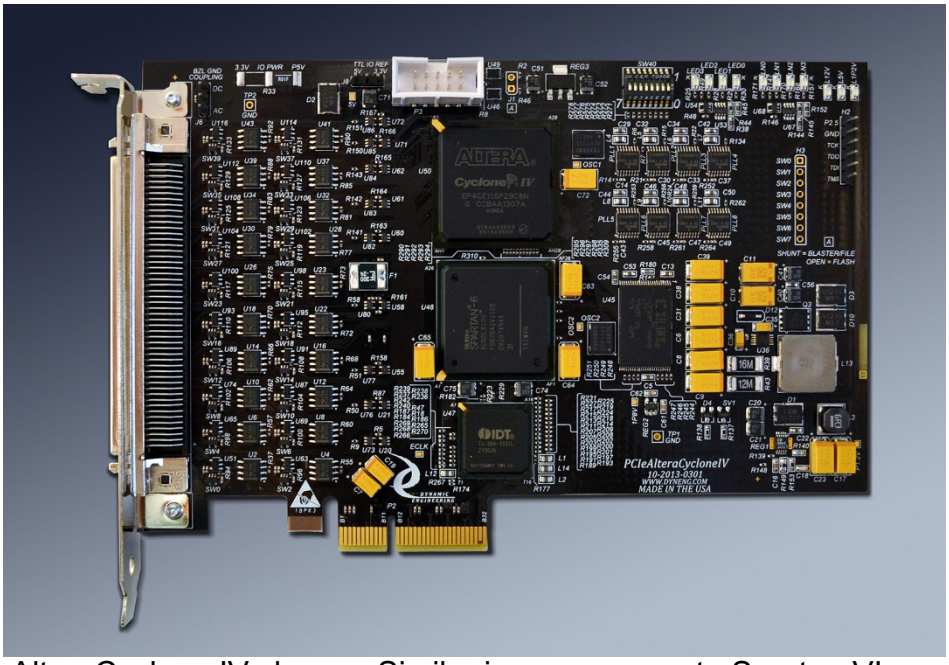
DYNAMIC ENGINEERING
150 DuBois, Suite B/C Santa Cruz, CA 95060
(831) 457-8891
www.dyneng.com
sales@dyneng.com
Est. 1988

User Manual

PCIe-Spartan-VI

Re-configurable Logic
with RS-485/LVDS and TTL IO



PCIeAlteraCyclone IV shown. Similar in appearance to Spartan VI version.

Revision 1p0
Corresponding Hardware:
Fab Number: 10-2023-0701

PCIe-Spartan-VI

Re-configurable Logic
PCIe Module
Dynamic Engineering
150 DuBois St. Suite B/C, Santa Cruz CA 95060
831-457-8891

This product has been designed to operate as a PCIe Module and compatible user-provided equipment. Connection of incompatible hardware is likely to cause serious damage.

Table of Contents

List of Figures

Product Description

PCIe-Spartan-VI is part of the PCIe Compatible family of modular I/O components. PCIe-Spartan-VI provides a user configurable Spartan VI FPGA [XC6SLX100-3FGG676I], along with 40 RS-485 or 40 LVDS transceivers, 12 TTL IO, 24 PLL based clock references and FIFO support, full DMA capabilities in a half-length single slot design.

The RS-485 and LVDS parts can be mixed. The standard RS-485 devices are rated for 50 MHz. The standard LVDS parts are rated for 200 MHz.

The PCIe bus implementation is 4 lanes using a [Pericom] PI7C9X130DNDE bridge. The bridge is connected through a Xilinx Bus Controller FPGA to the User Spartan VI FPGA. The Bus Controller provides a GPB [General Purpose Bus] which operates as a 32 bit, 50 MHz connection for programming registers and other set-up / status operation. In addition, there are 16 unidirectional data lanes arranged as 8 bidirectional pairs to support transmit and receive operations between the User and Controller FPGAs.

The Bus Controller supports multiple channels of data-flow to the User FPGA with FIFOs. There are 8 transmit and 8 receive FIFO paths. The “transmit” path is loaded using DMA transfers to read data from the host into the Bus Controller memory. Local transfer engines move the data to the User FPGA side using FIFO status to control the transfer. With 8 DMA controllers each data path is separate and can operate in parallel. The “receive” path works in reverse with transfer engines in the User FPGA moving data into the Bus Controller using the FIFO status to control the transfer. Data is DMA transferred from the local FIFO within the Bus Controller to host memory. 8 DMA engines are available to manage the receive direction allowing for 8 ports of full duplex operation. Within the Bus Controller is a local arbitrator to control access to the intermediate bus. The controller is automatic – no software required.

The data is moved between the two FPGAs with a “push”, the target side provides a FULL and Almost Full status to the data mover. The data transfer engine pushes data from the local FIFO to the target FIFO. When both the local and target FIFOs are not almost empty and not almost full respectively; the data is moved in a burst. 1 byte per clock. Due to pipelining, there are delays from write to current status being available to the control engine. When Master and or Target is “almost” (Full or Empty) the data transfer engine slows down to get exact status.

PCIe-Spartan-VI has drivers for Windows, and Linux. Both provide an automatic facility to reprogram the bridge for improved performance. Both come with a sample user application to demonstrate the use of the various HW capabilities. The VHDL for the User FPGA reference design is also available. The reference design can be modified to perform your unique function. The reference design has 8 ports plus a base function.

Built in support for internal and external loop-back of the data lanes is provided. The external loop-back [RS485 or LVDS] is in the form of a nibble wide data port with direction control[internal] and strobe. Your state machine can be substituted for the IO controller in the reference design.

There are 8 channels in the reference design. If you need more IO per channel and fewer channels or some other mix the VHDL can be modified. The GPB is supported with IOCTL style support in the driver allowing for user redefinition of the memory map within the User FPGA.

The reference software demonstrates DMA operation through the external IO and internal BIT tests. In addition, non-DMA versions are provided.

The base reference design has a controller for programming the PLLs. The reference software shows how to program the interface. Automatic software for taking the Cypress .JED file, parsing and loading the PLL.

The base reference design also has a simple register based, interface for the TTL IO.

An 8-bit "dip switch" is provided on the PCIe-Spartan-VI-485/LVDS. The switch configuration is readable via a register. The switch is for user-defined purposes. We envision the switch being used for software configuration control, PCIe board identification or test purposes.

LEDs are provided on board for user-defined purposes. The User FPGA has 4 LEDs, which can be used for whatever purpose the user desires. We have used the LEDs for test / debugging purposes – handy indications of what the hardware state is or what the current software process is. The LEDs in our reference design default to a non zero pattern to show the FLASH file has loaded properly. The register controlling the LEDs can overwrite the default.

In addition LEDs are provided to indicate that the regulators are operating properly

The UserAp reference software can load a User FPGA design file into the User FPGA at any time. UserAp also contains a function to reload from FLASH to allow the user to toggle back and forth between design versions rapidly. User Designs can also be programmed into FLASH with the programming tool [Impact in this case].

The User FPGA will be loaded on power transitions automatically from FLASH. Software can overwrite.

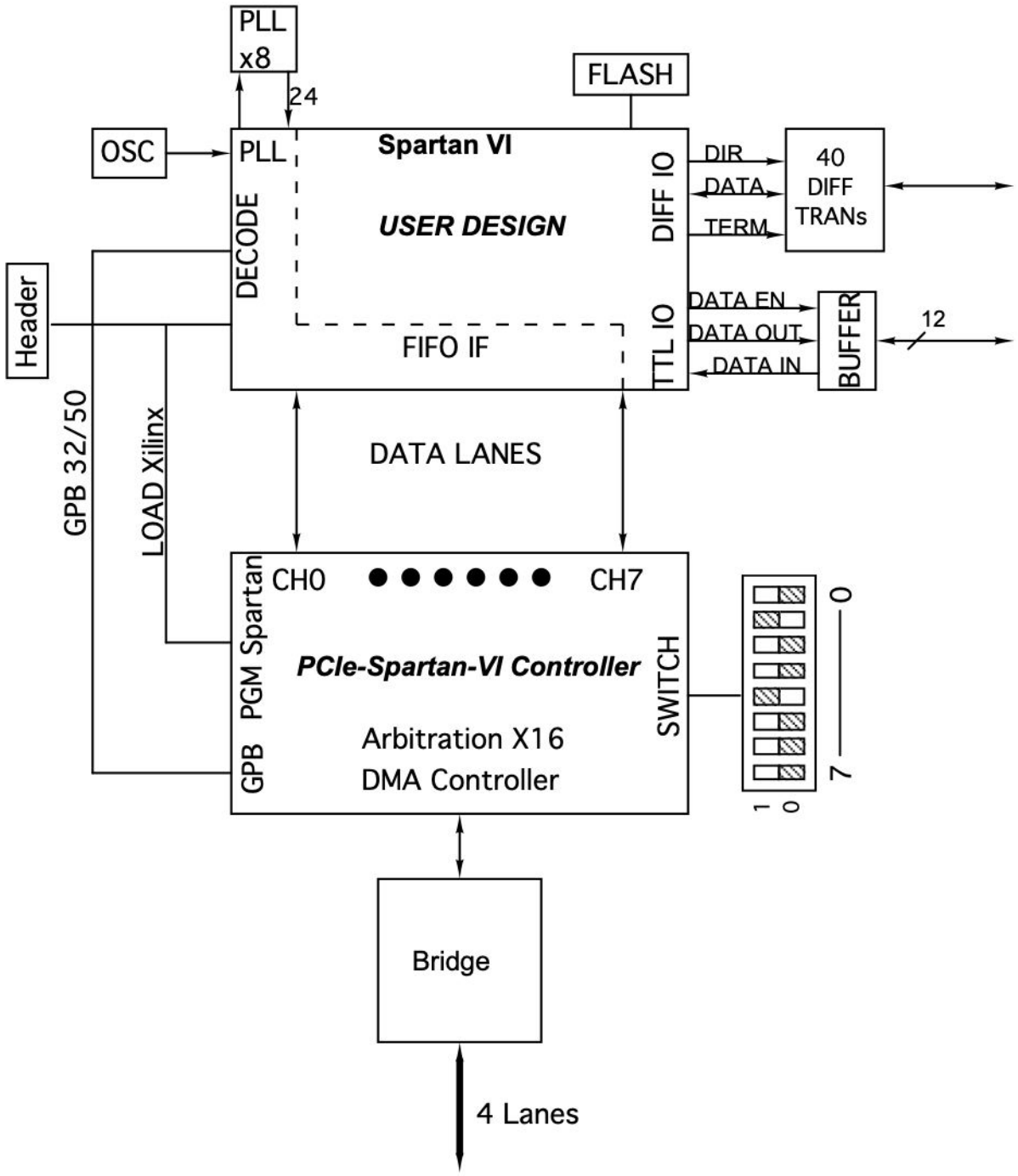


Figure 1 PCIe-Spartan-VI Block diagram

PCIe-Spartan-VI-485/LVDS has both RS-485/LVDS and TTL IO supported by a D100 connector. There are 40 un-committed RS-485 or LVDS IO. The standard RS-485 device supports 50 MHz operation. The LVDS parts have much larger bandwidth, 200 MHz receive and 400+ MHz transmit. The transceivers are designed for multi-drop

LVDS operation. The SN65MLVD201D is the part in the current design.

The 485/LVDS lines are routed as differential pairs with matched lengths and impedance control. The lengths are matched from the connector edge to the ball on the User FPGA. The right-angle connector is used for the matched length calculations. With the vertical connector option, the lengths will be slightly off. The differential signals are supported with programmable terminations. The terminations can be programmed from the User FPGA to be active or open.

The TTL IO is supported with open drain drivers with pull-ups. The receive side is also buffered to protect the User FPGA device. The pull-ups are referenced to a shunt selectable 3.3V or 5V. Data Out, Data In, and Data Enable controls for each IO. If Data Enable is programed to switch with Data Out the device will act as open drain. If set to Enabled or Disabled and Data Out operated independently the interface will be push-pull with 24mA source and sink for higher bandwidth than the open drain configuration.

The reference design has a pin configuration file, which can be reused for your specific implementation. The reference design is written in VHDL. For initial development and potentially as part of your system the D100 cable and the HDEterm100 are recommended. The HDEterm100 serves as a breakout from the cable to screw terminal block. The HDEterm100 has matched length, differential routing and several termination options that can be installed. For more information on the HDEterm100 please visit the web page https://www.dyneng.com/HDEterm100.html

Clocking options in the User FPGA are supported with 8 PLL devices. The PLLs are programmable with a frequency description file. The PLLs are programmed via the User FPGA. The reference design has an interface to the PLLs implemented along with software to load the PLLs with the requested frequency file.

The User FPGA is connected to the 50 MHz GPB clock and 133.33 MHz oscillator plus the PLLs. Locally generated frequencies or one of the clock inputs can be routed to the PLLs. The reference design divides the 133.33 to provide 66.666 to the PLLs. The references can be unique if desired. The PLLs are Cypress 22393 devices. Cypress has a utility available for calculating the frequency control words for the PLLs. The PLLs respond to one of two addresses [only one works]. Our test SW “discovers” the working address and then uses that address to program the PLL, read-back, and run frequency tests.

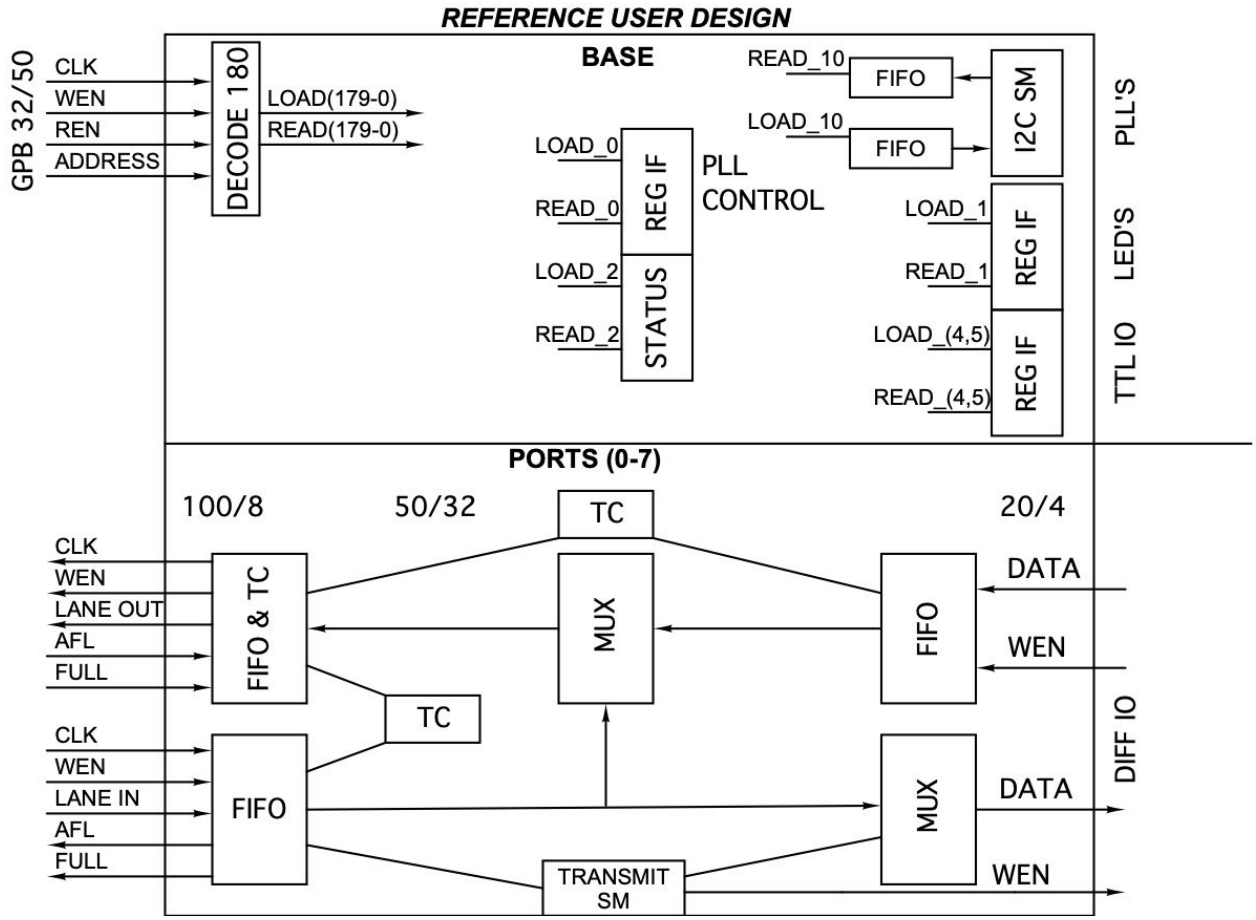


Figure 2 PCIe-Spartan-VI Reference design Block diagram

Theory of Operation

A wide range of interfaces and protocols can be implemented with PCIe-Spartan-VI. UART, Manchester encoding, serial or parallel, RS-422/485 or TTL, custom. The interfaces can be created using the hardware and development tools provided with the PCIe-Spartan-VI along with the AMD/Xilinx software.

The VHDL for the reference design is available The reference design was done with the ISE 14.7. The reference design is intended to be the starting point for a client design. The data lanes, GPB and other interfaces are implemented to allow the client to focus on the IO interface. The drivers for PCIe-Spartan-VI provide DMA and target accesses to the HW. The GPB interface is designed to use a passed in address rather than one hard coded in the driver so the user can change the memory map and not break the driver. The driver does put the address offset in so the client only needs to add the local [User] offset to create the pointer to the register to read or write.

8 channels are implemented each with a pair of data lanes attached. The reference design uses the 40 differential IO – 5 per channel. A simple nibble wide data transfer engine with strobe is implemented. The local 20 MHz clock [internal DCM] is used. The user design can remove the “end” code from the channel VHDL and replace with your state-machine etc. If you need different IO counts or Channel counts etc. a little editing will get you there. The code is commented, and the “find” command will help to navigate to the origin of signals etc. Nothing is “black boxed” so you can see what is being done and make changes as desired. While the VHDL is copyright to Dynamic Engineering, it is a purchase once and use many license. It is intended to be used, and reused by our clients. We want our clients to succeed, and try to keep the path to success open.

Once your requirements are known the design can be implemented with VHDL, Verilog, or schematics and compiled with the design software. The output file can then be “uploaded” to the User FPGA on PCIe-Spartan-VI. Because the FPGA can be re-loaded, your design can be implemented in phases. You can experiment and test out concepts and partial implementations during the design phase or perhaps simulate other hardware that needs to be implemented.

As an example, consider a serial interface with 8 channels. PCIe-Spartan-VI has 40 differential IO. Enough IO for 8 full duplex channels with a clock reference and two controls per channel. PCIe-Spartan-VI has 16 FIFOs with 8 oriented for transmitting data and 8 for receiving data, plus 8 PLLs. Each channel can be supported completely and independently. The User design could be based on the reference design; in which case the PLL interface, basic FIFO interface, and bus decoding are taken care of in the User FPGA plus all of the functions provided by the Bus Controller. The designer would need to implement the IO interface.

The data flow for transmission is Host memory transferred into the Bus Controller TX FIFO via DMA transfers. The data is transferred from the Bus Controller holding FIFOs to the User FPGA Holding FIFOs with the transfer control state-machines. Each channel is separately supported. The User FPGA side reports Almost Full and Full to allow the transfer control to burst when the Bus Controller FIFO is not Almost Empty and the User FIFO is not almost full. 100 MHz x8 per channel in each direction. The user state machine would read the data from the FIFO on the User FPGA side and apply the user protocol before transmitting. On the receive side, the data will flow into the User FPGA, be processed to convert to a format suitable for storing, and be written into the associated channels’ Rx FIFO. The data would be pushed from the User to the Bus Controller channel Rx FIFO using similar controls to the TX path. Almost Full and Full flags coming from the Bus Controller in this case. The DMA engine automatically senses the data in the channel FIFO and moves to the host memory [assuming DMA is set-up].

Flow control throughout. Arbitration within the Bus Controller to handle the 16 DMA

engines automatically. Scatter Gather DMA is supported for arbitrarily large data transfers. Large segments are allowed for OS with larger segment sizes [Linux compared to Windows®]. One interrupt per DMA at the end.

Other reference design features: The PLL i2c interface is provided with a state-machine rather than “bit banging”. Two FIFOs are provided to allow a complete PLL programming file to be loaded and transmitted to the intended PLL. PLL programming data can be retrieved from the selected PLL and stored into the receive FIFO. ACK/NAK status can be used to determine the address for the PLL. Reference software in the “UserAp” code that comes with each driver type.

Besides removing the overhead of the CPU doing the bit banging an advantage of this approach is the PLL being programmed more rapidly. The i2c SM can send the bits at the maximum rate the PLL is rated for. It is difficult to manage this in SW resulting in longer than necessary programming times.

The decoder shown in the diagram interfaces with the GPB to generate READ and LOAD pulses. To implement a register, tie the LOAD to the clock enable, DataIn to the D inputs and the reference clock to the clock. To read-back, enable the corresponding mux path with the READ strobe. There are 180 of each type of strobe available. The reference design allocates 0-19 to the base design, and the rest to the channels. Channel 0 using 20-39. Channel 1 using 40-59 etc. 20 – 32 bit decodes per port is frequently enough.

Feature List Current

- User Defined Spartan VI series FPGA
- DMA and target accesses
- 16 FIFO based data lanes – 8 dedicated “RX” and 8 dedicated “TX”
- 40 fully programmable RS485/422 or LVDS IO
- 12 TTL IO
- 8 PLLs each with 3 clocks supplied to Spartan VI
- 8 position "DIP Switch"
- User LEDs
- Power LEDs
- FLASH and SW loading of User FPGA

As Dynamic Engineering adds features to the hardware, we will update the PCIe-Spartan-VI page on the Dynamic Engineering website. If you want some of the new features, and have already purchased hardware, we will support you with a FLASH update.

VendorId = xDCBA
CardId = x0077.

Programming

PCIe-Spartan-VI is tested in a Windows environment. We use the Dynamic Engineering Driver to do the low level accesses to the hardware. We use MS Visual Studio in conjunction with the driver to write our test software. Please consider purchasing the engineering kit for the PCIe-Spartan-VI; the software kit includes our test suite. In addition, Linux reference suites are available

The drivers take care of discovery and the UserAp allows the client to select which installed board is selected for use.

If you are writing your own driver it is suggested to get the engineering kit and the Linux version of the SW. Usually, the defines and perhaps some of the code can be reused in your effort.

The following pages have the memory maps and bit maps for the Bus Controller "fixed" features and User FPGA reference design features. The User FPGA memory map is subject to change as you implement your design.

Address Map

Controller Base Address Map

PcieSpartanVI_BASE_BASE	0x0000 // 0 Base control register
PcieSpartanVI _BASE_USER_SWITCH	0x0004 // 1 User switch read port DIP switch read
PcieSpartanVI _BASE_XILINX_REV	0x0004 // 1 Xilinx revision read port
PcieSpartanVI _BASE_XILINX_DSN	0x0004 // 1 Xilinx Design Number read port
PcieSpartanVI _BASE_STATUS	0x0008 // 2 status Register offset
PcieSpartanVI _BASE_JTAG	0x000C // 3 JTAG interface Port
PcieSpartanVI _CH0	0x0050 // 20 starting address for channel 0
PcieSpartanVI _CH1	0x00A0 // 40 starting address for channel 1
PcieSpartanVI _CH2	0x00F0 // 60 starting address for channel 2
PcieSpartanVI _CH3	0x0140 // 80 starting address for channel 3
PcieSpartanVI _CH4	0x0190 // 100 starting address for channel 4
PcieSpartanVI _CH5	0x01E0 // 120 starting address for channel 5
PcieSpartanVI _CH6	0x0230 // 140 starting address for channel 6
PcieSpartanVI _CH7	0x0280 // 160 starting address for channel 7

Figure 3 PCIe-Spartan-VI Controller Base Address Map

The address map provided is for the local decoding performed within PCIe-Spartan-VI Bus Controller. The addresses are all offsets from a base address. The base address and interrupt level is provided by the host in which the PCIe-Spartan-VI is installed.

Controller Port Address Map

PcieSpartanVI _CHAN_CNTRL	0x00000000 // 0 General control register
PcieSpartanVI _CHAN_STATUS	0x00000004 // 1 Interrupt status port
PcieSpartanVI _CHAN_INT_CLEAR	0x00000004 // 1 Interrupt clear port
PcieSpartanVI _CHAN_WR_DMA_PNTR	0x00000008 // 2 Write DMA dpr physical PCI address
PcieSpartanVI _CHAN_TX_FIFO_COUNT	0x00000008 // 2 Tx FIFO count read port
PcieSpartanVI _CHAN_RD_DMA_PNTR	0x0000000C // 3 Read DMA dpr physical PCI address
PcieSpartanVI _CHAN_RX_FIFO_COUNT	0x0000000C // 3 Rx FIFO count read port
PcieSpartanVI _CHAN_FIFO	0x00000010 // 4 FIFO offset for single word access R/W
PcieSpartanVI _CHAN_TX_AMT	0x00000014 // 5 Tx almost empty count register - used for Urgent and pulsed interrupt
PcieSpartanVI _CHAN_RX_AFL	0x00000018 // 6 Rx almost full count register
PcieSpartanVI _CHAN_TX_AMT_LVL	0x00000028 // 10 Tx almost empty level register - used for level based FIFO interrupt
PcieSpartanVI _CHAN_RX_AFL_LVL	0x00000040 // 16 Rx almost full level register

Figure 4 PCIe-Spartan-VI Controller Port Address Map

There are 8 channels each with the memory map shown above. The offset to each channel relative to the base address as shown in the Base Address Map. Decodes 0-19 are reserved for the Base, 20-39 to channel 0, 40-59 to channel 1 etc.

The host system will enumerate to find the assets installed during power-on initialization. Interrupts are requested by the configuration space. Third party utilities can be useful to see how your system is configured. The interrupt level expected and style is also set in the registry. Dynamic Engineering recommends using the Dynamic Engineering Driver to take care of initialization and device registration.

Once the initialization process has occurred, and the system has assigned an address range to PCIe-Spartan-VI, software will need to determine what the address space is. We refer to this address as base in our software.

The next step is to initialize PCIe-Spartan-VI. The local Controller registers need to be configured as well as the registers within the User FPGA.

The following address maps for the User FPGA are based on the reference design. Your design implementation may change the addresses and bitmaps.

User Base Address Map

PcieSpartanVI_BASE_USER_ACCESS	0x8000 // PcieSpartanVI Base Set Addr(15) to use 32K dedicated to User control bus
PcieSpartanVI_USER_BASE	0x0000 // 0 User base address for PLL access
PcieSpartanVI_USER_LED	0x0004 // 1 User address for LED access Bit 4 enables to SW control
PcieSpartanVI_USER_STATUS	0x0008 // 2 User address for Status Register in Base
PcieSpartanVI_USER _TTL_DAT	0x0010 // 4 User address for TTL Data output and read-back of IO level 11-0
PcieSpartanVI_USER _TTL_EN	0x0014 // 5 User address for TTL Data Enables 11-0
PcieSpartanVI_USER_CNT	0x001C // 7 PcieSpartanVI Base Write Master Count, Read selected count
PcieSpartanVI_USER_PLL_FIFO	0x0028 // 10 User address for PLL FIFO
PcieSpartanVI_USER_CH_0	0x0050 // 20 User address pointer offset for channel 0
PcieSpartanVI_USER_CH_1	0x00A0 // 40 User address pointer offset for channel 1
PcieSpartanVI_USER_CH_2	0x00F0 // 60 User address pointer offset for channel 2
PcieSpartanVI_USER_CH_3	0x0140 // 80 User address pointer offset for channel 3
PcieSpartanVI_USER_CH_4	0x0190 // 100 User address pointer offset for channel 4
PcieSpartanVI_USER_CH_5	0x01E0 // 120 User address pointer offset for channel 5
PcieSpartanVI_USER_CH_6	0x0230 // 140 User address pointer offset for channel 6
PcieSpartanVI_USER_CH_7	0x0280 // 160 User address pointer offset for channel 7

Figure 5 PCIe-Spartan-VI User Base Address Map

User Port Address Map

PcieSpartanVI_USER _CH_BASE	0x0000 // 0 User address for Base Register
PcieSpartanVI_USER _CH_STATUS	0x0004 // 1 User address for Status
PcieSpartanVI_USER _CH_SP	0x0008 // 2 User address for Spare Register
PcieSpartanVI_USER _UB_FIFO_CNT	0x000C // 3 User address for User to Cont FIFO Count
PcieSpartanVI_USER _BU_FIFO_CNT	0x0010 // 4 User address for Cont to User FIFO Count
PcieSpartanVI_USER _FIFO_WR	0x0014 // 6 User address for Writing RX FIFO
PcieSpartanVI_USER _FIFO_RD	0x0018 // 6 User address for Reading TX FIFO

Figure 6 PCIe-Spartan-VI User Port Address Map

Dynamic Drivers provide a GPB write and GPB read utility which allows the user to pass an offset and data to write or an offset to read and return a LW. The utility assumes the offset to the User FPGA. User SW can use the above definitions or self-defined mapping to build a pointer to the intended offset. For example: PcieSpartanVI_USER_CH_7 + PcieSpartanVI_USER_CH_STATUS is the offset for the Status register in channel 7 of the User FPGA.

Register Definitions

Bus Controller Base Register Definitions

PcieSpartanVI_BASE_BASE

[0x0000 Main Control Register Port read/write]

DATA BIT	BASE REGISTER DESCRIPTION
0	BASE_USER_INT_MASK
1	BASE_USER_LOAD_USER
2	BASE_USER_LOAD_FLASH
3	BASE_USER_LOAD_USERDN
14	BASE_USER_PGM_RST
15	BASE_USER_RST

Figure 7 PCIe-Spartan-VI Xilinx Base Control Register

BASE_USER_INT_MASK : Set to enable interrupt from User device. Default is disabled. When ‘1’ the master enable is “enabled”.

This version of the design will implement JTAG programming of the User FPGA and FLASH. The definitions etc. are TBD. Ported from the D20MX and ASCB designs.

BASE_USER_RST: Set to reset User device low to enable

PcieSpartanVI_BASE_ID

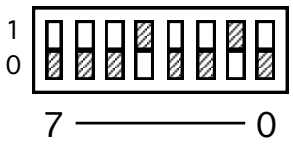
[0x04 Switch and Design number port read only]

DATA BIT	DESCRIPTION
31-24	Design Revision
23-16	Design Number
15-8	PCI Core Revision
7-0	DIP switch

Figure 8 PCIe-Spartan-VI Xilinx ID Register

The ID register is made up of the DIP switch port plus design revision and design number.

The DIP Switch is labeled for bit number and '1' '0' in the silk screen. The DIP Switch can be read from this port and used to determine which PCIe-Spartan-VI is which in a system with multiple cards installed. The DIP switch can also be used for other purposes – software revision etc. The switch shown would read back 0x12.



The Design Number and Design Revision allow for 256 Xilinx versions with 256 revisions of each. The standard Design Number x1 and the current revision is 0x01.

The PCI revision is the revision of the core in use. It is unlikely to change. The board ID can also be updated for clientized versions to allow drivers to differentiate between revisions and applications.

PcieSpartanVI_BASE_STATUS

[0x0008 status register read only]

STATUS REGISTER

DATA BIT	DESCRIPTION
31-18	Spare
17	User FPGA Load Status
16-10	Spare
9	User Int Masked
8	User Int
7	Ch7 Int Status
6	Ch6 Int Status
5	Ch5 Int Status
4	Ch4 Int Status
3	Ch3 Int Status
2	Ch2 Int Status
1	Ch1 Int Status
0	Ch0 Int Status

Figure 9 PCIe-Spartan-VI Status Port

The masks for the Controller Port interrupts are in the Port. When Ch(0:7) Int Status are set an interrupt is pending from the channel or channels indicated.

The User FPGA can generate an interrupt request. A second mask is provided within the Controller for this Interrupt request to allow for polling. See the Controller Base Register. User Int is the unmasked signal. User Int Masked is the version to check for an active interrupt, and is after the mask.

User FPGA Load Status is tied to the Done bit from the User FPGA. Done will be asserted low during programming and return high once completed.

PcieSpartanVI_BASE_JTAG

[0x000C program the Altera storage FIFO and status]

DATA BIT	DESCRIPTION
31	MUX select
30-4	spare
3	TDO
2	TMS
1	TCK
0	TDI

Figure 10 PCIe-Spartan-VI Programming port

TDI is set or cleared to create the data pattern to load JTAG data into the TDI pin on the User device. TCK and TMS are used in conjunction with TDI to create the programming stream. If programming the QSPI the FPGA will need to be programmed with the JTAG to QSPI module – automatic from the Impact program.

TDO is synchronized to the local clock and available for read-back.

The MUX Select when '0' provides a path from the Header to the User device for direct programming from the platform programming adapter. When set to '1' the mux provides a path from this device to the User FPGA allowing the User device to be programmed directly or the data stored into QSPI to load at power up.

The reference design has a different pattern on the User LEDs than the reference file provided with the UserAp. The UserAp test menu has two functions to allow user file loading and reloading from FLASH. The file name is entered as a string including the path. If you store the file with the UserAp executable PcieSpartanViUserAp.exe no path will be needed. The string look-up can be automated to provide other functionality; loading when Main is launched for example. This method can be used for remote site updates.

Bus Controller Port Register Definitions

PcieSpartanVI_CH_CNTRL

[0x0000 read/write]

Channel Control Register	
DESCRIPTION	**Bit**
CNTRL_TX_FIFO_RST	0x00000001 //0 set to clear FIFO's
CNTRL_RX_FIFO_RST	0x00000002 //1 set to clear FIFO's
CNTRL_FF_TEST	0x00000004 //2 bypass mode
CNTRL_MINTEN	0x00000008 //3 channel interrupt enable
CNTRL_FORCE_INT	0x00000010 //4 channel based force interrupt function
CNTRL_DMA_WREN	0x00000020 //5 DMA interrupt enable burst in
CNTRL_DMA_RDEN	0x00000040 //6 DMA interrupt enable burst out
CNTRL_DMA_INURGENT	0x00000080 //7 DMA prioritize DMA in
CNTRL_DMA_OUTURGENT	0x00000100 //8 DMA prioritize DMA out
CNTRL_PAC4_TXSTART	0x00010000 //16 enable Tx State machine
CNTRL_PAC4_RXSTART	0x00020000 //17 enable Rx State Machine
CNTRL_PAC4_TX_FF_AMT_INT	0x00040000 //18 enable Transmit FIFO Almost Empty Interrupt based on Pulse
CNTRL_PAC4_RX_FF_AFL_INT	0x00080000 //19 enable Receive FIFO Almost Full Interrupt based on Pulse
CNTRL_PAC4_OFL_INT	0x00100000 //20 enable Receive OverFlow interrupt
CNTRL_PAC4_TX_FF_AMT_INT_LVL	0x00200000 //21 enable Transmit FIFO Almost Empty Interrupt based on Level
CNTRL_PAC4_RX_FF_AFL_INT_LVL	0x00400000 //22 enable Receive FIFO Almost Full Interrupt based on Level

Figure 11 PCIe-Spartan-VI Controller Channel Control Register

CNTRL_TX_FIFO_RST and CNTRL_RX_FIFO_RST when set '1' cause the respective memories to be reset to the empty state. Clear to '0' for normal operation.

CNTRL_FF_TEST when set '1' routes data from the TX FIFO to the RX FIFO for loop-back testing within the Controller. Can be used with DMA or target accesses. Examples of how to operate are in the reference software.

CNTRL_MINTEN is the master interrupt enable. This bit needs to be set to have the particular channel able to cause an interrupt. In addition, individual masks for the interrupts are provided.

CNTRL_FORCE_INT when set causes an interrupt from the channel. Useful for debugging and checking interrupt handling software.

CNTRL_DMA_WREN [Burst In] and CNTRL_DMA_RDEN [Burst Out] when set enable DMA interrupts to be active. BurstIn refers to data coming from system memory to PCIe-Spartan-VI and BurstOut refers to data going to system memory.

CNTRL_INURGENT and CNTRL OUTURGENT are control bits which can provide additional priority to this channel relative to other channels. Usually not used, but can be useful when one channel has higher priority than the others.

CNTRL_PAC4_TX_FF_AMT_INT and CNTRL_PAC4_RX_FF_AFL_INT when set enable interrupts based on the FIFO level transition. From not Almost Full to Almost Full for the RX FIFO, and/or not Almost Empty to Almost Empty for the TX FIFO. These interrupts can be handy if using small DMA transfers that need to be managed. The interrupts are cleared by writing to the associated status register “sticky” bits.

CNTRL_PAC4_OFL_INT enables the interrupt on receive FIFO overflow. This condition is in theory “impossible” since there is HW flow control.

CNTRL_PAC4_TX_FF_AMT_INT_LVL and CNTRL_PAC4_RX_FF_AFL_INT_LVL when set enable interrupts based on the FIFO level. When Almost Full for the RX FIFO and/or Almost Empty for the TX FIFO. These interrupts can be handy if using small DMA transfers that need to be managed. These interrupts stay set until the level is returned above or below the threshold. The masks can be used to disable.

CNTRL_PAC4_TXSTART and CNTRL_PAC4_RXSTART enable the data transfer state-machines for the transmit [to User] and receive [from User] byte lanes. Normally both are enabled. For some test situations one or both can be disabled.

No interrupts are associated with the state-machines directly. Interrupts are available for the completion of a DMA or based on programmable FIFO levels to allow for user control.

PcieSpartanVi_CH_STATUS

[0x004 read/write]

Channel Control Register	
DESCRIPTION	**Bit**
STAT_TX_FIFO_MT	0x00000001 //0 set when TX FIFO is empty
STAT_TX_FIFO_AE	0x00000002 //1 set when TX FIFO is Almost Empty
STAT_TX_FIFO_FULL	0x00000004 //2 set when TX FIFO is Full
STAT_RX_FIFO_MT	0x00000010 //4 set when RX FIFO is Empty
STAT_RX_FIFO_AF	0x00000020 //5 set when RX FIFO is Almost Full
STAT_RX_FIFO_FULL	0x00000040 //6 set when RX FIFO is Full
STAT_TX_AMT_INT	0x00000100 //8 Transmit Almost Empty Int Occurred
STAT_RX_AFL_INT	0x00000200 //9 Receive Almost Full Int Occurred
STAT_TX_FF_INT_LAT	0x00000400 //10 Transmit Almost Full Int Occurred
STAT_RX_FF_INT_LAT	0x00000800 //11 Receive Almost Full Int Occurred
STAT_BURSTIN_ERR	0x00001000 //12 write DMA error
STAT_BURSTOUT_ERR	0x00002000 //13 read DMA error
STAT_WR_DMA_INT	0x00004000 //14 write DMA Interrupt
STAT_RD_DMA_INT	0x00008000 //15 read DMA Interrupt
STAT_RX_OVFL_LAT	0x00080000 //19 Rx OverFlow Latched
STAT_BO_IDLE	0x00400000 //22 Burst Out Idle
STAT_BI_IDLE	0x00800000 //23 Burst In Idle
LOC_INT	0x40000000 //30 channel interrupt
STAT_INT_ACTIVE	0x80000000 //31 channel interrupt is active

Figure 12 PCIe-Spartan-VI Controller Channel Status Register

Transmit FIFO Empty: When a one is read, the transmit data FIFO contains no data; when a zero is read, there is at least one data word in the FIFO.

Transmit FIFO Almost Empty: When a one is read, the number of data words in the transmit data FIFO is less than or equal to the value written to the TX_AMT_LVL register; when a zero is read, the FIFO level is more than that value.

Transmit FIFO Full: When a one is read, the transmit data FIFO is full; when a zero is read, there is room for at least one more data word in the FIFO.

Please note with the Receive side status; the status reflects the state of the FIFO and does not take the 4 deep pipeline into account. For example, the FIFO may be empty and there may be valid data within the pipeline. The data count is the combined FIFO and pipeline value and can be used for read size control.

Receive FIFO Empty: When a one is read, the receive data FIFO contains no data; when a zero is read, there is at least one data word in the FIFO.

Receive FIFO Almost Full: When a one is read, the number of data words in the receive data FIFO is greater or equal to the value written to the RX_AFL_LVL register; when a zero is read, the FIFO level is less than that value.

Receive FIFO Full: When a one is read, the receive data FIFO is full; when a zero is read, there is room for at least one more data-word in the FIFO.

STAT_TX_AMT_INT: When a one is read, the transmit FIFO almost empty is asserted. This is a level based interrupt. The status is before the mask to allow for non-interrupt driven SW operation. When the FIFO level is below the programmed level this bit is asserted.

STAT_TX_FF_INT_LAT: When a one is read, the transmit FIFO almost empty has been asserted. This is a triggered interrupt. The status is before the mask to allow for non-interrupt driven SW operation. When the FIFO level has dropped below the programmed level the status is captured and held. Clear by writing back to this bit.

STAT_RX_AFL_INT: When a one is read, the receive FIFO almost full is asserted. This is a level based interrupt. The status is before the mask to allow for non-interrupt driven SW operation. When the FIFO level is above the programmed level this bit is asserted.

STAT_RX_FF_INT_LAT: When a one is read, the receive FIFO almost full has been asserted. This is a triggered interrupt. The status is before the mask to allow for non-interrupt driven SW operation. When the FIFO level has risen the programmed level the status is captured and held. Clear by writing back to this bit.

Write/Read DMA Error Occurred: [STAT_BI_ERR, STAT_BO_ERR] When a one is read, a write or read DMA error has been detected. This will occur if there is a target or master abort or if the direction bit in the next pointer of one of the chaining descriptors is incorrect. A zero indicates that no write or read DMA error has occurred. These bits are latched and can be cleared by writing back to the Status register with a one in the appropriate bit position.

BO and BI Idle [STAT_BO_IDLE, STAT_BI_IDLE] are Burst Out and Burst In IDLE state status for the Receive and Transmit DMA actions. The bits will be 1 when in the IDLE state and 0 when processing a DMA. A new DMA should not be launched until the State machine is back in the IDLE state. Please note that the direction implied in the name has to do with the DMA direction – Burst data into the card for TX and burst data out of the card for Receive.

Write/Read DMA Interrupt Occurred: [STAT_WR_DMA_INT, STAT_RD_DMA_INT] When a one is read, a write/read DMA interrupt is latched. This indicates that the scatter-gather list for the current write or read DMA has completed, but the associated interrupt has yet to be processed. A zero indicates that no write or read DMA interrupt is pending.

Channel Interrupt Active: When a one is read, it indicates that a system interrupt is potentially asserted caused by an enabled channel interrupt condition. A zero indicates that no system interrupt is pending from an enabled channel interrupt condition. The Board level master interrupt enable will also need to be asserted to allow the active channel interrupt to become an interrupt request.

Local Interrupt: When a one is read, it indicates that any of the masked conditions other than DMA are active and enabled. Local Interrupt is or'd with the DMA interrupt sources to create the Channel Interrupt Active signal and to request the Interrupt.

STAT_RX_OVFL_LAT is set if the RX FIFO is overwritten. Should never occur as there is flow control in place. However, since this is a user design it is possible for user changes to break the flow control. Clear by writing back with this bit set.

PcieSpartanVi_CH_WR_DMA_PNTR

[0x008 Write only]

DMA Pointer Address Register

Data Bit	Description
31-2	First Chaining Descriptor Physical Address
1	direction [0]
0	end of chain

Figure 13 PCIe-Spartan-VI Controller Channel Write DMA Register

This write-only port is used to initiate a scatter-gather write [TX] DMA. When the address of the first chaining descriptor is written to this port, the DMA engine reads three successive long words beginning at that address. Essentially this data acts like a chaining descriptor value pointing to the next value in the chain.

The first is the address of the first memory block of the DMA buffer containing the data to read into the device, the second is the length in bytes of that block, and the third is the address of the next chaining descriptor in the list of buffer memory blocks. This process is continued until the end-of-chain bit in one of the next pointer values read indicates that it is the last chaining descriptor in the list.

All three values are on LW boundaries and are LW in size. Addresses for successive parameters are incremented. The addresses are physical addresses the HW will use on the PCI bus to access the Host memory for the next descriptor or to read the data to be transmitted. In most OS you will need to convert from virtual to physical. The length parameter is a number of bytes, and must be on a LW divisible number of bytes.

Status for the DMA activity can be found in the channel control register and channel status register.

Notes:

1. Writing a zero to this port will abort a write DMA in progress.
2. End of chain should not be set for the address written to the DMA Pointer Address Register. End of chain should be set when the descriptor follows the last length parameter.
3. The Direction should be set to '0' for Burst In DMA in all chaining descriptor locations.

PcieSpartanVI_CH_RD_DMA_PNTR

[0x00C Write only]

DMA Pointer Address Register	
Data Bit	**Description**
31-2	First Chaining Descriptor Physical Address
1	direction [1]
0	end of chain

Figure 14 PCIe-Spartan-VI Controller Channel Read DMA Register

This write-only port is used to initiate a scatter-gather read [RX] DMA. When the address of the first chaining descriptor is written to this port, the DMA engine reads three successive long words beginning at that address. Essentially this data acts like a chaining descriptor value pointing to the next value in the chain.

The first is the address of the first memory block of the DMA buffer to write data from the device to, the second is the length in bytes of that block, and the third is the address of the next chaining descriptor in the list of buffer memory blocks. This process is continued until the end-of-chain bit in one of the next pointer values read indicates that it is the last chaining descriptor in the list.

All three values are on LW boundaries and are LW in size. Addresses for successive parameters are incremented. The addresses are physical addresses the HW will use on the PCI bus to access the Host memory for the next descriptor or to read the data to be transmitted. In most OS you will need to convert from virtual to physical. The length parameter is a number of bytes, and must be on a LW divisible number of bytes.

Status for the DMA activity can be found in the channel control register and channel status register.

Notes:

1. Writing a zero to this port will abort a write DMA in progress.
2. End of chain should not be set for the address written to the DMA Pointer Address Register. End of chain should be set when the descriptor follows the last length parameter.
3. The Direction should be set to ‘1’ for Burst Out DMA in all chaining descriptor locations.

PcieSpartanVI_CH_TX_FIFO_CNT

[0x008 Port Read only]

TX FIFO Data Count Port	
Data Bit	**Description**
31-0	TX Data Words Stored

Figure 15 PCIe-Spartan-VI Controller Channel TX FIFO Count

This read-only register port reports the number of 32-bit data words in the transmit FIFO. The TX FIFO has 8K locations.

PcieSpartanVI_CH_RX_FIFO_CNT

[0x00C Port Read only]

RX FIFO Data Count Port	
Data Bit	**Description**
31-0	RX Data Words Stored

Figure 16 PCIe-Spartan-VI Controller Channel RX FIFO Count

This read-only register port reports the number of 32-bit data words in the receive FIFO. The pipeline for DMA processing has an additional 4 positions. The channel status register contains the combined pipeline and FIFO count. This design has 8K+4 locations possible in the FIFO + pipeline.

PcieSpartanVI_CH_FIFO

[0x010 Port Read/Write]

RX and TX FIFO Port	
Data Bit	**Description**
31-0	FIFO data word

Figure 17 PCIe-Spartan-VI Controller Channel FIFO Access

This port is used to make single-word accesses into the TX and out of the RX FIFO. Please note that reading is from the RX FIFO and writing is to the TX FIFO. Unless Bypass mode is established the data will not match.

PcieSpartanVI_CH_TX_AMT

[0x014 Port Read/Write]

TX Almost-Empty Pulse Register	
Data Bit	**Description**
31-16	Spare
15-0	TX FIFO Almost-Empty Level

Figure 18 PCIe-Spartan-VI Controller Channel TX Almost Empty Pulse

This read/write port accesses the transmitter almost-empty level register. When the number of data words in the transmit data FIFO is less than this value, the almost-empty status bit will be set. The register is R/W for 16 bits. The mask is valid for a size matching the depth of the FIFO. Used for the Pulsed interrupt and “Urgent” processing.

PcieSpartanVI_CH_RX_AFL

[0x018 Port Read/Write]

RX Almost-Full Pulse Register

Data Bit	Description
31-16	Spare
15-0	RX FIFO Almost-Full Level

Figure 19 PCIe-Spartan-VI Controller Channel RX Almost Full Pulse

This read/write port accesses the receiver almost-full level register. When the number of data words in the receive data FIFO is greater than this value, the almost-full status bit will be set. The register is R/W for 16 bits. The mask is valid for a size matching the depth of the FIFO. This value is applied to the entire FIFO chain [8K+4]. Used for the Pulsed interrupt and “Urgent” processing.

PcieSpartanVI_CH_TX_AMT_LVL

[0x028 Port Read/Write]

TX Almost-Empty Level Register

Data Bit	Description
31-16	Spare
15-0	TX FIFO Almost-Empty Level

Figure 20 PCIe-Spartan-VI Controller Channel TX Almost Empty Level

This read/write port accesses the transmitter almost-empty level register. When the number of data words in the transmit data FIFO is less than this value, the almost-empty status bit will be set. The register is R/W for 16 bits. The mask is valid for a size matching the depth of the FIFO. Used for the Level based interrupt.

PcieSpartanVI_CH_RX_AFL_LVL

[0x040 Port Read/Write]

RX Almost-Full Level Register

Data Bit	Description
31-16	Spare
15-0	RX FIFO Almost-Full Level

Figure 21 PCIe-Spartan-VI Controller Channel RX Almost Full Level

This read/write port accesses the receiver almost-full level register. When the number of data words in the receive data FIFO is greater than this value, the almost-full status bit will be set. The register is R/W for 16 bits. The mask is valid for a size matching the depth of the FIFO. This value is applied to the entire FIFO chain [8K+4]. Used for the Level based interrupt.

User FPGA Base Address Map

The User FPGA is completely programmable – the address map above and definitions below only have meaning if the reference VHDL is used as a starting point for your design. The reference software and reference User hardware implementation are used to perform the ATP on each board prior to shipment. The Engineering kit can include the HDEterm100 and a cable to interconnect the PCIe-Spartan-VI with the HDEterm100. For more information, please refer to the web page.

PcieSpartanVI_BASE_USER_ACCESS defines the offset from the Xilinx base address to the User Base address. Dynamic Drivers automatically include this offset. If you are doing your own driver you will need to include this offset. All other offsets shown are relative to the User base address.

PcieSpartanVI_USER_BASE

[0x000 Main Control Register Port read/write]

BASE REGISTER

DATA BIT	DESCRIPTION
31-24	InstNum
23	Spare
22	ForceInt
21	MIntEn
20	ClrPll
19-17	spare
16	PLL Programming Enable
15-8	Counter Select
7-0	PLL_CLK_EN_(7..0)

Figure 22 PCIe-Spartan-VI User Base Control Register

PCIe-Spartan-VI has 8 PLL devices which are programmed to produce the desired frequency with an i2c bus. Each PLL has a common data pin and independent clocks. The PLLs also have independent references.

PLL_CLK_EN_(7..0).when set selects that PLL for loading or reading. Reading is required to only have one PLL selected. Writing can be done in parallel or with separate writes to each device. For parallel writes to work the address of each PLL must be the same.

Counter Select is used to pick the counter to read-back with the “USER CNT” register. Once the PLLs are programmed the outputs can be used to count and check if the expected frequency is received.

The reference design contains the logic and software required to program the PLLs and to read the programmed frequency back. The software to determine the frequency command words is available from Cypress Semiconductor. The part number is CY22393FXI. Cypress has a utility available for calculating the frequency control words for the PLLs. https://www.dyneng.com/_Download/Utilities/CyberClocks.zip is the URL for the Cypress software used to calculate the PLL programming words.

The PLLs respond to one of two addresses [only one works]. As part of the ATP the reference software determines the address of each PLL and stores into an array. The remaining functions within the PLL test section use the data from the array to access each of the PLLs. Functions for loading, reading, comparing the read-back and expected load pattern, plus counting and checking against the expected count are provided. The software is part of the engineering kit and can be ported to your application.

PLL Programming Enable when set (‘1’) enables the i2c state-machine to begin operation. The state-machine will read the data from the FIFO and transfer to or read from the selected PLL as directed by the initial instruction read from the FIFO. Please refer to the “ALT PLL FIFO” section for the header information.

ClrPll when set ‘1’ resets the PLL interface – FIFOs etc. When ‘0’ the interface is in standard mode. Can set and clear on consecutive GPB accesses.

MIntEn is the master interrupt enable for User FPGA. Default is disabled. When ‘1’ the master enable is “enabled”.

ForceInt when ‘1’ and the MintEn is enabled causes an interrupt to be generated. This bit is useful for software debugging.

Please note the Controller enable for the User Interrupt must also be enabled.

InstNum is a R/W field where the driver can store the instance of the board. Useful for multiple board implementations.

PcieSpartanVI_USER_LED

[0x004 read/write]

LED REGISTER	
DATA BIT	**DESCRIPTION**
31-5	spare
4	LED Enable
3 - 0	LED Control

Figure 23 PCIe-Spartan-VI User LED Control Register

LED Control when set (‘1’) and LED Enable is set (‘1’) causes the individual LEDs to be illuminated. When ‘0’ the LED is off. When LED Enable is ‘0’ the LED pattern is controlled by the HW strapped default. The FLASH based reference design is set to “0x5”. Please note that the HW straps are counter-intuitive as the LED is on when ‘0’ is supplied and off when ‘1’ is supplied. The SW path has the inversion built in. A second version of the reference design has the pattern set to “0xA” so you have a visual for the FPGA loading properly from the SW load as opposed to the FLASH load.

PcieSpartanVI_USER_STATUS

[0x008 Status Register read/write]

Status Register	
DATA BIT	**DESCRIPTION**
31-24	DesignId
23-16	Int(7-0)
15-11	spare
10-8	PllPckDnCnt
7	PllNakLat
6	PllPacketDoneLat
5	PllReadFifoMt
4	PllWriteFifoMt
3	PllIdle
2	PllEn
1	LocalInt
0	spare

Figure 24 PCIe-Spartan-VI User Status Register

Int(7-0) when set indicate that a channel interrupt is pending. If the master interrupt is enabled the interrupt request from the User flows through to the Controller where it can

be passed onto the system. If the master enable is not set the INTx status can be used for polling. Additional masking is provided within the channels. INT0 is associated with Channel 0 etc. LocalInt is set if any of the INTx bits are set [before the master interrupt enable].

DesignId is set to allow the user software to read the Design Number and determine how to interact with the design.

When PllEn is set the PLLCounting test is in operation. The test is started by writing the control count to the master counter. When the master counter hits 0x00 the test completes. PllEn is set while the counter is operating.

PllIdle is set when the PLL controller is in the IDLE state. This bit can be polled to track status on the loading of the PLLs by the state-machine. Please note the state-machine will pass through the IDLE state during multiple packet operations.

PllWriteFifoMt is '1' when the FIFO associated with moving data to the PLL is empty.

PllReadFifoMt is '1' when the FIFO associated with data read from the PLL is empty. Please note: this FIFO will receive data when testing addresses etc.

PllPacketDoneLat is set and held when the PLL state-machine is done processing a packet. The bit is held until the same bit is written to – that is write to the status port with this bit '1' to clear.

PllNakLat is set when the PLL does not respond to an access attempt. This is normal during discovery and abnormal once the addresses are known. Clear by write-back to the status port with this bit set.

PllPckDnCnt is a 3 bit field which is incremented when a packet completes. During PLL programming address offsets are used which makes one load take more than one packet. The count is useful for determining when the complete PLL programming cycle has happened. Clear to 0x00 before using to check a load. Clear by ClrPll or writing back to the status register and clearing the PllPacketDoneLat [also clears this field].

PcieSpartanVI_USER_TTL_DAT

[0x010 TTL Data Port read/write]

TTL Data Register	
DATA BIT	**DESCRIPTION**
11-0	TTL Data 11-0

Figure 25 PCIe-Spartan-VI TTL Data Register

The TTL data pattern written to this port will be loaded to the output side for the bits enabled in the TTL_EN register. All bits when read are from the IO input. The IO may or may not match the output definition depending on which bits are enabled and what is connected on the IO side of the interface.

PcieSpartanVI_USERT_TTL_EN

[0x014 TTL Data Enable Port read/write]

485/LVDS IO REGISTERS	
DATA BIT	**DESCRIPTION**
11-0	TTL Data Enable

Figure 26 PCIe-Spartan-VI TTL Data Enable Register

The TTL IO each have individual enables. The bits set in this register are enabled to drive onto the cable side. Please note that the enables are inverted in HW to provide an active low enable at the ‘125 buffer devices. SW sets/reads as ‘1’ = enabled, ‘0’ = disabled [read this bit from the cable]. If the designer enables the bits in this register the IO are driven with 24 mA [drive / sink] buffers. If the bits are tied to the Output register the IO will be enabled for active low and disabled for active high signals – open drain operation. The reference for open drain operation is set by the header. The parts are also sourced by the selected voltage.

Application Note: Spare IO

Frequently a system will need some dedicated IO but not all of the IO. The reference design has all of the differential IO defined in the Channels, and all of the TTL IO defined in the Base. The mix can easily be changed to put TTL in the channels or Differential in the channels. "extra" IO can be used as a parallel port or other use. If not used please tie off to be controlled as an input to reduce system noise and avoid potential conflicts on the cabling.

PcieSpartanVI_USER_CNT

[0x01C Count read-back port]

Counter Port	
DATA BIT	**DESCRIPTION**
15-0	Count

Figure 27 PCIe-Spartan-VI Counter Results

There are 25 counters implemented to check the PLL programming. The Counter results to read are selected in the User Base register and read from this register. The 0x00 selection is the control count which will be 0 at the test completion. 0x01 corresponds to [silk screen] PLL1 output A. 0x02 to PLL1 output B etc, 0x18 = PLL8 output C.

The reference software has two counting tests. One test programs the PLLs to convert the 66.666 reference to 10 MHz on all channels and checks that all are within tolerance. The second test programs the PLLs to all different frequencies 10-33 MHz and again checks for the expected frequency to be measured via count. Most designs won't need this logic. It can be removed to create space in the FPGA if you are using the majority of the FPGA assets.

Design note: all 24 clocks are on FPGA clock inputs. Please see the PLL pin definitions in the pinout table. You may want to do some floor planning to have the clocks in the same vicinity as the logic. If only using a few clocks this is not necessary.

PcieSpartanVI_USER_PLL_FIFO

[0x028 PLL Data R/W port]

TTL IO REGISTERS	
DATA BIT	**DESCRIPTION**
31-0	Data to/from PLL

Figure 28 PCIe-Spartan-VI PLL FIFO

The transmit FIFO is monitored by the PLL state-machine. When the FIFO is written to the first word is read by the state-machine and parsed. The first word contains the mode on bit 0, address on 7-1, length on 15-8 [1-255], and the first byte or two to transfer. If multiple bytes – 3 or more are to be transferred the SW will need to make sure the data is in the FIFO for the 2nd LW before the end of the processing of the 2nd byte or an underflow condition will be detected. If your system timing is tough to manage it is suggested to disable the SM, load the FIFO and then enable the SM. A status bit for the idle condition is available to allow SW to know when the SM has responded to the disable.

The Length is the number of bytes in the data portion of the message + 1.

Please note: The PLLs have two data sets written to two address offsets per PLL programmed. The UserAp automatically converts the .jed file from the Cypress tool and generates the local buffers with the hex data to load to the PLL. The application software loads the FIFO with the correct address, length and data x2 for a complete programming operation.

The State-machine will parse the message and write or read based on bit 0. In either case the address and R/W are transmitted. An ACK is looked for from the Target. If a Write the data is then transmitted with the ACK being checked after each byte. Clocking is continuous until the message is completed. If a read is implemented, clocks are generated without data after the address. Data is captured during the high portion of the clock cycle, and the Master asserts the ACK until the last byte where a NAK is asserted. Data is stored into the receive FIFO in this case.

The reference software has examples of working with the PLLs and controlling HW.

User Channel Address Map

PcieSpartanVI_USER_CH_BASE

[0x000 Main Control Register Port read/write]

Channel Base Register

DATA BIT	DESCRIPTION
31-25	spare
24	RxIoEn
23	TxIoEn
22	RxFfAFIIntEnLvl
21	TxFfAMtIntEnLvl
20	spare
19	RxFfIntEn
18	TxFfIntEn
17	RxStart
16	TxStart
15-5	spare
4	IntForce
3	MIntEn
2	Bypass
1	RxRst
0	TxRst

Figure 29 PCIe-Spartan-VI Channel Base Register

TxRst When set (‘1’) causes Tx HW to be reset. FIFOs , state-machines etc.

RxRst when set (‘1’) causes Rx HW to be reset including FIFOs etc.

Bypass when set (‘1’) causes the HW to transfer data from the input data lane FIFO to the output data lane FIFO. The reference software has a loop-back test checking this data path. It is a good indicator that the User FPGA is properly loaded and the data lanes are functional. It is recommended to keep this logic in place to allow for BIT.

MIntEn when set (‘1’) allows the enabled channel status to cause interrupts via the connection to the Bus Controller. The Master in the Bus Controller also needs to be set for this to work.

IntForce when set ‘1’ and MintEn is set, causes an interrupt to be requested. Mainly used as a development tool to check interrupt paths and to simulate conditions when the external system is not in place.

TxStart when set enables data transfer via IO in the Bus Controller to User direction. Transmitting to external system.

RxStart when set enables data transfer via IO in the User to Bus Controller direction. Receiving from external system.

Please note: the Tx and Rx directions are independent byte lanes. Also, when Bypass is enabled the IO paths are automatically disabled.

TxFfIntEn when set enables the Tx direction FIFO Interrupt. The tx FIFO interrupt is latched [available in Status register] and triggered by the Tx FIFO going Almost Empty. If DMA is not used or if smaller DMA transfers are used this interrupt can be useful to trigger a new data transfer in the TX direction.

RxFfIntEn when set enables the Rx direction FIFO Interrupt. The Rx FIFO interrupt is latched [available in Status register] and triggered by the Rx FIFO going Almost Full. If DMA is not used or if smaller DMA transfers are used this interrupt can be useful to trigger a new data transfer in the RX direction.

Please see the status port for more information about the clearing of interrupt conditions.

TxFfAMtIntEnLvl when set enables the level based interrupt for the Tx FIFO Almost Empty condition. Since it is level controlled, the interrupt is cleared by adding more data to the FIFO or by disabling this enable. Since this is the data lane FIFO the write will be through the associated Bus Controller channel. Defined for future implementation or client use. Unused in reference design.

RxFfAFIIntEnLvl when set enables the level based interrupt for the Rx FIFO Almost Full condition. Since it is level controlled, the interrupt is cleared by adding reading data from the FIFO or by disabling this enable. Since this is the data lane FIFO the read will be through the associated Bus Controller channel. Defined for future implementation or client use. Unused in reference design.

TxIoEn when set causes the IO associated with the channel to be configured for transmission. In the case of the reference design with the bit set the directions are set to cause transmission and the terminations disabled. The default configuration is disabled.

RxIoEn when set causes the IO associated with the channel to be routed from the Receiver. When cleared the data path into the Rx FIFO is from the GPB. Set for standard IO operation, clear for Target writes to the Rx FIFO.

PcieSpartanVI_USER_CH_STATUS

[0x004 Status Register Port read/write]

Channel Status PORT

DATA BIT	DESCRIPTION
31	IntStat
30	LocInt
29-12	spare
11	UBFfIntLat
10	BUFfIntLat
9	spare
8	spare
7	UBFfFI
6	UBFfAfl
5	UBFfAMt
4	UBFfMt
3	BUFfFI;
2	BUFfAFI
1	BUFfAMt
0	BUFfMt

Figure 30 PCIe-Spartan-VI Channel Status Register

IntStat when set indicates that LocInt is set and the Mask is also enabled i.e. an interrupt is active for this channel.

LocInt is the combination of interrupts and single level masks before the channel level mask. Set when at least one Interrupt request is present.

note:
UB = User => Bus Controller
BU = Bus Controller => User

UBFfIntLat is latched and held until cleared by write to the status port with this bit position set. Almost Full User to Bus Controller FIFO.

UBFfIntLat is latched and held until cleared by write to the status port with this bit position set. Almost Empty User to Bus Controller FIFO.

UBFfFI is set when the User to Bus Controller FIFO is Full.
UBFfAFI is set when the User to Bus Controller FIFO is Almost Full.
UBFfAMt is set when the User to Bus Controller FIFO is Almost Empty.
UBFfMt is set when the AX FIFO is Empty.

BUFfFI is set when the Bus Controller to User FIFO is Full.

BUFfAFI is set when the Bus Controller to User FIFO is Almost Full.
BUFfAMt is set when the Bus Controller to User FIFO is Almost Empty.
BUFfMt is set when the Bus Controller to User FIFO is Empty.

PcieSpartanVI_USER_CH_SP

[0x008 Status Register Port read/write]

Channel Spare Port	
DATA BIT	**DESCRIPTION**
31 - 0	Spare Register

Figure 31 PCIe-Spartan-VI Channel Spare Register

It is always nice to have a spare register to R/W with and make sure your SW is operating properly. Each channel has one so you can do independence testing etc. Store values or ignore in your operational implementation.

PcieSpartanVI_USER_BUS_FIFO_CNT

[0x00C Status Register Port read]

Channel UB FIFO Data Count	
DATA BIT	**DESCRIPTION**
31-0	Data positions currently used in UB FIFO

Figure 32 PCIe-Spartan-VI Channel UB FIFO Count

Bit positions 9-0 have the count. 31-10 are set to 0x00. The count is the 32 bit data position count within the User to Bus Controller FIFO. Data is written as LW and read out as bytes before being transferred to the Bus Controller on the corresponding data lane. This count is used to compare against the programmed levels to determine Almost Full etc.

PcieSpartanVI_BUS_USER_FIFO_CNT

[0x010 Status Register Port read]

Channel BU FIFO Data Count	
DATA BIT	**DESCRIPTION**
31-0	Data positions currently used in BU FIFO

Figure 33 PCIe-Spartan-VI Channel BU FIFO Count

Bit positions 9-0 have the count. 31-10 are set to 0x00. The count is the 32 bit data position count within the XA FIFO. Data is written as bytes and read out as LW after being transferred from the Xilinx on the corresponding data lane. This count is used to compare against the programmed levels to determine Almost Empty etc.

The UB and BU FIFOs use flow control which is fixed in HW. Additional programmable [user] level control is provided to support interrupts and other purposes. The data is moved from one FPGA to the other by reading from the upstream port and writing to the downstream port. Since the data transfer is pipelined, there are delays with the status making it back to the controlling port. This is overcome by using burst mode transfers when the transmitting side is not Almost Empty and the Receive side is not Almost Full. When either side is Almost Empty /Full the transfer moves to a slower mode where the status can be checked after each byte is moved to allow the last bytes to be transferred without over-run or under-run issues.

PcieSpartanVI_USER_FIFO_WR

[0x014 UB FIFO Write Port]

Channel UB FIFO Write	
DATA BIT	**DESCRIPTION**
31-0	Data to write to UB FIFO

Figure 34 PCIe-Spartan-VI Channel UB FIFO Write

Writing to this port will put data into the UB FIFO. The data will transfer via the byte lanes through the Bus Controller and into user memory [if DMA is set-up etc.]. The reference software has a loop-back test where data is written to this port and DMA'd back to user space. Useful for debugging initial DMA into system memory without needing DMA out of system memory. Not normally used for system operation.

PcieSpartanVI_USER_FIFO_RD

[0x014 BU FIFO Read Port]

Channel BU FIFO Read	
DATA BIT	**DESCRIPTION**
31-0	Data read from BU FIFO

Figure 35 PCIe-Spartan-VI Channel BU FIFO Read

Reading from this port will retrieve data from the BU FIFO. The data from the Bus Controller side will transfer via the byte lanes from user memory [if DMA is set-up etc.] and into the User BU FIFO. The reference software incorporates a loop-back test where data is DMA'd to the channel's BU FIFO and read back with target accesses from this port. Useful for debugging initial DMA from system memory without needing DMA into system memory. Not normally used for system operation.

User FPGA Reference Design

The FPGA pin definitions are contained in the reference design. See the .UCF file for the project.

The reference design is hierarchical. The top module defined the connection to the UCF file and the underlying logic. Not strictly necessary – done for consistency with our other designs which frequently have cores or other logic tied in at the top level.

UserAp is the base level of the User Design. The decoding for the registers, clock generation [PLLs etc.] and ports are tied in at this level.

Each of the Ports is tied into the UserAp and provide independent controllers to provide the loop-back function. The FIFO and Control bus interfaces are distributed to provide a FIFO interface within each port. The idea is for the User design to replace the state-machines in the reference design to implement whatever protocol is required.

The IO are evenly distributed in the reference design. The allocation can be updated by the tie in at the UserAp level plus the logic within the ports. The number of ports can be reduced or potentially expanded depending on how the FIFO lanes are operated.

Loop-Back

Loop-back can be accomplished with the use of an HDEterm100 and HDEcabl100. The following cross connections are used to support the TTL and differential loop-back tests supplied with the driver and UserAp software.

TTL signal "from"	"to"
(11)100	(10)50
(9)44	(8)38
(7)82	(6)32
(5)74	(4)24
(3)68	(2)18
(1)12	(0)6

Each block of IO represents Channels in UserAp software (i.e. IO0-IO4 = Channel 0, IO5-IO9 = Channel 1)

Differential "from"	"to"
(IO0)1,51	(IO5)7,57
(IO1)2,52	(IO6)8,58
(IO2)3,53	(IO7)9,59
(IO3)4,54	(IO8)10,60
(IO4)5,55	(IO9)11,61
(IO10)13,63	(IO15)19,69
(IO11)14,64	(IO16)20,70
(IO12)15,65	(IO17)21,71
(IO13)16,66	(IO18)22,72
(IO14)17,67	(IO19)23,73
(IO20)27,77	(IO25)33,83
(IO21)28,78	(IO26)34,84
(IO22)29,79	(IO27)35,85
(IO23)30,80	(IO28)36,86
(IO24)31,81	(IO29)37,87
(IO30)39,89	(IO35)45,95
(IO31)40,90	(IO36)46,96
(IO32)41,91	(IO37)47,97
(IO33)42,92	(IO38)48,98
(IO34)43,93	(IO39)49,99

D100 Standard Pin Assignment

The pin assignment for the PCIe-Spartan-VI P1 connector.

IO_0P	IO_0M	1	51
IO_1P	IO_1M	2	52
IO_2P	IO_2M	3	53
IO_3P	IO_3M	4	54
IO_4P	IO_4M	5	55
TTL_0	GND*	6	56
IO_5P	IO_5M	7	57
IO_6P	IO_6M	8	58
IO_7P	IO_7M	9	59
IO_8P	IO_8M	10	60
IO_9P	IO_9M	11	61
TTL_1	GND*	12	62
IO_10P	IO_10M	15	63
IO_11P	IO_11M	14	64
IO_12P	IO_12M	15	65
IO_13P	IO_13M	16	66
IO_14P	IO_14M	17	67
TTL_2	TTL_8	18	68
IO_15P	IO_15M	19	69
IO_16P	IO_16M	20	70
IO_17P	IO_17M	21	71
IO_18P	IO_18M	22	72
IO_19P	IO_19M	23	73
TTL_3	TTL_9	24	74
fused fused VCCB		25	75
fused fused VCCB		26	76
IO_20P	IO_20M	27	77
IO_21P	IO_21M	28	78
IO_22P	IO_22M	29	79
IO_23P	IO_23M	30	80
IO_24P	IO_24M	31	81
TTL_4	TTL_10	32	82
IO_25P	IO_25M	33	83
IO_26P	IO_26M	34	84
IO_27P	IO_27M	35	85
IO_28P	IO_28M	36	86
IO_29P	IO_29M	37	87
TTL_5	GND*	38	88
IO_30P	IO_30M	39	89
IO_31P	IO_31M	40	90
IO_32P	IO_32M	41	91
IO_33P	IO_33M	42	92
IO_34P	IO_34M	43	93
TTL_6	GND*	44	94
IO_35P	IO_35M	45	95
IO_36P	IO_36M	46	96
IO_37P	IO_37M	47	97
IO_38P	IO_38M	48	98
IO_39P	IO_39M	49	99
TTL_7	TTL_11	50	100

Figure 36 PCIe-Spartan-VI D100 Pinout

VCCB is shunt selectable – 3.3V or 5V. Fused with 1.1A resettable fuse.

Applications Guide

Interfacing

Some general interfacing guidelines are presented below. Do not hesitate to contact the factory if you need more assistance.

ESD
Proper ESD handling procedures must be followed when handling PCIe-Spartan-VI. The card is shipped in an anti-static, shielded bag. The card should remain in the bag until ready for use. When installing the card, the installer must be properly grounded and the hardware should be on an anti-static work-station.

Start-up
Make sure that the "system" can see your hardware before trying to access it. Many BIOS will display the PCI devices found at boot up on a "splash screen" with the VendorID and CardId and an interrupt level. Look quickly! If the information is not available from the BIOS then a third party PCI device cataloging tool will be helpful. In Windows systems the device manager can be used.

Watch the system grounds. All electrically connected equipment should have a fail-safe common ground that is large enough to handle all current loads without affecting noise immunity. Power supplies and power consuming loads should all have their own ground wires back to a common point.

Construction and Reliability

PCIe Modules while commercial in nature can be conceived and engineered for rugged industrial environments. PCIe-Spartan-VI is constructed out of 0.062 inch thick High Temp FR4 material.

Surface mount components are used. Most devices are high pin count compared to mass of the device. For high vibration environments inductors and other higher mass per joint components can be glued down.

Conformal Coating is an option. For condensing environments conformal coating is required.

ROHS processing is an option. Standard lead solder is used unless “-ROHS” is added to the part number.

The D100 connector has Phosphor Bronze pins with Nickel plating for durability and Gold plating on the contact area on both plugs and receptacles. The connectors are keyed and shrouded. The pins are rated at 1 Amp per pin, 500 insertion cycles minimum [at a rate of 800 per hour maximum]. These connectors make consistent, correct insertion easy and reliable.

Thermal Considerations

PCIe-Spartan-VI is designed with CMOS circuits. The power dissipation due to internal circuitry is very low. With the one-degree differential temperature to the solder side of the board external cooling is easily accomplished.

Warranty and Repair

Please refer to the warranty page on our website for the current warranty offered and options.

https://www.dyneng.com/warranty.html

Service Policy

Before returning a product for repair, verify as well as possible that the suspected unit is at fault. Then call the Customer Service Department for a RETURN MATERIAL AUTHORIZATION (RMA) number. Carefully package the unit, in the original shipping carton if this is available, and ship prepaid and insured with the RMA number clearly written on the outside of the package. Include a return address and the telephone number of a technical contact. For out-of-warranty repairs, a purchase order for repair charges must accompany the return. Dynamic Engineering will not be responsible for damages due to improper packaging of returned items. For service on Dynamic Engineering Products not purchased directly from Dynamic Engineering contact your reseller. Products returned to Dynamic Engineering for repair by other than the original customer will be treated as out-of-warranty.

Out of Warranty Repairs

Out of warranty repairs will be billed on a material and labor basis. Customer approval will be obtained before repairing any item if the repair charges will exceed one half of the quantity one list price for that unit. Return transportation and insurance will be billed as part of the repair and is in addition to the minimum charge.

For Service Contact:

Customer Service Department
Dynamic Engineering
150 DuBois, Suite B/C
Santa Cruz, CA 95060
(831) 457-8891

support@dyneng.com

Specifications

PCIe Interfaces:	PCIe 4 lane interface
Access types:	Configuration and Memory space utilized
CLK rates supported:	133.33 MHz oscillator on User FPGA, 8 PLLs to provide programmable frequencies(24). 50 MHz GPB clock, internal DCMs.
Memory	FIFO memory is provided to support DMA both within the Controller and User FPGAs. Flow control is also implemented between the devices. 16 data lanes coupled to make 8 bidirectional full duplex channels.
IO	40 RS-485 or LVDS transceivers with programmable direction and termination - Twelve TTL with programmable direction
Interface:	D100 connector. [AMP] 787082-9 is the board side part number
Software Interface:	Control Registers within Controller and User FPGAs. VHDL reference design for User. Drivers provide generic calls for GPB access to allow any user modification to be programmed with the standard driver.
Initialization:	Programming procedure documented in this manual
Access Modes:	Registers on longword boundary. Standard target access read and write to registers and memory. DMA access to memory.
Access Time:	no wait states in DMA modes. 1-2 wait states in target access to Xilinx. Altera accesses are user defined.
Interrupt:	1 interrupt is supported with multiple sources. The interrupts are maskable and are supported with a status register.
Onboard Options:	Shunt for 3.3V or 5V reference to TTL IO. Header position for direct programming of User FPGA or attached QSPI FLASH.
Dimensions:	half-length PCIe board.
Construction:	High Temp FR4 Multi-Layer Printed Circuit, Surface Mount Components.
Power:	12V, 3.3V from PCIe bus. Local 5V and 2.5V, 1.8V, and 1.2V created with on-board power supplies.
User	8 position software readable switch 4 software controllable LEDs Power Supply LEDs

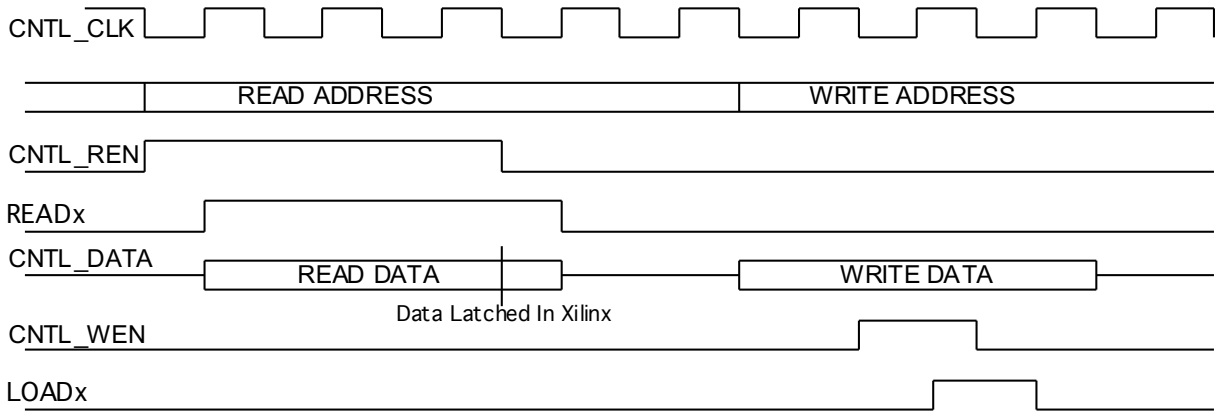
Order Information

Standard temperature range -40-85ºC components

PCIe-Spartan-VI	https://www.dyneng.com/PCIe-Spartan-VI.html half length PCIe card with user re-configurable Spartan VI [XC6S100LX-3FG676I is base device], 40 differential IO, 12 TTL IO, 8 PLL [24 clocks]
PCIe-Spartan-VI-ENG	Engineering Kit for the PCI-Spartan-VI is ala-carte Software, Cable,HDEterm100, reference User implementation. Please refer to the webpage for more information. Driver / reference SW: Win10/11, Linux.
-LVDS	Switch to LVDS IO instead of RS485(default selection).
-ROHS	Change to ROHS processing. Otherwise, leaded solder is used.
-CC	Add Conformal Coating for condensing environments
-Mixed	“Mixed” will be replaced with a new identifier. Combination of LVDS and RS485 IO installed.
HDEterm100	https://www.dyneng.com/HDEterm100.html 100-pin connectors (2) matching the PCIeAlteraCycloneIV-485/LVDS D100 interconnected with 100 screw terminals. DIN rail mounting. Optional terminations and test points.
HDEcable100	https://www.dyneng.com/HDEcabl100.html 100 pin connector matching PCIe-Spartan-VI and HDEterm100. Length options

Appendix

General Purpose Bus Timing



The basic timing for the GPB relative to the User device is shown. The CNTL clock operates at 50 MHz. Signals originating from the Bus Controller are registered to provide set-up and hold. Skew is kept to a minimum with edge FF's and controlled routing within the PCB.

A host read of the User over the GPB starts with the address and Read Enable being asserted. The User logic decodes the address and CNTL_REN to generate the local READx signal. The reference design provides 180 of the READ decodes. Read data is presented back to the CNTL_DATA bus and captured coincident with the edge of CNTL_REN as shown.

A host write to the User over the GPB starts with the address and data being asserted. After a 1 period delay CNTL_WEN is asserted. The User logic decodes the Address and CNTL_WEN to create the LOADx signals. LOADx when used as a write enable allows the CNTL_DATA to be loaded into the target device.

The included reference design [VHDL] provides the decoding and sample registers.

Further documentation is provided in-line with the VHDL.