DYNAMIC ENGINEERING
150 DuBois, Suite 3 Santa Cruz, CA 95060
(831) 457-8891 **Fax** (831) 457-4793
www.dyneng.com
sales@dyneng.com
Est. 1988

User Manual

PCI-Altera-485/LVDS

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

Revision H4
Corresponding Hardware: Revision E/F/G/H
10-2002-0705/0706/0707/0708

PCI-Altera-485/LVDS
PCI based Re-configurable logic
with RS-485/LVDS and TTL IO

Dynamic Engineering
150 DuBois, Suite 3
Santa Cruz, CA 95060
(831) 457-8891
FAX: (831) 457-4793

Manual Revision H4. Revised 11/29/07

Table of Contents

PRODUCT DESCRIPTION 6

THEORY OF OPERATION 9

Feature List Current 10

PROGRAMMING 11

ADDRESS MAP XILINX 12

Register Definitions 13
PCIALT_BASE 13
PCIALT_STATUS 17
PCIALT_DMAFIFO 18
PCIALT_FIFO_N 19
PCIALT_COMMAND 20
PCIALT_SWITCH 21
PCIALT_CMDSTAT 22
PCIALT_INTSTAT 23

ADDRESS MAP ALTERA 24
PCIALT_BASEA 25
PCIALT_RX_TX 27
PCIALT_DIR1,2 27
PCIALT_TERM1,2 28
PCIALT_IO1,2 28
Application Note: Spare IO 28
PCIALT_FIFO_NA 29
PCIALT_TTL 29
PCIALT_FIFOSTAT1 31
PCIALT_OSCCNT 32
PCIALT_OSCDAT 33
PCIALT_FIFOSTAT2 33

ALTERA PIN OUT 35

D100 STANDARD PIN ASSIGNMENT 47

D100 ALTERNATE PIN ASSIGNMENT 48

APPLICATIONS GUIDE 49

Interfacing 49

Construction and Reliability 50

Thermal Considerations 50

WARRANTY AND REPAIR 51

Service Policy 51
Out of Warranty Repairs 51

For Service Contact: 51

SPECIFICATIONS 52

ORDER INFORMATION 53

List of Figures

FIGURE 1 PCI-ALTERA-485/LVDS BLOCK DIAGRAM 7
FIGURE 2 PCI-ALTERA-485/LVDS XILINX ADDRESS MAP 12
FIGURE 3 PCI-ALTERA-485/LVDS XILINX BASE CONTROL REGISTER 13
FIGURE 4 PCI-ALTERA-485/LVDS INTERRUPT ENABLE PORT 16
FIGURE 5 PCI-ALTERA-485/LVDS STATUS PORT 17
FIGURE 6 PCI-ALTERA-485/LVDS DMA FIFO PORT 18
FIGURE 7 PCI-ALTERA-485/LVDS RX/TX FIFO PORTS 19
FIGURE 8 PCI-ALTERA-485/LVDS COMMAND PORT 20
FIGURE 9 PCI-ALTERA-485/LVDS USER SWITCH PORT 21
FIGURE 10 PCI-ALTERA-485/LVDS COMMAND STATUS PORT 22
FIGURE 11 PCI-ALTERA-485/LVDS INTERRUPT STATUS PORT 23
FIGURE 12 PCI-ALTERA-485/LVDS ALTERA-ATP ADDRESS MAP 24
FIGURE 13 PCI-ALTERA-485/LVDS ALTERA BASE CONTROL REGISTER 25
FIGURE 14 PCI-ALTERA-485/LVDS ALTERA BASE CONTROL REGISTER 27
FIGURE 15 PCI-ALTERA-485/LVDS DIRECTION CONTROL REGISTERS 27
FIGURE 16 PCI-ALTERA-485/LVDS TERMINATION CONTROL REGISTERS 28
FIGURE 17 PCI-ALTERA-485/LVDS IO CONTROL REGISTERS 28
FIGURE 18 PCI-ALTERA-485/LVDS ALTERA FIFO PORTS 29
FIGURE 19 PCI-ALTERA-485/LVDS TTL CONTROL REGISTER 29
FIGURE 20 PCI-ALTERA-485/LVDS FIFO STATUS PORT1 31
FIGURE 21 PCI-ALTERA-485/LVDS OSCILLATOR CONTROL REGISTER 32
FIGURE 22 PCI-ALTERA-485/LVDS OSCILLATOR DATA PORT 33
FIGURE 23 PCI-ALTERA-485/LVDS FIFO STATUS PORT2 33
FIGURE 24 PCI-ALTERA-485/LVDS D100 PINOUT 47
FIGURE 25 PCI-ALTERA-485/LVDS ALTERNATE D100 PINOUT 48

Product Description

PCI-Altera-485/LVDS is part of the PCI Compatible family of modular I/O components. The PCI-Altera-485/LVDS provides a user configurable 20K400E FPGA, along with 40 RS-485 or 40 LVDS transceivers, PLL and FIFO support, full DMA capabilities in a half-length single slot card. The 20K400E can be replaced with larger FPGAs for really large projects.

The RS-485 and LVDS parts can be mixed. The standard RS-485 devices are 5V parts with 40 MHz bandwidth. The standard LVDS parts are 3.3V parts. When mixed a different 485 device (3.3V) is used with lower bandwidth [Max3485].

The PCI bus implementation is 32 bits at 33 MHz, universal voltage. The hardware supports direct access software controlled read/write access to all locations plus DMA support to the high bandwidth ports. The hardware is optimized for back-to-back DMA accesses to support the multiple ports available on the PCI-Altera-485/LVDS.

The PCI-Altera-485/LVDS utilizes a PLX 9054 device for the PCI interface, and a dedicated Xilinx FPGA to manage the PCI DMA transfers through the 9054 and provide for loading the Altera FPGA.

The Xilinx supports multiple channels of data-flow to the Altera with FIFOs. There are 8 transmit and 8 receive FIFO paths. The “transmit” path is loaded from the Xilinx with data read from the host memory via DMA or with a direct host write and unloaded by the Altera. The receive path is loaded by the Altera and unloaded by the Xilinx. The Altera supplies the local clocks on the Altera side of the FIFOs. The Xilinx side is locked to the PCI reference clock times two (66 MHz) with high-speed clock buffers and matched length traces. Each of the “Altera FIFOs” is 8 bits wide and 4K deep.

The Xilinx has an “extra” FIFO - “DMA” - used to support DMA into and out of the board. The data is loaded and unloaded as 32-bit words with DMA. The Xilinx processes the data to load and unload the Altera FIFOs. The 32-bit PCI data (31-0) is written to the DMA FIFO from the PCI bus or from the Altera FIFOs, as bytes in the following order, 7-0, 15-8, 23-16, 31-24. The Xilinx maintains a command queue to allow back-to-back operations without interrupt delays for maximum through-put. The DMA FIFO is also used to load the Altera data file. The DMA FIFO reduces the PCI traffic and speeds operation.

An 8-bit "dip switch" is provided on the PCI-Altera-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, PCI board identification or test purposes.

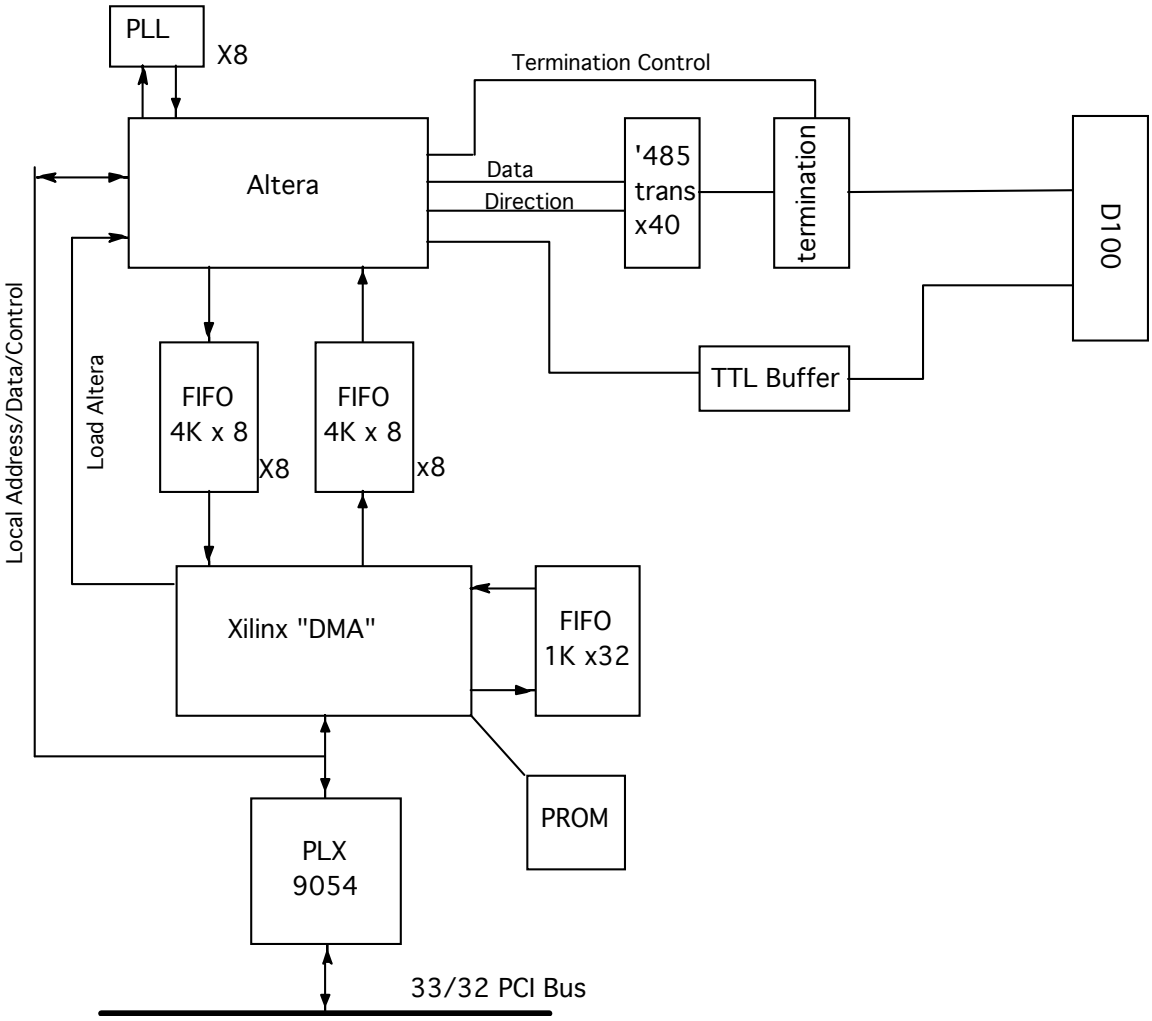


FIGURE 1 PCI-ALTERA-485/LVDS BLOCK DIAGRAM

LEDs are provided on board for user-defined purposes. The Xilinx has one LED, which is attached to a dual purpose pin. The LED indicates that data is being loaded into the Altera and after loading can be set or disabled. The Altera 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.

In addition an LED is provided to indicate that the 3.3V regulator is operating properly. Local regulation is provided for 3.3, 2.5, and 1.85 volts.

The PCI-Altera-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 40 MHz operation. The LVDS parts have much larger bandwidth, 166 Mhz receive and 400+ MHz transmit. The transcievers 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 constant space. The lengths are matched from the connector edge to the ball on the Altera. 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 Altera to be active or open.

The TTL IO is supported with open drain drivers with pull-ups and high-speed receivers [LVC244]. We have operated the board at 100 MHz through the TTL signals over short cables. The open drain drivers [LVTH125] have 64+ mA of sink capability.

The Altera FPGA is re-configurable by loading a new programming file into the device. The file can be generated with the standard Altera design software. The engineering kit includes a utility to load your file into the Altera FPGA. A reference file with our test configuration is also provided. The reference design has a pin configuration file, which can be reused for your specific implementation. The reference design is written in VHDL. The engineering kit also includes a cable and the HDEterm100. 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 http://www.dyneng.com/HDEterm100.html

Clocking options in the Altera FPGA are supported with 8 PLL devices. The PLLs are programmable with a frequency description file. The PLLs are loaded via the Altera. The reference design has an interface to the PLLs implemented along with software to load the PLLs with the requested frequency file. The Altera is connected to the 33 MHz PCI clock and 66.6667 MHz oscillator plus the PLL's. Either the 33 or the 66.6667 MHz can be routed to the PLLs. 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]. As part of our ATP our software determines the address of each PLL and prints it out. A label is attached to the shipping bag with the PLL addresses for the user's convenience. The software is part of the engineering kit and can be ported to your application.

The Altera can be programmed via the PCI bus or optionally from an installed FLASH PROM. If the PROM option is installed then the user can program the device with the provided header. The Altera will be loaded on power transitions automatically from the PROM. In some situations having the Altera programmed before waiting for the OS to start, and then the driver to load the Altera has benefits. When the PROM is installed a quickswitch is also installed to allow the Xilinx to re-load the Altera with PCI designs after power up.

Theory of Operation

A wide range of interfaces and protocols can be implemented with the PCI-Altera-485/LVDS. 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 PCI-Altera-485/LVDS along with the Quartus software.

Once your requirements are known the design can be implemented with VHDL, Verilog, or schematics and compiled with the Altera design software. The output file can then be "uploaded" to the Altera FPGA on the PCI-Altera-485/LVDS. 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. The PCI-Altera-485/LVDS has 40 differential IO. There are enough IO for 8 full duplex channels with a clock reference and two controls per channel. The PCI-Altera-485/LVDS 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 Altera 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 Altera plus all of the functions provided by the Xilinx. The designer would need to implement the IO interface.

The data flow for transmission would be Host memory transferred into the DMA FIFO via DMA transfers. From the DMA FIFO to the TX channel FIFO. The user state machine would read the data from the FIFO on the Altera side and apply the user protocol before serializing and transmitting. On the receive side the data would flow into the FPGA, be processed to convert to a format suitable for storing, and be written into the associated channels' RX FIFO. The data would be read from the RX FIFO by the Xilinx state-machine and be transferred into the DMA FIFO before being transferred to the host memory.

The Xilinx supports a command queue, DMA transfers, and FIFO level control. The user software can prioritize the channels by using interrupts and adjusting the command queue. The DMA transfers can be in either direction and can switch directions between commands. The full bandwidth of the PCI bus is utilized during DMA transfers. There is some overhead on the PCI bus side, which will limit the actual sustainable transfer rate somewhat compared to the theoretical limit. Looking at the other side of the equation: assuming serial data with 16 channels [8 tx and 8 rx] operating at 20 MHz creates a total of 10 Mwords per second on the PCI bus – approximately 30% loading from the theoretical maximum.

Once the state machine to control each channel is implemented the VHDL can be modified to provide the control bits and any new status to the bus interface and to instantiate the new IO.

Feature List Current

• User Defined Altera 20K series FPGA
• DMA capable 32/33 PCI bus interface
• Command buffer
• 16 FIFO paths – 8 dedicated “RX” and 8 dedicated “TX”
• 40 fully programmable RS485/422 or LVDS IO
• 12 TTL IO
• 8 position "DIP Switch"
• User LEDs
• Power LED
• On-going development with a "PROM" program

As Dynamic Engineering adds features to the hardware we will update the PCI-Altera-485/LVDS 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 PROM update. We will ship a new PROM with the updated program to you for shipping plus $25 per PROM. If you are interested please send a PO with shipping instructions, the serial numbers of the boards to upgrade and the programming charge.

The basic PCI identifying information will not change with the updates. The revision field will change to allow configuration control. Current revision is 0x0C.

Programming

The PCI-ALTERA-485/LVDS 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 C++ in conjunction with the driver to write our test software. Please consider purchasing the engineering kit for the PCI-ALTERA-485/LVDS; the software kit includes our test suite.

Before communication with the Xilinx or Altera devices can happen the PLX device requires some initialization. The local bus address space must be enabled and if interrupts are to be used the PLX must be enabled for those too.

Writing to the PLX local configuration address offset 0x4 [LAS0BA) with 0x01 will enable the local bus for memory space access, and re-map the local address to offset 0.

Writing to the Bus Region Descriptors [LBRD0] 0x18 with 0x40430343 will put the local bus into a well behaved state to inter-operate with the Xilinx and Altera. Specifically we are disabling the pre-fetch capability for the memory and ROM spaces. With the FIFO interfaces pre-fetching can lead to loss of data. More detail is available in the PLX 9054 HW design manual.

Writing 0x01200000 to [MARBR] 0x08 will set the Mode/DMA Arbitration to the correct state for operation.

To use interrupts from the Xilinx or Altera the [INTCSR] 0x68 will need to be programmed. 0x0F000900 will enable the local bus interrupt and PCI interrupt capability. To disable the local side interrupt clear bit 11.

Operation with DMA requires additional register programming within the PLX and Xilinx devices. The Dynamic Engineering Driver takes care of all of the initialization if it is used. Windows 2000 and XP are currently supported.

Access to the user FPGA is done with the same base address and all local addresses with address 13 set. 0x00000000 -> 0x0001FFF is allocated to the Xilinx and 0x0002000 -> 0x0003FFF to the user FPGA. The Xilinx decodes the lower 32 longwords. The internal registers for the Xilinx are defined in the following pages.

Address Map Xilinx

```
PCIALT_BASE       0x00000000 // base control register
PCIALT_INTEN      0x00000004 // interrupt enable register
PCIALT_STATUS     0x00000008 // status – read
PCIALT_DMAFIFO    0x0000000C // DMA FIFO read - write single word
PCIALT_FIFO_0     0x00000010 // TX_FIFO0 – write, RX_FIFO0 - read
PCIALT_FIFO_1     0x00000014 // TX_FIFO1 – write, RX_FIFO1 – read
PCIALT_FIFO_2     0x00000018 // TX_FIFO2 – write, RX_FIFO2 – read
PCIALT_FIFO_3     0x0000001C // TX_FIFO3 – write, RX_FIFO3 – read
PCIALT_FIFO_4     0x00000020 // TX_FIFO4 – write, RX_FIFO4 - read
PCIALT_FIFO_5     0x00000024 // TX_FIFO5 – write, RX_FIFO5 - read
PCIALT_FIFO_6     0x00000028 // TX_FIFO6 – write, RX_FIFO6 - read
PCIALT_FIFO_7     0x0000002C // TX_FIFO7 – write, RX_FIFO7 - read
PCIALT_COMMAND    0x00000030 // cmd queue – write, cmd count - read
PCIALT_SWITCH     0x00000034 // User Switch read back port
PCIALT_CMDSTAT    0x00000038 // command status - read
PCIALT_INTSTAT    0x0000003C // int stat clr – write, int stat – read
PCIALT_DMAFIFO    0x00000040 // DMA FIFO read - write DMA access
```

FIGURE 2 PCI-ALTERA-485/LVDS XILINX ADDRESS MAP

The address map provided is for the local decoding performed within PCI-Altera-485/LVDS Xilinx. The addresses are all offsets from a base address. The base address and interrupt level is provided by the host in which the PCII-Altera-485 is installed.

The host system will search the PCI bus to find the assets installed during power-on initialization. The VendorId = 0x10b5 and the CardId = 0x9054 for the PCI-Altera-485/LVDS. Interrupts are requested by the configuration space. PCIView and other 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 the PCI-Altera-485/LVDS card, the software will need to determine what the address space is. We refer to this address as base0 in our software.

The next step is to initialize the PCI-Altera-485/LVDS. The local Xilinx registers need to be configured and then the Altera loaded.

Register Definitions

PCIALT_BASE

[0x0000 Main Control Register Port read/write]

BASE REGISTER

DATA BIT	DESCRIPTION
31-24	fout_pae_ld(7..0) – individual FIFO almost empty level loads
23-16	fout_rst(7..0) – individual FIFO resets
15	ax_lnk_dir - direction of link signal 1 = Xilinx driving, 0 = tristate
14	ax_lnk_s - signal to/from altera
13	m_int_en - interrupt driver enable, master interrupt enable
12	force_int - cause an interrupt by enabling this bit
11	spare
10	cmd_rst - '1' resets command store and executive, '0' normal operation
9	ld_altera - '1' enables Altera configuration load - self clearing
8	led_on - '1' lights Xilinx LED, '0' turns LED off
7	en_wr_dma - enable the write FIFO state machine for DMA operation
6	en_rd_dma - enable the read FIFO state machine for DMA operation
5	en_wr_std - enable the write FIFO state machine for single word access
4	en_rd_std - enable the read FIFO state machine for single word access
3	fout_paelda '1' - programming almost mt level, '0' - Tx FIFO data writes
2	fout_rsta '1' resets Tx FIFOs, '0' for normal operation
1	spare
0	dma_rst '1' resets DMA FIFO, '0' for normal operation

FIGURE 3 PCI-ALTERA-485/LVDS XILINX BASE CONTROL REGISTER

Fout_paeld(7..0) -- '1' enables programming individual Tx FIFO almost empty levels, '0' allows Tx FIFO data writes.

Fout_rst(7..0) -- '1' resets individual Tx FIFOs, '0' for normal operation.

The AX_LNK_DIR bit is used to select the direction of the Xilinx – Altera link bit. A connection between the Altera and the Xilinx for user defined purposes and to provide a method for the Altera to create an interrupt request. If the bit is ‘1’ then the Xilinx is the driver and the Altera the receiver. The default is ‘0’ where the Altera will drive and the Xilinx receive the data.

AX_LNK_S is a register bit which defines the level of the AX_LNK when the Xilinx is programmed to be the master of the signal.

M_INT_EN is the master interrupt enable for all interrupts on the PCI-Altera-485/LVDS which are controlled by the Altera or Xilinx devices. Please note that the PLX interrupt

enable must also be enabled for a PCI interrupt to be generated. Default is disabled. When '1' the master enable is "enabled".

FORCE_INT when '1' and the master enabled causes an interrupt to be generated. This bit is useful for software debugging. Please note that this is a new feature on the second fab revision.

The CMD_RST bit when '1' clears the command queue.

LD_ALTERA when set to '1' begins the process of programming the Altera device. The data must be loaded into the DMA FIFO via software. When the hardware detects that the load is complete this bit is automatically cleared.

LED_ON when '1' enables the Xilinx controlled LED. During Altera loading the LED will toggle on and off with the data.

EN_WR_DMA when '1' enables the state-machine to support DMA transferred write operations into the DMA FIFO.

EN_RD_DMA when '1' enables the state-machine to support DMA transferred read operations into the DMA FIFO.

EN_WR_STD when '1' enables the state-machine to support single word write operations into the DMA FIFO.

EN_RD_STD when '1' enables the state-machine to support single word read operations into the DMA FIFO.

Fout_paelda when '1' selects the programming mode for the Programmable Almost Empty flag on all eight TX FIFOs. Please note that the PAF for the RX FIFO is programmed via the Altera. '0' is normal operation.

To program the FIFO flag first set the Fout_paelda bit, then set and clear the Fout_rsta bit. The next sequence of data written to the FIFOs will program the flags. When the sequence is completed take Fout_paelda low. Please note that you do not reset the part again. If reset occurs after programming the flags, the flags will revert to the default values.

The 4k part used on the PCI-Altera-485/LVDS has a 12-bit range for the PAE and PAF flags. The PAE flag is written first. The default is 7. The lower 8 bits are written followed by the upper 4. Bit positions 7-0 and 3-0 respectively. The PAF flag is not required to be written in order to program the PAE. If the PAF value is not written then it will require a reset to re-write the PAE values. If the PAF values are written then the pointer is returned to the PAE location. The PAE value can be re-written by setting the

PAELD bit and repeating the load sequence. Each of the TX FIFOs has an independent write port address allowing different values to be programmed for each channel.

Fout_rsta when ‘1’ resets all eight TX FIFO. Set to ‘0’ for normal operation.

To guarantee proper operation the FIFOs should be reset after power up. Set the reset control bit and then reset the control bit. The reset should be applied after the clocks are stable. The reset signal meet set- and hold times. The Xilinx takes care of these requirements automatically. The on time is short enough that software can toggle the bit as rapidly as desired.

DMA_RST when set to ‘1’ will reset the DMA FIFO. Set to ‘0’ for normal operation.

To guarantee proper operation the FIFOs [TX and DMA] should be reset after power up. Set the reset control bit and then clear the control bit. The reset should be applied after the clocks are stable. The reset must be “seen” by the FIFO. Apply the reset condition for a period longer than the clock period used on the FIFOs. The Xilinx side of the interface is referenced to the PCI clock – back to back operations will not violate the timing specification of 31+ nS.

PCIALT_INTEN

[0x0004 Interrupt Enable Port read/write]

INTERRUPT ENABLE REGISTER

DATA BIT	DESCRIPTION
20	inten_rx_dn
19	inten_tx_dn
18	inten_cmd_dn
17	inten_axlnk
16	inten_alt_dn
15-8	PAE 7-0
7-0	PAF 7-0

FIGURE 4 PCI-ALTERA-485/LVDS INTERRUPT ENABLE PORT

When PAF7..0 are set the corresponding Programmable Almost Full interrupt is enabled [FIFO 7 through 0].

When PAE7-0 are set the corresponding Programmable Almost Empty interrupt is enable [FIFO 7 through 0].

Inten_alt_dn when ‘1’ enables the interrupt associated with the Altera being done loading – fully programmed.

Inten_axlnk when ‘1’ enables the interrupt associated with the Xilinx – Altera link bit. Please note that the Altera should be the driver and the Xilinx the receiver if this bit is used to generate an interrupt.

Inten_cmd_dn when ‘1’ enables the command completion interrupt. If enabled, this interrupt will be asserted whenever a Tx FIFO load or Rx FIFO read command completes.

Inten_tx_dn when ‘1’ enables the Tx command completion interrupt. If enabled, this interrupt will be asserted whenever a Tx FIFO load command completes.

Inten_rx_dn when ‘1’ enables the Rx completion interrupt. If enabled, this interrupt will be asserted whenever an Rx FIFO read command completes.

PCIALT_STATUS

[0x0008 PCI-ALTERA-485/LVDS status register read only]

STATUS REGISTER

DATA BIT	DESCRIPTION
31-24	Xilinx revision
23	interrupt 1 = active request
22	ax_lnk state of AX_LNK IO line
21	alt_stat
20	alt_done
19	valid
18	dma_ff
17	undefined
16	dma_mt
15 - 8	FIFO OUT PAE 7 - 0
7 - 0	FIFO IN PAF 7 - 0

FIGURE 5 PCI-ALTERA-485/LVDS STATUS PORT

Interrupt when ‘1’ indicates that the Xilinx has an interrupt request pending. The Interrupt request is gated by the PLX device. The PLX must also be enabled for the interrupt request to become a PCI interrupt request. The Xilinx interrupt request can be from any of the interrupt sources in the Xilinx or Altera.

The AX_LNK bit mirrors the state of the AX_LNK signal between the Altera and Xilinx devices. Read the state of the bit here. Control the state of the bit in the base control register.

ALT_STAT = nSTATUS from Altera device

ALT_DONE = CONF_DONE from Altera device

VALID ‘1’ indicates data is available to read from the DMA FIFO. Please note that this bit can be set even when the MT flag is also set due to the pre-read feature built into the hardware to increase performance.

DMA_FF when ‘1’ indicates that the DMA FIFO is full.

DMA_MT when ‘1’ indicates that the DMA FIFO is empty.

FIFO OUT PAE are the eight “TX” FIFO Programmable Almost Empty flags. When ‘1’ the associated FIFO is “almost empty”. An interrupt can be generated when the FIFO becomes almost empty to queue another DMA transfer to that channel.

FIFO IN PAF are the eight “RX” FIFO Programmable Almost Full flags. When ‘1’ the associated FIFO is almost full. An interrupt can be generated when the FIFO becomes almost full to queue another DMA transfer from that channel.

The FIFO flag interrupts can happen in any order at any time depending on the interface defined in the Altera and external events. The status register must be read to determine which interrupt(s) are active. The software can then prioritize the command sequence to take care of the channels that require more data or need to have data read from them.

The bits in this register are unlatched and unmasked. The Interrupt Status latch contains latched data bits corresponding to each channel.

PCIALT_DMAFIFO

[0x000C, 0x0040 DMA FIFO PORT]

DMA FIFO Port	
DATA BIT	**DESCRIPTION**
31..0	FIFO data 31..0

FIGURE 6 PCI-ALTERA-485/LVDS DMA FIFO PORT

The DMA FIFO can be written to or read from via the PCI bus. The DMA FIFO can be accessed with “standard” [0x00C] target accesses or via DMA[0x0040]. The base control register must be properly programmed before accessing this port.

PCIALT_FIFO_N

[0x0010, 0x0014, 0x0018, 0x001C, 0x0020, 0x0024, 0x0028, 0x002C Altera-Xilinx FIFO PORTs]

RX and TX FIFO FIFO Ports	
DATA BIT	**DESCRIPTION**
7..0	FIFO Data 7..0

FIGURE 7 PCI-ALTERA-485/LVDS RX/TX FIFO PORTS

A write to the ports causes a write to the corresponding TX FIFO. A read from the port causes a read from the corresponding RX FIFO. The data width of the FIFO is 8 bits. When accessed via the DMA FIFO using the Xilinx state-machines the data is automatically converted to 32-bit and vice-versa. The direct read and write ports are used for special conditions – for example if an RX channel is stopped and there is data present in the FIFO, but not enough to trigger an interrupt request, the data can be read directly.

The ports can be used for loop-back testing with DMA'd or direct writes to the TX FIFO followed by reads from the TX FIFO port on the Altera side and vice-versa with the RX FIFOs. The reference design includes a state-machine implementation to loop data from the TX to the RX FIFO automatically. Allowing loop testing from the Xilinx port.

PCIALT_COMMAND

[0x0030 Command PORT]

COMMAND Port	
DATA BIT	**DESCRIPTION**
24	Direction 1 = TX 0 = RX
23..16	Channel Number 7 – 0
15	undefined
14 – 0	Word Count

FIGURE 8 PCI-ALTERA-485/LVDS COMMAND PORT

Each time data is written to the command port a new command is added to the Command Stack. The command can be associated with multiple channels for TX and only one channel for RX operations.

The Direction bit defines the direction of data flow. When '1' data flows from the host to the TX FIFOs. When '0' data flows from the RX FIFOs to the host.

The channel number selects the active channel(s). In receive mode only one channel can be processed per command. If more than one channel is selected with the Direction bit set to '0' then the command is ignored. In transmit mode multiple channels can transmit the same data. If this is desired then activate more than one channel in the Channel Number field. This will save PCI bus band width and not "cost" any extra loading time as the data is written in parallel to the multiple channels.

The word count is the number of 32-bit words to transfer from the DMA FIFO to the TX FIFO or from the RX FIFO to the DMA FIFO. For non-longword data boundaries the direct read / write capability can be used to complete the last 1-3 bytes.

Reading from this port returns the number of commands waiting to be processed not including the one currently executing. The command queue can hold up to 8 commands to process.

Please note that the commands will be executed in the order written. The commands written can be in any order of channel or direction. There are advantages to grouping the reads and writes together. For example a larger DMA transfer can break down to several smaller channel commands and be handled automatically in the hardware.

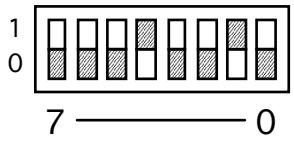
PCIALT_SWITCH

[0x0034 User Switch Port read only]

DipSwitch Port	
DATA BIT	**DESCRIPTION**
7..0	Sw7..0

FIGURE 9 PCI-ALTERA-485/LVDS USER SWITCH PORT

The user switch is read through this port. The bits are read as the lowest byte. Access the port as a long word and mask off the undefined bits. Read only. The dip-switch positions are defined in the silkscreen. For example the switch figure below indicates a 0x12.



PCIALT_CMDSTAT

[0x0038 Command Status PORT]

Command Status Ports

DATA BIT	DESCRIPTION
27-20	Active Channel 7-0
19	CMD RDY
18	CMD FULL
17	Running
16	DIR
15	‘0’
14..0	FIFO Count

FIGURE 10 PCI-ALTERA-485/LVDS COMMAND STATUS PORT

Active channel(s) indicates which channels are currently being serviced.

Command Ready [CMD RDY] if set indicates that there is at least one command waiting to be executed.

Command Full [CMD FULL] when set ‘1’ indicates that the command queue has 8 commands waiting and is full.

Running when set ‘1’ indicates that a command is currently being processed.

DIR when set ‘1’ indicates that the current command is in the TX direction, when ‘0’ the command being executed is in the RX direction.

The FIFO count is the number of 32-bit words in the DMA FIFO.

PCIALT_INTSTAT

[0x003C Interrupt Status PORT]

Command Status Ports

DATA BIT	DESCRIPTION
24	int_stat
23 - 21	spare
20	rx_dn
19	tx_dn
18	cmd_dn
17	axlnk
16	alt_dn
15 - 8	TX7-0_AE
7 - 0	RX7-0_AF

FIGURE 11 PCI-ALTERA-485/LVDS INTERRUPT STATUS PORT

Int_stat when ‘1’ indicates that an enabled interrupt condition is true. If the master interrupt enables are “true” then the interrupt request will be driven onto the PCI bus.

rx_dn, tx_dn, cmd_dn indicate that a receive, transmit, or either command has completed.

If AX_LNK is set-up to be an interruptor then the Altera can cause an interrupt to the PCI bus via the Xilinx. The axlnk bit will be set in this case. The bit is captured and held until explicitly cleared.

When programming the Altera device the alt_dn bit indicates that the process is complete. The bit can be used as an interrupt generator. The bit is held until explicitly cleared.

The Programmable Almost Empty and Programmable Almost Full flags are captured on the transition to active and held until explicitly cleared. The associated interrupt would be used to keep the data flowing to the transmit ports and to prevent overflow on the receive ports. The software can implement a priority based algorithm for processing the interrupts when more than one channel is active at a time.

To clear the status, write to the port with the associated bit set.

Address Map Altera

PCIALT_BASEA	0x00002000 // base control register
PCIALT_RXTX	0x00002004 // Enables for the 8 Tx and 8 Rx state machines
PCIALT_DIR1	0x00002008 // Direction control register lower 20 bits
PCIALT_DIR2	0x0000200C // Direction control register upper 20 bits
PCIALT_TERM1	0x00002010 // Termination control register lower 20 bits
PCIALT_TERM2	0x00002014 // Termination control register upper 20 bits
PCIALT_IO1	0x00002018 // 485/LVDS IO lower 20 bits
PCIALT_IO2	0x0000201C // 485/LVDS IO upper 20 bits
PCIALT_FIFO_0A	0x00002020 // TX_FIFO0 – read, RX_FIFO0 - write
PCIALT_FIFO_1A	0x00002024 // TX_FIFO1 – read, RX_FIFO1 - write
PCIALT_FIFO_2A	0x00002028 // TX_FIFO2 – read, RX_FIFO2 - write
PCIALT_FIFO_3A	0x0000202C // TX_FIFO3 – read, RX_FIFO3 - write
PCIALT_FIFO_4A	0x00002030 // TX_FIFO4 – read, RX_FIFO4 - write
PCIALT_FIFO_5A	0x00002034 // TX_FIFO5 – read, RX_FIFO5 - write
PCIALT_FIFO_6A	0x00002038 // TX_FIFO6 – read, RX_FIFO6 - write
PCIALT_FIFO_7A	0x0000203C // TX_FIFO7 – read, RX_FIFO7 - write
PCIALT_TTL	0x00002040 // Write - TTL data out, Read - TTL data bus in
PCIALT_FIFOSTAT1	0x00002044 // Altera FIFO empty, full status port
PCIALT_OSC_CNT	0x00002048 // Oscillator/Counter Mux Control Port
PCIALT_OSC_DATA	0x0000204C // Counter Data Read Port
PCIALT_FIFOSTAT2	0x00002050 // Altera FIFO programmable almost empty/full status port

FIGURE 12 PCI-ALTERA-485/LVDS ALTERA-ATP ADDRESS MAP

The Altera FPGA is completely programmable – the address map above and definitions below only have meaning if the Engineering Kit is used as a starting point for your design. The reference software and reference Altera hardware implementation are used to perform the ATP on each board prior to shipment. The Engineering kit also includes the HDEterm100 and a cable to interconnect the PCI-Altera-485/LVDS with the HDEterm100. For more information please refer to the web page.

Whether you use the reference design or not please note that the Altera decoding starts at address 0x2000.

PCIALT_BASEA

[0x2000 Main Control Register Port read/write]

BASE REGISTER

DATA BIT	DESCRIPTION
31-24	Design ID (read only when lockout = ‘0’)
28	Design ID lockout
27-24	LED 3-0 (write only unless lockout = ‘1’)
23	Lintn status - Asserted = ‘0’ (read only)
22	Local_int status - Asserted = ‘1’ (read only)
21	Force_int - cause an interrupt by enabling this bit
20	M_int_en - interrupt driver enable, master interrupt enable
19	FRX_LD
18	FRX Reset
17	PLL enable
16-9	s2 data
8	i2c Data
7-0	i2c clock PLL 7 - 0

FIGURE 13 PCI-ALTERA-485/LVDS ALTERA BASE CONTROL REGISTER

The PCI-Altera-485/LVDS 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 and upper “data” bit.

The common data line has a pull-up on the board. When the PLL is enabled and the i2c data bit is set to ‘0’ in the register the external line is driven low. When not enabled or i2c data is set to ‘1’ in the register, the external line is tri-stated and pulled-up by the resistor. For a read operation the data should be set to ‘1’ to allow the pll to drive the line.

The upper selection bit can be set in the register and directly driven to the PLLs. With separate i2c clock lines each of the PLLs can be programmed independently.

The clock line for the PLL to be programmed is toggled along with the data to create a bit stream with a “software clock”. Set the bit to the next state and toggle the clock line and repeat.

To read over the i2c bus a command is first written and then the bus read for the response. The i2c data register bit contains the state of the bus when read. The software will toggle the clock line and when the low-to-high transition is made, read the data bit then repeat until the message is captured.

The engineering kit contains the logic and software required to program the PLL and to read the programmed frequency back. The software to determine the frequency

command words is available from Cypress Semiconductor. The part number is CY22393FC. Cypress has a utility available for calculating the frequency control words for the PLLs. http://www.dyneng.com/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 our ATP our software determines the address of each PLL and prints it out. A label is attached to the shipping bag with the PLL addresses for the user's convenience. The software is part of the engineering kit and can be ported to your application.

When FRX Reset = '1' all of the receive FIFOs are reset. The default = '0' and is used for normal operation.

When FRX_LD is set to '1' the programmable flag for the receive FIFOs can be accessed. Make FRX_LD = '0' for normal operation.

The TX FIFOs are controlled from the Xilinx side of the interface. The RX FIFOs are controlled from the Altera side of the interface.

To guarantee proper operation, the FIFOs should be reset after power up. Set the reset control bit and then reset the control bit. The reset should be applied after the clocks are stable. The reset must be "seen" by the FIFO. Apply the reset condition for a period longer than the clock period used on the FIFOs.

To program the FIFO [PAF] flag first set the FRX_LD bit, then set and clear the FOUT_Reset bit. The next sequence of data written to the FIFOs will program the flags. When the sequence is completed take FRX_LD low. Please note that you do not reset the part again. If reset occurs after programming the flags, the flags will revert to the default values.

The 4k part used on the PCI-Altera-485/LVDS has a12 bit range for the PAE and PAF flags. The PAE flag is written first. The default is 7. The lower 8 bits are written followed by the upper 4. Bit positions 7-0 and 3-0 respectively. The PAE flag is written to in order to get to the programming position for the PAF. The PAF value can be re-written by setting the FRX_LD bit and repeating the load sequence. Each of the RX FIFOs has an independent write port address allowing different values to be programmed for each channel.
M_INT_EN is the master interrupt enable for Altera. Please note that the PLX interrupt enable must also be enabled for a PCI interrupt to be generated. Default is disabled. When '1' the master enable is "enabled".

FORCE_INT when '1' and the master enabled causes an interrupt to be generated. This bit is useful for software debugging.

The LEDs are enabled when the corresponding bit is set. The LEDs are for user purposes. One suggestion is to use them for debugging your Altera – the LEDs could be used to indicate what state the hardware is in or when a channel is active etc. The reference design uses them as register bits which could be used for software interaction.

PCIALT_RX_TX

[0x2004 Rx/Tx Control Register Port read/write]

BASE REGISTER

DATA BIT	DESCRIPTION
20	Ax_link_stat (read only)
17	Ax_link_dir '1' = output, '0' = input
16	Ax_link_dat
15 - 8	strt_tx7-0
7 - 0	strt_rx7-0

FIGURE 14 PCI-ALTERA-485/LVDS ALTERA BASE CONTROL REGISTER

The start bits are used to enable and disable the Rx and Tx state machines from running in the reference design. The state machines move data from a Tx FIFO to the corresponding Rx FIFO provided the Tx is not empty and the Rx is not full.

The Ax_link line is a bi-directional line between the Xilinx and Altera. The Ax_link_dir signal controls the direction of the line on the Altera side (1 = output, 0 = input).

The Ax_link_dat defines the logic state when the line is configured as an output.

The Ax_link_stat signal is used to read the state of the line when it is configured as an input.

PCIALT_DIR1,2

[0x2008, 0x200C Direction Control Ports read/write]

485/LVDS DIRECTION REGISTERS

DATA BIT	DESCRIPTION
19-0	dir39 - 20, dir19 - 0

FIGURE 15 PCI-ALTERA-485/LVDS DIRECTION CONTROL REGISTERS

The Direction Control registers are used to select transmit or receive on each of the RS-

485 lines. The Altera internal circuitry and the differential transceiver are affected by the selections made. '1' = transmit and '0' = receive in a particular bit position. All IO are independent. All IO default to receive on reset / power on.

PCIALT_TERM1,2

[0x2010, 0x2014 Termination Control Ports read/write]

485/LVDS TERMINATION REGISTERS	
DATA BIT	**DESCRIPTION**
19-0	term39 - 20, term19 - 0

FIGURE 16 PCI-ALTERA-485/LVDS TERMINATION CONTROL REGISTERS

The Termination Control registers are used to select the state of the programmable termination on each of the RS-485 differential pairs. '1' = activate termination and '0' = isolate termination in a particular bit position. All IO are independent. All IO default to not terminated. Please note that the termination is independent of the direction.

PCIALT_IO1,2

[0x2018, 0x201C IO Control Ports read/write]

485/LVDS IO REGISTERS	
DATA BIT	**DESCRIPTION**
19-0	io39 - 20, io19 - 0

FIGURE 17 PCI-ALTERA-485/LVDS IO CONTROL REGISTERS

For each bit in the IO control register that has the corresponding Direction bit set to transmit the output is enabled. When enabled the data in the IO register is driven onto the IO line to the transceiver and out onto the differential pair. If the direction bit is not set the corresponding bit in the io register is not driven onto the IO line. A read from the IO Register will return the value on the IO lines. If the direction is set to transmit then the value returned should match the value written to the register. If the direction is set to receive then the value returned will be set by the "system". The software should map out the bits set to transmit when reading from this port.

Application Note: Spare IO

Frequently a system will need some dedicated IO but not all of the IO. If the registers are left in the design and the intermediate buses used to combine with your

implementation the spare bits can be used as an auxiliary port with very little effort. Comment out the “io” definition for the lines that are under your state-machine control as well as the direction and termination bits [if you need real time control] then replace those definitions with your state-machine implementation.

PCIALT_FIFO_NA

[0x2020, 0x2024, 0x2028, 0x202C, 0x2030, 0x2034, 0x2038, 0x203C Altera FIFO Ports read/write]

FIFO PORTs	
DATA BIT	**DESCRIPTION**
7-0	FIFO 7 - 0

FIGURE 18 PCI-ALTERA-485/LVDS ALTERA FIFO PORTS

The Altera side of the RX can be written and the TX FIFOs can be read from these ports. Useful for loop-back testing. In normal operation the State-machine(s) within the Altera would be used to load data or read data from these ports.

PCIALT_TTL

[0x201C TTL Control Ports read/write]

TTL IO REGISTERS	
DATA BIT	**DESCRIPTION**
11-0	TTL 11 - 0

FIGURE 19 PCI-ALTERA-485/LVDS TTL CONTROL REGISTER

The TTL IO are designed with a ‘125 style gate and read-back buffer. The ‘125 provides an “open drain” tri-state gate. The PCI-Altera-485/LVDS has a pull-up on each line. With revision H and later hardware the enable for each ‘125 gate is tied to the Altera. The default footprint ties the enable to the data line for commonality with the previous versions. For increased TTL switching performance use the enables separately to leave the 125’s always enabled to benefit from the high side current, and improved switch delay time compared to tri-state delay time. When the gate is enabled the corresponding line is pulled low. When the gate is disabled the line is pulled-high with the pull-up.

Search for DATAEN within the UserApp.vhd module to make changes.

The TTL port resets to “FFF” to cause the IO to be in the tri-stated condition. The

software can write a ‘1’ or ‘0’ to any bit and cause the ‘1’ or ‘0’ on the IO line. If an external line is driving the IO bit then the line may remain at ‘0’ when set to the undriven state.

The read port returns the state of the IO lines – not necessarily the same as the write port.

PCIALT_FIFOSTAT1

[0x2044 FIFO Status Port read/write]

FIFO Status PORT

DATA BIT	DESCRIPTION
31 - 24	FTX FF 7 - 0
23 – 16	FTX MT 7 - 0
15 - 8	FRX FF 7 - 0
7 – 0	FRX MT 7 - 0

FIGURE 20 PCI-ALTERA-485/LVDS FIFO STATUS PORT1

The status from the Rx and Tx FIFOs is available in this port. The status bits are active high. When ‘1’ the status is active and when ‘0’ the status is inactive.

FTX FF = FIFO Transmit Full Flag. When ‘1’ the FIFO associated with the bit is full. When ‘0’ the FIFO is not full.

FTX MT = FIFO Transmit Empty Flag. When ‘1’ the FIFO associated with the bit is empty. When ‘0’ the FIFO is not empty.

FRX FF = FIFO Receive Full Flag. When ‘1’ the FIFO associated with the bit is full. When ‘0’ the FIFO is not full.

FRX MT = FIFO Receive Empty Flag. When ‘1’ the FIFO associated with the bit is empty. When ‘0’ the FIFO is not empty.

PCIALT_OSCCNT

[0x2048 Oscillator Control Register read/write]

Oscillator Control Register

DATA BIT	DESCRIPTION
30	CNT EN
29	CNT CLR
28 - 24	MUX select
23	PLL7_c
22	PLL7_b
21	PLL7_a
20	PLL6_c
19	PLL6_b
18	PLL6_a
17	PLL5_c
16	PLL5_b
15	PLL5_a
14	PLL4_c
13	PLL4_b
12	PLL4_a
11	PLL3_c
10	PLL3_b
9	PLL3_a
8	PLL2_c
7	PLL2_b
6	PLL2_a
5	PLL1_c
4	PLL1_b
3	PLL1_a
2	PLL0_c
1	PLL0_b
0	PLL0_a

FIGURE 21 PCI-ALTERA-485/LVDS OSCILLATOR CONTROL REGISTER

The PLLn_abc bits when set [‘1’] cause the corresponding IO bits to be controlled by the PLL clock as indicated. A clock waveform corresponding to the programmed PLL selected is driven to the IO. Please note that the direction register must also be programmed for the selected bits. The bits not selected are controlled by the IO registers. The upper bits: 39 – 24, are not affected by the pcialt485_osccnt register.

The CNT CLR bit when set clears the counters that count the PLL clock transitions. The CNT CLR function is used to reset the counters to a known state for functional test.

CNT EN when ‘1’ enables the counters to track the transitions of the PLL clocks. Each PLL output has a 32-bit counter. In addition a master counter based on the oscillator input frequency [66.6667 MHz] is available for reference. When bit 24 of the master counter transitions high [approximately 1/4 sec.], all counters stop counting. The counts

can be read to verify the frequency of the output clock.

The MUX select bits are used to select the counter to read. There is only one port to read the 24 counters from. The MUX select bits select which one drives the OSC Data port. The decoding is a straight binary encoding of the counter number corresponding to the control bit number in the Oscillator Control Register definition table. “00000” selects PLL0_a and “10000” selects PLL5_b etc. The master count is available on select 24.

PCIALT_OSCDAT

[0x204C Oscillator Data Port]

Oscillator Data PORT	
DATA BIT	**DESCRIPTION**
31 - 0	Counter Value

FIGURE 22 PCI-ALTERA-485/LVDS OSCILLATOR DATA PORT

When read the OSCDAT port provides the data from the counter selected with the mux in the OSCDAT register. The ports are used for checking that the PLLs are functioning correctly. The Master clock value is captured in an additional counter. The relative counts of the different PLLs can be compared with the reference count and checked for proper operation.

PCIALT_FIFOSTAT2

[0x2044 FIFO Status Port read/write]

FIFO Status PORT	
DATA BIT	**DESCRIPTION**
15 - 8	FTX PAE 7 - 0
7 – 0	FRX PAF 7 - 0

FIGURE 23 PCI-ALTERA-485/LVDS FIFO STATUS PORT2

The status from the Rx and Tx FIFOs is available in this port. The status bits are active high. When ‘1’ the status is active and when ‘0’ the status is inactive.

FTX PAE = FIFO Transmit Programmable Almost Empty flag. When ‘1’ the FIFO associated with the bit is almost empty. When ‘0’ the FIFO has more data than the Almost Empty level.

FRX PAF = FIFO Receive Programmable Almost Full. When ‘1’ the FIFO associated

with the bit is almost full. When ‘0’ the FIFO has more room than the Almost Full level is set to.

Altera Pin Out

The FPGA pin definitions are contained in the engineering kit and repeated here as a reference. The hardwired pins for power and ground are not shown.

CHIP "top" ASSIGNED TO AN EP20K400EBC652-3

DIR[0]	A2	output	3.3-V PCI
TERM[0]	A3	output	3.3-V PCI
IO[1]	A4	bidir	3.3-V PCI
IO[2]	A5	bidir	3.3-V PCI
TERM[3]	A6	output	3.3-V PCI
IO[5]	A7	bidir	3.3-V PCI
DIR[7]	A8	output	3.3-V PCI
TERM[8]	A9	output	3.3-V PCI
IO[10]	A10	bidir	3.3-V PCI
DIR[12]	A11	output	3.3-V PCI
DIR[13]	A12	output	3.3-V PCI
TERM[14]	A13	output	3.3-V PCI
TERM[17]	A15	output	3.3-V PCI
DIR[19]	A16	output	3.3-V PCI
IO[21]	A20	bidir	3.3-V PCI
DIR[23]	A21	output	3.3-V PCI
TERM[24]	A22	output	3.3-V PCI
IO[26]	A23	bidir	3.3-V PCI
IO[27]	A24	bidir	3.3-V PCI
DIR[29]	A25	output	3.3-V PCI
TERM[30]	A26	output	3.3-V PCI
IO[32]	A27	bidir	3.3-V PCI
DIR[34]	A28	output	3.3-V PCI
TERM[35]	A29	output	3.3-V PCI
DIR[37]	A30	output	3.3-V PCI
DIR[38]	A31	output	3.3-V PCI
TERM[38]	A32	output	3.3-V PCI
IO[39]	A33	bidir	3.3-V PCI
TERM[39]	A34	output	3.3-V PCI
IO[0]	B3	bidir	3.3-V PCI
DIR[1]	B4	output	3.3-V PCI
DIR[2]	B5	output	3.3-V PCI
IO[3]	B6	bidir	3.3-V PCI
DIR[5]	B7	output	3.3-V PCI
TERM[6]	B8	output	3.3-V PCI
IO[8]	B9	bidir	3.3-V PCI
DIR[10]	B10	output	3.3-V PCI

TERM[11]	B11	output	3.3-V PCI
TERM[12]	B12	output	3.3-V PCI
IO[14]	B13	bidir	3.3-V PCI
DIR[16]	B14	output	3.3-V PCI
DIR[17]	B15	output	3.3-V PCI
TERM[18]	B16	output	3.3-V PCI
IO[19]	B17	bidir	3.3-V PCI
TERM[19]	B19	output	3.3-V PCI
DIR[21]	B20	output	3.3-V PCI
TERM[22]	B21	output	3.3-V PCI
IO[24]	B22	bidir	3.3-V PCI
DIR[26]	B23	output	3.3-V PCI
DIR[27]	B24	output	3.3-V PCI
TERM[28]	B25	output	3.3-V PCI
IO[30]	B26	bidir	3.3-V PCI
DIR[32]	B27	output	3.3-V PCI
TERM[33]	B28	output	3.3-V PCI
DIR[35]	B29	output	3.3-V PCI
TERM[36]	B30	output	3.3-V PCI
TERM[37]	B31	output	3.3-V PCI
IO[38]	B32	bidir	3.3-V PCI
DIR[39]	B33	output	3.3-V PCI
IOP[0]	C1	bidir	3.3-V PCI
TERM[1]	C5	output	3.3-V PCI
DIR[3]	C6	output	3.3-V PCI
TERM[4]	C7	output	3.3-V PCI
IO[6]	C8	bidir	3.3-V PCI
DIR[8]	C9	output	3.3-V PCI
TERM[9]	C10	output	3.3-V PCI
IO[11]	C11	bidir	3.3-V PCI
IO[12]	C12	bidir	3.3-V PCI
DIR[14]	C13	output	3.3-V PCI
TERM[15]	C14	output	3.3-V PCI
IO[17]	C15	bidir	3.3-V PCI
TERM[20]	C20	output	3.3-V PCI
IO[22]	C21	bidir	3.3-V PCI
DIR[24]	C22	output	3.3-V PCI
TERM[25]	C23	output	3.3-V PCI
TERM[26]	C24	output	3.3-V PCI
IO[28]	C25	bidir	3.3-V PCI
DIR[30]	C26	output	3.3-V PCI
TERM[31]	C27	output	3.3-V PCI
IO[33]	C28	bidir	3.3-V PCI
IO[35]	C29	bidir	3.3-V PCI

IO[36]	C30	bidir	3.3-V PCI
IO[37]	C31	bidir	3.3-V PCI
IOP[1]	D1	bidir	3.3-V PCI
TERM[2]	D6	output	3.3-V PCI
IO[4]	D7	bidir	3.3-V PCI
DIR[6]	D8	output	3.3-V PCI
TERM[7]	D9	output	3.3-V PCI
IO[9]	D10	bidir	3.3-V PCI
DIR[11]	D11	output	3.3-V PCI
TERM[13]	D13	output	3.3-V PCI
IO[15]	D14	bidir	3.3-V PCI
TERM[16]	D15	output	3.3-V PCI
IO[18]	D16	bidir	3.3-V PCI
IO[20]	D20	bidir	3.3-V PCI
DIR[22]	D21	output	3.3-V PCI
TERM[23]	D22	output	3.3-V PCI
DIR[25]	D23	output	3.3-V PCI
DIR[28]	D25	output	3.3-V PCI
TERM[29]	D26	output	3.3-V PCI
IO[31]	D27	bidir	3.3-V PCI
DIR[33]	D28	output	3.3-V PCI
TERM[34]	D29	output	3.3-V PCI
DIR[36]	D30	output	3.3-V PCI
DOUT[0]	D35	output	3.3-V PCI
IOP[2]	E1	bidir	3.3-V PCI
IOP[3]	E2	bidir	3.3-V PCI
DIR[4]	E7	output	3.3-V PCI
TERM[5]	E8	output	3.3-V PCI
IO[7]	E9	bidir	3.3-V PCI
DIR[9]	E10	output	3.3-V PCI
TERM[10]	E11	output	3.3-V PCI
IO[13]	E13	bidir	3.3-V PCI
DIR[15]	E14	output	3.3-V PCI
IO[16]	E15	bidir	3.3-V PCI
DIR[18]	E16	output	3.3-V PCI
DIR[20]	E20	output	3.3-V PCI
TERM[21]	E21	output	3.3-V PCI
IO[23]	E22	bidir	3.3-V PCI
IO[25]	E23	bidir	3.3-V PCI
TERM[27]	E25	output	3.3-V PCI
IO[29]	E26	bidir	3.3-V PCI
DIR[31]	E27	output	3.3-V PCI
TERM[32]	E28	output	3.3-V PCI
IO[34]	E29	bidir	3.3-V PCI

DOUT[1]	E34	output	3.3-V PCI
DOUT[5]	E35	output	3.3-V PCI
IOP[4]	F1	bidir	3.3-V PCI
IOP[5]	F2	bidir	3.3-V PCI
IOP[6]	F4	bidir	3.3-V PCI
IOP[7]	F5	bidir	3.3-V PCI
IOP[8]	F6	bidir	3.3-V PCI
DOUT[3]	F32	output	3.3-V PCI
DOUT[2]	F33	output	3.3-V PCI
DOUT[6]	F34	output	3.3-V PCI
IOP[9]	G1	bidir	3.3-V PCI
IOP[10]	G2	bidir	3.3-V PCI
IOP[11]	G3	bidir	3.3-V PCI
IOP[12]	G4	bidir	3.3-V PCI
IOP[13]	G5	bidir	3.3-V PCI
IOP[14]	G6	bidir	3.3-V PCI
DOUT[4]	G31	output	3.3-V PCI
DOUT[8]	G32	output	3.3-V PCI
DOUT[7]	G33	output	3.3-V PCI
DOUT[10]	G34	output	3.3-V PCI
DIN[2]	G35	input	3.3-V PCI
IOP[15]	H2	bidir	3.3-V PCI
IOP[16]	H3	bidir	3.3-V PCI
IOP[17]	H4	bidir	3.3-V PCI
IOP[18]	H6	bidir	3.3-V PCI
DOUT[9]	H31	output	3.3-V PCI
DIN[0]	H32	input	3.3-V PCI
DOUT[11]	H33	output	3.3-V PCI
DIN[3]	H34	input	3.3-V PCI
DIN[6]	H35	input	3.3-V PCI
IOP[19]	J1	bidir	3.3-V PCI
IOP[20]	J2	bidir	3.3-V PCI
IOP[21]	J3	bidir	3.3-V PCI
IOP[22]	J4	bidir	3.3-V PCI
IOP[23]	J5	bidir	3.3-V PCI
IOP[24]	J6	bidir	3.3-V PCI
DIN[1]	J31	input	3.3-V PCI
DIN[5]	J32	input	3.3-V PCI
DIN[4]	J33	input	3.3-V PCI
DIN[7]	J34	input	3.3-V PCI
DIN[11]	J35	input	3.3-V PCI
IOP[25]	K1	bidir	3.3-V PCI
IOP[26]	K2	bidir	3.3-V PCI
IOP[27]	K3	bidir	3.3-V PCI

IOP[28]	K4	bidir	3.3-V	PCI
IOP[29]	K5	bidir	3.3-V	PCI
IOP[30]	K6	bidir	3.3-V	PCI
DIN[9]	K32	input	3.3-V	PCI
DIN[8]	K33	input	3.3-V	PCI
FRXRSTN[7]	K34	output	3.3-V	PCI
LED[2]	K35	output	3.3-V	PCI
IOP[31]	L1	bidir	3.3-V	PCI
ADD[11]	L2	input	3.3-V	PCI
ADD[10]	L4	input	3.3-V	PCI
ADD[9]	L5	input	3.3-V	PCI
ADD[8]	L6	input	3.3-V	PCI
DIN[10]	L31	input	3.3-V	PCI
FRXRSTN[6]	L32	output	3.3-V	PCI
FRXRSTN[5]	L33	output	3.3-V	PCI
TEST_IN1	L34	input	3.3-V	PCI
LED[1]	L35	output	3.3-V	PCI
ADD[7]	M2	input	3.3-V	PCI
ADD[6]	M3	input	3.3-V	PCI
ADD[5]	M4	input	3.3-V	PCI
ADD[4]	M5	input	3.3-V	PCI
ADD[3]	M6	input	3.3-V	PCI
FRXRSTN[4]	M31	output	3.3-V	PCI
FRXRSTN[3]	M32	output	3.3-V	PCI
FRXRSTN[2]	M33	output	3.3-V	PCI
TEST_IN0	M34	input	3.3-V	PCI
LED[0]	M35	output	3.3-V	PCI
ADD[2]	N2	input	3.3-V	PCI
ADD[1]	N3	input	3.3-V	PCI
ADD[0]	N5	input	3.3-V	PCI
LINTN	N6	bidir	3.3-V	PCI
PLL7_SCLK	N30	bidir	3.3-V	PCI
PLL7_S2	N31	output	3.3-V	PCI
PLL7_A	N32	input	3.3-V	PCI
PLL7_B	N33	input	3.3-V	PCI
PLL7_C	N34	input	3.3-V	PCI
BLASTN	P1	input	3.3-V	PCI
READYN	P3	output	3.3-V	PCI
RSTN	P4	input	3.3-V	PCI
W_R	P5	input	3.3-V	PCI
ADSN	P6	input	3.3-V	PCI
PLL6_SCLK	P30	bidir	3.3-V	PCI
PLL6_S2	P31	output	3.3-V	PCI
PLL6_A	P32	input	3.3-V	PCI

PLL6_B	P33	input	3.3-V	PCI
PLL6_C	P34	input	3.3-V	PCI
LED[3]	P35	output	3.3-V	PCI
XILINX_INT	R1	bidir	3.3-V	PCI
FTX0[0]	R6	input	3.3-V	PCI
PLL5_SCLK	R30	bidir	3.3-V	PCI
PLL5_S2	R31	output	3.3-V	PCI
PLL5_A	R32	input	3.3-V	PCI
PLL5_B	R33	input	3.3-V	PCI
PLL5_C	R35	input	3.3-V	PCI
FTX0[1]	T1	input	3.3-V	PCI
OSC	T2	input	3.3-V	PCI
FTX0[2]	T3	input	3.3-V	PCI
FTX0[3]	T5	input	3.3-V	PCI
PLL4_SCLK	T30	bidir	3.3-V	PCI
PLL4_S2	T31	output	3.3-V	PCI
PLL4_A	T32	input	3.3-V	PCI
PLL4_B	T35	input	3.3-V	PCI
PCI_CLK	U2	input	3.3-V	PCI
FTX0[4]	W5	input	3.3-V	PCI
PLL0_A	W34	input	3.3-V	PCI
FTX0[5]	Y4	input	3.3-V	PCI
FTX0[6]	Y6	input	3.3-V	PCI
PLL1_A	Y34	input	3.3-V	PCI
PLL4_C	Y35	input	3.3-V	PCI
FTX0[7]	AA1	input	3.3-V	PCI
FTXCLK[0]	AA2	output	3.3-V	PCI
FTXREN[0]	AA3	output	3.3-V	PCI
FTXMTN[0]	AA5	input	3.3-V	PCI
FTXPAEN[0]	AA6	input	3.3-V	PCI
PLL3_SCLK	AA30	bidir	3.3-V	PCI
PLL3_S2	AA32	output	3.3-V	PCI
PLL3_A	AA33	input	3.3-V	PCI
PLL3_B	AA34	input	3.3-V	PCI
FTXFFN[0]	AB1	input	3.3-V	PCI
FTX1[0]	AB2	input	3.3-V	PCI
FTX1[1]	AB3	input	3.3-V	PCI
FTX1[2]	AB4	input	3.3-V	PCI
FTX1[3]	AB5	input	3.3-V	PCI
FTX1[4]	AB6	input	3.3-V	PCI
PLL2_SCLK	AB30	bidir	3.3-V	PCI
PLL2_S2	AB31	output	3.3-V	PCI
PLL2_A	AB32	input	3.3-V	PCI
PLL2_B	AB33	input	3.3-V	PCI

PLL2_C	AB34	input	3.3-V PCI
PLL3_C	AB35	input	3.3-V PCI
FTX1[5]	AC1	input	3.3-V PCI
FTX1[6]	AC2	input	3.3-V PCI
FTX1[7]	AC4	input	3.3-V PCI
FTXCLK[1]	AC5	output	3.3-V PCI
FTXREN[1]	AC6	output	3.3-V PCI
PLL1_SCLK	AC30	bidir	3.3-V PCI
PLL1_S2	AC31	output	3.3-V PCI
PLL1_B	AC34	input	3.3-V PCI
PLL1_C	AC35	input	3.3-V PCI
FTXMTN[1]	AD1	input	3.3-V PCI
FTXPAEN[1]	AD2	input	3.3-V PCI
FTXFFN[1]	AD3	input	3.3-V PCI
FTX2[0]	AD4	input	3.3-V PCI
FTX2[1]	AD5	input	3.3-V PCI
FTX2[2]	AD6	input	3.3-V PCI
PLL0_SCLK	AD30	bidir	3.3-V PCI
PLL0_S2	AD31	output	3.3-V PCI
PLL0_B	AD33	input	3.3-V PCI
PLL0_C	AD34	input	3.3-V PCI
FTX2[3]	AE1	input	3.3-V PCI
FTX2[4]	AE3	input	3.3-V PCI
FTX2[5]	AE4	input	3.3-V PCI
FTX2[6]	AE5	input	3.3-V PCI
FTX2[7]	AE6	input	3.3-V PCI
PLL_REF	AE30	output	3.3-V PCI
FRXLDN[6]	AE31	output	3.3-V PCI
FRXLDN[7]	AE32	output	3.3-V PCI
FRXRSTN[1]	AE34	output	3.3-V PCI
PLL_DATA	AE35	bidir	3.3-V PCI
FTXCLK[2]	AF1	output	3.3-V PCI
FTXREN[2]	AF2	output	3.3-V PCI
FTXMTN[2]	AF3	input	3.3-V PCI
FTXPAEN[2]	AF4	input	3.3-V PCI
FTXFFN[2]	AF5	input	3.3-V PCI
FTX3[0]	AF6	input	3.3-V PCI
FRXFFN[7]	AF30	input	3.3-V PCI
FRXLDN[1]	AF31	output	3.3-V PCI
FRXLDN[2]	AF32	output	3.3-V PCI
FRXLDN[3]	AF33	output	3.3-V PCI
FRXLDN[4]	AF34	output	3.3-V PCI
FRXLDN[5]	AF35	output	3.3-V PCI
FTX3[1]	AG2	input	3.3-V PCI

FTX3[2]	AG3	input	3.3-V	PCI
FTX3[3]	AG4	input	3.3-V	PCI
FTX3[4]	AG5	input	3.3-V	PCI
FTX3[5]	AG6	input	3.3-V	PCI
FRX7[6]	AG30	output	3.3-V	PCI
FRX7[7]	AG31	output	3.3-V	PCI
FRXCLK[7]	AG32	output	3.3-V	PCI
FRXWEN[7]	AG33	output	3.3-V	PCI
FRXMTN[7]	AG34	input	3.3-V	PCI
FRXPAFN[7]	AG35	input	3.3-V	PCI
FTX3[6]	AH1	input	3.3-V	PCI
FTX3[7]	AH2	input	3.3-V	PCI
FTXCLK[3]	AH3	output	3.3-V	PCI
FTXREN[3]	AH5	output	3.3-V	PCI
FTXMTN[3]	AH6	input	3.3-V	PCI
FRX7[2]	AH30	output	3.3-V	PCI
FRX7[3]	AH32	output	3.3-V	PCI
FRX7[4]	AH33	output	3.3-V	PCI
FRX7[5]	AH34	output	3.3-V	PCI
FTXPAEN[3]	AJ1	input	3.3-V	PCI
FTXFFN[3]	AJ2	input	3.3-V	PCI
FTX4[0]	AJ3	input	3.3-V	PCI
FTX4[1]	AJ4	input	3.3-V	PCI
FTX4[2]	AJ5	input	3.3-V	PCI
FTX4[3]	AJ6	input	3.3-V	PCI
FRXWEN[6]	AJ30	output	3.3-V	PCI
FRXMTN[6]	AJ31	input	3.3-V	PCI
FRXPAFN[6]	AJ32	input	3.3-V	PCI
FRXFFN[6]	AJ33	input	3.3-V	PCI
FRX7[0]	AJ34	output	3.3-V	PCI
FRX7[1]	AJ35	output	3.3-V	PCI
FTX4[4]	AK1	input	3.3-V	PCI
FTX4[5]	AK2	input	3.3-V	PCI
FTX4[6]	AK3	input	3.3-V	PCI
FTX4[7]	AK4	input	3.3-V	PCI
FTXCLK[4]	AK5	output	3.3-V	PCI
FTXREN[4]	AK6	output	3.3-V	PCI
FRX6[4]	AK30	output	3.3-V	PCI
FRX6[5]	AK31	output	3.3-V	PCI
FRX6[6]	AK32	output	3.3-V	PCI
FRX6[7]	AK34	output	3.3-V	PCI
FRXCLK[6]	AK35	output	3.3-V	PCI
FTXMTN[4]	AL1	input	3.3-V	PCI
FTXPAEN[5]	AL7	input	3.3-V	PCI

FTX6[3]	AL8	input	3.3-V	PCI
FTXCLK[6]	AL9	output	3.3-V	PCI
FTX7[0]	AL10	input	3.3-V	PCI
FTX7[5]	AL11	input	3.3-V	PCI
FRXRSTN[0]	AL13	output	3.3-V	PCI
FRX0[3]	AL14	output	3.3-V	PCI
FRXCLK[0]	AL15	output	3.3-V	PCI
FRX1[0]	AL16	output	3.3-V	PCI
FRX1[5]	AL20	output	3.3-V	PCI
FRXMTN[1]	AL21	input	3.3-V	PCI
FRX2[2]	AL22	output	3.3-V	PCI
FRX2[7]	AL23	output	3.3-V	PCI
FRX3[2]	AL25	output	3.3-V	PCI
FRX3[7]	AL26	output	3.3-V	PCI
FRXFFN[3]	AL27	input	3.3-V	PCI
FRX4[4]	AL28	output	3.3-V	PCI
FRXWEN[4]	AL29	output	3.3-V	PCI
FRX6[1]	AL33	output	3.3-V	PCI
FRX6[2]	AL34	output	3.3-V	PCI
FRX6[3]	AL35	output	3.3-V	PCI
FTXPAEN[4]	AM1	input	3.3-V	PCI
FTX5[7]	AM6	input	3.3-V	PCI
FTXFFN[5]	AM7	input	3.3-V	PCI
FTX6[4]	AM8	input	3.3-V	PCI
FTXREN[6]	AM9	output	3.3-V	PCI
FTX7[1]	AM10	input	3.3-V	PCI
FTX7[6]	AM11	input	3.3-V	PCI
FRXLDN[0]	AM13	output	3.3-V	PCI
FRX0[4]	AM14	output	3.3-V	PCI
FRXWEN[0]	AM15	output	3.3-V	PCI
FRX1[1]	AM16	output	3.3-V	PCI
FRX1[6]	AM20	output	3.3-V	PCI
FRXPAFN[1]	AM21	input	3.3-V	PCI
FRX2[3]	AM22	output	3.3-V	PCI
FRXCLK[2]	AM23	output	3.3-V	PCI
FRX3[3]	AM25	output	3.3-V	PCI
FRXCLK[3]	AM26	output	3.3-V	PCI
FRX4[0]	AM27	output	3.3-V	PCI
FRX4[5]	AM28	output	3.3-V	PCI
FRXMTN[4]	AM29	input	3.3-V	PCI
FRX5[1]	AM30	output	3.3-V	PCI
FRX6[0]	AM35	output	3.3-V	PCI
FTX5[4]	AN5	input	3.3-V	PCI
FTXCLK[5]	AN6	output	3.3-V	PCI

FTX6[0]	AN7	input	3.3-V	PCI
FTX6[5]	AN8	input	3.3-V	PCI
FTXMTN[6]	AN9	input	3.3-V	PCI
FTX7[2]	AN10	input	3.3-V	PCI
FTX7[7]	AN11	input	3.3-V	PCI
FTXMTN[7]	AN12	input	3.3-V	PCI
FRX0[0]	AN13	output	3.3-V	PCI
FRX0[5]	AN14	output	3.3-V	PCI
FRXMTN[0]	AN15	input	3.3-V	PCI
FRX1[2]	AN16	output	3.3-V	PCI
FRX1[7]	AN20	output	3.3-V	PCI
FRXFFN[1]	AN21	input	3.3-V	PCI
FRX2[4]	AN22	output	3.3-V	PCI
FRXWEN[2]	AN23	output	3.3-V	PCI
FRXFFN[2]	AN24	input	3.3-V	PCI
FRX3[4]	AN25	output	3.3-V	PCI
FRXWEN[3]	AN26	output	3.3-V	PCI
FRX4[1]	AN27	output	3.3-V	PCI
FRX4[6]	AN28	output	3.3-V	PCI
FRXPAFN[4]	AN29	input	3.3-V	PCI
FRX5[2]	AN30	output	3.3-V	PCI
FRX5[5]	AN31	output	3.3-V	PCI
FTX5[0]	AP3	input	3.3-V	PCI
FTX5[2]	AP4	input	3.3-V	PCI
FTX5[5]	AP5	input	3.3-V	PCI
FTXREN[5]	AP6	output	3.3-V	PCI
FTX6[1]	AP7	input	3.3-V	PCI
FTX6[6]	AP8	input	3.3-V	PCI
FTXPAEN[6]	AP9	input	3.3-V	PCI
FTX7[3]	AP10	input	3.3-V	PCI
FTXCLK[7]	AP11	output	3.3-V	PCI
FTXPAEN[7]	AP12	input	3.3-V	PCI
FRX0[1]	AP13	output	3.3-V	PCI
FRX0[6]	AP14	output	3.3-V	PCI
FRXPAFN[0]	AP15	input	3.3-V	PCI
FRX1[3]	AP16	output	3.3-V	PCI
FRXCLK[1]	AP20	output	3.3-V	PCI
FRX2[0]	AP21	output	3.3-V	PCI
FRX2[5]	AP22	output	3.3-V	PCI
FRXMTN[2]	AP23	input	3.3-V	PCI
FRX3[0]	AP24	output	3.3-V	PCI
FRX3[5]	AP25	output	3.3-V	PCI
FRXMTN[3]	AP26	input	3.3-V	PCI
FRX4[2]	AP27	output	3.3-V	PCI

FRX4[7]	AP28	output	3.3-V	PCI
FRXFFN[4]	AP29	input	3.3-V	PCI
FRX5[3]	AP30	output	3.3-V	PCI
FRX5[6]	AP31	output	3.3-V	PCI
FRXCLK[5]	AP32	output	3.3-V	PCI
FRXMTN[5]	AP33	input	3.3-V	PCI
FTXFFN[4]	AR2	input	3.3-V	PCI
FTX5[1]	AR3	input	3.3-V	PCI
FTX5[3]	AR4	input	3.3-V	PCI
FTX5[6]	AR5	input	3.3-V	PCI
FTXMTN[5]	AR6	input	3.3-V	PCI
FTX6[2]	AR7	input	3.3-V	PCI
FTX6[7]	AR8	input	3.3-V	PCI
FTXFFN[6]	AR9	input	3.3-V	PCI
FTX7[4]	AR10	input	3.3-V	PCI
FTXREN[7]	AR11	output	3.3-V	PCI
FTXFFN[7]	AR12	input	3.3-V	PCI
FRX0[2]	AR13	output	3.3-V	PCI
FRX0[7]	AR14	output	3.3-V	PCI
FRXFFN[0]	AR15	input	3.3-V	PCI
FRX1[4]	AR16	output	3.3-V	PCI
FRXWEN[1]	AR20	output	3.3-V	PCI
FRX2[1]	AR21	output	3.3-V	PCI
FRX2[6]	AR22	output	3.3-V	PCI
FRXPAFN[2]	AR23	input	3.3-V	PCI
FRX3[1]	AR24	output	3.3-V	PCI
FRX3[6]	AR25	output	3.3-V	PCI
FRXPAFN[3]	AR26	input	3.3-V	PCI
FRX4[3]	AR27	output	3.3-V	PCI
FRXCLK[4]	AR28	output	3.3-V	PCI
FRX5[0]	AR29	output	3.3-V	PCI
FRX5[4]	AR30	output	3.3-V	PCI
FRX5[7]	AR31	output	3.3-V	PCI
FRXWEN[5]	AR32	output	3.3-V	PCI
FRXPAFN[5]	AR33	input	3.3-V	PCI
FRXFFN[5]	AR34	input	3.3-V	PCI
DATAEN[0]	L30	output	3.3-V	PCI
DATAEN[1]	Y31	output	3.3-V	PCI
DATAEN[2]	R2	output	3.3-V	PCI
DATAEN[3]	N1	output	3.3-V	PCI
DATAEN[4]	M1	output	3.3-V	PCI
DATAEN[5]	T6	output	3.3-V	PCI
DATAEN[6]	P2	output	3.3-V	PCI
DATAEN[7]	K30	output	3.3-V	PCI

DATAEN[8]	AD35	output	3.3-V PCI
DATAEN[9]	AC33	output	3.3-V PCI
DATAEN[10]	H30	output	3.3-V PCI
DATAEN[11]	J30	output	3.3-V PCI

The pin names match with the schematic names and the names found throughout this manual. The engineering kit contains a reference project with the pin numbers defined and the bus interfaces implemented. A lot of time will be saved on the first implementation starting with the reference design. The pinlist and following definitions are for those who want to "do it themselves".

Numbers in [] are member numbers – bit position or vector number.
Numbers not in [] are channel numbers in most cases.
"N" as a suffix indicates active low
The direction and voltage level are defined for each term.
There are 8 PLLs each with 3 (a,b,c) outputs.
Test_IN0,1 are connected to testpoints for user debugging.
FRX = RX FIFO
FTX = TX FIFO
PLL = Phase Locked Loop
WEN = Write Enable
REN = Read Enable
PAF = Programmable Almost Full
PAE = Programmable Almost Empty
FF = Full Flag
MT = Empty Flag
OSC is connected to the on-board 66.6667 MHz reference.
The Address, Data, ADS, BLAST, Ready, RST, W_R signals are the interface to the PLX device. Please refer to the 9054 data manual if you are not using the Engineering kit. Please note that the Ready signal must be open-drain because the Xilinx also drives this signal when it is accessed.

D100 Standard Pin Assignment

The pin assignment for the PCI-Altera-485/LVDS 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 +5	fused +5	25		75	
fused +5	fused +5	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 24 PCI-ALTERA-485/LVDS D100 PINOUT

D100 Alternate Pin Assignment

The Alternate pin assignment for the PCI-Altera-485/LVDS P1 connector.

CH0_TXDP	CH0_TXDM	1	51
CH0_TXCP	CH0_TXCM	2	52
CH0_RXDP	CH0_RXCM	3	53
CH0_RXCP	CH0_RXCM	4	54
CH0_TXCINP	CH0_TXCINM	5	55
CH0_MONITOR	GND*	6	56
CH1_TXDP	CH1_TXDM	7	57
CH1_TXCP	CH1_TXCM	8	58
CH1_RXDP	CH1_TXDM	9	59
CH1_RXCP	CH1_RXCM	10	60
CH1_TXCINP	CH1_TXCINM	11	61
CH1_MONITOR	GND*	12	62
CH2_TXDP	CH2_TXDM	15	63
CH2_TXCP	CH2_TXCM	14	64
CH2_RXDP	CH2_RXDM	15	65
CH2_RXCP	CH2_RXCM	16	66
CH2_TXCINP	CH2_TXCINM	17	67
CH2_MONITOR	TTL_IO_0	18	68
CH3_TXDP	CH3_TXDM	19	69
CH3_TXCP	CH3_TXCM	20	70
CH3_RXDP	CH3_RXDM	21	71
CH3_RXCP	CH3_RXCM	22	72
CH3_TXCINP	CH3_TXCINM	23	73
CH3_MONITOR	TTL_IO_1	24	74
fused +5 fused +5	25	75	
fused +	fused +5	26	76
CH4_TXDP	CH4_TXDM	27	77
CH4_TXCP	CH4_TXCM	28	78
CH4_RXDP	CH4_RXDM	29	79
CH4_RXCP	CH4_RXCM	30	80
CH4_TXCINP	CH4_TXCINM	31	81
CH4_MONITOR	TTL_IO_2	32	82
CH5_TXDP	CH5_TXDM	33	83
CH5_TXCP	CH5_TXCM	34	84
CH5_RXDP	CH5_RXDM	35	85
CH5_RXCP	CH5_RXCM	36	86
CH5_TXCINP	CH5_TXCINM	37	87
CH5_MONITOR	GND*	38	88
CH6_TXDP	CH6_TXDM	39	89
CH6_TXCP	CH6_TXCM	40	90
CH6_RXDP	CH6_RXDM	41	91
CH6_RXCP	CH6_RXCM	42	92
CH6_TXCINP	CH6_TXCINM	43	93
CH6_MONITOR	GND*	44	94
CH7_TXDP	CH7_TXDM	45	95
CH7_TXCP	CH7_TXCM	46	96
CH7_RXDP	CH7_RXDM	47	97
CH7_RXCP	CH7_RXCM	48	98
CH7_TXCINP	CH7_TXCINM	49	99
CH7_MONITOR	MONITOR_CLEAR	50	100

FIGURE 25 PCI-ALTERA-485/LVDS ALTERNATE D100 PINOUT

Note 1: GND* is connected to ground via an 0805 device. The device [R58] can be left open for isolated, filled with a 0Ω resistor for DC and filled with a .1uF cap for AC reference options. The default is for 0Ω installed. Please contact the factory if you would prefer one of the other options.

Note 2: fused +5 is connected to P5V via a 2.5A “self healing” fuse. If you prefer not to have power on the P1 connector please alert the factory when placing your order – we can leave the fuse off.

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 the PCI-Altera-485/LVDS. 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. We use PCIView.

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

PCI Modules while commercial in nature can be conceived and engineered for rugged industrial environments. The PCI-ALTERA-485/LVDS is constructed out of 0.062 inch thick FR4 material.

Through hole and surface mounting of components are used. High insertion and removal forces are required, which assists in the retention of components. If the application requires unusually high reliability or is in an environment subject to high vibration, the user may solder the corner pins of each socketed IC into the socket, using a grounded soldering iron.

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

The PCI-ALTERA-485/LVDS design consists of 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.

http://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. The current minimum repair charge is $100. 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 3
Santa Cruz, CA 95060
(831) 457-8891 - Fax (831) 457-4793

support@dyneng.com

Specifications

PCI Interfaces:	PCI Interface 33 MHz 32-bit
Access types:	Configuration and Memory space utilized
CLK rates supported:	33 MHz PCI, 66 MHz Tx FIFO write and Rx FIFO read, local 66.6667 MHz oscillator on Altera, 8 PLLs to provide programmable frequencies.
Memory	FIFO memory is provided to support DMA 1K x 32. In addition 16 8 x 4K FIFOs are provided to support Xilinx <-> Altera data flow with 8 TX and 8 RX
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 Xilinx. User defined within Altera.
Initialization:	Programming procedure documented in this manual
Access Modes:	Most 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 to the PCI bus is supported with multiple sources. The interrupts are maskable and are supported with a status register.
Onboard Options:	All Options are Software Programmable
Dimensions:	half-length PCI board.
Construction:	FR4 Multi-Layer Printed Circuit, Through Hole and Surface Mount Components. Programmable parts are socketed.
Power:	5V from PCI bus. Local 3.3 and 2.5 and 1.8 created with on-board power supplies.
User	8 position software readable switch 5 software controllable LED's 1 Power LED

Order Information

Standard temperature range 0-70ºC	
PCI-ALTERA-485/LVDS	http://www.dyneng.com/pci_altera_485.html half length PCI card with user re-configurable 20K400E, 40 RS-485 or LVDS IO, 12 TTL IO, 8 PLL
Extended temperature range -20 - 85ºC	
PCI-ALTERA-485/LVDS-ET	http://www.dyneng.com/pci_altera_485.html half length PCI card with user re-configurable 20K400E, 40 RS-485 or LVDS IO IO, 12 TTL IO, 8 PLL
PCI-ALTERA-485/LVDS-ENG	Engineering Kit for the PCI-ALTERA-485/LVDS Software, Schematic, Cable and HDEterm100, reference Altera implementation. Please refer to the webpage for more information
-PROM	Install the FLASH PROM and associated circuitry to allow "instant on" operation.
HDEterm100	http://www.dyneng.com/HDEterm100.html 100-pin connectors (2) matching the PCI-Altera-485/LVDS D100 interconnected with 100 screw terminals. DIN rail mounting. Optional terminations and testpoints.
HDEcable100	http://www.dyneng.com/HDEcabl100.html 100 pin connector matching PCI-Altera-485/LVDS and HDEterm100. Length options