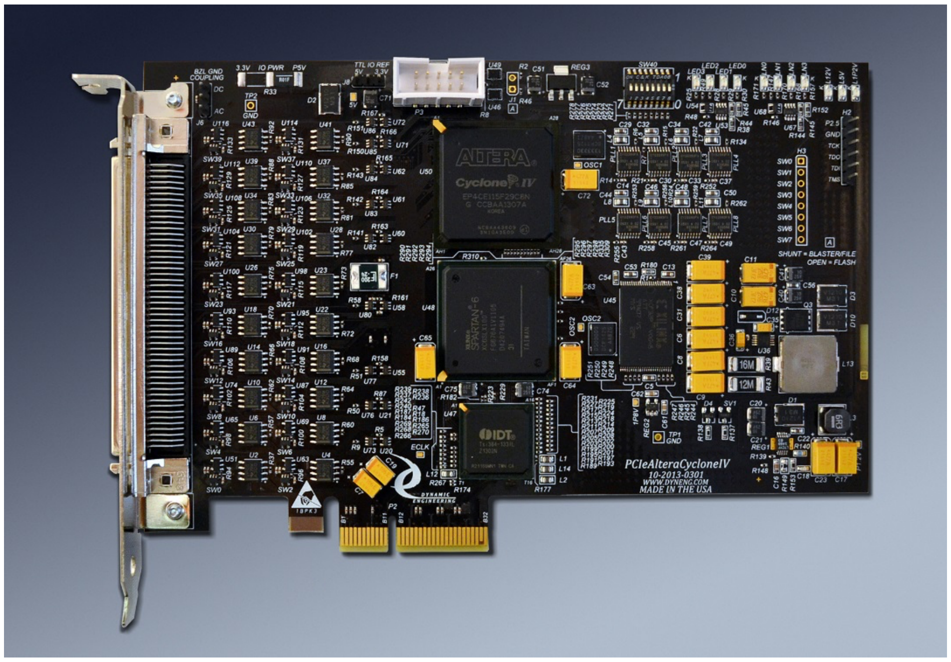
DYNAMIC ENGINEERING
150 DuBois, Suite B&C Santa Cruz, CA 95060
(831) 457-8891 www.dyneng.com
sales@dyneng.com
Est. 1988

User Manual

PCIe-AlteraCycloneIV

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



Revision 1p6
Corresponding Hardware:
Fab Number: 10-2013-0304 FLASH Rev 2.18

PCIe-AlteraCycloneIV
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 with PMC Module carriers and compatible user-provided equipment. Connection of incompatible hardware is likely to cause serious damage.

Table of Contents

List of Figures

Product Description

PCIe-AlteraCycloneIV is part of the PCIe Compatible family of modular I/O components. PCIe-AlteraCycloneIV provides a user configurable Cyclone IV FPGA [EP4CE115F29(i,c)8], along with 40 RS-485 or 40 LVDS transceivers, 8 PLLs [24 clocks] and FIFO support, full DMA capabilities in a half-length single slot card.

With Revision 4 of the PCB a new bridge and FLASH for the Xilinx FPGA are introduced. Both changes are due to EOL parts. The Altera pinout did not change. The control clock is now 50 MHz rather than 33.3 MHZ. Most designs will not "notice" the changes. If the User design for the Altera uses the Control Clock to create other frequencies it may be necessary to update the parameters to list the new control clock frequency. For example, our reference design created 20 MHz from the Control Clock and needed to be updated.

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 PCIe bus implementation is 4 lanes using a PI7C9X130 bridge. The bridge is connected through a Xilinx Spartan 6 to the Altera FPGA. The Spartan 6 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.

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 Spartan 6 has a multi-channel DMA engine that interacts with the TSI384 to provide separate support for each data lane. 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. When Master and or Target is "almost" [full or empty] the data transfer engine slows down to get exact status. Due to pipelining, there are delays from write to current status being available to the control engine.

PCIe-AlteraCycloneIV has drivers for Windows, and Linux. The packages come with a sample user application to demonstrate the use of the various HW capabilities. The VHDL for the Altera reference design is also available. The reference design can be modified to perform your unique function. The reference design has 8 channels plus a base function. Each channel is tied to 5 differential IO. 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 and strobe.

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 Altera.

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

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-AlteraCycloneIV-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. The switch is attached to the Xilinx FPGA to allow use by the driver.

LEDs are provided on board for user-defined purposes. 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. 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 the regulators are operating properly.

The UserAp reference software can load a User Altera design file [.rbf] into the Altera 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 targeted for FLASH should be configured for AS mode. User Designs targeted for direct loading should be configured for PS. The hardware takes care of controlling nConfig and the MSEL bits to effect the change in operation as part of the load control module within the PCIe-AlteraController.

The user can program the FLASH device with the provided header. The Altera will be loaded on power transitions automatically from the FLASH. Software can overwrite.

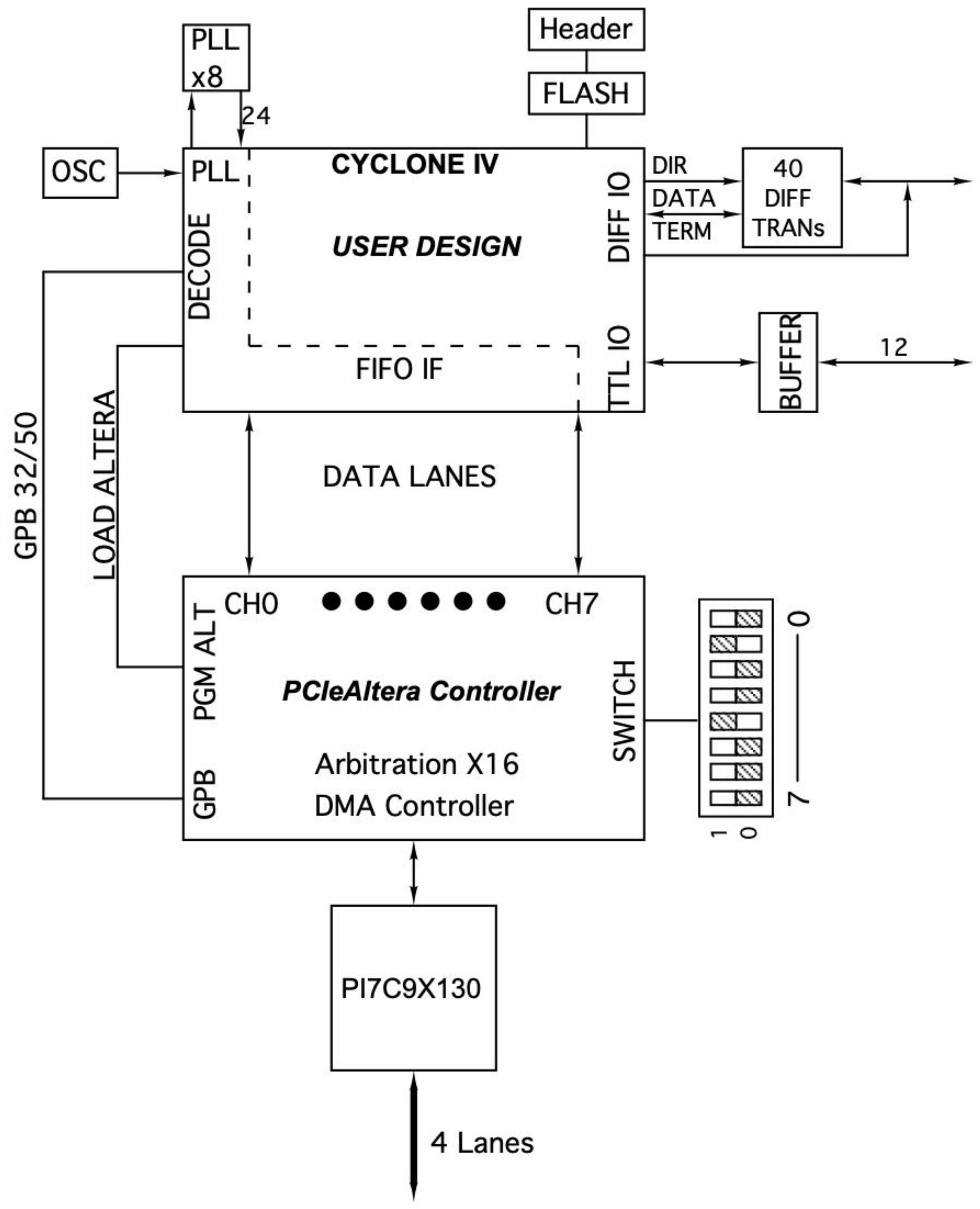


Figure 1 PCIe-AlteraCycloneIV Block diagram

PCIe-AlteraCycloneIV-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, 166 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 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.   The receive side is also buffered to protect the Altera device. The pull-ups are referenced to a shunt selectable 3.3V or 5V.   The shunt is located near the Altera FLASH header.

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 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.

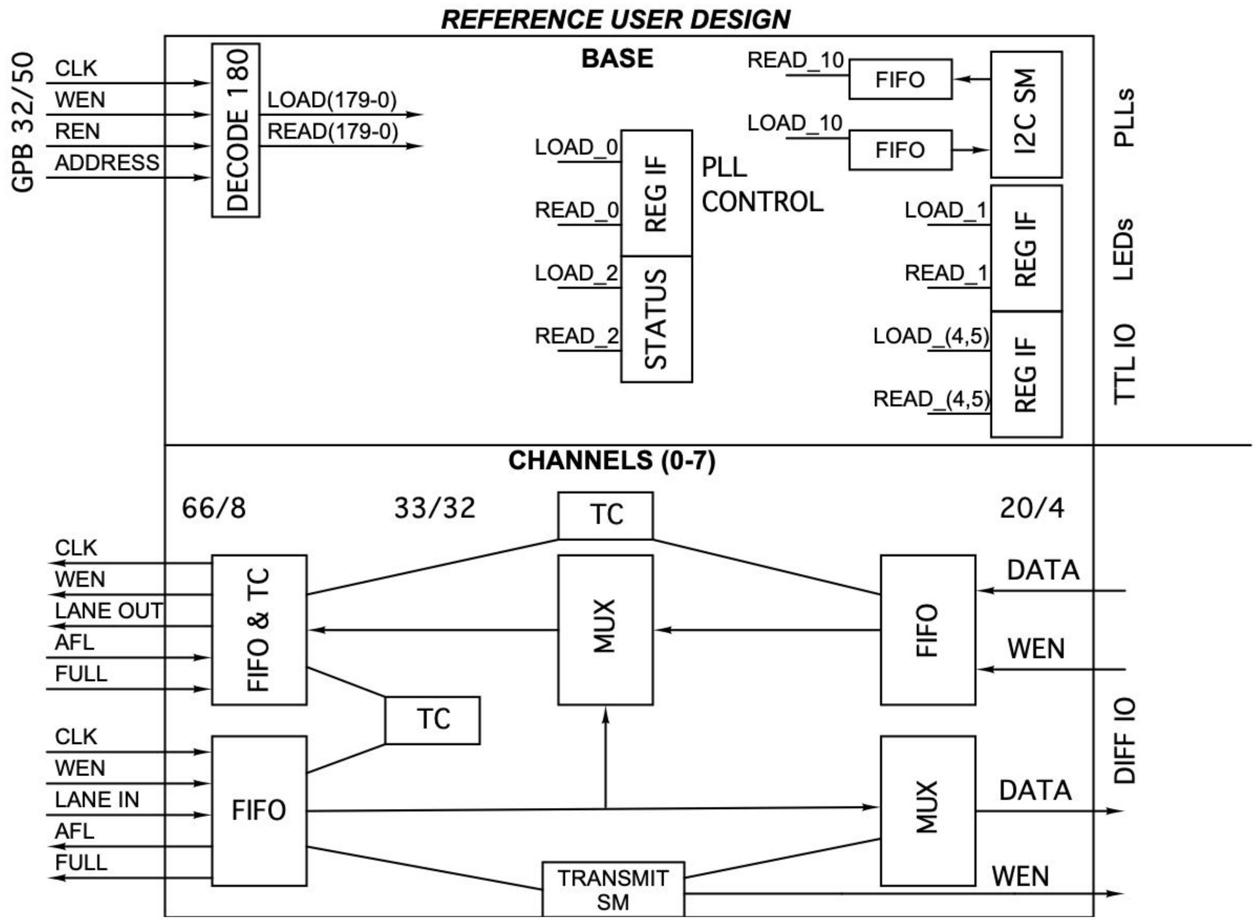


Figure 2 PCIe-AlteraCycloneIV Reference design Block diagram

Theory of Operation

A wide range of interfaces and protocols can be implemented with PCIe-AlteraCycloneIV. 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 PCIe-AlteraCycloneIV along with the Quartus software.

The Engineering Kit comes with the VHDL for the reference design. The reference design was done with the webpack version of Quartus II version 17.1. 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-AlteraCycloneIV 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 [Altera] 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 Altera PLL] 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 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 Altera design software. The output file can be “uploaded” to the Altera FPGA on PCIe-AlteraCycloneIV. 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-AlteraCycloneIV has 40 differential IO. PCIe-AlteraCycloneIV 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 is Host memory transferred into the Xilinx TX FIFO via DMA transfers. The data is transferred from the Xilinx holding FIFO to the Altera Holding FIFO with the transfer control state-machine within the Xilinx. Each channel is separately supported. The Altera side reports Almost Full and Full to allow the transfer control to burst when the Xilinx FIFO is not Almost Empty and the Altera FIFO is not almost full. 66 MHz x8 per channel per direction. The user state machine will read data from the FIFO on the Altera side and apply the user protocol before transmitting. On the receive side the data will flow into the Altera FPGA, be processed to convert to a format suitable for storing, and written into the associated channel’s RX FIFO. The data will be pushed from the Altera to the Xilinx channel Rx FIFO using similar controls to the TX path. Almost Full and Full flags coming from the Xilinx 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 all of the way through. Arbitration within the Xilinx to handle the 16 DMA engines automatically. Scatter Gather DMA is supported for large data transfers.

Large segments are allowed for OS with larger segment sizes [Linux compared to Windows®]. One interrupt per DMA[16] at the end.

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.

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 TTL interface in the reference design has two registers. One for controlling the direction [Tx/Rx] of each bit and one for data transfer. To operate in an open drain mode set the bit to RX to create a '1' and enable with the data set to '0' to get a low. This way the pull-up is creating the '1' and can be overwritten.

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 a tri-state function 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.

Feature List Current

- User Defined Altera Cyclone IV 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 Cyclone IV
- 8 position "DIP Switch"
- User LEDs
- Power LEDs
- FLASH and SW loading of Altera

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

VendorId = xDCBA
CardId = x0046.

Programming

PCIe-AlteraCycloneIV 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 PCIe-AlteraCycloneIV; the software kit includes our test suite. In addition, Linux drivers and reference suite 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 Xilinx “fixed” features and Altera Cyclone IV reference design features.   The Altera FPGA memory map is subject to change as you implement your design.

Address Map

Xilinx Base Address Map

PcieAlteraCycloneIV_BASE_BASE	0x0000 // 0 Base control register
PcieAlteraCycloneIV_BASE_USER_SWITCH	0x0004 // 1 User switch read port DIP switch read
PcieAlteraCycloneIV_BASE_XILINX_REV	0x0004 // 1 Xilinx revision read port
PcieAlteraCycloneIV_BASE_XILINX_DSN	0x0004 // 1 Xilinx Design Number read port
PcieAlteraCycloneIV_BASE_STATUS	0x0008 // 2 status Register offset
PcieAlteraCycloneIV_BASE_ALT_FIFO	0x000C // 3 Write to load FIFO with programming data for Altera device
PcieAlteraCycloneIV_BASE_ALT_FIFO_CNT	0x0010 // 4 Read for FIFO status [count and flags]
PcieAlteraCycloneIV_CH0	0x0050 // 20 starting address for channel 0
PcieAlteraCycloneIV_CH1	0x00A0 // 40 starting address for channel 1
PcieAlteraCycloneIV_CH2	0x00F0 // 60 starting address for channel 2
PcieAlteraCycloneIV_CH3	0x0140 // 80 starting address for channel 3
PcieAlteraCycloneIV_CH4	0x0190 // 100 starting address for channel 4
PcieAlteraCycloneIV_CH5	0x01E0 // 120 starting address for channel 5
PcieAlteraCycloneIV_CH6	0x0230 // 140 starting address for channel 6
PcieAlteraCycloneIV_CH7	0x0280 // 160 starting address for channel 7

Figure 3 PCIe-AlteraCycloneIV Xilinx Base Address Map

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

Xilinx Channel Address Map

PcieAlteraCycloneIV_CHAN_CNTRL	0x00000000 // 0 General control register
PcieAlteraCycloneIV_CHAN_STATUS	0x00000004 // 1 Interrupt status port
PcieAlteraCycloneIV_CHAN_INT_CLEAR	0x00000004 // 1 Interrupt clear port
PcieAlteraCycloneIV_CHAN_WR_DMA_PNTR	0x00000008 // 2 Write DMA dpr physical PCI address
PcieAlteraCycloneIV_CHAN_TX_FIFO_COUNT	0x00000008 // 2 Tx FIFO count read port
PcieAlteraCycloneIV_CHAN_RD_DMA_PNTR	0x0000000C // 3 Read DMA dpr physical PCI address
PcieAlteraCycloneIV_CHAN_RX_FIFO_COUNT	0x0000000C // 3 Rx FIFO count read port
PcieAlteraCycloneIV_CHAN_FIFO	0x00000010 // 4 FIFO offset for single word access R/W
PcieAlteraCycloneIV_CHAN_TX_AMT	0x00000014 // 5 Tx almost empty count register - used for Urgent and pulsed interrupt
PcieAlteraCycloneIV_CHAN_RX_AFL	0x00000018 // 6 Rx almost full count register
PcieAlteraCycloneIV_CHAN_TX_AMT_LVL	0x00000028 // 10 Tx almost empty level register - used for level based FIFO interrupt
PcieAlteraCycloneIV_CHAN_RX_AFL_LVL	0x00000040 // 16 Rx almost full level register

Figure 4 PCIe-AlteraCycloneIV Xilinx Channel 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 the PCIe-AlteraCycloneIV card, the 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 the PCIe-AlteraCycloneIV. The local Xilinx registers need to be configured as well as the registers within the Altera Cyclone IV FPGA.

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

Altera Base Address Map

PcieAlteraCycloneIV_BASE_ALT_ACCESS	0x8000 // PcieAlteraCycloneIVBase Set Addr(15) to use 32K dedicated to Altera control bus
PcieAlteraCycloneIV_ALT_BASE	0x0000 // 0 Altera base address for PLL access
PcieAlteraCycloneIV_ALT_LED	0x0004 // 1 Altera address for LED access Bit 4 enables to SW control
PcieAlteraCycloneIV_ALT_STATUS	0x0008 // 2 Altera address for Status Register in Base
PcieAlteraCycloneIV_ALT_TTL_DAT	0x0010 // 4 Altera address for TTL Data output and read-back of IO level 11-0
PcieAlteraCycloneIV_ALT_TTL_EN	0x0014 // 5 Altera address for TTL Data Enables 11-0
PcieAlteraCycloneIV_ALT_CNT	0x001C // 7 PcieAlteraCycloneIVBase Write Master Count, Read selected count
PcieAlteraCycloneIV_ALT_PLL_FIFO	0x0028 // 10 Altera address for PLL FIFO
PcieAlteraCycloneIV_ALT_CH_0	0x0050 // 20 Altera address pointer offset for channel 0
PcieAlteraCycloneIV_ALT_CH_1	0x00A0 // 40 Altera address pointer offset for channel 1
PcieAlteraCycloneIV_ALT_CH_2	0x00F0 // 60 Altera address pointer offset for channel 2
PcieAlteraCycloneIV_ALT_CH_3	0x0140 // 80 Altera address pointer offset for channel 3
PcieAlteraCycloneIV_ALT_CH_4	0x0190 // 100 Altera address pointer offset for channel 4
PcieAlteraCycloneIV_ALT_CH_5	0x01E0 // 120 Altera address pointer offset for channel 5
PcieAlteraCycloneIV_ALT_CH_6	0x0230 // 140 Altera address pointer offset for channel 6
PcieAlteraCycloneIV_ALT_CH_7	0x0280 // 160 Altera address pointer offset for channel 7

Figure 5 PCIe-AlteraCycloneIV Altera Base Address Map

Altera Channel Address Map

PcieAlteraCycloneIV_ALT_CH_BASE	0x0000 // 0 Altera address for Base Register
PcieAlteraCycloneIV_ALT_CH_STATUS	0x0004 // 1 Altera address for Status
PcieAlteraCycloneIV_ALT_CH_SP	0x0008 // 2 Altera address for Spare Register
PcieAlteraCycloneIV_ALT_AX_FIFO_CNT	0x000C // 3 Altera address for AX FIFO Count
PcieAlteraCycloneIV_ALT_XA_FIFO_CNT	0x0010 // 4 Altera address for XA FIFO Count
PcieAlteraCycloneIV_ALT_FIFO_WR	0x0014 // 6 Altera address for Writing RX FIFO
PcieAlteraCycloneIV_ALT_FIFO_RD	0x0018 // 6 Altera address for Reading TX FIFO

Figure 6 PCIe-AlteraCycloneIV Altera Channel 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 Altera. User SW can use the above definitions or self-defined mapping to build a pointer to the intended offset. For example: PcieAlteraCycloneIV_ALT_CH_7 + PcieAlteraCycloneIV_ALT_CH_STATUS is the offset for the Status register in channel 7 of the Cyclone IV.

Register Definitions

Xilinx Base Register Definitions

PcieAlteraCycloneIV_BASE_BASE

[0x0000 Main Control Register Port read/write]

DATA BIT	BASE REGISTER DESCRIPTION
0	BASE_ALTERA_INT_MASK
1	BASE_ALTERA_LOAD_USER
2	BASE_ALTERA_LOAD_FLASH
3	BASE_ALTERA_LOAD_USERDN
14	BASE_ALTERA_PGM_RST
15	BASE_ALTERA_RST

Figure 7 PCIe-AlteraCycloneIV Xilinx Base Control Register

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

BASE_ALTERA_LOAD_USER : Pulsed by writing to PCIeAlteraCycloneIVBASE_BASE with this bit set to start Altera User Load Function. See also BASE ALT FIFO for more information. [no need to clear]

BASE_ALTERA_LOAD_FLASH : Pulsed by writing to PCIeAlteraCycloneIVBASE_BASE with this bit set to start Altera FLASH Load Function [no need to clear]

BASE_ALTERA_LOAD_USERDN: Set when complete User Programming File has been loaded into BASE ALT FIFO.

BASE_ALTERA_PGM_RST: Set to reset FIFO and State-machine for Altera User File loading.

BASE_ALTERA_RST: Set to reset Altera device low to enable

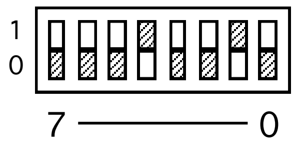
PcieAlteraCycloneIV_BASE_ID
[0x04 Switch and Design number port read only]

DATA BIT	DESCRIPTION
31-28	Revision Minor
27-24	Design Number
23-16	Revision Major
15-8	PCI Core Revision
7-0	DIP switch

Figure 8 PCIe-AlteraCycloneIV 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 in the silk screen. The DIP Switch can be read from this port and used to determine which PCIe-AlteraCycloneIV 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 is set to x1.
The current Major Revision is 0x02.
The current Minor Revision is x2.

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.

Please note: the definitions for the Upper 16 bits have changed between the PCB 03 and PCB 04 release.

PcieAlteraCycloneIV_BASE_STATUS

[0x0008 status register read only]

STATUS REGISTER

DATA BIT	DESCRIPTION
31-23	Spare
22	ConfDoneErrSLat
21	ConfDoneErrELat
20	PACIdleState
19-18	Spare
17	Altera nStatus
16-10	Spare
9	Altera Int Masked
8	Altera Int
7	Ch6 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-AlteraCycloneIV Status Port

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

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

Configuration Done Error Start Latched “ConfDoneErrSLat” is a sticky bit which is set if Conf_Done is not transitioned low in response to the start of a User Programming cycle. This bit is captured and held until cleared by writing to the status register with this bit set.

Configuration Done Error End Latched “ConfDoneErrSLat” is a sticky bit which is set if Conf_Done is not transitioned high in response to the end of a User Programming cycle. This bit is captured and held until cleared by writing to the status register with this bit set.

Program Altera Controller Idle State is set when the State-machine is in the Idle state waiting for a new User load or Load from FLASH command from the Base Control register. New commands will be ignored if the State-Machine is not idle. The state-machine can be reset to abort a current operation and return to IDLE.

AltnStatus is a secondary status bit from the Altera. While Conf_Done is used for the data transfer status, nStatus is pulsed low after nConfig is asserted. AltnStatus is of limited use for this application.

PcieAlteraCycloneIV_BASE_ALT_FIFO

PcieAlteraCycloneIV_BASE_ALT_FIFO_CNT

[0x000C 0x0010 program the Altera storage FIFO and status]

DATA BIT	DESCRIPTION
(0xC) 31-0	Program Altera Memory [PAM] FIFO load
(0x10) 31-18	spare
17	PAM FIFO FULL
16	PAM FIFO MT
15-10	spare
9-0	PAM Count

Figure 10 PCIe-AlteraCycloneIV Programming port

When the PAM FIFO offset is written to data is loaded into the 1Kx32 FIFO for storage. The data is read out by the PAM state-machine to serially load the Altera Cyclone IV FPGA.

The PAM count shows how much data is in the FIFO. The FPGA file is rather large; SW can poll the count, subtract from 1K and add new data to the FIFO while the transfer is taking place.

Our test loop uses the reset function to make sure the FIFO is empty and the State-Machine [PAM] is where you expect it. The software enables operation, and then loads 1K-1 into the FIFO. Once enabled the count is polled and more data added to the FIFO while the transfer is running. The Transfer happens at 50 MHz; with 32 bit data plus pad it takes about 1 uS per LW loaded into the FIFO.

The reference design has a different pattern on the User LEDs than the reference .rbf 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 "PCIeAlteraCycloneIV.rbf" or your equivalent including the path. If you store the .rbf with the UserAp executable PCIeAlteraCycloneIVUserAp.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.

When the User Load bit is set the control HW resets the Altera [nConfig], waits the required time and begins to look for data to load. When the FIFO has data the SM transfers to the Altera with the PS mode operation. This allows the clock to start and stop etc. When the user has loaded all data into the FIFO the user sets the USERDN bit to tell the state-machine to complete the transfer and to do the post transfer processing. The Altera an additional 300-650 uS to complete the data transfer once the Serial data transfer has completed.

Xilinx Channel Register Definitions

PcieAlteraCycloneIV_CH_CNTRL

[0x0000 read/write]

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

Channel Control Register

Figure 11 PCIe-AlteraCycloneIV Xilinx 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 Xilinx. 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 PCIeAlteraCycloneIV 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 Altera] and receive [from Altera] 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.

PcieAlteraCycloneIV_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-AlteraCycloneIV Xilinx 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 dipped 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.

PcieAlteraCycloneIV_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-AlteraCycloneIV Xilinx 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.

PcieAlteraCycloneIV_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-AlteraCycloneIV Xilinx 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.

PcieAlteraCycloneIV_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-AlteraCycloneIV Xilinx 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 a 8K locations.

PcieAlteraCycloneIV_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-AlteraCycloneIV Xilinx Channel RX FIFO Count

This read-only register port reports the number of 32-bit data words in the receive FIFO. The FIFO is actually made up of two FIFO's in series.   The first FIFO is 4Kx32 and the second is also 4Kx32. 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.

PcieAlteraCycloneIV_CH_FIFO

[0x010 Port Read/Write]

RX and TX FIFO Port

Data Bit	**Description**
31-0	FIFO data word

Figure 17 PCIe-AlteraCycloneIV Xilinx 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.

PcieAlteraCycloneIV_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-AlteraCycloneIV Xilinx 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.

PcieAlteraCycloneIV_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-AlteraCycloneIV Xilinx 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.

PcieAlteraCycloneIV_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-AlteraCycloneIV Xilinx 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.

PcieAlteraCycloneIV_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-AlteraCycloneIV Xilinx 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.

Altera Base 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 PCIeAlteraCycloneIV with the HDEterm100. For more information please refer to the web page.

PcieAlteraCycloneIV_BASE_ALT_ACCESS defines the offset from the Xilinx base address to the Altera 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 Altera Cyclone IV base address.

PcieAlteraCycloneIV_ALT_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-AlteraCycloneIV Altera Base Control Register

PCIeAlteraCycloneIV 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 PLL's also have independent references.

PLL_CLK_EN_(7..0).when set select 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 "ALT CNT" register. Once the PLL's are programmed the outputs can be used to count and check if the expected frequency is received.

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. 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 PLL's. 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.

Note: The "S2" bits are set to '0' in the RevA reference VHDL and will be grounded on the PCB with RevB and future revisions.

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 Altera. 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 Xilinx enable for the Altera 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.

PcieAlteraCycloneIV_ALT_LED

[0x004 read/write]

LED REGISTER

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

Figure 23 PCIe-AlteraCycloneIV Altera LED Control Register

LED Control when set (‘1’) and LED Enable is set (‘1’) causes the individual LED’s 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.

PcieAlteraCycloneIV_ALT_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-AlteraCycloneIV Altera 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 Altera flows through to the Xilinx 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 PLL's 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].

PcieAlteraCycloneIV_ALT_TTL_DAT

[0x010 TTL Data Port read/write]

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

Figure 25 PCIe-AlteraCycloneIV 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.

PcieAlteraCycloneIV_ALT_TTL_EN

[0x014 TTL Data Enable Port read/write]

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

Figure 26 PCIe-AlteraCycloneIV 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].

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.

PcieAlteraCycloneIV_ALT_CNT

[0x01C Count read-back port]

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

Figure 27 PCIe-AlteraCycloneIV Counter Results

There are 25 counters implemented to check the PLL programming. The Counter results to read are selected in the Altera 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 Cyclone assets.

Design note: not all 24 clocks are on Cyclone clock inputs. Please see the PLL pin definitions in the pinout table. It is recommended to put the higher rate clocks on the clock lines.

PcieAlteraCycloneIV_ALT_PLL_FIFO

[0x028 PLL Data R/W port]

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

Figure 28 PCIe-AlteraCycloneIV 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 PLL's and controlling HW.

Altera Channel Address Map

PcieAlteraCycloneIV_ALT_CH_BASE

[0x000 Main Control Register Port read/write]

Channel Base Register

DATA BIT	DESCRIPTION
31-25	spare
24	AxIoEn
23	XaIoEn
22	AxFfAFlIntEnLvl
21	XAFfAMtIntEnLvl
20	spare
19	AXFfIntEn
18	XAFfIntEn
17	AXStart
16	XAStart
15-5	spare
4	IntForce
3	MIntEn
2	Bypass
1	RxRst
0	TxRst

Figure 29 PCIe-AlteraCycloneIV 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 the Altera 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 Xilinx. The Master in the base level also needs to be set for this to work.

IntForce when set '1' and the MintEn is also 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.

XAStart when set enables data transfer via IO in the Xilinx to Altera direction. Transmitting to external system.

AXStart when set enables data transfer via IO in the Altera to Xilinx direction. Receiving from external system.

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

XAFfIntEn when set enables the XA direction FIFO Interrupt. The XA FIFO interrupt is latched [available in Status register] and triggered by the XA 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.

AXFfIntEn when set enables the AX direction FIFO Interrupt. The AX FIFO interrupt is latched [available in Status register] and triggered by the AF 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.

XAFfAMtIntEnLvl when set enables the level based interrupt for the XA 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 Xilinx channel. Defined for future implementation or client use. Unused in reference design.

AxFfAFlIntEnLvl when set enables the level based interrupt for the AX 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 Xilinx channel.   Defined for future implementation or client use. Unused in reference design.

XaIoEn 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 RX.

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

PcieAlteraCycloneIV_ALT_CH_STATUS

[0x004 Status Register Port read/write]

Channel Status PORT

DATA BIT	DESCRIPTION
31	IntStat
30	LocInt
29-12	spare
11	AXFfIntLat
10	XAFfIntLat
9	spare
8	spare
7	AXFfFl
6	AXFfAfl
5	AXFfAMt
4	AXFfMt
3	XAFfFl;
2	XAFfAFl
1	XAFfAMt
0	XAFfMt

Figure 30 PCIe-AlteraCycloneIV 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.

AXFfIntLat is latched and held until cleared by write to the status port with this bit position set. Almost Full is the FIFO condition.

XAFfIntLat is latched and held until cleared by write to the status port with this bit position set. Almost Empty condition in the FIFO triggers.

AXFfFl is set when the AX FIFO is Full.
AXFfAFl is set when the AX FIFO is Almost Full.
AXFfAMt is set when the AX FIFO is Almost Empty.
AXFfMt is set when the AX FIFO is Empty.

XAFfFl is set when the XA FIFO is Full.
XAFfAFl is set when the XA FIFO is Almost Full.
XAFfAMt is set when the XA FIFO is Almost Empty.
XAFfMt is set when the XA FIFO is Empty.

PcieAlteraCycloneIV_ALT_CH_SP

[0x008 Status Register Port read/write]

Channel Spare Port

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

Figure 31 PCIe-AlteraCycloneIV 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.

PcieAlteraCycloneIV_ALT_AX_FIFO_CNT

[0x00C Status Register Port read]

Channel AX FIFO Data Count

DATA BIT	**DESCRIPTION**
31-0	Data positions currently used in AX FIFO

Figure 32 PCIe-AlteraCycloneIV Channel AX 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 AX FIFO. Data is written as LW and read out as bytes before being transferred to the Xilinx on the corresponding data lane. This count is used to compare against the programmed levels to determine Almost Full etc.

PcieAlteraCycloneIV_ALT_XA_FIFO_CNT

[0x010 Status Register Port read]

Channel AX FIFO Data Count

DATA BIT	**DESCRIPTION**
31-0	Data positions currently used in XA FIFO

Figure 33 PCIe-AlteraCycloneIV Channel XA 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 AX and XA FIFO's 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 be 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 doing it".

PcieAlteraCycloneIV_ALT_FIFO_WR

[0x014 Status AX FIFO Write Port]

Channel AX FIFO Write

DATA BIT	**DESCRIPTION**
31-0	Data to write to AX FIFO

Figure 34 PCIe-AlteraCycloneIV Channel AX FIFO Write

Writing to this port will put data into the AX FIFO. The data will then transfer via the byte lanes through the Xilinx 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.

PcieAlteraCycloneIV_ALT_FIFO_RD

[0x014 Status AX FIFO Write Port]

Channel XA FIFO Read

DATA BIT	**DESCRIPTION**
31-0	Data read from XA FIFO

Figure 35 PCIe-AlteraCycloneIV Channel XA FIFO Read

Reading from this port will retrieve data from the XA FIFO. The data from the Xilinx side will transfer via the byte lanes from user memory [if DMA is set-up etc.] and into the Altera AX FIFO. The reference software has a loop-back test where data DMA'd to the channel's XA FIFO and then 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.

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 "PcieAlteraCycloneIV" ASSIGNED TO AN: EP4CE115F29I8

Pin Name/Usage	Location	Dir	I/O Standard	Bank	User Assignment
term[31]	A3	output	3.3-V LVTTL	8	Y
io[32]	A4	bidir	3.3-V LVTTL	8	Y
term[33]	A6	output	3.3-V LVTTL	8	Y
dir[34]	A7	output	3.3-V LVTTL	8	Y
term[35]	A8	output	3.3-V LVTTL	8	Y
ControlAddress[4]	A10	input	3.3-V LVTTL	8	Y
dir[36]	A11	output	3.3-V LVTTL	8	Y
term[37]	A12	output	3.3-V LVTTL	8	Y
PLL1_B	A14	input	3.3-V LVTTL	8	Y
PLL0_B	A15	input	3.3-V LVTTL	7	Y
dir[38]	A17	output	3.3-V LVTTL	7	Y
term[39]	A18	output	3.3-V LVTTL	7	Y
DATEN[11]	A19	output	3.3-V LVTTL	7	Y
ControlWen	A21	input	3.3-V LVTTL	7	Y
PLL3_S2	A22	output	3.3-V LVTTL	7	Y
PLL1_S2	A23	output	3.3-V LVTTL	7	Y
PLL7_S2	A25	output	3.3-V LVTTL	7	Y
PLL5_S2	A26	output	3.3-V LVTTL	7	Y
dir[9]	AA3	output	3.3-V LVTTL	2	Y
term[15]	AA4	output	3.3-V LVTTL	2	Y
term[13	AA5	output	3.3-V LVTTL	2	Y
DIN[3]	AA8	input	3.3-V LVTTL	3	Y
Ch5AXData[7]	AA14	output	3.3-V LVTTL	3	Y
Ch5XAData[3]	AA22	input	3.3-V LVTTL	5	Y
Ch4XAData[6]	AA24	input	3.3-V LVTTL	5	Y
Ch4XAData[5]	AA25	input	3.3-V LVTTL	5	Y
Ch4XAData[4]	AA26	input	3.3-V LVTTL	5	Y
io[9]	AB1	bidir	3.3-V LVTTL	2	Y
term[8]	AB2	output	3.3-V LVTTL	2	Y
dir[12]	AB3	output	3.3-V LVTTL	2	Y
io[12]	AB4	bidir	3.3-V LVTTL	2	Y
term[11]	AB5	output	3.3-V LVTTL	2	Y
Ch7XAData[6]	AB6	input	3.3-V LVTTL	2	Y
DOUT[3]	AB7	output	3.3-V LVTTL	3	Y
DATEN[3]	AB8	output	3.3-V LVTTL	3	Y
Ch6AXData[3]	AB11	output	3.3-V LVTTL	3	Y
Ch6AXData[4]	AB12	output	3.3-V LVTTL	3	Y
Ch6AXData[7]	AB13	output	3.3-V LVTTL	3	Y
Ch6AXEn	AB14	output	3.3-V LVTTL	3	Y
Ch6AXData[0]	AB15	output	3.3-V LVTTL	4	Y
Ch6AXData[2]	AB16	output	3.3-V LVTTL	4	Y
Ch6AXData[6]	AB17	output	3.3-V LVTTL	4	Y
Ch6XAData[0]	AB18	input	3.3-V LVTTL	4	Y

Ch5AXData[5]	AB19	output	3.3-V LVTTL	4	Y
Ch5AXData[3]	AB22	output	3.3-V LVTTL	4	Y
Ch5XAData[1	AB23	input	3.3-V LVTTL	5	Y
Ch5AXData[1]	AB24	output	3.3-V LVTTL	5	Y
Ch5XAData[2]	AB25	input	3.3-V LVTTL	5	Y
Ch4AXData[6]	AB26	output	3.3-V LVTTL	5	Y
Ch4AXData[0]	AB27	output	3.3-V LVTTL	5	Y
Ch4AXData[1]	AB28	output	3.3-V LVTTL	5	Y
io[7]	AC1	bidir	3.3-V LVTTL	2	Y
dir[7]	AC2	output	3.3-V LVTTL	2	Y
dir[10]	AC4	output	3.3-V LVTTL	2	Y
dir[8]	AC5	output	3.3-V LVTTL	2	Y
Ch7XAData[4]	AC7	input	3.3-V LVTTL	3	Y
DIN[2]	AC8	input	3.3-V LVTTL	3	Y
Ch7AXAFull	AC12	input	3.3-V LVTTL	3	Y
Ch6AXData[1]	AC15	output	3.3-V LVTTL	4	Y
Ch6AXData[5]	AC17	output	3.3-V LVTTL	4	Y
Ch6XAData[4]	AC18	input	3.3-V LVTTL	4	Y
Ch6XAData[2]	AC19	input	3.3-V LVTTL	4	Y
Ch5AXData[4]	AC21	output	3.3-V LVTTL	4	Y
Ch5AXData[2]	AC22	output	3.3-V LVTTL	4	Y
Ch5AXData[0]	AC24	output	3.3-V LVTTL	5	Y
PLL_REF6	AC25	output	3.3-V LVTTL	5	Y
Ch4AXData[7]	AC26	output	3.3-V LVTTL	5	Y
Ch4AXData[3]	AC27	output	3.3-V LVTTL	5	Y
Ch4AXData[2]	AC28	output	3.3-V LVTTL	5	Y
term[6]	AD1	output	3.3-V LVTTL	2	Y
dir[5]	AD2	output	3.3-V LVTTL	2	Y
io[10]	AD3	bidir	3.3-V LVTTL	2	Y
term[9]	AD4	output	3.3-V LVTTL	3	Y
dir[6]	AD5	output	3.3-V LVTTL	3	Y
DOUT[2]	AD7	output	3.3-V LVTTL	3	Y
DATEN[2]	AD8	output	3.3-V LVTTL	3	Y
Ch7XAData[1]	AD10	input	3.3-V LVTTL	3	Y
Ch7XAEn	AD11	input	3.3-V LVTTL	3	Y
Ch7AXData[1]	AD12	output	3.3-V LVTTL	3	Y
Ch7XAFull	AD14	output	3.3-V LVTTL	3	Y
Ch6XAData[5]	AD17	input	3.3-V LVTTL	4	Y
Ch6XAData[3]	AD18	input	3.3-V LVTTL	4	Y
Ch1XAEn	AD21	input	3.3-V LVTTL	4	Y
Ch1AXAFull	AD22	input	3.3-V LVTTL	4	Y
Ch2XAEn	AD24	input	3.3-V LVTTL	4	Y
Ch2AXFull	AD25	input	3.3-V LVTTL	4	Y
Ch5XAData[0]	AD26	input	3.3-V LVTTL	5	Y
Ch4AXData[5]	AD27	output	3.3-V LVTTL	5	Y
Ch4AXData[4]	AD28	output	3.3-V LVTTL	5	Y
io[5]	AE1	bidir	3.3-V LVTTL	2	Y
term[4]	AE2	output	3.3-V LVTTL	2	Y
io[8]	AE3	bidir	3.3-V LVTTL	2	Y
io[6]	AE4	bidir	3.3-V LVTTL	3	Y
term[5]	AE5	output	3.3-V LVTTL	3	Y
term[1]	AE6	output	3.3-V LVTTL	3	Y
dir[0]	AE7	output	3.3-V LVTTL	3	Y

DATEN[0]	AE8	output	3.3-V LVTTL	3	Y
io[0]	AE9	bidir	3.3-V LVTTL	3	Y
Ch7AXEn	AE10	output	3.3-VLVTTL	3	Y
Ch7AXData[5]	AE12	output	3.3-VLVTTL	3	Y
Ch7AXData[7]	AE13	output	3.3-VLVTTL	3	Y
Ch7AXData[0]	AE14	output	3.3-VLVTTL	3	Y
Ch6XAData[6]	AE15	input	3.3-VLVTTL	4	Y
Ch3XAFull	AE16	output	3.3-VLVTTL	4	Y
Ch3XAAFull	AE17	output	3.3-VLVTTL	4	Y
Ch2XAFull	AE18	output	3.3-VLVTTL	4	Y
Ch2XAAFull	AE19	output	3.3-VLVTTL	4	Y
Ch6XAData[1]	AE20	input	3.3-VLVTTL	4	Y
Ch1XAFull	AE21	output	3.3-VLVTTL	4	Y
Ch1AXFull	AE22	input	3.3-VLVTTL	4	Y
A_X_RefClk	AE23	output	3.3-VLVTTL	4	Y
Ch3AXFull	AE24	input	3.3-VLVTTL	4	Y
Ch2AXAFull	AE25	input	3.3-VLVTTL	4	Y
Ch5XAEn	AE26	input	3.3-VLVTTL	5	Y
Ch5AXAFull	AE27	input	3.3-VLVTTL	5	Y
Ch5AXFull	AE28	input	3.3-VLVTTL	5	Y
io[3]	AF2	bidir	3.3-VLVTTL	2	Y
term[7]	AF3	output	3.3-VLVTTL	3	Y
dir[4]	AF4	output	3.3-VLVTTL	3	Y
io[4]	AF5	bidir	3.3-VLVTTL	3	Y
io[2]	AF6	bidir	3.3-VLVTTL	3	Y
DIN[1]	AF7	input	3.3-VLVTTL	3	Y
DOUT[0]	AF8	output	3.3-VLVTTL	3	Y
Ch7AXFull	AF9	input	3.3-VLVTTL	3	Y
Ch7AXData[6]	AF10	output	3.3-VLVTTL	3	Y
Ch7XAAFull	AF11	output	3.3-VLVTTL	3	Y
Ch6XAData[7]	AF12	input	3.3-VLVTTL	3	Y
Ch7AXData[2]	AF13	output	3.3-VLVTTL	3	Y
Ch7AXData[4]	AF14	output	3.3-VLVTTL	3	Y
Ch7AXData[3]	AF15	output	3.3-VLVTTL	4	Y
Ch2AXEn	AF18	output	3.3-VLVTTL	4	Y
Ch1AXEn	AF20	output	3.3-VLVTTL	4	Y
Ch1XAAFull	AF21	output	3.3-VLVTTL	4	Y
Ch0XAFull	AF22	output	3.3-VLVTTL	4	Y
Ch3XAEn	AF24	input	3.3-VLVTTL	4	Y
Ch3AXAFull	AF25	input	3.3-VLVTTL	4	Y
Ch6AXFull	AF26	input	3.3-VLVTTL	4	Y
Ch6AXAFull	AF27	input	3.3-VLVTTL	5	Y
dir[3]	AG3	output	3.3-VLVTTL	3	Y
dir[1]	AG4	output	3.3-VLVTTL	3	Y
dir[2]	AG6	output	3.3-VLVTTL	3	Y
term[3]	AG7	output	3.3-VLVTTL	3	Y
DOUT[1]	AG8	output	3.3-VLVTTL	3	Y
Ch7XAData[5]	AG10	input	3.3-VLVTTL	3	Y
Ch7XAData[2]	AG11	input	3.3-VLVTTL	3	Y
Ch5AXData[6]	AG12	output	3.3-VLVTTL	3	Y
PLL4_A	AG14	input	3.3-VLVTTL	3	Y
PLL5_B	AG15	input	3.3-VLVTTL	4	Y
Ch6XAAFull	AG17	output	3.3-VLVTTL	4	Y

Ch5XAFull	AG18	output	3.3-VLVTTL	4	Y
Ch4AXEn	AG19	output	3.3-VLVTTL	4	Y
Ch4XAAFull	AG21	output	3.3-VLVTTL	4	Y
Ch0AXEn	AG22	output	3.3-VLVTTL	4	Y
Ch4AXFull	AG25	input	3.3-VLVTTL	4	Y
Ch6XAEn	AG26	input	3.3-VLVTTL	4	Y
term[2]	AH3	output	3.3-VLVTTL	3	Y
io[1]	AH4	bidir	3.3-VLVTTL	3	Y
term[0]	AH6	output	3.3-VLVTTL	3	Y
DATEN[1]	AH7	output	3.3-VLVTTL	3	Y
DIN[0]	AH8	input	3.3-VLVTTL	3	Y
Ch7XAData[7]	AH10	input	3.3-VLVTTL	3	Y
Ch7XAData[3]	AH11	input	3.3-VLVTTL	3	Y
Ch7XAData[0]	AH12	input	3.3-VLVTTL	3	Y
PLL4_B	AH14	input	3.3-VLVTTL	3	Y
X_A_RefClk	AH15	input	3.3-VLVTTL	4	Y
Ch6XAFull	AH17	output	3.3-VLVTTL	4	Y
Ch5AXEn	AH18	output	3.3-VLVTTL	4	Y
Ch5XAAFull	AH19	output	3.3-VLVTTL	4	Y
Ch4XAFull	AH21	output	3.3-VLVTTL	4	Y
Ch3AXEn	AH22	output	3.3-VLVTTL	4	Y
Ch0XAAFull	AH23	output	3.3-VLVTTL	4	Y
Ch4XAEn	AH25	input	3.3-VLVTTL	4	Y
Ch4AXAFull	AH26	input	3.3-VLVTTL	4	Y
dir[32]	B4	output	3.3-VLVTTL	8	Y
io[39]	B6	bidir	3.3-VLVTTL	8	Y
io[34]	B7	bidir	3.3-VLVTTL	8	Y
DIN[10]	B8	input	3.3-VLVTTL	8	Y
ControlData[13]	B10	bidir	3.3-VLVTTL	8	Y
io[36]	B11	bidir	3.3-VLVTTL	8	Y
PLL2_B	B14	input	3.3-VLVTTL	8	Y
ControlClk	B15	input	3.3-VLVTTL	7	Y
io[38]	B17	bidir	3.3-VLVTTL	7	Y
DIN[11]	B18	input	3.3-VLVTTL	7	Y
DOUT[11]	B19	output	3.3-VLVTTL	7	Y
ControlAddress[11]	B21	input	3.3-VLVTTL	7	Y
PLL2_S2	B22	output	3.3-VLVTTL	7	Y
PLL0_S2	B23	output	3.3-VLVTTL	7	Y
PLL6_S2	B25	output	3.3-VLVTTL	7	Y
PLL4_S2	B26	output	3.3-VLVTTL	7	Y
io[28]	C2	bidir	3.3-VLVTTL	1	Y
term[36]	C3	output	3.3-VLVTTL	8	Y
io[37]	C4	bidir	3.3-VLVTTL	8	Y
term[38]	C5	output	3.3-VLVTTL	8	Y
dir[39]	C6	output	3.3-VLVTTL	8	Y
DATEN[10]	C7	output	3.3-VLVTTL	8	Y
DOUT[10]	C8	output	3.3-VLVTTL	8	Y
ControlData[12]	C9	bidir	3.3-VLVTTL	8	Y
ControlData[15]	C10	bidir	3.3-VLVTTL	8	Y
ControlData[11]	C11	bidir	3.3-VLVTTL	8	Y
PLL_REF3	C12	output	3.3-VLVTTL	8	Y
ControlData[18]	C13	bidir	3.3-VLVTTL	8	Y
PLL_REF5	C14	output	3.3-VLVTTL	8	Y

LED[3]	C15	output	3.3-VLVTTL	7	Y
ControlData[23]	C16	bidir	3.3-VLVTTL	7	Y
ControlAddress[0]	C17	input	3.3-VLVTTL	7	Y
ControlData[26]	C18	bidir	3.3-VLVTTL	7	Y
ControlData[28]	C19	bidir	3.3-VLVTTL	7	Y
ControlData[31]	C20	bidir	3.3-VLVTTL	7	Y
ControlAddress[6]	C21	input	3.3-VLVTTL	7	Y
PLL3_C	C22	input	3.3-VLVTTL	7	Y
PLL7_A	C24	input	3.3-VLVTTL	7	Y
PLL6_A	C25	input	3.3-VLVTTL	7	Y
PLL5_A	C26	input	3.3-VLVTTL	7	Y
PLL7_C	C27	input	3.3-VLVTTL	6	Y
term[27]	D1	output	3.3-VLVTTL	1	Y
dir[28]	D2	output	3.3-VLVTTL	1	Y
io[35]	D4	bidir	3.3-VLVTTL	8	Y
dir[35]	D5	output	3.3-VLVTTL	8	Y
dir[37]	D6	output	3.3-VLVTTL	8	Y
PLL_REF4	D7	output	3.3-VLVTTL	8	Y
DIN[9]	D8	input	3.3-VLVTTL	8	Y
LED[0]	D9	output	3.3-VLVTTL	8	Y
ControlData[14]	D10	bidir	3.3-VLVTTL	8	Y
ControlData[17]	D11	bidir	3.3-VLVTTL	8	Y
ControlAddress[9]	D15	input	3.3-VLVTTL	7	Y
PLL_REF1	D16	output	3.3-VLVTTL	7	Y
ControlInt	D17	output	3.3-VLVTTL	7	Y
ControlAddress[3]	D18	input	3.3-VLVTTL	7	Y
Rst	D19	input	3.3-VLVTTL	7	Y
ControlAddress[10]	D20	input	3.3-VLVTTL	7	Y
ControlAddress[7]	D21	input	3.3-VLVTTL	7	Y
PLL2_C	D22	input	3.3-VLVTTL	7	Y
PLL1_C	D24	input	3.3-VLVTTL	7	Y
PLL0_C	D25	input	3.3-VLVTTL	7	Y
PLL2_SCLK	D26	output	3.3-VLVTTL	6	Y
PLL3_SCLK	D27	output	3.3-VLVTTL	6	Y
Ch0AXData[3]	D28	output	3.3-VLVTTL	6	Y
io[26]	E1	bidir	3.3-VLVTTL	1	Y
~term[34]	E3	output	3.3-VLVTTL	1	Y
io[33]	E4	bidir	3.3-VLVTTL	8	Y
dir[33]	E5	output	3.3-VLVTTL	8	Y
DOUT[9]	E7	output	3.3-VLVTTL	8	Y
DATEN[9]	E8	output	3.3-VLVTTL	8	Y
PLL_REF2	E10	output	3.3-VLVTTL	8	Y
PLL_REF7	E12	output	3.3-VLVTTL	8	Y
ControlData[16]	E14	bidir	3.3-VLVTTL	8	Y
ControlData[21]	E15	bidir	3.3-VLVTTL	7	Y
ControlData[24]	E17	bidir	3.3-VLVTTL	7	Y
ControlData[29]	E18	bidir	3.3-VLVTTL	7	Y
PLL_DATA	E19	bidir	3.3-VLVTTL	7	Y
ControlAddress[5]	E21	input	3.3-VLVTTL	7	Y
PLL6_C	E22	input	3.3-VLVTTL	7	Y
PLL5_C	E24	input	3.3-VLVTTL	7	Y
PLL1_SCLK	E25	output	3.3-VLVTTL	7	Y
PLL0_SCLK	E26	output	3.3-VLVTTL	6	Y

Ch0AXData[1]	E27	output	3.3-VLVTTL	6	Y
Ch0AXData[6]	E28	output	3.3-VLVTTL	6	Y
term[25]	F1	output	3.3-VLVTTL	1	Y
io[24]	F2	bidir	3.3-VLVTTL	1	Y
dir[26]	F3	output	3.3-VLVTTL	1	Y
term[32]	F5	output	3.3-VLVTTL	1	Y
DIN[8]	F8	input	3.3-VLVTTL	8	Y
ControlData[10]	F10	bidir	3.3-VLVTTL	8	Y
ControlData[8]	F12	bidir	3.3-VLVTTL	8	Y
ControlData[20]	F14	bidir	3.3-VLVTTL	8	Y
ControlData[19]	F15	bidir	3.3-VLVTTL	7	Y
ControlData[25]	F17	bidir	3.3-VLVTTL	7	Y
ControlData[30]	F19	bidir	3.3-VLVTTL	7	Y
ControlAddress[8]	F21	input	3.3-VLVTTL	7	Y
PLL4_C	F22	input	3.3-VLVTTL	7	Y
Ch2AXData[2]	F24	output	3.3-VLVTTL	6	Y
ControlAddress[2]	F25	input	3.3-VLVTTL	6	Y
Ch0AXData[0]	F26	output	3.3-VLVTTL	6	Y
Ch0AXData[2]	F27	output	3.3-VLVTTL	6	Y
Ch0AXData[5]	F28	output	3.3-VLVTTL	6	Y
dir[24]	G1	output	3.3-VLVTTL	1	Y
term[23]	G2	output	3.3-VLVTTL	1	Y
io[30]	G3	bidir	3.3-VLVTTL	1	Y
dir[30]	G4	output	3.3-VLVTTL	1	Y
term[29]	G5	output	3.3-VLVTTL	1	Y
DATEN[8]	G7	output	3.3-VLVTTL	8	Y
DOUT[8]	G8	output	3.3-VLVTTL	8	Y
ControlData[9]	G10	bidir	3.3-VLVTTL	8	Y
ControlData[6]	G12	bidir	3.3-VLVTTL	8	Y
ControlData[5]	G14	bidir	3.3-VLVTTL	8	Y
ControlData[4]	G15	bidir	3.3-VLVTTL	7	Y
ControlData[22]	G16	bidir	3.3-VLVTTL	7	Y
ControlData[1]	G17	bidir	3.3-VLVTTL	7	Y
ControlData[27]	G18	bidir	3.3-VLVTTL	7	Y
LED[2]	G20	output	3.3-VLVTTL	7	Y
PLL3_B	G21	input	3.3-VLVTTL	7	Y
ControlAddress[12]	G22	input	3.3-VLVTTL	7	Y
ControlAddress[1]	G23	input	3.3-VLVTTL	6	Y
Ch2AXData[3]	G24	output	3.3-VLVTTL	6	Y
Ch0AXData[4]	G25	output	3.3-VLVTTL	6	Y
Ch0AXData[7]	G26	output	3.3-VLVTTL	6	Y
Ch0XAData[7]	G27	input	3.3-VLVTTL	6	Y
Ch0XAData[6]	G28	input	3.3-VLVTTL	6	Y
dir[31]	H3	output	3.3-VLVTTL	1	Y
io[31]	H4	bidir	3.3-VLVTTL	1	Y
LED[1]	H5	output	3.3-VLVTTL	1	Y
DIN[7]	H8	input	3.3-VLVTTL	8	Y
ControlData[7]	H12	bidir	3.3-VLVTTL	8	Y
ControlRen	H14	input	3.3-VLVTTL	8	Y
ControlData[3]	H16	bidir	3.3-VLVTTL	7	Y
ControlData[2]	H17	bidir	3.3-VLVTTL	7	Y
PLL_REF0	H22	output	3.3-VLVTTL	6	Y
Ch0XAData[5]	H23	input	3.3-VLVTTL	6	Y

Ch1XAData[6]	H24	input	3.3-VLVTTL	6	Y
Ch2AXData[6]	H25	output	3.3-VLVTTL	6	Y
Ch0XAData[4]	H26	input	3.3-VLVTTL	6	Y
PLL3_A	J1	input	3.3-VLVTTL	1	Y
io[22]	J3	bidir	3.3-VLVTTL	1	Y
io[29]	J4	bidir	3.3-VLVTTL	1	Y
term[30]	J5	output	3.3-VLVTTL	1	Y
DATEN[7]	J7	output	3.3-VLVTTL	1	Y
ControlData[0]	J19	bidir	3.3-VLVTTL	7	Y
Ch0XAData[1]	J22	input	3.3-VLVTTL	6	Y
Ch0XAData[3]	J23	input	3.3-VLVTTL	6	Y
Ch0XAData[0]	J24	input	3.3-VLVTTL	6	Y
Ch1XAData[0]	J25	input	3.3-VLVTTL	6	Y
Ch0XAData[2]	J26	input	3.3-VLVTTL	6	Y
PLL0_A	J27	input	3.3-VLVTTL	6	Y
Osc	J28	input	3.3-VLVTTL	6	Y
dir[22]	K1	output	3.3-VLVTTL	1	Y
term[21]	K2	output	3.3-VLVTTL	1	Y
term[26]	K3	output	3.3-VLVTTL	1	Y
dir[29]	K4	output	3.3-VLVTTL	1	Y
DOUT[7]	K8	output	3.3-VLVTTL	1	Y
Ch1XAData[2]	K25	input	3.3-VLVTTL	6	Y
Ch1XAData[4]	K26	input	3.3-VLVTTL	6	Y
PLL7_SCLK	K27	output	3.3-VLVTTL	6	Y
Ch2AXData[0]	K28	output	3.3-VLVTTL	6	Y
io[20]	L1	bidir	3.3-VLVTTL	1	Y
dir[20]	L2	output	3.3-VLVTTL	1	Y
dir[27]	L3	output	3.3-VLVTTL	1	Y
term[28]	L4	output	3.3-VLVTTL	1	Y
io[27]	L5	bidir	3.3-VLVTTL	1	Y
DOUT[6]	L6	output	3.3-VLVTTL	1	Y
DIN[6]	L7	input	3.3-VLVTTL	1	Y
Ch2AXData[7]	L22	output	3.3-VLVTTL	6	Y
PLL6_SCLK	L23	output	3.3-VLVTTL	6	Y
Ch2AXData[4]	L24	output	3.3-VLVTTL	6	Y
Ch1XAData[7]	L25	input	3.3-VLVTTL	6	Y
PLL5_SCLK	L26	output	3.3-VLVTTL	6	Y
PLL4_SCLK	L27	output	3.3-VLVTTL	6	Y
Ch1XAData[3]	L28	input	3.3-VLVTTL	6	Y
io[19]	M1	bidir	3.3-VLVTTL	1	Y
dir[19]	M2	output	3.3-VLVTTL	1	Y
io[25]	M3	bidir	3.3-VLVTTL	1	Y
term[24]	M4	output	3.3-VLVTTL	1	Y
dir[25]	M5	output	3.3-VLVTTL	1	Y
DATEN[6]	M7	output	3.3-VLVTTL	1	Y
Ch1AXData[5]	M23	output	3.3-VLVTTL	6	Y
Ch1XAData[1]	M24	input	3.3-VLVTTL	6	Y
Ch1XAData[5]	M25	input	3.3-VLVTTL	6	Y
Ch1AXData[3]	M26	output	3.3-VLVTTL	6	Y
Ch1AXData[2]	M27	output	3.3-VLVTTL	6	Y
Ch1AXData[1]	M28	output	3.3-VLVTTL	6	Y
io[23]	N4	bidir	3.3-VLVTTL	1	Y
DIN[4]	N8	input	3.3-VLVTTL	1	Y

Ch2AXData[1]	N25	output	3.3-VLVTTL	6	Y
Ch3XAData[0]	N26	input	3.3-VLVTTL	6	Y
term[18]	P1	output	3.3-VLVTTL	1	Y
dir[17]	P2	output	3.3-VLVTTL	1	Y
Ch1AXData[7]	P21	output	3.3-VLVTTL	5	Y
Ch1AXData[4]	P26	output	3.3-VLVTTL	6	Y
Ch1AXData[6]	P27	output	3.3-VLVTTL	6	Y
io[17]	R1	bidir	3.3-VLVTTL	2	Y
term[16]	R2	output	3.3-VLVTTL	2	Y
term[22]	R4	output	3.3-VLVTTL	2	Y
io[21]	R5	bidir	3.3-VLVTTL	2	Y
dir[21]	R6	output	3.3-VLVTTL	2	Y
dir[23]	R7	output	3.3-VLVTTL	2	Y
Ch3AXData[6]	R22	output	3.3-VLVTTL	5	Y
Ch3AXData[4]	R23	output	3.3-VLVTTL	5	Y
Ch3AXData[5]	R24	output	3.3-VLVTTL	5	Y
Ch3AXData[3]	R25	output	3.3-VLVTTL	5	Y
Ch3AXData[1]	R26	output	3.3-VLVTTL	5	Y
Ch3XAData[1]	R27	input	3.3-VLVTTL	5	Y
Ch3XAData[2]	R28	input	3.3-VLVTTL	5	Y
io[15]	T3	bidir	3.3-VLVTTL	2	Y
Ch2AXData[5]	T22	output	3.3-VLVTTL	5	Y
Ch3AXData[2]	T25	output	3.3-VLVTTL	5	Y
Ch1AXData[0]	T26	output	3.3-VLVTTL	5	Y
dir[15]	U1	output	3.3-VLVTTL	2	Y
term[14]	U2	output	3.3-VLVTTL	2	Y
dir[18]	U3	output	3.3-VLVTTL	2	Y
io[18]	U4	bidir	3.3-VLVTTL	2	Y
term[19]	U5	output	3.3-VLVTTL	2	Y
term[20]	U6	output	3.3-VLVTTL	2	Y
DOUT[4]	U7	output	3.3-VLVTTL	2	Y
DATEN[4]	U8	output	3.3-VLVTTL	2	Y
Ch3AXData[7]	U21	output	3.3-VLVTTL	5	Y
Ch3AXData[0]	U22	output	3.3-VLVTTL	5	Y
Ch2XAData[0]	U23	input	3.3-VLVTTL	5	Y
Ch2XAData[6]	U24	input	3.3-VLVTTL	5	Y
Ch2XAData[4]	U25	input	3.3-VLVTTL	5	Y
Ch2XAData[3]	U26	input	3.3-VLVTTL	5	Y
Ch3XAData[3]	U27	input	3.3-VLVTTL	5	Y
Ch3XAData[5]	U28	input	3.3-VLVTTL	5	Y
dir[13]	V1	output	3.3-VLVTTL	2	Y
io[13]	V2	bidir	3.3-VLVTTL	2	Y
term[17]	V3	output	3.3-VLVTTL	2	Y
dir[16]	V4	output	3.3-VLVTTL	2	Y
DIN[5]	V5	input	3.3-VLVTTL	2	Y
Ch5XAData[7]	V21	input	3.3-VLVTTL	5	Y
Ch2XAData[1]	V22	input	3.3-VLVTTL	5	Y
Ch2XAData[7]	V23	input	3.3-VLVTTL	5	Y
Ch2XAData[2]	V24	input	3.3-VLVTTL	5	Y
Ch2XAData[5]	V25	input	3.3-VLVTTL	5	Y
Ch0AXAFull	V26	input	3.3-VLVTTL	5	Y
Ch3XAData[4]	V27	input	3.3-VLVTTL	5	Y
Ch3XAData[6]	V28	input	3.3-VLVTTL	5	Y

term[12]	W1	output	3.3-VLVTTL	2	Y
io[11]	W2	bidir	3.3-VLVTTL	2	Y
io[16]	W3	bidir	3.3-VLVTTL	2	Y
io[14]	W4	bidir	3.3-VLVTTL	2	Y
term[10]	W7	output	3.3-VLVTTL	2	Y
Ch5XAData[5]	W21	input	3.3-VLVTTL	5	Y
Ch5XAData[6]	W22	input	3.3-VLVTTL	5	Y
Ch3XAData[7]	W25	input	3.3-VLVTTL	5	Y
Ch4XAData[2]	W26	input	3.3-VLVTTL	5	Y
Ch4XAData[1]	W27	input	3.3-VLVTTL	5	Y
Ch4XAData[0]	W28	input	3.3-VLVTTL	5	Y
PLL2_A	Y1	input	3.3-VLVTTL	2	Y
PLL1_A	Y2	input	3.3-VLVTTL	2	Y
dir[11]	Y3	output	3.3-VLVTTL	2	Y
dir[14]	Y4	output	3.3-VLVTTL	2	Y
DOUT[5]	Y5	output	3.3-VLVTTL	2	Y
DATEN[5]	Y6	output	3.3-VLVTTL	2	Y
Ch5XAData[4]	Y22	input	3.3-VLVTTL	5	Y
Ch4XAData[7]	Y23	input	3.3-VLVTTL	5	Y
Ch0XAEn	Y24	input	3.3-VLVTTL	5	Y
Ch0AXFull	Y25	input	3.3-VLVTTL	5	Y
Ch4XAData[3]	Y26	input	3.3-VLVTTL	5	Y
PLL7_B	Y27	input	3.3-VLVTTL	5	Y
PLL6_B	Y28	input	3.3-VLVTTL	5	Y

The pin names match with the VHDL reference kit and are close enough to match against the schematic. 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.
The direction and voltage level are defined for each term.
There are 8 PLLs each with 3 (a,b,c) outputs.
PLL = Phase Locked Loop
WEN = Write Enable
REN = Read Enable
FF = Full Flag
MT = Empty Flag
OSC is connected to the on-board 133.33 MHz reference.

The FPGA type assigned in the project is for the industrial part. The standard part is commercial with Industrial temp being an option.

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 PCIeAlteraCycloneIV 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 VCCB		25	75
fused +5 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-AlteraCycloneIV D100 Pinout

VCCB is selected to match IO requirement for differential. 5V for RS485 and 3.3V for LVDS or mixed IO.

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 PCIe-AlteraCycloneIV. 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. The PCIe-AlteraCycloneIV 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 with 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 PCIe-AlteraCycloneIV 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.

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:	local 133.33 MHz oscillator on Altera, 8 PLLs to provide programmable frequencies. 50 MHz GPB clock, internal [Altera] PLLs.
Memory	FIFO memory is provided to support DMA both within the Xilinx and the Altera 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] 5787082-9 is the board side part number
Software Interface:	Control Registers within Xilinx. User defined within Altera. VHDL reference design for Altera. 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:	Shunts for 3.3V or 5V reference to TTL IO. Separate shunts for IC reference and IO reference. Recommend both set to the same value. "Active Serial" header for Altera FLASH programming. Bezel Grounding option [AC/DC/open] with J6.
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 and 1.2V created with on-board power supplies.
User	8 position software readable switch 4 software controllable LEDs Power Supply LEDs

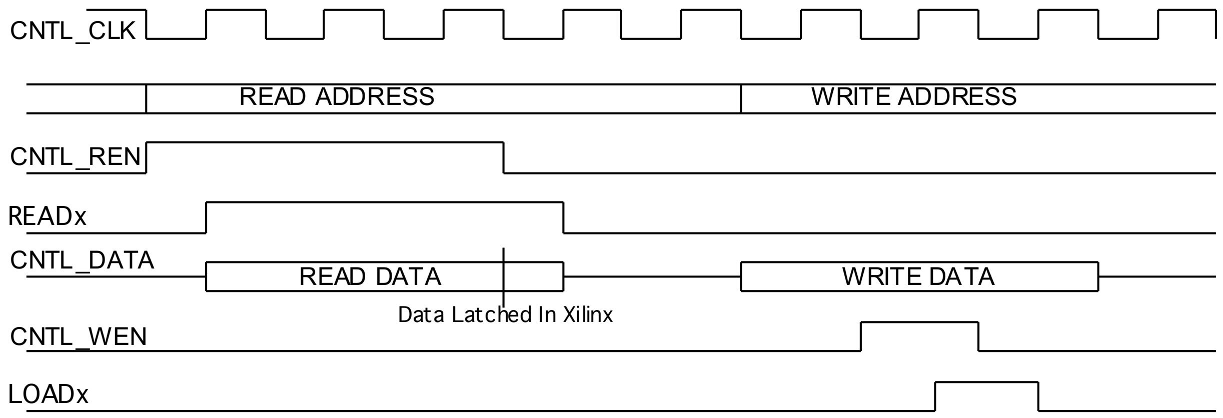
Order Information

Standard temperature range 0-70ºC

PCIe-AlteraCycloneIV	https://www.dyneng.com/PcieAlteraCycloneIV.html half-length PCIe card with user re-configurable Cyclone IV [EP4CE115F29C8], 40 RS-485 IO, 12 TTL IO, 8 PLL [24 clocks]
PCIe-AlteraCycloneIV-ENG	Engineering Kit for the PCIeAlteraCycloneIV Software, Schematic, Cable and HDEterm100, reference Altera implementation. Please refer to the webpage for more information. Win10 and Linux driver / reference SW currently available.
-ET	Extended temperature range -40 - 85ºC Contact Dynamic Engineering for this option.
-LVDS	Switch to LVDS IO instead of RS485.
-ROHS	Change to ROHS processing. Default is Sn/Pb solder.
-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-AlteraCycloneIV and HDEterm100. Length options

Appendix

General Purpose Bus Timing



The basic timing for the GPB relative to the Altera device is shown. The CNTL clock operates at 50 MHz. Signals originating from the Xilinx are clocked on the falling edge of the control clock to provide 1/2 period of set-up and hold minus the skew and switching times. Skew is kept to a minimum with edge FFs and controlled routing within the PCB.

A host read to the Altera over the GPB starts with the address and Read Enable being asserted. The Altera 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 falling edge of CNTL_REN as shown. Approximately 1/2 period of hold is provided to the Xilinx by the Altera, and the Altera has approximately 2 1/2 periods to provide the data to the Xilinx.

A host write to the Altera over the GPB starts with the address and data being asserted. After a 1 period delay CNTL_WEN is asserted. The Altera 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. Approximately 1/2 period of hold and 2 1/2 periods of set-up are provided to the Altera.

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

Further documentation is provided in-line with the VHDL.