DYNAMIC ENGINEERING
435 Park Dr., Ben Lomond, Calif. 95005
831-336-8891 Fax 831-336-3840
http://www.dyneng.com
sales@dyneng.com
Est. 1988

User Manual

PCI-Serial-ECL

Programmable ECL IO with PCI DMA

Revision A2
Corresponding Hardware: Revision A/B
10-2004-0301, 10-2004-0302

PCI-Serial-ECL
PCI based ECL IO w/ DMA

Dynamic Engineering
435 Park Drive
Ben Lomond, CA 95005
831-336-8891
831-336-3840 FAX

Manual Revision A. Revised 6/18/04

Dynamic Engineering has made every effort to ensure that this manual is accurate and complete. Still, the company reserves the right to make improvements or changes in the product described in this document at any time and without notice. Furthermore, Dynamic Engineering assumes no liability arising out of the application or use of the device described herein.

The electronic equipment described herein generates, uses, and can radiate radio frequency energy. Operation of this equipment in a residential area is likely to cause radio interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Dynamic Engineering's products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Dynamic Engineering.

Connection of incompatible hardware is likely

Table of Contents

List of Figures

Product Description Part I

PCI-Serial-ECL is part of the PCI Compatible family of modular I/O components. The PCI-Serial-ECL provides a Virtex II Pro FPGA, along with 38 ECL [NECL] and 12 TTL I/O lines, a programmable PLL and FIFO support with full DMA capabilities in a half-length single slot card.

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

The PCI-Serial-ECL utilizes a PLX 9054 device for the PCI interface, and a Xilinx FPGA to manage the 9054 and provide the transmit and receive state-machine control. The 9054 supports scatter-gather DMA and the FPGA supports burst transfers to allow high speed DMA data transfers.

The external FIFO is a 128K x 32 device that can operate at 66 MHz. The FPGA features Block RAM that can be configured to provide additional FIFO or other memory to support the IO.

The transmit data path in a typical configuration uses DMA to move data from system memory to the external holding FIFO. The transmit state-machine would then read the data from the holding FIFO, reformat as required and transmit out of the ECL or TTL ports. Similarly if configured for receive the data is reformatted, any error checking performed and then loaded into the external FIFO. DMA will move the data from the FIFO to the system memory.

The Cypress 22393 PLL is handy for creating user specific frequencies with which to operate the state-machines and IO. The driver supports programming the PLL over a serial I2C bus. Three clocks are received from the PLL onto FPGA long-lines. The clock routing uses matched lengths to provide in-phase references should they be necessary in your design. The FPGA DLLs provide further clock functionality. The base clock tree uses the PCI clock and a pair of DLLs for low-skew on-chip distribution to generate an inverted clock and a 66 MHz in phase copy. The 66 MHz is routed to the PLL for its clock reference. A user Oscillator position is also provided to allow for custom frequencies to be generated when

the PLL programming is not exact enough for your application.

Cypress has a utility available for calculating the frequency control words for the PLL. http://www.dyneng.com/CyberClocks.zip is the URL for the Cypress software used to calculate the PLL programming words. The PLL responds to one of two addresses [only one works]. As part of our ATP our software determines the address of the PLL and prints it out. A label is attached to the shipping bag with the PLL addresses for the user's convenience. The software is part of the engineering kit and can be ported to your application

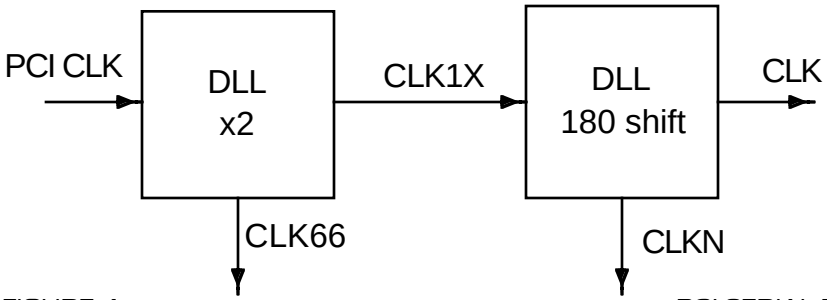


FIGURE 1 PCI-SERIAL-ECL CLOCK DISTRIBUTION

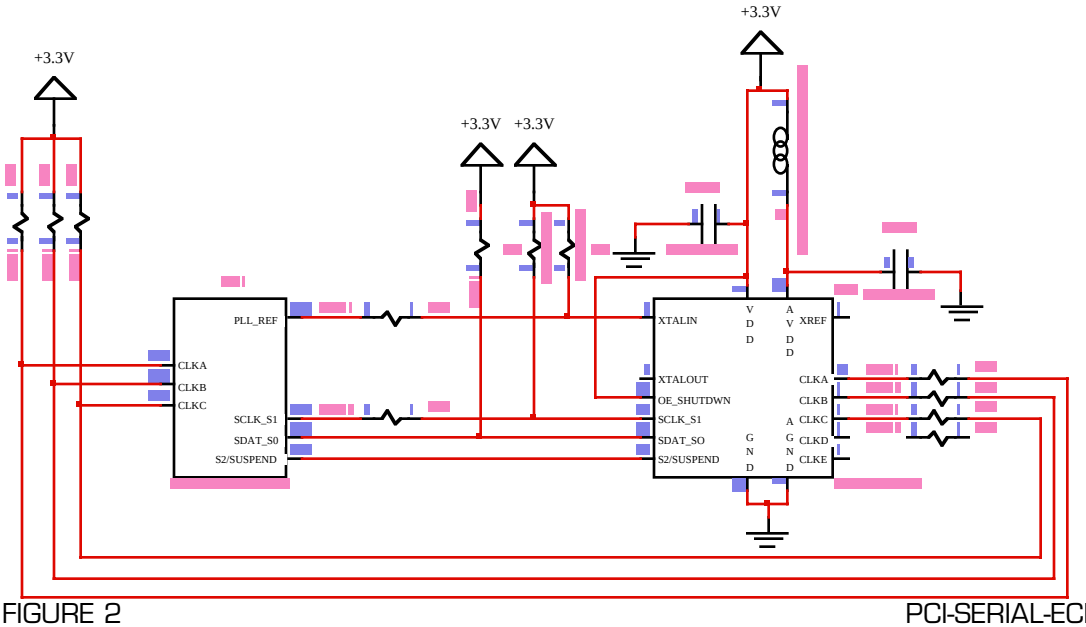


FIGURE 2 PCI-SERIAL-ECL PLL

An 8-bit "dip switch" is provided on the PCI-Serial-ECL. The switch configuration is readable via a register. The switch is for user-defined purposes. We envision the

switch being used for software configuration control, PCI board identification or test purposes.
LEDs are provided on the board. One LED is attached to the the Xilinx via a register controlled pin. The LED initially flashes based in on the PCI clock divided down to a human viewable rate. Once software has initialized the card and taken control of the LED through the control bit it can be used for any purpose, e.g. to indicate bus traffic etc. In its initial mode the flashing LED can be interpreted to mean that the FPGA has successfully loaded. With new implementations of VHDL it is handy to have "proof" that the Xilinx has been loaded, especially when the design is not behaving as expected.

An additional LED is provided to indicate that the 3.3V regulator is operating properly. Local regulation is provided for 3.3, 2.5,1.5, and -5V volts. The 3.3V supply has a shunt to select between the backplane 3.3 and the local regulator. The local regulator is a switching power supply which drops the 5V rail to 3.3V. The supply can source up to 10A. The -5V is needed for the ECL and is rated at 3A. The 1.5 and 2.5 are used by the Xilinx and have a lower capacity.

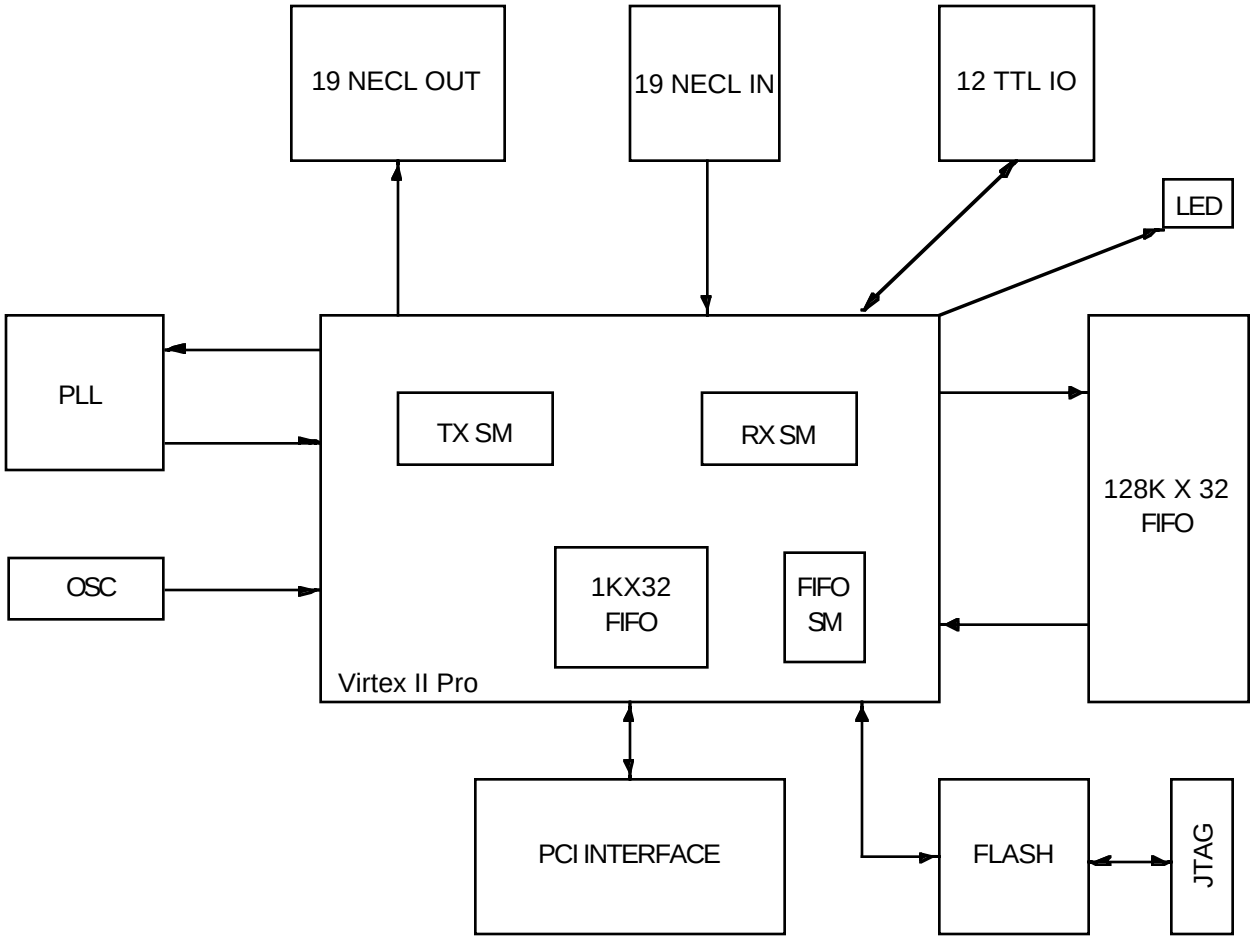


FIGURE 3 PCI-SERIAL-ECL BLOCK DIAGRAM

The PCI-Serial-ECL has both ECL and TTL IO interfaced by a D100 connector.

The TTL IO [11-0] is supported with open drain drivers with pull-ups and high-speed receivers [LVC244]. We have operated the board at 100 MHz through the TTL signals over short cables. The open drain drivers [LVTH125] have 64+ mA of sink capability.

There are 38 un-committed ECL IO. 19 devices are inputs and 19 are outputs. The inputs are terminated with 50Ω to -2V using a parallel equivalent circuit. [82/120] The ECL lines are routed as differential pairs with matched lengths and constant space. The lengths are matched from the connector edge to the Xilinx ball to allow for high-speed low skew operation.

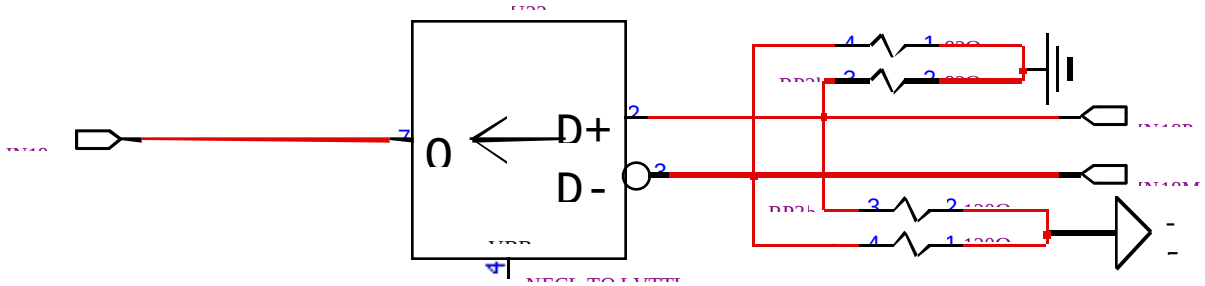


FIGURE 4 PCI-SERIAL-ECL TERMINATION

The Xilinx FPGA is re-configurable by loading a new programming file into the FLASH storage device. The file can be generated with the standard Xilinx design software. The standard Xilinx Parallel JTAG cable is connected to the on-board header to program the FLASH using the Xilinx ImPACT software. A reference file with our test configuration is also provided. The reference design has a pin configuration file, which can be reused for your specific implementation. The reference design is written in VHDL. The engineering kit also includes a cable and the HDEterm100. The HDEterm100 serves as a breakout from the cable to screw terminal block. The HDEterm100 has matched length, differential routing and several termination options that can be installed. For more information on the HDEterm100 please visit the web page http://www.dyneng.com/HDEterm100.html

Product Description Part II

A wide range of interfaces and protocols can be implemented with the PCI-Serial-ECL. UART, Manchester encoding, serial or parallel, ECL/NECL or TTL. The interfaces can be created using the hardware and development tools provided with the PCI-Serial-ECL along with the Xilinx software.

Once your requirements are known the design can be implemented with VHDL, Verilog, or schematics and compiled with the Xilinx design software. The output file can then be "uploaded" to the Xilinx FPGA [FLASH] on the PCI-Serial-ECL. 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 3 signals. The PCI-Serial-ECL has 38 ECL differential IO. There are enough IO for 1 full duplex channel with a clock reference, data and enable. The equivalent of a 16-bit parallel port would be left plus the 12 TTL IO. The serial channel would be supported with the 128Kx32 external FIFO plus any internal FIFOs that were instantiated out of block RAM. If the memory requirement per channel is small then additional channels can be supported on the one card. The FPGA is a Virtex II Pro model 4 and has plenty of additional room for more complex or additional data formatting requirements.

For systems with an external reference clock, the upper ECL input bit is received by the FPGA on a long line pin. IN18P/N can be routed through a DLL to create a low skew clock distribution based on an external reference.

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

The full bandwidth of the PCI bus is utilized during DMA transfers. There is some

overhead on the PCI bus side, which will limit the actual sustainable transfer rate somewhat compared to the theoretical limit. Looking at the other side of the equation: if we assume serial data with 1 channel operating at 170 MHz, this creates a total of 5.313 Mwords per second on the PCI bus – approximately 16% loading of the theoretical maximum.

Using the same example and looking at the external FIFO one can see that the OS can "go away" for 170Mbits/sec / 32 / 128K => 24.7 mS without over-running or under-running the FIFO. With Windows® and other high level OS based system the OS can loose track of the data movement due to other requirements - dealing with the keyboard or HDD for example. Having adequate storage can make a big difference in system performance.

Current Feature List

- User Definable Xilinx Virtex II Pro series FPGA
- DMA capable 32/33 PCI bus interface
- PLL
- 128K x 32 Data buffer
- 19 ECL Outputs
- 19 ECL Inputs
- 12 TTL IO
- 8 position "DIP Switch"
- User LED
- Power LED
- On going development with a "PROM" program

As Dynamic Engineering adds features to the hardware we will update the PCI-Serial-ECL page on the Dynamic Engineering website. If you want some of the new features, and have already purchased hardware, we will support you with a PROM update. We will reprogram the FLASH on your board for you or if you have the engineering kit and your own download cable, send you the new bit file. If you are interested please contact sales@dyneng.com for arrangements.

The basic PCI identifying information will not change with the updates. The revision field will change to allow configuration control. Current revision is 0x00.

Programming

The PCI-SERIAL-ECL is tested in a Windows® 2000 environment. We use our driver to support our test software. Please consider purchasing the engineering kit for the PCI-SERIAL-ECL; the kit has options and can include our test suite, driver and hardware support.

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

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

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

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

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

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

The internal registers for the Xilinx are defined in the following pages.

Address Map

```
PCISE_BASE_CNTL          0x00000000 //0 base control register
PCISE_INTEN              0x00000004 //1 interrupt enable register
PCISE_STATUS             0x00000008 //2 status - read
PCISE_SLAVE              0x0000000C //3 DMA FIFO read - write single word
PCISE_FIFO               0x00000010 //4 External FIFO read - write single word
PCISE_TTL_DATA_OUT       0x00000014 //5 TTL Reg read/write
PCISE_TTL_DATA_IN        0x00000018 //6 TTL IO read only
PCISE_ECL_OUT            0x0000001C //7 ECL output port for parallel data
PCISE_ECL_IN             0x00000020 //8 ECL input port for parallel data
PCISE_LOAD_TX_CNTR       0x00000024 //9 start value for transmit counter
PCISE_EXT_FIFO_CNT       0x00000028 //10 External FIFO data count - read
PCISE_DMA_FIFO_AF_LVL    0x0000002C //11 DMA FIFO almost full level read/write
PCISE_DMA_FIFO_AE_LVL    0x00000030 //12 DMA FIFO almost empty level read/write
PCISE_SWITCH             0x00000034 //13 User Switch read back port
PCISE_INTSTAT            0x0000003C //15 int stat clr – write, int stat – read
PCISE_DMAFIFO            0x00000040 //16 DMA FIFO read - write DMA access
```

FIGURE 5 PCI-SERIAL-ECL XILINX ADDRESS MAP

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

The host system will search the PCI bus to find the assets installed during power-on initialization and allocate memory and interrupt resources. The VendorId = 0x10b5 and the CardId = 0x9054 for the PCI-Serial-ECL. PCIView or other third party utilities can be useful to view your system configuration.

Once the initialization process has occurred and the system has assigned an address range to the PCI-Serial-ECL card, the software will need to determine what the address space is. We refer to this address as base0 in our software.

The next step is to initialize the PCI-Serial-ECL. The local Xilinx registers need to be configured.

Register Definitions

PCISE_BASE_CNTL

[$0000 Main Control Register Port read/write]

BASE REGISTER

DATA BIT	DESCRIPTION
31	Start_RX '1' enables RX state machine
30	Mux Sel '1' Enable TX counter data instead of FIFO data
29	Start_TX_B '1' enables transmit state machine to transmit
28	Start_TX_A '1' enables state machine to load external FIFO from internal FIFO
27-25	spare
24	Ext FIFO LD '1' enables programming almost full/empty FIFO flag levels
23	PLL En '1' = software enable to program PLL
22	PLL Data register bit
21	PLL S2
20	PLL SCLK output = PLL Command Clock
19-17	spare
16	EXT FIFO RST '1' resets external FIFO 0 = normal operation
15-14	spare
13	m_int_en - interrupt driver enable, master interrupt enable
12	force_int - cause an interrupt by enabling this bit
11-10	spare
9	led_on - '1' lights Xilinx LED, '0' turns LED off
8	led_en - '1' enabled to register control, 0 = blinking
7	en_wr_dma - enable the write FIFO state machine for DMA operation
6	en_rd_dma - enable the read FIFO state machine for DMA operation
5	en_wr_std - enable the write FIFO state machine for single word access
4	en_rd_std - enable the read FIFO state machine for single word access
3-1	spare
0	dma_rst '1' resets DMA FIFO, '0' for normal operation

FIGURE 6 PCI-SERIAL-ECL XILINX BASE CONTROL REGISTER

M_INT_EN is the master interrupt enable for all interrupts on the PCI-Serial-ECL which are controlled by the Xilinx. Please note that the PLX interrupt enable must also be enabled for a PCI interrupt to be generated. Default is disabled. When '1' the master enable is "enabled".

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

LED_ON when ‘1’ enables the Xilinx controlled LED. This bit works in conjunction with the LED_EN. When LED_EN = '0' the internal timer causes the LED to flash. When LED_EN = '1' the register bit LED_ON controls the LED. When the board is powered on and the LED is flashing the Xilinx has successfully loaded. When the software takes control the flashing will stop and a new user defined meaning attached to the LED. The user LED is located next to the 3.3V LED on the upper portion of the board.

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

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

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

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

Ext FIFO LD when ‘1’ selects the programming mode for the Programmable Almost Empty and Programmable Almost Full flag on the External FIFO. ‘0’ is normal operation.

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

The 128Kx32 part used on the PCI-Serial-ECL has a17-bit range for the PAE and PAF flags. The PAE flag is written first. The default is 127. The bit positions are LSB aligned [D16-0]. The PAF flag does not have to be written in order to program the PAE. If the PAF value is not written then it will require a reset to re-write the PAE values. If the PAF values are written then the pointer is returned to the PAE location. The PAE value can be re-written by setting the Ext FIFO LD bit and repeating the load sequence.

EXT FIFO RST when ‘1’ resets the external FIFO. Set to ‘0’ for normal operation.

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

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

The PCI-Serial-ECL has 1 PLL device that is programmed to produce the desired frequency with an I2C bus.

The data line has a pull-up on the board. When the I2C data bit is set to '0' in the register the external line is driven low. When the I2C data is set to '1' in the register the external line is pulled-up by the resistor. For a read operation the data should be set to '1' to allow the PLL to drive the line.

The upper selection bit can be set in the register and directly driven to the PLL.

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

To read over the I2C bus a command is first written and then the bus read for the response. The I2C clock and I2C data register bits contain the state of the bus when read. The software will poll the clock line and when the high to low transition is made read the data bit then repeat until the message is captured. Please note that the data bit in this register corresponds to the read/write register. The read-back data bit is located in the status register.

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. The command word generation program is downloadable from the Dynamic Engineering site.

In the base design the transmit clock is defined by the PLL output clock A. The reference to the PLL is the PCI clock doubled (66 MHz).

Start_TX_A and Start_TX_B are used to control the initialization of and transfer of data with the transmit state-machine. Start_TX_A when set causes the state-machine to start moving data from the internal "DMA FIFO" to the external data storage register. When there is sufficient data in the external FIFO Start_TX_B can be set to cause the TX state-machine to begin to send data to the Output. Depending on the system configuration different amounts of data should be stored into the external FIFO before starting the data transfer.

MUX_SEL when '1' causes the transmitter to use locally generated data instead of the stored External FIFO data for the transmission. The 32-bit counter is pre-loadable with an arbitrary offset. The counter is used to allow the receive section to be tested with the transmit section on the same card (there is only one storage FIFO).

Start_RX when set causes the RX state-machine to look for data. When data is recognized it is captured and then stored into the external FIFO using the pre-programmed algorithm.

PCISE_INTEN

[$0004 Interrupt Enable Port read/write]

INTERRUPT ENABLE REGISTER

DATA BIT	DESCRIPTION
6	inten_rx_or
5	inten_tx_dn
4	inten_ecl_in
3	inten_dma_ae
2	inten_dma_af
1	inten_pae
0	inten_paf

FIGURE 7 PCI-SERIAL-ECL INTERRUPT ENABLE PORT

Inten_paf when '1' enables the Programmable Almost Full interrupt for the external FIFO.

Inten_pae when '1' enables the Programmable Almost Empty interrupt for the external FIFO.

Inten_dma_af when '1' enables the Programmable Almost Full interrupt for the internal (DMA) FIFO.

Inten_dma_ae when '1' enables the Programmable Almost Empty interrupt for the internal (DMA) FIFO.

Inten_ecl_in when '1' enables the ECL input interrupt, which generates an interrupt when a low to high transition occurs on ECL_IN(0).

Inten_tx_dn when '1' enables the interrupt for the transmitter done condition. This occurs when the external FIFO does not have data when the transmitter is ready to read the next data word. This causes the START_TX_B bit to be cleared and the tx state-machine to halt.

Inten_rx_or when '1' enables the interrupt for the receiver over-run condition. Over-run occurs when the receiver is ready to load a data word and the external FIFO is full.

PCISE_STATUS

[$0008 Status Port read only]

STATUS REGISTER

DATA BIT	DESCRIPTION
31-25	undefined and special purpose bits
24	dma_valid
23	ext_valid
22	pll_sdat - read-back bit
20,21	spare
19	fifo_ff
18	fifo_paf
17	fifo_pae
16	fifo_mt
15	dma_ff
14	dma_paf
13	dma_pae
12	dma_mt
11 - 0	dma_fifo_count

FIGURE 8 PCI-SERIAL-ECL STATUS PORT

DMA_FIFO_COUNT indicates the number of data words in the DMA FIFO. There is always one more word available than this count indicates, since one read occurs automatically as soon as data is written to the FIFO. This data word is held in the output register until requested.

DMA_MT when '1' indicates that the DMA FIFO is empty.

DMA_PAE when '1' indicates that the DMA FIFO is almost empty as determined by the almost empty register value.

DMA_PAF when '1' indicates that the DMA FIFO is almost full as determined by the almost full register value.

DMA_FF when '1' indicates that the DMA FIFO is full.

FIFO_MT when '1' indicates that the external FIFO is empty.

FIFO PAE = External FIFO Programmable Almost Empty flag. When '1' the FIFO is almost empty.
FIFO PAF = External FIFO Programmable Almost Full flag. When '1' the external FIFO is almost full.

The programmable FIFO levels can be changed from the default values using the procedure described in the Base control register description.

FIFO_FF when '1' indicates that the external FIFO is full.

PLL_SDAT is the read-back bit to get the state of the data line connected to the PLL with the I2C bus. When reading data from the PLL this bit must be used.

EXT_VALID when '1' indicates that there is valid data available from the external FIFO. This bit may be set even when the FIFO is empty since there are two data words held in the FIFO output registers.

DMA_VALID when '1' indicates that there is valid data available from the DMA FIFO. This bit may be set even when the FIFO is empty since there is one data word held in the FIFO output register.

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

PCISE_DMA_FIFO

[$000C, 0x0040 DMA FIFO Port]

DMA FIFO PORT	
DATA BIT	DESCRIPTION
31..0	DMA FIFO data 31..0

FIGURE 9 PCI-SERIAL-ECL DMA FIFO PORT

The DMA FIFO can be written to or read from via the PCI bus. The DMA FIFO can be accessed with "standard" [0x000C] target accesses or via DMA[0x0040]. The base control register must be properly programmed before accessing this port.

PCISE_EXT_FIFO

[0x0010 External FIFO Port]

EXTERNAL FIFO PORT	
DATA BIT	DESCRIPTION
31..0	FIFO Data 31..0

FIGURE 10 PCI-SERIAL-ECL RX/TX FIFO PORT

A write to the port causes a write to the external FIFO. A read from the port causes a read from the external FIFO. The data width of the FIFO is 32 bits. The direct read and write port is used for testing and special conditions.

The port can be used for loop-back testing with DMA'd or direct writes to the External FIFO followed by reads from the External FIFO port.

PCISE_TTL

[0x14, 18 TTL Control Ports read/write] PCISE_TTL_DATA_OUT, PCISE_TTL_DATA_IN

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

FIGURE 11 PCI-SERIAL-ECL TTL CONTROL REGISTER

The TTL IO are designed with a '125 style gate and read-back buffer. The '125 provides an "open drain" tri-state gate. The PCI-Serial-ECL has a pull-up on each line. When the gate is enabled the corresponding line is pulled low. When the gate is disabled the line is pulled-high with the pull-up.

In order for the TTL port to be used for input, the output must be set to "FFF" to cause the IO to be in the tri-stated condition. The software can write a '1' or '0' to any bit and cause the '1' or '0' on the IO line. If an external line is driving the IO bit then the line may remain at '0' when set to the un-driven state.

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

PCISE_ECL

[Ox1C, 20 ECL Control Ports read/write] PCISE_ECL_OUT, PCISE_ECL_IN

ECL IO REGISTERS	
DATA BIT	DESCRIPTION
15-0	ECL 15 - 0

FIGURE 12 PCI-SERIAL-ECL ECL CONTROL REGISTER

The ECL IO lines not used for the serial port are organized as a parallel port. In the base design there are 16 input and 16 output lines. The outputs are driven from the register definition [PCISE_ECL_OUT]. The inputs are read from the receivers [PCISE_ECL_IN].

PCISE_TX_CNTR_LD

[Ox24 Control Port read/write]

TX COUNTER LOAD REGISTER	
DATA BIT	DESCRIPTION
31-0	TX Counter Initial Value

FIGURE 13 PCI-SERIAL-ECL TX COUNTER LOAD PORT

The serial transmit data can be derived from an internal 32-bit counter. This port allows that counter to be initialized to an arbitrary value. Thereafter each data value will be incremented by one from the previous value.

PCISE_EXT_FIFO_CNT

[Ox28 Status Port read only]

EXTERNAL FIFO WORD COUNT	
DATA BIT	DESCRIPTION
15-0	ECL 15 - 0

FIGURE 14 PCI-SERIAL-ECL EXTERNAL FIFO DATA COUNT PORT

Reading this port returns the number of data words in the External FIFO. There are always two more words available than this count indicates, since two reads occurs automatically as soon as data is written to the FIFO. These data words are held in the output registers until requested.

PCISE_DMA_AF_CNT

[Ox2C Control Port read/write]

DMA FIFO PAF REGISTER	
DATA BIT	DESCRIPTION
15-0	PAF Level 15 - 0

FIGURE 15 PCI-SERIAL-ECL DMA FIFO PAF LEVEL REGISTER

The value in this register is compared to the DMA FIFO count. If the count is greater than or equal to the value in this register, the DMA almost full status bit is set and the DMA almost full interrupt status bit is latched and may cause an interrupt if the proper enables are set.

PCISE_DMA_AE_CNT

[Ox30 Control Port read/write]

DMA FIFO PAE REGISTER	
DATA BIT	DESCRIPTION
15-0	PAE Level 15 - 0

FIGURE 16 PCI-SERIAL-ECL DMA FIFO PAE LEVEL REGISTER

The value in this register is compared to the DMA FIFO count. If the count is less than or equal to the value in this register, the DMA almost empty status bit is set and the DMA almost empty interrupt status bit is latched and may cause an interrupt if the proper enables are set.

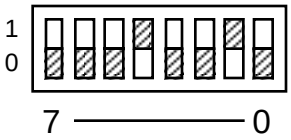
PCISE_SWITCH

[$0034 User Switch Port read only]

DIPSWITCH PORT	
DATA BIT	DESCRIPTION
7..0	Sw7..0

FIGURE 17 PCI-SERIAL-ECL USER SWITCH PORT

The user switch is read through this port. The bits are read as the lowest byte. Access the port as a long word and mask off the undefined bits. Read only. The dipswitch positions are defined in the silkscreen. For example the switch figure below indicates a Ox12.



PCISE_INTSTAT

[0x003C Interrupt Status Port]

INTERRUPT STATUS PORT

DATA BIT	DESCRIPTION
7	gnd
6	RX_OR_ERR
5	TX_DN
4	ECL_IN
3	DMA_AE
2	DMA_AF
1	EXT_PAE
0	EXT_PAF

FIGURE 18 PCI-SERIAL-ECL INTERRUPT STATUS PORT

The Programmable Almost Empty and Programmable Almost Full flags are captured on the transition to active and held until explicitly cleared. The associated interrupt would be used to keep the data flowing to the transmit port and to prevent overflow on the receive port.

ECL_IN is latched as a '1' when a low to high transition occurs on the ECL_IN(0) input data bit.

TX_DN is latched high when the transmit state-machine runs out of FIFO data to send.

RX_OR_ERR is the latched version of the over-run error condition. This occurs when the receive state machine tries to load data into a full FIFO.

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

Xilinx Pin Out

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

```
NET     "clk_osc"                LOC="E12";

NET     "clkFIFOout"             LOC="C22";
NET     "clkFIFOin"              LOC="F22";

NET     "FIN_REN"                LOC="E22";
NET     "FOUT_WEN"               LOC="D22";
NET     "FOUT_RSTN"              LOC="F18";
NET     "FOUT_W_LD"              LOC="F19";
NET     "FOUT_PAEN"              LOC="G18";
NET     "FIN_PAFN"               LOC="F21";
NET     "FIFO_MTN"               LOC="G19";
NET     "FIFO_FFN"               LOC="F20";

NET     "FDAT_IN<0>"             LOC = "U21";
NET     "FDAT_IN<1>"             LOC = "U20";
NET     "FDAT_IN<2>"             LOC = "T22";
NET     "FDAT_IN<3>"             LOC = "T21";
NET     "FDAT_IN<4>"             LOC = "T20";
NET     "FDAT_IN<5>"             LOC = "T19";
NET     "FDAT_IN<6>"             LOC = "T18";
NET     "FDAT_IN<7>"             LOC = "R22";
NET     "FDAT_IN<8>"             LOC = "R21";
NET     "FDAT_IN<9>"             LOC = "R20";
NET     "FDAT_IN<10>"            LOC = "R19";
NET     "FDAT_IN<11>"            LOC = "R18";
NET     "FDAT_IN<12>"            LOC = "P21";
NET     "FDAT_IN<13>"            LOC = "P19";
NET     "FDAT_IN<14>"            LOC = "P17";
NET     "FDAT_IN<15>"            LOC = "N21";
NET     "FDAT_IN<16>"            LOC = "N19";
NET     "FDAT_IN<17>"            LOC = "N17";
NET     "FDAT_IN<18>"            LOC = "M20";
NET     "FDAT_IN<19>"            LOC = "M18";
NET     "FDAT_IN<20>"            LOC = "L21";
NET     "FDAT_IN<21>"            LOC = "L19";
NET     "FDAT_IN<22>"            LOC = "L17";
NET     "FDAT_IN<23>"            LOC = "K21";
NET     "FDAT_IN<24>"            LOC = "K19";
NET     "FDAT_IN<25>"            LOC = "K17";
NET     "FDAT_IN<26>"            LOC = "J21";
NET     "FDAT_IN<27>"            LOC = "J19";
```

```
NET     "FDAT_IN<28>"               LOC = "J17";
NET     "FDAT_IN<29>"               LOC = "H22";
NET     "FDAT_IN<30>"               LOC = "H20";
NET     "FDAT_IN<31>"               LOC = "H18";

NET     "FDAT_OUT<0>"               LOC = "AA22";
NET     "FDAT_OUT<1>"               LOC = "Y21";
NET     "FDAT_OUT<2>"               LOC = "Y22";
NET     "FDAT_OUT<3>"               LOC = "W21";
NET     "FDAT_OUT<4>"               LOC = "W22";
NET     "FDAT_OUT<5>"               LOC = "V20";
NET     "FDAT_OUT<6>"               LOC = "V19";
NET     "FDAT_OUT<7>"               LOC = "V21";
NET     "FDAT_OUT<8>"               LOC = "V22";
NET     "FDAT_OUT<9>"               LOC = "U22";
NET     "FDAT_OUT<10>"              LOC = "P22";
NET     "FDAT_OUT<11>"              LOC = "P20";
NET     "FDAT_OUT<12>"              LOC = "P18";
NET     "FDAT_OUT<13>"              LOC = "N22";
NET     "FDAT_OUT<14>"              LOC = "N20";
NET     "FDAT_OUT<15>"              LOC = "N18";
NET     "FDAT_OUT<16>"              LOC = "M21";
NET     "FDAT_OUT<17>"              LOC = "M19";
NET     "FDAT_OUT<18>"              LOC = "M17";
NET     "FDAT_OUT<19>"              LOC = "L20";
NET     "FDAT_OUT<20>"              LOC = "L18";
NET     "FDAT_OUT<21>"              LOC = "K22";
NET     "FDAT_OUT<22>"              LOC = "K20";
NET     "FDAT_OUT<23>"              LOC = "K18";
NET     "FDAT_OUT<24>"              LOC = "J22";
NET     "FDAT_OUT<25>"              LOC = "J20";
NET     "FDAT_OUT<26>"              LOC = "J18";
NET     "FDAT_OUT<27>"              LOC = "H21";
NET     "FDAT_OUT<28>"              LOC = "H19";
NET     "FDAT_OUT<29>"              LOC = "G22";
NET     "FDAT_OUT<30>"              LOC = "G21";
NET     "FDAT_OUT<31>"              LOC = "G20";

### PLX Interface - In pin order on PLX device to help with Xilinx pin definitions
NET     "CLKIN"                     LOC = "C11"; #PLX CLOCK

NET     "ADDRESS<5>"                LOC = "AB21";
NET     "ADDRESS<4>"                LOC = "W18";
NET     "ADDRESS<3>"                LOC = "U19";
NET     "ADDRESS<2>"                LOC = "U18";
NET     "ADDRESS<1>"                LOC = "V16";
NET     "ADDRESS<0>"                LOC = "W16";

NET     "W_R"                       LOC = "Y16";
```

```
NET    "DATA_IOP<31>"          LOC = "R1";
NET    "DATA_IOP<30>"          LOC = "V15";
NET    "DATA_IOP<29>"          LOC = "W15";
NET    "DATA_IOP<28>"          LOC = "Y15";
NET    "DATA_IOP<27>"          LOC = "U14";
NET    "DATA_IOP<26>"          LOC = "V14";
NET    "DATA_IOP<25>"          LOC = "W14";
NET    "DATA_IOP<24>"          LOC = "W13";
NET    "DATA_IOP<23>"          LOC = "U13";
NET    "DATA_IOP<22>"          LOC = "V13";
NET    "DATA_IOP<21>"          LOC = "AA12";
NET    "DATA_IOP<20>"          LOC = "U12";
NET    "DATA_IOP<19>"          LOC = "V12";
NET    "DATA_IOP<18>"          LOC = "W12";
NET    "DATA_IOP<17>"          LOC = "Y12";
NET    "DATA_IOP<16>"          LOC = "Y11";
NET    "DATA_IOP<15>"          LOC = "W11";
NET    "DATA_IOP<14>"          LOC = "V11";
NET    "DATA_IOP<13>"          LOC = "U11";
NET    "DATA_IOP<12>"          LOC = "AA11";
NET    "DATA_IOP<11>"          LOC = "Y10";
NET    "DATA_IOP<10>"          LOC = "V10";
NET    "DATA_IOP<9>"           LOC = "U10";
NET    "DATA_IOP<8>"           LOC = "W10";
NET    "DATA_IOP<7>"           LOC = "W9";
NET    "DATA_IOP<6>"           LOC = "V9";
NET    "DATA_IOP<5>"           LOC = "U9";
NET    "DATA_IOP<4>"           LOC = "Y8";
NET    "DATA_IOP<3>"           LOC = "W8";
NET    "DATA_IOP<2>"           LOC = "Y7";
NET    "DATA_IOP<1>"           LOC = "W7";
NET    "DATA_IOP<0>"           LOC = "V7";

NET    "READYN"                LOC = "V6";
NET    "ADSN"                  LOC = "W6";
NET    "BLASTN"                LOC = "W5";
NET    "RSTN"                  LOC = "AB2";
NET    "LINTN"                 LOC = "AA1";

NET    "switch_in<0>"          LOC = "C10";
NET    "switch_in<1>"          LOC = "C8";
NET    "switch_in<2>"          LOC = "C7";
NET    "switch_in<3>"          LOC = "D7";
NET    "switch_in<4>"          LOC = "D6";
NET    "switch_in<5>"          LOC = "D5";
NET    "switch_in<6>"          LOC = "C2";
NET    "switch_in<7>"          LOC = "C1";
```

```
NET     "LED"                   LOC ="C21";

NET     "CLKA"                  LOC ="D11";
NET     "CLKB"                  LOC ="D12";
NET     "CLKC"                  LOC ="E11";
NET     "PLL_REF"               LOC ="B11";
NET     "PLL_SCLK"              LOC ="B12";
NET     "PLL_SDAT"              LOC ="C13";
NET     "PLL_S2"                LOC ="D13";


NET     "TTL_IN<0>"             LOC = "J1";
NET     "TTL_IN<1>"             LOC = "J2";
NET     "TTL_IN<2>"             LOC = "J3";
NET     "TTL_IN<3>"             LOC = "J4";
NET     "TTL_IN<4>"             LOC = "H1";
NET     "TTL_IN<5>"             LOC = "H2";
NET     "TTL_IN<6>"             LOC = "H3";
NET     "TTL_IN<7>"             LOC = "H4";
NET     "TTL_IN<8>"             LOC = "H5";
NET     "TTL_IN<9>"             LOC = "G3";
NET     "TTL_IN<10>"            LOC = "G4";
NET     "TTL_IN<11>"            LOC = "G5";

NET     "TTL_OUT<0>"            LOC = "G1";
NET     "TTL_OUT<1>"            LOC = "G2";
NET     "TTL_OUT<2>"            LOC = "F1";
NET     "TTL_OUT<3>"            LOC = "F2";
NET     "TTL_OUT<4>"            LOC = "F3";
NET     "TTL_OUT<5>"            LOC = "F4";
NET     "TTL_OUT<6>"            LOC = "E1";
NET     "TTL_OUT<7>"            LOC = "E2";
NET     "TTL_OUT<8>"            LOC = "E3";
NET     "TTL_OUT<9>"            LOC = "E4";
NET     "TTL_OUT<10>"           LOC = "D1";
NET     "TTL_OUT<11>"           LOC = "D2";

NET     "ECL_IN<0>"             LOC = "Y1";
NET     "ECL_IN<1>"             LOC = "Y2";
NET     "ECL_IN<2>"             LOC = "W1";
NET     "ECL_IN<3>"             LOC = "W2";
NET     "ECL_IN<4>"             LOC = "V1";
NET     "ECL_IN<5>"             LOC = "V2";
NET     "ECL_IN<6>"             LOC = "V3";
NET     "ECL_IN<7>"             LOC = "V4";
NET     "ECL_IN<8>"             LOC = "U1";
NET     "ECL_IN<9>"             LOC = "U2";
NET     "ECL_IN<10>"            LOC = "U3";
NET     "ECL_IN<11>"            LOC = "U4";
```

```
NET    "ECL_IN<12>"              LOC = "U5";
NET    "ECL_IN<13>"              LOC = "T1";
NET    "ECL_IN<14>"              LOC = "T2";
NET    "ECL_IN<15>"              LOC = "T3";
NET    "ECL_IN<16>"              LOC = "T4";
NET    "ECL_IN<17>"              LOC = "T5";
NET    "ECL_IN<18>"              LOC = "C12";

NET    "ECL_OUT<0>"              LOC = "N1";
NET    "ECL_OUT<1>"              LOC = "N2";
NET    "ECL_OUT<2>"              LOC = "N3";
NET    "ECL_OUT<3>"              LOC = "N4";
NET    "ECL_OUT<4>"              LOC = "N5";
NET    "ECL_OUT<5>"              LOC = "M2";
NET    "ECL_OUT<6>"              LOC = "M3";
NET    "ECL_OUT<7>"              LOC = "M4";
NET    "ECL_OUT<8>"              LOC = "M5";
NET    "ECL_OUT<9>"              LOC = "L2";
NET    "ECL_OUT<10>"             LOC = "L3";
NET    "ECL_OUT<11>"             LOC = "L4";
NET    "ECL_OUT<12>"             LOC = "L5";
NET    "ECL_OUT<13>"             LOC = "K1";
NET    "ECL_OUT<14>"             LOC = "K2";
NET    "ECL_OUT<15>"             LOC = "K3";
NET    "ECL_OUT<16>"             LOC = "K4";
NET    "ECL_OUT<17>"             LOC = "K5";
NET    "ECL_OUT<18>"             LOC = "J5";
```

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

Numbers in [] are member numbers – bit position or vector number.
Numbers not in [] are channel numbers in most cases.
“N” as a suffix indicates active low
The direction and voltage level are defined for each term.
FDAT_OUT = external FIFO output from Xilinx port
FDAT_IN = external FIFO input to Xilinx port
PLL = Phase Locked Loop
WEN = Write Enable
REN = Read Enable
PAF = Programmable Almost Full

PAE = Programmable Almost Empty
FF = Full Flag
MT = Empty Flag
OSC is unconnected on the standard board.
The Address, Data_IOP, ADS, BLAST, Ready, RST, W_R signals are the interface to the PLX device. Please refer to the 9054 data manual if you are not using the Engineering kit.

D100 Standard Pin Assignment

The pin assignment for the PCI-Serial-ECL P1 connector.

IN0P	IN0M	1	51
IN1P	IN1M	2	52
IN2P	IN2M	3	53
IN3P	IN3M	4	54
IN4P	IN4M	5	55
IN5P	IN5M	6	56
IN6P	IN6M	7	57
IN7P	IN7M	8	58
IN8P	IN8M	9	59
IN9P	IN9M	10	60
IN10P	IN10M	11	61
IN11P	IN11M	12	62
1N12P	IN12M	13	63
IN13P	IN13M	14	64
IN14P	IN14M	15	65
IN15P	IN15M	16	66
IN16P	IN16M	17	67
IN17P	IN17M	18	68
IN18P	IN18M	19	69
GND	GND	20	70
GND	GND	21	71
fused +5	fused +5	22	72
fused +5	fused +5	23	73
OUT0P	OUT0M	24	74
OUT1P	OUT1M	25	75
OUT2P	OUT2M	26	76
OUT3P	OUT3M	27	77
OUT4P	OUT4M	28	78
OUT5P	OUT5M	29	79
OUT6P	OUT6M	30	80
OUT7P	OUT7M	31	81
OUT8P	OUT8M	32	82
OUT9P	OUT9M	33	83
OUT10P	OUT10M	34	84
OUT11P	OUT11M	35	85
OUT12P	OUT12M	36	86
OUT13P	OUT13M	37	87
OUT14P	OUT14M	38	88
OUT15P	OUT15M	39	89
OUT16P	OUT16M	40	90
OUT17P	OUT17M	41	91
OUT18P	OUT18M	42	92
GND	GND	43	93
GND	GND	44	94
TTL_0	TTL_1	45	95
TTL_2	TTL_3	46	96
TTL_4	TTL_5	47	97
TTL_6	TTL_7	48	98
TTL_8	TTL_9	49	99
TTL_10	TTL_11	50	100

FIGURE 19 PCI-SERIAL-ECL D100 PINOUT

Note 1: fused +5 is connected to P5V via a 2.5A “self healing” fuse. If you prefer not to have power on the connector please alert the factory when placing your

order – we can leave the fuse off.
Note 2: IN0..18P/M and OUT0..18P/M refer to the ECL 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 PCI-Serial-ECL. The card is shipped in an anti-static, shielded bag. The card should remain in the bag until ready for use. When installing the card the installer must be properly grounded and the hardware should be on an anti-static work-station.

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

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

Construction and Reliability

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

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

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

Thermal Considerations

The PCI-SERIAL-ECL design consists of CMOS circuits. The power dissipation due to internal circuitry is very low. The installed IP Modules may require forced-air cooling. With the one degree differential temperature to the solder side of the board external cooling is easily accomplished.

Warranty and Repair

Dynamic Engineering warrants this product to be free from defects in workmanship and materials under normal use and service and in its original, unmodified condition, for a period of one year from the time of purchase. If the product is found to be defective within the terms of this warranty, Dynamic Engineering's sole responsibility shall be to repair, or at Dynamic Engineering's sole option to replace, the defective product. The product must be returned by the original customer, insured, and shipped prepaid to Dynamic Engineering. All replaced products become the sole property of Dynamic Engineering.

Dynamic Engineering's warranty of and liability for defective products is limited to that set forth herein. Dynamic Engineering disclaims and excludes all other product warranties and product liability, expressed or implied, including but not limited to any implied warranties of merchandisability or fitness for a particular purpose or use, liability for negligence in manufacture or shipment of product, liability for injury to persons or property, or for any incidental or consequential damages.

Dynamic Engineering's products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Dynamic Engineering.

Service Policy

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

Out of Warranty Repairs

Out of warranty repairs will be billed on a material and labor basis. The current minimum repair charge is $100. Customer approval will be obtained before repairing any item if the repair charges will exceed one half of the quantity one list price for that unit. Return transportation and insurance will be billed as part of the repair and is in addition to the minimum charge.

For Service Contact:

Customer Service Department
Dynamic Engineering
435 Park Dr.
Ben Lomond, CA 95005
831-336-8891
831-336-3840 fax
support@dyneng.com

Specifications

PCI Interfaces:	PCI Interface 33 MHz. 32 bit
Access types:	Configuration and Memory space utilized
CLK rates supported:	33 MHz. PCI, PLL with 66 MHz reference to provide programmable frequencies.
Memory	FIFO memory is provided to support DMA 1K x 32. In addition 128K x32 FIFO provided to support Xilinx data flow with TX and RX.
IO	19 ECL Transmitters. 19 ECL receivers. 12 TTL with programmable direction.
Interface:	D100 connector. [AMP] 787082-9 is the board side part number
Software Interface:	Control Registers within Xilinx.
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.
Interrupt:	1 interrupt to the PCI bus is supported with multiple sources. The interrupts are maskable and are supported with a status register.
Onboard Options:	All Options are Software Programmable
Dimensions:	half length PCI board.
Construction:	FR4 Multi-Layer Printed Circuit, Through Hole and Surface Mount Components. Programmable parts are socketed.
Power:	5V from PCI bus. Local 3.3 and 2.5, 1.8 and -5 created with on-board power supplies.
User	8 position software readable switch 1 software controllable LED's 1 Power LED

Order Information

Standard temperature range 0-70ºC

PCI-SERIAL-ECL	http://www.dyneng.com/pci_serial_ecl.html half length PCI card with user re-configurable Xilinx, 38 ECL, 12 TTL IO, 1 PLL

Extended temperature range -20 - 85ºC

PCI-SERIAL-ECL-ET	http://www.dyneng.com/pci_serial_ecl.html half length PCI card with user re-configurable Xilinx, 38 ECL, 12 TTL IO, 1 PLL
PCI-SERIAL-ECL-ENG	Engineering Kit for the PCI-SERIAL-ECL Software, Schematic, Cable and HDEterm100, reference Xilinx implementation. See webpage for more details and options including software drivers.
HDEterm100	http://www.dyneng.com/HDEterm100.html 100-pin connectors (2) matching the PCI-Serial-ECL D100 interconnected with 100 screw terminals. DIN rail mounting. Optional terminations and testpoints.
HDEcable100	http://www.dyneng.com/HDEcabl100.html 100 pin connector matching PCI-Serial-ECL and HDEterm100. Length options