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

## User Manual

# PCI-NECL-STE1

## Bidirectional NECL I/O with DMA

Revision B
Corresponding Hardware: Revision B
10-2004-0302
Corresponding Firmware: Revision C

**PCI-NECL-STE1**
PCI based Bidirectional DMA
With NECL and TTL I/O

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


Manual Revision A Revised June 25, 2010



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 to cause serious damage.

# Table of Contents

# List of Figures

## Product Description Part I

PCI-NECL-STE1 is part of the PCI Compatible family of modular I/O components. The PCI-NECL-STE1 provides a Virtex II Pro FPGA, along with 40 ECL [NECL] and 12 TTL I/O lines, a programmable PLL and FIFO support with full bidirectional 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 FIFO ports. The hardware is optimized for simultaneous bidirectional DMA access to support the high data rates available on the PCI-NECL-STE1.

The PCI-NECL-STE1 uses a PCI 9054 device from PLX Technology Inc. for the PCI interface and a Xilinx FPGA to manage the 9054 and provide transmit and receive state-machine control. The 9054 supports bi-directional scatter-gather DMA and the FPGA supports burst transfers to allow high-speed DMA data transfers.

The external FIFO is a 128K x 32-bit device that can operate at up to 66 MHz. The FPGA features Block RAM that can be configured to provide additional FIFO or other memory resources to support the I/O.

The Cypress 22393 PLL is handy for creating user specific frequencies with which to operate the state-machines and I/O. The driver supports programming the PLL over a serial I²C bus. Three clocks are received from the PLL onto FPGA long-lines see figure 1 below. The clock routing uses matched lengths to provide in-phase references should they be necessary in your design. The FPGA DCMs provide further clock functionality. The base clock tree uses the PCI clock and a DCM for low-skew on-chip distribution to generate a non-inverted and inverted 33 MHz clock, a (2x) 66 MHz clock and a (4x) 132 MHz clock. The 66 MHz is routed to the PLL for its clock reference. An unpopulated 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 device ids (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.

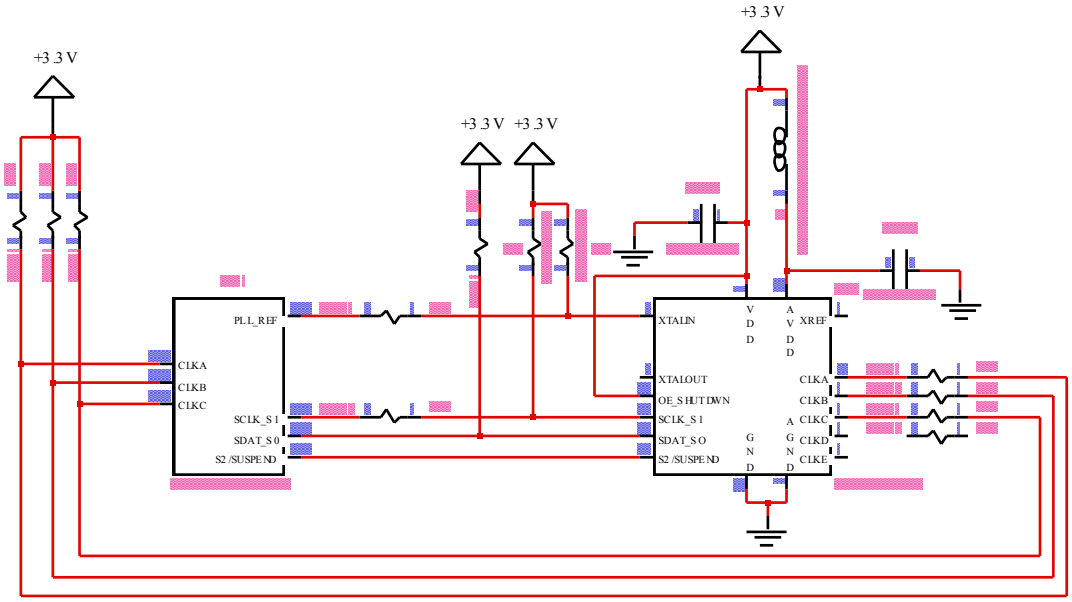



FIGURE 1 PCI-NECL-STE1 PLL

An 8-bit "dip switch" is provided on the PCI-NECL-STE1 for user-defined purposes. The switch configuration is readable via a memory-mapped I/O port. We envision the switch being used for software configuration control, PCI board identification or test purposes.

LED's are provided on the board. One LED is attached to the Xilinx via a register controlled pin. The LED initially flashes based on the PCI clock divided down to a humanly 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.

Additional LEDs are provided to indicate that the 3.3, 2.5, and -5V regulators are 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. This supply can source up to 10A. The -5V is needed for the ECL and is rated at 3A. The 1.5V and 2.5V are used by the Xilinx and have lower capacities.

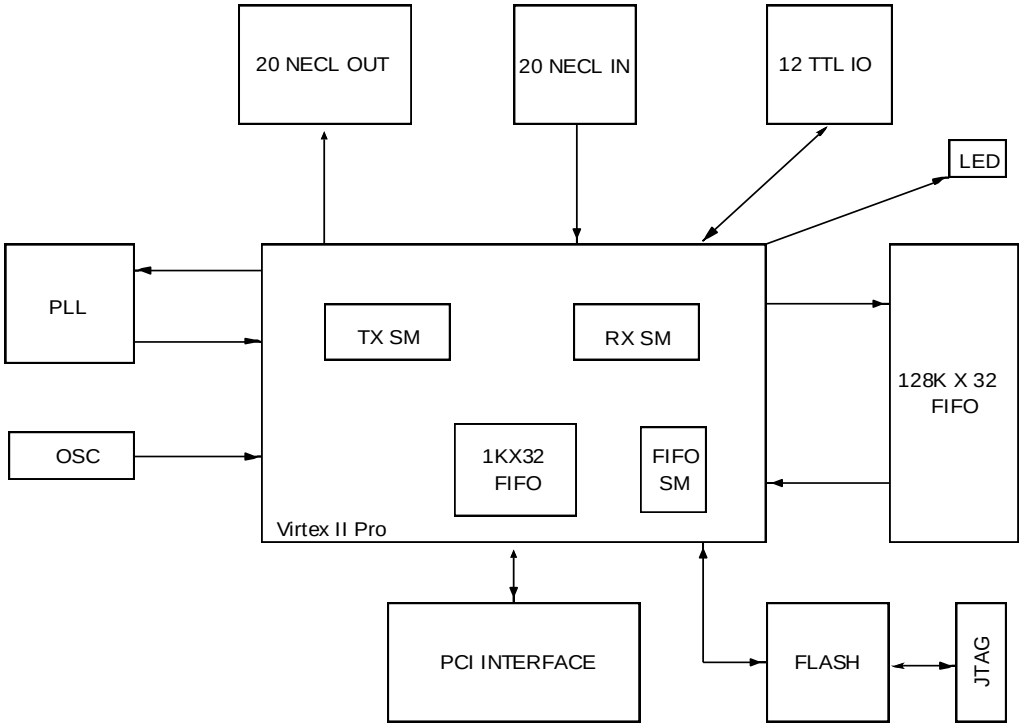



FIGURE 2 PCI-NECL-STE1 BLOCK DIAGRAM

The PCI-NECL-STE1 has both ECL and TTL I/O interfaced to the D100 connector. There is circuitry for 20 NECL inputs and 20 NECL outputs on the board as indicated in figure 2 above. However, for this design ECLIN10 and ECLOUT10 have been disconnected from the I/O connector to provide two additional control lines for the PCI9054 interface chip. These control lines are required to implement demand-mode DMA for DMA channel 0. This channel transfers received data to host memory and the additional controls are linked to the receive FIFO status so that the PCI bus is only requested when there is actually data available to transfer. Logic is also implemented to hold-off the bus request after the receive FIFO becomes nearly empty to allow time for a burst of data to accumulate. This should promote more efficient use of the PCI bus which may be critical when transferring data in both directions simultaneously.

The 12 TTL I/O lines are supported with open drain drivers with pull-ups and high-speed LVC244 receivers. We have operated the board at 100 MHz through the TTL signals over short cables. The LVTH125 open drain drivers have 64+ mA of sink capability.

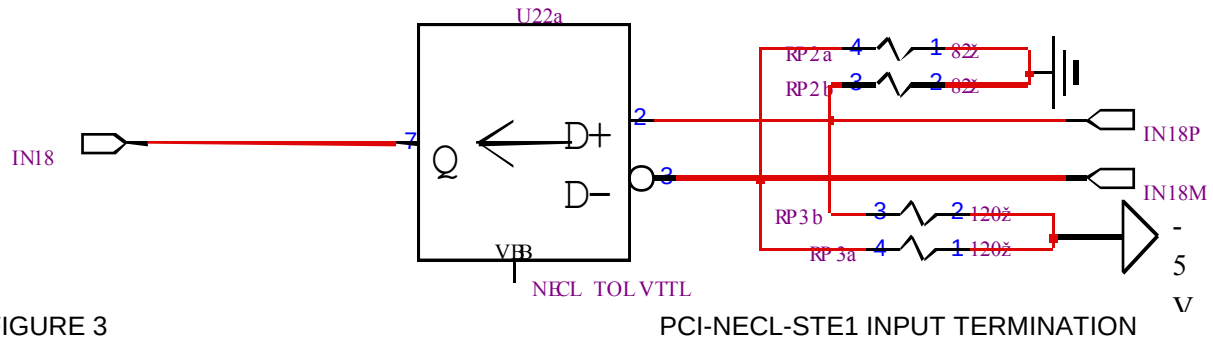



FIGURE 3 PCI-NECL-STE1 INPUT TERMINATION

The ECL inputs are terminated with 50Ω to -2V using a parallel 82Ω / 120Ω equivalent circuit as shown in figure 3 above. The outputs have optional 470Ω terminations to –5V. The ECL lines are routed as differential pairs with matched lengths and constant separation. The lengths are matched from the connector edge to the Xilinx ball to allow for high-speed low-skew operation.

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-NECL; 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-NECL 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 programmed into the Xilinx Flash PROM on the PCI-NECL. Because the Flash PROM is reprogrammable, 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 parallel interface with 16 data lines and 3 control signals. The PCI-NECL has 38 ECL differential I/O, so there is enough I/O for a full duplex implementation. The parallel channel would be supported with the 128Kx32 external FIFO plus any internal FIFOs that were instantiated out of block RAM. 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, ECL input bit 18 is received by the FPGA on a long line pin. IN18P/N can be routed through a Digital Clock Manager to create a low skew clock distribution based on an external reference.

The data flow for transmission would be Host memory transferred into the transmit FIFO via DMA transfers. The user state machine would read the data from the FIFO on the output side and apply the user protocol before 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 receive FIFO. The data would be read from the receive FIFO by the Xilinx FIFO control state-machine and be transferred into the host memory via DMA transfers.

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 parallel data with 1 channel operating at 35 Mbytes/second, this creates a total of 70 Mbytes/second on the PCI bus – approximately 53% 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 128K words x 4 bytes/word / 70Mbytes/sec => 7.3 mS without receive data over-running the FIFO when used for receive or under-running the FIFO when

used for transmit. With Windows® and other high level OS based system the OS can neglect 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
- Bidirectional DMA capable 32/33 PCI bus interface
- PLL
- 128K x 32 Data buffer
- 19 ECL Outputs
- 19 ECL Inputs
- 12 TTL I/O
- 8 position "DIP Switch"
- User LED
- Power LEDs
- On-going development with a "PROM" program

As Dynamic Engineering adds features to the hardware we will update the PCI-NECL-STE1 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 0x02.

## Programming

The PCI-NECL-STE1 is tested in a Windows® XP environment. We use our driver to support our test software. Please consider purchasing the engineering kit for the PCI-NECL-STE1; 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 as well.

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

Writing to the Bus Region Descriptors 0x18 (LBRD0) with 0x10430343 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 PCI9054 HW design manual.

Writing 0x01230880 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. 0x00000D00 will enable the Master abort interrupt and the local bus interrupt and enable the 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

| | | |
|---|---|---|
| PCIECL_BASE_CONTROL | 0x00000000 | // 0 Base control register |
| PCIECL_STATUS | 0x00000004 | // 1 Status register |
| PCIECL_FIFO | 0x0000000C | // 3 RX FIFO read/TX FIFO write single word |
| PCIECL_TTL_DATA_OUT | 0x00000014 | // 5 TTL register read/write |
| PCIECL_TTL_DATA_IN | 0x00000018 | // 6 TTL I/O read only |
| PCIECL_ECL_DATA_OUT | 0x0000001C | // 7 ECL register read/write to control auxiliary ECL I/O |
| PCIECL_ECL_DATA_IN | 0x00000020 | // 8 ECL I/O read only (Connected to ECL_B (0-7) |
| PCIECL_TX_FIFO_CNT | 0x00000024 | // 9 TX FIFO data count - read |
| PCIECL_RX_FIFO_CNT | 0x00000028 | // 10 RX FIFO data count - read |
| PCIECL_INT_AMT_LVL | 0x0000002C | // 11 Internal FIFO almost empty level read/write |
| PCIECL_EXT_FIFO_LVLS | 0x00000030 | // 12 External FIFO almost empty/full level read/write |
| PCIECL_SWITCH | 0x00000034 | // 13 User Switch and Xilinx revision read back port |
| PCIECL_INT_AFL_LVL | 0x00000038 | // 14 Internal FIFO almost full level read/write |
| PCIECL_FIFO_DMA | 0x00000040 | // 16 RX FIFO read/TX FIFO write DMA access |

FIGURE 4 PCI-NECL-STE1 XILINX ADDRESS MAP

The address map provided is for the local decoding performed within PCI-NECL-STE1 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-NECL-STE1 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-NECL-STE1. 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-NECL-STE1 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-NECL-STE1. The local Xilinx registers need to be configured.

## Register Definitions

### PCIECL_BASE_CNTL

[0x0000 Main Control Register Port read/write]

**Base Control Register**

| Data Bit | Description |
|---|---|
| 31-24 | Spare |
| 23 | PLL Enable |
| 22 | PLL Sdata Output |
| 21 | PLL S2 Output |
| 20 | PLL Sclk Output |
| 19-17 | Spare |
| 16 | LED On |
| 15 | LED Control Enable |
| 14 | TX Start/Stop Enable (not latched) |
| 13 | TX Start |
| 12 | RX Start |
| 11 | RX Almost Full Interrupt Enable |
| 10 | TX Almost Empty Interrupt Enable |
| 9 | RX FIFO Overrun Interrupt Enable |
| 8 | TX Done Interrupt Enable |
| 7 | TX Start Bit Clear Enable |
| 6 | TX Use External FIFO |
| 5 | Force Interrupt |
| 4 | Master Interrupt Enable |
| 3 | FIFO Loop-Back Enable |
| 2 | Receive FIFO Programmable Level Load |
| 1 | Receive FIFO Reset |
| 0 | Transmit FIFO Reset |

FIGURE 5 PCI-NECL-STE1 XILINX BASE CONTROL REGISTER

Transmit/Receive FIFO Reset: When one or both of these bits are set to a one, the corresponding data FIFO, packet length FIFO and control and status circuitry will be reset. When these bits are zero, normal FIFO operation is enabled. FIFO resets are referenced to the PCI clock; two periods are required for proper reset.

FIFO Loop-Back Enable: When this bit is set to a one, any data written to the transmit FIFO will be immediately transferred to the receive FIFO. This allows for fully testing the data FIFOs without an I/O loopback connection. When this bit is zero, normal operation is enabled.

Receive FIFO Programmable Level Load: The load bit must be active during FIFO reset to select the programmable level feature. Once selected, this bit must be set to zero for normal FIFO operation. When set to one, data accesses are instead directed to the almost empty and almost full level registers (See FIFO data sheet for details).

Master Interrupt Enable: When this bit is set to a one all enabled interrupts (except the DMA interrupts) will be gated through to the PCI host. When this bit is a zero, the interrupts can be used for status without interrupting the host.

Force Interrupt: When this bit is set to a one a system interrupt will occur provided the master interrupt enable is set. This is useful to test interrupt service routines.

TX Use External FIFO: When this bit is set to a one, the external 128K by 32-bit FIFO will be used to store data to be transmitted and the internal 2K by 32-bit FIFO will be used to store the received data. When this bit is zero, the internal 2K by 32-bit FIFO will be used to store data to be transmitted and the external 128K by 32-bit FIFO will be used to store the received data.

TX Start Bit Clear Enable: When this bit is set to a one, the TX start register bit will be automatically cleared by the TX done signal. When this bit is zero, the start bit must be explicitly cleared by an additional write to this register.

TX Done Interrupt Enable: When this bit is set to a one, an interrupt will be generated when the transmitter is done sending data, provided the master interrupt enable is asserted. This will occur when the TX FIFO runs out of data. When this bit is zero, an interrupt will not be generated, but the TX done status can still be read from the status register.

RX Overrun Interrupt Enable: When this bit is set to a one, an interrupt will be generated when the receiver attempts to write to a full FIFO, provided the master interrupt enable is asserted. When this bit is zero, an interrupt will not be generated, but the RX overrun status can still be read from the status register.

TX Almost Empty Interrupt Enable: When this bit is set to a one, an interrupt will be generated when the transmit FIFO level becomes equal or less than the value specified in the PCIECL_TX_FIFO_AMT_LVL register, provided the master interrupt enable is asserted. When this bit is zero, an interrupt will not be generated, but the status can still be read from the status register.

RX Almost Full Interrupt Enable: When this bit is set to a one, an interrupt will be generated when the receive FIFO level becomes equal or greater to the value specified in the PCIECL_RX_FIFO_AFL_LVL register, provided the master interrupt enable is asserted. When this bit is zero, an interrupt will not be generated, but the status can still be read from the status register.

RX Start: When this bit is set to a one, the receive state-machine is enabled to start looking for data. When data is detected it is captured, assembled into 32-bit words and stored in the RX FIFO. When this bit is zero, the receiver is disabled.

TX Start: When this bit is set to a one, the transmitter is enabled to send data provided the TX FIFO is not empty. When this bit is zero, the transmitter is disabled. The TX Start/Stop Enable bit described below must be set to a one along with this bit for the desired value to be set.

TX Start/Stop Enable: When a one is written to this bit, the value simultaneously written to the TX Start bit will be latched. If a zero is written to this bit, the TX Start bit will retain its current value regardless of the level written to it. This bit is not latched and must be explicitly written high each time the value of the TX Start bit is changed. This prevents inadvertent changes to the transmitter start/stop state from rewriting stored configuration values after the start bit has been automatically cleared.

LED Control Enable: When this bit is set to a one, the user LED will be controlled by the LED On bit in this register. When this bit is zero, the LED is controlled by a counter that flashes the LED approximately twice per second.

LED On: When this bit is set to a one and the LED Control Enable bit is also a one, the user LED will be lit. When this bit is zero and the LED Control Enable is a one, the user LED will be off.

PLL Sclk/Sdata Output: These signals are used to program the PLL over the I2C serial interface. Sclk is always an output whereas sdata is bi-directional. This is where the output value is specified. When sdata is an input it is read from the status register. Please note that the reference for the PLL is the PCI clock X 4 (132 MHz).

PLL S2 Output: This is an additional control line to the PLL that can be used to select additional pre-programmed frequencies.

PLL Enable: When this bit is set to a one, the signals used to program and read the PLL are enabled.

## PCIECL_STATUS

[0x0004 Status Port read/write to clear]

**Status Register**

| Data Bit | Description |
|---|---|
| 31-23 | Spare |
| 22 | PLL Sdata Input |
| 21-16 | Spare |
| 15 | Interrupt Active |
| 14-12 | Spare |
| 11 | Receive FIFO Almost Full Latch |
| 10 | Transmit FIFO Almost Empty Latch |
| 9 | Receiver Overrun Latch |
| 8 | Transmitter Done Latch |
| 7 | Receive Data Valid |
| 6 | Receive FIFO Full |
| 5 | Receive FIFO Almost Full |
| 4 | Receive FIFO Empty |
| 3 | Transmit Data Valid |
| 2 | Transmit FIFO Full |
| 1 | Transmit FIFO Almost Empty |
| 0 | Transmit FIFO Empty |

FIGURE 6 PCI-NECL-STE1 STATUS PORT

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 PCIECL_TX_FIFO_AMT_LVL register; when a zero is read, the level is greater 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.

Transmit Data Valid: When a one is read, there is at least one word of valid transmit data. When either the transmitter or the FIFO bypass is enabled, the first word written to the transmit FIFO will be read to be available to the transmitter or the FIFO bypass circuit. Therefore although the FIFO is empty, if this bit is set, there is one additional long-word of transmit data. A zero indicates that there is no valid transmit data.

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 PCIECL_RX_FIFO_AFL register; when a zero is read, the 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.

Receive Data Valid: When a one is read, there is at least one word of valid receive data. When data is written to the receive FIFO, the first word is read to be ready for a PCI read DMA or single-word read. Therefore although the FIFO is empty, if this bit is set, there is one additional long-word of receive data. A zero indicates that there is no valid receive data.

Transmitter Done Latch: When a one is read, it indicates that the transmitter has been started and then completed sending data. This will occur when the transmit FIFO contains no more data when the transmitter requests it. A zero indicates that the transmitter has either not been started or, if it has been started has not yet completed sending data. This bit is latched and can be cleared by writing to the Status register with a one in this bit position.

Receiver Overrun Latch: When a one is read, it indicates that an attempt has been made by the receive state-machine to write receive data to the receive FIFO when the FIFO is full. A zero indicates that a receiver overrun condition has not occurred. This bit is latched and can be cleared by writing to the Status register with a one in this bit position.

Transmit FIFO Almost Empty Latch: When a one is read, it indicates that the transmit FIFO data count has become less than or equal to the value in the PCIECL_TX_AMT_CNT register. A zero indicates that the FIFO has not become almost empty. This bit is latched and can be cleared by writing to the Status register with a one in this bit position.

Receive FIFO Almost Full Latch: When a one is read, it indicates that the receive FIFO data count has become greater than or equal to the value in the PCIECL_RX_AFL_CNT register. A zero indicates that the FIFO has not become almost full. This bit is latched and can be cleared by writing to the Status register with a one in this bit position.

Latched Receive FIFO Almost Full: When a one is read, it indicates that the receive FIFO data count has become greater than or equal to the value in the SPWR_CHAN_RX_AFL register. A zero indicates that the FIFO has not become almost full. This bit is latched and can be cleared by writing to the Status register with a one in this bit position.

Latched Transmit FIFO Almost Empty: When a one is read, it indicates that the transmit FIFO data count has become less than or equal to the value in the SPWR_CHAN_TX_AMT register. A zero indicates that the FIFO has not become almost empty. This bit is latched and can be cleared by writing to the Status register with a one in this bit position.

Interrupt Active: When a one is read, it indicates that an enabled interrupt condition is active. If the master interrupt enable bit is a one, this will result in a PCI interrupt being asserted. A zero indicates that no enabled interrupt condition is active.

PLL Sdata Input: The PLL Sdata bi-directional line is read using this bit. This line is used to read the register contents of the PLL.

### PCIECL_TX_FIFO

[0x000C, 0x0040 TX FIFO Port write only]

| TX FIFO Port | |
|---|---|
| **DATA BIT** | **DESCRIPTION** |
| 31-0 | FIFO data 31..0 |

FIGURE 7 PCI-NECL-STE1 TX FIFO PORT

The transmit FIFO can be written to from the PCI bus using single-word writes to address 0x0C or by an input DMA. Data is read from the transmit FIFO by either the FIFO bypass (used for FIFO testing) or the transmit state-machine.

### PCIECL_RX_FIFO

[0x000C, 0x0040 RX FIFO Port read only]

| RX FIFO Port | |
|---|---|
| **DATA BIT** | **DESCRIPTION** |
| 31-0 | FIFO Data 31..0 |

FIGURE 8 PCI-NECL-STE1 RX FIFO PORT

The receive FIFO can be read from the PCI bus using single-word reads from address 0x0C or using output DMA. Data is written to the receive FIFO by either the FIFO bypass (used for FIFO testing) or the receive state-machine.

## PCIECL_TTL

[0x0014, 0x0018 TTL Ports] PCIECL_TTL_DATA_OUT (read/write), PCIECL_TTL_DATA_IN (read only)

**TTL I/O Registers**

| DATA BIT | DESCRIPTION |
|---|---|
| 11-0 | TTL 11..0 |

FIGURE 9 PCI-NECL-STE1 TTL CONTROL REGISTERS

The TTL I/O are designed with a ‘125 style gate and read-back buffer. The ‘125 provides an “open drain” tri-state gate. The PCI-NECL-STE1 has a pull-up on each line. When the gate is set low (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 I/O to be in the tri-stated condition. The software can write a ‘1’ or ‘0’ to any bit and cause the ‘1’ or ‘0’ to be seen on the I/O line. If an external line is driving the I/O bit then the line may remain at ‘0’ when set to the un-driven state.

The PCINECL_TTL_DATA_IN read only port (0x0018) returns the state of the I/O lines – not necessarily the same as the write port.

## PCIECL_ECL

[0x001C, 0x0020 ECL Ports] PCIECL_ECL_DATA_OUT (read/write), PCIECL_ECL_DATA_IN (read only)

**ECL I/O Registers**

| DATA BIT | DESCRIPTION |
|---|---|
| 7-0 | ECL_OUTB 7-0 |

FIGURE 10 PCI-NECL-STE1 ECL CONTROL REGISTER

There are eight unused ECL I/O lines on the PCI-NECL-STE1. These can be used for any purpose and are controlled by the above register. A read from address 0x001C just returns the contents of the output data register. A read from address 0x0020 will return the data on the external data bus.

## PCIECL_TX_FIFO_CNT

[0x0024 Status Port read only]

| TX FIFO Word Count | |
|---|---|
| **DATA BIT** | **DESCRIPTION** |
| 31-0 | Transmit FIFO word count |

FIGURE 11 PCI-NECL-STE1 TX FIFO DATA COUNT PORT

Reading this port returns the number of data words in the transmit FIFO. If the FIFO loop-back or transmit state machine are enabled, there can be an additional data word available. This is indicated by the transmit data valid status bit. If the external FIFO is mapped to the transmitter (Base Control bit 6 = '1'), this count can be as much as 0x20000 (128 K) 32-bit words. If the internal FIFO is mapped to the transmitter (Base Control bit 6 = '0'), this count can only be a maximum of 0x800 (2 K) 32-bit words.

## PCIECL_RX_FIFO_CNT

[0x0028 Status Port read only]

| RX FIFO Word Count | |
|---|---|
| **DATA BIT** | **DESCRIPTION** |
| 31-0 | Receive FIFO word count |

FIGURE 12 PCI-NECL-STE1 RX FIFO DATA COUNT PORT

Reading this port returns the number of data words in the receive FIFO. There is always one more words available than this count indicates, since a read occurs automatically as soon as data is written to the FIFO. This data word is held in the output register until requested by a read DMA or single-word read. If the external FIFO is mapped to the receiver (Base Control bit 6 = '0'), this count can be as much as 0x20000 (128 K) 32-bit words. If the internal FIFO is mapped to the receiver (Base Control bit 6 = '1'), this count can only be a maximum of 0x800 (2 K) 32-bit words.

## PCIECL_INT_AMT_LVL

[0x002C Control Port read/write]

| Internal FIFO Almost Empty Register | |
|---|---|
| **DATA BIT** | **DESCRIPTION** |
| 15-0 | Transmit FIFO almost empty level register (only bits 0-11 significant) |

FIGURE 13 PCI-NECL-STE1 INT FIFO AMT LEVEL REGISTER

The value in this register is compared to the internal FIFO count. If the count is less than or equal to the value in this register, and the internal FIFO is mapped to the transmitter (Base Control bit 6 = '0'), the transmit FIFO almost empty status bit will be set. If the status transitions from not almost empty to almost empty, the transmit FIFO almost empty latch status bit will be latched and may cause an interrupt if the proper enables are set.

## PCIECL_EXT_FIFO_LVLS

[0x30 Control Port read/write]

| External FIFO Programmable Level Register | |
|---|---|
| **DATA BIT** | **DESCRIPTION** |
| 31-0 | Routed to External FIFO Data Bus (only bits 0-17 significant) |

FIGURE 14 PCI-NECL-STE1 EXT FIFO AMT/AFL LEVEL REGISTER

To set the programmable FIFO levels for the external FIFO, the programmable level mode must first be selected by setting the External FIFO Programmable Level Load bit high in the base control register while the FIFO reset sequence is performed. After doing this, the level load bit, when set to one, will redirect all subsequent data accesses to the external FIFO programmable almost empty/full internal registers.

Two points to note are: the first read/write accesses the almost empty register and the second accesses the almost full register (subsequent accesses repeat this in circular fashion); the second point is that unlike the internal FIFO, the external almost full level is the number of words from full that the almost full status is asserted not the absolute word count. The Dynamic Engineering device driver takes care of both of these issues so that setting the programmable levels for either external or internal FIFO varies only in the size of the programming fields. See the FIFO datasheet for more details if desired.

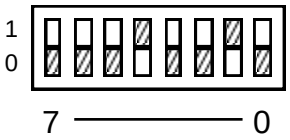
## PCIECL_SWITCH

[$0034 User Switch Port read only]

| DIP Switch Port | |
|---|---|
| **DATA BIT** | **DESCRIPTION** |
| 15..8 | Xilinx Revision |
| 7..0 | Sw7..0 |

FIGURE 15 PCI-NECL-STE1 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. The dipswitch positions are defined in the silkscreen. For example the switch figure below indicates a 0x12.




Xilinx Revision: The value of the second byte of this port is the rev. number of the Xilinx design (currently 0x03- rev. C).

## PCIECL_INT_AFL_LVL

[0x0038 Control Port read/write]

| Internal FIFO Almost Full Register | |
|---|---|
| **DATA BIT** | **DESCRIPTION** |
| 15-0 | Transmit FIFO almost empty level register (only bits 0-11 significant) |

FIGURE 16 PCI-NECL-STE1 INT FIFO AFL LEVEL REGISTER

The value in this register is compared to the internal FIFO count. If the count is greater than or equal to the value in this register, and the internal FIFO is mapped to the receiver (Base Control bit 6 = ‘1’) the receive FIFO almost full status bit will be set. If the status transitions from not almost full to almost full, the receive FIFO almost full latch status bit will be latched and may cause an interrupt if the proper enables are 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     "CLK_FIFO_OUT"          LOC = "C22";
NET     "CLK_FIFO_IN"           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     "DACK0N"                LOC = "U3";
NET     "DREQ0N"                LOC = "L3";

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>"            NC;
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_IN<19>"            LOC = "R2";

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>"           NC;
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";
NET     "ECL_OUT<19>"           LOC = "J6";
```

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, ADSN, BLASTN, READYN, RSTN, LINTN, W_R, DACK0N, and DREQ0N signals are the interface to the PLX device. Please refer to the 9054 data manual if you are not using the Engineering kit.

## Loop-Back

The Engineering kit uses the HDEterm100 with loop-back connections to provide a path to test the ECL I/O. The inputs are tied directly to the outputs.

| FROM | TO | | |
|---|---|---|---|
| OUT0P/OUT0M | IN0P/IN0M | 24/74 | 1/51 |
| OUT1P/OUT1M | IN1P/IN1M | 25/75 | 2/52 |
| OUT2P/OUT2M | IN2P/IN2M | 26/76 | 3/53 |
| OUT3P/OUT3M | IN3P/IN3M | 27/77 | 4/54 |
| OUT4P/OUT4M | IN4P/IN4M | 28/78 | 5/55 |
| OUT5P/OUT5M | IN5P/IN5M | 29/79 | 6/56 |
| OUT6P/OUT6M | IN6P/IN6M | 30/80 | 7/57 |
| OUT7P/OUT7M | IN7P/IN7M | 31/81 | 8/58 |
| OUT8P/OUT8M | IN8P/IN8M | 32/82 | 9/59 |
| OUT9P/OUT9M | IN9P/IN9M | 33/83 | 10/60 |
| | | | |
| OUT11P/OUT11M | IN11P/IN11M | 35/85 | 12/62 |
| OUT12P/OUT12M | IN12P/IN12M | 36/86 | 13/63 |
| OUT13P/OUT13M | IN13P/IN13M | 37/87 | 14/64 |
| OUT14P/OUT14M | IN14P/IN14M | 38/88 | 15/65 |
| OUT15P/OUT15M | IN15P/IN15M | 39/89 | 16/66 |
| OUT16P/OUT16M | IN16P/IN16M | 40/90 | 17/67 |
| OUT17P/OUT17M | IN17P/IN17M | 41/91 | 18/68 |
| OUT18P/OUT18M | IN18P/IN18M | 42/92 | 19/69 |
| OUT19P/OUT19M | IN19P/IN19M | 43/93 | 20/70 |
| | | | |
| 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 |

# D100 Standard Pin Assignment

The pin assignment for the PCI-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 |
| IN12P | 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 |
| IN19P | IN19M | 20 | 70 |
| GND | GND | 21 | 71 |
| GND | GND | 22 | 72 |
| GND | GND | 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 |
| OUT19P | OUT19M | 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 17 PCI-NECL-STE1 STANDARD D100 PINOUT

Note: IN0..19P/M and OUT0..19P/M refer to the ECL I/O.

# D100 -STE1 Pin Assignment

The pin assignment for the PCI-NECL-STE1 P1 connector.

| | | | |
|---|---|---|---|
| DATA_IN0P | DATA_IN0M | 1 | 51 |
| DATA_IN1P | DATA_N1M | 2 | 52 |
| DATA_IN2P | DATA_IN2M | 3 | 53 |
| DATA_IN3P | DATA_IN3M | 4 | 54 |
| DATA_IN4P | DATA_IN4M | 5 | 55 |
| DATA_IN5P | DATA_IN5M | 6 | 56 |
| DATA_IN6P | DATA_IN6M | 7 | 57 |
| DATA_IN7P | DATA_IN7M | 8 | 58 |
| ENABLE_IN_P | ENABLE_IN_M | 9 | 59 |
| CLK_IN_P | CLK_IN_M | 10 | 60 |
| N/C | N/C | 11 | 61 |
| DATA_INB0P | DATA_INB0M | 12 | 62 |
| DATA_INB1P | DATA_INB1M | 13 | 63 |
| DATA_INB2P | DATA_INB2M | 14 | 64 |
| DATA_INB3P | DATA_INB3M | 15 | 65 |
| DATA_INB4P | DATA_INB4M | 16 | 66 |
| DATA_INB5P | DATA_INB5M | 17 | 67 |
| DATA_INB6P | DATA_INB6M | 18 | 68 |
| REFCLK_IN_P | REFCLK_IN_M | 19 | 69 |
| DATA_INB7P | DATA_INB7M | 20 | 70 |
| GND | GND | 21 | 71 |
| GND | GND | 22 | 72 |
| GND | GND | 23 | 73 |
| DATA_OUT0P | DATA_OUT0M | 24 | 74 |
| DATA_OUT1P | DATA_OUT1M | 25 | 75 |
| DATA_OUT2P | DATA_OUT2M | 26 | 76 |
| DATA_OUT3P | DATA_OUT3M | 27 | 77 |
| DATA_OUT4P | DATA_OUT4M | 28 | 78 |
| DATA_OUT5P | DATA_OUT5M | 29 | 79 |
| DATA_OUT6P | DATA_OUT6M | 30 | 80 |
| DATA_OUT7P | DATA_OUT7M | 31 | 81 |
| ENABLE_OUT_P | ENABLE_OUT_M | 32 | 82 |
| CLK_OUT_P | CLK_OUT_M | 33 | 83 |
| N/C | N/C | 34 | 84 |
| DATA_OUTB0P | DATA_OUTB0M | 35 | 85 |
| DATA_OUTB1P | DATA_OUTB1M | 36 | 86 |
| DATA_OUTB2P | DATA_OUTB2M | 37 | 87 |
| DATA_OUTB3P | DATA_OUTB3M | 38 | 88 |
| DATA_OUTB4P | DATA_OUTB4M | 39 | 89 |
| DATA_OUTB5P | DATA_OUTB5M | 40 | 90 |
| DATA_OUTB6P | DATA_OUTB6M | 41 | 91 |
| REFCLK_OUTP | REFCLK_OUTM | 42 | 92 |
| DATA_OUTB7P | DATA_OUTB7M | 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 18 PCI-NECL-STE1 D100 PINOUT

# 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-NECL-STE1. 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 BI/OS 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 BI/OS, 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-NECL-STE1 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-NECL-STE1 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

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

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

## Service Policy

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

### Out of Warranty Repairs

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

## For Service Contact:

Customer Service Department
Dynamic Engineering
150 DuBois St., Suite C
Santa Cruz, CA 95060
(831) 457-8891
FAX: (831) 457-4793
support@dyneng.com

# Specifications

| | |
|---|---|
| PCI Interfaces: | PCI Interface 33 MHz. 32-bit |
| Access types: | Configuration and Memory space utilized |
| CLK rates supported: | 33 MHz. PCI, PLL with 66 MHz reference to provide programmable frequencies. |
| Memory | FIFO memory is provided 2K x 32-bit Internal FIFO and 128K x 32-bit External FIFO provided to support Xilinx data flow with TX and RX. Either FIFO can be assigned to transmit or receive. |
| I/O | 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 long-word boundaries. 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. |
| Power: | 5V from PCI bus. Local 3.3 and 2.5, 1.5 and -5 created with on-board power supplies. |
| User | 8 position software readable switch<br>1 software controllable LED's<br>1 Power LED |

# Order Information

| | |
|---|---|
| Standard temperature range 0-70ºC | |
| PCI-NECL-STE1 | http://www.dyneng.com/pci_ecl.html<br>half length PCI card with user re-configurable Xilinx, 38 ECL, 12 TTL I/O, 1 PLL |
| Extended temperature range -20 - 85ºC | |
| PCI-NECL-STE1-ET | http://www.dyneng.com/pci_ecl.html<br>half length PCI card with user re-configurable Xilinx, 38 ECL, 12 TTL I/O, 1 PLL |
| PCI-NECL-STE1-ENG | Engineering Kit for the PCI-NECL-STE1<br>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<br>100-pin connectors (2) matching the PCI-NECL-STE1 D100 interconnected with 100 screw terminals. DIN rail mounting. Optional terminations and testpoints. |
| HDEcable100 | http://www.dyneng.com/HDEcabl100.html<br>100 pin connector matching PCI-NECL-STE1 and HDEterm100. Length options |