

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*

WILDSTAR™-II Hardware Reference Manual

For all WILDSTAR™-II and WILDSTAR™-II PRO-Series PCI and VME Motherboards

12921 - 0000 Revision 6.0

© Copyright 1998-2005 by Annapolis Micro Systems, Inc. All Rights Reserved. Printed and published in the United States of America.
WILDFIRE™, WILDFIRE™-XL, WILDCHILD™, WILDFORCE™, WILDFORCE-XL™, WILD-ONE™, WILD-ONE™-XL, WILDTIME™, WILDCARD™, WILDSTAR™, WILDSTAR™-II, WILDSTAR™-II PRO, WILDSTAR™-E, WSDP™, WILDWARE™, C2WILD™, CoreFire™, WILD™ and FIREBIRD™ are trademarks of Annapolis Micro Systems, Inc. All other trademarks and registered trademarks are owned by their respective owners.



ANNAPOLIS MICRO SYSTEMS, INC. - LICENSE AGREEMENT

WILDSTAR™, WILDSTAR™-II, and WILDSTAR™-II PRO Host Software, Device Drivers, Models, VHDL, Examples, Tools and PCI Controllers are supplied with a License Agreement. This License Agreement also covers the PLD designs and Flash contents supplied with the board.

Do not install or use this product and/or break the seal on the CD-ROM until you have read and agreed to the following terms and conditions. If you agree to these terms and conditions, you should sign and return the License and Registration Certificate. You will not be entitled to support or updates until the License and Registration Certificate is received by Annapolis Micro Systems, Inc. Should you choose not to be bound by the terms and conditions of this agreement, you should promptly return this product.

YOU ARE BOUND TO THE TERMS OF THIS AGREEMENT BY BREAKING THE SEAL ON THE CD-ROM

Under the terms of this License, you:

- may make copies of the Licensed Product
- may not transfer the Licensed Product to an unlicensed party
- may modify the VHDL and the Examples
- may not modify any other parts of the Licensed Product
- may not decompile, reverse assemble or otherwise reverse engineer the Licensed Product
- may run this product ONLY on an Annapolis Micro Systems, Inc. board

The Licensed Product is owned and copyrighted by Annapolis Micro Systems, Inc. You may not remove the copyright notice from the Licensed Product. You must use your best efforts to prevent any unauthorized copying of the Licensed Product.

The Licensed Product is provided “as is” without warranty of any kind including warranties for merchantability or fitness for a particular purpose. Annapolis Micro Systems, Inc. shall not be liable for any loss of profits, loss of use, interruption of business, nor for indirect, special, incidental or consequential damages of any kind whether under this agreement or otherwise.

Although Annapolis Micro Systems, Inc. does not warrant the functions contained in the Licensed Product, the medium on which the Licensed Product is furnished is warranted to be free from defects in materials and workmanship under normal use for a period of 90 days from date of delivery to you as evidenced by a copy of your receipt. Annapolis Micro Systems’ entire liability to you and your exclusive remedy shall be replacement of the Licensed Product if the medium on which the Licensed Product is furnished proves to be defective.

You understand that the Licensed Product may require a license from the US Department of Commerce or other government agency before it may be taken or sent outside of the United States. You agree to obtain any required licenses before taking or sending the Licensed Product out of the United States. You will not permit the re-export of the Licensed Product without obtaining required licenses or letter of further assurance.

Annapolis Micro Systems, Inc. does not assume any liability arising out of the application or use of any product or circuit described herein.

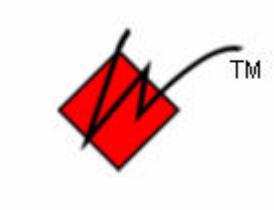

TM



TABLE OF CONTENTS

1 ABOUT THIS MANUAL 1-1

1.1 OVERVIEW 1-1

1.2 CONVENTIONS 1-3

1.3 ICONS 1-3

1.4 KEY WORDS AND DEFINITIONS 1-4

2 INTRODUCTION TO WILDSTAR™-II 2-1

2.1 WILDSTAR™-II BOARD ARCHITECTURE 2-1

2.1.1 WILDSTAR™-II PE MODULES 2-2
2.1.2 WILDSTAR™-II PRO PE MODULES 2-3
2.1.3 PE MODULE INTEROPERABILITY 2-4

2.2 WILDSTAR™-II BOARD TYPES 2-5

2.2.1 ABOUT THE WILDSTAR™-II /VME BOARD 2-5
2.2.2 ABOUT THE WILDSTAR™-II PRO /VME BOARD 2-8
2.2.3 ABOUT THE WILDSTAR™-II PRO ACE FOR VME BOARD 2-10
2.2.4 ABOUT THE WILDSTAR™-II /PCI BOARD 2-12
2.2.5 ABOUT THE WILDSTAR™-II PRO/PCI BOARD 2-16

2.3 WILDSTAR™ MAIN BOARD AND I/O CARD COMPATIBILITIES 2-19

3 GETTING STARTED 3-1

3.1 UNPACKING AND INSPECTING THE WILDSTAR™-II BOARD 3-1

3.2 BOARD ILLUSTRATIONS AND LED DEFINITIONS 3-2

3.2.1 WILDSTAR™-II / VME BOARD VIEWS 3-3
3.2.2 WILDSTAR™-II PRO/VME BOARD VIEWS 3-9
3.2.3 WILDSTAR™-II PRO ACE FOR VME (REV. B) BOARD VIEWS 3-13
3.2.4 WILDSTAR™-II /PCI BOARD VIEWS 3-17
3.2.5 WILDSTAR™-II PRO/PCI BOARD VIEWS 3-22

4 INSTALLING THE WILDSTAR™-II BOARD 4-1

4.1 WILDSTAR™-II /VME INSTALLATION 4-1

4.1.1 WILDSTAR™-II /VME SWITCH SETTINGS 4-2
4.1.2 HARDWARE INSTALLATION FOR WILDSTAR™-II /VME 4-4
4.1.3 POWER UP SYSTEM 4-4

4.2 WILDSTAR™-II PRO /VME INSTALLATION 4-5

4.2.1 WILDSTAR™-II PRO / VME JUMPER SETTINGS 4-7
4.2.2 HARDWARE INSTALLATION FOR WILDSTAR™-II PRO/ VME 4-8
4.2.3 POWER UP SYSTEM 4-9

4.3 WILDSTAR™-II PRO ACE FOR VME BOARD INSTALLATION 4-10

4.3.1 WILDSTAR™-II PRO ACE FOR VME JUMPER SETTINGS 4-10
4.3.2 HARDWARE INSTALLATION FOR WILDSTAR™-II PRO ACE FOR VME 4-12

4.4 WILDSTAR™-II /PCI BOARD INSTALLATION 4-13
4.4.1 HARDWARE INSTALLATION FOR WILDSTAR™-II /PCI 4-13
4.4.2 POWER UP SYSTEM 4-14
4.4.3 WILDSTAR™-II /PCI SWITCH SETTINGS 4-15
4.4.4 WILDSTAR™-II PRO/PCI SWITCH SETTINGS 4-17

4.5 WILDSTAR™-II RESET OPTIONS 4-18
4.5.1 PE RESETS 4-18
4.5.2 BOARD RESETS 4-18
4.5.3 SYSTEM RESETS 4-19

5 TECHNICAL SUPPORT 5-1

5.1 BOARD IDENTIFICATION NUMBERS 5-1

6 WILDSTAR™-II HARDWARE REFERENCE 6-1

6.1 WILDSTAR™-II BOARD SPECIFICATIONS 6-1

6.2 WILDSTAR™-II THERMAL MANAGEMENT 6-1

6.3 WILDSTAR™-II POWER FUNCTIONS 6-7
6.3.1 POWER MONITORING AND THRESHOLDS FOR WILDSTAR™-II / VME AND / PCI 6-7
6.3.2 WILDSTAR™-II /VME 3.3V POWER SUPPLY OPTIONS (REV. C ONLY) 6-9

6.4 WILDSTAR™-II PRO POWER FUNCTIONS 6-9

6.5 WILDSTAR™-II BOARD CLOCKS 6-11
6.5.1 CLOCK DISTRIBUTION DIAGRAMS 6-12
6.5.2 CLOCK DESCRIPTIONS AND FREQUENCIES 6-15

6.6 WILDSTAR-II PRO CLOCKS 6-18

6.7 WILDSTAR™-II BACKPLANE I/O AND POWER OPTIONS 6-18
6.7.1 WILDSTAR™-II/ VME BACKLPLANE I/O AND POWER CONFIGURATIONS 6-18
6.7.2 WILDSTAR™-II PRO /VME AND PRO ACE P2 BACKPLANE OPTIONS 6-25

6.8 WILDSTAR™-II PRO ROCKET I/O INTERFACES 6-32
6.8.1 ROCKET I/O CONNECTIONS FOR WILDSTAR™-II PRO BOARDS 6-32
6.8.2 ROCKET I/O CLOCKS 6-33

7 VHDL MODELS REFERENCE AND GUIDE 7-1

7.1 WILDSTAR™-II VHDL MODEL 7-1
7.1.1 VHDL MODEL SIMULATION ENVIRONMENT 7-1

7.2 VHDL INTERFACE COMPONENTS 7-3
7.2.1 CLOCK STANDARD INTERFACE (CLOCK_STD_IF) 7-4
7.2.2 LAD BUS STANDARD INTERFACE (LAD_BUS_STD_IF) 7-8
7.2.3 SRAM BASIC I/O (SRAMPORT_BASIC_IO) 7-15
7.2.4 SRAM DDR2 CORE (SRAM_DDR2_CORE) 7-17
7.2.5 DRAM STANDARD INTERFACE (DRAM_STD_IF) 7-20
7.2.6 DRAM BASIC I/O (DRAMPORT_BASIC_IO) 7-21
7.2.7 LED STANDARD INTERFACE (LED_STD_IF) 7-23
7.2.8 I/O CONNECTOR BASIC INTERFACE 7-24

7.2.9 PE TO PE BUS STANDARD INTERFACE (PE_BUS_STD_IF) 7-26
7.2.10 PE TO PE BUS BASIC INTERFACE (PE_BUS_BASIC_IO) 7-29

7.3 WILDSTAR™-II PRO VHDL MODEL 7-35

7.3.1 VHDL MODEL SIMULATION ENVIRONMENT 7-35

7.4 VHDL INTERFACE COMPONENTS 7-36

7.4.1 CLOCK INTERFACE (CLOCK_INTERFACE) 7-37
7.4.2 LAD BUS INTERFACE (LAD_INTERFACE) 7-39
7.4.3 SRAM DDRII INTERFACE (SRAM_DDRII_PORT*X*_INTERFACE) 7-45
7.4.4 SRAM QDRII INTERFACE (SRAM_QDRII_PORT*X*_INTERFACE) 7-47
7.4.5 DRAM INTERFACE (DRAM_INTERFACE) 7-50
7.4.6 LED INTERFACE (LED_INTERFACE) 7-52
7.4.7 PE TO PE BUS INTERFACE (PECONN_PORT*X*_INTERFACE) 7-53
7.4.8 ROCKET IO INTERFACE (RIO_PORT*X*_INTERFACE) (NOT AVAILABLE FOR VME SYSTEMS) 7-55

7.5 WILDSTAR™-II VHDL HOST MODEL 7-59

7.5.1 SIMULATED HOST API PROCEDURES 7-59
7.5.2 VHDL HOST MODEL SIMULATION ENVIRONMENT 7-71
7.5.3 USING TEMPLATES 7-71
7.5.4 USING VHDL RECORD TYPES 7-71
7.5.5 PCI CONTROLLER MODEL 7-71
7.5.6 ON-BOARD MEMORY MODEL 7-71
7.5.7 I/O CARD MODEL 7-72
7.5.8 VHDL MODEL CONFIGURATION 7-72
7.5.9 WILDSTAR™-II SYSTEM CONFIGURATION (SYSTEM_CONFIG) 7-72

8 WILDSTAR™ VHDL DESIGN CYCLE 8-1

8.1 WILDSTAR™-II /VME AND /PCI VHDL DESIGN 8-1

8.1.1 USING TEMPLATE VHDL DESIGN FILES 8-1
8.1.2 CREATING A PE DESIGN 8-1
8.1.3 CREATING A HOST DESIGN 8-1
8.1.4 CREATING A SYSTEM ARCHITECTURE 8-2
8.1.5 MODIFYING CONFIGURATION FILES 8-2
8.1.6 DESIGN SIMULATION 8-2
8.1.7 SYNTHESIZING A DESIGN 8-4
8.1.8 PLACING AND ROUTING A DESIGN 8-5
8.1.9 ANALYZING DESIGN PERFORMANCE 8-7

INDEX I

APPENDIX A: WILDSTAR™-II BACKPLANE STANDARD INTERFACE PIN MAPPING A1

APPENDIX B: WILDSTAR™-II PRO P2 BACKPLANE PIN MAPPING ... B1

APPENDIX C: WILDSTAR™-II P0/P2 PIN USAGE C1

APPENDIX D: WILDSTAR™-II VHDL SRAM EXAMPLE D1

SRAM EXAMPLE FOR WILDSTAR™-II /PCI AND /VME D1

MAIN FUNCTIONAL BLOCKS D2
SOFTWARE DESCRIPTION D3

COMMAND LINE PARAMETERS D3
PROGRAM SAMPLE OUTPUT D4

SRAM EXAMPLE FOR WILDSTAR™-II /PCI (REV. D) D5

MAIN FUNCTIONAL BLOCKS D5
SOFTWARE DESCRIPTION D6
COMMAND LINE PARAMETERS D6
PROGRAM SAMPLE OUTPUT D7

APPENDIX E: WILDSTAR™-II VHDL SDRAM EXAMPLE E1

SDRAM EXAMPLE FOR WILDSTAR™-II /PCI AND /VME E1

MAIN FUNCTIONAL BLOCKS E2
SOFTWARE DESCRIPTION E2
COMMAND LINE PARAMETERS E3
PROGRAM SAMPLE OUTPUT E3

APPENDIX F: COREFIRE™ SUPPORT F1

1 ABOUT THIS MANUAL

This *WILDSTAR™-II Reference Manual* is intended to help you install, program, and maintain your high performance WILDSTAR™-II motherboard from Annapolis Micro Systems, Inc. Both WILDSTAR™-II and WILDSTAR™-II PRO-series motherboards are discussed in this manual.

INFORMATION NOTE

Unless specifically indicated otherwise, references throughout this manual to **"WILDSTAR™-II"** typically pertain to the entire WILDSTAR™-II product line, including PRO-series boards.

All WILDSTAR™-II boards operate with WILDSTAR™-II Host Software. Detailed software information can be found in the *WILDSTAR™-II Software Reference Manual,* included in hardcopy and in a *.pdf* file on the WILDSTAR™-II documentation CD. Installation instructions for VHDL and host software are located in the accompanying release notes. A separate CD containing VHDL models and examples is also included with your board.

If you have questions or suggestions for improving the usefulness of the WILDSTAR™-II documentation, please send them to Annapolis Micro Systems, Inc. using the information provided in Chapter 5.

1.1 Overview

A brief description of each chapter appears below.

- Chapter 2, "**Introduction to WILDSTAR™-II**," discusses board architecture and performance features of WILDSTAR™-II boards.
- Chapter 3, "**Getting Started**," describes unpacking and inspection procedures for your WILDSTAR™-II, as well as board LED definitions and locations.
- Chapter 4, "**Installing WILDSTAR™-II**," describes host system requirements, installation instructions, reset options, and switch descriptions.
- Chapter 5, "**Technical Support**," provides information for contacting Annapolis Micro Systems, Inc. Technical Support.
- Chapter 6, **"WILDSTAR™-II Hardware Reference**," describes hardware specifications, along with voltage, clocking, and VME backplane options.

- Chapter 7, "**CoreFire™ Support**," provides a brief overview of the CoreFire™ Design Suite, an Annapolis Micro Systems application that allows you to quickly program and debug PEs on WILDSTAR™-II motherboards in a minimal number of steps.

- Chapter 8, "**VHDL Models Reference**," provides details of VHDL interfaces and models. It also describes simulation environments, use of WILDSTAR™-II VHDL templates, record types, and processing element (PE) standard interfaces.

- In Chapter 9, "**VHDL Design Cycle**," each phase of the VHDL design cycle is discussed, beginning with design creation and going through simulation, synthesis, place and route, and finally analysis of the design's performance.

- Appendix A lists **standard interface pin mapping locations** for the VME backplane.

- **WILDSTAR™-II PRO P0/P2 Backplane pin mapping** information appears in Appendix B

- **WILDSTAR™-II P0/P2 pin usage** information appears in Appendix C.

- Appendix D contains the **VHDL SRAM Example**.

- Appendix E contains the **VHDL SDRAM Example**.

- Appendix F contains a brief **CoreFire™ Design Suite Overview**

1.2 Conventions

Different text styles are used throughout the manual to call attention to specific items:

Convention	Description
Text represented as `screen display`	This typeface is used to represent displays appearing on the screen, such as at the "`A:\`" prompt.
Text represented as `commands`	This typeface is used to represent commands that are to be entered, such as: type "`setup.`"
Keys	When specific keys are referenced, they are designated by their labels, such as "the Enter key" or "the Escape key," or they may be shown as [Enter] or [Esc]. When two or more keys are to be pressed simultaneously, the keys are linked with a plus sign (+). For example: [Ctrl] + [Alt] + [Del].
Text represented as ***bold italics***	This typeface is used to designate equations being used in the text.

1.3 Icons

Throughout the manual, important information is highlighted with icons:

Icons	Description
	**Information Notes** call attention to important features or instructions.
	**Cautions** are directions that must be followed in order to avoid loss of system data and/or damage to hardware.
	**Warnings** are directions that must be followed to ensure personal safety.

1.4 Key Words and Definitions

Terms used throughout the manual are defined below.

ACE

WILDSTAR™-II PRO ACE for VME single-PE board

API

Application Programming Interface

Application Programming Interface

A set of functions coded in the C language allowing communication between an application and the board.

Backplane

A board within a chassis providing an interconnection point for logic and control elements among multiple VME boards.

Block RAM

An architectural feature of the Xilinx® Virtex™ II FPGA in which blocks are reserved specifically for Random Access Memory.

CLB

Configurable Logic Block (CLB). CLBs provide the functional elements for constructing logic within the PE.

CoreFire™ Design Suite

An FPGA design and debug application tool developed by Annapolis Micro Systems, Inc.

cPCI

Compact Peripheral Component Interconnect

DDR DRAM

Double Data Rate DRAM. A type of DRAM in which bursts of data are sent on both rising and falling edges of clock cycles.

DDR II SRAM

Double Data Rate SRAM. A high-performance, synchronous burst SRAM capable of single- and double-data operational modes.

Deprogram

A program is pulsed on an FPGA, which carries its configuration to be cleared and places the FPGA in a preconfiguration state.

Differential Signaling

Two-wire signaling, in which the difference in voltage between two wires is used to signal data

DLL

Dynamic Linkage Library

Driver

Software used to handle communication with one to four WILDSTAR™-II boards.

DRAM

Dynamic Random Access Memory

Euro I/O Card

Eurocard form factor I/O daughter card for WILDSTAR™/VME. Euro cards use a 228-pin plug and fit on both WILDSTAR™ and WILDSTAR™-II series boards. Euro II cards fit on WILDSTAR™-II boards only.

External I/O

External Input/Output

FPGA

Field Programmable Gate Array.

ICLK

Fixed, single-ended bus clock used for Local Address Data Bus transactions.

LAD Bus

Local Address Data Bus. Bus between the PCI controller, PE0, PE1, PE2, and the External I/O cards.

MCLK

Reprogrammable differential memory clock, asynchronous to PCLK.

ModelTech®

An application used to simulate VHDL.

N/C

No Connection.

PCI

Peripheral Component Interconnect.

PCLK

Reprogrammable Differential Processing Element Clock asynchronous to MCLK.

PE

Processing Element. A Xilinx® Virtex™ II Field Programmable Gate Array (FPGA) comprises the basic processing unit on the WILDSTAR™-II board. The Virtex-II Pro™ FPGA is used on PRO-Series boards. PEs are contained within *Processing Modules* (see definition below).

PE0

Processing Element 0

PE1

Processing Element 1

PE2

Processing Element 2

P0 VME Backplane

Connects the WILDSTAR™-II /VME motherboard to the backplane

P2 VME Backplane Connector

Connector on the WILDSTAR™-II / VME motherboard that supports backplane I/O functions.

Processing Module

A basic unit in the WILDSTAR™-II architecture consisting of a processing element (PE) and its associated ports, buses, clocks, and Flash.

QDR II SRAM

Quadruple-Data Rate II. A high-performance, synchronous burst SRAM capable of quadruple-data operational modes.

RACEway™

An ANSI-approved interface standard that provides routing and transfer protocol. RACEway™ provides interface signal connections for the VME P2 connector.

Rocket I/O™

Multi-Gigabit Transceivers (MGT) on the Virtex-II Pro™ FPGA, with transfer rates from 622 Mb/s to 3.125 Gb/s

Rx

Abbreviation of "Receive"

Single-ended Signaling

Uses one wire per signal, in which a single voltage is generated that the receiver compares with a fixed reference voltage

SRAM

Synchronous Random Access Memory

Slice

A slice contains two look-up tables and two flip-flops for implementing logic within a CLB (Configurable Logic Block). On Virtex™ II FPGAs, there are four slices per CLB.

Synplicity®

Manufacturer of Synplify ASIC®, an application used for synthesizing ASIC designs.

Tx

Abbreviation of "Transfer"

Virtex™-II FPGA

Virtex™-II Field Programmable Gate Array, used on WILDSTAR™-II motherboards.

Virtex-II Pro™ FPGA

Virtex™-II Pro™ Field Programmable Gate Array, used on WILDSTAR™-II PRO motherboards.

VME

Versa Module Eurocard.

WILDSTAR™-II Host Software (includes WILDSTAR™-II PRO)

Application Programming Interface, Device Driver, and utilities.

WILDSTAR™-II /PCI Second I/O Slot Adapter

A card designed to allow a second I/O card to be installed on the WILDSTAR™-II /PCI motherboard.

WILDSTAR™-II VHDL Models

Hardware models used for board-level VHDL simulation of application.

Xilinx®

Manufacturer of Virtex™-II FPGAs and place-and-route tools for FPGA designs.

2 INTRODUCTION TO WILDSTAR™-II

WILDSTAR™ computing is a revolutionary new technology combining reconfigurable system gates with very high memory and I/O bandwidth capacities. As an evolution of the WILDSTAR™ family of boards, WILDSTAR™-II delivers high levels of processing power, memory and flexibility while maintaining the proven reliability that distinguishes WILDSTAR™ products.

The WILDSTAR™-II board architecture supports three form factors: VME, PCI, and cPCI. WILDSTAR™-II-series boards are equipped with Virtex™-II FPGAs. WILDSTAR™-II PRO-series boards feature Virtex-II Pro™ FPGAs.



INFORMATION NOTE

Unless specifically indicated otherwise, references throughout this manual to **"WILDSTAR™-II"** pertain to the entire WILDSTAR™-II product line, including PRO-series boards.

To ensure safe and reliable processing, WILDSTAR™-II boards come equipped with a proactive thermal management system. Sensors across the board monitor power and temperature, with automatic shutdown capability to prevent excessive heat buildup. Depending on the WILDSTAR™-II board type, thermal management also includes heat sinks or fans.

WILDSTAR™-II boards support both VHDL design flow tools and the CoreFire™ Design Suite.

- The *VHDL* flow consists of four steps: 1) creation of VHDL design using supplied VHDL model; 2) simulation using the ModelTech® application; 3) synthesis using Synplicity®, and 4) place-and-routing using Xilinx® tools.

- The *CoreFire™* design flow involves two steps: creating a design using the CoreFire™ Application Builder, and then place-and-routing using Xilinx® FPGA design tools.

2.1 WILDSTAR™-II Board Architecture

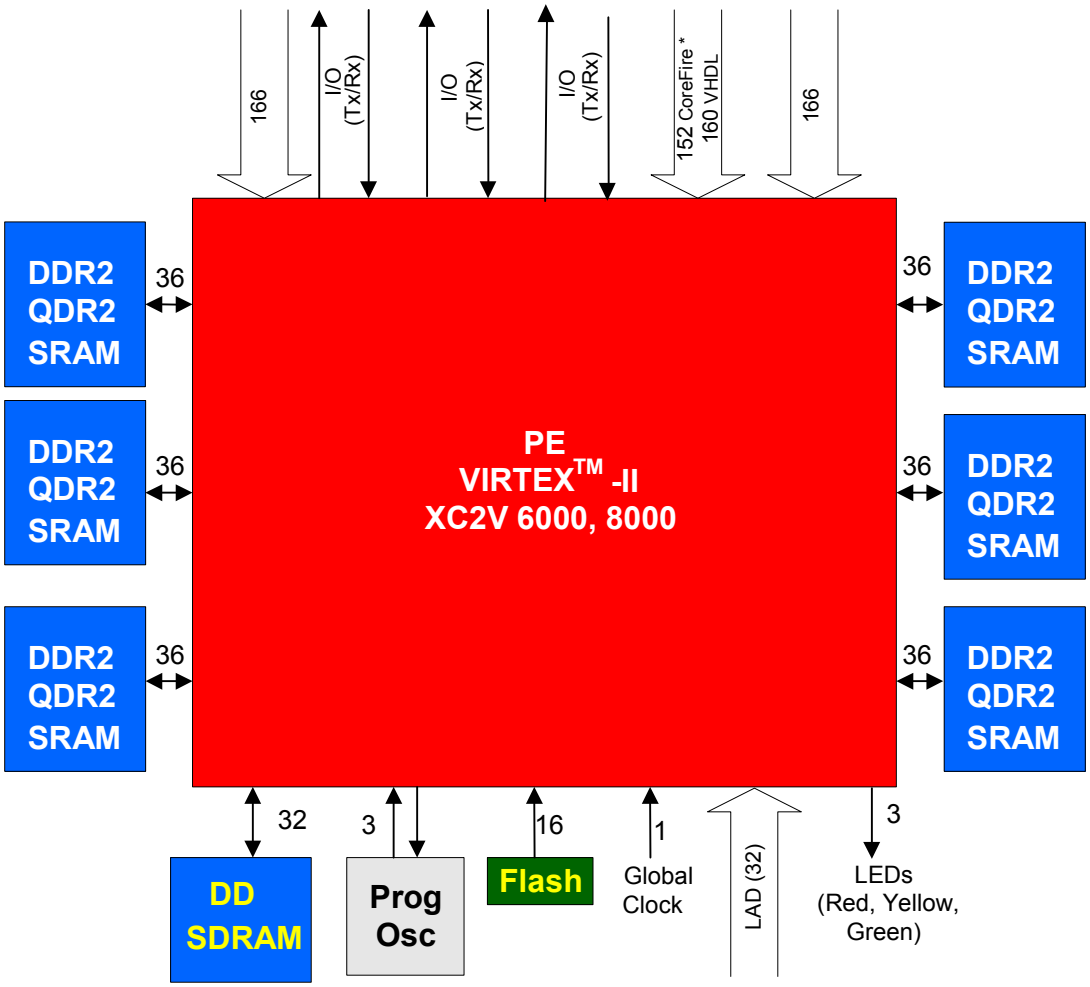
The WILDSTAR™-II architecture is best illustrated in terms of processing *modules*. A processing module is built around each of the board PEs and named accordingly (PE0, PE1, PE2).

Separate PE configuration modules for the WILDSTAR™-II and the WILDSTAR™-II PRO are described below.

2.1.1 WILDSTAR™-II PE Modules

Each module in the WILDSTAR™-II architecture (Figure 2-1) consists of the following:

- One Xilinx® Virtex™-II FPGA
- Six independent DDR2 SRAM ports
- One Bulk DDR DRAM port
- Three I/O data buses with dedicated Transmit (Tx) and Receive (Rx) clocks
- 32-bit LAD bus
- Flash storage of multiple FPGA images
- Three global clocks
- Three local clocks
- Three user LEDs



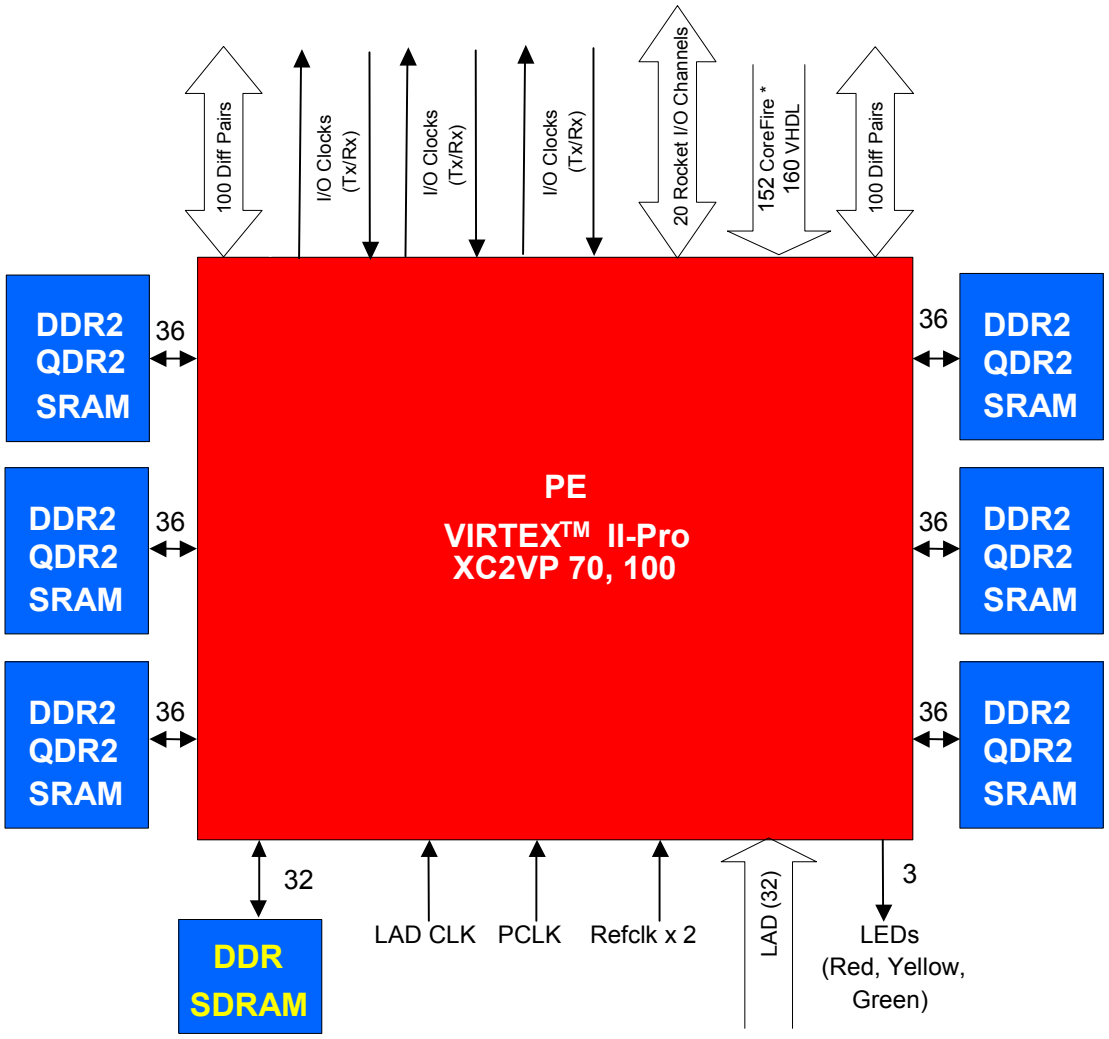
*CoreFire™ uses a small number of bits for overhead requirements.

Figure 2-1: WILDSTAR™-II Processing Module

2.1.2 WILDSTAR™-II PRO PE Modules

Each module in the WILDSTAR™-II PRO architecture (Figure 2-2) consists of the following:

- One Xilinx® Virtex-II Pro™ FPGA
- Six independent DDR2/QDR2 SRAM ports
- One Bulk DDR DRAM port
- Three I/O data buses with dedicated Transmit (Tx) and Receive (Rx) clocks
- 20 Rocket I/O Channels
- 32-bit LAD bus
- One global clock
- Three user LEDs



*CoreFire™ uses a small number of bits for overhead requirements.

Figure 2-2: WILDSTAR™-II PRO Processing Module

INFORMATION NOTE

Pin quantities specified in all diagrams are the default number as shipped to the customer.

2.1.3 PE Module Interoperability

Processing modules are architecturally compatible within the WILDSTAR™-II board family. For example, the processing module containing PE1 on a WILDSTAR™-II /VME is architecturally similar to the PE1 module on a WILDSTAR™-II /PCI. The host addresses the PEs in the architecture via the high-speed LAD Bus, allowing applications to be portable across platforms. In the same way, modules are compatible between board types in the WILDSTAR™-II PRO family.

INFORMATION NOTE

Significant design differences exist between WILDSTAR™-II and WILDSTAR™-II PRO boards. If you have specific questions about intercompatibility, please contact your Annapolis Micro Systems representative.

No bitfile compatibility exists between modules on different platform types. For an application to run on a platform different from the one on which it was originally compiled, it must be recompiled on the new platform.

I/O ports are configured as differential or single-ended, depending on where and how they are connected on a given platform. Each I/O bus has both transmit (Tx) and Receive (Rx) clocks, facilitating high-speed bus transfers.

INFORMATION NOTE

On the WILDSTAR™-II PRO boards, Virtex-II Pro™ FPGAs have built-in termination resistors for differential signaling. However, to perform differential signaling using a WILDSTAR™-II PRO board, the I/O card attached to it must also be capable of differential signaling.

Modules have flash (attached to PCI Controller) for fast, in-system programming.

"Stop light" LEDs are also incorporated into each processing module. They can be configured either as debug or indicator lights.

Depending on the number of processing modules on the board, functionality will vary. The WILDSTAR™-II PRO VME for ACE board carries only a single processing module, while the WILDSTAR™-II PRO/VME always carries three. Processing configurations specific to WILDSTAR™-II and WILDSTAR™-II PRO motherboards are described in the following sections.

2.2 WILDSTAR™-II Board Types

Features of WILDSTAR™-II /VME and /PCI boards are discussed in the sections that follow. PRO-series boards are discussed in separate sections.

INFORMATION NOTE

The WILDSTAR™-II family of boards hosts many of the special- use I/O daughter cards produced by Annapolis Micro Systems. Specific information about WILDSTAR™-II motherboards and I/O card compatibility can be found in the manual accompanying each I/O daughter card. For more information about I/O card options, contact Annapolis Micro Systems at (410) 841-2514 or by e-mail at wftech@annapmicro.com.

2.2.1 About the WILDSTAR™-II /VME Board

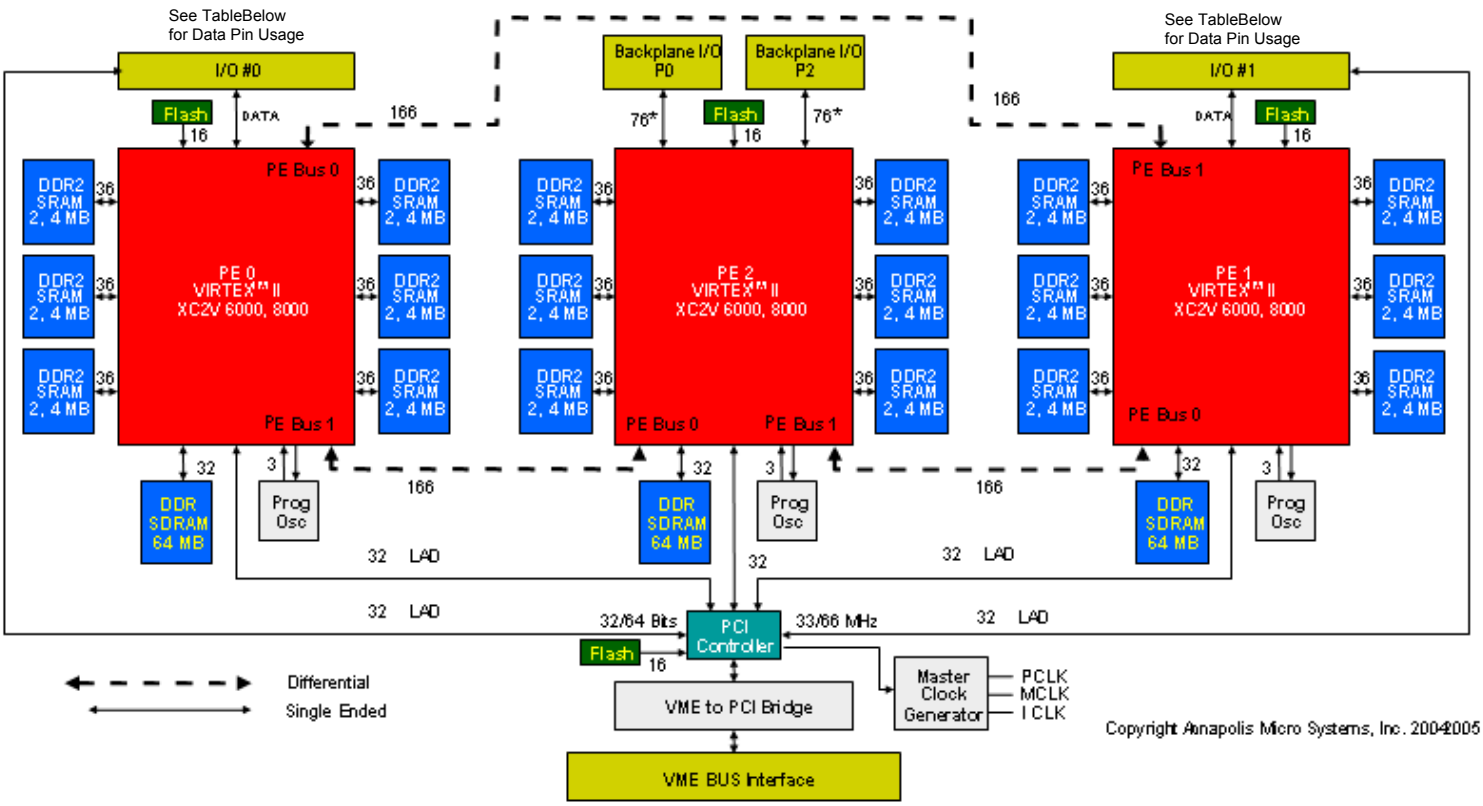
The WILDSTAR™-II /VME architecture has a number of processing configuration options (see section 2.2.1.2), with the full architecture containing three processing modules. As shown in Figure 2-3, each PE has two differential 166-pin I/O buses connected for PE-to-PE communication.

The remaining 160-pin I/O (152 pins if using CoreFire™) provides an extensive bus to I/O cards 0 and 1 from PEs 0 and 1, respectively, while the PE2 I/O bus communicates to the backplane I/O connectors P2 and P0. PEs are pin- and bit file-compatible with each other.

CAUTION

To prevent system damage, fire, and personal injury, the WILDSTAR™-II /VME and PRO /VME boards should have adequate cooling and airflow to ensure a Xilinx® junction temperature below 85˚C during application operation. Temperatures above this range may result in performance degradation and component damage.

The WILDSTAR™-II /VME is also equipped for power monitoring and thermal management. See Chapter Six for details of these functions.



*More pins are available with custom backplane configuration.

Figure 2-3: WILDSTAR™-II /VME Block Diagram

Table 2-1: Data Pin Usage for CoreFire™ and VHDL

I/O Daughter Card Type	CoreFire™	VHDL
Type I	152	154
Type II	158	160

A host computer communicates with the WILDSTAR™-II /VME board via the VME bus interface. The VME- to-PCI Bridge translates the VME cycles to PCI bus cycles in order to communicate with the WILDSTAR™-II /VME board's PCI controller. The host has direct register access and communication with PEs and I/O cards over the high-speed LAD bus, and the PCI controller has access to the PEs and up to two Euro I/O cards using independent Local Address Data (LAD) buses.

Euro I/O cards use a 228-pin plug that fits both WILDSTAR™ and WILDSTAR™-II series boards. Euro II cards fit on WILDSTAR™-II boards.

INFORMATION NOTE

Pin quantities specified in all board diagrams are the default number as shipped to the customer.

2.2.1.1 WILDSTAR™-II /VME Features

The WILDSTAR™-II /VME is equipped with the following system support features (some are only available on boards fully populated with PEs):

- Uses up to three Xilinx® Virtex™-II FPGAs with up to 24 million system gates. PEs can be populated with Virtex™ II 6000 to Virtex™ II 8000 parts
- Ability to program PEs from Flash on Power up
- Multi-Thread Safe Software Support
- Status LEDs for PEs
- Supports two I/O cards
- Approximately 4 GBytes/sec per I/O card
- DDR DRAM 800 MBytes/sec per PE for total of 2.4 GBytes/sec per board
- DDR II SRAM 960 MBytes/sec per port, for total of 17 GBytes/sec per board
- VME: 6U, Single Slot (Fully Populated)
- VME64X Bus Interface
- Local PCI Bus
- Clocks
  - PCLK: Processor Clock, independently programmable and differential
  - ICLK: LAD Clock, fixed frequency and single-ended
- Thermal Management
  - Host software can monitor PE junction temperature
  - Heat sinks on PEs
  - Board-level temperature monitoring
- Power Measurement and Control
  - Host software can monitor board current consumption and power supply voltage levels

- Individual PE power supplies and management
- PE temperature monitoring
- Board level I/V measurements
- Optional 3V power supply for use with a legacy VME chassis (available on Rev. C board only)
- Optional ability to supply power over the P0 connector (available on Rev. C boards only)

- CoreFire™ Support
- VHDL Development Environment Supported

2.2.1.2 WILDSTAR™-II / VME Processing Configurations

The functions available on WILDSTAR™-II / VME boards vary according to the processing configuration options described below. Each PE is the center of a single processing module, the contents of which are listed in section 2.1 .

Processing Configuration	Processing Modules	Functionality
"Carrier" board	None	No backplane P0/P2 access, LAD access only to I/O cards
One PE in location PE0	One	No backplane P0/P2 access, I/O card location 1 is limited to LAD access
One PE in location PE2	One	Both I/O card slots limited to LAD access, backplane P0/P2 access available
Two PEs in locations PE1 and PE0	Two	PE0 and PE1: no backplane P0/P2 access
Two PEs in locations PE2 and PE0	Two	PE0 and PE2: I/O card 1 limited to LAD access only
Three PEs	Three	Full functionality

2.2.2 About the WILDSTAR™-II PRO /VME Board

The WILDSTAR™-II PRO /VME architecture contains three processing modules. As shown in Figure 2-4, these PE modules use large buses to provide data throughput to I/O cards 0 and 1 respectively.

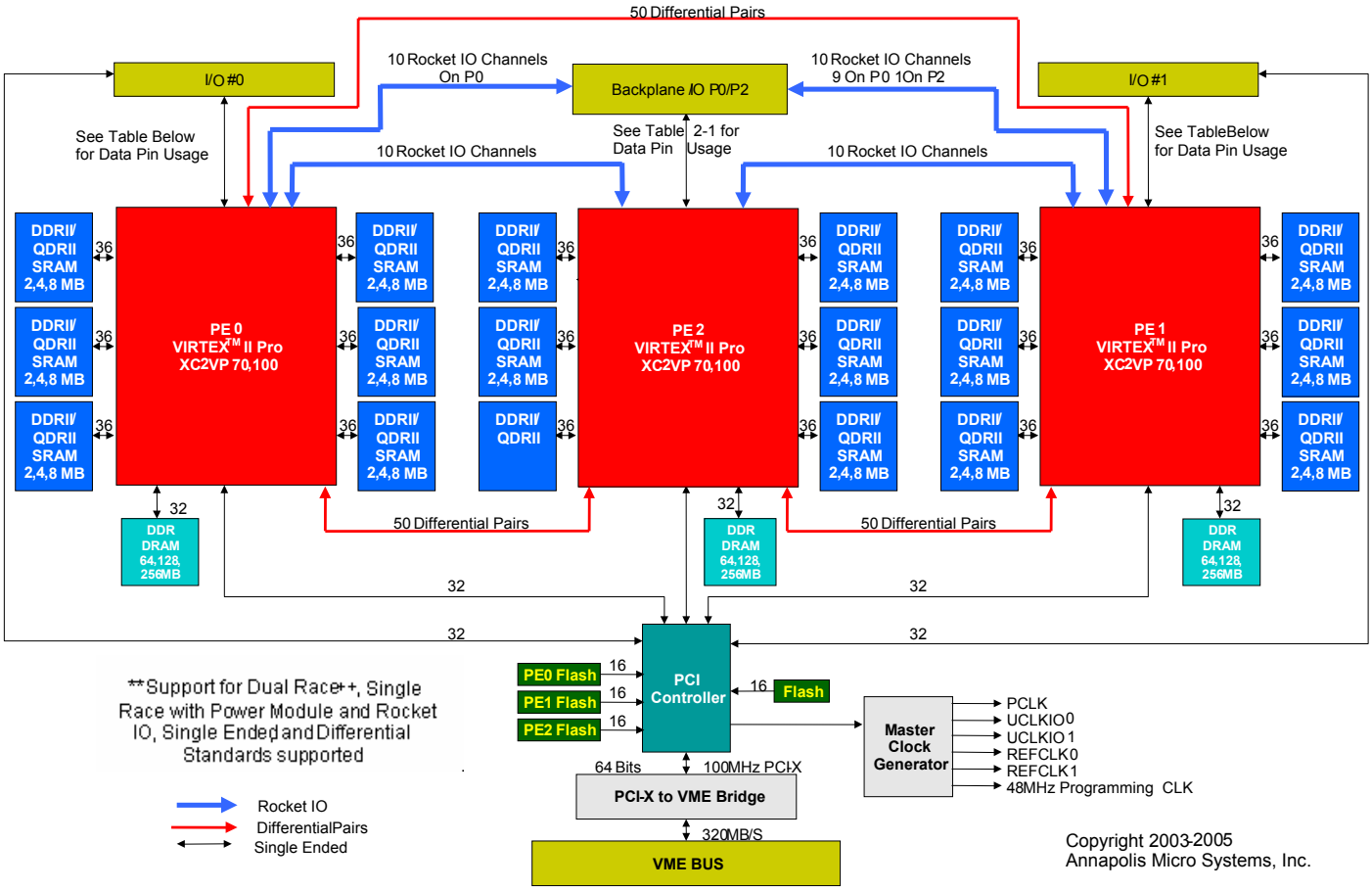


Figure 2-4: WILDSTAR™-II PRO/VME Block Diagram (Revs. A, B)

Table 2-2: Data Pin Usage for CoreFire™ and VHDL

I/O Daughter Card Type	CoreFire™	VHDL
Type I	152	154
Type II	158	160
Type III	77*	79*

* Differential Signaling (only on I/O cards populated with optional resistors)

2.2.2.1 WILDSTAR™-II PRO / VME Features

WILDSTAR™-II PRO / VME has the following system support features:

- Uses three Xilinx® Virtex-II Pro™ XC2V70 or XC2VP100 FPGAs.
- Status LEDs for PEs
- Six ports of DDR II SRAM at 960 MBytes/sec per port
- DDR DRAM 800 MBytes/sec

- Rocket I/O over backplane and PE-to-PE buses
- 2ESST (Two-edge source synchronous transfer) VME Interface
- Host software: Windows NT 4.0® and 2000®, Linux, Solaris™, VxWorks®. Includes API and device drivers
- Supports two I/O cards
- Clocks
  - PCLK: Processor clock, independently programmable and differential
  - UCLK0: User-programmable clock
  - UCLK1: User-programmable clock
  - 48 MHz Clock: Fixed 48 MHz clock
  - REFCLK0: Rocket I/O clock 0
  - REFCLK1: Rocket I/O clock 1
- Thermal Management
  - Optional heat sink combination attached to PE
  - PE temperature Monitoring
- Power Measurement and Control
  - Board level I/V measurements
- CoreFire™ Support
- VHDL Development Environment Supported

2.2.3 About the WILDSTAR™-II PRO ACE for VME Board

The WILDSTAR™-II PRO ACE for VME board, equipped with a single PE, is designed principally for rapid data transfer. These boards are capable of 2ESST VME transactions.

A host computer communicates with the WILDSTAR™-II PRO ACE for VME board via the VMEbus interface. The VME- to-PCI-X Bridge translates the VME cycles to PCI-X bus cycles in order to communicate with the WILDSTAR™-II PRO ACE board's PCI controller. The host has direct register access and communication with the PE and I/O cards over the high-speed LAD bus, and the PCI controller has access to the PE and up to two Euro I/O cards using independent Local Address Data (LAD) buses.

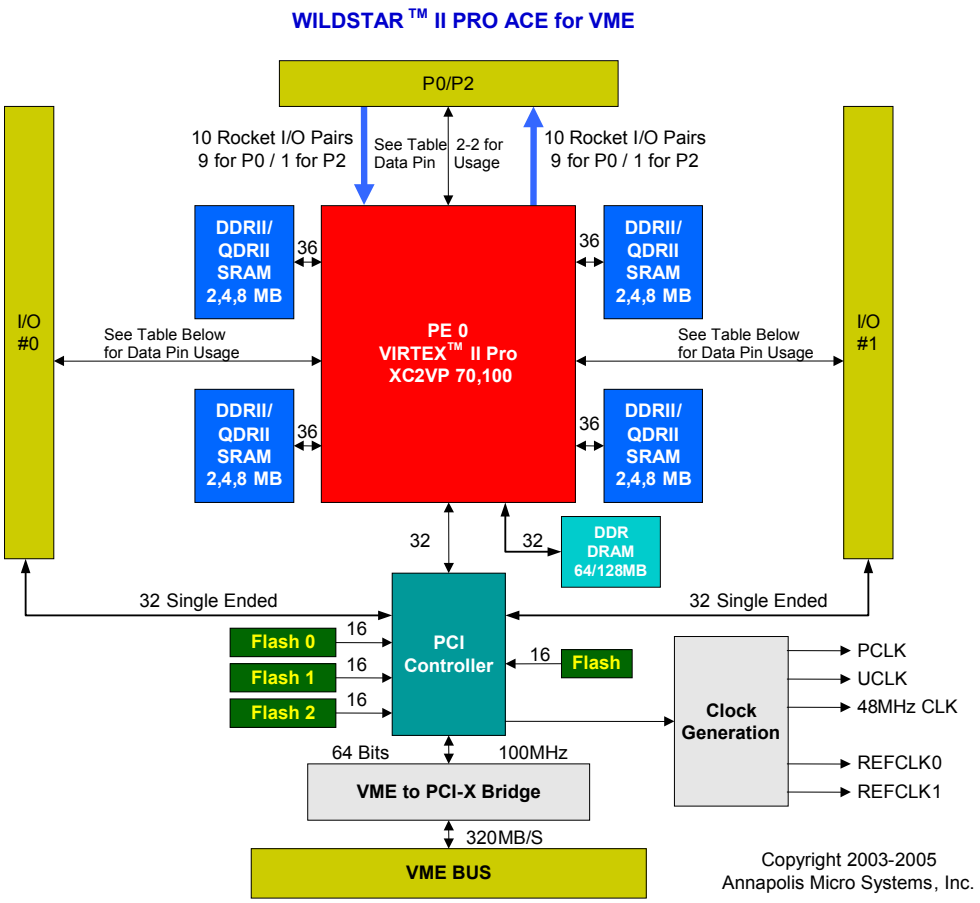


Figure 2-5: WILDSTAR™-II PRO ACE for VME Block Diagram

Table 2-3: Data Pin Usage for CoreFire™ and VHDL

I/O Daughter Card Type	CoreFire™	VHDL
Type I	152	154
Type II	158	160
Type III	77*	79*

* Differential Signaling (only on I/O cards populated with optional resistors)

As shown in the ACE block diagram, the PE0 module uses large buses to provide data throughput to I/O cards 0 and 1 respectively.

2.2.3.1 WILDSTAR™-II PRO ACE for VME Features

WILDSTAR™-II PRO ACE for VME has the following system support features:

- Uses one Xilinx® Virtex-II Pro™ XC2V70 or XC2VP100 FPGA.
- Status LEDs for PE
- Four ports of DDR II SRAM at 960 MBytes/sec per port

- DDR DRAM 800 MBytes/sec
- P2 backplane configuration options, including Dual RACE++
- Rocket I/O over P0 Connector
- 2ESST VME Interface
- Host software: Windows NT 4.0® and 2000®, Linux, Solaris™, VxWorks®. Includes API and device drivers
- Supports two I/O cards
- Clocks
  - PCLK: Processor clock, independently programmable and differential
  - UCLK: User-programmable clock (two for each I/O card used)
  - 48 MHz Clock: Fixed 48 MHz clock
  - REFCLK0: Rocket I/O clock 0
  - REFCLK1: Rocket I/O clock 1
- Thermal Management
  - PE temperature Monitoring
- Power Measurement and Control
  - Board level I/V measurements
- CoreFire™ Support
- VHDL Development Environment Supported

2.2.4 About the WILDSTAR™-II /PCI Board

The WILDSTAR™-II /PCI architecture has a number of processing configuration options (section 2.2.4.2), with the full architecture containing two processing modules. As shown in figure below, the PE0 and PE1 modules use large buses to provide data throughput to I/O cards 0 and 1 respectively.

PE0 and PE1 are pin- and bit file-compatible with each other. The PEs are connected by dual 166-pin differential data buses, providing double the data throughput as a single bus.

INFORMATION NOTE

WILDSTAR™-II /PCI and PRO /PCI board architectures are scalable to the cPCI form factor. Contact Annapolis Micro Systems, Inc. for details.

CAUTION

To prevent system damage, fire, and personal injury, *never* use the WILDSTAR™-II /PCI board without the heat sink fan installed and operational. In addition, the operational PCI chassis should have adequate cooling and airflow to ensure a T_j < 85˚C temperature during application operation. Temperatures above this range can result in performance degradation and component damage.

The WILDSTAR™-II /PCI is well-equipped for power monitoring and thermal management. See Chapter Six for details of these functions.

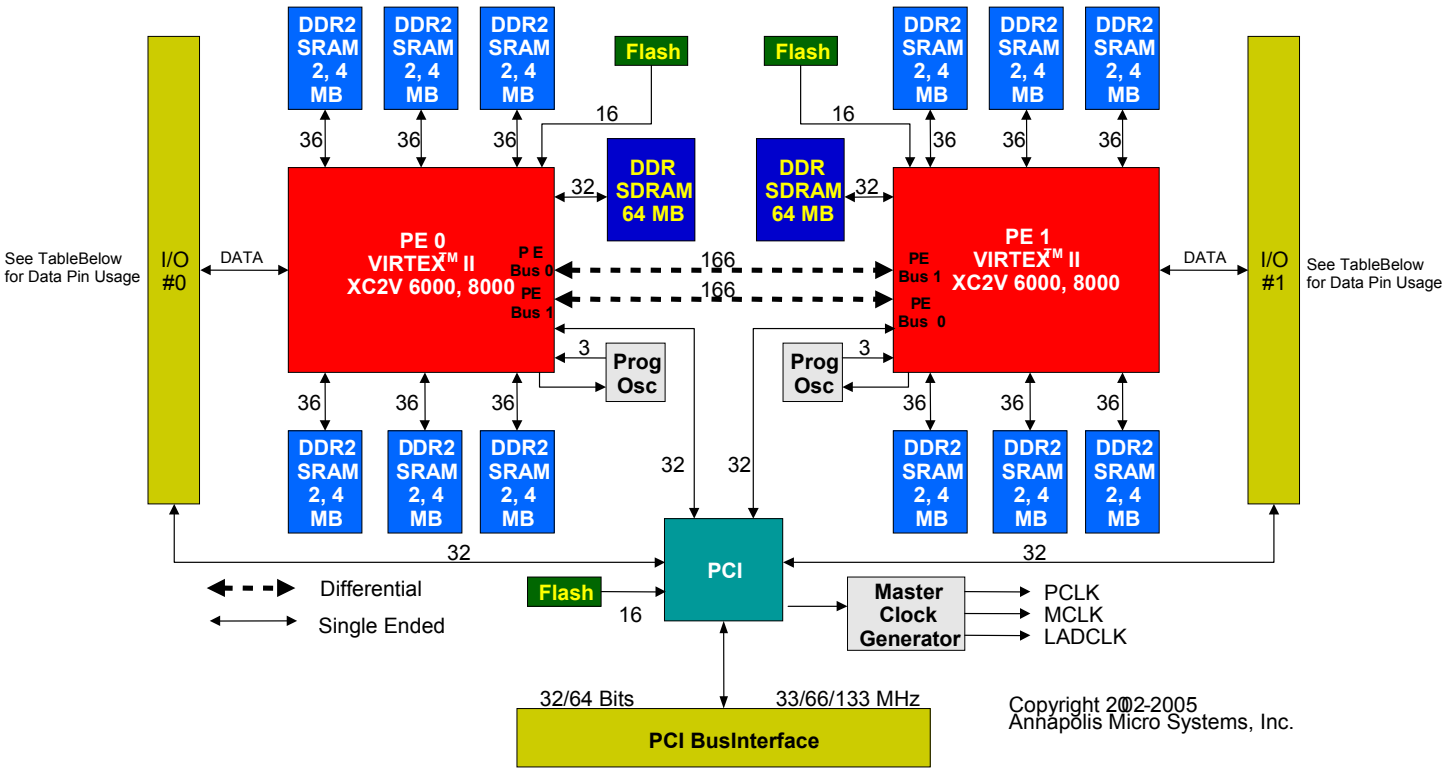


Figure 2-6: WILDSTAR™-II /PCI Block Diagram (All Revs.)

Table 2-4: Data Pin Usage for CoreFire™ and VHDL

I/O Daughter Card Type	CoreFire™	VHDL
Type I	152	154
Type II	158	160

A host computer can communicate with the WILDSTAR™-II /PCI board via the PCI bus interface, which in turn communicates with the board's PCI controller. The WILDSTAR™-II /PCI controller has access to the PEs and two Euro I/O cards using the Local Address Data (LAD) bus. The host also has direct register access and communication with PEs and external I/O cards over the LAD bus.



INFORMATION NOTE

If a second I/O card is placed on the WILDSTAR™-II /PCI, a special I/O slot adapter card must be used with it. This adapter plugs into the second I/O connector of the WILDSTAR™-II /PCI and requires a PCI chassis slot of its own.

2.2.4.1 WILDSTAR™-II /PCI Features

WILDSTAR™-II /PCI has the following system support features:

- Uses two Xilinx® Virtex™ II FPGAs with up to 16 million system gates. PEs can be populated with XC2V6000 or XC2V8000 FPGAs.
- Ability to program PEs from Flash on Power up
- Programmable Flash bank per PE for image storage.
- Status LEDs for PEs
- 12 ports of DDR II SRAM at 960 MBytes/sec per port, for approximately 11 GBytes/sec per board
- DDR DRAM 800 MBytes/sec per PE for total of 1.6 GBytes/sec per board
- PCI Bus Rev 3.0 compliant
- Board works in 3V/5V slots and supports up to 64-bit/66MHz operation
- Host software: Windows NT 4.0® and 2000®, Linux, Solaris™. Includes API and device drivers
- Supports two I/O cards

- I/O Bandwidth
  - 66Mhz PCI up to theoretical max 528 MBytes/sec
  - Approximately 4 GBytes/sec PE to I/O card
  - Approximately 12 GBytes/sec PE0 to PE1 Bandwidth
- Clocks
  - PCLK: Processor Clock, independently programmable and differential
  - ICLK: LAD Clock, fixed frequency and single-ended
- Thermal Management
  - Fan / heat sink combination attached to each PE
  - PE temperature Monitoring
  - Board level Temperature Monitoring
- Power Measurement and Control
  - Individual PE power Supplies and Management
  - PE temperature Monitoring
  - Board level I/V measurements
- CoreFire™ Support
- VHDL Development Environment Supported

2.2.4.2 WILDSTAR™-II /PCI Processing Configurations

The functions available on WILDSTAR™-II /PCI boards vary according to the processing configuration options described below. Each PE is the center of a single processing module, the contents of which are listed in section 2.1 .

Processing Configuration	Processing Modules	Functionality
One PE in location PE0	One	No PE port to port bus, only I/O card 0 slot has access to PE, second I/O slot is limited to LAD Access only.
Two PEs in locations PE0 and PE1	Two	Fully functional

2.2.5 About the WILDSTAR™-II PRO/PCI Board

A host computer can communicate with the WILDSTAR™-II PRO /PCI board via the PCI bus interface, which in turn communicates with the board's PCI controller. The WILDSTAR™-II PRO /PCI controller has access to the PEs and two I/O cards using the Local Address Data (LAD) bus. The host also has direct register access and communication with PEs and external I/O cards over the LAD bus.

The WILDSTAR™-II PRO/PCI architecture has a number of processing configuration options (section 2.2.5.2), with the full architecture containing two processing modules.

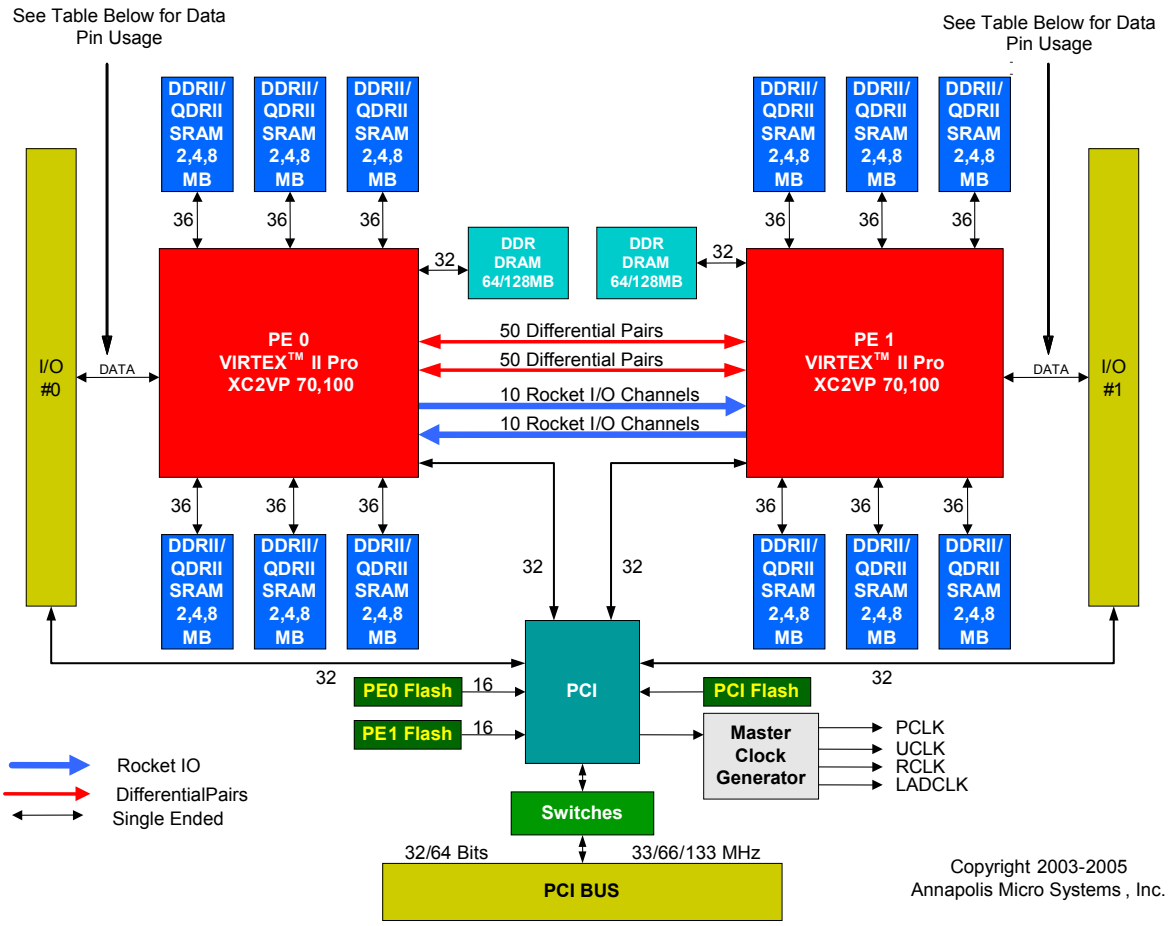


Figure 2-7: WILDSTAR™-II PRO/PCI (Revs. B, C, D)

Table 2-5: Data Pin Usage for CoreFire™ and VHDL

I/O Daughter Card Type	CoreFire™	VHDL
Type I	152	154
Type II	158	160
Type III	77*	79*

*Differential Signaling

As shown in Figure 2-7, the PE0 and PE1 modules use large buses to provide data throughput to I/O cards 0 and 1 respectively. PE0 and PE1 are pin- and bit file-compatible with each other. The PEs are connected by 100 differential pairs and 40 Rocket I/O pairs.

CAUTION

To prevent system damage, fire, and personal injury, *never* use the WILDSTAR™-II PRO/PCI board without the heat sink fan installed and operational. In addition, the operational PCI chassis should have adequate cooling and airflow to ensure a temperature below 85˚C during application operation. Higher temperatures can result in performance degradation and component damage.

The WILDSTAR™-II PRO/PCI incorporates a number of features for power monitoring and thermal management. See Chapter Six for details of these functions.

INFORMATION NOTE

If a second I/O card is placed on the WILDSTAR™-II PRO /PCI, a special I/O slot adapter card must be used with it. This adapter plugs into the second I/O connector of the WILDSTAR™-II PRO /PCI and requires a PCI chassis slot of its own.

2.2.5.1 WILDSTAR™-II PRO/PCI Features

WILDSTAR™-II /PCI has the following system support features:

- Uses two Xilinx® Virtex-II Pro™ FPGAs. PEs can be populated with XC2V70 or XC2VP100 FPGAs.
- Status LEDs for PEs
- 12 ports of DDR II SRAM at 960 MBytes/sec per port, for approximately 11 GBytes/sec per board
- DDR DRAM 800 MBytes/sec per PE for total of 1.6 GBytes/sec per board

- PCI Bus Rev 3.0 compliant
- PCI-x Rev. 2.0 compliant (in Mode 1 only)
- Board works in 3V slots only.
- Host software: Windows NT 4.0® and 2000®, Linux, Solaris™, Irix®. Includes API and device drivers
- Supports two I/O cards
- Clocks
  - PCLK: Processor clock, independently programmable and differential
  - LADCLK: Local Bus clock, single-ended
- Thermal Management
  - Fan / heat sink combination attached to each PE
  - PE temperature Monitoring
  - Board level Temperature Monitoring
- Power Measurement and Control
  - Individual PE power Supplies and Management
  - PE temperature Monitoring
  - Board level I/V measurements
- CoreFire™ Support
- VHDL Development Environment Supported

2.2.5.2 WILDSTAR™-II PRO/PCI Processing Configurations

The functions available on WILDSTAR™-II PRO /PCI boards vary according to the processing configuration options described below. Each PE is the center of a single processing module, the contents of which are listed in section 2.1 .

Processing Configuration	Processing Modules	Functionality
One PE in location PE0	One	No PE port to port bus, only I/O card 0 slot has access to PE, second I/O slot is limited to LAD Access only.
Two PEs in locations PE0 and PE1	Two	Fully functional

2.3 WILDSTAR™ Main Board and I/O Card Compatibilities

I/O card compatibility with Annapolis Micro Systems WILDSTAR™-II boards is determined by I/O card type II or III. Definitions of each type are as follows:

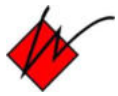
- **Type II** Euro I/O cards use the Virtex™-II FPGA
- **Type III** Euro I/O cards use the Virtex™-II Pro FPGA

Table 2-6: Main Board and I/O Card Compatibilities

Main Board	Type II	Type III
WILDSTAR™/PCI	--	--
WILDSTAR™-II/PCI	X	X[1]
WILDSTAR™-II PRO/PCI	X	X
WILDSTAR™-II/VME	X	X[1]
WILDSTAR™-II PRO/VME	X	X
WILDSTAR™-II PRO ACE for VME	X	X

[1]Requires modifications to support Type III cards. Contact Annapolis Micro Systems for details.


TM



3 GETTING STARTED

3.1 Unpacking and Inspecting the WILDSTAR™-II Board

The WILDSTAR™-II board is shipped in a sealed, static-sensitive package. In order to protect its sensitive components from moisture and static electricity damage, the board should remain in this package until installation time.



CAUTION
Before removing the board from its package, be sure to be grounded of all static electricity.

Your WILDSTAR™-II package includes the following items:

- One WILDSTAR™-II board (PRO/VME and PRO ACE for VME customers also receive a P0 Loopback plug)
- Hardcopy of WILDSTAR™-II Release Notes
- Hardcopy of *WILDSTAR™-II Hardware Reference Manual*
- Hardcopy of *WILDSTAR™-II Software Reference Manual*

- WILDSTAR™-II Documentation CD-ROM, containing:
  - WILDSTAR™-II Release Notes (containing installation instructions for host software and VHDL)
  - *WILDSTAR™-II Hardware Reference Manual*
  - *WILDSTAR™-II Software Reference Manual*

- WILDSTAR™-II Host Software CD-ROM, containing:
  - WILDSTAR™-II API
  - WILDSTAR™-II Device Drivers

- WILDSTAR™-II VHDL Models CD-ROM, containing:
  - Board VHDL Model
  - Application Examples

When handling the WILDSTAR™-II board, avoid touching any of the components on the board's surface, as they are sensitive and can be easily damaged.

Inspect the board thoroughly for damage that may have occurred during shipping. If there is any apparent damage to the board or any items missing from the shipment, contact Annapolis Micro Systems, Inc. using the information provided in Chapter 5 of this manual.

3.2 Board Illustrations and LED Definitions

The following illustrations show major component locations on all WILDSTAR™-II /VME and /PCI boards (WILDSTAR™-II PRO boards are shown in a separate section). LED definitions are also described for each board type.

3.2.1 WILDSTAR™-II / VME Board Views

Component and solder-side views of the WILDSTAR™-II/VME board are shown below. WILDSTAR™-II PRO/VME and PRO ACE for VME are illustrated separately.

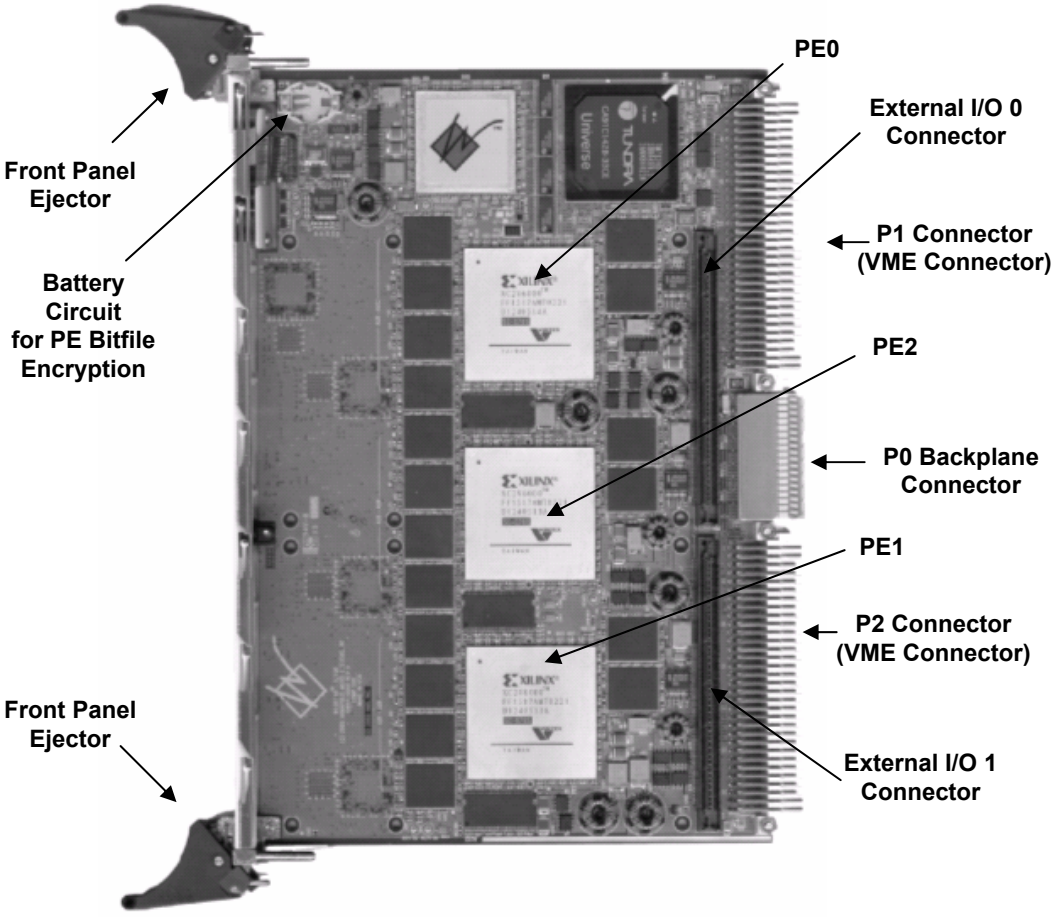


Figure 3-1: WILDSTAR™-II/VME Board, all Revs. (Component Side)

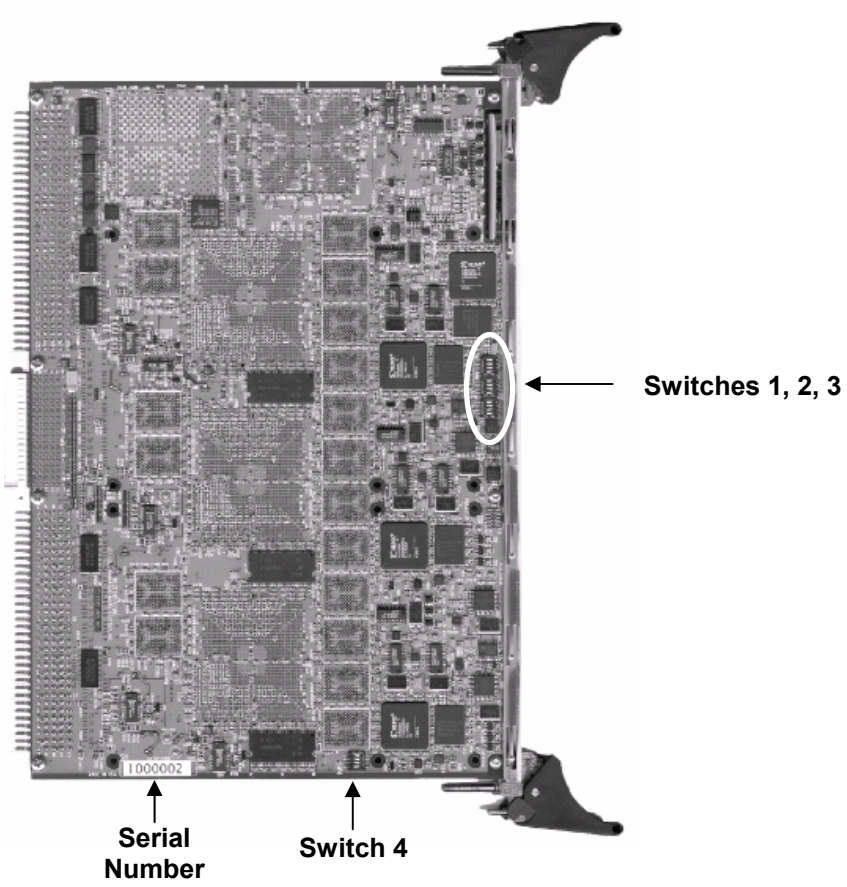


Figure 3-2: WILDSTAR™-II /VME Board, all Revs. (Solder Side)

INFORMATION NOTE

The four-character LED display on the WILDSTAR™-II /VME front panel includes a useful power-up feature. When the VME board is powered up, this display (Figure 3-3) initially shows the chassis slot number into which the board has been inserted, prefixed by an "S."

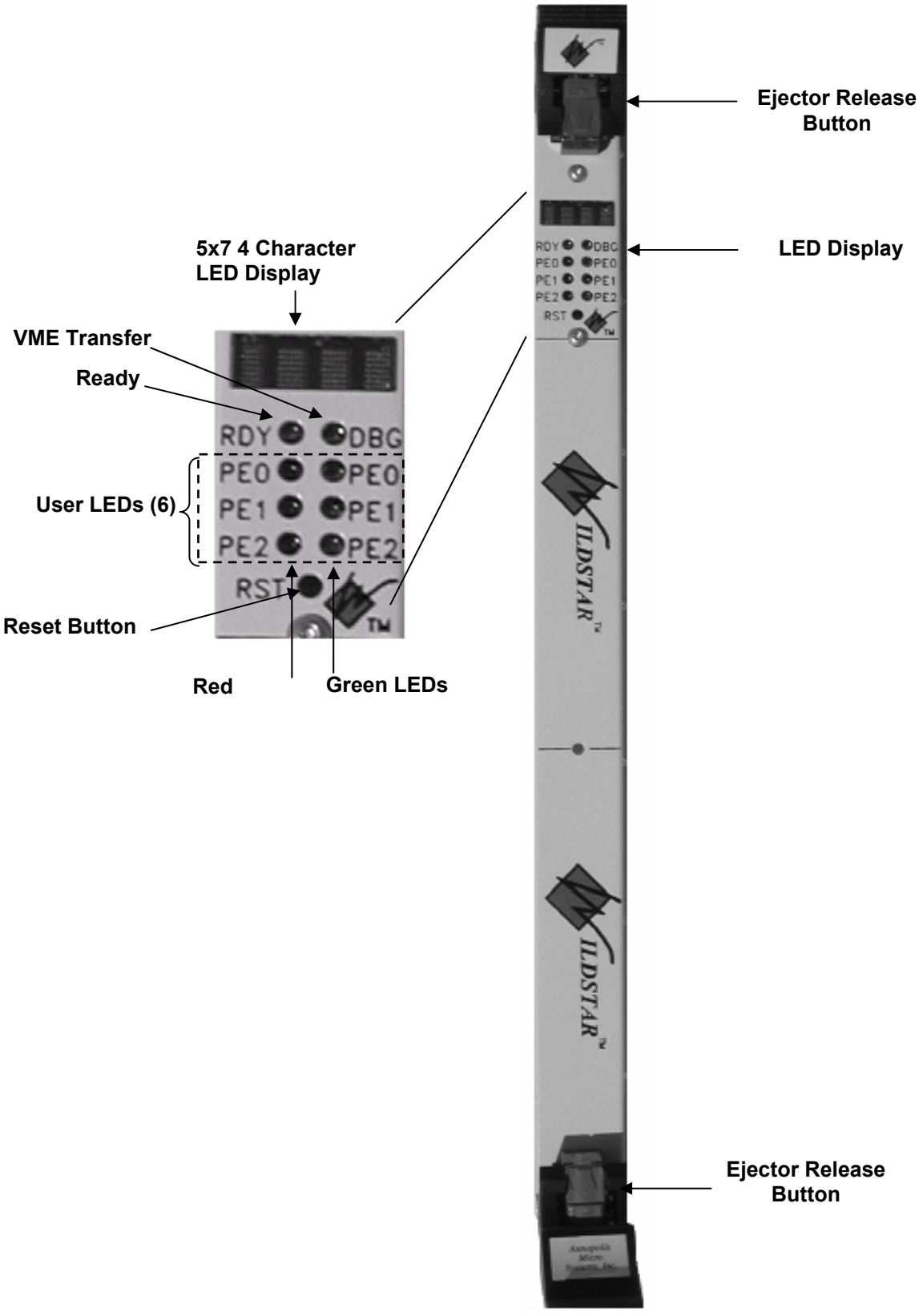


Figure 3-3: WILDSTAR™-II /VME LED Front Panel (without I/O Card Slots)

INFORMATION NOTE

The standard "blank" front panel illustrated in Figure 3-3 is for use without I/O cards. Front panels with openings are available to support I/O cards.

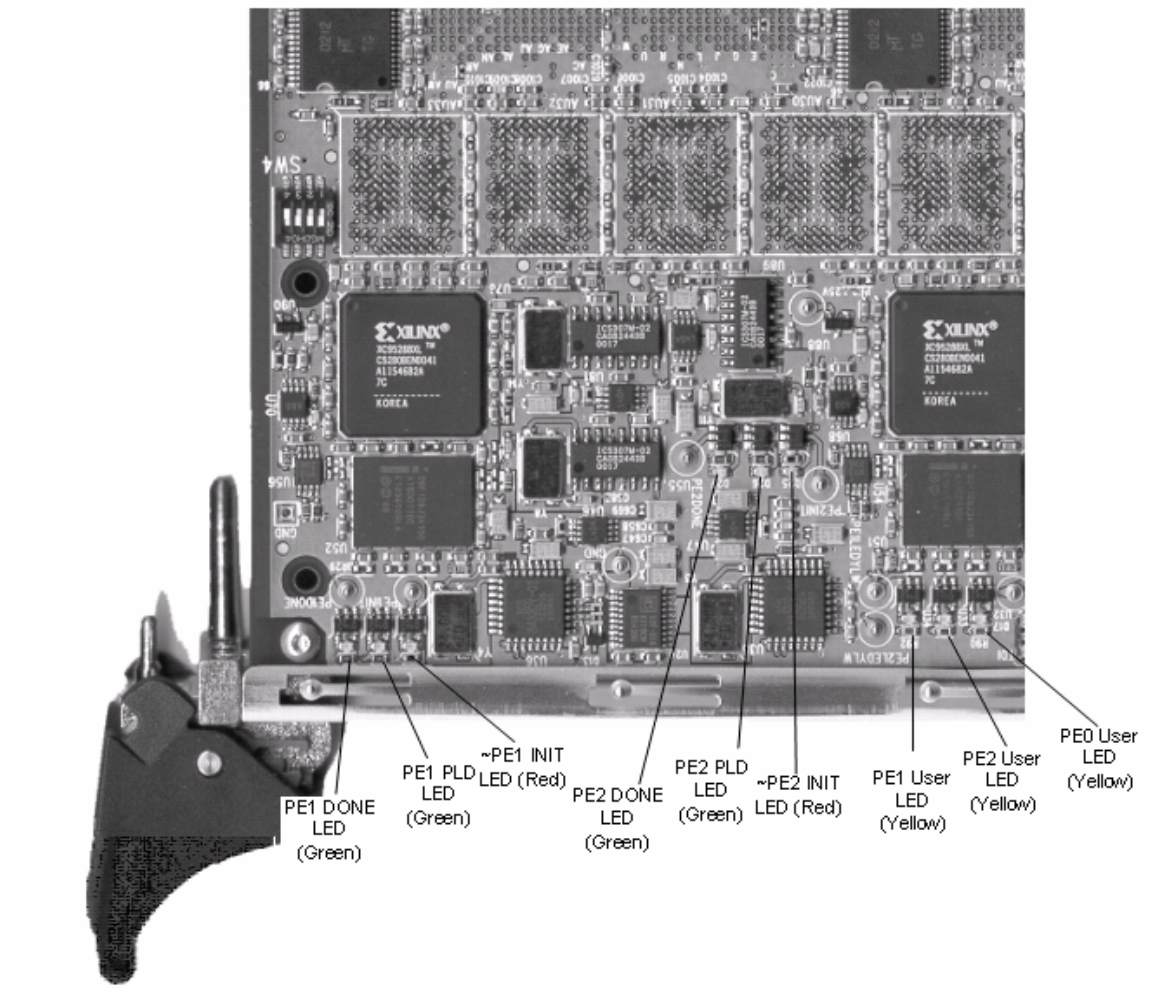


Figure 3-4: WILDSTAR™-II /VME Board LED Locations (all Revs.), Left Solder Side

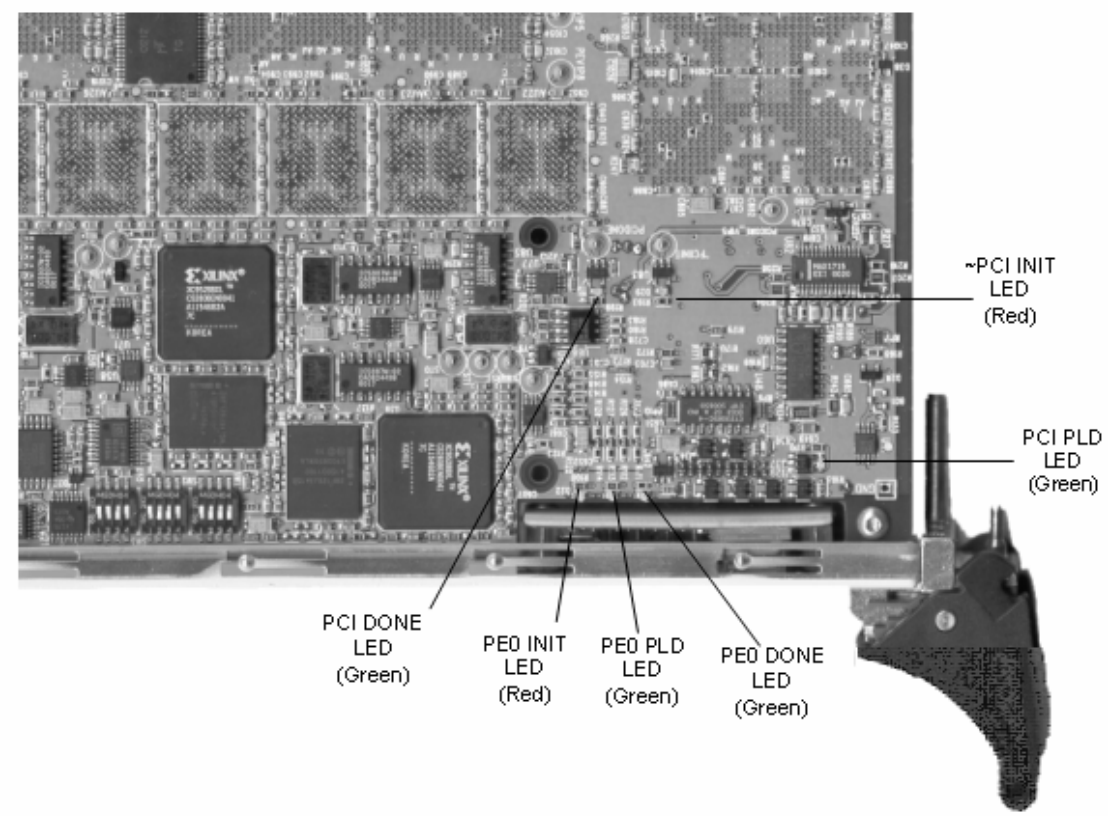


Figure 3-5: WILDSTAR™-II /VME Board LED Locations (all Revs.), Right Solder Side

Table 3-1: WILDSTAR™-II /VME Board LED Definitions

LED Name	LED Color	LED Default Power on State	LED Definition
PE1 DONE	Green	OFF	PE1 is programmed
PE1 PLD	Green	ON	Reserved
~PE1 INIT	Red	OFF	ON if error in PE1 programming, OFF if no error
PE2 DONE	Green	OFF	ON if PE2 is programmed
PE2 PLD	Green	ON	Reserved
~PE2 INIT	Red	OFF	Red if error in PE2 programming, OFF if no error

LED Name	LED Color	LED Default Power on State	LED Definition
PCI DONE	Green	ON	ON if PCI chip is programmed
~PCI INIT	Red	OFF	ON if error in programming, OFF if no error
~PE0 INIT	Red	OFF	ON if error in PCI programming, OFF if no error
PE0 PLD	Green	ON	Reserved
PE0 DONE	Green	OFF	ON if PE0 is programmed
PCI PLD	Green	ON	Reserved
PE0	Green	ON	User Definable (part of Processing Module)
PE0	Red	ON	User Definable (part of Processing Module)
PE1	Green	ON	User Definable (part of Processing Module)
PE1	Red	ON	User Definable (part of Processing Module)
PE2	Green	ON	User Definable (part of Processing Module)
PE2	Red	ON	User Definable (part of Processing Module)
PE1	Yellow	ON	User Definable (part of Processing Module)
PE2	Yellow	ON	User Definable (part of Processing Module)
PE0	Yellow	ON	User Definable (part of Processing Module)

3.2.2 WILDSTAR™-II PRO/VME Board Views

Component and solder-side views of the WILDSTAR™-II PRO/VME are shown below.

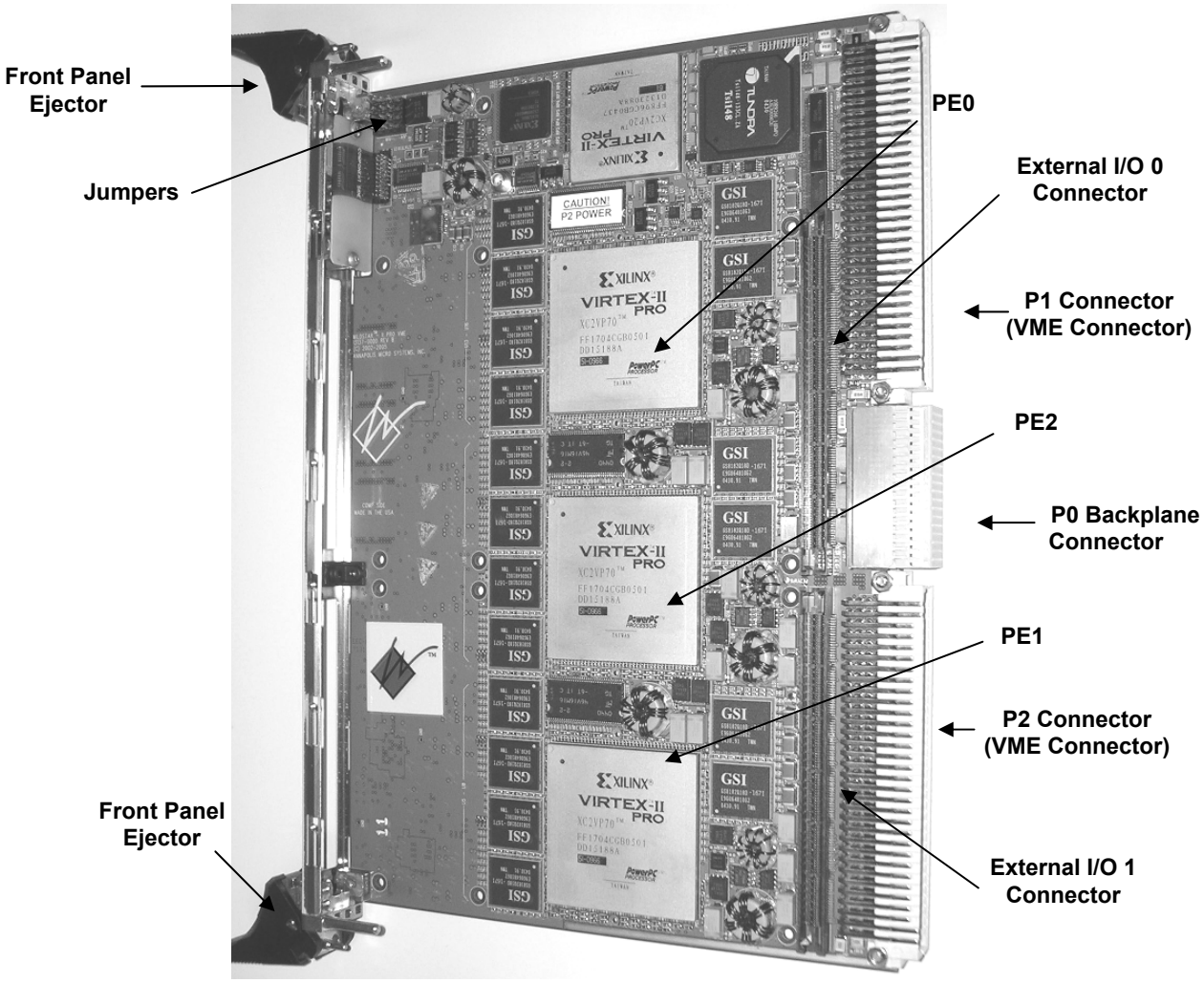


Figure 3-6: WILDSTAR™-II PRO/VME Rev. B (Component Side)

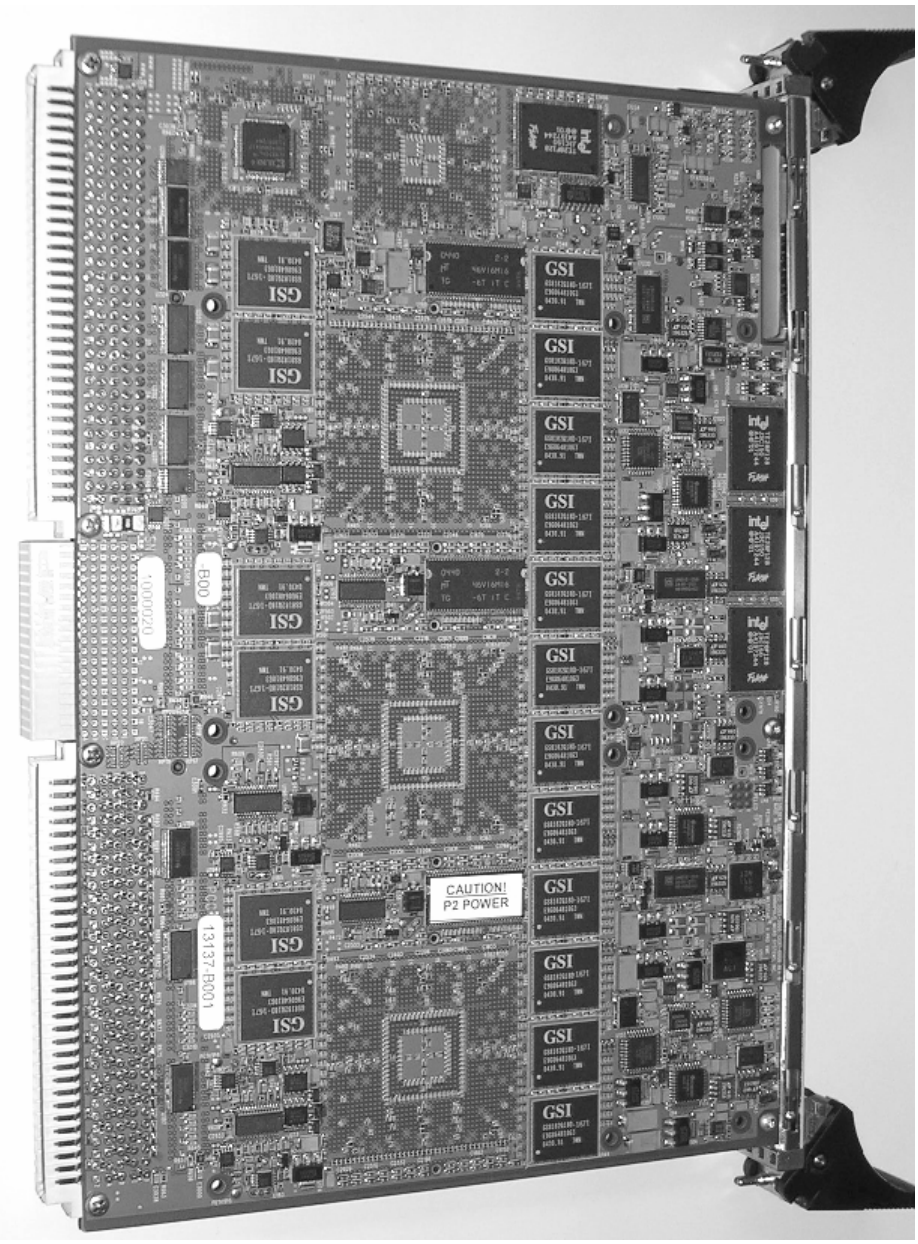


Figure 3-7: WILDSTAR™-II PRO/VME Rev. B (Solder Side)

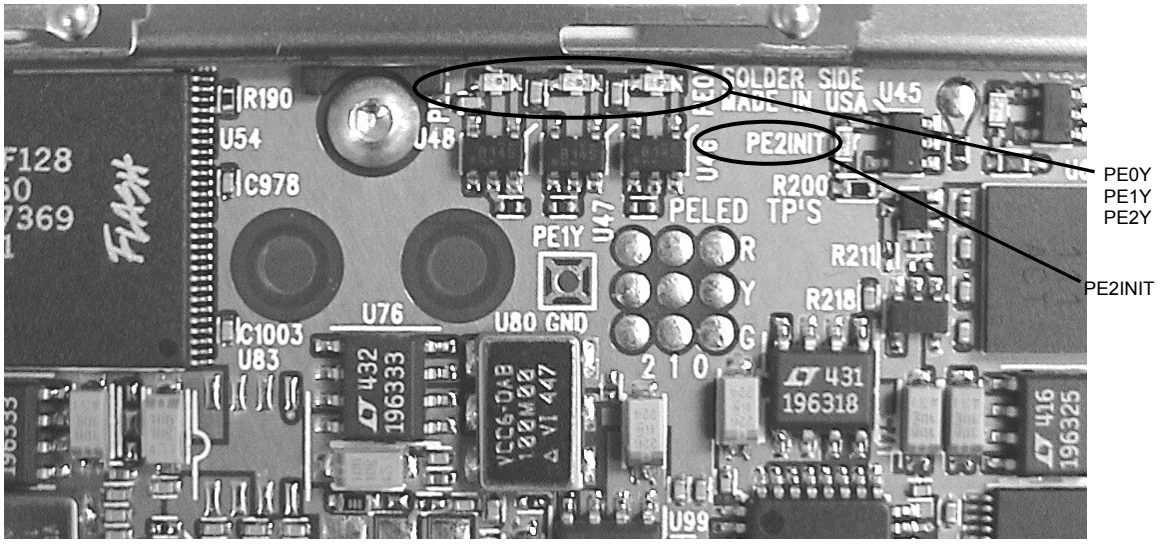


Figure 3-8: LEDs on WILDSTAR™-II PRO/VME (Solder Side, Detail 1)

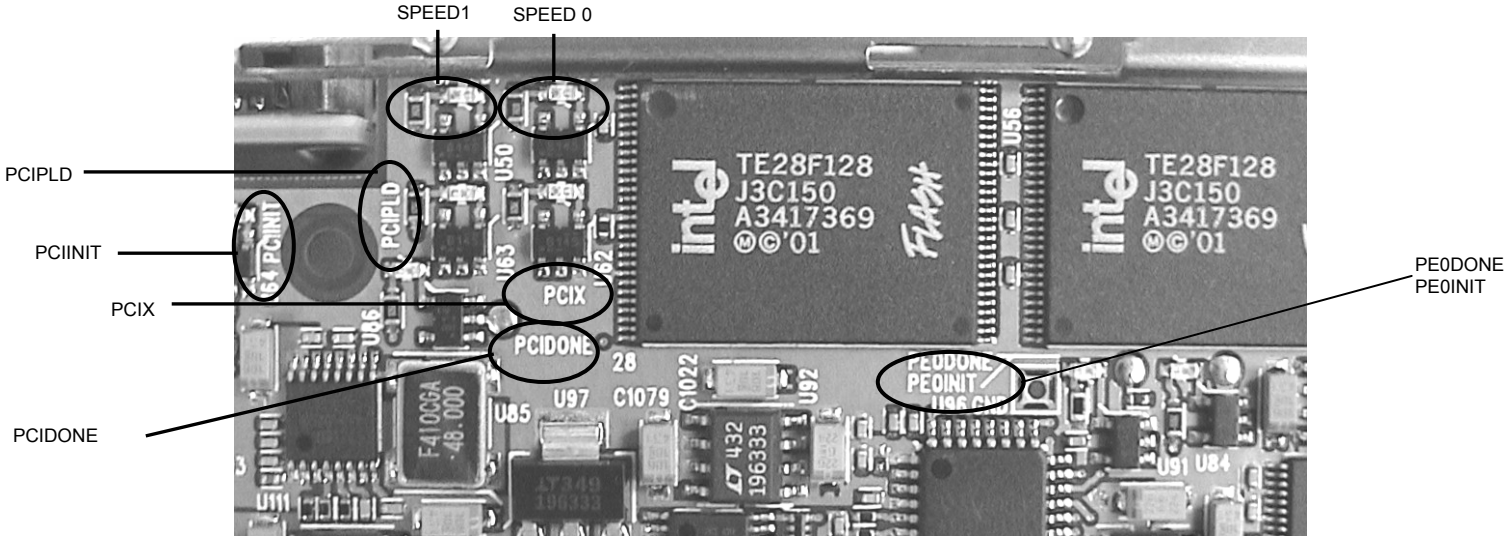


Figure 3-9: LEDs on WILDSTAR™-II PRO/VME (Solder Side, Detail 2)

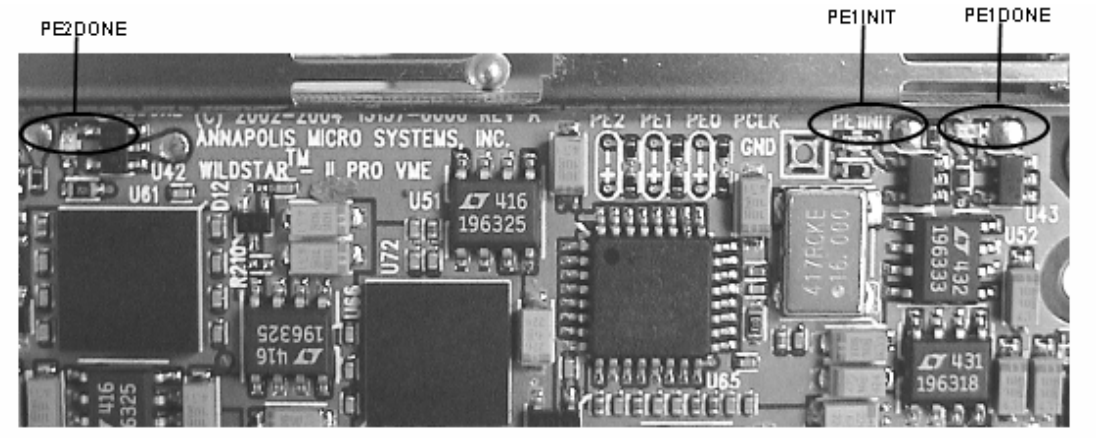


Figure 3-10: LEDs on WILDSTAR™-II PRO/VME (Solder Side, Detail 3)

Table 3-2: WILDSTAR™-II PRO/VME LED Definitions

LED Name	LED Color	LED Default Power on State	LED Definition
PCI DONE	Green	ON	PCI Controller is programmed
PCI INIT	Red	OFF	ON if error in programming, OFF if no error
PCI-X	Yellow	ON	Indicates PCI Bus speed
PCI SPEED0	Yellow	OFF	Indicates PCI Bus speed
PCI SPEED1	Yellow	OFF	Indicates PCI Bus speed
PCI PLD	Green	ON	Reserved
PE0DONE	Green	OFF	PE Programming
PE1DONE	Green	OFF	PE Programming
PE2DONE	Green	OFF	PE Programming
PE0INIT	Red	OFF	ON if error during configuration
PE1INIT	Red	OFF	ON if error during configuration
PE2INIT	Red	OFF	ON if error during configuration
PE0Y	Yellow	ON	User Definable
PE1Y	Yellow	ON	User Definable
PE2Y	Yellow	ON	User Definable

3.2.3 WILDSTAR™-II PRO ACE for VME (Rev. B) Board Views

Component and solder-side views of the WILDSTAR™-II PRO ACE for VME (Rev. B) are shown below.

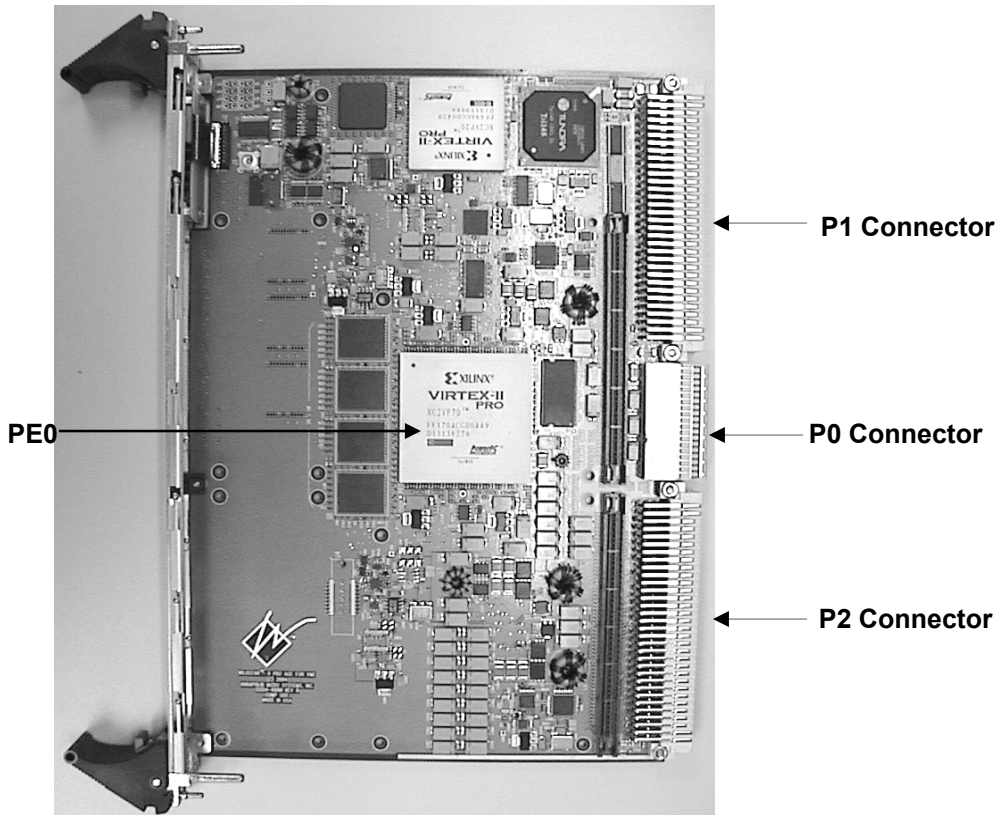


Figure 3-11: WILDSTAR™-II PRO ACE for VME Board, Rev. B (Component Side)

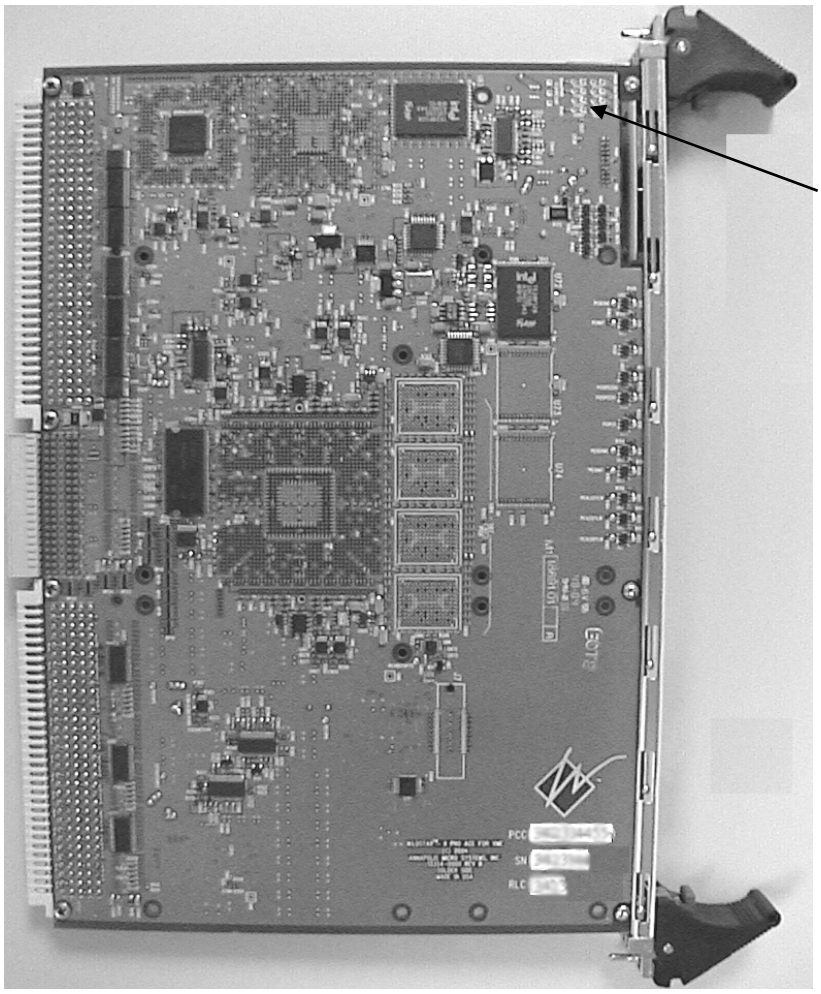


Figure 3-12: WILDSTAR™-II PRO ACE for VME Board, Rev. B (Solder Side)

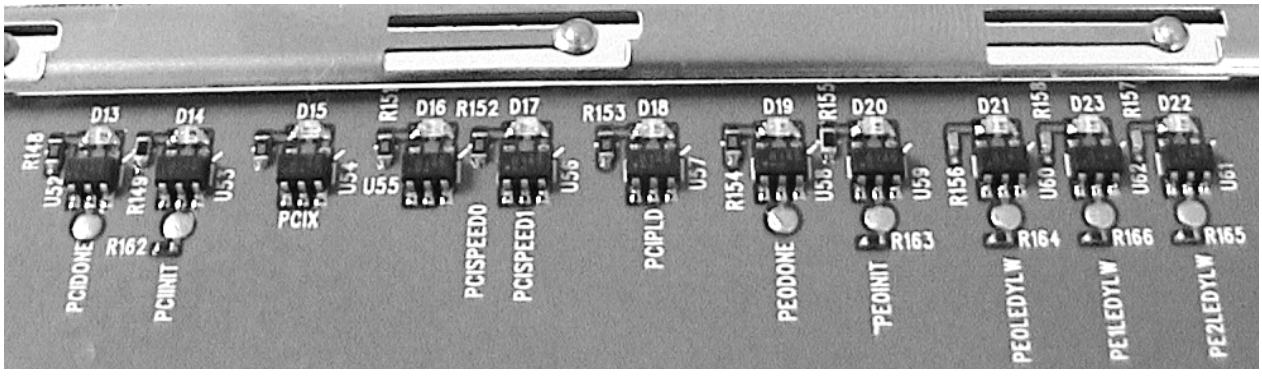


Figure 3-13: LEDs on WILDSTAR™-II PRO ACE for VME Rev. B (Solder Side, Detail)

Table 3-3: WILDSTAR™-II PRO ACE for VME (Rev. B) LED Definitions

LED Name	LED Color	LED Default Power on State	LED Definition
PCI DONE	Green	ON	PCI Controller is programmed
PCI INIT	Red	OFF	ON if error in programming, OFF if no error
PCI-X	Yellow	ON	Indicates PCI Bus speed
PCI SPEED0	Yellow	OFF	Indicates PCI Bus speed
PCI SPEED1	Yellow	OFF	Indicates PCI Bus speed
PCI PLD	Green	ON	Reserved
PE0DONE	Green	OFF	PE Programming
PE0INIT	Red	OFF	ON if error during configuration
PE0LEDYLW	Yellow	ON	User Definable
PE1LEDYLW	Yellow	ON	User Definable
PE2LEDYLW	Yellow	ON	User Definable

Table 3-4: LED Status and PRO ACE VME Bus Speed

PCI SPEED0 LED	PCI SPEED1 LED	PCI-X LED	Speed
OFF	OFF	ON	66 MHz PCI-X
OFF	OFF	OFF	33 MHz PCI
OFF	ON	ON	Reserved
OFF	ON	OFF	Reserved
ON	OFF	ON	100 MHz PCI-X
ON	OFF	OFF	Reserved
ON	ON	ON	133 MHz PCI-X
ON	ON	OFF	66MHz PCI

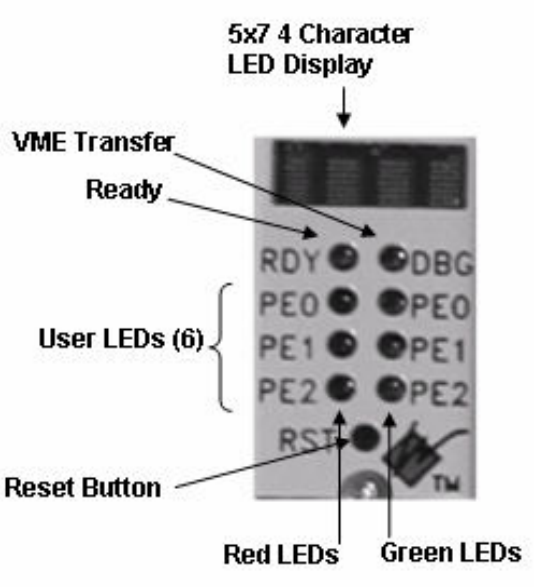


Figure 3-14: Front Panel Display for ACE Board (Detail)

INFORMATION NOTE

The four-character LED display on the WILDSTAR™-II PRO ACE for VME front panel includes a useful power-up feature. When the VME board is powered up, this display initially shows the chassis slot number into which the board has been inserted.

3.2.4 WILDSTAR™-II /PCI Board Views

Component and solder-side views of the WILDSTAR™-II /PCI are shown below.

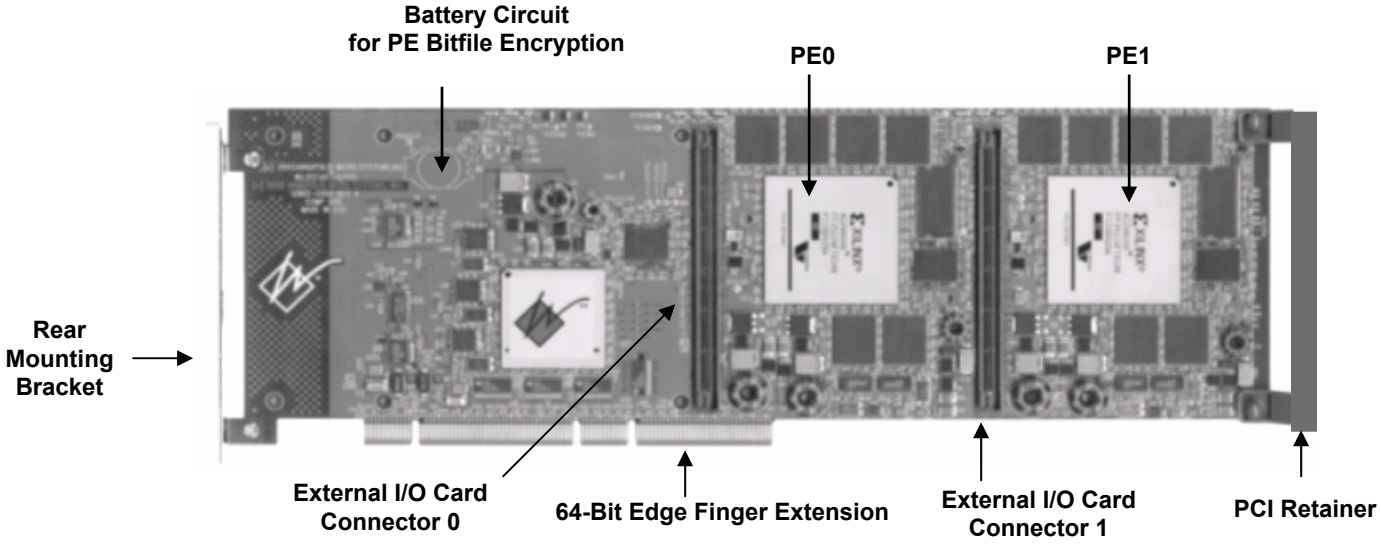


Figure 3-15: WILDSTAR™-II /PCI Board, Rev. A (Component Side, Shown without Heat Sinks or Fans)

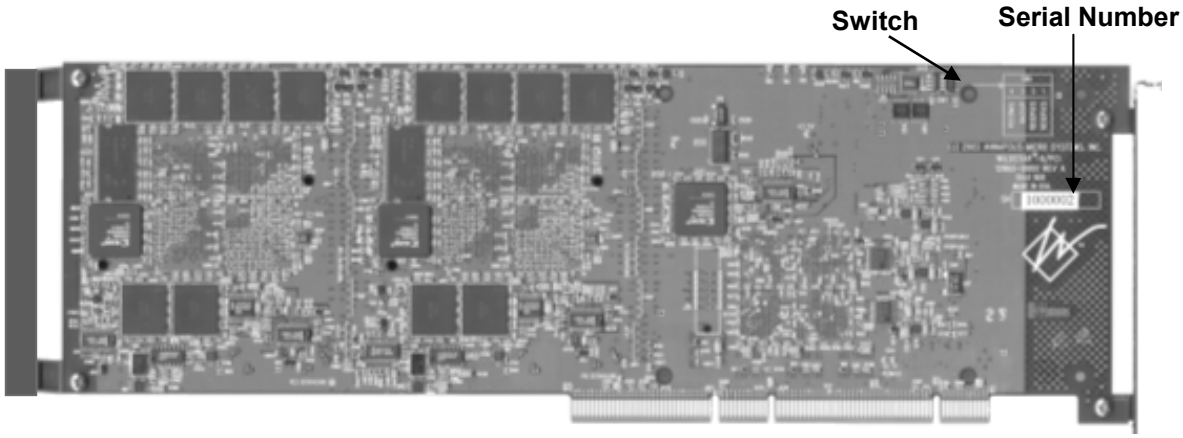


Figure 3-16: WILDSTAR™-II /PCI Board, Rev. A (Solder Side, Shown without Heat Sinks or Fans)

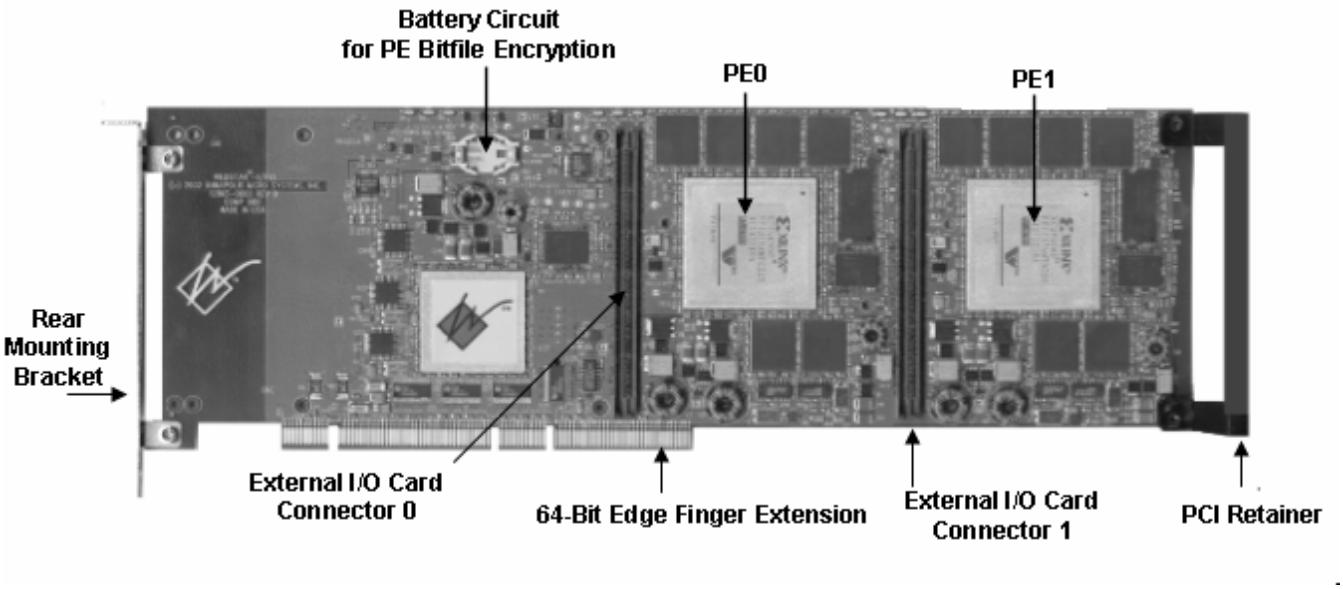


Figure 3-17: WILDSTAR™-II /PCI Board, Rev. B (Component Side, Shown without Heat Sinks or Fans)

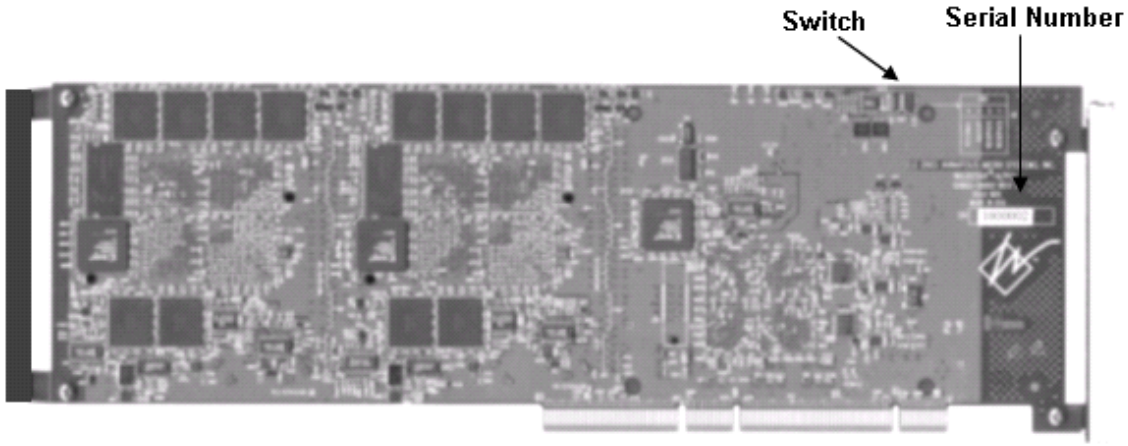


Figure 3-18: WILDSTAR™-II /PCI Board, Rev. B (Solder Side, Shown without Heat Sinks or Fans)

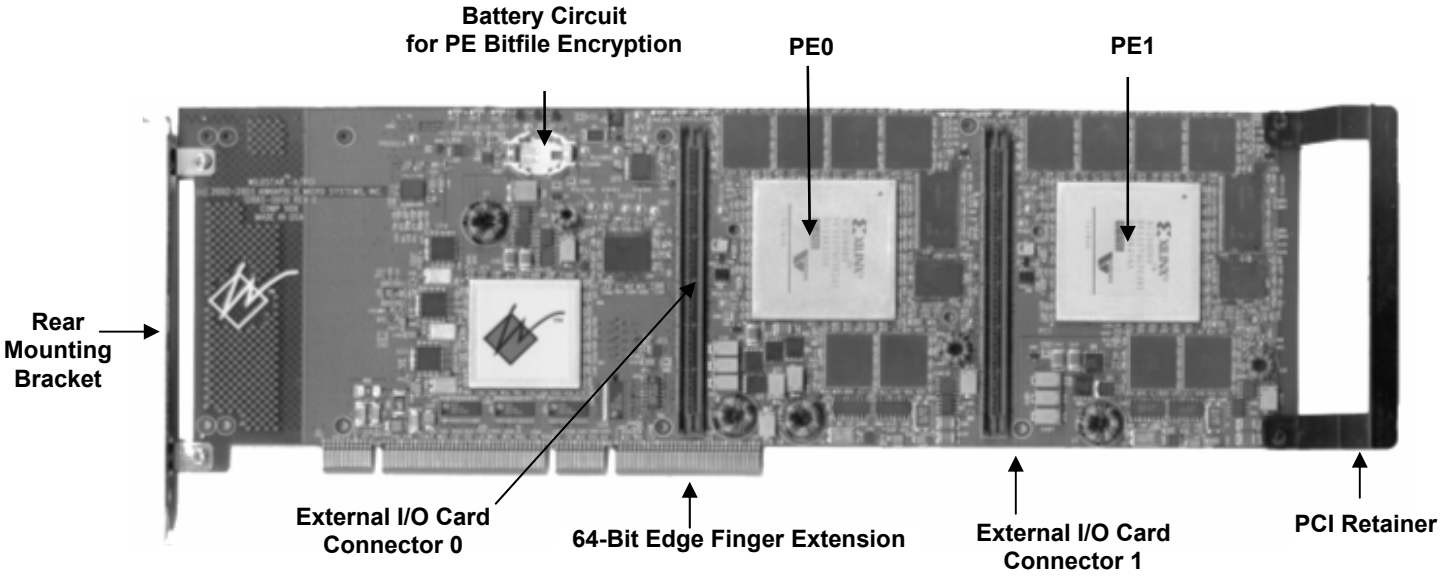


Figure 3-19: WILDSTAR™-II /PCI Board, Revs. C, D (Component Side, Shown without Heat Sinks or Fans)

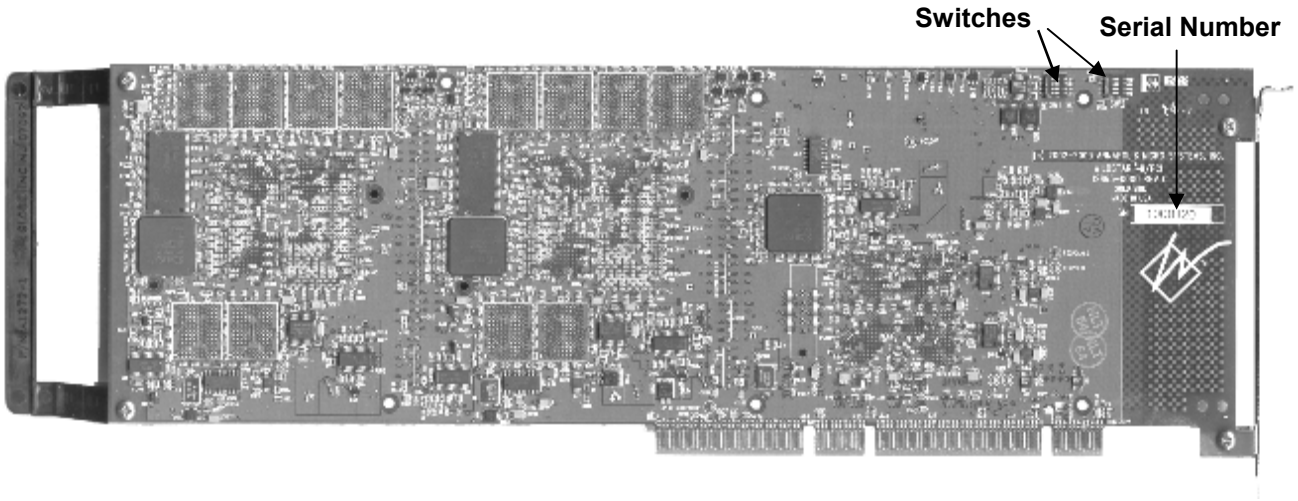


Figure 3-20: WILDSTAR™-II /PCI Board, Revs. C, D (Solder Side, Shown without Heat Sinks or Fans)

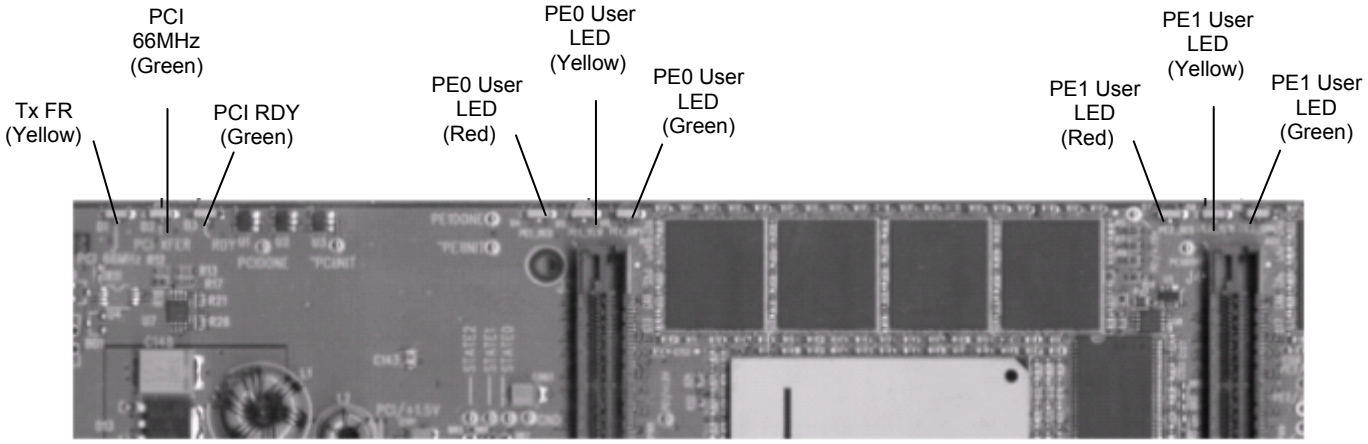


Figure 3-21: WILDSTAR™-II /PCI Board LEDs (all Revs), Top Component Side

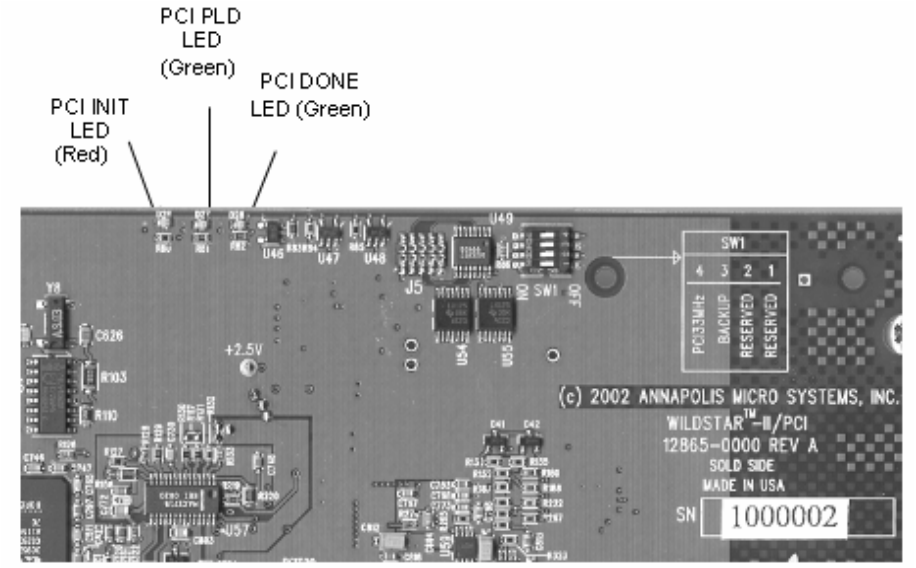


Figure 3-22: WILDSTAR™-II /PCI Board LEDs (all Revs), Top Right Solder Side

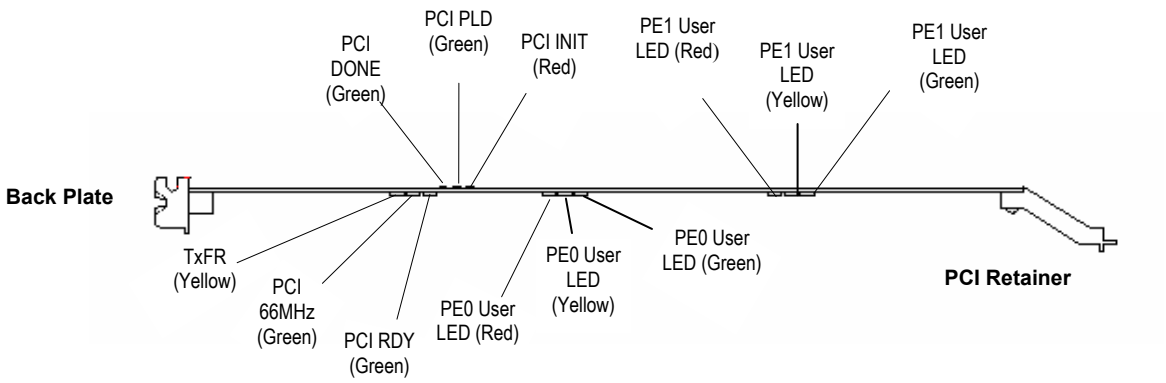


Figure 3-23: WILDSTAR™-II /PCI Board LEDs (all Revs), Top Edge View

Table 3-5: WILDSTAR™-II /PCI Board LED Definitions (all Revs)

LED Name	LED Color	LED Default Power on State	LED Definition
PCI TxFR	Yellow	OFF	PCI bus transfer in progress
PCI 66MHz	Green	Slot Dependent	PCI Bus is capable of 66 MHz operation
PCI RDY	Green	ON	PCI Controller is ready
PCI DONE	Green	ON	PCI Controller is programmed
PCI PLD	Green	ON	Reserved
PCI INIT	Red	OFF	ON if error in programming, OFF if no error
PE0 User	Red	ON	User Definable
PE0 User	Yellow	ON	User Definable
PE0 User	Green	ON	User Definable
PE1 User	Red	ON	User Definable
PE1 User	Yellow	ON	User Definable
PE1 User	Green	ON	User Definable

3.2.5 WILDSTAR™-II PRO/PCI Board Views

Component and solder-side views of the WILDSTAR™-II PRO/PCI are shown below.

INFORMATION NOTE

WILDSTAR™-II PRO/PCI motherboards can only be placed in 3.3V PCI slots.

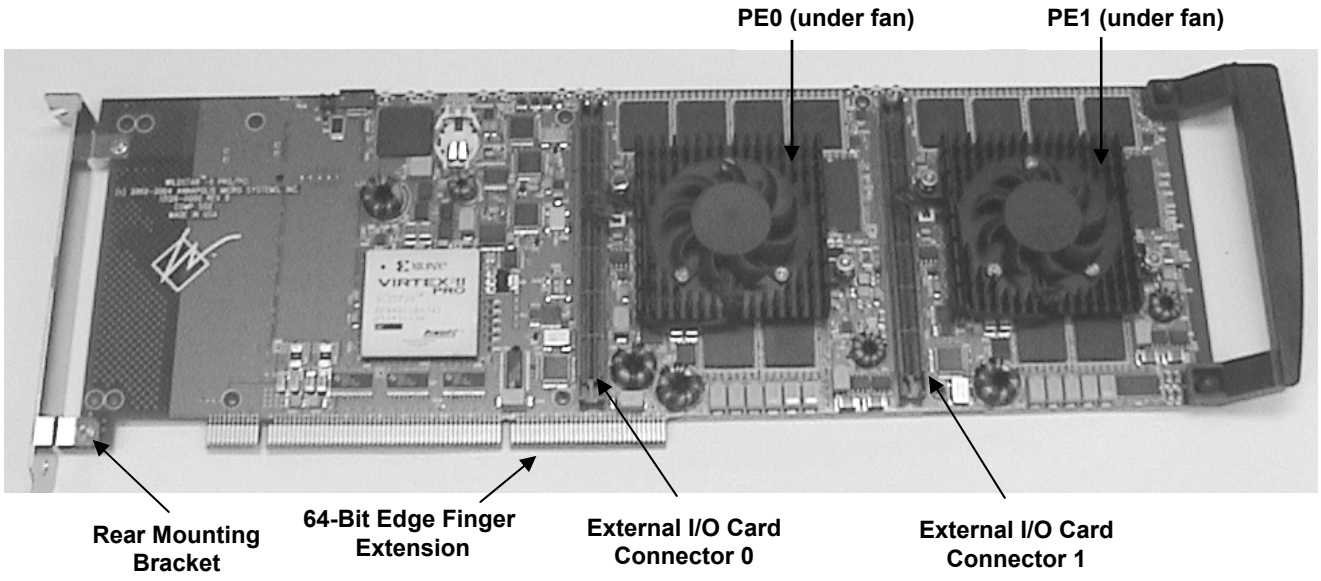


Figure 3-24: WILDSTAR™-II PRO/PCI Board, Rev. B, C, D (Component Side, Shown with Heat Sinks and Fans)

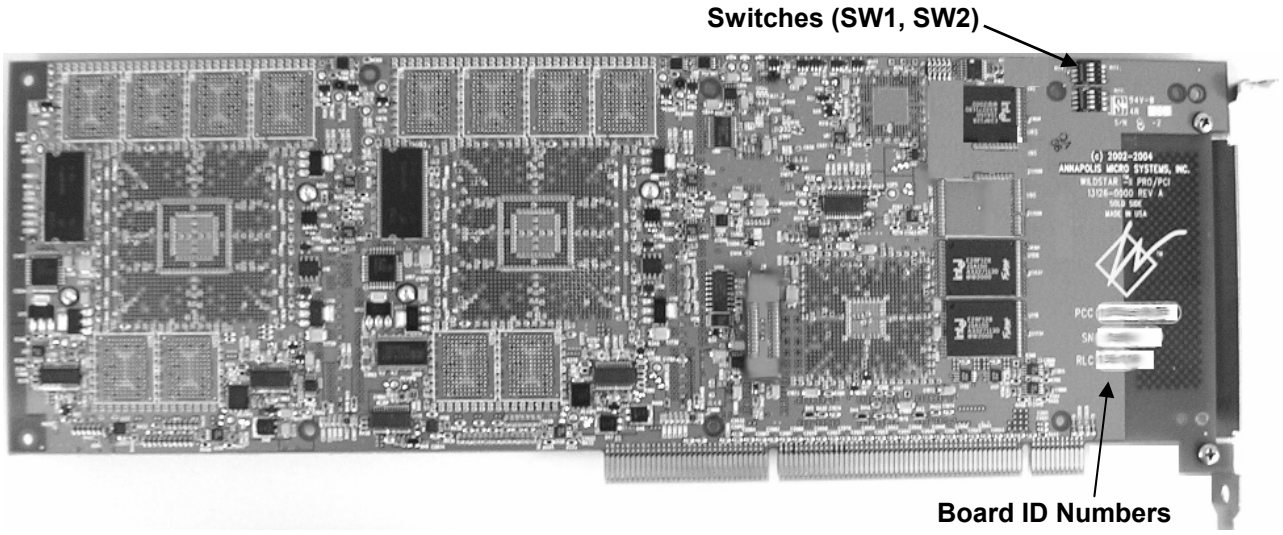


Figure 3-25: WILDSTAR™-II PRO/PCI Board, Rev. B (Solder Side)

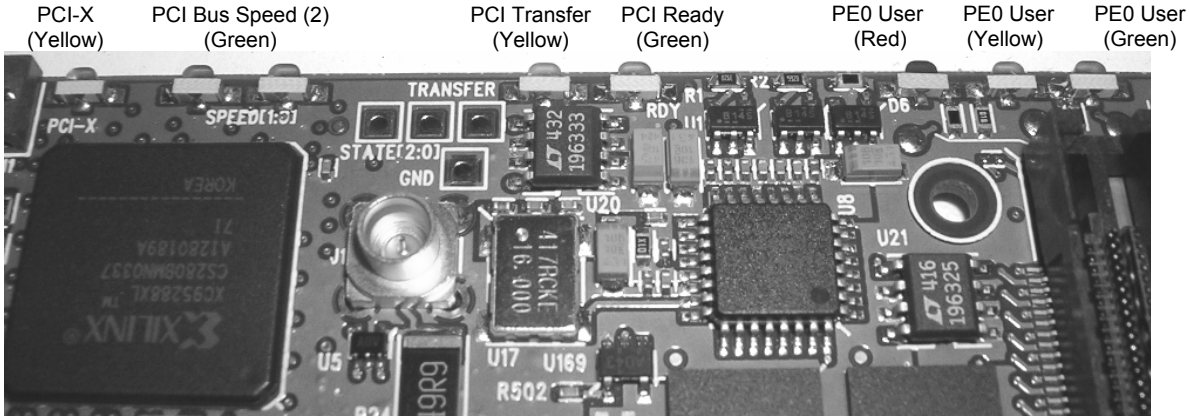


Figure 3-26: PRO /PCI Function and PE0 Status LEDs, Component Side

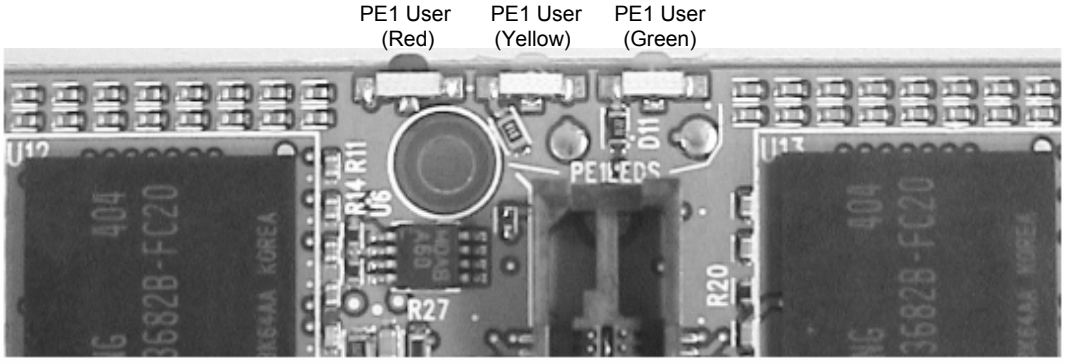


Figure 3-27: PRO /PCI PE1 Status LEDs, Component Side

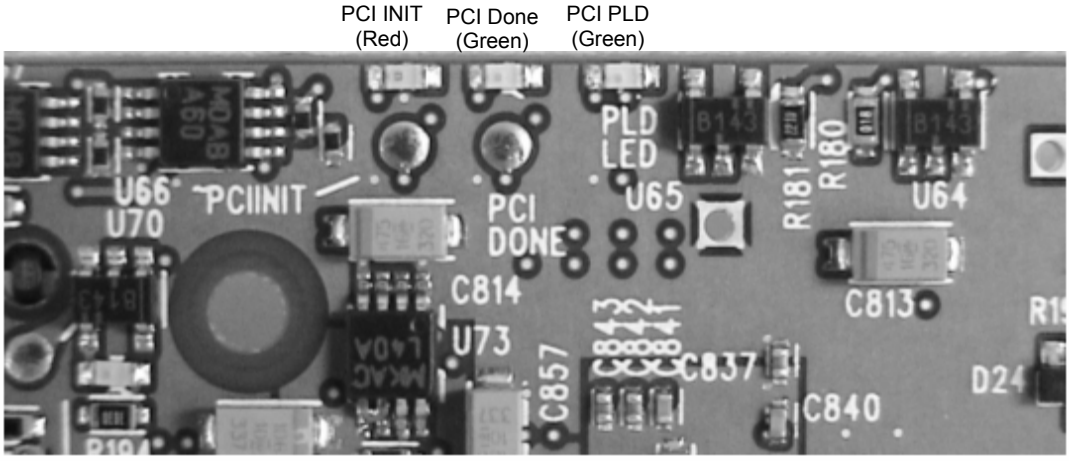


Figure 3-28: PRO /PCI Function LEDs, Solder Side

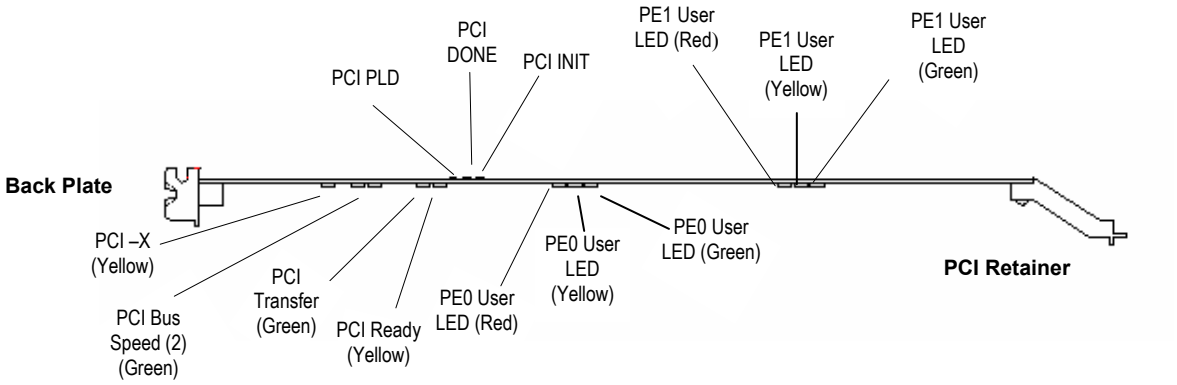


Figure 3-29: WILDSTAR™-II PRO /PCI Board LEDs, Top Edge View

Table 3-6: WILDSTAR™-II PRO /PCI Board LED Definitions

LED Name	LED Color	LED Default Power on State	LED Definition
PCI RDY	Green	ON	PCI Controller is ready
PCI TRANSFER			PCI bus transfer in progress
PCI DONE	Green	ON	PCI Controller is programmed
PLD LED	Green	ON	Reserved
PCI INIT	Red	OFF	ON if error in programming, OFF if no error
PE0 User	Red	ON	User Definable
PE0 User	Yellow	ON	User Definable
PE0 User	Green	ON	User Definable
PE1 User	Red	ON	User Definable
PE1 User	Yellow	ON	User Definable
PE1 User	Green	ON	User Definable
PCI SPEED 1	Yellow	OFF	Indicates PCI Bus speed
PCI SPEED 2	Yellow	OFF	Indicates PCI Bus speed
PCI-X	Green	Slot Dependent	Indicates PCI Bus speed

Table 3-7: LED Status and PRO /PCI Bus Speed

PCI SPEED 1 LED	PCI SPEED 2 LED	PCI-X LED	Speed
OFF	OFF	ON	66 MHz PCI-X
OFF	OFF	OFF	33 MHz PCI
OFF	ON	ON	Reserved
OFF	ON	OFF	Reserved
ON	OFF	ON	100 MHz PCI-X
ON	OFF	OFF	Reserved
ON	ON	ON	133 MHz PCI-X
ON	ON	OFF	66MHz PCI


TM

4 INSTALLING THE WILDSTAR™-II BOARD

This chapter provides hardware installation instructions for WILDSTAR™-II /VME and /PCI boards, as well as switch setting descriptions and PE, board, and system reset options. Separate sections are included here for VME and PCI series boards.

INFORMATION NOTE

PDFs of WILDSTAR™-II hardware and software manuals and host software release notes can be found on the Documentation CD included with your board.

4.1 WILDSTAR™-II /VME Installation

No onboard 3.3V power source is included on Revs A and B of the WILDSTAR™-II / VME. Subsequently, it can only be installed in a VME64X chassis, which has its own 3.3V power supply. (The standard VME chassis lacks a built-in 3.3V power supply.)

WILDSTAR™-II/ VME Rev. C boards have an optional, factory-configured 3.3V power supply for use with a legacy VME chassis. This option also supplies 3.3V power to all I/O cards and I/O PE pins.

Table 4-1: WILDSTAR™-II /VME (excluding PRO-Series) Board Revs and Chassis Compatibility

	VME64X	Legacy VME
Revs. A, B	Yes	No
Rev. C	Yes	Yes*

*Available only as a factory-configured option

To visually differentiate between a standard VME chassis and a VME64X chassis, examine the pins on the 6U backplane. The standard VME chassis has 32 rows of pins stacked three across (96 pins total). The VME64X adds a row of 32 additional pins to either side of the original 3 X 32 pin configuration and therefore has 32 rows of pins stacked five across (160 pins total).

WARNING
Do not attempt to insert a WILDSTAR™-II /VME board with a P0 Backplane Connector into a chassis that does not accept P0.

4.1.1 WILDSTAR™-II /VME Switch Settings

Switches on the solder side of the WILDSTAR™-II /VME board are used to set basic configuration options (see Figure 4-1). The board is delivered with all of its switches in their default settings. Confirm that all switches are set according to the information in the tables below.

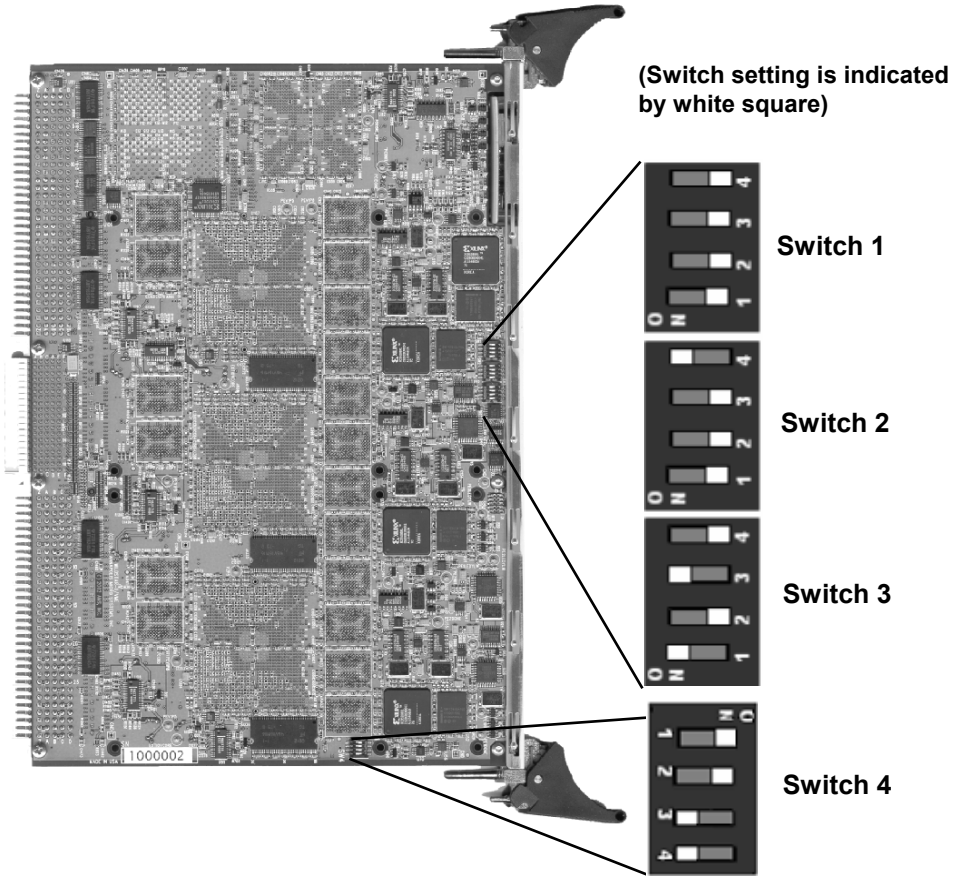


Figure 4-1: Location of WILDSTAR™-II /VME Switches, in Default Positions (Solder Side of Board)

Table 4-2: WILDSTAR™-II /VME Board Switch 1 (SW1) Settings

Position No.	Position Name	Switch Description	Default Setting
1	Flash Backup	When ON: Use Flash backup When OFF: Normal Flash operation	OFF
2	Reserved 2	Leave in OFF position	OFF
3	PCI Flash Protect	When ON, PCI Controller is Flash protected When OFF, PCI Controller Flash Update is enabled	OFF
4	Reserved 4	Leave in OFF position	OFF

Table 4-3: WILDSTAR™-II /VME Board Switch 2 (SW2) Settings

Position No.	Position Name	Switch Description		Default Setting
		ON	OFF	
1	Slot ID 1	Logic 0	Logic 1	OFF
2	Slot ID 2	Logic 0	Logic 1	OFF
3	Slot ID 3	Logic 0	Logic 1	OFF
4	Slot ID 4*	Logic 0	Logic 1	ON

* Slot ID [4:0] determines the VME address window to which the WILDSTAR™-II / VME board will respond. The SLOT ID bits are ignored when geographical addressing is used.

Table 4-4: WILDSTAR™-II /VME Board Switch 3 (SW3) Settings

Position No.	Position Name	Switch Description		Default Setting
1	Reserved	Leave in ON position		ON
2	PE Flash Protect	Leave in OFF position		OFF
3	Geographic Addressing Enable*	When ON, use geographic addressing. When OFF, use switches		ON
4	Slot ID 0	When ON: Logic 0	When OFF: Logic 1	OFF

* Geographical addressing determines which VME address window will be responded to by the WILDSTAR™-II /VME board. The SLOT ID switch settings are ignored when geographical addressing is used.

Table 4-5: WILDSTAR™-II /VME Board Switch 4 (SW4) Settings

Position No.	Position Name	Switch Description	Default Setting
1	Bi-Mode Disable	When ON, the board participates in VME transactions. When OFF, WILDSTAR-II/ VME is isolated from the VME backplane.	ON
2	VME Reset_In Enable	When ON, WILDSTAR-II / VME is reset by VME bus reset. When OFF, the board ignores VME bus reset.	ON
3	VME Reset_Out Enable	When ON, WILDSTAR-II / VME can cause VME bus resets. When OFF, the board cannot cause a VME bus reset.	OFF
4	Board Reset Select *(via front panel button or software)*	When ON and reset, all PEs and I/O cards are reset. User still has access to board. When OFF and reset, PCI, VME-PCI bridge, PEs, and I/O cards are all reset. User loses access to board and must re-open it.	OFF

4.1.2 Hardware Installation for WILDSTAR™-II /VME

Follow the steps below to install the WILDSTAR™-II /VME board into the host system.

1. Connect the ground strap to the user.
2. Shut down the host system and power off.

CAUTION

The WILDSTAR™-II /VME board must be inserted into the chassis gently to avoid scraping fragile components on the edges of the board.

3. Grasp the WILDSTAR™-II /VME board by the front panel ejectors with the VME connectors facing away from the user.
4. Gently slide the WILDSTAR™-II /VME board into the slot.
5. When the VME connectors first contact the backplane connectors, push firmly on the front panel ejectors until they snap into place. Engage the front panel ejector tabs with the chassis rails to fully seat WILDSTAR™-II /VME in the VME backplane.

To remove the VME board, connect a ground strap to the user and power off the host system. Gently press the red ejector release buttons, press down on the front panel ejectors, and slowly pull the board from the chassis.

4.1.3 Power Up System

After all cable connections have been made and the board has been properly plugged in, turn on power to the host system. Once the WILDSTAR™-II /VME board has been installed in the host system and the host machine has been powered up, the host software and VHDL models can be installed.

INFORMATION NOTE

Instructions for installing host software on WILDSTAR™-II boards can be found in the release notes accompanying the board.

4.2 WILDSTAR™-II PRO /VME Installation

Instructions for installing the WILDSTAR™-II PRO/VME board are included below. *This board can only be used in a VME64x chassis*.

Table 4-6: WILDSTAR™-II PRO/VME Board Revs and Chassis Compatibility

	VME64X	Legacy VME
Revs. A, B	Yes	No

WARNING

Check to see if your WILDSTAR™-II PRO/VME board has the P2 power adapter backplane option, indicated by the following label affixed to both sides of the board:





The label location is shown in the figures below. If your board has the P2 power option, *never* plug it into a chassis that uses any of the P2 signals (as with a RACEway® crossbar, for instance). Doing so could destroy the WILDSTAR™-II PRO/VME and other boards connected to P2.

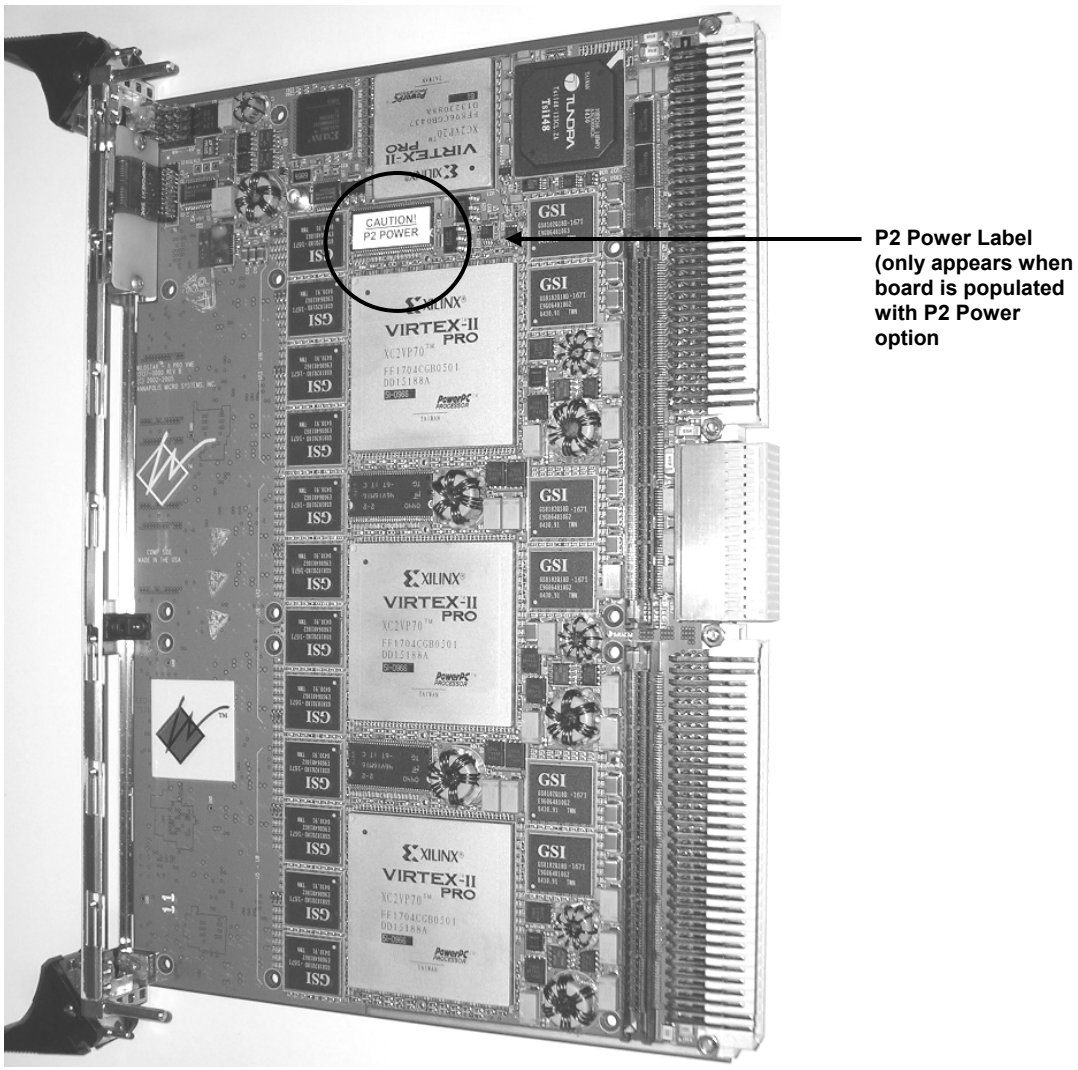


Figure 4-2: Label Indicating WILDSTAR™-II PRO/VME with P2 Power Option (Component Side)

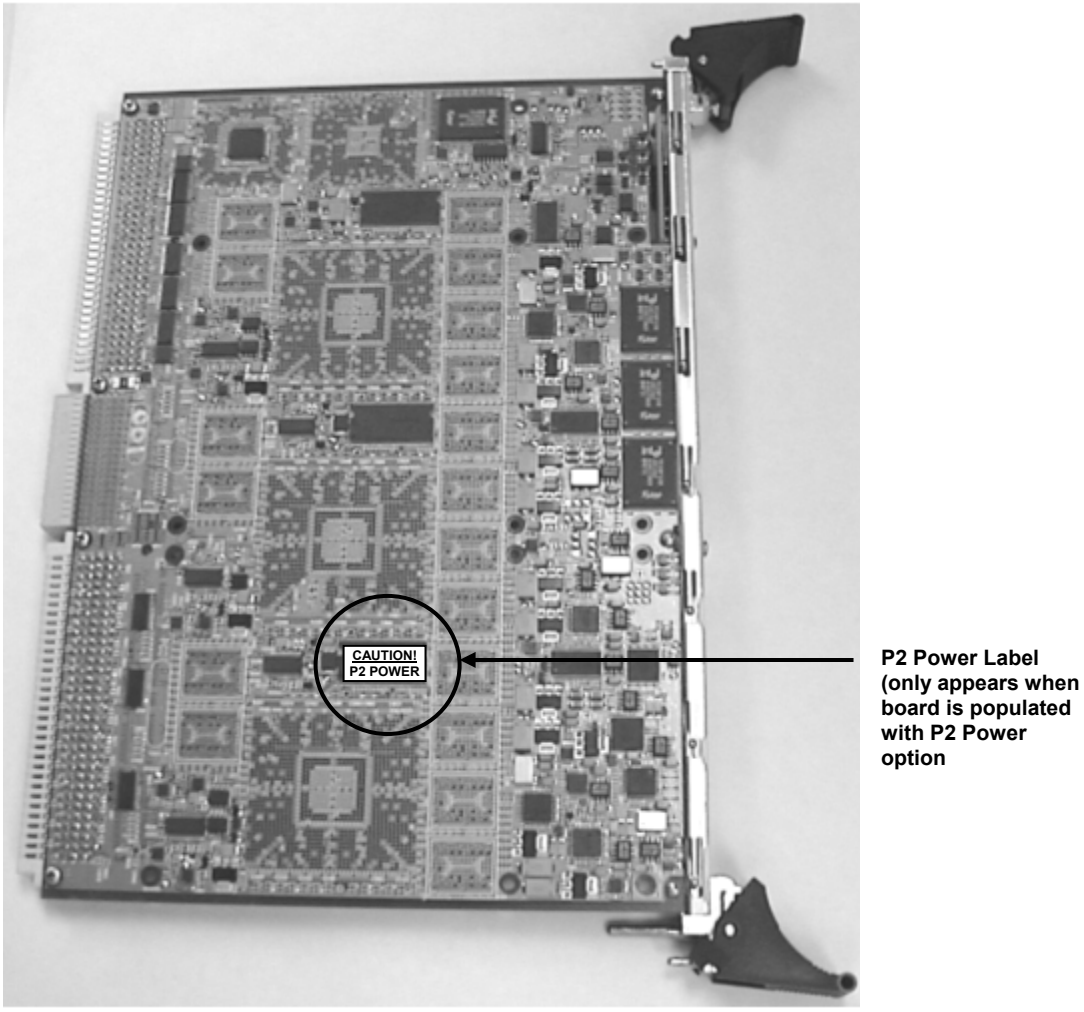


Figure 4-3: Label Indicating WILDSTAR™-II PRO/VME with P2 Power Option (Solder Side)

4.2.1 WILDSTAR™-II PRO / VME Jumper Settings

Jumper settings for the WILDSTAR™-II PRO / VME board are listed below. (See separate section for PRO ACE for VME settings.)

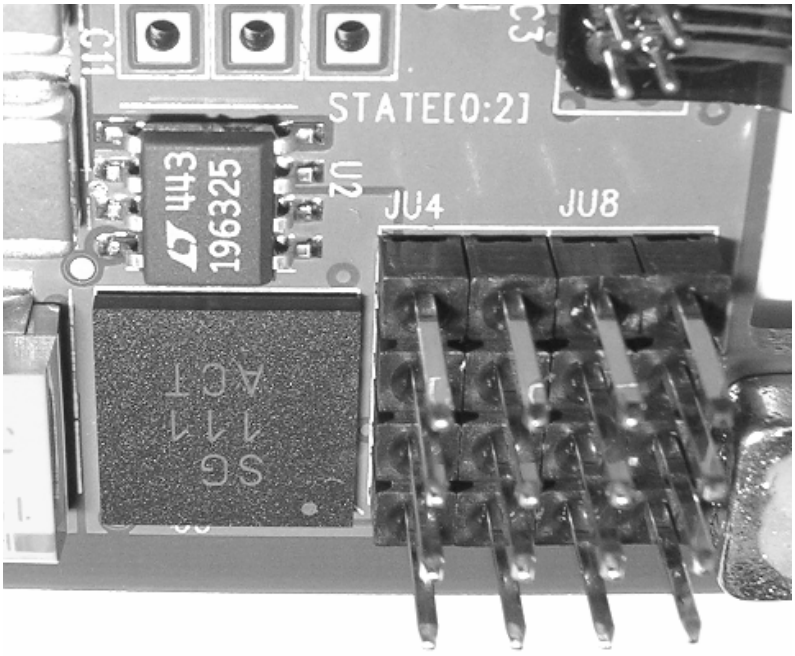


Figure 4-4: Jumper Detail for WILDSTAR™-II PRO/VME Board (Shown without Connecting Shunts)

Table 4-7: WILDSTAR™-II PRO/ VME Board Jumper Settings

Position No.	Jumper Name	Jumper Function Description	Default Setting
JU1	Flash Backup	Installed: Use Flash backup Uninstalled: Normal Flash operation	OFF
JU2	PLDJTAG CONN/~PCI	Installed: Drives PLD JTAG chain from connector or PCI controller.	OFF
JU3	PCI Flash Protect	Installed: PCI Controller is Flash protected Uninstalled: PCI Controller Flash Update is enabled	OFF
JU4	CHIPSCOPE	Installed: Selects PE JTAG chain	OFF
JU5	PE Flash Protect	Installed: PE Flash is protected Uninstalled: PE Flash can be written	OFF
JU6	VME Reset_In Enable	Installed: WILDSTAR-II / VME is reset by VME bus reset. Uninstalled: Board ignores VME bus reset.	OFF
JU7	VME Reset_Out Enable	Installed: WILDSTAR-II / VME can cause VME bus resets. Uninstalled: Board cannot cause a VME bus reset.	OFF
JU8	Board Reset Select *(via front panel button or software)*	When installed and reset, all PEs and I/O cards are reset. User still has access to board. When uninstalled and reset, PCI, VME-PCI bridge, PEs, and I/O cards are all reset. User loses access to board and must re-open it.	OFF

4.2.2 Hardware Installation for WILDSTAR™-II PRO/ VME

Follow the steps below to install the WILDSTAR™-II PRO/ VME board into the host system.

1. Ground yourself of static electricity with a groundstrap.

2. Shut down the host system and power off.



CAUTION
The WILDSTAR™-II PRO/VME board must be inserted into the chassis gently to avoid scraping fragile components on the edges of the board.

3. Grasp the WILDSTAR™-II PRO/ VME board by the front panel ejectors with the VME connectors facing away from the user.
4. Gently slide the WILDSTAR™-II PRO/ VME board into the slot.
5. When the VME connectors first contact the backplane connectors, push firmly on the front panel ejectors until they snap into place. Engage the front panel ejector tabs with the chassis rails to fully seat WILDSTAR™-II PRO/ VME in the VME backplane.

To remove the VME board, connect a ground strap to the user and power off the host system. Gently press the red ejector release buttons, press down on the front panel ejectors, and slowly pull the board from the chassis.

4.2.3 Power Up System

After all cable connections have been made and the board has been properly plugged in, turn on power to the host system.

Once the WILDSTAR™-II PRO /VME board has been installed in the host system and the host machine has been powered up, the host software and VHDL models can be installed.



INFORMATION NOTE
Instructions for installing host software on WILDSTAR™-II PRO boards can be found in the release notes accompanying the board.

4.3 WILDSTAR™-II PRO ACE for VME Board Installation

Instructions appear below for installing the WILDSTAR™-II PRO ACE for VME board in a VME64x chassis.

4.3.1 WILDSTAR™-II PRO ACE for VME Jumper Settings

Jumpers on the solder side of the WILDSTAR™-II PRO ACE for VME board are used to set basic configuration options.

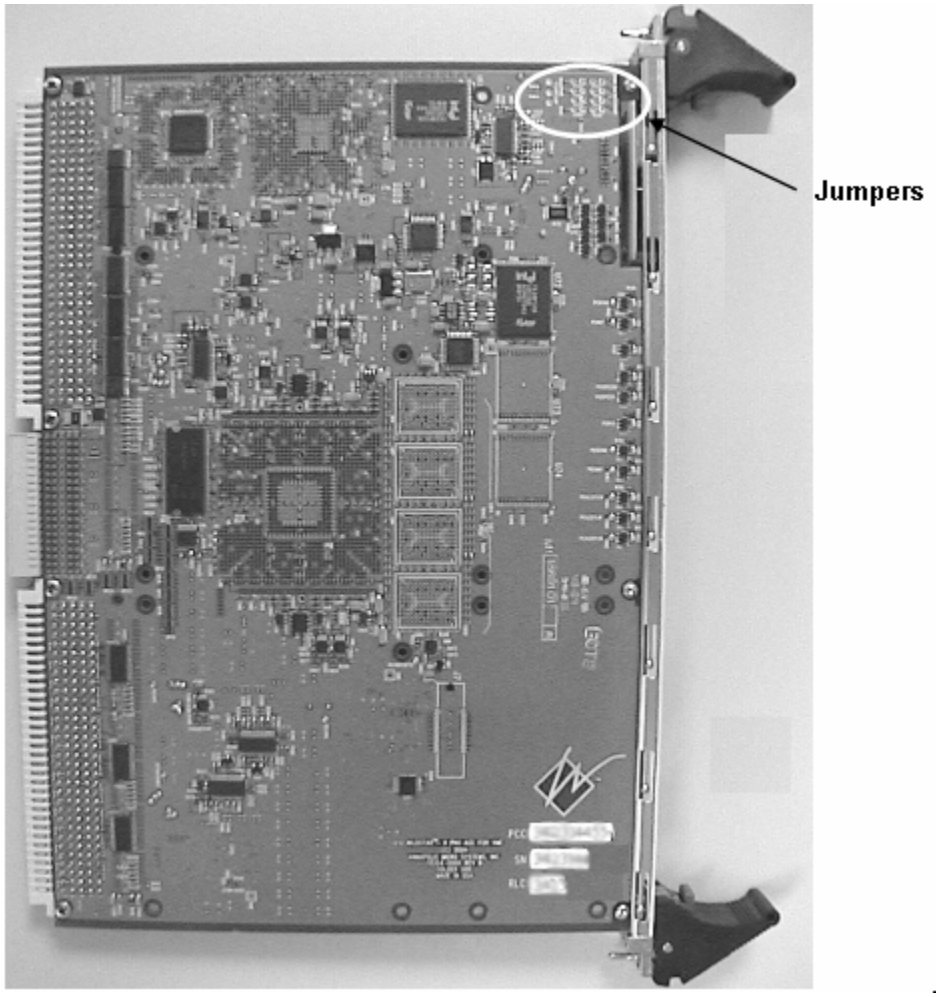


Figure 4-5: Jumpers on WILDSTAR™-II PRO ACE for VME (Solder Side of Board)

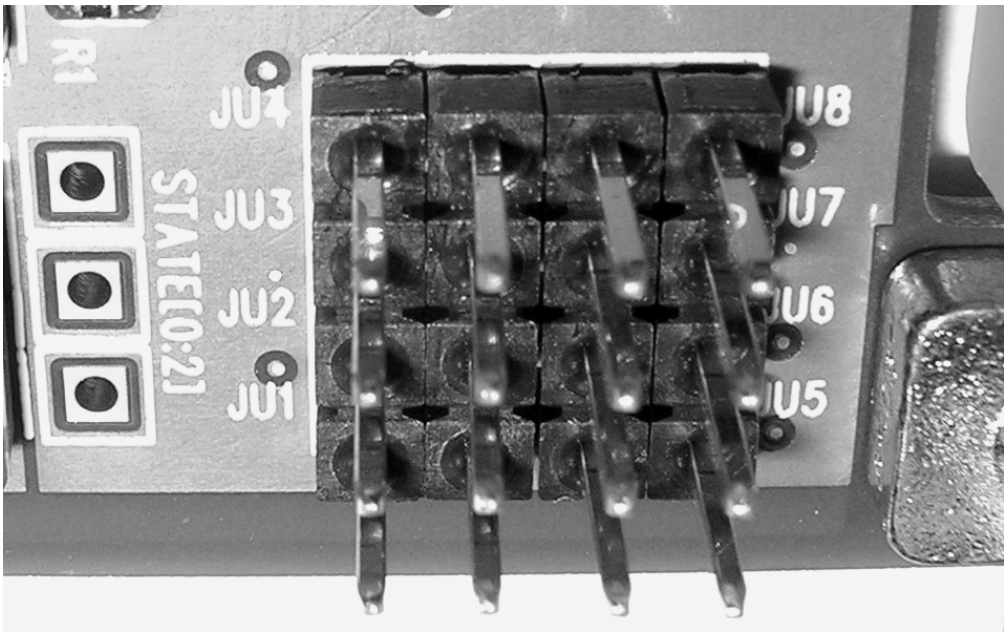


Figure 4-6: Jumper Detail for WILDSTAR™-II PRO ACE for VME Board (Rev. B, Shown without Connecting Shunts)

Table 4-8: WILDSTAR™-II PRO ACE for VME Board Jumper Settings

Position No.	Jumper Name	Jumper Function Description	Default Setting
JU1	Flash Backup	Installed: Use Flash backup Uninstalled: Normal Flash operation	OFF
JU2	PLDJTAG CONN/~PCI	Installed: Drives PLD JTAG chain from connector or PCI controller.	OFF
JU3	PCI Flash Protect	Installed: PCI Controller is Flash protected Uninstalled: PCI Controller Flash Update is enabled	OFF
JU4	CHIPSCOPE	Installed: Selects PE JTAG chain	OFF
JU5	PE Flash Protect	Installed: PE Flash is protected Uninstalled: PE Flash can be written	OFF
JU6	VME Reset_In Enable	Installed: WILDSTAR-II / VME is reset by VME bus reset. Uninstalled: Board ignores VME bus reset.	OFF
JU7	VME Reset_Out Enable	Installed: WILDSTAR-II / VME can cause VME bus resets. Uninstalled: Board cannot cause a VME bus reset.	OFF
JU8	Board Reset Select *(via front panel button or software)*	When installed and reset, all PEs and I/O cards are reset. User still has access to board. When uninstalled and reset, PCI, VME-PCI bridge, PEs, and I/O cards are all reset. User loses access to board and must re-open it.	OFF

INFORMATION NOTE

Setting the slot ID through the switches is *not supported* on WILDSTAR™-II PRO ACE for VME boards.

4.3.2 Hardware Installation for WILDSTAR™-II PRO ACE for VME

Follow the steps below to install the WILDSTAR™-II PRO ACE for VME into the host system.

1. Connect the ground strap to the user.
2. Shut down the host system and power off.

CAUTION

The WILDSTAR™-II PRO ACE for VME board must be inserted into the chassis gently to avoid scraping fragile components on the edges of the board.

3. Grasp the WILDSTAR™-II PRO ACE board by the front panel ejectors with the VME connectors facing away from the user.
4. Gently slide the WILDSTAR™-II PRO ACE board into the slot.
5. When the VME connectors first contact the backplane connectors, push firmly on the front panel ejectors until they snap into place. Engage the front panel ejector tabs with the chassis rails to fully seat WILDSTAR™-II PRO ACE in the VME backplane.

To remove the VME board, connect a ground strap to the user and power off the host system. Gently press the red ejector release buttons, press down on the front panel ejectors, and slowly pull the board from the chassis.

4.4 WILDSTAR™-II /PCI Board Installation

This section contains steps and suggestions for installing the WILDSTAR™-II /PCI motherboard. The PRO-series board can also be installed using these instructions.

Switch settings for the WILDSTAR™-II/PCI and PCI PRO-Series are described in separate sections following these installation instructions.

4.4.1 Hardware Installation for WILDSTAR™-II /PCI

INFORMATION NOTE

WILDSTAR™-II PRO /PCI motherboards can *only* be placed in 3.3V PCI slots.

Follow the steps below to install the WILDSTAR™-II /PCI board into the host system.

1. To prevent static electricity damage to the board, ensure that you are wearing a properly connected ground strap.
2. Shut down the host system and power off.
3. Remove the chassis cover to access the motherboard's PCI connectors.
4. For the available slot, remove the rear access panel (Figure 4-7).



Figure 4-7: Rear Access Panel

INFORMATION NOTE

If you plan on using two I/O cards with the WILDSTAR™-II /PCI, the second I/O card slot requires the use of the WILDSTAR™-II /PCI Second I/O Slot Adapter. Since this adapter requires its own chassis slot, ensure that this additional slot will be available before installing the WILDSTAR™-II /PCI motherboard.

INFORMATION NOTE

WILDSTAR™-II PRO/PCI motherboards can only be placed in 3.3V PCI slots.

CAUTION

The WILDSTAR™-II /PCI board uses the 64-bit PCI edge finger configuration. When the board is placed in a 32-bit PCI slot, the 64-bit edge finger extension will hang over the 32-bit connector socket. Be sure that the 64-bit edge finger extension does not touch any components on the motherboard for risk of possible electrical shortage or mechanical damage.

5. Insert the WILDSTAR™-II /PCI board into the PCI slot. Make sure that the rear-mounting bracket is lined up with the rear access panel.
6. Confirm that no two boards have any touching components.
7. Screw down the rear access panel of the WILDSTAR™-II /PCI board.
8. Replace the cover to the host system chassis.

Before removing any board, ensure that you are grounded of static electricity and that the host computer is powered off.

4.4.2 Power Up System

After all cable connections have been made and the board has been properly plugged in, turn on power to the host system.

Once the WILDSTAR™-II /PCI board has been installed in the host system and the host machine has been powered up, the WILDSTAR™-II host software and VHDL models can be installed.

INFORMATION NOTE

Instructions for installing host software on WILDSTAR™-II boards can be found in the release notes accompanying the board, as well as in .pdf format on the documentation CD.

INFORMATION NOTE

The WILDSTAR™-II /PCI board (including PRO-series) does not have a built-in 3.3V power supply and therefore assumes 3.3V power is available from the chassis backplane.

INFORMATION NOTE

The WILDSTAR™-II /PCI board is fitted with universal edge-fingers, capable of fitting into a 3.3V or 5V slot only. The WILDSTAR™-II PRO/PCI board should only be placed in a 3.3V slot.

4.4.3 WILDSTAR™-II /PCI Switch Settings

A switch on the back of each WILDSTAR™-II /PCI board controls basic configurations (Figure 4-8). The Rev. C board is equipped with two switches.

WILDSTAR™-II /PCI boards are delivered with all switches in their default settings. Confirm that switches are set according to the information in the tables below:

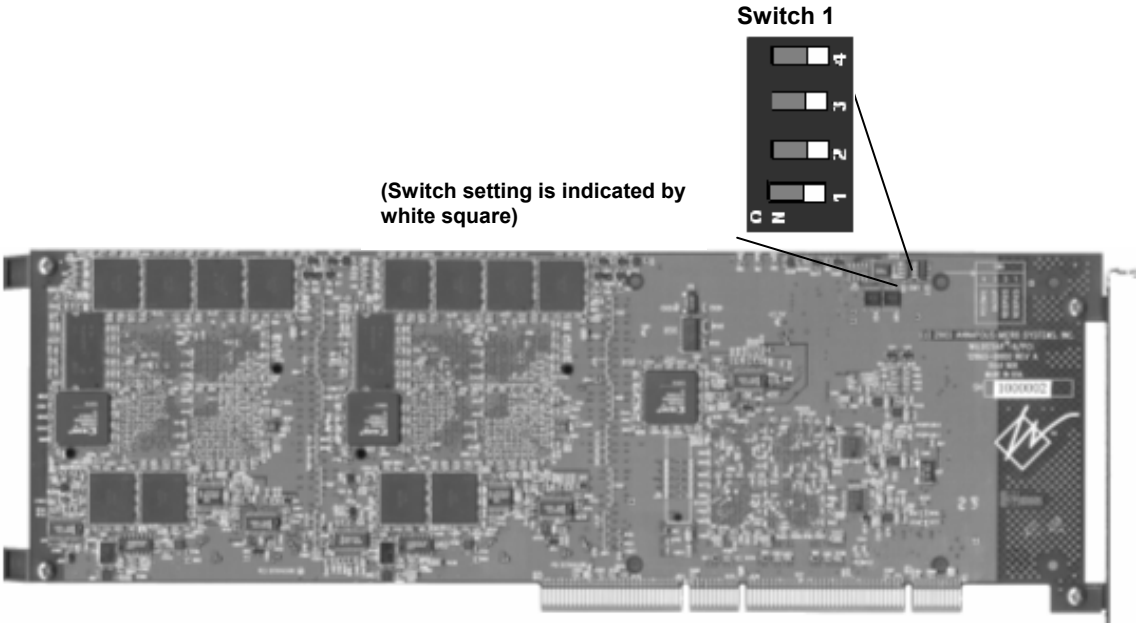


Figure 4-8: Location of WILDSTAR™-II /PCI Switch 1 (SW1), in Default Positions (Solder Side of Board)

Table 4-9: WILDSTAR™-II /PCI Board (All Revs.) Switch 1 Settings

Position No.	Position Name	Switch Description	Default Setting
1	Reserved	Leave in OFF position	OFF
2	Reserved	Leave in OFF position	OFF
3	Flash Protect (PCI and PE for Rev. B; PCI only for Rev. C)	When ON, Flash is Protected. When OFF, Flash is writeable.	OFF
4	33 MHz Enabled	When OFF, allows for 66 MHz Bus operation. When ON, switches PCI Bus to 33 MHz.	OFF

Table 4-10: WILDSTAR™-II /PCI Board (Rev. C, D) Switch 2 Settings

Position No.	Position Name	Switch Description	Default Setting
1	Reserved	Leave in ON position	ON
2	Reserved	Leave in ON position	ON
3	Reserved	Leave in ON position	ON
4	PE PLD Flash Protect	When ON, Flash is protected. When OFF, Flash is writeable.	OFF

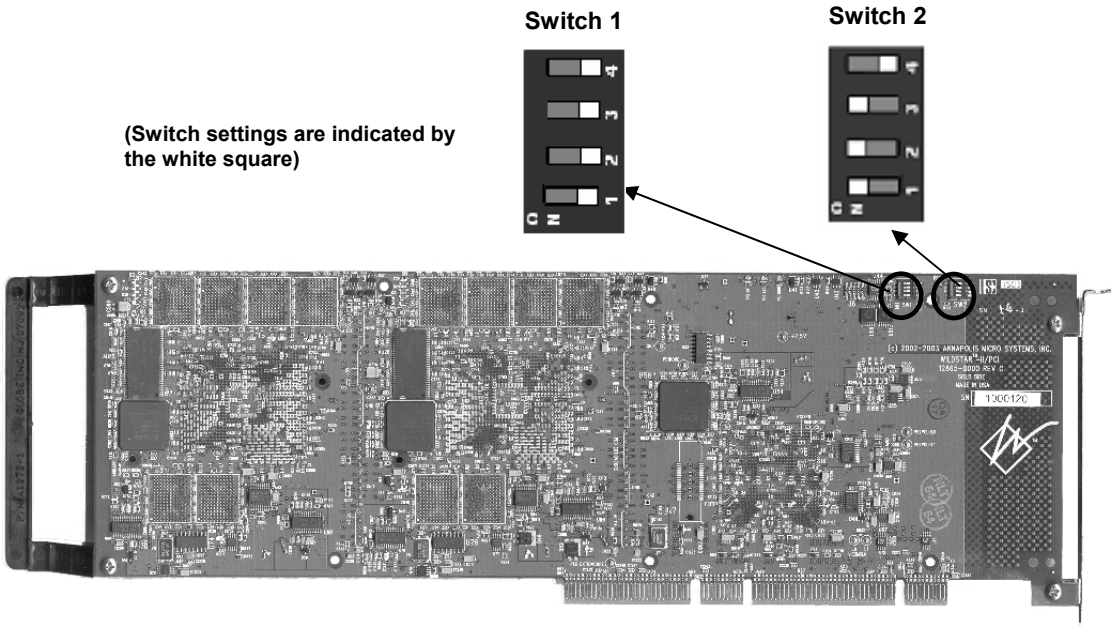


Figure 4-9: Location of WILDSTAR™-II /PCI Rev. C, D Switches (SW1, SW2), in Default Positions (Solder Side of Board)

4.4.4 WILDSTAR™-II PRO/PCI Switch Settings

Two switches on the back of each WILDSTAR™-II PRO /PCI board control basic configurations.

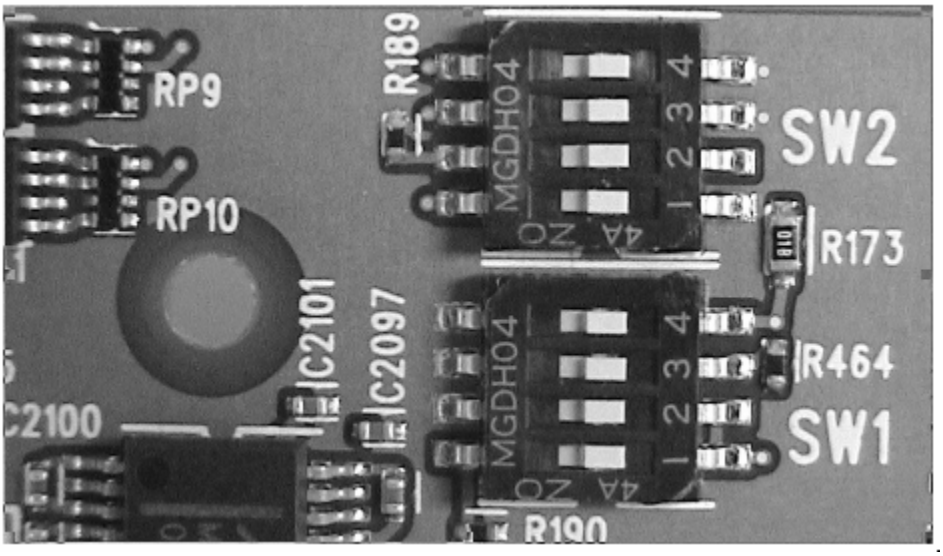


Figure 4-10: Location of WILDSTAR™-II PRO /PCI Switches (SW1, SW2), in Default Positions, all Revs. (Solder Side of Board)

Boards are delivered with all switches in their default settings. Confirm that switches are set according to the information in the tables below:

Table 4-11: WILDSTAR™-II PRO/PCI Board Switch 1 Settings

Switch 1 Position	Name	ON (Closed)	OFF (Open)	Default Position
1	PCI-X Bus Speed	Allow bus speed of 50-66MHz when on PCI-X bus	Allow bus speed of up to 133MHz when on PCI-X bus	OFF
2	PCI-X	Restrict bus to Conventional PCI	Allow the use of PCI-X on capable busses	OFF
3	Reserved			OFF
4	PE FLASH-PROTECT#	Do not permit PE Flash access.	Allow unrestricted access to PE Flash	OFF

Table 4-12: WILDSTAR™-II PRO/PCI Board Switch 2 Settings

Switch 2 Position	Name	ON (Closed)	OFF (Open)	Default Position
1	BACKUP	Use previous or original shipping PCI Controller version.	Use latest PCI Controller.	OFF
2	Reserved	Leave in OFF position		OFF
3	PCI FLASH-PROTECT#	Flash is protected	Flash is writeable	OFF
4	M66EN : Conventional PCI Bus Speed	Restrict Conventional PCI bus speed to 33MHz max.	Allow 66MHz Conventional PCI bus operation on busses that support 66MHz.	OFF

4.5 WILDSTAR™-II Reset Options

WILDSTAR™-II and WILDSTAR™-II PRO boards are equipped with a number of user-controlled reset options at the PE, board, and system level, each of which is explained below.

The following instructions apply to both WILDSTAR™-II and WILDSTAR™-II PRO series.

4.5.1 PE Resets

A reset pin on each mainboard PE (PE0 and PE1 on the PCI board; PE0, PE1 and PE2 on the VME board) can be set, cleared or pulsed using the following API call (described in the *WILDSTAR™-II Software Reference Manual*:

```
WSII_PeReset (WSII_BOARD hBoard, DWORD dPeMask,
WSII_RESET_OP ResetOperation);
```

This reset resets only the PE or PEs in the mask and will not disrupt any other board functionality.

4.5.2 Board Resets

One option exists for board resetting on a PCI board, while VME has two board reset options. On both board types, a board level reset can be done using the following API call:

```
WSII_RetCode WSII_BoardReset (WSII_BOARD hBoard);
```

This reset option causes all initial power-on conditions to be restored (for example, initial clock frequencies and PE programming on startup). All PEs will be deprogrammed (both mainboard and any IO card PEs) during this reset event.

The second option for a VME board reset is selected by setting the “board reset switch” to “OFF”, which is the default setting. This mode resets everything described in the API board reset mode and additionally resets the local PCI bus on the WILDSTAR™-II / VME and the VME-PCI bridge. This reset causes the host to lose access to the board until it is reopened, since all the setup for opening the board has been cleared. When the board is reset in this manner, it is as if power has been cycled to that board, with the VME slot ID being redisplayed on the front panel display.

4.5.3 System Resets

Below are details for system resets on the WILDSTAR™-II boards (includes PRO-series).

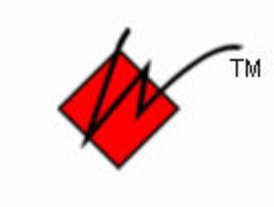
WILDSTAR™-II / PCI and PRO / PCI:

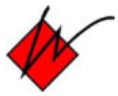
A reset from the PCI bus will cause a reset of the PCI controller on the WILDSTAR™-II / PCI, but will not deprogram the PEs. The board will need to be reopened after a PCI reset.

WILDSTAR™-II / VME and PRO / VME:

The WILDSTAR™-II / VME board can be configured to accept or ignore VME bus resets, or to cause a VME reset. By default, the board can receive and send VME resets from the VME bus.

When a VME bus reset is received, both the VME-PCI bridge and the PCI controller are reset because the local PCI bus on the WILDSTAR™-II / VME board is reset. This reset causes no PEs to be deprogrammed, but to have host access the board will need to be reopened. The VME slot ID will be redisplayed on the front panel display.


TM



5 TECHNICAL SUPPORT

If you have any questions about installing, programming, using, or maintaining your WILDSTAR™-II board, please call the WILDSTAR™ Technical Support team at (410) 841-2514, fax at (410) 841-2518, or send e-mail to wftech@annapmicro.com. Our web site address is http://www.annapmicro.com.

The suggestions listed below will help us respond to your questions more quickly.

5.1 Board Identification Numbers

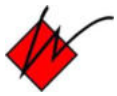
Each Annapolis Micro Systems board is prominently labeled with three unique codes: the *Product Configuration Code* (PCC), the *Serial Number* (SN), and the *Revision Level Code* (RLC). You can also find these codes by installing the board and running *wsinst.exe* from the host software CD.

- The **Product Configuration Code (PCC)** identifies PE type, memories, clocking, and other options selected for the particular board.
- The **Serial Number (SN)** is a unique number identifying each board.
- The **Revision Level Code (RLC)** includes information about revisions and engineering modifications made to the board.

When contacting Annapolis Micro Systems with board-related questions, please include these codes in your query, as well as the information listed below:

- Board operating system
- Host software version
- Host platform
- Host OS
- VHDL version number
- API version
- Driver versions
- CoreFire version being used (if applicable)


TM



6 WILDSTAR™-II HARDWARE REFERENCE

This chapter contains hardware reference information for the WILDSTAR™-II board, including thermal management, clocking, and backplane specifications.

6.1 WILDSTAR™-II Board Specifications

Table 6-1 specifies the physical dimensions and operating range for all WILDSTAR™-II board types:

Table 6-1: WILDSTAR™-II Specifications

WILDSTAR™-II /VME, PRO/VME, PRO ACE for VME	
Physical Dimensions:	Length: 233.35 mm/ 9.187 in Width: 160.00 mm/ 6.299 in Thickness: 1.57 mm/0.062 in
Operating Range:	Temperature: 0º to 70ºC

WILDSTAR™-II /PCI, WILDSTAR™-II PRO/PCI	
Physical Dimensions:	Length: 312.00 mm/12.283 in Width: 106.68 mm/4.200 in Thickness: 1.57 mm/0.062 in Weight: 3.9 oz.
Operating Range:	Temperature: 0º to 70ºC

6.2 WILDSTAR™-II Thermal Management

To minimize heating caused by high clock rates and power levels, WILDSTAR™-II boards have been equipped with special features that monitor and reduce heat buildup.



CAUTION
The Virtex™-II PE junction temperatures on all WILDSTAR™-II board types should *never* exceed 85˚C. Temperatures above this range may result in performance degradation and component damage.

The PCI Controller and PEs on each board contain devices that monitor the junction temperatures of these components. Additional sensing diodes are located at various board locations to monitor ambient temperature (sensing diodes are on non-PRO-series boards only).

Active heat remediation for the WILDSTAR™-II (and PRO-series) PCI includes a combination heat sink/fan with software and hardware controls. When an installed PE's T_j, or junction temperature, reaches 50˚C, the onboard PCI Controller defaults to turn on the fan automatically. (To protect the PE, the temperature has to be below 50˚C for some period before the fan actually shuts off.) A heat sink for PEs is an available option for the WILDSTAR™-II / VME and /cPCI boards.

CAUTION

If a WILDSTAR™-II board fan fails, it must promptly be replaced at the factory to avoid heat damage to the board.

CAUTION

To prevent system damage, fire, and personal injury, *never* use WILDSTAR™-II (and PRO-series) PCI boards without the heat sink fan installed and operational. In addition, the operational PCI chassis should have adequate cooling and airflow to ensure a $T_j < 85$˚C temperature during application operation. Temperatures above this range may result in performance degradation and component damage.

CAUTION

To prevent system damage, fire, and personal injury, the WILDSTAR™-II /VME board should have adequate cooling and airflow to ensure a $T_j < 85$˚C temperature during application operation. Temperatures above this range may result in performance degradation and component damage.

The following host software APIs provide a means for monitoring and controlling board temperature:

- **WSII_GetTemperature**: Returns the temperature, in Celsius, of a geographic location on the WILDSTAR™-II board or temperature of a specific Processing Element.
- **WSII_GetTemperatureThresh**: Returns the temperature, in Celsius, at which the temperature monitor will generate an interrupt and/or deprogram the part.
- **WSII_SetTemperatureThresh**: Sets the threshold used by the temperature monitor to determine when a part will be deprogrammed or warning event generated.

- **WSII_QueryTemperatureEvents**: Provides a map describing any temperature events that have occurred.
- **WSII_ResetTemperatureEvents**: Resets temperature events after they have occurred.
- **WSII_PeSetFanMode**: Sets the fan in one of two operating modes: WSII_FAN_ON or WSII_FAN_OFF. If WSII_FAN_ON is selected, the cooling fan will run constantly. If WSII_FAN_OFF is selected, the fan will only run when the on- board temperature exceeds 50 degrees Centigrade.

Details of these APIs can be found in the *WILDSTAR™-II Software Reference Manual*.

Figure 6-1 shows temperature monitoring points on the WILDSTAR™-II /VME board:

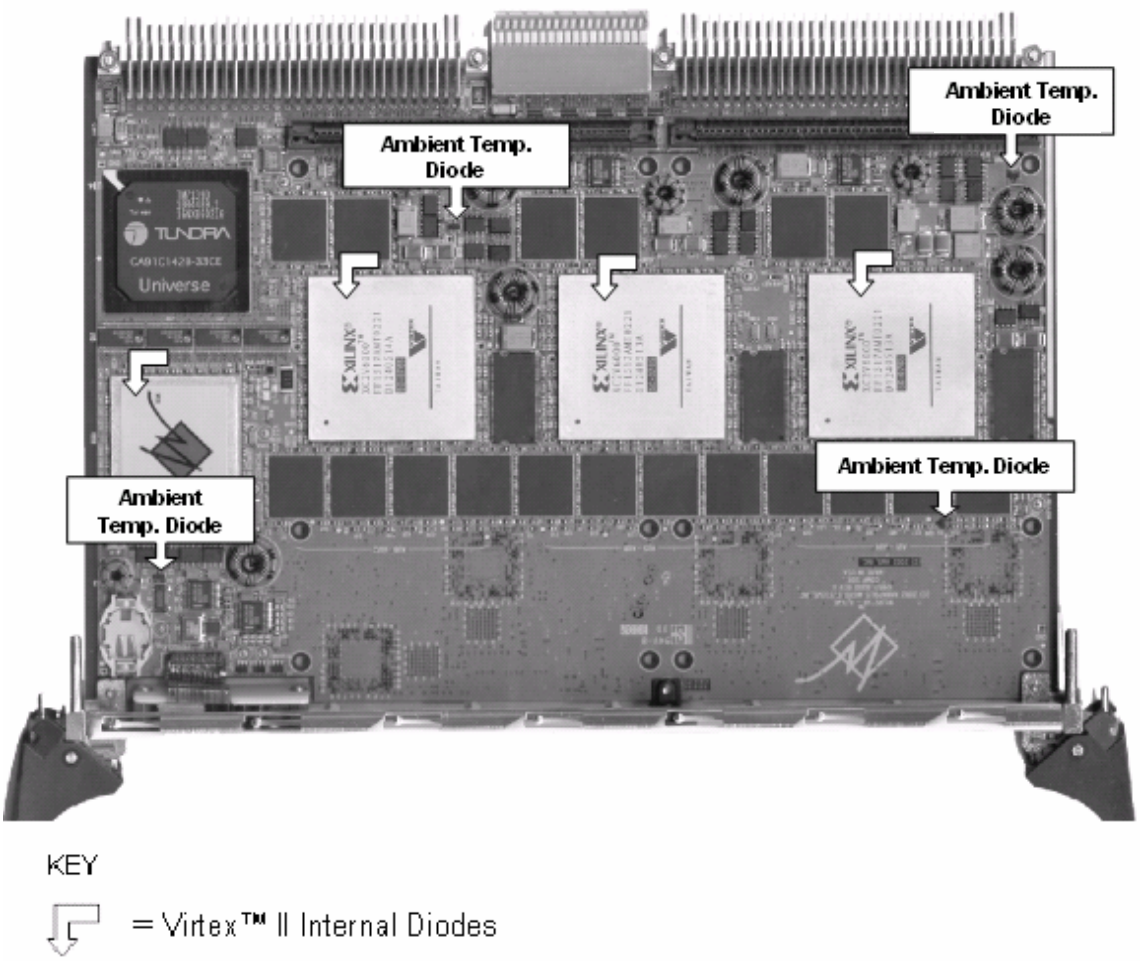
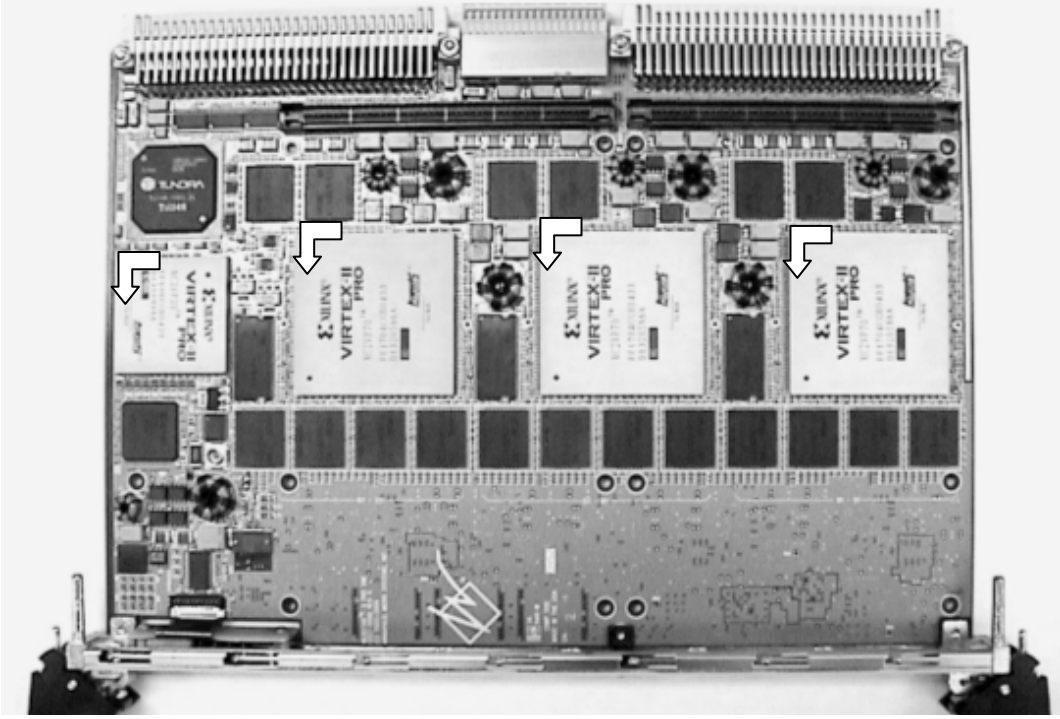
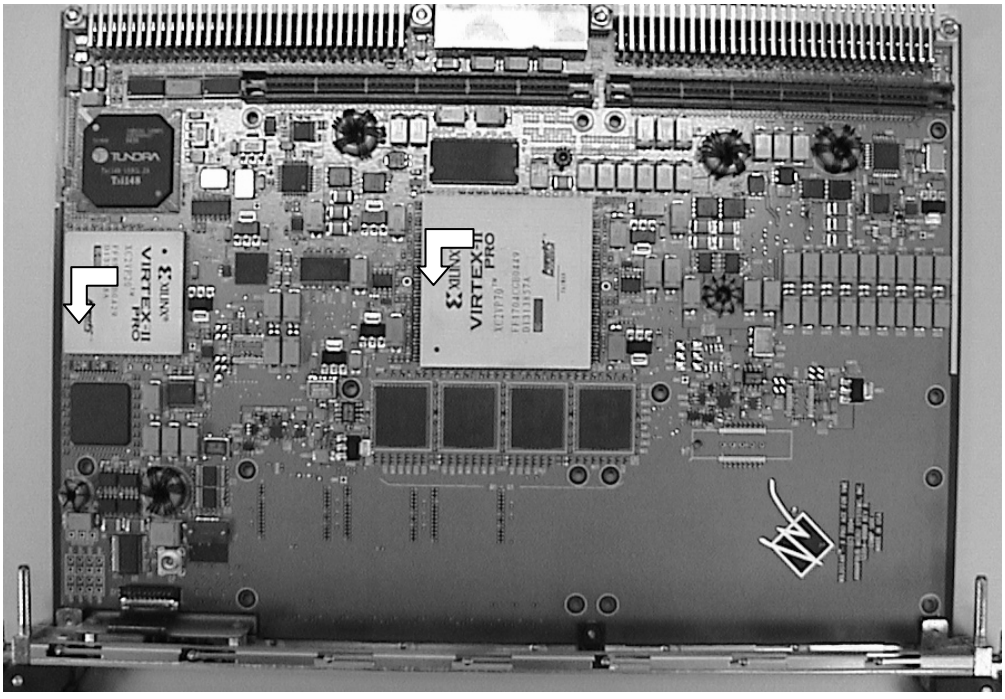


Figure 6-1: WILDSTAR™-II / VME Temperature Monitoring Points



= VIRTEX™-II Pro Internal Diodes

Figure 6-2: WILDSTAR™-II PRO/VME Temperature Monitoring Points



= VIRTEX™-II Pro Internal Diodes

Figure 6-3: WILDSTAR™-II PRO VME for ACE Temperature Monitoring Points

The illustrations below show temperature monitoring points on the WILDSTAR™-II / PCI and PRO/PCI boards:

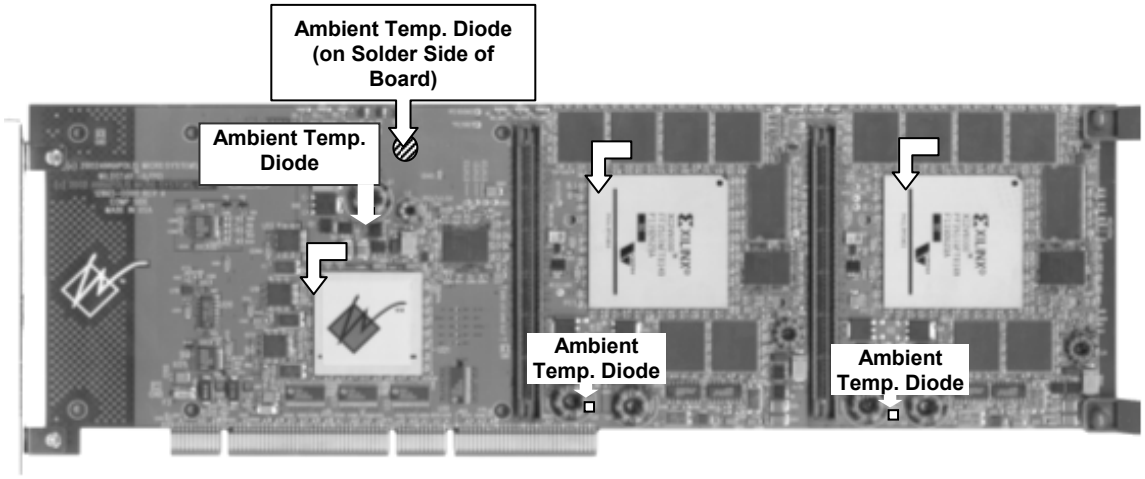
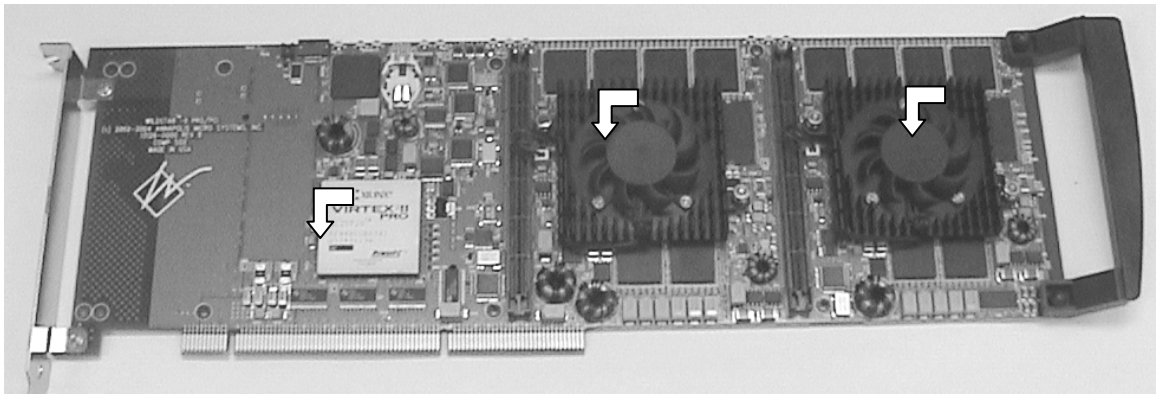


Figure 6-4: WILDSTAR™-II / PCI Temperature Monitoring Points (Revs. A-C; non-PRO-Series Boards only)



= VIRTEX™-II Pro Internal Diodes

Figure 6-5: WILDSTAR™-II PRO/PCI Temperature Monitoring Points

6.3 WILDSTAR™-II Power Functions

This section describes WILDSTAR™-II power monitoring, as well as special power supply options for the WILDSTAR™-II /VME Rev. C board. (Power options for PRO-series boards are discussed in a subsequent section of this chapter.)

6.3.1 Power Monitoring and Thresholds for WILDSTAR™-II / VME and / PCI



WARNING

To prevent system damage, fire, and personal injury, *never* use the WILDSTAR™-II /PCI board without the heat sink fan installed. In addition, the operational PCI chassis should have adequate cooling and airflow to ensure a $T_j < 85^{\circ}C$ temperature during application operation. Temperatures above this range may result in performance degradation and component damage.

The power consumed by the WILDSTAR™-II board directly correlates to the total number of flip-flops in the design and the frequency at which it is run. It is important to consider the power consumption of each target application so that the restrictions of the target chassis can be met with respect to available power and adequate airflow.

Figure 6-3 shows the power monitoring capabilities of the WILDSTAR™-II /VME and /PCI boards, respectively. Included here are features for PE and non-PE power sensing. On the WILDSTAR™-II /PCI board, the PCI chip turns on the fan automatically when PE junction temperatures reach 50 C. This threshold can be modified using software. (Refer to the *WILDSTAR™-II Software Reference* for API calls providing access to this information.)

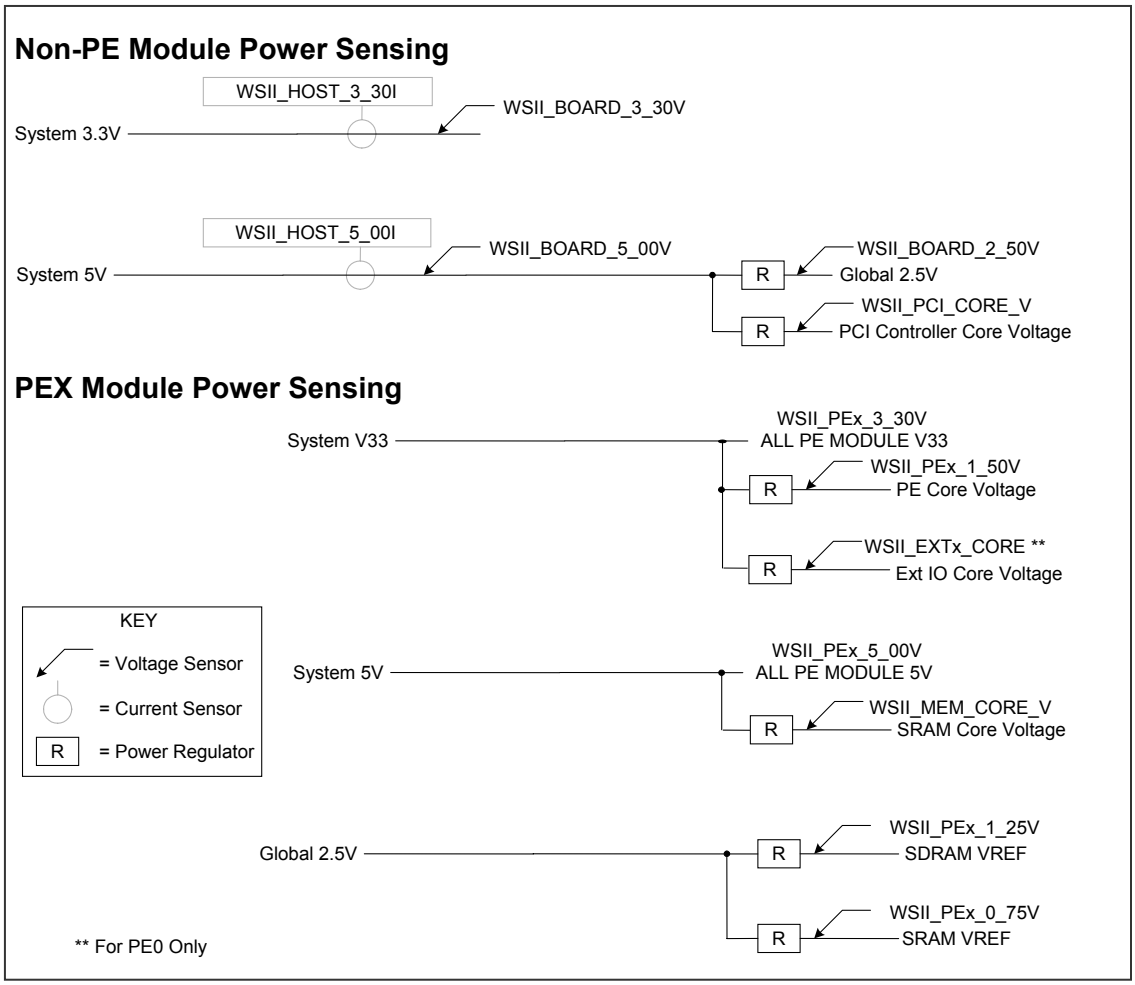


Figure 6-6: WILDSTAR™-II / VME and /PCI Power Monitoring

CAUTION

WILDSTAR™-II /VME Users:

The total power per slot should not exceed **120** watts (70 watts of 5V power, 50 watts of 3.3V power). The power consumption for an application must not exceed the recommended watts per slot. Ensure that the chassis is capable of supplying adequate power and cooling to run the application under conditions up to the maximum operating condition.

CAUTION

WILDSTAR™-II /PCI Users (all revs):

According to PCI Local Bus Revision 2.2 specifications, the total power should not exceed **25** watts per slot. If the power consumption for an application exceeds 25 watts per slot, the user must ensure that the chassis is capable of supplying adequate power and cooling to run the application.

6.3.2 WILDSTAR™-II /VME 3.3V Power Supply Options (Rev. C only)

WILDSTAR™-II /VME boards can be equipped with an optional 3.3V power supply. Installation of this supply allows the board to operate in a legacy VME chassis.

INFORMATION NOTE

Installing the 3.3V power supply limits the board power consumption to 70 watts.

6.4 WILDSTAR™-II PRO Power Functions

WILDSTAR™-II PRO boards are capable of measuring voltage and current values for 3V and 5V, allowing the total board power to be calculated. The board is also equipped with over/under voltage monitors watched by the PCI controller. The status of these monitors can be checked via the API.

Power measurement locations and ranges for PRO/VME and PRO ACE for VME are noted in Figure 6-7. Similar information for PRO/PCI is located in Figure 6-8.

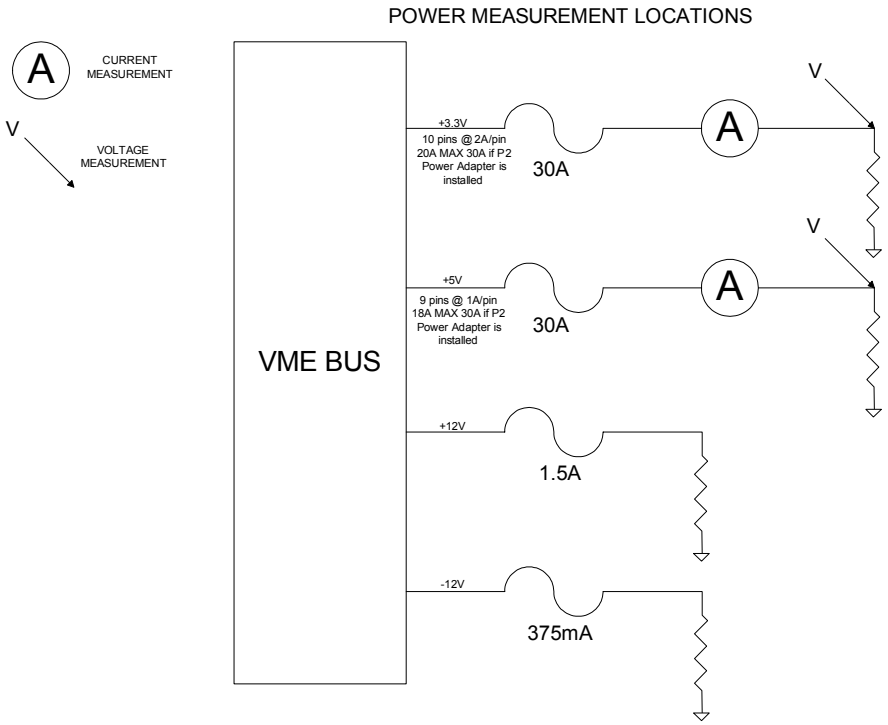


Figure 6-7: Power Measurement and Monitoring for WILDSTAR™-II PRO/VME and PRO ACE for VME

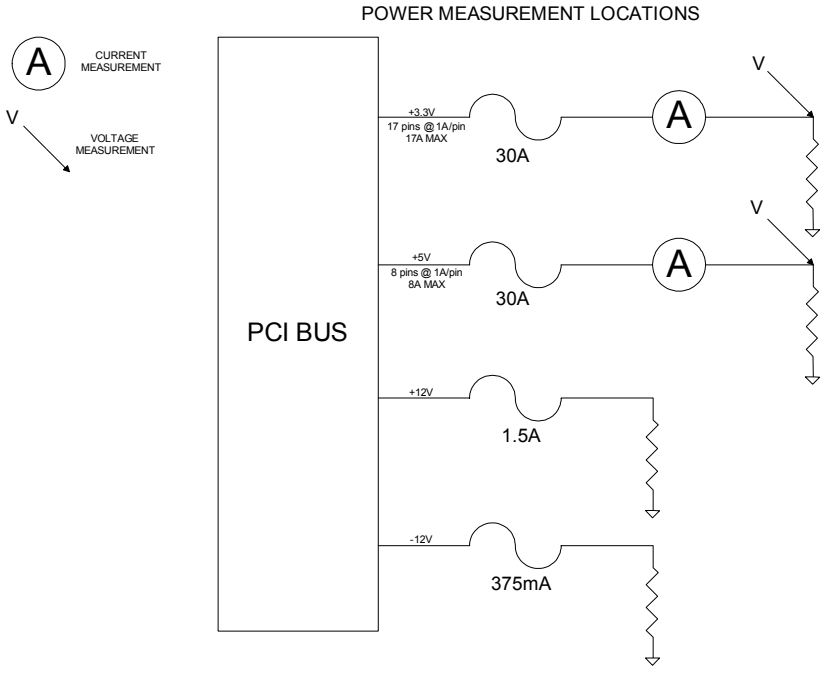


Figure 6-8: Power Measurement and Monitoring for WILDSTAR™-II PRO/PCI

6.5 WILDSTAR™-II Board Clocks

WILDSTAR™-II boards have two types of clocks: the global board clocks MCLK, PCLK, and ICLK; and local clocks for each PE, consisting of ACLK, BCLK, and CCLK. MCLK and PCLK are differential, while ICLK is fixed and single-ended. PE local clocks are also single-ended and individually configurable.

INFORMATION NOTE

It is strongly encouraged that you use a single clock in a PE design. Use of multiple clocks within a single PE may cause severe clock jitter and application failure. PCLK is recommended for most processing tasks.

An optional 1 pps time base clock input is available on WILDSTAR™-II /PCI (Rev. B only) and /VME boards. This option attaches the clock input to a global clock buffer on one of the PEs instead of an I/O board clock.

The optional 1PPS input on VME is a 5V signal terminated with a 50 Ohm resistor to GND. The input specs are 4.6V for Input Voltage High Threshold (VIHC) and 0.2V for Input Voltage Low Threshold (VILC). For other clock frequencies and destinations, please see Section 6.5.2.

6.5.1 Clock Distribution Diagrams

Below are diagrams detailing clock distribution signals on various PCI and VME boards.

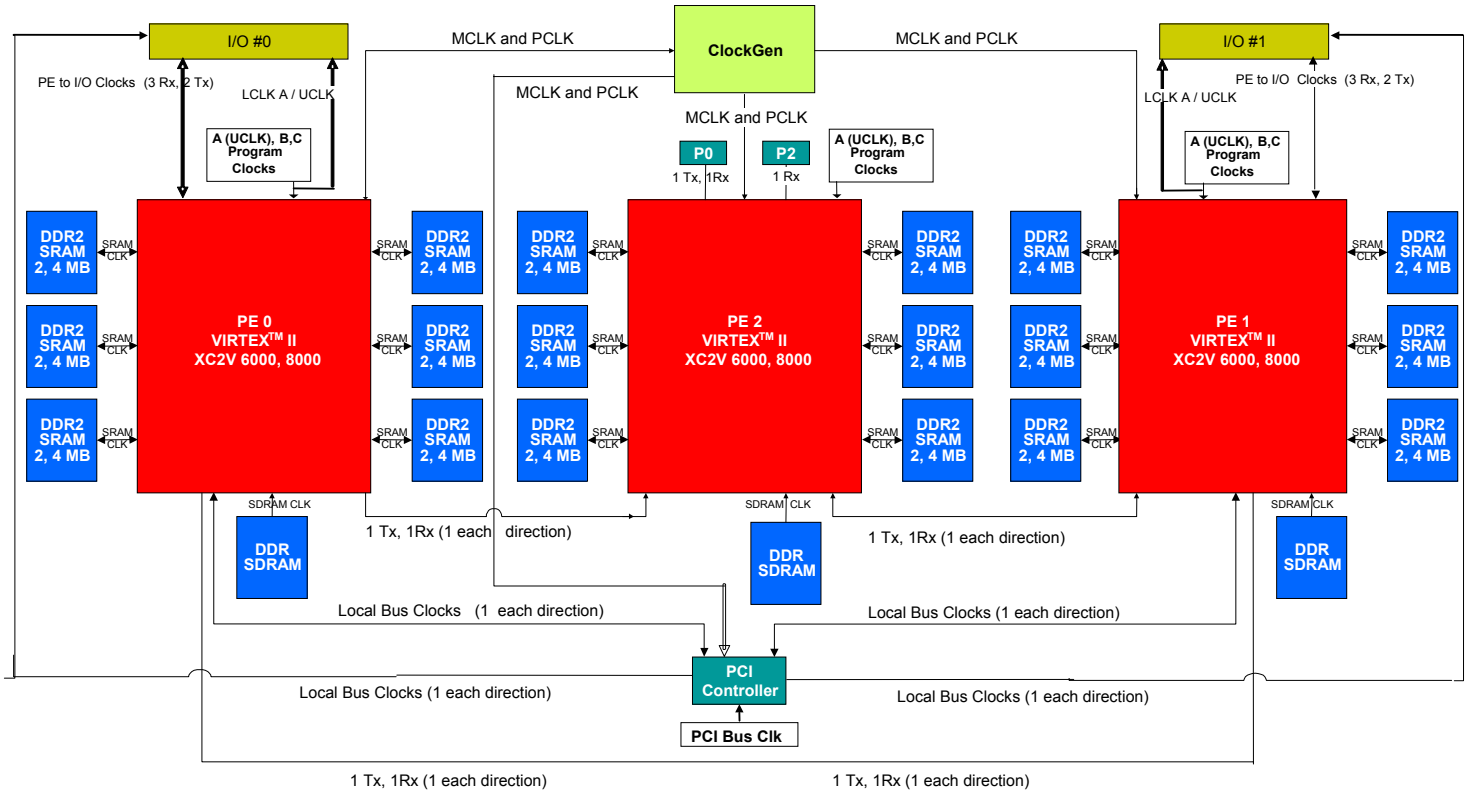
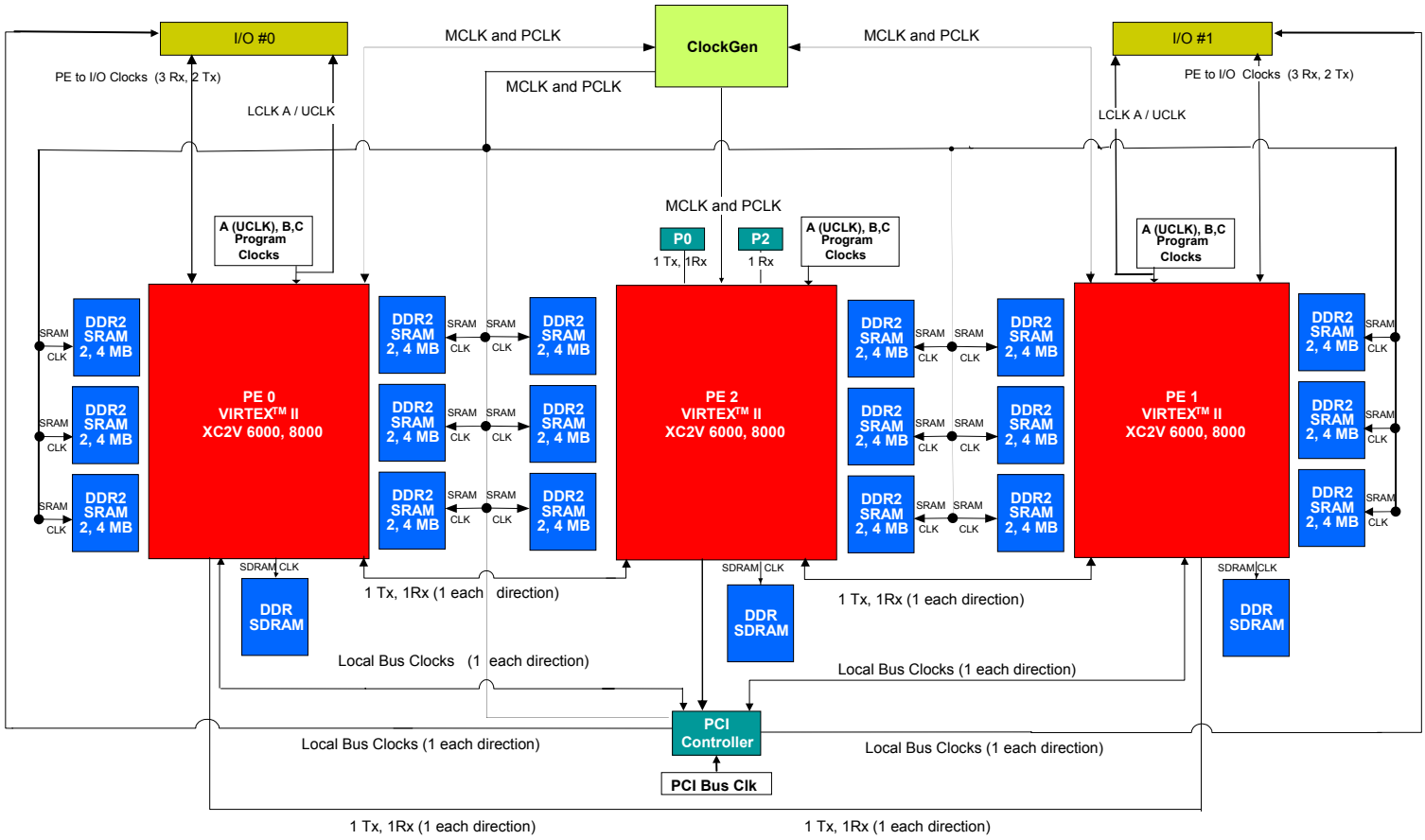


Figure 6-9: WILDSTAR™-II /VME Clock Distribution (Revs. A, B)



Copyright Annapolis Micro Systems, Inc. 2002, 2004

Figure 6-10: WILDSTAR™-II /VME Clock Distribution (Rev. C)

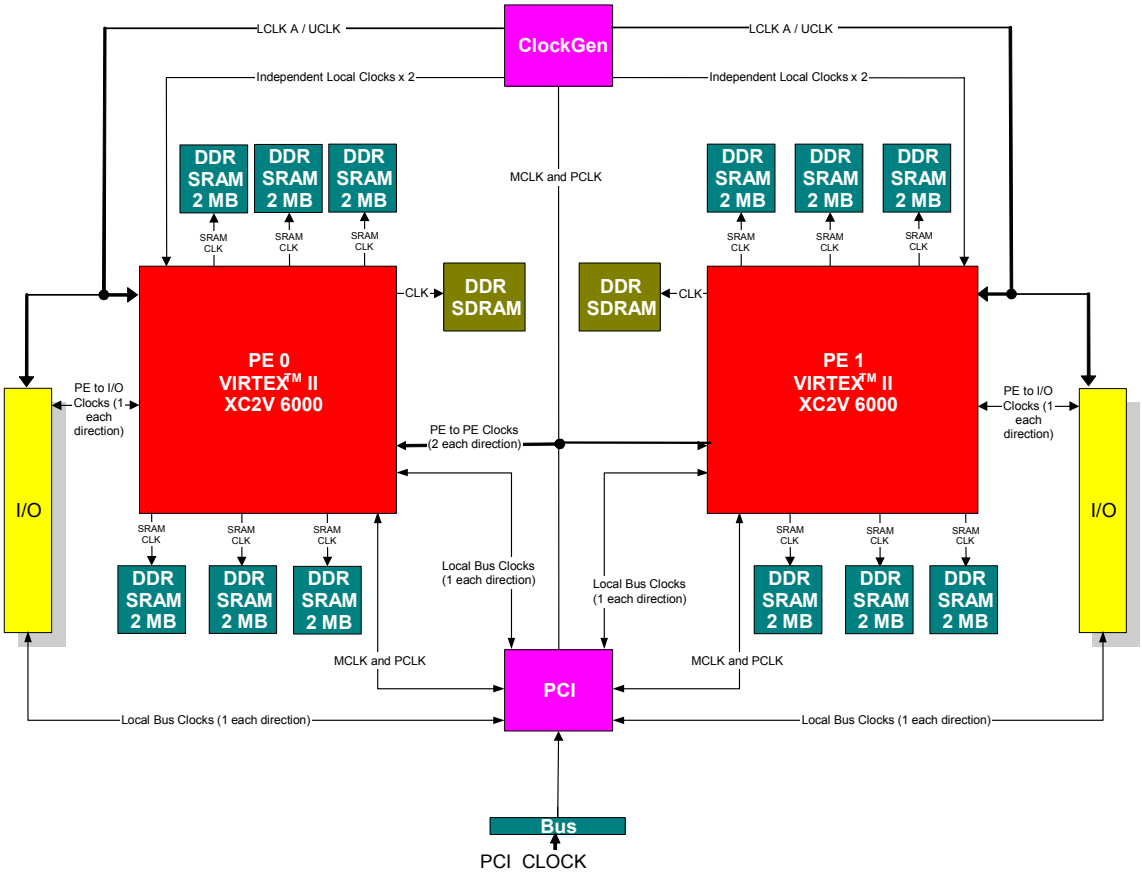


Figure 6-11: WILDSTAR™-II /PCI Clock Distribution (Revs. A, B, C,)

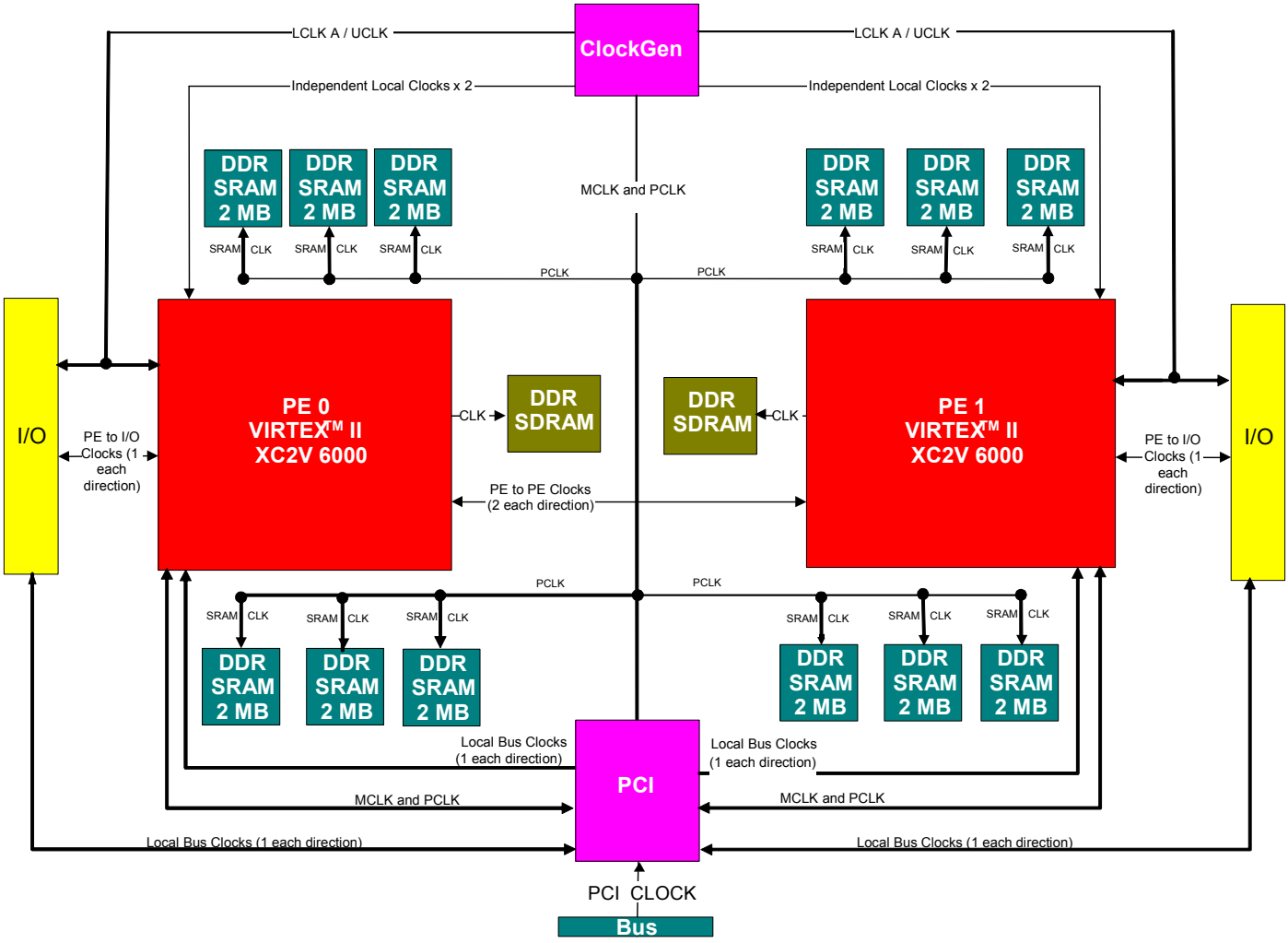


Figure 6-12: WILDSTAR™-II /PCI Clock Distribution (Rev. D)

6.5.2 Clock Descriptions and Frequencies

INFORMATION NOTE

The PEs' local A, B, and C clocks are highly accurate. Global M and P clocks can be set to an integral multiple of 3 MHz. The global ICLK cannot be modified.

MCLK

MCLK is differential and asynchronous to PCLK. It is configurable through the WILDSTAR™-II host software.

PCLK

PCLK is differential and asynchronous to MCLK. It is configurable through the WILDSTAR™-II host software.

ICLK

ICLK is the Local Address Data Bus (LAD Bus) clock. It is fixed at 132 MHz and single-ended. The PE uses this clock to interface to the PCI Controller for host access via the LAD bus.

The tables below describe the WILDSTAR™-II clocks. Note that the WILDSTAR™-II /PCI Rev. D clocks are listed in a separate table. With the exception of PCLK, the Rev. D clocks are identical to all other revisions of the board.

Table 6-2: WILDSTAR™-II Clock Description

Clock	Type	Single-Ended or Differential	Frequency Range (in MHz)		Destination	Resolution
			Minimum	Maximum		
PCLK	Processor Clock	Differential	24	420	PEs and I/O	3 MHz
MCLK	Memory Clock	Differential	24	420	PEs and I/O	3 MHz
ICLK	LAD Clock	Single-Ended	Fixed at 132 MHz		PCI, PEs and I/O	Better than 1 MHz
LCLK A (UCLK)*	Local Clock A	Single-Ended	6	200		
LCLK B	Local Clock B	Single-Ended	6	200	Local to PE	Better than 1 MHz
LCLK C	Local Clock C	Single-Ended	6	200	Local to PE	Better than 1 MHz

*For boards made prior to Rev. D, an Engineering Change Request (ECR) is required for UCLK to be functional.

Table 6-3: WILDSTAR™-II/ PCI (Rev. D) Clock Description

Clock	Type	Single-Ended or Differential	Frequency Range (in MHz)		Destination	Resolution
			Minimum	Maximum		
PCLK	Processor Clock	Differential	50	100	PEs, SRAMs	.25 MHz
			100	200		.5 MHz
			200	400		1 MHz
			400	800		2 MHz
MCLK	Memory Clock	Differential	24	420	PEs and I/O	3 MHz
ICLK	LAD Clock	Single-Ended	Fixed at 132 MHz		PCI, PEs and I/O Local to PE	Better than 1 MHz
LCLK A (UCLK)	Local Clock A	Single-Ended	6	200		
LCLK B	Local Clock B	Single-Ended	6	200	Local to PE	Better than 1 MHz
LCLK C	Local Clock C	Single-Ended	6	200	Local to PE	Better than 1 MHz

6.6 WILDSTAR-II PRO Clocks

Table 6-4: WILDSTAR™-II PRO Clock Description (all PRO-Series Boards)

Clock	Type	Single-Ended or Differential	Frequency Range (in MHz)		Destination	Resolution (MHz)
PCLK	Processor Clock	Differential	**Minimum**	**Maximum**		
			50	100	PEs, SRAMs, I/O, DRAMs	.25 MHz
			100	200		.5 MHz
48 MHz Clock	Fixed Clock	Single Ended	48		PEs, PCI Controller, PLDs	N/A
LADCLK	Local Bus Clock	Single Ended (PCLK/2)	PCLK/2		PEs, I/O	N/A
UCLK0	User Clock for IO Slot 0	Single Ended	50	100	PEs, I/O	0.25
			100	200		0.50
UCLK1	User Clock for IO Slot 1	Single Ended	50	100	PEs, I/O	0.25
			100	200		0.50

6.7 WILDSTAR™-II Backplane I/O and Power Options

WILDSTAR™- II/ VME, WILDSTAR™- II PRO/ VME, and WILDSTAR™- II PRO ACE for VME all support data transmission over the VME backplane. WILDSTAR™- II/ VME (Rev. C and higher), WILDSTAR™- II PRO/ VME, and WILDSTAR™- II PRO ACE/ VME are also capable of using the user-defined pins on the VME backplane to increase the amount of power that can be safely drawn from the backplane.

6.7.1 WILDSTAR™-II/ VME Backlplane I/O and Power Configurations

The WILDSTAR™-II /VME board supports backplane I/O functions through the P0 VME backplane and P2 backplane connector. Pin mapping information appears in Appendix A of this manual, with Appendix B describing P0/P2 pin usages.

CAUTION

Do *not* attempt to insert a WILDSTAR™-II /VME board with a P0 Backplane connector into a chassis that does not accept P0.

The WILDSTAR™-II /VME is shipped with a standard configuration for the P0 and P2 connectors (Figure 6-13). P0 is connected to PE2 via a 38 differential (3.3V LVDS) interface. Figure 6-14 shows the diagram for a WILDSTAR™-II /VME, Rev. C (factory-configured with 5 volt option).

In the standard configuration (Figure 6-13), P2 can be configured for support of single or dual RACE™++. The RACEway™ clock is connected to a global clock input buffer on PE2. A differential clock is connected to a global clock input buffer on PE2 from the P0 I/O Backplane connector.

An additional advantage to this configuration is point-to-point communication between WILDSTAR™-II /VME boards via a differential cable on the rear side of the VME backplane.

CAUTION

The P2-to-PE2 standard configuration is *not* 5 volt tolerant. Do not connect a 5 volt signal to this interface. Failure to adhere to this specification could result in damage to the product.

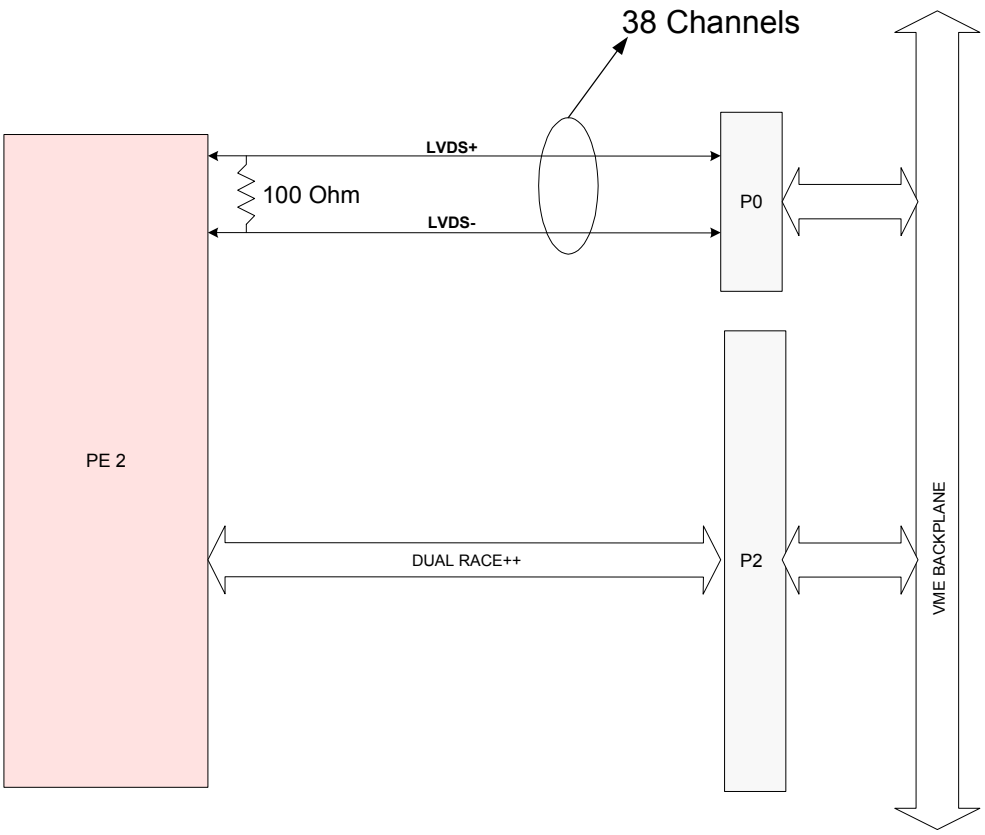


Figure 6-13: P0/P2 Backplane, Standard Configuration

As needs require, you can select customized P0/P2 Backplane options (see below), including the option of disconnecting P0/P2 from PE2 entirely. By special request, these options are implemented by Annapolis Micro Systems during manufacture of the board.

Figure 6-14 shows a standard configuration for a WILDSTAR™-II /VME with 5V option. This option requires the P0 5 Volt Adapter card.

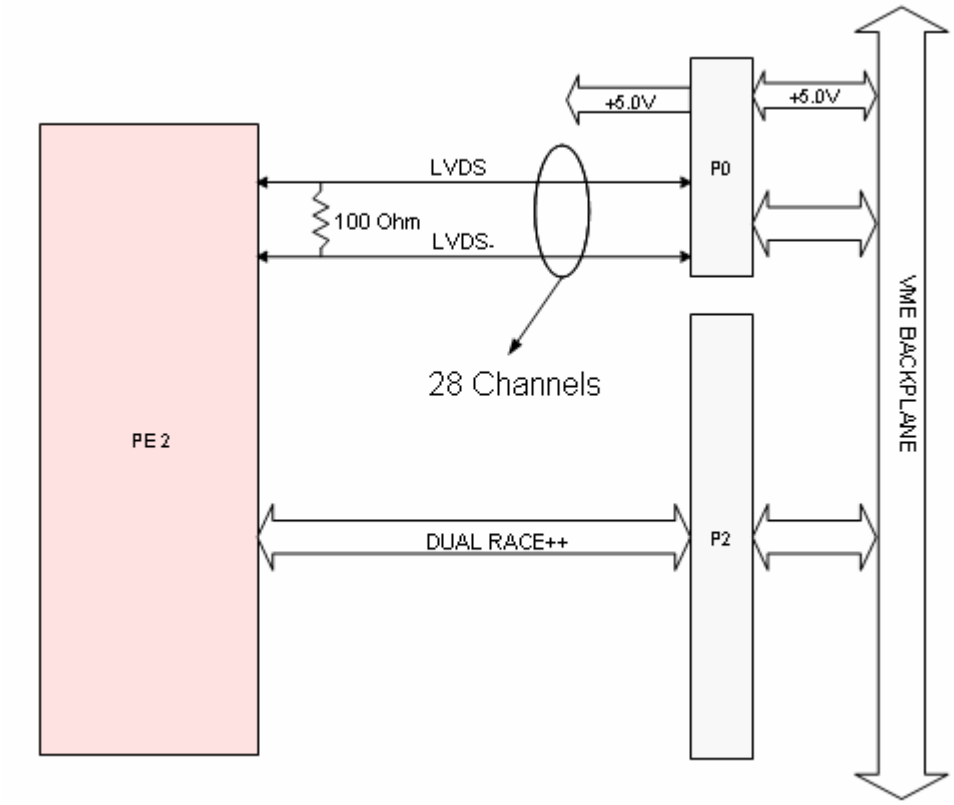


Figure 6-14: WILDSTAR™-II/VME (Rev C) P0 Backplane with 5 Volt Option

CAUTION

To prevent pin damage, be sure to plug the signal cables into the correct pins. If you have questions about which pins are involved, please contact Annapolis Micro Systems Technical Support using the information provided in Chapter 5.

The P0 Series Terminated, P2 RACE™++ optional configuration is shown in Figure 6-15. It features a single-ended clock to PE2 from P0 and a RACEway™ clock to PE2 from P2. In addition, when P0's signals are configured as single-ended and P2's are configured as RACE™++, 12 additional lines are available from P0 to the P2 I/O Backplane connector.

CAUTION

This P2-to-PE2 configuration is *not* 5 volt tolerant. Do not connect a 5 volt signal to this interface. Failure to adhere to this specification could result in damage to the product.

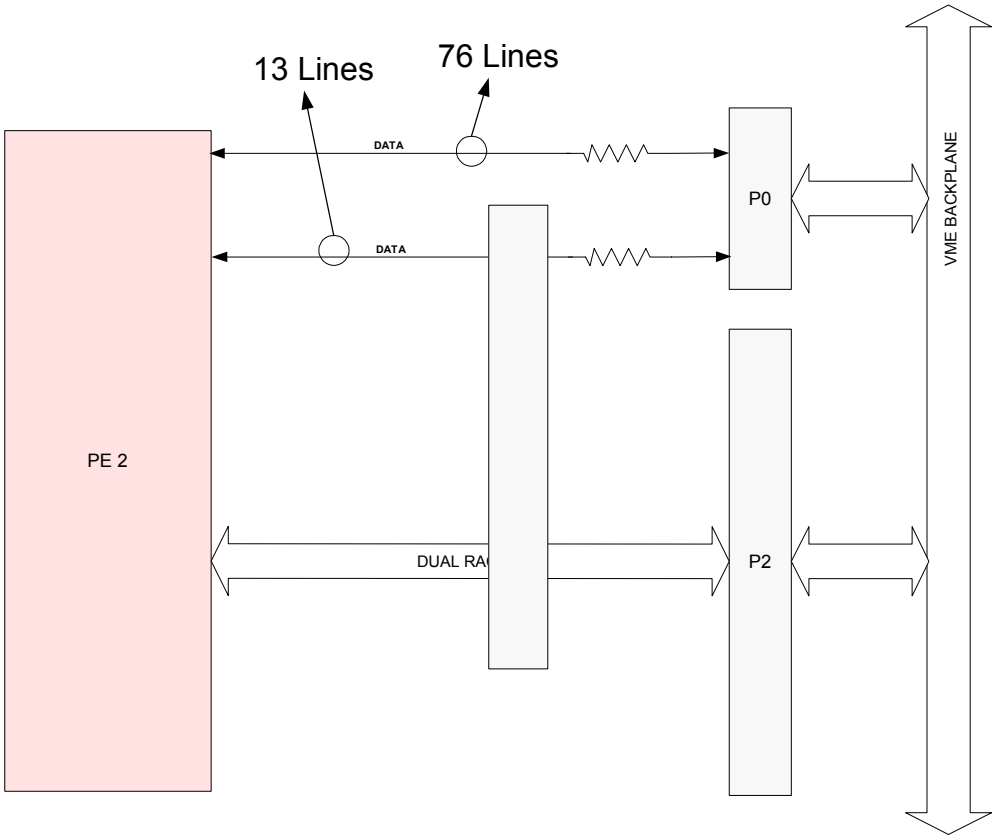


Figure 6-15: P0 Series Terminated, P2 RACE™ Optional Configuration

The P0 Connector Differential, P2 Series Terminated optional configuration is shown in Figure 6-16. It features a differential clock to PE2 from P0 and a single-ended clock to PE2 from P2.

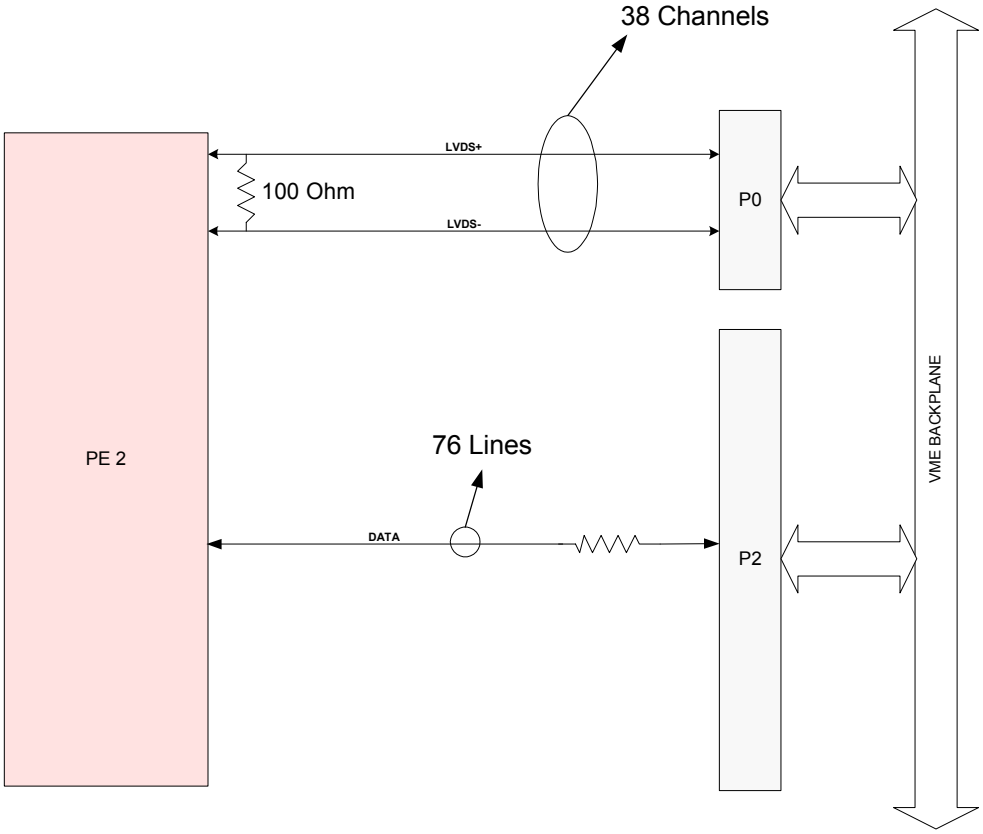


Figure 6-16: P0 Differential, P2 Series Terminated Optional Configuration

The P0 Differential, P2 Differential optional configuration is shown in Figure 6-17. It features differential clocks to PE2 from the P0 and P2 connectors.

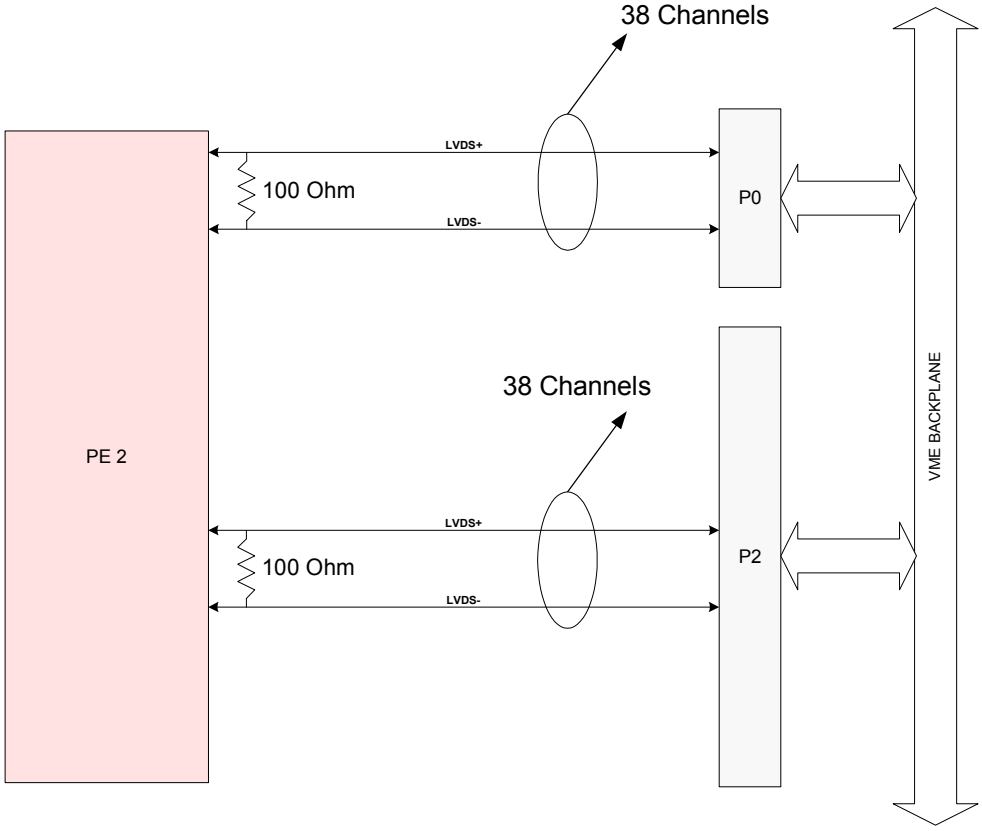


Figure 6-17: P0 Differential, P2 Differential Optional Configuration

6.7.2 WILDSTAR™-II PRO /VME and PRO ACE for VME P2 Backplane Options

The WILDSTAR™-II PRO /VME and PRO ACE for VME boards support backplane I/O communications through user-definable pins on the VME P2 backplane connector. These pins can also provide extra 3.3V and 5V power to the WILDSTAR™-II PRO/ VME and PRO ACE for VME boards.

P2 configuration options include:

- Dual RACEway
- P2 Power and 2.5V Differential I/O
- P2 Power and 3.3V Single-Ended I/O
- Differential
- Single-Ended

WILDSTAR-II PRO/ VME and PRO ACE for VME boards also support high-speed serial communication links using Rocket I/O through the user-definable pins on the VME P0 connector. See Section 6.8.1 for details of all Rocket IO connections on all WILDSTAR-II PRO Products.

Pin mapping information for P0 and P2 is included in Appendix B.



INFORMATION NOTE

Custom signaling and termination options are also available. Contact Annapolis Micro Systems, Inc. for details.



WARNING

Check to see if your WILDSTAR™-II PRO/VME board has the P2 power adapter backplane option, indicated by the following label affixed to both sides of the board:

CAUTION!
P2 POWER

If your board has the P2 power option, *never* plug it into a chassis that uses any of the P2 signals. Doing so could destroy the WILDSTAR™-II PRO/VME and other boards connected to P2.

CAUTION

All WILDSTAR™-II PRO boards use Rocket I/O for high-speed serial communication over the VME backplane and between PEs. Do *not* plug the P0 VME connector into any device that doesn't support Rocket I/O serial lines. Doing so could damage the WILDSTAR™-II PRO /VME or PRO ACE for VME.

If you have specific questions about this warning, please contact Annapolis Micro Systems, Inc. using the information provided in Chapter 5.

6.7.2.1 Dual RACEway

The P2 Connector on WILDSTAR™-II PRO /VME and PRO ACE for VME can be configured to support Dual RACE™++ communications over the backplane. The diagram below shows these connections.

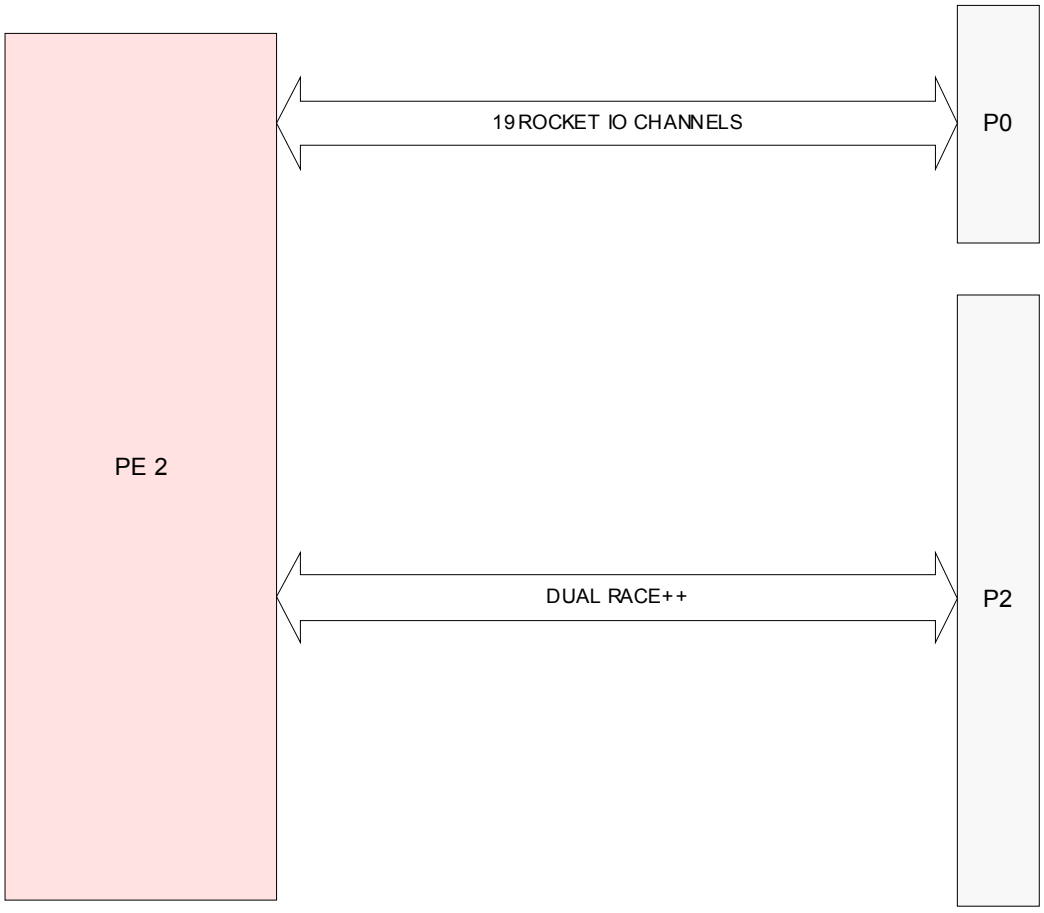


Figure 6-18: WILDSTAR™-II PRO /VME I/O Backplane Connection with Dual RACE++

6.7.2.2 Differential Signaling (2.5V)

The P2 connector on WILDSTAR™-II PRO /VME or PRO ACE for VME can be configured to support 2.5V LVDS I/O or 2.5V LVCMOS communication over the backplane (Figure 6-19). This configuration allows the highest bit rates over the P2 connector.

CAUTION

Do not expose 2.5V LVDS and LVCMOS connections to any signaling higher than 2.5V. Higher voltages can damage the board.

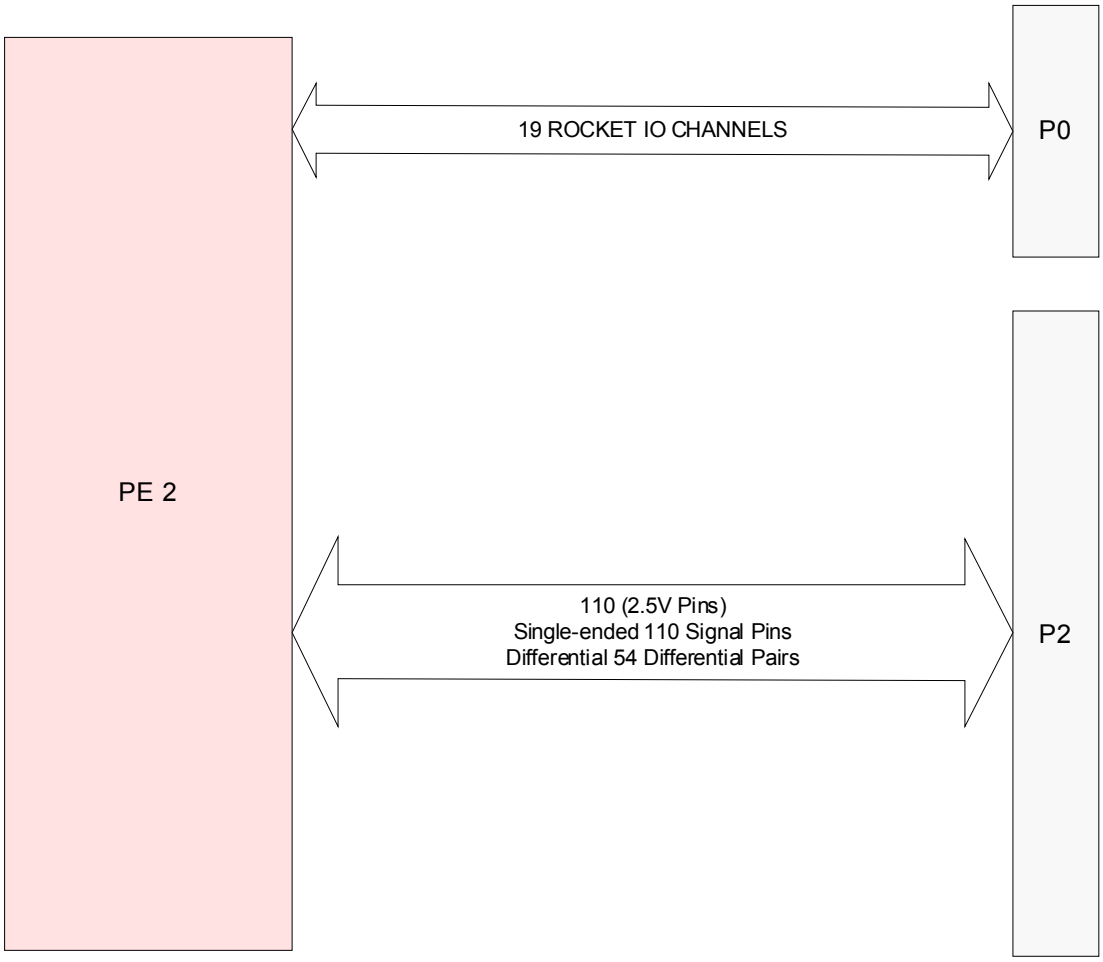


Figure 6-19: WILDSTAR™-II PRO /VME I/O Backplane Connection with 2.5V Differential Signaling

6.7.2.3 Single-Ended Signaling (3.3V)

The P2 connector on WILDSTAR™-II PRO /VME or PRO ACE for VME can be configured to support 3.3V Single-Ended connections over the backplane. Figure 6-20 shows these connections.

CAUTION

Do not expose 3.3V single-ended connections to any signaling higher than 3.3V. Higher voltages can damage the board.

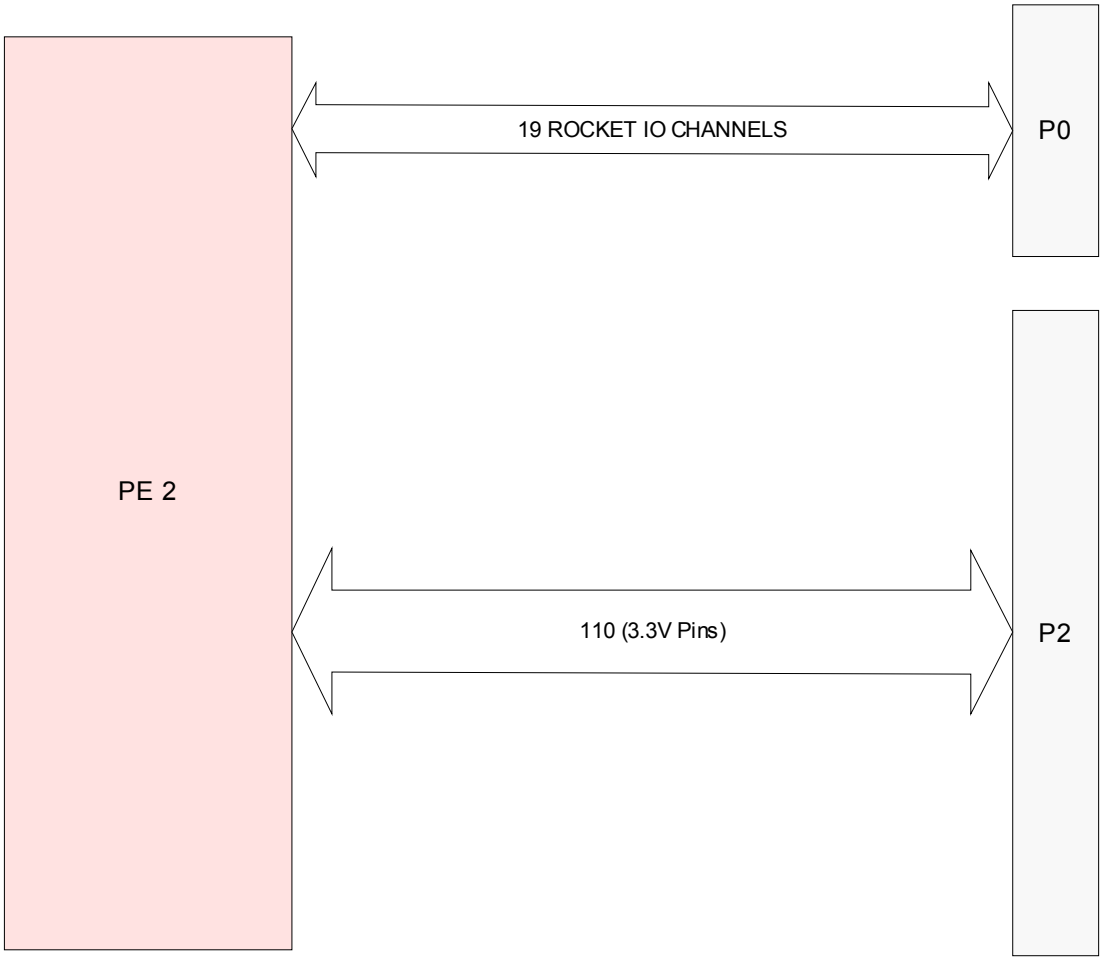


Figure 6-20: WILDSTAR™-II PRO /VME I/O Backplane Connection with 3.3V Single-Ended Signaling

6.7.2.4 P2 Power and 2.5V Differential I/O

The P2 Power and 2.5 Differential I/O option on WILDSTAR™-II PRO /VME or PRO ACE for VME supplies additional power and support for 2.5V LVDS or 2.5V

LVCMOS communication across the backplane. Figure 6-21 illustrates these connections.



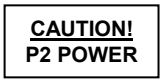
CAUTION

Do not expose 2.5V LVDS and LVCMOS connections to any signaling higher than 2.5V. Higher voltages can damage the board.

WARNING

Check to see if your WILDSTAR™-II PRO/VME board has the P2 power adapter backplane option, indicated by the following label affixed to both sides of the board:





If your board has the P2 power option, *never* plug it into a chassis that uses any of the P2 signals. Doing so could destroy the WILDSTAR™-II PRO/VME and other boards connected to P2.

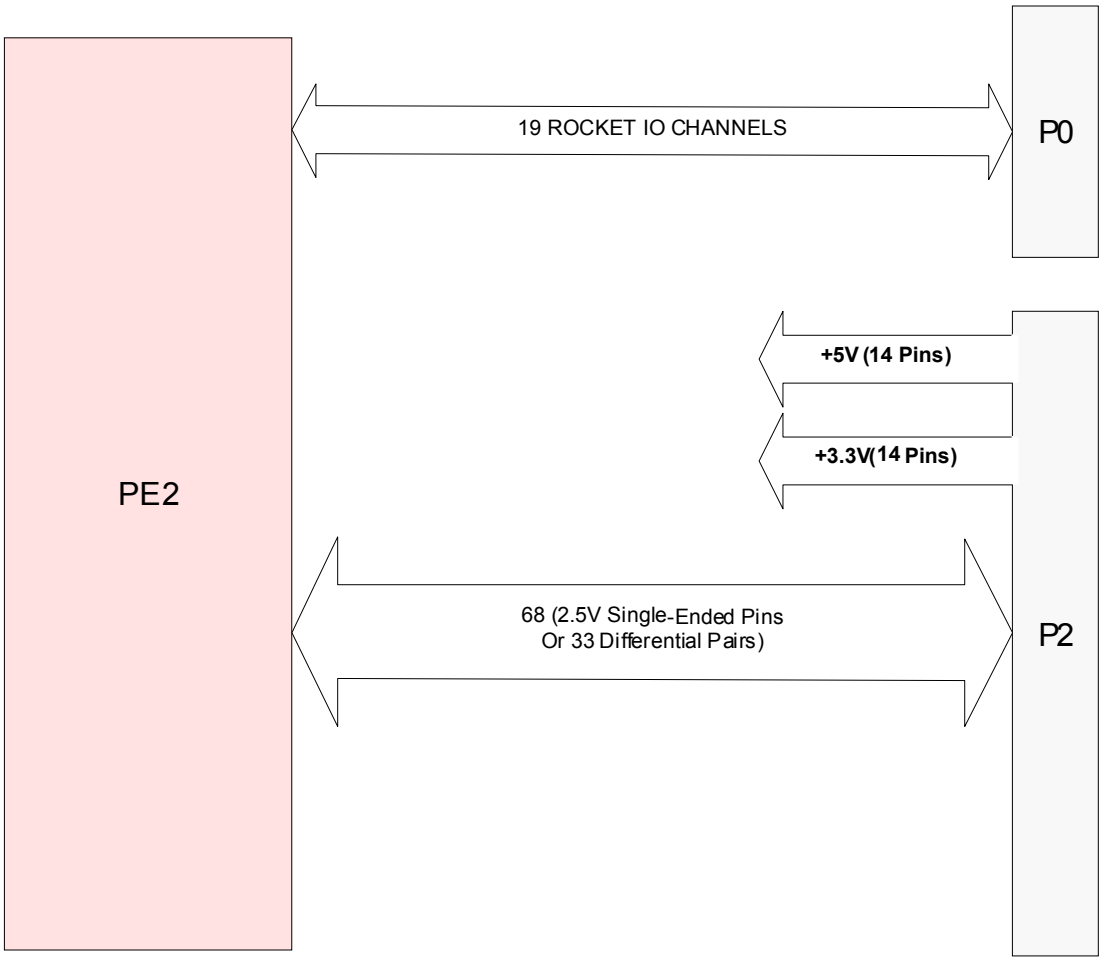


Figure 6-21: WILDSTAR™-II PRO /VME I/O Backplane Connection with 2.5V Differential Signaling and P2 Power

6.7.2.5 P2 Power and 3.3V Single-Ended I/O

The P2 Power and 3.3V Single-Ended I/O option on WILDSTAR™-II PRO /VME or PRO ACE for VME provides additional power and support for LVCMOS or LVTTL communication over the backplane. Figure 6-23 displays these connections.

CAUTION

Do not expose 3.3V single-ended connections to any signaling higher than 3.3V. Higher voltages can damage the board.

WARNING

Check to see if your WILDSTAR™-II PRO/VME board has the P2 power adapter backplane option, indicated by the following label affixed to both sides of the board:

CAUTION!
P2 POWER

If your board has the P2 power option, *never* plug it into a chassis that uses any of the P2 signals. Doing so could destroy the WILDSTAR™-II PRO/VME and other boards connected to P2.

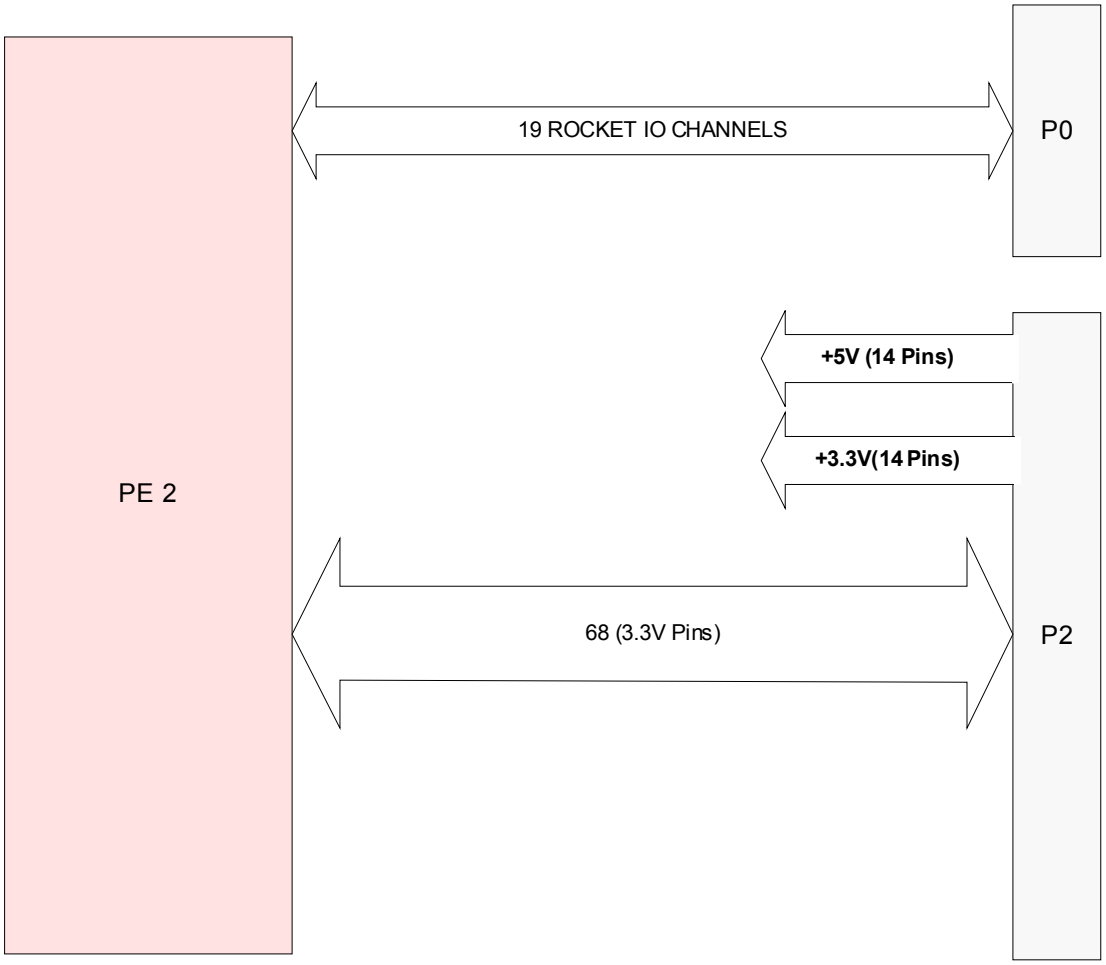


Figure 6-22: WILDSTAR™-II PRO /VME I/O Backplane Connection with 3.3V Differential Signaling and P2 Power

6.8 WILDSTAR™-II PRO Rocket I/O Interfaces

All WILDSTAR™-II PRO motherboards use XIlinx® Rocket I/Os for high speed serial communication. These connections allow for communication over the VME backplane and between PEs.

6.8.1 Rocket I/O Connections for WILDSTAR™-II PRO Boards

The following figures outline the Rocket IO connections for all three WILDSTAR™-II PRO boards: VME, VME ACE, and PCI..

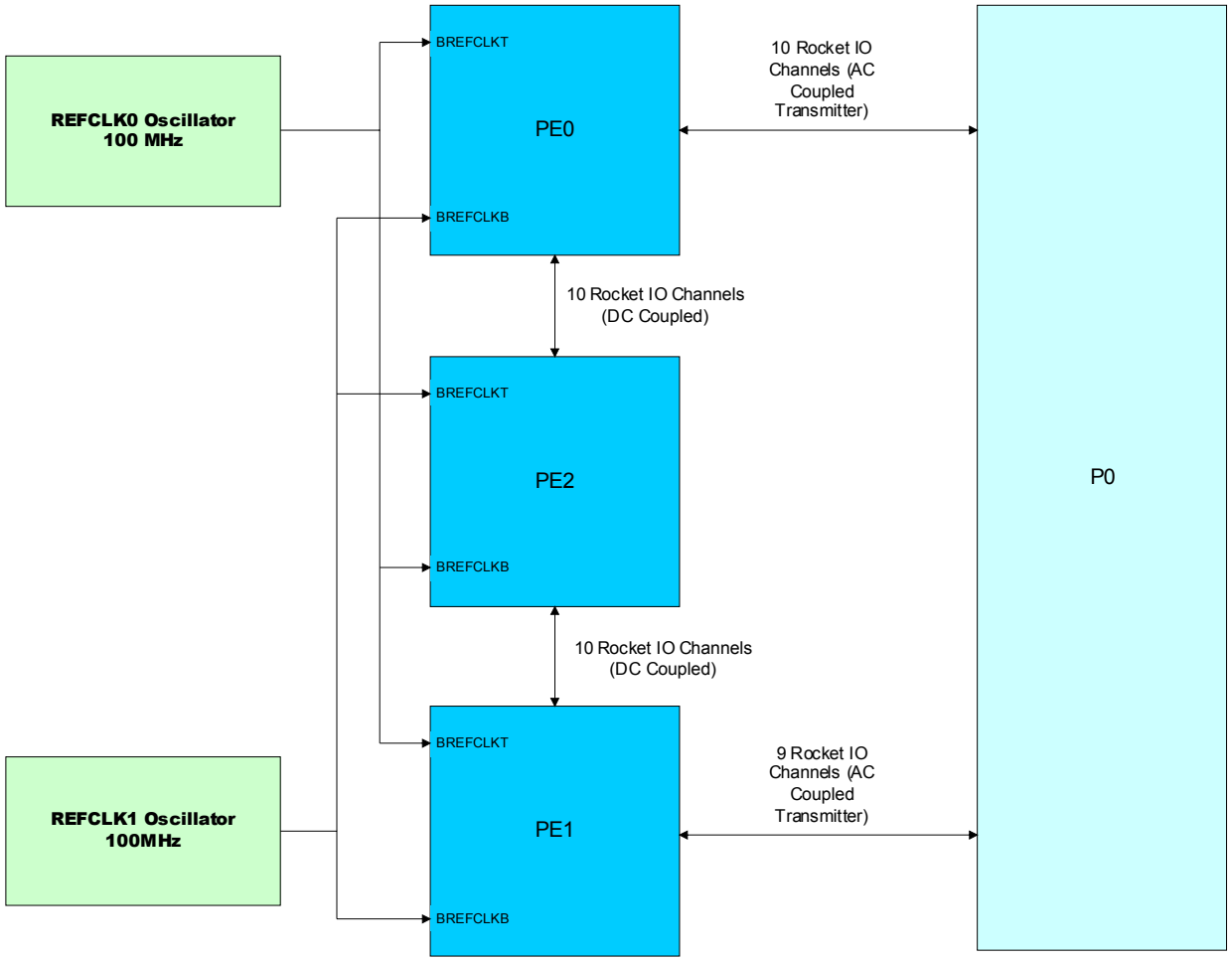


Figure 6-23: Rocket I/O Connections for WILDSTAR™-II PRO/VME

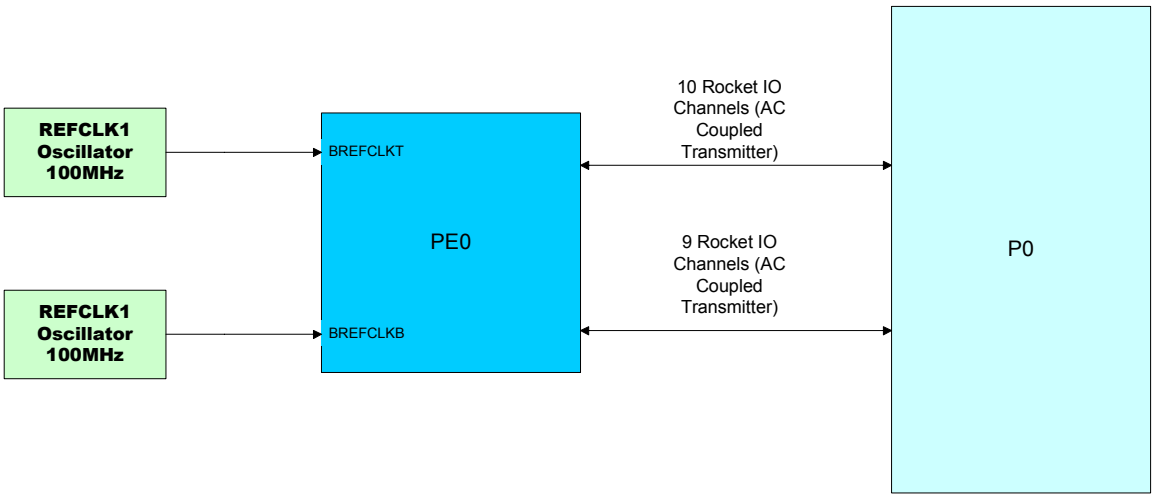


Figure 6-24: Rocket I/O Connections for WILDSTAR™-II PRO ACE for VME

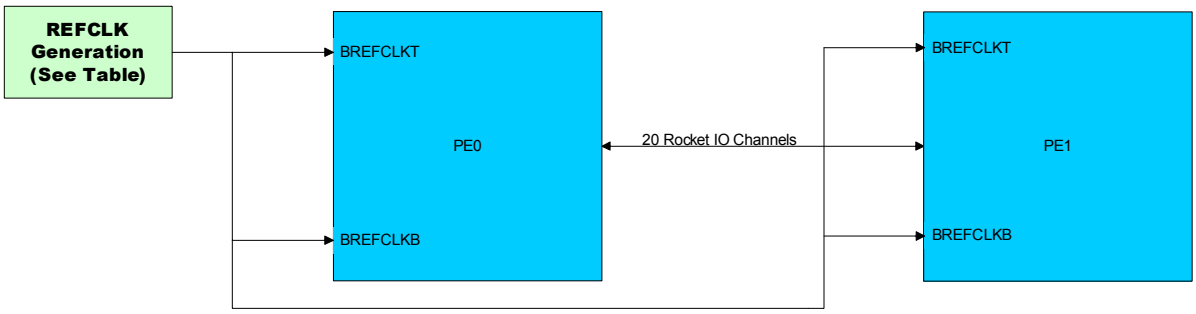


Figure 6-25: Rocket I/O Connections for WILDSTAR™-II PRO/PCI

6.8.2 Rocket I/O Clocks

There are specific clock input requirements for running Rocket I/O interfaces. Below are tables listing the clock frequencies for each type of WILDSTAR™-II PRO board using Rocket I/O.

Table 6-5: Clock Frequencies for Rocket I/O with WILDSTAR™-II PRO PCI, Rev. B

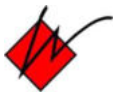
PE SPEED GRADE	REFCLK Frequency
-5	50-100MHz (Programmable)
-6, -7	50-156MHz (Programmable)

Table 6-6: Clock Frequencies for Rocket I/O with WILDSTAR™-II PRO PCI, Rev. C and D

PE SPEED GRADE	REFCLK Frequency
-5	100MHz
-6, -7	100 or 156.25MHz

Table 6-7: Clock Frequencies for Rocket I/O with WILDSTAR™-II PRO/VME and PRO ACE for VME

PE SPEED GRADE	REFCLK Frequency
-5, -6, -7	100MHz



7 VHDL MODELS REFERENCE AND GUIDE

The WILDSTAR™-II VHDL models are intended to help you design applications on WILDSTAR™-II boards. By simulating PE behavior, these models can verify the functionality of the reconfigurable PEs in a completed design.



INFORMATION NOTE

WILDSTAR™-II and WILDSTAR™-II PRO-series have separate VHDL models.



INFORMATION NOTE

VHDL Models are compiled and therefore require the ModelSim® simulator to simulate.

7.1 WILDSTAR™-II VHDL Model

The WILDSTAR™-II VHDL Model provides an accurate means of describing the behavior of the WILDSTAR™-II system. The Model encompasses the WILDSTAR™-II device and a simulation model of the host system.

The Model can be used to design the functionality of the reconfigurable processing elements (PEs) residing in the WILDSTAR™-II system. They can then simulate the behavior of the entire system to verify the functionality of the completed PE design.

The Model consists of two main components, a *simulation environment* and *PE interfaces*, both of which are described below.

7.1.1 VHDL Model Simulation Environment

The WILDSTAR™-II VHDL Model provides a full system simulation environment for debugging designs. This environment encompasses all functionality within the WILDSTAR™-II system, providing the designer with an accurate test bed for completed PE designs. As a result, processing element designs can be validated prior to synthesis, placement, and routing.

The simulation environment consists of three levels:

- **PE Design level**
- **Board level**
- **System level**

7.1.1.1 PE Design Level

The PE design is only the first level in the simulation environment. The purpose of this level is solely to drive the pins of the PE, and is therefore completely synthesizable. Code for this level will eventually become the bitstream that programs the PE on the board.

Both the PE interfaces and the user logic for driving the PE pins are contained in this level. The logic used at this level can include LAD bus logic, memory access logic, I/O logic and all other logic for processing data.

7.1.1.2 Board Level

Each PE is contained on a specific mainboard or I/O card. These boards contain components such as memories, buses, and I/O ports that interact with the PE. The board level simulation environment simulates all of these board components and provides the correct stimulus to the PE pads.

The board level also provides interconnections between PEs on a multi-PE board. The board level simulation environment comes completely written, and the user need only set configuration options.

7.1.1.3 System Level

The highest simulation environment level is the system level, which is used when one or more boards are placed into a host computer and connected together or connected to external data I/O. The system level simulation environment provides a means to simulate the host computer, place multiple boards in the system, and arbitrarily interconnect the boards to each other and to external sources.

The host computer is simulated through a set of VHDL procedures that mirror actual C API calls, such as register reading/writing, DMA reading/writing and interrupt handling. The user creates a 'program' using these VHDL procedures. This program creates stimulus on a simulated PCI bus, which is used by a simulated PCI controller on each board level to drive signals on the PE LAD bus pads.

The system level also provides signals connected to I/O ports on a board that can be driven or read by user logic, or simply connected to other I/O ports.

7.2 VHDL Interface Components

Each PE processing module on the WILDSTAR™-II board consists of numerous auxiliary components—clocks, memories, other PEs, and the host interface (LAD) controller—all of which can be used in any design.

Supplied with the VHDL model described are interfaces to each of these components:

Basic Interface

The Basic Interface components provide a direct interface to the PE pads with no intervening logic. The user logic is presented with two records—one containing all input pad signals, the other containing all output pad signals.

Standard Interface

The Standard Interface components comprise the layer above the basic interface. Each standard interface component contains a basic interface component, adds IOB registers to the PE pad signals, and simplifies the interface presented to the user. The user logic is presented with two records—one containing all input standard interface signals, the other containing all output standard interface signals.

For example, the basic interface component to a serial PLD device would only present the user with the one bit serial data stream and the control bits. All serialization and control would have to be created by the user. The standard interface component, however, would include this serialization *and* control logic inside the component. The user logic interface records for the standard interface component would contain a parallel multi-bit data word and a minimal number of control signals.

Core Interface

A Core Interface component replaces a standard Interface and usually contains a basic Interface embedded inside it. An example of a core would be the sram_DDR2_Core which has control logic and a FIFO to properly access the SRAM memories.

Each Basic, Standard, and Core interface component includes User_In, User_Out, and Pads ports. The Pads ports connect the component to the pads of the PE. The PE pads are divided into logical sets by record type. For example, a Pads.Clocks record maps to all the clock pads, and a Pads.LAD_Bus record maps to all the LAD bus pads. The Pads port of each interface component will be assigned one of these Pads records.

The User_In port is a record of signals that are outputs of the component, but inputs to user logic. Similarly, the User_Out port is a record of signals that are inputs to the component, but outputs of user logic.

In addition to User_In, User_Out, and Pads, each standard interface component has one or more clock ports. All logic into and out of the interface will be synchronous to one of these clocks. Additionally, all Pad output signals are registered in the IOB on their appropriate clock edge.

In addition to User_In, User_Out, and Pads, each core component has a "clk" port. All logic into and out of the interface will be synchronous to this "clk" port. Core components may also have other clock ports for internal use and generics if the core requires parameters. Additionally, all pad output signals are registered in the IOB on their appropriate clock edge.

The basic I/O components do not have a clock port. The User_In and User_Out ports for each basic I/O component are connected directly to the corresponding PE pad signal.

7.2.1 Clock Standard Interface (Clock_Std_IF)

INFORMATION NOTE

It is strongly encouraged that you use a single clock in a PE design. Use of multiple clocks within a single PE may cause severe clock jitter and application failure. PCLK is recommended for most processing tasks.

Clock_Std_IF provides a user interface to the various clock pads and other clock-related information in the VHDL design. The component declaration is shown below:

```
component Clock_Std_IF is
  generic
  (
    PLATFORM : PlatformType,
    M_CLK_DLL_TYPE                  : Clk_DLL_Type,
    M_CLK_CLKDV_DIVIDE              : string,
    M_CLK_DUTY_CYCLE_CORRECTION     : string,
    M_CLK_STARTUP_WAIT              : string,
    M_CLK_DFS_FREQUENCY_MODE        : string,
    M_CLK_CLKFX_MULTIPLY            : string,
    M_CLK_CLKFX_DIVIDE              : string,
    M_CLK_CLKOUT_PHASE_SHIFT        : string,
    M_CLK_PHASE_SHIFT               : string,
    M_CLK_DSS_MODE                  : string,

    P_CLK_DLL_TYPE                  : Clk_DLL_Type,
    P_CLK_CLKDV_DIVIDE              : string,
    P_CLK_DUTY_CYCLE_CORRECTION     : string,
    P_CLK_STARTUP_WAIT              : string,
    P_CLK_DFS_FREQUENCY_MODE        : string,
    P_CLK_CLKFX_MULTIPLY            : string,
    P_CLK_CLKFX_DIVIDE              : string,
    P_CLK_CLKOUT_PHASE_SHIFT        : string,
    P_CLK_PHASE_SHIFT               : string,
    P_CLK_DSS_MODE                  : string,

    M_CLK_FB_CLKDV_DIVIDE           : string,
    M_CLK_FB_DUTY_CYCLE_CORRECTION  : string,
    M_CLK_FB_STARTUP_WAIT           : string,
    M_CLK_FB_DFS_FREQUENCY_MODE     : string,
    M_CLK_FB_CLKFX_MULTIPLY         : string,
    M_CLK_FB_CLKFX_DIVIDE           : string,
    M_CLK_FB_CLKOUT_PHASE_SHIFT     : string,
    M_CLK_FB_PHASE_SHIFT            : string,
    M_CLK_FB_DSS_MODE               : string
  );
  port
  (
    Pads    : inout Clock_Pads_Type;
    User_In :   out Clock_Std_IF_In_Type;
    User_Out: in    Clock_Std_IF_Out_Type
  );
end component;
```

The Clock_Std_If component has three port signals: Pads, User_In, and User_Out. The Pads port is the component interface to the PE clock pads. This port should always be assigned to '`Pads.Clocks`'.

The User_In port, a record of type Clock_Std_IF_In_Type, is the user interface to the clock inputs. These signals should be used as the clock sources of the design.

Shown below is the definition of the Clock_Std_IF_In_Type record.

```
type Clock_Std_IF_In_Type is record
  M_Clk               : Clock_In_Type;
  M_ClkFb             : Clock_In_Type;
  P_Clk               : Clock_In_Type;
  A_Clk               : std_logic;
  B_Clk               : std_logic;
  C_Clk               : std_logic;
end record;
```

M_Clk, M_ClkFb, P_Clk, are routed through a DCM inside the Clock_Std_If component. Each clock uses the delay locked loop (DLL) feature of the Xilinx® Virtex™ II FPGA device. The DLLs help eliminate the effects of clock skew across the FPGA device, but they take time to acquire a lock to the feedback (or reference) clock. The "locked" indicator flags are provided to the design so you can determine when the DLL has acquired a lock.
The Clock_In_Type record contains all the DLL output signals, shown below:

```
type Clock_In_Type is record
  Clk       : std_logic;
  Clk90     : std_logic;
  Clk180    : std_logic;
  Clk270    : std_logic;
  Clk2x     : std_logic;
  Clk2x180  : std_logic;
  ClkDV     : std_logic;
  ClkFX     : std_logic;
  ClkFX180  : std_logic;
  Locked    : std_logic;
  Status    : std_logic_vector(7 downto 0);
  PSDone    : std_logic;
  Pad       : std_logic;
end record;
```

These record sets give you an interface to all of the DLL outputs for all three of these clocks.

A_Clk, B_Clk, and C_Clk are driven only by clock input buffers and are not routed through a DCM inside the Clock_Std_If. Instead, you need to selectively drive a DCM with any of these signals.

The User_Out port, of type `Clock_Std_IF_Out_Type`, provides you with an interface to any of the clock outputs. The only three clock outputs are the output signals for the M_Clk, M_ClkFb, P_Clk DCMs. The record definitions below show the available signals:

```
type Clock_Std_IF_Out_Type is record
  M_Clk   : Clock_Out_Type;
  P_Clk   : Clock_Out_Type;
  M_ClkFb : Clock_Out_Type;
end record;
```

```
type Clock_Out_Type is record
  Rst        : std_logic;
  DSSEn      : std_logic;
  PSIncDec   : std_logic;
  PSEn       : std_logic;
  PSClk      : std_logic;
end record;
```

Details of all the port signals of the Clock_Std_IF are shown in Table 7-1.

M_ClkFb is a special clock used for SRAM read transactions. It is generated by the Clock Standard Interface from P_Clk and sent externally through an IOB in the same manner that each SRAM clock is generated from the PE. This clock then returns through a PE clock pin and global buffer as M_ClkFb. It is only used to clock in SRAM read data because it was specifically designed to emulate a return clock from the SRAM chips.

Table 7-1: Clock Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads	NA	Bi-dir	Clock pad signals
User_In.M_Clk	1	Output	Memory clock DLL input record.
User_In.M_ClkFB	1	Output	Feedback memory clock DLL input record.
User_In.P_Clk	1	Output	Processor clock DLL input record.
User_In.A_Clk	1	Output	A Clk input signal. (Not connected to a DLL)
User_In.B_Clk	1	Output	B Clk input signal. (Not connected to a DLL)
User_In.C_Clk	1	Output	C Clk input signal. (Not connected to a DLL)
User_Out.M_Clk	1	Input	Memory Clock DLL output record.
User_Out.P_Clk	1	Input	Processor Clock DLL output record.
User_Out.M_ClkFB	1	Input	Feedback Memory Clock DLL output record.

The Clock Standard Interface generics are described below. For most applications they can be left out, since they are all given appropriate default values:

- **PLATFORM:** This generic should be set to VME_BUS for WILDSTAR™-II /PCI boards and PCI_BUS for WILDSTAR™-II /VME boards.
- **DLL_TYPE:** This generic should be set to LOW_FREQ (default) for nearly all applications. The Xilinx® documentation describes this DCM attribute in more detail.

- **CLKDV_DIVIDE**: defaults to “2.0”. This generic specifies what divisor the ClkDV DCM output will have compared to the DCM CLKIN. The Xilinx documentation describes this DCM attribute in more detail.
- **DUTY_CYCLE_CORRECTION**: defaults to “TRUE”. This attribute specifies whether the DCM will use duty cycle correction on the input clock. The Xilinx® documentation describes this DCM attribute in more detail.
- **STARTUP_WAIT**: defaults to “FALSE”. This attribute delays the completion of the device configuration process until after the DCM locks. The Xilinx documentation describes this DCM attribute in more detail.
- **DFS_FREQUENCY_MODE**: defaults to “LOW”. The Xilinx documentation describes this DCM attribute in more detail.
- **CLKFX_MULTIPLY**: defaults to 4. The Xilinx documentation describes this DCM attribute in more detail.
- **CLKFX_DIVIDE**: defaults to 1. The Xilinx documentation describes this DCM attribute in more detail.
- **CLKOUT_PHASE_SHIFT**: defaults to "NONE". Can also be "FIXED" or "VARIABLE". The Xilinx documentation describes this DCM attribute in more detail.
- **PHASE_SHIFT**: defaults to 0. Can be -255 to 255 if CLKOUT_PHASE_SHIFT is set to “FIXED”. The Xilinx documentation describes this DCM attribute in more detail.
- **DSS_MODE**: defaults to “NONE”. The Xilinx documentation describes this DCM attribute in more detail.

7.2.2 LAD Bus Standard Interface (LAD_Bus_Std_IF)

The LAD Bus is the primary means of communicating with the host system. Using the LAD bus, the PE can send and receive programmed I/O data from the host and DMA data to and from the host.

The component declarations for the LAD_Bus_Std_IF and port record types are shown below:

```
component LAD_Bus_Std_IF is
  port
  (
    Global_Reset  : in    std_logic;
    Pads          : inout LAD_Bus_Pads_Type;
    User_In       :   out LAD_Bus_Std_IF_In_Type;
    User_Out      : in    LAD_Bus_Std_IF_Out_Type
  );
end component;
```

```
type LAD_Bus_Std_IF_In_Type is record
  Addr            : std_logic_vector ( 18 downto 0 );
  Data_In         : std_logic_vector ( 31 downto 0 );
  Reg_Strobe      : std_logic;
  DMA_Strobe      : std_logic;
  DMAInProgress   : std_logic;
  Write           : std_logic;
  PciRdy          : std_logic;
  BusGnt          : std_logic;
  RxClk_In        : Clock_In_Type;
end record;
```

```
type LAD_Bus_Std_IF_Out_Type is record
  Data_Out        : std_logic_vector ( 31 downto 0 );
  Strobe_Out      : std_logic;
  DMA_Init        : std_logic;
  PeRdy           : std_logic;
  IntReq          : std_logic;
  BusReq          : std_logic;
  RxClk_Out       : Clock_Out_Type;
end record;
```

The "User_In" and "User_Out" port signals of the LAD_Bus_Std_IF component should be used by the design to receive data from and send data to the LAD bus of the WILDSTAR™-II device, respectively. The "Pads" signals of the LAD_Bus_Std_IF component are only used to connect the LAD_Bus_Std_IF component to the pads, and should always be assigned to Pads.LAD_Bus. A definition of each of the port signals is given in Table 7-2 below.

Table 7-2: LAD Bus Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads	NA	Bi-dir	LAD Bus Pad signals
Global_Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARTUP_Virtex component
User_In.Addr	19	Output	User interface to the LAD bus address; note that this is a DWORD address; also note that this address is automatically incremented each clock cycle during burst LAD bus cycles
User_In.Data_In	32	Output	User interface to the input data from the LAD bus
User_In.Reg_Strobe	1	Output	Register space access strobe signal; indicates a valid register space cycle when '1'
User_In.DMA_Strobe	1	Output	DMA access strobe signal; indicates a valid DMA-To-PE cycle when '1'
User_In.DMAInProgress	1	Output	This signal goes low when all DMA-From-PE data has been transmitted to the host.

Signal Name	Width	Dir	Description
User_In.Write	1	Output	Write select interface signal; indicates a write cycle when '1' or a read cycle when '0' (when strobe is '1')
User_In.PCIRdy	1	Output	Set to '1' when the PCI controller can accept DMA data
User_In.BusGnt	1	Output	Set to '1' when the PE can master the LAD bus
User_In.Reset	1	Output	PE Reset Signal. Can be toggled by a host API call
User_In.Rx_Clk_In	1	Output	PCI to PE transmit clock DLL input signals.
User_Out.Data_Out	32	Input	User interface to the output data to the LAD bus
User_Out.Strobe_Out	1	Input	The PE drives this signal to '1' during DMA Init, DMA-From-PE and register read LAD bus cycles to indicate that the requested data has been driven on the LAD bus.
User_Out.DMA_Init	1	Input	The PE drives this signal during a DMA initialization cycle.
User_Out.PeRdy	1	Input	'1' indicates that the PE can accept DMA data '0' indicates that the PE cannot accept DMA data
User_Out.IntReq	1	Input	User interface to the interrupt request signal; Driving this line high for >90ns will create an interrupt to the host.
User_Out.BusReq	1	Input	The PE drives this signal to '1' to request LAD bus mastering. The PCI controller will respond by setting User_In.BusGnt to '1' when mastering is possible.
User_Out.Rx_Clk_Out	1	Input	PCI to PE transmit clock DLL output signals.

7.2.2.1 LAD Bus Transactions

There are six types of LAD Bus transaction cycles: two register space cycles, two DMA init cycles and two DMA data transfer cycles. Register space read cycles and register space write cycles, both of which are initiated by the PCI controller, are generated when the host calls a register read or write API call. The two DMA init cycles are generated by the PE to initialize the PCI controller for DMA transactions.

The DMA-to-PE transaction, a DMA data transfer cycle, is generated by the PCI controller to transfer DMA data to the PE. The DMA-From-PE data transfer cycle is generated by the PE to send DMA data to the PCI controller, and ultimately to the host memory.

7.2.2.2 Register Space Transactions

A typical LAD bus register write access is illustrated in Figure 7-1. The PE will receive an Address, Data, and Reg_Strobe for every piece of data transferred to the PE. User_In.Write will be high for register writes.

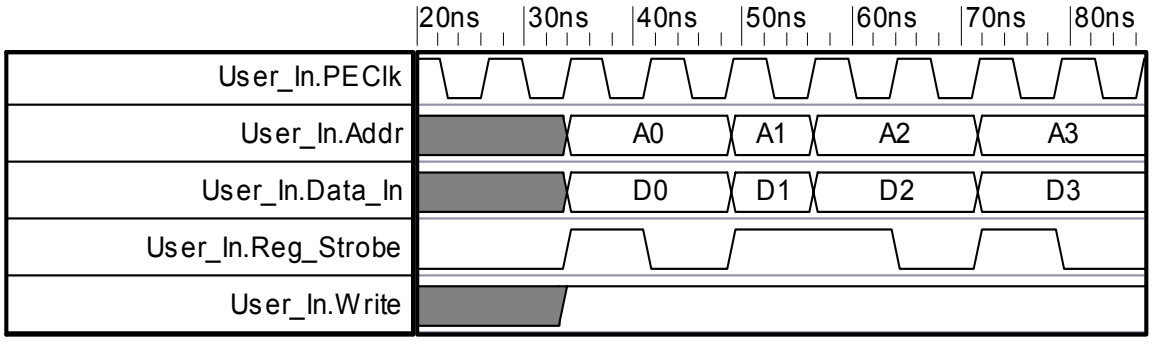


Figure 7-1: Typical LAD Bus Write Cycle from Register Space

Unlike register writes, LAD bus register reads are not bursts. Only a single data word is expected for every address and strobe on the LAD bus. Within ~64 clock cycles of receiving a register read LAD bus transfer, the PE must simultaneously drive LAD_Bus_Out.Data Out with the requested data and User_Out.Strobe out to '1'.

INFORMATION NOTE

For WILDSTAR™-II boards, 9us (approximately 1200) clock cycles are allowed, and for WILDSTAR™-II PRO boards, 1 us (about 120) cycles are allowed. However, long delays will halt all PCI bus traffic and degrade system performance. Delays should be kept as short as possible.

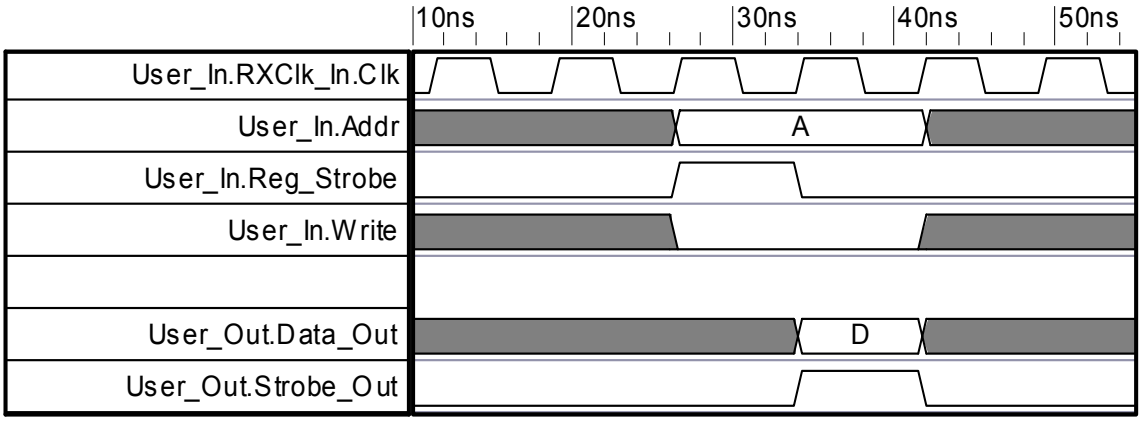


Figure 7-2: Shortest Possible Register Space LAD Bus Read Cycle

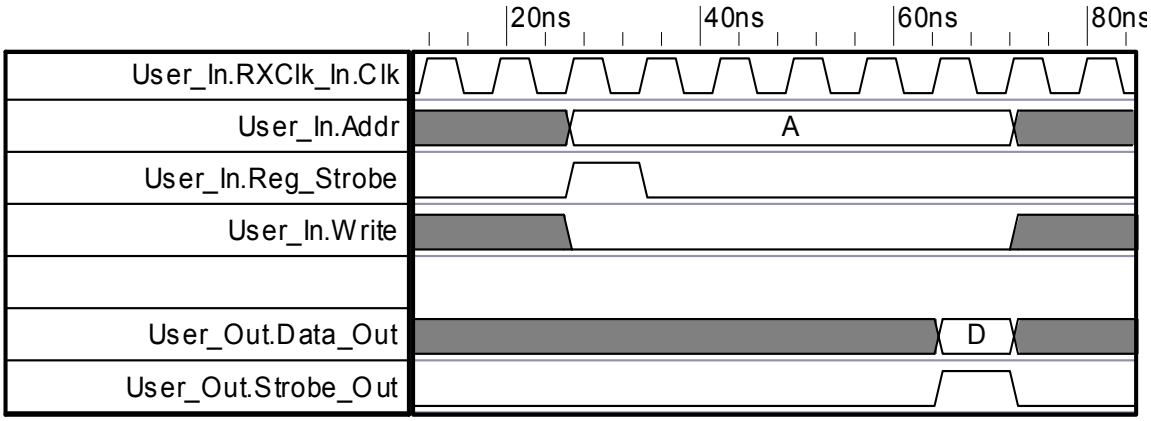


Figure 7-3: Longer Register Space LAD Bus Read Cycle

7.2.2.3 DMA Initialization Cycles

There are two types of DMA initialization LAD bus cycles, known as DMA-To-PE and DMA-From-PE cycles. DMA-To-PE cycles are used to set up a DMA transaction from host memory to the PE. The PE generates both of these cycles on the LAD bus to set up a DMA transaction to host memory.

After sending the DMA-To-PE initialization cycle, the PE will start receiving the requested DMA data. DMA-From-PE cycles are used to set up a DMA transaction from the PE to the host memory. After a DMA-From-PE initialization cycle, the PCI controller will set User_In.PCIRdy high signifying that the PE can start sending DMA data to the PCI via LAD bus DMA-From-PE data cycles.

A DMA-To-PE initialization cycle has the form shown in Figure 7-4 below.

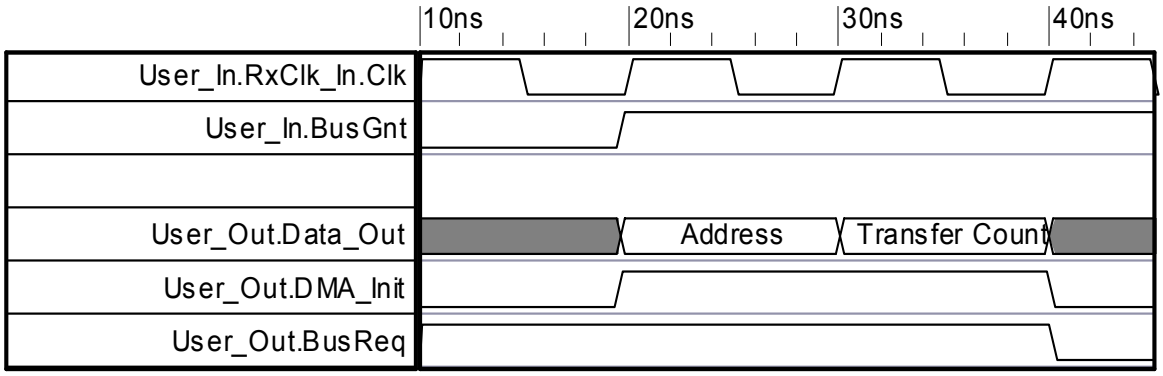


Figure 7-4: DMA-To-PE Initialization Cycle

Before the PE can generate cycles on the LAD bus, it must be granted the bus. To request the bus grant to the PE, the User_Out.BusReq line must be asserted. Some number of clock cycles later, when User_In.BusGnt goes high, the PE can start generating the DMA-To-PE initialization cycle.

The DMA-To-PE initialization cycle consists of first sending a DMA hardware address and then a transfer count to the PCI controller. These two must be sent in consecutive clock cycles.

The transfer count is the number of DWORDS (32-bit values) to transfer from host memory to the PE. The host must have allocated and bound a DMA buffer of at least this many DWORDS before the PE executes the DMA-To-PE initialization cycle. The address is a slightly modified version of the hardware address returned by the WSII_DmaBind API call. The PE must set the bottom two bits of the HardwareAddress to “00”, which tells the PCI controller that this cycle initializes a DMA-To-PE transaction.

Upon completing the initialization cycle, the PE should deassert User_Out.BusReq. The PCI controller then begins fetching the requested data

from memory and sending it to the PE via the DMA-To-PE data cycle described below.

A DMA-From-PE initialization Cycle has the following form:

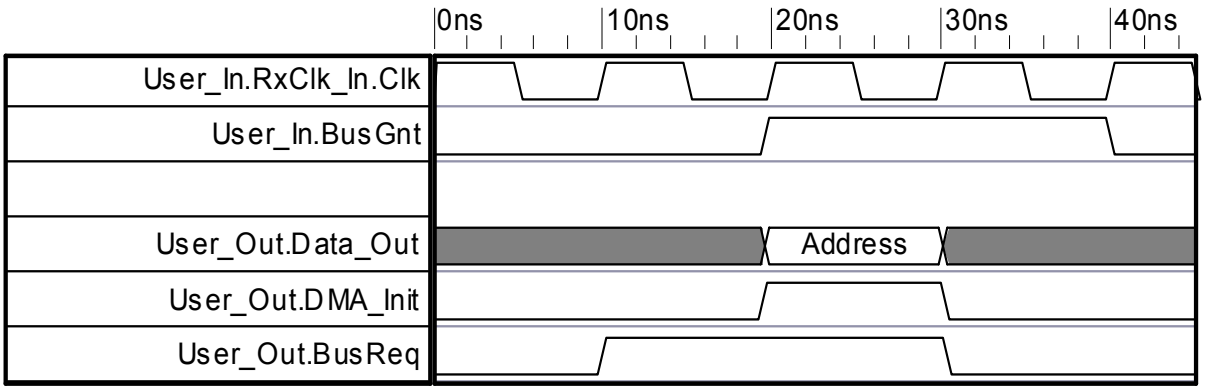


Figure 7-5: DMA-From-PE Initialization Cycle

As with the DMA-To-PE initialization cycles, before the PE may generate cycles on the LAD bus it must be granted the bus.

The DMA-From-PE initialization cycle consists only of sending a DMA hardware address. The number of DWORDs transferred is controlled by the PE. The PCI controller accepts as many pieces of data from the PE as the host has allocated with the WSII_DmaBind API call.

The address is a slightly modified version of the hardware address returned by the WSII_DmaBind API call. The PE must set the bottom two bits of the HardwareAddress to "01". This tells the PCI controller that this cycle initializes a DMA-From-PE transaction.

Upon completion of the initialization cycle the PE should deassert User_Out.BusReq. If the hardware address was valid, the PCI controller will then set User_In.PCIRdy high and the PE can start sending data to the PCI controller via the DMA-From-PE data cycles described below.

7.2.2.4 DMA Data Cycles

While the DMA initialization cycles set up the DMA transfers, the DMA data cycles are used to actually transfer the data. There are two types of DMA data cycles—the DMA-To-PE and DMA-From-PE data cycles. The DMA-To-PE data cycles are driven by the PCI controller, while the PE generates the DMA-From-PE cycles.

Figure 7-6 shows a typical DMA-To-PE data cycle.

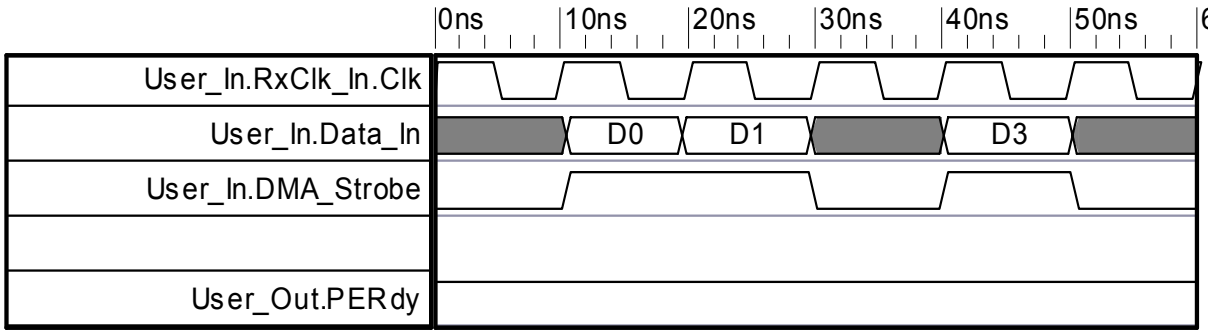


Figure 7-6: DMA-To-PE Data Cycle

For every clock cycle in which User_In.DMA_Strobe is high, the PE is given a new piece of DMA data. The data does not come with a corresponding address; however, the PCI controller will always send the data in order, starting with the address requested in the DMA-To-PE initialization cycle.

The User_Out.PERdy line is used by the PE to tell the PCI controller when it can accept data. The PCI controller will stop sending data to the PE if User_Out.PERdy is set low. The PE, however, must be able to accept at least seven more pieces of data following the deassertion of User_Out.PERdy.

Figure 7-7 shows a typical DMA-From-PE data cycle.

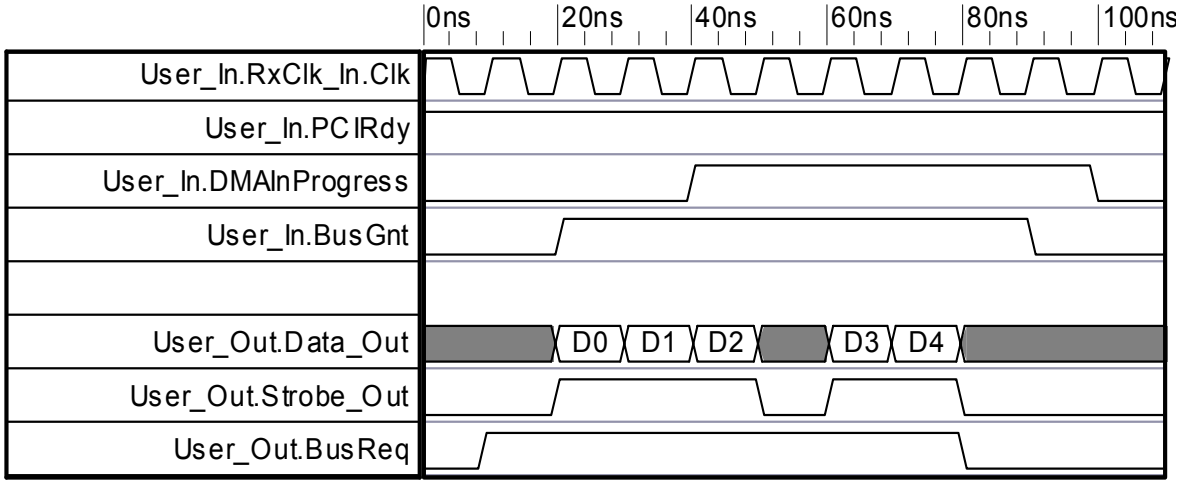


Figure 7-7: DMA-From-PE Data Cycle

Before the PE can start a DMA-From-PE data cycle, it must request the LAD bus by asserting User_Out.BusReq and wait for User_In.BusGnt to go high. Once the bus is granted, the PE sends DMA data to the PCI controller by asserting the Strobe_Out signal.

For every clock that the PCI controller samples User_Out.Strobe_Out high, User_Out.Data_Out is DMA'ed back to the host. This DMA transaction does not

have an address. The PCI controller assumes the data comes in order, starting with the data intended for the starting address given in the DMA-From-PE initialization cycle.

The PCI controller signals its ability to accept DMA data through the User_In.PCIRdy signal. The PE should not request the bus to transmit DMA data if User_In.PCIRdy is not high. Additionally, if User_In.PCIRdy goes low during a DMA-From-PE transaction, the PE must stop sending data within 200 clock cycles.

Once the PE has completed the DMA-From-PE transaction, it should deassert User_Out.BusReq. The User_In.DMAInProgress signal stays active until the PCI controller has transferred all the data to the host memory. This signal may stay high for a number of clock cycles after the PE has finished the DMA-From-PE transaction.

7.2.3 SRAM Basic I/O (SRAMPort_Basic_IO)

The SRAM port consists of one or two of the following SRAM memories:

- Micron® 18Mb DDRb2 (MT57V512H36A)
- Micron® 9Mb DDRb4 (MT57V256H36P)
- Micron® 18 Mb DDRII CIO 2 word burst (MT57W512H36B)
- NEC® 18 Mb DDRII CIO 2 word burst (UPD44164362)
- Samsung® 18 Mb DDRII CIO 2 word burst (K7I163682B-FC20)

The WSII_GetInformation API call is used to determine memory type, burst length and population.

The SRAM basic I/O component provides a simple user interface to the six DDR SRAM ports connected to the WILDSTAR™-II PE device. For additional information for interfacing the DDR or DDRII SRAM, see the datasheets at www.micron.com, www.nec.com, or www.samsung.com.

A description of the SRAM VHDL Example is included in Appendix C of this manual.

The SRAM basic I/O is defined as:

```
component SramPort_Basic_IO is
  generic
  (
    SRAMTYPE  : SRAM_Type;
  ),
  port
  (
    Pads      : inout SramPort_Pads_Type;
    User_In   :   out SramPort_Basic_IO_In_Type;
    User_Out  : in    SramPort_Basic_IO_Out_Type
  );
end component;
```

The SRAMPort_Basic_IO_In_Type and SRAMPort_Basic_IO_Out_Type are defined as:

```
type SramPort_Basic_IO_In_Type is record
  Clk_In    : std_logic;
  Addr_In   : std_logic_vector ( 20 downto 0 );
  Data_In   : std_logic_vector ( 35 downto 0 );
  WE_In_n   : std_logic;
  Load_In_n : std_logic_vector ( 1 downto 0 );
end record;
```

```
type SramPort_Basic_IO_Out_Type is record
  Data_Out  : std_logic_vector ( 35 downto 0 );
  Data_OE_n : std_logic_vector ( 35 downto 0 );
  Addr      : std_logic_vector ( 20 downto 0 );
  BW_n      : std_logic_vector ( 3 downto 0);
  Load_n    : std_logic_vector ( 1 downto 0);
  WE_n      : std_logic;
  Clk       : std_logic;
  Clk_n     : std_logic;
end record;
```

The port signals of the SRAM basic I/O are summarized below. User_In.Addr_In, User_In.WE_In_n, and User_In.Load_In_n can be used to facilitate clocking control signals on the return clock (M_ClkFb). It is possible to cross the P_CLK/M_ClkFb clock boundary inside the PE, but the supplied "sram_DDR2_Core" uses these signals instead.

Table 7-3: SRAM Basic I/O Signal Mapping

Signal Name	Width	Dir	Description	Datasheet Name
User_Out.Addr	20	Input	Address Bus	SA[1:14]*
User_Out.Data_Out	36	Input	Data Bus Output	DQ_
User_Out.Data_OE_n	36	Input	Data Bus Output Enable	NA
User_Out.BW_n	4	Input	Byte Write Enable	BW_#
User_Out.Load_n	2	Input	Load	0=LD#, chip 1 1=LD#,chip 2 (if present)
User_Out.WE_n	1	Input	Write Enable	R/W#
User_Out.Clk	1	Input	Clock	K
User_Out.Clk_n	1	Input	Clock	K#
User_In.Data_In	36	Output	Data Bus Input	DQ_
User_In.Clk_In	1	Output	Clock Input (Input Buffer on User_out.Clk pad)	NA
User_In.Addr_In	20	Output	Addr Input (Input Buffer on User_Out.Addr pad)	NA
User_In.WE_In_n	1	Output	Write Enable Input (Input Buffer on User_Out.WE_n pad)	NA
User_In.Load_In_n	2	Output	Load Input (Input Buffer on User_Out.Load_n pad)	NA

*Note: SA[0] is always grounded.

7.2.4 SRAM DDR2 Core (SRAM_DDR2_Core)

The SRAM DDR2 core is designed to provide a high-level interface to the SRAM memories. The SRAM_Basic_IO is included inside the core, so it is not necessary to include this elsewhere. The SRAM DDR2 core supports the following SRAM memories:

- Micron® 18 Mb DDRII CIO 2 word burst (MT57W512H36B)
- NEC® 18 Mb DDRII CIO 2 word burst (UPD44164362)
- Samsung® 18 Mb DDRII CIO 2 word burst (K7I163682B-FC20)

The SRAM DDR2 Core is defined as:

```
component SRAM_DDR2_Core is
   generic
   (
     SRAMTYPE     : SRAM_Type  := DDRII;
     SRAMMANUF    : SRAM_Manuf := MICRON;
     ADDRZ        : integer    := 0;
     SRAM_PORT    : natural    := 0
   );
   port
   (
     Clk          : in    Clock_In_Type;
     ClkFB        : in    Clock_In_Type;
     Global_Reset : in    std_logic;
     Pads         : inout SramPort_Pads_Type;
     User_In      :   out SRAM_DDR2_Core_In_Type;
     User_Out     : in    SRAM_DDR2_Core_Out_Type
   );
 end component;
```

All signals into and out of this core are synchronous to the "Clk" port. This *must* be set to Clocks_In.P_Clk or the core will produce memory errors. All processing logic in the PE should also run off Clocks_In .P_Clk, since using other clocks can cause jitter on P_Clk and the SRAM signals.

The ClkFB port must be Clocks_In.M_ClkFB, which clocks in read data and which may need to be phase-shifted depending on the clock frequency (same as P_Clk). The equations for phase shifting M_ClkFb are as follows for each memory:

- Micron®: Phase_Shift = ((1.2/((1/fClockFreq)*1000))*256)
- NEC®: Phase_Shift = ((3.5/((1/fClockFreq)*1000))*256)
- Samsung®: Phase_Shift = ((5/((1/fClockFreq)*1000))*256)

It is also important that the makefile include the following line in the BITOPTIONS if using the phase shift:

For **VME** boards: -g Centered_x3y1:0
For **PCI** boards: -g Centered_x2y1:0

This line allows the maximum phase shift to go from 0 to 10ns. Without this, the phase shift is limited to +/- 5ns.

The SRAM_DDR2_Core_In_Type and SRAM_DDR2_Core_Out_Type are defined as:

```
type SRAM_DDR2_Core_Out_Type is record
  RxFifoReset  : std_logic;
  Data_Out     : std_logic_vector ( 71 downto 0 );
  Addr         : std_logic_vector ( 21 downto 0 );
  BWE          : std_logic_vector ( 3 downto 0);
  Strobe       : std_logic;
  Write        : std_logic;
end record;
```

```
type SRAM_DDR2_Core_In_Type is record
  Data_In      : std_logic_vector ( 71 downto 0 );
  Data_Valid   : std_logic;
end record;
```

The port signals of the SRAM DDR2 core are summarized below.

Table 7-4: SRAM DDR2 Signal Mapping

Signal Name	Width	Dir	Description
User_Out.RxFifoReset	1	Input	Reset the internal data receive FIFO (must always be valid, not quantified by strobe)
User_Out.Addr	20	Input	Address Bus
User_Out.Data_Out	32	Input	Data Bus Output
User_Out.BWE	4	Input	Byte Write Enable
User_Out.Strobe	2	Input	Strobe signal to initiate a SRAM access (all other User_Out signals must be valid when high)
User_Out.Write	1	Input	Select a write to SRAM when high
User_In.Data_In	32	Output	Read Data from SRAM
User_In.Data_Valid	1	Output	Read Data Valid from SRAM

To perform an SRAM access, provide valid signals for User_Out (Addr, Data_Out if write, BWE, Write) and set User_Out.Strobe high. Read data will come back on the User_In port and is quantified by the User_In.Data_Valid signal.

Some rules must be followed when using the SRAMs and doing a read followed by a write:

- For all memories, a NOOP must be inserted between a write and read request.
- For the Micron® memories, a second NOOP must be added above 132 MHz.
- For the NEC® memories, a second NOOP must be added above 60 MHz.

- For the Samsung® memories, a second NOOP must be added above 150 MHz.

The SRAM DDR2 core component generics are described below:

- **SRAMTYPE:** Defaults to DDRII. Other options are not supported.
- **SRAMMANUF:** Defaults to MICRON. Other options are NEC, SAMSUNG and CYPRESS.
- **ADDRZ:** Defaults to 0. Should not be changed unless a PCI Rev A board is used.
- **SRAM_PORT**: Must be set for each port. Valid numbers are 0-5.

7.2.5 DRAM Standard Interface (DRAM_Std_IF)

The DRAM_Std_IF standard interface provides the user interface for DRAM.

The DRAM standard interface is defined as:

```
component DramPort_Std_IF is
  port
  (
    Pads         : inout DramPort_Pads_Type;
    P_Clk        : in    Clock_In_Type;
    Global_Reset : in    std_logic;
    User_In      : out   DramPort_Std_IF_In_Type;
    User_Out     : in    DramPort_Std_IF_Out_Type
  );
end component;
```

The DramPort_Std_IF_Out_Type is defined as:

```
type DramPort_Std_IF_Out_Type is record
  Global_Reset    : std_logic;
  Addr            : std_logic_vector(24 downto 0);
  data_out        : std_logic_vector(63 downto 0);
  write           : std_logic;
  strobe          : std_logic;
end record;
```

The DramPort_Std_IF_In_Type is defined as:

```
type DramPort_Std_IF_In_Type is record

  data_in          : std_logic_vector(63 downto 0);
  data_valid       : std_logic;
  write_rdy        : std_logic;
  read_rdy         : std_logic;
end record;
```

The port signals of the DRAM standard interface are shown below.

Table 7-5: DRAM Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Global_Reset	1	Input	Global reset (or set) signal
P_Clk	1	Input	Input Clock. All data bits are synchronous to this clock.
Pads	NA	Bi-Dir	DRAM Pad signals
User_Out.Data_Out	64	Input	User Data to be outputted by DRAM
User_In.Data_In	64	Output	Data from the DRAM to the user
User_Out.write	1	Output	User Write control signal
User_Out.Strobe	1	Output	User Strobe control signal
User_In.write_rdy	64	Output	User Write Ready signal
User_Out.read_rdy	1	Output	User Read Ready signal
User_Out.addr	24	Output	User Strobe control signal

To write data to the SDRAM, first check that the 'write_rdy' signal is high. This step ensures that the write FIFOs are not full. If 'write_rdy' is high, the interface is ready to accept write data. Next, put the address on the 'addr' lines and the 64-bit data on the 'data_out' lines and then pulse 'strobe' and 'write'. Data can be sent in continuously as long as 'write_rdy' is high.

To read data back, first check that 'read_rdy' is high to make sure that the read address FIFOs are not full. Then, put the address on the 'addr' lines and pulse 'strobe'. Again, the address of data to be read can be continuously written into the interface as long as 'read_rdy' is high. Soon afterwards, the 'data_valid' signal will go high and the 64-bit read data will be available on the 'data_in' lines. 'Data valid' will stay high as long as data is still being read out.

7.2.6 DRAM Basic I/O (DRAMPort_Basic_IO)

The DRAM port consists of two parallel Micron® 256Mb x16 DDR SDRAM chips (MT46V16M16). Chip 1 stores the lower 16 bits, and chip 2 stores the upper 16 bits. The DRAM basic I/O component provides a simple user interface to the DDR SDRAM port connected to the WILDSTAR™-II PE device. For additional information for interfacing the DDR SDRAM, see the datasheets at www.micron.com.

The DRAM basic I/O is defined as:

```
  component DramPort_Basic_IO is
   port
   (
     Pads      : inout DramPort_Pads_Type;
     User_In   :   out DramPort_Basic_IO_In_Type;
     User_Out  : in    DramPort_Basic_IO_Out_Type
   );
 end component;
```

The DRAMPort_Basic_IO_In_Type and DRAMPort_Basic_IO_Out_Type are defined as:

```
type DramPort_Basic_IO_In_Type is record
  Data_In     : std_logic_vector ( 31 downto 0);
  DQS_In      : std_logic_vector (  3 downto 0);
end record;
```

```
type DramPort_Basic_IO_Out_Type is record
  Addr      : std_logic_vector ( 14 downto 0 );
  Data_Out  : std_logic_vector ( 31 downto 0 );
  Data_OE_n : std_logic_vector ( 31 downto 0 );
  BA        : std_logic_vector (  1 downto 0 );
  DM        : std_logic_vector (  1 downto 0 );
  DQS_Out   : std_logic_vector (  3 downto 0 );
  DQS_OE_n  : std_logic_vector (  3 downto 0 );
  RAS_n     : std_logic;
  CAS_n     : std_logic;
  CS_n      : std_logic;
  WE_n      : std_logic;
  CKE       : std_logic;
  Clk       : std_logic;
  Clk_n     : std_logic;
end record;
```

The port signals of the DRAM basic I/O are summarized below.

Table 7-6: DRAM Basic I/O Signal Mapping

Signal Name	Width	Dir	Description	Datasheet Name
User_Out.Addr	14*	Input	Address Bus	A[0:14]*
User_Out.Data_Out	32	Input	Data Bus Output	DQ[0:31]
User_Out.Data_OE_n	32	Input	Data Bus Output Enable	NA
User_Out.BA	2	Input	Bank Address	BA[0-1]
User_Out.DM	2	Input	Data Mask	0=LDM, 1=UDM
User_Out.DQS_Out	4	Input	Data Strobe Output	0=LDQS, Chip 1 1=UDQS, Chip 1 2=LDQS, Chip 2 3=UDQS, Chip 2
User_Out.DQS_OE_n	4	Input	Data Strobe Output Enable	NA
User_Out.RAS_n	1	Input	Row Address Strobe	RAS#
User_Out.CAS_n	1	Input	Column Address Strobe	CAS#
User_Out.WE_n	1	Input	Write Enable	WE#
User_Out.CKE	1	Input	Clock Enable	CKE
User_Out.Clk	1	Input	Clock	CK
User_Out.Clk_n	1	Input	Clock	CK#
User_In.Data_In	32	Output	Data Bus Input	DQ[0:31]
User_In.DQS_In	4	Output	Data Strobe Input	0=LDQS, Chip 0 1=UDQS, Chip 0 2=LDQS, Chip 1 3=UDQS, Chip 1

*On WILDSTAR™-II PCI Rev A, only A[0:12] can be used

7.2.7 LED Standard Interface (LED_Std_IF)

The LED_Std_IF standard interface provides the user interface to the three light emitting diodes (LEDs) of the WILDSTAR™-II PE device. The LEDs can be used to indicate status or processing activity, and can even be used to help debug a PE design.

The LED standard interface is defined as:

```
component LED_Std_IF is
  port
  (
    Pads         : inout LED_Pads_Type;
    Clk          : in    std_logic;
    Global_Reset : in    std_logic;
    User_Out     : in    LED_Std_IF_Out_Type
  );
end component;
```

The LED_Std_IF_Out_Type is defined as:

```
type LED_Std_IF_Out_Type is record
  Red       : std_logic;
  Green     : std_logic;
  Yellow    : std_logic;
end record;
```

The port signals of the LED standard interface are shown below. The "User_Out" signals should be used to drive the LEDs of the PE device.

Table 7-7: LED Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads	3	Output	LED pad signals
User_Out.Red	1	Input	User interface to the red LED, '1' = ON
User_Out.Green	1	Input	User interface to the green LED, '1' = ON
User_Out.Yellow	1	Input	User interface to the Yellow LED, '1' = ON

7.2.8 I/O Connector Basic Interface

The I/O card connector interface components provide the user an interface to the external I/O card devices. Two interface components are available: IO_ConnWSII_Basic_IO and IO_ConnWS_Basic_IO. The IO_ConnWSII_ Basic_IO should be used when interfacing a newer generation I/O card (i.e., Fibre Channel and Gigabit Ethernet), while the IO_ConnWS_ Basic_IO is needed for the older generation I/O cards.

7.2.8.1 IO_ConnWS_Basic_IO

The IO_ConnWS_Std_IF component provides the user interface to 153 bi-directional connections to external I/O cards, two clock synchronizing signals, four DLL control signals, and five I/O card clock signals. The 153-bit bus may be used to transfer data between the WILDSTAR™-II and an external I/O card. The direction of each bit is controlled by the PE though individual output enable lines.

The two clock synchronizing lines, TxClk and RxClk, are dedicated for source-synchronous I/O data clocking, TxClk being the transmit clock and RxClk the receive clock.

The four DLL control signals, Dll_Reset_In, Dll_Reset_Out, Dll_Locked_In, and Dll_Locked_Out, are dedicated for I/O card PEs to reset WILDSTAR™-II PE DLLs, and for WILDSTAR™-II PEs to reset I/O card PE DLLs.

The five clock signals, UClk_Out, MClk_Out, UClk_In, MClk_In, and UClk_Mstr are used to source the M_Clk and U_Clk signals in the I/O card PE. MClk_Out and UClk_Out become U_Clk and M_Clk respectively in the I/O Card

PE. MClk_In and U_Clk in are driven by the U clock source and M clock source of the I/O card PE. (See the Clock Standard interface in the appropriate I/O card reference manual on setting the U clock source and M clock source from the I/O card PE.)

The UClk_Mstr line is currently reserved, and should not be used.

The IO_ConnWS_Basic_IO is defined as:

```
component IO_ConnWS_Basic_IO is
  port
  (
    Pads      : inout IO_ConnWSII_Pads_Type;
    User_In   :   out IO_ConnWS_Basic_IO_In_Type;
    User_Out  : in    IO_ConnWS_Basic_IO_Out_Type
  );
end component;
```

The User_In and User_Out signals should be used in the user logic to send data to the I/O card. The record types, IO_ConnWS_Basic_IO_In_Type and IO_ConnWS_Basic_IO_Out_Type, are defined as:

```
type IO_ConnWS_Basic_IO_In_Type is record
  Dll_Reset_In    : std_logic;
  Dll_Locked_In   : std_logic;
  UClk_In         : std_logic;
  UClk_Mstr       : std_logic;
  MClk_In         : std_logic;
  Data_In         : std_logic_vector ( 152 downto 0 );
  RxClk           : std_logic;
end record;;
```

```
type IO_ConnWS_Basic_IO_Out_Type is record
  Dll_Reset_Out   : std_logic;
  Dll_Locked_Out  : std_logic;
UClk_Out        : std_logic;
  MClk_Out        : std_logic;
  Data_Out        : std_logic_vector ( 152 downto 0 );
  Data_OE_n       : std_logic_vector ( 152 downto 0 );
  TxClk           : std_logic;
end record;;
```

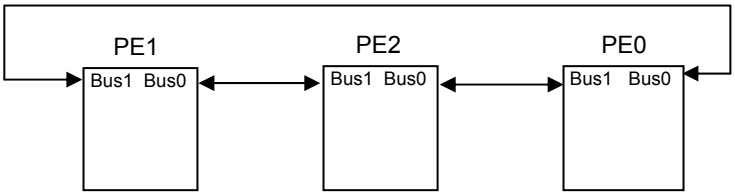
Table 7-8: I/O Connector Basic Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads	NA	Bi-dir	I/O Connector Pad signals
User_In.Data_In	153	Output	User interface to the I/O connector data input
User_In.DLL_Reset_In	1	Output	Dedicated Input Line: Suggested use as a DLL reset signal.
User_In.DLL_Locked_In	1	Output	Dedicated Input Line: Suggested use as a DLL locked signal.

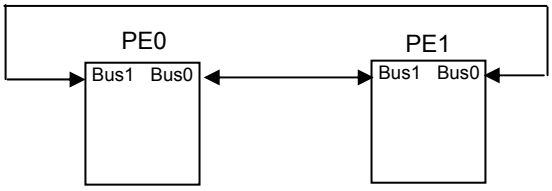
Signal Name	Width	Dir	Description
User_In.MClk_In	1	Output	Dedicated Input Line: Suggested use as an M Clock feedback line from the I/O PE.
User_In.UClk_In	1	Output	Dedicated Input Line: Suggested use as an U Clock feedback line from the I/O PE.
User_In.UClk_Mstr	1	Output	RESERVED: Should not be used.
User_In.RxClk	1	Output	Dedicated Input Line: Suggested use as the receive clock for the User_In.Data_In Lines.
User_Out.Data_Out	153	Input	User interface to the I/O connector data output
User_Out.Data_OE_n	153	Input	User interface to the I/O connector data output enables; each data bit driver is enabled when its respective output enable is '0' or disabled (for reading) when '1'
User_Out.MClk_out	1	Input	The WILDSTAR™-II PE supplies M clock and U clock to the I/O card PE. This signal becomes M Clock for the I/O card.
User_Out.UClk_out	1	Input	The WILDSTAR™-II PE supplies M clock and U clock to the I/O card PE. This signal becomes U Clock for the I/O card.
User_Out.DLL_Reset_Out	1	Input	Dedicated Output Line: Suggested use as a DLL reset signal.
User_Out.DLL_Locked_Out	1	Input	Dedicated Output Line: Suggested use as a DLL locked signal.
User_Out.MClk_Out	1	Input	Connecte to the M Clock input pin on the I/O PE. This signal will become M Clock for the I/O board.
User_Out.UClk_Out	1	Input	Connecte to the U Clock input pin on the I/O PE. This signal will become U Clock for the I/O board.
User_Out.TxClk	1	Input	Dedicated Output Line: Suggested use as the transmit clock for the User_Out.Data_Out Lines.

7.2.9 PE to PE Bus Standard Interface (PE_Bus_Std_IF)

Each PE has two 174-pin buses connecting it with adjacent PEs. Below are diagrams showing PE bus connections on the WILDSTAR™-II /VME and /PCI boards.



WILDSTAR™-II /VME PE Bus Connections



WILDSTAR™-II /PCI PE Bus Connections

The PE_Bus_Std_IF divides 174 pins as follows:

160 pins	Divided into five groups of 32 signals labeled GroupA to GroupE
6 pins	Three groups of 2 signals labeled Group2A to Group2C
2 pins	Transmit clock
2 pins	Receive clock
4 pins	Four DLL Status Signals

The component declaration for the PE_Bus_Std_If is:

```
component PE_Bus_Std_IF is
  generic
  (
    GROUPA_DIR      : DIR_Type;
    GROUPB_DIR      : DIR_Type;
    GROUPC_DIR      : DIR_Type;
    GROUPD_DIR      : DIR_Type;
    GROUPE_DIR      : DIR_Type;
    GROUP2A_DIR     : DIR_Type;
    GROUP2B_DIR     : DIR_Type;
    GROUP2C_DIR     : DIR_Type;
    PLATFORM        : PlatformType;
    PEPORT          : integer
  );
  port
  (
    Clk           : in    Clock_In_Type;
    Global_Reset  : in    std_logic;
    Pads          : inout PE_Bus_Pads_Type;
    User_In       :   out PE_Bus_Std_IF_In_Type;
    User_Out      : in    PE_Bus_Std_IF_Out_Type
  );
end component;
```

The PE_Bus_Std_IF_In_Type, PE_Bus_Std_IF_Out_Type and associated records are defined as:

```
type PE_Bus_Std_IF_In_Type is record
  RxReset_In      : std_logic;
  TxReady_In      : std_logic;
  Data_Valid      : std_logic;
  Data_In         : PEData_Type;
end record;
```

```
type PE_Bus_Std_IF_Out_Type is record
  TxReset_Out    : std_logic;
  Data_Out       : PEData_Type;
end record;
```

```
type PEData_Type is record
   BusA         : std_logic_vector ( 31 downto 0 );
   BusB         : std_logic_vector ( 31 downto 0 );
   BusC         : std_logic_vector ( 31 downto 0 );
   BusD         : std_logic_vector ( 31 downto 0 );
   BusE         : std_logic_vector ( 31 downto 0 );
   Bus2A        : std_logic_vector (  1 downto 0 );
   Bus2B        : std_logic_vector (  1 downto 0 );
   Bus2C        : std_logic_vector (  1 downto 0 );
 end record;
```

The direction of each group (into the PE or out of the PE) is set by the eight generics in the interface.

- **PLATFORM, PEPORT:** These generics are used to configure the interface for a specific port. They should not be modified.
- **GROUPA_DIR:** Set to INPUT for the PE to receive the GroupA signals. Set to OUTPUT for the PE to transmit the GroupA signals.
- **GROUPB_DIR:** Set to INPUT for the PE to receive the GroupB signals. Set to OUTPUT for the PE to transmit the GroupB signals.
- **GROUPC_DIR:** Set to INPUT for the PE to receive the GroupC signals. Set to OUTPUT for the PE to transmit the GroupC signals.
- **GROUPD_DIR:** Set to INPUT for the PE to receive the GroupD signals. Set to OUTPUT for the PE to transmit the GroupD signals.
- **GROUPE_DIR:** Set to INPUT for the PE to receive the GroupE signals. Set to OUTPUT for the PE to transmit the GroupE signals.

The same instructions apply to GROUP2A_DIR, GROUP2B_DIR, and GROUP2C_DIR.

The Clk and Clk90 ports should be driven by the desired transmit clock. Although all input data is not received synchronous to this transmit clock, the data is passed through a FIFO, so that the User_In data is synchronous to the transmit clock.

The port signals of the PE Bus standard interface are shown below. The "User_Out" and "User_In" signals should be used to read and write to the PE busses.

Table 7-9: PE Bus Standard Interface Port Signals

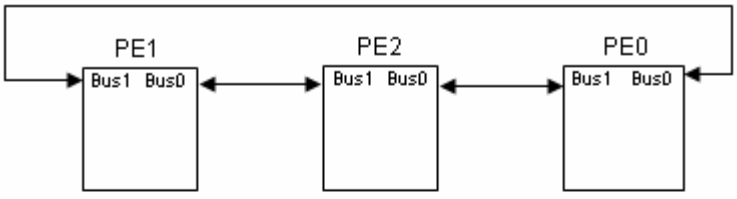
Signal Name	Width	Dir	Description
Pads	NA	Bi-Dir	PE bus pad signals
Global_Reset	1	Input	Global reset signal
Clk	1	Input	Input Clock. All data bits are synchronous to this clock.
User_Out.Data_Out.BusA	32	Input	User interface to the Group A output pins.
User_Out.Data_Out.BusB	32	Input	User interface to the Group B output pins.
User_Out.Data_Out.BusC	32	Input	User interface to the Group C output pins.

Signal Name	Width	Dir	Description
User_Out.Data_Out.BusD	32	Input	User interface to the Group D output pins.
User_Out.Data_Out.BusE	32	Input	User interface to the Group E output pins.
User_Out.Data_Out.BusA	2	Input	User interface to the Group 2A output pins.
User_Out.Data_Out.Bus2B	2	Input	User interface to the Group 2B output pins.
User_Out.Data_Out.Bus2C	2	Input	User interface to the Group 2C output pins.
User_Out.TxResetOut	1	Output	A '1' on this port resets the transmitting DCM.
User_In.Data_In.BusA	32	Output	User interface to the Group A input pins.
User_In.Data_In.BusB	32	Output	User interface to the Group B input pins.
User_In.Data_In.BusC	32	Output	User interface to the Group C input pins.
User_In.Data_In.BusD	32	Output	User interface to the Group D input pins.
User_In.Data_In.BusE	32	Output	User interface to the Group E input pins.
User_In.Data_In.BusA	2	Output	User interface to the Group 2A input pins.
User_In.Data_In.Bus2B	2	Output	User interface to the Group 2B input pins.
User_In.Data_In.Bus2C	2	Output	User interface to the Group 2C input pins.
User_In.Data_Valid	1	Output	Asserted '1' when data on User_In.Data_In is unique and valid.
User_In.RxReset_In	1	Output	Asserted '1' when the receiving DCM is in reset.
User_In.TxReset_In	1	Output	Asserted '1' when the transmitting DCM is in reset.

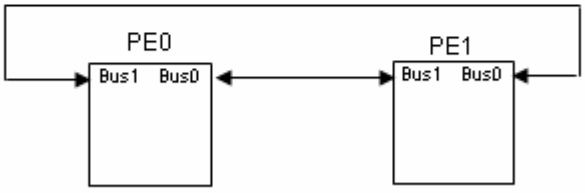
7.2.10 PE to PE Bus Basic Interface (PE_Bus_Basic_IO)

The PE_Bus_Basic_IO provides additional flexibility on the PE to PE bus if it is needed. This interface is consists only of differential input and output buffers.

Below are diagrams showing PE bus connections on the WILDSTAR™-II /VME and /PCI boards.



WILDSTAR™-II /VME PE Bus Connections



WILDSTAR™-II /PCI PE Bus Connections

The PE_Bus_Basic_IO divides 174 pins as follows:

166 pins	Divided into 83 differential pairs.
2 pins	Transmit clock
2 pins	Receive clock
4 pins	Four DLL Status Signals

The component declaration for the PE_Bus_Basic_IO is:

```
component PE_Bus_Basic_IO is
  generic
  (
    OUTPUT_ENABLE   : std_logic_vector(83 downto 0)
  );
  port
  (
    Pads      : inout PE_Bus_Pads_Type;
    User_In   :   out PE_Bus_Basic_IO_In_Type;
    User_Out  : in    PE_Bus_Basic_IO_Out_Type
  );
end component;
```

The PE_Bus_Basic_IO_In_Type, PE_Bus_Basic_IO_Out_Type and associated records are defined as:

```
type PE_Bus_Basic_IO_In_Type is record
 Data_In          : std_logic_vector ( 83 downto 0 )
 Clocks_In        : PE_Bus_Basic_IO_Clock_In_Type;
end record;
```

```
type PE_Bus_Basic_IO_Out_Type is record
  Data_Out         : std_logic_vector ( 83 downto 0 )
  Clocks_Out       : PE_Bus_Basic_IO_Clock_Out_Type;
end record;
```

```
type PE_Bus_Basic_IO_Clock_In_Type is record
  Dll_Reset_In    : std_logic;
  Dll_Locked_In   : std_logic;
  RxClk           : std_logic;
end record;
```

```
type PE_Bus_Basic_IO_Clock_Out_Type is record
  Dll_Reset_Out   : std_logic;
  Dll_Locked_Out  : std_logic;
  TxClk           : std_logic;
end record;
```

The direction of each bit group (into the PE or out of the PE) is set by the OUTPUT_ENABLE generics in the interface. Each bit in this generic configures the corresponding data bit (i.e., bit 13 configures data bit 13 to be input or output). Set the bit to 1 for an output or a 0 for input.

The port signals of the PE Bus basic interface are shown below. The "User_Out" and "User_In" signals should be used to read and write to the PE busses.

Table 7-10: PE Bus Standard Interface Port Signals

Signal Name	Width	Dir	Description
Pads	NA	Bi-Dir	PE bus pad signals
Global_Reset	1	Input	Global reset signal
Clk	1	Input	Input Clock. All data bits are synchronous to this clock.
User_Out.Data_Out	83	Input	User interface to the output pins.
User_Out.DLL_Reset_Out	1	Input	Dedicated Output Line: Suggested use as a DLL reset signal.
User_Out.DLL_Locked_Out	1	Input	Dedicated Output Line: Suggested use as a DLL locked signal.
User_Out.TxClk	1	Input	Dedicated Output Line: Suggested use as the transmit clock for the User_Out.Data_Out Lines.
User_In.Data_In	83	Output	User interface to the input pins.
User_In.DLL_Reset_In	1	Output	Dedicated Input Line: Suggested use as a DLL reset signal.
User_In.DLL_Locked_In	1	Output	Dedicated Input Line: Suggested use as a DLL locked signal.
User_In.RxClk	1	Output	Dedicated Input Line: Suggested use as the receive clock for the User_In.Data_In Lines.

7.2.10.1 P0 Backplane Interface (P0_Backplane_Std_IF)

On WILDSTAR™-II /VME, the PE2 processing element has access to the P0 and P2 buses of the VME backplane. The P0_Backplane_Std_IF component is used to communicate with the P0 backplane.

The P0_Backplane_Std_IF is defined as follows:

```
component P0_Backplane_Std_IF is
  generic
  (
    DIFFERENTIAL    : boolean;
    P2_DIFFERENTIAL : boolean;
    OUTPUT_ENABLE   : std_logic_vector(86 downto 0)
  );
  port
  (
    Clk           : in    std_logic;
    Global_Reset  : in    std_logic;
    Pads          : inout P0_Backplane_Pads_Type;
    User_In       :   out P0_Backplane_Std_If_In_Type;
    User_Out      : in    P0_Backplane_Std_If_Out_Type
  );
end component;
```

PE designs should use the User_In and User_Out ports to communicate with the P0 backplane. These records are defined below.

```
type P0_Backplane_Std_If_In_Type is record
  Data_In      : std_logic_vector ( 86 downto 0 );
  RxClk        : std_logic;
end record;

type P0_Backplane_Std_If_Out_Type is record
  Data_Out     : std_logic_vector ( 86 downto 0 );
  TxClk        : std_logic;
end record;
```

The P0_Backplane_Std_IF divides the P0 bus into two clock lines and 36 differential signals or 87 single-ended series terminated signals, depending on the board configuration. The User_In.Data_In port receives data from the P0 backplane, while the User_Out.Data_Out port drives data onto the P0 backplane. All of these signals are registered in the IOB of their respective pad on the signal rising edge on the Clk port.

When set for differential operation, only bits 0 to 35 of User_In.Data_In and User_Out.Data_Out can be used. When set to series terminated mode, all 87 bits are usable only if the P2 backplane interface is not also in series terminated mode. If both the P2 and P0 backplanes are series terminated, only bits 0 to 71 are usable.

The RxClk line is a special input line that is not registered in the IOB. Therefore, it may be used to receive a clock from the P0 backplane. Similarly, the TxClk output line is not registered in the IOB, and can be used to transmit a clock on the P0 backplane.

The P0_Backplane_Std_IF has three generics to configure the interface.

- **DIFFERENTIAL:** This generic can be set to TRUE or FALSE. Set this to TRUE to configure the interface for differential signaling or set it to FALSE for single-ended signaling.

- **P2_DIFFERENTIAL:** This generic can be set to TRUE or FALSE. Set this to TRUE if the P2 interface is configured for differential signaling, or set it to FALSE if the P2 interface is configured for single-ended signaling.

- **OUTPUT_ENABLE:** This generic is used to configure the direction of each data bit in the interface. Each bit in this generic configures the corresponding data bit. (i.e., bit 13 configures data bit 13 to be input or output) Set the bit to 1 for an output or a 0 for input.

The following table summarizes the ports to the P0_Backplane_Std_IF.

Table 7-11: P0_Backplane_Std_IF Ports

Signal Name	Width	Dir	Description
Global_Reset	1	Input	Global reset (or set) signal
Clk	1	Input	Input Clock. All data bits are synchronous to this clock.
Pads	NA	Bi-Dir	P0 Backplane Pad signals
User_Out.Data_Out	87	Input	User interface to the P0 backplane output pins.
User_Out.TxClk	1	Input	Special un-registered clock output pin.
User_In.Data_In	87	Output	User interface to the P0 backplane input pins.
User_In.RxClk	1	Output	Special un-registered clock input pin.

See Appendix A for the mapping of these signals to their respective P0 backplane pins.

7.2.10.2 P2 Backplane Interface (P2_Backplane_Std_IF)

On WILDSTAR™-II /VME, the PE2 processing element has access to the P0 and P2 buses of the VME backplane. The P2_Backplane_Std_IF component is used to communicate with the P2 backplane.

The P2_Backplane_Std_IF is defined as follows:

```
component P2_Backplane_Std_IF is
  generic
  (
    DIFFERENTIAL    : Boolean   ;
    OUTPUT_ENABLE   : std_logic_vector(91 downto 0)
  );
  port
  (
    Clk           : in    std_logic;
    Global_Reset  : in    std_logic;
    Pads          : inout P2_Backplane_Pads_Type;
    User_In       :   out P2_Backplane_Std_If_In_Type;
    User_Out      : in    P2_Backplane_Std_If_Out_Type
  );
end component;
```

PE designs should use the User_In and User_Out ports to communicate with the P2 backplane. These records are defined below.

```
  type P2_Backplane_Std_If_In_Type is record
    Data_In      : std_logic_vector ( 91 downto 0 );
  end record;

  type P2_Backplane_Std_If_Out_Type is record
    Data_Out     : std_logic_vector ( 91 downto 0 );
  end record;
```

The P2_Backplane_Std_IF divides the P2 bus into 46 differential signals or 92 single-ended series terminated signals, depending on the board configuration. The User_In.Data_In port is used to receive data from the P2 backplane, while the User_Out.Data_Out port is used to drive data onto the P2 backplane. All of these signals are registered in the IOB of their respective pad on the rising edge of the signal on the Clk port.

When set for differential operation, only bits 0 to 45 of User_In.Data_In and User_Out.Data_Out can be used. When set to series terminated mode, all 92 bits are usable.

The P2_Backplane_Std_IF has three generics to configure the interface.

- **DIFFERENTIAL:** This generic can be set to TRUE or FALSE. Set this to TRUE to configure the interface for differential signaling of set it to FALSE for single-ended signaling.

- **P2_DIFFERENTIAL:** This generic can be set to TRUE or FALSE. Set this to TRUE if the P2 interface is configured for differential signaling, or set it to FALSE if the P2 interface is configured for single-ended signaling.

- **OUTPUT_ENABLE:** This generic is used to configure the direction of each data bit in the interface. Each bit in this generic configures the corresponding data bit (i.e., bit 13 configures data bit 13 to be input or output). Set the bit to 1 for an output or a 0 for input.

The following table summarizes the ports to the P2_Backplane_Std_IF.

Table 7-12: P2_Backplane_Std_IF Ports

Signal Name	Width	Dir	Description
Global_Reset	1	Input	Global reset (or set) signal
Clk	1	Input	Input Clock. All data bits are synchronous to this clock.
Pads	NA	Bi-Dir	P2 Backplane pad signals
User_Out.Data_Out	87	Input	User interface to the P2 backplane output pins.
User_In.Data_In	87	Output	User interface to the P2 backplane input pins.

7.3 WILDSTAR™-II PRO VHDL Model

The WILDSTAR™-II PRO VHDL Model provides an accurate means of describing the behavior of the WILDSTAR™-II PRO system. The Model encompasses the WILDSTAR™-II PRO device and a simulation model of the host system.

The Model can be used to design the functionality of the reconfigurable processing elements (PEs) residing in the WILDSTAR™-II PRO system. It can then simulate the behavior of the entire system to verify the functionality of the completed PE design.

The Model consists of two main components, a *simulation environment* and *PE interfaces*, both of which are described below.

7.3.1 VHDL Model Simulation Environment

The WILDSTAR™-II PRO VHDL Model provides a full system simulation environment for debugging designs. This environment encompasses all functionality within the WILDSTAR™-II PRO system, providing the designer with an accurate test bed for completing PE designs. As a result, processing element designs can be validated prior to synthesis, placement, and routing.

The simulation environment consists of three levels:

- **PE Design level**
- **Board level**
- **System level**

7.3.1.1 PE Design Level

The PE design is only the first level in the simulation environment. The purpose of this level is solely to drive the pins of the PE, and is therefore completely synthesizable. Code for this level will eventually become the bitstream that programs the PE on the board.

Both the PE interfaces and the user logic for driving the PE pins are contained in this level. The logic used at this level can include LAD bus logic, memory access logic, I/O logic and all other logic for processing data.

7.3.1.2 Board Level

Each PE is contained on a specific mainboard or I/O card. These boards contain components such as memories, buses, and I/O ports that interact with the PE. The board level simulation environment simulates all of these board components and provides the correct stimulus to the PE pads.

The board level also provides interconnections between PEs on a multi-PE board. The board level simulation environment comes completely written, and the user need only set configuration options.

7.3.1.3 System Level

The highest simulation environment level is the system level, which is used when one or more boards are placed into a host computer and connected together or connected to external data I/O. The system level simulation environment provides a means to simulate the host computer, place multiple boards in the system, and arbitrarily interconnect the boards to each other and to external sources.

The host computer is simulated through a set of VHDL procedures that mirror actual C API calls, such as register reading/writing, DMA reading/writing and interrupt handling. The user creates a 'program' using these VHDL procedures. This program creates stimulus on a simulated PCI bus, which is used by a simulated PCI controller on each board level to drive signals on the PE LAD bus pads.

The system level also provides signals connected to I/O ports on a board that can be driven or read by user logic, or simply connected to other I/O ports.

7.4 VHDL Interface Components

Each PE processing module on the WILDSTAR™-II PRO board consists of numerous auxiliary components—clocks, memories, other PEs, and the host interface (LAD) controller—all of which can be used in any design.

Supplied with the VHDL model described are interfaces to each of these components. Each interface component contains logic necessary to communicate with the component and simplifies the interface presented to the user. The user logic is presented with two records—one containing all input standard interface signals, the other containing all output standard interface signals.

Each interface component includes User_In, User_Out, and Pads ports. The Pads ports connect the component to the pads of the PE. The PE pads are divided into logical sets by record type. For example, a Pads.Clocks record maps to all the clock pads, and a Pads.LAD_Bus record maps to all the LAD bus pads. The Pads port of each interface component will be assigned one of these Pads records.

The User_In port is a record of signals that are outputs of the component, but inputs to user logic. Similarly, the User_Out port is a record of signals that are inputs to the component, but outputs of user logic.
In addition to User_In, User_Out, and Pads, each interface component has one or more clock ports. All logic into and out of the interface is synchronous to one of these clocks. Additionally, all Pad output signals are registered in the IOB on their appropriate clock edge.

In addition to User_In, User_Out, and Pads, each core component has a various clock ports. All logic into and out of the interface will be synchronous to one of these clock ports.

7.4.1 Clock Interface (Clock_Interface)



INFORMATION NOTE

It is strongly encouraged that you use only a single clock in a PE design. Use of multiple clocks within a single PE may cause severe clock jitter and application failure. P_clock is recommended for most processing tasks.

Clock_interface provides you with an interface to the various clock pads and other clock-related information in the VHDL design. The component declaration is shown below:

```
component clock_interface is
generic (
    P_CLK_DCM_FREQUENCY_MODE    : string    := "LOW"
);
port (
    pads                        : inout clock_pads_type;
    reset                       : in std_logic;
    reserved                    : inout reserved_type;
    user_in                     : out clock_interface_in_type;
    user_out                    : in clock_interface_out_type
);
end component;
```

The clock_interface component has four port signals: pads, reserved, user_in, and user_out. The Pads port is the component interface to the PE clock pads. This port should always be assigned to 'pads.clocks'. The Reserved port is for "behind-the-scene" signals. This port should always be assigned to 'reserved'. The Reset port simply needs to be connected to the main reset signal.

The user_in port, a record of type clock_interface_in_type, is the user interface to the clock inputs. These signals should be used as the clock sources of the design.

Shown below is the definition of the clock_interface_in_type record.

```
type clock_interface_in_type is record
    p_clock         : clock_in_type;
    ref48_clock     : std_logic;
    ONEPPS_clock    : std_logic;
end record;
```

Ref48_clock is a reference clock permanently set at 48 MHz. P_clock is the main processing clock that most of the user code runs on. P_clock uses a Xilinx® DCM, which helps eliminate the effects of clock skew across the FPGA device, but it

takes time to acquire a lock to the feedback (or reference) clock. The "locked" indicator flags are provided to the design so you can determine when the DCM has acquired a lock. The host also has access to this information. ONEPPS_clock is the clock that is generated from the ONEPPS connector on the mainboard.

The clock_in_type record contains all the DCM output signals, shown below:

```
type clock_in_type is record
   clock       : std_logic;
   clock90     : std_logic;
   clock180    : std_logic;
   clock270    : std_logic;
   clock2x     : std_logic;
   clock2x180  : std_logic;
   clockdv     : std_logic;
   locked      : std_logic;
   pad         : std_logic;
end record;
```

This record sets give you an interface to most of the DCM outputs for p_clock.

The user_out port, of type clock_interface_out_type, provides you with an interface to any of the clock outputs. The record definitions below show the available signals:

```
type clock_interface_out_type is record
    p_clock     : clock_out_type;
end record;
```

The "reset" signal in clock_out_type gives you the ability to reset the p_clock DCM. This is also available to the host.

```
type clock_out_type is record
    reset       : std_logic;
end record;
```

Details of all the port signals of the clock_interface are shown in Table 7-13.

Table 7-13: Clock Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads	NA	Bi-dir	Clock pad signals
Reserved	NA	Bi-dir	Reserved signals
User_In.p_clock	1	Output	Processor clock DCM input record.
User_In.ref48_clock	1	Output	48 MHz reference clock. (Not connected to a DCM)
User_In.ONEPPS_clock	1	Output	One Pulse Per Second Clock. (Not connected to a DCM
User_Out.p_clock	1	Input	Processor Clock DCM output record.

The clock interface generics are described below. For most applications they can be left out, since they are all given appropriate default values:

- **P_CLK_DCM_FREQUENCY_MODE:** This generic should be set to LOW_FREQ (default) for nearly all applications. The Xilinx® documentation describes this DCM attribute in more detail.

7.4.2 LAD Bus Interface (Lad_Interface)

The LAD Bus is the primary means of communicating with the host system. Using the LAD bus, the PE can send and receive programmed I/O data from the host and DMA data to and from the host.

The component declarations for the lad_interface and port record types are shown below:

```
component lad_interface is
port (
    pads                : inout lad_pads_type;
    reset               : in std_logic;
    reserved            : inout reserved_type;
    user_in             : out lad_interface_in_type;
    user_out            : in lad_interface_out_type
);
end component;
```

```
type lad_interface_in_type is record
    addr            : std_logic_vector (18 downto 0);
    data_in         : std_logic_vector (31 downto 0);
    reg_strobe      : std_logic;
    dma_strobe      : std_logic;
    dma_in_progress : std_logic;
    write           : std_logic;
    pci_rdy         : std_logic;
    bus_gnt         : std_logic;
    rxclk_in        : clock_in_type;
end record;
```

```
type lad_interface_out_type is record
    data_out    : std_logic_vector (31 downto 0);
    strobe_out  : std_logic;
    dma_init    : std_logic;
    pe_rdy      : std_logic;
    int_req     : std_logic;
    bus_req     : std_logic;
end record;
```

The “user_in” and “user_out” port signals of the lad_interface component should be used by the design to receive data from and send data to the LAD bus of the WILDSTAR™-II PRO device, respectively. The “pads” signals of the

lad_interface component are only used to connect the lad_interface component to the pads, and should always be assigned to pads.LAD_Bus.

A definition of each of the port signals is given in Table 7-14 below.

Table 7-14: LAD Bus Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
pads	NA	Bi-dir	LAD Bus Pad signals
reset	1	Input	Global reset (or set) signal; connected to the GSR input of a STARTUP_Virtex component
Reserved	NA	Bi-dir	Reserved signals
user_in.addr	19	Output	User interface to the LAD bus address; note that this is a DWORD address; also note that this address is automatically incremented each clock cycle during burst LAD bus cycles
user_in.data_in	32	Output	User interface to the input data from the LAD bus
user_in.reg_strobe	1	Output	Register space access strobe signal; indicates a valid register space cycle when '1'
user_in.dma_strobe	1	Output	DMA access strobe signal; indicates a valid DMA-To-PE cycle when '1'
user_in.dma_in_progress	1	Output	This signal goes low when all DMA-From-PE data has been transmitted to the host.
user_in.write	1	Output	Write select interface signal; indicates a write cycle when '1' or a read cycle when '0' (when strobe is '1')
user_in.pci_rdy	1	Output	Set to '1' when the PCI controller can accept DMA data
user_in.bus_gnt	1	Output	Set to '1' when the PE can master the LAD bus
user_in.reset	1	Output	PE Reset Signal. Can be toggled by a host API call
user_in.rxclk_in	1	Output	PCI to PE transmit clock DLL input signals.
user_out.data_out	32	Input	User interface to the output data to the LAD bus
user_out.strobe_out	1	Input	The PE drives this signal to '1' during DMA Init, DMA-From-PE and register read LAD bus cycles to indicate that the requested data has been driven on the LAD bus.
user_out.dma_init	1	Input	The PE drives this signal during a DMA initialization cycle.
user_out.pe_rdy	1	Input	'1' indicates that the PE can accept DMA data '0' indicates that the PE cannot accept DMA data
user_out.int_req	1	Input	User interface to the interrupt request signal; Driving this line high for >90ns will create an interrupt to the host.
user_out.bus_req	1	Input	The PE drives this signal to '1' to request LAD bus mastering. The PCI controller will respond by setting User_In.BusGnt to '1' when mastering is possible.

7.4.2.1 LAD Bus Transactions

There are six types of LAD Bus transaction cycles: two register space cycles, two DMA init cycles and two DMA data transfer cycles. Register space read cycles and register space write cycles, both of which are initiated by the PCI controller, are generated when the host calls a register read or write API call. The two DMA init cycles are generated by the PE to initialize the PCI controller for DMA transactions.

The DMA-to-PE transaction, a DMA data transfer cycle, is generated by the PCI controller to transfer DMA data to the PE. The DMA-From-PE data transfer

cycle is generated by the PE to send DMA data to the PCI controller, and ultimately to the host memory.

7.4.2.2 Register Space Transactions

A typical LAD bus register write access is illustrated in Figure 7-8. The PE receives an Address, Data, and Reg_Strobe for every piece of data transferred to the PE. User_In.Write is high for register writes.

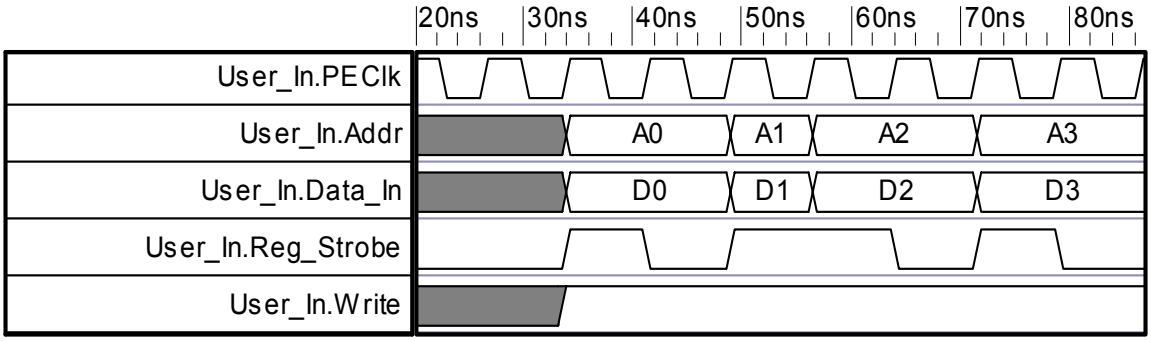


Figure 7-8: Typical LAD Bus Write Cycle from Register Space

Unlike register writes, LAD bus register reads are not bursts. Only a single data word is expected for every address and strobe on the LAD bus. Within ~64 clock cycles of receiving a register read LAD bus transfer, the PE must simultaneously drive lad_bus_out.data_out with the requested data and user_out.strobe_out to '1'.

INFORMATION NOTE

For WILDSTAR™-II PRO boards, 1 us (about 120) cycles are allowed. However, long delays will halt all PCI bus traffic and degrade system performance. Delays should be kept as short as possible.

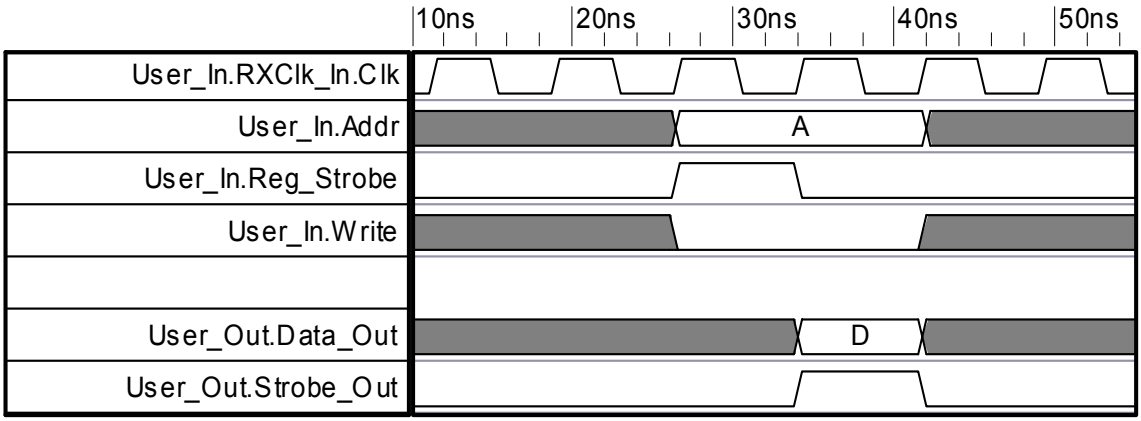


Figure 7-9: Shortest Possible Register Space LAD Bus Read Cycle

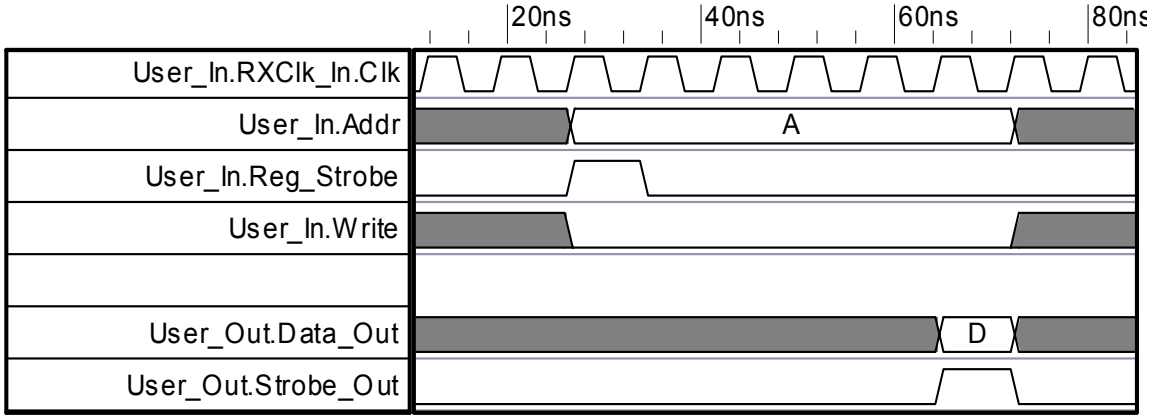


Figure 7-10: Longer Register Space LAD Bus Read Cycle

7.4.2.3 DMA Initialization Cycles

There are two types of DMA initialization LAD bus cycles, known as DMA-To-PE and DMA-From-PE cycles. DMA-To-PE cycles are used to set up a DMA transaction from host memory to the PE. The PE generates both of these cycles on the LAD bus to set up a DMA transaction to host memory.

After sending the DMA-To-PE initialization cycle, the PE starts receiving the requested DMA data. DMA-From-PE cycles, conversely, are used to set up a DMA transaction from the PE to the host memory. After a DMA-From-PE initialization cycle, the PCI controller sets User_In.PCIRdy high signifying that the PE can start sending DMA data to the PCI via LAD bus DMA-From-PE data cycles.

A DMA-To-PE initialization cycle has the form shown in the figure below.

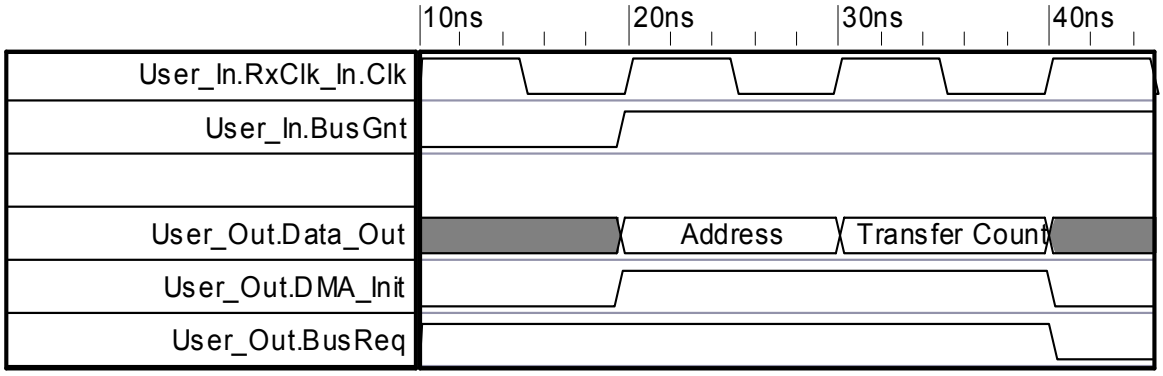


Figure 7-11: DMA-To-PE Initialization Cycle

Before the PE can generate cycles on the LAD bus, it must be granted the bus. To request the bus grant to the PE, the User_Out.BusReq line must be asserted. Some number of clock cycles later, when User_In.BusGnt goes high, the PE can start generating the DMA-To-PE initialization cycle.

The DMA-To-PE initialization cycle consists of first sending a DMA hardware address and then a transfer count to the PCI controller. These two must be sent in consecutive clock cycles.

The transfer count is the number of DWORDS (32-bit values) to transfer from host memory to the PE. The host must have allocated and bound a DMA buffer of at least this many DWORDS before the PE executes the DMA-To-PE initialization cycle. The address is a slightly modified version of the hardware address returned by the WSII_DmaBind API call. The PE must set the bottom two bits of the HardwareAddress to "00", which tells the PCI controller that this cycle initializes a DMA-To-PE transaction.

Upon completing the initialization cycle, the PE should deassert User_Out.BusReq. The PCI controller then begins fetching the requested data from memory and sending it to the PE via the DMA-To-PE data cycle described below.

A DMA-From-PE initialization Cycle has the form shown below.

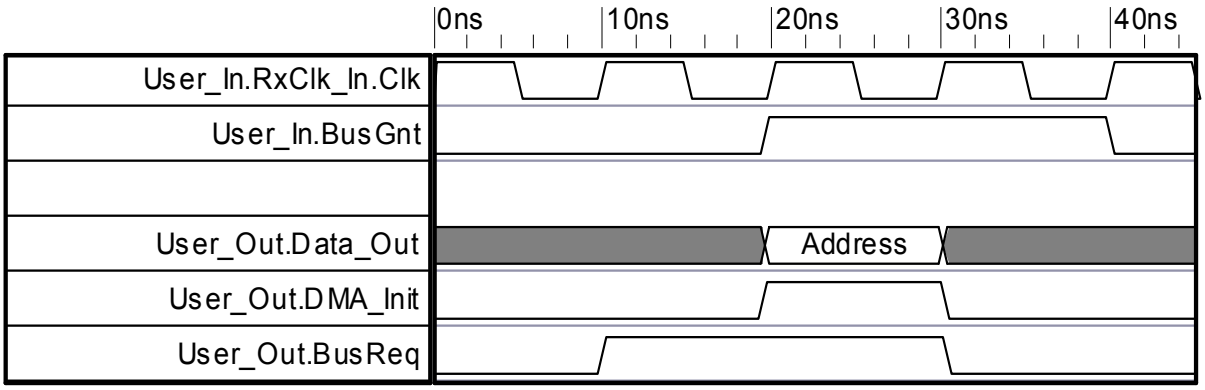


Figure 7-12: DMA-From-PE Initialization Cycle

As with the DMA-To-PE initialization cycles, before the PE may generate cycles on the LAD bus it must be granted the bus.

The DMA-From-PE initialization cycle consists only of sending a DMA hardware address. The number of DWORDs transferred is controlled by the PE. The PCI controller accepts as many pieces of data from the PE as the host has allocated with the WSII_DmaBind API call.

The address is a slightly modified version of the hardware address returned by the WSII_DmaBind API call. The PE must set the bottom two bits of the HardwareAddress to "01". This tells the PCI controller that this cycle initializes a DMA-From-PE transaction.

Upon completion of the initialization cycle the PE should deassert User_Out.BusReq. If the hardware address was valid, the PCI controller sets User_In.PCIRdy high and the PE can start sending data to the PCI controller via the DMA-From-PE data cycles described below.

7.4.2.4 DMA Data Cycles

While the DMA initialization cycles set up the DMA transfers, the DMA data cycles are used to actually transfer the data. There are two types of DMA data cycles—the DMA-To-PE and DMA-From-PE data cycles. The DMA-To-PE data cycles are driven by the PCI controller, while the PE generates the DMA-From-PE cycles.

Figure 7-13 shows a typical DMA-To-PE data cycle.

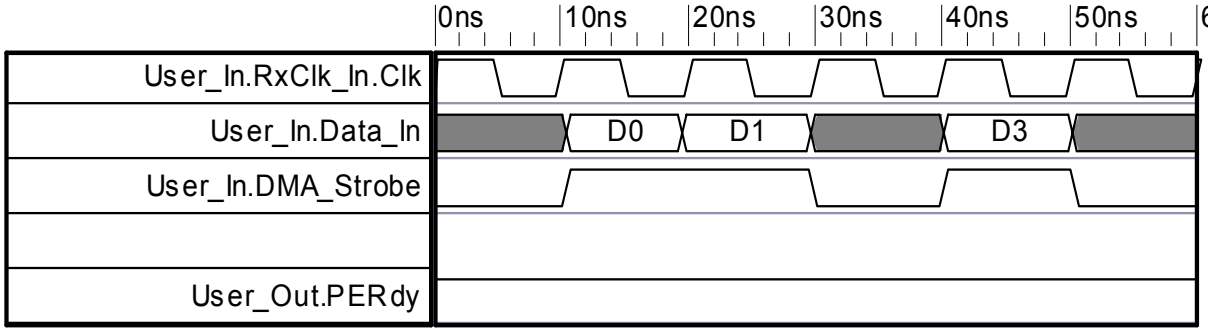


Figure 7-13: DMA-To-PE Data Cycle

For every clock cycle in which user_in.dma_strobe is high, the PE is given a new piece of DMA data. The data does not come with a corresponding address; however, the PCI controller will always send the data in order, starting with the address requested in the DMA-To-PE initialization cycle.

The user_out.pe_rdy line is used by the PE to tell the PCI controller when it can accept data. The PCI controller stops sending data to the PE if user_out.pe_rdy is set low. The PE, however, must be able to accept at least seven more pieces of data following the deassertion of user_out.pe_rdy.

Figure 8-10 shows a typical DMA-From-PE data cycle.

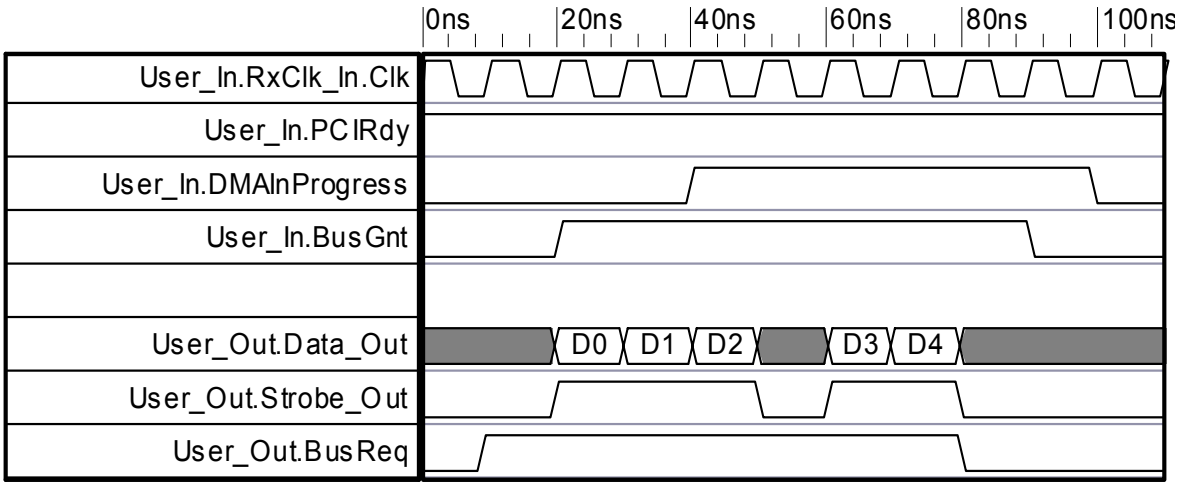


Figure 7-14: DMA-From-PE Data Cycle

Before the PE can start a DMA-From-PE data cycle, it must request the LAD bus by asserting user_out.bus_req and wait for user_in.bus_gnt to go high. Once the bus is granted, the PE sends DMA data to the PCI controller by asserting the strobe_out signal.

For every clock that the PCI controller samples user_out.strobe_out high, user_out.data_out is sent by DMA back to the host. This DMA transaction does not have an address. The PCI controller assumes the data comes in order, starting with the data intended for the starting address given in the DMA-From-PE initialization cycle.

The PCI controller signals its ability to accept DMA data through the user_in.pci_rdy signal. The PE should not request the bus to transmit DMA data if user_in.pci_rdy is not high. Additionally, if user_in.pci_rdy goes low during a DMA-From-PE transaction, the PE must stop sending data within 200 clock cycles.

Once the PE has completed the DMA-From-PE transaction, it should deassert User_Out.BusReq. The user_in.dma_in_progress signal stays active until the PCI controller has transferred all the data to the host memory. This signal may stay high for a number of clock cycles after the PE has finished the DMA-From-PE transaction.

7.4.3 SRAM DDRII Interface (sram_ddrii_port*x*_interface)

The SRAM interface currently supports half or full population of the following parts:

- Samsung® 18Mb (512Kx36-bit) DDRII CIO b2 (K7I323682B)
- Samsung® 36Mb (1Mx36-bit) DDRII CIO b2 (K7I163682B)

The WSII_GetInformation API call is used to determine memory type, burst length and population.

The sram_ddrii_port*x*_interface component provides a simple user interface to the DDR SRAM ports connected to the WILDSTAR™-II PRO PE device. The '*x*' in 'sram_ddrii_port*x*_interface' corresponds to the port in which the user wishes to interface.

For additional information for interfacing the DDRII SRAM, see the datasheets at www.samsung.com.
The sram_ddrii_port*x*_interface is defined as:

```
component sram_ddrii_port0_interface is
generic (
    SRAM_SIZE   : natural       := 18;  -- In Mb
    SRAM_POP    : POPULATION    := HALF
);
port (
    pads        : inout sram_pads_type;
    reset       : in std_logic;
    reserved    : in reserved_type;
    p_clock     : in std_logic;
    user_in     : out sram_interface_in_type;
    user_out    : in sram_interface_out_type
);
end component;
```

All signals into and out of this interface are synchronous to the p_clock.

The sram_interface_in_type and sram_interface_out_type are defined as:

```
type sram_interface_in_type is record
    data_in     : std_logic_vector (71 downto 0);
    data_valid  : std_logic;
end record;
```

```
type sram_interface_out_type is record
    data_out    : std_logic_vector (71 downto 0);
    addr        : std_logic_vector (19 downto 0);
    bwe         : std_logic_vector (3 downto 0);
    write       : std_logic;
    strobe      : std_logic;
end record;
```

The port signals of the SRAM interface are summarized below.

Table 7-15: SRAM Interface Signal Mapping

Signal Name	Width	Dir	Description
user_out.addr	20	Input	Address Bus
user_out.data_out	32	Input	Data Bus Output
user_out.bwe	4	Input	Byte Write Enable
user_out.strobe	2	Input	Strobe signal to initiate a SRAM access (all other User_Out signals must be valid when high)
user_out.write	1	Input	Select a write to SRAM when high
user_in.data_in	32	Output	Read Data from SRAM
user_in.data_valid	1	Output	Read Data Valid from SRAM

To perform an SRAM access, provide valid signals for user_out (addr, data_out if write, bwe, write) and set user_out.strobe high. Read data will come back on the user_in port and is quantified by the user_in.data_valid signal.

Some rules must be followed when using the SRAMs and doing a read followed by a write:

- For all memories, a NOOP must be inserted between a write and read request.
- For the Samsung® memories, a second NOOP must be added above 150 MHz.

The sram_port*x*_interface component generics are described below:

- **SRAM_SIZE:** Defaults to 18 (as in Mb). Other options are 36 or 72. This refers to the total size of the SRAM parts.
- **SRAM_POP:** Defaults to HALF (as in populated). Other option is FULL.

7.4.4 SRAM QDRII Interface (sram_qdrii_port*x*_interface)

The SRAM interface currently supports half or full population of the following parts:

- Samsung® 18Mb QDRII b2 (K7R161882M)
- Samsung® 36Mb QDRII b2 (K7R321882M)
- GSI® 18Mb QDRII (GS8182Q18D-167)
- GSI® 36Mb QDRII (GS8182Q36D-167)

The WSII_GetInformation API call is used to determine memory type, burst length and population.

The sram_qdrii_port*x*_interface component provides a simple user interface to the DDR SRAM ports connected to the WILDSTAR™-II PRO PE device. The '*x*' in 'sram_qdrii_port*x*_interface' corresponds to the port in which the user wishes to interface.

For additional information for interfacing the DDRII SRAM, see the datasheets at www.samsung.com.
The sram_qdrii_port*x*_interface is defined as:

```
component sram_qdrii_port0_interface is
generic (
    SRAM_SIZE   : natural       := 18;  -- In Mb
    SRAM_POP    : POPULATION    := HALF
);
port (
    pads        : inout sram_pads_type;
    reset       : in std_logic;
    reserved    : in reserved_type;
    p_clock     : in std_logic;
    user_in     : out sram_qdrii_interface_in_type;
    user_out    : in sram_qdrii_interface_out_type
);
end component;
```

All signals into and out of this interface are synchronous to the p_clock.

The sram_interface_in_type and sram_interface_out_type are defined as:

```
type sram_qdrii_interface_in_type is record
    read_data       : std_logic_vector(35 downto 0);
    read_data_valid : std_logic;
end record;
```

```
type sram_qdrii_interface_out_type is record
    write_data      : std_logic_vector(35 downto 0);
    write_addr      : std_logic_vector(20 downto 0);
    read_addr       : std_logic_vector(20 downto 0);
    write           : std_logic;
    read            : std_logic;
end record;
```

The port signals of the SRAM interface are summarized below.

Table 7-16: SRAM QDRII Interface Signal Mapping

Signal Name	Width	Dir	Description
user_out.write_addr	21	Input	Write Address Bus
user_out.write_data	36	Input	Write Data Bus Output
user_out.read_addr	21	Input	Read Address Bus
user_out.write	1	Input	Write enable
user_out.read	1	Input	Read enable
user_in.read_data	36	Output	Read Data from SRAM
user_in.read_data_valid	1	Output	Read Data Valid from SRAM

The unique feature of the QDRII SRAMs is that you can write and read at the same time, both from the part itself and also through our interface. To write data, simply put the address on write_addr, the data on write_data, and set write to '1'. To read data, put the address on read_addr, and set read to '1'. When the data is available, read_data_valid will go high and the data will appear on read_data.

The sram_qdrii_port*x*_interface component generics are described below:

- **SRAM_SIZE:** Defaults to 18 (as in Mb). Other options are 36 or 72. This refers to the total size of the SRAM parts.
- **SRAM_POP:** Defaults to HALF (as in populated). Other option is FULL.

7.4.4.1 WILDSTAR™-II PRO Reserved Registers

Four permanent registers built into the WILDSTAR™-II PRO VHDL model are located at the beginning of the address map for each PE.

INFORMATION NOTE

An entire block of address space per PE is blocked off for internal use. This space goes from 0x00 to 0x3C on each PE.
Registers should not be created in these locations.

Below are descriptions of each register:

Version Register (0x0): This register contains the VHDL version. Bits 23 to 16 contain the major release number. Bits 15 to 8 contain the minor release number. Bits 7 to 0 contain the second minor release number.

I/O Control Register (0x01): This register contains a code describing the type of I/O interface currently implemented in the PE and should only be written by the API. In conjunction with the API, it is used to prevent any IO card mismatches.

DCM Control Register (0x02): This register controls most of the PEs clock DCMs. The included example source (C and VHDL) show how to use this register. Table 7-17 describes its contents.

Signature Register (0x3C): This register is reserved by the software as a test register only.

Table 7-17: Description of DCM Control Register

Bit	R/W	Default	Description
31-24	R/W	0000000	SRAM Clock Phase Shift Value
23	R	0	SRAM Clock Phase Shift Done
23-21	R	0	Reserved
20	R/W	0	SRAM Chip Onboard DLL Disable
19	R	0	SRAM Clock DCM Locked
18-17	R	0	Reserved
16	R/W	0	SRAM Clock DCM Reset
15	R	0	Rocket IO Port1 Clock DCM Locked
14	R/W	0	Rocket IO Port1 Clock DCM Reset
13	R	0	Rocked IO Port0 Clock DCM Locked
12	R/W	0	Rocked IO Port0 Clock DCM Reset
11	R	0	P2 Connector Clock DCM Locked
10	R/W	0	P2 Connector Clock DCM Reset
9	R	0	IO Connector Clock DCM Locked
8	R/W	0	IO Connector Clock DCM Reset
7	R	0	PE to PE Connector Port1 Clock DCM Locked
6	R/W	0	PE to PE Connector Port1 Clock DCM Reset
5	R	0	PE to PE Connector Port0 Clock DCM Locked
4	R/W	0	PE to PE Connector Port0 Clock DCM Reset
3	R	0	P Clock DCM Locked
2	R	0	Reserved
1	R/W	0	P Clock DCM Clock Output Enable
0	R/W	0	P Clock DCM Reset

The Samsung memories used on WILDSTAR™-II PRO contain onboard DLLs. These DLLs should be turned on when P Clock is 124 MHz or higher. Below 124 MHz, this DLL should be turned off. This option is controlled via the DCM Control Register in the WILDSTAR™-II PRO Reserved Register space. It is called the SRAM Chip Onboard DLL Disable bit.

7.4.5 DRAM Interface (dram_interface)

The dram_interface provides the user interface for DRAM.

The DRAM interface is defined as:

```
component dram_interface is
generic (
    DRAM_SIZE       : natural := 256
);
port (
    pads            : inout dram_pads_type;
    reset           : in std_logic;
    reserved        : in reserved_type;
    p_clock         : in clock_in_type;
    ref48_clock     : in std_logic;
    user_in         : out dram_interface_in_type;
    user_out        : in dram_interface_out_type
);
end component;
```

The dram_interface_in_type is defined as:

```
type dram_interface_in_type is record
    data_in             : std_logic_vector(63 downto
0);
    data_in_available   : std_logic;
    write_rdy           : std_logic;
    read_rdy            : std_logic;
end record;
```

The dram_interface_out_type is defined as:

```
type dram_interface_out_type is record
    addr            : std_logic_vector(24 downto 0);
    data_out        : std_logic_vector(63 downto 0);
    data_in_ce      : std_logic;
    write           : std_logic;
    read            : std_logic;
end record;
```

The port signals of the DRAM interface are shown in the following table.

Table 7-18: DRAM Interface Component Port Signals

Signal Name	Width	Dir	Description
user_in.data_in	64	Input	Read data in from the interface
User_in.data_in_available	1	Input	Read data in available signal from interface
User_in.write_rdy_	1	Input	Signal from interface stating whether user can write
User_in.read_rdy_	1	Input	Signal from interface stating whether user can read
user_out.addr	25	Output	Address lines to interface
user_out.data_out	64	Output	Write data out from user to interface
User_out.data_in_ce	1	Output	Signal to interface telling it to send the next set of data
user_out.write	1	Output	signal to interface for writing data
User_out.read	1	Output	signal to interface for reading data
user_in.write_rdy	64	Output	User Write Ready signal
user_out.read_rdy	1	Output	User Read Ready signal

To write data to the SDRAM, first check that the ‘write_rdy’ signal is high. This step ensures that the write FIFOs are not full. If ‘write_rdy’ is high, the interface

is ready to accept write data. Next, put the address on the 'addr' lines and the 64-bit data on the 'data_out' lines and then pulse 'write'. Data can be sent in continuously as long as 'write_rdy' is high.

To read data back, first check that 'read_rdy' is high to make sure that the read address FIFOs are not full. Then, put the address on the 'addr' lines and pulse 'read'. Again, the address of data to be read can be continuously written into the interface as long as 'read_rdy' is high. Soon afterwards, the 'data_in_available' signal will go high. To get the read data, the data_in_ce line must be pulsed (or held high to retrieve data continuously). One clock later, data will be available on the 'data in' lines .

7.4.6 LED Interface (led_interface)

The LED interface provides the user access to the three light emitting diodes (LEDs) of the WILDSTAR™-II PRO PCI and VME PE devices. On the WILDSTAR™-II PRO ACE for VME, there are six LEDs. The LEDs can be used to indicate status or processing activity, and can even be used to help debug a PE design.

The led_interface is defined as:

```
component led_interface is
port (
    pads            : inout led_pads_type;
    reset           : in std_logic;
    clock           : in std_logic;
    user_out        : in led_interface_out_type
);
end component;
```

The led_interface_out_type for WILDSTAR™-II PRO PCI and VME is defined as:

```
type led_interface_out_type is record
    red             : std_logic;
    yellow          : std_logic;
    green           : std_logic;
end record;
```

The led_interface_out_type for WILDSTAR™-II PRO ACE for VME is defined as:

```
type led_interface_out_type is record
    red0            : std_logic;
    yellow0         : std_logic;
    green0          : std_logic;
    red1            : std_logic;
    yellow1         : std_logic;
    green1          : std_logic;
    red2            : std_logic;
    yellow2         : std_logic;
    green2          : std_logic;
end record;
```

The port signals of the LED standard interface are shown below. The user_out signals should be used to drive the LEDs of the PE device.

Table 7-19: LED Interface Component Port Signals for non-ACE

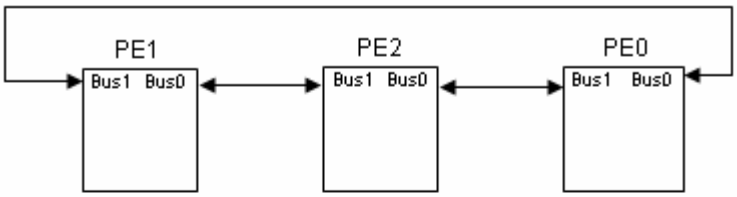
Signal Name	Width	Dir	Description
pads	3	Output	LED pad signals
user_out.red	1	Input	User interface to the red LED, '1' = ON
user_out.green	1	Input	User interface to the green LED, '1' = ON
user_out.yellow	1	Input	User interface to the Yellow LED, '1' = ON

Table 7-20: LED Interface Component Port Signals for ACE

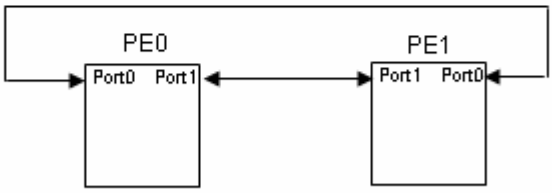
Signal Name	Width	Dir	Description
Pads	3	Output	LED pad signals
user_out.red0	1	Input	User interface to the red 0 LED, '1' = ON
user_out.green0	1	Input	User interface to the green 0 LED, '1' = ON
user_out.yellow0	1	Input	User interface to the Yellow 0 LED, '1' = ON
user_out.red1	1	Input	User interface to the red 1 LED, '1' = ON
user_out.green1	1	Input	User interface to the green 1 LED, '1' = ON
user_out.yellow1	1	Input	User interface to the Yellow 1 LED, '1' = ON
user_out.red2	1	Input	User interface to the red 2 LED, '1' = ON
user_out.green2	1	Input	User interface to the green 2 LED, '1' = ON
user_out.yellow2	1	Input	User interface to the Yellow 2 LED, '1' = ON

7.4.7 PE to PE Bus Interface (peconn_port*x*_interface)

Each PE has two 50-bit differential buses connecting it with the adjacent PE. Below are diagrams showing PE bus connections on the WILDSTAR™-II PRO boards (not available on ACE boards).



WILDSTAR™-II PRO VME PE Bus Connections



WILDSTAR™-II PRO PCI PE Bus Connections

The component declaration for the peconn_interface is:

```
component peconn_port0_interface is
generic (
    -- '0' = input / '1' = output
    DATA_DIR    : std_logic_vector(49 downto 0) := "00" & x"000000000000");
port (
    pads        : inout peconn_pads_type;
    reset       : in std_logic;
    reserved    : inout reserved_type;
    p_clock     : in std_logic;
    user_in     : out peconn_interface_in_type;
    user_out    : in peconn_interface_out_type
);
end component;
```

The peconn_interface_in_type is as follows:

```
type peconn_interface_in_type is record
    data_in         : std_logic_vector (49 downto 0);
    rx_clock        : std_logic;
end record;
```

The peconn_interface_out_type is as follows:

```
type peconn_interface_out_type is record
    data_out        : std_logic_vector (49 downto 0);
end record;
```

The 'x' in peconn_portx_interface refers to the port number. Each PE has two 50-bit ports, called port0 and port1. PE0's peconn_port0_interface hooks up to PE1's peconn_port0_interface. PE0's peconn_port1_interface connects to PE1's peconn_port1_interface. Therefore, if you set all 50 bits of PE0's peconn_port0_interface to outputs, you need to set all 50 bits of PE1's peconn_port0_interface to inputs.

Data going into the interface on the user_out.data_out lines should be based off of p_clock. Data coming in from the interface on the data_in.data_in lines is clocked on the user_in.rx_clock clock.

The generic for the peconn_portx_interface is as follows:

- **DATA_DIR:** Since this generic sets the direction of each individual bit on the port, it is 50 bits long. When a bit is set to '0', the bit is an input. When a bit is set to '1', the bit is an output.

The port signals of the pe_conn_port*x*_interface are shown below. The "user_out" and "user_in" signals should be used to read and write to the PE buses.

Table 7-21: PE to PE Interface Port Signals

Signal Name	Width	Dir	Description
Pads	NA	Bi-Dir	PE bus pad signals
reset	1	Input	Global reset signal
Reserved	NA	Bi-Dir	Reserved signals
p_clock	1	Input	Input Clock.
user_out.data_out	50	Input	Data to be sent out of the PE
user_in.data_in	50	Output	Data coming into the PE
user_in.rx_clock	1	Output	Clock that the input data comes in on

7.4.8 Rocket IO Interface (rio_port*x*_interface) (not available for VME Systems)

Each PE has two Rocket IO interfaces included in it, and each interface has 10 2-byte ports associated with it. When combined, the interfaces provide access to a 320-bit wide bi-directional bus which can currently run between 60 MHz and 100 MHz (not available on ACE boards).

The component declaration for the rio_port*x*_interface is:

```
component rio_port0_interface is
generic (
    PE              : natural   := 0;
    NO_OF_BYTES     : natural   := 2
);
port (
    pads            : inout rio_pads_type;
    reset           : in std_logic;
    p_clock         : in std_logic;
    reserved        : inout reserved_type;
    user_in         : out rio_interface_in_type;
    user_out        : in rio_interface_out_type
);
end component;
```

The peconn_interface_in_type is as follows:

```
type rio_interface_in_type is record
    rxdata_port9    : std_logic_vector(15 downto 0);
    rxdata_port8    : std_logic_vector(15 downto 0);
    rxdata_port7    : std_logic_vector(15 downto 0);
    rxdata_port6    : std_logic_vector(15 downto 0);
    rxdata_port5    : std_logic_vector(15 downto 0);
    rxdata_port4    : std_logic_vector(15 downto 0);
    rxdata_port3    : std_logic_vector(15 downto 0);
    rxdata_port2    : std_logic_vector(15 downto 0);
    rxdata_port1    : std_logic_vector(15 downto 0);
    rxdata_port0    : std_logic_vector(15 downto 0);
    txready         : std_logic;
    rxready         : std_logic;
end record;
```

The peconn_interface_out_type is as follows:

```
type rio_interface_out_type is record
    txdata_port9    : std_logic_vector(15 downto 0);
    txdata_port8    : std_logic_vector(15 downto 0);
    txdata_port7    : std_logic_vector(15 downto 0);
    txdata_port6    : std_logic_vector(15 downto 0);
    txdata_port5    : std_logic_vector(15 downto 0);
    txdata_port4    : std_logic_vector(15 downto 0);
    txdata_port3    : std_logic_vector(15 downto 0);
    txdata_port2    : std_logic_vector(15 downto 0);
    txdata_port1    : std_logic_vector(15 downto 0);
    txdata_port0    : std_logic_vector(15 downto 0);
end record;
```

The ‘x’ in rio_portx_interface refers to the main port number. PE0’s rio_port0_interface hooks up to PE1’s rio_port0_interface, and PE0’s rio_port1_interface connects to PE1’s rio_port1_interface. As noted earlier, each main port has 10 data ports in it, with each of the data ports being two bytes wide.

Note that the Rocket IO interface includes no method for determining whether the data is valid or not. The example included with the VHDL uses one of the 16-bits as a data valid signal.

To write on any data port, simply put data on the rio_port*x*_interface’s user_out.txdata_port*y* lines. This data shows up in the other PE’s rio_port*x*_interface’s user_in.rxdata_port*y* after a couple hundred clocks.

The data going into and out of these interfaces should be based on p_clock.

The generics for the rio_port*x*_interface are as follows:

- **PE:** Defaults to ‘0’. This is simply the PE number. It can either be ‘0’ for PE0 or ‘1’ for PE1.

- **NO_OF_BYTES:** Defaults to ‘2’. This is the number of bytes per data port. At this time, this is the only amount supported. Do not change this number.

The port signals of the rio_port*x*_interface are shown below. The “user_out” and “user_in” signals should be used to read and write to the Rocket IO buses.

Table 7-22: Rocket IO Interface Port Signals

Signal Name	Width	Dir	Description
pads	NA	Bi-Dir	PE bus pad signals
reset	1	Input	Global reset signal
reserved	NA	Bi-Dir	Reserved signals
p_clock	1	Input	Input Clock.
user_out.txdata_port9	16	Input	Data to be sent out of the PE
user_out.txdata_port8	16	Input	Data to be sent out of the PE
user_out.txdata_port7	16	Input	Data to be sent out of the PE
user_out.txdata_port6	16	Input	Data to be sent out of the PE
user_out.txdata_port5	16	Input	Data to be sent out of the PE
user_out.txdata_port4	16	Input	Data to be sent out of the PE
user_out.txdata_port3	16	Input	Data to be sent out of the PE
user_out.txdata_port2	16	Input	Data to be sent out of the PE
user_out.txdata_port1	16	Input	Data to be sent out of the PE
user_out.txdata_port0	16	Input	Data to be sent out of the PE
user_in.rxdata_port9	16	Output	Data coming into the PE
user_in.rxdata_port8	16	Output	Data coming into the PE
user_in.rxdata_port7	16	Output	Data coming into the PE
user_in.rxdata_port6	16	Output	Data coming into the PE
user_in.rxdata_port5	16	Output	Data coming into the PE
user_in.rxdata_port4	16	Output	Data coming into the PE
user_in.rxdata_port3	16	Output	Data coming into the PE
user_in.rxdata_port2	16	Output	Data coming into the PE
user_in.rxdata_port1	16	Output	Data coming into the PE
user_in.rxdata_port0	16	Output	Data coming into the PE
user_in.txready	1	Output	Transmit is ready signal
user_in.rxready	1	Input	Receive is ready signal

7.5 WILDSTAR™-II VHDL Host Model

The simulated VHDL host model is used to access the PE resources of the WILDSTAR™-II and WILDSTAR™-II PRO boards via the host (LAD cycles). The model is similar to the actual host system in that you can write a "program" consisting of standard API functions that interact with the board. In fact, the simulated VHDL host API procedures have nearly the same structure and usage as the actual software API functions.

7.5.1 Simulated Host API Procedures

The following simulated VHDL host API procedures are available:

WSII_SetClockFrequency	Sets various board clock frequencies
WSII_PeReset	Pulses, Asserts or Deasserts the reset line for a set of PEs.
WSII_InterruptWait	Waits for an interrupt from a set of PEs.
WSII_InterruptQueryStatus	Gets the interrupt status information from a particular board
WSII_InterruptReset	Resets the interrupt
WSII_ReadRegs_32	Reads from register locations in the PE Register space
WSII_WriteRegs_32	Writes to register locations in the PE Register space
WSII_DmaMemAlloc	Allocates a DMA buffer and DMA handle.
WSII_DmaMemFree	Frees a DMA buffer and DMA handle.
WSII_DmaBind	Binds a DMA buffer.
WSII_DmaUnBind	Unbinds a DMA buffer.

Syntax:

```
procedure WSII_SetClockFrequency
  (
    signal Cmd_Req      :   out boolean;
    signal Cmd_Ack      : in    boolean;
           Board        :       WSII_BOARD;
           dClock       :       DWORD;
           fClkFreqMhz  :       float
  );
```

Description:

Sets various PE clock frequencies.

Arguments:

Cmd_Req	Host model command request signal
Cmd_Ack	Host model command acknowledge signal
board	WILDSTAR™ board identifier (zero-based) variable
dClock	WILDSTAR™ board clock to set. This parameter can be WSII_CLK_PE0_A, WSII_CLK_PE0_B, WSII_CLK_PE0_C, WSII_CLK_PE1_A, WSII_CLK_PE1_B, WSII_CLK_PE1_C, WSII_CLK_PE2_A, WSII_CLK_PE2_B, WSII_CLK_PE2_C, WSII_CLK_GLOBAL_P, or WSII_CLK_GLOBAL_M. ***Note:*** *The parameters for the WILDSTAR™-II PRO board are WSII_CLK_GLOBAL_P and WSII_CLK_GLOBAL_REF only.*
dFreq	Specifies the clock frequency in MHz

Comments:

This function causes the PCI Controller to set the specified clock to the frequency specified by *dFreq*.

Syntax:

```
procedure WSII_PeReset
  (
    signal Cmd_Req        :   out boolean;
    signal Cmd_Ack        : in    boolean;
           board          : in    WSII_BOARD;
           dPeMask        : in    DWORD;
           ResetOperation : in    DWORD
  );
```

Description:

This function controls the global reset of a set of PEs. The reset line can be asserted, deasserted or pulsed for one I Clock cycle.

Arguments:

Cmd_Req	Host model command request signal
Cmd_Ack	Host model command acknowledge signal
board	WILDSTAR™ board identifier (zero-based) variable
dPEMask	Determines which set of WILDSTAR™ boards to reset. This parameter can be WSII_MASK_PE0, WSII_MASK_PE1, WSII_MASK_EX0, or any combination of these values or-ed together.
ResetOperation	Determines the operation to perform on the reset line. This parameter can be WSII_RESET_ASSERT, WSII_RESET_DEASSERT,WSII_RESET_PULSE to pulse the reset line for one I Clock cycle.

Syntax:

```
procedure WSII_InterruptQueryStatus
  (
    signal Cmd_Req        :   out boolean;
    signal Cmd_Ack        : in    boolean;
           board          : in    WSII_BOARD;
           pIntMask       :   out DWORD
  );
```

Description:

Returns the status of interrupts for a particular board. The return value format is:

Bit 0 : 1 if PE0 has generated an interrupt that has not been cleared
Bit 1 : 1 if PE1 has generated an interrupt that has not been cleared
Bit 3: 1 if external I/O card 0 has generated an interrupt that has not been cleared

Arguments:

Cmd_Req	Host model command request signal
Cmd_Ack	Host model command acknowledge signal
board	WILDSTAR™ board identifier (zero-based) variable
pIntMask	Returns the status of the interrupts using the format in the description above.

Syntax:

```
procedure WSII_InterruptQueryStatus
(
    signal Cmd_Req :   out boolean;
    signal Cmd_Ack : in    boolean;
    board          : in    WSII_BOARD;
    ResetMask      : in    DWORD
);
```

Description:

If a bit is set, that interrupt source will be cleared; otherwise, it will be left alone.
Bit 0 : 0 clears PE0's interrupt
Bit 1 : 0 Clears PE1's interrupt
Bit 3 : 0 Clars I/O card 0 interrupt

Arguments:

Cmd_Req	Host model command request signal
Cmd_Ack	Host model command acknowledge signal
board	WILDSTAR™-II board identifier (zero-based) variable
ResetMask	Mask of bits to clear

Syntax:

```
procedure WSII_InterruptWait
  (
    signal Cmd_Req        :   out boolean;
    signal Cmd_Ack        : in    boolean;
           board          : in    WSII_BOARD;
           WaitMask       : in    DWORD;
           pIntMask       :   out DWORD;
           dTimeoutMs     : in    DWORD
  ); WSII_SetClockFrequency
```

Description:

Waits for an interrupt from a set of PEs.

Arguments:

Cmd_Req	Host model command request signal
Cmd_Ack	Host model command acknowledge signal
board	WILDSTAR™ board identifier (zero-based) variable
WaitMask	Determines the set of PE interrupts to wait for. This parameter can be WSII_MASK_PE0, WSII_MASK_PE1, WSII_MASK_EX0, or any combination of these values "OR-ed" together.
pIntMask	Set to the current interrupt status value when this procedure returns.
dTimeoutMs	This parameter is not used in simulation.

Syntax:

```
procedure WSII_ReadRegs_32
  (
    signal Cmd_Req      :   out boolean;
    signal Cmd_Ack      : in    boolean;
           board        : in    WSII_BOARD;
           dPeRegion    : in    DWORD;
           dOffset      : in    DWORD;
           dNumDwords   : in    DWORD;
           pData        :   out DWORD_array
  );
```

Description:

Reads from the register space of the PE on the specified board.

Arguments:

Cmd_Req	Host model command request signal
Cmd_Ack	Host model command acknowledge signal
board	WILDSTAR™ board identifier (zero-based) variable
dPeRegion	Specifies which PE to access. This parameter can be WSII_PE0, WSII_PE1, WSII_PE2, WSII_EXTIO_PE0, or WSII_EXTIO_PE1.
dOffset	Specifies the offset of the first DWORD in the PE register space to be accessed. Zero (0) is the lowest DWORD offset, 0x7FFFF is the highest.
dNumDwords	Number of DWORDs to read from PE register space, starting at *dOffset*
pData	DWORD array into which the read data is stored (must have at least *dNumDwords* elements in the array)

Comments:

This function causes the PCI Controller to initiate one or more LAD bus read transactions to the PE device.

See Also:

WSII_WriteRegs_32

Syntax:

```
procedure WSII_WriteRegs_32
  (
    signal Cmd_Req      :   out boolean;
    signal Cmd_Ack      : in    boolean;
           board        : in    WSII_BOARD;
           dPeRegion    : in    DWORD;
           dOffset      : in    DWORD;
           dNumDwords   : in    DWORD;
           pData        : in    DWORD_array
  );
```

Description:

Writes to the register space of the PE on the specified board.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
board	WILDSTAR™ board identifier (zero-based) variable
dPeRegion	Specifies which PE to access
dOffset	Specifies the offset of the first DWORD in the PE register space to be accessed. Zero (0) is the lowest DWORD offset, 0x7FFFF is the highest.
dNumDwords	Number of DWORDs to write to PE register space, starting at *dOffset*
pRegData	DWORD array from which the write data is retrieved (must have at least *dNumDwords* elements in the array)

Comments:

This function causes the PCI Controller to initiate one or more LAD bus read transactions to the PE device.

See Also:

WSII_ReadRegs_32

Syntax:

```
procedure WSII_DmaMemAlloc

    (
      signal   Cmd_Req      :   out boolean;
      signal   Cmd_Ack      : in    boolean;
               Board        :       WSII_BOARD;
      variable DmaHandle    :   out WSII_DMA_HANDLE;
               dNumDwords   : in    DWORD;
      variable pBuffer      :   out DWORD_Pointer
    );
```

Description:

Allocates a DMA handle and buffer usable for DMA transactions.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
board	WILDSTAR™ board identifier (zero-based) variable
DMAhandle	The returned DMA handle. This handle is used in WSII_DmaBind to bind the allocated memory buffer.
dNumDwords	Number of DWORDs allocate.
pBuffer	Return a new DWORD_Pointer a memory buffer of size dNumDwords. (DWORD_Pointer is an access type to an array of DWORDS)

Comments:

This is the first step to any DMA transaction. The DMA handle returned is used in all subsequent DMA API calls.

See Also:

WSII_DmaMemFree, WSII_DmaBind, WSII_DmaUnbind

Syntax:

```
procedure WSII_DmaMemFree

    (
      signal   Cmd_Req      :   out boolean;
      signal   Cmd_Ack      : in    boolean;
               Board        :       WSII_BOARD;
      variable DmaHandle    : inout WSII_DMA_HANDLE
    );
```

Description:

This procedure frees a previously allocated DMA buffer and DMA handle.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
board	WILDSTAR™ board identifier (zero-based) variable
DMAhandle	DMA handle to free. This procedure also frees the DMA memory buffer associated with this DMA handle.

Comments:

If the buffer to be freed is also bound, it must be unbound (using WSII_DmaUnbind) before calling WSII_DmaMemFree.

See Also:

WSII_DmaMemAlloc, WSII_DmaBind, WSII_DmaUnbind

Syntax:

```
procedure WSII_DmaBind

    (
      signal   Cmd_Req                :   out boolean;
      signal   Cmd_Ack                : in    boolean;
               Board                  :       WSII_BOARD;
      variable DmaHandle              : inout WSII_DMA_HANDLE;
      variable pBuffer                : in    DWORD_Pointer;
               dRequestedLengthDwords : in    DWORD;
      variable dGrantedLengthDwords   :   out DWORD;
      variable HardwareAddress        :   out DWORD
    );
```

Description:

This procedure binds an allocated buffer for use with DMA. All DMA buffers must be bound before they can be used. In rare occasions dGrantedLengthDwords may be less than the dRequestedLengthDwords due to finite resources. The DMA transfer may be a maximum of dGrantedLengthDwords DWORDS in length.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
board	WILDSTAR™ board identifier (zero-based) variable
DMAhandle	DMA handle for the buffer that is to be bound.
pBuffer	DMA buffer to be bound.
dRequestedLengthDwords	Number of DWORDS of this DMA buffer to bind.
dGrantedLengthDwords	Number of DWORDS of this DMA buffer that were bound.
HardwareAddress	Hardware address that the PE uses in the DMA initialization LAD bus cycles to DMA to this host DMA buffer.

See Also:

WSII_DmaMemFree, WSII_DmaBind, WSII_DmaUnbind

Syntax:

```
procedure WSII_DmaUnbind

    (
      signal   Cmd_Req               :   out boolean;
      signal   Cmd_Ack               : in    boolean;
               Board                 :       WSII_BOARD;
      variable DmaHandle             : inout WSII_DMA_HANDLE
    );
```

Description:

This procedure unbinds a previously bound DMA buffer.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
board	WILDSTAR™ board identifier (zero-based) variable
DMAhandle	The handle to the DMA buffer to be unbound.

See Also:

WSII_DmaMemFree, WSII_DmaBind, WSII_DmaUnbind

7.5.2 VHDL Host Model Simulation Environment

The WILDSTAR™-II system has been modeled in behavioral VHDL to give you a board simulation environment in which to develop applications. This enables you to validate the PE designs before the synthesis, placement, and routing steps.



INFORMATION NOTE

VHDL Models are compiled and therefore require the ModelSim® simulator to simulate.

The WILDSTAR™-II simulation model is hierarchical in design. It starts with a model of the system that contains a host and one or more WILDSTAR™-II boards. The WILDSTAR™-II board contains the PE, memories, host interface, I/O card, and clock generation circuitry.

7.5.3 Using Templates

Template files are used throughout the WILDSTAR™-II VHDL design cycle as a means to accelerate the design cycle and promote the understanding of the model components. As new components are added to the VHDL model, the template files will reflect how these new components should be used in a design. It is always a good idea to start with a simple design and work toward the final solution in stages.

7.5.4 Using VHDL Record Types

All of the user interfaces to the logical I/O ports of the PEs are described using VHDL record types. Records are used to collect all of the signals of an interface into a single data structure in order to provide an easy way to identify all of the signals associated with that interface. Interface signals that are record types are also much easier to declare and connect to their respective interface components.

7.5.5 PCI Controller Model

The PCI Controller is used to handle all LAD bus transactions, configure the clocks, and handle PE interrupts. These PCI Controller functions can be accessed from the simulated host API function interfaces described in the previous section.

7.5.6 On-Board Memory Model

The 36-bit on-board synchronous SRAM and 64 bit synchronous DRAM memory models are currently based on a static memory model in order to facilitate the inspection of the memory contents from the VHDL simulator tool. However, if no memory is needed by the application, the on-board memory model can be configured to an “Empty” architecture in order to speed up the simulation time.

7.5.7 I/O Card Model

By default, the "Empty" I/O card model is used. The empty I/O card tri-states all signals to the WILDSTAR™-II board. When additional I/O cards are installed, VHDL templates and configurations are installed as well.

7.5.8 VHDL Model Configuration

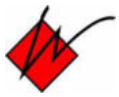
A WILDSTAR™-II can be made up of one or more WILDSTAR™-II boards (each with two possible I/O cards) and a host processor. However, each WILDSTAR™-II board in the system can be configured differently (e.g., different PE architectures, memory types and I/O cards). The WILDSTAR™-II system simulation model can be customized to a particular application by modifying the VHDL configuration.

7.5.9 WILDSTAR™-II System Configuration (System_Config)

The parts of the WILDSTAR™-II and VHDL system configurations that can be modified by the user are as follows:

For WILDSTAR™-II /VME and /PCI:

- **Host**: The host model architecture can be changed to the name used by a particular application.
- **PE0**: The PE0 architecture can be changed to the name used by a particular PE in the application.
- **PE1**: The PE1 architecture can be changed to the name used by a particular PE in the application.
- **PE2**: The PE2 architecture can be changed to the name used by a particular PE in the application.
- **I/O card**: The default VHDL configuration is "Empty_IO_Card_Config." When the VHDL supplied with the WILDSTAR™-II I/O card product is installed, the proper I/O card configuration will also be installed. However, users may also supply their own VHDL configuration for any custom-designed I/O cards.
- **WSII_PCI_Config:** Several generics can be changed to configure the types of memories used for each of the PEs.



8 WILDSTAR™ VHDL DESIGN CYCLE

The WILDSTAR™-II VHDL design cycle consists of several simple steps that help you create, simulate, synthesize, and analyze a WILDSTAR™-II system design. This chapter describes how to create a design using the WILDSTAR™-II and system VHDL models.

8.1 WILDSTAR™-II /VME and /PCI VHDL Design

8.1.1 Using Template VHDL Design Files

Template files should be used when creating a WILDSTAR™-II design since the template files have been pre-designed to contain most of the boilerplate interface logic that is needed to by every design. Starting from scratch would not only take much longer, but would also increase the risk of implementing interface logic that is incorrect or incomplete.

The following VHDL template files are all found in the sub-directory called "$ANNAPOLISII_BASE/<Board>/template/sim" (where $ANNAPOLISII_BASE is the installation path to the WILDSTAR™-II /VME or WILDSTAR™-II /PCI VHDL model, and <Board> is either "wsii_pci" or "wsii_vme"), and should be copied to a design project directory and used as a starting point for the design:

project_vcom.do	Modeltech® macro script file
project_vsim.do	Modeltech® macro script file
host_template_arch.vhd	Host design template architecture
pe_template_arch.vhd	PE0/PE1 design template architecture
pe_revb_template_arch.vhd	PE0/PE1 design template architecture (Rev B boards only)
pe2_template_arch.vhd	PE2 design template architecture.
system_template_arch.vhd	System template architecture.
system_cfg.vhd	System design configuration

8.1.2 Creating a PE Design

The PE template file should be modified by removing the comments from the standard interfaces which are needed by the design. The user-defined portion of the application should be made to reside in a separate component and file. Hierarchical design should be used in order to keep the complexity of any single component to a minimum.

8.1.3 Creating a Host Design

Commands such as WSII_WriteRegs_32 or WSII_ReadRegs_32 can be issued to the ModelSim® simulator by using VHDL to create a host model.

8.1.3.1 Using the VHDL Host Template

The host template file should be modified by removing the comments from the API procedures and modifying the program flow to suit the design. The host program will usually reside in a single sequential main process, but it is possible to mimic a "multi-threaded" program by having more than one process. Currently, however, a lock or a mutual exclusion mechanism must be built for issuing commands to the PCI Controller.

8.1.4 Creating a System Architecture

The template system architecture file, system_template_arch.vhd, creates a system with only one board, labeled "board zero". To add more boards to the system, uncomment the desired number of board architectures and their corresponding command processing lines in the P_CMD process in the system_template_arch.vhd file.

8.1.5 Modifying Configuration Files

The system configuration (system_cfg.vhd) file must also be modified to meet the needs of the design. Model details such as memory bank type, PE architecture, and host architecture can be set inside this configuration file. Simply follow the instructions inside the configuration file on how to change the user-modifiable portions of the WILDSTAR™-II VHDL model.

8.1.6 Design Simulation

This section discusses how to simulate a WILDSTAR™-II VHDL design using the Model Technology, Inc., ModelSim® VHDL simulator.

8.1.6.1 Compiling the UNISIM library

The UNISIM library provides a simulation model for the various Xilinx® Virtex-II primitives used in the VHDL model. This library must be compiled once before your first simulation.

Xilinx provides the executable "compxlib" to compile the UNISIM library for various platforms. To compile the UNISIM library for the WILDSTAR-II, use the following parameters:

- ***-w*** : To overwrite any existing compiled files
- ***-f virtex2:u*** : To select the Virtex-II UNISIM library
- ***-l vhdl*** : To select VHDL as the target language
- ***-o <Annapolis base>\<board>\vhdl***: To set the target output directory. Set <Annapolis Base> to you root Annapolis directory (typically c:\annapolis), and <board> to wsii_vme or wsii_pci for WILDSTAR™-II/VME or WILDSTAR-II/PCI respectively.
- ***-s <mti_se/mti_pe>*** : To select the simulator. Use mti_se for ModelSim SE versions and mti_pe for ModelSim PE.

- ***-p <path>*** : To set the path to the ModelSim program. <Path> is the directory where the ModelSim executable is located.

For example, to compile the UNISIM library for a WILDSTAR™-II/VME, the following command might be used:

```
compxlib -w -f virtex2:u -l vhdl -s mti_se -p
c:\ModelTech5.7\win32 -o c:\annapolis\wsii_vme\vhdl
```

8.1.6.2 Using Template ModelSim® Macro Files



INFORMATION NOTE

To ensure that all ModelSim® templates are current, the *refresh.do* script should be run (located in the annapolis\wsii_pci\template\sim directory). This script updates all the precompiled libraries shipped with the WILDSTAR™-II VHDL model. If you have installed VHDL in a directory other than <\annapolis>, you will need to change the ANNAPOLIS_BASE variable in the *refresh.do* script to point to the proper directory.

The following ModelSim® template files are all found in the sub-directory called "$ANNAPOLISII_BASE/<Board>/template/sim" (where $ANNAPOLISII_BASE is the installation path to the WILDSTAR™-II /VME or WILDSTAR™-II /PCI VHDL model and <Board> is either "wsii_pci" or "wsii_vme"), and should be copied to a design project directory and used as a starting point for the design:

project_vcom.do	Project VHDL compilation macro for ModelSim™
project_vsim.do	Project VHDL simulation macro for ModelSim®

Once the macro files have been copied, they should be modified to meet the needs of the design project. Simply follow the step-by-step instructions located inside each of the macro files to customize them for the current design project.

8.1.6.3 Compiling a VHDL Design

Once the macro files have been configured, the design can be compiled using the ModelSim® tool. Follow these simple steps to compile the design:

1. Start the ModelSim® tool
2. In the **Tools** menu (versions 5.6 and newer), select the **Execute Macro…** option.

3. Using the file browser, locate the design project directory.
4. Select the **project_vcom.do** compilation macro file.
5. Click on the **Open** button.

At this point, the compilation messages will begin to scroll in the **ModelSim** window (or **Transcript** window in ModelSim v4.7 or older). If any errors occur during compilation, the error message(s) will also appear in the **ModelSim**/**Transcript** window. Once the errors are corrected, repeat steps 1 through 5 above until the design has been completely compiled.

8.1.6.4 Simulating a VHDL Design

Once the VHDL design has been compiled, the design can be simulated using the ModelSim® tool. Follow these simple steps to simulate the design:

1. Start the ModelSim® tool
2. In the **Tools** menu (versions 5.6 and newer), select the **Execute Macro…** option
3. Using the file browser, locate the design project directory
4. Select the **project_vsim.do** compilation macro file
5. Click on the **Open** button

At this point, the loading messages will begin to scroll in the **ModelSim** window (or **Transcript** window in ModelSim® Version 4.7 or older). If any errors occur during the loading of the design, the error message(s) will also appear in the **ModelSim**/**Transcript** window.

Once the errors are corrected, repeat steps 1 through 5 above until the design has been completely loaded and is ready for simulation. Refer to the ModelSim® manual for details on how to run the simulation.

8.1.7 Synthesizing a Design

This section discusses how to synthesize a design using the Synplicity Synplify® VHDL synthesis tool.

8.1.7.1 Using Template Synplify® Project Files

The following Synplify® project files are found in the sub-directory called "$ANNAPOLISII_BASE/<Board>/template/syn" (where $ANNAPOLISII_BASE is the installation path to the WILDSTAR™-II /VME or WILDSTAR™-II /PCI VHDL model and <Board> is either "wsii_pci" or "wsii_vme"). The file should be copied to a design project directory and used as a starting point for the design:

pe\pe.prj	PE0/PE1 project VHDL synthesis project file for Synplify®
pe2\pe2.prj	PE2 project VHDL synthesis project file for

Once the PE project file has been copied, it should be modified to meet the needs of the design project. Simply follow the step-by-step instructions located inside the project file to customize it for the current design project.

8.1.7.2 Setting up Synthesis Constraints

Certain design constraints can be configured using the Synplify® synthesis tool. Some of these constraints are located at the end of the Synplify® project file. Other design constraints may be added directly to the VHDL code as VHDL attributes, while others can be added to the Synplify® design constraints file (*.sdc). Refer to the Synplify® manual for more information regarding synthesis design constraints.

8.1.7.3 Synthesize the Design

Once the Synplify® project file and design constraints have been configured, the PE design can be synthesized. Follow these steps to synthesize the PE design:

1. Start the Synplify® tool
2. Close any project window that might be open
3. In the **File** menu, select the **Open Project…** option
4. Using the file browser, locate the design project directory
5. For Select the **pe.prj** or **pe2.prj** Synplify® project file
6. Click on the **Open** button
7. Click on the **RUN** button

At this point, compiling and mapping messages will begin to appear in the project window. If any warnings or errors occur during the loading of the design, the error indicator will also appear in the project window. Click on the View Log button to review the warning and/or error messages.

Due to the comprehensive nature of the WILDSTAR™-II PE model, you will undoubtedly encounter warnings during the synthesis process. These warnings are usually related to PE I/O signals that are simply unused by the current design project. As a rule, you can ignore any warnings that do not appear in your own design files.

Once the errors (and user warnings) are corrected, repeat steps 1 through 7 above until the design has been completely loaded and is ready for place-and-route. Refer to the Synplify manual for more details on how to use the Synplify® tool.

8.1.8 Placing and Routing a Design

Once the design has been synthesized and an EDIF design file has been produced, the Xilinx® Alliance Series tools can be used to place-and-route the design. The Xilinx® Foundation Series tools include all of the necessary Alliance Series tools.

8.1.8.1 Setting Environment Variables

There are two environment variables which should be added or modified to the user profile. The first is only necessary under Windows NT® systems. Click Start -> Settings -> Control Panel -> system. Click on the environment tab. Add a new variable called MAKE_MODE and set the value to UNIX. The second variable, which should already exist, is the path variable. Add to the path variable "$ANNAPOLISII_BASE/shared_ii/bin" (where ANNAPOLISII_BASE is the installation path to the WILDSTAR™- II VHDL model). These changes/modifications are necessary for proper makefile execution.

8.1.8.2 Using Template Makefiles

The following Xilinx® Alliance Series makefile files are found in the sub-directory called "$ANNAPOLISII_BASE/<Board>/template/syn" (where $ANNAPOLISII_BASE is the installation path to the WILDSTAR™-II /VME or WILDSTAR™-II /PCI VHDL model and <Board> is either "wsii_pci" or "wsii_vme") and should be copied to a design project directory and used as a starting point for the design:

pe\makefile	PE0/PE1 project makefile for Xilinx Alliance tools
pe2\makefile	PE2 project makefile for Xilinx Alliance tools

Once the makefile has been copied, it should be edited so that the ANNAPOLISII_BASE macro properly reflects the WILDSTAR™-II installation directories. Be sure to use forward slashes ('/') in the path name, even if using an operating system that usually requires backslashes in a path name.

There are several other variables which must be modified in order for the makefile to produce the correct binary file for WILDSTAR™-II.

1. **DEF_PART**: must be set to match the PE part type.
2. **DEF_SPEED**: should be set to match the speed grade of the PE. *If the speed grade does not match, erroneous timing reports will be generated.*

8.1.8.3 Setting Up Timing Constraints

In addition to setting up synthesis constraints, you can also set up other timing constraints by using what is called a user constraints file (UCF). The UCF file for the design can be created by copying the **<design>.ncf** file that was created by the Synplify® tool to a UCF file called **<design>.ucf**. The UCF file can then be edited and changed to meet the timing requirements of the current design project.

The UCF file will automatically be incorporated into the place-and-route process as long as the base part of the UCF filename matches the base part of the EDIF filename (e.g., pe0_design.ucf matches pe0_design.edf).

8.1.8.4 Running the Xilinx® Place-and-Route Makefile

Once the Xilinx® makefile has been edited and the timing constraints are completed, the design is ready to be placed and routed. Follow these simple steps to place-and-route a WILDSTAR™-II PE design:

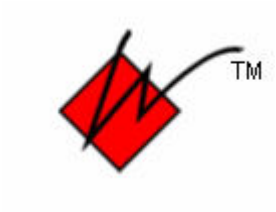
1. If using an operating system other than Microsoft Windows®, open a DOS command prompt.
2. Open a UNIX-like shell from which the **make** command will operate.
3. Change directories to the design project directory
4. Type "**make** <pe design name>.x86"

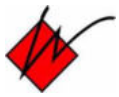
8.1.9 Analyzing Design Performance

Once the design has been placed and routed, the timing and area performance can be analyzed by referring to the following files.

<design>.par	Area and timing results generated by the **par** tool
<design>.twr	Timing report generated by the **trce** tool

If the desired timing was not met, you can choose to modify the timing constraints or to modify the Xilinx® Alliance Series makefile options.


TM



INDEX

A

API, 1-4
Application Builder Online Help, 1
Application Programming Interface, 1-4

B

Block RAM, 1-4

C

CLB, 1-4
Clocks, 6-11
Configurable Logic Block, 1-4
Conventions, 1-3
CoreFire, i, 1-2, 1-4, 2-1, 2-2, 2-3, 2-5, 2-8, 2-10, 2-12, 2-15, 2-18, 1, 2

D

Definitions, 1-3
DLL, 1-5
DMA, 7-8, 7-9, 7-10, 7-12, 7-13, 7-14, 7-15, 7-40, 7-41, 7-43, 7-44, 7-45, 7-46, 7-60, 7-68, 7-69, 7-70, 7-71
DRAM, 1-4, 2-7, 2-9, 2-12, 2-14, 2-18
Driver, 1-5

E

Euro, 2-6, 2-10, 2-14
External I/O, 1-5

F

FPGA, 1-5, 1

G

General Specifications, 6-1

H

Host software, 2-7, 2-10, 2-12, 2-14, 2-18

I

ICLK, 1-5, 6-11, 6-16, 6-17
Icons, 1-3
Introduction, 1

K

KCLK, 1-5
Key Words, 1-3

L

LED, 1-1, 3-2, 3-5, 3-6, 3-7, 3-12, 3-15, 3-20, 3-21, 3-23, 3-24, 3-25, 7-21, 7-22, 7-24, 7-25, 7-52, 7-54
LED Definition, 3-7, 3-12, 3-15, 3-21, 3-25
Lower-Level Cores, 1

M

MCLK, 1-6
ModelSim, 8-1, 8-2, 8-3, 8-4
ModelTech, 1-6, 2-1

P

PCI, 1-6
PCLK, 1-6
PE. See Processing Element
PE0, 1-6
PE1, 1-6
PE2, 1-6
PMC, 8-6
Power Requirements, 6-7

Q

QDR, 1-7

R

RACEway, 1-7
Register Space Transactions, 7-10, 7-42
Rocket I/O, 1-7, 2-3

S

Slice, 1-7
Switches, 4-2
Synplify, 8-4, 8-5, 8-6
System Support Features, 2-7, 2-9, 2-11, 2-14, 2-17

T

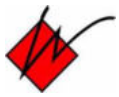
Thermal Management, 2-7, 2-10, 2-12, 2-15, 2-18, 6-1
Tools, 1
trace points, 1

V

VIRTEX™ FPGA, 1-8
Virtex™-II PRO, 1-7, 1-8, 2-1, 2-3
VME, 1-8
VME64X, 2-7, 4-1

W

WILDSTAR™ II, i, iii, 1-1, 1-2, 1-5, 1-6, 2-1, 2-5, 2-6, 2-7, 2-8, 2-9, 2-10, 2-11, 2-12, 2-13, 2-14, 2-15, 2-16, 2-17, 2-18, 3-1, 3-2, 3-3, 3-4, 3-5, 3-6, 3-7, 3-9, 3-11, 3-12, 3-13, 3-14, 3-15, 3-17, 3-18, 3-19, 3-20, 3-21, 3-22, 3-23, 3-24, 3-25, 4-1, 4-2, 4-4, 4-5, 4-8, 4-9, 4-10, 4-11, 4-12, 4-13, 4-14, 4-15, 4-16, 4-17, 5-1, 6-1, 6-7, 6-9, 6-11, 6-12, 6-13, 6-14, 6-15, 6-16, 6-17, 6-18, 7-1, 7-9, 7-25, 7-27, 7-36, 7-40, 7-60, 7-72, 7-73, 8-1, 8-2, 8-3, 8-4, 8-5, 8-6, 8-7
WILDSTAR™ II /PCI, 4-15, 8-1, 8-3, 8-4, 8-6
WILDSTAR™ II /VME, 2-6, 2-10
WILDSTAR™/VME, 4-3, 4-8, 4-11
WILDSTAR™-II PRO, i, 2-1, 2-3, 2-16, 2-17, 3-22, 6-1
WILDSTAR™-II PRO/PCI, 2-16
WSDP™, i



APPENDIX A: WILDSTAR™-II BACKPLANE STANDARD INTERFACE PIN MAPPING

Table A-1: P0 Backplane Standard Interface Pin Mapping in Differential Mode

Interface Pin Number	Positive Differential Pin Location	Negative Differential Pin Location
0	A01	B01
1	A02	B02
2	A03	B03
3	A04	B04
4	A05	B05
5	A06	B06
6	A07	B07
7	A08	B08
8	A09	B09
TxClk	A10	B10
9	A11	B11
10	A12	B12
11	A13	B13
12	A14	B14
13	A15	B15
14	A16	B16
15	A17	B17
16	A18	B18
17	A19	B19
18	D01	E01
19	D02	E02
20	D03	E03
21	D04	E04
22	D05	E05
23	D06	E06
24	D07	E07
25	D08	E08
26	D09	E09
RxClk	D10	E10
27	D11	E11
28	D12	E12
29	D13	E13
30	D14	E14
31	D15	E15
32	D16	E16
33	D17	E17
34	D18	E18
35	D19	E19

Table A-2: WILDSTAR™-II P0 Backplane Standard Interface Pin Mapping in Single-Ended Mode

Interface Pin Number	Pin Location
0	A01
1	A02
2	A03
3	A04
4	A05
5	A06
6	A07
7	A08
8	A09
TxClk	A10
9	A11
10	A12
11	A13
12	A14
13	A15
14	A16
15	A17
16	A18
17	A19
18	B01
19	B02
20	B03
21	B04
22	B05
23	B06
24	B07
25	B08
26	B09
27	B10
28	B11
29	B12
30	B13
31	B14
32	B15
33	B16
34	B17
35	B18
36	B19
37	D01
38	D02
39	D03
40	D04
41	D05

Interface Pin Number	Pin Location
42	D06
43	D07
44	D08
45	D09
RxClk	D10
46	D11
47	D12
48	D13
49	D14
50	D15
51	D16
52	D17
53	D18
54	D19
55	E01
56	E02
57	E03
58	E04
59	E05
60	E06
61	E07
62	E08
63	E09
64	E10
65	E11
66	E12
67	E13
68	E14
69	E15
70	E16
71	E17
72	E18
73	E19
74	C07
75	C08
76	C09
77	C10
78	C11
79	C12
80	C13
81	C14
82	C15
83	C16
84	C17
85	C18
86	C19

Table A-3: WILDSTAR™-II P2 Backplane Standard Interface Pin Mapping in Differential Mode

Interface Pin Number	Positive Differential Pin Location	Negative Differential Pin Location
0	A02	Z01
1	A03	A04
2	A06	Z07
3	A08	A09
4	A11	A12
5	A14	Z13
6	A15	Z15
7	A17	Z17
8	A18	A19
9	A20	Z19
10	A21	Z21
11	A22	A23
12	A24	Z23
13	A25	Z25
14	A26	A27
15	A29	Z29
16	A30	A31
17	A32	Z31
18	C01	D01
19	C02	D03
20	C04	D05
21	C05	D06
22	C07	D07
23	C09	D08
24	C10	D10
25	C12	D11
26	C13	D13
27	C15	D16
28	C18	D19
29	C21	D22
30	C23	D23
31	C24	D24
32	C25	C26
33	C27	D26
34	C28	C29
35	C30	C31
36	C32*	D25*
37	D03	D04
38	D14	D15
39	D17	D18
40	D20	D21

Interface Pin Number	Positive Differential Pin Location	Negative Differential Pin Location
41	D27	D28
42	D29	D30
43	Z03	Z05
44	Z09	Z11
45	Z28	Z27

*Bit 36 is NOT CONNECTED on Rev. A boards

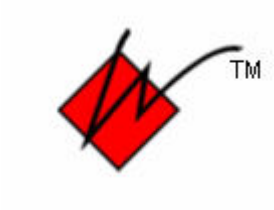
Table A-4: WILDSTAR™-II P2 Backplane Standard Interface Pin Mapping in Single Ended Mode

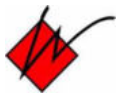
Interface Pin Number	Pin Location
0	A02
1	A03
2	A04
3	A06
4	A08
5	A09
6	A11
7	A12
8	A14
9	A15
10	A17
11	A18
12	A19
13	A20
14	A21
15	A22
16	A23
17	A24
18	A25
19	A26
20	A27
21	A28
22	A29
23	A30
24	A31
25	A32
26	C01
27	C02
28	C03
29	C04
30	C05
31	C07
32	C09
33	C10

Interface Pin Number	Pin Location
34	C12
35	C13
36	C15
37	C16
38	C18
39	C19
40	C21
41	C22
42	C23
43	C24
44	C25
45	C26
46	C27
47	C28
48	C29
49	C30
50	C31
51	C32
52	D01
53	D03
54	D04
55	D05
56	D06
57	D07
58	D08
59	D10
60	D11
61	D13
62	D14
63	D15
64	D17
65	D18
66	D20
67	D21
68	D23
69	D24
70	D26
71	D27
72	D28
73	D29
74	D30
75	D25*
76	Z01
77	Z03
78	Z05
79	Z07
80	Z09

Interface Pin Number	Pin Location
81	Z11
82	Z13
83	Z15
84	Z17
85	Z19
86	Z21
87	Z23
88	Z25
89	Z27
90	Z29
91	Z31

*Bit 75 is NOT CONNECTED on Rev. A boards


TM



APPENDIX B: WILDSTAR™-II PRO P2 BACKPLANE PIN MAPPING FOR STANDARD OPTIONS

Table B-1: WILDSTAR™-II PRO P2 Backplane Pin Mapping with Dual RACEway

P2 Pin	Raceway Signal Name	PRO ACE PIN	PRO VME PE2 Pin
A1	XCLKI	AP22	AT21
A2	GND	AF36	AU21
A3	XB1_IO09	AH31	AU12
A4	XB1_IO08	AH32	AT12
A5	GND	AH37	AV12
A6	XB1_IO06	AH38	AW12
A7	GND	AH33	AL13
A8	XB1_IO10	AH34	AM13
A9	XB1_IO04	AJ31	AN13
A10	GND	AJ32	AP13
A11	XB1_IO05	AJ41	AR13
A12	XB1_IO03	AJ42	AT13
A13	GND	AJ37	AU13
A14	XB1_RDCON	AJ38	AV13
A15	XB1_PAR**	AJ33	AY13
A16	GND	AJ34	AW13
A17	XB1_IO00	AJ35	AL14
A18	XB1_IO15	AJ36	AL15
A19	GND	AK41	AU15
A20	XB1_IO24	AK42	AV15
A21	XB1_IO31	AK31	AY15
A22	GND	AK32	AY14
A23	XB1_IO28	AK39	AL16
A24	XB1_IO27	AK40	AM16
A25	GND	AK37	AN16
A26	XB1_IO22	AK38	AP16
A27	XB1_IO20	AK33	AT16
A28	GND	AK34	AR16
A29	XB1_IO18	AK35	AU16
A30	XB1_IO17	AK36	AV16
A31	GND	AL40	AL17
A32	XB1_IO13	AL41	AL18
B1	VME BUS RESERVED		
B2	VME BUS RESERVED		
B3	VME BUS RESERVED		
B4	VME BUS RESERVED		
B5	VME BUS RESERVED		

P2 Pin	Raceway Signal Name	PRO ACE PIN	PRO VME PE2 Pin
B6	VME BUS RESERVED		
B7	VME BUS RESERVED		
B8	VME BUS RESERVED		
B9	VME BUS RESERVED		
B10	VME BUS RESERVED		
B11	VME BUS RESERVED		
B12	VME BUS RESERVED		
B13	VME BUS RESERVED		
B14	VME BUS RESERVED		
B15	VME BUS RESERVED		
B16	VME BUS RESERVED		
B17	VME BUS RESERVED		
B18	VME BUS RESERVED		
B19	VME BUS RESERVED		
B20	VME BUS RESERVED		
B21	VME BUS RESERVED		
B22	VME BUS RESERVED		
B23	VME BUS RESERVED		
B24	VME BUS RESERVED		
B25	VME BUS RESERVED		
B26	VME BUS RESERVED		
B27	VME BUS RESERVED		
B28	VME BUS RESERVED		
B29	VME BUS RESERVED		
B30	VME BUS RESERVED		
B31	VME BUS RESERVED		
B32	VME BUS RESERVED		
C1	XRESETIO*	AL31	AN17
C2	XB2_PAR**	AL32	AM17
C3	XSYNCI*	AL38	AP17
C4	GND	AL39	AR17
C5	XB1_IO07	AL35	AT17
C6	GND	AL36	AU17
C7	XB1_IO11	AL33	AM18
C8	GND	AL34	AN18
C9	XB1_STRBIO	AM41	AR18
C10	XB1_RPLYIO	AM42	AT18
C11	GND	AN40	AV18
C12	XB1_REQI	AM40	AV17
C13	XB1_REQO	AM34	AY17
C14	GND	AM35	AY18
C15	XB1_IO02	AM38	AL19
C16	XB1_IO01	AM39	AM19
C17	GND	AM36	AN19
C18	XB1_IO12	AM37	AP19

P2 Pin	Raceway Signal Name	PRO ACE PIN	PRO VME PE2 Pin
C19	XB1_IO25	AM33	AW17
C20	GND	AN34	AW16
C21	XB1_IO29	AN41	AU19
C22	XB1_IO30	AN42	AV19
C23	GND	AN37	AY19
C24	XB1_IO26	AN38	AW19
C25	XB1_IO23	AN35	AL20
C26	GND	AN36	AL21
C27	XB1_IO19	AP41	AM20
C28	XB1_IO21	AP42	AN20
C29	GND	AP38	AR20
C30	XB1_IO16	AP39	AP20
C31	XB1_IO14	AP35	AU20
C32	GND	AR36	AV20
D1	XB2_IO00	AP36	AW20
D2	GND	AR40	AY23
D3	XB2_IO01	AP37	AY20
D4	XB2_IO02	AR41	AW23
D5	GND	AR37	AV23
D6	XB2_IO03	AT38	AU23
D7	XB2_IO04	AR38	AP23
D8	XB2_IO05	AR39	AR23
D9	GND	AT41	AN23
D10	XB2_IO06	AT42	AM23
D11	XB2_IO07	AT39	AL22
D12	GND	AT40	AL23
D13	XB2_IO08	AU41	AW24
D14	XB2_IO09	AU42	AY24
D15	XB2_IO10	AU39	AV24
D16	GND	AU40	AU24
D17	XB2_IO11	AW40	AT24
D18	XB2_IO12	AV40	AR24
D19	GND	AV41	AP24
D20	XB2_IO13	AV42	AN24
D21	XB2_IO14	AW41	AM24
D22	GND	AW42	AL24
D23	XB2_IO15	AU35	AY25
D24	XB2_IO16	AT35	AY26
D25	GND	AU36	AV26
D26	XB2_IO17	AT37	AV25
D27	XB2_IO18	AY37	AT25
D28	GND	AY38	AR25
D29	XB2_IO19	AW26	AN25
D30	XB2_IO20	AV36	AM25
D31	GND		

P2 Pin	Raceway Signal Name	PRO ACE PIN	PRO VME PE2 Pin
D32	+5V		
Z1	XB2_RPLYIO	AF33	AP10
Z2	GND		
Z3	XB2_REQI	AF34	AR10
Z4	GND		
Z5	XB2_REQO	AG40	AT10
Z6	GND		
Z7	XB2_RDCON	AG41	AU10
Z8	GND		
Z9	XB2_STRBIO	AG32	AW10
Z10	GND		
Z11	XB2_IO21	AG33	AV10
Z12	GND		
Z13	XB2_IO22	AG38	AP11
Z14	GND		
Z15	XB2_IO23	AG39	AR11
Z16	GND		
Z17	XB2_IO24	AG36	AU11
Z18	GND		
Z19	XB2_IO25	AG37	AV11
Z20	GND		
Z21	XB2_IO26	AH35	AY11
Z22	GND		
Z23	XB2_IO27	AG35	AY10
Z24	GND		
Z25	XB2_IO28	AH41	AM12
Z26	GND		
Z27	XB2_IO29	AH42	AN12
Z28	GND		
Z29	XB2_IO30	AJ40	AP12
Z30	GND		
Z31	XB2_IO31	AH40	AR12
Z32	GND		

Table B-2: WILDSTAR™-II PRO P2 Backplane Pin Mapping with P2 Power and 2.5V Differential I/O

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
A1	P2CLKIN+	AF35	AT21
A2	P2CLKIN-	AF36	AU21
A3	P2DATA08+	AH31	AU12
A4	P2DATA08-	AH32	AT12
A5	P2DATA09+	AH37	AV12
A6	P2DATA09-	AH38	AW12

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
A7	P2DATA10+	AH33	AL13
A8	P2DATA10-	AH34	AM13
A9	P2DATA11+	AJ31	AN13
A10	P2DATA11-	AJ32	AP13
A11	P2DATA12+	AJ41	AR13
A12	P2DATA12-	AJ42	AT13
A13	P2DATA13+	AJ37	AU13
A14	P2DATA13-	AJ38	AV13
A15	P2DATA14+	AJ33	AY13
A16	P2DATA14-	AJ34	AW13
A17	P2DATA15+	AJ35	AL14
A18	P2DATA15-	AJ36	AL15
A19	P2DATA16+	AK41	AU15
A20	P2DATA16-	AK42	AV15
A21	P2DATA17+	AK31	AY15
A22	P2DATA17-	AK32	AY14
A23	P2DATA18+	AK39	AL16
A24	P2DATA18-	AK40	AM16
A25	P2DATA19+	AK37	AN16
A26	P2DATA19-	AK38	AP16
A27	P2DATA20+	AK33	AT16
A28	P2DATA20-	AK34	AR16
A29	P2DATA21+	AK35	AU16
A30	P2DATA21-	AK36	AV16
A31	P2DATA22+	AL40	AL17
A32	P2DATA22-	AL41	AL18
B1	VME BUS RESERVED		
B2	VME BUS RESERVED		
B3	VME BUS RESERVED		
B4	VME BUS RESERVED		
B5	VME BUS RESERVED		
B6	VME BUS RESERVED		
B7	VME BUS RESERVED		
B8	VME BUS RESERVED		
B9	VME BUS RESERVED		
B10	VME BUS RESERVED		
B11	VME BUS RESERVED		
B12	VME BUS RESERVED		
B13	VME BUS RESERVED		
B14	VME BUS RESERVED		
B15	VME BUS RESERVED		
B16	VME BUS RESERVED		
B17	VME BUS RESERVED		
B18	VME BUS RESERVED		
B19	VME BUS RESERVED		

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
B20	VME BUS RESERVED		
B21	VME BUS RESERVED		
B22	VME BUS RESERVED		
B23	VME BUS RESERVED		
B24	VME BUS RESERVED		
B25	VME BUS RESERVED		
B26	VME BUS RESERVED		
B27	VME BUS RESERVED		
B28	VME BUS RESERVED		
B29	VME BUS RESERVED		
B30	VME BUS RESERVED		
B31	VME BUS RESERVED		
B32	VME BUS RESERVED		
C1	P2DATA23+	AL31	AN17
C2	P2DATA23-	AL32	AM17
C3	P2DATA24+	AL38	AP17
C4	P2DATA24-	AL39	AR17
C5	P2DATA25+	AL35	AT17
C6	P2DATA25-	AL36	AU17
C7	P2DATA26+	AL33	AM18
C8	P2DATA26-	AL34	AN18
C9	P2DATA27+	AM41	AR18
C10	P2DATA27-	AM42	AT18
C11	P2DATA28+	AN40	AV18
C12	P2DATA28-	AM40	AV17
C13	P2DATA29+	AM34	AY17
C14	P2DATA29-	AM35	AY18
C15	P2DATA30+	AM38	AL19
C16	P2DATA30-	AM39	AM19
C17	P2DATA31+	AM36	AN19
C18	P2DATA31-	AM37	AP19
C19	P2DATA32+(SE ONLY)	AM33	AW17
C20	P2DATA32- (SE ONLY)	AN34	AW16
C21	P2DATA33+	AN41	AU19
C22	P2DATA33-	AN42	AV19
C23	P2DATA34+	AN37	AY19
C24	P2DATA34-	AN38	AW19
C25	P2DATA35+	AN35	AL20
C26	P2DATA35-	AN36	AL21
C27	P2DATA36+	AP41	AM20
C28	P2DATA36-	AP42	AN20
C29	P2DATA37+	AP38	AR20
C30	P2DATA37-	AP39	AP20
C31	P2DATA38+	AP35	AU20
C32	P2DATA38-	AR36	AV20

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
D1	P2DATA39+	AP36	AW20
D2	GND		
D3	P2DATA39-	AP37	AY20
D4	GND		
D5	GND		
D6	5V		
D7	GND		
D8	5V		
D9	GND		
D10	5V		
D11	5V		
D12	GND		
D13	5V		
D14	GND		
D15	5V		
D16	GND		
D17	5V		
D18	5V		
D19	GND		
D20	GND		
D21	5V		
D22	GND		
D23	5V		
D24	5V		
D25	GND		
D26	GND		
D27	5V		
D28	GND		
D29	5V		
D30	5V		
D31	VME BUS RESERVED		
D32	VME BUS RESERVED		
Z1	P2DATA00+		
Z2	GND		
Z3	P2DATA00-		
Z4	GND		
Z5	V33		
Z6	GND		
Z7	V33		
Z8	GND		
Z9	V33		
Z10	GND		
Z11	V33		
Z12	GND		
Z13	V33		

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
Z14	GND		
Z15	V33		
Z16	GND		
Z17	V33		
Z18	GND		
Z19	V33		
Z20	GND		
Z21	V33		
Z22	GND		
Z23	V33		
Z24	GND		
Z25	V33		
Z26	GND		
Z27	V33		
Z28	GND		
Z29	V33		
Z30	GND		
Z31	V33		
Z32	GND		

Table B-3: WILDSTAR™-II PRO P2 Backplane Pin Mapping with P2 Power and 3.3V Single-Ended I/O

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
A1	P2CLKIN0	AF35	AT21
A2	P2CLKIN1	AF36	AU21
A3	P2DATA16	AH31	AU12
A4	P2DATA17	AH32	AT12
A5	P2DATA18	AH37	AV12
A6	P2DATA19	AH38	AW12
A7	P2DATA20	AH33	AL13
A8	P2DATA21	AH34	AM13
A9	P2DATA22	AJ31	AN13
A10	P2DATA23	AJ32	AP13
A11	P2DATA24	AJ41	AR13
A12	P2DATA25	AJ42	AT13
A13	P2DATA26	AJ37	AU13
A14	P2DATA27	AJ38	AV13
A15	P2DATA28	AJ33	AY13
A16	P2DATA29	AJ34	AW13
A17	P2DATA30	AJ35	AL14
A18	P2DATA31	AJ36	AL15
A19	P2DATA32	AK41	AU15
A20	P2DATA33	AK42	AV15
A21	P2DATA34	AK31	AY15

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
A22	P2DATA35	AK32	AY14
A23	P2DATA36	AK39	AL16
A24	P2DATA37	AK40	AM16
A25	P2DATA38	AK37	AN16
A26	P2DATA39	AK38	AP16
A27	P2DATA40	AK33	AT16
A28	P2DATA41	AK34	AR16
A29	P2DATA42	AK35	AU16
A30	P2DATA43	AK36	AV16
A31	P2DATA44	AL40	AL17
A32	P2DATA45	AL41	AL18
B1	VME BUS RESERVED		
B2	VME BUS RESERVED		
B3	VME BUS RESERVED		
B4	VME BUS RESERVED		
B5	VME BUS RESERVED		
B6	VME BUS RESERVED		
B7	VME BUS RESERVED		
B8	VME BUS RESERVED		
B9	VME BUS RESERVED		
B10	VME BUS RESERVED		
B11	VME BUS RESERVED		
B12	VME BUS RESERVED		
B13	VME BUS RESERVED		
B14	VME BUS RESERVED		
B15	VME BUS RESERVED		
B16	VME BUS RESERVED		
B17	VME BUS RESERVED		
B18	VME BUS RESERVED		
B19	VME BUS RESERVED		
B20	VME BUS RESERVED		
B21	VME BUS RESERVED		
B22	VME BUS RESERVED		
B23	VME BUS RESERVED		
B24	VME BUS RESERVED		
B25	VME BUS RESERVED		
B26	VME BUS RESERVED		
B27	VME BUS RESERVED		
B28	VME BUS RESERVED		
B29	VME BUS RESERVED		
B30	VME BUS RESERVED		
B31	VME BUS RESERVED		
B32	VME BUS RESERVED		
C1	P2DATA46	AL31	AN17
C2	P2DATA47	AL32	AM17

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
C3	P2DATA48	AL38	AP17
C4	P2DATA49	AL39	AR17
C5	P2DATA50	AL35	AT17
C6	P2DATA51	AL36	AU17
C7	P2DATA52	AL33	AM18
C8	P2DATA53	AL34	AN18
C9	P2DATA54	AM41	AR18
C10	P2DATA55	AM42	AT18
C11	P2DATA56	AN40	AV18
C12	P2DATA57	AM40	AV17
C13	P2DATA58	AM34	AY17
C14	P2DATA59	AM35	AY18
C15	P2DATA60	AM38	AL19
C16	P2DATA61	AM39	AM19
C17	P2DATA62	AM36	AN19
C18	P2DATA63	AM37	AP19
C19	P2DATA64	AM33	AW17
C20	P2DATA65	AN34	AW16
C21	P2DATA66	AN41	AU19
C22	P2DATA67	AN42	AV19
C23	P2DATA68	AN37	AY19
C24	P2DATA69	AN38	AW19
C25	P2DATA70	AN35	AL20
C26	P2DATA71	AN36	AL21
C27	P2DATA72	AP41	AM20
C28	P2DATA73	AP42	AN20
C29	P2DATA74	AP38	AR20
C30	P2DATA75	AP39	AP20
C31	P2DATA76	AP35	AU20
C32	P2DATA77	AR36	AV20
D1	P2DATA78	AP36	AW20
D2	GND		
D3	P2DATA80	AP37	AY20
D4	GND		
D5	GND		
D6	5V		
D7	GND		
D8	5V		
D9	GND		
D10	5V		
D11	5V		
D12	GND		
D13	5V		
D14	GND		
D15	5V		

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
D16	GND		
D17	5V		
D18	5V		
D19	GND		
D20	GND		
D21	5V		
D22	GND		
D23	5V		
D24	5V		
D25	GND		
D26	GND		
D27	5V		
D28	GND		
D29	5V		
D30	5V		
D31	VME BUS RESERVED		
D32	VME BUS RESERVED		
Z1	P2DATA00	AF33	AP10
Z2	GND		
Z3	P2DATA01	AF34	AR10
Z4	GND		
Z5	P2DATA02	AG40	AT10
Z6	GND		
Z7	P2DATA03	AG41	AU10
Z8	GND		
Z9	P2DATA04	AG32	AW10
Z10	GND		
Z11	P2DATA05	AG33	AV10
Z12	GND		
Z13	P2DATA06	AG38	AP11
Z14	GND		
Z15	P2DATA07	AG39	AR11
Z16	GND		
Z17	P2DATA08	AG36	AU11
Z18	GND		
Z19	P2DATA09	AG37	AV11
Z20	GND		
Z21	P2DATA10	AH35	AY11
Z22	GND		
Z23	P2DATA11	AG35	AY10
Z24	GND		
Z25	P2DATA12	AH41	AM12
Z26	GND		
Z27	P2DATA13	AH42	AN12
Z28	GND		

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
Z29	P2DATA14	AJ40	AP12
Z30	GND		
Z31	P2DATA15	AH40	AR12
Z32	GND		

Table B-4: WILDSTAR™-II PRO P2 Backplane Pin Mapping with 2.5V Differential I/O

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
A1	P2CLKIN+	AF35	AT21
A2	P2CLKIN-	AF36	AU21
A3	P2DATA08+	AH31	AU12
A4	P2DATA08-	AH32	AT12
A5	P2DATA09+	AH37	AV12
A6	P2DATA09-	AH38	AW12
A7	P2DATA10+	AH33	AL13
A8	P2DATA10-	AH34	AM13
A9	P2DATA11+	AJ31	AN13
A10	P2DATA11-	AJ32	AP13
A11	P2DATA12+	AJ41	AR13
A12	P2DATA12-	AJ42	AT13
A13	P2DATA13+	AJ37	AU13
A14	P2DATA13-	AJ38	AV13
A15	P2DATA14+	AJ33	AY13
A16	P2DATA14-	AJ34	AW13
A17	P2DATA15+	AJ35	AL14
A18	P2DATA15-	AJ36	AL15
A19	P2DATA16+	AK41	AU15
A20	P2DATA16-	AK42	AV15
A21	P2DATA17+	AK31	AY15
A22	P2DATA17-	AK32	AY14
A23	P2DATA18+	AK39	AL16
A24	P2DATA18-	AK40	AM16
A25	P2DATA19+	AK37	AN16
A26	P2DATA19-	AK38	AP16
A27	P2DATA20+	AK33	AT16
A28	P2DATA20-	AK34	AR16
A29	P2DATA21+	AK35	AU16
A30	P2DATA21-	AK36	AV16
A31	P2DATA22+	AL40	AL17
A32	P2DATA22-	AL41	AL18
B1	VME BUS RESERVED		
B2	VME BUS RESERVED		
B3	VME BUS RESERVED		
B4	VME BUS RESERVED		
B5	VME BUS RESERVED		
B6	VME BUS RESERVED		
B7	VME BUS RESERVED		
B8	VME BUS RESERVED		
B9	VME BUS RESERVED		
B10	VME BUS RESERVED		

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
B11	VME BUS RESERVED		
B12	VME BUS RESERVED		
B13	VME BUS RESERVED		
B14	VME BUS RESERVED		
B15	VME BUS RESERVED		
B16	VME BUS RESERVED		
B17	VME BUS RESERVED		
B18	VME BUS RESERVED		
B19	VME BUS RESERVED		
B20	VME BUS RESERVED		
B21	VME BUS RESERVED		
B22	VME BUS RESERVED		
B23	VME BUS RESERVED		
B24	VME BUS RESERVED		
B25	VME BUS RESERVED		
B26	VME BUS RESERVED		
B27	VME BUS RESERVED		
B28	VME BUS RESERVED		
B29	VME BUS RESERVED		
B30	VME BUS RESERVED		
B31	VME BUS RESERVED		
B32	VME BUS RESERVED		
C1	P2DATA23+	AL31	AN17
C2	P2DATA23-	AL32	AM17
C3	P2DATA24+	AL38	AP17
C4	P2DATA24-	AL39	AR17
C5	P2DATA25+	AL35	AT17
C6	P2DATA25-	AL36	AU17
C7	P2DATA26+	AL33	AM18
C8	P2DATA26-	AL34	AN18
C9	P2DATA27+	AM41	AR18
C10	P2DATA27-	AM42	AT18
C11	P2DATA28+	AN40	AV18
C12	P2DATA28-	AM40	AV17
C13	P2DATA29+	AM34	AY17
C14	P2DATA29-	AM35	AY18
C15	P2DATA30+	AM38	AL19
C16	P2DATA30-	AM39	AM19
C17	P2DATA31+	AM36	AN19
C18	P2DATA31-	AM37	AP19
C19	P2DATA32+(SE ONLY)	AM33	AW17
C20	P2DATA32- (SE ONLY)	AN34	AW16
C21	P2DATA33+	AN41	AU19
C22	P2DATA33-	AN42	AV19

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
C23	P2DATA34+	AN37	AY19
C24	P2DATA34-	AN38	AW19
C25	P2DATA35+	AN35	AL20
C26	P2DATA35-	AN36	AL21
C27	P2DATA36+	AP41	AM20
C28	P2DATA36-	AP42	AN20
C29	P2DATA37+	AP38	AR20
C30	P2DATA37-	AP39	AP20
C31	P2DATA38+	AP35	AU20
C32	P2DATA38-	AR36	AV20
D1	P2DATA39+	AP36	AW20
D2	P2DATA40+	AR40	AY23
D3	P2DATA39-	AP37	AY20
D4	P2DATA40-	AR41	AW23
D5	P2DATA41+	AR37	AV23
D6	P2DATA41-	AT38	AU23
D7	P2DATA42+	AR38	AP23
D8	P2DATA42-	AR39	AR23
D9	P2DATA43+	AT41	AN23
D10	P2DATA43-	AT42	AM23
D11	P2DATA44+	AT39	AL22
D12	P2DATA44-	AT40	AL23
D13	P2DATA45+	AU41	AW24
D14	P2DATA45-	AU42	AY24
D15	P2DATA46+	AU39	AV24
D16	P2DATA46-	AU40	AU24
D17	P2DATA47+	AW40	AT24
D18	P2DATA47-	AV40	AR24
D19	P2DATA48+	AV41	AP24
D20	P2DATA48-	AV42	AN24
D21	P2DATA49+	AW41	AM24
D22	P2DATA49-	AW42	AL24
D23	P2DATA50+	AU35	AY25
D24	P2DATA50-	AT35	AY26
D25	P2DATA51+	AU36	AV26
D26	P2DATA51-	AT37	AV25
D27	P2DATA52+	AY37	AT25
D28	P2DATA52-	AY38	AR25
D29	P2DATA53+	AW26	AN25
D30	P2DATA53-	AV36	AM25
D31	VME BUS RESERVED		
D32	VME BUS RESERVED		
Z1	P2DATA00+	AF33	AP10
Z2	GND		
Z3	P2DATA00-	AF34	AR10

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
Z4	GND		
Z5	P2DATA01+	AG40	AT10
Z6	GND		
Z7	P2DATA01-	AG41	AU10
Z8	GND		
Z9	P2DATA02+	AG32	AW10
Z10	GND		
Z11	P2DATA02-	AG33	AV10
Z12	GND		
Z13	P2DATA03+	AG38	AP11
Z14	GND		
Z15	P2DATA03-	AG39	AR11
Z16	GND		
Z17	P2DATA04+	AG36	AU11
Z18	GND		
Z19	P2DATA04-	AG37	AV11
Z20	GND		
Z21	P2DATA05+	AH35	AY11
Z22	GND		
Z23	P2DATA05-	AG35	AY10
Z24	GND		
Z25	P2DATA06+	AH41	AM12
Z26	GND		
Z27	P2DATA06-	AH42	AN12
Z28	GND		
Z29	P2DATA07+	AJ40	AP12
Z30	GND		
Z31	P2DATA07-	AH40	AR12
Z32	GND		

Table B-5: WILDSTAR™-II PRO P2 Backplane Pin Mapping with 3.3V Single-Ended I/O

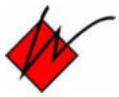
P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
A1	P2CLKIN0	AF35	AT21
A2	P2CLKIN1	AF36	AU21
A3	P2DATA16	AH31	AU12
A4	P2DATA17	AH32	AT12
A5	P2DATA18	AH37	AV12
A6	P2DATA19	AH38	AW12
A7	P2DATA20	AH33	AL13
A8	P2DATA21	AH34	AM13

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
A9	P2DATA22	AJ31	AN13
A10	P2DATA23	AJ32	AP13
A11	P2DATA24	AJ41	AR13
A12	P2DATA25	AJ42	AT13
A13	P2DATA26	AJ37	AU13
A14	P2DATA27	AJ38	AV13
A15	P2DATA28	AJ33	AY13
A16	P2DATA29	AJ34	AW13
A17	P2DATA30	AJ35	AL14
A18	P2DATA31	AJ36	AL15
A19	P2DATA32	AK41	AU15
A20	P2DATA33	AK42	AV15
A21	P2DATA34	AK31	AY15
A22	P2DATA35	AK32	AY14
A23	P2DATA36	AK39	AL16
A24	P2DATA37	AK40	AM16
A25	P2DATA38	AK37	AN16
A26	P2DATA39	AK38	AP16
A27	P2DATA40	AK33	AT16
A28	P2DATA41	AK34	AR16
A29	P2DATA42	AK35	AU16
A30	P2DATA43	AK36	AV16
A31	P2DATA44	AL40	AL17
A32	P2DATA45	AL41	AL18
B1	VME BUS RESERVED		
B2	VME BUS RESERVED		
B3	VME BUS RESERVED		
B4	VME BUS RESERVED		
B5	VME BUS RESERVED		
B6	VME BUS RESERVED		
B7	VME BUS RESERVED		
B8	VME BUS RESERVED		
B9	VME BUS RESERVED		
B10	VME BUS RESERVED		
B11	VME BUS RESERVED		
B12	VME BUS RESERVED		

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
B13	VME BUS RESERVED		
B14	VME BUS RESERVED		
B15	VME BUS RESERVED		
B16	VME BUS RESERVED		
B17	VME BUS RESERVED		
B18	VME BUS RESERVED		
B19	VME BUS RESERVED		
B20	VME BUS RESERVED		
B21	VME BUS RESERVED		
B22	VME BUS RESERVED		
B23	VME BUS RESERVED		
B24	VME BUS RESERVED		
B25	VME BUS RESERVED		
B26	VME BUS RESERVED		
B27	VME BUS RESERVED		
B28	VME BUS RESERVED		
B29	VME BUS RESERVED		
B30	VME BUS RESERVED		
B31	VME BUS RESERVED		
B32	VME BUS RESERVED		
C1	P2DATA46	AL31	AN17
C2	P2DATA47	AL32	AM17
C3	P2DATA48	AL38	AP17
C4	P2DATA49	AL39	AR17
C5	P2DATA50	AL35	AT17
C6	P2DATA51	AL36	AU17
C7	P2DATA52	AL33	AM18
C8	P2DATA53	AL34	AN18
C9	P2DATA54	AM41	AR18
C10	P2DATA55	AM42	AT18

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
C11	P2DATA56	AN40	AV18
C12	P2DATA57	AM40	AV17
C13	P2DATA58	AM34	AY17
C14	P2DATA59	AM35	AY18
C15	P2DATA60	AM38	AL19
C16	P2DATA61	AM39	AM19
C17	P2DATA62	AM36	AN19
C18	P2DATA63	AM37	AP19
C19	P2DATA64	AM33	AW17
C20	P2DATA65	AN34	AW16
C21	P2DATA66	AN41	AU19
C22	P2DATA67	AN42	AV19
C23	P2DATA68	AN37	AY19
C24	P2DATA69	AN38	AW19
C25	P2DATA70	AN35	AL20
C26	P2DATA71	AN36	AL21
C27	P2DATA72	AP41	AM20
C28	P2DATA73	AP42	AN20
C29	P2DATA74	AP38	AR20
C30	P2DATA75	AP39	AP20
C31	P2DATA76	AP35	AU20
C32	P2DATA77	AR36	AV20
D1	P2DATA78	AP36	AW20
D2	P2DATA79	AR40	AY23
D3	P2DATA80	AP37	AY20
D4	P2DATA81	AR41	AW23
D5	P2DATA82	AR37	AV23
D6	P2DATA83	AT38	AU23
D7	P2DATA84	AR38	AP23
D8	P2DATA85	AR39	AR23
D9	P2DATA86	AT41	AN23
D10	P2DATA87	AT42	AM23
D11	P2DATA88	AT39	AL22
D12	P2DATA89	AT40	AL23
D13	P2DATA90	AU41	AW24
D14	P2DATA91	AU42	AY24
D15	P2DATA92	AU39	AV24
D16	P2DATA93	AU40	AU24
D17	P2DATA94	AW40	AT24
D18	P2DATA95	AV40	AR24
D19	P2DATA96	AV41	AP24
D20	P2DATA97	AV42	AN24
D21	P2DATA98	AW41	AM24
D22	P2DATA99	AW42	AL24
D23	P2DATA100	AU35	AY25

P2 Pin	Signal Name	PRO ACE PIN	PRO VME PE2 Pin
D24	P2DATA101	AT35	AY26
D25	P2DATA102	AU36	AV26
D26	P2DATA103	AT37	AV25
D27	P2DATA104	AY37	AT25
D28	P2DATA105	AY38	AR25
D29	P2DATA106	AW26	AN25
D30	P2DATA107	AV36	AM25
D31	VME BUS RESERVED		
D32	VME BUS RESERVED		
Z1	P2DATA00	AF33	AP10
Z2	GND		
Z3	P2DATA01	AF34	AR10
Z4	GND		
Z5	P2DATA02	AG40	AT10
Z6	GND		
Z7	P2DATA03	AG41	AU10
Z8	GND		
Z9	P2DATA04	AG32	AW10
Z10	GND		
Z11	P2DATA05	AG33	AV10
Z12	GND		
Z13	P2DATA06	AG38	AP11
Z14	GND		
Z15	P2DATA07	AG39	AR11
Z16	GND		
Z17	P2DATA08	AG36	AU11
Z18	GND		
Z19	P2DATA09	AG37	AV11
Z20	GND		
Z21	P2DATA10	AH35	AY11
Z22	GND		
Z23	P2DATA11	AG35	AY10
Z24	GND		
Z25	P2DATA12	AH41	AM12
Z26	GND		
Z27	P2DATA13	AH42	AN12
Z28	GND		
Z29	P2DATA14	AJ40	AP12
Z30	GND		
Z31	P2DATA15	AH40	AR12
Z32	GND		



APPENDIX C: WILDSTAR™-II P0/P2 PIN USAGE

P0 Pins

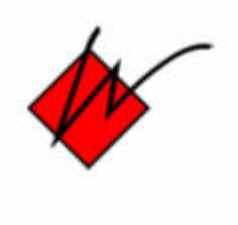
Item	P0 Pins	# of Pins	Optional GND	Notes
1	B1/A1	2		
2	B2/A2	2		
3	B3/A3	2		
4	B4/A4	2		
5	B5/A5	2		
6	B6/A6	2		
7	B7/A7	2		
8	B8/A8	2		
9	B9/A9	2		
10	A11/B11	2		
11	B12/A12	2		
12	A13/B13	2		
13	B14/A14	2		
14	B15/A15	2		
15	B16/A16	2		
16	B17/A17	2		
17	B18/A18	2		
18	B19/A19	2		
19	E1/D1	2		
20	E2/D2	2		
21	E3/D3	2		
22	E4/D4	2		
23	E5/D5	2		
24	E6/D6	2		
25	E7/D7	2		
26	E8/D8	2		
27	E9/D9	2		
28	E11/D11	2		
29	E12/D12	2		
30	E13/D13	2		
31	E14/D14	2		
32	E15/D15	2		
33	E16/D16	2		
34	E17/D17	2		
35	E18/D18	2		
36	E19/D19	2		
37	C07	1	1	
38	C08	1	1	
39	C09	1	1	
40	C10	1	1	
41	C11	1	1	

Item	P0 Pins	# of Pins	Optional GND	Notes
42	C12	1	1	
43	C13	1	1	
44	C14	1	1	
45	C15	1	1	
46	C16	1	1	
47	C17	1	1	
48	C18	1	1	
49	C19	1	1	
50	D10	1		
51	A10	1		
52	B10/A10	2		Global Clock Pins
53	E10/D10	2		Global Clock Pins
		91		

P2 Pins

Item	P2 Pin	# of Pins	Optional GND	Notes
1	Z01/A02	2	A2	
2	A04/A03	2		
3	Z07/A06	2		
4	A09/A08	2		
5	A12/A11	2		
6	Z13/A14	2		
7	Z15/A15	2		
8	A17/Z17	2		
9	A18/A19	2	A19	
10	Z19/A20	2		
11	Z21/A21	2		
12	A22/A23	2	A22	
13	A24/Z23	2		
14	A25/Z25	2	A25	
15	A27/A26	2		
16	A29/Z29	2		
17	A31/A30	2	A31	
18	A32/Z31	2		
19	C01/D01	2		
20	C03/C02	2		
21	D05/C04	2	D05/C04	
22	D06/C05	2		
23	D07/C07	2		
24	C09/D08	2		
25	D10/C10	2		
26	C12/D11	2		
27	D13/C13	2		

Item	P2 Pin	# of Pins	Optional GND	Notes
28	C16/C15	2		
29	C19/C18	2		
30	C22/C21	2		
31	D23/C23	2	C23	
32	D24/C24	2		
33	C25/C26	2	C26	
34	D26/C27	2		
35	C29/C28	2	C29	
36	C30/C31	2		
37	D04/D03	2		
38	D15/D14	2		
39	D18/D17	2		
40	D21/D20	2		
41	D28/D27	2	D28	
42	D30/D29	2		
43	Z03/Z05	2		
44	Z09/Z11	2		
45	Z27/A28	2	A28	
46	A1	1		Global Clock Pin
		93		



APPENDIX D: WILDSTAR™-II VHDL SRAM EXAMPLE

This section describes SRAM Examples for the WILDSTAR™-II /PCI and /VME mainboards. Because of design differences, the WILDSTAR™-II /PCI Rev. D SRAM Example is discussed separately.

SRAM Example for WILDSTAR™-II /PCI and /VME

This example illustrates the use of SRAM on a WILDSTAR™-II /PCI or /VME board. All six SRAM ports are used in this example and the data integrity checking is done at the hardware level. Host software is used to program the test Virtex™-II FPGA, set up the board processing clock (PCLK), and run the test.

This test writes the entire memory in an incrementing pattern to all six memories simultaneously, then reads back all six memories to check for data errors. If any errors are found, an error bit is set in the FPGA, which is read by the host software to indicate whether the test passed.

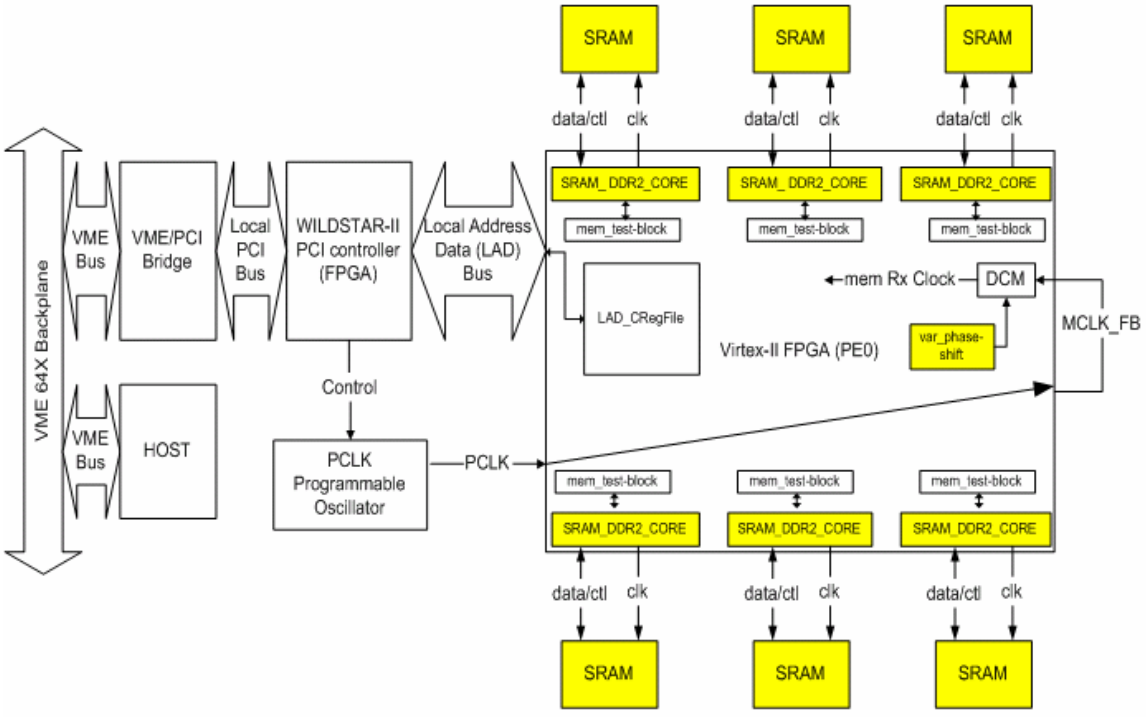


Figure D-1: Block Diagram for WILDSTAR™-II VME-Based Example

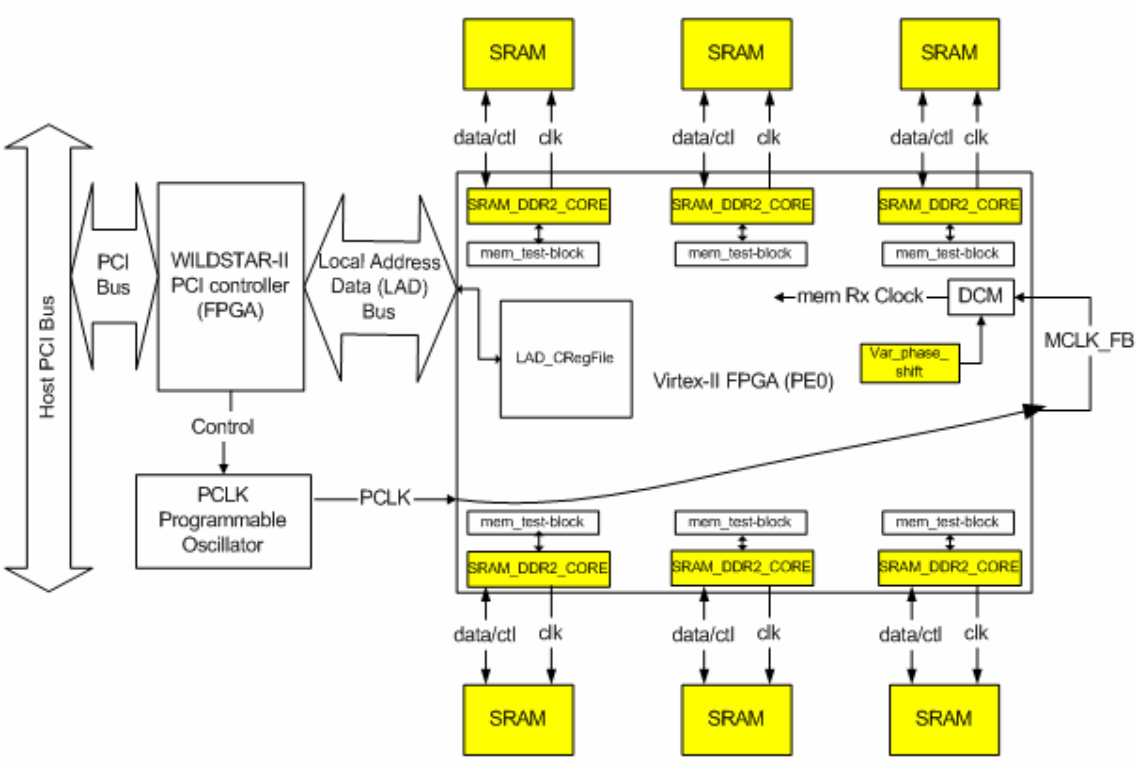


Figure D-2: Block Diagram for WILDSTAR™-II PCI-Based Example

Main Functional Blocks

The main functional blocks in this example are:

- **mem_test_block(6x)-** This block runs the SRAM test. It accepts some global input signals to determine how much memory to test and when to start. Its output indicates Done status and whether an error was detected.

- **SRAM_DDR2_CORE(6x)-** This block, included with the VHDL model, is used to interface the SRAM memory. P_CLK is used to drive the memory signals to the SRAM, and the M_ClkFb clock is used to clock the SRAM read data back in to the PE. This core also supplies the SRAM with its clock (P_Clk). SRAM read data is clocked in with the M_ClkFb signal.

- **var_phase_shift(1x)-** This core phase shifts a particular clock. In this example, it phase shifts the M_ClkFb clock, which mimics a return clock from the SRAM. As the P_Clk frequency is increased, this M_ClkFb must be shifted to clock in the SRAM read data at the ideal time. This block is controlled from the host.

- **LAD_CRegFile(1x)-** This is a host-accessible register block used to pass parameters to the example such as memory size and start. It also passes error information and Done status back to the host.

Software Description

The host code for this example does the following:

1) Opens the WILDSTAR™-II board.
2) Reads the ID prom to determine what type of SRAM is present and its size.
3) Sets the board processing clock (P_Clk).
4) Programs the PE with the current image for the SRAM found in Step 2.
5) Resets the PE.
6) Programs the phase shift into the "var_phase_shift" block based on P_Clk.
7) Writes to the PE's control register to start the PE writing data into memory.
8) Waits for the write to finish.
9) Writes to the PE's control register to start the PE reading data from memory.
10) Waits for the read to finish.
11) Checks the error bit in the PE to see if any data errors were detected.
12) Closes the WILDSTAR™-II board.

Command Line Parameters

These can also be seen by using the –H option at the command line.

Usage:
sram_ex [-S <slot number>] [-F <P clock frequency>] [-E <U clock frequency>]
[-P <PE number] [-V] [-B]

Options:

-S <Slot #>	Set board slot number.	(default = 0)
-F <Frequency>	Set the P Clock Frequency in MHz. (default = 138; minimum acceptable frequency is 40MHz, and the maximum acceptable frequency is 140MHz)	
-P <PE #>	Set the PE to test. {0,1,2}	(default = 0)
-V	Set platform to VMIC	(default = disabled)
-B	Set platform to SBS BIT3	(default = disabled)
-H	Print this help	

Program Sample Output

Below is a sample output for running the test. Note that because of granularity, the P Clock setting requested by the user (100.000000) and the actual frequency (102.000000) are slightly different.

```
D:\annapolis\examples\wsii\sram_ex\win>sram_ex -s 5 -B -f 100
Setting Board Number to 5
Opening Board ... DONE
Setting P Clock to 100.000000.
P Clock Set to 102.000000
Programming PE with XC2V6000\FF1517\wsii_vme_pe_ddr2_reva.x86 ...
DONE

**********************************************************************
* This Example tests the SRAM ports on the WILDSTAR(tm) –II board. *
**********************************************************************

 Testing 262144 QWORDS, per port ( 2.0 MB)

  Resetting PE0                 ... DONE
  Setting Phase shift for PE0   ... DONE
  Waiting for clocks to lock    ... DONE
  Starting Write                ... DONE
  Waiting for write to complete ... DONE
  Starting Read                 ... DONE
  Waiting for read to complete  ... DONE
  Verify                        ... SUCCESS
```

SRAM Example for WILDSTAR™-II /PCI (Rev. D)

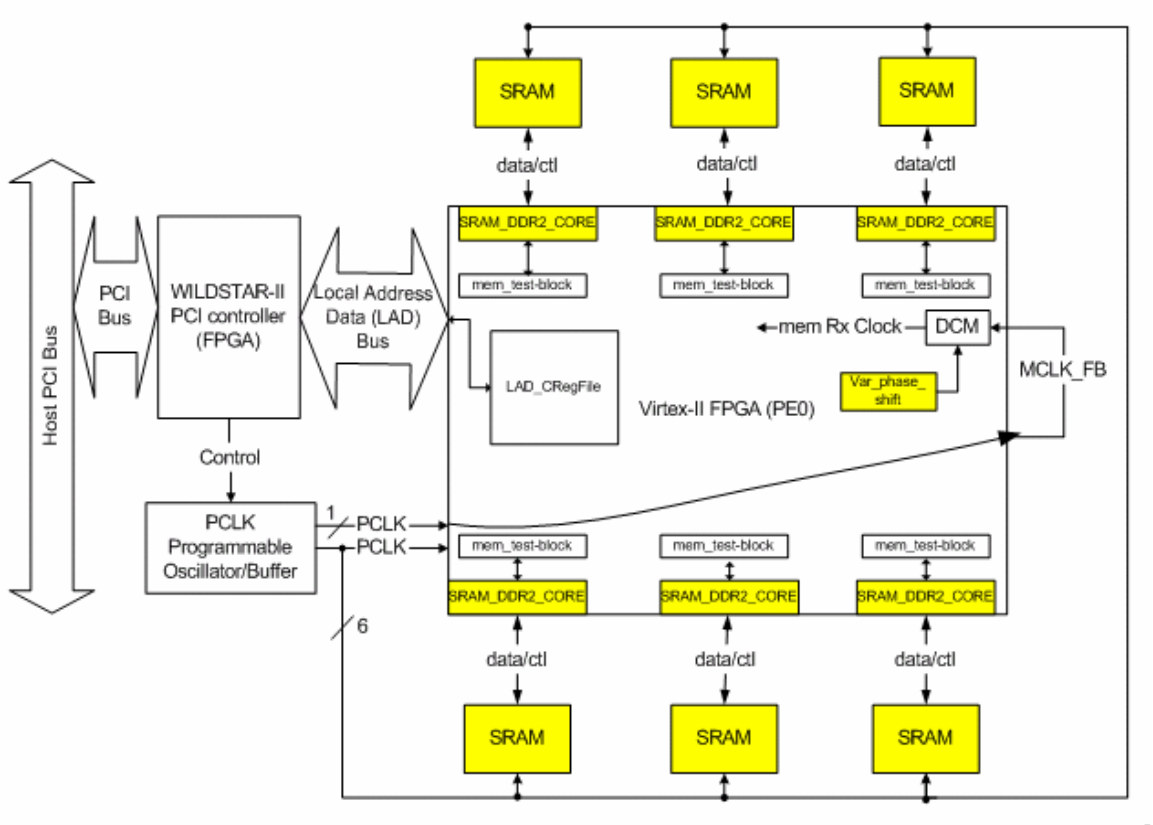


Figure D-3: Block Diagram for WILDSTAR™-II PCI (Rev. D) Example

Main Functional Blocks

The main functional blocks in this example are:

- **mem_test_block(6x)**- This block runs the SRAM test. It accepts some global input signals to determine how much memory to test and when to start. Its output indicates Done status and whether an error was detected.

- **SRAM_DDR2_CORE(6x)**- This block, included with the VHDL model, is used to interface the SRAM memory. P_CLK is used to drive the memory signals to the SRAM, and the M_ClkFb clock is used to clock the SRAM read data back in to the PE. This core also supplies the SRAM with its clock (P_Clk). SRAM read data is clocked in with the M_ClkFb signal.

- **var_phase_shift(1x)**- This core phase shifts a particular clock. In this example, it phase-shifts the M_ClkFb clock, which mimics a return clock from the SRAM. As the P_Clk frequency is increased, this M_ClkFb must be shifted to clock in the SRAM read data at the ideal time. This block is controlled from the host.

- **LAD_CRegFile(1x)**- This is a host-accessible register block used to pass parameters to the example such as memory size and start. It also passes error information and Done status back to the host.

Software Description

The host code for this example does the following:

1) Opens the WILDSTAR™-II board.
2) Reads the ID prom to determine what type of SRAM is present and its size.
3) Sets the board processing clock (P_Clk).
4) Programs the PE with the current image for the SRAM found in Step 2.
5) Resets the PE.
6) Programs the phase shift into the "var_phase_shift" block based on P_Clk.
7) Writes to the PE's control register to start the PE writing data into memory.
8) Waits for the write to finish.
9) Writes to the PE's control register to start the PE reading data from memory.
10) Waits for the read to finish.
11) Checks the error bit in the PE to see if any data errors were detected.
12) Closes the WILDSTAR™-II board.

Command Line Parameters

These can also be seen by using the –H option at the command line.

Usage:
sram_ex [-S <slot number>] [-F <P clock frequency>] [-E <U clock frequency>]
[-P <PE number] [-V] [-B]

Options:

-S <Slot #>	Set board slot number.	(default = 0)
-F <Frequency>	Set the P Clock Frequency in MHz. (default = 138; minimum acceptable frequency is 40MHz, and the maximum acceptable frequency is 140MHz)	
-P <PE #>	Set the PE to test. {0,1,2}	(default = 0)
-V	Set platform to VMIC	(default = disabled)

-B	Set platform to SBS BIT3	(default = disabled)
-H	Print this help	

Program Sample Output

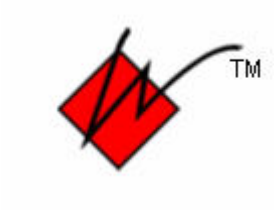
Below is a sample output for running the test. Note that because of granularity, the P Clock setting requested by the user (100.000000) and the actual frequency (102.000000) are slightly different.

```
D:\annapolis\examples\wsii\sram_ex\win>sram_ex -s 0 -f 100
Setting Board Number to 0
Opening Board ... DONE
Setting P Clock to 100.000000.
P Clock Set to 100.000000
Programming PE with XC2V6000\FF1517\wsii_pci_pe_ddr2_revd.x86 ... DONE

**********************************************************************
* This Example tests the SRAM ports on the WILDSTAR(tm) –II board. *
**********************************************************************

 Testing 262144 QWORDS, per port ( 2.0 MB)

  Resetting PE0                  ... DONE
  Setting Phase shift for PE0    ... DONE
  Waiting for clocks to lock     ... DONE
  Starting Write                 ... DONE
  Waiting for write to complete  ... DONE
  Starting Read                  ... DONE
  Waiting for read to complete   ... DONE
   Verify                        ... SUCCESS
```


TM

APPENDIX E: WILDSTAR™-II VHDL SDRAM EXAMPLE

SDRAM Example for WILDSTAR™-II /PCI and /VME

This example illustrates the use of SDRAM on a WILDSTAR™-II /PCI or /VME board. Host software is used to program the test Virtex™-II FPGA, set up the board processing clock (PCLK), and run the test.

The test consists of writing a user-settable amount of data to the SDRAM and then reading it back and verifying the contents. The results are displayed on the screen.

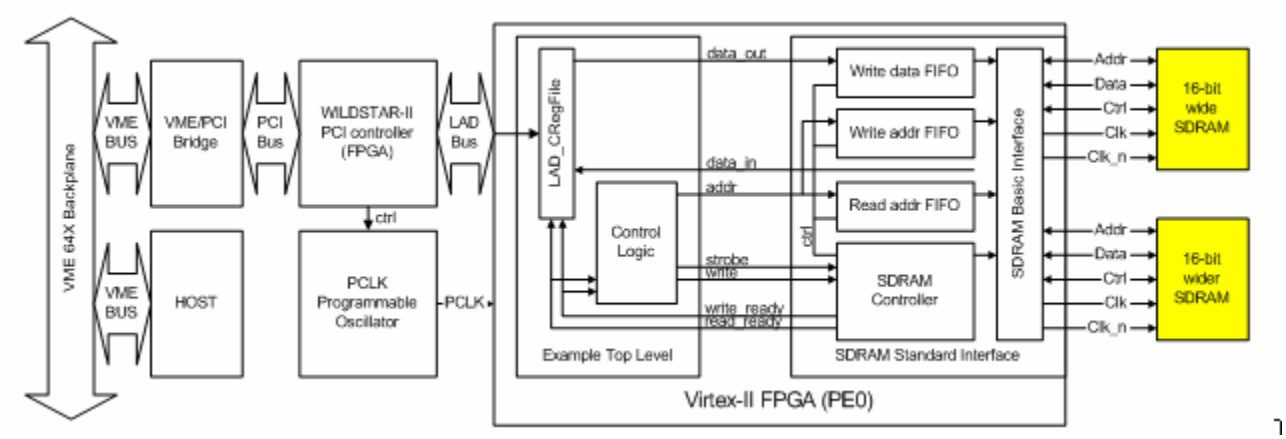


Figure E-1: Block Diagram for WILDSTAR™-II /VME-Based Example

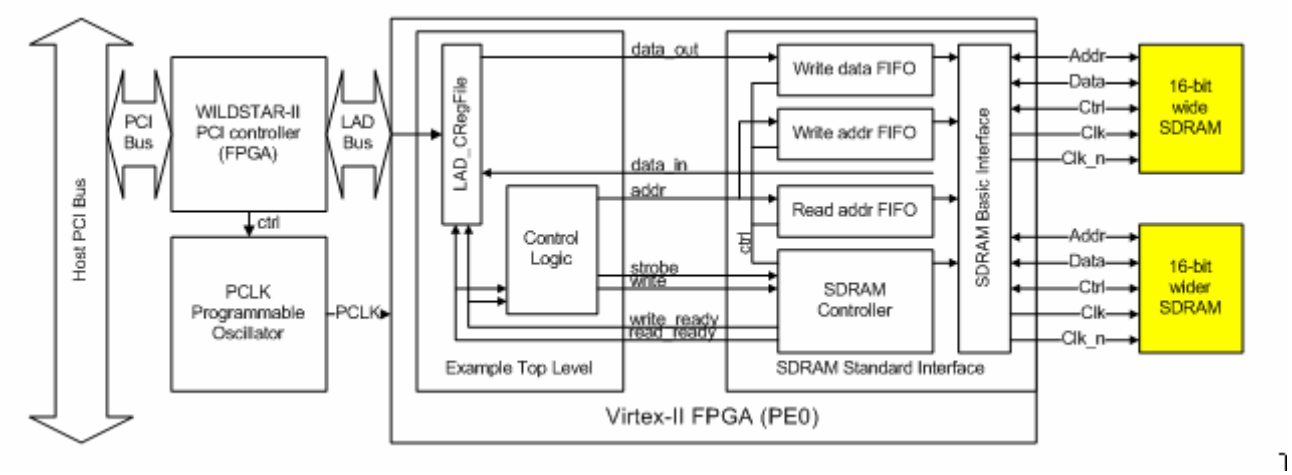


Figure E-2: Block Diagram for WILDSTAR™-II /PCI-Based Example

Main Functional Blocks

The main functional blocks in this example are:

- **Pe_dram_ex_arch** - This block is the top level of the SDRAM test. It interfaces the host for control, generates write data, and interfaces the SDRAM through the SDRAM standard interface.

- **LAD_CRegFile(1x)**- This is a host-accessible register block used to pass read and write status to the host. It is also used to get write data from the host and send read data back.

Software Description

The host code for this example does the following:

1) Opens the WILDSTAR™-II board.
2) Sets the board processing clock (P_Clk).
3) Programs the PE with the SDRAM example image.
4) Resets the PE.
5) Reads the status register to make sure that the image is ready to write.
6) Writes incremental data to the data register.
7) Repeats steps 5 and 6 until the user-settable amount of data is written.
8) Reads the status register to make sure that the image is ready to read.
9) Reads data from the data register.
10) Verifies that the data correct and prints the results to the screen.
11) Repeats steps 8, 9, and 10 until the user-settable amount of data is read.

12) Prints the final results to the screen.

13) Closes the WILDSTAR™-II board.

Command Line Parameters

These can also be seen by using the –H option at the command line.

Usage:
dram_ex [-S <slot number>] [-F <P clock frequency>] [-P <PE number] [-V] [-B] [-D quantity of data (even number)>]

Options:

-S <Slot #>	Set board slot number.	(default = 0)
-F <Frequency>	Set the P Clock Frequency in MHz.	(default = 133)
-P <PE #>	Set the PE to test. {0,1,2}	(default = 0)
-V	Set platform to VMIC	(default = disabled)
-B	Set platform to SBS BIT3	(default = disabled)
-D	Set quantity of data (even number)	(default = 10)
-H	Print this help	

Program Sample Output

Below is a sample output for running the test.

```
C:\annapolis\examples\wsii\dram_ex\win>dram_ex -s 4 -b -f 133 -d 16
Setting Board Number to 4
Opening Board ... DONE
Setting P Clock to 133.000000.
P Clock Set to 133.000000
Programming PE with XC2V6000\FF1517\wsii_vme_pe_revb.x86 ... DONE

  Resetting PE                ... DONE

WSII VME SDRAM Interface Example begin
Write section begin
Writing word #1       DRAM data = 0x1.
Writing word #2       DRAM data = 0x2.
Writing word #3       DRAM data = 0x3.
Writing word #4       DRAM data = 0x4.
Writing word #5       DRAM data = 0x5.
```

```
Writing word #6         DRAM data = 0x6.
Writing word #7         DRAM data = 0x7.
Writing word #8         DRAM data = 0x8.
Writing word #9         DRAM data = 0x9.
Writing word #10        DRAM data = 0xa.
Writing word #11        DRAM data = 0xb.
Writing word #12        DRAM data = 0xc.
Writing word #13        DRAM data = 0xd.
Writing word #14        DRAM data = 0xe.
Writing word #15        DRAM data = 0xf.
Writing word #16        DRAM data = 0x10.
Write section complete
Read back section begin
Read word #1    Correct data = 0x1      Actual data = 0x1      Pass
Read word #2    Correct data = 0x2      Actual data = 0x2      Pass
Read word #3    Correct data = 0x3      Actual data = 0x3      Pass
Read word #4    Correct data = 0x4      Actual data = 0x4      Pass
Read word #5    Correct data = 0x5      Actual data = 0x5      Pass
Read word #6    Correct data = 0x6      Actual data = 0x6      Pass
Read word #7    Correct data = 0x7      Actual data = 0x7      Pass
Read word #8    Correct data = 0x8      Actual data = 0x8      Pass
Read word #9    Correct data = 0x9      Actual data = 0x9      Pass
Read word #10   Correct data = 0xa      Actual data = 0xa      Pass
Read word #11   Correct data = 0xb      Actual data = 0xb      Pass
Read word #12   Correct data = 0xc      Actual data = 0xc      Pass
Read word #13   Correct data = 0xd      Actual data = 0xd      Pass
Read word #14   Correct data = 0xe      Actual data = 0xe      Pass
Read word #15   Correct data = 0xf      Actual data = 0xf      Pass
Read word #16   Correct data = 0x10     Actual data = 0x10     Pass
Read back section complete
Success ... there were no errors.
WSII VME SDRAM Interface Example complete
```

APPENDIX F: COREFIRE™ SUPPORT

The CoreFire™ Design Suite, an FPGA programming tool produced by Annapolis Micro Systems, allows you to build PE designs quickly and load them onto WILDSTAR™-II boards. You simply create dataflow diagrams by choosing cores from the available libraries, dropping them into an editing field and connecting their ports. The CoreFire™ Design Suite includes the Application Builder and an Application Debugger, the latter being useful for monitoring dataflow through a diagram.

When you build a diagram, CoreFire™ generates the hardware design and a corresponding software model. The software model functions as an interface between the board hardware and the user application. The CoreFire™ Application Debugger also relies on the software model to access trace points that have been built into the design.

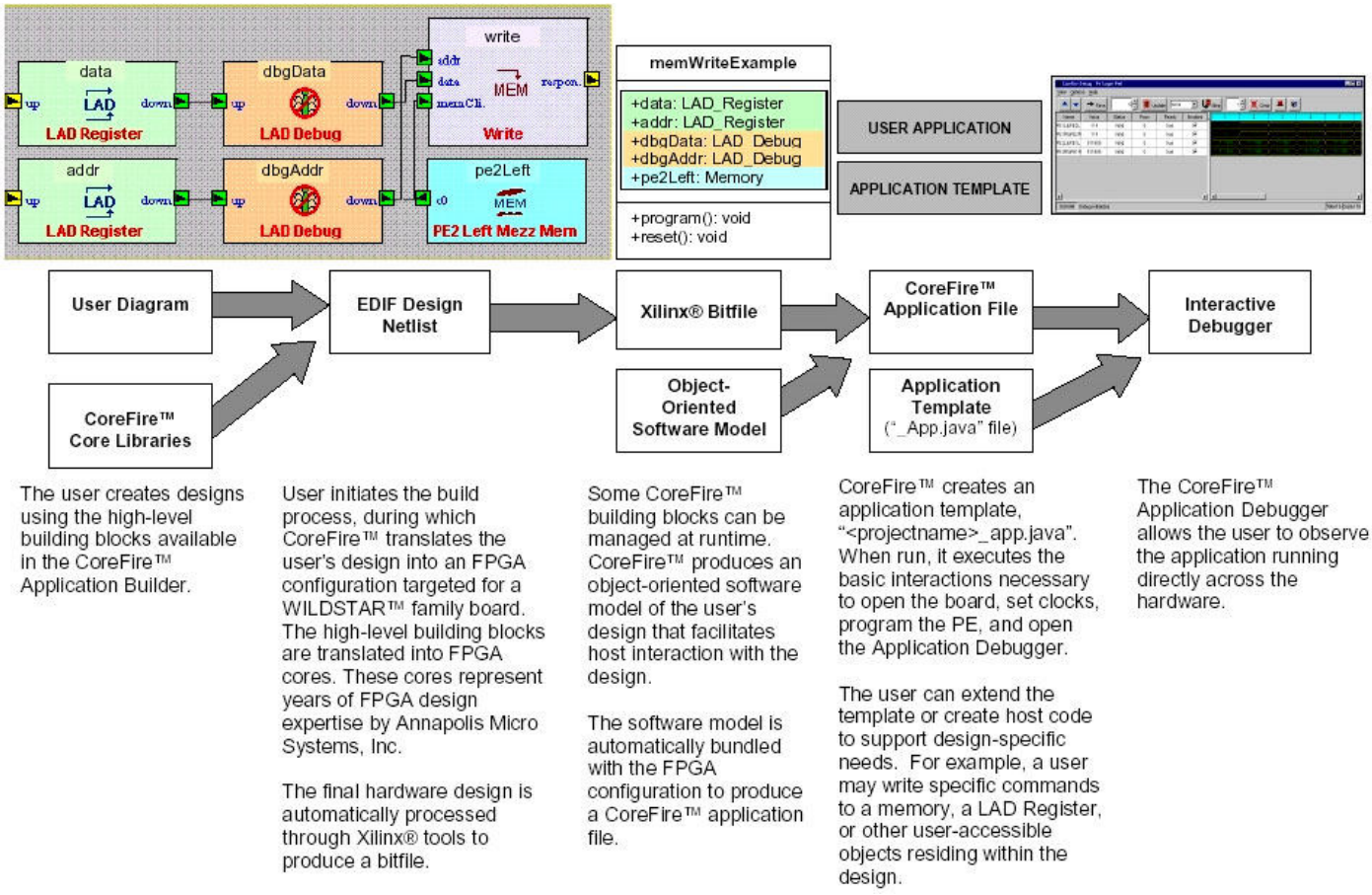


Figure F-1: CoreFire Design Flow Model

CoreFire™ "cores" are the building blocks used for creating dataflow diagrams. Cores are grouped in libraries according to function, with libraries dedicated to specific PEs on

WILDSTAR™ motherboards and I/O cards manufactured by Annapolis Micro Systems. You can also build customized cores, called Core Macros, using cores from the libraries.

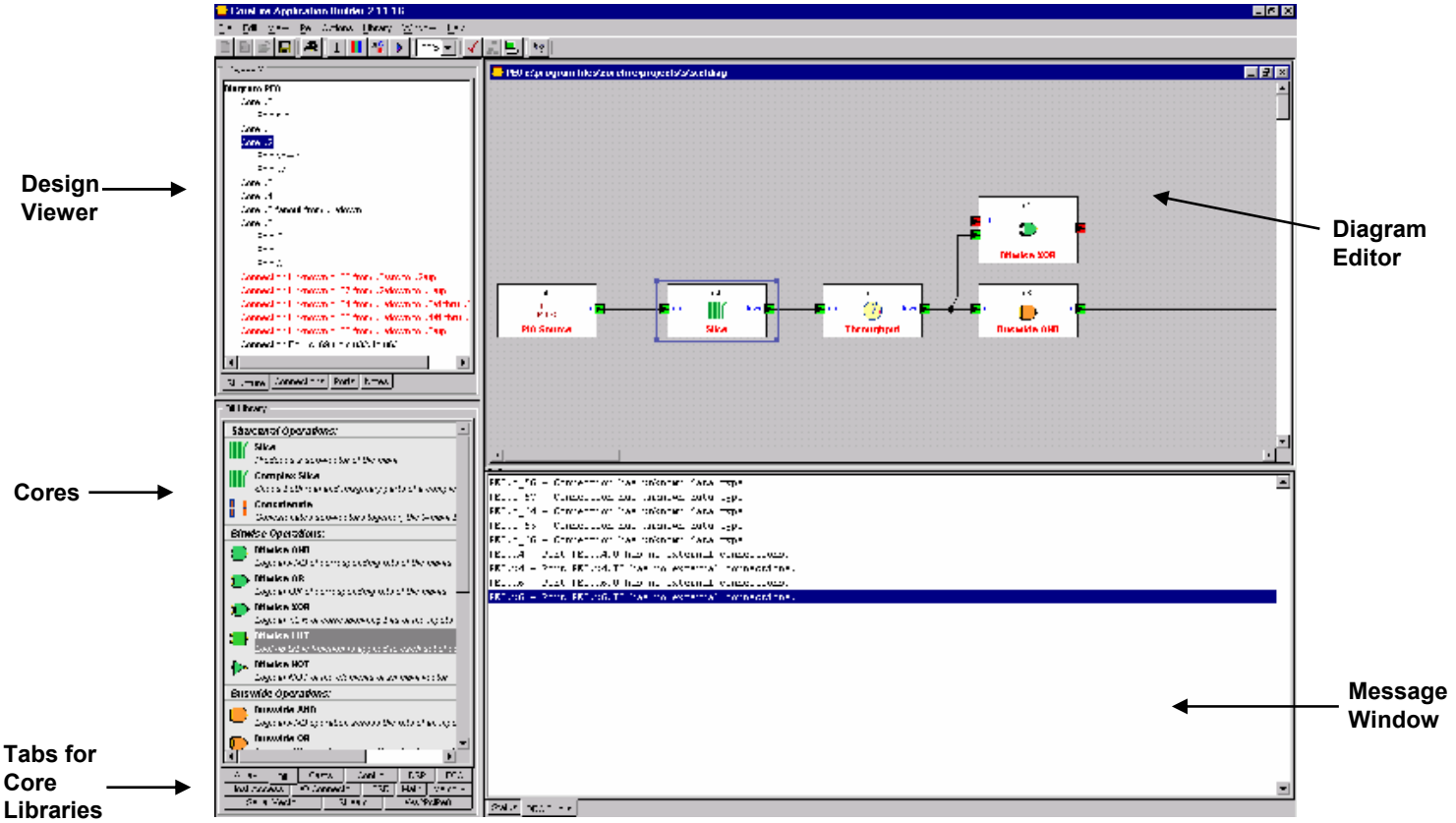


Figure F-2: CoreFire™ Application Builder—Main Interface

For more information about the CoreFire™ Design Suite, contact Annapolis Micro Systems, Inc. at (410) 841-2514 or wfinfo@annapmicro.com, or visit www.annapmicro.com.

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP