

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.

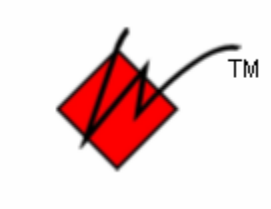
Find the *Annapolis Firebird Dual G-LINK* at our website: ***Click HERE***

FIREBIRD™ Hardware Reference Manual

(For FIREBIRD™/PCI, c/PCI, /PMC Dual WSDP™, and PMC Dual G-Link)

12727-0000 Revision 3.0

© Copyright 2000-2004 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™, STARFIRE™, STARFIRE™ II, WILDSTAR™, WILDSTAR™-II, 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.


TM

ANNAPOLIS MICRO SYSTEMS, INC. - LICENSE AGREEMENT

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

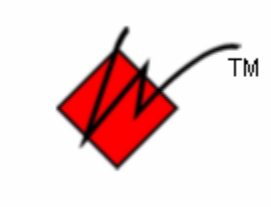


TABLE OF CONTENTS

1 ABOUT THIS MANUAL 1-1

1.1 Introduction 1-1
1.2 How This Manual Is Organized 1-1
1.3 Attachments 1-2
1.4 Conventions 1-2
1.5 Icons 1-3
1.6 Key Words and Definitions 1-3

2 INTRODUCTION TO FIREBIRD™ 2-1

2.1 FIREBIRD™/PCI Features 2-1
2.2 Features Common to PMC Dual WSDP™ and PMC Dual G-Link 2-2

3 GETTING STARTED 3-1

3.1 Unpacking 3-1
3.2 Visual Inspection 3-1

4 FIREBIRD™ HARDWARE INSTALLATION 4-1

4.1 Installing FIREBIRD™/PCI 4-2
4.1.1 3.3V and 5V PCI Signaling 4-2
4.1.2 FIREBIRD™/PCI Hardware Installation (All Board Revisions) 4-3
4.1.3 FIREBIRD™/PCI Rev. A Board 4-4
4.1.3.1 FIREBIRD™/PCI Rev. A Switch Settings 4-4
4.1.3.2 FIREBIRD™/PCI Rev. A LED Definitions 4-6
4.1.4 FIREBIRD™/PCI Board (Revs. B, C, D) 4-7
4.1.4.1 FIREBIRD™/PCI Rev. B, C, D-- Switch Settings 4-8
4.1.4.2 FIREBIRD™/PCI Rev. B , C, D-- LED Definitions 4-9
4.2 Installing FIREBIRD™/PMC 4-10
4.2.1 3.3V and 5V PCI Signaling (Dual WSDP™, Dual G-Link) 4-10
4.2.2 FIREBIRD™/PMC Dual WSDP™ Hardware Installation 4-10
4.2.2.1 Cable Requirements and Installation for PMC Dual WSDP™ 4-11
4.2.3 FIREBIRD™/PMC Dual WSDP™ Board 4-12
4.2.3.1 FIREBIRD™/PMC Switch Settings (Dual WSDP™, Dual G-Link) 4-13
4.2.3.2 FIREBIRD™/PMC Dual WSDP™ LED Definitions 4-14
4.2.4 FIREBIRD™/PMC Dual G-Link Hardware Installation 4-16
4.2.4.1 Cable Requirements and Installation for Dual G-Link 4-16
4.2.5 FIREBIRD™/PMC Dual G-Link Board 4-17
4.2.6 FIREBIRD™/PMC Dual G-Link Switch Settings 4-19
4.2.7 FIREBIRD™/PMC Dual G-Link LED Definitions 4-19
4.3 Host Software for FIREBIRD™/PCI and PMC 4-20

5 TECHNICAL SUPPORT 5-1

6 HARDWARE REFERENCE 6-1

6.1 FIREBIRD™ Hardware 6-1
6.2 General Specifications 6-1
6.3 Power Consumption 6-1
6.4 FIREBIRD™ Board Clocks 6-2
6.4.1 Frequency Parameters 6-2
6.4.2 Clock Skew 6-3
6.4.3 UCLK 6-3
6.4.4 MCLK 6-3
6.4.5 KCLK 6-3
6.4.6 XCLK (FIREBIRD™/PMC only) 6-3
6.4.7 FIREBIRD™/PCI Clock Diagram 6-4
6.4.8 FIREBIRD™/PMC Dual WSDP™ Clock Diagram 6-5
6.4.9 FIREBIRD™/PMC Dual G-Link Clock Diagram 6-6
6.4.10 Front Panel Clocks for FIREBIRD™/PMC Dual G-Link 6-7
6.4.10.1 LVPECL Differential Receive 6-8
6.4.10.2 50 Ohm (3.3v max.), Single-Ended Receive (Default Setting) 6-9
6.4.10.3 Series Terminated 6-10
6.4.10.4 Send/Receive TRIG—Receive on FPINCLK 6-11
6.4.10.5 Dual G-Link Termination Options 6-12
6.4.10.6 Transmit and Receive Clocks 6-12
6.4.10.7 Sourcing Motherboard Clocks (MCLK, UCLK) 6-13
6.5 FIREBIRD™/PMC Dual G-Link Fiber Optic Channels 6-13
6.5.1 Receive Modes 6-13
6.5.1.1 Normal Mode 6-13
6.5.1.2 PASS Mode 6-14
6.5.2 Transmit Modes 6-16
6.5.3 Transceiver Configuration Registers 6-17
6.6 DC Characteristics 6-20
6.6.1 FIREBIRD™/PCI 6-20
6.7 FIREBIRD™ External Interface Options 6-21
6.7.1 External I/O Cards 6-21
6.7.1.1 Euro I/O Cards 6-21
6.8 FIREBIRD™/PMC Dual WSDP™ User VHDL MICTOR® Pin Mapping 6-21

7 VHDL MODELS REFERENCE AND GUIDE 7-1

7.1 FIREBIRD™ Block Diagrams 7-1
7.2 VHDL Components Common to all FIREBIRD™ Types 7-4
7.2.1 LAD 64 Bus Standard Interface (LAD64_Bus_Std_IF) 7-4
7.2.2 LED Standard Interface (LED_Std_IF) 7-5
7.2.3 On-board 64-bit Memory Standard Interface (Mem64_Std_IF) 7-6
7.2.4 On-Board 32-bit Memory Standard Interface (Mem32_Std_IF) 7-8
7.3 VHDL Models Specific to FIREBIRD™/PCI 7-10
7.3.1 Clock Standard Interface (Clock_Std_IF) 7-10
7.4 VHDL Models Specific to FIREBIRD™/PMC (WSDP™ and G-Link) ... 7-13
7.4.1 Clock Standard Interface (Clock_Std_IF) 7-13
7.4.2 Backplane Standard Interface (Backplane_Std_IF) 7-17
7.5 VHDL Models Specific to FIREBIRD™/PMC Dual WSDP™ Only 7-19
7.5.1 FIREBIRD™/PMC Dual WSDP™ PE0 Model 7-19
7.5.1.1 Clock Standard Interface (Clock_Std_IF) 7-19
7.5.1.2 FIREBIRD™/PMC Dual WSDP™ Standard Interface (WSDP_Std_IF) 7-23
7.5.2 FIREBIRD™/PMC Dual WSDP™ BPE Model 7-24

7.5.2.1 FIREBIRD™/PMC – BPE Pad Definitions and Locations 7-24
7.5.2.2 Clock Standard Interface (Clock_Std_IF) 7-25
7.5.2.3 WSDP™ Standard Interface (WSDP_Std_IF) 7-30
7.5.2.4 IO_Conn Standard Interface (IO_Conn_Std_IF) 7-31
7.5.2.5 LED Standard Interface (LED_Std_IF) 7-33
7.6 VHDL Models Specific to FIREBIRD™/PMC Dual G-Link Only 7-34
7.6.1 Clock Standard Interface (Clock_Std_IF) 7-34
7.6.1.1 Dual G-Link Standard Interface (Glink_RX_Std_IF,Glink_TX_Std_IF) 7-38
7.6.1.2 Dual G-Link PLD Standard Interface (GLink_PLD_Std_If) 7-40
7.6.1.3 Dual G-Link TX Standard FIFO Interface (GLink_TX_STD_FIFO_IF) 7-41
7.6.1.4 Dual G-Link TX Multiplexing FIFO Interface 7-41
7.6.1.5 Dual G-Link RX Multiplexing FIFO Interface (GLink_RX_Mux_FIFO_IF) 7-43
7.6.1.6 Dual G-Link RX Standard FIFO Interface (GLink_RX_STD_FIFO_IF) 7-44
7.6.1.7 Dual G-Link PLD LAD Multiplexing Interface (LAD_Mux_GLink_PLD_IF) .. 7-45
7.7 FIREBIRD™ Family Mux Library 7-48
7.7.1 Standard PE Templates 7-48
7.7.2 LAD64_Mux Library 7-49
7.7.2.1 LAD64_Mux_IF 7-50
7.7.2.2 LAD64_Mux_Reset 7-51
7.7.2.3 LAD64_Mux_RegFile 7-52
7.7.2.4 LAD64_Mux_CRegFile 7-53
7.7.2.5 LAD64_Mux_BlockRAM 7-53
7.7.2.6 LAD64_Mux_BlockRAM64 7-54
7.7.2.7 LAD64_Mux_Arb 7-54
7.7.2.8 Assigning the BASE to the LAD64 Mux Components 7-55
7.7.3 Mem_Mux Library 7-57
7.7.3.1 Memory Arbitration Schemes 7-58
7.7.3.2 Mem32_Mux Control 7-60
7.7.3.3 Mem64_Mux Control 7-62
7.7.4 Programmed I/O Memory Bridges 7-64
7.7.4.1 LAD64_Mem32_Bridge 7-65
7.7.4.2 LAD64_Mem64_Bridge 7-65
7.7.4.3 Assigning the BASE to the LAD Memory Bridges 7-66
7.7.5 DMA_Mux Library 7-68
7.7.5.1 LAD64_DMA_Write_Mux_IF 7-69
7.7.5.2 LAD64_DMA_Read_Mux_IF 7-70
7.7.6 DMA Memory Bridges 7-72
7.7.6.1 DMA Write Bridges 7-73
7.7.6.2 DMA Read Bridges 7-76
7.7.6.3 Assigning the BASE to the LAD Memory Bridges 7-79
7.8 Host Model 7-80
7.8.1 Simulated Host API Procedures 7-80
7.8.2 FIREBIRD™/PMC System Configuration (System_Config) 7-97
7.9 Introduction to FIREBIRD™ VHDL 7-98
7.9.1 Simulation Environment 7-98
7.9.2 Using Templates 7-98
7.9.3 Using VHDL Record Types 7-98
7.9.4 Using Processing Element Standard Interfaces 7-98
7.10 FIREBIRD™ VHDL Model 7-100
7.10.1 PE0 Model 7-100
7.10.1.1 PE0 Register Space 7-100
7.10.1.2 PE0 Memory Space 7-100
7.10.1.3 LAD Bus Transactions 7-100
7.10.1.4 On-board Memory timing 7-103

7.10.2 Host Model 7-105
7.10.3 PCI Controller Model 7-106
7.10.4 On-board Memory Model 7-106
7.10.5 I/O Card Model 7-106
7.10.6 VHDL Model Configuration 7-106

8 FIREBIRD™ VHDL DESIGN CYCLE 8-1

8.1 FIREBIRD™/PCI and /PMC 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 Modifying Configuration Files 8-2
8.1.5 Simulating a Design 8-2
8.1.5.1 Using Template ModelSim™ Macro Files 8-2
8.1.5.2 Setting Up Memory Initialization Files 8-2
8.1.5.3 Compiling a VHDL Design 8-3
8.1.5.4 Simulating a VHDL Design 8-3
8.2 Design Synthesis 8-4
8.2.1 Using Template Synplify® Project Files 8-4
8.2.2 Setting Up Synthesis Constraints 8-4
8.2.3 Synthesizing the Design 8-4
8.2.4 Place-and-Routing a Design 8-5
8.2.5 Setting Environment Variables 8-5
8.2.6 Using Template Makefiles 8-6
8.2.7 Setting Up Timing Constraints 8-6
8.2.8 Running the Xilinx® Place-and-Route Makefile 8-7
8.2.9 Analyzing Design Performance 8-7
8.2.10 Transferring a PE design 8-7

ATTACHMENT A: PE PAD LOCATIONS AND DEFINITIONS A1

ATTACHMENT B: BPE PAD LOCATIONS AND DEFINITIONS B1

ATTACHMENT C: PE0 GENERIC QUICK REFERENCE GUIDE C1

ATTACHMENT D: BPE GENERIC QUICK REFERENCE GUIDE D1

ATTACHMENT E: EURO I/O DAUGHTER CARD REFERENCE GUIDE FOR FIREBIRD™/PCI E1

ATTACHMENT F: FIREBIRD™/PMC DUAL WSDP™: USER VHDL MAPPING TO MICTOR PINS F1

1 ABOUT THIS MANUAL

1.1 Introduction

FIREBIRD™ computing, a revolutionary technology developed by Annapolis Micro Systems, Inc., combines a high number of reconfigurable system gates with exceptional memory and I/O bandwidth. All FIREBIRD™ boards feature a single processing element with five independent memory ports. The FIREBIRD™ architecture supports PCI, cPCI, and PMC bus formats.



INFORMATION NOTE
In this document, **"FIREBIRD™"** stands for FIREBIRD™/PCI (all board revisions), FIREBIRD™/PMC Dual WSDP™, and FIREBIRD™/PMC Dual G-Link, unless otherwise noted.

1.2 How This Manual Is Organized

The FIREBIRD™ Reference Manual describes how to install, maintain, and program each version of the FIREBIRD™ reconfigurable computing board. It is organized into eight chapters designed for easy reference.



INFORMATION NOTE
The FIREBIRD™ motherboard operates with WILDSTAR™ Host Software. Detailed WILDSTAR™ software information can be found in the *WILDSTAR™ Software Reference Manual* included in the FIREBIRD™ package.

- Chapter 1, "**About This Manual**," outlines the conventions, icons, and key words used throughout the manual.
- Chapter 2, "**Introduction to FIREBIRD™**," discusses board architecture and performance features.
- Chapter 3, "**Getting Started**," describes how to properly unpack and visually inspect the FIREBIRD™ board.
- Chapter 4, "**Installing FIREBIRD™**," explains system requirements and hardware installation.

- Chapter 5, "**Technical Support**," provides information for contacting the Annapolis Micro Systems, Inc. Technical Support team.

- Chapter 6, "**Hardware Reference**," includes general hardware specifications, switching characteristics, DC characteristics, and external interface options.

- Chapter 7, "**VHDL Models Reference**," elaborates on the VHDL design cycle, describes the simulation environment, and details how to use the FIREBIRD™ VHDL templates, record types, and Processing Element (PE) standard interfaces. Each phase of the VHDL design cycle is discussed, beginning with creation, and going through simulation, synthesis, place and route, and finally analysis of the design's performance.

1.3 Attachments

The following attachments are included with this manual.

- **Attachment A** contains PE pad locations and definitions.
- **Attachment B** contains BPE pad locations and definitions.
- **Attachment C** contains the *PE0 Generic Quick Reference Guide*.
- **Attachment D** contains the *BPE Generic Quick Reference Guide*.
- **Attachment E** contains the *Euro I/O Daughter Card Reference Guide for FIREBIRD™/PCI.*
- **Attachment F contains** the *FIREBIRD™/PMC Dual WSDP™ User VHDL Mapping for MICTOR® Pins*.

1.4 Conventions

A variety of 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].

Convention	Description
	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.

1.5 Icons

Throughout the manual, important information is highlighted with icons.

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

1.6 Key Words and Definitions

Some of the terms used throughout the manual are defined below.

API

Application Programming Interface. A set of functions coded in the C language allowing communication between an application and the WILDSTAR™ Board.

Block RAM

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

CLB

Configurable Logic Block. CLBs provide the functional elements for constructing logic within the PE. Each Virtex™ CLB contains 4 Logic Cells, organized in 2 similar Slices.

DMA

Direct Memory Access.

Driver

Software to handle communication with FIREBIRD™ boards.

Euro I/O Card

Eurocard form factor I/O daughter card for FIREBIRD™/PCI and FIREBIRD™ cPCI boards.

External I/O

External Input/Output.

External I/O Cards

External Input/Output cards for WILDSTAR™/VME and PCI. Include both Euro I/O cards and PCI I/O cards.

FIREBIRD™ VHDL Models

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

FPGA

Field Programmable Gate Array.

KCLK

System bus clock used for Local Address Data Bus transactions.

LAD Bus

Local Address Data Bus. Bus between the PCI Controller, PE0, and the External I/O card.

MCLK

Reprogrammable Memory clock.

N/C

No Connection.

PCI

Peripheral Component Interconnect.

PCLK

Reprogrammable Processing Element Clock synchronous to MCLK.

PE0

Processing Element 0. PE0 is used for I/O and data processing in the WILDSTAR™ architecture.

PMC

PCI Mezzanine Card

Processing Element

A Xilinx® Virtex™ Field Programmable Gate Array (FPGA), comprises the basic processing unit on the WILDSTAR™ board.

Slice

A slice contains two look-up tables and two flip-flops for implementing logic within a CLB. There are two slices per CLB.

UCLK

Configurable User clock for use in PEs and external I/O cards.

Virtex™ FPGA

Virtex™ Field Programmable Gate Array.

Virtex™-E FPGA

High density FPGA offering higher performance, larger capacity, and higher speed than the Virtex™ FPGA.

VME

Versa Module Eurocard.

WILDSTAR™ Development Toolkit

Model Technology, Inc. ModelSim, Synplicity® Synplify®, Xilinx® Alliance Series.

WILDSTAR™ Host Software

Application Programming Interface, Device Driver, and utilities.

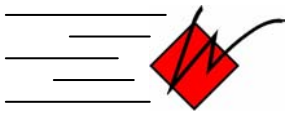
WILDSTAR™ VHDL Models

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

ZBT

Zero Bus Turnaround.


TM

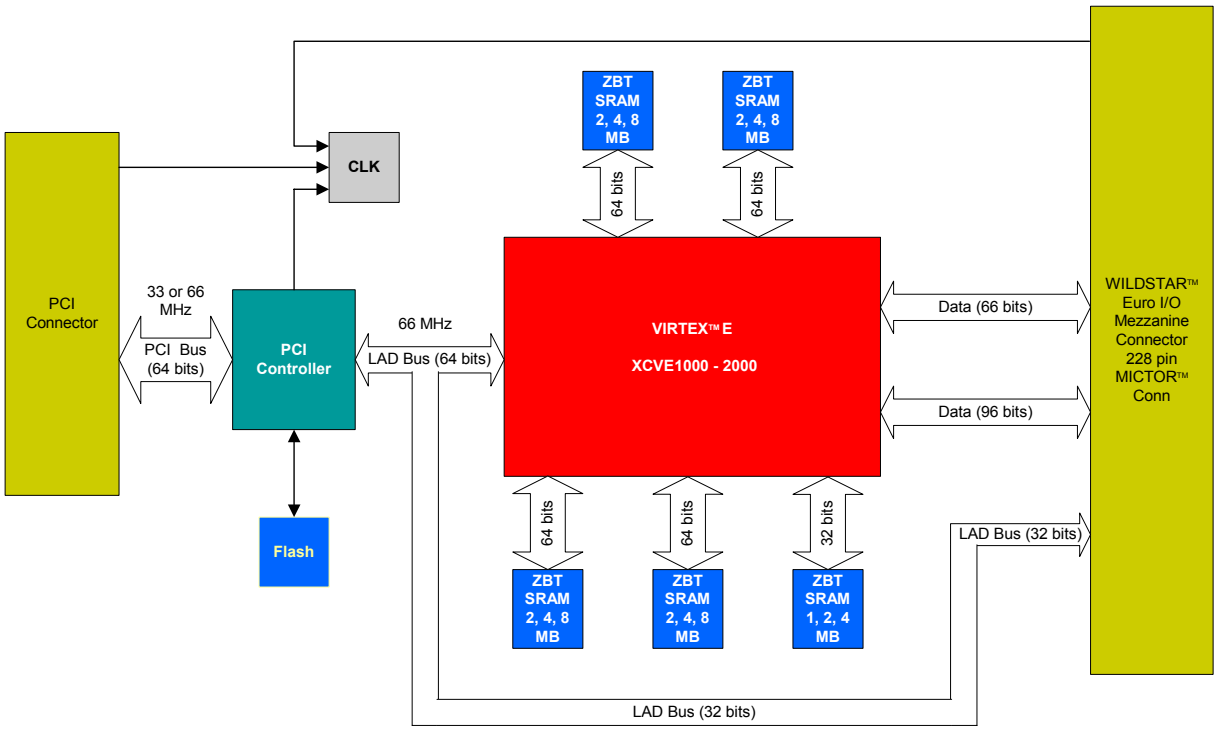


2 INTRODUCTION TO FIREBIRD™

The FIREBIRD™ architecture supports three form factors—FIREBIRD™/PCI, FIREBIRD™/cPCI, and FIREBIRD™/PMC. The features of the FIREBIRD™ PCI and PMC boards are highlighted below and discussed in more detail throughout this manual.

2.1 FIREBIRD™/PCI Features

- Uses Xilinx® Virtex™-E FPGAs XCV1000E to XCV2000E for PE0
- Processing clocks up to 150MHz
- Five independent memory banks, containing nine to 36 Mbytes of synchronous ZBT SRAM
- 5.4Gbytes/sec of memory bandwidth
- Support for power-aware applications
- Accepts a single WILDSTAR™ Euro I/O card
- Up to 3Gbytes/sec of I/O bandwidth to the WILDSTAR™ Euro I/O card
- 66MHz/64bit PCI transactions (3.3V PCI signaling only)
- PE0 can optionally be programmed from flash on power up



Copyright 2000-2001 Annapolis MIcro Systems, Inc.

Figure 2–1: FIREBIRD™/PCI Block Diagram

2.2 Features Common to PMC Dual WSDP™ and PMC Dual G-Link

FIREBIRD™/PMC boards combine the functionality of an I/O card with the power and versatility of the FIREBIRD™ architecture. As an industry-standard board, the FIREBIRD™/PMC can be used with any PMC-compatible motherboard. The PMC board features either dual WSDP™ or dual fiber optic G-Link ports.

Other features include the following:

- Uses Xilinx® Virtex™-E FPGAs XCV1000E to XCV2000E for PE0
- Processing clocks up to 150 MHz
- MCLK and UCLK can be optionally programmed on power up.
- Five independent memory banks, containing nine or 18 Mbytes of synchronous ZBT SRAM
- 5.4 GBytes/sec of memory bandwidth

- Factory-configurable P14 I/O options:
  - Single-channel RACE++/RACEway support available
  - Differential I/O (31 LVDS or 32 LVPECL)
  - +5V tolerant I/O (series terminated)
  - Custom (combination of above)
- 66 MHz/64 bit PCI transactions (3V PCI signaling only)

Features Specific to Dual WSDP™

- WSDP™ uses Xilinx® Virtex™-E FPGAs XCV200E to XCV300E for BPE0
- PE0 and BPE can optionally be programmed on power up
- Two front panel WSDP™ ports supporting various I/O standards, including LVDS
- Front panel LEDs
- WSDP™ ports controlled directly by BPE
- Can synchronize MCLK or UCLK to either WSDP™ clock

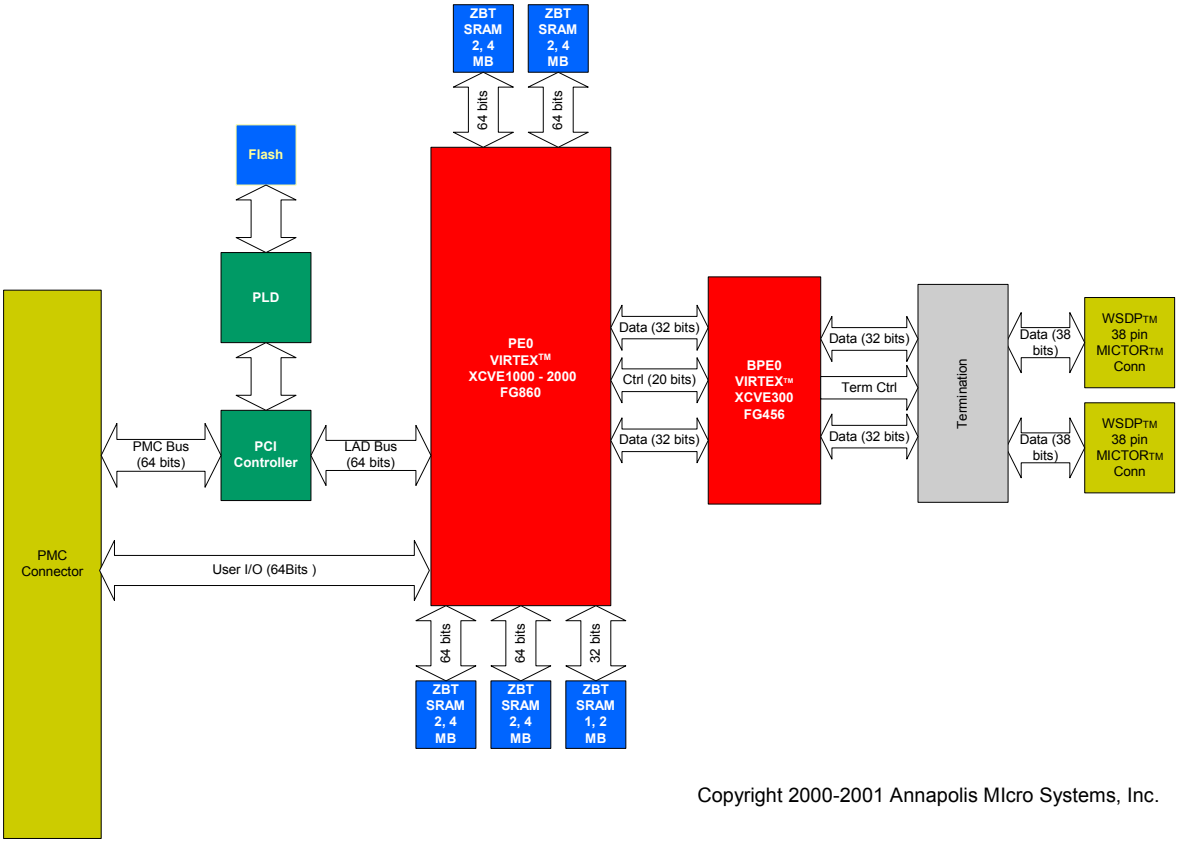


Figure 2-2: FIREBIRD™/PMC Dual WSDP™ Block Diagram

Features Specific to Dual G-Link

- Motherboard with I/O card(s) installed occupies only a single VME or PCI slot.
- PE0 can optionally be programmed on power up
- Rates of transmitting and receiving data up to a combined 1.1 GBytes/sec over two full-duplex fiber optic channels per card.
- Ability to lock PMC Dual G-Link and motherboard to external single-ended or differential clock source.
- Ability to lock board to recovered clock-off channel.
- Supports multiple Agilest® small form factor fiber optic transceivers.

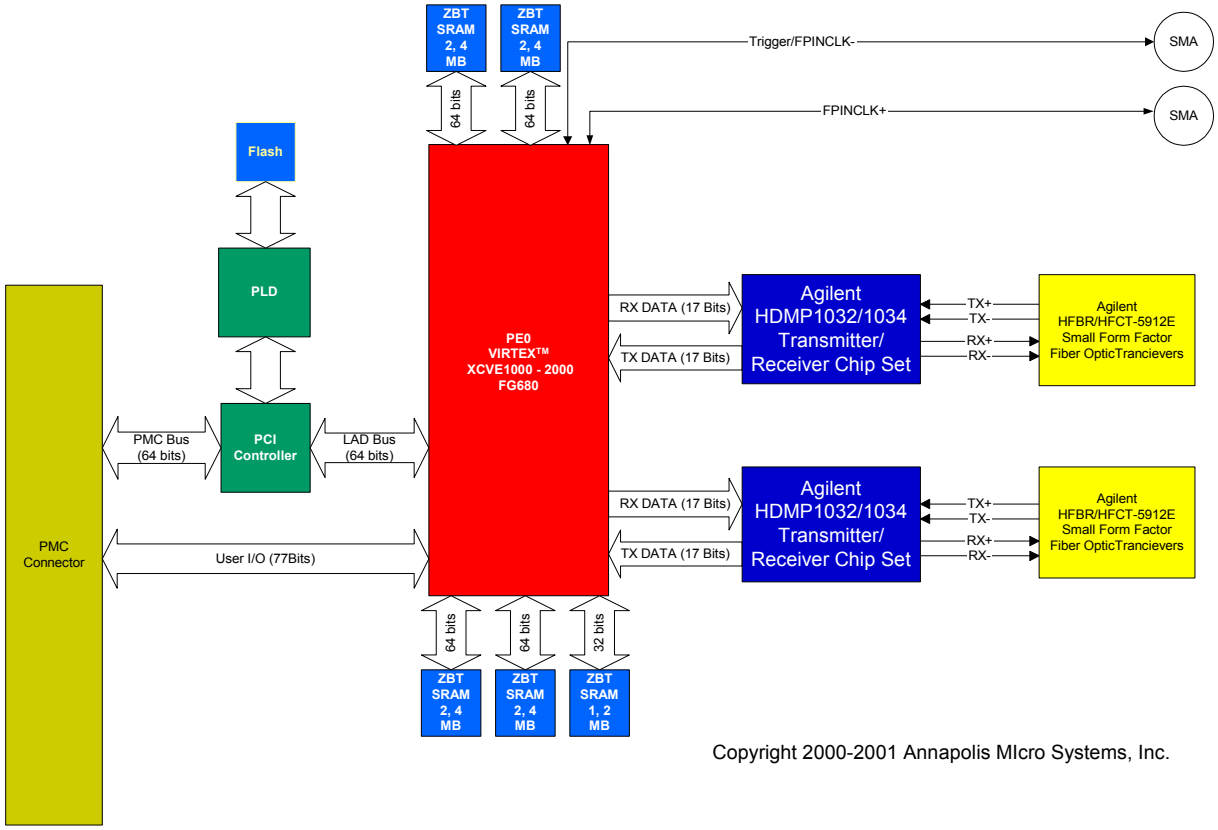
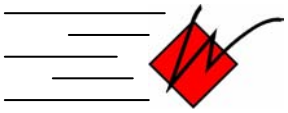


Figure 2-3: FIREBIRD™/PMC Dual G-Link Block Diagram



3 GETTING STARTED

The FIREBIRD™ board uses WILDSTAR™ host software, which is included with the board. Details for software installation can be found in the FIREBIRD™ Release Notes, also included.

3.1 Unpacking

The FIREBIRD™ board is shipped in a static-sensitive pack.



CAUTION

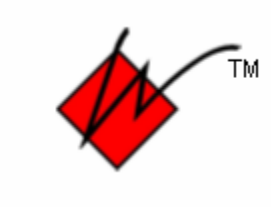
Before removing the board from the static pack, be sure to be grounded of all static electricity. Static discharge to the FIREBIRD™ board could damage sensitive components.

3.2 Visual Inspection

The FIREBIRD™ package includes:

- FIREBIRD™ board
- *FIREBIRD™ Reference Manual*
- *WILDSTAR™ Software Reference Manual*
- Release Notes
- Software license agreement
- Installation Hardware
- WILDSTAR™ Host Software CD-ROM for FIREBIRD™, containing:
  - WILDSTAR™ API
  - WILDSTAR™ Device Drivers
  - Application Examples
  - FIREBIRD™ VHDL Models
  - PDFs (portable document files) of manuals

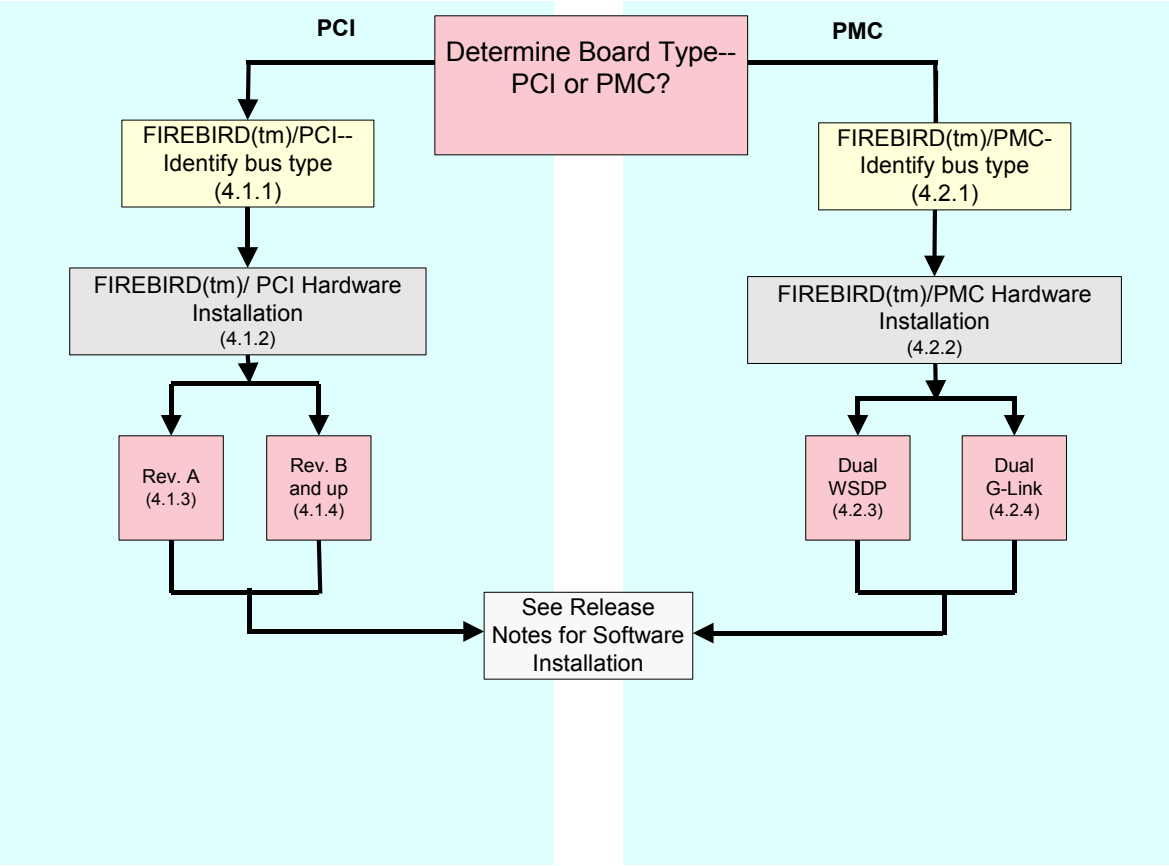
Before installing the FIREBIRD™ board in the computer, visually inspect it for damage that may have occurred during shipping. If you note any damage, contact Annapolis Micro Systems, Inc. (See Chapter Five for contact information.)


TM

4 FIREBIRD™ HARDWARE INSTALLATION

This section explains the procedures for installing FIREBIRD™ boards and hardware. Directions for installation are organized according to board type. For installation of software for FIREBIRD™, please consult the Release Notes included with your board.

The diagram below will help you locate the instructions needed for installing your particular FIREBIRD™ board type.



4.1 Installing FIREBIRD™/PCI

4.1.1 3.3V and 5V PCI Signaling

The FIREBIRD™/PCI motherboard can be manufactured to adhere to 3.3V or 5V PCI bus signaling capability. If manufactured as 3.3V PCI bus signaling, the motherboard will **only** fit into a 3.3V PCI slot (see Figure 4-1). If manufactured as 5V PCI bus signaling, the motherboard will fit into a 3.3V or a 5V PCI slot (see Figure 4-2).

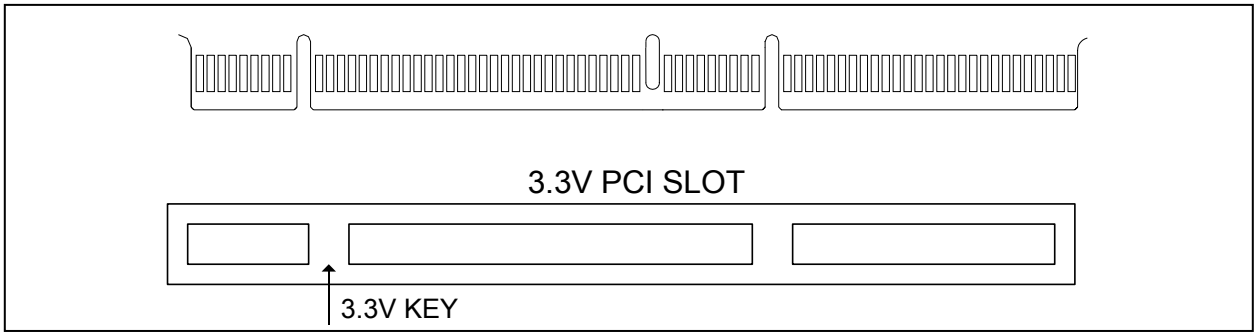


Figure 4–1: 3.3V PCI Slot

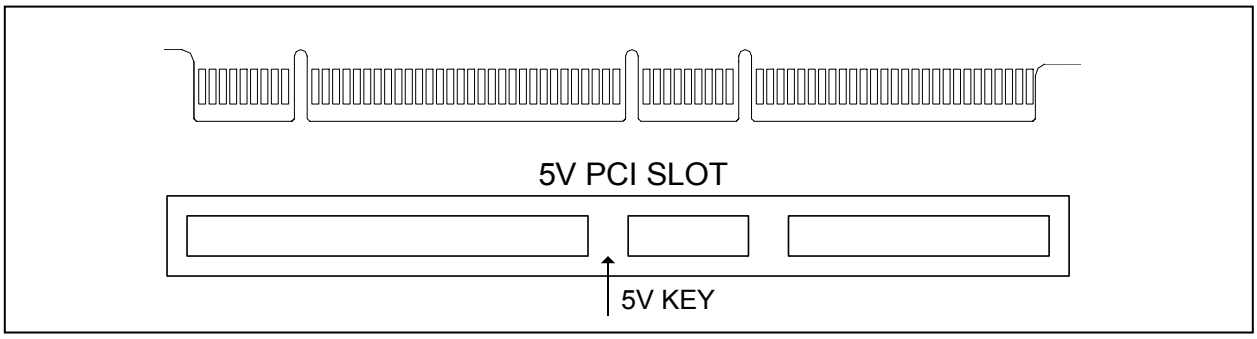


Figure 4–2: 5V PCI Slot

INFORMATION NOTE

During power up, the PCI Controller will automatically be loaded on the FIREBIRD™ 3.3V and FIREBIRD™ 5V motherboards.

4.1.2 FIREBIRD™/PCI Hardware Installation (All Board Revisions)

Follow the steps below to install the motherboard into the host system.

1. Connect the ground strap to the user.
2. Shut down the host system and power off.
3. Remove the cover to the host chassis in order to gain access to the motherboard's PCI connectors.
4. For the available slot, remove the rear access panel.
5. Insert the FIREBIRD™/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 FIREBIRD™ board.
8. Replace the cover to the host system chassis.

INFORMATION NOTE

The front mounting bracket (see Figure 4-3) must be secured to the front of the slot to ensure the FIREBIRD™/PCI board does not warp.

INFORMATION NOTE

The FIREBIRD™ board uses 64-bit PCI edge finger configuration. When the FIREBIRD™/PCI board is placed in a 32-bit PCI slot, the 64 bit edge fingers 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.

4.1.3 FIREBIRD™/PCI Rev. A Board

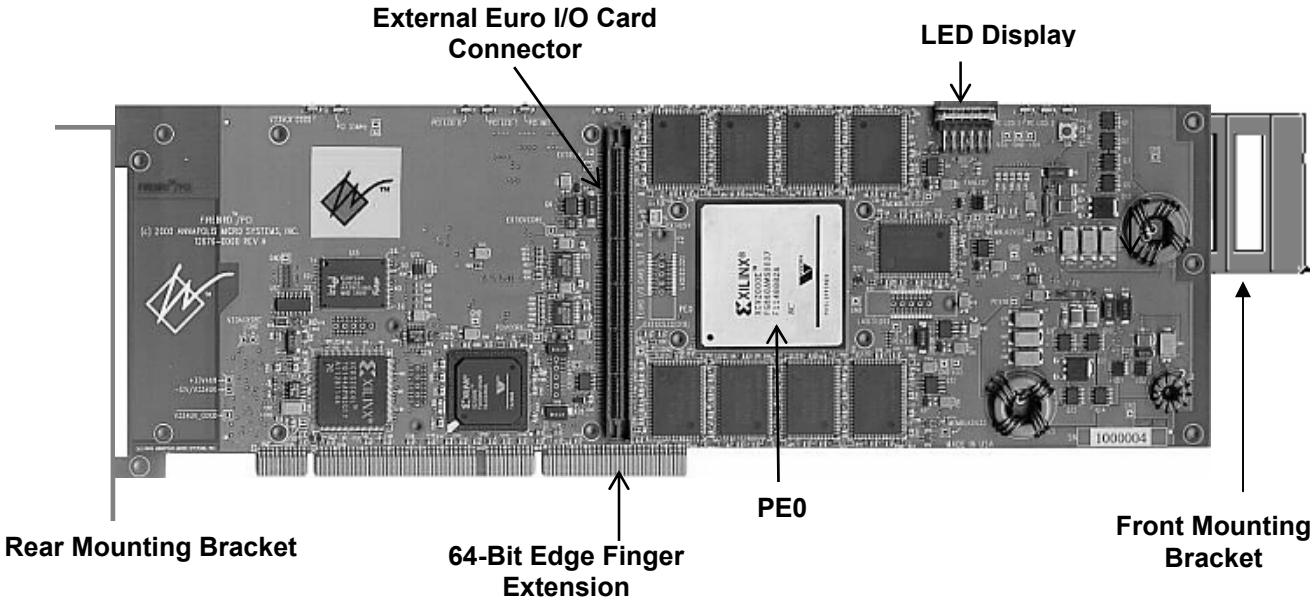


Figure 4–3: FIREBIRD™/PCI Rev. A Motherboard - Component Side

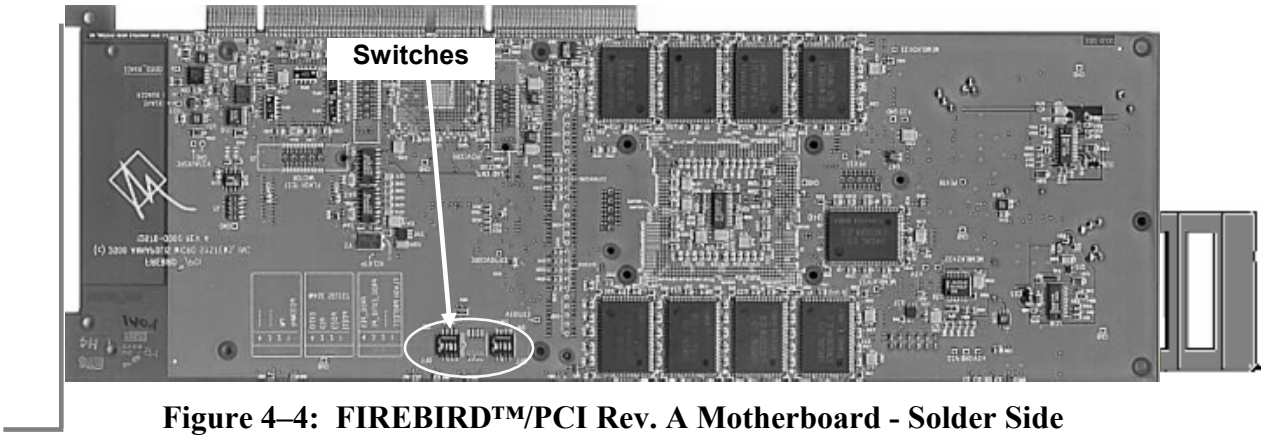


Figure 4–4: FIREBIRD™/PCI Rev. A Motherboard - Solder Side

4.1.3.1 FIREBIRD™/PCI Rev. A Switch Settings

The FIREBIRD™/PCI Rev A board is delivered with all of its switches in their default settings. The switch settings are defined in the following tables. Refer to Figure 4-4 for the exact location of the switches. Figure 4-5 shows an enlarged view of the two switches and their default settings.

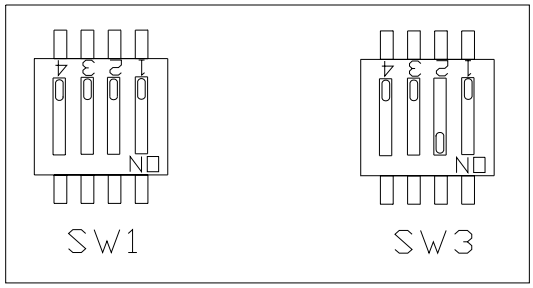


Figure 4–5: FIREBIRD™/PCI Rev A Default Switch Settings

Table 4-1: FIREBIRD™/PCI Rev A SW1 Options

Position	Name	Description	Default Setting
1	PCI33MHz	ON = Restricts PCI clock, and therefore KCLK, to 33MHz max. On Universal Boards, this switch has no effect. All FIREBIRD™/PCI boards configured with 5V signaling run up to 33MHz max. OFF = PCI clock permitted to run up to 66MHz.	OFF
2	Reserved.	Reserved.	OFF
3	Reserved.	Reserved.	OFF
4	Reserved.	Reserved.	OFF

Table 4-2: FIREBIRD™/PCI Rev A SW3 Options

Position	Name	Description	Default Setting
1	Reserved.	Reserved.	OFF
2	Reserved.	Reserved.	ON
3	RECOVER	ON = Recovers original PCI controller version	OFF
4	-	Reserved.	OFF

INFORMATION NOTE

When upgrading the PCI controller on the FIREBIRD™/PCI Rev. A board, set Switch 3, position 1, to "on." Doing so will disable the flash protect. Once the PCI controller has been updated, be sure to set the switch back to the "off" position to avoid accidentally overwriting the flash.

4.1.3.2 FIREBIRD™/PCI Rev. A LED Definitions

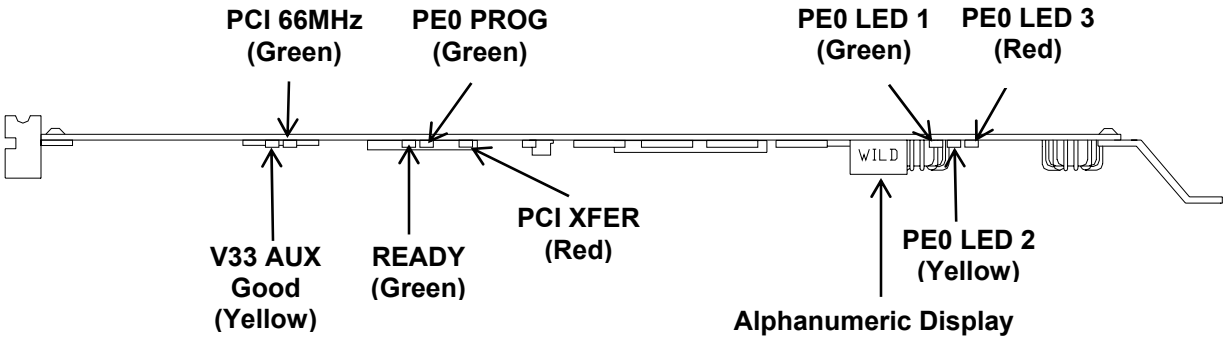


Figure 4–6: FIREBIRD™/PCI Rev. A LED Locations and Names

INFORMATION NOTE

The alphanumeric display (shown in Figure 4-6) is user configurable using the API call "WS_UpdateDisplay."

Table 4-3: FIREBIRD™/PCI Rev. A LED Definitions

LED	LED Color	LED Default POWER ON State	Led Definition
V33 AUX Good	Yellow	System Dependent	Auxiliary 3.3V is present
PCI 66MHz	Green	System Dependent	PCI/LAD Bus is capable of 66Mhz operation
READY	Green	ON	PCI Controller is programmed—board ready
PE0 PROG	Green	OFF	PE0 is programmed
PCI XFER	Red	OFF	FIREBIRD™/PCI bus transfer in progress
PE0 LED 1	Green	OFF	User defined LED 1
PE0 LED 2	Yellow	OFF	User defined LED 2
PE0 LED 3	Red	OFF	User defined LED 3

4.1.4 FIREBIRD™/PCI Board (Revs. B, C, D)

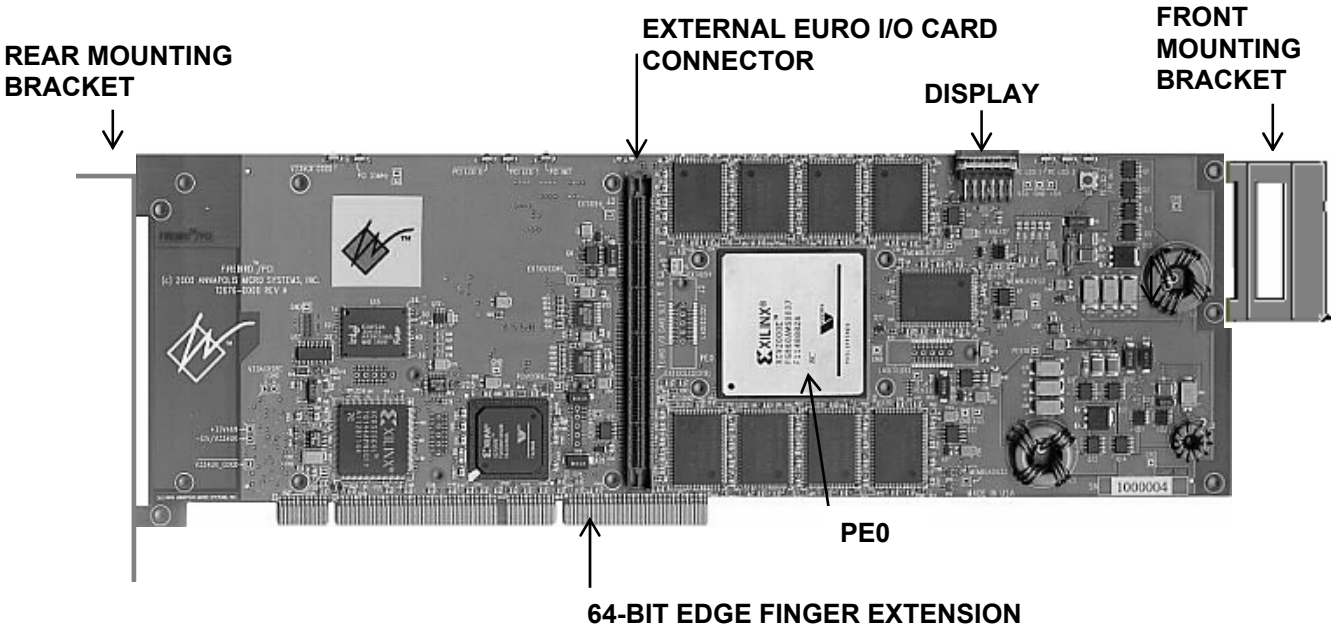


Figure 4–7: FIREBIRD™/PCI (Rev. B, C, D) Motherboard - Component Side

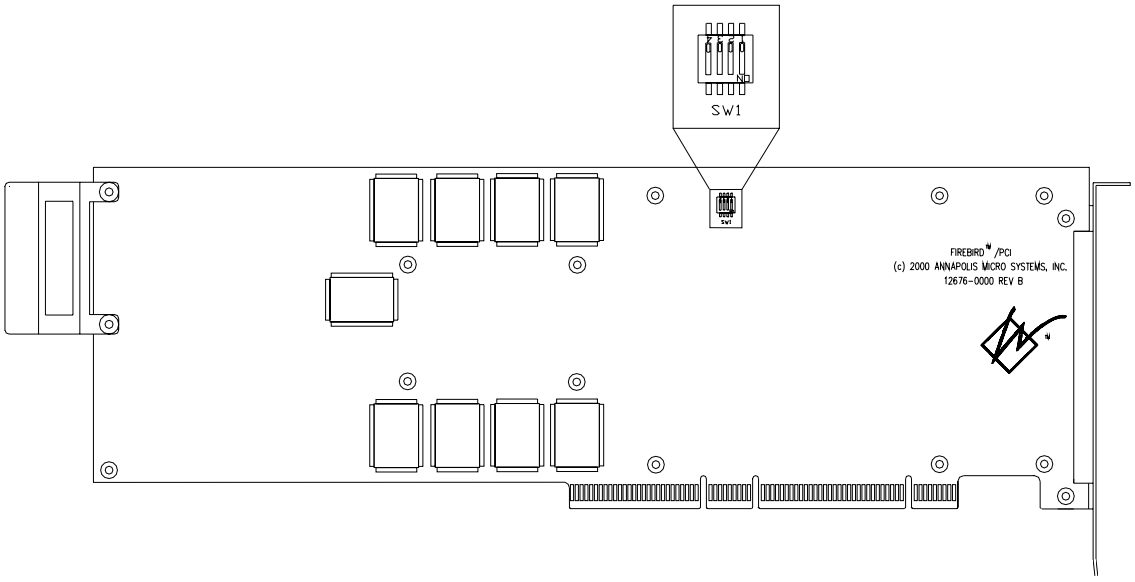


Figure 4–8: FIREBIRD™/PCI Motherboard (Rev. B, C, D) - Solder Side

4.1.4.1 FIREBIRD™/PCI Rev. B, C, D-- Switch Settings

The FIREBIRD™/PCI boards (Rev. B, C, D) are delivered with all of its switches in their default settings. The switch settings are defined in the following tables. Refer to Figure 4-8 for the exact location of the switches. Figure 4-9 shows an enlarged view of the switch and its default settings

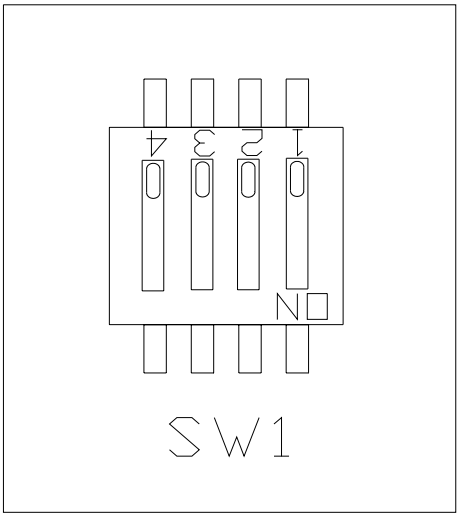


Figure 4–9: FIREBIRD™/PCI (Rev B, C, D) Default Switch Settings

Table 4-4: FIREBIRD™/PCI (Rev. B, C, D) SW1 Options

Position	Name	Description	Default Setting
4	PCI33MHz	ON = Restricts PCI clock, and therefore KCLK, to 33MHz max. On Universal Boards, this switch has no effect. All FIREBIRD™ boards configured with 5V signaling run up to 33MHz max. OFF = PCI clock permitted to run up to 66MHz.	OFF
3	Backup PCI	ON = Program PCI with "backup" image. OFF = Program PCI with image determined by FLASH register.	OFF
2	Reserved	Reserved	OFF
1	Reserved	Reserved	OFF

4.1.4.2 FIREBIRD™/PCI Rev. B , C, D-- LED Definitions

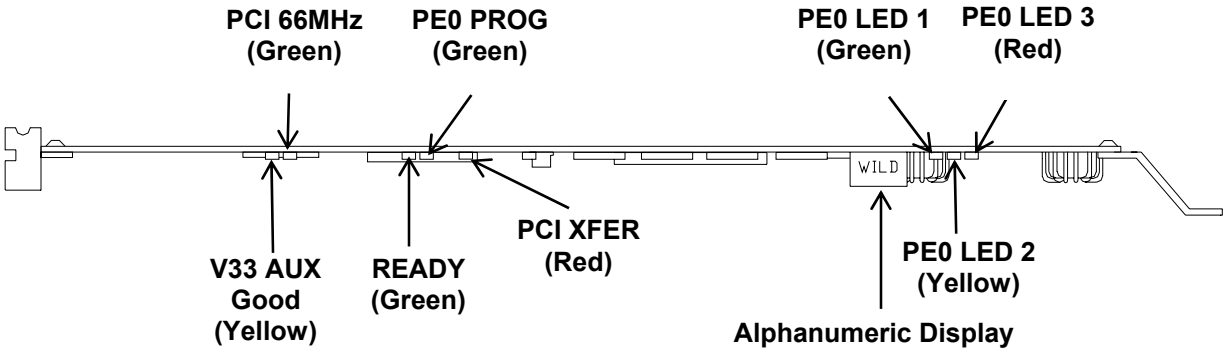


Figure 4–10: FIREBIRD™/PCI (Rev. B, C, D) LED Locations and Names

INFORMATION NOTE

The alphanumeric display (shown in Figure 4-10) is user configurable using the API call "WS_UpdateDisplay."

Table 4-5: FIREBIRD™/PCI Rev. B, C, D-- LED Definitions

LED	LED Color	LED Default POWER ON State	Led Definition
V33 AUX Good	Yellow	System Dependent	Auxiliary 3.3V is present
PCI 66MHz	Green	System Dependent	PCI/LAD Bus is capable of 66Mhz operation
READY	Green	ON	PCI Controller is programmed—board ready
PE0 PROG	Green	OFF	PE0 is programmed
PCI XFER	Red	OFF	FIREBIRD™/PCI bus transfer in progress
PE0 LED 1	Green	OFF	User defined LED 1
PE0 LED 2	Yellow	OFF	User defined LED 2
PE0 LED 3	Red	OFF	User defined LED 3

4.2 Installing FIREBIRD™/PMC

4.2.1 3.3V and 5V PCI Signaling (Dual WSDP™, Dual G-Link)

The FIREBIRD™ /PMC board is engineered for 3.3 volts and 5 volts. In the illustration below, note the different locations of the 3.3 and 5 volt keys on the component side of the board.

If manufactured as 5 volt (5V) bus signaling (with the 5V keyhole open), the motherboard will fit into *either* a 3.3V or a 5V PCI slot. Boards having 3.3 volt (3.3V) bus signaling (with the 5V keyhole closed) will *only* work in a 3.3V PMC slot.

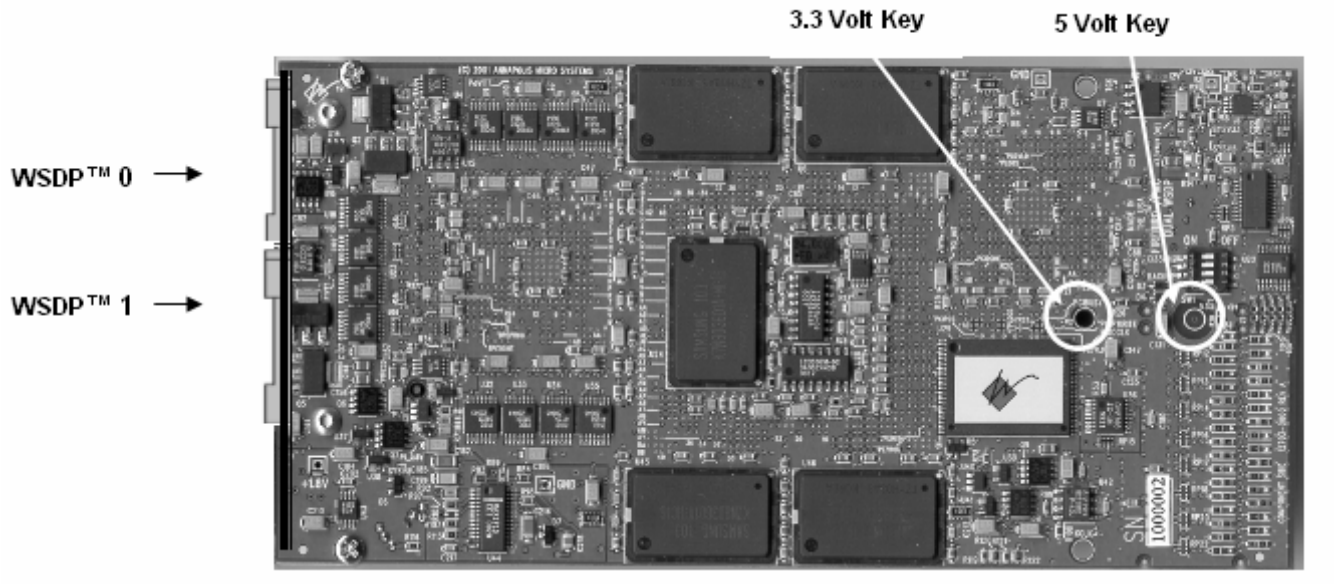


Figure 4-11: Voltage Key Locations for FIREBIRD™/PMC (both WSDP™ and G-Link)

4.2.2 FIREBIRD™/PMC Dual WSDP™ Hardware Installation

Follow the steps below to install the motherboard into the host system.

1. Connect the ground strap to the user.
2. Shut down the host system and power off.
3. Remove motherboard from host chassis and lay face up on an electrostatic-protected workstation.

4. Lift the FIREBIRD™/PMC Dual WSDP™ board by its sides, with the metal front plate and PMC connectors pointing down. Ensure that the rubber gasket surrounding the front plate is securely in place.
5. Align the PMC Dual WSDP™ board connectors with the receptacles on the motherboard and press together snugly.
6. Align the two standoffs with the corresponding holes between the motherboard and the PMC board. (These standoffs will be secured with screws.)
7. With the screws provided, fasten the PMC board to motherboard assembly through the component side of the PMC board. Turn the assembly over and fasten down the motherboard with the remaining screws.
8. Connect all cables as required.

INFORMATION NOTE

The front panel is secured by hex screws and clear Mylar spacers. If the front panel is removed from the board, be sure to save the screws and spacers for later re-installation.

4.2.2.1 Cable Requirements and Installation for PMC Dual WSDP™

FIREBIRD™/PMC Dual WSDP™ boards require a WSDP™ cable for each port (Figure 4-12, sold separately). Each cable is equipped with 32 data lines, four control lines, and two clock lines. Depending on I/O throughput needs, one or both WSDP™ ports can be used to pass data across the cables.

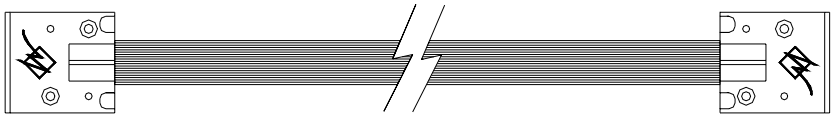


Figure 4-12: WSDP™ Cable

4.2.3 FIREBIRD™/PMC Dual WSDP™ Board

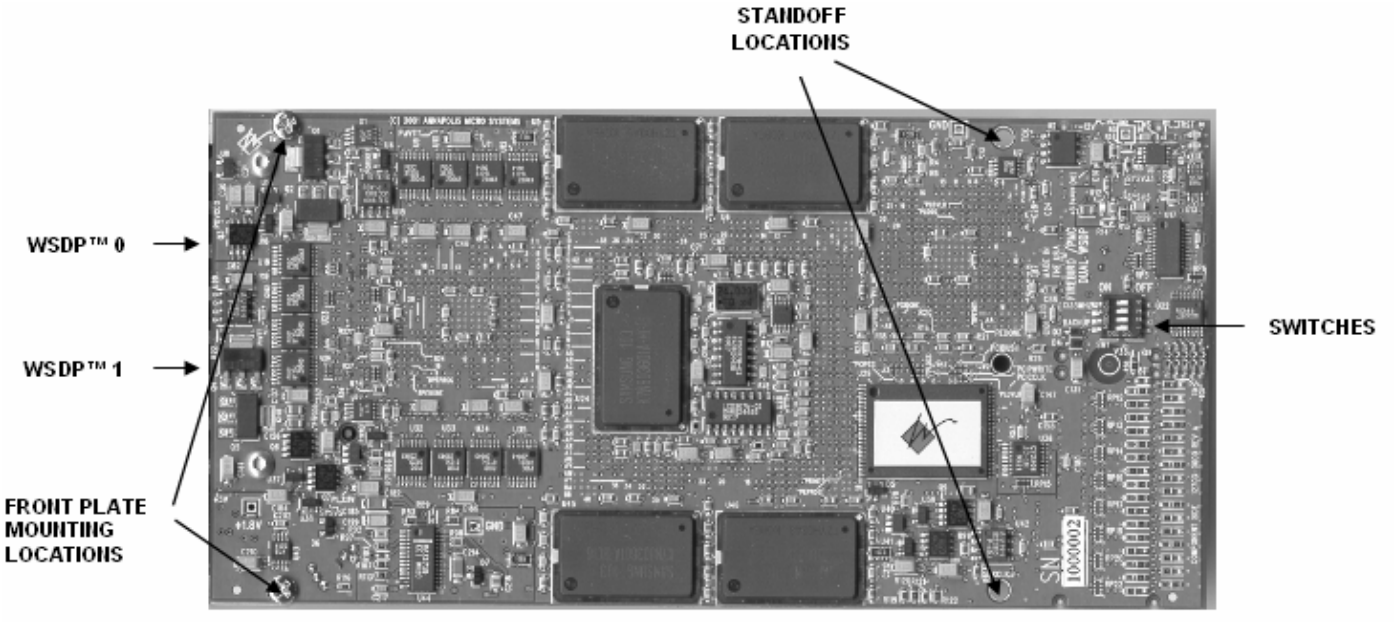


Figure 4–13: FIREBIRD™/PMC Dual WSDP™ - Component Side

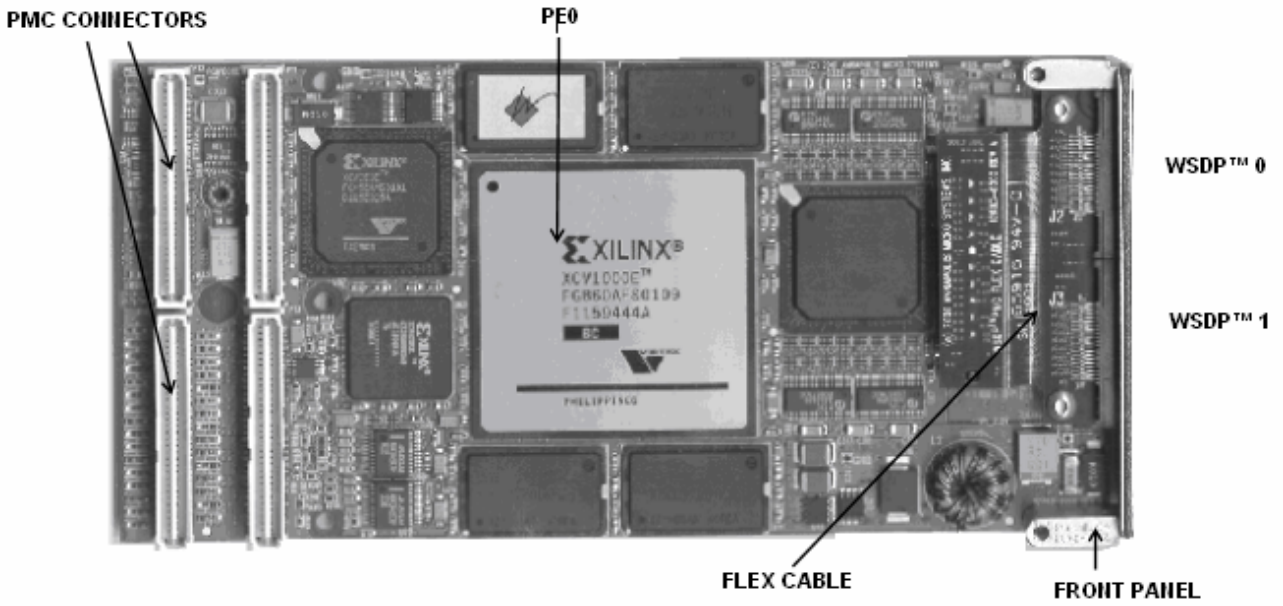


Figure 4-14: FIREBIRD™/PMC Dual WSDP™ -Solder Side

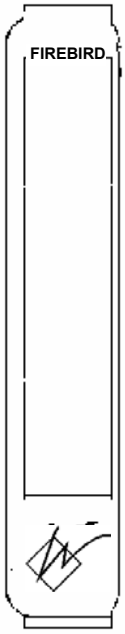


Figure 4–15: FIREBIRD™/PMC Dual WSDP™ - Front Panel

4.2.3.1 FIREBIRD™/PMC Switch Settings (Dual WSDP™, Dual G-Link)

FIREBIRD™/PMC boards are delivered with all switches in their default settings. The switch settings are defined in the following tables. Refer to Figure 4-13 for the location of the switches. Figure 4-16 shows an enlarged view of the switch and its default settings

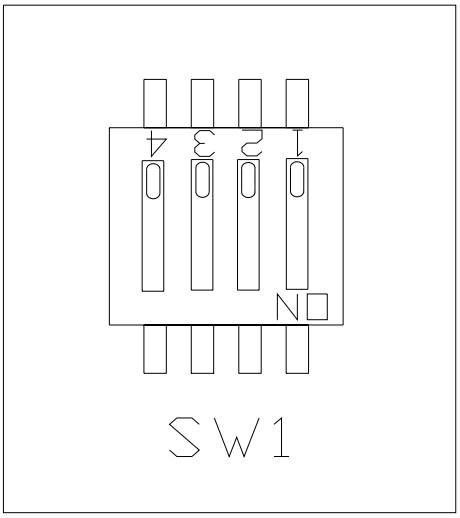


Figure 4–16: FIREBIRD™/PMC Default Switch Settings

Table 4-6: FIREBIRD™/PMC Options

Position	Name	Description	Default Setting
1	PCI33MHz	ON = Restricts PCI clock, and therefore KCLK, to 33MHz max. On Universal Boards, this switch has no effect. All FIREBIRD™ boards configured with 5V signaling run up to 33MHz max. OFF = PCI clock permitted to run up to 66MHz.	OFF
2	Backup PCI	ON = Program PCI with "backup" image. OFF = Program PCI with image determined by FLASH register.	OFF
3	Reserved	Reserved	OFF
4	Reserved	Reserved	OFF

4.2.3.2 FIREBIRD™/PMC Dual WSDP™ LED Definitions

The following illustrations indicate the locations of the nine LEDs residing on the FIREBIRD™/PMC Dual WSDP™ board. Table 4-5 provides names and descriptions of each of the LEDs shown.

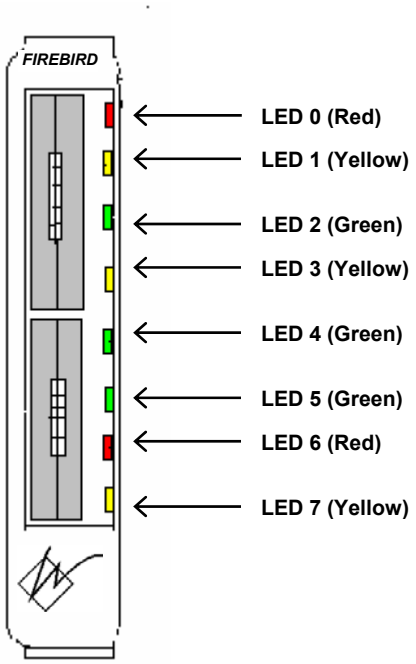


Figure 4-17: FIREBIRD™/PMC Dual WSDP™ - Front Panel LEDs

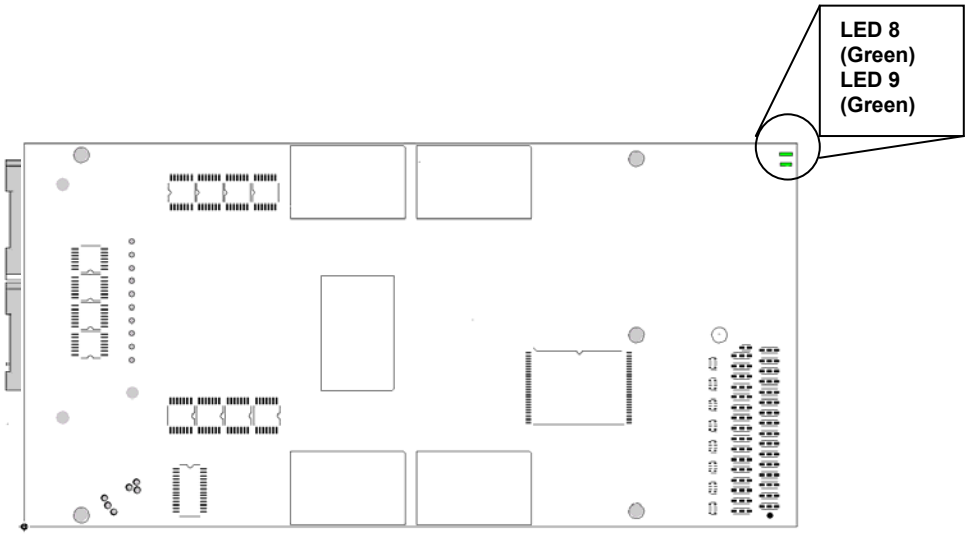


Figure 4–18: FIREBIRD™/PMC Dual WSDP™ - LEDs on Component Side

Table 4–5: FIREBIRD™/PMC Dual WSDP™ - LED Definitions

LED NUMBER	Name	Color	Description
0	User LED	Red	Connected to PE0
1	User LED	Yellow	Connected to PE0
2	User LED	Green	Connected to PE0
3	PCI Transfer	Yellow	Denotes that A PCI transfer is taking place.
4	66MHz PCI	Green	Denotes that the PMC slot is 66Mhz
5	PLD Ready	Green	PLD is ready (Flashes at startup programming)
6	WSDP Port 0 User LED	Red	Controlled by BPE0
7	WSDP Port 1 User LED	Yellow	Controlled by BPE0
8	~PCI Ready	Green	PCI controller is ready.
9	PE Done	Green	PE0 is programmed.

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

4.2.4 FIREBIRD™/PMC Dual G-Link Hardware Installation

Follow the steps below to install the motherboard into the host system.

1. Connect the ground strap to the user.
2. Shut down the host system and power off.
3. Remove motherboard from host chassis and lay face up on an electrostatic-protected workstation.
4. Lift the FIREBIRD™/PMC Dual G-Link board by its sides, with the metal front plate and PMC connectors pointing down. Ensure that the rubber gasket surrounding the front plate is securely in place.
5. Align the PMC Dual G-Link board connectors with the receptacles on the motherboard and press together snugly.
6. Align the two standoffs with the corresponding holes between the motherboard and the PMC board. (These standoffs will be secured with screws.)



INFORMATION NOTE

As with the Dual WSDP™, the Dual G-Link board front panel is secured by hex screws and clear Mylar spacers. If the front panel is removed from the board, be sure to save the screws and spacers for later re-installation.

7. With the screws provided, fasten the PMC board to the motherboard assembly with screws through the component side of the PMC board. Turn the assembly over and fasten down the motherboard with the remaining screws.
8. Connect all cables as required (see cable details below).

4.2.4.1 Cable Requirements and Installation for Dual G-Link

Two types of cables are required for proper operation of the FIREBIRD™/PMC Dual G-Link board: fiber optic cables for data, and SMA coaxial cables for clock connection.

Fiber Optic Cables

Single-mode fiber cables should be used when connecting the FIREBIRD™/PMC Dual G-Link board. These cables must have MT-RJ plugs on the end to be compatible with the board.

SMA Coaxial Cables

50Ω cables with a standard 50 Ω cable plug SMA should be used to connect to the SMA clock connectors on the FIREBIRD™/PMC Dual G-Link board. AMP PN 1056443-1 with RG-174 cable is one acceptable option, for example.

4.2.5 FIREBIRD™/PMC Dual G-Link Board

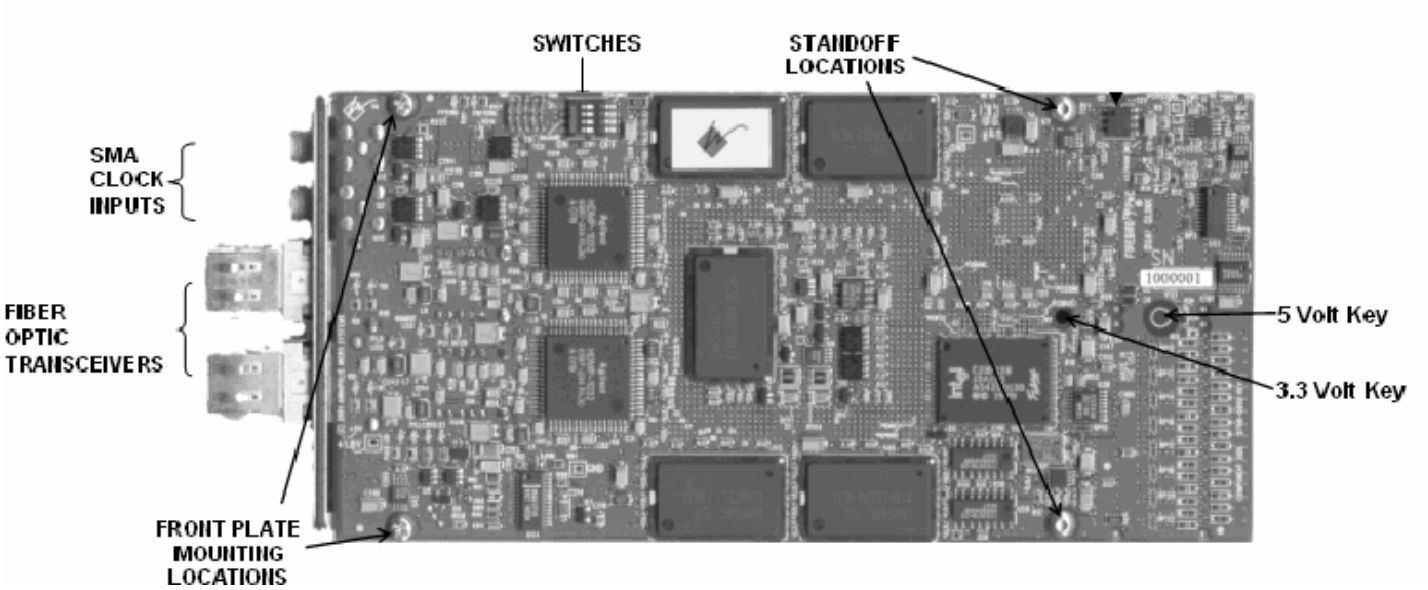


Figure 4–19: FIREBIRD™/PMC Dual G-Link - Component Side

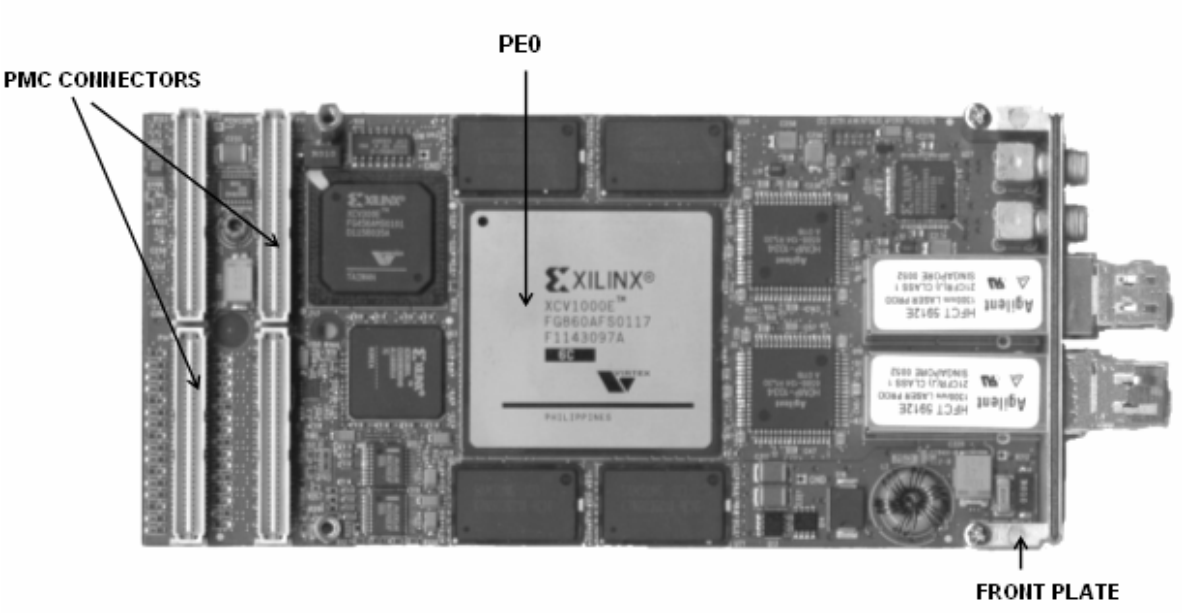


Figure 4–20: FIREBIRD™/PMC Dual G-Link - Solder Side

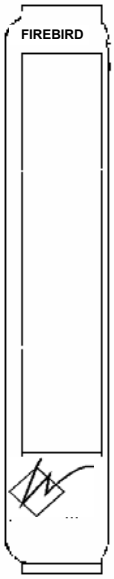


Figure 4–21: FIREBIRD™/PMC Dual G-Link - Front Panel

4.2.6 FIREBIRD™/PMC Dual G-Link Switch Settings

See Section 4.2.3.1.

4.2.7 FIREBIRD™/PMC Dual G-Link LED Definitions

The illustration below shows identifies each LED and its location on the Dual G-Link board. LEDs are numbered in the illustration from either left to right or top to bottom. Table 4-6 lists the names and descriptions of each LED.

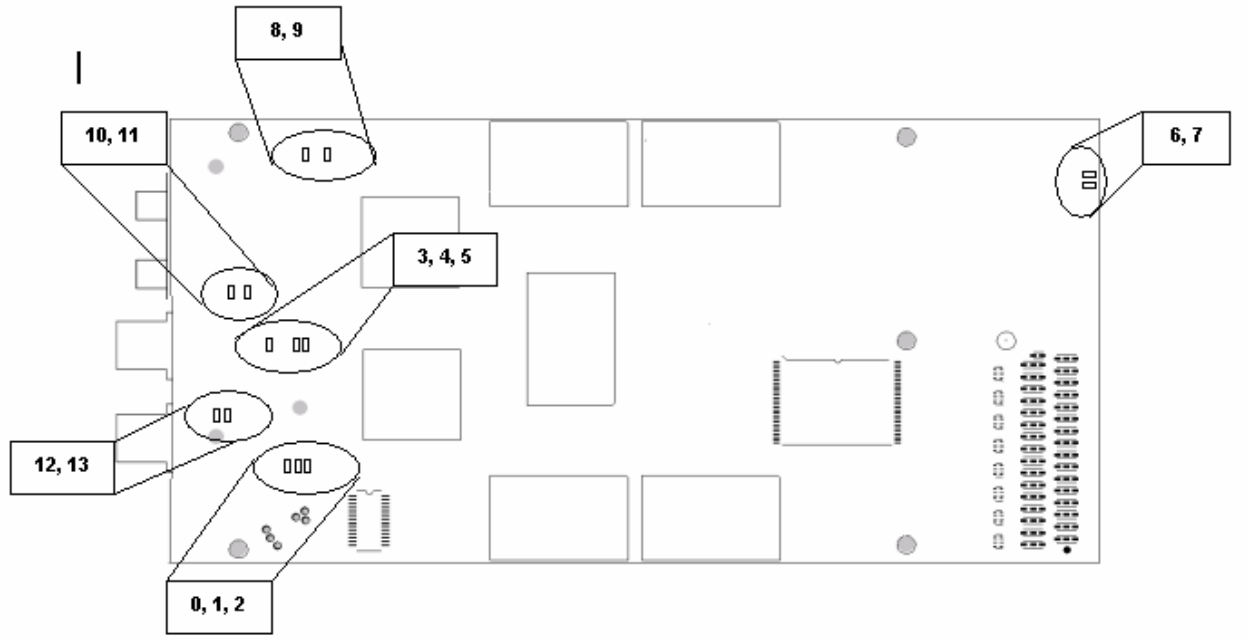


Figure 4–22: FIREBIRD™/PMC Dual G-Link - LEDs on Component Side

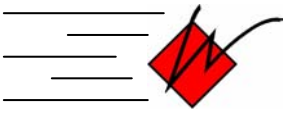
Table 4-6: PMC Dual G-Link LED Definitions

LED NUMBER	Name	Color	Description
0	User LED	Red	Connected to PE0
1	User LED	Yellow	Connected to PE0
2	User LED	Green	Connected to PE0
3	PCI Transfer	Yellow	Denotes that A PCI transfer is taking place.
4	66MHz PCI	Green	Denotes that the PMC slot is 66Mhz
5	PLD Ready	Green	PLD is ready (Flashes at startup programming)
6	~PCI Ready	Green	PCI controller is ready.
7	PE Done	Green	PE0 is programmed.

LED NUMBER	Name	Color	Description
8	FPTERM	Yellow	See front panel clock termination options, Chapter 6.
9	TRIGTERM	Yellow	See front panel clock termination options, Chapter 6.
10	RX0	Red	On when signal is detected on receiver channel 0
11	TX0	Red	On when channel 0 transmitter is enabled and power is on.
12	RX1	Yellow	On when signal is detected on receiver channel 1.
13	TX1	Yellow	On when channel 1 transmitter is enabled and power is on.

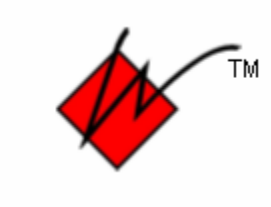
4.3 Host Software for FIREBIRD™/PCI and PMC

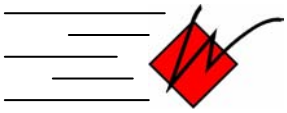
Once the FIREBIRD™ board has been installed in the host system and the host machine has been powered up, the WILDSTAR™ Host Software and FIREBIRD™ VHDL Models can be installed. See the Release Notes included with the FIREBIRD™ board for installation instructions.



5 TECHNICAL SUPPORT

If you have any questions about installing, programming, using, or maintaining the FIREBIRD™ 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. The web site address is http://www.annapmicro.com.


TM



6 HARDWARE REFERENCE

6.1 FIREBIRD™ Hardware

This chapter contains hardware reference information including switching and DC electrical specifications, I/O connector pinouts, and power consumption specifications for the FIREBIRD™/PCI and PMC boards.

6.2 General Specifications

Table 6-1 specifies the physical dimensions and operating range for the FIREBIRD™/PCI and PMC boards.

Table 6-1: FIREBIRD™ Specifications

FIREBIRD™/PCI	
Physical Dimensions:	Length: 312.00mm/12.283in Width: 106.68mm/4.200in Thickness: 1.57mm/0.062in
Operating Range:	Temperature: 0° to 70° C
FIREBIRD™/PMC (Dual WSDP™, Dual G-Link)	
Physical Dimensions:	Length: 149.0mm/5.87in Width: 74.0mm/2.91in Thickness: 1.6mm/0.063in
Operating Range:	Temperature: 0° to 70° C

6.3 Power Consumption

The power consumed by the FIREBIRD™ board directly correlates to the total number of flip-flops in the design and the frequency at which it is run.



CAUTION

According to **PCI** Local Bus Revision 2.1 specifications, the total power should not exceed **25** watts per slot. If the estimated power consumption for an application exceeds 25 watts per slot, the user must ensure that the base system is capable of supplying adequate power and cooling to run the application.

CAUTION

According to **PMC** Local Bus Revision 2.1 specifications, the total power should not exceed **7.5** watts per slot. If the estimated power consumption for an application exceeds 7.5 watts per slot, the user must ensure that the base system is capable of supplying adequate power and cooling to run the application.

6.4 FIREBIRD™ Board Clocks

All FIREBIRD™ boards have three main clocks for use in applications—MCLK, UCLK, and KCLK. FIREBIRD™/PMC has an additional XCLK clock. All clocks are distributed across the FIREBIRD™ board with very low skew. MCLK and UCLK can fundamentally be driven by two sources independently, selectable by the host.

6.4.1 Frequency Parameters

Table 6-2 describes each clock, its frequency ranges, and its destinations for the FIREBIRD™/PCI and /PMC boards.

Table 6-2: FIREBIRD™/PCI and PMC Clock Description

Clock	Full Name	Clock Source	Frequency	Range (MHz)	Destination
UCLK	User Clock	PGM OSC_1	**Minimum**	**Maximum**	PE0, Euro I/O Card
			10	150	
		Euro I/O Card	0	150	
MCLK	Memory Clock	PGM OSC_0	10	150	PE0, Euro I/O Card, Memories
		Euro I/O Card	0	150	
KCLK	System Clock	PCI Bus	10	66	PE, Euro I/O Card
XCLK *(PMC only)*	VME Backplane Generated Clock	P14 Connector	0	100 MHz	VME Backplane Generated Clock

6.4.2 Clock Skew

Table 6–3: FIREBIRD™ /PCI and PMC Clock Skew

Symbol	Description		Typical	Maximum	Units
T_{SKEW}	Maximum Clock Skew (UCLK, MCLK, KCLK)	Single Board	<0.5	1.5	ns

6.4.3 UCLK

UCLK is a user configurable clock. It is asynchronous to MCLK and KCLK. The source of UCLK is selectable via the WILDSTAR™ host software between the FIREBIRD™ programmable oscillator and the external I/O card.

6.4.4 MCLK

All PE memory is run exclusively from MCLK. MCLK asynchronous to UCLK and KCLK. The source of MCLK is selectable via the WILDSTAR™ Host Software between the FIREBIRD™ programmable oscillator and the external I/O card.

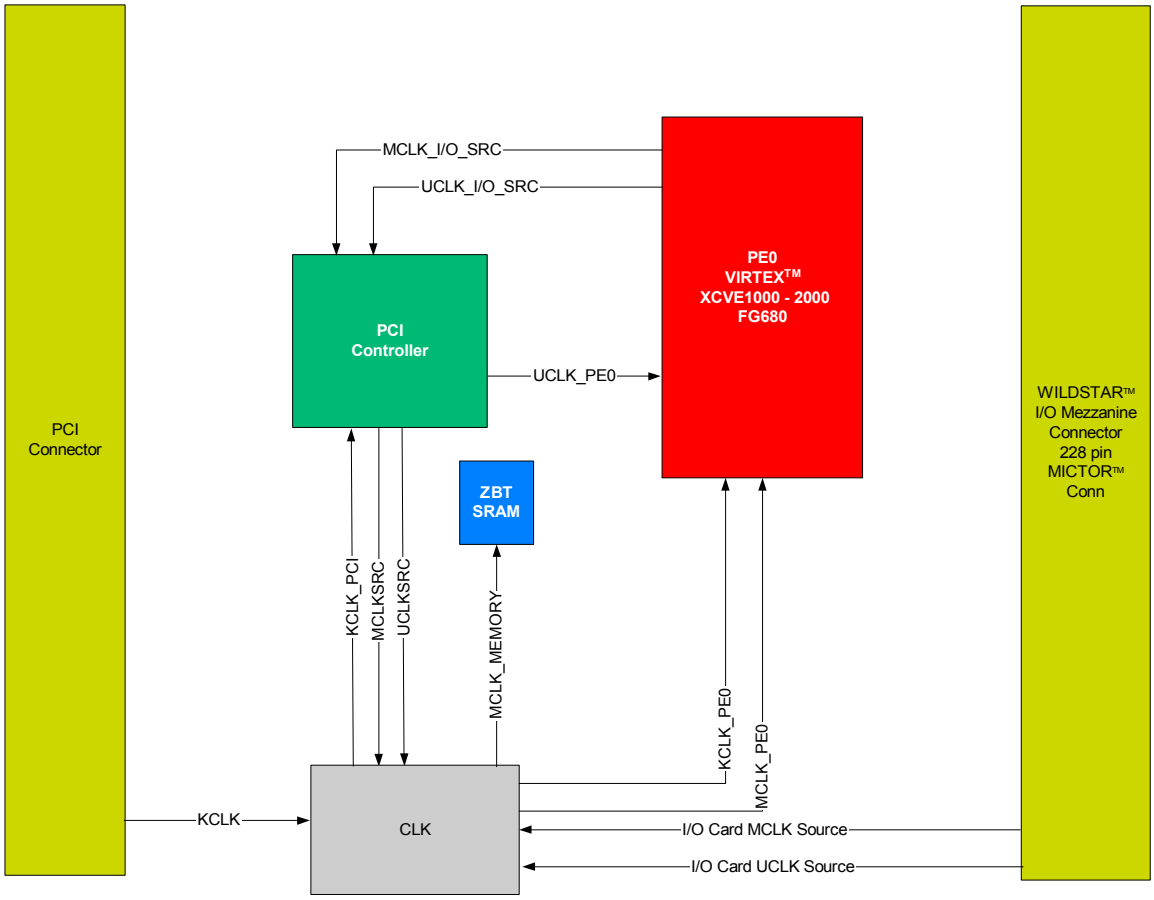
6.4.5 KCLK

KCLK is the Local Address Data Bus (LAD Bus) clock. KCLK is asynchronous to UCLK and MCLK. The PE uses this clock to interface to the PCI Controller for host access via the LAD bus. The LAD bus speed is set by the speed of the PCI bus, which is effectively KCLK.

6.4.6 XCLK (FIREBIRD™/PMC only)

XCLK comes from the VME backplane via the P14 PMC connector on the host motherboard. This pin is the signal “XCLKI” when plugged into a RACEway™ backplane.

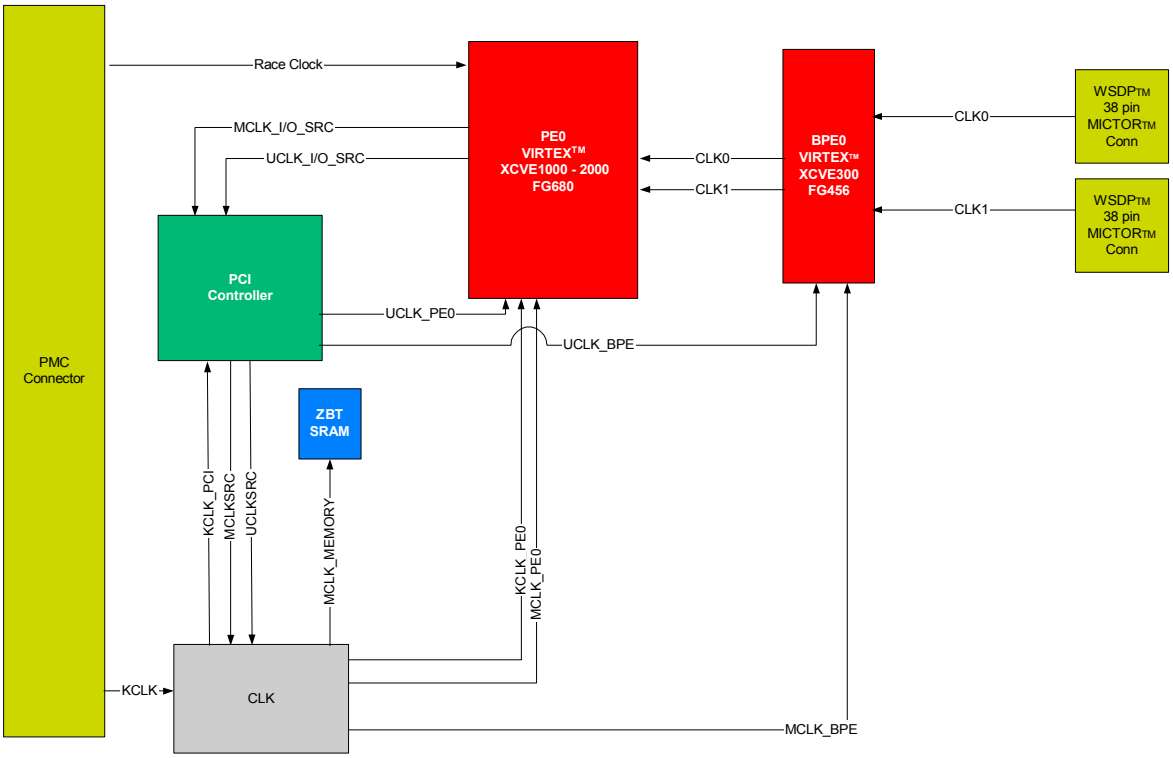
6.4.7 FIREBIRD™/PCI Clock Diagram



Copyright 2000-2001 Annapolis Micro Systems, Inc.

Figure 6–1: FIREBIRD™ /PCI Clock Structure

6.4.8 FIREBIRD™/PMC Dual WSDP™ Clock Diagram

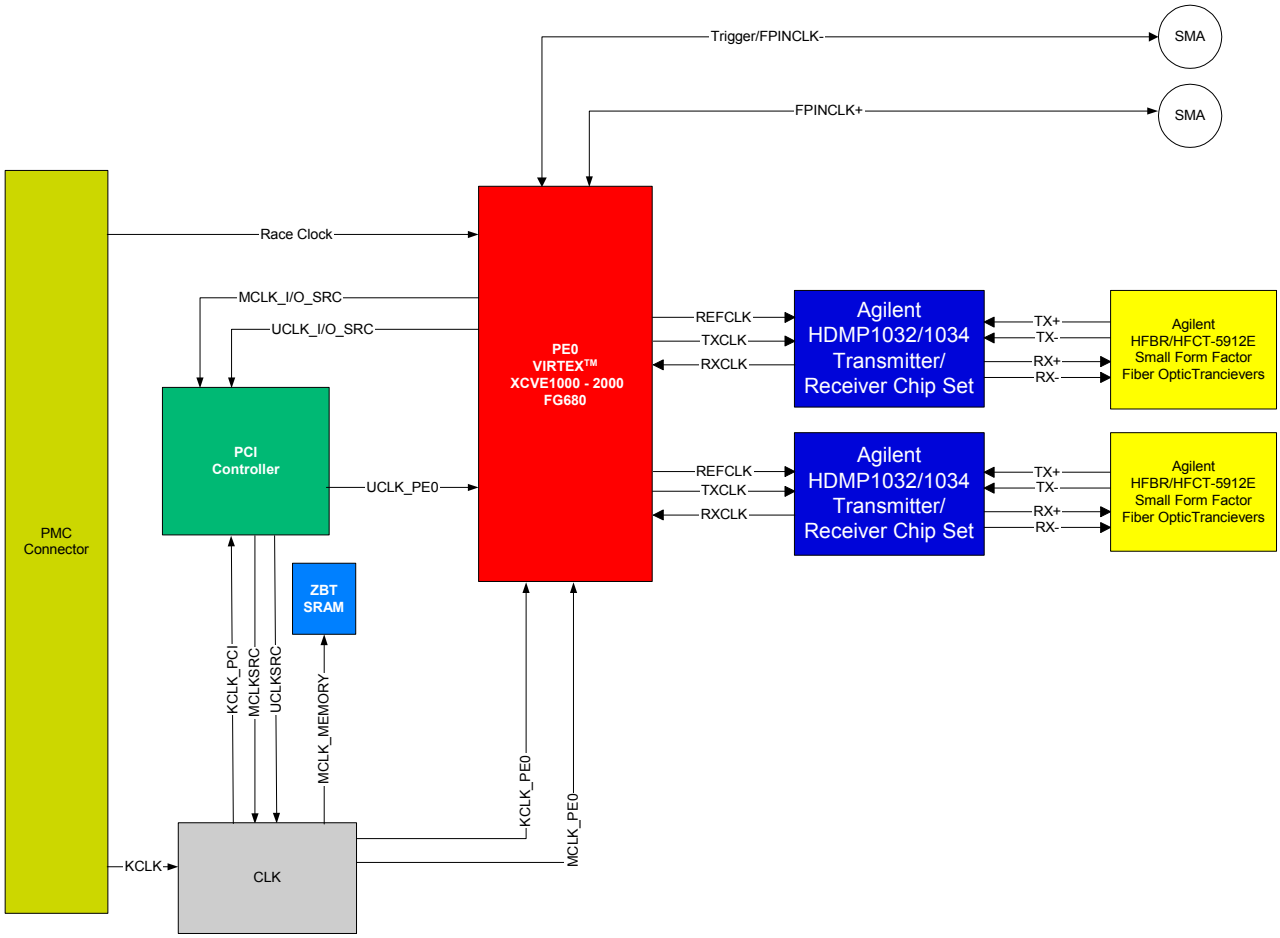


Note: 66 MHz requires 3.3V board

Copyright 2000-2001 Annapolis Micro Systems, Inc.

Figure 6-2: FIREBIRD™/PMC Dual WSDP™ Clock Structure

6.4.9 FIREBIRD™/PMC Dual G-Link Clock Diagram



Copyright 2000-2001 Annapolis MIcro Systems, Inc.

Note: 66 MHz requires 3.3V board

Figure 6-3: FIREBIRD™/PMC Dual G-Link™ Clock Structure

6.4.10 Front Panel Clocks for FIREBIRD™/PMC Dual G-Link

CAUTION

When driving clock inputs, the maximum voltage should not exceed 3.3 volts, nor should the input go below ground. If these limits are exceeded, serious damage to the FPGA may result.

Two SMA connectors are located on the front panel of the FIREBIRD™/PMC Dual G-Link board. Each clock input is equipped with protection diodes. These inputs provide the ability to receive clocks from LVTTL, LVPECL, and 50 ohm clock sources by setting various termination options. The ability to generate an LVTTL clock is also provided.

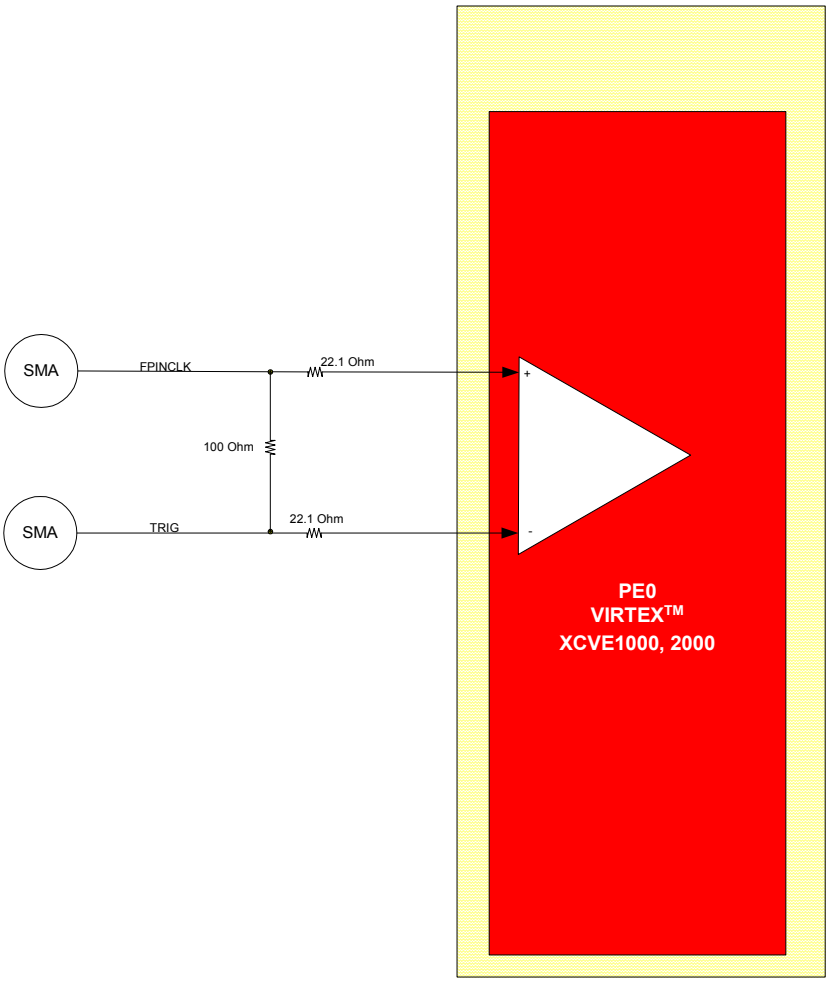
INFORMATION NOTE

Since SMA clock inputs are LVTTL, input levels should not exceed 3.3 volts.

There are four termination options available for the FIREBIRD™/PMC Dual G-Link board front panel clocks (explained below). Details needed for setting each of these options appear in Table 6-4.

6.4.10.1 LVPECL Differential Receive

This termination option is recommended if long clock cabling requirements apply, and works well for frequencies up to 70 MHz. The termination input circuit is shown in Figure 6-4.

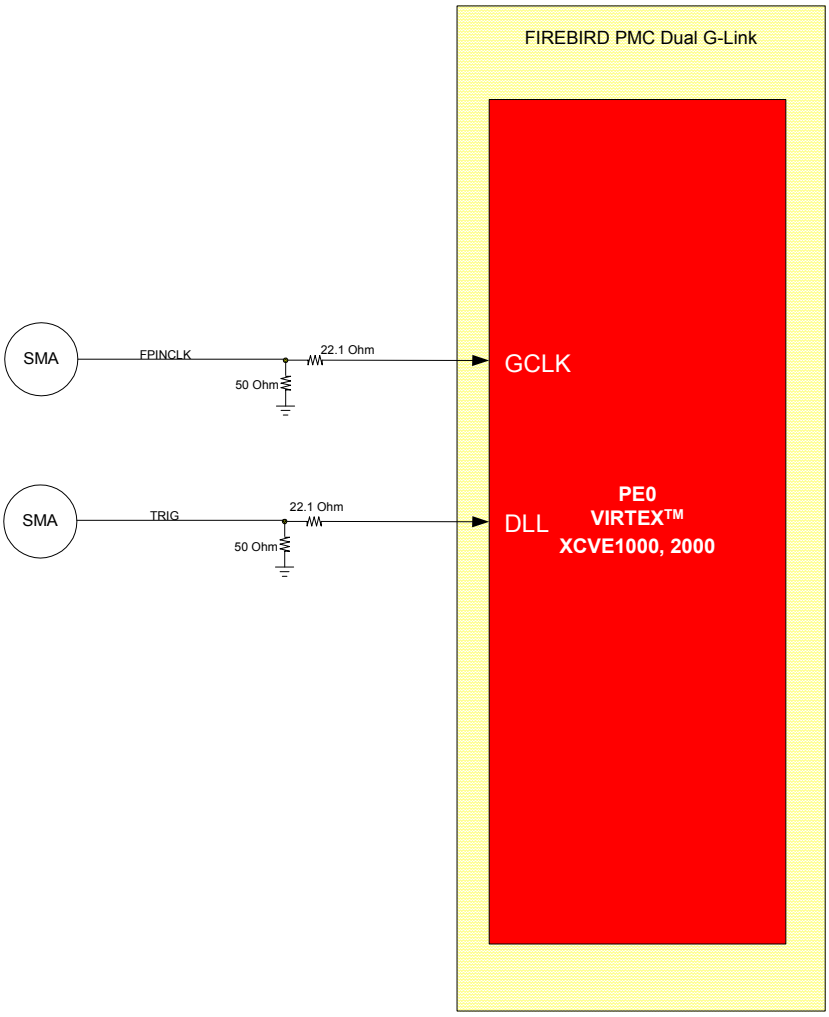


Copyright Annapolis Micro Systems, Inc. 2001

Figure 6-4: LVPECL Differential Receive

6.4.10.2 50 Ohm (3.3v max.), Single-Ended Receive (Default Setting)

This termination option is the power up default setting, and is typically used when receiving from a 50 ohm source. The maximum recommended frequency is [TBD]. The termination input circuit is shown in Figure 6-5.

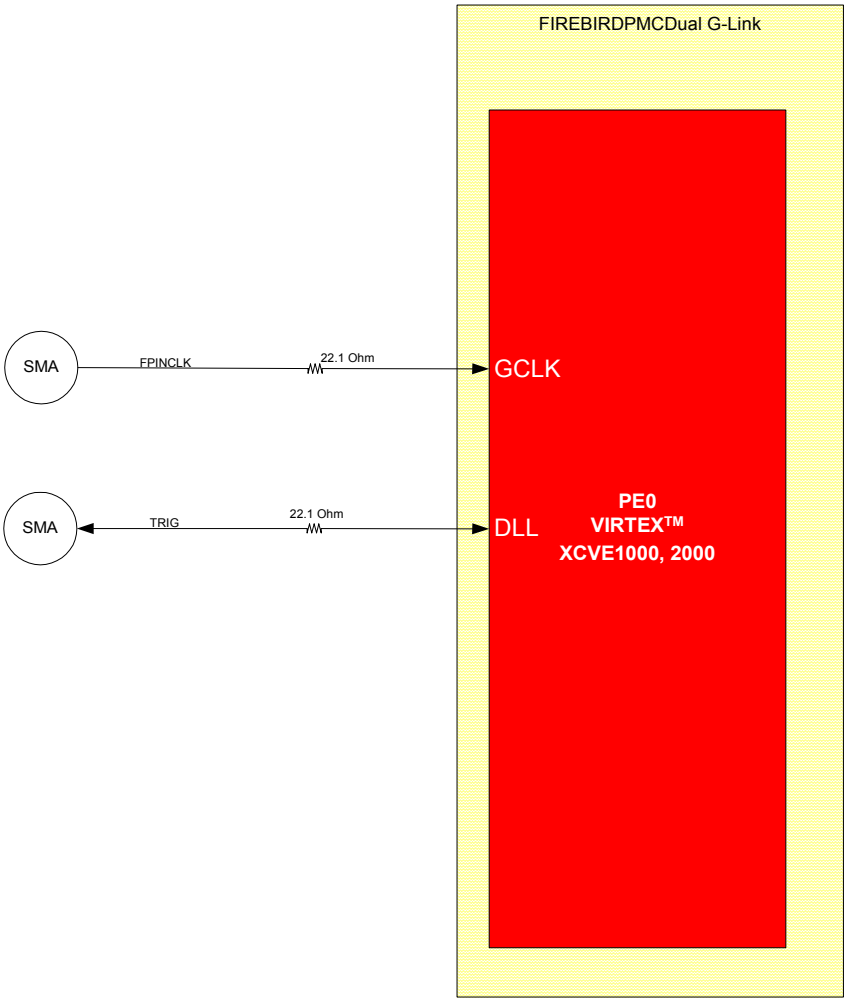


Copyright Annapolis Micro Systems, Inc. 2001

Figure 6-5: 50 Ohm (3.3v max.), Single-Ended Receive (Default Setting)

6.4.10.3 Series Terminated

The TRIG input for this termination option is configured either as input or output, with a 22 ohm series termination. The FPINCLK is configured as input only, with a series termination of 22 ohms also. Maximum frequency is [TBD].

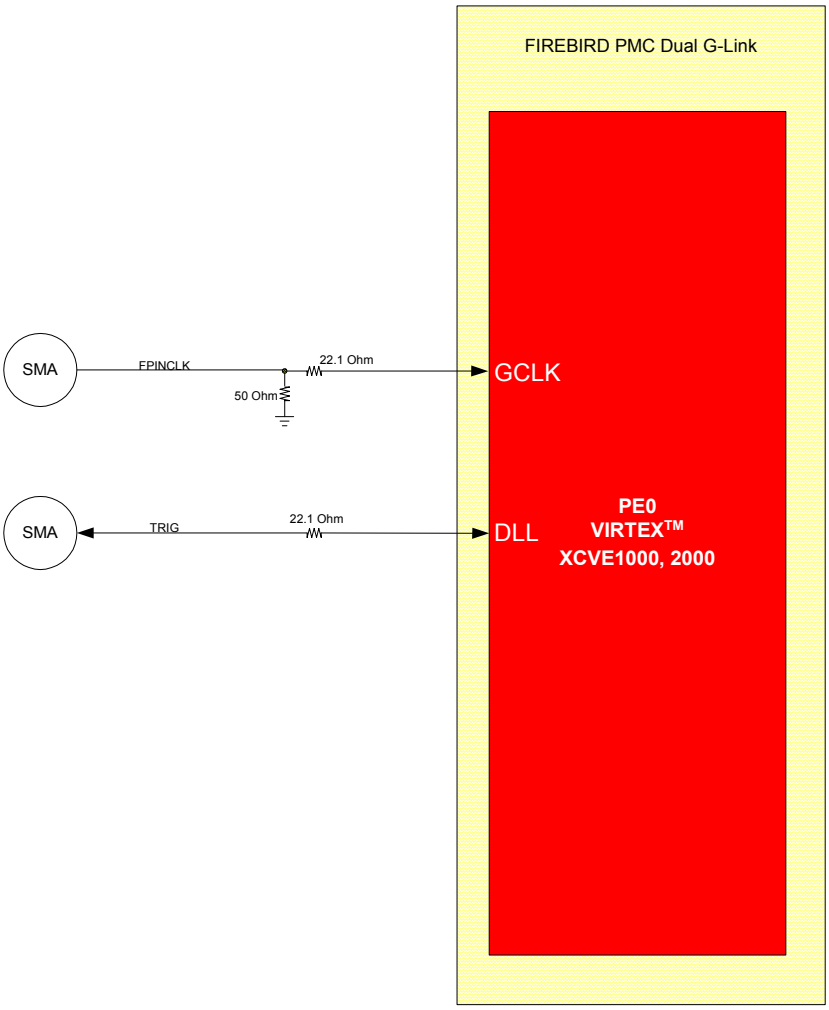


Copyright Annapolis Micro Systems, Inc. 2001

Figure 6-6: Series Terminated

6.4.10.4 Send/Receive TRIG—Receive on FPINCLK

As with the Series Terminated option, the TRIG clock is configured as input or output, with a 22 ohm series resistor. FPINCLK is the same as the 50 ohm option described earlier.



Copyright Annapolis Micro Systems, Inc. 2001

Figure 6-7: Send/Receive TRIG—Receive on FPINCLK

CAUTION

When using a 50 ohm driver to supply a clock to the FIREBIRD™/PMC Dual G-Link board, you must make sure that the termination is correctly set before applying the clock. Failure to do so could damage the board. The Dual G-Link board is set to this termination on power up and whenever the PE is programmed. Care must be taken when setting PASS mode options since they require writes to the register where the termination control bits are located.

6.4.10.5 Dual G-Link Termination Options

Table 6-4: Termination Options

Termination Mode	FPCLKTERMENB	TRIGTERMENB
LVPECL differential receive	0	0
Series termination send/receive on Trigger. Series termination receive on FPINCLK	0	1
Receive with 50 ohm driver termination on both clock inputs. (Power on default)	1	0
Send/Receive on Trigger with series termination. Receive on FPINCLK with 50 driver termination.	1	1

6.4.10.6 Transmit and Receive Clocks

Each of the two fiber optic channels on the FIREBIRD™/PMC Dual G-Link board uses three clocks: TXCLK, RXCLK, and REFCLK.

- TXCLK is used to send data out of each fiber optic channel and is sourced by PE0.
- RXCLK represents the "recovered" clock from the fiber optic line. This clock is used to receive data when the fiber optic chipset is receiving data in Normal mode.
- REFCLK is used when the receive chipset is receiving data in PASS mode.

6.4.10.7 Sourcing Motherboard Clocks (MCLK, UCLK)

The FIREBIRD™/PMC Dual G-Link board provides the ability to synchronize an external clock to one of the motherboard clocks. MCLK, UCLK, or both can be synchronized from the front panel clock or from one of the fiber optic channels on the board. MCLK and UCLK are distributed on the board with very low skew.

6.5 FIREBIRD™/PMC Dual G-Link Fiber Optic Channels

Each FIREBIRD™/PMC Dual G-Link board has two full-duplex fiber optic channels. Each channel is capable of 280 MB/sec. The next few sections describe the receive and transmit modes of the fiber optic channels.

6.5.1 Receive Modes

The FIREBIRD™/PMC Dual G-Link board employs two different receive modes—Normal Mode and Pass Mode—to receive data from the fiber optic transceivers. The choice of which mode to implement depends on the user's system configuration. (Please consult the Agilest HDMP-1032/1034 transceiver chip set documentation at **www.semiconductor.agilent.com** for more details on the operation of transmit and receive modes.)

6.5.1.1 Normal Mode

During normal operation, RXCLK is recovered from the fiber optic serial data stream by the receiver chip. Figure 6-8 shows the Normal mode structure that applies to each of the two channels. On the falling edge of RXCLK, the data, control, and status bits from the receiver are clocked out of the receiver chip. This usually requires that a FIFO be used to cross the clock boundary between RXCLK and an internal PE0 clock. This mode is typically employed if the transmit clock cannot easily be distributed to the FIREBIRD™/PMC Dual G-Link board.

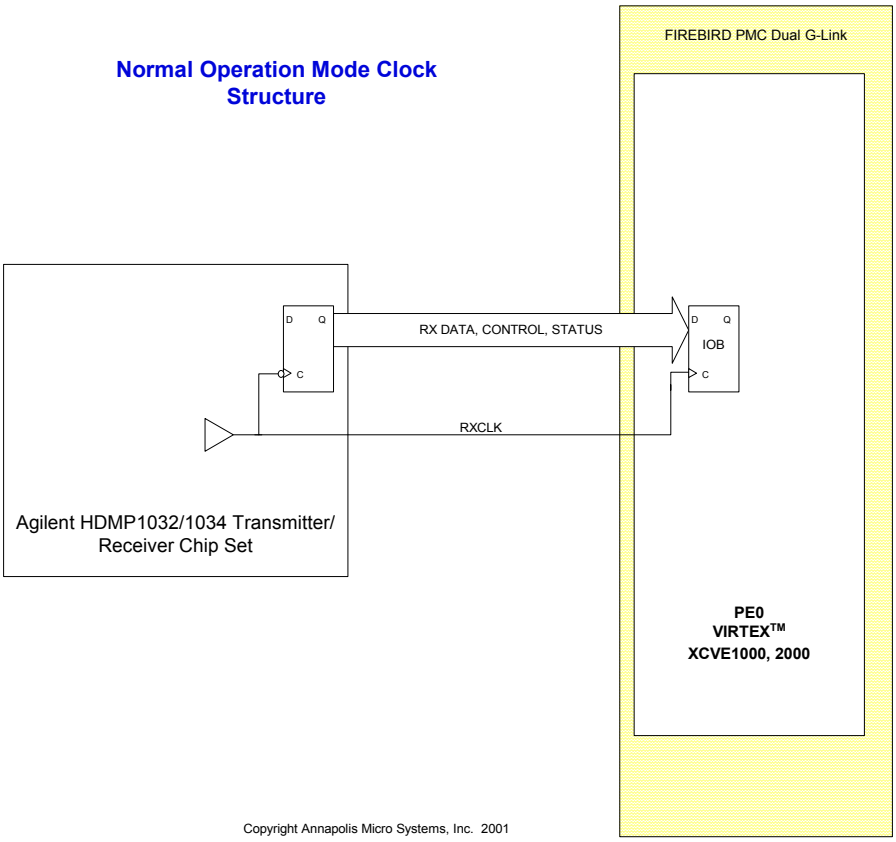


Figure 6-8: Normal Operation Mode Clock Structure

6.5.1.2 PASS Mode

PASS mode can be used to potentially avoid the use of a receive FIFO if the transmit clock is available to the FIREBIRD™/PMC Dual G-Link board. Figure 6-9 illustrates the PASS mode structure that applies to both channels. When operating in PASS mode, the Dual G-Link board uses the clock from the transmitter to run REFCLK. The receive chips align data with the rising edge of REFCLK. Multiple channels can be put in parallel PASS mode so that data sizes of greater than 17 bits can be received synchronously.

Parallel Pass Mode involves two or more channels that are concatenated in PASS mode together, forming a single data stream. These combined channels are able to compensate for a phased difference between REFCLK and recovered RXCLK.

INFORMATION NOTE

The REFCLK must be the *same clock* as the transmitted clock. This is typically accomplished by bringing the transmit clock into the Dual G-Link board through the front panel connector, or by synchronizing a clock in PE0 to the transmit clock source.

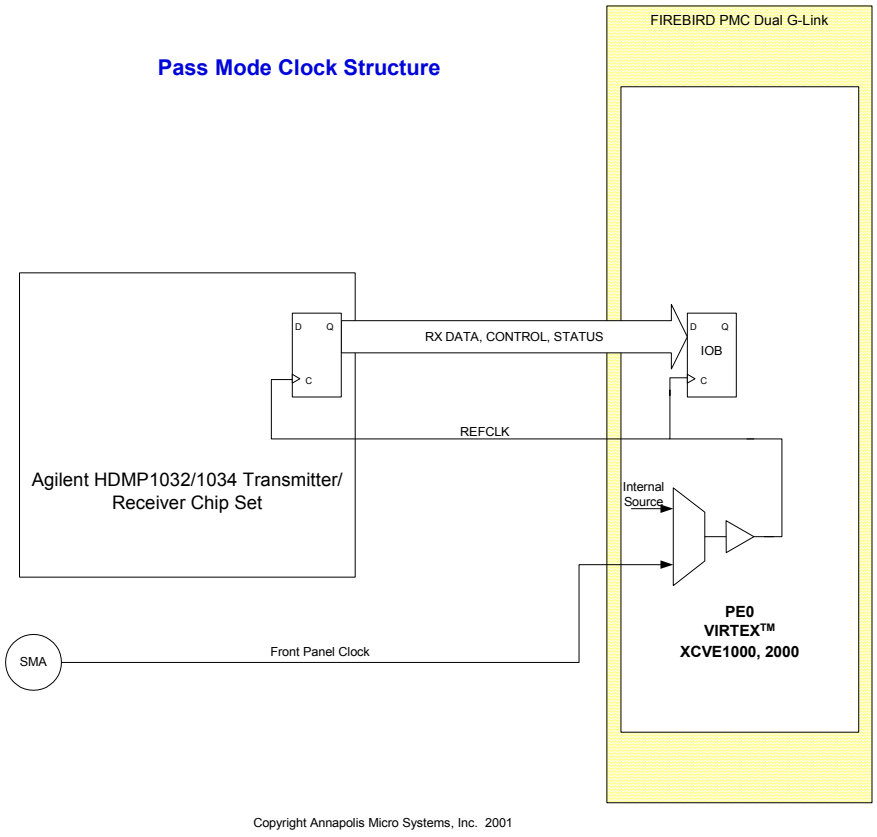


Figure 6-9: Pass Mode Clock Structure

6.5.2 Transmit Modes

As shown in Figure 6-10, transmit mode PE0 sources the clock to the transmitter chips, and the transmit data is clocked out of PE0 IOB on the falling edge of the selected transmit clock source.

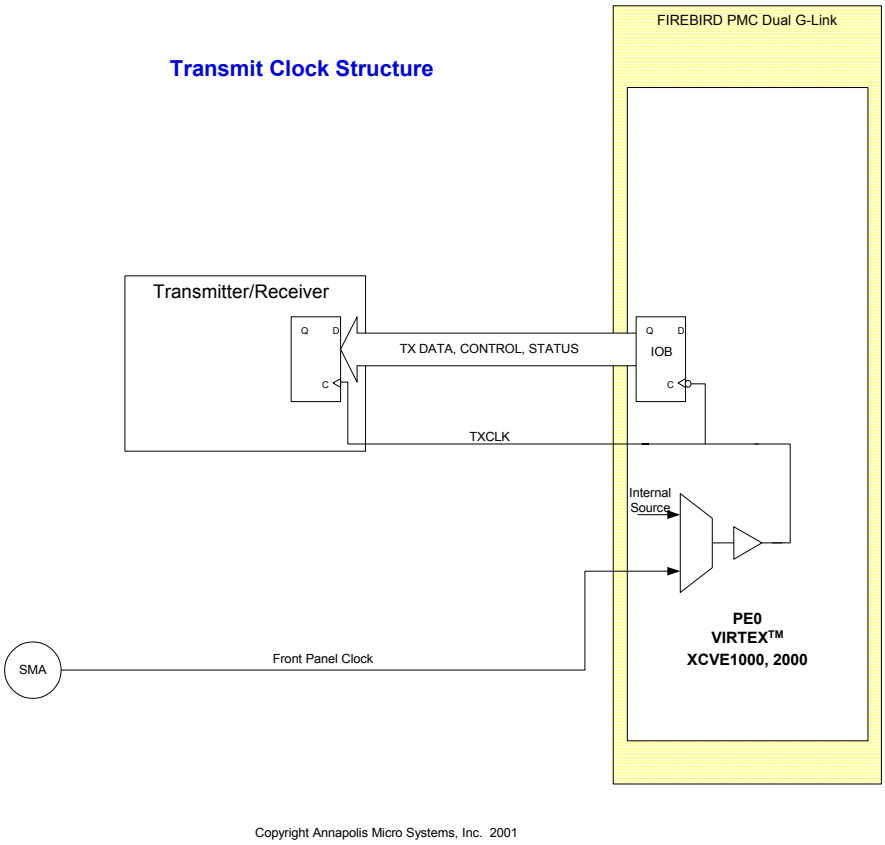


Figure 6-10: Transmit Mode Clock Structure

6.5.3 Transceiver Configuration Registers

(NOTE: All registers are 10 bits)

INFORMATION NOTE

To ease the transferring of designs between the FIREBIRD™/PMC Dual WSDP™ and the Quad G-Link Fiber I/O card, the control registers on the PMC Dual WSDP™ should be considered as a subset of the control registers on the Quad G-Link Fiber I/O card...

Table 6-5: G-Link PLD Register Definition

Register Name	Address	Description
Channel 0 Control Reg.	000	Contains static control signals for G-Link Channel 0
Channel 1 Control Reg.	001	Contains static control signals for G-Link Channel 1
Pass Mode Routing Control Reg.	100	Selects PASS mode configuration and clock termination options.
Version Register	101	Tells current version of PLD design

Table 6-6: Channel [1:0] Control Registers

(This register is identical for each port.)

Bit Number	Description
9	RXPASSENB - 1 Enables Pass Mode 0 disables. Disabled on power up
8	RXESMPXENB - 1 Enables Enhanced Simplex Mode, 0 Disables. It should be disabled when it is used with the older chipset (HDMP -1022/1012). Enabled on power up
7	RXFLGENB – 1 Enables 17th Data Bit, 0 Disables. Disabled on power up.
6	RXDIV1 – Sets frequency range, see table below Set to highest range on power up.
5	RXDIV0 – Sets frequency range, see table below
4	TXDISABLE – 1 Disables transmitter, 0 enables. Disabled on power up.
3	TXESMPXENB - 1 Enables Enhanced Simplex Mode, 0 Disables. It should be disabled when it is used with the older chipset (HDMP -1022/1012).

Bit Number	Description
	Enabled on power up
2	TXFLGENB – 1 Enables 17th Data Bit, 0 Disables Disabled on power up
1	TXDIV1 – Sets frequency range, see table below. Set to highest range on power up.
0	TXDIV0 – Sets frequency range, see table below. Set to highest range on power up.

Table 6-7: Clock Frequency Range

Parallel Word Rate (Mword/sec)	TX/RX DIV[1:0]
40 – 70 (max)	0 0
20 – 45	0 1
13 (min) – 26	1 0
Used only in test mode	1 1

Table 6-8: PASS Mode Control Register

Bit Number	Description
9	RESERVED
8	RESERVED
7	RESERVED
6	RESERVED
5	FPCLKTERMENB, see table below for termination options
4	TRIGTERMENB, see table below for termination options
3	Select 3 – See table below for PASS mode options
2	Select 2 – See table below for PASS mode options
1	Select 1 – See table below for PASS mode options
0	Select 0 – See table below for PASS mode options

Table 6-9: PASS Mode and Clock Termination Options

MODE	Control Word
Channels 0 and 1 chained together in parallel PASS mode.	X X 0 0 0 0
Channels 0 and 1 functioning independently.	X X 1 1 1 1
LVPECL differential receive	0 0 X X X X
Series termination send/receive on Trig. Series termination receive on FPINCLK	0 1 X X X X

MODE	Control Word
Receive with 50 ohm driver termination on both clock inputs. (Power on default)	1 0 X X X X
Send/Receive on Trigger with series termination. Receive on FPINCLK with 50 driver termination	1 1 X X X X

Table 6-10: LED Definitions

LED Name	Definition
Parallel PASS LED	Lights when channels 0 and 1 are configured to be chained together in PASS mode.
RX0 LED	Lights when signal detect goes high.
RX1 LED	Lights when signal detect goes high.
TX0 LED	Lights when power is on and when transmitter is enabled.
TX1 LED	Lights when power is on and when transmitter is enabled.



CAUTION

When using a 50 ohm driver to supply a clock to the FIREBIRD™/PMC Dual G-Link board, you must make sure that the termination is correctly set before applying the clock. Failure to do so could damage the board. The G-Link is set to this termination on power up and whenever the PE is programmed. Care must be taken when setting the PASS mode options since they require writes to the register where the termination control bits are located.

6.6 DC Characteristics

The following tables describe the DC characteristics for the I/O pins that interface to PE0 from the I/O card.

6.6.1 FIREBIRD™/PCI

Table 6–11: FIREBIRD™/PCI External I/O Signal Electrical Characteristics

Symbol	Description	Minimum	Maximum	Units
V_{OH}	High-level output voltage @ I_{OH} = -24.0 mA	2.4		V
V_{OL}	Low-level output voltage @ I_{OL} = 24.0 mA		0.4	V
V_{IH}	High-level input voltage @ I_{OH} = -24.0 mA	2.0	3.6	V
V_{IL}	Low-level input voltage @ I_{OH} = -24.0 mA	-0.5	0.8	V
CIN	Input capacitance (sample tested)		8.0	pF

6.7 FIREBIRD™ External Interface Options

INFORMATION NOTE
FIREBIRD™/PMC boards *are not* designed to accept I/O daughter cards. They can only be used with FIREBIRD™/PCI boards.

6.7.1 External I/O Cards

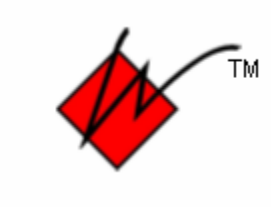
External I/O cards present a high bandwidth, flexible conduit for transmitting data to and from the FIREBIRD™/PCI board. FIREBIRD™/PCI will accept one WILDSTAR™ Euro I/O daughter card. More detailed information can be found in the manual that accompanies the daughter card.

6.7.1.1 Euro I/O Cards

The Euro I/O cards have the ability to communicate between external I/O and PE0 using 66 and 96 bit-directional busses. Physical specifications and connection signals for the Euro I/O card are presented in Attachment E.

6.8 FIREBIRD™/PMC Dual WSDP™ User VHDL MICTOR® Pin Mapping

For information regarding the mapping of user VHDL to MICTOR pins for the FIREBIRD™/PMC Dual WSDP™, see Attachment F.

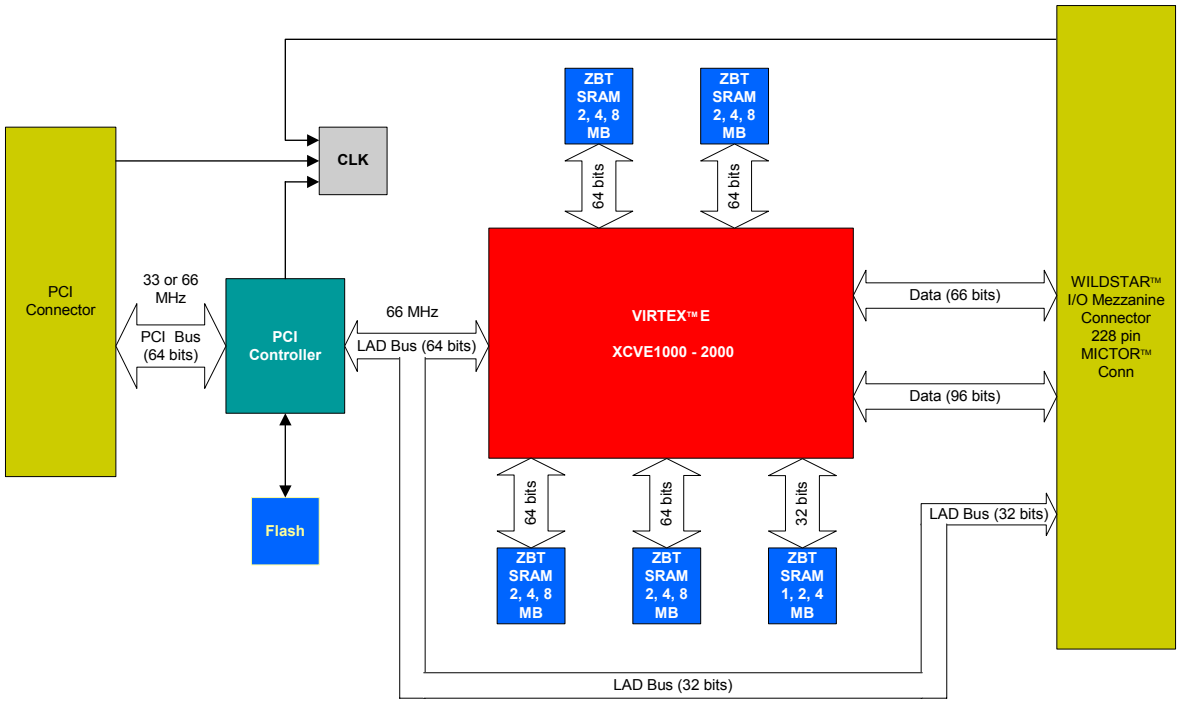

TM

7 VHDL MODELS REFERENCE AND GUIDE

This section discusses the VHDL model of the FIREBIRD™ system. This model includes the FIREBIRD™ board and a simulation model of the host system. The following sections describe the various VHDL user interfaces to the FIREBIRD™ system.

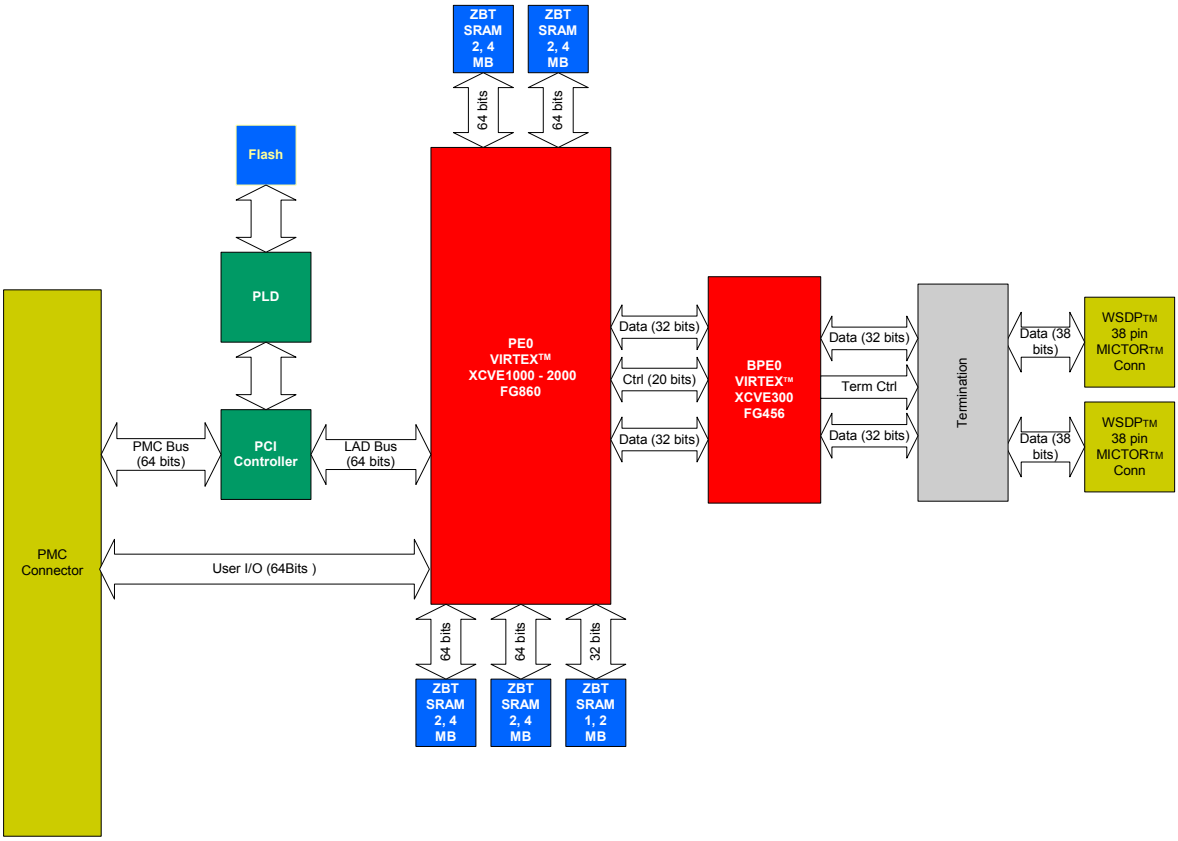
7.1 FIREBIRD™ Block Diagrams

The following block diagrams illustrate the structure, layout, and connectivity of the FIREBIRD™ /PCI and /PMC systems.



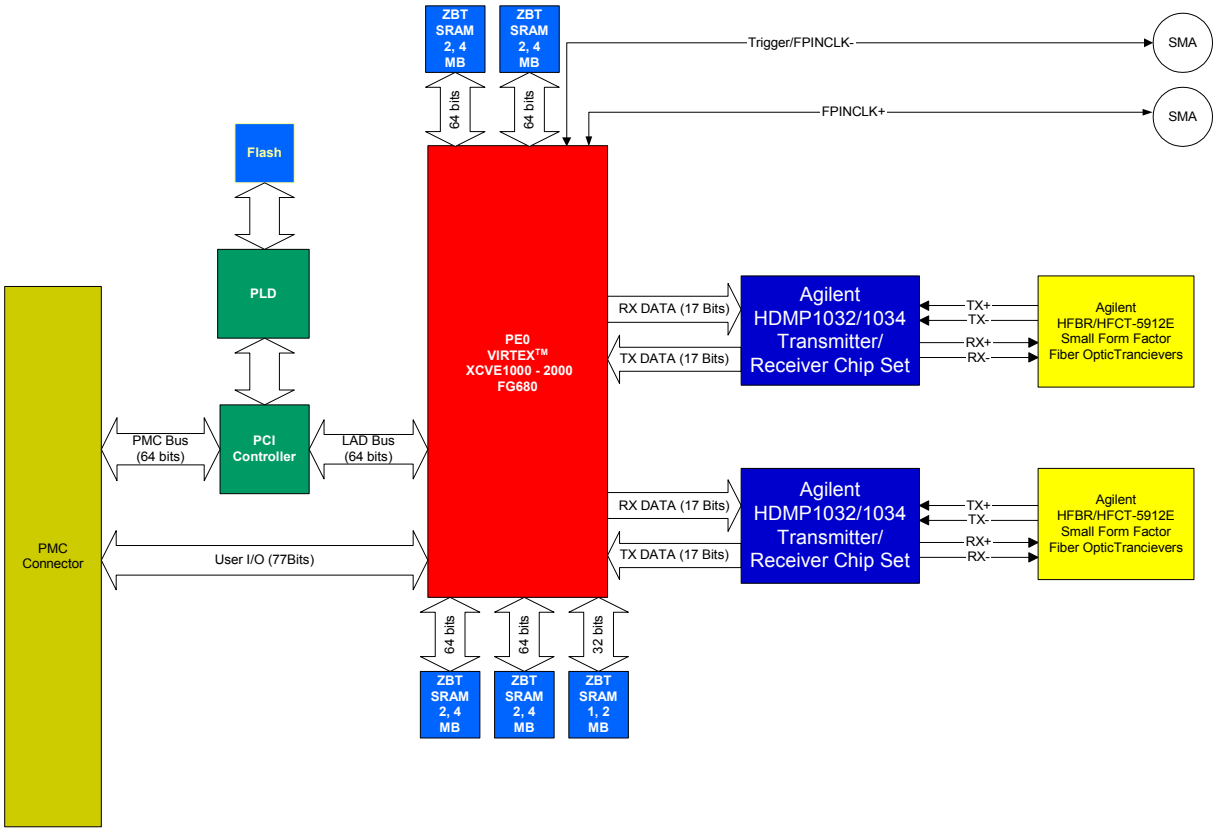
Copyright 2000-2001Annapolis MIcro Systems, Inc.

Figure 7-1: FIREBIRD™/PCI



Copyright 2000-2001 Annapolis MIcro Systems, Inc.

Figure 7-2: FIREBIRD™/PMC Dual WSDP™



Copyright 2000-2001 Annapolis MIcro Systems, Inc.

Figure 7-3: FIREBIRD™/PMC Dual G-Link

7.2 VHDL Components Common to all FIREBIRD™ Types

7.2.1 LAD 64 Bus Standard Interface (LAD64_Bus_Std_IF)

The local address data (LAD) bus is the primary means of communicating with the host system. The LAD Bus Standard Interface (LAD64_Bus_Std_IF) component provides the user interface to the LAD bus from within the PE0 device. The LAD bus is also the communication path between the PCI Controller and PE0 device. The LAD bus is a single master (PCI Controller), 64-bit, shared address/data bus. Every cycle on the LAD bus is initiated by the PCI Controller and may last anywhere from four to hundreds of clock (K_Clk) cycles.

The "User_In" and "User_Out" port signals, described in Table 7-1, should be used by the design to receive data from and send data to the LAD bus respectively. The "pads" signals of the LAD64_Bus_Std_IF component are only used to connect the LAD64_Bus_Std_IF component to the pads of PE0.

K_Clk is the LAD bus clock and it will run at the same frequency as the PCI bus clock, which means that it can be 10-33MHz or 66MHz. Because LAD bus signaling rules are no longer dependent on the LAD clock speed, the PE need not be aware of the K_Clk frequency as long as it is able to meet the worst possible (66MHz) timing. Users will not have the ability to freely select the frequency of this clock since it is dependent on the host system, but they can restrict the clock to a maximum frequency of 33MHz if necessary. All Local Bus signals and address/data lines are synchronous to K_Clk.

Table 7-1: LAD Bus Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
K_Clk	1	Input	PCI/LAD bus clock signal; 33 or 66 MHz.
Global_Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARTUP_Virtex component
Pads.LAD.Addr_Data	64	Bi-dir	Shared address/data bus pad signals; note that the address is a BYTE address
Pads.LAD.DWordL_n	1	Input	Data strobe low word pad signal; indicates when read data has been accepted by PCI bus or when write data is valid on LAD bus
Pads.LAD.DWordH_n	1	Input	Data strobe high word pad signal; indicates when read data has been accepted by PCI bus or when write data is valid on LAD bus
Pads.LAD.WR_n	1	Input	Write select pad signal; indicates a write cycle when '0' or read cycle when '1' (when Pads.CS_n is '0')
Pads.LAD.CS_n	1	Input	Chip select pad signal; indicates a valid LAD bus cycle when '0,' otherwise no LAD bus cycle is taking place
Pads.LAD.Int_Req_n	1	Output	Interrupt request pad signal; high-to-low transition generates an PCI interrupt back to the host system

Signal Name	Width	Dir	Description
Pads.LAD.DMA_WStat	2	Output	DMA Write Status Flags; used to report status of DMA Writes in PE0 to the PCI controller.
PadsLAD.DMA_Rstat	2	Output	DMA Read Status Flags; used to report status of DMA Reads in PE0 to the PCI controller.
Pads.LAD.Reset		Input	PCI controller generated reset.
User_In.Addr	23	Output	User interface to the LAD bus address; note that this is a QWORD address; also note that this address is automatically incremented each clock cycle during burst LAD bus cycles
User_In.Data_In	64	Output	User interface to the input data from the LAD bus
User_In.Reg_Strobe_n	1	Output	Register space access strobe signal; indicates a valid register space cycle when '0'
User_In.Mem_Strobe_n	1	Output	Memory space access strobe signal; indicates a valid memory space cycle when '0'
User_In.DwordL_n	1	Output	User interface to data strobe low data signal; used to indicate when valid data is available on the lower 32-bits of the data bus.
User_In.DwordH_n	1	Output	User interface to data strobe high data signal; used to indicate when valid data is available on the upper 32-bits of the data bus.
User_In.Write_Sel_n	1	Output	Write select interface signal; indicates a write cycle when '0' or a read cycle when '1' (when strobe is '0')
User_Out.Data_Out	64	Input	User interface to the output data to the LAD bus
User_Out.Int_Req_n	1	Input	User interface to the interrupt request signal; high-to-low transition on this signal generates a single pulse on the interrupt request pad signal (which in turn will generate a PCI interrupt to the host)
User_In.DMA_RStat	2	Output	User interface to the DMA Read status flags; indicates the read status of a DMA operation.
User_Out.DMA_WStat	2	Input	User interface to the DMA Write status flags; indicates the write status of a DMA operation.

The LAD64_Std_IF component has one generic that can be set by the user upon instantiation of the component (this generic is for the FIREBIRD™/PMC board only):

- **V25_ENABLED**: This generic is set to TRUE when the backplane interface BUFFER_TYPE is set to a drive level which required 2.5V; otherwise this generic should be set to FALSE. The default value is FALSE. For more details about this generic, see Section 7.4.2.

7.2.2 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 FIREBIRD™ PE0 device. The LEDs can be used to indicate status or processing activity, and can even be used to debug a PE0 design. The port signals of the LED standard interface are shown in Table 7-2.

The "User_Out" port signals of the LED_Std_IF component should be used by the design to control the LEDs of the FIREBIRD™ PE0 device. The

"Pads" signals of the LED_Std_IF component are only used to connect the LED_Std_IF component to the pads of PE0.

Table 7-2: LED Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads	3	Output	LED pad signals
User_Out_Red_n	1	Input	User interface to the red LED
User_Out_Yellow_n	1	Input	User interface to the yellow LED
User_Out_Green_n	1	Input	User interface to the green LED

The LED_Std_IF component has one generic that can be set by the user upon instantiation of the component (this generic is for the FIREBIRD™/PMC board only):

- **V25_ENABLED**: This generic is set to TRUE when the backplane interface BUFFER_TYPE is set to a drive level which required 2.5V; otherwise, this generic should be set to FALSE. The default value is FALSE. For more details about this generic, see Section 7.4.2.

7.2.3 On-board 64-bit Memory Standard Interface (Mem64_Std_IF)

The PE0 device has access to four identical 64-bit on-board synchronous SRAM memory ports that are referred to as Mem_0, Mem_1, Mem_2 and Mem_3 ports. Each of these ports is accessed through a standard interface called the Mem64_Std_IF. The port signals of the memory standard interface are shown in Table 7-3. The "User_In" and "User_Out" port signals should be used in the design to receive data from and send data to the on-board memory devices, respectively. In the table below, "Mem_x" is used to indicate memory ports 0 through 3.

Table 7-3: Memory Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
M_Clk	1	Input	Memory bus clock signal
Global_Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARTUP_Virtex component
Pads.Mem_x.Addr	19	Output	Address bus pad signals
Pads.Mem_x.Data	64	Bi-dir	Data bus pad signals
Pads.Mem_x.Addr_CS_n	1	Output	This signal is reserved for further use.
Pads.Mem_x.CSH_n	1	Output	Chip select high pad signal; enables both SRAM devices that make up the port for the high 32-bits
Pads.Mem_x.CSL_n	1	Output	Chip select low pad signal; enables both SRAM devices that make up the port for the low 32-bits
Pads.Mem_x.CE_n	1	Output	Clock enable pad signal; enables the clock to the SRAM
Pads.Mem_x.WE_n	1	Output	Write enable pad signal; indicates a write cycle when '0' or a read cycle when '1'
User_In.Data_In	64	Output	User interface to the data input from memory
User_In.Low_Valid_n	1	Output	User interface to the low data input valid flag; indicates that the User_In.Data_In(31 downto 0) signal is valid when '0'
User_In.High_Valid_n	1	Output	User interface to the high data input valid flag; indicates that the User_In.Data_In(63 downto 32) signal is valid when '0'
User_Out.Addr	20	Input	User interface to the address to the memory
User_Out.Data_Out	64	Input	User interface to the output data to the memory
User_Out.Low_Strobe_n	1	Input	User interface to the memory low access strobe; indicates a memory access for the low 32 bits is started when '0'
User_Out.High_Strobe_n	1	Input	User interface to the memory high access strobe; indicates a memory access for the high 32 bits is started when '0'
User_Out.Write_Sel_n	1	Input	User interface to the memory write select; indicates a write access when '0' or a read access when '0' (when User_Out.Strobe_n is '0')
User_Out.CE_n	1	Input	User interface to the clock enable; gives user access to conserve power by turning off the clock inside the memory.

The Mem64_Std_IF component has several generics that can be set by the user upon instantiation of the component:

- **INFF_DELAY**: The INFF_DELAY generic controls whether a delay will be added to the input Flip-Flop. When set to NODELAY, the input signal bypasses the delay. When INFF_DELAY is set to DELAY, the input signal goes through the delay circuit before it is routed out of the pad. The default setting is NODELAY.

- **OBUF_DRIVE**: The OBUF_DRIVE generic controls the output drive strength of the pad. There are several drive strengths associated with this generic. *Note:* if a differential signal is selected in the generic BUFFER_TYPE, is generic is not used.

- **HALF_POPULATED**: The half-populated generic should *never* be instantiated by the user. This generic is used so the memory interface is compatible between Firebird™/PCI and Firebird™/PMC. The default and only condition for this generic is TRUE.

7.2.4 On-Board 32-bit Memory Standard Interface (Mem32_Std_IF)

The PE0 device has access to one 32-bit on-board synchronous SRAM memory port that is referred to as Mem_4. This port is accessed through a standard interface called the Mem32_Std_IF. The port signals of the memory standard interface are shown in Table 7-4. The "User_In" and "User_Out" port signals should be used in the design to receive data from and send data to the on-board memory devices, respectively.

Table 7-4: Memory Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
M_Clk	1	Input	Memory bus clock signal
Global_Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARTUP_Virtex component
Pads.Mem_4.Addr	19	Output	Address bus pad signals
Pads.Mem_4.Data	32	Bi-dir	Data bus pad signals
Pads.Mem_4.Addr_CS_n	1	Output	This signal is reserved for further use.
Pads.Mem_4.CS_n	1	Output	Chip select pad signal; enables both SRAM devices that make up the port for the 32-bits
Pads.Mem_4.CE_n	1	Output	Clock enable pad signal; enables the clock to the SRAM
Pads.Mem_4.WE_n	1	Output	Write enable pad signal; indicates a write cycle when '0' or a read cycle when '1' (when Pads.CS_n is '0')
User_In.Data_In	32	Output	User interface to the data input from memory
User_In.Data_Valid_n	1	Output	User interface to the data input valid flag; indicates that the User_In.Data_In signal is valid when '0'
User_Out.Addr	20	Input	User interface to the address to the memory
User_Out.Data_Out	32	Input	User interface to the output data to the memory
User_Out.Strobe_n	1	Input	User interface to the memory access strobe; indicates a memory access is started when '0'
User_Out.Write_Sel_n	1	Input	User interface to the memory write select; indicates a write access when '0' or a read access when '0' (when User_Out.Strobe_n is '0')
User_Out.CE_n	1	Input	User interface to the clock enable; gives user access to conserve power by turning off the clock inside the memory.

The Mem32_Std_IF component has several generics that can be set by the user upon instantiation of the component:

- **INFF_DELAY**: The INFF_DELAY generic controls whether a delay will be added to the input Flip-Flop. When set to NODELAY, the input signal bypasses the delay. When INFF_DELAY is set to DELAY, the input signal goes through the delay circuit before it is routed out of the pad. The default setting is NODELAY.

- **OBUF_DRIVE**: The OBUF_DRIVE generic controls the output drive strength of the pad. There are several drive strengths associated with this

generic. Note: if a differential signal is selected in the generic BUFFER_TYPE, is generic is not used.

- **HALF_POPULATED**: The half populated generic should never be instantiated by the user. This generic is used so the memory interface is compatible between FIREBIRD™/ PCI and FIREBIRD™/ PMC. The default and only condition for this generic is TRUE.

7.3 VHDL Models Specific to FIREBIRD™/PCI

7.3.1 Clock Standard Interface (Clock_Std_IF)

The PE Clocks_Std_IF is a VHDL component that provides the user interface to the various clock pads and other clock-related information in the VHDL design. The port signals of the Clock_Std_IF are shown in Table 7-5. The "Pads" signals of the Clock_Std_IF component are only used to connect the Clock_Std_IF component to the pads of PE0. The "User_In" and "User_Out" port signals of the Clock_Std_IF component should be used to receive clock sources, receive status information from the DLLs, reset the DLLs manually, and send clock information to the selectable clock interface for the design.

Table 7-5: Clock Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads.Clocks.M_Clk	1	Input	Memory clock pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
Pads.Clocks.K_Clk	1	Input	PCI/LAD bus clock pad signal; always either 33 or 66 MHz.
Pads.Clocks.U_Clk	1	Input	User clock pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.M_Clk	1	Output	User interface to the M_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.K_Clk	1	Output	User interface to the K_Clk pad signal; either 33 or 66 MHz.
User_In.U_Clk	1	Output	User interface to the U_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.M_Clk_2x	1	Output	User interface to the clock doubled version of the M_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.K_Clk_2x	1	Output	User interface to the clock doubled version of the K_Clk pad signal; always 66 MHz when active.
User_In.U_Clk_2x	1	Output	User interface to the clock doubled version of the U_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.M_Clk_Locked	1	Output	M_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_In.K_Clk_Locked	1	Output	K_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_In.U_Clk_Locked	1	Output	U_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_Out.M_Clk_DllRst	1	Input	User interface which allows manual reset of the M_Clk DLL.
User_Out.K_Clk_DllRst	1	Input	User interface which allows manual reset of the K_Clk DLL.
User_Out.U_Clk_DllRst	1	Input	User interface which allows manual reset of the U_Clk DLL.

The Clock_Std_IF component also has several generics that can be set by the user upon instantiation of the component:

- **M_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **U_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **M_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.M_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **K_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.K_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.K_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.K_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **U_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component.

When set to NONE, the User_In.U_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **M_CLK_USE_BUFG**: When set to TRUE, the output of the M_CLK DLL's CLK0, CLK2x or both will be connected to a BUFG. When set to FALSE, the output M_CLK DLL will not be routed to a DLL, but will be routed into the design, if this clock net is used. The default setting is TRUE.

- **U_CLK_USE_BUFG**: When set to TRUE, the output of the U_CLK DLL's CLK0, CLK2x or both will be connected to a BUFG. When set to FALSE, the output U_CLK DLL will not be routed to a DLL, but will be routed into the design, if this clock net is used. The default setting is TRUE.



CAUTION

When the BOTH option is used for the CLK_DLL_OUT generics, two BUFGs are used by that clock. There are four BUFGS available in the Xilinx® part. If the BOTH option is selected, then only two rather than three other BUFGs are available.



CAUTION

The clock doubled output is only available if the low frequency DLL is selected for that clock. The USE_2x and BOTH options are not available if CLKDLLHF is selected for DLL_TYPE.

7.4 VHDL Models Specific to FIREBIRD™/PMC (WSDP™ and G-Link)

7.4.1 Clock Standard Interface (Clock_Std_IF)

The PE Clocks_Std_IF is a VHDL component that provides the user interface to the various clock pads and other clock-related information in the VHDL design. The port signals of the Clock_Std_IF are shown in Table 7-6. The "Pads" signals of the Clock_Std_IF component are only used to connect the Clock_Std_IF component to the pads of PE0. The "User_In" and "User_Out" port signals of the Clock_Std_IF component should be used to receive clock sources, receive status information from the DLLs, reset the DLLs manually, and send clock information to the selectable clock interface for the design.

Table 7-6: Clock Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads.Clocks.RACE_Clk	1	Input	RACEway Clock input from backplane_std_if. This takes the RACEway clock and sends it through a DLL.
Pads.Clocks.M_Clk_Src	1	Output	M Clock source which will be provided to the on-board PLL
Pads.Clocks.U_Clk_Src	1	Output	U Clock source which will be provided to the on-board PLL
User_In.Race_Clk	1	Output	User interface to the RACEway pad signal.
User_In.RACE_Clk_2x	1	Output	User interface to the doubled RACEway pad signal.
User_In.Race_Clk_Locked	1	Output	Race_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_Out.RACE_Clk_DllRst	1	Input	User interface which allows manual reset of the RACE_Clk DLL.
User_Out.M_Clk_Src	1	Input	User interface the Pads.Clocks.M_Clk_Src signal
User_Out.U_Clk_Src	1	Input	User interface the Pads.Clocks.U_Clk_Src signal
Pads.Clocks.M_Clk	1	Input	Memory clock pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
Pads.Clocks.K_Clk	1	Input	PCI/LAD bus clock pad signal; always either 33 or 66 MHz.
Pads.Clocks.U_Clk	1	Input	User clock pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.M_Clk	1	Output	User interface to the M_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.K_Clk	1	Output	User interface to the K_Clk pad signal; either 33 or 66 MHz.
User_In.U_Clk	1	Output	User interface to the U_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.M_Clk_2x	1	Output	User interface to the clock doubled version of the M_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.K_Clk_2x	1	Output	User interface to the clock doubled version of the K_Clk pad signal; always 66 MHz when active.
User_In.U_Clk_2x	1	Output	User interface to the clock doubled version of the U_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.

Signal Name	Width	Dir	Description
User_In.M_Clk_Locked	1	Output	M_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_In.K_Clk_Locked	1	Output	K_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_In.U_Clk_Locked	1	Output	U_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_Out.M_Clk_DllRst	1	Input	User interface which allows manual reset of the M_Clk DLL.
User_Out.K_Clk_DllRst	1	Input	User interface which allows manual reset of the K_Clk DLL.
User_Out.U_Clk_DllRst	1	Input	User interface which allows manual reset of the U_Clk DLL.

The Clock_Std_IF component also has several generics that can be set by the user upon instantiation of the component:

- **RACE_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **M_CLK_SOURCE:** When set to OSCILLATOR, a 0 is routed out the M_Clk_Src pad. When set to WSDP_A, WSDPA_Clk is routed out the M_Clk_Src pad. When set to WSDP_B, WSDPB_Clk is routed out the M_Clk_Src pad.

- **U_CLK_SOURCE**: When set to OSCILLATOR, a 0 is routed out the U_Clk_Src pad. When set to WSDP_A, WSDPA_Clk is routed out the U_Clk_Src pad. When set to WSDP_B, WSDPB_Clk is routed out the U_Clk_Src pad.

- **RACE_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.RACE_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.RACE_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.RACE_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **RACE_CLK_USE_BUFG**: When set to TRUE, the output of the RACE_CLK DLL's CLK0, CLK2x or both will be connected to a BUFG. When set to FALSE, the output RACE_CLK DLL will not be routed to a DLL, but will be routed into the design, if this clock net is used. The default setting is FALSE.

- **M_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **U_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **M_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.M_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **K_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.K_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.K_Clk signal is

driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.K_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **U_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.U_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **M_CLK_USE_BUFG**: When set to TRUE, the output of the M_CLK DLL's CLK0, CLK2x or both will be connected to a BUFG. When set to FALSE, the output M_CLK DLL will not be routed to a DLL, but will be routed into the design, if this clock net is used. The default setting is TRUE.

- **U_CLK_USE_BUFG**: When set to TRUE, the output of the U_CLK DLL's CLK0, CLK2x or both will be connected to a BUFG. When set to FALSE, the output U_CLK DLL will not be routed to a DLL, but will be routed into the design, if this clock net is used. The default setting is TRUE.



CAUTION

When the BOTH option is used for the CLK_DLL_OUT generics, two BUFGs are used by that clock. There are four BUFGS available in the Xilinx® part. If the BOTH option is selected, then only two rather than three other BUFGs are available.



CAUTION

The clock doubled output is only available if the low frequency DLL is selected for that clock. The USE_2x and BOTH options are not available if CLKDLLHF is selected for DLL_TYPE.

7.4.2 Backplane Standard Interface (Backplane_Std_IF)

The Backplane_Std_IF component provides the user interface to the bi-directional connections to the FIREBIRD™/PMC P14 connector. The Backplane interface is 64 bits wide and is presented to the user as a synchronous input and output port (with Enable). The backplane_Std_IF component has several pad drive levels, including two differential modes.

The "User_In" and "User_Out" port signals, described in Table 7-7, should be used by the design to receive data from and send data to the I/O connector. The "Pads" signals of the Backplane_Std_IF component are only used to connect the Backplane_Std_IF component to the pads the VME backplane.

Table 7-7: Backplane Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Clk	64	Input	User selectable clock. Drives input and output IOB Registers.
Clk_x2	1	Input	User selectable doubled clock. Used when a clock is driven out the Backplane at rate, i.e. not halved.
Global_Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARTUP_Virtex component
Pads	64	Bi-dir	Backplane pad signals
User_In.Data_In	64	Output	User interface to the backplane data input
User_Out.Data_Out	64	Input	User interface to the backplane connector data output
User_Out.Data_OE_n	64	Input	User interface to the backplane connector data output enables; each data bit driver is enabled when its respective output enable is '0' or disabled (for reading) when '1'

If the buffer type is set to a differential mode, i.e. buffer type is set to LVDS or LVPECL, the backplane_Std_IF is modified slightly. In LVDS mode, there are only 31 differential pairs. The inputs, outputs, and output enables are only valid on bits 0 through 30; the upper 33 bits are not connected. The backplane_Std_IF takes the lower bits and makes them differential. The same is true for LVPECL, except that 32 differential pairs are available instead of 31. This difference is due to fact that the upper two bits are on non-voltage selectable banks of the PE.

CAUTION

If LVDS is selected, there are only 31 differential pairs (0 – 30) available. If LVPECL is selected, there are 32 differential pairs (0 – 31) available.

The Backplane_Std_IF component also has several generics that can be set by the user upon instantiation of the component:

- **INFF_DELAY**: The INFF_DELAY generic controls whether a delay will be added to the input Flip-Flop. When set to NODELAY, the input signal bypasses the delay. When INFF_DELAY is set to DELAY, the input signal goes through the delay circuit before it is routed out of the pad. The default setting is NODELAY.

- **OBUF_DRIVE:** The OBUF_DRIVE generic controls the output drive strength of the pad. There are several drive strengths associated with this generic. *Note*: if a differential signal is selected in the generic BUFFER_TYPE, is generic is not used.

- **CLK_OUTPUT_EN:** The CLK_OUTPUT_EN generic is used when the user desires an output clock be driven onto the VME backplane. This generic is associated with three other generics, DIV_XMIT_CLOCK, XMIT_CLK_BIT, and INVERT_CLK. When set to true a clock will be transmitted out of the backplane_Std_IF. When set to FALSE, only data will be transmitted out of the interface. The default setting is FALSE.

- **DIV_XMIT_CLOCK:** This generic is used when the clock which is being transmitted onto the backplane needs to be divided by two. When set to TRUE, the clock sent in on the input specified by the XMIT_CLK_BIT generic will be divided by two. When set to FALSE, the clock specified by the XMIT_CLK_BIT will be sent out "as is." If a differential clock is being transmitted and the DIV_XMIT_CLK generic is set to FALSE, the user must pass a 2x clock into the interface on the Clk_x2 input.

- **XMIT_CLK_BIT:** The XMIT_CLK_BIT generic determines which bit of the backplane interface, a clock will be sent out. The XMIT_CLK_BIT generic is an integer which must be between 1 and 30.

- **INVERT_CLK:** When set to TRUE, the clock which is being sent out will be inverted. When set to FALSE, the clock being sent out will not be inverted.

- **BUFFER_TYPE:** The BUFFER_TYPE generic determines the output pad drive level. *Note*: if LVDS is selected, there are only 31 pairs of differential signals. If LVPECL is selected, there are 32 differential pairs.

7.5 VHDL Models Specific to FIREBIRD™/PMC Dual WSDP™ Only

INFORMATION NOTE

In this section, WSDP™ ports A and B correspond respectively to ports labeled WSDP 0 and WSDP 1 in other sections of this manual.

INFORMATION NOTE

The definitions and locations of the FIREBIRD™/PMC Dual WSDP™ PE0 pads are contained in Attachment A of this manual.

7.5.1 FIREBIRD™/PMC Dual WSDP™ PE0 Model

7.5.1.1 Clock Standard Interface (Clock_Std_IF)

The PE Clocks_Std_IF is a VHDL component that provides the user interface to the various clock pads and other clock-related information in the VHDL design. The port signals of the Clock_Std_IF are shown in Table 7-8. The "Pads" signals of the Clock_Std_IF component are only used to connect the Clock_Std_IF component to the pads of PE0. The "User_In" and "User_Out" port signals of the Clock_Std_IF component should be used to receive clock sources, receive status information from the DLLs, reset the DLLs manually, and send clock information to the selectable clock interface for the design.

Table 7-8: Clock Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads.Clocks.WSDPA_Clk	1	Input	WSDP Clock A from the BPE
Pads.Clocks.WSDPB_Clk	1	Input	WSDP Clock B from the BPE
User_In.WSDPA_Clk	1	Output	User interface to the WSDP A clock signal
User_In.WSDPB_Clk	1	Output	User interface to the WSDP B clock signal.
User_In.WSDPA_Clk_2x	1	Output	User interface to the doubled WSDP A clock signal
User_In.WSDPB_Clk_2x	1	Output	User interface to the doubled WSDP B clock signal
User_In.WSDPA_Clk_Locked	1	Output	WSDPA_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_In.WSDPB_Clk_Locked	1	Output	WSDPB_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_Out.WSDPA_Clk_DllRst	1	Input	User interface which allows manual reset of the WSDPA_Clk DLL.

Signal Name	Width	Dir	Description
User_Out.WSDPB_Clk_DllRst	1	Input	User interface which allows manual reset of the WSDPB_Clk DLL.
Pads.Clocks.M_Clk	1	Input	Memory clock pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
Pads.Clocks.K_Clk	1	Input	PCI/LAD bus clock pad signal; always either 33 or 66 MHz.
Pads.Clocks.U_Clk	1	Input	User clock pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.M_Clk	1	Output	User interface to the M_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.K_Clk	1	Output	User interface to the K_Clk pad signal; always 33 or 66 MHz.
User_In.U_Clk	1	Output	User interface to the U_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.M_Clk_2x	1	Output	User interface to the clock doubled version of the M_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.K_Clk_2x	1	Output	User interface to the clock doubled version of the K_Clk pad signal; always 66 MHz when active.
User_In.U_Clk_2x	1	Output	User interface to the clock doubled version of the U_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.M_Clk_Locked	1	Output	M_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_In.K_Clk_Locked	1	Output	K_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_In.U_Clk_Locked	1	Output	U_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_Out.M_Clk_DllRst	1	Input	User interface which allows manual reset of the M_Clk DLL.
User_Out.K_Clk_DllRst	1	Input	User interface which allows manual reset of the K_Clk DLL.
User_Out.U_Clk_DllRst	1	Input	User interface which allows manual reset of the U_Clk DLL.

- **WSDPA_CLK_USE_BUFG**: When set to TRUE, a the output of the WSDPA_CLK DLL's CLK0, CLK2x or both will be connected to a BUFG. When set to FALSE, the output WSDPA_CLK DLL will not be routed to a DLL, but will be routed into the design, if this clock net is used. The default setting is FALSE.

- **WSDPB_CLK_USE_BUFG**: When set to TRUE, the output of the WSDPB_CLK DLL's CLK0, CLK2x or both will be connected to a BUFG. When set to FALSE, the output WSDPB_CLK DLL will not be routed to a DLL, but will be routed into the design, if this clock net is used. The default setting is FALSE.

- **WSDPA_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.WSDPA_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.WSDPA_Clk signal is driven by the CLK0 output of a low

frequency CLKDLL component. When set to NONE, the User_In.WSDPA_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **WSDPB_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.WSDPB_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.WSDPB_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.WSDPB_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **WSDPA_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **WSDPB_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

The Clock_Std_IF component also has several generics that can be set by the user upon instantiation of the component:

- **M_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **U_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **M_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.M_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **K_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.K_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.K_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.K_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **U_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.U_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **M_CLK_USE_BUFG**: When set to TRUE, the output of the M_CLK DLL's CLK0, CLK2x or both will be connected to a BUFG. When set to FALSE, the output M_CLK DLL will not be routed to a DLL, but will be routed into the design, if this clock net is used. The default setting is TRUE.

- **U_CLK_USE_BUFG**: When set to TRUE, the output of the U_CLK DLL's CLK0, CLK2x or both will be connected to a BUFG. When set to FALSE, the output U_CLK DLL will not be routed to a DLL, but will be routed into the design, if this clock net is used. The default setting is TRUE.



CAUTION
When the BOTH option is used for the CLK_DLL_OUT generics, two BUFGs are used by that clock. There are four BUFGS available in the Xilinx® part. If the BOTH option is selected, then only two rather than three other BUFGs are available.



CAUTION
The clock doubled output is only available if the low frequency DLL is selected for that clock. The USE_2x and BOTH options are not available if CLKDLLHF is selected for DLL_TYPE.

7.5.1.2 FIREBIRD™/PMC Dual WSDP™ Standard Interface (WSDP_Std_IF)

The WSDP™ ports include the handshake line and the control and data bus pairs. These signals are directly connected between the BPE and PE0. There are two WSDP™ ports, WSDPA and WSDPB. The port signals of the WSDP™ standard interface are described in Table 7-9. The "User_In" and "User_Out" port signals should be used in the design to receive data from

and send data to the WSDP™ BPE port, respectively. The "x" in the table below should be replaced with "A" or "B" depending upon which component is instantiated.

Table 7-9: PMC Dual WSDP™ Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Clk	1	Input	User selectable clock. Drives input and output IOB Registers.
Global_Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARTUP_Virtex component.
Pads.WSDPx.Data	32	Bi-dir	WSDP™ Data pads signals.
Pads.WSDPx.Ctrl	10	Bi-dir	WSDP™ Control pads signals.
Pads.WSDPx.HS	1	Bi-dir	WSDP™ Handshake pad signal.
User_In.Data_In	32	Output	User interface to the WSDP™ data input
User_Out.Data_Out	32	Input	User interface to the WSDP™ data output
User_Out.Data_OE_n	32	Input	User interface to the WSDP™ data output enables; each data bit driver is enabled when its respective output enable is '0' or disabled (for reading) when '1'
User_In.Ctrl_In	10	Output	User interface to the WSDP™ control input
User_Out.Ctrl_Out	10	Input	User interface to the WSDP™ control output
User_Out.Ctrl_OE_n	10	Input	User interface to the WSDP™ control output enables; each control bit driver is enabled when its respective output enable is '0' or disabled (for reading) when '1'
User_In.HS_In	1	Output	User interface to the WSDP™ handshake input
User_Out.HS_Out	1	Input	User Interface to the WSDP™ handshake output
User_Out.HS_OE_n	1	Input	User interface to the WSDP™ handshake output enables; each data bit driver is enabled when its respective output enable is '0' or disabled (for reading) when '1'

7.5.2 FIREBIRD™/PMC Dual WSDP™ BPE Model

The Bridge Processing Element (BPE) is connected to the main Processing Element (PE0). This section describes the characteristics of the BPE.

7.5.2.1 FIREBIRD™/PMC – BPE Pad Definitions and Locations

The definitions and locations of the BPE pad locations are contained in Attachment B. This information is also contained in the BPE user constraints file (bpe.ucf) which is used by the Xilinx® Alliance tools to place the BPE pads in the proper locations.



INFORMATION NOTE

The pad index corresponds to the logical pad index of the BPE VHDL model, not the physical I/O pin number of the FPGA die.

7.5.2.2 Clock Standard Interface (Clock_Std_IF)

The Clock_Std_IF is a VHDL component that provides a user interface to the various clock pads and other clock-related information in the VHDL design. The port signals of the Clock_Std_IF are shown in the Table 7-10. The "User_In" port signals should be used as the various clock signals in the design.

Table 7-10: BPE Clock Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads.Clocks.M_Clk	1	Input	Memory clock pad signal; source can be selected from a user programmable clock, or one of the I/O cards.
Pads.Clocks.U_Clk	1	Input	User clock pad signal; source can be selected from a user programmable clock, or one of the I/O cards.
Pads.Clocks.W_Clk_A	1	Input	WSDP™ port A clock signal.
Pads.Clocks.W_Clk_B	1	Input	WSDP™ port B clock signal.
Pads.Clocks.WSDPA_Clk_Out	1	Output	External Clock pad which can be used to source the WSDP™ and WILDSTAR™ board to an incoming clock signal.
Pads.Clocks.WSDPB_Clk_Out	1	Output	External Clock pad which can be used to source the WSDP™ and WILDSTAR™ board to an incoming clock signal.
User_In.M_Clk	1	Output	User interface to the M_Clk pad signal; source can be selected from a user programmable clock, or one of the I/O cards.
User_In.U_Clk	1	Output	User interface to the U_Clk pad signal; source can be selected from a user programmable clock, or one of the I/O cards.
User_In.W_Clk_A	1	Output	User interface to the W_Clk_A pad signal.
User_In.W_Clk_B	1	Output	User interface to the W_Clk_B pad signal.
User_In.M_Clk_2x	1	Output	Doubled user interface to the M_Clk pad signal; source can be selected from a user programmable clock, or one of the I/O cards.
User_In.U_Clk_2x	1	Output	Doubled user interface to the U_Clk pad signal; source can be selected from a user programmable clock, or one of the I/O cards.
User_In.W_Clk_A_2x	1	Output	Doubled user interface to the W_Clk_A pad signal.
User_In.W_Clk_B_2x	1	Output	Doubled user interface to the W_Clk_B pad signal.
User_In.M_Clk_Locked	1	Output	M_Clk DLL lock flag; indicates DLL has achieved a lock when '1'.
User_In.U_Clk_Locked	1	Output	U_Clk DLL lock flag; indicates DLL has achieved a lock when '1'.
User_In.W_Clk_A_Locked	1	Output	W_Clk_2 DLL lock flag; indicates DLL has achieved a lock when '1'.
User_In.W_Clk_B_Locked	1	Output	W_Clk_1 DLL lock flag; indicates DLL has achieved a lock when '1'.
User_Out.WSDPA_Clk_Out	1	Output	Clock from I/O Connector A which will be sent down to the PE.
User_Out.WSDPB_Clk_Out	1	Output	Clock from I/O Connector B which will be sent down to the PE.
User_Out.M_Clk_DllRst	1	Input	User interface which allows manual reset of the M_Clk DLL.
User_Out.U_Clk_DllRst	1	Input	User interface which allows manual reset of the K_Clk DLL.
User_Out.W_Clk_A_DllRst	1	Input	User interface which allows manual reset of the U_Clk_0 DLL.
User_Out.W_Clk_B_DllRst	1	Input	User interface which allows manual reset of the U_Clk_1 DLL.

Signal Name	Width	Dir	Description
Pads.Clocks.M_Clk	1	Input	Memory clock pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
Pads.Clocks.U_Clk	1	Input	User clock pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.M_Clk	1	Output	User interface to the M_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.U_Clk	1	Output	User interface to the U_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.M_Clk_2x	1	Output	User interface to the clock doubled version of the M_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.U_Clk_2x	1	Output	User interface to the clock doubled version of the U_Clk pad signal; source can be selected from a user programmable clock, one of the I/O cards, or the VME backplane.
User_In.M_Clk_Locked	1	Output	M_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_In.U_Clk_Locked	1	Output	U_Clk DLL lock flag; indicates DLL has achieved a lock when '1.'
User_Out.M_Clk_DllRst	1	Input	User interface which allows manual reset of the M_Clk DLL.
User_Out.U_Clk_DllRst	1	Input	User interface which allows manual reset of the U_Clk DLL.

The Clock_Std_IF component also has several generics that can be set by the user upon instantiation of the component:

- **U_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **M_CLK_DLL_OUT:** When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected.

When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **W_CLK_A_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **W_CLK_B_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **M_CLK_SRC**: When set to OSCILLATOR, the User_In.M_Clk signal is driven by the motherboard oscillator. When set to W_CLK_2, the User_In.M_Clk signal is driven by the external clock received on WSDP™ Port 2. The default setting is OSCILLATOR.

- **U_CLK_SRC**: When set to OSCILLATOR, the User_In.U_Clk signal is driven by the motherboard oscillator. When set to W_CLK_1, the User_In.U_Clk signal is driven by the external clock received on WSDP™ port 1. The default setting is OSCILLATOR.

- **W_CLK_A_BUFFER_TYPE**: This controls the output buffer type for the WSDP™ Port A clock signal.

- **W_CLK_B_BUFFER_TYPE**: This controls the output buffer type for the WSDP™ port 1 clock signal.

- **M_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.M_Clk signal is driven directly by the clock pad signal (i.e. the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **U_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.U_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **W_CLK_2_DLL_TYPE**: When set to HIGH_FREQ, the User_In.W_Clk_2 signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.W_Clk_2 signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.W_Clk_2 signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.
- **W_CLK_1_DLL_TYPE**: When set to HIGH_FREQ, the User_In.W_Clk_1 signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.W_Clk_1 signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.W_Clk_1 signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.
- **DIV_Ext_Clk**: When Set to True, the external clock is halved before it is sent down to the motherboard. When set to False, the external clock is sent directly down to the motherboard.

- **M_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x

output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **U_CLK_DLL_OUT**: When set to USE_1x, the CLK0 output of the DLL is routed to the Global Clock Buffer (BUFG). The output of the BUFG is routed back into the CLKFB input of the DLL. When set to USE_2x, the CLK2X output of the DLL is routed to the BUFG. The output of the BUFG is routed back into the CLKFB input of the DLL. When set to both, the CLK0 output and the CLK2X output of the DLL are routed to BUFGs. The output of the BUFG connected to the CLK2X output is routed back into the CLKFB input of the DLL.* When the 1x output is selected, the doubled output of the DLL will be disconnected. When the 2x output is selected, the normal 1x output of the DLL will be disconnected. The default setting is USE_1x.

- **M_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.M_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **U_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.U_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **M_CLK_USE_BUFG**: When set to TRUE, the output of the M_CLK DLL's CLK0, CLK2x or both will be connected to a BUFG. When set to FALSE, the output M_CLK DLL will not be routed to a DLL, but will be routed into the design, if this clock net is used. The default setting is TRUE.

- **U_CLK_USE_BUFG**: When set to TRUE, the output of the U_CLK DLL's CLK0, CLK2x or both will be connected to a BUFG. When set to FALSE, the output U_CLK DLL will not be routed to a DLL, but will be

routed into the design, if this clock net is used. The default setting is TRUE.

CAUTION

When the BOTH option is used for the CLK_DLL_OUT generics, two BUFGs are used by that clock. There are four BUFGS available in the Xilinx® part. If the BOTH option is selected, then only two rather than three other BUFGs are available.

CAUTION

The clock doubled output is only available if the low frequency DLL is selected for that clock. The USE_2x and BOTH options are not available if CLKDLLHF is selected for DLL_TYPE.

7.5.2.3 WSDP™ Standard Interface (WSDP_Std_IF)

The WSDP™ ports include the handshake line and the control and data bus pairs. These signals are directly connected between the BPE and PE0. There are two WSDP™ ports, WSDPA and WSDPB.

INFORMATION NOTE

WSDP™ ports A and B correspond respectively to ports labeled WSDP 0 and WSDP 1 in other sections of this manual.

The port signals of the WSDP™ standard interface are described in Table 7-11. The "User_In" and "User_Out" port signals should be used in the design to receive data from and send data to the WSDP™ BPE port, respectively. The "x" in the table below should be replaced with "A" or "B", depending upon which component is instantiated.

Table 7-11: WSDP™ Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Clk	1	Input	User selectable clock. Drives input and output IOB Registers.
Global_Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARTUP_Virtex component.
Pads.WSDPx.Data	32	Bi-dir	WSDP™ Data pads signals.
Pads.WSDPx.Ctrl	10		WSDP™ Control pads signals.
Pads.WSDPx.HS	1		WSDP™ Handshake pad signal.
User_In.Data_In	32	Output	User interface to the WSDP™ data input
User_Out.Data_Out	32	Input	User interface to the WSDP™ data output
User_Out.Data_OE_n	32	Input	User interface to the WSDP™ data output enables; each data bit driver is enabled when its respective output enable is '0' or disabled (for reading) when '1'
User_In.Ctrl_In	10	Output	User interface to the WSDP™ control input
User_Out.Ctrl_Out	10	Input	User interface to the WSDP™ control output
User_Out.Ctrl_OE_n	10	Input	User interface to the WSDP™ control output enables; each control bit driver is enabled when its respective output enable is '0' or disabled (for reading) when '1'
User_In.HS_In	1	Output	User interface to the WSDP™ handshake input
User_Out.HS_Out	1	Input	User Interface to the WSDP™ handshake output
User_Out.HS_OE_n	1	Input	User interface to the WSDP™ handshake output enables; each data bit driver is enabled when its respective output enable is '0' or disabled (for reading) when '1'

The WSDP_Std_IF component has generics that can be set by the user upon instantiation of the component:

- **V25_ENABLED**: This generic is set to true when the I/O connector, interface BUFFER_TYPE, is set to a drive level which required 2.5V; otherwise this generic should be set to false. If I/O Connector A is set to a drive level which requires 2.5V only WSDP_A needs to be set to true. Likewise if I/O connector B is set to a drive level which required 2.5V only WSDP_B needs to be set to TRUE. The default value is FALSE.

- **PORT_SEL**: This generic is set to PORTA when the WSDP_A component is instantiated. The generic is set to PORTB when the WSDP_B component is instantiated. These generics are preset in the BPE template file.

7.5.2.4 IO_Conn Standard Interface (IO_Conn_Std_IF)

The IO_Conn is a bi-directional interface between the BPE devices of the PMC Dual WSDP™ card and the external WSDP™ ports. The IO_Conn bus provides a 38-bit communication path between the BPE device and the external I/O WSDP™ port. The port signals of the IO_Conn bus standard interface are described in Table 7-12. The "User_In" and "User_Out" port signals should be used in the design to receive data from and send data to the IO Connectors, respectively.

Table 7-12: IO_Conn Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads	38	Bi-dir	IO_Conn bus pad signals
User_In.Data_In	38	Output	User interface to the IO Conn bus data input
User_Out.Data_Out	38	Input	User interface to the IO Conn bus data output
User_Out.Data_OE_n	38	Input	User interface to the IO Conn bus data output enables; each data bit driver is enabled when its respective output enable is '0' or disabled (for reading) when '1'

The IO_Conn_Std_IF component has several generics that can be set by the user upon instantiation of the component:

- **CLK_OUTPUT_EN:** The CLK_OUTPUT_EN generic is used when the user desires an output clock be driven onto the VME backplane. This generic is associated with three other generics, DIV_XMIT_CLOCK, XMIT_CLK_BIT, and INVERT_CLK. When set to true a clock will be transmitted out of the backplane_Std_IF. When set to false only data will be transmitted out of the interface. The default setting is FALSE.

- **DIV_XMIT_CLOCK:** This generic is used when the clock which is being transmitted onto the backplane needs to be divided by two. When set to true the clock sent in on the input specified by the XMIT_CLK_BIT generic will be divided by two. When set to false the clock specified by the XMIT_CLK_BIT will be sent out as is. If a differential clock is being transmitted and the DIV_XMIT_CLK generic is set to false, the user must pass a 2x clock into the interface on the Clk_x2 input.
- **XMIT_CLK_BIT:** The XMIT_CLK_BIT generic determines which bit of the backplane interface, a clock will be sent out. The XMIT_CLK_BIT generic is an integer which must be between 1 and 30.

- **INVERT_CLK:** When set to true, the clock which is being sent out will be inverted. When set to false, the clock, which is being sent out will not be inverted.

- **BUFFER_TYPE:** The BUFFER_TYPE generic determines the output pad drive level. There are several types of buffer types available. *Note*: if LVDS is selected there are only 31 pairs of differential signals. If LVPECL is selected there are 32 differential pairs.

- **DIFF_TERM**: The DIFF_TERM generic is used when a differential BUFFER_TYPE is selected; otherwise this generic is not used. Each bit

controls the termination of the individual data bit being transmitted or received.

- **WSDP_TERM**: The WSDP_TERM generic is used when a non-differential BUFFER TYPE is selected; otherwise this generic is not used. Please refer to Table 7-13 for individual bit control.

Table 7-13: WSDP™ Termination Generics

WSDP™ Termination Generic	Termination Control
WSDP_TERM(0)	Byte termination enable for Data Bits 0-7
WSDP_TERM(1)	Byte termination enable for Data Bits 8 - 15
WSDP_TERM(2)	Byte termination enable for Data Bits 16 – 23
WSDP_TERM(3)	Byte termination enable for Data Bits 24 – 31
WSDP_TERM(4-7)	Bit termination enable for Data Bits 34-37
WSDP_TERM(8)	Bit termination enable for WSDP™ Clock Out
WSDP_TERM(9)	Bit termination enable for WSDP™ Clock In

7.5.2.5 LED Standard Interface (LED_Std_IF)

The LED_Std_IF is a VHDL component that provides a user interface to two LEDs. The User LED can always be used to indicate status or processing activity, and is useful in debugging a Dual WSDP™ card design. The WSDP™ LEDs are also available to indicate status and processing activity. The port signals of the LED_Std_IF are shown in Table 7-14 below. The "User_Out" signals should be used to drive the LEDs of the PMC Dual WSDP™ card.

Table 7-14: LED Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads	1	Output	LED pad signals
User_Out	1	Input	User interface to the LED

The WSDP_Std_IF component has generics that can be set by the user upon instantiation of the component:

- **V25_ENABLED_PORTx**: This generic is set to true when the I/O connector interface, BUFFER_TYPE, is set to a drive level which required 2.5V; otherwise this generic should be set to FALSE. If I/O Connector A is set to a drive level which requires 2.5V only WSDP_A needs to be set to true. Likewise, if I/O connector B is set to a drive level which required 2.5V, only WSDP_B needs to be set to TRUE. The default value is FALSE.

7.6 VHDL Models Specific to FIREBIRD™/PMC Dual G-Link Only

7.6.1 Clock Standard Interface (Clock_Std_IF)

The PE Clock_Std_IF is a VHDL component that provides a user interface to the various clock pads and other clock-related information in the VHDL design. The port signals of the Clock_Std_IF are shown in Table 7-15. The "User_In" port signals should be used as the various clock signals in the design.

The Clock_Std_IF component also has several generics that can be set by the user upon instantiation of the component:

- **K_CLK_DLL_OUT**: When set to USE_1x, the User_In.K_Clk signal is driven by the 1x output of the K Clock DLL. User_In.K_Clk_2x is tied low. When set to USE_2x, the User_In.K_Clk_2x signal is driven by the 2x output of the K Clock DLL. User_In.K_Clk is tied low. When set to BOTH, both User_In.K_Clk and User_In.K_Clk_2x are driven by the 1x and 2x K clock DLL outputs respectively.

- **M_CLK_DLL_OUT**: When set to USE_1x, the User_In.M_Clk signal is driven by the 1x output of the M Clock DLL. User_In.M_Clk_2x is tied low. When set to USE_2x, the User_In.M_Clk_2x signal is driven by the 2x output of the M Clock DLL. User_In.M_Clk is tied low. When set to BOTH, both User_In.M_Clk and User_In.M_Clk_2x are driven by the 1x and 2x M clock DLL outputs respectively.

- **U_CLK_DLL_OUT**: When set to USE_1x, the User_In.U_Clk signal is driven by the 1x output of the U Clock DLL. User_In.U_Clk_2x is tied low. When set to USE_2x, the User_In.U_Clk_2x signal is driven by the 2x output of the U Clock DLL. User_In.U_Clk is tied low. When set to BOTH, both User_In.U_Clk and User_In.U_Clk_2x are driven by the 1x and 2x U Clock DLL outputs respectively.

- **SMA_CLK_DLL_OUT**: When set to USE_1x, the User_In.SMA_Clk signal is driven by the 1x output of the SMA Clock DLL. User_In.SMA_Clk_2x is tied low. When set to USE_2x, the User_In.SMA_Clk_2x signal is driven by the 2x output of the SMA Clock DLL. User_In.SMA_Clk is tied low. When set to BOTH, both User_In.SMA_Clk and User_In.SMA_Clk_2x are driven by the 1x and 2x SMA Clock DLL outputs respectively.

- **TRIG_CLK_DLL_OUT**: When set to USE_1x, the User_In.Trig_Clk signal is driven by the 1x output of the Trig Clock DLL. User_In.Trig_Clk_2x is tied low. When set to USE_2x, the User_In.Trig_Clk_2x signal is driven by the 2x output of the Trig Clock DLL. User_In.Trig_Clk is tied low. When set to BOTH, both User_In.Trig_Clk and User_In.Trig_Clk_2x are driven by the 1x and 2x Trig Clock DLL outputs respectively.

- **RACE_CLK_DLL_OUT**: When set to USE_1x, the User_In.RACE_Clk signal is driven by the 1x output of the RACE Clock DLL. User_In.RACE_Clk_2x is tied low. When set to USE_2x, the User_In.RACE_Clk_2x signal is driven by the 2x output of the RACE Clock DLL. User_In.RACE_Clk is tied low. When set to BOTH, both User_In.RACE_Clk and User_In.RACE_Clk_2x are driven by the 1x and 2x RACE Clock DLL outputs respectively.

- **M_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.M_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.M_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **U_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.U_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.U_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **SMA_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.SMA_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.SMA_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.SMA_Clk signal is driven directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **TRIG_CLK_DLL_TYPE**: When set to HIGH_FREQ, the User_In.TRIG_Clk signal is driven by the CLK0 output of a high frequency CLKDLLHF component. When set to LOW_FREQ, the User_In.TRIG_Clk signal is driven by the CLK0 output of a low frequency CLKDLL component. When set to NONE, the User_In.TRIG_Clk signal is driven

directly by the clock pad signal (i.e., the CLKDLL component is bypassed). The default setting is LOW_FREQ.

- **M_CLK_SOURCE:** When set to OSCILLATOR, the User_In.M_Clk and the User_In.M_Clk_2x signals are driven by the motherboard oscillator. When set to SMA_Clk, TRIG_CLK, RX1_CLK, RX3_CLK, or RACE_CLK the User_In.M_Clk and User_In.M_Clk_2x signals are driven by the SMA_Clk, Trig_Clk, or RACE_Clk inputs respectively. The 1x signal is also sent back to the mail board to become the source of M_Clk for the entire board.

- **U_CLK_SOURCE:** When set to OSCILLATOR, the User_In.U_Clk and the User_In.U_Clk_2x signals are driven by the motherboard oscillator. When set to SMA_Clk, TRIG_CLK, or RACE_CLK the User_In.U_Clk and User_In.U_Clk_2x signals are driven by the SMA_Clk, Trig_Clk, or RACE_Clk inputs respectively. This signal is also sent back to the mail board to become the source of U_Clk for the entire board.

- **M_CLK_USE_BUFG**: When set to TRUE, User_In.M_Clk and/or User_In.M_Clk_2x are routed using one of the four the global clocking networks, and corresponding global clock buffer, on the FPGA. If M_CLK_DLL_OUT is set to USE_1x or USE_2x then only one global clock buffer is used. If M_CLK_DLL_OUT is set to BOTH then two global clock buffers are used.

- **U_CLK_USE_BUFG**: When set to TRUE, User_In.U_Clk and/or User_In.U_Clk_2x are routed using one of the four the global clocking networks, and corresponding global clock buffer, on the FPGA. If U_CLK_DLL_OUT is set to USE_1x or USE_2x then only one global clock buffer is used. If U_CLK_DLL_OUT is set to BOTH then two global clock buffers are used.

- **SMA_CLK_USE_BUFG**: When set to TRUE, User_In.SMA_Clk and/or User_In.SMA_Clk_2x are routed using one of the four the global clocking networks, and corresponding global clock buffer, on the FPGA. If SMA_CLK_DLL_OUT is set to USE_1x or USE_2x then only one global clock buffer is used. If SMA_CLK_DLL_OUT is set to BOTH then two global clock buffers are used.

- **TRIG_CLK_USE_BUFG**: When set to TRUE, User_In.TRIG_Clk and/or User_In.TRIG_Clk_2x are routed using one of the four the global clocking networks, and corresponding global clock buffer, on the FPGA. If TRIG_CLK_DLL_OUT is set to USE_1x or USE_2x then only one global

clock buffer is used. If TRIG_CLK_DLL_OUT is set to BOTH then two global clock buffers are used.

- **RACE_CLK_USE_BUFG**: When set to TRUE, User_In.RACE_Clk and/or User_In.RACE_Clk_2x are routed using one of the four the global clocking networks, and corresponding global clock buffer, on the FPGA. If RACE_CLK_DLL_OUT is set to USE_1x or USE_2x then only one global clock buffer is used. If RACE_CLK_DLL_OUT is set to BOTH then two global clock buffers are used.

- **SMA_MODE**: When set to DUAL_INPUT, both User_In.SMA_Clk and User_In.Trig_Clk are driven by the two SMA clock ports. When set to DIFF_INPUT, User_In.SMA_Clk is driven by a differential clock input on the two SMA clock ports and User_In.Trig_Clk is tied low. When set to BI_DIRECTIONAL, User_In.SMA_Clk is driven by the FP CLK SMA port and User_Out.Trig_Clk drives the TRG CLK SMA port.

- **SMA_BUFFER**: Sets the input and output drive strength of the SMA connectors. It also can be set to LVTTL, LVCMOS2, or LVDS.

Table 7-15: Clock Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Pads.Clocks.M_Clk	1	Input	Memory clock pad signal; source can be selected from a user programmable clock, one of the PMC Dual G-Link card clocks.
Pads.Clocks.K_Clk	1	Input	VME/LAD bus clock pad signal; can be either 33 MHz or 66MHz
Pads.Clocks.U_Clk	1	Input	User clock pad signal; source can be selected from a user programmable clock, or one of the PMC Dual G-Link card clocks.
Pads.Clocks.SMA_Clk	1	Input	Pad signal for the FP CLK SMA connector
Pads.Clocks.Trig_Clk	1	Input	Pad signal for the TRG CLK SMA connector
Pads.Clocks.RACE_Clk	1	Input	Raceway Clock off the P2 backplane on VME systems
Pads.Clocks.M_Clk_Src	1	Output	Clock signal sent to the motherboard to become the global source for M_Clk across the board.
Pads.Clocks.U_Clk_Src	1	Output	Clock signal sent to the motherboard to become the global source for U_Clk across the board.
User_In.M_Clk	1	Output	User interface to the M_Clk pad signal; source can be selected from a user programmable clock, or one of the PMC Dual G-Link cards
User_In.K_Clk	1	Output	User interface to the K_Clk pad signal; can be either 33 MHz or 66MHz
User_In.U_Clk	1	Output	User interface to the U_Clk pad signal; source can be selected from a user programmable clock, or one of the PMC Dual G-Link cards
User_In.SMA_Clk	1	Output	User interface to the FP CLK SMA connector pad signal
User_In.Trig_Clk	1	Output	User interface to the TRG CLK SMA connector pad signal
User_In.RACE_Clk	1	Output	User interface to Raceway Clock off the P2 backplane on VME systems

Signal Name	Width	Dir	Description
User_In.M_Clk_2x	1	Output	User interface to the DLL doubled M_Clk pad signal; source can be selected from a user programmable clock, or one of the PMC Dual G-Link cards
User_In.K_Clk_2x	1	Output	User interface to the DLL doubled K_Clk pad signal; can be either 33 MHz or 66MHz
User_In.U_Clk_2x	1	Output	User interface to the DLL doubled U_Clk pad signal; source can be selected from a user programmable clock, or one of the PMC Dual G-Link cards
User_In.SMA_Clk_2x	1	Output	User interface to the DLL doubled FP CLK SMA connector pad signal
User_In.Trig_Clk_2x	1	Output	User interface to the DLL doubled TRG CLK SMA connector pad signal
User_In.RACE_Clk_2x	1	Output	User interface to the DLL doubled Raceway Clock off the P2 backplane on VME systems
User_In.K_Clk_En	1	Output	Clock enable used for PLDs when running 66Mhz K_Clk designs
User_In.M_Clk_Locked	1	Output	M_Clk DLL lock flag; indicates DLL has achieved a lock when '1'
User_In.K_Clk_Locked	1	Output	K_Clk DLL lock flag; indicates DLL has achieved a lock when '1'
User_In.U_Clk_Locked	1	Output	U_Clk DLL lock flag; indicates DLL has achieved a lock when '1'
User_In.SMA_Clk_Locked	1	Output	SMA_Clk DLL lock flag; indicates DLL has achieved a lock when '1'
User_In.RACE_Clk_Locked	1	Output	RACE_Clk DLL lock flag; indicates DLL has achieved a lock when '1'
User_Out.K_Clk_DLLRst	1	Input	Resets the K_Clk DLL when set to '1'
User_Out.M_Clk_DLLRst	1	Input	Resets the M_Clk DLL when set to '1'
User_Out.U_Clk_DLLRst	1	Input	Resets the U_Clk DLL when set to '1'
User_Out.SMA_Clk_DLLRst	1	Input	Resets the SMA_Clk DLL when set to '1'
User_Out.Trig_Clk_DLLRst	1	Input	Resets the Trig_Clk DLL when set to '1'
User_Out.RACE_Clk_DLLRs t	1	Input	Resets the RACE_Clk DLL when set to '1'

7.6.1.1 Dual G-Link Standard Interface (Glink_RX_Std_IF,Glink_TX_Std_IF)

The standard interface to each PMC Dual G-Link card transceiver is broken into two distinct parts: the receiver (typically called RX) and the transmitter (typically called TX). There can be up to two transceivers per PMC Dual G-Link card, and therefore up to two pairs of standard interfaces.

The Glink_RX_Std_IF component is the standard interface to the receiver. This interface has one generic:

- **PASS_Mode**: When set to TRUE, the input flip-flops are clocked on REFCLK. When set to FALSE, the input flip-flops are clocked on RXCLK.

This interface also has five port signals: RX_Clock, Ref_Clock, Global_Reset, Pads, and User_In. These are described in Table 7-16 below.

Table 7-16: PMC Dual G-Link RX Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Global_Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARUP_Virtex component
RX_Clock	1	Output	User interface to the recovered data clock
Ref_Clock	1	Input	User interface to the PASS mode reference clock signal
Pads.Data_In	17	Input	Data pad signals
Pads.Slip	1	Input	Used during PASS mode. Set to '1' if the receiver needed to slip one word to compensate for high phase difference between REFCLK and RXCLK
Pads.Ctrl_Xfer	1	Input	Set to '1' if the word on Pads.Data_In is a control word.
Pads.Data_Xfer	1	Input	Set to '1' if the word on Pads.Data_In is a data word.
Pads.Error	1	Input	Set to '1' if the receiver received a word that was not a valid data, control or idle word.
Pads.Ready	1	Input	Set to '1' if the receiver has received 64 to 128 consecutive valid words.
Pads.RX_Clock	1	Input	Recovered data clock pad signal
Pads.Ref_Clock	1	Output	Reference clock output pad signal
User_In.Data_In	17	Output	User interface to data pads signals
User_In.Slip	1	Output	Used during PASS mode. Set to '1' if the receiver needed to slip one word to compensate for high phase difference between REFCLK and RXCLK
User_In.Ctrl_Xfer	1	Output	Set to '1' if the word on Pads.Data_In is a control word.
User_In.Data_Xfer	1	Output	Set to '1' if the word on Pads.Data_In is a data word.
User_In.Error	1	Output	Set to '1' if the receiver received a word that was not a valid data, control or idle word.
User_In.Ready	1	Output	Set to '1' if the receiver has received 64 to 128 consecutive valid words.

The Glink_TX_Std_IF also has five port signals: Global_Reset, TX_Clock, TX_Locked, Pads, and User_Out. These are described in the table below.

Table 7-17: Interface Port Signals

Signal Name	Width	Dir	Description
Global_Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARUP_Virtex component
TX_Clock	1	Input	User interface to the transmit clock
TX_Locked	1	Output	User interface to Pads.TX_Locked. Set to '1' when the PLL in the transmitter has locked to the transmit clock
Pads.Data_Out	17	Output	Data pad signals
Pads.Ctrl_Xfer	1	Output	Set to '1' to transmit Pads.Data_Out as a control word.
Pads.Data_Xfer	1	Output	Set to '1' to transmit Pads.Data_Out as a data word.
Pads.TX_Locked	1	Input	Set to '1' when the PLL in the transmitter has locked to the transmit clock
User_Out.Data_Out	17	Input	User interface to data pads signals
User_Out.Ctrl_Xfer	1	Output	Set to '1' to transmit Pads.Data_Out as a control word.
User_Out.Data_Xfer	1	Output	Set to '1' to transmit Pads.Data_Out as a data word.

7.6.1.2 Dual G-Link PLD Standard Interface (GLink_PLD_Std_If)

The PMC Dual G-Link PLD standard interface (GLink_PLD_Std_IF) is a VHDL component that provides a user interface to the port expansion PLD from within the processing element.

The Glink_PLD_Std_If has six generics that can be set at compile time.

- **INIT_DATA_CH0**: The channel 0 Control Register will be initialized to these 10 bits when this interface is reset.
- **INIT_DATA_CH1**: The channel 1 Control Register will be initialized to these 10 bits when this interface is reset.
- **INIT_DATA_PASS**: The PASS Mode Control Register will be initialized to these 10 bits when this interface is reset.

The port signals of the GLink_PLD_Std_IF component are described in the table below. The "User_In" and "User_Out" port signals should be used in the design to receive data from and send data to the PMC Dual G-Link PLD, respectively.

Table 7-18: G-Link PLD Standard Interface Component Port Signals

Signal Name	Width	Dir	Description
Clk	1	Input	Serial Data Clock. Should be set to Clocks_In.K_Clk only.
Global_Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARUP_Virtex component
Pads.Data	1	Bi-dir	Data pad to the PMC Dual G-Link PLD
Pads.CS_n	1	Output	Chip Select signal to the PMC Dual G-Link PLD. Must be set to '0' for the duration of all PLD transactions and set to '1' otherwise
Pads.Dir	1	Output	Set to '0' to transmit to the PLD Set to '1' to receive data from the PLD
User_Out.Data_Out	10	Input	10 bits of data to be programmed into the PLD when User_Out.Load_n is strobed low.
User_Out.Reg_Sel	3	Input	Address of the PLD register to read and write from when User_Out.Load_n is strobed low.
User_Out.Load_n	1	Input	When strobed low, the data in User_Out.Data_Out is programmed into the PLD register at the address in User_Out.Reg_Sel. The same PLD register is then read back and stored in the interface.
User_In.Data_CH0_In	10	Output	Locally stored value for PLD register "000"
User_In.Data_CH1_In	10	Output	Locally stored value for PLD register "001"
User_In.Data_PASS_In	10	Output	Locally stored value for PLD register "100"
User_In.Data_PWR_In	10	Output	Locally stored value for PLD register "110"
User_In.Data_CH0_Valid	1	Output	Set to '1' when PLD register "000" has successfully been read back
User_In.Data_CH1_Valid	1	Output	Set to '1' when PLD register "001" has successfully been read back
User_In.Data_PASS_Valid	1	Output	Set to '1' when PLD register "100" has successfully been read back
User_In.Data_PWR_Valid	1	Output	Set to '1' when PLD register "110" has successfully been read back

7.6.1.3 Dual G-Link TX Standard FIFO Interface (GLink_TX_STD_FIFO_IF)

The standard interface to the PMC Dual G-Link transmitter drives its output signals directly user interface signal. This means that all user logic must be clocked with TX Clock

The GLink_TX_Std_FIFO_IF overcomes this restriction by putting a FIFO between the user logic and the pad output signals. In this interface, data is stored temporarily in a 256-word deep block RAM FIFO before going to the pads.

Table 7-19: PMC Dual G-Link TX Standard FIFO Interface Component Port Signals

Signal Name	Width	Dir	Description
Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARUP_Virtex component
TX Clock	1	Input	User interface to the TX clock signal
W_Clock	1	Input	All logic on the user side of the FIFO is clocked on this clock
Pads.Data_Out	17	Output	Data pad signals
Pads.Ctrl_Xfer	1	Output	Set to '1' if the word on Pads.Data_Out is a control word.
Pads.Data_Xfer	1	Output	Set to '1' if the word on Pads.Data_Out is a data word.
Pads.TX_Locked	1	Input	Set to '1' if the receiver received a word that was not a valid data, control or idle word.
User_Out.Data_Out	17	Input	Data received from the Fiber Optic Interface
User_Out.Ctrl_Xfer	1	Input	Set to '1' if the word on Data_Out is a control word
User_Out.Data_Xfer	1	Input	Set to '1' if the word on Data_Out is a data word
User_Out.Push	1	Input	Set to '1' to push User_Out.Data_Out onto the FIFO
User_In.FIFO_Full	1	Input	Set to '1' when the FIFO is full
User_In.FIFO_1Qrtr	1	Input	Set to '1' when the FIFO is at least 1 quarter full
User_In.FIFO_Half	1	Input	Set to '1' when the FIFO is at least half full
User_In.FIFO_3Qrtr	1	Input	Set to '1' when the FIFO is at least 3 quarters full

7.6.1.4 Dual G-Link TX Multiplexing FIFO Interface

The standard interface to the PMC Dual G-Link transmitter limits the number of components that may use a single channel. Therefore, in addition to the standard interface to the PMC Dual G-Link transmitter, there are two corresponding MUX (Multiplexing) interfaces allowing any number of components to share a single transmitting channel. These two transmitter interfaces are the GLink_TX_MUX_FIFO_Fair_IF and the GLink_TX_MUX_FIFO_Priority_IF.

The GLink_TX_MUX_FIFO_Priority_IF has several generics to control the type of arbitration used, as well as other general characteristics of the interface. These generics are:

- **ROTATE_PRIORITY**: This generic may be set to TRUE or FALSE. Setting this to TRUE allows the arbiter to give highest priority to any client on a rotating basis. When this is set to FALSE, the client with the lowest vector index will always have priority.

- **STICKY_PRIORITY**: This generic may be set to TRUE or FALSE, and is only valid when ROTATE_PRIORITY is TRUE. When STICKY_PRIORITY is set to TRUE, any client currently receiving AKKs will continue to have priority until that client deasserts the request line. When set to FALSE, a client that is currently receiving AKKs may stop receiving AKKs if another client, with a higher priority, asserts the request line.

- **REGISTERED_AKKS**: This generic may be set to TRUE or FALSE. When set to TRUE, all AKK signals are registered going out of the interface. This allows for a faster timing, but also halves the transfer bandwidth to the interface for single client systems. When set to FALSE, the AKK signals are generated in combination and may not make high frequency timing.

- **REGISTERED_REQS**: This generic may be set to TRUE or FALSE. When set to TRUE, the request signal is registered going into the interface. This allows for a faster timing, but also halves the transfer bandwidth to the interface for single client systems. When set to FALSE, the request signals are decoded in combination and may not make high frequency timing.

The port interface for the GLink_TX_MUX_FIFO_Priority_IF and the GLink_TX_MUX_FIFO_Fair_IF are identical. These signals are described in Table 7-20 below.

Table 7-20: PMC Dual G-Link Transfer MUX FIFO Interface Component Port Signals

Signal Name	Width	Dir	Description
Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARUP_Virtex component
TX_Clock	1	Input	User interface to the transmit clock. All logic on the PMC Dual G-Link transmitter side of the FIFO is clocked using this clock.

Signal Name	Width	Dir	Description
W_Clock	1	Input	All logic on the Client side of the FIFO is clocked using this clock. It may be completely asynchronous to TX_Clock
Pads.Data_Out	17	Output	Data pad signals
Pads.Ctrl_Xfer	1	Output	Set to '1' to transmit Pads.Data_Out as a control word.
Pads.Data_Xfer	1	Output	Set to '1' to transmit Pads.Data_Out as a data word.
Pads.TX_Locked	1	Input	Set to '1' when the PLL in the transmitter has locked to the transmit clock
Clients(x).Data_Out	17	Output	Data to be sent to the PMC Dual G-Link Transmitter
Clients(x).Ctrl_Xfer	1	Output	Set to '1' to transmit Data_Out as a control word
Clients(x).Data_Xfer	1	Output	Set to '1' to transmit Data_Out as a data word
Clients(x).TX_Locked	1	Output	Set to '1' when the PLL in the transmitter has locked to the transmit clock
Clients(x).Req	1	Input	Set to '1' to request a data transmission
Clients(x).Akk	1	Output	Set to '1' when the interface has accepted the data transmission request.

7.6.1.5 Dual G-Link RX Multiplexing FIFO Interface (GLink_RX_Mux_FIFO_IF)

The standard interface to the PMC Dual G-Link receiver limits the number of components that may use a single channel. Therefore, in addition to the standard interface to the PMC Dual G-Link transmitter there is a corresponding MUX (Multiplexing) interface that allows any number of components to share a single receiving channel. Unlike the transmitting MUX, which has several choices for arbitrations schemes, there is only one arbitration scheme for the receive side.

On the receive side in the MUX interface, if one or multiple clients assert req, and the FIFO is not empty, all clients that assert req will receive an AKK in the same clock cycle. On the next clock cycle, the requested piece of data will be presented to all clients and valid data will be asserted for all clients that are asserting req.

The MUX interface to the PMC Dual G-Link receiver is the GLink_RX_MUX_FIFO_IF. It has one generic to control the general characteristics of the interface:

- **PASS_MODE**: When set to TRUE the PMC Dual G-Link receiver side of the FIFO as well as all input flip-flops are clocked on REFCLK. When set to FALSE the input flip-flops and FIFO are clocked on RXCLK.

Table 7-21: PMC Dual G-Link RX MUX FIFO Interface Component Port Signals

Signal Name	Width	Dir	Description
Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARUP_Virtex component
Ref_Clock	1	Input	User interface to the PASS mode reference clock signal

Signal Name	Width	Dir	Description
R_Clock	1	Input	All logic on the client side of the FIFO is clocked on this clock
Pads.Data_In	17	Input	Data pad signals
Pads.Slip	1	Input	Used during PASS mode. Set to '1' if the receiver needed to slip one word to compensate for high phase difference between REFCLK and RXCLK
Pads.Ctrl_Xfer	1	Input	Set to '1' if the word on Pads.Data_In is a control word.
Pads.Data_Xfer	1	Input	Set to '1' if the word on Pads.Data_In is a data word.
Pads.Error	1	Input	Set to '1' if the receiver received a word that was not a valid data, control or idle word.
Pads.Ready	1	Input	Set to '1' if the receiver has received 64 to 128 consecutive valid words.
Pads.RX_Clock	1	Input	Recovered data clock pad signal
Pads.Ref_Clock	1	Output	Reference clock output pad signal
Client(x).Data_In	17	Output	Data received from the Fiber Optic Interface
Client(x).Ctrl_Xfer	1	Output	Set to '1' if the word on Data_In is a control word
Client(x).Data_Xfer	1	Output	Set to '1' if the word on Data_In is a data word
Client(x).Data_Valid	1	Output	Set to '1' when Data_In, Ctrl_Xfer and Data Xfer are valid
Client(x).RX_Clock	1	Output	RX clock recovered from the data stream by the PMC Dual G-Link receiver
Clients(x).Req	1	Input	Set to '1' to request a data from the receiver
Clients(x).Akk	1	Output	Set to '1' when the interface has accepted the request for data

7.6.1.6 Dual G-Link RX Standard FIFO Interface (GLink_RX_STD_FIFO_IF)

The standard interface to the PMC Dual G-Link receiver drives its output signals directly from the corresponding pad signals. This means that all received data will only be valid for a single RX Clock or Ref Clock cycle, and that all user logic must be clocked with Ref Clock or RX Clock for PASS mode and non-PASS mode respectively.

The GLink_RX_Std_FIFO_IF overcomes both of these restrictions by putting a FIFO between the pad input signals and the user output signals. In this interface, data is stored temporarily in a 256-word deep block RAM FIFO as it comes in off the pads. Data may later be taken off the FIFO, to be used by logic in the FPGA.

The GLink_RX_Std_FIFO_IF has one generic to control the general characteristics of the interface:

- **PASS_MODE**: When set to TRUE, the PMC Dual G-Link receiver side of the FIFO as well as all input flip-flops are clocked on REFCLK. When set to FALSE, the input flip-flops and FIFO are clocked on RXCLK.

Table 7-22: PMC Dual G-Link RX Standard FIFO Interface Component Port Signals

Signal Name	Width	Dir	Description
Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARUP_Virtex component
Ref_Clock	1	Input	User interface to the PASS mode reference clock signal
R_Clock	1	Input	All logic on the user side of the FIFO is clocked on this clock
Pads.Data_In	17	Input	Data pad signals
Pads.Slip	1	Input	Used during PASS mode. Set to '1' if the receiver needed to slip one word to compensate for high phase difference between REFCLK and RXCLK
Pads.Ctrl_Xfer	1	Input	Set to '1' if the word on Pads.Data_In is a control word.
Pads.Data_Xfer	1	Input	Set to '1' if the word on Pads.Data_In is a data word.
Pads.Error	1	Input	Set to '1' if the receiver received a word that was not a valid data, control or idle word.
Pads.Ready	1	Input	Set to '1' if the receiver has received 64 to 128 consecutive valid words.
Pads.RX_Clock	1	Input	Recovered data clock pad signal
Pads.Ref_Clock	1	Output	Reference clock output pad signal
User_In.Data_In	17	Output	Data received from the Fiber Optic Interface
User_In.Ctrl_Xfer	1	Output	Set to '1' if the word on Data_In is a control word
User_In.Data_Xfer	1	Output	Set to '1' if the word on Data_In is a data word
User_In.FIFO_Empty	1	Output	Set to '1' when there is data in the FIFO
User_In.RX_Clock	1	Output	RX clock recovered from the data stream by the PMC Dual G-Link receiver
User_Out.Pull	1	Input	Set to '1' to request the next piece of data from the FIFO. This signal may be kept high without fear of under running the FIFO.

7.6.1.7 Dual G-Link PLD LAD Multiplexing Interface (LAD_Mux_GLink_PLD_IF)

The LAD_Mux_Glink_PLD_IF is similar to the GLink_PLD_Std_IF, but the LAD_Mux_Glink_PLD_IF connects directly to a LAD_Mux_Vector. This allow the user to read and write to the G-Link PLD from the host.

Table 7-23: PMC Dual G-Link PLD MUX Interface Component Port Signals

Signal Name	Width	Dir	Description
Clk	1	Input	Serial Data Clock. Should be set to Clocks_In.K_Clk only.
Global_Reset	1	Input	Global reset (or set) signal; usually connected to the GSR input of a STARUP_Virtex component
Pads.Data	1	Bi-dir	Data pad to the PMC Dual G-Link PLD
Pads.CS_n	1	Output	Chip Select signal to the PMC Dual G-Link PLD. Must be set to '0' for the duration of all PLD transactions and set to '1' otherwise
Pads.Dir	1	Output	Set to '0' to transmit to the PLD Set to '1' to receive data from the PLD
LAD	1	Bi-Dir	Should be set to a free LAD_Mux record

The LAD_Mux_GLink_PLD_IF has five generics:

- **BASE**: Base LAD address to read from and write to for host register access
- **INIT_DATA _CH0**: The Channel 0 Control Register will be initialized to these 10 bits when this interface is reset.
- **INIT_DATA _CH1**: The Channel 1 Control Register will be initialized to these 10 bits when this interface is reset.
- **INIT_DATA_PASS**: The PASS Mode Control Register will be initialized to these 10 bits when this interface is reset.
- **INIT_DATA_PWR**: The Power Control Register will be initialized to these 10 bits when this interface is reset.

When a write is performed to the G-Link PLD, the address must be presented along with the new data. When a read is performed from the G-Link PLD, the address must be presented, and the new data will appear in each register on the PE after approximately 30 clock cycles.

The Valid signal for the particular register will go high, acknowledging the new data has been set in the PLD and it has been read back by the PE. The following three tables define the PLD address necessary to perform reads and writes to the PLD. G-Link PLD Address locations are necessary when accessing the PLD from the host using the LAD_Mux_Glink_PLD_IF.

Table 7-24: G-Link PLD Address Location

Register Name	Address	Data Bits	Description
Channel 0	0	10	Channel 0 Control Register
Channel 1	1	10	Channel 1 Control Register
PASS Mode	4	10	PASS Mode Control Register
PWR Control	6	10	Power Control Register

When new data is written to the G-Link PLD, a single 64-bit write by the host is necessary. Please refer to Table 7-25 for the G-Link LAD Mux PLD Write definitions.

Table 7-25: G-Link LAD Mux PLD Write Definitions

Signal Name	Data Bits	Dir	Description
Data_Out	0-9	Output	Data which will be written to the PLD.
Reg_Sel	10-12	Output	PLD register select, determine which 10 bit register will get the new data.

Upon reset or after new data is written to the PLD, the data is automatically read back and stored in local registers in the PE. The entire PLD's values are contained in a single 64-bit register.

Table 7-26: G-Link LAD Mux PLD Read Definitions

Signal Name	Data Bits	Dir	Description
Data_CH0_In	0-9	Input	Channel 0 register Data
Data_CH1_In	10-19	Input	Channel 1 register Data
Data_PASS_In	20-29	Input	Pass Mode register Data
Data_PWR_In	30-39	Input	Power Control register Data
Data_CH0_Valid	40	Input	Channel 0 register data valid signal.
Data_CH1_Valid	41	Input	Channel 1 register data valid signal.
Data_PASS_Valid	42	Input	Pass Mode register data valid signal
Data_PWR_Valid	43	Input	Power control register data valid signal.

7.7 FIREBIRD™ Family Mux Library

The FIREBIRD™ Family Mux Library is a set of components and interface specifications that simplify application development by providing commonly used interfaces and promoting reusable components. The current release of the FIREBIRD™ Family Mux Library contains two major component groups—the LAD64_Mux library and the Mem_Mux library. The LAD64_Mux library simplifies the most common LAD Bus module development. The Mem_Mux library abstracts the interface to the available memory ports on the FIREBIRD™ board and provides the capability to share those interfaces between PEs.

Two types of components are provided by the libraries—server and client. A "server" is a component that allows multiple modules to share a single resource. Servers are characterized by the rules used to arbitrate between modules competing for access to the resource. A "client" is a component that requires access to a resource controlled by a server. A client must obey certain protocols when negotiating with a server for access. Figure 7-4 illustrates the General Mux Library Scheme.

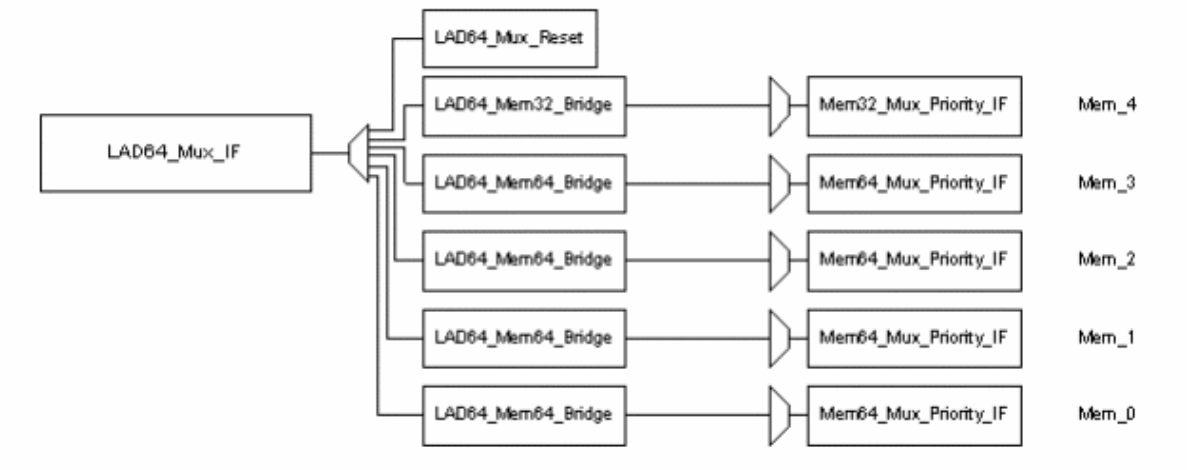


Figure 7-4: General Mux Library Scheme

7.7.1 Standard PE Templates

The FIREBIRD™ Family Library components that replace the Interface Standard components are provided in the standard template files for each of the FIREBIRD™ PEs. The LAD64_Std_If is replaced by the LAD64_Mux_If. Each of the Memory standard interfaces is replaced by the Memory Mux interfaces. The template files also include the

LAD64_Mux_Reset component as a replacement for the standard global reset logic.

7.7.2 LAD64_Mux Library

The LAD64_Mux Library provides a reusable set of components that allows the user to create fully functional LAD bus interfaces with minimum VHDL coding. The library consists of an LAD bus server unit and a set of possible client components that connect to an arbiter. There are two arbitration schemes available—Priority and Fair. The LAD bus server is used in place of the LAD64_Std_IF. The arbiter has a user-defined number of connections for client components. To construct an LAD interface, the user simply connects appropriate clients to the server.

Clients of the LAD64_Mux have a port called LAD. The definition of a LAD64_Mux is shown in Table 7-27:

Table 7-27: LAD64_Mux Record Signals

Signal Name	Width	Dir	Description
Addr	23	Output	User interface to the LAD bus address; note that this is a QWORD address; also note that this address is automatically incremented each clock cycle during burst LAD bus cycles
Write	1	Output	Write select signal; indicates a write cycle when '1' or read cycle when '0'
High_Strobe	1	Output	Data strobe high DWORD signal; indicates write data is valid on LAD bus
Low_Strobe	1	Output	Data strobe low DWORD signal; write data is valid on LAD bus
DMA_High_Strobe	1	Output	DMA Data strobe high DWORD signal; indicates write DMA data is valid on LAD bus
DMA_Low_Strobe	1	Output	DMA Data strobe high DWORD signal; indicates write DMA data is valid on LAD bus
Reset	1	Input	User interface to the reset (or set) signal; usually connected to the Global_Reset signal of a LAD64_MUX_Reset component.
Data_In	64	Output	User interface to the input data from the LAD bus
Data_Out	64	Input	User interface to the output data to the LAD bus
Akk	1	Input	User interface to the LAD acknowledge signal. Indicates which LAD64_Mux_Vector has access to the LAD bus. Only one AKK should be valid at a time.
Int_Req	1	Input	User interface to the interrupt request signal; high-to-low transition on this signal generates a single pulse on the interrupt request pad signal (which in turn will generate a PCI interrupt to the host)
DMA_RStat	2	Input	User interface to the DMA Read status flags; indicates the read status of a DMA operation.
DMA_WStat	2	Input	User interface to the DMA Write status flags; indicates the write status of a DMA operation.
DMA_RAkk	1	Input	User interface to the DMA Read acknowledge signal. Indicates which LAD64_Mux_Vector Read status flags are valid. Only one DMA_RAkk should be valid at a time.

Signal Name	Width	Dir	Description
DMA_WAkk	1	Input	User interface to the DMA Read acknowledge signal. Indicates which LAD64_Mux_Vector Write status flags are valid. Only one DMA_RAkk should be valid at a time.

Clients are connected to the LAD64_Mux server by assigning them an unused element of the LAD64_Bus vector. Clients cannot share elements of the Clients vector—each must have its own.

INFORMATION NOTE

All available elements of an LAD64_Mux_vector must be connected to a client. Unconnected elements will cause mapping failures in Synplify®.

7.7.2.1 LAD64_Mux_IF

The component instantiation for the LAD64_Mux_IF is shown in the following code listing:

```
U_LAD64_MUX_IF : LAD64_Mux_IF
   generic map
   (
     Register_Mux => true,
     V25_Enabled  => false,
   )
   port map
   (
     Kclk    => K_Clk,
     Reset   => Global_Reset,
     pads    => Pads.LAD64_Bus,
     Clients => LAD64_Bus
   );
```

The Clients port is bound to an array of LAD64_Mux_vector records. Each record corresponds to a single LAD Mux client, such as a Register File, Block RAM, or Memory Bridge. The size of the Clients array must equal the number of clients connected to the LAD bus. The KClk, Reset, and Pads ports should be connected to the appropriate resources. There are also two generics which can be instantiated by the user:

- **Register Mux**: This Boolean generic is used to register the LAD bus to provide correct timing for LAD Bus operations. In the Firebird™/PMC this value defaults to true and should not be changed.
- **V25_Enabled**: The Boolean generic should be set to true when the backplane interface BUFFER_TYPE is set to a drive level which required 2.5V; otherwise this generic should be set to false. The default value is false.

7.7.2.2 LAD64_Mux_Reset

The component instantiation for the LAD64_Mux_Reset is shown in the following code listing:

```
u_LAD64_Mux_Reset: LAD64_Mux_Reset
 generic map
 (
  Base => x"7ff8"
 )
 port map
 (
  Rclk => M_Clk,
  Kclk => K_Clk,
  LAD  => LAD64_Bus(4),

  Reset       => Global_Reset,
  DLL_Reset_0 => Clocks_Out.M_Clk_DllRst,
  DLL_Reset_1 => Clocks_Out.K_Clk_DllRst,
  DLL_Reset_2 => Clocks_Out.U_Clk_DllRst,
  DLL_Reset_3 => Clocks_Out.P_Clk_DllRst,
  DLL_Reset_4 => User_Reset_0,
  DLL_Reset_5 => User_Reset_1,
  DLL_Reset_6 => User_Reset_2,
  DLL_Reset_7 => User_Reset_3
 );
```

The LAD64_Mux_Reset unit provides an LAD accessible reset unit for the PE. It encapsulates a Virtex_STARTUP block. When a '1' is written to the unit's single control address, it will generate a reset pulse on the global reset line that lasts for several Kclk periods. The reset pulse is also available at the Reset port for use in simulation.

In addition, the reset unit provides a standard method for resetting the on-chip clock DLLs. While Bit 0 of the unit's control register provides the global reset pulse, bits 1-8 provide control over the eight DLLs. Writing *0x1fe* to the reset unit will place all eight DLLs in reset. Writing *0x0* to the reset unit will remove the reset applied to the DLLs, and bring them back out of reset. Although DLL_Reset ports are intended for DLLs, they can be used as additional resets for any logic inside the PE0 design.

The LAD64_Mux_Reset unit uses a single address. In the case above, the LAD64_Mux_Reset unit responds only to LAD bus address 0x7ff8.

INFORMATION NOTE

In the above instantiation, the LAD64_Mux_Reset unit has been connected to the fourth element of the LAD64_Bus signal. As stated above, this element must not be shared with another client.

7.7.2.3 LAD64_Mux_RegFile

The component declaration for the LAD64_Mux_Regfile is shown in the following code listing:

```
component LAD64_Mux_RegFile is
   generic
   (
     Base  : std_logic_vector(15 downto 0) := x"0000";
     L2Num : natural := 0
   );
   port
   (
     Kclk  : in    std_logic;
     LAD   : inout LAD64_Mux;
     Regs  : inout LAD64_Mux_register_vector(0 to 2**L2Num-1)
   );
 end component;
```

The LAD64_Mux_RegFile provides an LAD accessible register file on a PE. Each register in the file is 64-bits.

The Regs port presents the values in the register file to the PE. The *Regs.Data_In* field is an output from the LAD64_Mux_RegFile that provides the value written to the register file from the host over the LAD bus. The value presented to the LAD bus when the host attempts to read the register is retrieved from the *Regs.Data_Out* field. *Regs.Data_Out* is an input to the LAD64_Mux_RegFile unit. To configure a register that can Read the values written by the host from the PE, connect the *Regs.Data_In* field to the *Regs.Data_Out* field.

The Regs.Strobe field is an output from the LAD64_Mux_RegFile. It presents a single *K_Clk* pulse when the value of the corresponding register is updated by the host.

Other than the Base generic the LAD64_Mux_RegFile has one additional generic:

- **L2Num**: The number of registers is controlled by the *L2Num* variable. The number of registers in the file is *2**L2Num*. It occupies LAD addresses from *Base* to *Base+2**L2Num-1*.

7.7.2.4 LAD64_Mux_CRegFile

The component declaration for the LAD64_Mux_CRegfile is shown in the following code listing:

```
component LAD64_Mux_CRegFile is
  generic
  (
    Base: std_logic_vector(15 downto 0) := x"0000";
    L2Num : natural := 0
  );
  port
  (
    Kclk  : in    std_logic;
    LAD   : inout LAD64_Mux;

    Rclk  : in    std_logic;
    Regs  : inout LAD64_Mux_register_vector(0 to 2**L2Num-1)
  );
end component;
```

The LAD64_Mux_CRegFile client provides the same services as the LAD64_Mux_RegFile, except that it reliably crosses clock domains. The outputs from the Regs record are updated synchronously with respect to the clock connected to the *Rclk* port.

7.7.2.5 LAD64_Mux_BlockRAM

The component declaration for the LAD64_Mux_BlockRAM is shown in the following code listing:

```
component LAD64_Mux_BlockRAM is
  generic
  (
    Base : std_logic_vector(15 downto 0) := x"0000"
  );
  port
  (
    KClk       : in    std_logic;
    LAD        : inout LAD64_Mux;
    MClk       : in    std_logic;
    Addr       : in    std_logic_vector(7 downto 0);
    WriteL     : in    std_logic;
    WriteH     : in    std_logic;
    WData      : in    std_logic_vector(31 downto 0);
    RDataL     :   out std_logic_vector(31 downto 0);
    RDataH     :   out std_logic_vector(31 downto 0)
  );
end component;
```

The LAD64_Mux_BlockRAM instantiates four dual-ported block RAM with one port connected to the LAD bus and two ports presented for use on the PE. The RAMs in this component which are each 32 bits wide and 256 words deep. *Mclk*, *Addr*, *WriteL, WriteH*, and *WData* are inputs to the LAD64_Mux_BlockRAM. *RDataL* and *RdataH* are outputs.

The appended “L” and “H” on the “Write” and “RData” signals are inputs and outputs of the Low and High BlockRAMs, respectively. Since this unit is based on Virtex Block RAM, Reads are performed synchronously. In other words, when an address is presented on the Addr field, the value of that address from the memory is not valid until after the next clock edge. The unit occupies address space from Base to Base+255.

7.7.2.6 LAD64_Mux_BlockRAM64

The component declaration for the LAD64_Mux_BlockRAM64 is shown in the following code listing:

```
component LAD64_Mux_BlockRAM64 is
  generic
  (
    Base : std_logic_vector(15 downto 0) := x"0000"
  );
  port
  (
    KClk       : in    std_logic;
    LAD        : inout LAD64_Mux;
    MClk       : in    std_logic;
    Addr       : in    std_logic_vector(7 downto 0);
    Write      : in    std_logic;
    WData      : in    std_logic_vector(63 downto 0);
    RData      :   out std_logic_vector(63 downto 0)
  );
end component;
```

The LAD64_Mux_BlockRAM64 instantiates four dual-ported block RAM with one port connected to the LAD bus and one port presented for use on the PE. The RAMs in this component which are each 64 bits wide and 256 words deep. *Mclk*, *Addr*, *Write*, and *WData* are inputs to the LAD64_Mux_BlockRAM64. *Rdata* is an output. Since this unit is based on Virtex Block RAM, Reads are performed synchronously. In other words, when an address is presented on the Addr field, the value of that address from the memory is not valid until after the next clock edge. The unit occupies address space from Base to Base+255.

7.7.2.7 LAD64_Mux_Arb

The component declaration for the LAD64_Mux_Arb is shown in the following code listing:

```
component LAD64_Mux_Arb is
  generic
  (
    Register_Mux : boolean := false
  );
  port
  (
    Multi  : inout LAD64_Mux_vector;
    Single : inout LAD64_Mux;
    K_Clk  : in std_logic
```

```
  );
end component;
```

The LAD64_Mux_Arb provide a means of taking a single LAD64_Mux_Bus client and creating multiple LAD64_Mux_vectors. The arbiter allows the user to create hierarchical designs and pass a single LAD64_Mux_Bus down to the lower levels of the hierarchy and still use the LAD Mux Components within the lower levels. The generic Register_Mux should be set to false to meet the timing requirements of a read on the LAD Bus.

7.7.2.8 Assigning the BASE to the LAD64 Mux Components

Most elements have a BASE generic, which is used to map the component into each PE's LAD bus address space. Each component has a certain number of addresses it uses. Table 7-28 shows each LAD Mux component, the number of address spaces each component consumes, and the mask created for each component.

Table 7-28: LAD64 Mux Address Spaces

Component	# Address Spaces	Mask
LAD64_Mux_IF	0	N/A
LAD64_Mux_Reset	1	0x7FFF
LAD64_Mux_RegFile	2^{L2Num}	See Table 7-29
LAD64_Mux_CRegFile	2^{L2Num}	See Table 7-29
LAD64_Mux_BlockRAM	256	0x7F00
LAD64_Mux_BlockRAM64	512	0x7E00
LAD64_Mux_Arb	0	N/A

The mask for the LAD64_Mux_RegFile and the LAD64_Mux_CRegFile is automatically calculated from the L2Num generic. The mask starts out as 0x7FFF and for each L2num the least significant bits become zero. See the table below for examples:

Table 7-29: LAD64 Mux Regfile Masks

L2Num	# Registers	Generated Mask
0	1	0x7FFF
1	2	0x7FFE
2	4	0x7FFC
4	16	0x7FF0

When selecting a base address for the component the "Mask" must be taken into consideration. If the address on the LAD bus AND'ed with the Mask is equal to the Base, the component considers that LAD bus transaction to be within the component's address space. The examples below will help when deciding on a base address:

7.7.2.8.1 LAD64_Mux_Reset

This component only uses one address and it can be place at any unused location in the PE's address space.

7.7.2.8.2 LAD64_Mux_RegFile and LAD64_Mux_CRegFile

These components use 2^{L2Num} number of address. If the L2Num is 0, the component uses only one address and it can be placed at any unused location in the PE's address space. If L2Num is 1, two registers are now used.

Table 7-30: Example Base Address Table for RegFiles

Case	L2Num	Base Address	Mask	Produces Registers at	Correct/Incorrect
1	1	0x1001	0x7FFE	0x1001	Incorrect
	1	0x1000	0x7FFE	0x1000, 0x1001	**Correct**
2	2	0x2152	0x7FFC	None	Incorrect
	2	0x2150	0x7FFC	0x2150 – 0x2153	**Correct**

Case #1

Incorrect Base Choice:

The first LAD bus address will be 0x1001. When the LAD Address is AND'ed with the mask, the result will be 0x1001. This matches the base and the component considers this inside its address range. However, the second LAD bus address will be 0x1002. When the LAD Address is AND'ed with the mask, the result will be 0x1000. This doesn't match the base and the transaction is not considered to be within range and the write to the second register is not performed.

Correct Base Choice

The first LAD bus address will be 0x1000. When the LAD Address is AND'ed with the mask, the result will be 0x1001. This matches the base and the component considers this inside its address range. The second LAD bus address will be 0x1001. When the LAD Address is AND'ed with the mask, the result will be 0x1000. This also matches the base and the component considers this inside its address range.

Case #2

Incorrect Base Choice:

The first LAD bus address will be 0x2152. When the LAD Address is AND'ed with the mask, the result will be 0x2150. This doesn't match the base and the transaction is not considered to be within range, so the write to the second register is not performed. This will be true for all cases with this base.

Correct Base Choice

The first LAD bus address will be 0x2150. When the LAD Address is AND'ed with the mask, the result will be 0x2150. This matches the base and the component considers this inside its address range. The second LAD bus address will be 0x2151. When the LAD Address is AND'ed with the mask the result will be 0x2150. This also matches the base and the component considers this inside its address range. The next two address will produce a result of 0x2150 when the LAD address is AND'ed with the mask. Therefore all four registers will be accessible.

7.7.2.8.3 LAD64_Mux_BlockRAM and LAD64_Mux_BlockRAM64

The LAD64_Mux_BlockRAM and the LAD64_Mux_BlockRAM64 use 256 addresses, and both use a mask of 0x7F00. The addressing of the LAD64_MuxBlockRAM is described below:

Table 7-31: Example Base Addresses for LAD64_Mux_BlockRAM

Base Address	Mask	Produces Registers at	Correct/Incorrect
0x1130	0x7F00	None	Incorrect
0x1100	0x7F00	0x1100 - 0x11FF	**Correct**

Incorrect Base Choice:

The first LAD bus address will be 0x1130. When the LAD Address is AND'ed with the mask, the result will be 0x1100. This doesn't match the base and the transaction is not considered to be within range. All subsequent LAD addresses from 0x1131 to 0x122F will have the same result—no address will match the base. (When address 0x1200 – 0x122F is AND'ed with the mask, the result will be 0x1200 instead of 0x1100.)

Correct Base Choice

In this case, any address from 0x1100 – 0x11FF, when AND'ed with the mask, will be 0x1100. Therefore all addresses will be accessible.

The LAD64_Mux_BlockRAM64 is very similar to LAD64_Mux_BlockRAM however its mask is 0x7E00 instead of 0x7F00. Please use the table and examples above to calculate correct base selections for the LAD64_Mux_BlockRAM64.

7.7.3 Mem_Mux Library

The Mem_Mux Library defines a standard way of multiplexing memory ports between different user logic modules contained in the PE. The Mem_Mux Library provides a memory server that is used in place of the standard memory interface in the PE. The server is capable of providing

memory access to multiple clients. The clients must obey a simple protocol when interacting with the memory server.

Compliant clients can be designed without knowledge of which other clients may be attached to the memory port. This promotes reuse of components between designs and allows components to be easily migrated between different memory ports. The Mem32_Mux Library contains a standard client that allows the host to access 32-bit memory through the LAD bus. A sample instantiation of a Mem_Mux server is shown below.

```
U_MEM_1 : Mem64_Mux_Priority_IF
  port map
  (
    Mclk    => Clocks_In.M_Clk,
    Reset   => Global_Reset,
    Pads    => Pads.Mem_1,
    Clients => Mem64_Mux_1
  );
```

This is the instantiation of the Mem_1 memory port from the PE0 template. *Mclk*, *Reset*, and *Pads* are all connected to the appropriate resources. The Clients port is a vector of records that connects the multiplexed clients to the server. This is the same approach used with the LAD64_Mux server.

Clients are connected to the Mem64_Mux server by assigning them an unused element of the Mem64_Mux vector. Clients cannot share elements of the Clients vector—each must have its own.



INFORMATION NOTE

All available elements of an Mem_Mux_vector must be connected to a client. Unconnected elements will cause mapping failures in Synplify®.

7.7.3.1 Memory Arbitration Schemes

Two types of memory arbitration schemes can be used, both of which are described below.

7.7.3.1.1 Priority Arbitration

Mem_Mux servers that use priority arbitration resolve multiple simultaneous requests for access to memory. During a cycle, the client requesting access who has been assigned the lowest numbered element of the client vector is granted access over the others. It is possible, therefore, for a single high

priority client to monopolize memory access. Memory access patterns must be carefully designed so that lower priority clients have enough memory bandwidth available to perform their functions.

The Priority Arbiter has generics that can be set by the user upon instantiation of the component. Please refer to Table 7-32: Priority Arbiter Generics, for a complete description of these generics

Table 7-32: Priority Arbiter Generics

Generic Name	Values	Description
Avoid_Overflow	True/ False	This adds logic that guarantees that an arbiter's internal AkkFIFO does not overflow. If the user knows that the latency from the MemMux to memory and back is less than the FIFO depth, this generic can be set to false to avoid the additional logic and timing troubles.
Num_Akk_FIFOs	Integer	This sets the number of FIFOs used to store previous Akks for routing return values to their destinations. The value 0 is a special case for backwards compatibility, in which a single SRLFifo is used (worse timing), instead of an SRLFDFifo. Values greater than 0 include that many SRLFDFifos. Each FIFO can store 17 values, so the maximum number of outstanding requests which can be stored is 17*Num_Akk_FIFOs, or 16 when Num_Akk_FIFOs=0.
Rotate_Priority	True/ False	When arbitrating between multiple clients, the priority of who will be acknowledged first can either be fixed, in which case client 0 will always have the highest priority, or it can change based upon who was acknowledged previously.
Sticky_Priority	True/ False	With a rotating priority, the client that was acknowledged on the previous clock cycle may either be the highest priority on this clock cycle (TRUE), or it may become the lowest priority (FALSE). When the previously acknowledged client becomes the highest priority, that client can hog the bus by performing requests on every clock cycle, thus making the priority stick at that client. If Rotating_Priority is FALSE, the value of Sticky_Priority is irrelevant.
Registered_Reqs	True/ False	An arbiter may either react combinatorially with respect to the requests being made, or it may register these requests and process them a clock cycle later. The latter mode is better for timing reasons, if not all client requests already come from registers.
Registered_Akks	True/ False	Once a decision is made on which client to akk, the akk may either be presented immediately, or it may be delayed one clock cycle. Delaying akks is usually better for timing reasons since the akks typically go straight into a multiple XOR. The disadvantage of registering Reqs and Akks is the response time with which a client is acknowledged. Each time a client asserts or deasserts its request line, there may be up to a two clock cycle period during which the memory remains idle because an inactive client is still being acknowledged. If only one of Reqs and Akks is registered, this dead time is reduced to one clock cycle, maximum. Usually, this effect is never an issue, but if it is, using rotating non-sticky priorities and placing Mem*_Mux_IFIFOs in line will alleviate these problems
Half_populated	True/ False	This generic should be set to (TRUE) when a design will be used with a FIREBIRD™/PCI board that does not have memories on the bottom of the board.
Register_Data	True/ False	This generic is used to add registers between the IOB registers and the interfaces which connect to the priority arbiters. When using this generic it is important to remember that the delay when performing a read is increased by two clock cycles.

7.7.3.1.2 Fair Arbitration

Another Mem_Mux server with a different arbitration scheme is also available— the fair arbiter. The fair arbiter simply rotates access to memory among all of its clients. The more clients present in the system, the less memory bandwidth each is granted. Each client is granted an equal slice of memory bandwidth whether or not it will actually be used.

The Fair Arbiter has several generics that can be set by the user upon instantiation of the component. Please refer to Table 7-33: Fair Arbiter Generics, for a complete description of these generics.

Table 7-33: Fair Arbiter Generics

Generic Name	Values	Description
Avoid_Overflow	True/ False	This adds logic that guarantees that an arbiter's internal AkkFIFO does not overflow. If the user knows that the latency from the MemMux to memory and back is less than the FIFO depth, this generic can be set to false to avoid the additional logic and timing troubles.
Half_populated	True/ False	This generic should be set to (TRUE) when a design will be used with a FIREBIRD™/PCI board which does not have memories on the bottom of the board.
Register_Data	True/ False	This generic is used to add registers between the IOB registers and the interfaces which connect to the priority arbiters. When using this generic it is important to remember that the delay when performing a read is increased by two clock cycles.

The FIREBIRD™/PMC PE0 uses two similar Mem_Mux components. The Mem64_Mux is used when connecting to 64-bit wide memories. The Mem32_Mux is used when connecting to the 32-bit wide memory. The two components are described below.

7.7.3.2 Mem32_Mux Control

The Mem32_Mux record definitions are shown in .

Table 7-34: Mem32_Mux Record Signals

Signal Name	Width	Dir	Description
Addr	32	Input	User interface to the Memory bus address
Write	1	Input	Write select signal; indicates a write cycle when '1' or read cycle when '0'
Data_In	32	Output	User interface to the data from the 32 Bit memory port.
Data_Out	32	Input	User interface to the data to the 32 bit memory port.
Data_Valid	1	Output	Data Valid signal which is high when valid data is present from a 32 bit memory port.
Req	1	Input	Indicate that an operation, either read or write, to the memory is valid.
Akk	1	Output	Indicates the memory interface as accepted a requested transaction and is attempting to complete it.

When a client wishes to perform a Write cycle, the address of the operation on the *Addr* port should be presented. The outgoing data is placed on the *Data_Out* lines. Asserting the *Write* line indicates a Write. Finally, asserting the *Req* line indicates that the operation being presented by the client is valid. The client must continue to present its memory operation until it samples *Akk* high on a rising clock edge. A high *Akk* signal indicates that the memory interface has accepted the transaction and will attempt to complete it.

Situations where the memory operation is never completed are indicative of design errors or application malfunctions. The Mem32_Mux server provides arbitration between its various clients. Some arbitration schemes can be misused to produce situations where operations may not be able to complete. If, for instance, a priority arbitration scheme is being used and a high priority client never relinquishes control of the memory, lower priority clients can never complete their memory transactions.

Mem32_Mux clients should make no assumptions about the behavior of the *Akk* signal. *Akk* may be asserted or deasserted at any time. *Akk* may be asserted whether or not a request is being made. In some cases, the priority arbiter that comes with the library, *Akk,* is a combination function that includes the client's *Req*. In other cases, like the fair arbiter that comes with the library, the assertion of *Akk* is completely decoupled from the client.

The fair arbiter simply provides the *Akk* to clients in round-robin fashion. Even if a designer knows beforehand which arbiter will be used, to comply with the Mem32_Mux specification, and thereby be reusable within the context of the Mux Library, clients must be capable of interoperating with an arbitrary arbiter.

The timing for a Read cycle is illustrated in the following timing diagram. To read from memory, a client should assert the desired address and deassert

the *Write* line. Asserting the *Req* line indicates that the current request is valid. As with a Write request, the Read request is accepted when *Akk* is sampled high on a rising clock edge. No assumptions should be made about the behavior of the *Akk* line. After a Read request has been accepted, the corresponding data from memory will be returned on the *Data_In* lines accompanied by an asserted *Data_Valid* line. Returning data is guaranteed to arrive in the order in which it was requested. Moreover, Mem32_Mux clients should make no assumptions about the latency between the acceptance of a request and the arrival of the requested data.

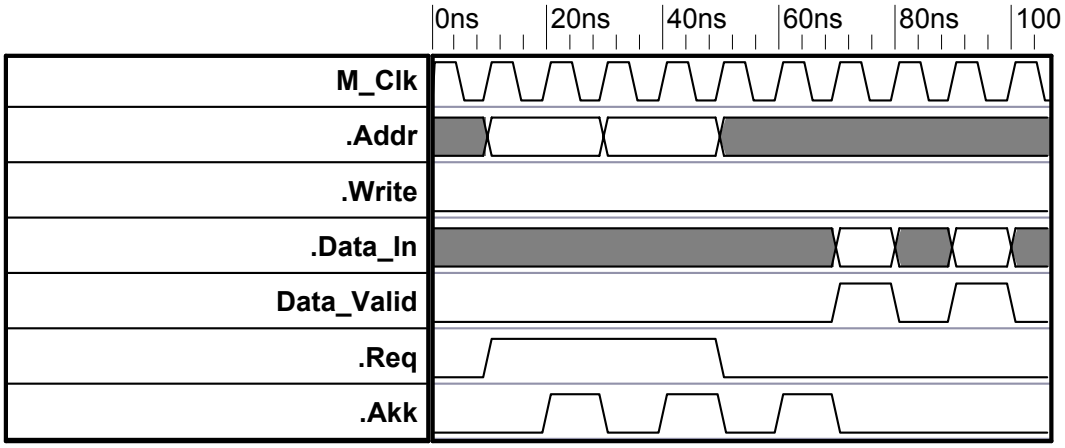


Figure 7-5: Mem32_MuxRead Cycle

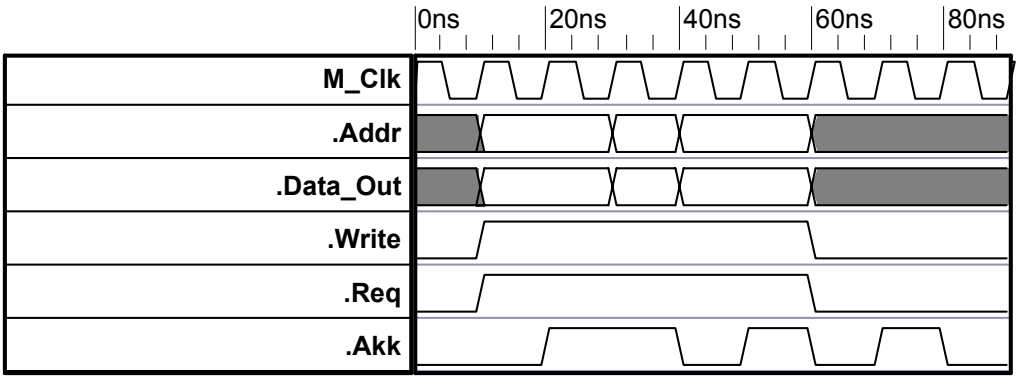


Figure 7-6: Mem32_MuxWrite Cycle

7.7.3.3 Mem64_Mux Control

The Mem64_Mux definitions are shown in

Table 7-35:

Table 7-35: Mem64_Mux Record Signals

Signal Name	Width	Dir	Description
Addr	32	Output	User interface to the Memory bus address
Write	1	Output	Write select signal; indicates a write cycle when '1' or read cycle when '0'
Data_In	64/32	Output	User interface to the data from the 64 Bit memory port.
Data_Out	64/32	Input	User interface to the data to the 64 bit memory port.
Low_Data_Valid	1	Output	Indicates the lower 32 bits have valid data from a 64 bit memory port during a read operation.
High_Data_Valid	1	Output	Indicates the upper 32 bits have valid data from a 64 bit memory port during a read operation.
Req	1	Input	Indicate that an operation, either read or write, to the memory is valid.
Low_Enable	1	Input	Indicates the lower 32 bits have valid data which should be sent to a 64 bit memory during a write or read operation.
High_Enable	1	Input	Indicates the upper 32 bits have valid data which should be sent to a 64 bit memory during a write or read operation.
Akk	1	Output	Indicates the memory interface as accepted a requested transaction and is attempting to complete it.

The Mem64_Mux control signaling is almost identical to the signaling for a Mem32_Mux port. The only differences are the wider data path and the addition of individual enables and valid signals for the upper and lower DWORDs. To perform a Read or Write on the entire 64-bit data range, both *Low_Enable* and *High_Enable* should be asserted. The *Low_Data_Valid* and *High_Data_Valid* signals indicate when valid data from a previous Read request is available. If a Read request is performed with both enables asserted, the interface guarantees that the upper and lower DWORDs of the requested data will arrive on the same clock.

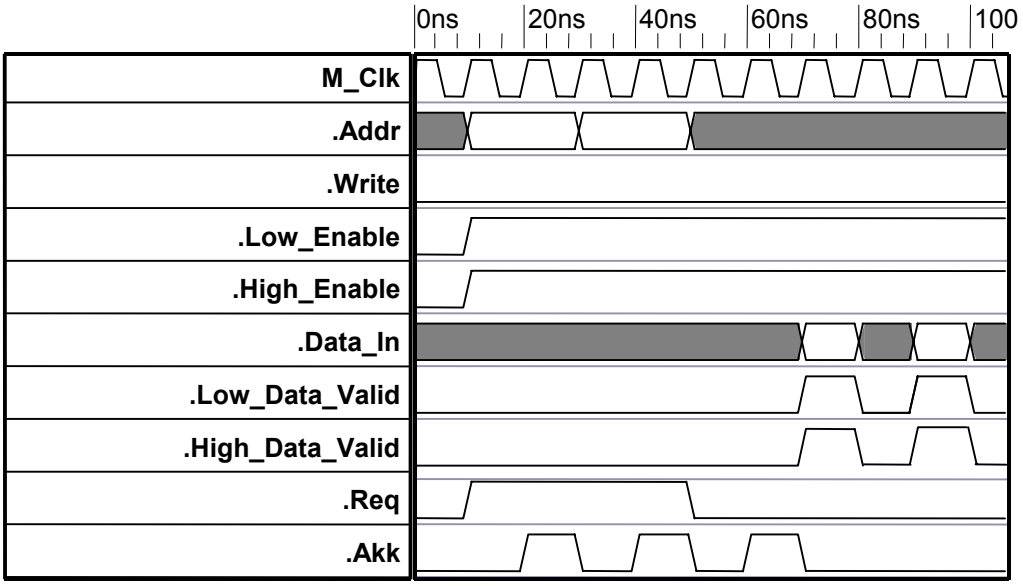


Figure 7-7: Mem64_MuxRead Cycle

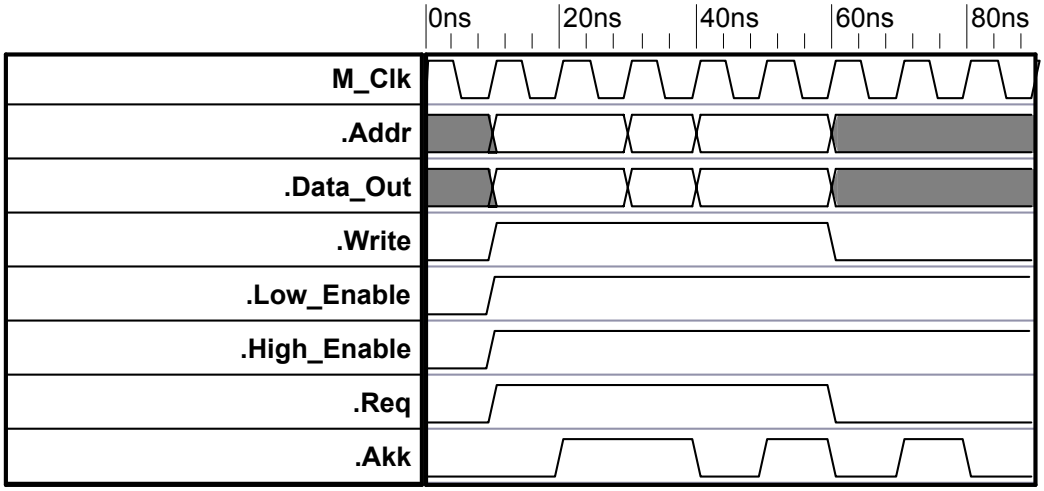


Figure 7-8: Mem64_MuxWrite Cycle

7.7.4 Programmed I/O Memory Bridges

The LAD64_Mem32_Bridge and LAD64_Mem64_Bridge provide LAD access to the Mem64_Mux memory ports and the Mem32_Mux memory port.

Many elements have Mask and Base generics which are used to map the component into the LAD bus address space(The Base generic is described here; additional generics are described in the sections below.) If the address on the LAD bus AND'ed with the Mask is equal to the Base, the component considers that LAD bus transaction to be within the component's address space. Different clients consume different amounts of address space.

Component	# Address Spaces
LAD64_Mem32_Bridge	256
LAD64_Mem364_Bridge	512

7.7.4.1 LAD64_Mem32_Bridge

The component declaration for the LAD64_Mem32_Bridge is shown in the following code listing:

```
component LAD64_Mem32_Bridge is
 generic
 (
  BASE : std_logic_vector(15 downto 0) := x"0000";
  REG_MEM_DATA : boolean := FALSE
 );
 port
 (
  Kclk      : in    std_logic;
  Mclk      : in    std_logic;
  LAD       : inout LAD64_Mux;
  Mem       : inout Mem32_Mux
 );
end component;
```

The LAD64_Mem32_Bridge provides LAD access to a Mem32_Mux memory port. An indirect addressing scheme is used. Data headed for memory is buffered up from the LAD bus in an on-PE Block RAM and then copied into memory. Data being retrieved from memory is copied from memory to the Block RAM and then read out across the LAD Bus. The Standard Library provides a set of C-based memory access functions that provide a simplified mechanism for interacting with memory through the bridge. The LAD64_Mem32_Bridge occupies addresses from Base through Base+255.

7.7.4.2 LAD64_Mem64_Bridge

The component declaration for the LAD64_Mem64_Bridge is shown in the following code listing:

```
component LAD64_Mem64_Bridge is
 generic
 (
  Base : std_logic_vector(15 downto 0) := x"0000";
  REG_MEM_DATA : boolean := FALSE
 );
 port
 (
  Kclk      : in    std_logic;
  Mclk      : in    std_logic;
  LAD       : inout LAD64_Mux;
  Mem       : inout Mem64_Mux
 );
```

```
end component;
```

The LAD64_Mem64_Bridge provides the same functionality as the LAD64_Mem32_Bridge except that it provides access to 64-bit memory ports. It occupies address from Base through Base+511.

The Mem64_Mux memory interface specification is based on the signals encapsulated in

Table 7-35: Mem64_Mux Record Signals.

7.7.4.3 Assigning the BASE to the LAD Memory Bridges

The LAD memory bridges have a BASE generic which is used to map the component into each PE's LAD bus address space. The LAD_Mem32_Bridge uses 512 addresses, while the LAD_Mem64_Bridge uses 1024 address.

When selecting a base address for the bridges, the "Mask" must be taken into consideration. If the address on the LAD bus AND'ed with the Mask is equal to the Base, the component considers that LAD bus transaction to be within the component's address space. The examples below will help when deciding on a base address:

7.7.4.3.1 LAD64_Mem32_Bridge

As noted, the LAD64_Mem32_Bridge uses 512 addresses. The table below shows the address map of the LAD64_Mem32_Bridge. The base for this example is 0x0; therefore the addresses consumed by this component is 0x0 - 0x200.

Table 7-36: Address in the LAD_Mem32_Bridge

Addresses	Consumed By
0x000-0x0FF	LAD64_Mux_BlockRAM
0x100-0x101	LAD64_Mux_CRegfile w/L2Num = 1
0x102-0x1FF	Reserved

The example below will help when deciding on a base address:

Table 7-37: Example Base Address Table for the Mem32 Bridge

Base Address	Mask	Correct/Incorrect
0x1150	0x7F00	Incorrect
0x1000	0x7F00	**Correct**
0x1100	0x7F00	**Correct**

Incorrect Base Choice:

Because this component contains a LAD64_Mux_BlockRAM, the same problems described in section 7.7.2.8.3, will occur. Consequently, no information from the LAD bus will be written/read to/from the memories.

Correct Base Choice
When the address on the LAD bus ranges from 0x1000 – 0x10FF and is AND'ed with the mask the result will be 0x1000. Data from the host will be allowed to write/read to/from the block ram. The Mem Bridge also contains a LAD64_Mux_CRegFile, which sets at address "BASE+0x100", in this case 0x1300. Since the mask is 0x7F00, when the LAD address is AND'ed with the mask the result will be 0x1100. This matches "BASE+0x100" and the component considers this inside its address range

0x1100 is also an acceptable "BASE" address selection for the same reasons as above.

When using two or more LAD64_Mem32_Bridges the base addresses must be at least 0x200 apart or the addresses from one component will overlap and will cause erroneous behavior in the design.

7.7.4.3.2 LAD64_Mem64_Bridge

As stated in section 7.7.4.3, the LAD64_Mem64_Bridge uses 1024 addresses. The table below shows the address map of the LAD64_Mem64_Bridge. The base for this example is 0x0; therefore the addresses consumed by this component are 0x0 - 0x400.

Table 7-38: Address in the LAD64_Mem64_Bridge

Addresses	Consumed By
0x000-0x1FF	LAD64_Mux_BlockRAM64
0x200-0x201	LAD64_Mux_CRegfile w/L2Num = 1
0x202-0x3FF	Reserved

The example below will help when deciding on a base address:

Table 7-39: Example Base Address Table for the Mem64 Bridge

Base Address	Mask	Correct/Incorrect
0x1150	0x7E00	Incorrect
0x1000	0x7E00	**Correct**
0x1100	0x7E00	Incorrect

Incorrect Base Choice:
Because this component contains a LAD_Mux_BlockRAM64, the same problems described in section 7.7.2.8.3, will occur. Therefore no information from the LAD bus will be written/read to/from the memories.

0x1100 is also an incorrect choice for the “BASE”. When the LAD Address is AND’ed with the mask the result will be 0x1000. Since this address doesn’t match the result, no data will be written into the Block RAM.

Correct Base Choice

When the address on the LAD bus ranges from 0x1000 – 0x11FF and is AND’ed with the mask, the result will be 0x1000. Data from the host will be allowed to write/read to/from the Block RAM. The Mem Bridge also contains a LAD_Mux_CRegFile, which sets at address “BASE+0x200”, in this case 0x1200. Since the mask is 0x7F00, when the LAD address is AND’ed with the mask, the result will be 0x1200. This matches “BASE+0x200” and the component considers this inside its address range

When using two or more LAD_Mem64_Bridge’s the base addresses must be at least 0x400 apart or the addresses from one component will overlap and will cause erroneous behavior in the design.

7.7.5 DMA_Mux Library

The DMA_Mux Library provides a reusable set of components that allows the user to create fully functional LAD bus interfaces with minimum VHDL coding. Figure 7-9 illustrates the General DMA Mux Library Scheme.

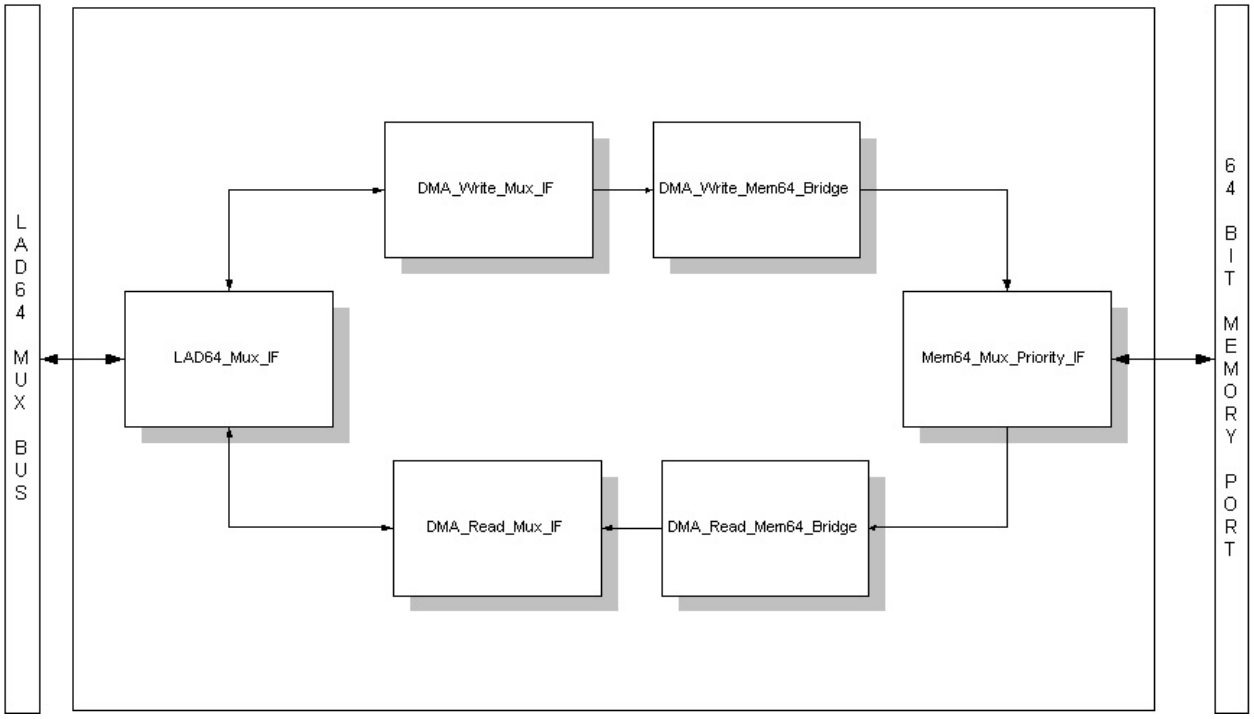


Figure 7-9: General DMA Mux Library Scheme

7.7.5.1 LAD64_DMA_Write_Mux_IF

The component declaration for the LAD64_DMA_Write_Mux_IF is shown in the following code listing:

```
                component LAD64_DMA_Write_Mux_IF is
  port
  (
    Kclk        : in    std_logic;
    LAD         : inout LAD64_Mux;
    Rclk        : in    std_logic;
    Reset       : in    std_logic;
    DMA         : inout LAD64_DMA_Write_Mux_vector
  );
end component;
```

The LAD64_DMA_Write_Mux_IF component provides the user with DMA data from the host system and controls the DMA_Wstat status flags. The DMA port signals are described in Table 7-40:

Table 7-40: LAD64_DMA_Write_Mux_Vector Record Signals

Signal Name	Width	Dir	Description
Data_LValid	1	Output	User interface which will be '1' when valid data is "Pulled" from the lower 32 bit FIFO.
Data_HValid	1	Output	User interface which will be '1' when valid data is "Pulled" from the upper 32 bit FIFO.
PullL	1	Input	Active high signal used to pull data from the lower FIFO.
PullH	1	Input	Active high signal used to pull data from the upper FIFO.
Data_In	64/32	Output	User interface to the input data from the LAD bus. The data will either be 64 or 32 bits depending on which "Pull" was used.
Fifo_Status	5	Output	User interface which indicate the status of both FIFO.
Fifo_EmptyL	1	Output	User interface signal, which indicates the empty status of the lower FIFO.
Fifo_EmptyH	1	Output	User interface signal, which indicates the empty status of the upper FIFO.
Fifo_Full	1	Output	User interface signal, which indicates that either the upper or lower FIFO has reach the ¾ full mark. The ¾ mark is used to prevent over running the input FIFOs

The PCI controller on a FIREBIRD™/PMC can provide data to the PE in two different ways. The optimal way is when the data comes to the PE 64 bits at a time. The other way, is when data is sent to the PE as two 32-bit writes. To handle both cases, the LAD64_DMA_Write_Mux_IF uses two Block RAM FIFOs, which are 256 DWORD deep.

The LAD port is a unique LAD64_Mux_Vector, described in Table 7-27: LAD64_Mux Record Signals. The Rclk signal is user defined except when the LAD64_DMA_Write_Mux_IF is connected to a DMA_Write_Bridge, in this case Rclk must be Mclk. The Reset signal should be connected to the Global_Reset output of a LAD64_Mux_Reset component.

The LAD64_DMA_Write_Mux_Vector provides the user with interactive control signals to and from the Block RAM FIFOs. The Data_LValid and Data_HValid indicate which DWORD, either upper or lower, is valid on the Data_In bus. This interface has individual Pulls for the high and low FIFOs, but it is recommended that the PullL and PullH control signals be performed simultaneously. To avoid an overflow, the PullL and PullH signals are "AND"ed with their respective Fifo_Empty signals. The Fifo_Status lines are taken from the lower FIFO and indicate when the FIFO is empty, one DWORD to ¼ full, ¼ to ½ full, ½ to ¾ full, and ¾ full to full.

The Fifo_EmptyL and Fifo_EmptyH are derived directly from the empty signal of their respective FIFOs. The Fifo_EmptyL and Fifo_EmptyH signals do not necessarily go high at the same time; therefore it is necessary, if 64 bits are being pulled from the interface at a time, to "AND" the empty signals together before pulling data from the FIFOs. The LAD64_DMA_Write_Mux_IF adds a 10ps delay, in simulation only, to aid in the user designs. The Fifo_Full signal is high when either FIFO becomes ¾ full.

Only one LAD64_DMA_Write_Mux_IF can be instantiated in a design, however multiple LAD64_DMA_Write_Mux_Vectors can be connected to this interface. The LAD64_DMA_Write_Mux_Vector follows the same rules as a LAD64_Mux_Vector, in that vector must have a unique connection and no connections can be skipped.

The LAD64_DMA_Write_Mux_IF can be connected directly into the PE, or it can be used in conjunction with a LAD64_DMA_Write_Mem32_Bridge or a LAD64_DMA_Write_Mem64_Bridge to write data to memory.

7.7.5.2 LAD64_DMA_Read_Mux_IF

The component declaration for the LAD64_DMA_Read_Mux_IF is shown in the following code listing:

```
component LAD64_DMA_Read_Mux_IF is
  port
  (
    Kclk        : in    std_logic;
    LAD         : inout LAD64_Mux;
    Rclk        : in    std_logic;
```

```
    Reset       : in    std_logic;

    DMA         : inout LAD64_DMA_Read_Mux_vector
  );
end component;
```

The LAD64_DMA_Read_Mux_IF component sends DMA data to the host system and controls the DMA_Rstat status flags. The DMA port signals are described in Table 7-41.

Table 7-41: LAD64_DMA_Read_Mux_Vector Record Signals

Signal Name	Width	Dir	Description
Data_Out	64	Input	User interface which will be '1' when valid data is "Pulled" from the lower 32 bit FIFO.
Push	1	Input	User interface which will be '1' when valid data is "Pulled" from the upper 32 bit FIFO.
Fifo_Status	5	Output	User interface which indicate the status of both FIFO.
Fifo_Empty	1	Output	User interface signal, which indicates the empty status both FIFOs.
Fifo_Full	1	Output	User interface signal, which indicates that either the upper or lower FIFO has reach the ¾ full mark. The ¾ mark is used to prevent over running the input FIFOs

The PCI controller on a FIREBIRD™/PMC expects data 64 bits at a time. Therefore the user must provide data to the LAD64_DMA_Read_Mux_IF 64 bits at a time.

The ports on LAD64_DMA_Read_Mux_IF consist of Kclk, LAD, Rclk, Reset and DMA. The Kclk port is derived from the PCI clock and can be either 33 or 66 MHz. The LAD port is a unique LAD64_Mux_Vector which is described in Table 7-27: LAD64_Mux Record Signals. The Rclk signal is user defined except when the LAD64_DMA_Write_Mux_IF is used with a DMA_Write_Bridge, in this case Rclk must be Mclk. The Reset signal should be connected to the Global_Reset output of a LAD64_Mux_Reset component.

The LAD64_DMA_Read_Mux_Vector provides the user with interactive control signals to and from the Block RAM FIFOs. The Push signal is used with the Data_Out signal to "Push" data onto the data from the PE. . The Fifo_Status lines are taken from the lower FIFO and indicate when the FIFO is empty, 1 DWORD to ¼ full, ¼ to ½ full, ½ to ¾ full, and ¾ full to full. The Fifo_Empty is high when there are no more Qword left in the FIFO. The Fifo_Full signal is high when either FIFO becomes ¾ full.

Only one LAD64_DMA_Read_Mux_IF can be instantiated in a design, however multiple LAD64_DMA_Read_Mux_Vectors can be connected to this interface. The LAD64_DMA_Read_Mux_Vector follows the same rules as a LAD64_Mux_Vector, in that each vector must have a unique connection and no connections can be skipped.

The LAD64_DMA_Read_Mux_IF can be connected directly to the PE, or it can be used in conjunction with a LAD64_DMA_Read_Mem32_Bridge or a LAD64_DMA_Read_Mem64_Bridge to read data from memory.

7.7.6 DMA Memory Bridges

The DMA Memory Bridges are used to transfer DMA data to and from the host system to the local memories on the FIREBIRD™/PMC. The component instantiation for the LAD64_DMA_Write_Mem64_Bridge and the LAD64_DMA_Read_Mem64_Bridge are shown in the following code listing:

```
U_DMA_Write64_Bridge : LAD64_DMA_Write_Mem64_Bridge
  generic map
  (
    BASE        => x"1000",
    RELAOD_EN   => FALSE
  )
  port map
  (
    Kclk        => Clocks_In.K_Clk,
    Reset       => Global_Reset,
    LAD         => LAD64_Mux_Bus(4),

    Mclk        => Clocks_In.M_Clk,
    Mem         => Mem64_Mux_0(0),
    DMA         => DMA_Write_Mux_Bus(0),
    Status      => DMA_Status_0
  );

U_DMA_Read64_Bridge : LAD64_DMA_Read_Mem64_Bridge
  generic map
  (
    BASE        => x"1200",
    RELAOD_EN   => FALSE
  )
  port map
  (
    Kclk        => Clocks_In.K_Clk,
    Reset       => Global_Reset,
    LAD         => LAD64_Mux_Bus(4),

    Mclk        => Clocks_In.M_Clk,
    Mem         => Mem64_Mux_0(1),
    DMA         => DMA_Read_Mux_Bus(0),
    Status      => DMA_Status_1
  );
```

The ports on LAD64_DMA_Write_Mem64_Bridge and LAD64_DMA_Read_Mem64_Bridge are similar, each consisting of three generics and the ports Kclk, Reset, LAD, Mclk, Mem, DMA, and status. The Mask and Base generics are necessary in all DMA bridges because they contain a single LAD64_Mux_CRegfile. The CRegfile is used to control the start and stop memory addresses. Kclk is derived from the PCI clock and can be either 33 or 66 MHz. The Reset signal should be connected to the Global_Reset output of a LAD64_Mux_Reset component. The LAD port is a unique LAD64_Mux_Vector which is described in Table 7-27: LAD64_Mux Record Signals. The Mclk signal is the memory interface clock. The Mem port is either a Mem32_Mux Vector or a Mem64_Mux Vector. The DMA port as well as the RELOAD_EN generic are described in sections 7.7.6.1 and 7.7.6.2.

Each DMA memory bridge contains a status record which can be used in the PE to determine if a DMA operation is being performed and what percentage of the DMA operation has been completed. The DMA port signals are described in Table 7-42.

Table 7-42: LAD64_DMA_Status Record Signals

Signal Name	Width	Dir	Description
Start_Addr	32	Output	Starting address of memory address counter.
Stop_Addr	32	Output	Stopping address of memory address counter.
Curr_Addr	32	Output	Current address of the DMA operation. This number can be subtracted from the Stop_Addr to determine how many QWORDS are left in the current DMA operation.
Busy	1	Output	Active high signal indicating an active DMA operation.
Init_Xfer	1	Input*	Used with the RELOAD_EN generic. See the individual Memory Bridge definition for further explanation.

* In the DMA_Read_Bridges, this signal is driven low if RELOAD_EN is set to false. Otherwise, it must be driven by the user.

Note: While the memory bridges all operate in the same manner, they are described in separate sections to avoid confusion.

7.7.6.1 DMA Write Bridges

The LAD64_DMA_Write_Mem32_Bridge and the LAD64_DMA_Write_Mem64_Bridge are used in conjunction with a LAD64_DMA_Write_Mux_IF to take data from the host system and write it to memory. Both DMA write bridges operate the same except LAD64_DMA_Write32_Bridge is connected to a 32-bit memory port while the LAD64_DMA_Write_Mem64_Bridge is connected to a 64-bit memory port..

As stated in Section 7.7.6, the Mask and Base generics are necessary in the DMA bridges because the bridges contain a single LAD64_Mux_CRegfile. The CRegfile is used to control the start and stop memory addresses. The following exert of ANSI C code shows the steps which are necessary in the host program to set up a memory write bridge and complete a DMA transfer:

```
dReadAddr[0] = 0;
dReadAddr[1] = 0x1500;

      /* Set start and stop addresses */
      printf ("\tSetting Starting Address ... ");
      rc = WS_WritePeReg64( WS_Board, 0x0, 0x1000, 2, dReadAddr );
      if ( rc != WS_SUCCESS )
      {
        printf( "ERROR %d: %s\n", rc, WS_ErrorString( rc ) );
        WS_DmaMemFree ( WS_Board, WriteBuffer );
        return(rc);
                                      }
      printf ("Done\n");


      /* DMA Data to the selected PE  /*
      printf ("\tDMAing Data to PE[0] ... " );
      rc = WS_DmaWrite ( WS_Board, 0x0, 0x1500, WriteBuffer, &Cnt, 0x0,
     0x0 );
      if ( rc != WS_SUCCESS )
      {
        printf( "ERROR %d: %s\n", rc, WS_ErrorString( rc ) );
        WS_DmaMemFree ( WS_Board, WriteBuffer );
        return(rc);
      }
      printf ("Done\n");
```

In the lines above, there are only two API calls. The first API, WS_WritePeReg64, is used to set the start and stop memory addresses. In this case, the start address is 0 and the ending address is 0x1500.

The second API, WS_DmaWrite is used to transfer the data from the host system to the memory. In this case, WriteBuffer would contain 0x1500 QWORDS or 0x3000 DWORDS of data. If the calls are successful the "rc" values of the API calls will be WS_SUCCESS.

INFORMATION NOTE

All data being written to or read from memory, when using the DMA memory bridges are based on QWORDS.

The write bridges contain the generic RELOAD_EN, which allows the start address to be reset after a DMA transfer is completed. Although several write bridges can be instantiated at a time, only one DMA write bridge should be active at time. For this reason, even if the RELOAD_EN generic is

set to true, the address counter will not reset unless the Status.Init_Xfer signal is high at the end of a DMA operation.

The Status.Init_Xfer line to a particular bridge can be held high during a DMA write operation, but the address will only be reset when the memory address reaches the count provided by the programmed I/O write. This is necessary so the data coming from the LAD64_DMA_Write_Mux_IF only goes to one bridge. The bridges use the fifo_empty signal to determine when data should be pulled from the write mux interface.

7.7.6.1.1 LAD64_DMA_Write_Mem32_Bridge

The component declaration for the LAD64_DMA_Write_Mem32_Bridge is shown in the following code listing:

```
Component LAD64_DMA_Write_Mem32_Bridge is
  generic
  (
    BASE       : std_logic_vector(15 downto 0) := x"0000";
    RELOAD_EN  : boolean := FALSE
  );
  port
  (
    Kclk   : in    std_logic;
    Reset  : in    std_logic;
    LAD    : inout LAD64_Mux;
    Mclk   : in    std_logic;
    Mem    : inout Mem32_Mux;
    DMA    : inout LAD64_DMA_Write_Mux;
    Status : inout LAD64_DMA_Status
  );
end component;
```

As stated in Section 7.7.6.1, the LAD64_DMA_Write_Mem32_Bridge is used with a LAD64_DMA_Write_Mux_IF to write data to the 32 bit memory port on the FIREBIRD™/PMC.

7.7.6.1.2 LAD64_DMA_Write_Mem64_Bridge

The component declaration for the LAD64_DMA_Write_Mem64_Bridge is shown in the following code listing:

```
Component LAD64_DMA_Write_Mem64_Bridge is
  generic
  (
    BASE       : std_logic_vector(15 downto 0) := x"0000";
    RELOAD_EN  : boolean := FALSE
  );
  port
  (
    Kclk   : in    std_logic;
    Reset  : in    std_logic;
    LAD    : inout LAD64_Mux;
```

```
    Mclk   : in    std_logic;
    Mem    : inout Mem64_Mux;
    DMA    : inout LAD64_DMA_Write_Mux;
    Status : inout LAD64_DMA_Status
  );
end component;
```

The LAD64_DMA_Write_Mem64_Bridge is used with a LAD64_DMA_Write_Mux_IF to write data to the 64 bit memory port on the FIREBIRD™/PMC.

7.7.6.1.3 LAD64_DMA_Write_Mux_Arb

The component declaration for the LAD64_DMA_Write_Mux_Arb is shown in the following code listing:

```
component LAD64_DMA_Write_Mux_Arb is
  generic
  (
    Register_Mux : boolean := false
  );
  port
                                (
    Multi  : inout LAD64_DMA_Write_Mux_vector;
    Single : inout LAD64_DMA_Write_Mux;
    Rclk   : in    std_logic
  );
end component;
```

The LAD64_DMA_Write_Mux provides a means of taking a single LAD64_DMA_Write_Mux client and creating multiple LAD64_DMA_Write_Mux_Vectors. The arbiter allows the user to create hierarchical designs and pass a single LAD64_DMA_Write_Mux client down to the lower levels of the hierarchy and still use the LAD Mux Components within the lower levels. The generic Register_Mux can be set to true, if needed, to improve design timing.

7.7.6.2 DMA Read Bridges

The LAD64_DMA_Read_Mem32_Bridge and the LAD64_DMA_Read_Mem64_Bridge are used in conjunction with a LAD64_DMA_Read_Mux_IF to from the Firebird™ PE0 memory and send it to the host system. Both DMA read bridges operate the same except LAD64_DMA_Read_Mem32_Bridge is connected to a 32-bit memory port while the LAD64_DMA_Read_Mem64_Bridge is connected to a 64-bit memory port..

As stated in Section 7.7.6, the Mask and Base generics are necessary in the DMA bridges because the bridges contain a single LAD64_Mux_CRegfile. The CRegfile is used to control the start and stop memory addresses. The

following exert of ANSI C code shows the steps which, are necessary in the host program, to set up a memory read bridge and complete a DMA transfer:

```
dReadAddr[0] = 0;
dReadAddr[1] = 0x1500;

 /* Set start and stop addresses */
printf ("\tSetting Starting Address ... ");
rc = WS_WritePeReg64( WS_Board, 0x0, 0x1000, 2, dReadAddr );
if ( rc != WS_SUCCESS )
{
  printf( "ERROR %d: %s\n", rc, WS_ErrorString( rc ) );
  WS_DmaMemFree ( WS_Board, ReadBuffer );
  return(rc);
                                }
printf ("Done\n");


/* DMA Data from PE0  /*
printf ("\tDMAing Data from PE[0] ... " );
rc = WS_DmaRead ( WS_Board, 0x0, 0x1500, ReadBuffer, &Cnt, 0x0, 0x0
);
if ( rc != WS_SUCCESS )
{
  printf( "ERROR %d: %s\n", rc, WS_ErrorString( rc ) );
  WS_DmaMemFree ( WS_Board, ReadBuffer );
  return(rc);
}
printf ("Done\n");
```

In the lines above, there are only two API calls. The first API, WS_WritePeReg64, is used to set the start and stop memory addresses. In this case, the start address is 0 and the ending address is 0x1500. The second API, WS_DmaRead is used to transfer the FIREBIRD™ local memory to the host system. If the calls are successful, the "rc" values of the API calls will be WS_SUCCESS.

INFORMATION NOTE

All data being written to or read from memory, when using the DMA memory bridges, are based on QWORDS.

The read bridges contain the generic RELOAD_EN. This allows the start address to be reset after a DMA transfer is completed. Although several read bridges can be instantiated at a time, only one DMA read bridge *can* be active at time.

The read bridge acts as a prefetch unit for the LAD64_DMA_Read_Mux_If. As soon as the stop address doesn't match the start address, data is retrieved from memory and stored in the FIFOs in the read interface. For this reason,

the PE assumes responsibility for the reset of the start counter. Once the PE drives Status.Init_Xfer, the start and stop counters will be reset to their original values written by the programmed I/O write. This action begins the prefetching of data from memory for the LAD64_DMA_Read_Mux_IF. The bridges use the fifo_full signal to determine when a pause in fetching data should occur.

7.7.6.2.1 LAD64_DMA_Read_Mem32_Bridge

The component declaration for the LAD64_DMA_Read_Mem32_Bridge is shown in the following code listing:

```
            Component LAD64_DMA_Read_Mem64_Bridge is
  generic
  (
    BASE       : std_logic_vector(15 downto 0) := x"0000";
    RELOAD_EN  : boolean := FALSE
  );
  port
  (
    Kclk   : in    std_logic;
    Reset  : in    std_logic;
    LAD    : inout LAD64_Mux;
    Mclk   : in    std_logic;
    Mem    : inout Mem64_Mux;
    DMA    : inout LAD64_DMA_Read_Mux;
    Status : inout LAD64_DMA_Status
  );
end component;
```

As stated in Section 7.7.6.1, the LAD64_DMA_Write_Mem32_Bridge is used with a LAD64_DMA_Write_Mux_IF to read from a 32-bit memory port on the FIREBIRD™/PMC and send it to the host system.

7.7.6.2.2 LAD64_DMA_Read_Mem64_Bridge

The component declaration for the LAD64_DMA_Write_Mem64_Bridge is shown in the following code listing:

```
            Component LAD64_DMA_Read_Mem64_Bridge is
  generic
  (
    BASE       : std_logic_vector(15 downto 0) := x"0000";
    RELOAD_EN  : boolean := FALSE
  );
  port
  (
    Kclk   : in    std_logic;
    Reset  : in    std_logic;
    LAD    : inout LAD64_Mux;
    Mclk   : in    std_logic;
    Mem    : inout Mem64_Mux;
    DMA    : inout LAD64_DMA_Read_Mux;
    Status : inout LAD64_DMA_Status
  );
```

```
end component;
```

The LAD64_DMA_Read_Mem64_Bridge is used with a LAD64_DMA_Read_Mux_IF to read from a 64-bit memory port on the FIREBIRD™/PMC and send it to the host system.

7.7.6.2.3 LAD64_DMA_Read_Mux_Arb

The component declaration for the LAD64_DMA_Read_Mux_Arb is shown in the following code listing:

```
component LAD64_DMA_Write_Mux_Arb is
  generic
  (
    Register_Mux : boolean := false
  );
  port
                              (
    Multi  : inout LAD64_DMA_Read_Mux_vector;
    Single : inout LAD64_DMA_Read_Mux;
    Rclk   : in    std_logic
  );
end component;
```

The LAD64_DMA_Read_Mux provides a means of taking a single LAD64_DMA_Read_Mux client and creating multiple LAD64_DMA_Read_Mux_Vectors. The arbiter allows the user to create hierarchical designs and pass a single LAD64_DMA_Write_Mux client down to the lower levels of the hierarchy and still use the LAD Mux components within the lower levels. The generic Register_Mux can be set to true, if needed, to improve design timing.

7.7.6.3 Assigning the BASE to the LAD Memory Bridges

The DMA Bridges each contain a LAD64_Mux_CRegFile with the L2Num set to '2'. This means that there are four address spaces consumed by these components. Please refer to Section 7.7.2.8 when creating a Base for these components.

7.8 Host Model

The simulated host model is used to access the PE resources of the FIREBIRD™ board via the PCI Controller. It can also be used to initialize and analyze the contents of any memory device in the FIREBIRD™ system. The simulated host model is similar to the actual host system in that the user writes a "program" that consists of standard API functions that interact with the FIREBIRD™ board. In fact, the simulated VHDL host API procedures have nearly identical structure and usage as the actual software API functions.

7.8.1 Simulated Host API Procedures

The following simulated VHDL host API procedures are available to the user:

WS_Open	Sets up initial board configuration
WS_ReadPeReg	Reads from register locations in the PE Register Space (DWORD Boundary)
WS_WritePeReg	Writes to register locations in the PE Register Space (DWORD Boundary)
WS_ReadPeReg64	Reads from register locations in the PE Register Space (QWORD Boundary)
WS_WritePeReg64	Writes to register locations in the PE Register Space (QWORD Boundary)
WS_DMARead	Reads from the PE Space using DMA
WS_DMAWrite	Writes to the PE Space using DMA
WS_MClkSetConfig	Sets the configuration of the memory and PE clocks
WS_UClkSetConfig	Sets the configuration of the user clocks
WS_WaitOnInterrupt	Waits for PE interrupt signal
WS_QueryInterruptStatus	Returns current interrupt status of all PEs
WS_ResetInterrupt	Clears pending interrupts

The following simulation-specific host procedures that are not part of the standard host API library are also available to the user:

Debug_Load_Mem_From_File	Loads a simulated FIREBIRD™ system memory component from a file
Debug_Dump_Mem_To_File	Dumps the contents of a simulated FIREBIRD™ system memory component to a file

WS_Open

Syntax:

```
procedure
WS_Open( signal Cmd_Req :   out boolean;
         signal Cmd_Ack : in    boolean;
         board          :       WS_BoardNum;
         dFlags         :       DWORD )
```

Description:

Sets initial board configuration based on dFlags argument

Arguments:

Cmd_Req	Host model command request signal
Cmd_Ack	Host model command acknowledge signal
board	FIREBIRD™ board identifier (zero-based) variable
dFlags	DWORD flag input

Comments:

This function configures the LAD bus speed. This function corresponds to the C API call WS_Open.

WS_SetLadBusSpeed

Syntax:

```
procedure
WS_SetLadBusSpeed( signal Cmd_Req :   out boolean;
                   signal Cmd_Ack : in    boolean;
                   board          :       WS_BoardNum;
                   dFreq          :       DWORD )
```

Description:

Sets the Lad Bus data rate.

Arguments:

Cmd_Req	Host model command request signal
Cmd_Ack	Host model command acknowledge signal
board	FIREBIRD™ board identifier (zero-based) variable
dFreq	Specifies Lad bus rate

Comments:

This function causes the PCI Controller to set the LAD bus rate.

WS_DmaWrite

Syntax:

```
 procedure
WS_DmaWrite ( signal Cmd_Req  :   out boolean;
              signal Cmd_Ack  : in    boolean;
              board           : in    WS_BoardNum;
              dPeNum          : in    DWORD;
              dDwordCount     : in    DWORD;
              pRegData        : in    DWORD_array;
              pDwordsWritten  :   out DWORD;
              dTimeoutMS      : in    DWORD;
              dFlags          : in    DWORD  )
```

Description :

Writes DMA data to the memory 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 FIREBIRD™ board identifier (zero-based) variable.

dPeNum

Specifies which Processing Element to Access.

dDwordCount

Number of DWORDs to write to memory space, the PE design must handle where the data goes because the address on the LAD bus is not valid.

PRegData

DWORD array from which the write data is retrieved (must have at least dDwordCount elements in the array) pDwordsWritten Number of DWORDs written during a DMA write operation.

dTimeoutMS

Reserved for future usage.

dFlags

Reserved for future usage.

Comments:

This function causes the PCI Controller to initiate a LAD bus DMA write transaction to the PEX device.

See Also:

Ws_DmaRead

WS_DmaRead

Syntax:

```
procedure
WS_DmaRead  ( signal Cmd_Req :   out boolean;
              signal Cmd_Req :   out boolean;
              signal Cmd_Ack : in    boolean;
              board          : in    WS_BoardNum;
              dPeNum         : in    DWORD;
              dDwordCount    : in    DWORD;
              pRegData       :   out DWORD_array;
              pDwordsRead    :   out DWORD;
              dTimeoutMS     : in    DWORD;
              dFlags         : in    DWORD
```

Description :

Reads DMA data to the memory 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 FIREBIRD™ board identifier (zero-based) variable.

dPeNum

Specifies which PE to Access.

dDwordCount

Number of DWORDs to read from memory space, the PE design must handle where the data comes from because the address on the LAD bus is not valid.

PRegData

DWORD array from which the read data is stored (must have at least dDwordCount elements in the array).

pDwordsRead

Number of DWORDs read during a DMA read operation.

dTimeoutMS

Reserved for future usage.

dFlags

Reserved for future usage.

Comments:

This function causes the PCI controller to initiate a LAD bus DMA write transaction to the PEX device.

See Also:

Ws_DmaRead

WS_ReadPeReg

Syntax:

```
procedure
WS_ReadPeReg( signal Cmd_Req :   out boolean;
              signal Cmd_Ack : in    boolean;
              board          :       WS_BoardNum;
              dPeNum         :       DWORD;
              dDwordOffset   :       DWORD;
              dDwordCount    :       DWORD;
              pRegData       :   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	FIREBIRD™ board identifier (zero-based) variable
dPeNum	Specifies which PE to access
dDwordOffset	Specifies the offset of the first DWORD in the PE register space to be accessed. Zero (0) is the lowest DWORD offset.
dDwordCount	Number of DWORDs to read from PE register space, starting at *dDwordOffset*
pRegData	DWORD array into which the read data is stored (must have at least *dDWordCount* elements in the array)

Comments:

This function causes the PCI Controller to initiate a LAD bus read transaction to the PEX device.

See Also:

WS_WritePeReg

WS_ReadPeReg64

Syntax:

```
procedure
WS_ReadPeReg( signal Cmd_Req :   out boolean;
              signal Cmd_Ack : in    boolean;
              board          :       WS_BoardNum;
              dPeNum         :       DWORD;
              dQwordOffset   :       DWORD;
              dQwordCount    :       DWORD;
              pRegData       :   out QWORD_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	FIREBIRD™ board identifier (zero-based) variable
dPeNum	Specifies which PE to access
dQwordOffset	Specifies the offset of the first QWORD in the PE register space to be accessed. Zero (0) is the lowest QWORD offset.
dQwordCount	Number of Qword to read from PE register space, starting at *dQwordOffset*
pRegData	QWORD array into which the read data is stored (must have at least *dQWordCount* elements in the array)

Comments:

This function causes the PCI Controller to initiate a LAD bus read transaction to the PEX device.

See Also:

WS_WritePeReg64

WS_WritePeReg

Syntax:

```
procedure
WS_WritePeReg( signal Cmd_Req :   out boolean;
               signal Cmd_Ack : in    boolean;
               board          :       WS_BoardNum;
               dPeNum         :       DWORD;
               dDwordOffset   :       DWORD;
               dDwordCount    :       DWORD;
               pRegData       :       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	FIREBIRD™ board identifier (zero-based) variable
dPeNum	Specifies which PE to access
dDwordOffset	Specifies the offset of the first DWORD in the PE register space to be accessed. Zero (0) is the lowest DWORD offset.
dDwordCount	Number of DWORDs to write to PE register space, starting at *dDwordOffset*
pRegData	DWORD array from which the write data is retrieved (must have at least *dDWordCount* elements in the array)

Comments:

This function causes the PCI Controller to initiate a LAD bus write transaction to the PEX device.

See Also:

WS_ReadPeReg

WS_WritePeReg64

Syntax:

```
procedure
WS_WritePeReg( signal Cmd_Req :   out boolean;
               signal Cmd_Ack : in    boolean;
               board          :       WS_BoardNum;
               dPeNum         :       DWORD;
               dQwordOffset   :       DWORD;
               dQwordCount    :       DWORD;
               pRegData       :       QWORD_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	FIREBIRD™ board identifier (zero-based) variable
dPeNum	Specifies which PE to access
dQwordOffset	Specifies the offset of the first QWORD in the PE register space to be accessed. Zero (0) is the lowest QWORD offset.
dQwordCount	Number of Qword to write to PE register space, starting at *dQwordOffset*
pRegData	QWORD array from which the write data is retrieved (must have at least *dQWordCount* elements in the array)

Comments:

This function causes the PCI Controller to initiate a LAD bus write transaction to the PEX device.

See Also:

WS_ReadPeReg64

WS_MClkSetConfig

Syntax:

```
procedure
WS_MClkSetConfig( signal Cmd_Req :   out boolean;
                  signal Cmd_Ack : in    boolean;
                  board          :       WS_BoardNum;
                  dMclkSource    :       WS_CLOCK_SOURCE;
                  pMclkFreq      :       float;
                  dPclkDivisor   :       DWORD )
```

Description:

Configures both the memory clock (M_Clk) and PE clock (Extra_Clk) of the specified board.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
board	FIREBIRD™ board identifier (zero-based) variable
dMclkSource	Specifies the source of the memory clock: PROG_OSCILLATOR sets the source as the on-board programmable oscillator EXT_IO sets the source as the I/O card clock
pMclkFreq	Frequency to set the memory clock which is given in kHz or MHz (must be within the range from 30 MHz to 140 MHz).
dPclkDivisor	This legacy field is ignored for FIREBIRD™.

Comments:

This function not only configures the source and frequency of the memory clock (M_Clk), but also configures the PE clock (Extra_Clk) generation circuitry by setting the Extra_Clk divisor value.

See Also:

WS_UClkSetConfig

WS_UClkSetConfig

Syntax:

```
procedure
WS_UClkSetConfig( signal Cmd_Req :   out boolean;
                  signal Cmd_Ack : in    boolean;
                  board          :       WS_BoardNum;
                  dUclkSource    :       WS_CLOCK_SOURCE;
                  pUclkFreq      :       float )
```

Description:

Configures use clock (U_Clk) of the specified board.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
board	FIREBIRD™ board identifier (zero-based) variable
dUclkSource	Specifies the source of the user clock: PROG_OSCILLATOR sets the source as the on-board programmable oscillator EXT_IO sets the source as the I/O card clock
pUclkFreq	Frequency to set the user clock which is given in kHz or MHz (must be within the range from 30 MHz to 140 MHz).

Comments:

This function configures the source and frequency of the user clock (U_Clk).

See Also:

WS_MClkSetConfig

WS_WaitOnInterrupt

Syntax:

```
procedure
WS_WaitOnInterrupt( signal Cmd_Req  :   out boolean;
                    signal Cmd_Ack  : in    boolean;
                    board           :       WS_BoardNum;
                    dIntMask        :       DWORD;
                    pIntMask        :       DWORD;
                    dTimeoutMS      :       DWORD
 )
```

Description:

Holds the host procedure until an interrupt is received from the specified PE.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
board	FIREBIRD™ board identifier (zero-based) variable
dIntMask	Bitmask that specifies which PE interrupts are expected.
pIntMask	Bitmask that defines which PE interrupts were actually received.
dTimeoutMS	Timeout value in milliseconds. Not currently implemented in board model.

Comments:

This function causes the host model to wait, performing no more functions, until the interrupt is received.

See Also:

WS_QueryInterruptStatus
WS_ResetInterrupt

WS_QueryInterruptStatus

Syntax:

```
procedure
WS_QueryInterruptStatus( signal Cmd_Req  :   out boolean;
                         signal Cmd_Ack  : in    boolean;
                         board           :       WS_BoardNum;
                         pIntMask        :       DWORD )
```

Description:

Returns the current PE interrupt status.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
board	FIREBIRD™ board identifier (zero-based) variable
pIntMask	Bitmask that defines which PE interrupts are pending.

Comments:

This function returns which PE interrupts are currently pending (sent by the PE but not acknowledged by the host).

See Also:

WS_WaitOnInterrupt
WS_ResetInterrupt

WS_ResetInterrupt

Syntax:

```
procedure
WS_ResetInterrupt( signal Cmd_Req  :   out boolean;
                   signal Cmd_Ack  : in    boolean;
                   board           :       WS_BoardNum;
                   dIntMask        :       DWORD )
```

Description:

Clears the specified interrupts.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
board	FIREBIRD™ board identifier (zero-based) variable
dIntMask	Bitmask that specifies which PE interrupts are to be cleared.

Comments:

This function causes pending interrupts to be cleared.

See Also:

WS_QueryInterruptStatus
WS_WaitOnInterrupt

Debug_Load_Mem_From_File

Syntax:

```
procedure
Debug_Load_Mem_From_File( signal Cmd_Req   :   out boolean;
                          signal Cmd_Ack   : in    boolean;
                          File_Name        :       string;
                          Board_Num        :       integer;
                          Mem_Bank         :       Mem_Bank_Type )
```

Description:

Loads a memory bank from a formatted initialization file.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
File_Name	Specifies the file that will be used to load the memory.
Board_Num	FIREBIRD™ board identifier (zero-based) variable
Mem_Bank	Specifies the memory bank that is to be loaded:
	MEM_0_BANK
	MEM_1_BANK
	MEM_2_BANK
	MEM_3_BANK
	MEM_4_BANK
	EXT_IO_CARD_MEM0_BANK
	EXT_IO_CARD_MEM1_BANK

Comments:

Each line of the initialization file specified by *File_Name* must conform to the following format rules:

<comment>	Lines beginning with "--" denotes a comment line. Note that "--" must occur at the beginning of a line.
fill <val>	Fill the entire memory with the value <val>. Note that <val> can be either a decimal value or hexadecimal value (when preceded with "0x" or “0X").
offset <val>	Set the current internal address to <val>. Note that <val> can be either a decimal value or hexadecimal value (when preceded with "0x" or "0X"). The initial default address is set to 0.
<val>	Data value to be written to the current internal address. The internal address is automatically incremented after <val> is stored. Note that <val> can be either a decimal value or hexadecimal value (when preceded with "0x" or "0X"). Also note that more than one <val> can occur on a line (when separated by a space character).

See Also:

Debug_Dump_Mem_To_File

Debug_Dump_Mem_To_File

Syntax:

```
procedure
Debug_Dump_Mem_To_File( signal Cmd_Req    :   out boolean;
                        signal Cmd_Ack    : in    boolean;
                        File_Name         :       string;
                        Board_Num         :       integer;
                        Mem_Bank          :       Mem_Bank_Type
                        Dword_Offset      :       integer;
                        Dword_Count       :       integer )
```

Description:

Dumps the contents of a memory bank to an output file.

Arguments:

Cmd_Req	Host model command request output signal
Cmd_Ack	Host model command acknowledge input signal
File_Name	Specifies the file that will be used to load the memory.
Board_Num	FIREBIRD™ board identifier (zero-based) variable
Mem_Bank	Specifies the memory bank that is to be loaded: MEM_0_BANK MEM_1_BANK MEM_2_BANK MEM_3_BANK MEM_4_BANK EXT_IO_CARD_MEM0_BANK EXT_IO_CARD_MEM1_BANK
Dword_Offset	Specifies the offset of the first DWORD in the Processing Element register space to be accessed.
Dword_Count	Number of DWORDs to dump from the memory bank, starting at *Dword_Offset*

Comments:

Dumps the contents of a memory bank to an ASCII text file.

See Also:

Debug_Load_Mem_From_File

7.8.2 FIREBIRD™/PMC System Configuration (System_Config)

The parts of the FIREBIRD™ system VHDL configuration that can be modified by the user include:

- **Host**: The host model architecture can be changed to the name used by a particular application.
- **PE0**: The PE0 model architecture should be changed from "inactive" to the name used by a particular application.
- **BPE**: The BPE model architecture should be changed from "inactive" to the name used by a particular application.
- **PE0 memory bank 0**: The architecture of the memory bank can be changed to "Empty" (when no memory is needed) or "Static" (when memory is needed). Also, the "MEM_SIZE" generic can be changed to match the needs of the application.
- **PE0 memory bank 1**: The architecture of the memory bank can be changed to "Empty" (when no memory is needed) or "Static" (when memory is needed). Also, the "MEM_SIZE" generic can be changed to match the needs of the application.
- **PE0 memory bank 2**: The architecture of the memory bank can be changed to "Empty" (when no memory is needed) or "Static" (when memory is needed). Also, the "MEM_SIZE" generic can be changed to match the needs of the application.
- **PE0 memory bank 3**: The architecture of the memory bank can be changed to "Empty" (when no memory is needed) or "Static" (when memory is needed). Also, the "MEM_SIZE" generic can be changed to match the needs of the application.
- **PE0 memory bank 4**: The architecture of the memory bank can be changed to "Empty" (when no memory is needed) or "Static" (when memory is needed). Also, the "MEM_SIZE" generic can be changed to match the needs of the application.
- **WSDP interface :** The WSDP™ interface architecture can be changed to the name used by a particular application. The WSDP™ interface can be used to model signals coming and going through the 38 pin WSDP™ MICTOR® connector.

7.9 Introduction to FIREBIRD™ VHDL

The FIREBIRD™ VHDL Models are intended to provide an accurate means of describing the behavior of the FIREBIRD™ system. These models can be used to design the functionality of the reconfigurable PEs that reside in the FIREBIRD™ system. They can then be used to simulate the behavior of the entire system, as a means of verifying the functionality of the completed design.

7.9.1 Simulation Environment

The FIREBIRD™ system has been modeled in behavioral VHDL to give the designer an accurate simulation environment in which to develop applications. The simulation environment encapsulates the FIREBIRD™ system functionality, providing the designer with an accurate test bed for the completed PE design. This enables the designer to validate the PE design before the synthesis, placement, and routing steps.

The FIREBIRD™ simulation model is hierarchical in design. It starts with a model of the system that contains a host and one or more FIREBIRD™ boards. The FIREBIRD™ board contains the PE, memories, PCI Controller, I/O card, and clock generation circuitry.

7.9.2 Using Templates

Template files are used throughout the FIREBIRD™ 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.

7.9.3 Using VHDL Record Types

All of the user interfaces to the logical I/O ports of the PE 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.

7.9.4 Using Processing Element Standard Interfaces

Standard interfaces to the PE I/O pads are provided in order to present the user with an organized and efficient interface to the resources outside the PE. However, if the standard interfaces that are provided with the FIREBIRD™ VHDL Model do not meet the requirements of the design, the designer is not

required to use them. The designer can use the standard interfaces as a starting point to develop custom pad interfaces.

7.10 FIREBIRD™ VHDL Model

7.10.1 PE0 Model

7.10.1.1 PE0 Register Space

The PE0 register space is the range of LAD DWORD addresses on the 24-bit **Addr** signal that will generate a register strobe on the **Reg_Strobe_n** signal. The **Reg_Strobe_n** signal will be active (logical '0') when **Addr** is in the 32k DWORD range of 0x08000 to 0x0FFFF. The designer may further partition the PE0 register space by decoding this 24-bit value. Refer to Chapter 6—Software Reference and Chapter 9—Programmers Guide for more details on how to generate LAD bus register space cycles.

7.10.1.2 PE0 Memory Space

The PE0 memory space is the range of LAD DWORD addresses on the 24-bit **Addr** signal that will generate a memory strobe on the **Mem_Strobe_n** signal. At this time, the memory space of the PE0 device is inactive and the **Mem_Strobe_n** signal is always inactive (logical '1').

7.10.1.3 LAD Bus Transactions

There are four types of LAD Bus Transaction cycles—register space read cycles, register space write cycles, memory space read cycles, and memory space write cycles. The register and memory cycles are differentiated from one another by the fact that the Reg_Strobe_n signal is '0' only during register space cycles while the Mem_Strobe_n signal is '0' only during memory space cycles. Similarly, write and read cycles are differentiated from one another by the fact that the Write_Sel_n signal is '0' during write cycles and is '1' during read cycles.

The register or memory strobe is asserted during every K_Clk cycle of a valid register space or memory space LAD bus transaction. The combination of the strobe signal and the write select is used to determine when to assert/accept data to/from the LAD bus. If during any given clock cycle the strobe and write select are both active, then a write cycle is taking place and both the address and data are valid during that clock cycle. If during any given clock cycle the strobe is active and the write select is inactive, then a read cycle is taking place and the address is valid during that clock cycle. The data for the read should be driven onto the outgoing data bus during the next cycle.

The various types of LAD bus access are further illustrated in the timing diagrams labeled Figure 7-10 through Figure 7-13.

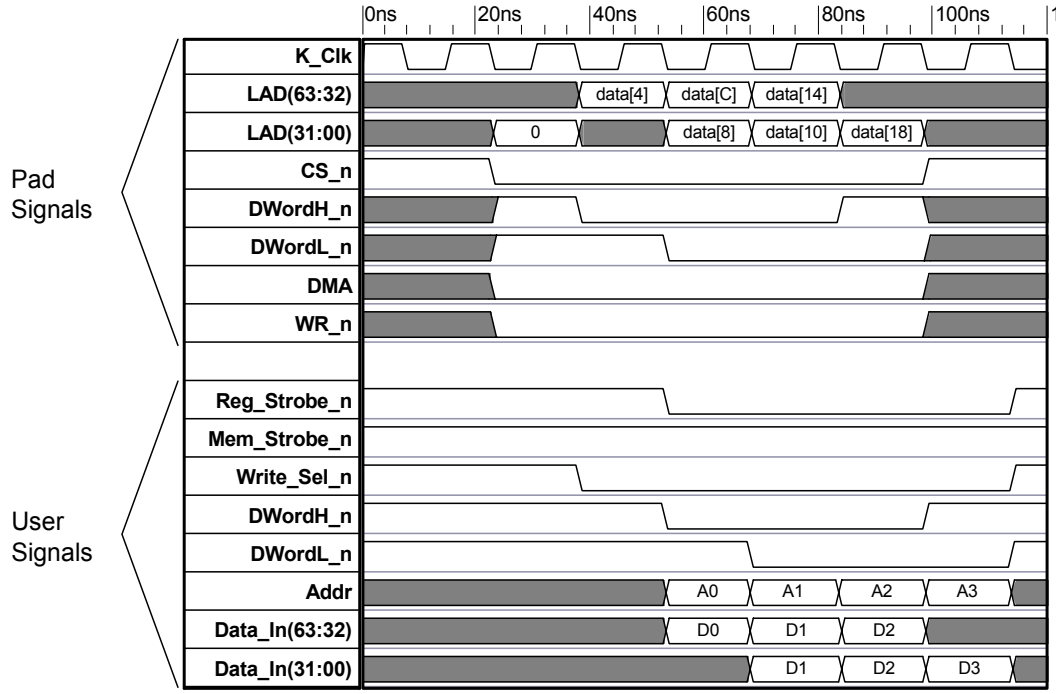


Figure 7-10 : Typical LAD bus burst write cycle to register space

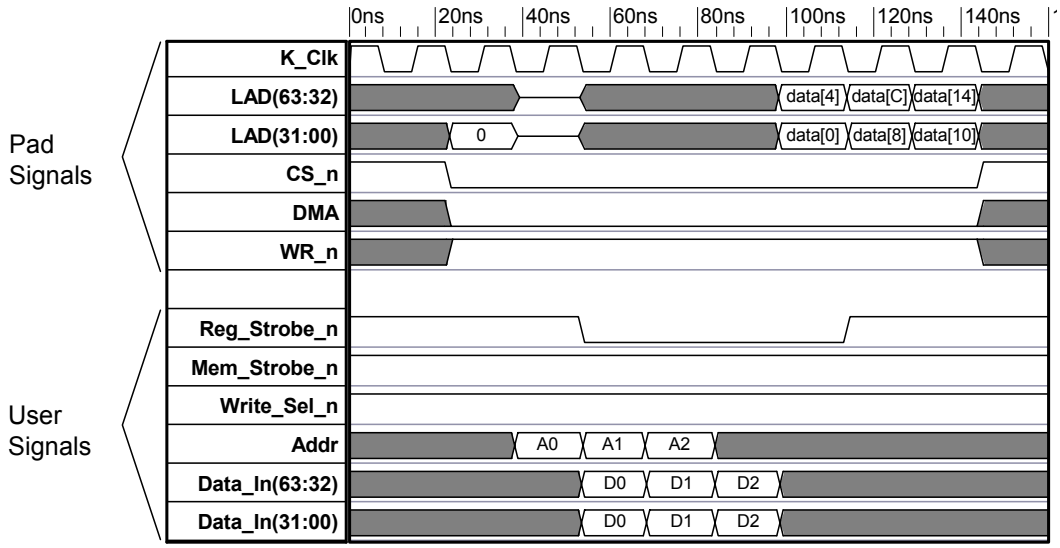


Figure 7-11 : Typical LAD bus burst read cycle to register space

DMA transactions on the Local Address Data Bus are very similar to the existing LAD bus transactions. The LAD64_Bus_In.Mem_Strobe_n signal is used to signal a DMA transaction. DMA transactions are "address-less." That is, DMA transactions are streamed into and out of the PE. It is up to the PE and host code design to control where the DMA data is going and where it is coming from.

In addition, there are two status bits, LAD64_Bus_Out.DMA_Stat, that signify whether the PE is able to receive or provide data. Bit 0 signifies that the PE can receive or provide at least four pieces of data. Bit 1 signifies that the PE can provide or receive at least one piece of data. The status signals are valid for the selected DMA channel. The current DMA channel, read or write, can be checked by LAD64_Bus_In.DMA_Chan(0) signal. A 0 indicates the write channel, a 1 indicates the read channel.

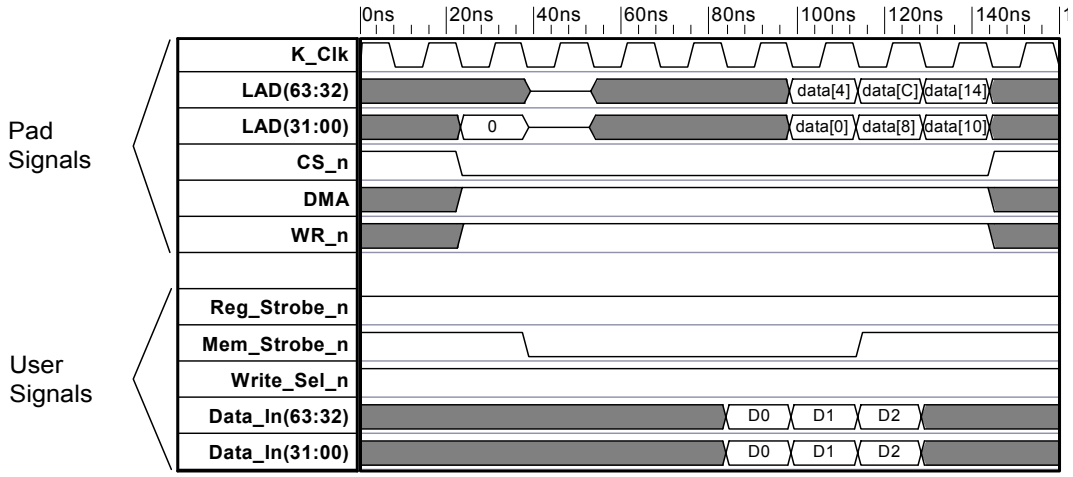


Figure 7-12: Typical DMA Read

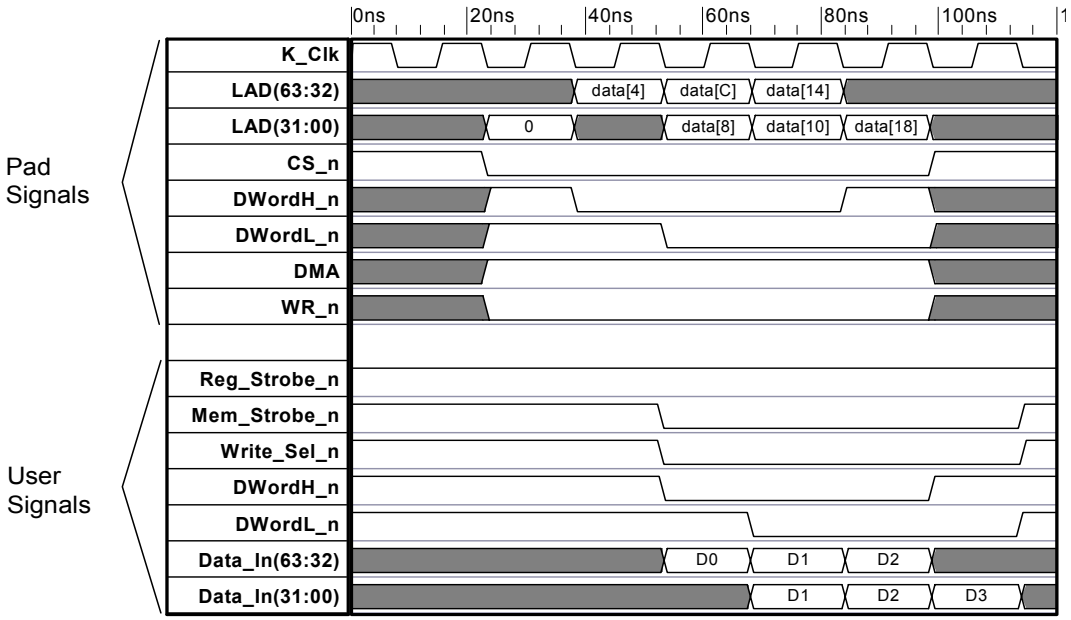


Figure 7-13: Typical DMA Write

7.10.1.4 On-board Memory timing

The memory port is synchronous to the memory clock (M_Clk), which means that all accesses are pipelined and have a latency of several clock cycles. However, since the memory port is completely synchronous, the memory port can operate up to 150 MHz. The memory standard interface simplifies access to the memory port by reducing the control signals to a strobe and write select. The strobe signal (Strobe_n = '0') is used in conjunction with the write select signal to start a write (Write_Sel_n = '0') or read (Write_Sel_n = '1') cycle. The following timing diagrams Figure 7-14 through Figure 7-17 illustrate the various types and combinations of memory port access cycles.

INFORMATION NOTE

64-bit memory transaction timing is the same as the 32-bit timing shown below with the inclusion of High_Strobe_n and High_Data_Valid_n. The pads will reflect these changes with CSH_n.

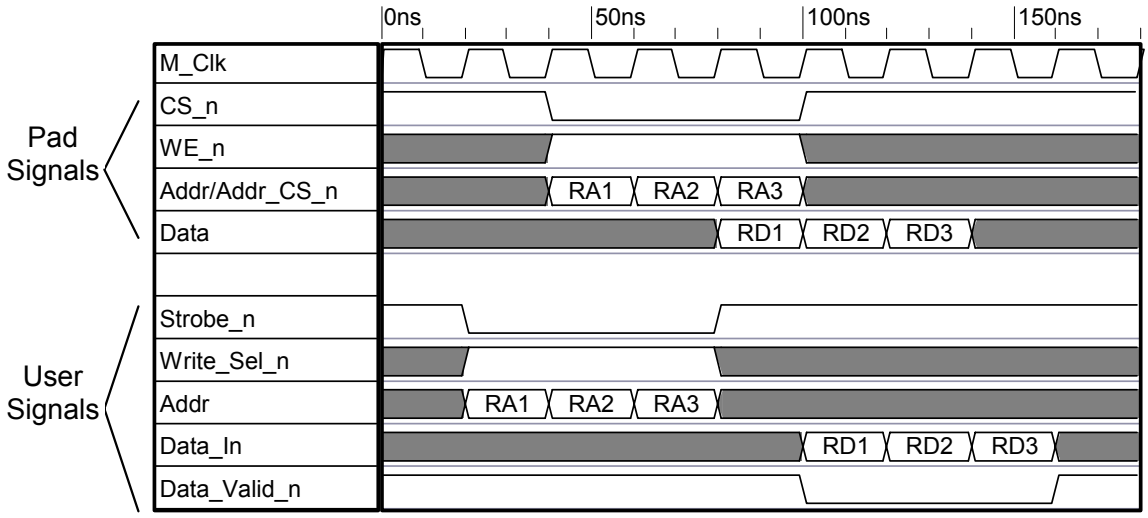


Figure 7-14: Typical Burst Read Cycle from On-Board Memory

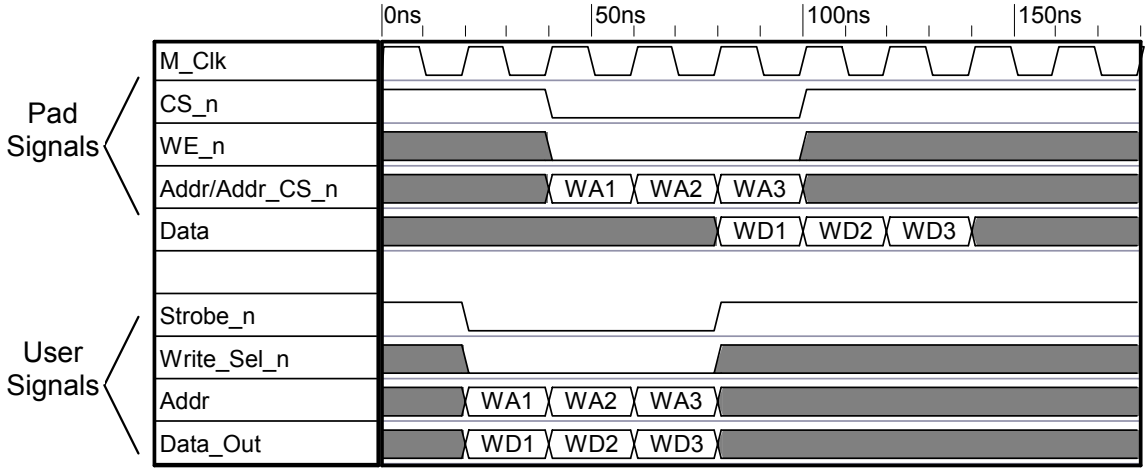


Figure 7-15: Typical Burst Write Cycle to On-Board Memory

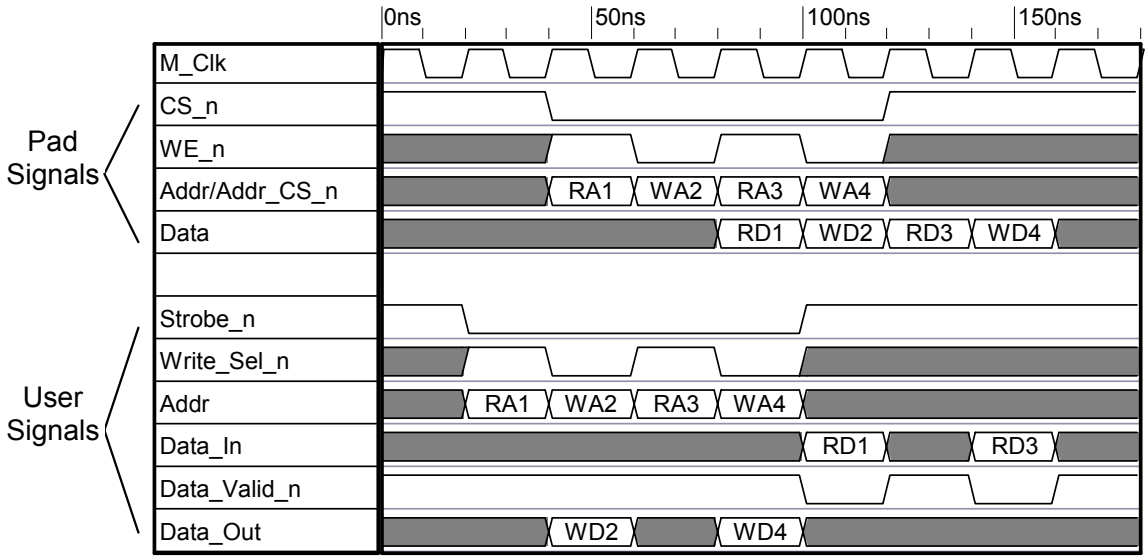


Figure 7-16: Alternating Reads From and Writes To On-Board Memory

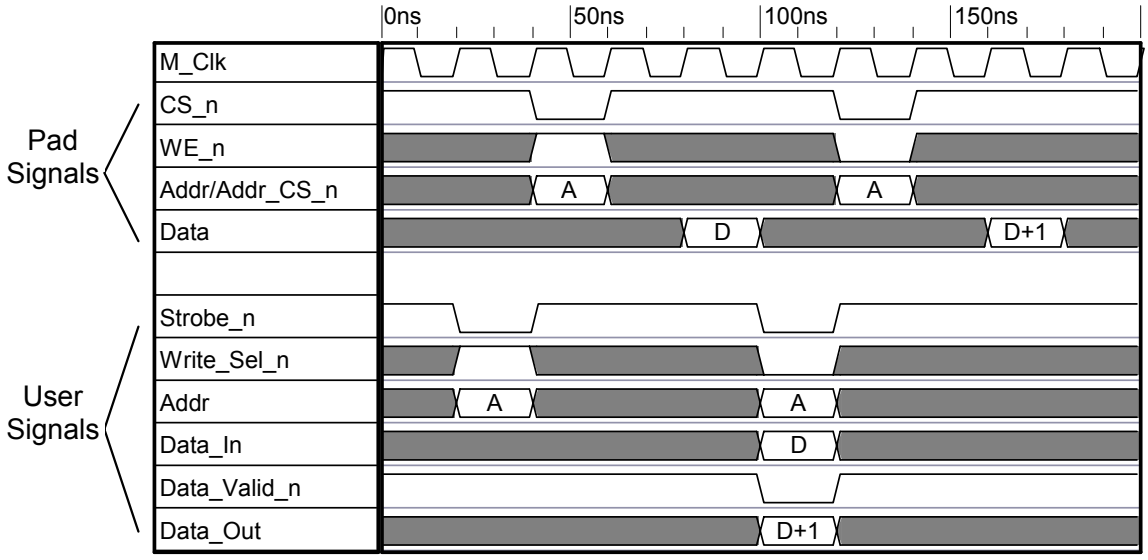


Figure 7-17: Typical Read-Modify-Write Cycle to On-Board Memory

7.10.2 Host Model

The simulated host model is used to access the PE devices via the PCI Controller. It can also be used to initialize and analyze the contents of any memory device in the FIREBIRD™ system. The simulated host model is similar to the actual host system in that the user writes a "program" that consists of standard API functions that interact with the FIREBIRD™ board.

In fact, the simulated VHDL host API procedures have nearly identical structure and usage as the actual software API functions.

7.10.3 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 all be accessed from the simulated host API function interfaces described in the previous section.

7.10.4 On-board Memory Model

The 64-bit and 32-bit on-board synchronous SRAM memory model is 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 model.

7.10.5 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 FIREBIRD™ board. When additional I/O cards are installed, VHDL templates and configurations are installed as well.

7.10.6 VHDL Model Configuration

A FIREBIRD™ can be made up of one or more FIREBIRD™ boards (each with a mezzanine card and I/O card) and a host processor. However, each FIREBIRD™ board in the system can be configured differently (e.g., different mezzanine cards, PE architectures, memory sizes). The FIREBIRD™ system simulation model can be customized to a particular application by modifying the VHDL configuration.

8 FIREBIRD™ VHDL DESIGN CYCLE

The FIREBIRD™ VHDL design cycle is made up of several simple steps that help an application developer create, simulate, synthesize, and analyze a FIREBIRD™ system design.

8.1 FIREBIRD™/PCI and /PMC VHDL Design

This section describes how to create a design using the FIREBIRD™ system VHDL model.

8.1.1 Using Template VHDL Design Files

Template files should be used when creating a FIREBIRD™ 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 "$FIREBIRD_BASE/ template/" (where FIREBIRD_BASE is the installation path to the FIREBIRD™ VHDL model) and should be copied to a design project directory and used as a starting point for the design:

pmc_pe0_template_arch.vhd	PE0 design template architecture
pmc_bpe_template_arch.vhd	BPE design template architecture
host_template_arch.vhd	Host design template architecture
wsdp_if_template_arch.vhd	WSDP™ simulation 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 that 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. Also, hierarchical design will promote design reuse, especially for common components such as LAD bus registers, and global reset logic.

8.1.3 Creating a Host Design

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, the designer must build their own lock and/or mutual exclusion mechanism for issuing commands to the PCI Controller.

8.1.4 Modifying Configuration Files

The system configuration file must also be modified to meet the needs of the design. Model details such as memory bank size, PE architecture, and host architecture can be set inside the configuration files. Simply follow the instructions inside each configuration file on how to change the user-modifiable portions of the FIREBIRD™ VHDL model.

8.1.5 Simulating a Design

This section discusses how to simulate a FIREBIRD™ VHDL design using the Model Technology, Inc., ModelSim™ VHDL simulator.

8.1.5.1 Using Template ModelSim™ Macro Files

The following ModelSim™ template files are all found in the sub-directory "$ANNAPOLIS_BASE/firebird_wsdp/template/sim" (where ANNAPOLIS_BASE is the installation path to the AMS Firebird™/PMC VHDL model) 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.

8.1.5.2 Setting Up Memory Initialization Files

The on-board memory bank models in the FIREBIRD™ system VHDL model can be initialized from a file. However, each line of the memory initialization file must be formatted using the following special rules:

<comment>	Lines beginning with "--" denote a comment line. Note that "--" must occur at the beginning of a line.
fill <val>	Fill the entire memory with the value <val>. Note that <val> can be either a decimal value or hexadecimal value (when preceded with "0x" or "0X").

offset <val> Set the current internal address to <val>. Note that <val> can be either a decimal value or hexadecimal value (when preceded with "0x" or "0X"). The initial default address is set to 0.

<val> Data value to be written to the current internal address. The internal address is automatically incremented after <val> is stored. Note that <val> can be either a decimal value or hexadecimal value (when preceded with "0x" or "0X"). Also note that more than one <val> can occur on a line (when separated by a space character).

Once the files are constructed, they must be loaded into the appropriate memory bank using the simulated host API procedure called Debug_Load_Mem_From_File.

8.1.5.3 Compiling a VHDL Design

Once the macro files and memory bank initialization files have been configured, the design can be compiled using the ModelSim™ tool or the C2WILD™ API Simulator (by Annapolis Micro Systems, Inc., sold separately). Follow these simple steps to compile the design:

1. Start the ModelSim™ tool.
2. In the **Macro** menu (or in the **File** menu in ModelSim™ v4.7 or older), 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.

8.1.5.4 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 **Macro** menu (or in the **File** menu in ModelSim™ v4.7 or older), 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™ v4.7 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.

8.2 Design Synthesis

This section discusses how to synthesize a design using the Synplicity® Synplify® VHDL synthesis tool.

8.2.1 Using Template Synplify® Project Files

The following Synplify® project file is found in the sub-directory called "$ANNAPOLIS_BASE/firebird_wsdp/template/syn" (where ANNAPOLIS_BASE is the installation path to the AMS Firebird™/PMC VHDL model) and should be copied to a design project directory and used as a starting point for the design:

pe0.prj	PE0 project VHDL synthesis project file for Synplify®
bpe.prj	BPE project VHDL synthesis project file for Synplify®

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.

8.2.2 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.

8.2.3 Synthesizing the Design

Once the Synplify® project file and design constraints have been configured, the PE design can be synthesized. Follow these simple steps to synthesize the PE0 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 **pe0.prj** Synplify® project file
6. Click on the **Open** button
7. Click on the **RUN** button

At this point, the 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 FIREBIRD™ PE model, the user 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.

8.2.4 Place-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 Xilinx® Alliance Series tools.

8.2.5 Setting Environment Variables

There are two environment variables which should be added/modified to the users 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 "$ANNAPOLIS_BASE/shared/bin" (where ANNAPOLIS_BASE is the installation path to the FIREBIRD™/PMC VHDL model). These changes/modifications are necessary for proper makefile execution.

8.2.6 Using Template Makefiles

The following Xilinx® Alliance Series makefile file is found in the sub-directory called "$FIREBIRD_BASE/template/syn" (where FIREBIRD_BASE is the installation path to the FIREBIRD™/PCI VHDL model) and should be copied to a design project directory and used as a starting point for the design:

pe0/makefile	PE0 project makefile for Xilinx Alliance tools
bpe/makefile	BPE project makefile for Xilinx Alliance tools

Once the makefile has been copied, it should be edited so that the ANNAPOLIS_BASE macro properly reflects the FIREBIRD™/PMC 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 FIREBIRD™/PMC.

1. **DEF_PART**: must be set to match the PE or BPE part type.
2. **DEF_SPEED**: should be set to match the speed grade of the PE or BPE. IF the speed grade does not match, erroneous timing reports will be generated.
3. **DEF_PROMOUT**: This only needs to be set to HEX if the Xilinx® part type is XCV2000E or greater.
4. **DEF_ENDIAN**: This produces either a little-endian or a Big-endian binary file.

8.2.7 Setting Up Timing Constraints

In addition to setting up synthesis constraints, the user 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).

8.2.8 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 FIREBIRD™ PE0 design:

1. If using an operating system other than Microsoft® Windows®, open a dos command prompt. If using an operating system other than Microsoft® Windows®, open a UNIX-like shell from which the **make** command will operate.
2. Change directories to the design project directory
3. Type **make** <design>**.hex**"

8.2.9 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, the designer can choose to modify the timing constraints or to modify the Xilinx® Alliance Series makefile options.

8.2.10 Transferring a PE design

The resulting binary file is used by the FIREBIRD™ API to load the Xilinx® image using the WS_ProgramPe function. On little-Endian machines (Intel), the binary file needs to be an .x86 file; on big-Endian machines (Sun, ALPHA, VxWorks), the binary file needs to be an .m68 file. This file is produced by the peutil program on Windows NT ®, and mcs2bin program on all other platforms.

If the VHDL development system and the host system are the same, *mcs2bin* and *peutil* can be run without transferring any files. If the host system is not the same as the VHDL development system and is not the same type of system (i.e., VHDL development performed on Intel, host system is SUN), the ASCII text <design>.mcs program must be transferred to the host system. The mcs2bin program then must be run on the host system.

To run the mcs2bin program under unix, execute
"<ANNAPOLIS_BASE>**\host\tools\mcs2bin** <design>**.mcs"**
for each design in the project (where <ANNAPOLIS_BASE> is the installation path to the FIREBIRD™/PMC VHDL model).



INFORMATION NOTE

If the VHDL development system and the FIREBIRD™ host system are not of the same type, the hex2bin program must be run on the host system. Files produced by hex2bin on a different type of system from the host system will not work on the host system.

INDEX

A

API, 1-3

B

Block RAM, 1-3
bus signaling, 4-2, 4-10

C

CLB, 1-3
clock, 6-7, 6-8, 6-11, 6-12, 6-13, 6-14, 6-15
Clock Standard Interface, 7-10, 7-13, 7-19, 7-24, 7-25, 1
Clocks, 6-2
connectors, 6-7
Conventions, 1-2

D

DC Characteristics, 6-20
Definitions, 1-3
DMA, 1-3
Driver, 1-3

E

Euro I/O Card, 1-4
External I/O, 1-4
External I/O Cards, 1-4

F

FIREBIRD™ VHDL Models, 1-4
FIREBIRD™/PCI Rev. A, 4-4, 4-6
FIREBIRD™/PCI Rev. B and C, 4-6, 4-8
FPGA, 1-4, 6-7
FPINCLK, 6-10, 6-11, 6-12

G

G-Link, 1-1, 2-2, 2-4, 4-10, 4-13, 4-16, 4-17, 4-19, 6-1, 6-6, 7-3

H

hardware, 4-1
Host Software, 1-3, 1-1, 3-1, 4-20, 6-3

I

Icons, 1-3
installation, 4-1
IO_Conn Standard Interface, 7-31

K

KCLK, 1-4
Key Words, 1-3

L

LAD Bus, 1-4
LED Definitions, 4-6, 4-8, 4-9, 4-14, 4-15, 4-19
LED Standard Interface, 7-5, 7-6, 7-33, 3

M

MCLK, 1-4, 6-13
MUX, 7-45

P

PASS mode, 6-12, 6-14
PCI, 1-4
PCLK, 1-4
PE0, 1-4
PE3, 6-2, 6-12, 6-13, 6-15
PLD Standard Interface, 7-39, 7-40
PMC, 1-1, 1-4, 2-1, 2-2, 2-3, 2-4, 4-10, 4-11, 4-12, 4-13, 4-14, 4-15, 4-16, 4-17, 4-18, 4-19, 4-20, 6-1, 6-2, 6-3, 6-5, 6-6, 6-7, 6-12, 6-13, 6-14, 6-17, 6-19, 6-21, 7-1, 7-2, 7-3, 7-7, 7-9, 7-13, 7-19, 7-23, 7-24, 7-31, 7-33, 7-50, 7-60, 7-69, 7-71, 7-72, 7-75, 7-76, 7-78, 7-79, 7-97, 8-2, 8-4, 8-5, 8-6, 8-7, 1, 16, 1, 3, 4
power, 6-9, 6-12
Power Consumption, 6-1

Processing Element, 1-4

R

RACE++, 2-3
RACEway™, 6-3
REFCLK, 6-12, 6-14, 6-15
RXCLK, 6-12, 6-13

S

SLICE, 1-5
Specifications, 6-1
Switching Characteristics, 6-1

T

Termination, 6-13
TRIG, 6-10, 6-11

U

UCLK, 1-5, 6-13

V

VIRTEX™ FPGA, 1-5
VIRTEX™-E FPGA, 1-5
VME, 1-5

W

WILDSTAR™ Development Toolkit, 1-5
WILDSTAR™ Host Software, 1-5
WILDSTAR™ VHDL Models, 1-5
WSDP™, 1-1, 2-2, 2-3, 4-10, 4-11, 4-12, 4-13, 4-14, 4-15, 4-16, 4-17, 4-18, 6-5, 7-2
WSDP™ Standard Interface, 7-23, 7-24, 7-30

X

XCLK, 6-3

Z

ZBT, 1-5

ATTACHMENT A: PE PAD LOCATIONS AND DEFINITIONS

PE0 Pad Definitions and Locations

The definitions and locations of the FIREBIRD™/PCI and PMC PE0 pads are contained in Tables A-1 and A-2 respectively. This information is also contained in the PE0 user constraints file (pe0.ucf) which is used by the Xilinx® Alliance Series to place the PE0 pads in the proper locations.



INFORMATION NOTE

The pad index corresponds to the logical pad index of the PE0 VHDL Model, not the physical I/O pin number of the FPGA die.

Table A-1: FIREBIRD™/PCI PE0 Pad Locations and Definitions

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
0	Clocks.M_Clk	BA22	Input	Memory clock
1	Clocks.K_Clk	B22	Input	PCI/LAD bus clock
2	Clocks.U_Clk	C22	Input	User clock
3	Clocks.Extra_Clk	AY22	Input	Reserved
4	LAD64_Bus.Addr_Data[00]	BB17	Bi-dir	LAD bus shared address/data bus bit 0
5	LAD64_Bus.Addr_Data[01]	AU16	Bi-dir	LAD bus shared address/data bus bit 1
6	LAD64_Bus.Addr_Data[02]	BB18	Bi-dir	LAD bus shared address/data bus bit 2
7	LAD64_Bus.Addr_Data[03]	AV16	Bi-dir	LAD bus shared address/data bus bit 3
8	LAD64_Bus.Addr_Data[04]	AY18	Bi-dir	LAD bus shared address/data bus bit 4
9	LAD64_Bus.Addr_Data[05]	AW16	Bi-dir	LAD bus shared address/data bus bit 5
10	LAD64_Bus.Addr_Data[06]	BA18	Bi-dir	LAD bus shared address/data bus bit 6
11	LAD64_Bus.Addr_Data[07]	AV17	Bi-dir	LAD bus shared address/data bus bit 7
12	LAD64_Bus.Addr_Data[08]	BB19	Bi-dir	LAD bus shared address/data bus bit 8
13	LAD64_Bus.Addr_Data[09]	AW17	Bi-dir	LAD bus shared address/data bus bit 9
14	LAD64_Bus.Addr_Data[10]	AY19	Bi-dir	LAD bus shared address/data bus bit 10
15	LAD64_Bus.Addr_Data[11]	AV18	Bi-dir	LAD bus shared address/data bus bit 11
16	LAD64_Bus.Addr_Data[12]	BA19	Bi-dir	LAD bus shared address/data bus bit 12
17	LAD64_Bus.Addr_Data[13]	AW18	Bi-dir	LAD bus shared address/data bus bit 13
18	LAD64_Bus.Addr_Data[14]	BB20	Bi-dir	LAD bus shared address/data bus bit 14
19	LAD64_Bus.Addr_Data[15]	AY20	Bi-dir	LAD bus shared address/data bus bit 15
20	LAD64_Bus.Addr_Data[16]	AV19	Bi-dir	LAD bus shared address/data bus bit 16
21	LAD64_Bus.Addr_Data[17]	BA20	Bi-dir	LAD bus shared address/data bus bit 17

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
22	LAD64_Bus.Addr_Data[18]	BB21	Bi-dir	LAD bus shared address/data bus bit 18
23	LAD64_Bus.Addr_Data[19]	AW19	Bi-dir	LAD bus shared address/data bus bit 19
24	LAD64_Bus.Addr_Data[20]	AY21	Bi-dir	LAD bus shared address/data bus bit 20
25	LAD64_Bus.Addr_Data[21]	AV20	Bi-dir	LAD bus shared address/data bus bit 21
26	LAD64_Bus.Addr_Data[22]	BA21	Bi-dir	LAD bus shared address/data bus bit 22
27	LAD64_Bus.Addr_Data[23]	AW20	Bi-dir	LAD bus shared address/data bus bit 23
28	LAD64_Bus.Addr_Data[24]	AW21	Bi-dir	LAD bus shared address/data bus bit 24
29	LAD64_Bus.Addr_Data[25]	BB22	Bi-dir	LAD bus shared address/data bus bit 25
30	LAD64_Bus.Addr_Data[26]	AW22	Bi-dir	LAD bus shared address/data bus bit 26
31	LAD64_Bus.Addr_Data[27]	BB23	Bi-dir	LAD bus shared address/data bus bit 27
32	LAD64_Bus.Addr_Data[28]	AW23	Bi-dir	LAD bus shared address/data bus bit 28
33	LAD64_Bus.Addr_Data[29]	AY23	Bi-dir	LAD bus shared address/data bus bit 29
34	LAD64_Bus.Addr_Data[30]	AV23	Bi-dir	LAD bus shared address/data bus bit 30
35	LAD64_Bus.Addr_Data[31]	BA23	Bi-dir	LAD bus shared address/data bus bit 31
36	LAD64_Bus.Addr_Data[32]	AV9	Bi-dir	LAD bus shared address/data bus bit 32
37	LAD64_Bus.Addr_Data[33]	BA11	Bi-dir	LAD bus shared address/data bus bit 33
38	LAD64_Bus.Addr_Data[34]	BB11	Bi-dir	LAD bus shared address/data bus bit 34
39	LAD64_Bus.Addr_Data[35]	AY12	Bi-dir	LAD bus shared address/data bus bit 35
40	LAD64_Bus.Addr_Data[36]	AY9	Bi-dir	LAD bus shared address/data bus bit 36
41	LAD64_Bus.Addr_Data[37]	BA12	Bi-dir	LAD bus shared address/data bus bit 37
42	LAD64_Bus.Addr_Data[38]	BB12	Bi-dir	LAD bus shared address/data bus bit 38
43	LAD64_Bus.Addr_Data[39]	AY13	Bi-dir	LAD bus shared address/data bus bit 39
44	LAD64_Bus.Addr_Data[40]	BA13	Bi-dir	LAD bus shared address/data bus bit 40
45	LAD64_Bus.Addr_Data[41]	AW10	Bi-dir	LAD bus shared address/data bus bit 41
46	LAD64_Bus.Addr_Data[42]	BB13	Bi-dir	LAD bus shared address/data bus bit 42
47	LAD64_Bus.Addr_Data[43]	AY10	Bi-dir	LAD bus shared address/data bus bit 43
48	LAD64_Bus.Addr_Data[44]	AY14	Bi-dir	LAD bus shared address/data bus bit 44
49	LAD64_Bus.Addr_Data[45]	AV11	Bi-dir	LAD bus shared address/data bus bit 45
50	LAD64_Bus.Addr_Data[46]	BA14	Bi-dir	LAD bus shared address/data bus bit 46
51	LAD64_Bus.Addr_Data[47]	AW11	Bi-dir	LAD bus shared address/data bus bit 47
52	LAD64_Bus.Addr_Data[48]	BB14	Bi-dir	LAD bus shared address/data bus bit 48
53	LAD64_Bus.Addr_Data[49]	AV12	Bi-dir	LAD bus shared address/data bus bit 49
54	LAD64_Bus.Addr_Data[50]	BA15	Bi-dir	LAD bus shared address/data bus bit 50
55	LAD64_Bus.Addr_Data[51]	AW12	Bi-dir	LAD bus shared address/data bus bit 51
56	LAD64_Bus.Addr_Data[52]	AY15	Bi-dir	LAD bus shared address/data bus bit 52
57	LAD64_Bus.Addr_Data[53]	AW13	Bi-dir	LAD bus shared address/data bus bit 53
58	LAD64_Bus.Addr_Data[54]	BB15	Bi-dir	LAD bus shared address/data bus bit 54
59	LAD64_Bus.Addr_Data[55]	AV14	Bi-dir	LAD bus shared address/data bus bit 55
60	LAD64_Bus.Addr_Data[56]	BA16	Bi-dir	LAD bus shared address/data bus bit 56
61	LAD64_Bus.Addr_Data[57]	AW14	Bi-dir	LAD bus shared address/data bus bit 57
62	LAD64_Bus.Addr_Data[58]	AY16	Bi-dir	LAD bus shared address/data bus bit 58
63	LAD64_Bus.Addr_Data[59]	BB16	Bi-dir	LAD bus shared address/data bus bit 59
64	LAD64_Bus.Addr_Data[60]	AV15	Bi-dir	LAD bus shared address/data bus bit 60
65	LAD64_Bus.Addr_Data[61]	BA17	Bi-dir	LAD bus shared address/data bus bit 61
66	LAD64_Bus.Addr_Data[62]	AY17	Bi-dir	LAD bus shared address/data bus bit 62
67	LAD64_Bus.Addr_Data[63]	AW15	Bi-dir	LAD bus shared address/data bus bit 63
68	LAD64_Bus. DWordL_n	AW9	Input	LAD bus data low strobe

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
69	LAD64_Bus. DWordH_n	BA10	Input	LAD bus data high strobe
70	LAD64_Bus.WR_n	AY11	Input	LAD bus write select
71	LAD64_Bus.CS_n	BB10	Input	LAD bus chip select
72	LAD64_Bus.Int_Req_n	AY24	Output	LAD bus interrupt request
73	LAD64_Bus.DMA	AY8	Input	LAD bus DMA/register mode select
74	LAD64_Bus.DMA_Chan[00]	AW8	Input	LAD bus DMA channel number indicator bit 0
75	LAD64_Bus.DMA_Chan[01]	BB9	Input	LAD bus DMA channel number indicator bit 1
76	LAD64_Bus.DMA_Stat[00]	BA9	Output	LAD bus DMA status flags bit 0
77	LAD64_Bus.DMA_Stat[01]	AV8	Output	LAD bus DMA status flags bit 1
78	Mem_0 .Addr[00]	K1	Output	Mem_0 Address bit 0
79	Mem_0 .Addr[01]	H4	Output	Mem_0 Address bit 1
80	Mem_0 .Addr[02]	K2	Output	Mem_0 Address bit 2
81	Mem_0 .Addr[03]	H5	Output	Mem_0 Address bit 3
82	Mem_0 .Addr[04]	J1	Output	Mem_0 Address bit 4
83	Mem_0 .Addr[05]	G3	Output	Mem_0 Address bit 5
84	Mem_0 .Addr[06]	J2	Output	Mem_0 Address bit 6
85	Mem_0 .Addr[07]	H1	Output	Mem_0 Address bit 7
86	Mem_0 .Addr[08]	G4	Output	Mem_0 Address bit 8
87	Mem_0 .Addr[09]	H2	Output	Mem_0 Address bit 9
88	Mem_0 .Addr[10]	F3	Output	Mem_0 Address bit 10
89	Mem_0 .Addr[11]	G1	Output	Mem_0 Address bit 11
90	Mem_0 .Addr[12]	G5	Output	Mem_0 Address bit 12
91	Mem_0 .Addr[13]	F1	Output	Mem_0 Address bit 13
92	Mem_0 .Addr[14]	E3	Output	Mem_0 Address bit 14
93	Mem_0 .Addr[15]	G2	Output	Mem_0 Address bit 15
94	Mem_0 .Addr[16]	F4	Output	Mem_0 Address bit 16
95	Mem_0 .Addr[17]	E1	Output	Mem_0 Address bit 17
96	Mem_0 .Addr[18]	F2	Output	Mem_0 Address bit 18
97	Mem_0 .Data[00]	AC2	Bi-dir	Mem_0 Data bit 0
98	Mem_0 .Data[01]	AC5	Bi-dir	Mem_0 Data bit 1
99	Mem_0 .Data[02]	AB1	Bi-dir	Mem_0 Data bit 2
100	Mem_0 .Data[03]	AC4	Bi-dir	Mem_0 Data bit 3
101	Mem_0 .Data[04]	AB3	Bi-dir	Mem_0 Data bit 4
102	Mem_0 .Data[05]	AB4	Bi-dir	Mem_0 Data bit 5
103	Mem_0 .Data[06]	AB2	Bi-dir	Mem_0 Data bit 6
104	Mem_0 .Data[07]	AA2	Bi-dir	Mem_0 Data bit 7
105	Mem_0 .Data[08]	AA4	Bi-dir	Mem_0 Data bit 8
106	Mem_0 .Data[09]	AA3	Bi-dir	Mem_0 Data bit 9
107	Mem_0 .Data[10]	Y4	Bi-dir	Mem_0 Data bit 10
108	Mem_0 .Data[11]	AA1	Bi-dir	Mem_0 Data bit 11
109	Mem_0 .Data[12]	Y5	Bi-dir	Mem_0 Data bit 12
110	Mem_0 .Data[13]	Y1	Bi-dir	Mem_0 Data bit 13
111	Mem_0 .Data[14]	Y3	Bi-dir	Mem_0 Data bit 14
112	Mem_0 .Data[15]	W4	Bi-dir	Mem_0 Data bit 15
113	Mem_0 .Data[16]	Y2	Bi-dir	Mem_0 Data bit 16
114	Mem_0 .Data[17]	W1	Bi-dir	Mem_0 Data bit 17
115	Mem_0 .Data[18]	W5	Bi-dir	Mem_0 Data bit 18

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
116	Mem_0 .Data[19]	W3	Bi-dir	Mem_0 Data bit 19
117	Mem_0 .Data[20]	V4	Bi-dir	Mem_0 Data bit 20
118	Mem_0 .Data[21]	W2	Bi-dir	Mem_0 Data bit 21
119	Mem_0 .Data[22]	V5	Bi-dir	Mem_0 Data bit 22
120	Mem_0 .Data[23]	V1	Bi-dir	Mem_0 Data bit 23
121	Mem_0 .Data[24]	U4	Bi-dir	Mem_0 Data bit 24
122	Mem_0 .Data[25]	U5	Bi-dir	Mem_0 Data bit 25
123	Mem_0 .Data[26]	V2	Bi-dir	Mem_0 Data bit 26
124	Mem_0 .Data[27]	T4	Bi-dir	Mem_0 Data bit 27
125	Mem_0 .Data[28]	U1	Bi-dir	Mem_0 Data bit 28
126	Mem_0 .Data[29]	T5	Bi-dir	Mem_0 Data bit 29
127	Mem_0 .Data[30]	U3	Bi-dir	Mem_0 Data bit 30
128	Mem_0 .Data[31]	R4	Bi-dir	Mem_0 Data bit 31
129	Mem_0 .Data[32]	U2	Bi-dir	Mem_0 Data bit 32
130	Mem_0 .Data[33]	R5	Bi-dir	Mem_0 Data bit 33
131	Mem_0 .Data[34]	T1	Bi-dir	Mem_0 Data bit 34
132	Mem_0 .Data[35]	P4	Bi-dir	Mem_0 Data bit 35
133	Mem_0 .Data[36]	T3	Bi-dir	Mem_0 Data bit 36
134	Mem_0 .Data[37]	P5	Bi-dir	Mem_0 Data bit 37
135	Mem_0 .Data[38]	T2	Bi-dir	Mem_0 Data bit 38
136	Mem_0 .Data[39]	N4	Bi-dir	Mem_0 Data bit 39
137	Mem_0 .Data[40]	R1	Bi-dir	Mem_0 Data bit 40
138	Mem_0 .Data[41]	M4	Bi-dir	Mem_0 Data bit 41
139	Mem_0 .Data[42]	R3	Bi-dir	Mem_0 Data bit 42
140	Mem_0 .Data[43]	M5	Bi-dir	Mem_0 Data bit 43
141	Mem_0 .Data[44]	R2	Bi-dir	Mem_0 Data bit 44
142	Mem_0 .Data[45]	P1	Bi-dir	Mem_0 Data bit 45
143	Mem_0 .Data[46]	L4	Bi-dir	Mem_0 Data bit 46
144	Mem_0 .Data[47]	L5	Bi-dir	Mem_0 Data bit 47
145	Mem_0 .Data[48]	N1	Bi-dir	Mem_0 Data bit 48
146	Mem_0 .Data[49]	K3	Bi-dir	Mem_0 Data bit 49
147	Mem_0 .Data[50]	N3	Bi-dir	Mem_0 Data bit 50
148	Mem_0 .Data[51]	K4	Bi-dir	Mem_0 Data bit 51
149	Mem_0 .Data[52]	N2	Bi-dir	Mem_0 Data bit 52
150	Mem_0 .Data[53]	K5	Bi-dir	Mem_0 Data bit 53
151	Mem_0 .Data[54]	M1	Bi-dir	Mem_0 Data bit 54
152	Mem_0 .Data[55]	J3	Bi-dir	Mem_0 Data bit 55
153	Mem_0 .Data[56]	M3	Bi-dir	Mem_0 Data bit 56
154	Mem_0 .Data[57]	J4	Bi-dir	Mem_0 Data bit 57
155	Mem_0 .Data[58]	M2	Bi-dir	Mem_0 Data bit 58
156	Mem_0 .Data[59]	J5	Bi-dir	Mem_0 Data bit 59
157	Mem_0 .Data[60]	L1	Bi-dir	Mem_0 Data bit 60
158	Mem_0 .Data[61]	L3	Bi-dir	Mem_0 Data bit 61
159	Mem_0 .Data[62]	L2	Bi-dir	Mem_0 Data bit 62
160	Mem_0 .Data[63]	H3	Bi-dir	Mem_0 Data bit 63
161	Mem_0 . Addr_CS_n	C4	Output	Mem_0 Address Chip Select
162	Mem_0 . CSH_n	E2	Output	Mem_0 Chip Select High

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
163	Mem_0 . CSL_n	D3	Output	Mem_0 Chip Select Low
164	Mem_0 . CE_n	E4	Output	Mem_0 Clock Enable
165	Mem_0 . WE_n	A5	Output	Mem_0 Write Enable
166	Mem_1 .Addr[00]	AG2	Output	Mem_1 Address bit 0
167	Mem_1 .Addr[01]	AJ5	Output	Mem_1 Address bit 1
168	Mem_1 .Addr[02]	AG1	Output	Mem_1 Address bit 2
169	Mem_1 .Addr[03]	AJ4	Output	Mem_1 Address bit 3
170	Mem_1 .Addr[04]	AF3	Output	Mem_1 Address bit 4
171	Mem_1 .Addr[05]	AH5	Output	Mem_1 Address bit 5
172	Mem_1 .Addr[06]	AF2	Output	Mem_1 Address bit 6
173	Mem_1 .Addr[07]	AH4	Output	Mem_1 Address bit 7
174	Mem_1 .Addr[08]	AF1	Output	Mem_1 Address bit 8
175	Mem_1 .Addr[09]	AG5	Output	Mem_1 Address bit 9
176	Mem_1 .Addr[10]	AE3	Output	Mem_1 Address bit 10
177	Mem_1 .Addr[11]	AG4	Output	Mem_1 Address bit 11
178	Mem_1 .Addr[12]	AE2	Output	Mem_1 Address bit 12
179	Mem_1 .Addr[13]	AF5	Output	Mem_1 Address bit 13
180	Mem_1 .Addr[14]	AD2	Output	Mem_1 Address bit 14
181	Mem_1 .Addr[15]	AE5	Output	Mem_1 Address bit 15
182	Mem_1 .Addr[16]	AD3	Output	Mem_1 Address bit 16
183	Mem_1 .Addr[17]	AE4	Output	Mem_1 Address bit 17
184	Mem_1 .Addr[18]	AF4	Output	Mem_1 Address bit 18
185	Mem_1 .Data[00]	BB8	Bi-dir	Mem_1 Data bit 0
186	Mem_1 .Data[01]	AY7	Bi-dir	Mem_1 Data bit 1
187	Mem_1 .Data[02]	BA8	Bi-dir	Mem_1 Data bit 2
188	Mem_1 .Data[03]	AW7	Bi-dir	Mem_1 Data bit 3
189	Mem_1 .Data[04]	BB7	Bi-dir	Mem_1 Data bit 4
190	Mem_1 .Data[05]	AV7	Bi-dir	Mem_1 Data bit 5
191	Mem_1 .Data[06]	BA7	Bi-dir	Mem_1 Data bit 6
192	Mem_1 .Data[07]	AY6	Bi-dir	Mem_1 Data bit 7
193	Mem_1 .Data[08]	BB6	Bi-dir	Mem_1 Data bit 8
194	Mem_1 .Data[09]	AY5	Bi-dir	Mem_1 Data bit 9
195	Mem_1 .Data[10]	BA6	Bi-dir	Mem_1 Data bit 10
196	Mem_1 .Data[11]	BB5	Bi-dir	Mem_1 Data bit 11
197	Mem_1 .Data[12]	AW6	Bi-dir	Mem_1 Data bit 12
198	Mem_1 .Data[13]	BA5	Bi-dir	Mem_1 Data bit 13
199	Mem_1 .Data[14]	AY4	Bi-dir	Mem_1 Data bit 14
200	Mem_1 .Data[15]	BA4	Bi-dir	Mem_1 Data bit 15
201	Mem_1 .Data[16]	AV6	Bi-dir	Mem_1 Data bit 16
202	Mem_1 .Data[17]	AW3	Bi-dir	Mem_1 Data bit 17
203	Mem_1 .Data[18]	AU2	Bi-dir	Mem_1 Data bit 18
204	Mem_1 .Data[19]	AU5	Bi-dir	Mem_1 Data bit 19
205	Mem_1 .Data[20]	AU1	Bi-dir	Mem_1 Data bit 20
206	Mem_1 .Data[21]	AT2	Bi-dir	Mem_1 Data bit 21
207	Mem_1 .Data[22]	AV4	Bi-dir	Mem_1 Data bit 22
208	Mem_1 .Data[23]	AT1	Bi-dir	Mem_1 Data bit 23
209	Mem_1 .Data[24]	AR2	Bi-dir	Mem_1 Data bit 24

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
210	Mem_1 .Data[25]	AV3	Bi-dir	Mem_1 Data bit 25
211	Mem_1 .Data[26]	AR1	Bi-dir	Mem_1 Data bit 26
212	Mem_1 .Data[27]	AP2	Bi-dir	Mem_1 Data bit 27
213	Mem_1 .Data[28]	AP1	Bi-dir	Mem_1 Data bit 28
214	Mem_1 .Data[29]	AU4	Bi-dir	Mem_1 Data bit 29
215	Mem_1 .Data[30]	AN2	Bi-dir	Mem_1 Data bit 30
216	Mem_1 .Data[31]	AU3	Bi-dir	Mem_1 Data bit 31
217	Mem_1 .Data[32]	AN1	Bi-dir	Mem_1 Data bit 32
218	Mem_1 .Data[33]	AT5	Bi-dir	Mem_1 Data bit 33
219	Mem_1 .Data[34]	AM3	Bi-dir	Mem_1 Data bit 34
220	Mem_1 .Data[35]	AT4	Bi-dir	Mem_1 Data bit 35
221	Mem_1 .Data[36]	AM2	Bi-dir	Mem_1 Data bit 36
222	Mem_1 .Data[37]	AT3	Bi-dir	Mem_1 Data bit 37
223	Mem_1 .Data[38]	AM1	Bi-dir	Mem_1 Data bit 38
224	Mem_1 .Data[39]	AR5	Bi-dir	Mem_1 Data bit 39
225	Mem_1 .Data[40]	AL3	Bi-dir	Mem_1 Data bit 40
226	Mem_1 .Data[41]	AR4	Bi-dir	Mem_1 Data bit 41
227	Mem_1 .Data[42]	AL2	Bi-dir	Mem_1 Data bit 42
228	Mem_1 .Data[43]	AR3	Bi-dir	Mem_1 Data bit 43
229	Mem_1 .Data[44]	AL1	Bi-dir	Mem_1 Data bit 44
230	Mem_1 .Data[45]	AP5	Bi-dir	Mem_1 Data bit 45
231	Mem_1 .Data[46]	AK3	Bi-dir	Mem_1 Data bit 46
232	Mem_1 .Data[47]	AP4	Bi-dir	Mem_1 Data bit 47
233	Mem_1 .Data[48]	AK2	Bi-dir	Mem_1 Data bit 48
234	Mem_1 .Data[49]	AP3	Bi-dir	Mem_1 Data bit 49
235	Mem_1 .Data[50]	AK1	Bi-dir	Mem_1 Data bit 50
236	Mem_1 .Data[51]	AN5	Bi-dir	Mem_1 Data bit 51
237	Mem_1 .Data[52]	AJ3	Bi-dir	Mem_1 Data bit 52
238	Mem_1 .Data[53]	AN4	Bi-dir	Mem_1 Data bit 53
239	Mem_1 .Data[54]	AJ2	Bi-dir	Mem_1 Data bit 54
240	Mem_1 .Data[55]	AN3	Bi-dir	Mem_1 Data bit 55
241	Mem_1 .Data[56]	AH3	Bi-dir	Mem_1 Data bit 56
242	Mem_1 .Data[57]	AM4	Bi-dir	Mem_1 Data bit 57
243	Mem_1 .Data[58]	AH2	Bi-dir	Mem_1 Data bit 58
244	Mem_1 .Data[59]	AL5	Bi-dir	Mem_1 Data bit 59
245	Mem_1 .Data[60]	AH1	Bi-dir	Mem_1 Data bit 60
246	Mem_1 .Data[61]	AL4	Bi-dir	Mem_1 Data bit 61
247	Mem_1 .Data[62]	AG3	Bi-dir	Mem_1 Data bit 62
248	Mem_1 .Data[63]	AK4	Bi-dir	Mem_1 Data bit 63
249	Mem_1 . Addr_CS_n	AC3	Output	Mem_1 Address Chip Select
250	Mem_1 . CSH_n	AD5	Output	Mem_1 Chip Select High
251	Mem_1 . CSL_n	AD1	Output	Mem_1 Chip Select Low
252	Mem_1 . CE_n	AC1	Output	Mem_1 Clock Enable
253	Mem_1 . WE_n	AD4	Output	Mem_1 Write Enable
254	Mem_2 .Addr[00]	AF42	Output	Mem_2 Address bit 0
255	Mem_2 .Addr[01]	AJ39	Output	Mem_2 Address bit 1
256	Mem_2 .Addr[02]	AF40	Output	Mem_2 Address bit 2

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
257	Mem_2 .Addr[03]	AF41	Output	Mem_2 Address bit 3
258	Mem_2 .Addr[04]	AH38	Output	Mem_2 Address bit 4
259	Mem_2 .Addr[05]	AE42	Output	Mem_2 Address bit 5
260	Mem_2 .Addr[06]	AH39	Output	Mem_2 Address bit 6
261	Mem_2 .Addr[07]	AE40	Output	Mem_2 Address bit 7
262	Mem_2 .Addr[08]	AG38	Output	Mem_2 Address bit 8
263	Mem_2 .Addr[09]	AE41	Output	Mem_2 Address bit 9
264	Mem_2 .Addr[10]	AG39	Output	Mem_2 Address bit 10
265	Mem_2 .Addr[11]	AD42	Output	Mem_2 Address bit 11
266	Mem_2 .Addr[12]	AF38	Output	Mem_2 Address bit 12
267	Mem_2 .Addr[13]	AD40	Output	Mem_2 Address bit 13
268	Mem_2 .Addr[14]	AF39	Output	Mem_2 Address bit 14
269	Mem_2 .Addr[15]	AD41	Output	Mem_2 Address bit 15
270	Mem_2 .Addr[16]	AE38	Output	Mem_2 Address bit 16
271	Mem_2 .Addr[17]	AC42	Output	Mem_2 Address bit 17
272	Mem_2 .Addr[18]	AE39	Output	Mem_2 Address bit 18
273	Mem_2 .Data[00]	BA35	Bi-dir	Mem_2 Data bit 0
274	Mem_2 .Data[01]	AY37	Bi-dir	Mem_2 Data bit 1
275	Mem_2 .Data[02]	BB36	Bi-dir	Mem_2 Data bit 2
276	Mem_2 .Data[03]	BA36	Bi-dir	Mem_2 Data bit 3
277	Mem_2 .Data[04]	AW37	Bi-dir	Mem_2 Data bit 4
278	Mem_2 .Data[05]	BB37	Bi-dir	Mem_2 Data bit 5
279	Mem_2 .Data[06]	BA37	Bi-dir	Mem_2 Data bit 6
280	Mem_2 .Data[07]	AY38	Bi-dir	Mem_2 Data bit 7
281	Mem_2 .Data[08]	BB38	Bi-dir	Mem_2 Data bit 8
282	Mem_2 .Data[09]	AY39	Bi-dir	Mem_2 Data bit 9
283	Mem_2 .Data[10]	AV41	Bi-dir	Mem_2 Data bit 10
284	Mem_2 .Data[11]	AV42	Bi-dir	Mem_2 Data bit 11
285	Mem_2 .Data[12]	AW40	Bi-dir	Mem_2 Data bit 12
286	Mem_2 .Data[13]	AU41	Bi-dir	Mem_2 Data bit 13
287	Mem_2 .Data[14]	AV39	Bi-dir	Mem_2 Data bit 14
288	Mem_2 .Data[15]	AU42	Bi-dir	Mem_2 Data bit 15
289	Mem_2 .Data[16]	AT41	Bi-dir	Mem_2 Data bit 16
290	Mem_2 .Data[17]	AU38	Bi-dir	Mem_2 Data bit 17
291	Mem_2 .Data[18]	AT42	Bi-dir	Mem_2 Data bit 18
292	Mem_2 .Data[19]	AV40	Bi-dir	Mem_2 Data bit 19
293	Mem_2 .Data[20]	AR41	Bi-dir	Mem_2 Data bit 20
294	Mem_2 .Data[21]	AU39	Bi-dir	Mem_2 Data bit 21
295	Mem_2 .Data[22]	AR42	Bi-dir	Mem_2 Data bit 22
296	Mem_2 .Data[23]	AU40	Bi-dir	Mem_2 Data bit 23
297	Mem_2 .Data[24]	AP41	Bi-dir	Mem_2 Data bit 24
298	Mem_2 .Data[25]	AT38	Bi-dir	Mem_2 Data bit 25
299	Mem_2 .Data[26]	AP42	Bi-dir	Mem_2 Data bit 26
300	Mem_2 .Data[27]	AN41	Bi-dir	Mem_2 Data bit 27
301	Mem_2 .Data[28]	AT39	Bi-dir	Mem_2 Data bit 28
302	Mem_2 .Data[29]	AN42	Bi-dir	Mem_2 Data bit 29
303	Mem_2 .Data[30]	AT40	Bi-dir	Mem_2 Data bit 30

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
304	Mem_2 .Data[31]	AM40	Bi-dir	Mem_2 Data bit 31
305	Mem_2 .Data[32]	AR38	Bi-dir	Mem_2 Data bit 32
306	Mem_2 .Data[33]	AM41	Bi-dir	Mem_2 Data bit 33
307	Mem_2 .Data[34]	AR39	Bi-dir	Mem_2 Data bit 34
308	Mem_2 .Data[35]	AM42	Bi-dir	Mem_2 Data bit 35
309	Mem_2 .Data[36]	AR40	Bi-dir	Mem_2 Data bit 36
310	Mem_2 .Data[37]	AL40	Bi-dir	Mem_2 Data bit 37
311	Mem_2 .Data[38]	AP38	Bi-dir	Mem_2 Data bit 38
312	Mem_2 .Data[39]	AL41	Bi-dir	Mem_2 Data bit 39
313	Mem_2 .Data[40]	AP39	Bi-dir	Mem_2 Data bit 40
314	Mem_2 .Data[41]	AL42	Bi-dir	Mem_2 Data bit 41
315	Mem_2 .Data[42]	AP40	Bi-dir	Mem_2 Data bit 42
316	Mem_2 .Data[43]	AK40	Bi-dir	Mem_2 Data bit 43
317	Mem_2 .Data[44]	AN38	Bi-dir	Mem_2 Data bit 44
318	Mem_2 .Data[45]	AK41	Bi-dir	Mem_2 Data bit 45
319	Mem_2 .Data[46]	AN39	Bi-dir	Mem_2 Data bit 46
320	Mem_2 .Data[47]	AK42	Bi-dir	Mem_2 Data bit 47
321	Mem_2 .Data[48]	AN40	Bi-dir	Mem_2 Data bit 48
322	Mem_2 .Data[49]	AJ40	Bi-dir	Mem_2 Data bit 49
323	Mem_2 .Data[50]	AM38	Bi-dir	Mem_2 Data bit 50
324	Mem_2 .Data[51]	AJ41	Bi-dir	Mem_2 Data bit 51
325	Mem_2 .Data[52]	AJ42	Bi-dir	Mem_2 Data bit 52
326	Mem_2 .Data[53]	AM39	Bi-dir	Mem_2 Data bit 53
327	Mem_2 .Data[54]	AH40	Bi-dir	Mem_2 Data bit 54
328	Mem_2 .Data[55]	AH41	Bi-dir	Mem_2 Data bit 55
329	Mem_2 .Data[56]	AL38	Bi-dir	Mem_2 Data bit 56
330	Mem_2 .Data[57]	AH42	Bi-dir	Mem_2 Data bit 57
331	Mem_2 .Data[58]	AL39	Bi-dir	Mem_2 Data bit 58
332	Mem_2 .Data[59]	AG41	Bi-dir	Mem_2 Data bit 59
333	Mem_2 .Data[60]	AK39	Bi-dir	Mem_2 Data bit 60
334	Mem_2 .Data[61]	AG40	Bi-dir	Mem_2 Data bit 61
335	Mem_2 .Data[62]	AJ38	Bi-dir	Mem_2 Data bit 62
336	Mem_2 .Data[63]	AG42	Bi-dir	Mem_2 Data bit 63
337	Mem_2 . Addr_CS_n	AD39	Output	Mem_2 Address Chip Select
338	Mem_2 . CSH_n	AD38	Output	Mem_2 Chip Select High
339	Mem_2 . CSL_n	AC40	Output	Mem_2 Chip Select Low
340	Mem_2 . CE_n	AC41	Output	Mem_2 Clock Enable
341	Mem_2 . WE_n	AB42	Output	Mem_2 Write Enable
342	Mem_3 .Addr[00]	H40	Output	Mem_3 Address bit 0
343	Mem_3 .Addr[01]	K42	Output	Mem_3 Address bit 1
344	Mem_3 .Addr[02]	H39	Output	Mem_3 Address bit 2
345	Mem_3 .Addr[03]	K41	Output	Mem_3 Address bit 3
346	Mem_3 .Addr[04]	H38	Output	Mem_3 Address bit 4
347	Mem_3 .Addr[05]	J42	Output	Mem_3 Address bit 5
348	Mem_3 .Addr[06]	J41	Output	Mem_3 Address bit 6
349	Mem_3 .Addr[07]	G40	Output	Mem_3 Address bit 7
350	Mem_3 .Addr[08]	H42	Output	Mem_3 Address bit 8

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
351	Mem_3 .Addr[09]	G39	Output	Mem_3 Address bit 9
352	Mem_3 .Addr[10]	H41	Output	Mem_3 Address bit 10
353	Mem_3 .Addr[11]	G38	Output	Mem_3 Address bit 11
354	Mem_3 .Addr[12]	G42	Output	Mem_3 Address bit 12
355	Mem_3 .Addr[13]	G41	Output	Mem_3 Address bit 13
356	Mem_3 .Addr[14]	F40	Output	Mem_3 Address bit 14
357	Mem_3 .Addr[15]	F42	Output	Mem_3 Address bit 15
358	Mem_3 .Addr[16]	F41	Output	Mem_3 Address bit 16
359	Mem_3 .Addr[17]	F39	Output	Mem_3 Address bit 17
360	Mem_3 .Addr[18]	E42	Output	Mem_3 Address bit 18
361	Mem_3 .Data[00]	AC38	Bi-dir	Mem_3 Data bit 0
362	Mem_3 .Data[01]	AB40	Bi-dir	Mem_3 Data bit 1
363	Mem_3 .Data[02]	AC39	Bi-dir	Mem_3 Data bit 2
364	Mem_3 .Data[03]	AB41	Bi-dir	Mem_3 Data bit 3
365	Mem_3 .Data[04]	AA40	Bi-dir	Mem_3 Data bit 4
366	Mem_3 .Data[05]	AA41	Bi-dir	Mem_3 Data bit 5
367	Mem_3 .Data[06]	AB39	Bi-dir	Mem_3 Data bit 6
368	Mem_3 .Data[07]	AA42	Bi-dir	Mem_3 Data bit 7
369	Mem_3 .Data[08]	Y41	Bi-dir	Mem_3 Data bit 8
370	Mem_3 .Data[09]	AA39	Bi-dir	Mem_3 Data bit 9
371	Mem_3 .Data[10]	Y40	Bi-dir	Mem_3 Data bit 10
372	Mem_3 .Data[11]	Y39	Bi-dir	Mem_3 Data bit 11
373	Mem_3 .Data[12]	Y42	Bi-dir	Mem_3 Data bit 12
374	Mem_3 .Data[13]	Y38	Bi-dir	Mem_3 Data bit 13
375	Mem_3 .Data[14]	W41	Bi-dir	Mem_3 Data bit 14
376	Mem_3 .Data[15]	W40	Bi-dir	Mem_3 Data bit 15
377	Mem_3 .Data[16]	W39	Bi-dir	Mem_3 Data bit 16
378	Mem_3 .Data[17]	W42	Bi-dir	Mem_3 Data bit 17
379	Mem_3 .Data[18]	W38	Bi-dir	Mem_3 Data bit 18
380	Mem_3 .Data[19]	V41	Bi-dir	Mem_3 Data bit 19
381	Mem_3 .Data[20]	V39	Bi-dir	Mem_3 Data bit 20
382	Mem_3 .Data[21]	V40	Bi-dir	Mem_3 Data bit 21
383	Mem_3 .Data[22]	V38	Bi-dir	Mem_3 Data bit 22
384	Mem_3 .Data[23]	V42	Bi-dir	Mem_3 Data bit 23
385	Mem_3 .Data[24]	U39	Bi-dir	Mem_3 Data bit 24
386	Mem_3 .Data[25]	U41	Bi-dir	Mem_3 Data bit 25
387	Mem_3 .Data[26]	U38	Bi-dir	Mem_3 Data bit 26
388	Mem_3 .Data[27]	U40	Bi-dir	Mem_3 Data bit 27
389	Mem_3 .Data[28]	U42	Bi-dir	Mem_3 Data bit 28
390	Mem_3 .Data[29]	T39	Bi-dir	Mem_3 Data bit 29
391	Mem_3 .Data[30]	T41	Bi-dir	Mem_3 Data bit 30
392	Mem_3 .Data[31]	T38	Bi-dir	Mem_3 Data bit 31
393	Mem_3 .Data[32]	T40	Bi-dir	Mem_3 Data bit 32
394	Mem_3 .Data[33]	R39	Bi-dir	Mem_3 Data bit 33
395	Mem_3 .Data[34]	T42	Bi-dir	Mem_3 Data bit 34
396	Mem_3 .Data[35]	R42	Bi-dir	Mem_3 Data bit 35
397	Mem_3 .Data[36]	R38	Bi-dir	Mem_3 Data bit 36

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
398	Mem_3 .Data[37]	R40	Bi-dir	Mem_3 Data bit 37
399	Mem_3 .Data[38]	P39	Bi-dir	Mem_3 Data bit 38
400	Mem_3 .Data[39]	R41	Bi-dir	Mem_3 Data bit 39
401	Mem_3 .Data[40]	P38	Bi-dir	Mem_3 Data bit 40
402	Mem_3 .Data[41]	P42	Bi-dir	Mem_3 Data bit 41
403	Mem_3 .Data[42]	N39	Bi-dir	Mem_3 Data bit 42
404	Mem_3 .Data[43]	P40	Bi-dir	Mem_3 Data bit 43
405	Mem_3 .Data[44]	M39	Bi-dir	Mem_3 Data bit 44
406	Mem_3 .Data[45]	P41	Bi-dir	Mem_3 Data bit 45
407	Mem_3 .Data[46]	M38	Bi-dir	Mem_3 Data bit 46
408	Mem_3 .Data[47]	N42	Bi-dir	Mem_3 Data bit 47
409	Mem_3 .Data[48]	L39	Bi-dir	Mem_3 Data bit 48
410	Mem_3 .Data[49]	N40	Bi-dir	Mem_3 Data bit 49
411	Mem_3 .Data[50]	L38	Bi-dir	Mem_3 Data bit 50
412	Mem_3 .Data[51]	N41	Bi-dir	Mem_3 Data bit 51
413	Mem_3 .Data[52]	K40	Bi-dir	Mem_3 Data bit 52
414	Mem_3 .Data[53]	M42	Bi-dir	Mem_3 Data bit 53
415	Mem_3 .Data[54]	K39	Bi-dir	Mem_3 Data bit 54
416	Mem_3 .Data[55]	M40	Bi-dir	Mem_3 Data bit 55
417	Mem_3 .Data[56]	K38	Bi-dir	Mem_3 Data bit 56
418	Mem_3 .Data[57]	M41	Bi-dir	Mem_3 Data bit 57
419	Mem_3 .Data[58]	J40	Bi-dir	Mem_3 Data bit 58
420	Mem_3 .Data[59]	L42	Bi-dir	Mem_3 Data bit 59
421	Mem_3 .Data[60]	J39	Bi-dir	Mem_3 Data bit 60
422	Mem_3 .Data[61]	L40	Bi-dir	Mem_3 Data bit 61
423	Mem_3 .Data[62]	J38	Bi-dir	Mem_3 Data bit 62
424	Mem_3 .Data[63]	L41	Bi-dir	Mem_3 Data bit 63
425	Mem_3 . Addr_CS_n	D41	Output	Mem_3 Address Chip Select
426	Mem_3 . CSH_n	E41	Output	Mem_3 Chip Select High
427	Mem_3 . CSL_n	E40	Output	Mem_3 Chip Select Low
428	Mem_3 . CE_n	E39	Output	Mem_3 Clock Enable
429	Mem_3 . WE_n	F38	Output	Mem_3 Write Enable
430	Mem_4 .Addr[00]	AY33	Output	Mem_4 Address bit 0
431	Mem_4 .Addr[01]	BA30	Output	Mem_4 Address bit 1
432	Mem_4 .Addr[02]	AW33	Output	Mem_4 Address bit 2
433	Mem_4 .Addr[03]	BB31	Output	Mem_4 Address bit 3
434	Mem_4 .Addr[04]	AV33	Output	Mem_4 Address bit 4
435	Mem_4 .Addr[05]	AY31	Output	Mem_4 Address bit 5
436	Mem_4 .Addr[06]	AY34	Output	Mem_4 Address bit 6
437	Mem_4 .Addr[07]	BA31	Output	Mem_4 Address bit 7
438	Mem_4 .Addr[08]	AW34	Output	Mem_4 Address bit 8
439	Mem_4 .Addr[09]	BB32	Output	Mem_4 Address bit 9
440	Mem_4 .Addr[10]	AV34	Output	Mem_4 Address bit 10
441	Mem_4 .Addr[11]	BA32	Output	Mem_4 Address bit 11
442	Mem_4 .Addr[12]	AY35	Output	Mem_4 Address bit 12
443	Mem_4 .Addr[13]	BB33	Output	Mem_4 Address bit 13
444	Mem_4 .Addr[14]	AW35	Output	Mem_4 Address bit 14

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
445	Mem_4 .Addr[15]	BA33	Output	Mem_4 Address bit 15
446	Mem_4 .Addr[16]	AV35	Output	Mem_4 Address bit 16
447	Mem_4 .Addr[17]	BB34	Output	Mem_4 Address bit 17
448	Mem_4 .Addr[18]	AY36	Output	Mem_4 Address bit 18
449	Mem_4 .Data[00]	BB25	Bi-dir	Mem_4 Data bit 0
450	Mem_4 .Data[01]	AW26	Bi-dir	Mem_4 Data bit 1
451	Mem_4 .Data[02]	AY25	Bi-dir	Mem_4 Data bit 2
452	Mem_4 .Data[03]	AV26	Bi-dir	Mem_4 Data bit 3
453	Mem_4 .Data[04]	BA25	Bi-dir	Mem_4 Data bit 4
454	Mem_4 .Data[05]	AW27	Bi-dir	Mem_4 Data bit 5
455	Mem_4 .Data[06]	BB26	Bi-dir	Mem_4 Data bit 6
456	Mem_4 .Data[07]	AV27	Bi-dir	Mem_4 Data bit 7
457	Mem_4 .Data[08]	AY26	Bi-dir	Mem_4 Data bit 8
458	Mem_4 .Data[09]	AU27	Bi-dir	Mem_4 Data bit 9
459	Mem_4 .Data[10]	BA26	Bi-dir	Mem_4 Data bit 10
460	Mem_4 .Data[11]	AW28	Bi-dir	Mem_4 Data bit 11
461	Mem_4 .Data[12]	BB27	Bi-dir	Mem_4 Data bit 12
462	Mem_4 .Data[13]	AY27	Bi-dir	Mem_4 Data bit 13
463	Mem_4 .Data[14]	AV28	Bi-dir	Mem_4 Data bit 14
464	Mem_4 .Data[15]	BA27	Bi-dir	Mem_4 Data bit 15
465	Mem_4 .Data[16]	AW29	Bi-dir	Mem_4 Data bit 16
466	Mem_4 .Data[17]	BB28	Bi-dir	Mem_4 Data bit 17
467	Mem_4 .Data[18]	AV29	Bi-dir	Mem_4 Data bit 18
468	Mem_4 .Data[19]	AY28	Bi-dir	Mem_4 Data bit 19
469	Mem_4 .Data[20]	AW30	Bi-dir	Mem_4 Data bit 20
470	Mem_4 .Data[21]	BA28	Bi-dir	Mem_4 Data bit 21
471	Mem_4 .Data[22]	AW31	Bi-dir	Mem_4 Data bit 22
472	Mem_4 .Data[23]	BB29	Bi-dir	Mem_4 Data bit 23
473	Mem_4 .Data[24]	AV31	Bi-dir	Mem_4 Data bit 24
474	Mem_4 .Data[25]	AY29	Bi-dir	Mem_4 Data bit 25
475	Mem_4 .Data[26]	AY32	Bi-dir	Mem_4 Data bit 26
476	Mem_4 .Data[27]	BA29	Bi-dir	Mem_4 Data bit 27
477	Mem_4 .Data[28]	AW32	Bi-dir	Mem_4 Data bit 28
478	Mem_4 .Data[29]	BB30	Bi-dir	Mem_4 Data bit 29
479	Mem_4 .Data[30]	AV32	Bi-dir	Mem_4 Data bit 30
480	Mem_4 .Data[31]	AY30	Bi-dir	Mem_4 Data bit 31
481	Mem_4 . Addr_CS_n	BB35	Bi-dir	Mem_4 Address Chip Select
482	Mem_4 . CS_n	BA34	Bi-dir	Mem_4 Chip Select
483	Mem_4 . CE_n	AW36	Bi-dir	Mem_4 Clock Enable
484	Mem_4 . WE_n	AV36	Bi-dir	Mem_4 Write Enable
485	IO_Conn[0]	D27	Bi-dir	I/O Connector bit 0
486	IO_Conn[1]	C25	Bi-dir	I/O Connector bit 1
487	IO_Conn[2]	E27	Bi-dir	I/O Connector bit 2
488	IO_Conn[3]	A26	Bi-dir	I/O Connector bit 3
489	IO_Conn[4]	F27	Bi-dir	I/O Connector bit 4
490	IO_Conn[5]	B26	Bi-dir	I/O Connector bit 5
491	IO_Conn[6]	D28	Bi-dir	I/O Connector bit 6

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
492	IO_Conn[7]	C26	Bi-dir	I/O Connector bit 7
493	IO_Conn[8]	E28	Bi-dir	I/O Connector bit 8
494	IO_Conn[9]	A27	Bi-dir	I/O Connector bit 9
495	IO_Conn[10]	D29	Bi-dir	I/O Connector bit 10
496	IO_Conn[11]	B27	Bi-dir	I/O Connector bit 11
497	IO_Conn[12]	E29	Bi-dir	I/O Connector bit 12
498	IO_Conn[13]	C27	Bi-dir	I/O Connector bit 13
499	IO_Conn[14]	D30	Bi-dir	I/O Connector bit 14
500	IO_Conn[15]	A28	Bi-dir	I/O Connector bit 15
501	IO_Conn[16]	D31	Bi-dir	I/O Connector bit 16
502	IO_Conn[17]	B28	Bi-dir	I/O Connector bit 17
503	IO_Conn[18]	E31	Bi-dir	I/O Connector bit 18
504	IO_Conn[19]	C28	Bi-dir	I/O Connector bit 19
505	IO_Conn[20]	D32	Bi-dir	I/O Connector bit 20
506	IO_Conn[21]	A29	Bi-dir	I/O Connector bit 21
507	IO_Conn[22]	E32	Bi-dir	I/O Connector bit 22
508	IO_Conn[23]	B29	Bi-dir	I/O Connector bit 23
509	IO_Conn[24]	C33	Bi-dir	I/O Connector bit 24
510	IO_Conn[25]	C29	Bi-dir	I/O Connector bit 25
511	IO_Conn[26]	D33	Bi-dir	I/O Connector bit 26
512	IO_Conn[27]	A30	Bi-dir	I/O Connector bit 27
513	IO_Conn[28]	E33	Bi-dir	I/O Connector bit 28
514	IO_Conn[29]	B30	Bi-dir	I/O Connector bit 29
515	IO_Conn[30]	C34	Bi-dir	I/O Connector bit 30
516	IO_Conn[31]	C30	Bi-dir	I/O Connector bit 31
517	IO_Conn[32]	D34	Bi-dir	I/O Connector bit 32
518	IO_Conn[33]	A31	Bi-dir	I/O Connector bit 33
519	IO_Conn[34]	E34	Bi-dir	I/O Connector bit 34
520	IO_Conn[35]	B31	Bi-dir	I/O Connector bit 35
521	IO_Conn[36]	C35	Bi-dir	I/O Connector bit 36
522	IO_Conn[37]	C31	Bi-dir	I/O Connector bit 37
523	IO_Conn[38]	D35	Bi-dir	I/O Connector bit 38
524	IO_Conn[39]	A32	Bi-dir	I/O Connector bit 39
525	IO_Conn[40]	E35	Bi-dir	I/O Connector bit 40
526	IO_Conn[41]	B32	Bi-dir	I/O Connector bit 41
527	IO_Conn[42]	C36	Bi-dir	I/O Connector bit 42
528	IO_Conn[43]	C32	Bi-dir	I/O Connector bit 43
529	IO_Conn[44]	A33	Bi-dir	I/O Connector bit 44
530	IO_Conn[45]	D36	Bi-dir	I/O Connector bit 45
531	IO_Conn[46]	B33	Bi-dir	I/O Connector bit 46
532	IO_Conn[47]	E36	Bi-dir	I/O Connector bit 47
533	IO_Conn[48]	A34	Bi-dir	I/O Connector bit 48
534	IO_Conn[49]	C37	Bi-dir	I/O Connector bit 49
535	IO_Conn[50]	B34	Bi-dir	I/O Connector bit 50
536	IO_Conn[51]	D37	Bi-dir	I/O Connector bit 51
537	IO_Conn[52]	A35	Bi-dir	I/O Connector bit 52
538	IO_Conn[53]	B35	Bi-dir	I/O Connector bit 53

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
539	IO_Conn[54]	A36	Bi-dir	I/O Connector bit 54
540	IO_Conn[55]	C38	Bi-dir	I/O Connector bit 55
541	IO_Conn[56]	B36	Bi-dir	I/O Connector bit 56
542	IO_Conn[57]	C39	Bi-dir	I/O Connector bit 57
543	IO_Conn[58]	A37	Bi-dir	I/O Connector bit 58
544	IO_Conn[59]	E37	Bi-dir	I/O Connector bit 59
545	IO_Conn[60]	B37	Bi-dir	I/O Connector bit 60
546	IO_Conn[61]	D38	Bi-dir	I/O Connector bit 61
547	IO_Conn[62]	A38	Bi-dir	I/O Connector bit 62
548	IO_Conn[63]	D40	Bi-dir	I/O Connector bit 63
549	IO_Conn[64]	AW25	Bi-dir	I/O Connector bit 64 - HandShake 0
550	IO_Conn[65]	BA24	Bi-dir	I/O Connector bit 65 - HandShake 1
551	IO_Conn[66]	B25	Bi-dir	I/O Connector bit 66
552	IO_Conn[67]	E26	Bi-dir	I/O Connector bit 67
553	IO_Conn[68]	A25	Bi-dir	I/O Connector bit 68
554	IO_Conn[69]	D26	Bi-dir	I/O Connector bit 69
555	IO_Conn[70]	A24	Bi-dir	I/O Connector bit 70
556	IO_Conn[71]	E25	Bi-dir	I/O Connector bit 71
557	IO_Conn[72]	C24	Bi-dir	I/O Connector bit 72
558	IO_Conn[73]	D25	Bi-dir	I/O Connector bit 73
559	IO_Conn[74]	B24	Bi-dir	I/O Connector bit 74
560	IO_Conn[75]	E24	Bi-dir	I/O Connector bit 75
561	IO_Conn[76]	A23	Bi-dir	I/O Connector bit 76
562	IO_Conn[77]	D24	Bi-dir	I/O Connector bit 77
563	IO_Conn[78]	C23	Bi-dir	I/O Connector bit 78
564	IO_Conn[79]	E23	Bi-dir	I/O Connector bit 79
565	IO_Conn[80]	B23	Bi-dir	I/O Connector bit 80
566	IO_Conn[81]	D23	Bi-dir	I/O Connector bit 81
567	IO_Conn[82]	A22	Bi-dir	I/O Connector bit 82
568	IO_Conn[83]	D22	Bi-dir	I/O Connector bit 83
569	IO_Conn[84]	C21	Bi-dir	I/O Connector bit 84
570	IO_Conn[85]	D21	Bi-dir	I/O Connector bit 85
571	IO_Conn[86]	B21	Bi-dir	I/O Connector bit 86
572	IO_Conn[87]	D20	Bi-dir	I/O Connector bit 87
573	IO_Conn[88]	A21	Bi-dir	I/O Connector bit 88
574	IO_Conn[89]	E20	Bi-dir	I/O Connector bit 89
575	IO_Conn[90]	C20	Bi-dir	I/O Connector bit 90
576	IO_Conn[91]	D19	Bi-dir	I/O Connector bit 91
577	IO_Conn[92]	B20	Bi-dir	I/O Connector bit 92
578	IO_Conn[93]	E19	Bi-dir	I/O Connector bit 93
579	IO_Conn[94]	A20	Bi-dir	I/O Connector bit 94
580	IO_Conn[95]	D18	Bi-dir	I/O Connector bit 95
581	IO_Conn[96]	A19	Bi-dir	I/O Connector bit 96
582	IO_Conn[97]	E18	Bi-dir	I/O Connector bit 97
583	IO_Conn[98]	C19	Bi-dir	I/O Connector bit 98
584	IO_Conn[99]	D17	Bi-dir	I/O Connector bit 99
585	IO_Conn[100]	B19	Bi-dir	I/O Connector bit 100

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
586	IO_Conn[101]	E17	Bi-dir	I/O Connector bit 101
587	IO_Conn[102]	A18	Bi-dir	I/O Connector bit 102
588	IO_Conn[103]	D16	Bi-dir	I/O Connector bit 103
589	IO_Conn[104]	C18	Bi-dir	I/O Connector bit 104
590	IO_Conn[105]	E16	Bi-dir	I/O Connector bit 105
591	IO_Conn[106]	B18	Bi-dir	I/O Connector bit 106
592	IO_Conn[107]	F16	Bi-dir	I/O Connector bit 107
593	IO_Conn[108]	A17	Bi-dir	I/O Connector bit 108
594	IO_Conn[109]	D15	Bi-dir	I/O Connector bit 109
595	IO_Conn[110]	C17	Bi-dir	I/O Connector bit 110
596	IO_Conn[111]	E15	Bi-dir	I/O Connector bit 111
597	IO_Conn[112]	B17	Bi-dir	I/O Connector bit 112
598	IO_Conn[113]	D14	Bi-dir	I/O Connector bit 113
599	IO_Conn[114]	A16	Bi-dir	I/O Connector bit 114
600	IO_Conn[115]	E14	Bi-dir	I/O Connector bit 115
601	IO_Conn[116]	C16	Bi-dir	I/O Connector bit 116
602	IO_Conn[117]	D13	Bi-dir	I/O Connector bit 117
603	IO_Conn[118]	B16	Bi-dir	I/O Connector bit 118
604	IO_Conn[119]	D12	Bi-dir	I/O Connector bit 119
605	IO_Conn[120]	A15	Bi-dir	I/O Connector bit 120
606	IO_Conn[121]	E12	Bi-dir	I/O Connector bit 121
607	IO_Conn[122]	C15	Bi-dir	I/O Connector bit 122
608	IO_Conn[123]	C11	Bi-dir	I/O Connector bit 123
609	IO_Conn[124]	B15	Bi-dir	I/O Connector bit 124
610	IO_Conn[125]	D11	Bi-dir	I/O Connector bit 125
611	IO_Conn[126]	A14	Bi-dir	I/O Connector bit 126
612	IO_Conn[127]	E11	Bi-dir	I/O Connector bit 127
613	IO_Conn[128]	C14	Bi-dir	I/O Connector bit 128
614	IO_Conn[129]	C10	Bi-dir	I/O Connector bit 129
615	IO_Conn[130]	B14	Bi-dir	I/O Connector bit 130
616	IO_Conn[131]	D10	Bi-dir	I/O Connector bit 131
617	IO_Conn[132]	A13	Bi-dir	I/O Connector bit 132
618	IO_Conn[133]	E10	Bi-dir	I/O Connector bit 133
619	IO_Conn[134]	C13	Bi-dir	I/O Connector bit 134
620	IO_Conn[135]	C9	Bi-dir	I/O Connector bit 135
621	IO_Conn[136]	B13	Bi-dir	I/O Connector bit 136
622	IO_Conn[137]	D9	Bi-dir	I/O Connector bit 137
623	IO_Conn[138]	A12	Bi-dir	I/O Connector bit 138
624	IO_Conn[139]	E9	Bi-dir	I/O Connector bit 139
625	IO_Conn[140]	C12	Bi-dir	I/O Connector bit 140
626	IO_Conn[141]	C8	Bi-dir	I/O Connector bit 141
627	IO_Conn[142]	B12	Bi-dir	I/O Connector bit 142
628	IO_Conn[143]	D8	Bi-dir	I/O Connector bit 143
629	IO_Conn[144]	A11	Bi-dir	I/O Connector bit 144
630	IO_Conn[145]	E8	Bi-dir	I/O Connector bit 145
631	IO_Conn[146]	B11	Bi-dir	I/O Connector bit 146
632	IO_Conn[147]	C7	Bi-dir	I/O Connector bit 147

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
633	IO_Conn[148]	A10	Bi-dir	I/O Connector bit 148
634	IO_Conn[149]	C6	Bi-dir	I/O Connector bit 149
635	IO_Conn[150]	B10	Bi-dir	I/O Connector bit 150
636	IO_Conn[151]	D7	Bi-dir	I/O Connector bit 151
637	IO_Conn[152]	A9	Bi-dir	I/O Connector bit 152
638	IO_Conn[153]	B9	Bi-dir	I/O Connector bit 153
639	IO_Conn[154]	A8	Bi-dir	I/O Connector bit 154
640	IO_Conn[155]	E7	Bi-dir	I/O Connector bit 155
641	IO_Conn[156]	B8	Bi-dir	I/O Connector bit 156
642	IO_Conn[157]	C5	Bi-dir	I/O Connector bit 157
643	IO_Conn[158]	A7	Bi-dir	I/O Connector bit 158
644	IO_Conn[159]	A6	Bi-dir	I/O Connector bit 159
645	IO_Conn[160]	B7	Bi-dir	I/O Connector bit 160
646	IO_Conn[161]	D6	Bi-dir	I/O Connector bit 161
647	LEDs.Red_n	AW5	Output	Light emitting diode
648	LEDs.Yellow_n	BB24	Output	Light emitting diode
649	LEDs.Green_n	AW24	Output	Light emitting diode
650	ToPCI1	AV10	Output	Spare pin between PCI & PE
651	Reset_n	AV24	Output	Reset from PCI

Table A-2: FIREBIRD™/PMC PE0 Pad Locations and Definitions

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
0	Clocks.M_Clk	BA22	Input	Memory clock
1	Clocks.K_Clk	B22	Input	PCI/LAD bus clock
2	Clocks.U_Clk	C22	Input	User clock
3	Clocks.RACE_Clk	A22	Input	RACEway input clock
4	Clocks.WSDPA_Clk	AY22	Input	BPE WSDPA Input Clock
5	Clocks.WSDPB_Clk	AW21	Input	BPE WSDPB Input Clock
6	Clocks.M_Clk_Src	N4	Output	Memory clock sync signal
7	Clocks.U_Clk_Src	E3	Output	User clock sync signal
8	LAD64_Bus.Addr_Data[00]	L4	Bi-dir	LAD bus shared address/data bus bit 0
9	LAD64_Bus.Addr_Data[01]	L5	Bi-dir	LAD bus shared address/data bus bit 1
10	LAD64_Bus.Addr_Data[02]	N1	Bi-dir	LAD bus shared address/data bus bit 2
11	LAD64_Bus.Addr_Data[03]	K3	Bi-dir	LAD bus shared address/data bus bit 3
12	LAD64_Bus.Addr_Data[04]	N3	Bi-dir	LAD bus shared address/data bus bit 4
13	LAD64_Bus.Addr_Data[05]	K4	Bi-dir	LAD bus shared address/data bus bit 5
14	LAD64_Bus.Addr_Data[06]	N2	Bi-dir	LAD bus shared address/data bus bit 6
15	LAD64_Bus.Addr_Data[07]	K5	Bi-dir	LAD bus shared address/data bus bit 7
16	LAD64_Bus.Addr_Data[08]	M1	Bi-dir	LAD bus shared address/data bus bit 8
17	LAD64_Bus.Addr_Data[09]	J3	Bi-dir	LAD bus shared address/data bus bit 9
18	LAD64_Bus.Addr_Data[10]	M3	Bi-dir	LAD bus shared address/data bus bit 10
19	LAD64_Bus.Addr_Data[11]	J4	Bi-dir	LAD bus shared address/data bus bit 11
20	LAD64_Bus.Addr_Data[12]	M2	Bi-dir	LAD bus shared address/data bus bit 12
21	LAD64_Bus.Addr_Data[13]	J5	Bi-dir	LAD bus shared address/data bus bit 13
22	LAD64_Bus.Addr_Data[14]	L1	Bi-dir	LAD bus shared address/data bus bit 14
23	LAD64_Bus.Addr_Data[15]	L3	Bi-dir	LAD bus shared address/data bus bit 15
24	LAD64_Bus.Addr_Data[16]	L2	Bi-dir	LAD bus shared address/data bus bit 16
25	LAD64_Bus.Addr_Data[17]	H3	Bi-dir	LAD bus shared address/data bus bit 17
26	LAD64_Bus.Addr_Data[18]	K1	Bi-dir	LAD bus shared address/data bus bit 18
27	LAD64_Bus.Addr_Data[19]	H4	Bi-dir	LAD bus shared address/data bus bit 19
28	LAD64_Bus.Addr_Data[20]	K2	Bi-dir	LAD bus shared address/data bus bit 20
29	LAD64_Bus.Addr_Data[21]	H5	Bi-dir	LAD bus shared address/data bus bit 21
30	LAD64_Bus.Addr_Data[22]	J1	Bi-dir	LAD bus shared address/data bus bit 22
31	LAD64_Bus.Addr_Data[23]	G3	Bi-dir	LAD bus shared address/data bus bit 23
32	LAD64_Bus.Addr_Data[24]	J2	Bi-dir	LAD bus shared address/data bus bit 24
33	LAD64_Bus.Addr_Data[25]	H1	Bi-dir	LAD bus shared address/data bus bit 25
34	LAD64_Bus.Addr_Data[26]	G4	Bi-dir	LAD bus shared address/data bus bit 26
35	LAD64_Bus.Addr_Data[27]	H2	Bi-dir	LAD bus shared address/data bus bit 27
36	LAD64_Bus.Addr_Data[28]	F3	Bi-dir	LAD bus shared address/data bus bit 28
37	LAD64_Bus.Addr_Data[29]	G1	Bi-dir	LAD bus shared address/data bus bit 29
38	LAD64_Bus.Addr_Data[30]	G5	Bi-dir	LAD bus shared address/data bus bit 30
39	LAD64_Bus.Addr_Data[31]	F1	Bi-dir	LAD bus shared address/data bus bit 31
40	LAD64_Bus.Addr_Data[32]	AA2	Bi-dir	LAD bus shared address/data bus bit 32
41	LAD64_Bus.Addr_Data[33]	AA4	Bi-dir	LAD bus shared address/data bus bit 33
42	LAD64_Bus.Addr_Data[34]	AA3	Bi-dir	LAD bus shared address/data bus bit 34
43	LAD64_Bus.Addr_Data[35]	Y4	Bi-dir	LAD bus shared address/data bus bit 35
44	LAD64_Bus.Addr_Data[36]	AA1	Bi-dir	LAD bus shared address/data bus bit 36

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
45	LAD64_Bus.Addr_Data[37]	Y5	Bi-dir	LAD bus shared address/data bus bit 37
46	LAD64_Bus.Addr_Data[38]	Y1	Bi-dir	LAD bus shared address/data bus bit 38
47	LAD64_Bus.Addr_Data[39]	Y3	Bi-dir	LAD bus shared address/data bus bit 39
48	LAD64_Bus.Addr_Data[40]	W4	Bi-dir	LAD bus shared address/data bus bit 40
49	LAD64_Bus.Addr_Data[41]	Y2	Bi-dir	LAD bus shared address/data bus bit 41
50	LAD64_Bus.Addr_Data[42]	W1	Bi-dir	LAD bus shared address/data bus bit 42
51	LAD64_Bus.Addr_Data[43]	W5	Bi-dir	LAD bus shared address/data bus bit 43
52	LAD64_Bus.Addr_Data[44]	W3	Bi-dir	LAD bus shared address/data bus bit 44
53	LAD64_Bus.Addr_Data[45]	V4	Bi-dir	LAD bus shared address/data bus bit 45
54	LAD64_Bus.Addr_Data[46]	W2	Bi-dir	LAD bus shared address/data bus bit 46
55	LAD64_Bus.Addr_Data[47]	V5	Bi-dir	LAD bus shared address/data bus bit 47
56	LAD64_Bus.Addr_Data[48]	V1	Bi-dir	LAD bus shared address/data bus bit 48
57	LAD64_Bus.Addr_Data[49]	U4	Bi-dir	LAD bus shared address/data bus bit 49
58	LAD64_Bus.Addr_Data[50]	U5	Bi-dir	LAD bus shared address/data bus bit 50
59	LAD64_Bus.Addr_Data[51]	V2	Bi-dir	LAD bus shared address/data bus bit 51
60	LAD64_Bus.Addr_Data[52]	T4	Bi-dir	LAD bus shared address/data bus bit 52
61	LAD64_Bus.Addr_Data[53]	U1	Bi-dir	LAD bus shared address/data bus bit 53
62	LAD64_Bus.Addr_Data[54]	T5	Bi-dir	LAD bus shared address/data bus bit 54
63	LAD64_Bus.Addr_Data[55]	U3	Bi-dir	LAD bus shared address/data bus bit 55
64	LAD64_Bus.Addr_Data[56]	R4	Bi-dir	LAD bus shared address/data bus bit 56
65	LAD64_Bus.Addr_Data[57]	U2	Bi-dir	LAD bus shared address/data bus bit 57
66	LAD64_Bus.Addr_Data[58]	R5	Bi-dir	LAD bus shared address/data bus bit 58
67	LAD64_Bus.Addr_Data[59]	T1	Bi-dir	LAD bus shared address/data bus bit 59
68	LAD64_Bus.Addr_Data[60]	P4	Bi-dir	LAD bus shared address/data bus bit 60
69	LAD64_Bus.Addr_Data[61]	T3	Bi-dir	LAD bus shared address/data bus bit 61
70	LAD64_Bus.Addr_Data[62]	P5	Bi-dir	LAD bus shared address/data bus bit 62
71	LAD64_Bus.Addr_Data[63]	T2	Bi-dir	LAD bus shared address/data bus bit 63
72	LAD64_Bus. DWordL_n	M5	Input	LAD bus data low strobe
73	LAD64_Bus. DWordH_n	R3	Input	LAD bus data high strobe
74	LAD64_Bus.WR_n	P1	Input	LAD bus write select
75	LAD64_Bus.CS_n	M4	Input	LAD bus chip select
76	LAD64_Bus.Int_Req_n	R1	Output	LAD bus interrupt request
77	LAD64_Bus.Req_n	R2	Input	LAD bus DMA/register mode select
78	LAD64_Bus.DMA_WStat[00]	AC2	Output	LAD bus DMA write status flag bit 0
79	LAD64_Bus.DMA_WStat[01]	AD1	Output	LAD bus DMA write status flag bit 1
80	LAD64_Bus.DMA_RStat[00]	AB4	Output	LAD bus DMA read status flag bit 0
81	LAD64_Bus.DMA_RStat[01]	AB2	Output	LAD bus DMA read status flag bit 1
82	LAD64_Bus.Reset_n	AW20	Input	LAD bus Reset bit
83	LEDs.Green_n	AJ2	Output	Green Light emitting diode
84	LEDs.Red_n	AH3	Output	Red Light emitting diode
85	LEDs.Yellow_n	AN3	Output	Yellow Light emitting diode
86	IO_Conn[0]	AC4	Bi-dir	I/O Connector bit 0
87	IO_Conn[1]	AB3	Bi-dir	I/O Connector bit 1
88	IO_Conn[2]	AC5	Bi-dir	I/O Connector bit 2
89	IO_Conn[3]	AB1	Bi-dir	I/O Connector bit 3
90	IO_Conn[4]	AC3	Bi-dir	I/O Connector bit 4
91	IO_Conn[5]	AD4	Bi-dir	I/O Connector bit 5

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
92	IO_Conn[6]	AD5	Bi-dir	I/O Connector bit 6
93	IO_Conn[7]	AC1	Bi-dir	I/O Connector bit 7
94	IO_Conn[8]	AD3	Bi-dir	I/O Connector bit 8
95	IO_Conn[9]	AE4	Bi-dir	I/O Connector bit 9
96	IO_Conn[10]	AD2	Bi-dir	I/O Connector bit 10
97	IO_Conn[11]	AE5	Bi-dir	I/O Connector bit 11
98	IO_Conn[12]	AG4	Bi-dir	I/O Connector bit 12
99	IO_Conn[13]	AE2	Bi-dir	I/O Connector bit 13
100	IO_Conn[14]	AF1	Bi-dir	I/O Connector bit 14
101	IO_Conn[15]	AG5	Bi-dir	I/O Connector bit 15
102	IO_Conn[16]	AF2	Bi-dir	I/O Connector bit 16
103	IO_Conn[17]	AH4	Bi-dir	I/O Connector bit 17
104	IO_Conn[18]	AJ4	Bi-dir	I/O Connector bit 18
105	IO_Conn[19]	AF3	Bi-dir	I/O Connector bit 19
106	IO_Conn[20]	AJ5	Bi-dir	I/O Connector bit 20
107	IO_Conn[21]	AG1	Bi-dir	I/O Connector bit 21
108	IO_Conn[22]	AK4	Bi-dir	I/O Connector bit 22
109	IO_Conn[23]	AG2	Bi-dir	I/O Connector bit 23
110	IO_Conn[24]	AL4	Bi-dir	I/O Connector bit 24
111	IO_Conn[25]	AG3	Bi-dir	I/O Connector bit 25
112	IO_Conn[26]	AL5	Bi-dir	I/O Connector bit 26
113	IO_Conn[27]	AH1	Bi-dir	I/O Connector bit 27
114	IO_Conn[28]	AM4	Bi-dir	I/O Connector bit 28
115	IO_Conn[29]	AH2	Bi-dir	I/O Connector bit 29
116	IO_Conn[30]	AJ3	Bi-dir	I/O Connector bit 30
117	IO_Conn[31]	AN4	Bi-dir	I/O Connector bit 31
118	IO_Conn[32]	AK1	Bi-dir	I/O Connector bit 32
119	IO_Conn[33]	AN5	Bi-dir	I/O Connector bit 33
120	IO_Conn[34]	AP4	Bi-dir	I/O Connector bit 34
121	IO_Conn[35]	AK2	Bi-dir	I/O Connector bit 35
122	IO_Conn[36]	AP5	Bi-dir	I/O Connector bit 36
123	IO_Conn[37]	AK3	Bi-dir	I/O Connector bit 37
124	IO_Conn[38]	AL2	Bi-dir	I/O Connector bit 38
125	IO_Conn[39]	AR3	Bi-dir	I/O Connector bit 39
126	IO_Conn[40]	AL3	Bi-dir	I/O Connector bit 40
127	IO_Conn[41]	AR4	Bi-dir	I/O Connector bit 41
128	IO_Conn[42]	AT3	Bi-dir	I/O Connector bit 42
129	IO_Conn[43]	AM1	Bi-dir	I/O Connector bit 43
130	IO_Conn[44]	AT4	Bi-dir	I/O Connector bit 44
131	IO_Conn[45]	AM2	Bi-dir	I/O Connector bit 45
132	IO_Conn[46]	AN1	Bi-dir	I/O Connector bit 46
133	IO_Conn[47]	AT5	Bi-dir	I/O Connector bit 47
134	IO_Conn[48]	AN2	Bi-dir	I/O Connector bit 48
135	IO_Conn[49]	AU3	Bi-dir	I/O Connector bit 49
136	IO_Conn[50]	AP2	Bi-dir	I/O Connector bit 50
137	IO_Conn[51]	AP1	Bi-dir	I/O Connector bit 51
138	IO_Conn[52]	AV3	Bi-dir	I/O Connector bit 52

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
139	IO_Conn[53]	AR1	Bi-dir	I/O Connector bit 53
140	IO_Conn[54]	AT1	Bi-dir	I/O Connector bit 54
141	IO_Conn[55]	AR2	Bi-dir	I/O Connector bit 55
142	IO_Conn[56]	AT2	Bi-dir	I/O Connector bit 56
143	IO_Conn[57]	AV4	Bi-dir	I/O Connector bit 57
144	IO_Conn[58]	AU5	Bi-dir	I/O Connector bit 58
145	IO_Conn[59]	AU1	Bi-dir	I/O Connector bit 59
146	IO_Conn[60]	AW3	Bi-dir	I/O Connector bit 60
147	IO_Conn[61]	AU2	Bi-dir	I/O Connector bit 61
148	IO_Conn[62]	BA4	Bi-dir	I/O Connector bit 62
149	IO_Conn[63]	AV6	Bi-dir	I/O Connector bit 63
150	IO_Conn.25V_En	AP3	Output	2.5V Enable signal
151	Mem_0.Addr[00]	BA23	Output	Mem_0 Address bit 0
152	Mem_0.Addr[01]	AV23	Output	Mem_0 Address bit 1
153	Mem_0.Addr[02]	AY23	Output	Mem_0 Address bit 2
154	Mem_0.Addr[03]	AW23	Output	Mem_0 Address bit 3
155	Mem_0.Addr[04]	BB23	Output	Mem_0 Address bit 4
156	Mem_0.Addr[05]	AW22	Output	Mem_0 Address bit 5
157	Mem_0.Addr[06]	BB22	Output	Mem_0 Address bit 6
158	Mem_0.Addr[07]	BA21	Output	Mem_0 Address bit 7
159	Mem_0.Addr[08]	BB8	Output	Mem_0 Address bit 8
160	Mem_0.Addr[09]	AY7	Output	Mem_0 Address bit 9
161	Mem_0.Addr[10]	BA8	Output	Mem_0 Address bit 10
162	Mem_0.Addr[11]	AW7	Output	Mem_0 Address bit 11
163	Mem_0.Addr[12]	BB7	Output	Mem_0 Address bit 12
164	Mem_0.Addr[13]	AV7	Output	Mem_0 Address bit 13
165	Mem_0.Addr[14]	BA7	Output	Mem_0 Address bit 14
166	Mem_0.Addr[15]	AY6	Output	Mem_0 Address bit 15
167	Mem_0.Addr[16]	BB6	Output	Mem_0 Address bit 16
168	Mem_0.Addr[17]	AY5	Output	Mem_0 Address bit 17
169	Mem_0.Addr[18]	BA6	Output	Mem_0 Address bit 18
170	Mem_0.Data[00]	AV20	Bi-dir	Mem_0 Data bit 0
171	Mem_0.Data[01]	AY21	Bi-dir	Mem_0 Data bit 1
172	Mem_0.Data[02]	AW19	Bi-dir	Mem_0 Data bit 2
173	Mem_0.Data[03]	BB21	Bi-dir	Mem_0 Data bit 3
174	Mem_0.Data[04]	BA20	Bi-dir	Mem_0 Data bit 4
175	Mem_0.Data[05]	AV19	Bi-dir	Mem_0 Data bit 5
176	Mem_0.Data[06]	AY20	Bi-dir	Mem_0 Data bit 6
177	Mem_0.Data[07]	BB20	Bi-dir	Mem_0 Data bit 7
178	Mem_0.Data[08]	AW18	Bi-dir	Mem_0 Data bit 8
179	Mem_0.Data[09]	BA19	Bi-dir	Mem_0 Data bit 9
180	Mem_0.Data[10]	AV18	Bi-dir	Mem_0 Data bit 10
181	Mem_0.Data[11]	AY19	Bi-dir	Mem_0 Data bit 11
182	Mem_0.Data[12]	AW17	Bi-dir	Mem_0 Data bit 12
183	Mem_0.Data[13]	BB19	Bi-dir	Mem_0 Data bit 13
184	Mem_0.Data[14]	AV17	Bi-dir	Mem_0 Data bit 14
185	Mem_0.Data[15]	BA18	Bi-dir	Mem_0 Data bit 15

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
186	Mem_0.Data[16]	AW16	Bi-dir	Mem_0 Data bit 16
187	Mem_0.Data[17]	AY18	Bi-dir	Mem_0 Data bit 17
188	Mem_0.Data[18]	AV16	Bi-dir	Mem_0 Data bit 18
189	Mem_0.Data[19]	BB18	Bi-dir	Mem_0 Data bit 19
190	Mem_0.Data[20]	AU16	Bi-dir	Mem_0 Data bit 20
191	Mem_0.Data[21]	BB17	Bi-dir	Mem_0 Data bit 21
192	Mem_0.Data[22]	AW15	Bi-dir	Mem_0 Data bit 22
193	Mem_0.Data[23]	AY17	Bi-dir	Mem_0 Data bit 23
194	Mem_0.Data[24]	BA17	Bi-dir	Mem_0 Data bit 24
195	Mem_0.Data[25]	AV15	Bi-dir	Mem_0 Data bit 25
196	Mem_0.Data[26]	BB16	Bi-dir	Mem_0 Data bit 26
197	Mem_0.Data[27]	AY16	Bi-dir	Mem_0 Data bit 27
198	Mem_0.Data[28]	AW14	Bi-dir	Mem_0 Data bit 28
199	Mem_0.Data[29]	BA16	Bi-dir	Mem_0 Data bit 29
200	Mem_0.Data[30]	AV14	Bi-dir	Mem_0 Data bit 30
201	Mem_0.Data[31]	BB15	Bi-dir	Mem_0 Data bit 31
202	Mem_0.Data[32]	AW13	Bi-dir	Mem_0 Data bit 32
203	Mem_0.Data[33]	AY15	Bi-dir	Mem_0 Data bit 33
204	Mem_0.Data[34]	AW12	Bi-dir	Mem_0 Data bit 34
205	Mem_0.Data[35]	BA15	Bi-dir	Mem_0 Data bit 35
206	Mem_0.Data[36]	AV12	Bi-dir	Mem_0 Data bit 36
207	Mem_0.Data[37]	BB14	Bi-dir	Mem_0 Data bit 37
208	Mem_0.Data[38]	AW11	Bi-dir	Mem_0 Data bit 38
209	Mem_0.Data[39]	BA14	Bi-dir	Mem_0 Data bit 39
210	Mem_0.Data[40]	AV11	Bi-dir	Mem_0 Data bit 40
211	Mem_0.Data[41]	AY14	Bi-dir	Mem_0 Data bit 41
212	Mem_0.Data[42]	AY10	Bi-dir	Mem_0 Data bit 42
213	Mem_0.Data[43]	BB13	Bi-dir	Mem_0 Data bit 43
214	Mem_0.Data[44]	AW10	Bi-dir	Mem_0 Data bit 44
215	Mem_0.Data[45]	BA13	Bi-dir	Mem_0 Data bit 45
216	Mem_0.Data[46]	AY13	Bi-dir	Mem_0 Data bit 46
217	Mem_0.Data[47]	AV10	Bi-dir	Mem_0 Data bit 47
218	Mem_0.Data[48]	BB12	Bi-dir	Mem_0 Data bit 48
219	Mem_0.Data[49]	BA12	Bi-dir	Mem_0 Data bit 49
220	Mem_0.Data[50]	AY9	Bi-dir	Mem_0 Data bit 50
221	Mem_0.Data[51]	AY12	Bi-dir	Mem_0 Data bit 51
222	Mem_0.Data[52]	AW9	Bi-dir	Mem_0 Data bit 52
223	Mem_0.Data[53]	BB11	Bi-dir	Mem_0 Data bit 53
224	Mem_0.Data[54]	BA11	Bi-dir	Mem_0 Data bit 54
225	Mem_0.Data[55]	AV9	Bi-dir	Mem_0 Data bit 55
226	Mem_0.Data[56]	AY11	Bi-dir	Mem_0 Data bit 56
227	Mem_0.Data[57]	AY8	Bi-dir	Mem_0 Data bit 57
228	Mem_0.Data[58]	BB10	Bi-dir	Mem_0 Data bit 58
229	Mem_0.Data[59]	BA10	Bi-dir	Mem_0 Data bit 59
230	Mem_0.Data[60]	AW8	Bi-dir	Mem_0 Data bit 60
231	Mem_0.Data[61]	BB9	Bi-dir	Mem_0 Data bit 61
232	Mem_0.Data[62]	AV8	Bi-dir	Mem_0 Data bit 62

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
233	Mem_0.Data[63]	BA9	Bi-dir	Mem_0 Data bit 63
234	Mem_0.Addr_CS_n	N/A	N/A	Not Used
235	Mem_0.CSH_n	AW6	Output	Mem_0 Chip Select High
236	Mem_0.CSL_n	BA5	Output	Mem_0 Chip Select Low
237	Mem_0.CE_n	BB5	Output	Mem_0 Clock Enable
238	Mem_0.WE_n	AY4	Output	Mem_0 Write Enable
239	Mem_1.Addr[00]	AP41	Output	Mem_1 Address bit 0
240	Mem_1.Addr[01]	AU40	Output	Mem_1 Address bit 1
241	Mem_1.Addr[02]	AR42	Output	Mem_1 Address bit 2
242	Mem_1.Addr[03]	AU39	Output	Mem_1 Address bit 3
243	Mem_1.Addr[04]	AR41	Output	Mem_1 Address bit 4
244	Mem_1.Addr[05]	AV40	Output	Mem_1 Address bit 5
245	Mem_1.Addr[06]	AT42	Output	Mem_1 Address bit 6
246	Mem_1.Addr[07]	AU38	Output	Mem_1 Address bit 7
247	Mem_1.Addr[08]	AV27	Output	Mem_1 Address bit 8
248	Mem_1.Addr[09]	BB26	Output	Mem_1 Address bit 9
249	Mem_1.Addr[10]	AW27	Output	Mem_1 Address bit 10
250	Mem_1.Addr[11]	BA25	Output	Mem_1 Address bit 11
251	Mem_1.Addr[12]	AV26	Output	Mem_1 Address bit 12
252	Mem_1.Addr[13]	AY25	Output	Mem_1 Address bit 13
253	Mem_1.Addr[14]	AW26	Output	Mem_1 Address bit 14
254	Mem_1.Addr[15]	BB25	Output	Mem_1 Address bit 15
255	Mem_1.Addr[16]	AV25	Output	Mem_1 Address bit 16
256	Mem_1.Addr[17]	BA24	Output	Mem_1 Address bit 17
257	Mem_1.Addr[18]	AW25	Output	Mem_1 Address bit 18
258	Mem_1.Data[00]	AT41	Bi-dir	Mem_1 Data bit 0
259	Mem_1.Data[01]	AU42	Bi-dir	Mem_1 Data bit 1
260	Mem_1.Data[02]	AV39	Bi-dir	Mem_1 Data bit 2
261	Mem_1.Data[03]	AU41	Bi-dir	Mem_1 Data bit 3
262	Mem_1.Data[04]	AW40	Bi-dir	Mem_1 Data bit 4
263	Mem_1.Data[05]	AV42	Bi-dir	Mem_1 Data bit 5
264	Mem_1.Data[06]	AV41	Bi-dir	Mem_1 Data bit 6
265	Mem_1.Data[07]	AY39	Bi-dir	Mem_1 Data bit 7
266	Mem_1.Data[08]	BB38	Bi-dir	Mem_1 Data bit 8
267	Mem_1.Data[09]	AY38	Bi-dir	Mem_1 Data bit 9
268	Mem_1.Data[10]	BA37	Bi-dir	Mem_1 Data bit 10
269	Mem_1.Data[11]	BB37	Bi-dir	Mem_1 Data bit 11
270	Mem_1.Data[12]	AW37	Bi-dir	Mem_1 Data bit 12
271	Mem_1.Data[13]	BA36	Bi-dir	Mem_1 Data bit 13
272	Mem_1.Data[14]	BB36	Bi-dir	Mem_1 Data bit 14
273	Mem_1.Data[15]	AY37	Bi-dir	Mem_1 Data bit 15
274	Mem_1.Data[16]	BA35	Bi-dir	Mem_1 Data bit 16
275	Mem_1.Data[17]	AV36	Bi-dir	Mem_1 Data bit 17
276	Mem_1.Data[18]	BB35	Bi-dir	Mem_1 Data bit 18
277	Mem_1.Data[19]	AW36	Bi-dir	Mem_1 Data bit 19
278	Mem_1.Data[20]	BA34	Bi-dir	Mem_1 Data bit 20
279	Mem_1.Data[21]	AY36	Bi-dir	Mem_1 Data bit 21

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
280	Mem_1.Data[22]	BB34	Bi-dir	Mem_1 Data bit 22
281	Mem_1.Data[23]	AV35	Bi-dir	Mem_1 Data bit 23
282	Mem_1.Data[24]	BA33	Bi-dir	Mem_1 Data bit 24
283	Mem_1.Data[25]	AW35	Bi-dir	Mem_1 Data bit 25
284	Mem_1.Data[26]	BB33	Bi-dir	Mem_1 Data bit 26
285	Mem_1.Data[27]	AY35	Bi-dir	Mem_1 Data bit 27
286	Mem_1.Data[28]	BA32	Bi-dir	Mem_1 Data bit 28
287	Mem_1.Data[29]	AV34	Bi-dir	Mem_1 Data bit 29
288	Mem_1.Data[30]	BB32	Bi-dir	Mem_1 Data bit 30
289	Mem_1.Data[31]	AW34	Bi-dir	Mem_1 Data bit 31
290	Mem_1.Data[32]	BA31	Bi-dir	Mem_1 Data bit 32
291	Mem_1.Data[33]	AY34	Bi-dir	Mem_1 Data bit 33
292	Mem_1.Data[34]	AY31	Bi-dir	Mem_1 Data bit 34
293	Mem_1.Data[35]	AV33	Bi-dir	Mem_1 Data bit 35
294	Mem_1.Data[36]	BB31	Bi-dir	Mem_1 Data bit 36
295	Mem_1.Data[37]	AW33	Bi-dir	Mem_1 Data bit 37
296	Mem_1.Data[38]	BA30	Bi-dir	Mem_1 Data bit 38
297	Mem_1.Data[39]	AY33	Bi-dir	Mem_1 Data bit 39
298	Mem_1.Data[40]	AY30	Bi-dir	Mem_1 Data bit 40
299	Mem_1.Data[41]	AV32	Bi-dir	Mem_1 Data bit 41
300	Mem_1.Data[42]	BB30	Bi-dir	Mem_1 Data bit 42
301	Mem_1.Data[43]	AW32	Bi-dir	Mem_1 Data bit 43
302	Mem_1.Data[44]	BA29	Bi-dir	Mem_1 Data bit 44
303	Mem_1.Data[45]	AY32	Bi-dir	Mem_1 Data bit 45
304	Mem_1.Data[46]	AY29	Bi-dir	Mem_1 Data bit 46
305	Mem_1.Data[47]	AV31	Bi-dir	Mem_1 Data bit 47
306	Mem_1.Data[48]	BB29	Bi-dir	Mem_1 Data bit 48
307	Mem_1.Data[49]	AW31	Bi-dir	Mem_1 Data bit 49
308	Mem_1.Data[50]	BA28	Bi-dir	Mem_1 Data bit 50
309	Mem_1.Data[51]	AW30	Bi-dir	Mem_1 Data bit 51
310	Mem_1.Data[52]	AY28	Bi-dir	Mem_1 Data bit 52
311	Mem_1.Data[53]	AV29	Bi-dir	Mem_1 Data bit 53
312	Mem_1.Data[54]	BB28	Bi-dir	Mem_1 Data bit 54
313	Mem_1.Data[55]	AW29	Bi-dir	Mem_1 Data bit 55
314	Mem_1.Data[56]	BA27	Bi-dir	Mem_1 Data bit 56
315	Mem_1.Data[57]	AV28	Bi-dir	Mem_1 Data bit 57
316	Mem_1.Data[58]	AY27	Bi-dir	Mem_1 Data bit 58
317	Mem_1.Data[59]	BB27	Bi-dir	Mem_1 Data bit 59
318	Mem_1.Data[60]	AW28	Bi-dir	Mem_1 Data bit 60
319	Mem_1.Data[61]	BA26	Bi-dir	Mem_1 Data bit 61
320	Mem_1.Data[62]	AU27	Bi-dir	Mem_1 Data bit 62
321	Mem_1.Data[63]	AY26	Bi-dir	Mem_1 Data bit 63
322	Mem_1.Addr_CS_n	N/A	N/A	Not Used
323	Mem_1.CSH_n	BB24	Output	Mem_1 Chip Select High
324	Mem_1.CSL_n	AW24	Output	Mem_1 Chip Select Low
325	Mem_1.CE_n	AV24	Output	Mem_1 Clock Enable
326	Mem_1.WE_n	AY24	Output	Mem_1 Write Enable

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
327	Mem_2.Addr[00]	F38	Output	Mem_2 Address bit 0
328	Mem_2.Addr[01]	D41	Output	Mem_2 Address bit 1
329	Mem_2.Addr[02]	E39	Output	Mem_2 Address bit 2
330	Mem_2.Addr[03]	E41	Output	Mem_2 Address bit 3
331	Mem_2.Addr[04]	E40	Output	Mem_2 Address bit 4
332	Mem_2.Addr[05]	D40	Output	Mem_2 Address bit 5
333	Mem_2.Addr[06]	A38	Output	Mem_2 Address bit 6
334	Mem_2.Addr[07]	D38	Output	Mem_2 Address bit 7
335	Mem_2.Addr[08]	D26	Output	Mem_2 Address bit 8
336	Mem_2.Addr[09]	A24	Output	Mem_2 Address bit 9
337	Mem_2.Addr[10]	E25	Output	Mem_2 Address bit 10
338	Mem_2.Addr[11]	C24	Output	Mem_2 Address bit 11
339	Mem_2.Addr[12]	D25	Output	Mem_2 Address bit 12
340	Mem_2.Addr[13]	B24	Output	Mem_2 Address bit 13
341	Mem_2.Addr[14]	E24	Output	Mem_2 Address bit 14
342	Mem_2.Addr[15]	A23	Output	Mem_2 Address bit 15
343	Mem_2.Addr[16]	D24	Output	Mem_2 Address bit 16
344	Mem_2.Addr[17]	C23	Output	Mem_2 Address bit 17
345	Mem_2.Addr[18]	E23	Output	Mem_2 Address bit 18
346	Mem_2.Data[00]	B37	Bi-dir	Mem_2 Data bit 0
347	Mem_2.Data[01]	E37	Bi-dir	Mem_2 Data bit 1
348	Mem_2.Data[02]	A37	Bi-dir	Mem_2 Data bit 2
349	Mem_2.Data[03]	C39	Bi-dir	Mem_2 Data bit 3
350	Mem_2.Data[04]	B36	Bi-dir	Mem_2 Data bit 4
351	Mem_2.Data[05]	C38	Bi-dir	Mem_2 Data bit 5
352	Mem_2.Data[06]	A36	Bi-dir	Mem_2 Data bit 6
353	Mem_2.Data[07]	B35	Bi-dir	Mem_2 Data bit 7
354	Mem_2.Data[08]	A35	Bi-dir	Mem_2 Data bit 8
355	Mem_2.Data[09]	D37	Bi-dir	Mem_2 Data bit 9
356	Mem_2.Data[10]	B34	Bi-dir	Mem_2 Data bit 10
357	Mem_2.Data[11]	C37	Bi-dir	Mem_2 Data bit 11
358	Mem_2.Data[12]	A34	Bi-dir	Mem_2 Data bit 12
359	Mem_2.Data[13]	E36	Bi-dir	Mem_2 Data bit 13
360	Mem_2.Data[14]	B33	Bi-dir	Mem_2 Data bit 14
361	Mem_2.Data[15]	D36	Bi-dir	Mem_2 Data bit 15
362	Mem_2.Data[16]	A33	Bi-dir	Mem_2 Data bit 16
363	Mem_2.Data[17]	C32	Bi-dir	Mem_2 Data bit 17
364	Mem_2.Data[18]	C36	Bi-dir	Mem_2 Data bit 18
365	Mem_2.Data[19]	B32	Bi-dir	Mem_2 Data bit 19
366	Mem_2.Data[20]	E35	Bi-dir	Mem_2 Data bit 20
367	Mem_2.Data[21]	A32	Bi-dir	Mem_2 Data bit 21
368	Mem_2.Data[22]	D35	Bi-dir	Mem_2 Data bit 22
369	Mem_2.Data[23]	C31	Bi-dir	Mem_2 Data bit 23
370	Mem_2.Data[24]	C35	Bi-dir	Mem_2 Data bit 24
371	Mem_2.Data[25]	B31	Bi-dir	Mem_2 Data bit 25
372	Mem_2.Data[26]	E34	Bi-dir	Mem_2 Data bit 26
373	Mem_2.Data[27]	A31	Bi-dir	Mem_2 Data bit 27

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
374	Mem_2.Data[28]	D34	Bi-dir	Mem_2 Data bit 28
375	Mem_2.Data[29]	C30	Bi-dir	Mem_2 Data bit 29
376	Mem_2.Data[30]	C34	Bi-dir	Mem_2 Data bit 30
377	Mem_2.Data[31]	B30	Bi-dir	Mem_2 Data bit 31
378	Mem_2.Data[32]	E33	Bi-dir	Mem_2 Data bit 32
379	Mem_2.Data[33]	A30	Bi-dir	Mem_2 Data bit 33
380	Mem_2.Data[34]	D33	Bi-dir	Mem_2 Data bit 34
381	Mem_2.Data[35]	C29	Bi-dir	Mem_2 Data bit 35
382	Mem_2.Data[36]	C33	Bi-dir	Mem_2 Data bit 36
383	Mem_2.Data[37]	B29	Bi-dir	Mem_2 Data bit 37
384	Mem_2.Data[38]	E32	Bi-dir	Mem_2 Data bit 38
385	Mem_2.Data[39]	A29	Bi-dir	Mem_2 Data bit 39
386	Mem_2.Data[40]	D32	Bi-dir	Mem_2 Data bit 40
387	Mem_2.Data[41]	C28	Bi-dir	Mem_2 Data bit 41
388	Mem_2.Data[42]	E31	Bi-dir	Mem_2 Data bit 42
389	Mem_2.Data[43]	B28	Bi-dir	Mem_2 Data bit 43
390	Mem_2.Data[44]	D31	Bi-dir	Mem_2 Data bit 44
391	Mem_2.Data[45]	A28	Bi-dir	Mem_2 Data bit 45
392	Mem_2.Data[46]	D30	Bi-dir	Mem_2 Data bit 46
393	Mem_2.Data[47]	C27	Bi-dir	Mem_2 Data bit 47
394	Mem_2.Data[48]	E29	Bi-dir	Mem_2 Data bit 48
395	Mem_2.Data[49]	B27	Bi-dir	Mem_2 Data bit 49
396	Mem_2.Data[50]	D29	Bi-dir	Mem_2 Data bit 50
397	Mem_2.Data[51]	A27	Bi-dir	Mem_2 Data bit 51
398	Mem_2.Data[52]	E28	Bi-dir	Mem_2 Data bit 52
399	Mem_2.Data[53]	C26	Bi-dir	Mem_2 Data bit 53
400	Mem_2.Data[54]	D28	Bi-dir	Mem_2 Data bit 54
401	Mem_2.Data[55]	B26	Bi-dir	Mem_2 Data bit 55
402	Mem_2.Data[56]	F27	Bi-dir	Mem_2 Data bit 56
403	Mem_2.Data[57]	A26	Bi-dir	Mem_2 Data bit 57
404	Mem_2.Data[58]	E27	Bi-dir	Mem_2 Data bit 58
405	Mem_2.Data[59]	C25	Bi-dir	Mem_2 Data bit 59
406	Mem_2.Data[60]	D27	Bi-dir	Mem_2 Data bit 60
407	Mem_2.Data[61]	B25	Bi-dir	Mem_2 Data bit 61
408	Mem_2.Data[62]	E26	Bi-dir	Mem_2 Data bit 62
409	Mem_2.Data[63]	A25	Bi-dir	Mem_2 Data bit 63
410	Mem_2.Addr_CS_n	N/A	N/A	Not Used
411	Mem_2.CSH_n	D23	Output	Mem_2 Chip Select High
412	Mem_2.CSL_n	D22	Output	Mem_2 Chip Select Low
413	Mem_2.CE_n	C21	Output	Mem_2 Clock Enable
414	Mem_2.WE_n	B23	Output	Mem_2 Write Enable
415	Mem_3.Addr[00]	D21	Output	Mem_3 Address bit 0
416	Mem_3.Addr[01]	B21	Output	Mem_3 Address bit 1
417	Mem_3.Addr[02]	D20	Output	Mem_3 Address bit 2
418	Mem_3.Addr[03]	A21	Output	Mem_3 Address bit 3
419	Mem_3.Addr[04]	E20	Output	Mem_3 Address bit 4
420	Mem_3.Addr[05]	C20	Output	Mem_3 Address bit 5

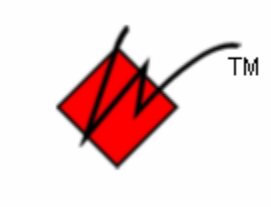
Pad Index	Pad Name	Pin Loc	Dir	Pad Description
421	Mem_3.Addr[06]	D19	Output	Mem_3 Address bit 6
422	Mem_3.Addr[07]	B20	Output	Mem_3 Address bit 7
423	Mem_3.Addr[08]	C5	Output	Mem_3 Address bit 8
424	Mem_3.Addr[09]	A7	Output	Mem_3 Address bit 9
425	Mem_3.Addr[10]	A6	Output	Mem_3 Address bit 10
426	Mem_3.Addr[11]	B7	Output	Mem_3 Address bit 11
427	Mem_3.Addr[12]	D6	Output	Mem_3 Address bit 12
428	Mem_3.Addr[13]	A5	Output	Mem_3 Address bit 13
429	Mem_3.Addr[14]	C4	Output	Mem_3 Address bit 14
430	Mem_3.Addr[15]	F5	Output	Mem_3 Address bit 15
431	Mem_3.Addr[16]	E4	Output	Mem_3 Address bit 16
432	Mem_3.Addr[17]	E2	Output	Mem_3 Address bit 17
433	Mem_3.Addr[18]	D3	Output	Mem_3 Address bit 18
434	Mem_3.Data[00]	E19	Bi-dir	Mem_3 Data bit 0
435	Mem_3.Data[01]	A20	Bi-dir	Mem_3 Data bit 1
436	Mem_3.Data[02]	D18	Bi-dir	Mem_3 Data bit 2
437	Mem_3.Data[03]	A19	Bi-dir	Mem_3 Data bit 3
438	Mem_3.Data[04]	E18	Bi-dir	Mem_3 Data bit 4
439	Mem_3.Data[05]	C19	Bi-dir	Mem_3 Data bit 5
440	Mem_3.Data[06]	D17	Bi-dir	Mem_3 Data bit 6
441	Mem_3.Data[07]	B19	Bi-dir	Mem_3 Data bit 7
442	Mem_3.Data[08]	E17	Bi-dir	Mem_3 Data bit 8
443	Mem_3.Data[09]	A18	Bi-dir	Mem_3 Data bit 9
444	Mem_3.Data[10]	D16	Bi-dir	Mem_3 Data bit 10
445	Mem_3.Data[11]	C18	Bi-dir	Mem_3 Data bit 11
446	Mem_3.Data[12]	E16	Bi-dir	Mem_3 Data bit 12
447	Mem_3.Data[13]	B18	Bi-dir	Mem_3 Data bit 13
448	Mem_3.Data[14]	F16	Bi-dir	Mem_3 Data bit 14
449	Mem_3.Data[15]	A17	Bi-dir	Mem_3 Data bit 15
450	Mem_3.Data[16]	D15	Bi-dir	Mem_3 Data bit 16
451	Mem_3.Data[17]	C17	Bi-dir	Mem_3 Data bit 17
452	Mem_3.Data[18]	E15	Bi-dir	Mem_3 Data bit 18
453	Mem_3.Data[19]	B17	Bi-dir	Mem_3 Data bit 19
454	Mem_3.Data[20]	D14	Bi-dir	Mem_3 Data bit 20
455	Mem_3.Data[21]	A16	Bi-dir	Mem_3 Data bit 21
456	Mem_3.Data[22]	E14	Bi-dir	Mem_3 Data bit 22
457	Mem_3.Data[23]	C16	Bi-dir	Mem_3 Data bit 23
458	Mem_3.Data[24]	D13	Bi-dir	Mem_3 Data bit 24
459	Mem_3.Data[25]	B16	Bi-dir	Mem_3 Data bit 25
460	Mem_3.Data[26]	D12	Bi-dir	Mem_3 Data bit 26
461	Mem_3.Data[27]	A15	Bi-dir	Mem_3 Data bit 27
462	Mem_3.Data[28]	E12	Bi-dir	Mem_3 Data bit 28
463	Mem_3.Data[29]	C15	Bi-dir	Mem_3 Data bit 29
464	Mem_3.Data[30]	C11	Bi-dir	Mem_3 Data bit 30
465	Mem_3.Data[31]	B15	Bi-dir	Mem_3 Data bit 31
466	Mem_3.Data[32]	D11	Bi-dir	Mem_3 Data bit 32
467	Mem_3.Data[33]	A14	Bi-dir	Mem_3 Data bit 33

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
468	Mem_3.Data[34]	E11	Bi-dir	Mem_3 Data bit 34
469	Mem_3.Data[35]	C14	Bi-dir	Mem_3 Data bit 35
470	Mem_3.Data[36]	C10	Bi-dir	Mem_3 Data bit 36
471	Mem_3.Data[37]	B14	Bi-dir	Mem_3 Data bit 37
472	Mem_3.Data[38]	D10	Bi-dir	Mem_3 Data bit 38
473	Mem_3.Data[39]	A13	Bi-dir	Mem_3 Data bit 39
474	Mem_3.Data[40]	E10	Bi-dir	Mem_3 Data bit 40
475	Mem_3.Data[41]	C13	Bi-dir	Mem_3 Data bit 41
476	Mem_3.Data[42]	C9	Bi-dir	Mem_3 Data bit 42
477	Mem_3.Data[43]	B13	Bi-dir	Mem_3 Data bit 43
478	Mem_3.Data[44]	D9	Bi-dir	Mem_3 Data bit 44
479	Mem_3.Data[45]	A12	Bi-dir	Mem_3 Data bit 45
480	Mem_3.Data[46]	E9	Bi-dir	Mem_3 Data bit 46
481	Mem_3.Data[47]	C12	Bi-dir	Mem_3 Data bit 47
482	Mem_3.Data[48]	C8	Bi-dir	Mem_3 Data bit 48
483	Mem_3.Data[49]	B12	Bi-dir	Mem_3 Data bit 49
484	Mem_3.Data[50]	D8	Bi-dir	Mem_3 Data bit 50
485	Mem_3.Data[51]	A11	Bi-dir	Mem_3 Data bit 51
486	Mem_3.Data[52]	E8	Bi-dir	Mem_3 Data bit 52
487	Mem_3.Data[53]	B11	Bi-dir	Mem_3 Data bit 53
488	Mem_3.Data[54]	C7	Bi-dir	Mem_3 Data bit 54
489	Mem_3.Data[55]	A10	Bi-dir	Mem_3 Data bit 55
490	Mem_3.Data[56]	C6	Bi-dir	Mem_3 Data bit 56
491	Mem_3.Data[57]	B10	Bi-dir	Mem_3 Data bit 57
492	Mem_3.Data[58]	D7	Bi-dir	Mem_3 Data bit 58
493	Mem_3.Data[59]	A9	Bi-dir	Mem_3 Data bit 59
494	Mem_3.Data[60]	B9	Bi-dir	Mem_3 Data bit 60
495	Mem_3.Data[61]	A8	Bi-dir	Mem_3 Data bit 61
496	Mem_3.Data[62]	E7	Bi-dir	Mem_3 Data bit 62
497	Mem_3.Data[63]	B8	Bi-dir	Mem_3 Data bit 63
498	Mem_3.Addr_CS_n	N/A	N/A	Not Used
499	Mem_3.CSH_n	F4	Output	Mem_3 Chip Select High
500	Mem_3.CSL_n	G2	Output	Mem_3 Chip Select Low
501	Mem_3.CE_n	E1	Output	Mem_3 Clock Enable
502	Mem_3.WE_n	F2	Output	Mem_3 Write Enable
503	Mem_4.Addr[00]	AF39	Output	Mem_4 Address bit 0
504	Mem_4.Addr[01]	AD40	Output	Mem_4 Address bit 1
505	Mem_4.Addr[02]	AF38	Output	Mem_4 Address bit 2
506	Mem_4.Addr[03]	AD42	Output	Mem_4 Address bit 3
507	Mem_4.Addr[04]	AG39	Output	Mem_4 Address bit 4
508	Mem_4.Addr[05]	AE41	Output	Mem_4 Address bit 5
509	Mem_4.Addr[06]	AG38	Output	Mem_4 Address bit 6
510	Mem_4.Addr[07]	AE40	Output	Mem_4 Address bit 7
511	Mem_4.Addr[08]	AP38	Output	Mem_4 Address bit 8
512	Mem_4.Addr[09]	AL40	Output	Mem_4 Address bit 9
513	Mem_4.Addr[10]	AR40	Output	Mem_4 Address bit 10
514	Mem_4.Addr[11]	AM42	Output	Mem_4 Address bit 11

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
515	Mem_4.Addr[12]	AR39	Output	Mem_4 Address bit 12
516	Mem_4.Addr[13]	AM41	Output	Mem_4 Address bit 13
517	Mem_4.Addr[14]	AR38	Output	Mem_4 Address bit 14
518	Mem_4.Addr[15]	AM40	Output	Mem_4 Address bit 15
519	Mem_4.Addr[16]	AT40	Output	Mem_4 Address bit 16
520	Mem_4.Addr[17]	AN42	Output	Mem_4 Address bit 17
521	Mem_4.Addr[18]	AT39	Output	Mem_4 Address bit 18
522	Mem_4.Data[00]	AH39	Bi-dir	Mem_4 Data bit 0
523	Mem_4.Data[01]	AE42	Bi-dir	Mem_4 Data bit 1
524	Mem_4.Data[02]	AH38	Bi-dir	Mem_4 Data bit 2
525	Mem_4.Data[03]	AF41	Bi-dir	Mem_4 Data bit 3
526	Mem_4.Data[04]	AF40	Bi-dir	Mem_4 Data bit 4
527	Mem_4.Data[05]	AJ39	Bi-dir	Mem_4 Data bit 5
528	Mem_4.Data[06]	AF42	Bi-dir	Mem_4 Data bit 6
529	Mem_4.Data[07]	AG42	Bi-dir	Mem_4 Data bit 7
530	Mem_4.Data[08]	AJ38	Bi-dir	Mem_4 Data bit 8
531	Mem_4.Data[09]	AG40	Bi-dir	Mem_4 Data bit 9
532	Mem_4.Data[10]	AK39	Bi-dir	Mem_4 Data bit 10
533	Mem_4.Data[11]	AG41	Bi-dir	Mem_4 Data bit 11
534	Mem_4.Data[12]	AL39	Bi-dir	Mem_4 Data bit 12
535	Mem_4.Data[13]	AH42	Bi-dir	Mem_4 Data bit 13
536	Mem_4.Data[14]	AL38	Bi-dir	Mem_4 Data bit 14
537	Mem_4.Data[15]	AH41	Bi-dir	Mem_4 Data bit 15
538	Mem_4.Data[16]	AH40	Bi-dir	Mem_4 Data bit 16
539	Mem_4.Data[17]	AM39	Bi-dir	Mem_4 Data bit 17
540	Mem_4.Data[18]	AJ42	Bi-dir	Mem_4 Data bit 18
541	Mem_4.Data[19]	AJ41	Bi-dir	Mem_4 Data bit 19
542	Mem_4.Data[20]	AM38	Bi-dir	Mem_4 Data bit 20
543	Mem_4.Data[21]	AJ40	Bi-dir	Mem_4 Data bit 21
544	Mem_4.Data[22]	AN40	Bi-dir	Mem_4 Data bit 22
545	Mem_4.Data[23]	AK42	Bi-dir	Mem_4 Data bit 23
546	Mem_4.Data[24]	AN39	Bi-dir	Mem_4 Data bit 24
547	Mem_4.Data[25]	AK41	Bi-dir	Mem_4 Data bit 25
548	Mem_4.Data[26]	AN38	Bi-dir	Mem_4 Data bit 26
549	Mem_4.Data[27]	AK40	Bi-dir	Mem_4 Data bit 27
550	Mem_4.Data[28]	AP40	Bi-dir	Mem_4 Data bit 28
551	Mem_4.Data[29]	AL42	Bi-dir	Mem_4 Data bit 29
552	Mem_4.Data[30]	AP39	Bi-dir	Mem_4 Data bit 30
553	Mem_4.Data[31]	AL41	Bi-dir	Mem_4 Data bit 31
554	Mem_4.Addr_CS_n	N/A	N/A	Not Used
555	Mem_4.CS_n	AP42	Output	Mem_4 Chip Select
556	Mem_4.CE_n	AN41	Output	Mem_4 Clock Enable
557	Mem_4.WE_n	AT38	Output	Mem_4 Write Enable
558	PLD.Dir	AH5	Output	PLD direction bit
559	PLD.Data	AF5	Bi-Dir	PLD Data
560	PLD.CS_n	AE3	Output	PLD chip select
561	WSDP_A.Data[0]	P38	Bi-dir	WSDP™ Connector Data Bit

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
562	WSDP_A.Data[1]	P42	Bi-dir	WSDP™ Connector A Data Bit
563	WSDP_A.Data[2]	N39	Bi-dir	WSDP™ Connector A Data Bit
564	WSDP_A.Data[3]	P40	Bi-dir	WSDP™ Connector A Data Bit
565	WSDP_A.Data[4]	M39	Bi-dir	WSDP™ Connector A Data Bit
566	WSDP_A.Data[5]	P41	Bi-dir	WSDP™ Connector A Data Bit
567	WSDP_A.Data[6]	M38	Bi-dir	WSDP™ Connector A Data Bit
568	WSDP_A.Data[7]	N42	Bi-dir	WSDP™ Connector A Data Bit
569	WSDP_A.Data[8]	N40	Bi-dir	WSDP™ Connector A Data Bit
570	WSDP_A.Data[9]	L39	Bi-dir	WSDP™ Connector A Data Bit
571	WSDP_A.Data[10]	M40	Bi-dir	WSDP™ Connector A Data Bit
572	WSDP_A.Data[11]	N41	Bi-dir	WSDP™ Connector A Data Bit
573	WSDP_A.Data[12]	K39	Bi-dir	WSDP™ Connector A Data Bit
574	WSDP_A.Data[13]	M42	Bi-dir	WSDP™ Connector A Data Bit
575	WSDP_A.Data[14]	K40	Bi-dir	WSDP™ Connector A Data Bit
576	WSDP_A.Data[15]	L38	Bi-dir	WSDP™ Connector A Data Bit
577	WSDP_A.Data[16]	K38	Bi-dir	WSDP™ Connector A Data Bit
578	WSDP_A.Data[17]	M41	Bi-dir	WSDP™ Connector A Data Bit
579	WSDP_A.Data[18]	J40	Bi-dir	WSDP™ Connector A Data Bit
580	WSDP_A.Data[19]	L42	Bi-dir	WSDP™ Connector A Data Bit
581	WSDP_A.Data[20]	H40	Bi-dir	WSDP™ Connector A Data Bit
582	WSDP_A.Data[21]	L40	Bi-dir	WSDP™ Connector A Data Bit
583	WSDP_A.Data[22]	J38	Bi-dir	WSDP™ Connector A Data Bit
584	WSDP_A.Data[23]	L41	Bi-dir	WSDP™ Connector A Data Bit
585	WSDP_A.Data[24]	J39	Bi-dir	WSDP™ Connector A Data Bit
586	WSDP_A.Data[25]	K42	Bi-dir	WSDP™ Connector A Data Bit
587	WSDP_A.Data[26]	H39	Bi-dir	WSDP™ Connector A Data Bit
588	WSDP_A.Data[27]	J41	Bi-dir	WSDP™ Connector A Data Bit
589	WSDP_A.Data[28]	H38	Bi-dir	WSDP™ Connector A Data Bit
590	WSDP_A.Data[29]	K41	Bi-dir	WSDP™ Connector A Data Bit
591	WSDP_A.Data[30]	J42	Bi-dir	WSDP™ Connector A Data Bit
592	WSDP_A.Data[31]	G40	Bi-dir	WSDP™ Connector A Data Bit
593	WSDP_A.Ctrl[0]	G39	Bi-dir	WSDP™ Connector A Control Bit
594	WSDP_A.Ctrl[1]	H41	Bi-dir	WSDP™ Connector A Control Bit
595	WSDP_A.Ctrl[2]	G38	Bi-dir	WSDP™ Connector A Control Bit
596	WSDP_A.Ctrl[3]	G42	Bi-dir	WSDP™ Connector A Control Bit
597	WSDP_A.Ctrl[4]	G41	Bi-dir	WSDP™ Connector A Control Bit
598	WSDP_A.Ctrl[5]	F40	Bi-dir	WSDP™ Connector A Control Bit
599	WSDP_A.Ctrl[6]	F42	Bi-dir	WSDP™ Connector A Control Bit
600	WSDP_A.Ctrl[7]	F41	Bi-dir	WSDP™ Connector A Control Bit
601	WSDP_A.Ctrl[8]	F39	Bi-dir	WSDP™ Connector A Control Bit
602	WSDP_A.Ctrl[9]	E42	Bi-dir	WSDP™ Connector A Control Bit
603	WSDP_A.HS	H42	Bi-dir	WSDP™ Connector A Handshake Bit
604	WSDP_B.Data[0]	Y41	Bi-dir	WSDP™ Connector B Data Bit
605	WSDP_B.Data[1]	AA39	Bi-dir	WSDP™ Connector B Data Bit
606	WSDP_B.Data[2]	Y40	Bi-dir	WSDP™ Connector B Data Bit
607	WSDP_B.Data[3]	Y39	Bi-dir	WSDP™ Connector B Data Bit
608	WSDP_B.Data[4]	Y42	Bi-dir	WSDP™ Connector B Data Bit

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
609	WSDP_B.Data[5]	Y38	Bi-dir	WSDP™ Connector B Data Bit
610	WSDP_B.Data[6]	W41	Bi-dir	WSDP™ Connector B Data Bit
611	WSDP_B.Data[7]	W40	Bi-dir	WSDP™ Connector B Data Bit
612	WSDP_B.Data[8]	W39	Bi-dir	WSDP™ Connector B Data Bit
613	WSDP_B.Data[9]	W42	Bi-dir	WSDP™ Connector B Data Bit
614	WSDP_B.Data[10]	W38	Bi-dir	WSDP™ Connector B Data Bit
615	WSDP_B.Data[11]	V40	Bi-dir	WSDP™ Connector B Data Bit
616	WSDP_B.Data[12]	V39	Bi-dir	WSDP™ Connector B Data Bit
617	WSDP_B.Data[13]	V38	Bi-dir	WSDP™ Connector B Data Bit
618	WSDP_B.Data[14]	V41	Bi-dir	WSDP™ Connector B Data Bit
619	WSDP_B.Data[15]	V42	Bi-dir	WSDP™ Connector B Data Bit
620	WSDP_B.Data[16]	U41	Bi-dir	WSDP™ Connector B Data Bit
621	WSDP_B.Data[17]	U39	Bi-dir	WSDP™ Connector B Data Bit
622	WSDP_B.Data[18]	U38	Bi-dir	WSDP™ Connector B Data Bit
623	WSDP_B.Data[19]	U40	Bi-dir	WSDP™ Connector B Data Bit
624	WSDP_B.Data[20]	U42	Bi-dir	WSDP™ Connector B Data Bit
625	WSDP_B.Data[21]	T39	Bi-dir	WSDP™ Connector B Data Bit
626	WSDP_B.Data[22]	T40	Bi-dir	WSDP™ Connector B Data Bit
627	WSDP_B.Data[23]	T38	Bi-dir	WSDP™ Connector B Data Bit
628	WSDP_B.Data[24]	T41	Bi-dir	WSDP™ Connector B Data Bit
629	WSDP_B.Data[25]	R39	Bi-dir	WSDP™ Connector B Data Bit
630	WSDP_B.Data[26]	T42	Bi-dir	WSDP™ Connector B Data Bit
631	WSDP_B.Data[27]	R42	Bi-dir	WSDP™ Connector B Data Bit
632	WSDP_B.Data[28]	R38	Bi-dir	WSDP™ Connector B Data Bit
633	WSDP_B.Data[29]	R40	Bi-dir	WSDP™ Connector B Data Bit
634	WSDP_B.Data[30]	P39	Bi-dir	WSDP™ Connector B Data Bit
635	WSDP_B.Data[31]	R41	Bi-dir	WSDP™ Connector B Data Bit
636	WSDP_B.Ctrl[0]	AB39	Bi-dir	WSDP™ Connector B Control Bit
637	WSDP_B.Ctrl[1]	AA41	Bi-dir	WSDP™ Connector B Control Bit
638	WSDP_B.Ctrl[2]	AA40	Bi-dir	WSDP™ Connector B Control Bit
639	WSDP_B.Ctrl[3]	AB41	Bi-dir	WSDP™ Connector B Control Bit
640	WSDP_B.Ctrl[4]	AC39	Bi-dir	WSDP™ Connector B Control Bit
641	WSDP_B.Ctrl[5]	AB40	Bi-dir	WSDP™ Connector B Control Bit
642	WSDP_B.Ctrl[6]	AC38	Bi-dir	WSDP™ Connector B Control Bit
643	WSDP_B.Ctrl[7]	AB42	Bi-dir	WSDP™ Connector B Control Bit
644	WSDP_B.Ctrl[8]	AD39	Bi-dir	WSDP™ Connector B Control Bit
645	WSDP_B.Ctrl[9]	AC41	Bi-dir	WSDP™ Connector B Control Bit
646	WSDP_B.HS	AA42	Bi-dir	WSDP™ Connector B Handshake Bit


TM

ATTACHMENT B: BPE PAD LOCATIONS AND DEFINITIONS

BPE Pad Definitions and Locations

The definitions and locations of the FIREBIRD™/PCI PE0 pads are contained in Table B-1. This information is also contained in the PE0 user constraints file (pe0.ucf) which is used by the Xilinx® Alliance Series to place the PE0 pads in the proper locations.



INFORMATION NOTE

The pad index corresponds to the logical pad index of the PE0 VHDL Model, not the physical I/O pin number of the FPGA die.

Table B-1: FIREBIRD™/PMC WSDP BPE Pad Locations and Definitions

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
0	Clocks.M_Clk	A11	Input	Memory clock
1	Clocks.U_Clk	W12	Input	PCI/LAD bus clock
2	Clocks.W_CLK_A	C11	Input	User clock
3	Clocks.W_CLK_B	Y11	Input	RACEway input clock
4	Clocks.WSDPA_Clk_Out	H21	Output	BPE WSDPA Input Clock
5	Clocks.WSDPB_Clk_Out	R22	Output	BPE WSDPB Input Clock
6	LEDs(0)	J19	Output	Memory clock sync signal
7	LEDs(1)	M19	Output	User clock sync signal
8	IO_Conn_A.Data(0)	D16	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 0
9	IO_Conn_A.Data(1)	C17	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 1
10	IO_Conn_A.Data(2)	D15	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 2
11	IO_Conn_A.Data(3)	A17	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 3
12	IO_Conn_A.Data(4)	D14	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 4
13	IO_Conn_A.Data(5)	C16	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 5
14	IO_Conn_A.Data(6)	B15	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 6
15	IO_Conn_A.Data(7)	C15	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 7
16	IO_Conn_A.Data(8)	F12	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 8
17	IO_Conn_A.Data(9)	A15	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 9
18	IO_Conn_A.Data(10)	B14	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 10
19	IO_Conn_A.Data(11)	C14	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 11
20	IO_Conn_A.Data(12)	C13	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 12
21	IO_Conn_A.Data(13)	D13	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 13
22	IO_Conn_A.Data(14)	B12	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 14

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
23	IO_Conn_A.Data(15)	A13	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 15
24	IO_Conn_A.Data(16)	C12	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 16
25	IO_Conn_A.Data(17)	D12	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 17
26	IO_Conn_A.Data(18)	D9	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 18
27	IO_Conn_A.Data(19)	A8	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 19
28	IO_Conn_A.Data(20)	C8	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 20
29	IO_Conn_A.Data(21)	B8	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 21
30	IO_Conn_A.Data(22)	A7	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 22
31	IO_Conn_A.Data(23)	E9	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 23
32	IO_Conn_A.Data(24)	D7	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 24
33	IO_Conn_A.Data(25)	C7	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 25
34	IO_Conn_A.Data(26)	A5	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 26
35	IO_Conn_A.Data(27)	B6	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 27
36	IO_Conn_A.Data(28)	A4	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 28
37	IO_Conn_A.Data(29)	E7	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 29
38	IO_Conn_A.Data(30)	D5	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 30
39	IO_Conn_A.Data(31)	B3	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 31
40	IO_Conn_A.Data(32)	B11	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 32
41	IO_Conn_A.Data(33)	D11	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 33
42	IO_Conn_A.Data(34)	F11	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 34
43	IO_Conn_A.Data(35)	B10	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 35
44	IO_Conn_A.Data(36)	C9	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 36
45	IO_Conn_A.Data(37)	E10	Bi-dir	WSDP™ External I/O Connector A (WSDP™ 1) Bit 37
46	IO_Conn_A.Byte_Term(0)	G21	Output	WSDP™ I/O Connector A Data Byte Termination Bit 0
47	IO_Conn_A.Byte_Term(1)	G18	Output	WSDP™ I/O Connector A Data Byte Termination Bit 1
48	IO_Conn_A.Byte_Term(2)	E13	Output	WSDP™ I/O Connector A Data Byte Termination Bit 2
49	IO_Conn_A.Byte_Term(3)	A14	Output	WSDP™ I/O Connector A Data Byte Termination Bit 3
50	IO_Conn_A.Diff_Term(0)	G20	Output	WSDP™ I/O Connector A Differential Termination Bit 0
51	IO_Conn_A.Diff_Term(1)	G19	Output	WSDP™ I/O Connector A Differential Termination Bit 1
52	IO_Conn_A.Diff_Term(2)	F22	Output	WSDP™ I/O Connector A Differential Termination Bit 2
53	IO_Conn_A.Diff_Term(3)	F19	Output	WSDP™ I/O Connector A Differential Termination Bit 3
54	IO_Conn_A.Diff_Term(4)	F20	Output	WSDP™ I/O Connector A Differential Termination Bit 4
55	IO_Conn_A.Diff_Term(5)	F18	Output	WSDP™ I/O Connector A Differential Termination Bit 5
56	IO_Conn_A.Diff_Term(6)	E22	Output	WSDP™ I/O Connector A Differential Termination Bit 6
57	IO_Conn_A.Diff_Term(7)	E21	Output	WSDP™ I/O Connector A Differential Termination Bit 7
58	IO_Conn_A.Diff_Term(8)	E20	Output	WSDP™ I/O Connector A Differential Termination Bit 8
59	IO_Conn_A.Diff_Term(9)	D21	Output	WSDP™ I/O Connector A Differential Termination Bit 9
60	IO_Conn_A.Diff_Term(10)	C22	Output	WSDP™ I/O Connector A Differential Termination Bit 10
61	IO_Conn_A.Diff_Term(11)	C21	Output	WSDP™ I/O Connector A Differential Termination Bit 11
62	IO_Conn_A.Diff_Term(12)	B19	Output	WSDP™ I/O Connector A Differential Termination Bit 12
63	IO_Conn_A.Diff_Term(13)	C18	Output	WSDP™ I/O Connector A Differential Termination Bit 13
64	IO_Conn_A.Diff_Term(14)	E12	Output	WSDP™ I/O Connector A Differential Termination Bit 14
65	IO_Conn_A.Diff_Term(15)	A10	Output	WSDP™ I/O Connector A Differential Termination Bit 15
66	IO_Conn_A.Diff_Term(16)	C10	Output	WSDP™ I/O Connector A Differential Termination Bit 16
67	IO_Conn_A.Diff_Term(17)	B9	Output	WSDP™ I/O Connector A Differential Termination Bit 17
68	IO_Conn_A.Diff_Term(18)	D10	Output	WSDP™ I/O Connector A Differential Termination Bit 18

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
69	IO_Conn_A.Ctrl_Term(0)	E16	Output	WSDP™ I/O Connector A Ctrl Byte Termination Bit 0
70	IO_Conn_A.Ctrl_Term(1)	B17	Output	WSDP™ I/O Connector A Ctrl Byte Termination Bit 1
71	IO_Conn_A.Ctrl_Term(2)	E15	Output	WSDP™ I/O Connector A Ctrl Byte Termination Bit 2
72	IO_Conn_A.Ctrl_Term(3)	A16	Output	WSDP™ I/O Connector A Ctrl Byte Termination Bit 3
73	IO_Conn_A.Clk_In_Term	D17	Output	WSDP™ I/O Connector A Clock In Termination Bit
74	IO_Conn_A.Clk_Out_Term	B18	Output	WSDP™ I/O Connector A Clock Out Termination Bit
75	IO_Conn_A.VCCO25_Sel	H19	Output	WSDP™ I/O Connector A 2.5V Select Bit
76	IO_Conn_B.Data(0)	AB3	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 0
77	IO_Conn_B.Data(1)	V7	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 1
78	IO_Conn_B.Data(2)	W6	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 2
79	IO_Conn_B.Data(3)	Y7	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 3
80	IO_Conn_B.Data(4)	AA6	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 4
81	IO_Conn_B.Data(5)	AB6	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 5
82	IO_Conn_B.Data(6)	AA7	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 6
83	IO_Conn_B.Data(7)	W7	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 7
84	IO_Conn_B.Data(8)	AA8	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 8
85	IO_Conn_B.Data(9)	V9	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 9
86	IO_Conn_B.Data(10)	W9	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 10
87	IO_Conn_B.Data(11)	AB9	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 11
88	IO_Conn_B.Data(12)	Y9	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 12
89	IO_Conn_B.Data(13)	V10	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 13
90	IO_Conn_B.Data(14)	V12	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 14
91	IO_Conn_B.Data(15)	Y12	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 15
92	IO_Conn_B.Data(16)	AA12	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 16
93	IO_Conn_B.Data(17)	W13	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 17
94	IO_Conn_B.Data(18)	V13	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 18
95	IO_Conn_B.Data(19)	AB14	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 19
96	IO_Conn_B.Data(20)	AA14	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 20
97	IO_Conn_B.Data(21)	V14	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 21
98	IO_Conn_B.Data(22)	Y14	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 22
99	IO_Conn_B.Data(23)	W15	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 23
100	IO_Conn_B.Data(24)	AB15	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 24
101	IO_Conn_B.Data(25)	AA15	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 25
102	IO_Conn_B.Data(26)	V15	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 26
103	IO_Conn_B.Data(27)	Y16	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 27
104	IO_Conn_B.Data(28)	AA17	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 28
105	IO_Conn_B.Data(29)	W16	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 29
106	IO_Conn_B.Data(30)	V16	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 30
107	IO_Conn_B.Data(31)	AA18	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 31
108	IO_Conn_B.Data(32)	AA11	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 32
109	IO_Conn_B.Data(33)	U12	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 33
110	IO_Conn_B.Data(34)	W10	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 34
111	IO_Conn_B.Data(35)	AB10	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 35
112	IO_Conn_B.Data(36)	W11	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 36
113	IO_Conn_B.Data(37)	AB11	Bi-dir	WSDP™ External I/O Connector B (WSDP™ 4) Bit 37
114	IO_Conn_B.Byte_Term(0)	W14	Output	WSDP™ I/O Connector B Data Byte Termination Bit 0
115	IO_Conn_B.Byte_Term(1)	Y15	Output	WSDP™ I/O Connector B Data Byte Termination Bit 1

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
116	IO_Conn_B.Byte_Term(2)	R18	Output	WSDP™ I/O Connector B Data Byte Termination Bit 2
117	IO_Conn_B.Byte_Term(3)	T21	Output	WSDP™ I/O Connector B Data Byte Termination Bit 3
118	IO_Conn_B.Diff_Term(0)	AA9	Output	WSDP™ I/O Connector B Differential Termination Bit 0
119	IO_Conn_B.Diff_Term(1)	V11	Output	WSDP™ I/O Connector B Differential Termination Bit 1
120	IO_Conn_B.Diff_Term(2)	AA10	Output	WSDP™ I/O Connector B Differential Termination Bit 2
121	IO_Conn_B.Diff_Term(3)	AB13	Output	WSDP™ I/O Connector B Differential Termination Bit 3
122	IO_Conn_B.Diff_Term(4)	Y13	Output	WSDP™ I/O Connector B Differential Termination Bit 4
123	IO_Conn_B.Diff_Term(5)	W17	Output	WSDP™ I/O Connector B Differential Termination Bit 5
124	IO_Conn_B.Diff_Term(6)	AA19	Output	WSDP™ I/O Connector B Differential Termination Bit 6
125	IO_Conn_B.Diff_Term(7)	AA20	Output	WSDP™ I/O Connector B Differential Termination Bit 7
126	IO_Conn_B.Diff_Term(8)	W18	Output	WSDP™ I/O Connector B Differential Termination Bit 8
127	IO_Conn_B.Diff_Term(9)	V20	Output	WSDP™ I/O Connector B Differential Termination Bit 9
128	IO_Conn_B.Diff_Term(10)	AA22	Output	WSDP™ I/O Connector B Differential Termination Bit 10
129	IO_Conn_B.Diff_Term(11)	W21	Output	WSDP™ I/O Connector B Differential Termination Bit 11
130	IO_Conn_B.Diff_Term(12)	U19	Output	WSDP™ I/O Connector B Differential Termination Bit 12
131	IO_Conn_B.Diff_Term(13)	V21	Output	WSDP™ I/O Connector B Differential Termination Bit 13
132	IO_Conn_B.Diff_Term(14)	T18	Output	WSDP™ I/O Connector B Differential Termination Bit 14
133	IO_Conn_B.Diff_Term(15)	W22	Output	WSDP™ I/O Connector B Differential Termination Bit 15
134	IO_Conn_B.Diff_Term(16)	T20	Output	WSDP™ I/O Connector B Differential Termination Bit 16
135	IO_Conn_B.Diff_Term(17)	T19	Output	WSDP™ I/O Connector B Differential Termination Bit 17
136	IO_Conn_B.Diff_Term(18)	V22	Output	WSDP™ I/O Connector B Differential Termination Bit 18
137	IO_Conn_B.Ctrl_Term(0)	AB17	Output	WSDP™ I/O Connector B Ctrl Byte Termination Bit 0
138	IO_Conn_B.Ctrl_Term(1)	AB18	Output	WSDP™ I/O Connector B Ctrl Byte Termination Bit 1
139	IO_Conn_B.Ctrl_Term(2)	Y17	Output	WSDP™ I/O Connector B Ctrl Byte Termination Bit 2
140	IO_Conn_B.Ctrl_Term(3)	P18	Output	WSDP™ I/O Connector B Ctrl Byte Termination Bit 3
141	IO_Conn_B.Clk_In_Term	Y18	Output	WSDP™ I/O Connector B Clock In Termination Bit
142	IO_Conn_B.Clk_Out_Term	V17	Output	WSDP™ I/O Connector B Clock Out Termination Bit
143	IO_Conn_B.VCCO25_Sel	U22	Output	WSDP™ I/O Connector B 2.5V Select Bit
144	WSDP_A.Data(0)	L4	Bi-dir	WSDP™ Connector A Data Bit 0
145	WSDP_A.Data(1)	L3	Bi-dir	WSDP™ Connector A Data Bit 1
146	WSDP_A.Data(2)	L5	Bi-dir	WSDP™ Connector A Data Bit 2
147	WSDP_A.Data(3)	L1	Bi-dir	WSDP™ Connector A Data Bit 3
148	WSDP_A.Data(4)	L6	Bi-dir	WSDP™ Connector A Data Bit 4
149	WSDP_A.Data(5)	K2	Bi-dir	WSDP™ Connector A Data Bit 5
150	WSDP_A.Data(6)	K4	Bi-dir	WSDP™ Connector A Data Bit 6
151	WSDP_A.Data(7)	K3	Bi-dir	WSDP™ Connector A Data Bit 7
152	WSDP_A.Data(8)	K1	Bi-dir	WSDP™ Connector A Data Bit 8
153	WSDP_A.Data(9)	K5	Bi-dir	WSDP™ Connector A Data Bit 9
154	WSDP_A.Data(10)	J1	Bi-dir	WSDP™ Connector A Data Bit 10
155	WSDP_A.Data(11)	J3	Bi-dir	WSDP™ Connector A Data Bit 11
156	WSDP_A.Data(12)	J5	Bi-dir	WSDP™ Connector A Data Bit 12
157	WSDP_A.Data(13)	J2	Bi-dir	WSDP™ Connector A Data Bit 13
158	WSDP_A.Data(14)	H1	Bi-dir	WSDP™ Connector A Data Bit 14
159	WSDP_A.Data(15)	J4	Bi-dir	WSDP™ Connector A Data Bit 15
160	WSDP_A.Data(16)	H4	Bi-dir	WSDP™ Connector A Data Bit 16
161	WSDP_A.Data(17)	H3	Bi-dir	WSDP™ Connector A Data Bit 17

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
162	WSDP_A.Data(18)	G1	Bi-dir	WSDP™ Connector A Data Bit 18
163	WSDP_A.Data(19)	H2	Bi-dir	WSDP™ Connector A Data Bit 19
164	WSDP_A.Data(20)	F1	Bi-dir	WSDP™ Connector A Data Bit 20
165	WSDP_A.Data(21)	E2	Bi-dir	WSDP™ Connector A Data Bit 21
166	WSDP_A.Data(22)	H5	Bi-dir	WSDP™ Connector A Data Bit 22
167	WSDP_A.Data(23)	G3	Bi-dir	WSDP™ Connector A Data Bit 23
168	WSDP_A.Data(24)	G4	Bi-dir	WSDP™ Connector A Data Bit 24
169	WSDP_A.Data(25)	F2	Bi-dir	WSDP™ Connector A Data Bit 25
170	WSDP_A.Data(26)	F4	Bi-dir	WSDP™ Connector A Data Bit 26
171	WSDP_A.Data(27)	D2	Bi-dir	WSDP™ Connector A Data Bit 27
172	WSDP_A.Data(28)	G5	Bi-dir	WSDP™ Connector A Data Bit 28
173	WSDP_A.Data(29)	F3	Bi-dir	WSDP™ Connector A Data Bit 29
174	WSDP_A.Data(30)	E1	Bi-dir	WSDP™ Connector A Data Bit 30
175	WSDP_A.Data(31)	E3	Bi-dir	WSDP™ Connector A Data Bit 31
176	WSDP_A.Ctrl(0)	F5	Bi-dir	WSDP™ Connector A Control Bit 0
177	WSDP_A.Ctrl(1)	E4	Bi-dir	WSDP™ Connector A Control Bit 1
178	WSDP_A.Ctrl(2)	B1	Bi-dir	WSDP™ Connector A Control Bit 2
179	WSDP_A.Ctrl(3)	C1	Bi-dir	WSDP™ Connector A Control Bit 3
180	WSDP_A.Ctrl(4)	E6	Bi-dir	WSDP™ Connector A Control Bit 4
181	WSDP_A.Ctrl(5)	D8	Bi-dir	WSDP™ Connector A Control Bit 5
182	WSDP_A.Ctrl(6)	C5	Bi-dir	WSDP™ Connector A Control Bit 6
183	WSDP_A.Ctrl(7)	B5	Bi-dir	WSDP™ Connector A Control Bit 7
184	WSDP_A.Ctrl(8)	D6	Bi-dir	WSDP™ Connector A Control Bit 8
185	WSDP_A.Ctrl(9)	A3	Bi-dir	WSDP™ Connector A Control Bit 9
186	WSDP_A.HS	B7	Bi-dir	WSDP™ Connector A Handshake Bit
187	WSDP_B.Data(0)	W2	Bi-dir	WSDP™ Connector B Data Bit 0
188	WSDP_B.Data(1)	Y1	Bi-dir	WSDP™ Connector B Data Bit 1
189	WSDP_B.Data(2)	W1	Bi-dir	WSDP™ Connector B Data Bit 2
190	WSDP_B.Data(3)	T3	Bi-dir	WSDP™ Connector B Data Bit 3
191	WSDP_B.Data(4)	U4	Bi-dir	WSDP™ Connector B Data Bit 4
192	WSDP_B.Data(5)	T4	Bi-dir	WSDP™ Connector B Data Bit 5
193	WSDP_B.Data(6)	V2	Bi-dir	WSDP™ Connector B Data Bit 6
194	WSDP_B.Data(7)	V1	Bi-dir	WSDP™ Connector B Data Bit 7
195	WSDP_B.Data(8)	T5	Bi-dir	WSDP™ Connector B Data Bit 8
196	WSDP_B.Data(9)	U2	Bi-dir	WSDP™ Connector B Data Bit 9
197	WSDP_B.Data(10)	R5	Bi-dir	WSDP™ Connector B Data Bit 10
198	WSDP_B.Data(11)	U1	Bi-dir	WSDP™ Connector B Data Bit 11
199	WSDP_B.Data(12)	R2	Bi-dir	WSDP™ Connector B Data Bit 12
200	WSDP_B.Data(13)	R4	Bi-dir	WSDP™ Connector B Data Bit 13
201	WSDP_B.Data(14)	T1	Bi-dir	WSDP™ Connector B Data Bit 14
202	WSDP_B.Data(15)	T2	Bi-dir	WSDP™ Connector B Data Bit 15
203	WSDP_B.Data(16)	R1	Bi-dir	WSDP™ Connector B Data Bit 16
204	WSDP_B.Data(17)	P5	Bi-dir	WSDP™ Connector B Data Bit 17
205	WSDP_B.Data(18)	P3	Bi-dir	WSDP™ Connector B Data Bit 18
206	WSDP_B.Data(19)	P4	Bi-dir	WSDP™ Connector B Data Bit 19
207	WSDP_B.Data(20)	P2	Bi-dir	WSDP™ Connector B Data Bit 20
208	WSDP_B.Data(21)	N5	Bi-dir	WSDP™ Connector B Data Bit 21

Pad Index	Pad Name	Pin Loc	Dir	Pad Description
209	WSDP_B.Data(22)	P1	Bi-dir	WSDP™ Connector B Data Bit 22
210	WSDP_B.Data(23)	N4	Bi-dir	WSDP™ Connector B Data Bit 23
211	WSDP_B.Data(24)	N3	Bi-dir	WSDP™ Connector B Data Bit 24
212	WSDP_B.Data(25)	N2	Bi-dir	WSDP™ Connector B Data Bit 25
213	WSDP_B.Data(26)	M5	Bi-dir	WSDP™ Connector B Data Bit 26
214	WSDP_B.Data(27)	M3	Bi-dir	WSDP™ Connector B Data Bit 27
215	WSDP_B.Data(28)	M4	Bi-dir	WSDP™ Connector B Data Bit 28
216	WSDP_B.Data(29)	N1	Bi-dir	WSDP™ Connector B Data Bit 29
217	WSDP_B.Data(30)	M6	Bi-dir	WSDP™ Connector B Data Bit 30
218	WSDP_B.Data(31)	M1	Bi-dir	WSDP™ Connector B Data Bit 31
219	WSDP_B.Ctrl(0)	W5	Bi-dir	WSDP™ Connector B Control Bit 0
220	WSDP_B.Ctrl(1)	Y2	Bi-dir	WSDP™ Connector B Control Bit 1
221	WSDP_B.Ctrl(2)	V3	Bi-dir	WSDP™ Connector B Control Bit 2
222	WSDP_B.Ctrl(3)	V4	Bi-dir	WSDP™ Connector B Control Bit 3
223	WSDP_B.Ctrl(4)	AB5	Bi-dir	WSDP™ Connector B Control Bit 4
224	WSDP_B.Ctrl(5)	AA4	Bi-dir	WSDP™ Connector B Control Bit 5
225	WSDP_B.Ctrl(6)	AB4	Bi-dir	WSDP™ Connector B Control Bit 6
226	WSDP_B.Ctrl(7)	W3	Bi-dir	WSDP™ Connector B Control Bit 7
227	WSDP_B.Ctrl(8)	V8	Bi-dir	WSDP™ Connector B Control Bit 8
228	WSDP_B.Ctrl(9)	Y8	Bi-dir	WSDP™ Connector B Control Bit 9
229	WSDP_B_HS	AB8	Bi-dir	WSDP™ Connector B Handshake Bit
230	Reset	P20	Input	Reset signal

ATTACHMENT C: PE0 GENERIC QUICK REFERENCE GUIDE

Clk_Std_IF Generics

The Clock Standard Interface has several generics that can be set by the user upon instantiation of this component. The table below indicates the generic name, type associated with the generic, valid values which can be used, and a description of each of the generic values:

Table C-1: Clk_Std_IF Generic Quick Reference Guide

Generic	Type	Values	Description
M_CLK_DLL_OUT	Clk_DLL_Out_Type	USE_1X, USE_2X, BOTH	Select which output of the DLL is routed to the global buffer.
U_CLK_DLL_OUT			
WSDPA_CLK_DLL_OUT			
WSDPB_CLK_DLL_OUT			
RACE_CLK_DLL_OUT			
M_CLK_SOURCE	M_Clk_Source_Type	OSCILLATOR, WSDP_A, WSDP_B	Select which Clock to send to the Mclk PLL.
U_CLK_SOURCE	U_Clk_Source_Type	OSCILLATOR, WSDP_A, WSDP_B	Select which Clock to send to the UCLK PLL.
M_CLK_DLL_TYPE	Clk_DLL_Type	LOW_FREQ, HIGH_FREQ, NONE	Select which type of DLL to use.
K_CLK_DLL_TYPE			
U_CLK_DLL_TYPE			
WSDPA_CLK_DLL_TYPE			
WSDPB_CLK_DLL_TYPE			
RACE_CLK_DLL_TYPE			
M_CLK_USE_BUFG	Boolean	TRUE, FALSE	Selects whether to route the output of a DLL to a BUFG or sent the output of the DLL directly into the part.
U_CLK_USE_BUFG			
WSDPA_CLK_USE_BUFG			
WSDPB_CLK_USE_BUFG			
RACE_CLK_USE_BUFG			

Backplane_Std_IF Generics

The Backplane Standard Interface has several generics that can be set by the user upon instantiation of this component. The table below indicates the generic name, type associated with the generic, valid values which can be used, and a description of each of the generic values:

Table C-2: Backplane_Std_IF Generic Quick Reference Guide

Generic	Type	Values	Description
INFF_DELAY	INFF_Delay_Type	DELAY, NODELAY	Selects whether to add delay to the input pad signal or bypass the delay
OBUF_DRIVE	OBUF_Drive_Type	SLOW_24mA, SLOW_12mA, SLOW_8mA, SLOW_6mA, SLOW_4mA, SLOW_2mA, FAST_24mA, FAST_12mA, FAST_8mA, FAST_6mA	Selects which drive strength to use on the output pad when a LVTTL buffer type is selected
CLK_OUTPUT_EN	Boolean	TRUE, FALSE	Selects whether or not to drive a clock out on the backplane.
DIV_XMIT_CLOCK	Boolean	TRUE, FALSE	Selects whether or not to divide the output by 2
XMIT_CLK_BIT	integer	1-62/ 1-30	Select which bit to send the clock out. If a non-differential mode is selected 62 different bits can be used to transmit the clock. If a differential mode is selected only 29 possible bits can be selected to drive the clock
INVERT_CLK	Boolean	TRUE, FALSE	Selects whether or not to invert the output clock.
BUFFER_TYPE	BUF_TYPE	LVTTL, LVCMOS2, LVDS, LVPECL	Selects which buffer type to use in the backplane interface

LAD64_Std_IF Generics

The LAD64 Standard Interface has one generic that can be set by the user upon instantiation of this component. The table below indicates the generic name, type associated with the generic, valid values which can be used, and a description of each of the generic values:

Table C-3: Backplane_Std_IF Generic Quick Reference Guide

Generic	Type	Values	Description
V25_ENABLED	Boolean	TRUE, FALSE	This generic is set to true when the backplane interface BUFFER_TYPE is set to a drive level which required 2.5V; otherwise this generic should be set to false. The default value is false.

LED_Std_IF Generics

The LED Standard Interface has one generic that can be set by the user upon instantiation of this component. The table below indicates the generic name, type associated with the generic, valid values which can be used, and a description of each of the generic values:

Table C-4: LED_Std_IF Generic Quick Reference Guide

Generic	Type	Values	Description
V25_ENABLED	Boolean	TRUE, FALSE	This generic is set to true when the backplane interface BUFFER_TYPE is set to a drive level which required 2.5V; otherwise this generic should be set to false. The default value is false.

Mem32_Std_IF Generics

The 32 Bit Memory Standard Interface has several generic that can be set by the user upon instantiation of this component. The table below indicates the generic name, type associated with the generic, valid values which can be used, and a description of each of the generic values:

Table C-5: Mem32_Std_IF Generic Quick Reference Guide

Generic	Type	Values	Description
INFF_DELAY	INFF_Delay_Type	DELAY, NODELAY	Selects whether to add delay to the input pad signal or bypass the delay
OBUF_DRIVE	OBUF_Drive_Type	SLOW_24mA, SLOW_12mA, SLOW_8mA, SLOW_6mA, SLOW_4mA, SLOW_2mA, FAST_24mA, FAST_12mA, FAST_8mA, FAST_6mA	Selects which drive strength to use on the output pad when a LVTTL buffer type is selected
HALF_POPULATED	Boolean	TRUE, FALSE	For Firebird™/PMC this generic should always be set to TRUE.*

* The 32 bit memory interface VHDL is used in both the FIREBIRD™/PMC model and in the FIREBIRD ™/PCI model. This generic controls the ordering of the address being sent to memory. Since FIREBIRD ™/PMC doesn't contain an address chip select line, the ordering of the memory address is exactly the same as a half populated FIREBIRD ™/PCI. This is why the generic must always be set to True.

Mem64_Std_IF Generics

The 64-Bit Memory Standard Interface has several generics that can be set by the user upon instantiation of this component. The table below indicates the generic name, type associated with the generic, valid values which can be used, and a description of each of the generic values:

Table C-6: Mem64_Std_IF Generic Quick Reference Guide

Generic	Type	Values	Description
INFF_DELAY	INFF_Delay_Type	DELAY, NODELAY	Selects whether to add delay to the input pad signal or bypass the delay

Generic	Type	Values	Description
OBUF_DRIVE	OBUF_Drive_Type	SLOW_24mA, SLOW_12mA, SLOW_8mA, SLOW_6mA, SLOW_4mA, SLOW_2mA, FAST_24mA, FAST_12mA, FAST_8mA, FAST_6mA	Selects which drive strength to use on the output pad when a LVTTL buffer type is selected
HALF_POPULATED	Boolean	TRUE, FALSE	For Firebird™/PMC this generic should always be set to TRUE.*

* The 64 bit memory interface VHDL is used in both the FIREBIRD™/PMC model and in the FIREBIRD™/PCI model. This generic controls the ordering of the address being sent to memory. Since FIREBIRD™/PMC doesn't contain an address chip select line, the ordering of the memory address is exactly the same as a half populated FIREBIRD™/PCI. This is why the generic must always be set to True.

ATTACHMENT D: BPE GENERIC QUICK REFERENCE GUIDE

Clk_Std_IF Generics

The Clock Standard Interface has several generics that can be set by the user upon instantiation of this component. The table below indicates the generic name, type associated with the generic, valid values which can be used, and a description of each of the generic values:

Table D-1: Clock Standard Interface Generics

Generic	Type	Value	Description
U_CLK_DLL_OUT	Clk_DLL_Out_Type	USE_1X, USE_2X, BOTH	Select which output of the DLL is routed to the global buffer.
M_CLK_DLL_OUT			
W_CLK_A_DLL_OUT			
W_CLK_B_DLL_OUT			
M_CLK_SOURCE	M_Clk_Source_Type	OSCILLATOR, W_CLK_A, W_CLK_B	Select which source each clock is derived from.
U_CLK_SOURCE	U_Clk_Source_Type	OSCILLATOR, W_CLK_A, W_CLK_B	Select which source each clock is derived from.
M_CLK_DLL_TYPE	Clk_DLL_Type	LOW_FREQ, HIGH_FREQ, NONE	Select which type of DLL to use.
U_CLK_DLL_TYPE			
W_CLK_A_DLL_TYPE			
W_CLK_B_DLL_TYPE			
W_CLK_A_BUFFER_TYPE	BUF_TYPE	LVTTL, SSTL2_I, SSTL2_II, SSTL3_I, SSTL3_II, LVCMOS2, LVDS, LVPECL	Selects which buffer type to use in the clock interface
W_CLK_B_BUFFER_TYPE			
DIV_Ext_Clk	Boolean	TRUE, FALSE	Selects whether or not to divide the output by 2
V25_ENABLED_PORTA	Boolean	TRUE, FALSE	If SSTL2_I, SSTL2_II, LVCMOS2, or LVDS is selected for the IO_CONN_A standard interface, this must be set to TRUE. If LVTTL, SSTL3_I, SSTL3_II, LVPECL is selected, for the IOCONN_A standard interface, this must be set to FALSE.
V25_ENABLED_PORTB	Boolean	TRUE, FALSE	If SSTL2_I, SSTL2_II, LVCMOS2, or LVDS is selected for the IO_CONN_B standard interface, this must be set to TRUE. If LVTTL, SSTL3_I, SSTL3_II, LVPECL is selected, for the IO_CONN_B standard interface, this must be set to FALSE.

WSDP_Std_IF Generics

The WSDP Standard Interface has one generic that can be set by the user upon instantiation of this component. The table below indicates the generic name, type associated with the generic, valid values which can be used, and a description of each of the generic values:

Table D-2: WSDP_Std_IF Generic Quick Reference Guide

Generic	Type	Values	Description
V25_ENABLED	Boolean	TRUE, FALSE	If the component which is instantiated is WSDP_A, this generic must be set to true when the io_conn_a interface BUFFER_TYPE is set to a drive level which required 2.5V; otherwise this generic should be set to false. If the component which is instantiated is WSDP_B, this generic must be set to true when the io_conn_b interface BUFFER_TYPE is set to a drive level which required 2.5V; otherwise this generic should be set to false.
PORT_SEL	PORT_TYPE	PORTA, PORTB	If the component is instantiated as WSDP_A, the PORT_SEL must be set to PORTA. If the component is instantiated as WSDP_B, the PORT_SEL must be set to PORTB.

IO_CONN_Std_IF Generics

The I/O Connector Standard Interface has one generic that can be set by the user upon instantiation of this component. The table below indicates the generic name, type associated with the generic, valid values which can be used, and a description of each of the generic values:

Table D-3: IO_Conn_Std_IF Generic Quick Reference Guide

Generic	Type	Values	Description
CLK_OUTPUT_EN	Boolean	TRUE, FALSE	Selects whether or not to drive a clock out on the backplane.
DIV_XMIT_CLOCK	Boolean	PORTA, PORTB	Selects whether or not to divide the output by 2
XMIT_CLK_BIT	integer	1-37/ 1-18	Select which bit to send the clock out. Typically, if a non-differential mode is selected, the clock is sent out on bit 32. If a differential mode is selected, typically the clock is sent out on bit 16
INVERT_CLK	Boolean	TRUE, FALSE	Selects whether or not to invert the output clock
BUFFER_TYPE	BUF_TYPE	LVTTL, SSTL2_I, SSTL2_II, SSTL3_I, SSTL3_II, LVCMOS2, LVDS, LVPECL	Selects which buffer type to use in the clock interface

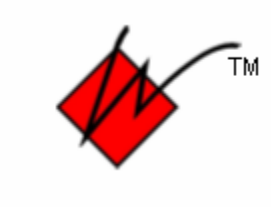
Generic	Type	Values	Description
WSDP_TERM	WSDP_TERM_TYPE	‘0’ or ‘1’ for all bits	Sets the output termination values when a non differential buffer type is selected. If a differential buffer type is selected the termination values are zero.
DIFF_TERM	DIFF_TERM_TYPE	‘0’ or ‘1’ for all bits	Sets the output termination values when a differential buffer type is selected. If a non-differential buffer type is selected the termination values are zero.

LED_Std_IF Generics

The LED Standard Interface has one generic that can be set by the user upon instantiation of this component. The table below indicates the generic name, type associated with the generic, valid values which can be used, and a description of each of the generic values:

Table D-4: Backplane_Std_IF Generic Quick Reference Guide

Generic	Type	Values	Description
V25_ENABLED_PORTA	Boolean	TRUE, FALSE	This generic is set to true when the I/O connector interface BUFFER_TYPE, is set to a drive level which required 2.5V; otherwise this generic should be set to false. If I/O Connector A is set to a drive level which requires 2.5V only WSDP_A needs to be set to true. Likewise if I/O connector B is set to a drive level which required 2.5V only WSDP_B needs to be set to true. The default value is false.


TM

ATTACHMENT E: EURO I/O DAUGHTER CARD REFERENCE GUIDE FOR FIREBIRD™/PCI

Introduction

The Euro I/O card presents a high bandwidth, flexible conduit for transmitting data to and from the FIREBIRD™/PCI board. FIREBIRD™/PCI is capable of accepting two of the Euro I/O cards detailed in this section.

The following figure details the specifications for the Euro I/O cards, with the connection signals listed in Table E-1.

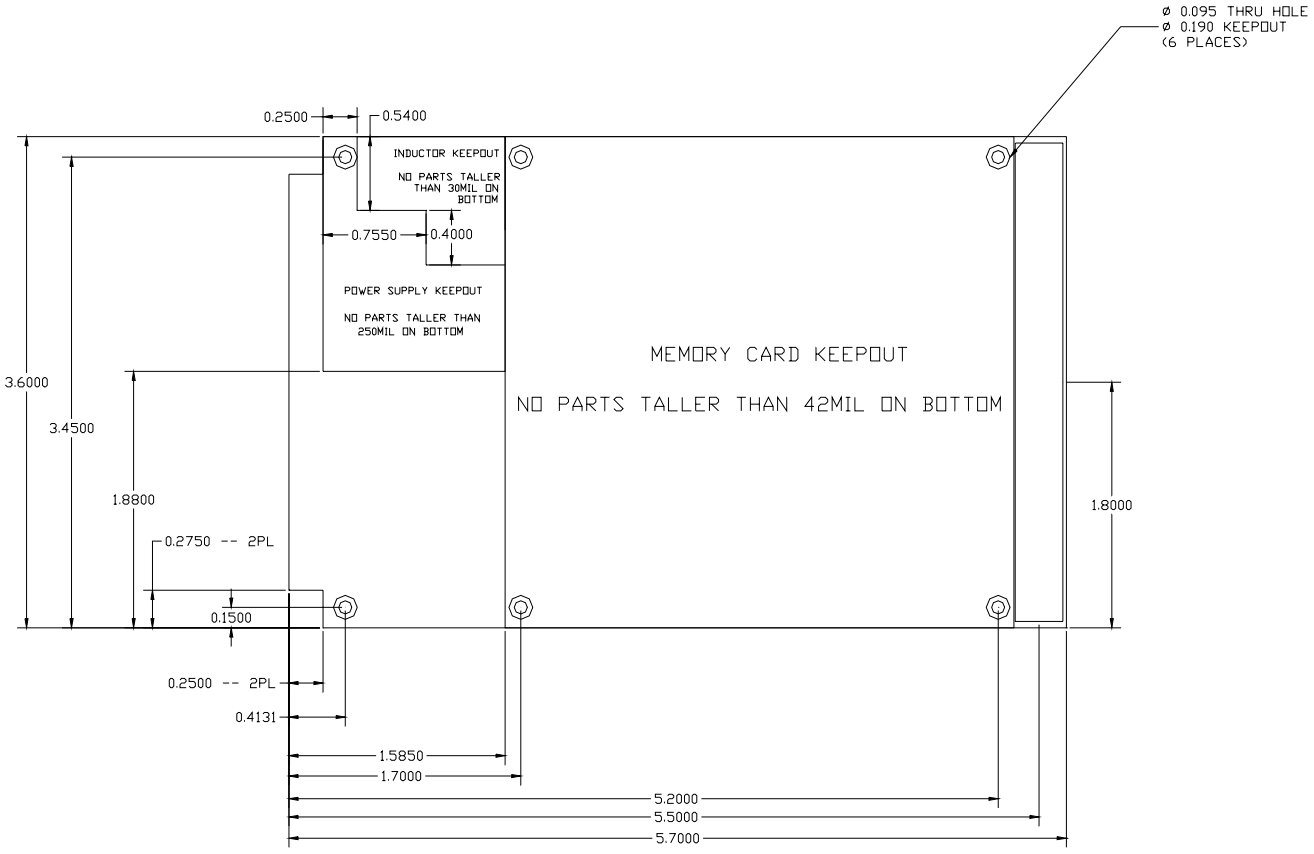


Figure E-1: Euro I/O Card Specifications

INFORMATION NOTE

The combined height of the WILDSTAR™ board, board connectors, Euro I/O cards thickness, and component side parts cannot exceed 14.48 mm (.570 inches). The maximum component height on the back side of the board is not to exceed 2.67 mm (.105 inches).

Table E-1: WILDSTAR™ External I/O_0 Connector Signal Description

External I/O Connector Pin Number	Port Signal	Direction of Signal	P0 Connector Name	P0 connector Pin Number
IOC0-1	KCLK	Input	N/C	N/C
IOC0-2	UCLKUSRC	Output	N/C	N/C
IOC0-3	INIT_n	Output	N/C	N/C
IOC0-4	PEAD63	Bi-dir.	N/C	N/C
IOC0-5	PEAD62	Bi-dir.	N/C	N/C
IOC0-6	PEAD61	Bi-dir.	N/C	N/C
IOC0-7	PEAD60	Bi-dir.	N/C	N/C
IOC0-8	PEAD59	Bi-dir.	N/C	N/C
IOC0-9	PEAD58	Bi-dir.	N/C	N/C
IOC0-10	PEAD57	Bi-dir.	N/C	N/C
IOC0-11	PEAD56	Bi-dir.	N/C	N/C
IOC0-12	PEAD55	Bi-dir.	N/C	N/C
IOC0-13	PEAD54	Bi-dir.	N/C	N/C
IOC0-14	PEAD53	Bi-dir.	N/C	N/C
IOC0-15	PEAD52	Bi-dir.	N/C	N/C
IOC0-16	PEAD51	Bi-dir.	N/C	N/C
IOC0-17	PEAD50	Bi-dir.	N/C	N/C
IOC0-18	PEAD49	Bi-dir.	N/C	N/C
IOC0-19	PEAD48	Bi-dir.	N/C	N/C
IOC0-20	PEAD47	Bi-dir.	N/C	N/C
IOC0-21	PEAD46	Bi-dir.	N/C	N/C
IOC0-22	PEAD45	Bi-dir.	N/C	N/C
IOC0-23	PEAD44	Bi-dir.	N/C	N/C
IOC0-24	PEAD43	Bi-dir.	N/C	N/C
IOC0-25	PEAD42	Bi-dir.	N/C	N/C
IOC0-26	PEAD41	Bi-dir.	N/C	N/C
IOC0-27	PEAD40	Bi-dir.	N/C	N/C
IOC0-28	PEAD39	Bi-dir.	N/C	N/C
IOC0-29	PEAD38	Bi-dir.	N/C	N/C
IOC0-30	PEAD37	Bi-dir.	N/C	N/C
IOC0-31	PEAD36	Bi-dir.	N/C	N/C
IOC0-32	PEAD35	Bi-dir.	N/C	N/C
IOC0-33	PEAD34	Bi-dir.	N/C	N/C
IOC0-34	PEAD33	Bi-dir.	N/C	N/C
IOC0-35	PEAD32	Bi-dir.	N/C	N/C
IOC0-36	PEAD31	Bi-dir.	N/C	N/C

External I/O Connector Pin Number	Port Signal	Direction of Signal	P0 Connector Name	P0 connector Pin Number
IOC0-37	PEAD30	Bi-dir.	N/C	N/C
IOC0-38	PEAD29	Bi-dir.	N/C	N/C
IOC0-39	PEAD28	Bi-dir.	N/C	N/C
IOC0-40	PEAD27	Bi-dir.	N/C	N/C
IOC0-41	PEAD26	Bi-dir.	N/C	N/C
IOC0-42	PEAD25	Bi-dir.	N/C	N/C
IOC0-43	PEAD24	Bi-dir.	N/C	N/C
IOC0-44	PEAD23	Bi-dir.	N/C	N/C
IOC0-45	PEAD22	Bi-dir.	N/C	N/C
IOC0-46	PEAD21	Bi-dir.	N/C	N/C
IOC0-47	PEAD20	Bi-dir.	N/C	N/C
IOC0-48	PEAD19	Bi-dir.	N/C	N/C
IOC0-49	PEAD18	Bi-dir.	N/C	N/C
IOC0-50	PEAD17	Bi-dir.	N/C	N/C
IOC0-51	PEAD16	Bi-dir.	N/C	N/C
IOC0-52	PEAD15	Bi-dir.	N/C	N/C
IOC0-53	PEAD14	Bi-dir.	N/C	N/C
IOC0-54	PEAD13	Bi-dir.	N/C	N/C
IOC0-55	PEAD12	Bi-dir.	N/C	N/C
IOC0-56	PEAD11	Bi-dir.	N/C	N/C
IOC0-57	PEAD10	Bi-dir.	N/C	N/C
IOC0-58	PEAD09	Bi-dir.	N/C	N/C
IOC0-59	PEAD08	Bi-dir.	N/C	N/C
IOC0-60	PEAD07	Bi-dir.	N/C	N/C
IOC0-61	PEAD06	Bi-dir.	N/C	N/C
IOC0-62	PEAD05	Bi-dir.	N/C	N/C
IOC0-63	PEAD04	Bi-dir.	N/C	N/C
IOC0-64	PEAD03	Bi-dir.	N/C	N/C
IOC0-65	PEAD04	Bi-dir.	N/C	N/C
IOC0-66	PEAD01	Bi-dir.	N/C	N/C
IOC0-67	PEAD00	Bi-dir.	N/C	N/C
IOC0-68	DEVSEL	Input	N/C	N/C
IOC0-69	TDI	Input	N/C	N/C
IOC0-70	UCLKFBK	Input	N/C	N/C
IOC0-71	TCK	Input	N/C	N/C
IOC0-72	UCLKMSTR	Input	N/C	N/C
IOC0-73	TMS	Input	N/C	N/C
IOC0-74	UCLKMSRC	Output	N/C	N/C
IOC0-75	TDO	Output	N/C	N/C
IOC0-76	HSO	Bi-dir.	N/C	N/C
IOC0-77	EX0DATA01	Bi-dir.	B01	P0-20
IOC0-78	EX0DATA00	Bi-dir.	A01	P0-1
IOC0-79	EX0DATA03	Bi-dir.	D01	P0-58
IOC0-80	EX0DATA02	Bi-dir.	C01	P0-39
IOC0-81	EX0DATA05	Bi-dir.	A02	P0-2
IOC0-82	EX0DATA04	Bi-dir.	E01	P0-77
IOC0-83	EX0DATA07	Bi-dir.	C02	P0-40
IOC0-84	EX0DATA06	Bi-dir.	B02	P0-21
IOC0-85	EX0DATA09	Bi-dir.	E02	P0-78
IOC0-86	EX0DATA08	Bi-dir.	D02	P0-59
IOC0-87	EX0DATA11	Bi-dir.	B03	P0-22
IOC0-88	EX0DATA10	Bi-dir.	A03	P0-3
IOC0-89	EX0DATA13	Bi-dir.	D03	P0-60

External I/O Connector Pin Number	Port Signal	Direction of Signal	P0 Connector Name	P0 connector Pin Number
IOC0-90	EX0DATA12	Bi-dir.	C03	P0-41
IOC0-91	EX0DATA15	Bi-dir.	A04	P0-4
IOC0-92	EX0DATA14	Bi-dir.	E03	P0-79
IOC0-93	EX0DATA17	Bi-dir.	C04	P0-42
IOC0-94	EX0DATA16	Bi-dir.	B04	P0-23
IOC0-95	EX0DATA19	Bi-dir.	E04	P0-80
IOC0-96	EX0DATA18	Bi-dir.	D04	P0-61
IOC0-97	EX0DATA21	Bi-dir.	B05	P0-24
IOC0-98	EX0DATA20	Bi-dir.	A05	P0-5
IOC0-99	EX0DATA23	Bi-dir.	D05	P0-62
IOC0-100	EX0DATA22	Bi-dir.	C05	P0-43
IOC0-101	EX0DATA25	Bi-dir.	A06	P0-6
IOC0-102	EX0DATA24	Bi-dir.	E05	P0-31
IOC0-103	EX0DATA27	Bi-dir.	C06	P0-44
IOC0-104	EX0DATA26	Bi-dir.	B06	P0-25
IOC0-105	EX0DATA29	Bi-dir.	E06	P0-82
IOC0-106	EX0DATA28	Bi-dir.	D06	P0-63
IOC0-107	EX0DATA31	Bi-dir.	B07	P0-25
IOC0-108	EX0DATA30	Bi-dir.	A07	P0-7
IOC0-109	EX0DATA33	Bi-dir.	D07	P0-64
IOC0-110	EX0DATA32	Bi-dir.	C07	P0-45
IOC0-111	EX0DATA35	Bi-dir.	A08	P0-8
IOC0-112	EX0DATA34	Bi-dir.	E07	P0-83
IOC0-113	EX0DATA37	Bi-dir.	C08	P0-46
IOC0-114	EX0DATA36	Bi-dir.	B08	P0-27
IOC0-115	EX0DATA39	Bi-dir.	E08	P0-84
IOC0-116	EX0DATA38	Bi-dir.	D08	P0-65
IOC0-117	EX0DATA41	Bi-dir.	B09	P0-28
IOC0-118	EX0DATA40	Bi-dir.	A09	P0-9
IOC0-119	EX0DATA43	Bi-dir.	D09	P0-66
IOC0-120	EX0DATA42	Bi-dir.	C09	P0-47
IOC0-121	EX0DATA45	Bi-dir.	A10	P0-10
IOC0-122	EX0DATA44	Bi-dir.	E09	P0-85
IOC0-123	EX0DATA47	Bi-dir.	C10	P0-48
IOC0-124	EX0DATA46	Bi-dir.	B10	P0-29
IOC0-125	EX0DATA49	Bi-dir.	E10	P0-86
IOC0-126	EX0DATA48	Bi-dir.	D10	P0-67
IOC0-127	EX0DATA51	Bi-dir.	B11	P0-30
IOC0-128	EX0DATA50	Bi-dir.	A11	P0-11
IOC0-129	EX0DATA53	Bi-dir.	D11	P0-68
IOC0-130	EX0DATA52	Bi-dir.	C11	P0-49
IOC0-131	EX0DATA55	Bi-dir.	A12	P0-12
IOC0-132	EX0DATA54	Bi-dir.	E11	P0-87
IOC0-133	EX0DATA57	Bi-dir.	C12	P0-50
IOC0-134	EX0DATA56	Bi-dir.	B12	P0-31
IOC0-135	EX0DATA59	Bi-dir.	E12	P0-88
IOC0-136	EX0DATA58	Bi-dir.	D12	P0-69
IOC0-137	EX0DATA61	Bi-dir.	B13	P0-32
IOC0-138	EX0DATA60	Bi-dir.	A13	P0-13
IOC0-139	EX0DATA63	Bi-dir.	D13	P0-70

External I/O Connector Pin Number	Port Signal	Direction of Signal	P0 Connector Name	P0 connector Pin Number
IOC0-140	EX0DATA62	Bi-dir.	C13	P0-51
IOC0-141	EX0DATA65	Bi-dir.	A14	P0-14
IOC0-142	EX0DATA64	Bi-dir.	E13	P0-89
IOC0-143	EX0DATA67	Bi-dir.	C14	P0-52
IOC0-144	EX0DATA66	Bi-dir.	B14	P0-33
IOC0-145	EX0DATA69	Bi-dir.	E14	P0-90
IOC0-146	EX0DATA68	Bi-dir.	D14	P0-71
IOC0-147	EX0DATA71	Bi-dir.	B15	P0-34
IOC0-148	EX0DATA70	Bi-dir.	A15	P0-15
IOC0-149	EX0DATA73	Bi-dir.	D15	P0-72
IOC0-150	EX0DATA72	Bi-dir.	C15	P0-53
IOC0-151	EX0DATA75	Bi-dir.	A16	P0-16
IOC0-152	EX0DATA74	Bi-dir.	E15	P0-91
IOC0-153	EX0DATA77	Bi-dir.	C16	P0-54
IOC0-154	EX0DATA76	Bi-dir.	B16	P0-35
IOC0-155	EX0DATA79	Bi-dir.	E16	P0-92
IOC0-156	EX0DATA78	Bi-dir.	D16	P0-73
IOC0-157	EX0DATA81	Bi-dir.	B17	P0-36
IOC0-158	EX0DATA80	Bi-dir.	A17	P0-17
IOC0-159	EX0DATA83	Bi-dir.	D17	P0-74
IOC0-160	EX0DATA82	Bi-dir.	C17	P0-55
IOC0-161	EX0DATA85	Bi-dir.	A18	P0-18
IOC0-162	EX0DATA84	Bi-dir.	E17	P0-93
IOC0-163	EX0DATA87	Bi-dir.	C18	P0-56
IOC0-164	EX0DATA86	Bi-dir.	B18	P0-37
IOC0-165	EX0DATA89	Bi-dir.	E18	P0-94
IOC0-166	EX0DATA88	Bi-dir.	D18	P0-75
IOC0-167	EX0DATA91	Bi-dir.	B19	P0-38
IOC0-168	EX0DATA90	Bi-dir.	A19	P0-19
IOC0-169	EX0DATA93	Bi-dir.	D19	P0-76
IOC0-170	EX0DATA92	Bi-dir.	C19	P0-57
IOC0-171	N/C	N/A	N/A	N/A
IOC0-172	EX0DATA94	Bi-dir.	E19	P0-95
IOC0-173	IDCE_n	Input	N/C	N/C
IOC0-174	HS1	Bi-dir.	N/C	N/C
IOC0-175	DXP	Output	N/C	N/C
IOC0-176	IDCLK	Input	N/C	N/C
IOC0-177	DXN	Output	N/C	N/C
IOC0-178	IDDATA	Output	N/C	N/C
IOC0-179	INT_n	Output	N/C	N/C
IOC0-180	DMACH0	Output	N/C	N/C
IOC0-181	DS_n	Input	N/C	N/C
IOC0-182	DMACH1	Output	N/C	N/C
IOC0-183	AS_n	Input	N/C	N/C
IOC0-184	DMAST1	Output	N/C	N/C
IOC0-185	CS_n	Input	N/C	N/C
IOC0-186	DMAST0	Output	N/C	N/C
IOC0-187	REG_n	Input	N/C	N/C
IOC0-188	WR_n	Input	N/C	N/C
IOC0-189	LAD01	Bi-dir.	N/C	N/C
IOC0-190	LAD00	Bi-dir.	N/C	N/C
IOC0-191	LAD03	Bi-dir.	N/C	N/C
IOC0-192	LAD02	Bi-dir.	N/C	N/C

External I/O Connector Pin Number	Port Signal	Direction of Signal	P0 Connector Name	P0 connector Pin Number
IOC0-193	LAD05	Bi-dir.	N/C	N/C
IOC0-194	LAD04	Bi-dir.	N/C	N/C
IOC0-195	LAD07	Bi-dir.	N/C	N/C
IOC0-196	LAD06	Bi-dir.	N/C	N/C
IOC0-197	LAD09	Bi-dir.	N/C	N/C
IOC0-198	LAD08	Bi-dir.	N/C	N/C
IOC0-199	LAD11	Bi-dir.	N/C	N/C
IOC0-200	LAD10	Bi-dir.	N/C	N/C
IOC0-201	LAD13	Bi-dir.	N/C	N/C
IOC0-202	LAD12	Bi-dir.	N/C	N/C
IOC0-203	LAD15	Bi-dir.	N/C	N/C
IOC0-204	LAD14	Bi-dir.	N/C	N/C
IOC0-205	LAD17	Bi-dir.	N/C	N/C
IOC0-206	LAD16	Bi-dir.	N/C	N/C
IOC0-207	LAD19	Bi-dir.	N/C	N/C
IOC0-208	LAD18	Bi-dir.	N/C	N/C
IOC0-209	LAD21	Bi-dir.	N/C	N/C
IOC0-210	LAD20	Bi-dir.	N/C	N/C
IOC0-211	LAD23	Bi-dir.	N/C	N/C
IOC0-212	LAD22	Bi-dir.	N/C	N/C
IOC0-213	LAD25	Bi-dir.	N/C	N/C
IOC0-214	LAD24	Bi-dir.	N/C	N/C
IOC0-215	LAD27	Bi-dir.	N/C	N/C
IOC0-216	LAD26	Bi-dir.	N/C	N/C
IOC0-217	LAD29	Bi-dir.	N/C	N/C
IOC0-218	LAD28	Bi-dir.	N/C	N/C
IOC0-219	LAD31	Bi-dir.	N/C	N/C
IOC0-220	LAD30	Bi-dir.	N/C	N/C
IOC0-221	DONE	Output	N/C	N/C
IOC0-222	RESET_n	Input	N/C	N/C
IOC0-223	MCLK	Input	N/C	N/C
IOC0-224	PCLK	Input	N/C	N/C
IOC0-225	PROG_n	Input	N/C	N/C
IOC0-226	GND	Input	N/C	N/C
IOC0-227	+12V	Input	N/C	N/C
IOC0-228	-12V	Input	N/C	N/C

Installation Instructions

Euro I/O card installation instructions are provided in the manual that accompanies the card.

ATTACHMENT F: FIREBIRD™/PMC DUAL WSDP™: USER VHDL MAPPING TO MICTOR PINS

Table F-1: FIREBIRD™/PMC DUAL WSDP™ LVDS PAIRS

PORT 0 (Port A)

WSDP™ Signal* (Single Ended)	WSDP™ Signal* (Differential)	WSDP™ Signal (Differential)	Polarity	VTT Termination Control	Differential Termination Control	BPE0 Pin	Mictor Pin on Front panel
IO_Conn_x.Data_x.0		Data 00	-	WSDP_TERM(0)		D16	38
IO_Conn_x.Data_x.1	IO_Conn_x.Data_x.0	Data 01	+	WSDP_TERM(0)	DIFF_TERM(0)	C17	36
IO_Conn_x.Data_x.2		Data 02	-	WSDP_TERM(0)		D15	34
IO_Conn_x.Data_x.3	IO_Conn_x.Data_x.1	Data 03	+	WSDP_TERM(0)	DIFF_TERM(1)	A17	32
IO_Conn_x.Data_x.4		Data 04	-	WSDP_TERM(0)		D14	30
IO_Conn_x.Data_x.5	IO_Conn_x.Data_x.2	Data 05	+	WSDP_TERM(0)	DIFF_TERM(2)	C16	28
IO_Conn_x.Data_x.6		Data 06	-	WSDP_TERM(0)		B15	26
IO_Conn_x.Data_x.7	IO_Conn_x.Data_x.3	Data 07	+	WSDP_TERM(0)	DIFF_TERM(3)	C15	24
IO_Conn_x.Data_x.8		Data 08	-	WSDP_TERM(1)		F12	22
IO_Conn_x.Data_x.9	IO_Conn_x.Data_x.4	Data 09	+	WSDP_TERM(1)	DIFF_TERM(4)	A15	20
IO_Conn_x.Data_x.10		Data 10	-	WSDP_TERM(1)		B14	18
IO_Conn_x.Data_x.11	IO_Conn_x.Data_x.5	Data 11	+	WSDP_TERM(1)	DIFF_TERM(5)	C14	16
IO_Conn_x.Data_x.12		Data 12	-	WSDP_TERM(1)		C13	14
IO_Conn_x.Data_x.13	IO_Conn_x.Data_x.6	Data 13	+	WSDP_TERM(1)	DIFF_TERM(6)	D13	12
IO_Conn_x.Data_x.14		Data 14	-	WSDP_TERM(1)		B12	10
IO_Conn_x.Data_x.15	IO_Conn_x.Data_x.7	Data 15	+	WSDP_TERM(1)	DIFF_TERM(7)	A13	8
IO_Conn_x.Data_x.16		Data 16	-	WSDP_TERM(2)		C12	37
IO_Conn_x.Data_x.17	IO_Conn_x.Data_x.8	Data 17	+	WSDP_TERM(2)	DIFF_TERM(8)	D12	35

WSDP™ Signal* (Single Ended)	WSDP™ Signal* (Differential)	WSDP™ Signal (Differential)	Polarity	VTT Termination Control	Differential Termination Control	BPE0 Pin	Mictor Pin on Front panel
IO_Conn_x.Data_x.18		Data 18	-	WSDP_TERM(2)		D9	33
IO_Conn_x.Data_x.19	IO_Conn_x.Data_x.9	Data 19	+	WSDP_TERM(2)	DIFF_TERM(9)	A8	31
IO_Conn_x.Data_x.20		Data 20	-	WSDP_TERM(2)		C8	29
IO_Conn_x.Data_x.21	IO_Conn_x.Data_x.10	Data 21	+	WSDP_TERM(2)	DIFF_TERM(10)	B8	27
IO_Conn_x.Data_x.22		Data 22	-	WSDP_TERM(2)		A7	25
IO_Conn_x.Data_x.23	IO_Conn_x.Data_x.11	Data 23	+	WSDP_TERM(2)	DIFF_TERM(11)	E9	23
IO_Conn_x.Data_x.24		Data 24	-	WSDP_TERM(3)		D7	21
IO_Conn_x.Data_x.25	IO_Conn_x.Data_x.12	Data 25	+	WSDP_TERM(3)	DIFF_TERM(12)	C7	19
IO_Conn_x.Data_x.26		Data 26	-	WSDP_TERM(3)		A5	17
IO_Conn_x.Data_x.27	IO_Conn_x.Data_x.13	Data 27	+	WSDP_TERM(3)	DIFF_TERM(13)	B6	15
IO_Conn_x.Data_x.28		Data 28	-	WSDP_TERM(3)		A4	13
IO_Conn_x.Data_x.29	IO_Conn_x.Data_x.14	Data 29	+	WSDP_TERM(3)	DIFF_TERM(14)	E7	11
IO_Conn_x.Data_x.30		Data 30	-	WSDP_TERM(3)		D5	9
IO_Conn_x.Data_x.31	IO_Conn_x.Data_x.15	Data 31	+	WSDP_TERM(3)	DIFF_TERM(15)	B3	7
IO_Conn_x.Data_x.34		CNTL 0	-	WSDP_TERM(0)		F11	4
IO_Conn_x.Data_x.35	IO_Conn_x.Data_x.17	CNTL 2	+	WSDP_TERM(1)	DIFF_TERM(17)	B10	2
IO_Conn_x.Data_x.36		CNTL 1	-	WSDP_TERM(2)		C9	3
IO_Conn_x.Data_x.37	IO_Conn_x.Data_x.18	CNTL 3	+	WSDP_TERM(3)	DIFF_TERM(18)	E10	1
IO_Conn_x.Data_x.33		WSDPCLKIN	+	WSDP_TERM(9)		D11,C11	5
IO_Conn_x.Data_x.32	IO_Conn_x.Data_x.16	WSDPCLKOUT	-	WSDP_TERM(8)	DIFF_TERM(16)	B11	6

* "x" can be Input or Output

Table F-2: FIREBIRD™/PMC DUAL WSDP™ LVDS PAIRS

PORT 1 (Port B)

WSDP™ Signal* (Single Ended)	WSDP™ Signal* (Differential)	WSDP™ Signal (Differential)	Polarity	Paired with when terminated as differential	VTT Termination Control	Differential Termination Control	BPE0 Pin	Mictor Pin on Front panel
IO_Conn_x.Data_x.0		Data 00	-	Data 01	WSDP_TERM(0)		AB3	38
IO_Conn_x.Data_x.1	IO_Conn_x.Data_x.0	Data 01	+	Data 00	WSDP_TERM(0)	DIFF_TERM(0)	V7	36
IO_Conn_x.Data_x.2		Data 02	-	Data 03	WSDP_TERM(0)		W6	34
IO_Conn_x.Data_x.3	IO_Conn_x.Data_x.1	Data 03	+	Data 02	WSDP_TERM(0)	DIFF_TERM(1)	Y7	32
IO_Conn_x.Data_x.4		Data 04	-	Data 05	WSDP_TERM(0)		AA6	30
IO_Conn_x.Data_x.5	IO_Conn_x.Data_x.2	Data 05	+	Data 04	WSDP_TERM(0)	DIFF_TERM(2)	AB6	28
IO_Conn_x.Data_x.6		Data 06	-	Data 07	WSDP_TERM(0)		AA7	26
IO_Conn_x.Data_x.7	IO_Conn_x.Data_x.3	Data 07	+	Data 06	WSDP_TERM(0)	DIFF_TERM(3)	W7	24
IO_Conn_x.Data_x.8		Data 08	-	Data 09	WSDP_TERM(1)		AA8	22
IO_Conn_x.Data_x.9	IO_Conn_x.Data_x.4	Data 09	+	Data 08	WSDP_TERM(1)	DIFF_TERM(4)	V9	20
IO_Conn_x.Data_x.10		Data 10	-	Data 11	WSDP_TERM(1)		W9	18
IO_Conn_x.Data_x.11	IO_Conn_x.Data_x.5	Data 11	+	Data 10	WSDP_TERM(1)	DIFF_TERM(5)	AB9	16
IO_Conn_x.Data_x.12		Data 12	-	Data 13	WSDP_TERM(1)		Y9	14
IO_Conn_x.Data_x.13	IO_Conn_x.Data_x.6	Data 13	+	Data 12	WSDP_TERM(1)	DIFF_TERM(6)	V10	12
IO_Conn_x.Data_x.14		Data 14	-	Data 15	WSDP_TERM(1)		V12	10
IO_Conn_x.Data_x.15	IO_Conn_x.Data_x.7	Data 15	+	Data 14	WSDP_TERM(1)	DIFF_TERM(7)	Y12	8
IO_Conn_x.Data_x.16		Data 16	-	Data 17	WSDP_TERM(2)		AA12	37
IO_Conn_x.Data_x.17	IO_Conn_x.Data_x.8	Data 17	+	Data 16	WSDP_TERM(2)	DIFF_TERM(8)	W13	35
IO_Conn_x.Data_x.18		Data 18	-	Data 19	WSDP_TERM(2)		V13	33
IO_Conn_x.Data_x.19	IO_Conn_x.Data_x.9	Data 19	+	Data 18	WSDP_TERM(2)	DIFF_TERM(9)	AB14	31
IO_Conn_x.Data_x.20		Data 20	-	Data 21	WSDP_TERM(2)		AA14	29
IO_Conn_x.Data_x.21	IO_Conn_x.Data_x.10	Data 21	+	Data 20	WSDP_TERM(2)	DIFF_TERM(10)	V14	27
IO_Conn_x.Data_x.22		Data 22	-	Data 23	WSDP_TERM(2)		Y14	25
IO_Conn_x.Data_x.23	IO_Conn_x.Data_x.11	Data 23	+	Data 22	WSDP_TERM(2)	DIFF_TERM(11)	W15	23
IO_Conn_x.Data_x.24		Data 24	-	Data 25	WSDP_TERM(3)		AB15	21
IO_Conn_x.Data_x.25	IO_Conn_x.Data_x.12	Data 25	+	Data 24	WSDP_TERM(3)	DIFF_TERM(12)	AA15	19
IO_Conn_x.Data_x.26		Data 26	-	Data 27	WSDP_TERM(3)		V15	17
IO_Conn_x.Data_x.27	IO_Conn_x.Data_x.13	Data 27	+	Data 26	WSDP_TERM(3)	DIFF_TERM(13)	Y16	15
IO_Conn_x.Data_x.28		Data 28	-	Data 29	WSDP_TERM(3)		AA17	13
IO_Conn_x.Data_x.29	IO_Conn_x.Data_x.14	Data 29	+	Data 28	WSDP_TERM(3)	DIFF_TERM(14)	W16	11
IO_Conn_x.Data_x.30		Data 30	-	Data 31	WSDP_TERM(3)		V16	9
IO_Conn_x.Data_x.31	IO_Conn_x.Data_x.15	Data 31	+	Data 30	WSDP_TERM(3)	DIFF_TERM(15)	AA18	7
IO_Conn_x.Data_x.34		CNTL 0	-	CNTL 2	WSDP_TERM(0)		W10	4
IO_Conn_x.Data_x.35	IO_Conn_x.Data_x.17	CNTL 2	+	CNTL 0	WSDP_TERM(1)	DIFF_TERM(17)	AB10	2
IO_Conn_x.Data_x.36		CNTL 1	-	CNTL 3	WSDP_TERM(2)		W11	3
IO_Conn_x.Data_x.37	IO_Conn_x.Data_x.18	CNTL 3	+	CNTL 1	WSDP_TERM(3)	DIFF_TERM(18)	AB11	1

WSDP™ Signal* (Single Ended)	WSDP™ Signal* (Differential)	WSDP™ Signal (Differential)	Polarity	Paired with when terminated as differential	VTT Termination Control	Differential Termination Control	BPE0 Pin	Mictor Pin on Front panel
IO_Conn_x.Data_x.33		WSDPCLKIN	+	WSDPCLKOUT	WSDP_TERM(9)		U12,Y11	5
IO_Conn_x.Data_x.32	IO_Conn_x.Data_x.16	WSDPCLKOUT	-	WSDPCLKIN	WSDP_TERM(8)	DIFF_TERM(16)	AA11	6

* "x" can be Input or Output

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