PLB RapidIO LVDS

DS455 June 4, 2004 **Product Specification**

Introduction

This document presents the design specification for the Xilinx PLB RapidIO™ LVDS Intellectual Property (IP) solution. This LogiCORE™ module provides an interface between the IBM® CoreConnect™ Processor Local Bus (PLB) and an LVDS based RapidIO interface standard.

The PLB RapidIO LVDS design provides an interface between the PPC405 (via PLB CoreConnect Bus) and a RapidIO protocol network. The physical interface to the RapidIO bus uses the 8 bit LVDS standard.

Features

The PLB RapidIO LVDS is a soft IP core designed for Xilinx FPGAs incorporating PPC405 and MicroBlaze processing elements. The design provides the following features:

- Front end Interface to the IBM CoreConnect PLB Bus
  - Supports PLB signaling per the IBM *64-Bit Processor Local Bus, Architectural Specification*
  - Integrates easily with the Xilinx Platform Studio for PPC405 System Development.
  - 64 bit wide data transfers
  - 512x64 Tx and Rx Packet Buffers (Up to 8 maximally sized packets can be queued for Tx and Rx)
  - PLB Cacheline and Burst Transfer Interface Support with Packet Buffers.
  - Parameterized System Address Block.
- Back end Interface to RapidIO Bus.
  - Incorporates Xilinx RapidIO Physical Layer
    - Supports *RapidIO Physical Layer 8/16 LP-LVDS Interconnect Specification v1.1.*
    - 8-bit LVDS PHY (TX and Rx functions)
    - 500 MBytes/sec Peak Transfer Rate at the PHY Tx and Rx ports.
- Processor Accessible Registers
  - Interrupt Enable and Status Registers
  - S/W Reset/MIR Register
  - RapidIO PHY Link Status Register
  - RapidIO PHY Management Register Set
- System Interrupt Support
  - Tx and Rx Flow Control Interrupts
  - Programmable Enables/Disables.
  - Interrupt Status Registers can support S/W Polled Mode control flow in place of Interrupt Control Flow
- PLB System clock frequency up to 100 MHz

LogiCORE™ Facts		
Core Specifics		
Supported Device Family	Virtex-II Pro™, Virtex-II™,	
Version of Core	plb_rapidio_lvds	v1.00c
Resources Used		
	Min	Max
I/O	40	40
LUTs	6,077	6,345
FFs	2,907	3,032
Block RAMs	4	4
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	Example UCF	
Verification	N/A	
Instantiation Template	None	
Reference Designs	None	
Design Tool Requirements		
Xilinx Implementation Tools	6.2i	
Verification	N/A	
Simulation	ModelSim 5.7e	
Synthesis	XST	
Support		
Support provided by Xilinx, Inc.		

PLB RapidIO LVDS Parameters

The PLB RapidIO LVDS provides for user interface tailoring via 20 input parameters. These parameters are detailed in Table 1. The parameters are separated into three groups. The PLB Interface group allows for customizing the PLB bus interface to the user's PLB system.

The FPGA Family Type parameter is used to select the target FPGA family type. Currently, this design only supports Virtex™II type devices.

The RapidIO PHY Configuration parameters allows the user to customize the RapidIO PHY interface.

Table 1: **PLB RapidIO LVDS Design Parameters**

Grouping / Number		Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
PLB Interface	G1	Registers and FIFOs Base Address	C_REG_FIFO_BASEADDR	0000_0000 to FFFF_8000	6000_0000	std_logic_vector
	G2	Registers and FIFOs High Address	C_REG_FIFO_HIGHADDR	C_BASEADDR + 0000_3FFF	6000_1FFF	std_logic_vector
	G3	Packet Buffers Base Address	C_PBUF_BASEADDR	0000_0000 to FFFF_8000	6000_2000	std_logic_vector
	G4	Packet Buffers High Address	C_PBUF_HIGHADDR	0000_0000 to FFFF_8000	6000_3FFF	std_logic_vector
	G5	Device Block ID	C_DEV_BLK_ID	0 to 255	90	integer
	G6	Burst Support Enable	C_DEV_BURST_ENABLE	0 = Burst Support Disabled 1 = Burst Support Enabled	0 (Disabled)	integer
	G7	PLB Master ID Bus Width	C_PLB_MID_WIDTH	$\log_2$(**G8**) (min value of 1)	4	integer
	G8	Number of PLB Masters	C_PLB_NUM_MASTERS	1 to 16	16	integer
	G9	Width of the PLB Address Bus	C_PLB_AWIDTH	32	32 (1)	integer
	G10	Width of the PLB Data Bus	C_PLB_DWIDTH	64	64 (1)	integer
	G11	PLB Clock Period in picoseconds	C_PLB_CLK_PERIOD_PS	up to 10000 (100MHz)	10000	integer
FPGA Family Type	G12	Xilinx FPGA Family	C_FAMILY (2)	virtex2, virtex2p	virtex2p	string

Table 1: **PLB RapidIO LVDS Design Parameters *(Continued)***

Grouping / Number		Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
RapidIO PHY Configuration	G13	Host Device Indication at reset	C_GEN_CTRL_HOST_RESET_VALUE	'0' = Agent or Slave Device '1' = Host Device	'1'	std_logic
	G14	Master Enable value at reset	C_GEN_CTRL_MASTER_ENABLE_RESET_VALUE	'0' = Response Only Device '1' = Request and Response Device	'1'	std_logic
	G15	Host Discovered value at reset	C_GEN_CTRL_DISCOVERED_RESET_VALUE	'0' = Not Discovered by Host '1' = Discovered by Host	'0'	std_logic
	G16	Extended Features Pointer	C_PORT_MAINT_EF_PTR	X"0000 - FFFF"	X"0000"	std_logic_vector(0 to 15)
	G17	Generic End Point Configuration	C_CSR_GENERIC_END_POINT_MAP	'0' = Generic End Point Free Device '1' = Generic End Point Device	'1'	std_logic
	G18	Output port driver enable at reset	C_OUTPUT_PORT_ENABLE_RST (3)	'0' = Output port disabled at reset '1' = Outut port enabled at reset	'1'	std_logic
	G19	Multicast event participant	C_MULTICAST_EVENT_PARTIC_RST (3)	'0' = Multicast Event participation disabled '1' = Multicast Event participation enabled	'1'	std_logic
	G20	Port Map Selection	C_PORT_MAP_SEL (4)	X"0" to X"F"	X"0"	std_logic_vector(0 to 3)
	G21	Throttle Interval	C_THROTTLE_INTERVAL	X"00" to X"FF"	X"0F"	std_logic_vector(0 to 7)
	G22	Pacing Idle	C_PACING_IDLE	X"0" to X"F"	X"5"	std_logic_vector(0 to 3)

Table 1: **PLB RapidIO LVDS Design Parameters *(Continued)***

Grouping / Number		Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
PHY Tx Clocking Source	G23	Internal Tx DCM omit/include selection	C_INCLUDE_TX_DCM (5)	0 = Omit internal Tx DCM and BUFG logic 1 = Include internal Tx DCM and BUFG logic	1	Integer
PLB Abort Support	G24	Selection for PLB Abort support	C_SUPPORT_PLB_ABORT (6)	0 = Disable PLB_abort support 1 = Enable PLB_abort support	1	Integer

Notes:
1. This parameter is provided for future growth. It must be set to the default value shown.
2. This design currently supports only Virtex™-II and Virtex-II Pro family devices.
3. This PHY configuration parameter will be mapped to the appropriate address based upon the value of **G20**. For example, if **G20** is set to 0x1, the reset values will map to offset 0x78 and 0x7C.
4. This PHY configuration parameter sets the port response address mapping for Port N Error, Status CSR, and Port N Control CSR. The values range from 0x0 to 0xF. For example, if **G20** is set to 0x1, the PHY will respond to offsets in the range of 0x60 to 0x78.
5. This parameter allows the User to omit or include logic (DCM and BUFGs) for the generation of the Tx clocking signals required by the PHY. If omitted, the User must provide the 4 Tx clocking inputs to this core.
6. This parameter allows the User to disable support of PLB_abort. This is only be necessary if the design F_{max} timing is being missed due to the combinational delays incurred with the PLB_abort input signal. If PLB_abort is disabled, **the User must ensure** that no PLB Master device accessing the PLB RapidIO LVDS Slave will abort a transaction addressed to it.

Allowable Parameter Combinations

Parameter **G1** and **G3** must be an address that is evenly divisible by 0000_4000_{hex} to assure proper address decoding.

Parameter **G2** must assign a minimum PLB address space usage of $13FF_{hex}$ for the PLB RapidIO LVDS registers. It must comply with the following relationship:

G2 >= **G1** + 0000_13FF_{hex} .

Parameter **G4** must assign a minimum PLB address space usage of $1FFF_{hex}$ for the PLB RapidIO LVDS data buffers. It must comply with the following relationship:

G4 >= **G3** + 0000_1FFF_{hex} .

Parameter **G7** is dependent on **G8**. These two parameters must follow the relationship:

G7 = $\log_2$(**G8**) or 1, whichever is greater (a value of zero is not allowed).

Parameters **G9** and **G10** should only be set to 32 and 64 respectively for this version of the PLB RapidIO LVDS design.

PLB RapidIO LVDS I/O Signals

The PLB RapidIO LVDS has two major interfaces. These are to the PLB Bus and the RapidIO physical interface. The device also generates an interrupt for use by a processing element. The I/O signals for the design are listed in Table 2. The interfaces referenced in this table are shown in Figure 13 in the PLB RapidIO LVDS block diagram.

Table 2: **PLB RapidIO LVDS I/O Signals**

Grouping		Signal Name	Interface	I/O	Description	Page
PLB Bus Input Signals	P1	PLB_clk	PLB Bus	I	PLB main bus clock. See table note 1.	
	P2	PLB_Rst	PLB Bus	I	PLB main bus reset. See table note 1.	
	P3	PLB_ABus(0:**G9**-1)	PLB Bus	I	See table note 1.	
	P4	PLB_PAValid	PLB Bus	I	See table note 1.	
	P5	PLB_SAValid (2)	PLB Bus	I	See table note 1.	
	P6	PLB_rdPrim (2)	PLB Bus	I	See table note 1.	
	P7	PLB_wrPrim (2)	PLB Bus	I	See table note 1.	
	P8	PLB_masterID(0:**G7**-1)	PLB Bus	I	See table note 1.	
	P9	PLB_abort	PLB Bus	I	See table note 1.	
	P10	PLB_busLock (2)	PLB Bus	I	See table note 1.	
	P11	PLB_RNW	PLB Bus	I	See table note 1.	
	P12	PLB_BE(0:[**G10**/8]-1)	PLB Bus	I	See table note 1.	
	P13	PLB_MSize(0:1)	PLB Bus	I	See table note 1.	
	P14	PLB_size(0:3)	PLB Bus	I	See table note 1.	
	P15	PLB_type(0:2)	PLB Bus	I	See table note 1.	
	P16	PLB_compress (2)	PLB Bus	I	See table note 1.	
	P17	PLB_guarded (2)	PLB Bus	I	See table note 1.	
	P18	PLB_ordered (2)	PLB Bus	I	See table note 1.	
	P19	PLB_lockerr (2)	PLB Bus	I	See table note 1.	
	P20	PLB_wrDBus(0:**G10**-1)	PLB Bus	I	See table note 1.	
	P21	PLB_wrBurst	PLB Bus	I	See table note 1.	
	P22	PLB_rdBurst	PLB Bus	I	See table note 1.	
	P23	PLB_pendReq (2)	PLB Bus	I	See table note 1.	
	P24	PLB_pendPri(0:1) (2)	PLB Bus	I	See table note 1.	
	P25	PLB_reqPri(0:1) (2)	PLB Bus	I	See table note 1.	

Table 2: **PLB RapidIO LVDS I/O Signals *(Continued)***

Grouping		Signal Name	Interface	I/O	Description	Page
PLB Bus Slave Reply Signals	P26	Sln_addrAck	PLB Bus	O	See table note 1.	
	P27	Sln_SSize(0:1)	PLB Bus	O	See table note 1.	
	P28	Sln_wait	PLB Bus	O	See table note 1.	
	P29	Sln_rearbitrate	PLB Bus	O	See table note 1.	
	P30	Sln_wrDack	PLB Bus	O	See table note 1.	
	P31	Sln_wrComp	PLB Bus	O	See table note 1.	
	P32	Sln_wrBTerm	PLB Bus	O	See table note 1.	
	P33	Sln_rdBus(0:**G10**-1)	PLB Bus	O	See table note 1.	
	P34	Sln_rdWdAddr(0:3)	PLB Bus	O	See table note 1.	
	P35	Sln_rdDAck	PLB Bus	O	See table note 1.	
	P36	Sln_rdComp	PLB Bus	O	See table note 1.	
	P37	Sln_rdBTerm	PLB Bus	O	See table note 1.	
	P38	Sln_MBusy(0:**G8**-1)	PLB Bus	O	See table note 1.	
	P39	Sln_MErr(0:**G8**-1)	PLB Bus	O	See table note 1.	
Device Interrupt Output	P40	Dev_Intr_Out	Interrupt Control	O	Device interrupt output. Registered level type, active high.	
RapidIO PHY Tx Signals	P41	RIO_Tclk	RapidIO	O	RapidIO Transmit Clock (250MHz) (3)	
	P42	RIO_Td(0:7)	RapidIO	O	RapidIO Transmit Data Bus (3)	
	P43	RIO_Tframe	RapidIO	O	RapidIO Transmit Frame signal (3)	
RapidIO PHY Rx Signals	P44	RIO_Rclk	RapidIO	I	RapidIO Receive Clock (~250MHz) (3)	
	P45	RIO_Rd(0:7)	RapidIO	I	RapidIO Receive Data bus (3)	
	P46	RIO_Rframe	RapidIO	I	RapidIO Receive Frame signal (3)	

Table 2: **PLB RapidIO LVDS I/O Signals *(Continued)***

Grouping		Signal Name	Interface	I/O	Description	Page
RapidIO PHY Tx Clocking Inputs	P47	RIO_Sys_clk (4)	User Clock Source	I	System Clock (125MHz) used by the internal Tx Clock generation logic (when it is included).	
	P48	TXDCM_250 (5)	User Tx clock input	I	RapidIO PHY Tx clock source (250 MHz).	
	P49	TXDCM_250i (5)	User Tx clock input	I	RapidIO PHY Tx clock source (250 MHz) that is 180º phase shifted relative to TXDCM_250.	
	P50	TXDCM_62 (5)	User Tx clock input	I	RapidIO PHY Tx clock source (62 MHz).	
	P51	TXDCM_locked (5)	User Tx DCM	I	RapidIO PHY Tx clock source status intended to be 'locked' signal from source DCM.	

Notes:
1. This signal's function and timing is defined in the IBM® **64-Bit Processor Local Bus Architecture Specification Version 3.x.**
2. This PLB input signal is not used by the PLB RapidIO LVDS design but is reserved for future support.
3. This signal's function and timing is defined in the Xilinx LogiCORE™ **RapidIO 8-bit Port Physical Layer Interface Product Specification**.
4. This input port is used only when parameter **G23** is set to 1.
5. This input port is used only when parameter **G23** is set to 0.

Parameter - Port Dependencies

The PLB RapidIO LVDS parameterization has effects on some I/O port sizes and other parameter values. These dependencies are summarized in Table 3.

Table 3: **Parameter/Port Dependencies**

		Name	Affects	Depends	Relationship Description
Design Parameters	**G7**	C_PLB_MID_WIDTH	**P8**	**G8**	**PLB_masterID** Bus Width is set by this parameter. It is dependant on **G8** which specifies the number of PLB masters connected to the PLB.
	G8	C_PLB_NUM_MASTERS	**P38**, **P39**, **G7**	-	This parameter specifies the number of PLB masters connected to the PLB. This intern sets the size of the **SIn_MErr** and **SIn_Busy** response buses and the width of the **PLB_masterID** bus (via **G7)**.
	G9	C_PLB_AWIDTH	**P3**	-	Specifies the width of the **PLB_ABus** bus.
	G10	C_PLB_DWIDTH	**P12**, **P20**, **P33**	-	Specifies the width of the **PLB_wrDBus**, **PLB_rdDBus**, and indirectly the number of byte enables required. The number of byte enables sets the size of the **PLB_BE** bus.
	G23	C_INCLUDE_TX_DCM	**P48, P49, P50, P51**	-	Ports **P48-P51** are used only when parameter **G23** is set to 0.
	G23	C_INCLUDE_TX_DCM	**P47**		Port **P47** (RIO_Sys_clk) is only used when parameter **G23** is set to 1.

PLB RapidIO LVDS Register and Memory Descriptions

The PLB RapidIO LVDS design contains memory mapped address spaces that are needed for processor interaction with the module. These memory spaces are summarized in Table 4 and are then described in more detail in the text following the table. Note that each address space is relative to the input parameter **G1** (C_BASEADDR).

Table 4: **PLB RapidIO LVDS Resource Address Assignments**

Functional Group	Register Name	PLB Address	Access	Data Width (bits)
IPIF Support	Device Global Interrupt Enable Register (1)	C_REG_FIFO_BASEADDR + $001C_{hex}$	Read/Write	32
	Interrupt Status Register (1)	C_REG_FIFO_BASEADDR + 0020_{hex}	Read/TOW	32
	Interrupt Enable Register (1)	C_REG_FIFO_BASEADDR + 0028_{hex}	Read/Write	32
	Device MIR / Reset Register (1)	C_REG_FIFO_BASEADDR + 0040_{hex}	Read/Write	32

Table 4: **PLB RapidIO LVDS Resource Address Assignments *(Continued)***

Functional Group	Register Name	PLB Address	Access	Data Width (bits)
Transmit and Receive Control	RapidIO Link Status Register (2)	C_REG_FIFO_BASEADDR + 0050_{hex}	Read	8
	Tx Length FIFO (2)	C_REG_FIFO_BASEADDR + 0100_{hex}	Write	32
	Tx Status FIFO (2)	C_REG_FIFO_BASEADDR + 0120_{hex}	Read	32
	Rx Length FIFO (2)	C_REG_FIFO_BASEADDR + 0200_{hex}	Read	32
	Rx Status Register(2)	C_REG_FIFO_BASEADDR + 0220_{hex}	Read	32
PHY Configuration	RapidIO PHY Management Interface Registers (3)	C_REG_FIFO_BASEADDR + 1000_{hex} to C_REG_FIFO_BASEADDR + $13FF_{hex}$	Read/Write	32
Transmit Packet Data Buffer (2) (512x64)	Bin # 0 (63.5 dblwords)	C_PBUF_BASEADDR + 0000_{hex} to C_PBUF_BASEADDR + $01F7_{hex}$	Read/Write	64
	Bin #0 TLC Reserved	C_PBUF_BASEADDR + $01F8_{hex}$ to C_PBUF_BASEADDR + $01FF_{hex}$	Read/Write (restricted)	32
	Bin # 1 (63.5 dblwords)	C_PBUF_BASEADDR + 0200_{hex} to C_PBUF_BASEADDR + $03F7_{hex}$	Read/Write	64
	Bin #1 TLC Reserved	C_PBUF_BASEADDR + $03F8_{hex}$ to C_PBUF_BASEADDR + $03FF_{hex}$	Read/Write (restricted)	32
	Bin # 2 (63.5 dblwords)	C_PBUF_BASEADDR + 0400_{hex} to C_PBUF_BASEADDR + $05F7_{hex}$	Read/Write	64
	Bin #2 TLC Reserved	C_PBUF_BASEADDR + $05F8_{hex}$ to C_PBUF_BASEADDR + $05FF_{hex}$	Read/Write (restricted)	32
	Bin # 3 (63.5 dblwords)	C_PBUF_BASEADDR + 0600_{hex} to C_PBUF_BASEADDR + $07F7_{hex}$	Read/Write	64
	Bin #3 TLC Reserved	C_PBUF_BASEADDR + $07F8_{hex}$ to C_PBUF_BASEADDR + $07FF_{hex}$	Read/Write (restricted)	32

Table 4: **PLB RapidIO LVDS Resource Address Assignments *(Continued)***

Functional Group	Register Name	PLB Address	Access	Data Width (bits)
Transmit Packet Data Buffer (2) (512x64)	Bin # 4 (63.5 dblwords)	C_PBUF_BASEADDR + 0800_{hex} to C_PBUF_BASEADDR + $09F7_{hex}$	Read/Write	64
	Bin #4 TLC Reserved	C_PBUF_BASEADDR + $09F8_{hex}$ to C_PBUF_BASEADDR + $09FF_{hex}$	Read/Write (restricted)	32
	Bin # 5 (63.5 dblwords)	C_PBUF_BASEADDR + $0A00_{hex}$ to C_PBUF_BASEADDR + $0BF7_{hex}$	Read/Write	64
	Bin #5 TLC Reserved	C_PBUF_BASEADDR + $0BF8_{hex}$ to C_PBUF_BASEADDR + $0BFF_{hex}$	Read/Write (restricted)	32
	Bin # 6 (63.5 dblwords)	C_PBUF_BASEADDR + $0C00_{hex}$ to C_PBUF_BASEADDR + $0DF7_{hex}$	Read/Write	64
	Bin #6 TLC Reserved	C_PBUF_BASEADDR + $0DF8_{hex}$ to C_PBUF_BASEADDR + $0DFF_{hex}$	Read/Write (restricted)	32
	Bin # 7 (63.5 dblwords)	C_PBUF_BASEADDR + $0E00_{hex}$ to C_PBUF_BASEADDR + $0FF7_{hex}$	Read/Write	64
	Bin #7 TLC Reserved	C_PBUF_BASEADDR + $0FF8_{hex}$ to C_PBUF_BASEADDR + $0FFF_{hex}$	Read/Write (restricted)	32

Table 4: **PLB RapidIO LVDS Resource Address Assignments *(Continued)***

Functional Group	Register Name	PLB Address	Access	Data Width (bits)
Receive Packet Data Buffer (2) (512x64)	Bin # 0 (64 dblwords)	C_PBUF_BASEADDR + 1000_{hex} to C_PBUF_BASEADDR + $11FF_{hex}$	Read/Write	64
	Bin # 1 (64 dblwords)	C_PBUF_BASEADDR + 1200_{hex} to C_PBUF_BASEADDR + $13FF_{hex}$		
	Bin # 2 (64 dblwords)	C_PBUF_BASEADDR + 1400_{hex} to C_PBUF_BASEADDR + $15FF_{hex}$		
	Bin # 3 (64 dblwords)	C_PBUF_BASEADDR + 1600_{hex} to C_PBUF_BASEADDR + $17FF_{hex}$		
	Bin # 4 (64 dblwords)	C_PBUF_BASEADDR + 1800_{hex} to C_PBUF_BASEADDR + $19FF_{hex}$		
	Bin # 5 (64 dblwords)	C_PBUF_BASEADDR + $1A00_{hex}$ to C_PBUF_BASEADDR + $1BFF_{hex}$		
	Bin # 6 (64 dblwords)	C_PBUF_BASEADDR + $1C00_{hex}$ to C_PBUF_BASEADDR + $1DFF_{hex}$		
	Bin # 7 (64 dblwords)	C_PBUF_BASEADDR + $1E00_{hex}$ to C_PBUF_BASEADDR + $1FFF_{hex}$		

Notes:
1. These registers are part of the IPIF support logic.
2. These elements are part of the RapidIO Processor Buffer sub-module.
3. These elements are accessed via the Processor Buffer but are part of the RapidIO PHY module.

Device Global Interrupt Enable Register (GIE)

This 32 bit register contains 1 R/W bit that allows the user to enable or disable the driving of an internally generated interrupt onto the Dev_Intr_Out port (**P40**). The register's format is graphically shown in Figure 1 and detailed in Table 5.

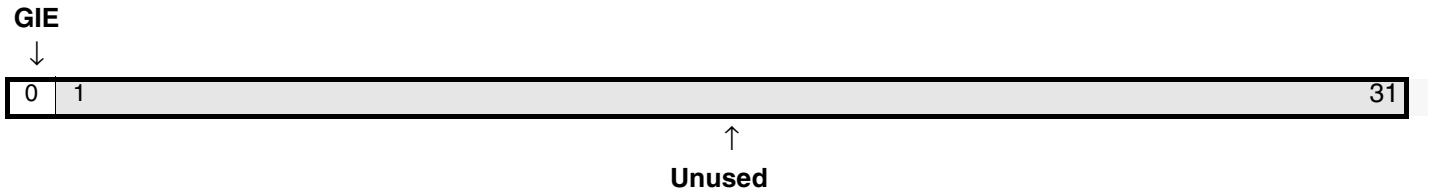

Figure 1: **Device Global Interrupt Enable Register Bit Layout**

Table 5: **Device Global Interrupt Enable Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0	**GIE**	Read/Write	'0'(1)	Global Interrupt Enable. • '0' - Disable Global Interrupt Output • '1' - Enable Global Interrupt Output
1 to 31	**Unused**	N/A	all zeroes	• Unused. • Reserved

Notes:
1. Interrupts are disabled after reset and initialization.

Interrupt Status Register (ISR)

This 32 bit register contains 7 R/TOW bits that capture events generated by the transmit and receive operations within the PLB RapidIO LVDS module. The register's format is graphically shown in Figure 2 and detailed in Table 6.

These interrupts are the source for the device interrupt driven onto the Dev_Intr_Out port (**P40**). Four of the interrupts are used for operational purposes. The remaining three are provided as error condition checking that can be used to support User Application development.

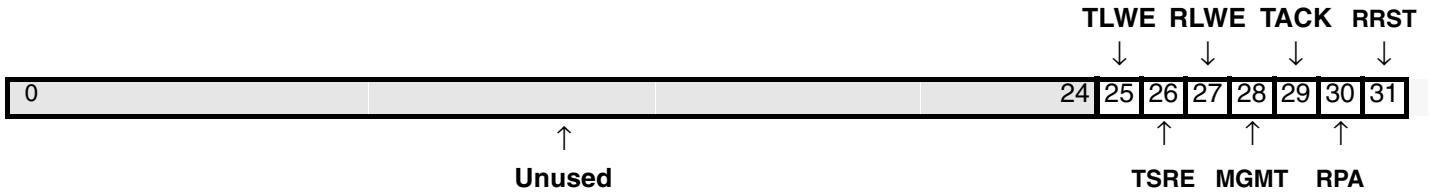

Figure 2: **Interrupt Status Register Bit Layout**

Table 6: **Interrupt Status Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 24	**Unused**	N/A	all zeroes	Unused. • Reserved
25	**TLWE**	R / TOW (2)	'0' (1)	Tx Length FIFO Write Error. • '1' = Error Detected. A write of the Tx Length FIFO (by the User Application) was attempted when the FIFO was 'Full'. • '0' = No Error
26	**TSRE**	R / TOW (2)	'0' (1)	Tx Status FIFO Read Error. • '1' = Error Detected. A read of the Tx Status FIFO (by the User Application) was attempted when the FIFO was 'EMPTY'. • '0' = No Error
27	**RLWE**	R / TOW (2)	'0' (1)	Rx Length FIFO Write Error. • '1' = Error Detected. The Rx Link Controller attempted to write to the Rx Length FIFO when it was 'FULL'. • '0' = No Error.
28	**MGMT**	R / TOW (2)	'0' (1)	PHY Management Interrupt. • '1' = Interrupt asserted. The PHY Management Interface is requesting service. • '0' = No interrupt.
29	**TACK**	R / TOW (2)	'0' (1)	Tx Acknowledge Interrupt. • '1' = Interrupt asserted. This interrupt is asserted whenever the Tx Status FIFO has at least one entry in it (it is 'not EMPTY'). The entry in the FIFO indicates the Address (Bin#) of a Tx Packet that has been successfully transmitted by the PHY. • '0' = No interrupt
30	**RPA**	R / TOW (2)	'0' (1)	Rx Packet Available Interrupt. • '1' = Interrupt asserted. This interrupt is asserted whenever the Rx Length FIFO has at least one entry in it (it is 'not EMPTY'). This means that at least one Rx packet has been received and is loaded in the Rx Packet Buffer. • '0' = No interrupt
31	**RRST**	R / TOW (2)	'0' (1)	Rx Reset Interrupt. • '1' = Interrupt asserted. A Rx Link Reset request from the PHY has been detected. • '0' = No interrupt.

Notes:

1. Interrupts are cleared after reset and initialization.
2. TOW refers to a Toggle On Write function where a write of a '1' to a register bit position will cause the corresponding register bit to toggle state. This allows the interrupt handling S/W to clear (or set for testing) an interrupt bit without interfering with the remaining interrupts in the register (by writing zeros to those bit positions).

Interrupt Enable Register (IER)

This 32 bit register contains 7 R/W bits that allows the user to enable or disable an individual interrupt contributing to the Dev_Intr_Out signal. These enable bits have a one-to-one correspondence to the bits in the Interrupt Status Register. The register's format is graphically shown in Figure 3 and detailed in Table 7.

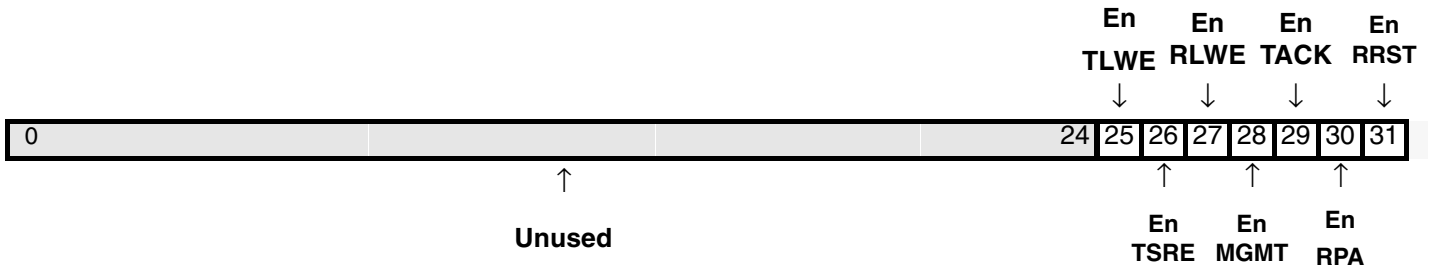

Figure 3: **Interrupt Enable Register Bit Layout**

Table 7: **Interrupt Enable Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 24	**Unused**	N/A	all zeroes	Unused. • Reserved
25	**EnTLWE**	R/W	'0'	Enable Tx Length Write Error. • '1' = Enabled • '0' = Disabled
26	**EnTSRE**	R / W	'0'(1)	Enable Tx Status FIFO Read Error. • '1' = Enabled • '0' = Disabled
27	**EnRLWE**	R / W	'0'(1)	Enable Rx Length FIFO Write Error. • '1' = Enabled • '0' = Disabled
28	**EnMGMT**	R / W	'0'(1)	Enable PHY Management Interrupt. • '1' = Enabled • '0' = Disabled
29	**EnTACK**	R / W	'0'(1)	Enable Tx Acknowledge Interrupt. • '1' = Enabled • '0' = Disabled
30	**EnRPA**	R / W	'0'(1)	Enable Rx Packet Available Interrupt. • '1' = Enabled • '0' = Disabled
31	**EnRRST**	R / W	'0'(1)	Enable Rx Reset Interrupt. • '1' = Enabled • '0' = Disabled

Notes:
1. Interrupt enables are cleared (interrupts disabled) after reset and initialization.

Module Information Register (MIR) and Reset Port

This 32 bit register provides a dual purpose. A read operation returns a 32 bit value known as the Module Information Register (MIR), In this design application, the MIR is primarily version information (hard coded value) about the imbedded IPIF but it also includes an echo of the C_DEV_BLOCK_ID parameter (**G5**). The register's format is graphically shown in Figure 4 and detailed in Table 8.

Since the idea behind the device block ID is that it is unique for each device in a processor controlled sub-system, it is very useful in verifying that communication uniquely exists between the PLB RapidIO LVDS and the User Application during system integration.

The second function of this register is to provide a means for a User Application to independently reset the PLB Rapidio LVDS module. The reset is triggered by a write to this register address with a data key value of **0000_000A$_{hex}$** . The reset forces re-initialization of the module including the RapidIO PHY. The write has no effect on the register's contents.

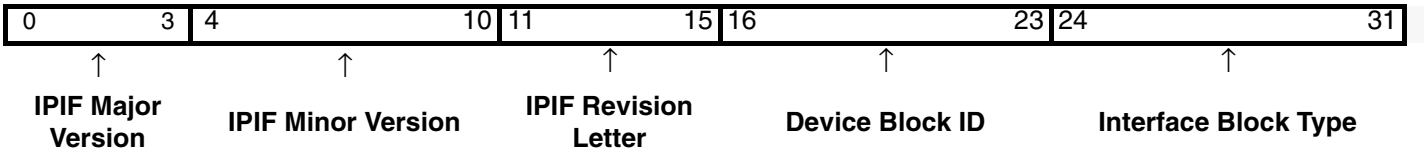

Figure 4: **MIR / Reset Register Bit Layout (read value only)**

Table 8: **MIR / Reset Register Bit Definitions (read value only)**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 3	**IPIF Major Version**	R	1 (1)	IPIF Major Version. This is a binary encoded number. • 1 = major version 1 • 2 = major version 2 • and so on.
4 to 10	**IPIF Minor Version**	R	00 (1)	IPIF Minor Version. This is a binary encoded number. • 00 = minor version 0 • 01 = minor version 1 • and so on.
25	**IPIF Revision Letter**	R	04 (1)	IPIF Revision Letter. This is a binary coded letter. • 00 = a • 01 = b • 02 = c • and so on.

Table 8: **MIR / Reset Register Bit Definitions *(Continued)*(read value only) *(Continued)***

Bit(s)	Name	Core Access	Reset Value	Description
26	**Device Block ID**	R	Value of parameter **G5**	Device Block ID Echo. • This is an echo of the **G5** parameter value.
27	**Interface Block Type**	R	08	Interface Block Type. • 01 = OPB IPIF • 08 = PLB IPIF

Notes:
1. This value shown corresponds to PLB IPIF version V1.00e

Link Status Register

This 8 bit register provides a general status of the PHY Tx and Rx Link port interface with the internal Processor Buffer Module. The User Application should monitor this register's value to determine if the PHY is operational. In addition, the Tx Packet Buffer Bin # being accessed by the PHY is echoed in this register. This is an aid to system integration of the User Application. The register's format is graphically shown in Figure 5 and detailed in Table 9.

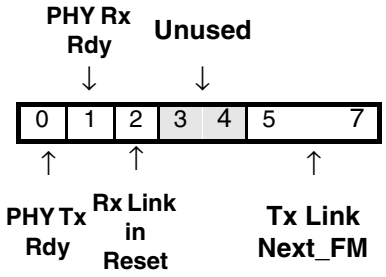

Figure 5: **Link Status Register Bit Layout**

Table 9: **Link Status Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0	**PHY Tx Rdy**	R	0 (1)	PHY Tx Ready. • '0' = PHY Tx Link Interface is not Ready • '1' = PHY Tx Link Interface is Ready
1	**PHY Rx Rdy**	R	0 (1)	PHY Rx Ready. • '0' = PHY Rx Link Interface is not Ready • '1' = PHY Rx Link Interface is Ready
2	**Rx Link in Reset**	R	Based on RapidIO Bus State	Rx Link in Reset. • '0' = Normal Operation • '1' = PHY is detecting a RapidIO bus reset request

Table 9: **Link Status Register Bit Definitions *(Continued)***

Bit(s)	Name	Core Access	Reset Value	Description
3 to 4	**Unused**	R	"00"	Unused. • Reserved bits
5 to 7	**Tx Link Next_FM**	R	"000"	Tx Link Next_FM. This bit field indicates the Tx Packet Bin # currently being addressed by the PHY Tx Link Port. • "000" = Tx Pkt Buffer Bin 0 • "001" = Tx Pkt Buffer Bin 1 • ... • "111" = Tx Pkt Buffer Bin 7

Notes:
1. These values are initially zeroed but get set to '1' when the RapidIO PHY becomes operational.

Tx Length FIFO

This 15wrd x 16bit Asynchronous FIFO (appears to the User Application as 32 bits wide) provides the mechanism for the User Application to indicate to the Processor Buffer logic that an Tx packet is available in the Tx Packet Buffer and can be considered for transmission.

The data value written by the User Application (the Tx Packet Descriptor) specifies the Bin # of the packet in the Tx Packet Buffer and the number of bytes making up the packet The data format is graphically shown in Figure 6 and detailed in Table 10.

The presence of at least one packet descriptor entry in the Tx Length FIFO (its not EMPTY) will cause the Transmit Link Controller to unload the descriptor and put it in the Tx queue.

Note: This mechanism requires the User Application to write the packet data to the Tx Packet Buffer first and then write the Packet Descriptor to the Tx Length FIFO second.

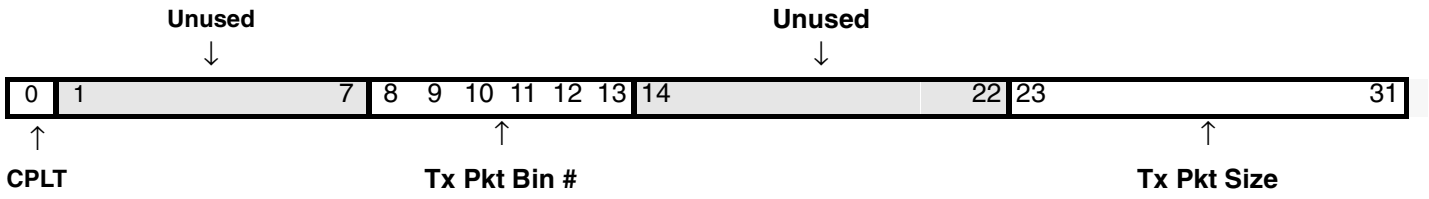

Figure 6: **Tx Length FIFO Data Format (write value only)**

Table 10: **Tx Length FIFO Data Bit Definitions (write only)**

Bit(s)	Name	Core Access	Reset Value	Description
0	**CPLT**	R	'0'	Packet Complete. This bit field indicates the Tx Packet data in the designated Pkt Bin # is a complete packet (not a partial packet). • The User Application always sets this to a '1' in the Tx Length value
1 to7	**Unused**	R	Zeros	Unused. • Reserved bits
8 to 13	**Tx Pkt Bin #**	R	Zeros	Tx Packet Bin Number. This 6 bit field specifies the holding bin number for the corresponding Tx Packet in the Tx Packet Buffer. • "000000" = Bin 0 • "000001" = Bin 1 • "000010" = Bin 2 • and so on (1)
14 to 22	**Unused**	R	Zeros	Unused. • Reserved bits
23 to 31	**Tx Pkt Size**	R	Zeros	Tx Packet Size. This 9 bit field specifies the number of valid bytes comprising the Tx Packet. Big endian byte ordering is used on last dblword which may contain fewer than 8 valid bytes.

Notes:
1. .Since the Tx Packet Buffer Memory currently has 8 bins, the values in this bit field can only be in the range 0 to 7.

Tx Status FIFO

This 32 bit read only register (actually a 15x8 FIFO) provides feedback to the User Application regarding Tx packets that have been successfully transmitted by the PHY. This asynchronous FIFO is loaded by the Processor Buffer Tx Link Controller (TLC) and is read by the User Application.

One or more entries in this FIFO will also cause the setting of the **TACK** interrupt bit in the Interrupt Status Register. The FIFO's format is graphically shown in Figure 7 and detailed in Table 11.

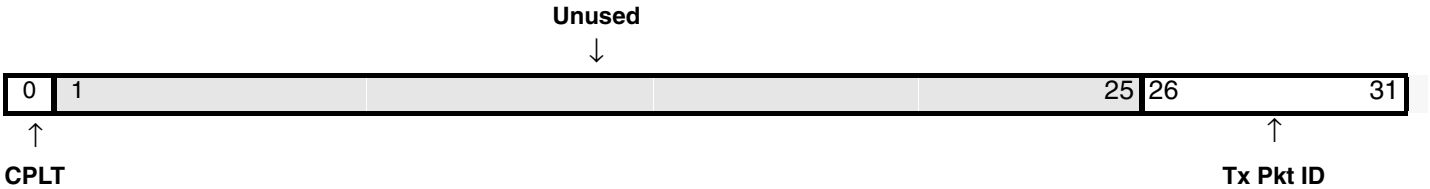

Figure 7: **Tx Status FIFO Data Format (Read value only)**

Table 11: **Tx Status FIFO Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0	**CPLT**	R	'0'	Tx Packet Complete. • Always set to '1' when FIFO has a valid entry (not empty)
1 to 25	**Unused**	R	Zeros	Unused • Reserved bits
26 to 31	**Tx Pkt ID**	R	Zeros	Tx Packet ID This bit field echoes the Packet ID value of the Tx packet successfully transmitted by the PHY. Currently, the Tx Pkt ID matches the Tx Pkt Bin # where the Tx Packet was loaded by the User Application. • "000000" = Pkt ID 0 • "000001" = Pkt ID 1 • "000010" = Pkt ID 2 • and so on (1)

Notes:
1. Since there are only 8 Tx bins in the Tx Packet Buffer memory, the ID values will only be in the range 0 to 7.

Rx Length FIFO

This 15wrd x 16bit Asynchronous FIFO (appears to the User Application as a 32 bit wide read only entity) provides the mechanism for the Processor Buffer logic to indicate to the User Application that an Rx packet is available in the Rx Packet Buffer and can be retrieved.

The data value written by the Processor Buffer logic (the packet descriptor) specifies the Bin # of the packet in the Rx Packet Buffer and the number of bytes comprising the packet. The register's format is graphically shown in Figure 8 and detailed in Table 12.

The presence of at least one packet descriptor entry in the Rx Length FIFO (its not EMPTY) will cause the **RPA** interrupt to be set in the Interrupt Status Register.

Note: This mechanism requires the User Application to read the packet descriptor out of the Rx Length FIFO first and then, using the descriptor information, read the packet data out of the Rx Packet Buffer.

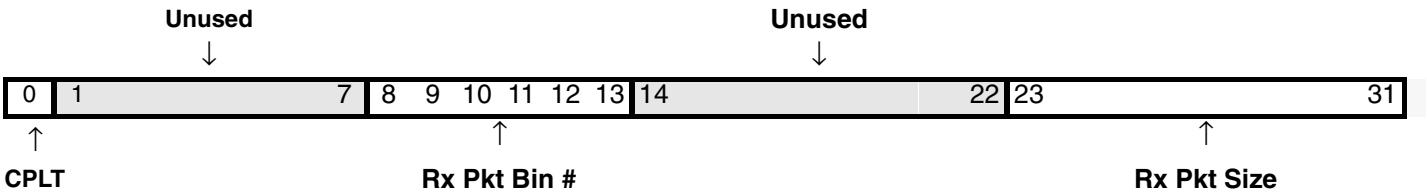

Figure 8: **Rx Length FIFO Data Format (read value only)**

Table 12: **Rx Length FIFO Data Bit Definitions (read only)**

Bit(s)	Name	Core Access	Reset Value	Description
0	**CPLT**	R	'0'	Packet Complete. This bit field indicates the Rx Packet data in the designated Pkt Bin # is a complete packet (not a partial packet). • Always = 1 when a valid entry is read from the Rx Length FIFO
1 to7	**Unused**	R	zeros	Unused. • Reserved bits
8 to 13	**Rx Pkt Bin #**	R	zeros	Rx Packet Bin Number. This 3 bit field specifies the holding bin number for the corresponding Rx Packet in the Rx Packet Buffer. • "000000" = Bin 0 • "000001" = Bin 1 • "000010" = Bin 2 • and so on (1)
14 to 22	**Unused**	R	zeros	Unused. • Reserved bits
23 to 31	**Rx Pkt Size**	R	zeros	Rx Packet Size. This 9 bit field specifies the number of valid bytes comprising the Rx Packet. Big endian byte ordering is used on last dblword which may contain fewer than 8 valid bytes.

Notes:
1. .Since the Rx Packet Buffer has only 8 bins, the value in this bit field will range 0 to 7.

Rx Status Port

This 32 bit port provides the mechanism by which the User Application notifies the Processor Buffer logic that the next sequential Rx Packet has been retrieved from the Rx Packet Buffer.

The User Application reads this port address. The read operation sends a signal to the Processor Buffer logic that causes the Rx Packet Buffer Bin holding the Rx Packet to be released from the 'filled' queue and added to the 'available' queue. The register's format is graphically shown in Figure 9 and detailed in Table 13.

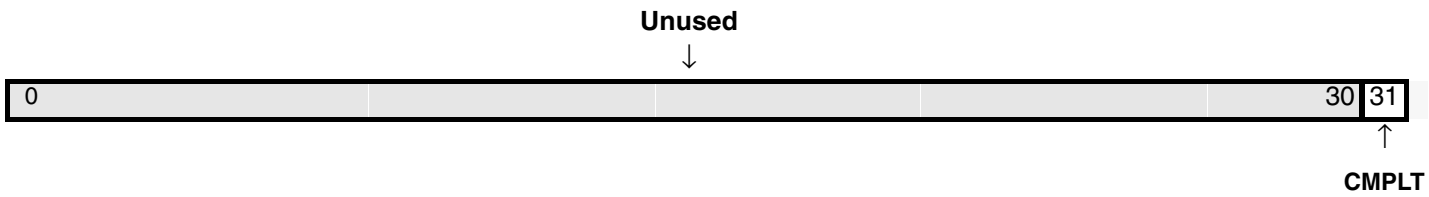

Figure 9: **Rx Status Data Format (read value only)**

Table 13: **Rx Acknowledge Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 30	**Unused**	R	zeros	Unused. • Reserved
31	**CMPLT**	R	’1’	Rx Packet Receive Complete • Always set to ’1’

Notes:
1.

RapidIO PHY Management Interface Registers

The RapidIO Physical Layer module incorporates the required Management and Configuration registers per the Xilinx ***RapidIO 8-bit Port Physical Layer Interface Product Specification***. These register are 32 bits wide and are summarized below in Table 14. Please refer to the ***RapidIO Physical Layer 8/16 LP-LVDS V1.1*** specification for the individual bit definitions.

Table 14: **RapidIO PHY Configuration Space Registers**

Port	Register Name (word 0)	Register Name (word 1)	Address
General	8/16 LP-LVDS Port Maintenance Block Header		C_REG_FIFO_BASEADDR + 1000h
	Reserved		C_REG_FIFO_BASEADDR + 1008h to C_REG_FIFO_BASEADDR + 1018h
	Port Link Time-Out Control CSR	Port Response Time-Out Control CSR	C_REG_FIFO_BASEADDR + 1020h
	Reserved		C_REG_FIFO_BASEADDR + 1028h
	Reserved		C_REG_FIFO_BASEADDR + 1030h
	Reserved	Port General Control CSR	C_REG_FIFO_BASEADDR + 1038h
Port 0	Reserved		C_REG_FIFO_BASEADDR + 1040h
	Reserved		C_REG_FIFO_BASEADDR + 1048h
	Reserved		C_REG_FIFO_BASEADDR + 1050h
	Port 0 Error and Status CSR	Port 0 Control CSR	C_REG_FIFO_BASEADDR + 1058h

Table 14: **RapidIO PHY Configuration Space Registers *(Continued)***

Port	Register Name (word 0)	Register Name (word 1)	Address
Port 1 (1)	Reserved		C_REG_FIFO_BASEADDR + 1060h
	Reserved		C_REG_FIFO_BASEADDR + 1068h
	Reserved		C_REG_FIFO_BASEADDR + 1070h
	Port 1 Error and Status CSR	Port 1 Control CSR	C_REG_FIFO_BASEADDR + 1078h
Port 2 - 14 (1)	Assigned to Port 2 - 14 CSRs		C_REG_FIFO_BASEADDR + 1080h to C_REG_FIFO_BASEADDR + 1218h
Port 15 (1)	Reserved		C_REG_FIFO_BASEADDR + 1220h
	Reserved		C_REG_FIFO_BASEADDR + 1228h
	Reserved		C_REG_FIFO_BASEADDR + 1230h
	Port 15 Error and Status CSR	Port 15 Control CSR	C_REG_FIFO_BASEADDR + 1238h

Notes:
1. Shaded ports are shown for completeness but are not supported at this time by the RapidIO PHY.

Transmit Packet Data Buffer Memory

RapidIO Transmit Information is stored in the Tx Packet Buffer Memory by the User Application. The buffer is a dual port memory that provides sumultaneous access to the User Application and the Tx Link Controller.

The Tx Packet Buffer memory is organized into 8 Bins that are each 64 entries deep by 64 bits wide. The memory's design allows the User Application to access each Bin entry as two independent word (32 bit) accesses or 1 dblword (64 bit) access. Byte and halfword accesses are not supported.

Each Tx Packet Bin bin is sized to hold the largest allowable Tx Packet size with some room left over. The Proc Buffer Tx Link Controller uses the last dblword in each Bin for local housekeeping. The User Application must not write to these reserved dblword locations.

Tx Packet Data

The Tx Packet Data must be formatted into 64 bit dblword entries as shown in Figure 10 and detailed Table 15. These entries are stored in the Tx Packet Buffer when the User Application is ready to have the Packet transmitted by the PHY to the RapidIO interface. The format of the Tx packet data is graphically summarized in the document ***LogiCORE™ RapidIO Physical Layer Interface Design Guide*** Appendix A.

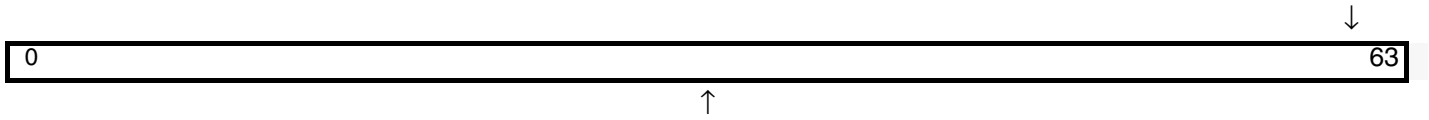

Figure 10: **Tx Packet Buffer Packet Data Dblword Bit Layout**

Table 15: **Tx Packet Buffer Data Dblword Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 63	**Tx Packet Buffer Data Dblword**	R/W (1) (2)	See Note 3	Tx Packet Buffer Data Dblword. • This is the 64 bit wide data that is used to form a Tx Packet in the Tx Packet Buffer memory. Each Packet Bin in the Tx Packet Buffer can store up to 63.5 dblword entries (127 words). The last word (32 bits) in each Bin is reserved for the Packet Descriptor information.

Notes:
1. In normal operation the User Application writes to the Tx Packet Buffer and the Processor Buffer Logic reads from it. The User Application may read from it for test and verification purposes.
2. Each element in the Tx Packet Buffer can be written to and read from as a single dblword (64bit) access or two word (32 bit) accesses. Byte and halfword accesses are not supported.
3. The Tx Packet Buffer is built from dual port memory primitives (BRAM). It's contents are undefined until the User Application writes data to it.

Receive Packet Data Buffer

The User Application reads received Packet Data from the Rx Packet Buffer. The data is formatted by the Rx Link Controller into 64 bit dblword entries. This is shown in Figure 12 and detailed in Table 16. The data format for the RapidIO Rx packet data is made up of a header, Data Payload, followed by 2 or 4 bytes of CRC and/or padding.

The make up of the header is dependent on the ftype, see ***LogiCORE™ RapidIO Physical Layer Interface Design Guide*** Appendix A. The Physical Layer strips the first byte of the header, passing the remaining bytes of the header to the user. The Data Payload that follows, when considered with the complete header, is always 64-bit aligned. The remaining bytes can then be discarded by the user. Also any intermediate CRC and padding are also stripped by the Physical Layer.

For example if a packet of ftype 6 is recieved and Rx Length of 89 bytes is read from the RX Length FIFO, the user can then extract the valid fields as follows.

Based on an ftype of 6, the header is made up of 8 bytes. The first byte is used and stripped by the Physical Layer, leaving 7 bytes of header being passed to the user. Therefore, the remaining 89 bytes - 7 bytes = 82 bytes make up the Data Payload, CRC and possible padding. Since the Data Payload is always 64-bit aligned, it is evenly divisible by 8. To determine the number of Data Payload bytes, simply divide the remaining byte count of 82 by 8 which equals 10.25. The whole portion, 10, is the number of double words of Data Payload. The fraction of 0.25 is the number of double words that can be discarded at the end of the packet, i.e. 0.25 Double Words X 8 Bytes/Double Word = 2 Bytes. For this example there are 7 bytes of Header, 80 bytes of Data Payload, and 2 bytes on the end that can be discarded.

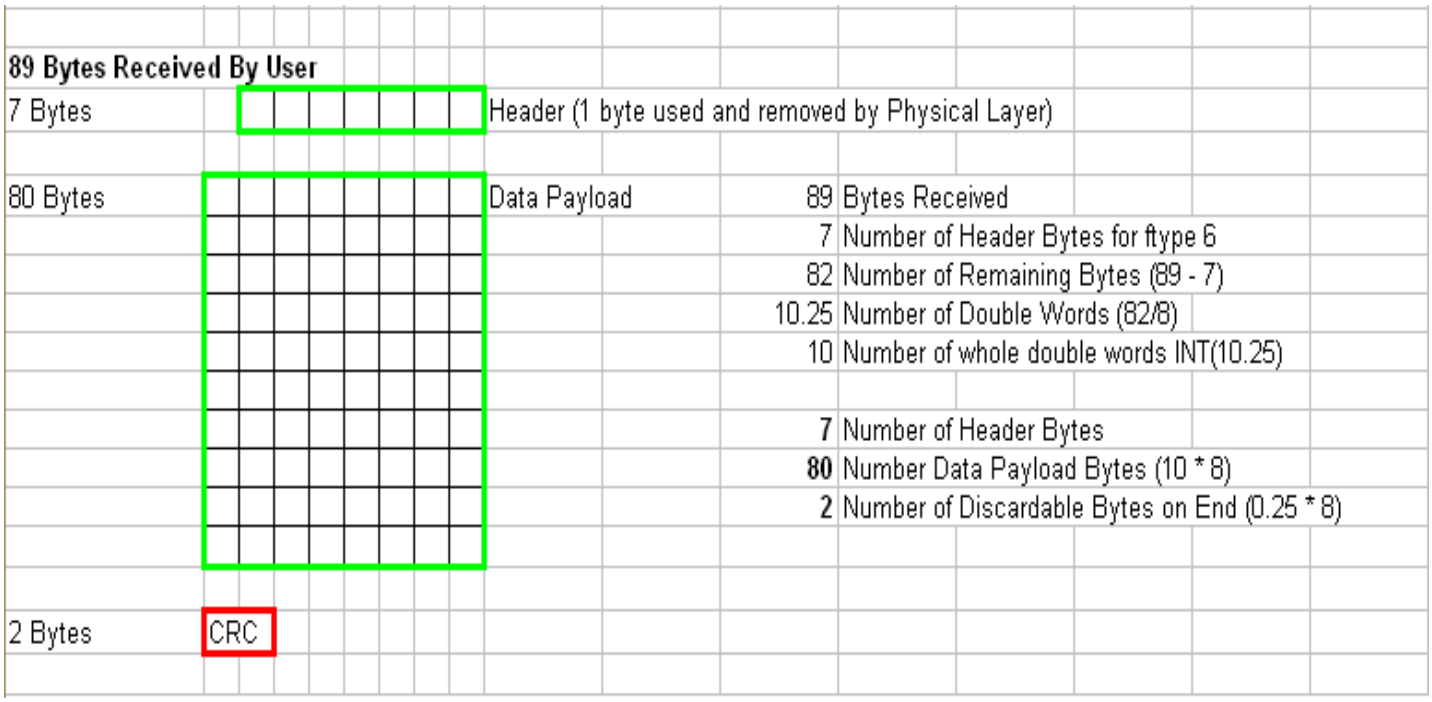

Figure 11: **Example RapidIO Rx Packet Data**

The data format of the RapidIO Rx packet data is graphically summarized in the document ***LogiCORE™ RapidIO Physical Layer Interface Design Guide*** Appendix A.

The Rx Buffer design allows the User Application to access each buffer word as two independent word (32 bit) accesses or one dblword (64 bit) access. Byte and halfword accesses are not supported. These entries are stored in the Rx Packet Buffer by the Rx Link Controller during packet receive operations.

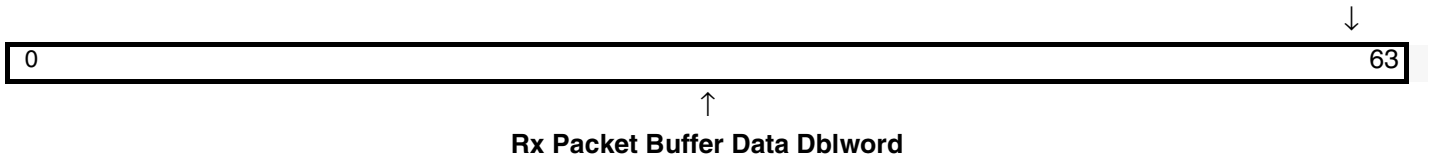

Figure 12: **Rx Packet Buffer Data Dblword Bit Layout**

Table 16: **Rx Packet Buffer Data Dblword Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to 63	**Rx Packet Buffer Data Dblword**	R/W (1) (2)	See Note **3**	Rx Packet Buffer Data Dblword. • This is the 64 bit wide data that is used to form a Rx Packet in the Rx Packet Buffer memory. Each Packet Bin in the Rx Packet Buffer can store up to 63.5 entries.

Notes:
1. In normal operation the User Application reads from the Rx Packet Buffer and the Processor Buffer Logic writes to it. The User Application may write to it for test and verification purposes.
2. Each element in the Rx Packet Buffer can be written to and read from as a single dblword (64bit) access or two word (32 bit) accesses. Byte and halfword accesses are not supported.
3. The Rx Packet Buffer is built from dual port memory primitives (BRAM). It's contents are undefined until the User Application or Processor Buffer logic writes data to it.

PLB RapidIO LVDS Interrupt Descriptions

Interrupts

The PLB RapidIO LVDS provides 4 operational interrupts and 2 integration support interrupts. These interrupts are accessed via the Interrupt Status Register (ISR) and Enabled/Disabled via the Interrupt Enable Register (IER). These registers and their contents are detailed in the register description section.

At the device I/O (Port **P40**), the PLB RapidIO LVDS outputs a single active high interrupt signal that can be used as an input to a System Interrupt Controller. The interrupt is a registered level interrupt. This means that the once the interrupt is set high, it cannot go low until the User Application clears the source of the interrupt in the ISR or disables it via the IER.

This device interrupt is generated internally by 'ORing" together the User Application enabled interrupts (via the IER) that are captured in the ISR. The User Application can enable/disable the single device interrupt via the Global Interrupt Enable Register (GIE). This is independent of any settings in the IER. In the disabled state, the device interrupt is driven low.

PLB RapidIO LVDS Block Diagram

The PLB RapidIO LVDS design includes the top level wrapper and three sub-blocks. The block diagram for the module is shown in Figure 13. The module interfaces to a processing element via the 64 bit IBM® PLB bus. The RapidIO Tx and Rx interfaces are intended for the 8 bit LVDS RapidIO signaling protocol. The plb_rapidio_lvds_v1_00_b adds the ability for the User to select between using external logic to provide the Tx clocking signals to the PHY or internal DCM/BUFG logic pre-configured for the PHY Tx requirements.

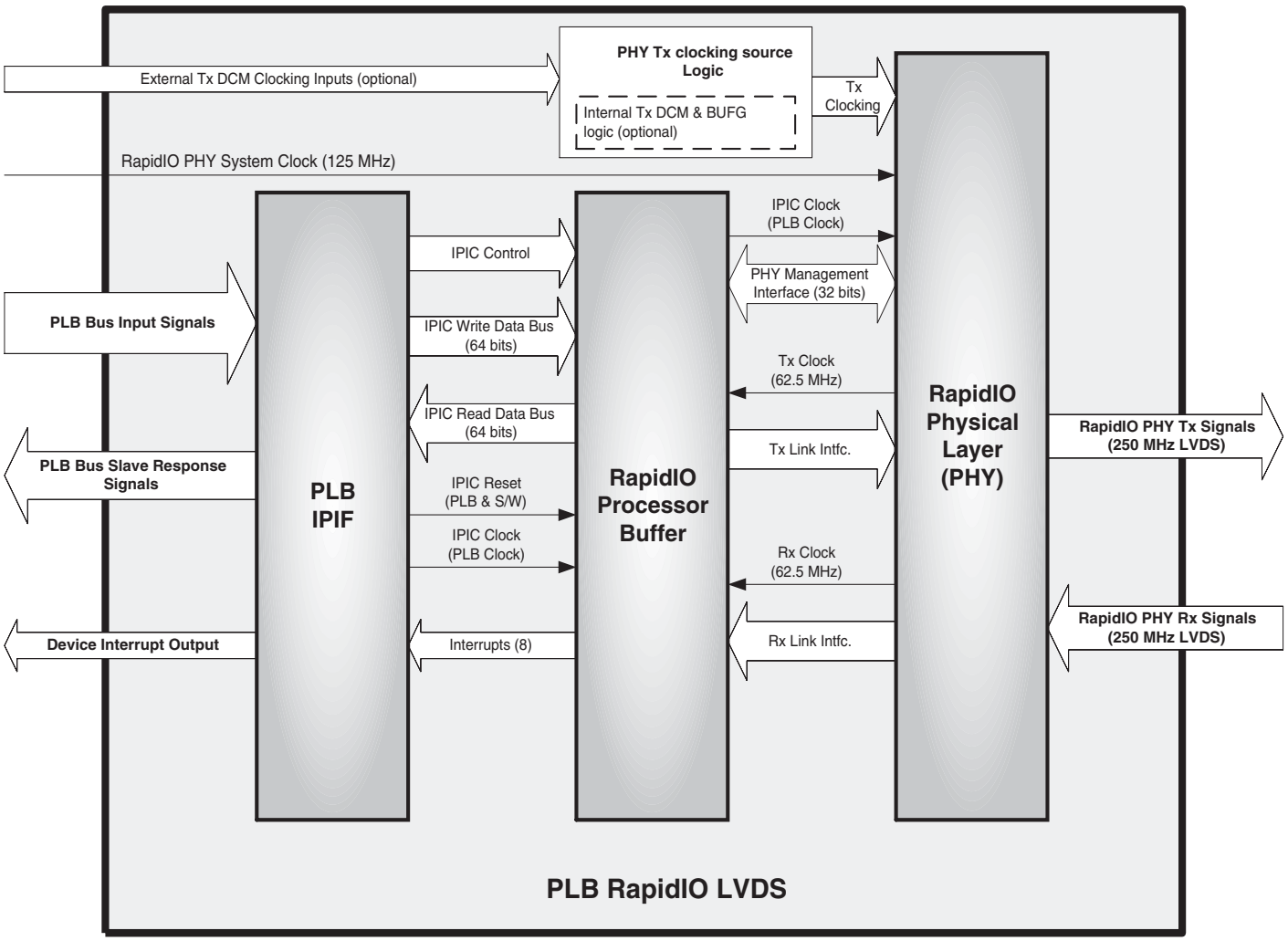

Figure 13: **PLB RapidIO LVDS Top-level Block Diagram**

PLB IPIF Block Diagram

The block diagram for the PLB IPIF is shown in Figure 14. It provides for the interface with the PLB Bus. The PLB IPIF is highly parameterized and therefore, the functions within the IPIF reflect the settings of the parameters for the PLB RapidIO LVDS requirements.

The PLB IPIF also provides for the IP Interface (IPIC) protocol which is used to communicate with the Processor Buffer block. This is a Xilinx internal standard interface for IP cores needing processing element interfaces. It provides a common signal set and function regardless of the processor host bus type.

The block diagram shown reflects the parameter settings for the RapidIO LVDS requirements. The function is comprised of 4 main blocks; the PLB Slave Attachment, the Byte Steering Module, the MIR/Reset Module, and the Interrupt Source Controller. Each of these is summarized in the following sections.

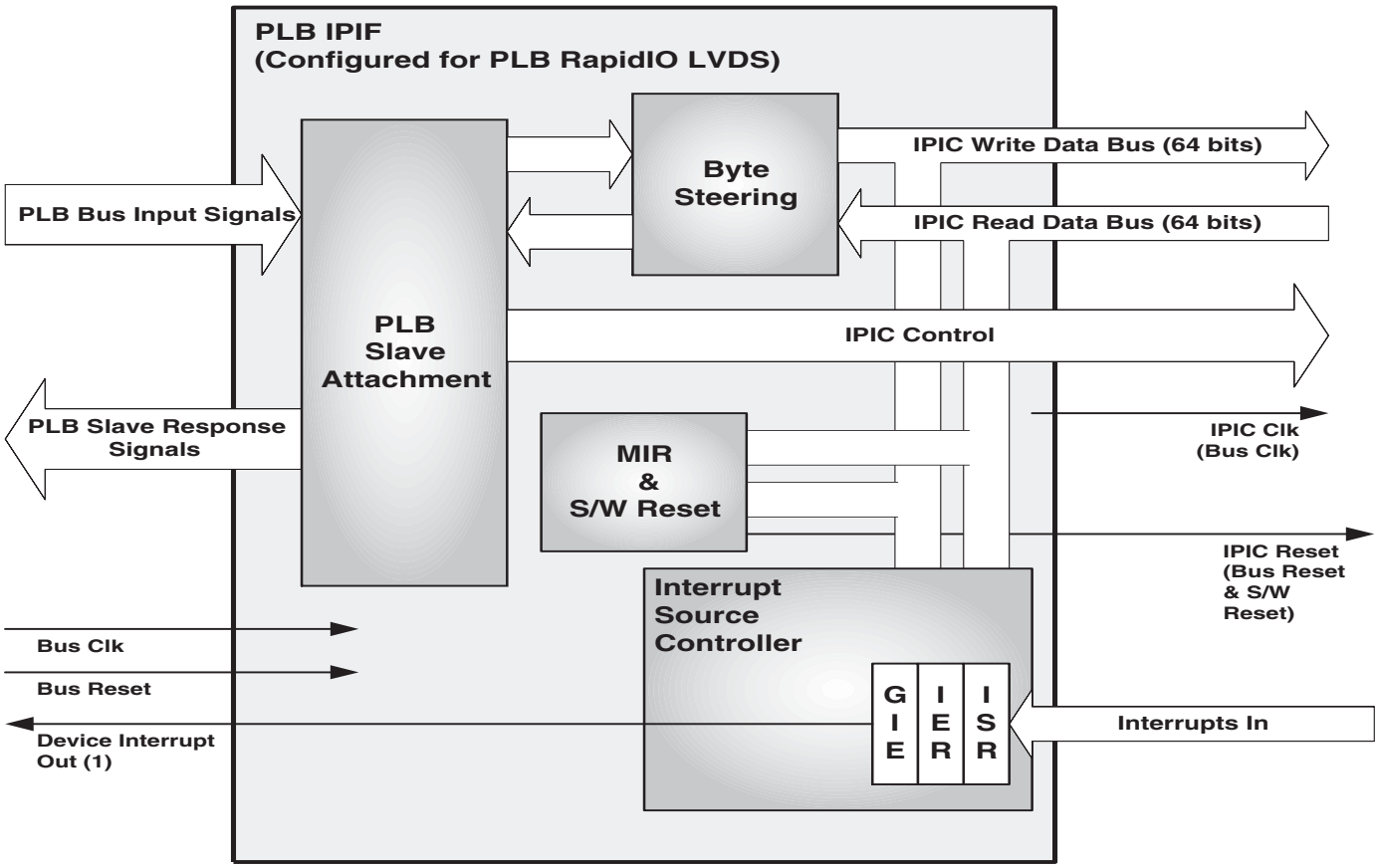

Figure 14: **PLB IPIF (RapidIO Parameterization) Logic Block**

PLB Slave Attachment

The PLB Slave Attachment is designed to perform the mechanization of the PLB Bus interface and the IPIC Interface. It interprets the PLB Input signals, stimulates the IPIC bus with the necessary controls and timing to access the target function, and then responds to the PLB Bus via the PLB Slave Response signals.

In order to perform these duties, the Slave Attachment design incorporates these functions:

- PLB transaction validation
- PLB transaction decode
- PLB address decoding
- PLB write data buffering (burst enabled)
- PLB write response and timing
- IPIC write control and timing
- PLB read response and timing
- IPIC read control and timing
- Data Phase Watch Dog Timeout
- Error handling
- PLB abort handling (can be User optioned in plb_rapidio_lvds_v1_00_b)

The PLB Slave Attachment is designed to conform to the IBM® ***64-Bit Processor Local Bus Architecture Specification 3.x***.

Byte Steering Module

The Byte Steering Module is used during read or write operations where the target function's data bus is not the same width as the PLB data bus (64 bits).

This module, coupled with target specific address decoding functions in the Slave Attachment, allows registers and memory widths from 8 bits to 64 bits (in byte increments) to be dynamically accessed according to the byte mirroring requirements of the PLB Bus.

The module performs the needed steering/mirroring for both Read and Write data buses as well as the byte lane enables (data transfer qualifies).

MIR & S/W Reset Module

This module incorporates the Module Information Register (MIR) and the Software initiated Reset Port. These functions are detailed in the section titled PLB RapidIO LVDS Register and Memory Descriptions.

Interrupt Source Controller

This module incorporates the Interrupt Status Register, the Interrupt Enable Register, and the Global Interrupt Enable Register. These registers are detailed in the section titled PLB RapidIO LVDS Register and Memory Descriptions.

RapidIO Processor Buffer

The Processor Buffer is a VHDL coded design that provides an interface between the Xilinx PLB IPIF and the Xilinx RapidIO Physical Layer module. Figure 15 shows a block diagram of the RapidIO Processor Buffer.

The module interfaces the Xilinx PLB IPIF and the Xilinx RapidIO Physical Layer Interface. The buffer has five major functions:

- Clock domain transformation
- Physical Layer Management Interface access
- Packet delimiting transformation
  - Tx Link Interface
  - Rx Link Interface
- Packet Data storage.
  - Tx Packet Buffer Memory
  - Rx Packet Buffer Memory
- Process Control
  - Link Status Register
  - Tx Status FIFO
  - Tx Length FIFO
  - Rx Length FIFO
  - Rx Status Register

These requirements are incorporated one way or another into the functions designed into the Processor Buffer. These functions are described in the following sections.

Clock Domain Transformation

The Processor Buffer provides the necessary clock domain transformation for three independently clocked interfaces.

The first clocked interface is the IPIF IP Interconnect (IPIC). This interface is connected to the IPIF and is synchronous to the PLB Bus Clock.

The second clocked interface is the PHY Transmit Interface. This interface is clocked at a 62.5MHz rate and the clock source is the PHY Tx Link Interface.

The third clocked interface is the PHY Receive interface. It is also operates at 62.5MHz and is sourced by the PHY Rx Link interface.

The PHY Management Interface is clocked using the PLB Bus clock.

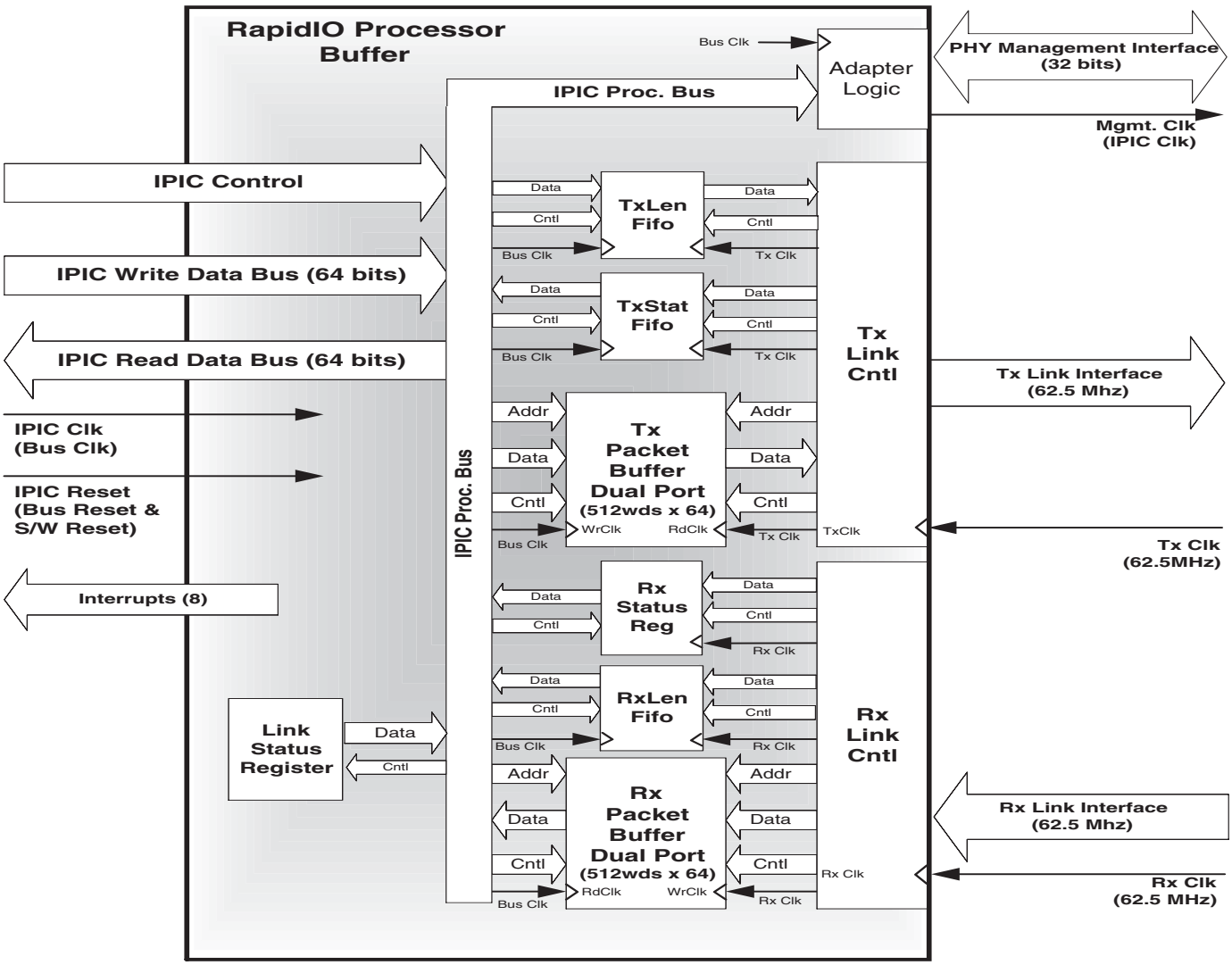

Figure 15: **Processor Buffer Logic Block**

Management Interface for Physical Layer

The Processor Buffer provides access to the PHY Management Interface via simple translation function to adapt the IPIC protocol and signal set to that needed by the PHY. The interface's data bus is 32 bits wide and uses a simple read/write protocol.

The PHY Management Interface is provided the Bus Clock for synchronization. Thus, the Processor Buffer does not have to provide clock domain transformation on this interface.

Tx Length FIFO

The Tx Length FIFO is used by the User Application to communicate to the Tx Link Controller that a Tx Packet is available in the Tx Packet Buffer and can be transmitted. The FIFO is a dual clock design and therefor provides clock domain transformation between the Bus Clock and the Tx Link Clock.

Each entry in the FIFO corresponds to a unique Tx Packet in the Tx Packet Buffer. Each entry includes the length of the packet (in bytes) and the Bin number of the Rx Packet Buffer used to hold the packet.

The data format is shown in the Register and Memory Description section of this document. Data is written to the FIFO by the User Application and read by the Tx Link Controller.

Tx Packet Buffer Memory

The Tx Packet Buffer is the temporary storage agent for the RapidIO Transmit Packet Data and Packet Descriptors. The buffer is composed of Xilinx Block RAM and is sized (512wds x 64bits) to contain eight (8) bins of 64 dblwords each. This is large enough to hold a maximum sized RapidIO packet within each bin.

The buffer is by nature a Dual Port Memory configuration that supports independent clocking of the A and B interface ports. Transmit packet data is written into the port A by the User Application via the IPIC interface. The packet data is then read from the buffer B port by the Tx Link Controller logic and sent to the PHY Tx Link Interface. For system debugging, the User Application can also read the packet data it writes into the buffer via the port A read interface

Tx Status FIFO

The Tx Status FIFO is the mechanism by which the Tx Link Controller can inform the User Application that a Tx Packet transmission has completed. When a Packet transmission completes, the Tx Link Controller writes the Ack ID of the Packet (essentially the buffer ID of the Packet Buffer where the packet was written by the User Application) into the FIFO.

The data format is shown in the Register and Memory Description section of this document. The presence of one or more entries in the Tx Status FIFO causes an interrupt to be asserted to the User Application. The interrupt is intended to prompt the User Application to read the Tx Status FIFO data (the packet ID) and use it to manage its Tx Packet completion queue.

Tx Link Controller

The actual writing of Tx packet data into the Tx Link Port of the RapidIO PHY is controlled by the Tx Link Controller (TLC) in the Processor Buffer. The TLC monitors changes in the Lnk_TNext_FM bus from the PHY. When the bus changes value, the PHY is indicating that it is ready to accept the next packet.

The TLC then searches the Tx Packet Buffer to select the next packet to be transmitted. The TLC extracts the packet tag, priority, and type information provided by the Packet Descriptor data value written into the last word of each packet bin containing valid packet data. This information is sorted by the TLC to select the best packet to be transmitted.

Once a packet is selected for transmission, the TLC extracts the Bin#, REM value, and word count. from the Packet Descriptor. Using this information, the TLC sequentially reads the packet words from the Tx Packet Buffer and writes it to the PHY Tx Link Port. The TLC writes the data using the Tx interface protocol as defined by the RapidIO Physical Layer documentation.

The PHY signals the completion of a transmission by changing the value of the LNK_TLAST_ACK bus. The TLC monitors this bus for changes. A change to a new value indicates that the packet transmitted when the LNK_TNEXT_FM signal matched the new value of the LNK_TLAST_ACK value has successfully competed.

The TLC then performs the housekeeping necessary to remove the packet from the pending completion list and write the completion status data to the Tx Status FIFO.

Link Status Register

A Link Status Register is provided for the User Application. The data format is shown in the Register and Memory Description section of this document. The register is Read Only and provides general status about the Tx and Rx Links between the RapidIO PHY and the Processor Buffer.

The Link Status Register should be used by the User Application for power up and reset initialization to determine when the Tx and Rx links are ready for operation or if the Rx port is reporting a reset condition.

The register also provides the state of the lnk_next_fm bus that is part of the Tx link signal set. These three bits indicate which packet buffer address the PHY is currently using for an active transmission or the next packet buffer to be used for the next transmission

Rx Packet Buffer Memory

The Rx Packet Buffer is the temporary storage agent for the RapidIO Receive Packet Data. The buffer is composed of Xilinx Block RAM and is sized (512wds x 64bits) to contain eight (8) bins of 64 dblwords each. This is large enough to hold a maximum sized RapidIO packet within each bin.

The data format is shown in the Register and Memory Description section of this document and is identical to the Tx Packet Buffer organization. The buffer is by nature a Dual Port Memory configuration that supports independent clocking of the A and B interface ports.

Receive packet data is received from the PHY Rx Link Interface by the Rx Link Controller and written into the Rx Packet Buffer port B. The packet data is then read from the Rx Packet Buffer A port by the User Application. For system debugging, the User Application can also write to the Rx Packet Buffer A port but does not do so in normal operation.

Rx Length FIFO

The Rx Length FIFO is used by the Rx Link Controller to communicate with the User Application. An entry in the FIFO signals the User Application (via interrupt) that a Rx Packet is available in the Rx Packet Buffer and can be read.

The FIFO is a dual clock design and therefor provides clock domain transformation between the processor bus clock and the PHY supplied Rx Link Clock. Each entry that is loaded into the FIFO (16 bits) corresponds to a unique Rx Packet in the Rx Packet Buffer.

Each entry includes the length of the packet (in bytes) and the Bin number in the Rx Packet Buffer used to hold the packet.

The data format is shown in the Register and Memory Description section of this document. This FIFO is read by the User Application and written by the Rx Link Controller.

Rx Status Register

The User Application acknowledges the receipt of an Rx Packet by reading the Rx Status Register. The data read by the User Application should be discarded. The operation of reading this register causes the Rx Link Controller to update its Rx Packet Buffer Bin# pointers and the buffer availability status to the PHY.

Rx Link Controller

The Processor Buffer manages the PHY Rx Link Interface via the Rx Link Controller (RLC). This function is comprised of state machines and logic. The RLC is responsible for receiving data from the PHY interface and writing it into the Rx Packet Buffer.

The RLC manages the Bins in the Rx Packet Buffer like a ring buffer. There is a bin head pointer and a bin tail pointer. The tail pointer indicates the packet bin# that is being processed (or is next in line to be processed) by the User Application.

Each received packet is stored in the next available sequential bin number (the bin head pointer). Up to eight packets can be stored in the Rx Packet Buffer. If the Rx Packet Buffer becomes full (the bin head pointer wraps around and bumps against the tail pointer), the RLC will no longer accept Rx data from the PHY.

The RLC also provides the PHY with a count value of bins that are available for writing. When the Rx Buffer is filled, this count value is zero and should keep the PHY from attempting to send packet data to the Rx Link Interface.

At the completion of a packet write into the Rx Packet Buffer, the RLC then writes the associated packet descriptor data into the Rx Length FIFO. This will cause an interrupt to the User Application. When the User Application retrieves the packet data, it must write to the Rx Acknowledge port. This causes the RLC to advance the ring buffer tail pointer by one bin number as well as increment the available bin count to the PHY by 1.

RapidIO Physical Layer Interface

The LogiCORE™ RapidIO Physical Layer interface performs several functions. It is partitioned into two modules, the OSI Physical Layer Module (OPLM) and the OSI Link Layer Module (OLLM). These modules are shown in Figure 16.

The OPLM is responsible for the serialization/de serialization function, transmit and internal clock generation, link initialization, and training.

The OLLM is responsible for CRC generation and verification, symbol generation and decoding, packet exchange protocol handshake, and buffer management.

A more detailed description of the Physical Layer interface is presented in the LogiCORE™ ***RapidIO 8-bit Port Physical Layer Interface*** Product Specification.

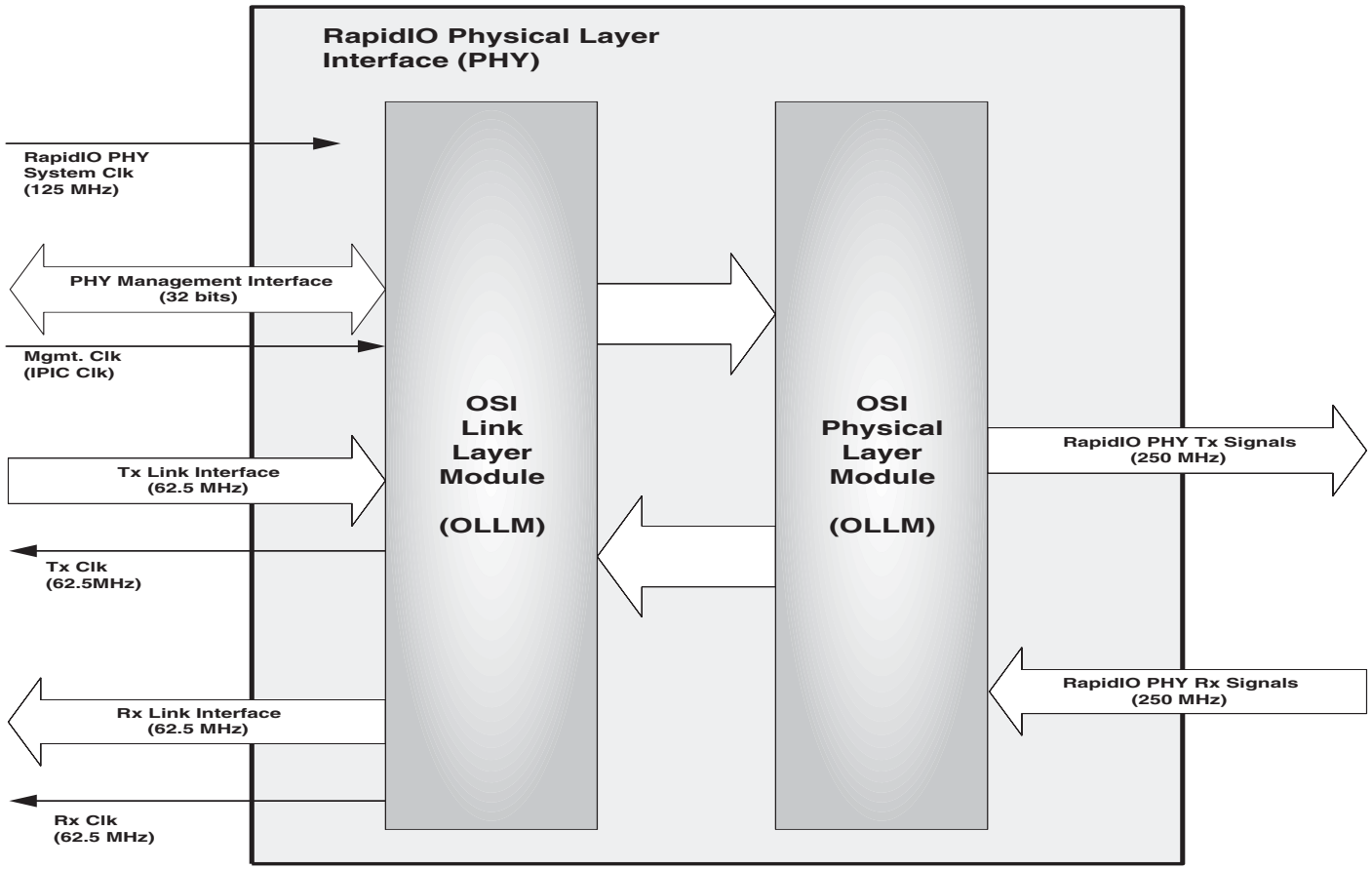

Figure 16: **RapidIO Physical Layer Interface Logic Block**

FPGA Design Application Hints

RapidIO System Clock

When enabled via parameter **G23** (= 1), the internal Tx Clocking logic requires 125MHz to be supplied on the input port RIO_Sys_clk (**P47**). If parameter **G23** is set to 0, then this input is not used and should be tied to logic low.

Other Clocking Information

The RapidIO PHY consumes some clocking resources that the user must account for in the overall FPGA resource planning effort. The PHY incorporates one Digital Clock Module (DCM) in its design. These are not accessible to the user and count towards the total DCM usage in the FPGA resources. The PHY also consumes four global clock buffers (GCLKs) .

If the Tx clocking is included internally via setting parameter **G23** equal to 1, the Tx clocking logic will consume 1 DCM and 3 global clock buffers (GCLKs).

Tx Clocking Design Example

The following VHDL snippet is a design example for generating the Tx clocking signals.

```
-------------------------------------------------------------
   -- If Generate
   --
   -- Label: INCLUDE_TX_DCM
   --
   -- If Generate Description:
   --   This IfGen implements the Tx clock DCM and associated
   -- BUFG's.
   --
   --
   -------------------------------------------------------------
   INCLUDE_TX_DCM : If (C_INCLUDE_TX_DCM = 1) generate

          -- Local DCM Signals
            Signal sig_clk250_fb  : std_logic;
            Signal sig_clk250_180 : std_logic;
            Signal sig_clk62      : std_logic;


            component DCM
              port (
                CLKIN      : in std_logic;
                CLKFB      : in std_logic;
                DSSEN      : in std_logic;
                PSINCDEC   : in std_logic;
                PSEN       : in std_logic;
                PSCLK      : in std_logic;
                RST        : in std_logic;
                CLK0       : out std_logic;
                CLK90      : out std_logic;
                CLK180     : out std_logic;
                CLK270     : out std_logic;
                CLK2X      : out std_logic;
                CLK2X180   : out std_logic;
                CLKDV      : out std_logic;
                CLKFX      : out std_logic;
                CLKFX180   : out std_logic;
                LOCKED     : out std_logic;
                PSDONE     : out std_logic;
                STATUS     : out std_logic_vector(7 downto 0)
                );
            end component DCM;

            component BUFG
              port (
                O : out std_logic;
                I : in std_logic
                );
            end component BUFG;


            ------------------------------------------------------------------------------
            -- DCM Attribute declarations
            ------------------------------------------------------------------------------

             attribute CLKIN_PERIOD          : string;
             attribute CLK_FEEDBACK          : string;
             attribute DLL_FREQUENCY_MODE    : string;
             attribute DUTY_CYCLE_CORRECTION : string;
             attribute STARTUP_WAIT          : string;
             attribute DFS_FREQUENCY_MODE    : string;
             attribute DSS_MODE              : string;
```

```
    attribute CLKDV_DIVIDE          : string;
    attribute CLKFX_DIVIDE          : string;
    attribute CLKFX_MULTIPLY        : string;
    attribute CLKOUT_PHASE_SHIFT    : string;
    attribute PHASE_SHIFT           : string;


    attribute CLKIN_PERIOD           of I_TX_DCM : label is "8.0 ns"; -- 125 MHz
    attribute CLK_FEEDBACK           of I_TX_DCM : label is "2X";
    attribute DLL_FREQUENCY_MODE     of I_TX_DCM : label is "LOW";
    attribute DUTY_CYCLE_CORRECTION  of I_TX_DCM : label is "TRUE";
    attribute STARTUP_WAIT           of I_TX_DCM : label is "FALSE";
    attribute DFS_FREQUENCY_MODE     of I_TX_DCM : label is "LOW";
    attribute DSS_MODE               of I_TX_DCM : label is "NONE";
    attribute CLKDV_DIVIDE           of I_TX_DCM : label is "2.0";
    attribute CLKFX_DIVIDE           of I_TX_DCM : label is "001";
    attribute CLKFX_MULTIPLY         of I_TX_DCM : label is "004";
    attribute CLKOUT_PHASE_SHIFT     of I_TX_DCM : label is "NONE";
    attribute PHASE_SHIFT            of I_TX_DCM : label is "000";


begin

  ------------------------------------------------------------------------
  -- Primary DCM for Tx clock frequency generation derived from the
  -- 125MHz System input clock
  ------------------------------------------------------------------------
  I_TX_DCM: DCM
     port map (
          CLKIN       => RIO_Sys_clk,      --.CLKIN(sys_clk),     [in] -- 125 MHz
          CLKFB       => sig_clk250_fb,    --.CLKFB(w_dc250),     [in]
          DSSEN       => '0',              --.DSSEN(1'b0),        [in]
          PSINCDEC    => '0',              --.PSINCDEC(1'b0),     [in]
          PSEN        => '0',              --.PSEN(1'b0),         [in]
          PSCLK       => '0',              --.PSCLK(1'b0),        [in]
          RST         => sig_bus2ip_reset, --.RST(~sys_reset_n),  [in]
          CLK0        => open,             --.CLK0(w_nocon_0),    [out]
          CLK90       => open,             --.CLK90(w_nocon_4),   [out]
          CLK180      => open,             --.CLK180(w_nocon_1),  [out]
          CLK270      => open,             --.CLK270(w_nocon_2),  [out]
          CLK2X       => sig_clk250_fb,    --.CLK2X(w_dc250),     [out]
          CLK2X180    => sig_clk250_180,   --.CLK2X180(w_dc250i), [out]
          CLKDV       => sig_clk62,        --.CLKDV(w_dc62),      [out]
          CLKFX       => open,             --.CLKFX(w_nocon_5),   [out]
          CLKFX180    => open,             --.CLKFX180(w_nocon_6),[out]
          LOCKED      => sig_tx_locked,    --.LOCKED(tx_locked),  [out]
          PSDONE      => open,             --.PSDONE(w_nocon_7))  [out]
          STATUS      => open              --.STATUS(w_pss[7:0]), [out]
    );

  ------------------------------------------------------------------------
  -- TX Clocks BufG instantiation
  ------------------------------------------------------------------------
    -- Global Buffer the 250 MHz clock to generate
    -- the Tx 250 Mhz clk
    I_CLK250_BUFG: BUFG
      port map (
        O => sig_tx_250,   --[out]
        I => sig_clk250_fb --[in]
      );
```

```
            -- Global Buffer the 62 MHz clock to generate
            -- the Tx 62 Mhz clk
            -- Verilog Source example
                -- BUFG  G62 (.I(w_dc62), .O(tx_62));
            I_CLK62_BUFG: BUFG
              port map (
                O => sig_tx_62,   --[out]
                I => sig_clk62    --[in]
              );

              -- Global Buffer the DCM 180 degree phase shifted 250 MHz clock to generate
              -- the 180 degree phase shifted Tx 250 Mhz clk
              I_CLK250I_BUFG: BUFG
                port map (
                  O => sig_tx_250i,      --[out]
                  I => sig_clk250_180    --[in]
                );

End generate INCLUDE_TX_DCM;
```

Reset Assertion

The PLB RapidIO LVDS module incorporates reset assertion logic that guarantees a minimum reset pulse to the RapidIO PHY and the Processor Buffer logic for 32 PLB Bus clock periods. This extended reset is activated by either a PLB Reset assertion or a commanded reset via a write to the Reset/MIR Register by the User Application.

LVDS Driver Instantiation

The PLB RapidIO LVDS design does not incorporate the I/O drivers for the RapidIO LVDS signals. Drivers are needed for signals output on Tx Ports **P41**, **P42**, and **P43** and receivers are needed for signals input to the Rx Ports **P44**, **P45**, and **P46**. This requires the user to include the appropriate LVDS I/O drivers and receivers in the top level FPGA design. This shown graphically in Figure 17.

There are three ways for the user to get the Xilinx toolset to incorporate LVDS I/O Buffers. The first is direct instantiation of the Xilinx buffer primitives in the user's HDL source for the FPGA top level.

The second method is by inference through UCF assignments on the RapidIO signal port names used at the FPGA top level design. An LVDS I/O Buffer standard would be assigned to the signal names.

The third way (also by inference) requires the user to assign an I/O standard attribute to the RapidIO I/O signal nets declared within the HDL of the top level FPGA design.

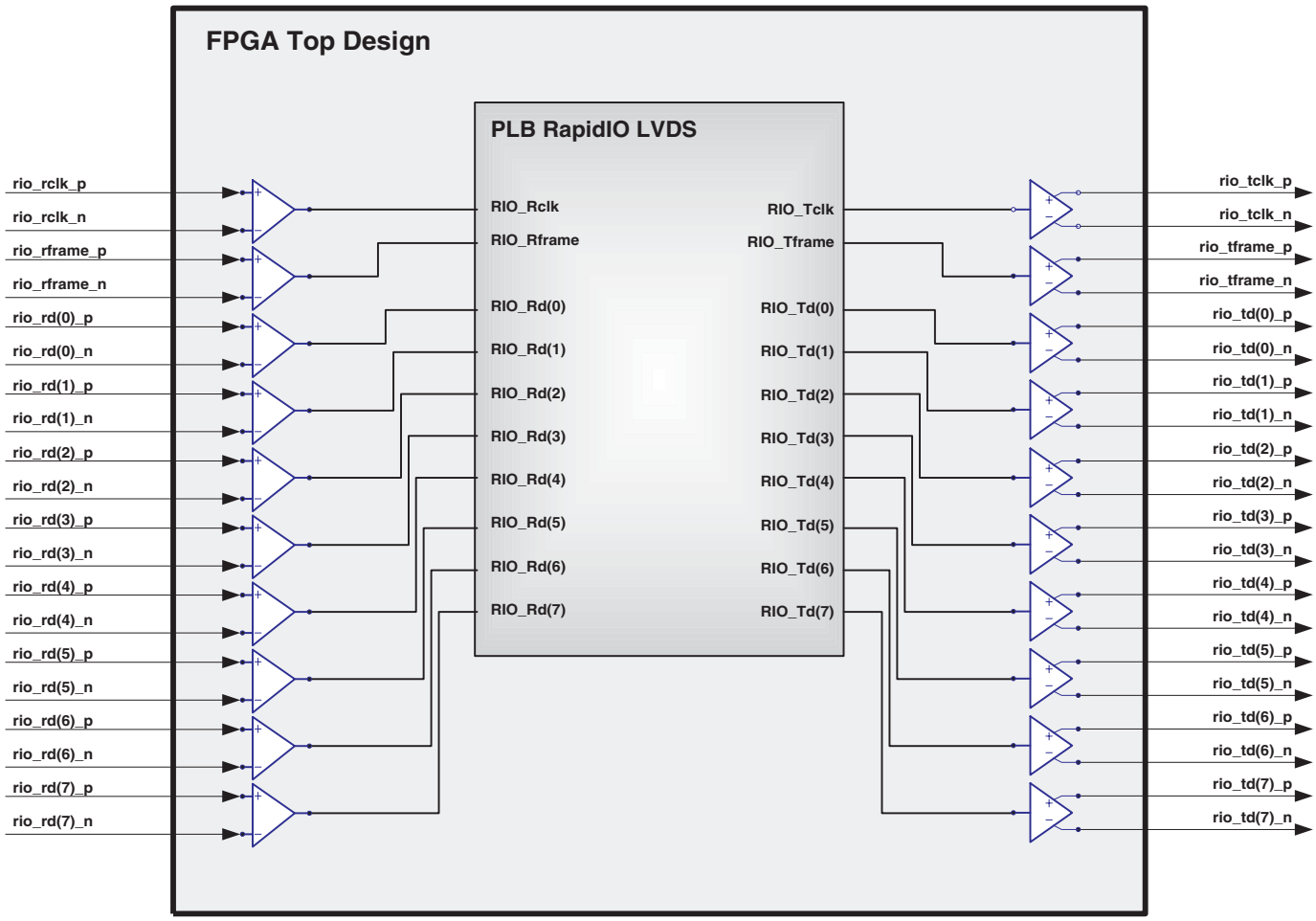

Figure 17: **LVDS Buffer Instantiation at FPGA Top Level**

User Application Hints

General Hardware Initialization

The PLB RapidIO LVDS has some minimal initialization steps that need to be performed by the User Application. These are listed below:

1. Issue a software reset command to the Device by writing the data word 0x0000000A to the MIR/Reset Register address. The commanded reset causes the reset signal to the Processor Buffer and the RapidIO PHY to be asserted for 32 PLB Bus clock periods. This assures a positive reset regardless of the frequency relationships between the PLB, System, Tx, and Rx clocks

2. Set the Device global interrupt enable by writing data word 0x80000000 to the Device Global Interrupt Enable register address.

3. Enable the operational interrupts by writing 0x000F to the Interrupt Enable Register. (0x007F may optionally be written to include the device's error/debug interrupts).

4. Monitor the Link Status Register. The User Application should not attempt any operations until the Tx Link and Rx Link Ready bits are set. This indicates that PHY has completed clock acquisition and initialization and the Tx and Rx Link clocks are stable.

5. If necessary, query or configure the PHY Management Interface registers per the ***RapidIO 8-bit Port Physical Layer Interface*** Product Specification. This is done via the Management Interface Register memory mapped ports of this core.

General Tx Packet Processing Flow

The User Application is responsible for the formatting of the Tx Packet header and data fields. The User Application is also required to maintain a Tx Packet queue that keeps track of up to 8 packets pending transmission completion. The User Application must also keep track of used and available packet bins in the Tx Packet Buffer memory. The following steps describe the high level process that is necessary to complete a single Tx Packet transmission.

1. The User Application formats the Tx Packet header and data information and writes it into the next sequentially available Bin in the Tx Packet Buffer. The User Application must keep track of used bins and available bins as part of the Tx Packet Queue management. The User Application should only load up to seven request packets at a time into the Tx Packet Buffer. This will assure that at least one bin is always available for a Tx Response Packet to be loaded.

2. The User Application formats the associated Packet Descriptor data value and writes it to the Tx Length FIFO. This places the packet in the "available for transmission" queue that is managed by the Tx Link Controller (TLC).

3. The User Application waits for the Tx Status interrupt to occur. When it does, the User Application must read the Tx Status FIFO. The Packet ID field within the data word is the source bin# of the Tx packet that the PHY is reporting as completed. The User Application uses this value to update it’s local Tx Packet queue information.

4. The User Application must clear the TX Acknowledge Interrupt in the IP Interrupt Status Register. In most processor applications this will involve clearing the Tx Status TX Acknowledge interrupt in the IP Status register of the Processor Buffer IPIF and then clearing the RapidIO Device interrupt in the User System's main Interrupt Controller (INTC).

5. Send the next Packet.

General Rx Packet Processing Flow

The User Application is required to unpack and process all pertinent Rx Packet header and data information. The following flow describes the high level flow of receiving a packet.

1. The User Application waits until a Rx Packet Available Interrupt occurs. This interrupt is defined to mean that at least one RapidIO Rx packet is resident in the Rx Packet Buffer and a corresponding packet descriptor is in the Rx Length FIFO.

2. The User Application must read the Rx Length FIFO. The read data value is an Rx Packet descriptor that indicates the Bin# where the associated packet data is stored in the Rx Packet Buffer and the byte count for the packet.

3. Using the Packet Descriptor data, the User Application reads the packet data long words out of the Rx Packet Buffer.

4. The User Application acknowledges the receipt of the Rx Packet to the hardware by reading from the Rx Status port in the Processor Buffer. This causes the Processor Buffer Rx Link Controller to release for use the packet Bin that held the Rx Packet.

5. The User Application must clear the Rx Packet Available Interrupt. In most processor applications this will involve clearing the Rx Packet Available interrupt in the IP Interrupt Status register and then clearing the RapidIO Device interrupt in the System's main Interrupt Controller (INTC).

6. Ready for next packet.

Design Constraints

The PLB RapidIO LVDS core has numerous design constraint requirements needed to achieve optimal performance. These are primarily for the RapidIO PHY. An example UCF for this core is available and must be modified for use in the system.

The file is named **plb_rapidio_lvds_example.ucf**. This file should be viewed when generating the constraints for the PLB RapidIO LVDS part of the user's system UCF. This file is found in the core's data subdirectory in the core EDK library. Please refer to the *EDK Getting Started Guide* for more information on the location of this file.

Example Contraints File - plb_rapidio_lvds_example.ucf

The example constraint file is divided into several sections. A breif discussion of each section follows to aid in creating a constraints file for a particular user application. Refer to the example constraints file, plb_rapidio_lvds_example.ucf for the following discussion.

PLB_RAPIDIO_LVDS Time Spec

The PLB_RAPIDIO_LVDS Time Spec section defines Timing Name (TNM) groups to which timing requirments can be applied. This section sets the minimum allowable net delays for critical top level signals for the plb_rapidio_lvds module. If different clock rates are used, then the timing numbers in this section should be modified appropriatly.

PLB_RAPIDIO_LVDS DCM/BUFG Constraints

This section constrains the physical placement of the plb_rapidio_lvds module's DCM along with the placement of the supporting global buffers. The placement of the DCM should be modified appropriatly for a users specific application. It should be noted these constraints are only valid if the C_INCLUDE_TX_DCM is set to a logical 1.

RapidIO FPGA I/O Pin Assignment

The RapidIO pin assignment physically assigns the lvds signal pairs to the FPGA pins and will be unique for a particular user's custom design. Because the names of these signals are unique to your design, they should be modified appropriatly. These signals feed in/out LVDS buffers that are instantiated by the user, usually in the system top level.

LVDS I/O Standard Assignment

This section is shown as an example of how to assign the I/O Standard to the differential I/O Buffer. The example instantiations in the constraints file do NOT exist in the plb_rapidio_lvds module. These instantiation names will need to be modified to match the users instantiation of the differential IO Buffers.

The following code snippet shows an example of how to instantiate the differential I/O Buffers.

```
architecture implementation of lvds_ibuf is

component IBUFDS
        port (
          I : In  std_logic;
          IB : In  std_logic;
          O  : Out std_logic
```

```
          );
end component IBUFDS;

begin

I_IBD0 : IBUFDS
port map(
       I  => RIO_Rclk_p,
       IB => RIO_Rclk_n,
       O  =>  RIO_Rclk
       );
end implementation;
```

Where RIO_Rclk is mapped to the port on the plb_rapidio_lvds module and RIO_Rclk_p and RIO_Rclk_n are assigned to a differential FPGA pin pair.

The IOSTANDARD for the differential buffer can then be set to LVDS_25 in the contraints file. An example is shown below.

INST "I_IBD0" IOSTANDARD=LVDS_25;

Or the IOSTANDARD can be set using an attribute for the component instantiation. An example is shown below.

attribute IOSTANDARD : string

attribute IOSTANDARD of I_IBD0 : label is "LVDS_25"

PHY I/O Double Data Rate Flops

The PHY I/O Double Data Rate Flops sections assigns the PHY signals that need to be registered in the IOB for double data rate clocking. These signals sould not be modified except for the instantiation path.

PHY Time Specs

The PHY Time Spec section defines Timing Name (TNM) groups to which timing requirments can be applied. This section sets the minimum allowable net delays for critical signals withing the PHY. If different clock rates are used, then the timing numbers in this section should be modified appropriatly.

PHY RLOC Constraints for OPLM

This section contains the Relative Location (RLOC) Constraints for the PHY. The RLOC constraint groups logic elements into discrete sets. The mapper will maintain the relative order of the group and move the whole group as a single unit. This section should be copied as is with modifications being made only to the instantiation path names as appropriate. Within this section the U_SET mapping constraint is also used. This constraint groups design elements with RLOC constraints into a single set. Except for instantiation path these constraints should also not be modified.

Design Implementation

Design Tools

The VHDL portions of the PLB RapidIO LVDS V1.00c design are compatible with the Xilinx ISE 6.2i tool suite. Depending on the type of User licensing acquired for the RapidIO PHY, the RapidIO PHY is deliveres either as an evaluation version or a fully functional version. The PHY module is provided as a pre-synthesized '.ngo' netlist file.

Design Verification

Design simulation was performed with the Model Technology Simulation tool version 5.7e in conjunction with the IBM Bus Functional Model (BFM) Toolkit (PLB_TOOLKIT4X_090902) for the CoreConnect™ PLB.

The RapidIO PHY was simulated in the design with the phy_8_lvds.vhd simulation model. It was configured in an external loop back mode where the Tx signals were tied to the Rx signals.

Target Technology

The intended target technology family is Virtex™II Pro FPGAs. The design is also compatible with Virtex™II devices.

Device Utilization and Performance Benchmarks

The PLB RapidIO LVDS is a module that will be used with other design modules and system configurations within an FPGA. As a result, the utilization and timing numbers reported in this section can only be estimates. As the PLB RapidIO LVDS is combined with other pieces of the FPGA design, the utilization of FPGA resources and timing will vary from the results reported here.

Resource Utilization was analyzed in the following manner. An HDL wrapper was created that instantiated the PLB RapidIO LVDS module and the RapidIO LVDS buffers. This wrapper facilitated the changing of the parameterization of the module for each implementation run.

Timing was analyzed in the following manner. An HDL wrapper was created that instantiated the PLB RapidIO LVDS module and the RapidIO LVDS buffers. This wrapper placed registers on all of the module's PLB inputs and outputs. The RapidIO interface signals, being DDR registered signals in IOBs, had no additional registering.

Each of the wrapper files described above was then implemented in a Virtex-II Pro XC2VP7-7 device using the Xilinx implementation tools. The results of the analysis are shown in Table 17.

Table 17: **PLB RapidIO LVDS FPGA Performance and Resource Utilization Benchmarks (Virtex-II Pro XC2VP7-7)**

Parameter Values	Device Resources Used			f_{MAX} (1)			
C_DEV_BURST_ ENABLE	Slices	Slice Flip-Flops	4-input LUTs	PLB_Clk	RIO_Sys_clk (2)	RIO_Rclk (2)	RIO_Tclk (2)
= 0	2,920	2,907	6,077	153 MHz	125 MHz	250 MHz	250 MHz
= 1	3,045	3,032	6,345	122 MHz	125 MHz	250 MHz	250 MHz

Notes:
1. Timing benchmark numbers approach the performance ceiling rather than representing performance under typical user conditions.
2. These clocks are fixed at the frequencies shown. The module design supports these frequencies when constrained properly.

Specification Exceptions

PLB Signaling Exceptions

This PLB RapidIO LVDS design does not use the PLB signal set as indicated by cell shading in Table 2.

Reference Documents

The following documents contain reference information important to understanding the PLB RapidIO LVDS design:

- IBM ***64-Bit Processor Local Bus, Architectural Specification*** (v3.x)
- Xilinx LogiCORE™ ***RapidIO 8-bit Port Physical Layer Interface*** Product Specification
- Xilinx LogiCORE™ *RapidIO Physical Layer Interface Design Guide (v1.x)*

Revision History

The following table shows the revision history for this document.

Date	Version	Revision	Ed.
01/13/02	1.0	Initial Xilinx release.	
07/09/03	1.2	Update to new template	
02/25/03	1.3	Updated to reflect v1.00b of the core	
03/24/04	1.4	Updated to reflect v1.00c of the core. Added detail to Design Contraints section.	
04/20/04	1.4.1	Updated FPGA Utilization Numbers	
06/04/04	1.4.2	Updated per Yong Zhu.	lcw
06/10/04	1.4.3	Added clarity and an example to Receive Packet format	gb