



XAPP775 (v1.0) August 25, 2004

10 Gigabit Ethernet/FibreChannel PCS Reference Design

Author: Justin Gaither and Marc Cimadevilla

Summary

This application note describes the 10 Gigabit Ethernet Physical Coding Sublayer (PCS) reference design for Xilinx Virtex-II™ and Virtex-II Pro™ FPGAs. The PCS connects between a Xilinx RocketPHY™ 10 Gb/s transceiver and the Xilinx LogicCORE™ 10 Gigabit Ethernet Media Access Controller (MAC) core, LogicCORE XAUI core or 10 Gigabit Media Independent Interface (XGMII) Reference Design (XAPP606). All source files are included with the reference design to allow customization for specific applications. The interface between the RocketPHY and the PCS can be a dual data rate implementation of a 10 Gigabit 16-Bit Interface (XSBI) which takes advantage of the RocketPHY device's custom DDR modes. This reference design can also be used as the PCS layer of a 10 Gigabit FibreChannel implementation.

Introduction

Figure 1 (from the IEEE 802.3ae 10 Gigabit Ethernet standard) illustrates the placement of this PCS reference design relative to other layers of the Ethernet protocol.

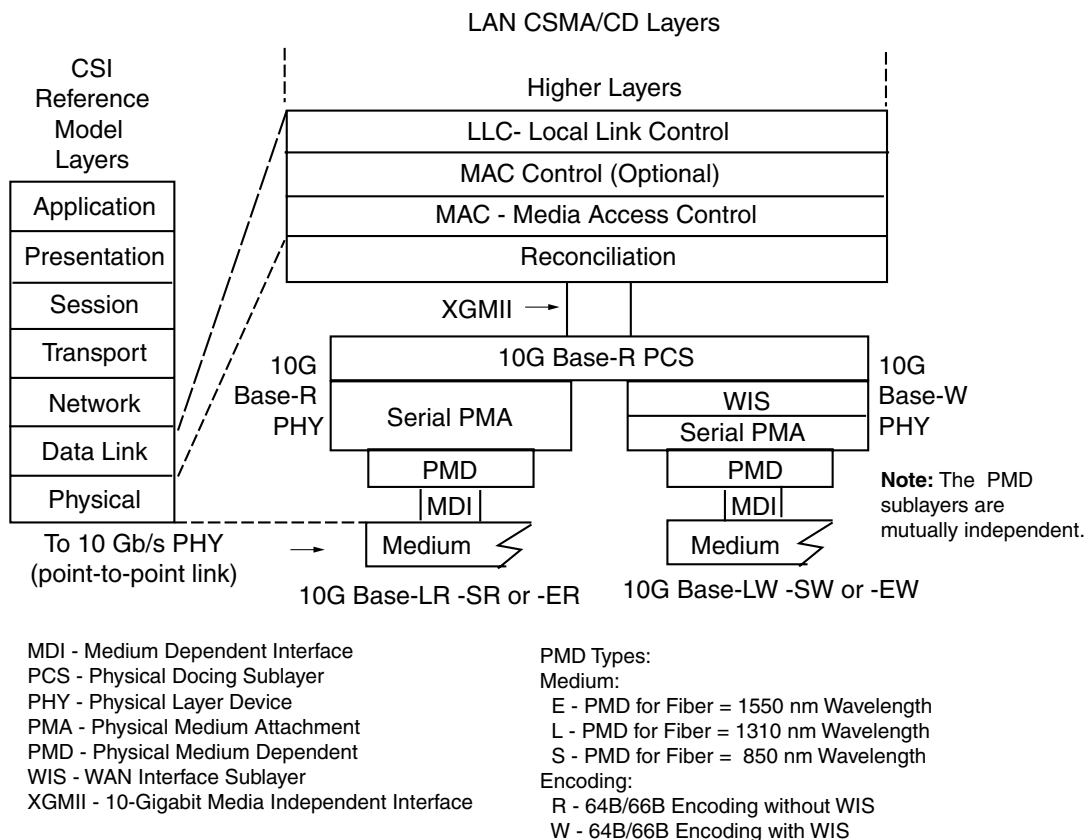


Figure 1: PCS Design Placement

© 2004 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and further disclaimers are as listed at <http://www.xilinx.com/legal.htm>. All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

The Serial Physical Medium Attachment (PMA) is implemented using a Xilinx RocketPHY 10 Gigabit Transceiver and the MAC layer can be implemented using the Xilinx LogiCORE 10 Gigabit Ethernet MAC. There are Alliance core partners who can also provide the WAN Interface Sublayer (WIS), if needed, and there are multiple vendors of XFP modules, which can be used to implement the Physical Medium Dependent (PMD) sublayer.

Figure 2 shows the basic architecture of the reference design. This design has been optimized to reduce the number of clocking resources needed within the FPGA. The Core_clk is setup to be a global clock resource that is used by other circuits inside the FPGA, such as the 10 Gigabit Ethernet MAC. The 4:1 mux and demux circuits leverage the XAPP265 DDR interface reference design or the XAPP622 SDR interface reference design. Source code for both implementations is included in this reference design. An important feature of this reference design is that both TX and RX data paths have clock tolerance buffers included, thereby allowing the clk156 (Core_clk) to be asynchronous to the TXPCLKP/N and RXPCLK clocks. This greatly simplifies clock synchronization of the system and allows independent clocks to be used for the 644.53125 MHz 16-bit XSBI interface and the 156.25 MHz system clock (Core_clk). Synchronizing these two clocks would require an external phase-locked loop (PLL).

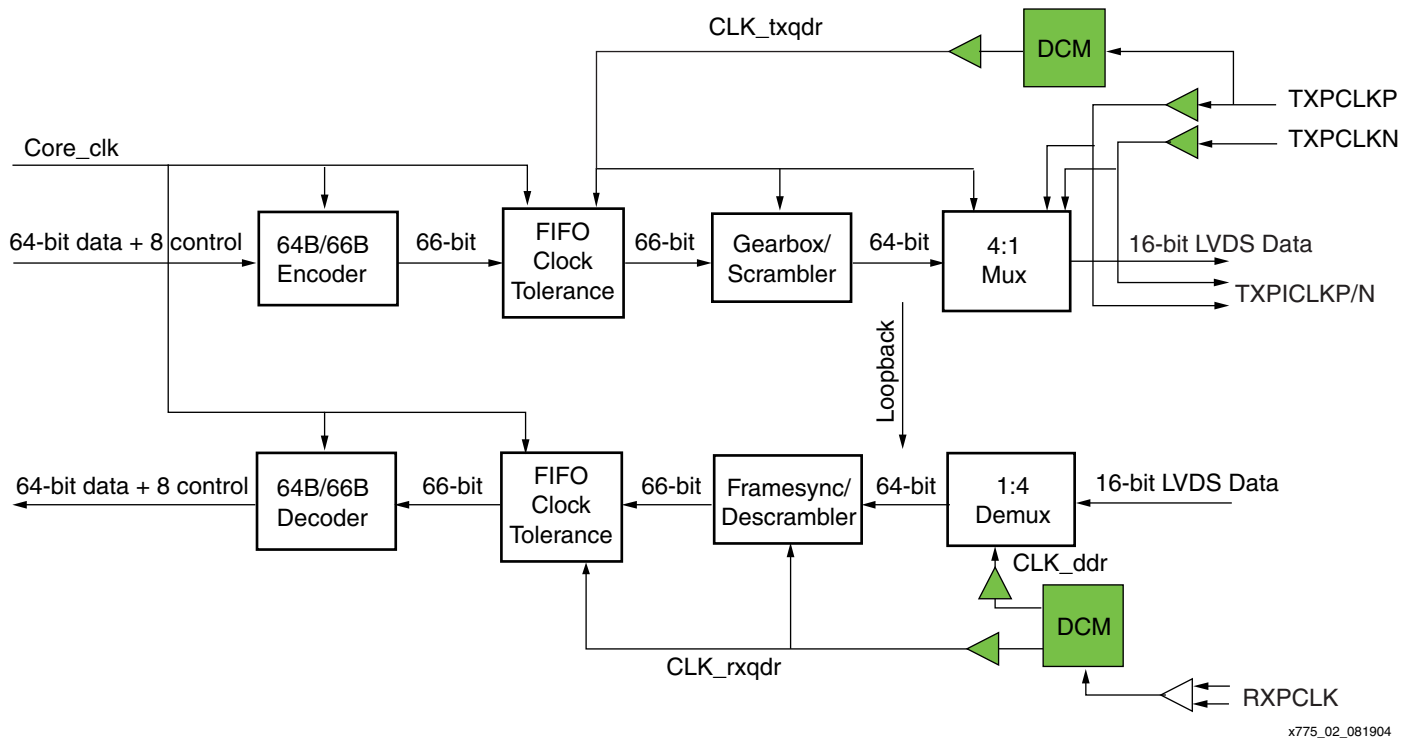


Figure 2: Reference Design Basic Architecture

Resource Requirements

The resource requirements are shown in Table 1.

Table 1: Resource Requirements

Description	Slices	LUT	Registers	Block RAM	BUFGMUX	DCM
PCS SDR	4234	6888	3185	8	5	2
PCS DDR	4167	6338	3623	8	4	2

FPGA Implementation

The FPGA implementation consists of Verilog source code, constraint files, and project files to build the design inside your application. The interface side already has LVDS IBUFs and OBUFs for connection to the external 10 Gb/s transceiver as part of the interface application notes. The MAC side consists of a 64-bit data bus and 8-bit control bus for each transmit and receive direction. This internal interface can connect directly to the 10 Gigabit Ethernet MAC.

Encoder/Decoder

The encoder and decoder blocks translate from/to the 64-bit XGMII data to 66-bit data bus that is 64B/66B encoded according to the IEEE 802.3ae standard.

FIFO Clock Tolerance

The FIFO clock tolerance block is responsible for synchronizing the data packets to the higher speed interface clock domain. The write side of the FIFO removes all available idles and (repeated) consecutive Sequence ordered sets before writing into the FIFO. The read side inserts idles as needed in order to keep the FIFO half full. The FIFO also holds data for the gearbox and Framesync blocks so that they can properly transmit and receive data. This same block is used on both TX and RX data paths.

Scrambler/Gearbox

The scrambler is used on the TX path to ensure sufficient transitions in the serial interface to support clock recovery and optical transmission. The polynomial used for scrambling is specified in the IEEE 802.3ae specification as $(1 + x^{39} + x^{58})$. The gearbox is used to translate the data bus from 66 bits to 64 bits wide, so that the 64 bits can be multiplexed into a 16-bit interface.

Descrambler/Framesync

The descrambler is used to reverse the scrambling operation performed on the transmit side, so that the data can be properly decoded. It operates using the same polynomial as the scrambler. The Framesync is used to translate the data bus from a 64-bit non-aligned data bus to a 66-bit word aligned data bus. The alignment is performed by searching the data field for the two Framesync bits.

Top-Level Signals

The top-level signals and their descriptions are shown in [Table 2](#).

Table 2: Top-Level Signals

Signal Name	Type	Description
xgmii_txd[63:0]	Internal-Input	Input data (double-width XGMII)
xgmii_txc[7:0]	Internal-Input	Input control signals (double-width XGMII)
TXP_data, TXN_data[15:0]	Output-LVDS	
TXP_clk, TXN_clk	Output-LVDS	
clk156	Internal-Input	156.25 MHz clock associated with 10-Gigabit Media (XG) data inputs and outputs (BUFG should be placed external to PCS core)
xgmii_rxd[63:0]	Internal-Output	Output received data (double-width XGMII)
xgmii_rxc[7:0]	Internal-Output	Output control signals (double-width XGMII)
RXP_data, RXN_data[15:0]	Input-LVDS	
RXP_clk, RXN_clk	Input-LVDS	

Table 2: Top-Level Signals (Continued)

Signal Name	Type	Description
txpclkp/n	Input-LVDS	Interface clock used to source the TX data and clock
tx_fifo_spill	Internal-Output	Indicates a spill has occurred in TX FIFO
rx_fifo_spill	Internal-Output	Indicates a spill has occurred in RX FIFO
hi_ber	Internal-Output	Indicates a hi_ber condition see IEEE 802.3ae standard
ber_cnt[5:0]	Internal-Latched Output	Defined by IEEE 802.3ae 49.2.14.2
txlf	Internal-Output	PCS transmit_fault variable defined by IEEE 802.3ae 49.2.14.2
rxlf	Internal-Output	PCS receive_fault variable defined by IEEE 802.3ae 49.2.14.2
blk_lock	Internal-Output	PCS block_lock variable defined by IEEE 802.3ae 49.2.14.2
errd_blks[7:0]	Internal-Latched Output	Defined by IEEE 802.3ae 49.2.14.2
ready	Internal-Output	Indicates the dynamic phase adjustment on RX path is complete
txpsdone	Internal-Output	Indicates the dynamic phase adjustment on TX path is complete.
txdcmlock	Internal-Output	Indicates TX path DCM is locked
rxdcmlock	Internal-Output	Indicates RX path DCM is locked
linkstatus	Internal-Output	Defined by IEEE 802.3ae
seedA[57:0]	Internal-Input	Seed inputs for IEEE 802.3ae test pattern
seedB[57:0]	Internal-Input	Seed inputs for IEEE 802.3ae test pattern
jtest_errc[15:0]	Internal-Output	Defined by IEEE 802.3ae 49.2.14.2
signal_ok	Internal-Input	Defined by IEEE 802.3ae
rx_jtm_en	Internal-Input	Defined by IEEE 802.3ae
tx_jtm_en	Internal-Input	Defined by IEEE 802.3ae
bypass_descram	Internal-Input	Enables the descrambler bypass
bypass_scram	Internal-Input	Enables the scrambler bypass
bypass_66decoder	Internal-Input	Enables the 66-bit decoder bypass
bypass_66encoder	Internal-Input	Enables the 66-bit encoder bypass
txprbs31_sel	Internal-Input	Defined by IEEE 802.3ae
rxprbs31_sel	Internal-Input	Defined by IEEE 802.3ae
jtm_dps_0/1	Internal-Input	Defined by IEEE 802.3ae
clear_jtest_errc	Internal-Input	Clears test_pattern_error_count
clear_ber_cnt	Internal-Input	Clears ber_count
clear_err_blks	Internal-Input	Clears erroredblock_cnt
arstb	Internal-Input	Resets all digital circuits and RocketIO X device
rxdcmrst	Internal-Input	Reset for RX DCM
rxpsen	Internal-Input	Phase shift enable for RX DCM
rxpsdir	Internal-Input	INC/DEC phase shift for RX DCM

Table 2: Top-Level Signals (Continued)

Signal Name	Type	Description
rxpsclk	Internal-Input	Clock for RX DCM phase shift
txdcmrst	Internal-Input	Reset for TX DCM
txpsen	Internal-Input	Phase shift enable for TX DCM
txpsdir	Internal-Input	INC/DEC phase shift for TX DCM
txpsclk	Internal-Input	Clock for TX DCM phase shift

Verification Testbench

The design was verified using the RocketPHY development kit HW-RPHY-DVLP-1. All necessary files are included to demonstrate PCS operation on this set of Xilinx development boards.

The simulation and hardware test benches share a common interface. See Figure 3. The main difference in the hardware testbench is that ChipScope Pro™ analyzer is used to monitor the RX data bus instead of a waveform viewer. The testbench uses block RAMs to store the pattern information. One memory (Data) is used to store 72-bit XGMII data words. There is a fixed set of available data words available. A second memory (Control) is used to store the index and count for the testbench. The index points to a location in the Data memory and the count indicates how many clock cycles this word is transmitted.

The Test Controller reads the first location of the Control memory and transmits data from the Data memory as indicated by the index and count. It then reads the next location in the Control memory and repeats. After it has reached the end of the Control memory, it starts over. On the receive side, we can view the XGMII RX data and verify that the correct data is received. It will be common to have more or fewer number of idle words between packets due to clock tolerance adjustments. The simulation model does not include the RocketPHY SmartModel, which is available upon request from your local Xilinx FAE.

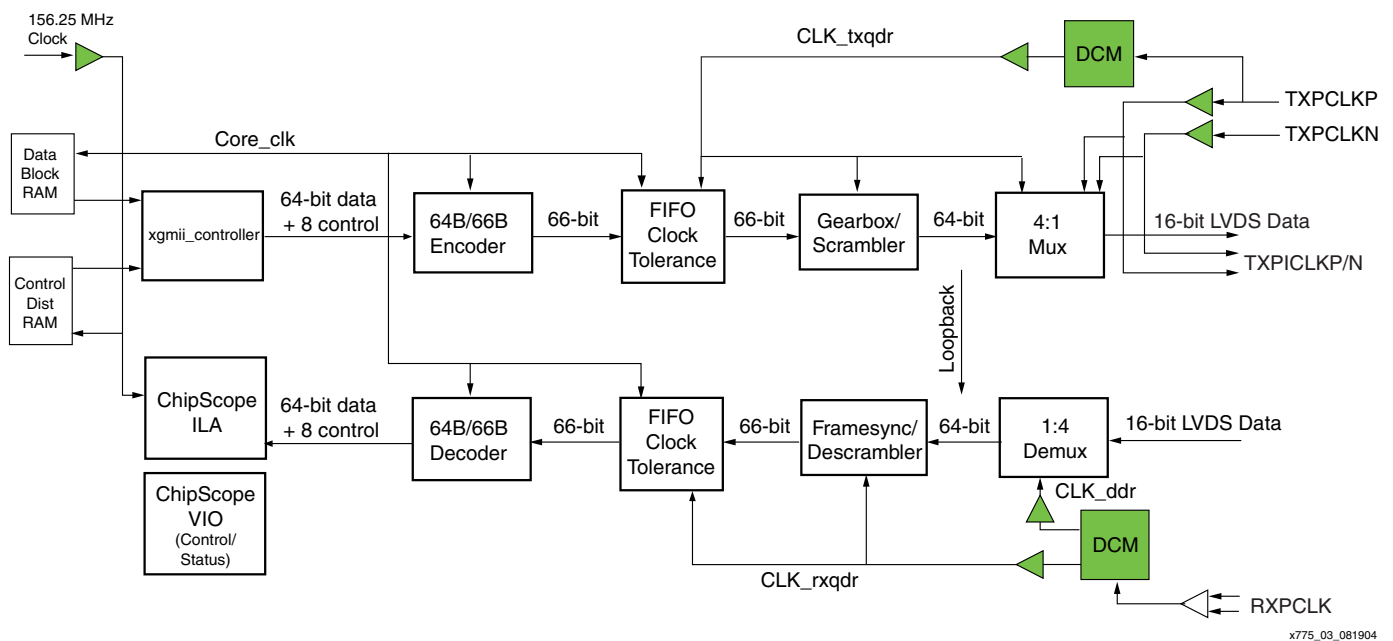


Figure 3: Verification Testbench Diagram

Reference Design

The reference design file consists of a zip file ([xapp775.zip](#)) as described below:

Root directory	
sim	Simulation directory
	run_sim script used to run NCVerilog simulation
	testbench.vc NCVerilog command options
verilog	Source directory
	PCS all Verilog files for PCS
	sim_test all Verilog files for simulation testbench
	hw_test all Verilog files for hardware testbench
	include Verilog include files
	ddr Verilog files for DDR interface
projnav	Xilinx Project Navigator directory
	xst xst working directory
	chipscope directory containing ChipScope modules (i.e., icon, ila, vio)
	*.bit bit file for ML10G
	*.ucf Constraint file for ML10G
	*.prj ProjNav project file
	MK_PCS_MGT.cpj ChipScope project file
coregen	directory containing coregen data
vhdl	
	pcs VHDL files for PCS
	hwtest VHDL to implement hardware test case
	coregen VHDL versions of cores

Source code is provided for SDR interface in Verilog and VHDL. For DDR interface, the Verilog files are provided which replace the SDR versions. The project directory has all necessary files to regenerate the Verilog version of the SDR interface hardware test platform.

