



XAPP759 (v1.1) March 4, 2005

Configurable Physical Coding Sublayer

Author: Dai Huang, Jack Lo, and Shalin Sheth

Summary

This application note describes a Configurable Physical Coding Sublayer (CPCS) reference design that extends the functionality of the Xilinx RocketIO™ multi-gigabit transceiver (MGT) blocks in the Virtex™-II Pro FPGA family.

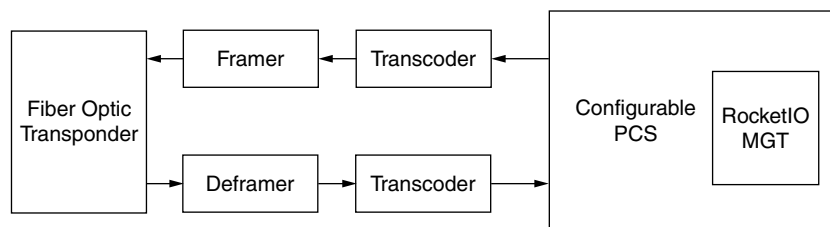
Introduction

The CPCS reference design provides a multi-mode PCS layer for Fibre Channel (both 1.0625 Gb/s and 2.125 Gb/s), ESCON/SBCON, and Gigabit Ethernet. This PCS layer is dynamically configurable under software control. Client signals can be protocol data unit (PDU) oriented frames, or a block-code oriented constant bit rate stream. PDU oriented data transmission is referred to as *framed mode* transmission. Block-code oriented data transmission is referred to as *transparent mode* transmission. CPCS supports both framed mode and transparent mode transmission for Gigabit Ethernet. CPCS supports only transparent mode transmission for Fibre Channel and ESCON/SBCON.

Table 1 lists PCS modes that are supported by the CPCS reference design. Figure 1 illustrates a typical application.

Table 1: PCS Mode

PCS Mode	Application	Line Rate	Data Rate	Reference
2G FC	Transparent mode 2G Fibre Channel	2.125 Gb/s	1.7 Gb/s	ANSI X3.230 (1994) FC-PH clause 12
1G FC	Transparent mode 1G Fibre Channel	1.0625 Gb/s	850 Mb/s	ANSI X3.230 (1994) FC-PH clause 12
1000BASE-X	Frame mode and transparent mode 1G Ethernet	1.25 Gb/s	1.0 Gb/s	IEEE802.3-2002 clause 36 and 37
ESCON	Transparent mode ESCON/SBCON	200 Mb/s	160 Mb/s	ANSI X3.296 (1997) SBCON clause 7



X759_01_030504

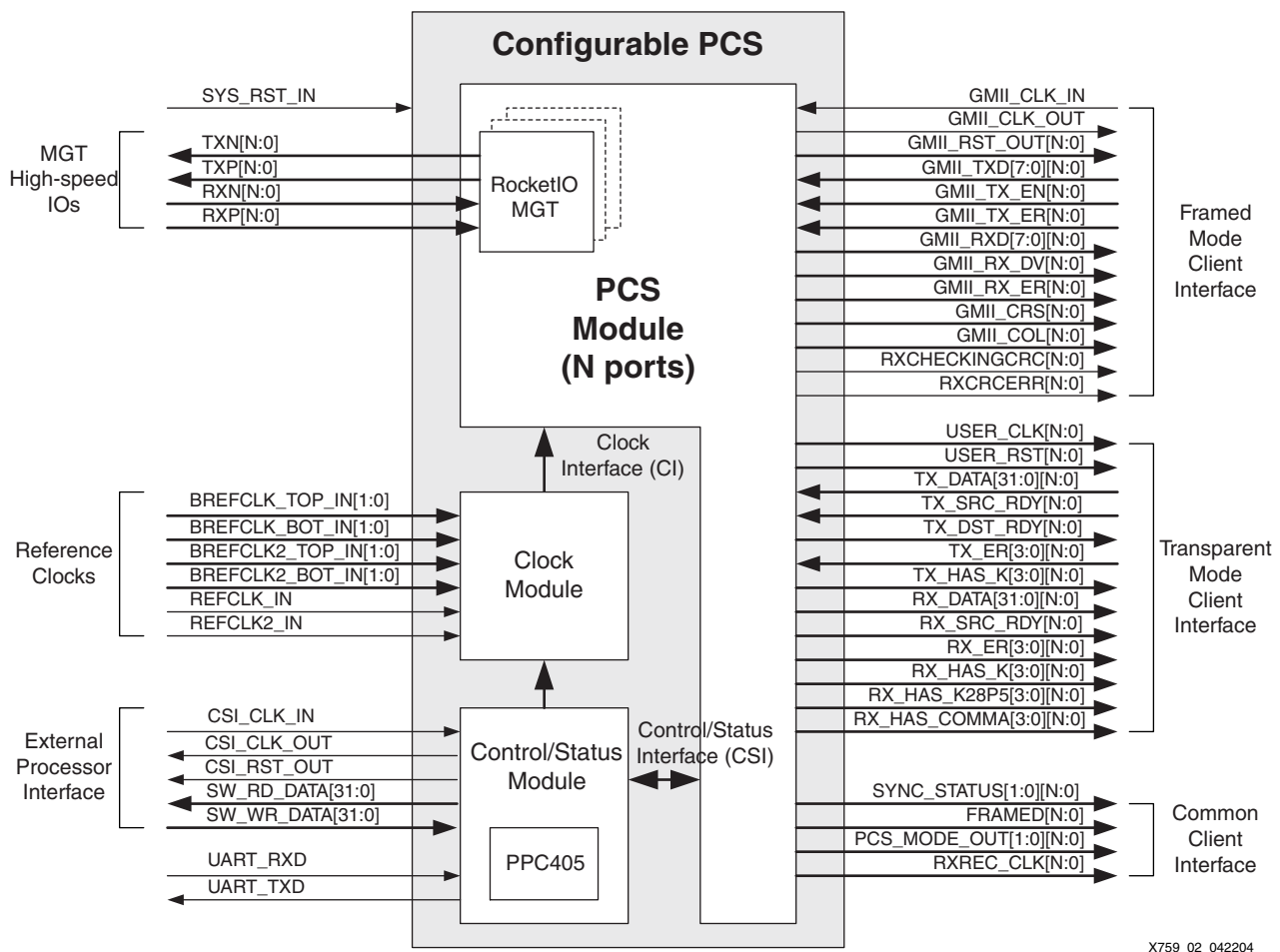
Figure 1: Application of Configurable PCS

Architecture and Interfaces

Interface Description

Figure 2 shows the overall architecture of the CPCS reference design and its interfaces. CPCS provides two types of client interfaces to accommodate framed mode and transparent mode transmission. The framed mode client interface allows for Gigabit Ethernet framed mode transmission through a built-in Gigabit Media Independent Interface (GMII). The transparent mode client interface supports only transparent mode transmission. CPCS enables one of these two client interfaces on each port at a time, depending on the PCS mode choice of the port. The common client interface provides common signals to the client used in both framed and transparent mode transmission.

A single-port CPCS block encapsulates one MGT, which connects to high-speed serial I/O pads through the differential pair signals TXN, TXP and RXN, RXP for the transmitting and receiving directions, respectively. A multi-port CPCS block contains multiple MGTs and provides dedicated client interfaces and high-speed I/O pads for each port. The external processor interface is provided as an option to connect the CPCS to an external processor in place of the embedded PowerPC 405 processor (PPC405) in the control/status module. The remaining interfaces, which include the control/status interface (CSI) and clock interface (CI), connect the internal modules in the CPCS.



X759_02_042204

Figure 2: Interfaces to the Configurable PCS

Table 2 and Table 3 list the framed mode client interface and transparent mode client interface signals, respectively. The common client interface signals are shown in Table 4. The framed mode client interface supports GMII for interfacing to a MAC device through an 8-bit data path. The transparent mode client interface provides additional side-band signals in addition to the data signals for comma detection and other PCS control characters and information. As the data signal on the transparent mode interface is 32 bits wide, a side-band signal is 4 bits wide, each for one 8B/10B character.

Table 2: Framed Mode Client Interface Signals

Name	Width	Direction	Description
GMII_CLK_IN	1	Input	Optional external 125 MHz GMII clock input. Use when USE_EXT_GMII_CLK parameter is set to TRUE.
GMII_CLK_OUT	1	Output	125 MHz GMII clock to the client. Client should always use this clock to drive its GMII related circuits.
GMII_RST_OUT	N x 1	Output	Synchronous reset signal to each port of the client. Client should always use this signal to reset its GMII related circuits.
GMII_TXD	N x 8	Input	Framed data from client to transmit.
GMII_TX_EN	N x 1	Input	Data enable control signal from client.
GMII_TX_ER	N x 1	Input	Error control signal from client.
GMII_RXD	N x 8	Output	Received framed data to client.
GMII_RX_DV	N x 1	Output	Data valid control signal to client.
GMII_RX_ER	N x 1	Output	Error control signal to client.
GMII_CRS	N x 1	Output	Carrier sense control signal to client.
GMII_COL	N x 1	Output	Collision control signal to client.
RXCHECKINGCRC	N x 1	Output	CRC status from the MGT. Asserted high to indicate that the MGT receiver has recognized the end of a data packet. This signal is only valid when the USE_MGT_CRC parameter is set to TRUE.
RXCRCERR	N x 1	Output	Asserted high to indicate the CRC code is incorrect in the received frame. If the received frame contains a CRC error, this signal is asserted along with RXCHECKINGCRC. This signal is only valid when the USE_MGT_CRC parameter is set to TRUE.

Note: N is the number of CPCS ports implemented in the system. N = 1, 2,...8

Table 3: Transparent Mode Client Interface Signals

Name	Width	Direction	Description
USER_CLK	N x 1	Output	Clock signal to each port of the client. This clock rate will adapt to the PCS mode selected on each CPCS port. This clock is frequency locked to the reference clock inputs as listed in Table 5 .
USER_RST	N x 1	Output	Synchronous reset signal to the client.
RX_DATA	N x 32	Output	Received data or control character to the client.
RX_SRC_RDY	N x 1	Output	Asserted when the interface is ready to pass valid data. Deasserted when data on the interface is invalid.
RX_ER	N x 4	Output	Error control signal to the client. Each bit corresponds to a byte on RX_DATA. This signal indicates a code violation error in the RX_DATA on the corresponding byte. Such error may result from a bit error or disparity error detected in the receiving data stream.
RX_HAS_K	N x 4	Output	Asserted if a K character is found in RX_DATA. Each bit corresponds to a byte on RX_DATA.
RX_HAS_K28P5	N x 4	Output	Asserted if a K28.5 character is found in RX_DATA. Each bit corresponds to a byte on RX_DATA. Each bit is invalid when RX_ER is high on corresponding bit.
RX_HAS_COMMA	N x 4	Output	Asserted if a comma character is found in RX_DATA. Each bit corresponds to a byte on RX_DATA. Each bit is invalid when RX_ER is high on corresponding bit.
TX_DATA	N x 32	Input	Data or control character to transmit.
TX_SRC_RDY	N x 1	Input	Asserted when client is ready to send data.
TX_DST_RDY	N x 1	Output	Asserted when the interface is ready to accept data. When deasserted, TX_DATA, TX_ER, TX_HAS_K inputs to the interface are required to hold.
TX_ER	N x 4	Input	Reserved. Tied to low.
TX_HAS_K	N x 4	Input	Asserted if a K character is present in TX_DATA. Each bit corresponds to a byte on TX_DATA.

Note: N is the number of CPCS ports implemented in the system. N = 1, 2,...8

Table 4: Common Client Interface Signals

Name	Width	Direction	Description
SYNC_STATUS	N x 2	Output	PCS synchronization status on each CPCS port. Bit 0 is asserted when synchronization is acquired. Bit 1 is reserved.
FRAMED	N x 1	Output	Asserted if framed mode transmission is currently in use on a CPCS port. Deasserted if transparent mode transmission is in use.
PCS_MODE_OUT	N x 2	Output	PCS mode in use on a CPCS port.
RXREC_CLK	N x 1	Output	Non-buffered MGT RX recovered clock. This clock can be used for diagnostic purposes.

Note: N is the number of CPCS ports implemented in the system. N = 1, 2,...8

Table 5 lists the rest of the interface signals on the CPCS block.

Table 5: Other CPCS Interface Signals

Name	Width	Direction	Description
TXP and TXN	N	Output	High-speed differential serial output from the RocketIO MGT in each CPCS port.
RXP and RXN	N	Output	High-speed differential serial input to the RocketIO MGT in each CPCS port.
BREFCLK_TOP_IN BREFCLK_BOT_IN	2	Input	When IBUFG_INSERTION is set to TRUE, this port is used to supply 106.25 MHz differential reference clock inputs for FC PCS (2G). When MGTs on the top bank of the device are used, BREFCLK_TOP_IN must be connected to the dedicated pads on the top bank of the FPGA. Otherwise, it can be tied to low. Similarly, BREFCLK_BOT_IN is used for the bottom bank of MGTs. When IBUFG_INSERTION is set to FALSE, only bit 0 of this port is used to supply the single-ended clock.
BREFCLK2_TOP_IN BREFCLK2_BOT_IN	2	Input	53.125 MHz differential or single-ended reference clock inputs for FC PCS (1G).
REFCLK_IN	1	Input	62.5 MHz reference clock input for 1000BASE-X PCS.
REFCLK2_IN	1	Input	80 MHz reference clock input for ESCON PCS.
SYS_RST_IN	1	Input	Asynchronous reset to the CPCS block.
CSI_CLK_IN	1	Input	Optional external control/status interface clock input. Use when USE_EXT_CSI_CLK parameter is set to TRUE. This clock speed is typically 125 MHz.
CSI_CLK_OUT	1	Output	Control/status interface clock output. Use when USE_EXT_PROCESSOR parameter is set to TRUE. This clock is either generated internally or from CSI_CLK_IN. This clock speed is typically 125 MHz.
CSI_RST_OUT	1	Output	Synchronous control/status interface reset output. Use to reset the external processor when USE_EXT_PROCESSOR parameter is set to TRUE.
SW_RD_DATA	32	Output	Data to the external processor. Use when USE_EXT_PROCESSOR parameter is set to TRUE. This port should connect to a processor through general purpose I/O (GPIO) interface. See Table 12, page 18 for bit definition of this port.

Table 5: Other CPCS Interface Signals (Continued)

Name	Width	Direction	Description
SW_WR_DATA	32	Input	Data from the external processor. Use when USE_EXT_PROCESSPR parameter is set to TRUE. This port should connect to a processor through general purpose I/O (GPIO) interface. See Table 13, page 18 for bit definition of this port.
UART_TXD	1	Output	Data output to the serial port.
UART_RXD	1	Input	Data input from the serial port.

Note: N is the number of CPCS ports implemented in the system. N = 1, 2,...8

Transmission Order and Byte Mapping

The most significant byte (MSB) of the data bus on the transparent mode client interface (TX_DATA and RX_DATA) is transmitted and received first.

Most of the 4-bit wide status and control buses (TX_ER, TX_HAS_K, RX_ER, RX_HAS_K, etc.) on the transparent mode client interface correlate to a specific byte of TX_DATA or RX_DATA. This scheme is shown in Table 6.

Table 6: Byte Mapping of Status/Control Buses on Transparent Mode Client Interface

Bit of Status/Control Buses (e.g. RX_ER)	Data Bits of Data Buses (e.g. RX_DATA)
3	[31:24]
2	[23:16]
1	[15:8]
0	[7:0]

Timing Waveforms

The timing waveforms of frame transmission and reception on the framed mode client interface can be found in the the *Ethernet 1000BASE-X PCS/PMA LogiCORE* data sheet at http://www.xilinx.com/ipcenter/catalog/logicore/docs/gig_eth_pcs_pma.pdf.

Figure 3 shows a timing waveform of data transmission on the transparent mode client interface.

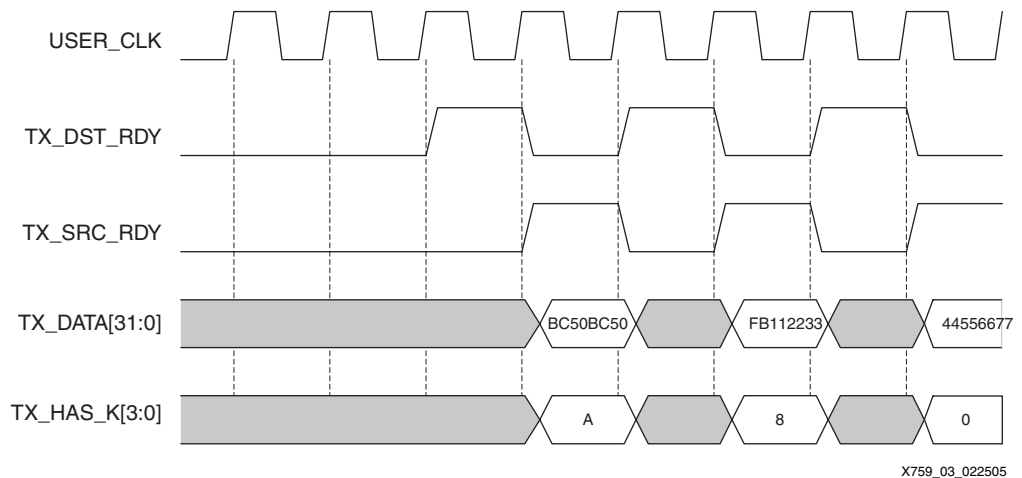
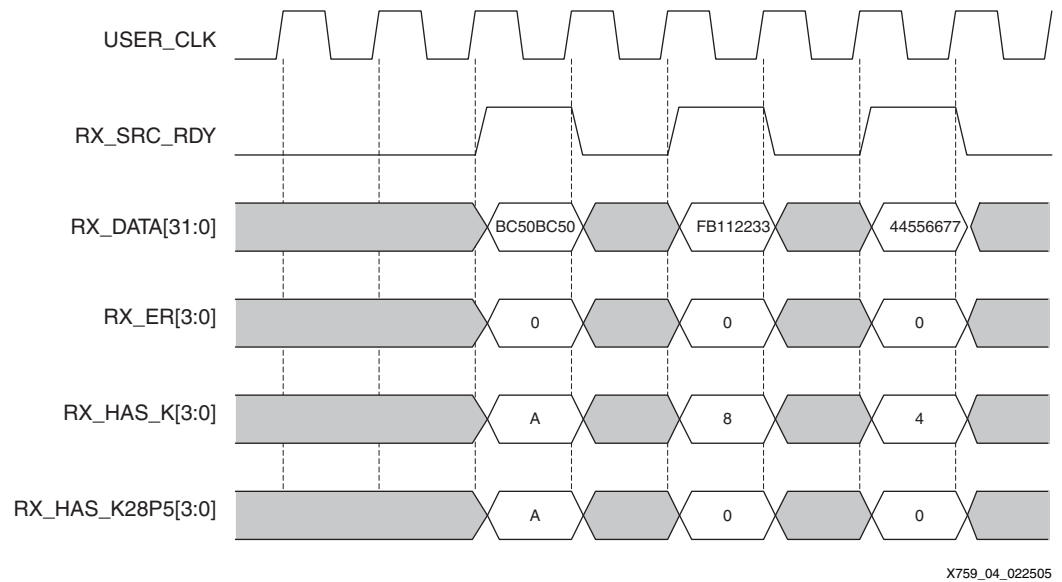


Figure 3: Frame Transmission on Transparent Mode Client Interface

The client should present valid data to the TX_DATA bus one cycle after the TX_DST_RDY is asserted. In order to maintain the proper data rate on the 32-bit transparent mode client interface, the TX_DST_RDY output is normally toggled every cycle in 1000BASE-X and FC

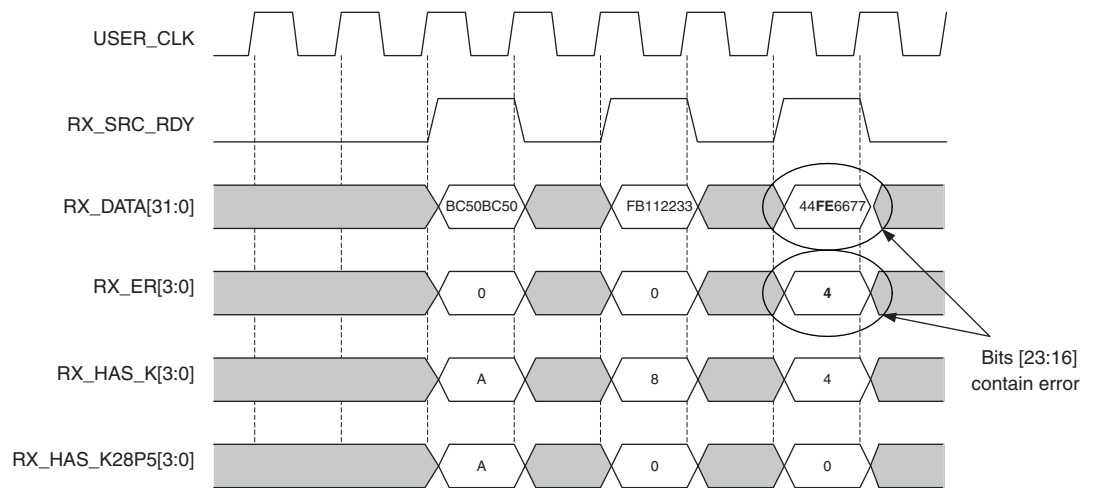
transparent PCS modes, and toggled every eight cycles in ESCON transparent PCS mode. The CPCS will treat any value placed on TX_DATA and TX_HAS_K within the TX_SRC_RDY assertion window as data to transmit.

The timing waveform of normal data reception on the transparent mode client interface is illustrated in Figure 4. The client should latch in values such as RX_DATA, RX_HAS_K, RX_HAS_K28P5 within the RX_SRC_RDY assertion window. Figure 5 shows the timing waveform of data reception with errors. In this example one of the RX_ER bits is asserted high to indicate a code violation on the corresponding byte on the RX_DATA bus. The erroneous byte is replaced by the specific error replacement code, which is one of the hardware configuration parameters on the CPCS. The corresponding bits on RX_HAS_K28P5 and RX_HAS_COMMA become invalid when RX_ER bit is high.



X759_04_022505

Figure 4: Normal Data Reception on Transparent Mode Client Interface



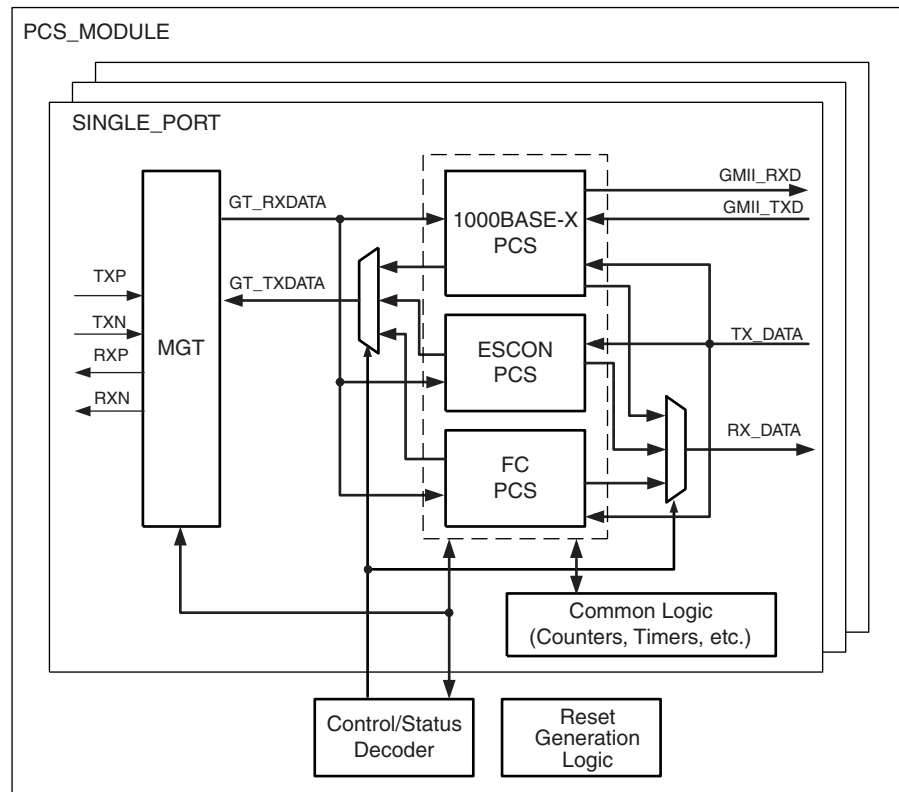
X759_05_022505

Figure 5: Data Reception with Errors on Transparent Mode Client Interface

Hardware Functional Description

Overview

The major functional block of the CPCS reference design is a multi-mode, multi-port PCS module. Figure 6 illustrates the block diagram of the PCS module. As shown in Figure 6, each SINGLE_PORT module consists of three PCS sub modules, dedicating to 1000BASE-X, Fibre Channel, and ESCON/SBCON PCS respectively. All three PCS submodules share one MGT. A multiplexer on the receiving and transmission data path allows data flowing between the MGT and the client interface through the selected PCS submodule. All three PCS submodules also share some common logic to reduce the size of the design, including global counters and timers. The PCS_MODULE block wraps multiple instantiations of SINGLE_PORT modules, propagates resets to each port, and decodes control/status data from the control/status module.



X759_06_042104

Figure 6: Multi-mode Multi-port PCS Module Block Diagram

1000BASE-X PCS

The 1000BASE-X PCS module in the CPCS block supports both framed mode and transparent mode data transmission. It is implemented using a Xilinx Ethernet 1000BASE-X PCS/PMA LogiCORE module designed to the IEEE 802.3-2002 standard. The LogiCORE module is ideally suited for the development of high-density Gigabit Ethernet communications and storage equipment. For details, see the *Ethernet 1000BASE-X PCS/PMA LogiCORE* data sheet at http://www.xilinx.com/ipcenter/catalog/logicore/docs/gig_eth_pcs_pma.pdf.

Framed Mode

The framed mode 1000BASE-X PCS module in CPCS provides the following features:

- Implements a full duplex single-speed 1 Gb/s Ethernet PCS with 1000BASE-X Physical Medium Attachment (PMA) using an integrated RocketIO MGT. Incorporates the RocketIO MGT to provide the SERDES, 8B/10B encoding/decoding, clock recovery and clock correction, bit synchronization and comma detection functions.

- Implements the Gigabit Media Independent Interface (GMII) as specified in IEEE802.3, clause 35.
- Implements the PCS transmit engine, receive engine, carrier sense and synchronization as specified in IEEE802.3, clause 36.
- Supports 1000BASE-X Auto-Negotiation as specified in IEEE802.3, clause 37. The auto-negotiation block can be omitted from the system using the parameter HAS_GE_AN.
- Supports configuration and monitoring through a simple control/status interface as replacement to the serial MDIO interface.
- Supports CRC generation on the transmitting data path and CRC checking on the receiving data path using RocketIO MGT's built-in CRC logic. Although this functionality is outside the scope of the 1000BASE-X PCS and PMA sublayers, this logic is an option of the 1000BASE-X PCS module. This functionality can be turned off using the parameter USE_MGT_CRC.
- Supports IDLE insertion on the transmission and IDLE termination on the reception.
- Uses Fibre Channel OLS primitive sequence as the secondary clock correction sequence.
- Provides a single 125 MHz clock to the client GMII interface. Supports clocking the GMII from an external 125 MHz clock input or internal DCM. When user selects internal DCM to generate the GMII clock, one additional clock buffer (BUFG) will be used.

Transparent Mode

The transparent mode 1000BASE-X PCS module in CPCS provides the following features:

- Implements a full duplex single-speed 1 Gb/s Ethernet PCS to support transparent mode transmission, which does not delineate frames, generate carrier sense, and handle auto-negotiation.
- Implements 1000BASE-X Physical Medium Attachment (PMA) using an integrated RocketIO MGT. Incorporates the RocketIO MGT to provide the SERDES, 8B/10B encoding/decoding, clock recovery and clock correction, bit synchronization, and comma detection functions.
- Implements a 32-bit transparent mode client interface for transmission and reception of data and control characters.
- Implements a subset of the PCS transmit engine, receive engine, and synchronization as specified in IEEE802.3, clause 36.
- Supports configuration and monitoring through a generic control/status interface as replacement to the serial MDIO interface.
- Supports IDLE insertion on transmission data path if data underrun condition on the client interface is detected. Received IDLEs are passed through to the client.
- Reports 8B/10B code violations and disparity errors through RX_ER signal on the client interface.
- Supports replacement of an erroneous character with a parameterizable valid data or special K character. An 8B/10B code violation or disparity error detected in the receiver can result in this erroneous data.
- Provides a single 62.5 MHz clock to the client.

Fibre Channel PCS

Fibre Channel (FC) PCS implemented in CPCS supports transparent mode transmission only.

The transparent mode FC PCS module provides the following features:

- Implements a dual-speed Fibre Channel PCS running at either 1.0625 Gb/s or 2.125 Gb/s to support transparent mode transmission, which does not delineate frames or terminate IDLEs.
- Incorporates the RocketIO MGT to provide the SERDES, 8B/10B encoding/decoding, clock recovery and clock correction, bit synchronization, and comma detection functions.
- Implements FC-1 receiver, transmitter, and synchronization as specified in ANSI INCITS X3.230 (1994) FC-PH, clause 12.
- Supports Point-to-Point and Fabric topologies. It supports all non-arbitrated loop port types, such as N-Port, F-Port, and E_Port.
- Implements a 32-bit transparent mode client interface for transmission and reception of data and control characters.
- Supports 32-bit word synchronization on the received data by aligning data to 32-bit ordered set boundaries. Requires that the transmitting data from the client is already aligned properly to the 32-bit ordered set boundaries.
- Supports configuration and monitoring through a generic control/status interface.
- Reports 8B/10B code violations and disparity errors through RX_ER signal on the client interface.
- Supports replacement of an erroneous character with a parameterizable valid data character or special K character. An 8B/10B code violation or disparity error detected in the receiver can result in this erroneous data.
- Supports IDLE insertion on transmission data path if data underrun condition on the client interface is detected. Received IDLEs are passed through to the client.
- Provides a single 53.125 MHz (for 1.0625 Gb/s) or 106.25 MHz (for 2.125 Gb/s) clock to the client.

ESCON PCS

ESCON PCS implemented in CPCS supports transparent mode transmission only.

The IBM ESCON (Enterprise System CONnection) interface is presently being processed to become an ANSI standard interface, known as Single-Byte Command Code Sets CONnection (SBCON) architecture, for computer-to-peripheral interconnect. ESCON data rate, 200 Mb/s, is below the clock data recovery range of the RocketIO MGT PLL. CPCS deploys an over-sampling technique to accommodate such a low data rate on RocketIO MGTs.

The transparent mode FC PCS module provides the following features:

- Implements a 200 Mb/s ESCON/SBCON PCS to support transparent mode transmission, which does not delineate frames or terminate IDLEs.
- Incorporates the RocketIO MGT to provide the SERDES, clock recovery and clock correction, and bit synchronization functions.
- Implements data recovery, code group synchronization, 8B/10B encoding/decoding, and comma detection circuits in FPGA fabric using over-sampling techniques.
- Implements ESCON/SBCON link transmission and reception as specified in ANSI X3.296 (1997) clause 7.
- Implements a 32-bit transparent mode client interface for transmission and reception of data and control characters.
- Supports configuration and monitoring through a generic control/status interface.

- Reports 8B/10B code violations and disparity errors through RX_ER signal on the client interface.
- Supports replacement of an erroneous character with a parameterizable valid data or special K character. An 8B/10B violation or disparity error detected in the receiver may result in this erroneous data.
- Supports IDLE insertion on transmission data path if data underrun condition on the client interface is detected. Note that Link-level facility shall ensure that at least four idle characters are transmitted between frames. Received IDLEs are passed through to the client.
- Provides a single 80 MHz clock to the client.

Clock Module

Figure 7 is a block diagram of the clock module and clock interface (CI). The clock interface connects to the clock module that provides the clock signals to both MGT and CPCS internal logic, usually one for each CPCS port. The number of separate clock inputs depends upon the number of modes supported, as shown in Table 7, page 12. The clock module provides up to eight buffered clock outputs, each selected by the PCS_MODE inputs, and each driving one CPCS port in a system through the clock interface.

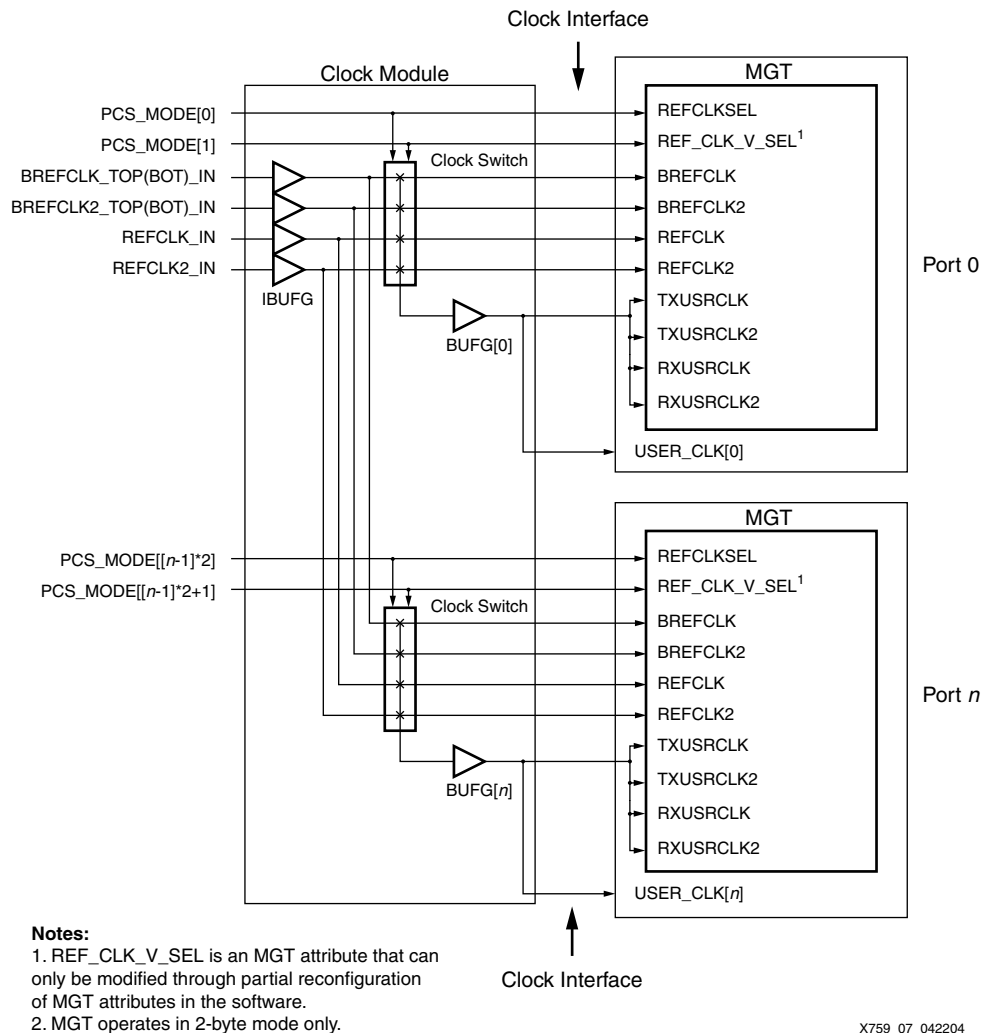


Figure 7: Clock Module Block Diagram

Table 7: Clock Interface Signals

Name	Width	Direction	Clock Freq. (MHz)	Description
PCS_MODE	Nx2	Input	N/A	PCS Mode input from the control/status module
BREFCLK_TOP and BREFCLK_BOT	1	Output	106.25	BREFCLK output to top and bottom MGTs
BREFCLK2_TOP and BREFCLK2_BOT	1	Output	53.125	BREFCLK2 output to top and bottom MGTs
REFCLK	1	Output	62.50	REFCLK output to all MGTs
REFCLK2	1	Output	80	REFCLK2 output to all MGTs
USER_CLK	N	Output	106.25, 53.125, 62.5, or 80	User clock to internal logic in each CPCS port

Note: N is the number of CPCS ports implemented in the system. N = 1, 2,...8

The multi-mode PCS module in the CPCS requires a common user clock running at a dedicated clock rate for each PCS mode in a CPCS port. This common user clock (i.e. USER_CLK) must adapt to the clock rate to accommodate the data rate of each protocol, and must be controlled and adjusted independently to each other among multiple ports. Using a common user clock can reduce the number of global clock buffers (BUFGs) consumed in the overall system, and provide a simple clocking scheme to the MGT and the client logic.

CPCS implements a clock switch for each port, which is a 4-to-1 clock multiplexer between the four reference clocks and the global clock buffer, to conduct rate adaptation among the four PCS modes. As shown in Figure 8, this clock switch contains three 2-to-1 look-up table (LUT) multiplexers (MUXF5 and MUXF6) in half of a configurable logic block (CLB).

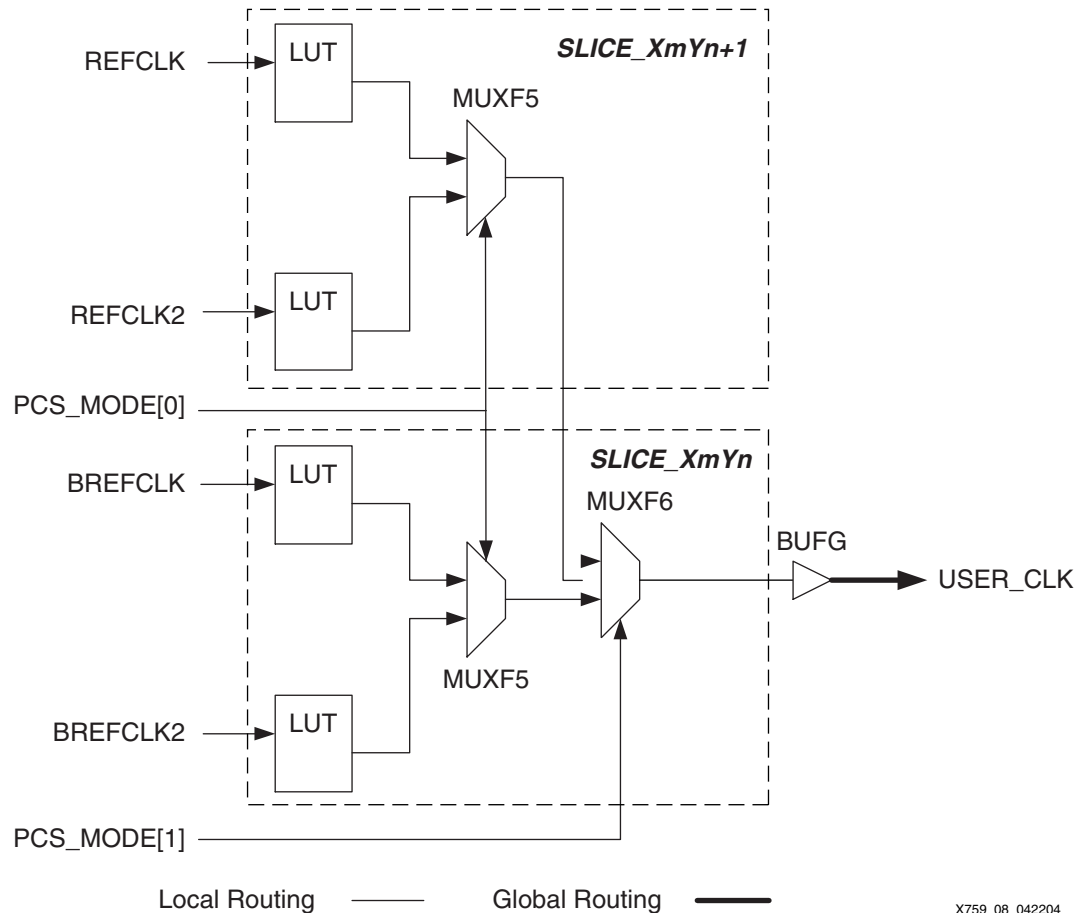
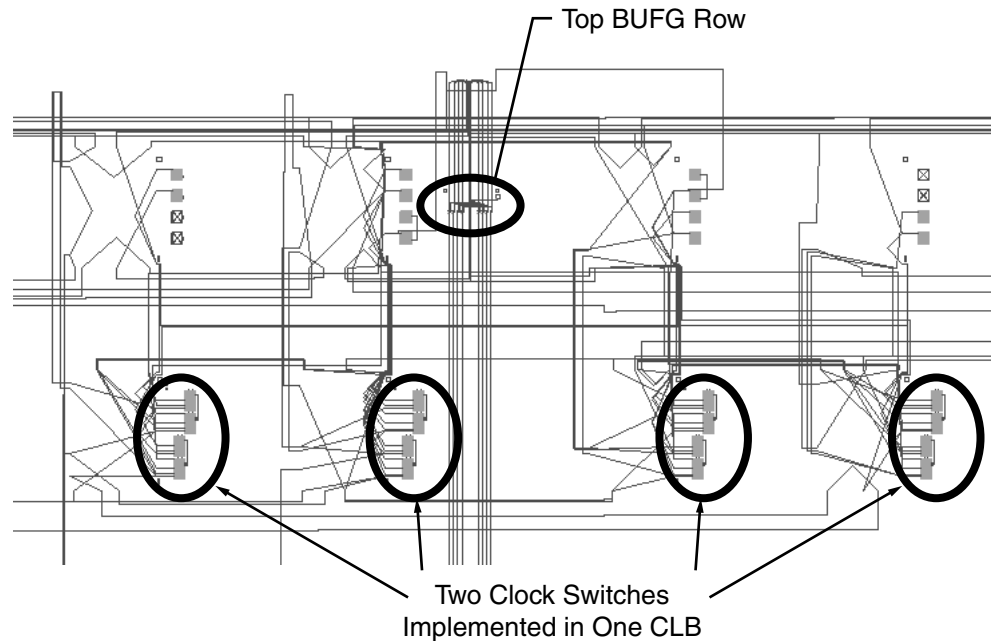


Figure 8: Implementation of Clock Switch

X759_08_042204

The reference clock inputs go through local routing in the clock switch, then come back to BUFG and the global clock network to drive the logic in the PCS module and the client. To minimize the routing delay, a clock switch must use two adjacent slices in a CLB, and must be placed into a CLB that is closest to the BUFG row on the top or bottom of the FPGA die. Figure 9 illustrates an implementation of eight clock switches in four CLBs placed close to the top BUFG row in an FPGA die.

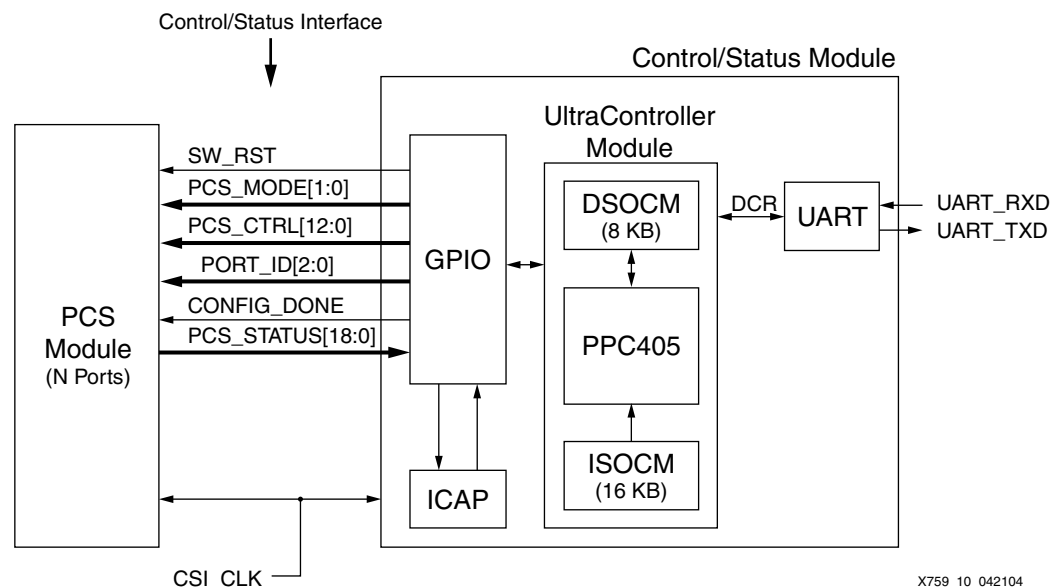


X759_09_042104

Figure 9: Placement of Clock Switch as Shown in FPGA Editor

Control/Status Module

The control/status interface (CSI) connects the PCS module to the control/status module, as shown in Figure 10.



X759_10_042104

Figure 10: Control/Status Module Using the Embedded PPC405 Processor

The control/status module embeds a PowerPC 405 processor (PPC405) using a Xilinx UltraController design that provides a GPIO interface to access internal control and status registers in the CPCS. The control/status module contains the Virtex-II Pro internal configuration access port (ICAP) for partial reconfiguration on the RocketIO MGT attributes. The control/status module also communicates to an external serial port through a UART module.

Table 8 shows the signals for the control/status interface.

Table 8: Control/Status Interface Signals

Name	Width	Direction	Description
CSI_CLK	1	Input	Control/status interface clock. This clock is typically 125 MHz.
CONFIG_DONE	1	Output	Asserted when PCS configuration on a particular port is done. When asserted, PCS_MODE, PCS_CTRL, PORT_ID must contain valid data. In a multi-port system, this signal is asserted multiple times to indicate completion on each port.
PORT_ID	3	Output	Identifies the port to which CONFIG_DONE, PCS_MODE, PCS_CTRL, SW_RST, and PCS_STATUS are associated.
PCS_MODE	2	Output	To instruct the mode of operation on a specified port. Only becomes valid when CONFIG_DONE is asserted.
PCS_CTRL	13	Output	Control signals to the specified port. Only becomes valid when CONFIG_DONE is asserted.
SW_RST	1	Output	Software reset signal to the specified port. Software may hold each individual port in reset when configuration on such port is in progress.
PCS_STATUS	19	Input	Status from the specified port. PORT_ID is used to select which port to read the status.

Table 9 lists the bit definition of the PCS_CTRL bus in the CPCS block.

Table 9: Bit Definition of PCS_CTRL Bus

PCS_CTRL Bits	PCS Mode Applied	Functional Description
[1:0]	All	Select the two loopback test modes on the RocketIO MGT in target CPCS port. Bit 1 is for serial loopback and bit 0 is for internal parallel loopback. If both bits are low, the target CPCS port operates in normal mode.
2	All	Shuts down the RocketIO MGT in target CPCS port. This decreases the power consumption while the MGT is shut down.
3	All	Specifies whether or not to invert the transmitter output of the RocketIO MGT in target CPCS port, i.e., to reverse the polarity on the TXN and TXP lines. Deasserted sets regular polarity. Asserted reverses polarity.

Table 9: Bit Definition of PCS_CTRL Bus (Continued)

PCS_CTRL Bits	PCS Mode Applied	Functional Description
4	All	Specifies whether or not to invert the receiver input of the RocketIO MGT in target CPCS port, i.e., to reverse the polarity on the RXN and RXP lines. Deasserted sets regular polarity. Asserted reverses polarity.
5	ESCON	Reserved
6	1000BASE-X Framed Mode	Enable auto-negotiation function in framed mode 1000BASE-X on target CPCS port. Only usable when HAS_GE_AN = TRUE.
7	1000BASE-X	Asserted to apply framed mode transmission on target CPCS port. Deasserted to apply transparent mode transmission. Framed mode transmission is only available in 1000BASE-X.
[12:8]	N/A	Reserved

[Table 10](#) lists the bit definition of PCS_STATUS in the CPCS block.

Table 10: Bit Definition of PCS_STATUS Bus

PCS_STATUS Bits	PCS Mode Applied	Functional Description
0	All	Asserted to indicate the link-level synchronization is acquired on target CPCS port. Deasserted to indicate the synchronization is not acquired.
[2:1]	All	Reports the PCS mode applied on target CPCS port.
[18:3]	N/A	Reserved

ICAP Module

The CPCS reference design uses the Virtex-II Pro internal configuration access port (ICAP) for partial reconfiguration on the RocketIO MGT attributes. This will allow changing the protocol specific settings of each MGT in the FPGA on the fly. The ICAP module is the fundamental module used to perform in-circuit reconfiguration in Virtex-II Pro devices. The ICAP module is used to access the device configuration registers as well as to transfer configuration data using the Slave SelectMAP™ protocol. Please refer to UG012, "Virtex-II Pro and Virtex-II Pro X FPGA User Guide, Chapter 4: Configuration" for more details about the Slave SelectMAP mode.

Table 11 lists all ports on the ICAP module implemented in the CPCS. The ICAP_CCLK synchronizes all reading and writing of the ICAP data for reconfiguring the FPGA.

Table 11: ICAP Ports

Name	Direction	Description
ICAP_CCLK	Input	ICAP configuration clock. This clock can be driven either by a free running oscillator or a controlled signal. The maximum frequency of this clock is 50 MHz ⁽¹⁾ .
ICAP_CE	Input	ICAP port enable for write/read. Active low.
ICAP_WRITE	Input	ICAP write/read select. (write = 0, read = 1)
ICAP_IN[7:0]	Input	ICAP data input bus. Bit 7 is considered the MSB of each byte.
ICAP_OUT[7:0]	Output	ICAP data output bus.
ICAP_BUSY	Output	When ICAP_CE is asserted, indicates when the ICAP can accept another byte from ICAP_IN. If ICAP_BUSY is low, the ICAP reads the data bus on the next rising ICAP_CCLK edge where both ICAP_CE and ICAP_WRITE are asserted low. If ICAP_BUSY is High, the current byte is ignored and must be reloaded on the next rising ICAP_CCLK edge when ICAP_BUSY is low. This normally occurs when the ICAP ABORT Sequence is in progress ⁽²⁾ . When ICAP_CE is not asserted, ICAP_BUSY is in a high impedance state.

Notes:

1. Refer to DS083, *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet, Module 3: DC and Switching Characteristics* for details on configuration timing.
2. Refer to UG012, *Virtex-II Pro and Virtex-II Pro X FPGA User Guide, Chapter 4: Configuration* for more details about SelectMAP Abort Sequence.

Note: Important Settings

1. The CPCS reference design requires the FPGA configuration mode NOT to be Boundary-Scan, i.e., the configuration pin setting (M2:M0) cannot be 101, because the Boundary-Scan configuration mode will disable the ICAP interface. If JTAG will be used as the primary configuration method, select another mode pin setting (recommendation is Slave Serial, M2:M0 = 111) to avoid disabling the ICAP interface. JTAG configuration will still be available because it overrides other means of configuration, and the MGT reconfiguration in CPCS will function as intended.
2. When implementing the CPCS reference design using Xilinx ISE tool, the BitGen PERSIST option must be set to "NO" in order for the ICAP interface to take control of the device configuration logic. If PERSIST=YES, then the configuration logic is NOT accessible through the ICAP interface and the MGT reconfiguration in CPCS will not function properly.

The CPCS reference design supports two use models of ICAP.

1. By default, the CPCS implements the PPC405 based UltraController and ICAP in the Control/Status module. The software running on the PPC405 processor accesses the 32-bit GPIO interface. Part of the GPIO interface directly connects to the ICAP. Therefore, the software can perform bit banging on the ICAP ports, especially to generate a controlled ICAP_CCLK. The ICAP module and accompanying software will easily suffice to enable MGT reconfiguration flow.
2. The CPCS also provides an external processor interface directly connected to the ICAP. This use model allows using an external processor and accompanying software to drive the ICAP. The ICAP_CCLK must be a controlled clock signal from the processor if the processor also drives other ICAP signals. User can adopt the software API provided in the CPCS reference design for accessing the ICAP from the processor.

Figure 11 shows the ICAP read timing when using controlled ICAP_CCLK.

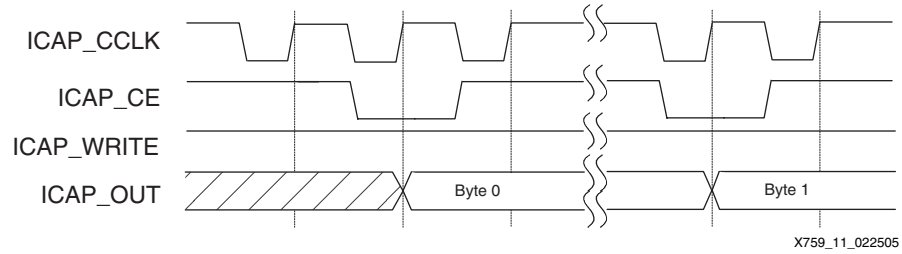


Figure 11: ICAP Read Timing with Controlled ICAP Clocking

Figure 12 shows the ICAP write timing when using controlled ICAP_CCLK.

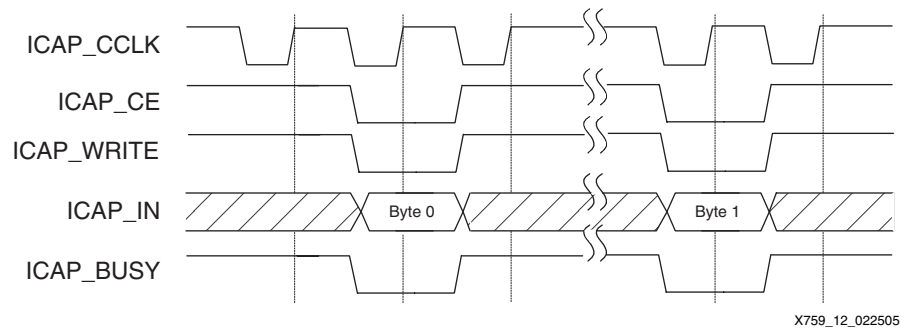


Figure 12: ICAP Write Timing with Controlled ICAP Clocking

Figure 13 shows the ICAP abort timing when using controlled ICAP_CCLK. As described in UG012, *Virtex-II Pro and Virtex-II Pro X FPGA User Guide, Chapter 4: Configuration*, an ABORT is an interruption in the SelectMAP read or write which occurs when the state of RDWR_B on the SelectMAP interface changes while CS_B is asserted. Same scenario also applies to the ICAP module since it uses a subset of the SelectMAP protocol. After the ABORT sequence completes, the user must resynchronize the device. After resynchronizing, configuration can resume by sending the last configuration packet that was in progress when the ABORT occurred, or configuration can be restarted from the beginning.

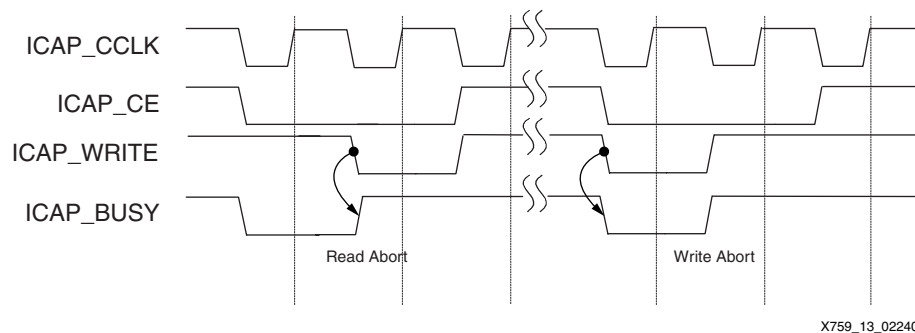


Figure 13: ICAP Read/Write Abort with Controlled ICAP Clocking

Since the ICAP module is consistent with the SelectMAP interface, user can refer to DS083, *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet, Module 3: DC and Switching Characteristics* as a reference to ICAP configuration timing.

External Processor Interface

The CPCS provides an option to use an external processor in place of the embedded PPC405 processor. When the user selects this option, the UltraController and UART modules are removed from the system. Users should use SW_RD_DATA and SW_WR_DATA GPIO ports to connect to an external processor, as shown in Figure 14.

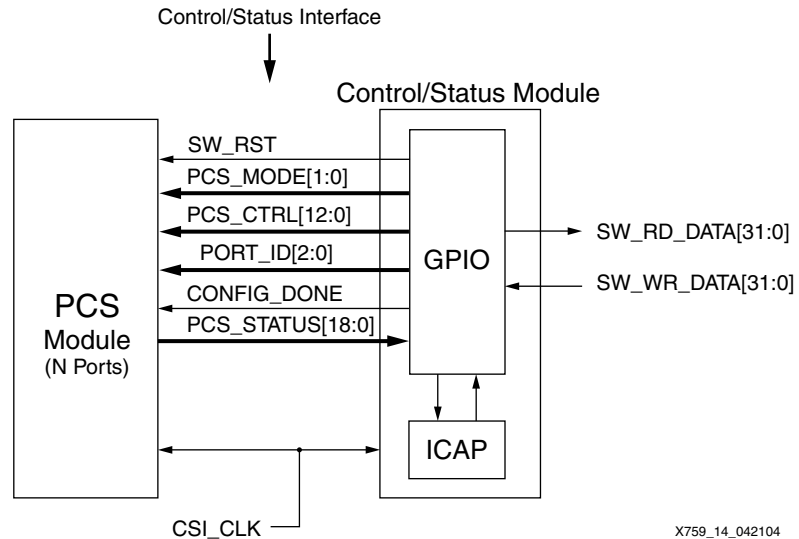


Figure 14: Control/Status Module Using External Processor

Table 12 lists the bit definition of the SW_RD_DATA port.

Table 12: Bit Definition of SW_RD_DATA Port

SW_RD_DATA Bits	Description
[31:24]	Connect to ICAP data output port (ICAP_OUT)
23	Connect to ICAP busy signal (ICAP_BUSY)
[22:20]	Reserved
[19]	Report the value of the USE_MGT_CRC parameter to the software. If USE_MGT_CRC is set to true, this bit is high.
[18:0]	Connect to PCS_STATUS[18:0] on control/status interface

Table 13 lists the bit definition of the SW_WR_DATA port.

Table 13: Bit Definition of SW_WR_DATA Port

SW_WR_DATA Bits	Description
[31:24]	Connect to ICAP data input port (ICAP_IN)
23	Connect to ICAP clock input port (ICAP_CCLK)
22	Connect to ICAP clock enable (ICAP_CE)
21	Connect to ICAP write enable input (ICAP_WRITE)
20	Connect to CONFIG_DONE on control/status interface
[19:18]	Connect to PCS_MODE[1:0] on control/status interface
[17:15]	Connect to PORT_ID[2:0] on control/status interface
[14]	Connect to SW_RST on control/status interface
[13]	Reserved
[12:0]	Connect to PCS_CTRL[12:0] on control/status interface

Most of bits on the SW_WR_DATA and SW_RD_DATA buses are asynchronous signals except for ICAP signals, such as ICAP_IN, ICAP_OUT, ICAP_BUSY, ICAP_CE, ICAP_WRITE and ICAP_CCLK, which should be synchronous to the ICAP clock (ICAP_CCLK). Figure 15 shows the timing of bit groups on the external processor interface. Table 14 lists the critical timing parameters.

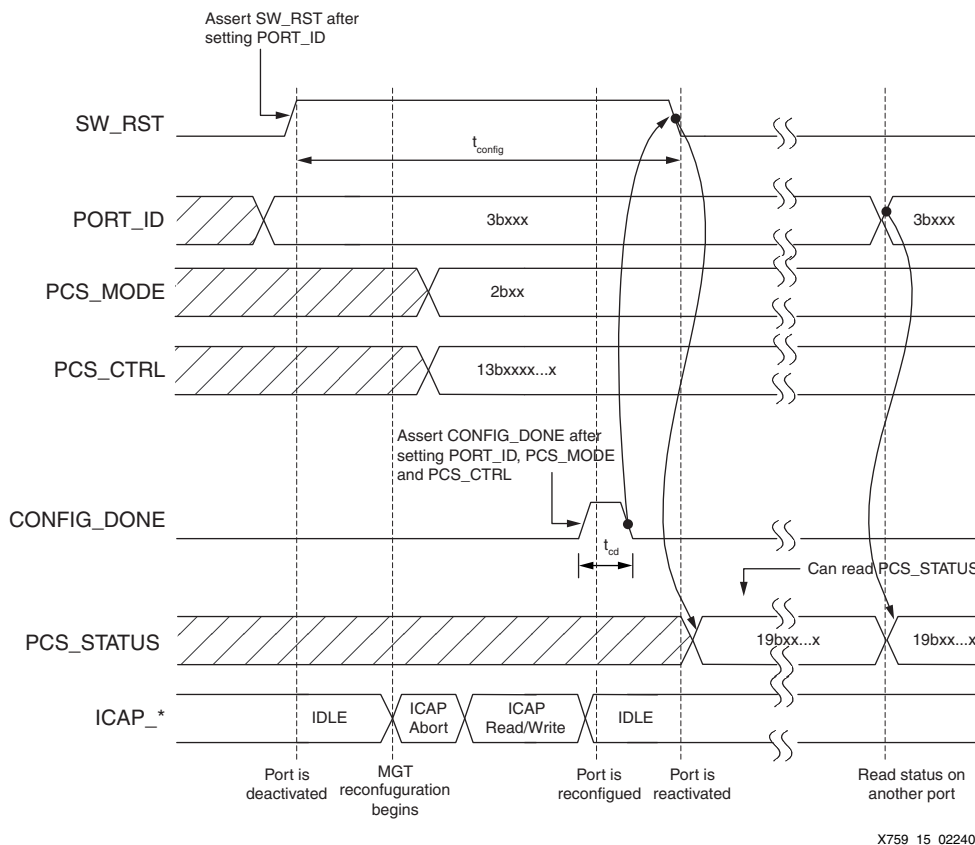


Figure 15: External Processor Interface Timing

Table 14: External Processor Interface Timing Parameter

Symbol	Description	Value
Tcd	Minimum pulse width of CONFIG_DONE. CONFIG_DONE is sampled at the rising edge of CSI_CLK.	$\geq \text{CSI_CLK period} * 2$. Note: CSI_CLK is typical 125 MHz.
Tconfig	Port configuration duration, starting from assertion of SW_RST, ending at the deassertion of SW_RST.	Typical 26 μs .

Hardware Configuration Parameters

Available implementation options of CPCS operation are specified as hardware configuration parameters to the CPCS at compile time, as shown in the following tables. [Table 15](#) contains the list of general configuration parameters that apply to the entire CPCS block.

Table 15: General Configuration Parameters

Parameter	Value	Description
HAS_GE_PCS	Boolean: true or false	When true (default), implement the PCS for 1000BASE-X in each CPCS port; otherwise, remove the logic for 1000BASE-X from CPCS block.
HAS_FC_PCS	Boolean: true or false	When true (default), implement the PCS for Fibre Channel in each CPCS port; otherwise, remove the logic for Fibre Channel from CPCS block.
HAS_ESCON_PCS	Boolean: true or false	When true (default), implement the PCS for ESCON in each CPCS port; otherwise, remove the logic for ESCON from CPCS block.
NUM_PORTS	Integer: 1 to 8	Specify the number of ports implemented in CPCS block.
IBUFG_INSERTION	Boolean: true or false	When true (default), instantiate IBUFG and/or IBUFGDS for clock inputs; otherwise, do not instantiate IBUFG and IBUFGDS inside CPCS block.
BUFG_INSERTION	Boolean: true or false	When true (default), instantiate BUFG for GMII_CLK and CSI_CLK. When false, do not instantiate BUFG for these two clocks in the CPCS.
USE_EXT_CSI_CLK	Boolean: true or false	When false (default), the clock for control/status interface is generated inside CPCS block using the buffered GMII clock. When true, the clock for control/status interface is supplied from an external clock input.
USE_EXT_GMII_CLK	Boolean: true or false	When false (default), the 125MHz GMII clock is generated inside CPCS block using a DCM. When true, the GMII clock is supplied from an external clock input.
USE_ASYNC_GMII_CLK	Boolean: true or false	When false (default), remove asynchronous FIFOs in 1000BASE-X PCS. When true, implement asynchronous FIFOs in 1000BASE-X PCS to support using an external GMII_CLK that is asynchronous to the MGT user clocks.
USE_EXT_PROCESSOR	Boolean: true or false	When false (default), the UltraController™ module with embedded PPC405 processor is implemented in CPCS. When true, uses external processor.
PORT0_INIT, PORT1_INIT, ...PORT7_INIT	16-bit binary. For example: 0010000000000000	CPCS port initialization value. Bits 14 and 13 are to set default PCS_MODE value on the specific port. Bits 12 down to 0 are to set default PCS_CTRL value on the specific port. Bit 15 is reserved. Each of these initialization values is overridden by PCS_MODE and PCS_CTRL inputs when CONFIG_DONE is asserted high on CSI on a specific port. See “Control/Status Module,” page 13 .
CHIPSCOPE_EN	Boolean: true or false	When false (default), do not include the embedded ChipScope Pro ILA core. When true, attach the ChipScope Pro ILA core to the specified CPCS port for in-system debugging.
CHIPSCOPE_PORT	Integer: 0 to NUM_PORTS-1	Specify which CPCS port is connected to ChipScope Pro ILA core. The ILA core is clocked by the user clock (USER_CLK) on this port.

Table 16 lists the configuration parameters for the 1000BASE-X PCS module.

Table 16: 1000BASE-X PCS Configuration Parameters

Parameter	Transmission Mode	Value	Description
HAS_GE_AN	Framed	Boolean: true or false	When true (default), implement auto-negotiation for Gigabit Ethernet; otherwise, remove auto-negotiation logic for Gigabit Ethernet.
USE_MGT_CRC	Framed	Boolean: true or false	When true (Default), enable RocketIO transceiver's CRC logic for Gigabit Ethernet. When false, disable RocketIO transceiver's CRC logic. See details in "Hardware Functional Description," page 8.
GE_ER_REPLACE_CHR	Transparent	9-bit binary. For example: 11111110	Error replacement character used in the receiver to replace a received data character that contains 8B/10B code violation or disparity error. The replacement character can be a valid 8B/10B data character or a valid 8B/10B special character (K character). The most significant bit on this parameter indicates the type of the character. If it is 1, the character is specified as a K character; otherwise, it is a data character.

Note: Use these parameters only when HAS_GE_PCS is set to true.

Table 17 lists the configuration parameters for ESCON PCS module.

Table 17: ESCON PCS Configuration Parameters

Parameter	Value	Description
ESCON_ER_REPLACE_CHR	9-bit binary. For example: 11111110	Error replacement character used in the receiver to replace a received data character that contains 8B/10B code violation or disparity error. The replacement character can be a valid 8B/10B data character or a valid 8B/10B special character (K character). The most significant bit on this parameter indicates the type of the character. If it is 1, the character is specified as a K character; otherwise, it is a data character.

Note: Use this parameter only when HAS_ESCON_PCS is set to true.

Table 18 lists the configuration parameters for Fibre Channel PCS module.

Table 18: Fibre Channel PCS Configuration Parameters

Parameter	Value	Description
FC_ER_REPLACE_CHR	9-bit binary. For example: 11111110	Error replacement character used in the receiver to replace a received data character that contains 8B/10B code violation or disparity error. The replacement character can be a valid 8B/10B data character or a valid 8B/10B special character (K character). The most significant bit on this parameter indicates the type of the character. If it is 1, the character is specified as a K character; otherwise, it is a data character.

Note: Use this parameter only when HAS_FC_PCS is set to true.

Table 19 lists the implementation parameters that apply to the entire CPCS block.

Table 19: Implementation Parameters

Parameter	Value	Description
GT0_LOC,GT1_LOC,...GT7_LOC	String of 14 characters. For example: "GT_X0Y0"	Use to define the MGT location to place the MGT associated with a specific port in CPCS. These parameters also determine which BREFCLK inputs (top or bottom) should be connected. Characters in this string should be left aligned. Consult Xilinx ISE 6 Software Manuals for available MGT location constraints.
BUFG0_LOC, BUFG1_LOC, ... BUFG7_LOC	String of 14 characters. For example: "BUFGMUX0S" or "BUFGMUX1P"	Use to define the BUFG location to place the BUFG component instantiated in a specific CPCS port. The location of the BUFG in each port is determined by the placement of the MGT in this port. If a port instantiates a MGT on the top bank of the FPGA, this port should also use a BUFG on the top bank of the FPGA for best timing performance. Characters in this string should be left aligned. Consult Xilinx ISE 6 Software Manuals for available BUFG location constraints.
MUX0_LOC, MUX1_LOC, ... MUX7_LOC	String of 14 characters. For example: "SLICE_X67Y17 4"	Use to define the slice location to place the CLB that is used to construct a clock switch in the clock module for a specific port. A clock switch is a 4-to-1 clock multiplexer implemented using two adjacent slices in a CLB. This CLB must be placed close to the BUFG row on the top or the bottom of a FPGA die to minimize the skew. These parameters are device dependent and must correlate to BUFG*_LOC parameters. Characters in this string should be left-aligned.
ENALIGN_REG0_LOC, ENALIGN_REG1_LOC, ... ENALIGN_REG7_LOC	String of 14 characters. For example: "SLICE_X15Y16 9"	Use to define the slice location to place the ENPCOMMAALIGN and ENMCOMMAALIGN flip-flops. These flip-flops must be placed near the associated MGT. These parameters are device dependent and must correlate to GT*_LOC parameters. Consult the RocketIO User Guide - "SERDES Alignment" for details.

Notes:

1. When changing a target FPGA device, user needs to modify these device dependent parameters, such as GT*_LOC, MUX*_LOC and ENALIGN_REG*_LOC.
2. Please consult the Excel sheet (CPCS_imp_param_table.xls) supplied in the reference design for proper settings on these parameters for different FPGA devices.

Clock Management

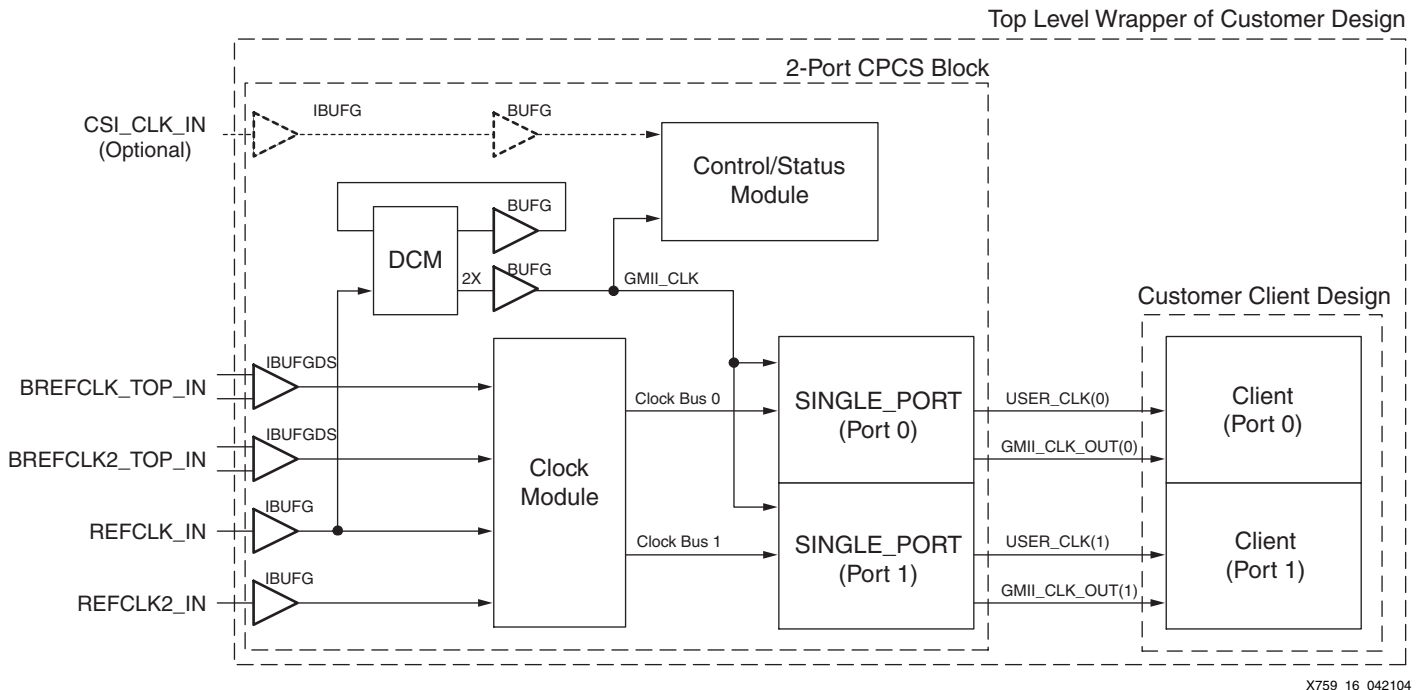
Reference Clock Inputs

The four reference clock inputs into the CPCS block, REFCLK_IN, REFCLK2_IN, BREFCLK_TOP(BOT)_IN, and BREFCLK2_TOP(BOT)_IN, are generated from external clock sources with four different clock frequencies that correlate to the four supported PCS modes. These reference clocks connect to the (B)REFCLK and (B)REFCLK2 ports of the RocketIO MGTs, and also connect to the clock module to generate the USER_CLK for each CPCS port. The USER_CLK connects to the TXUSRCLK(2) and RXUSRCLK(2) on the RocketIO MGT as well as the internal logic and transparent mode client interface in each CPCS port.

The BREFCLK(2)_TOP(BOT)_IN clock inputs use dedicated routing resources. If MGTs on the top bank of a Virtex-II Pro FPGA are to be used in CPCS, BREFCLK(2)_TOP_IN on the top of the chip must be used. The same rule applies to MGTs and BREFCLK(2)_BOT_IN clock inputs at the bottom. When both top and bottom MGTs are used in CPCS (e.g., four ports use top MGTs and four ports use bottom MGTs), both BREFCLK(2)_TOP_IN and BREFCLK(2)_BOT_IN must be connected.

Clocks to Client Logic

Figure 16 illustrates basic clock management with an instance of a multiple-port CPCS block, using two ports as an example. In this example, only two MGTs on the top bank are used in the CPCS. Client signals on the transparent mode client interface of the CPCS block are synchronous to the USER_CLK. Port 0 should use USER_CLK(0). Port 1 should use USER_CLK(1). Client signals on the framed mode client interface (GMII) of the CPCS block are synchronous to the GMII_CLK_OUT.



X759_16_042104

Figure 16: Basic Clock Management

DCM and BUFG Usage

The RocketIO MGTs inside the CPCS block operate in 2-byte mode and at full rate. This operation mode allows each MGT to accept a single buffered clock (USER_CLK) driving its TXUSRCLK(2) and RXUSRCLK(2) inputs and does not require use of a digital clock manager (DCM). Multiple MGTs can operate at independent rates in CPCS without using up DCM resources in Virtex-II Pro FPGAs.

For transparent mode transmission, each CPCS port consumes only one global clock buffer (BUFG), which resides in the clock module.

For framed mode transmission, an additional BUFG is used to buffer the GMII clock. The GMII_CLK can be generated from the 62.5 MHz REFCLK_IN input using a DCM as shown in Figure 16, page 23. However, this deploys an extra BUFG for the clock feedback on the DCM. By changing the parameter USE_EXT_GMII_CLK, the CPCS supports using an external GMII clock input (GMII_CLK_IN) to save a BUFG, as shown in Figure 17. If the external GMII clock is asynchronous to the 62.5 MHz REFCLK_IN input, the user can set the USE_ASYNC_GMII_CLK to true. This option will implement a shallow asynchronous FIFO in 1000BASE-X PCS module for each CPCS port.

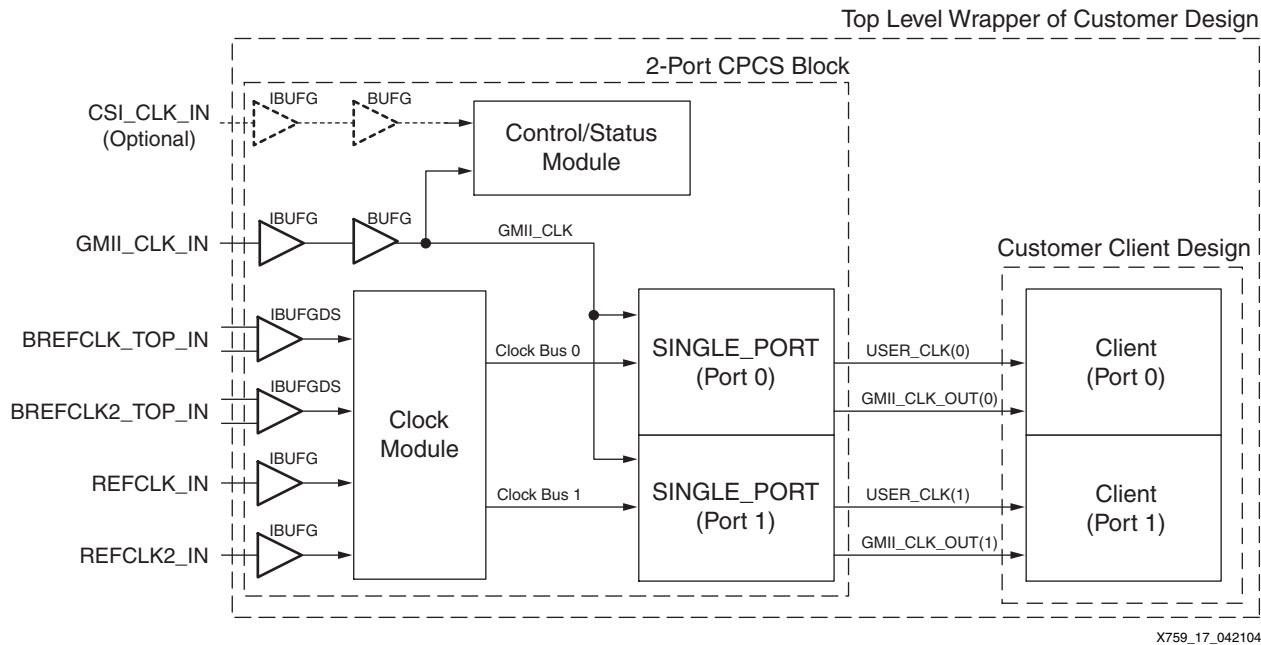


Figure 17: Clock Management Using External GMII Clock

The clock to the control/status module can share the buffered GMII clock. By changing the USE_EXT_CSI_CLK parameter, the CPCS can use an external CSI clock input (CSI_CLK_IN) that deploys an extra IBUFG and a BUFG. This CSI clock is typically 125 MHz.

Table 20 lists the BUFG counts in a multi-port CPCS with different hardware configurations.

Table 20: BUFG Counts in Multi-Port CPCS

Hardware Configuration Parameter			BUFG Count
PORT_NUM	USE_EXT_GMII_CLK	USE_EXT_CSI_CLK	
N	TRUE	TRUE	N+2
N	TRUE	FALSE	N+1
N	FALSE	TRUE	N+3
N	FALSE	FALSE	N+2

Note: N is the number of ports implemented in the CPCS block. N = 1, 2,...8

IBUFG and BUFG Insertion

The user can remove IBUFG instances in the CPCS using the IBUFG_INSERTION parameter. When IBUFG_INSERTION is set to FALSE, the user can instantiate IBUFGDS primitives for the differential reference clocks (BREFCLK*_IN, BREFCLK2*_IN) outside the CPCS block, then feed the singled-ended clock to CPCS.

The user can remove BUFG instances for GMII_CLK and CSI_CLK in the CPCS using the BUFG_INSERTION parameter. When these BUFGs are removed from the CPCS block, they must be instantiated in the customer design outside the CPCS.

Software Functional Description

Configuration of CPCS

Software configuration on the CPCS is primarily used for configuring the MGT attributes for the different PCS modes. In addition, different clock sources must be configured for different PCS modes under software control. Each CPCS port operates independently of other CPCS ports in the device, so that different ports on the same device can be configured to a different PCS mode on the fly.

The user instructs the software configuration process through the terminal control. Hence a UART interface is implemented in the control/status module.

In a multi-port CPCS block, configuration on each port is done sequentially, indicated by the CONFIG_DONE pulse signal on the CSI. When this signal is asserted, the software must also present the valid data to the PCS_MODE, PCS_CTRL, and PORT_ID buses.

Each port must go through three stages to complete the configuration process, as listed below.

1. MGT Attribute Selection

The CPCS applies in-circuit partial reconfiguration of MGT attributes through the Virtex-II Pro internal configuration access port (ICAP). Using this solution, CPCS software can modify MGT attributes at run time to program each port into a specific PCS mode. These attributes are listed in [Table 21](#).

Table 21: MGT Attribute Values

Attributes	1000BASE-X (GbE)	Fibre Channel (1G and 2G)	ESCON/SBCON
REF_CLK_V_SEL	0	1	0
SERDES_10B	FALSE	FALSE	TRUE
RX_DECODE_USE	TRUE	TRUE	FALSE
CLK_COR_SEQ_LEN	2	4	4
CLK_COR_SEQ_1_1	00110111100	00110111100	10000000011
CLK_COR_SEQ_1_2	00001010000	00010010101	11111111111
CLK_COR_SEQ_1_3	00000000000	00010110101	11111111100
CLK_COR_SEQ_1_4	00000000000	00010110101	10011110000
CLK_COR_SEQ_2_USE	FALSE	TRUE	TRUE
CLK_COR_SEQ_2_1	00000000000	00110111100	11111111100
CLK_COR_SEQ_2_2	00000000000	00000110101	10000000000
CLK_COR_SEQ_2_3	00000000000	00010001010	10000000011
CLK_COR_SEQ_2_4	00000000000	00001010101	11100001111
CLK_COR_KEEP_IDLE	FALSE	FALSE	TRUE

Table 21: MGT Attribute Values (Continued)

Attributes	1000BASE-X (GbE)	Fibre Channel (1G and 2G)	ESCON/SBCON
CLK_COR_REPEAT_WAIT	1	2	2
ALIGN_COMMA_MSB	FALSE	FALSE	TRUE
COMMA_10B_MASK	1111111000	1111111000	1111111111
MCOMMA_10B_VALUE	1100000000	1100000000	1111111100
PCOMMA_10B_VALUE	0011111000	0011111000	0000000011
RX_CRC_USE	TRUE/FALSE	FALSE	FALSE
TX_CRC_USE	TRUE/FALSE	FALSE	FALSE
CRC_FORMAT	ETHERNET	FIBRE_CHAN	USER_MODE
TX_DIFF_CTRL	400/500/600/700/800	400/500/600/700/800	400/500/600/700/800
TX_PREEMPHASIS	0/1/2/3	0/1/2/3	0/1/2/3

Note: Gray area can be reconfigured independent to the PCS Mode

2. Clock Selection

With a selected PCS mode, the dedicated reference clock for such PCS mode must be selected for the MGT and the client logic. The CPCS control/status module conducts the following actions in both hardware and software with regards to clock selection on each CPCS port:

- a. Outputs a select signal (PCS_MODE) to the REFCLKSEL port on the MGT.
- b. Reconfigures the REF_CLK_V_SEL attribute on the MGT through ICAP.
- c. Outputs a select signal (PCS_MODE) to the clock switch in the clock module. Selects the common user clock (USER_CLK) from one of the reference clocks. Uses this common user clock to drive the TXUSRCLK(2) and RXUSRCLK(2) on the MGT, and the client logic.

Table 22 lists the clock sources and frequencies for four PCS modes.

Table 22: Clock Selection

	1000BASE-X	Fibre Channel 1G	Fibre Channel 2G	ESCON
PCS_MODE[1:0]	2b00	2b11	2b10	2b01
Clock Source	REFCLK_IN	BREFCLK2_TOP(BOT)_IN	BREFCLK_TOP(BOT)_IN	REFCLK2_IN
Clock Frequency	62.5 MHz	53.125 MHz	106.25 MHz	80 MHz

3. Selection of Protocol-Specific Features

With selected PCS mode from the user interface, the control/status module will first control each CPCS port to enter the correct PCS mode. This operation includes enabling selected PCS logic and disabling non-selected PCS logic, and setting the multiplexer on the data paths among multiple PCS modules. The control/status module then sends out control signals to each CPCS port through the PCS_CTRL bus to toggle protocol specific functions. Note that each CPCS port can be controlled independently through a dedicated PCS_CTRL bus, therefore operates differently to each other.

Status Gathering

After configuration, the software can switch to status gathering mode. The software can only gather status from one port at a time. It can scan through different ports and read their status sequentially through the PCS_STATUS bus.

Software Operation Flow

Figure 18 shows the software operation flow diagram applied in the CPCS reference design.

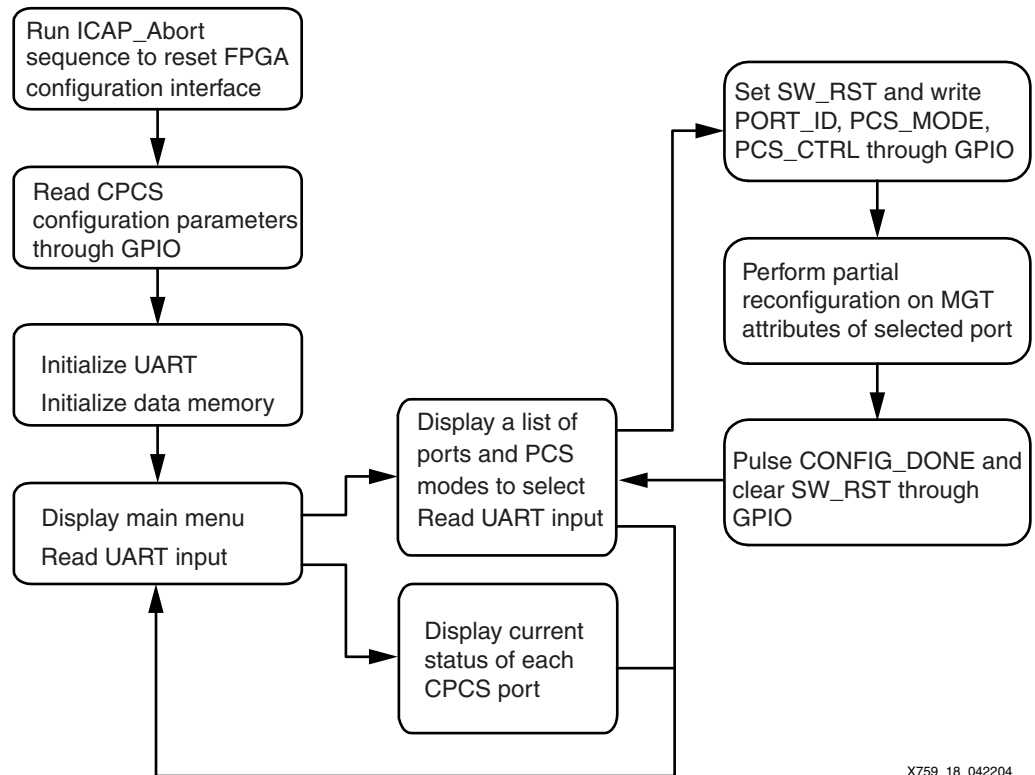


Figure 18: Operation Flow Diagram of CPCS Software

X759_18_042204

Bit Banging

The software running on the PPC405 or an external processor can perform bit banging on the ICAP ports through the GPIO. For example, the software can toggle the ICAP_CCLK on the ICAP to generate a controlled ICAP_CCLK.

Below is a section of C codes that shows how to perform bit banging on the GPIO interface while accessing the ICAP.

```
Xuint32 Read_Sequence(void) /* Read data from ICAP */
{
    Xuint32 temp_word;

    /* First run through a CCLK cycle */
    GPIO_WR_ICAP(0x00600000); /* Assert ICAP_CCLK low */
    GPIO_WR_ICAP(0x00E00000); /* Assert ICAP_CCLK high */

    /* Turn on the ICAP_CE (active_low) and continue to clock the ICAP. */
    GPIO_WR_ICAP(0x00200000); /* ICAP_CCLK=0, ICAP_CE=0, ICAP_WE=1 */
    GPIO_WR_ICAP(0x00A00000); /* ICAP_CCLK=1, ICAP_CE=0, ICAP_WE=1 */

    /* Turn off the ICAP_CE (active low). */
    GPIO_WR_ICAP(0x00600000); /* ICAP_CCLK=0, ICAP_CE=1, ICAP_WE=1 */
    GPIO_WR_ICAP(0x00E00000); /* ICAP_CCLK=1, ICAP_CE=1, ICAP_WE=1 */

    temp_word = GPIO_RD_ICAP();
    return temp_word;
}

void Write_Sequence(Xuint32 data_word) /* Write data to ICAP */
{
    Xuint32 temp_word;

    /* First run through a CCLK cycle */
    GPIO_WR_ICAP(0x00600000); /* ICAP_CCLK low, CE high, WE high */
    GPIO_WR_ICAP(0x00E00000); /* ICAP_CCLK high, CE high, WE high */

    /* Set the data and turn on the ICAP_CE and ICAP_WE (active low).
       Continue to clock the ICAP. */
    GPIO_WR_ICAP(data_word & 0xFF0FFFFFF); /* data, CCLK=0, ICAP_CE=0,
    ICAP_WE=0 */
    GPIO_WR_ICAP(data_word | 0x00800000); /* data, CCLK=1, ICAP_CE=0,
    ICAP_WE=0 */

    /* Clear the data and turn off the ICAP_CE and ICAP_WE (active low). */
    GPIO_WR_ICAP(0x00600000);
    GPIO_WR_ICAP(0x00E00000);
}
}
```

Software Configuration Parameters

The header file (includes.h) in the EDK project directory defines configuration parameters for the CPCS software, as listed in [Table 23](#).

Table 23: Software Configuration Parameters

Parameter	Value	Description
NUM_OF_PORTS	Integer: 1 to 8	Specify the number of ports implemented in CPCS reference design. This software parameter must match the NUM_PORTS hardware parameter listed in Table 15, page 20 .
MGT_NUM_ARRAY	Array of integers	Map the hardware port number to software MGT number (MGT_NUM). Data element at index n of this array should store the MGT_NUM for port n in CPCS.

Each MGT in an FPGA is assigned a unique number (MGT_NUM) in the software in order to identify such MGT and perform partial reconfiguration. [Table 24](#) lists the MGT_NUM with its associated MGT location in some of the Virtex-II Pro FPGA devices. Such numbering methods can be applied to other FPGA devices. By mapping between the software MGT_NUM with the hardware port number, the software can properly identify the physical location of an MGT in each port and complete partial reconfiguration on its attributes.

Table 24: Association of Software MGT_NUM Value to MGT Physical Location

MGT_NUM	MGT Physical Location	XC2VP7, XC2VP20, XC2VP30	XC2VP40	XC2VP50	XC2VP70, XC2VP100
0	GT_X0Y0	X	X	X	X
1	GT_X0Y1	X	X	X	X
2	GT_X1Y0	X	X	X	X
3	GT_X1Y1	X	X	X	X
4	GT_X2Y0	X	X	X	X
5	GT_X2Y1	X	X	X	X
6	GT_X3Y0	X	X	X	X
7	GT_X3Y1	X	X	X	X
8	GT_X4Y0		X	X	X
9	GT_X4Y1		X	X	X
10	GT_X5Y0		X	X	X
11	GT_X5Y1		X	X	X
12	GT_X6Y0			X	X
13	GT_X6Y1			X	X
14	GT_X7Y0			X	X
15	GT_X7Y1			X	X
16	GT_X8Y0				X
17	GT_X8Y1				X
18	GT_X9Y0				X
19	GT_X9Y1				X

Performance

Latency

Table 25 and Table 26 show the latency through the CPCS transmission and receiving side components, respectively.

Table 25: Latency Through CPCS Transmission Side Components

Component	Hardware Configuration	Latency
RocketIO MGT	USE_MGT_CRC = TRUE	14.5 USER_CLK cycles
	USE_MGT_CRC = FALSE	8.5 USER_CLK cycles
Framed mode 1000BASE-XPCS module	USE_ASYNC_GMII_CLK = FALSE	4 GMII_CLK cycles
	USE_ASYNC_GMII_CLK = TRUE	10 GMII_CLK cycles
Transparent mode 1000BASE-XPCS module	USE_ASYNC_GMII_CLK = FALSE	5 USER_CLK cycles
	USE_ASYNC_GMII_CLK = TRUE	9 USER_CLK cycles
Transparent mode 1G/2G FC PCS module	N/A	5 USER_CLK cycles
Transparent mode ESCON PCS module	N/A	4 USER_CLK cycles

Table 26: Latency through CPCS Receiving Side Components

Component	Hardware Configuration	Latency
RocketIO MGT	N/A	24 to 25 USER_CLK cycles
Framed mode 1000BASE-XPCS module	USE_ASYNC_GMII_CLK = FALSE	8 GMII_CLK cycles
	USE_ASYNC_GMII_CLK = TRUE	12 GMII_CLK cycles
Transparent mode 1000BASE-XPCS module	USE_ASYNC_GMII_CLK = FALSE	6 USER_CLK cycles
	USE_ASYNC_GMII_CLK = TRUE	10 USER_CLK cycles
Transparent mode 1G/2G FC PCS module	N/A	6 USER_CLK cycles
Transparent mode ESCON PCS module	N/A	11 USER_CLK cycles

Resource Utilization

Table 27 shows the resource utilization of the CPCS reference design.

Table 27: Resource Utilization for Virtex-II Pro

Modules	LUTs	Flip-Flops	BRAMs (1 BRAM = 2KB)	BUFGs	DCMs	PPC405s
N-Port CPCS Block						
ESCON PCS	364 x N	266 x N	If N is even, 1.5 x N If N is odd, 1.5 x (N + 1)	0	0	0
1000BASE-X PCS ⁽¹⁾	260 x N to 708 x N	361 x N to 417 x N	0	0	0	0
FC PCS	196 x N	258x N	0	0	0	0
Global Logic (Timers, counters, multiplexers)	164 x N	81 x N	0	0	0	0
Sub Total	984 x N to 1432 x N	966 x N to 1022 x N	If N is even, 1.5 x N If N is odd, 1.5 x (N + 1)	0	0	0
Clock Module and Interface ⁽²⁾	N	66	0	N + 1 to N + 3	0 to 1	0
Control/Status Module and Interface ⁽³⁾	0 to 156	0 to 219	0 to 12	0	0	0 to 1
Total	984 x N to 1432 x N + 156	966 x N + 66 to 1022 x N + 285	If N is even, 12 + 1.5 x N If N is odd, 12 + 1.5 x (N + 1)	N + 1 to N + 3	0 to 1	0 to 1

Notes:

- Resources used depend on the settings of HAS_GE_AN, USE_MGT_CRC, and USE_ASYNC_GMII_CLK parameters.
- Resources used depend on the settings of USE_EXT_CSI_CLK and USE_EXT_GMII_CLK parameters.
- Software code size is 24K bytes. Resources used depend on the setting of the USE_EXT_PROCESSOR parameter.
- N is the number of CPCS ports implemented in the system. N = 1, 2,...8

Reference Design

Design Hierarchy

The directory structure of the CPCS reference design is shown below. This tree does not show temporary directories that are generated during the design implementation.

```

<CPCS_ROOT>                                Readme file, environment setup scripts
|
|--- src                                    Design source codes
|   |--- vhdl
|
|--- ppc_sub_system                          EDK project files, scripts
|   |--- __xps                               Option files for various EDK tools
|   |--- implementation                     Generated netlists for the EDK project
|   |--- pcores                             Customized IP cores for the EDK project
|   |--- sw                                  Software codes
|       |--- data                            Device dependent data files
|
|--- test
|   |--- func_sim                            Scripts, test vector files for simulation
|       |--- vhdl
|   |--- testbench                           testbench source codes
|       |--- vhdl                             tester source codes
|       |--- verilog
|
|--- build
|   |--- vhdl                                Scripts for synthesis and implementation
|       |--- syn                             Synthesis project files and pre-compiled
|                                           netlists
|
|--- chipscope                               ChipScope netlists and project file
|
|--- demo                                    CPCS demo bitstreams

```

The CPCS source code is provided under the **<CPCS_ROOT>/src/vhdl** directory, which does not include the source codes for ESCON, Fibre Channel, and 1000BASE-X PCS designs. The netlists for the ESCON, Fibre Channel, and 1000BASE-X PCS blocks are pre-generated and placed under the **<CPCS_ROOT>/build/vhdl/syn** directory. The source codes for these designs are not available under the CPCS reference design license.

The EDK project for the control/status module using the embedded PPC405 processor is located under the **<CPCS_ROOT>/ppc_sub_system** directory. There are two EDK MHS files in this directory. The **<CPCS_ROOT>/ppc_sub_system/system_1cpu.mhs** file targets FPGAs with a single embedded PPC405 core (e.g. XC2VP7). The **<CPCS_ROOT>/ppc_sub_system/system_2cpu.mhs** file targets FPGAs with dual embedded PPC405 cores (e.g. XC2VP50). Select the proper MHS file and rename it into **<CPCS_ROOT>/ppc_sub_system/system.mhs** before compiling the EDK project.

In order to support MGT partial reconfiguration on different Virtex-II Pro devices, the CPCS software provides a list of data files (under the **<CPCS_ROOT>/ppc_sub_system/sw/data** directory) that contains device-dependent data structures for MGT reconfiguration. Users should select data files in the **MGT_reconfig.c** (under the **<CPCS_ROOT>/ppc_sub_system/sw** directory) by enabling a proper header file for the target device.

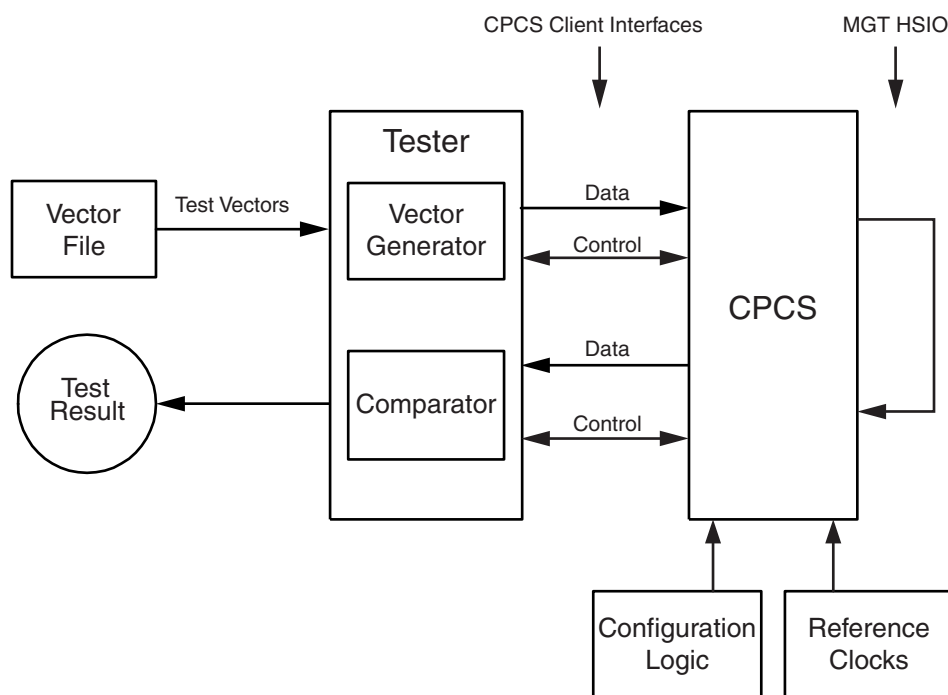
The software can also be used with an external processor. Refer to **<CPCS_ROOT>/ppc_sub_system/sw/sw_user_guide.txt** for further information.

Design Implementation and Simulation Instruction

For a complete description of the various design processes (environment setup, design tools, design configuration, synthesis, EDK project compilation, implementation, and simulation) on the CPCS reference design, please refer to the readme file in the provided xapp759.zip file.

Design Simulation Testbench

The CPCS reference design provides a simulation suite under the `<CPCS_ROOT>/test` directory, which contains a series of testbenches, test vectors, and scripts dedicated to each PCS mode. Since the simulation model for the ICAP module in Virtex-II Pro FPGA is currently not available, it is not applicable to simulate the entire CPCS design with the MGT partial reconfiguration functions in the software. Therefore, simulation requires the CPCS to be configured into a specific PCS mode through the external processor interface, and the protocol-dependent MGT attributes be preset properly. The CPCS testbench omits the control/status module and replaces it with a simple configuration logic driving the external processor interface. As shown in Figure 19, a simulation testbench contains a tester module that is connected to the framed mode and transparent mode client interfaces on the CPCS block. The MGT high-speed I/O (HSIO) ports are connected in loopback mode on the CPCS interface. The tester module reads a test vector file and generates data and control signals to the TX interfaces on the CPCS. The tester module also reads the data from the RX interfaces on the CPCS, compares it to the expected data and generates the test result.



X759_19_042104

Figure 19: CPCS Simulation Testbench

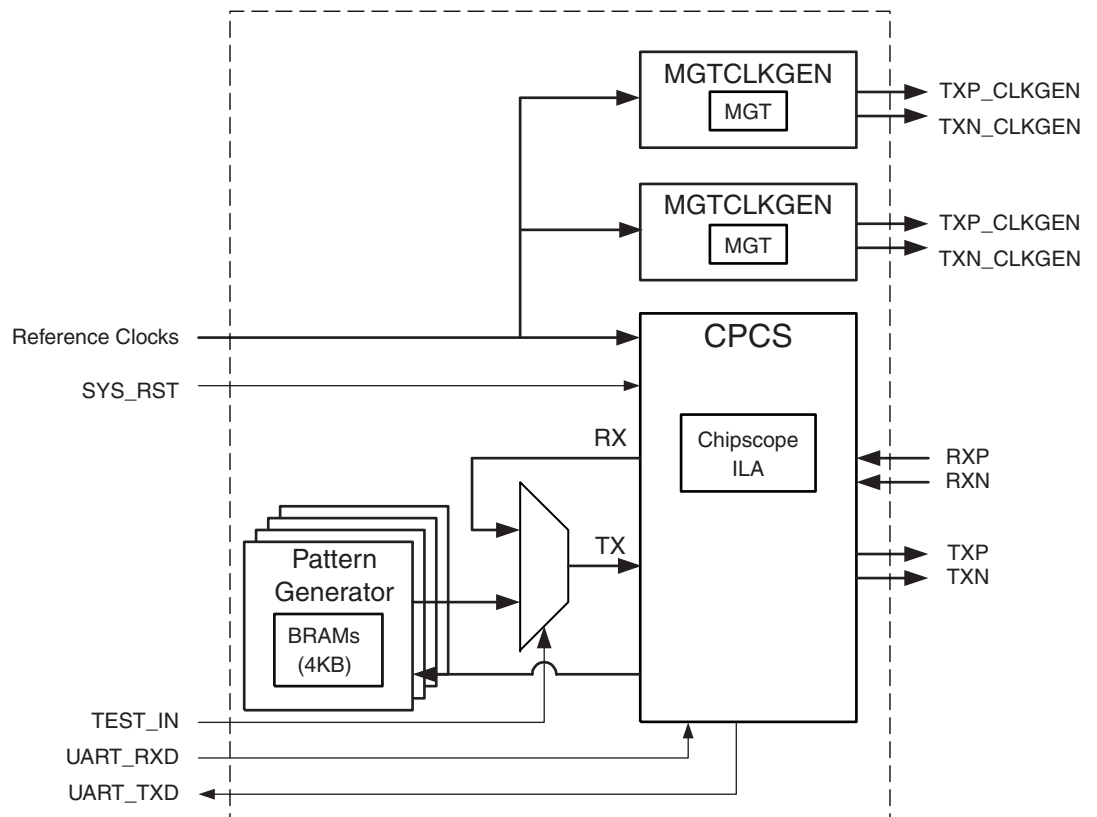
Hardware Test and CPCS Demonstration

The CPCS reference design provides a test design (called CPCS_TEST) to test and demonstrate CPCS on Xilinx ML323 and ML321 development platform boards. The ML323 board has a XC2VP50 Virtex-II Pro FPGA device in an FF1152 package. The ML321 board has a XC2VP7 Virtex-II Pro FPGA device in an FF672 package. For detailed usage of the ML323 and ML321 platforms, please refer to the *Virtex-II Pro ML320, ML321, ML323 Platform User Guide*.

Figure 20, page 34 shows a block diagram of the CPCS_TEST design. The CPCS_TEST design implements a four-port CPCS block. As listed in Table 28, page 34, the CPCS_TEST design places two MGTs (two CPCS ports) on the top bank of the FPGA, and two MGTs on the bottom bank of the FPGA. The CPCS_TEST design uses one of the embedded PPC405 cores in the control/status module to run the software. The CPCS_TEST supports two test modes: the pattern generation (PATGEN) mode that initiates test patterns through the pattern

generator module connected to each CPCS port, and the echo mode that reflects received data back to the transmission on the CPCS client interfaces. By switching between the PATGEN mode and the echo mode, users can validate CPCS_TEST with an external traffic analyzer, or perform a self test on ML323 or ML321 board. The CPCS_TEST design also instantiate two ChipScope™ ILA cores attached to the MGT interfaces of port 0 and port 1 in CPCS. User can monitor the data traffic on these two CPCS ports through the ChipScope analyzer.

The pattern generator module in the CPCS_TEST allocates 4 KB block SelectRAM memories (BRAMs) for each CPCS port, which contains the test vector. Users can run a PERL script (<CPCS_ROOT>/src/vhdl/PATGEN/gen_pat_table.pl) to convert the test vector file used in simulation into lines of BRAM initialization codes, and use them to initialize the BRAMs in the pattern generator. This solution allows users to share test vectors between simulation and hardware testing on the CPCS.



X759_20_042104

Figure 20: Block Diagram of CPCS_TEST Design

Table 28: Port Distribution of CPCS_TEST on ML323/ML321 Platform

CPCS Port Number	MGT Location	CPCS Port Number	MGT Location
0	GT_X0Y1	2	GT_X0Y0
1	GT_X1Y1	3	GT_X1Y0

The ML323 or ML321 board has four onboard crystals and two sets of SMA clock inputs, which can be used to provide reference clocks to the CPCS design, as listed in Table 29. To provide reference clocks to the SMA clock inputs without using external clock generators, the CPCS_TEST design implements two MGT clock generator modules (MGTCLKGEN), one uses a MGT on the top bank, one uses a MGT on the bottom bank. The MGT in the top MGTCLKGEN module is clocked from the top differential oscillator (53.125 MHz). Such MGT generates 53.125 MHz differential clock signals on the SMA ports. These clock output signals can be used to drive the differential SMA clock inputs on the bottom of the board, which is the BREFCLK2_BOT_IN inputs to the CPCS. Similarly the bottom MGTCLKGEN module is used to generate the clock for the BREFCLK_TOP_IN inputs to the CPCS. Table 30 lists the location of MGTCLKGEN modules and their clock outputs.

Table 29: Distribution of CPCS Reference Clocks on ML323/ML321 Platform Board

Component	Location on ML323	Reference Clock Connection on CPCS	Clock Frequency
Clock Crystal in the Top Oscillator Socket	X3	REFCLK_IN	62.5 MHz
Clock Crystal in the Bottom Oscillator Socket	X5	REFCLK2_IN	80 MHz
Differential Oscillator on the top of the board	X2	BREFCLK2_TOP_IN	53.125 MHz
Differential Oscillator on the bottom of the board	X4	BREFCLK_BOT_IN	106.25 MHz
Differential SMA Clock on the top of the board	J21 and J23	BREFCLK_TOP_IN	106.25 MHz
Differential SMA Clock on the bottom of the board	J42 and J41	BREFCLK2_BOT_IN	53.125 MHz

Table 30: Distribution of CPCS_TEST MGTCLKGEN Modules

Component	MGT Location	Clock Output Location	Designate Reference Clock Input on CPCS
Top MGTCLKGEN module	ML323: GT_X7Y1 ML321: GT_X3Y1	ML323: MGT11 ML321: MGT9	BREFCLK2_BOT_IN
Bottom MGTCLKGEN module	ML323: GT_X7Y0 ML321: GT_X3Y0	ML323: MGT14 ML321: MGT16	BREFCLK_TOP_IN

Download Reference Design

The CPCS reference design can be downloaded from <http://www.xilinx.com/bvdocs/appnotes/xapp759.zip>

References

1. Xilinx, Inc., UG012: Virtex-II Pro and Virtex-II Pro X FPGA User Guide, <http://www.xilinx.com/bvdocs/userguides/ug012.pdf>
2. Xilinx, Inc., UG024: RocketIO Transceiver User Guide, <http://www.xilinx.com/bvdocs/userguides/ug024.pdf>
3. IEEE, IEEE Std 802.3, 2002 Edition
4. ANSI, X3.230 (1994) Fibre Channel - Physical and Signaling Interface (FC-PH) Specification
5. ANSI, X3.296 (1997) Single-Byte Command Code Sets CONnection Architecture (SBCON) Specification
6. Xilinx, Inc., DS264: Ethernet 1000BASE-X PCS/PMA LogiCORE data sheet, http://www.xilinx.com/ipcenter/catalog/logicore/docs/gig_eth_pcs_pma.pdf
7. Xilinx, Inc., XAPP662: In-Circuit Partial Reconfiguration of RocketIO Attributes, <http://www.xilinx.com/bvdocs/appnotes/xapp662.pdf>
8. Xilinx, Inc., XAPP672: The UltraController Solution: A Lightweight PowerPC Microcontroller, <http://www.xilinx.com/bvdocs/appnotes/xapp672.pdf>
9. Xilinx, Inc., DS083: Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet, <http://www.xilinx.com/bvdocs/publications/ds083.pdf>

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/27/04	1.0	Initial Xilinx release.
03/04/05	1.1	Added test design for ML321 boards. Updated resource utilization.