



XpressPCS IP

Reference Manual

Version 1.0.2 December 2008
Copyright © PLDA 1996-2008

XpressPCS IP

Reference Manual

Documentation Change History

Date	Version Number	Changes
December 2008	1.0.2	<ul style="list-style-type: none">• No documentation changes
September 2008	1.0.1	<ul style="list-style-type: none">• Updated for 5.0 Gbps link speed support
January 2008	1.0.0	<ul style="list-style-type: none">• First release

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks owned by PLDA SA. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by PLDA in good faith. This document is provided "as is" with no warranties whatsoever, including any warranty of merchantability, non infringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample.

This document is intended only to assist the reader in the use of the product. PLDA shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product. Nor shall PLDA be liable for infringement of proprietary rights relating to use of information in this document. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Product Status

The information in this document is final content pertaining to the PLDA XpressPCS IP Core.

Web Address

<http://www.plda.com>

Table of Contents

List of Tables	4
List of Figures	5
Preface	6
About this Document	6
Additional Reading	6
Feedback and Contact Information	7
Chapter 1 XpressPCS IP Architecture	8
1.1 PCI Express Architecture	8
1.2 Physical Layer Architecture	8
1.3 XpressPCS and the PCS Sub-Layer	10
1.4 Physical Media Attachment (PMA) Sub-layer	12
Chapter 2 XpressPCS Interfaces and Signals	13
2.1 Introduction	13
2.2 Global Signals	13
2.3 PIPE Interface Signals	14
2.4 PMA Interface Signals	15
2.5 Test and Control Signals	18
Chapter 3 Implementing XpressPCS	19
3.1 Implementing XpressPCS on a Multi-Lane Link	19
3.1.1 Connecting Per-Lane Signals	19
3.1.2 Connecting Clock and Reset Signals	20
3.1.3 Connecting MAC to PHY Signals	20
3.1.4 Connecting PHY to MAC Signals	21
3.2 Multi-Lane Implementation Example	21

List of Tables

Global clock and reset signals	13
PIPE interface signals	14
PMA interface signals	15
Test and control signals	18
Shared and per-lane PIPE signals	19
Clock and reset signals	20
Description of v4fx125_phyx4_v2 architecture	22

List of Figures

Standard PCI Express Architecture	8
Architecture of the Physical Layer	10
XpressPCS architecture	11
All XpressPCS signals	13
External clock and rest signals	20
v4fx125_phyx4_v2 architecture	21

Preface

About this Document

Intended Audience

This document has been written for design managers, system engineers, and designers who are evaluating or using the PLDA XpressPCS IP.

Scope

This document provides the complete functional description of the PLDA XpressPCS IP.

Typographical Conventions

italic

Highlights important notes or publications

bold

Highlights interface elements.

COURIER NEW

DENOTES TEXT USED IN A CODE EXAMPLE OR A SIGNAL.

Additional Reading

This section lists additional resources from PLDA and third-parties.

PLDA periodically updates its documentation. Please contact PLDA at support@plda.com or check the Web site at <http://www.plda.com> for current versions.

PLDA Publications

Please refer to the following documents for further information:

- *Build History*: The *Build History* lists changes made to the packaging of each build.
- *Revision History*: The *Revision History* lists changes made to the RTL of the Core.

Other Publications

Please refer to the following documents for information on specification standards:

- *PCI Express™ 2.0 Base Specification Revision 1.0*
- *PHY Interface for the PCI Express™ Architecture Revision 2.0*
- *PHY Interface for the PCI Express™ Architecture Revision 1.87*

Feedback and Contact Information

Feedback about this Document

PLDA welcomes comments and suggestions pertaining to this documentation. Please contact PLDA at support@plda.com and provide the following information:

- the title of the document
- the page number to which your comments refer
- a description of your comments

Contact information

Corporate Headquarters

PLDA
Parc club du golf - Bât. 11a
Rue Guilibert
13856 Aix-en-Provence Cedex 3 - France

Tel: USA +1 408 273 4528 - International +33 442 393 600

Fax: +33 442 394 902

Sales

For sales questions, please contact sales@plda.com.

Technical Support

For technical support questions, please contact support@plda.com.

Chapter 1 XpressPCS IP Architecture

The XpressPCS IP is a specially-designed PCS core for use in the PHY sub-layer of PCI Express components. It enables designers to quickly and efficiently integrate PCS logic into their cores.

1.1 PCI Express Architecture

The PCI Express protocol is based on three layers:

- **Transaction layer:** The Transaction Layer is primarily responsible for the assembly and disassembly of Transaction Layer packets (TLPs) and for the storage of configuration information. It also converts received Completion packets into data payloads, updates status information, and is responsible for flow control services, ordering rules, and power management services.
- **Data Link Layer:** The Data Link Layer of a component (along with its counterpart on the other side of a PCI Express Link) is responsible for link management, including: TLP acknowledgement, a retry mechanism in case of a non-acknowledged packet, flow control across the PCI Express link (transmission and reception), power management, CRC generation and CRC checking, error reporting, and logging.
- **Physical Layer:** The Physical Layer is responsible for power management, width and lane negotiation, reset/hot-plug control, 8-bit/10-bit encoding/decoding, scrambling/de-scrambling, embedded clock tuning and alignment, transmission and reception circuit, and elastic buffer and multi-lane de-skew on the receiving side. The frequency is scalable up to 10 Gbps per Lane, the maximum frequency of a Printed Circuit Board (PCB), however, only 5.0 Gbps per Lane is currently defined.

The diagram below shows the standard architecture of a PCI Express component:

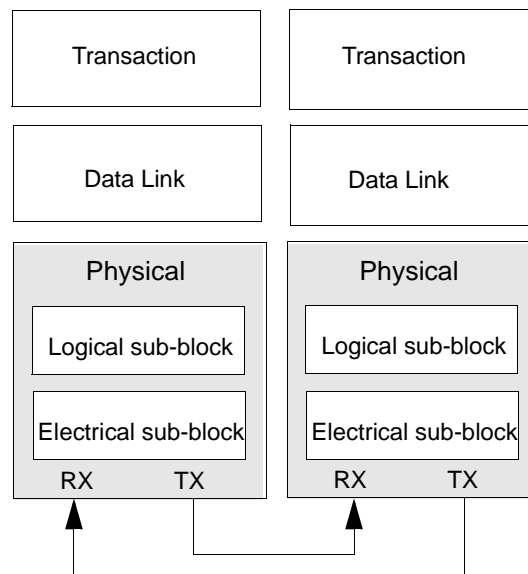


Figure 1: Standard PCI Express Architecture

1.2 Physical Layer Architecture

The Physical Layer is the layer closest to the PCI Express link. It connects to the link through a high-speed SERDES and is responsible for:

- Initializing the link
- Scrambling and 8b/10b encoding of 2.5 Gbps or 2.5/5.0 Gbps per lane
- Serializing and deserializing data

The Physical Layer can be divided into three sub-blocks:

- **Media Access Controller (MAC):** The MAC block includes the Link Training and Status State Machine (LTSSM) and the Scrambling/Descrambling and Multi-Lane de-skew functions.
- **Physical Coding Sub-layer (PCS):** The PCS houses the Rx elastic buffer and performs 8B/10B encoding/decoding. It also performs receiver detection control and loopback.
- **Physical Media Attachment (PMA):** The PMA implements a SERDES, with Clock Data Recover (CDR) and PLL. It also includes the link serializer & deserializer block to transform the data flow rate as described in the following table:

Table 1: Serializer/Deserializer Block Data Flow Rate Transformation

PIPE Width	Rate	PIPE Interface Frequency
8-bit	2.5 Gbps	250 MHz
16-bit	2.5 Gbps	125 MHz
8-bit	5.0 Gbps	500 MHz
16-bit	5.0 Gbps	250 MHz

An example of Physical Layer architecture is shown in the diagram below:

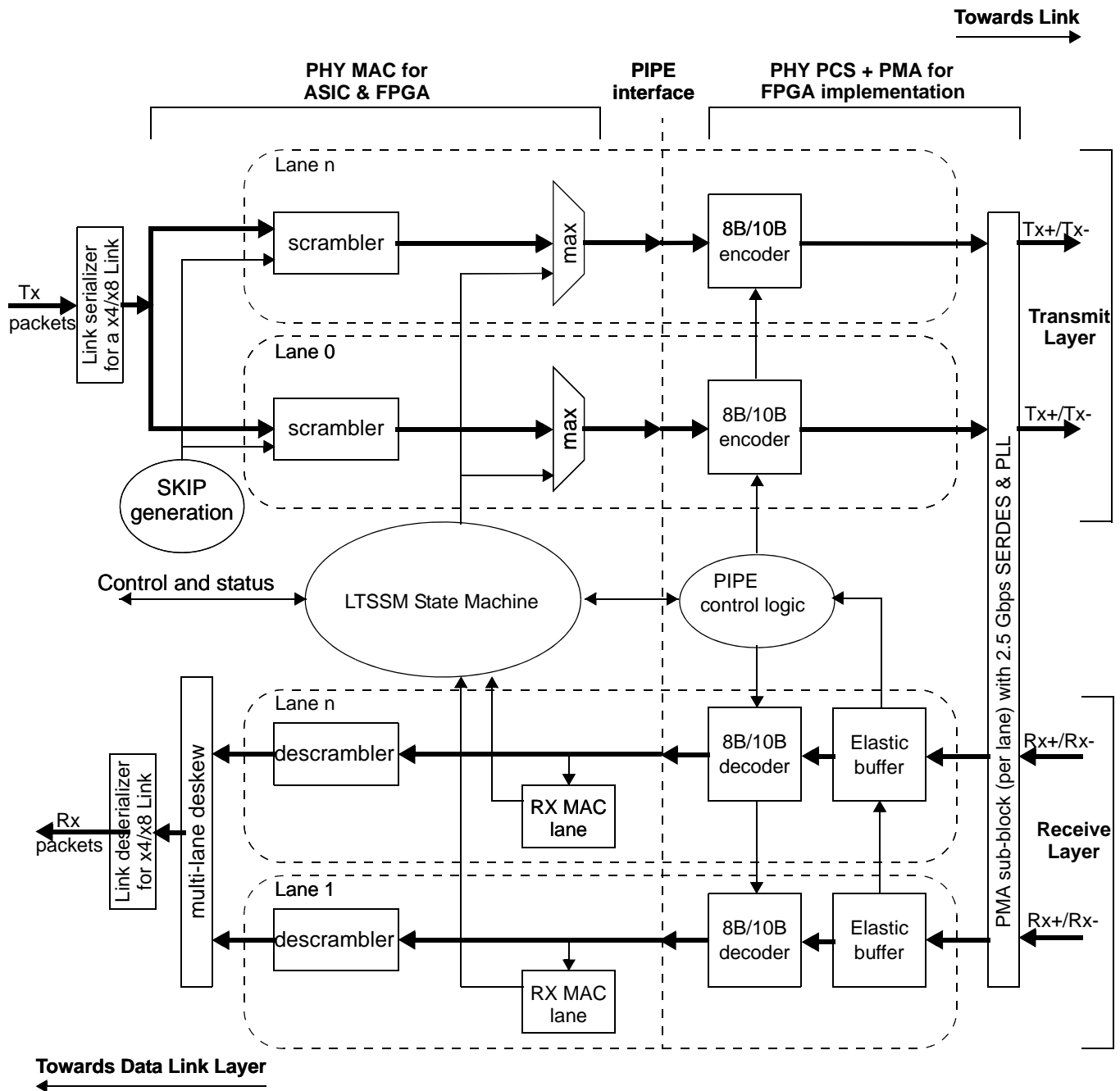


Figure 2: Architecture of the Physical Layer

The Physical Layer integrates both digital and analog elements, prompting Intel® to design the PIPE interface in order to separate the MAC from the other two layers. The Core is compliant with the PIPE interface, enabling integration with other PIPE-compliant PHY layer vendors.

1.3 XpressPCS and the PCS Sub-Layer

The PLDA XpressPCS IP is a fully-functional instance of the PCS sub-layer.

The Physical Coding Sub-layer (PCS) is located between the MAC sub-layer and the PMA sub-layer. Its main function is to communicate between the MAC and PMA layers. It houses the Rx elastic buffer and performs 8B/10B encoding/decoding, as well as receiver detection control and loopback.

The interface between the MAC sub-layer and the XpressPCS uses PIPE-compliant signals, and data is transmitted between the XpressPCS and the PMA using parallel 10-bit (or 20-bit in 125 MHz version) encoding.

The XpressPCS is implemented per lane, so a multi-lane core will implement multiple instances of the XpressPCS. Each XpressPCS contains all PIPE signals which enable a multi-lane link with single PHY or multiple PHY. Multi-lane implementation is described in [Section 3.1: Implementing XpressPCS on a Multi-Lane Link](#).

The following diagram shows the architecture of the XpressPCS:

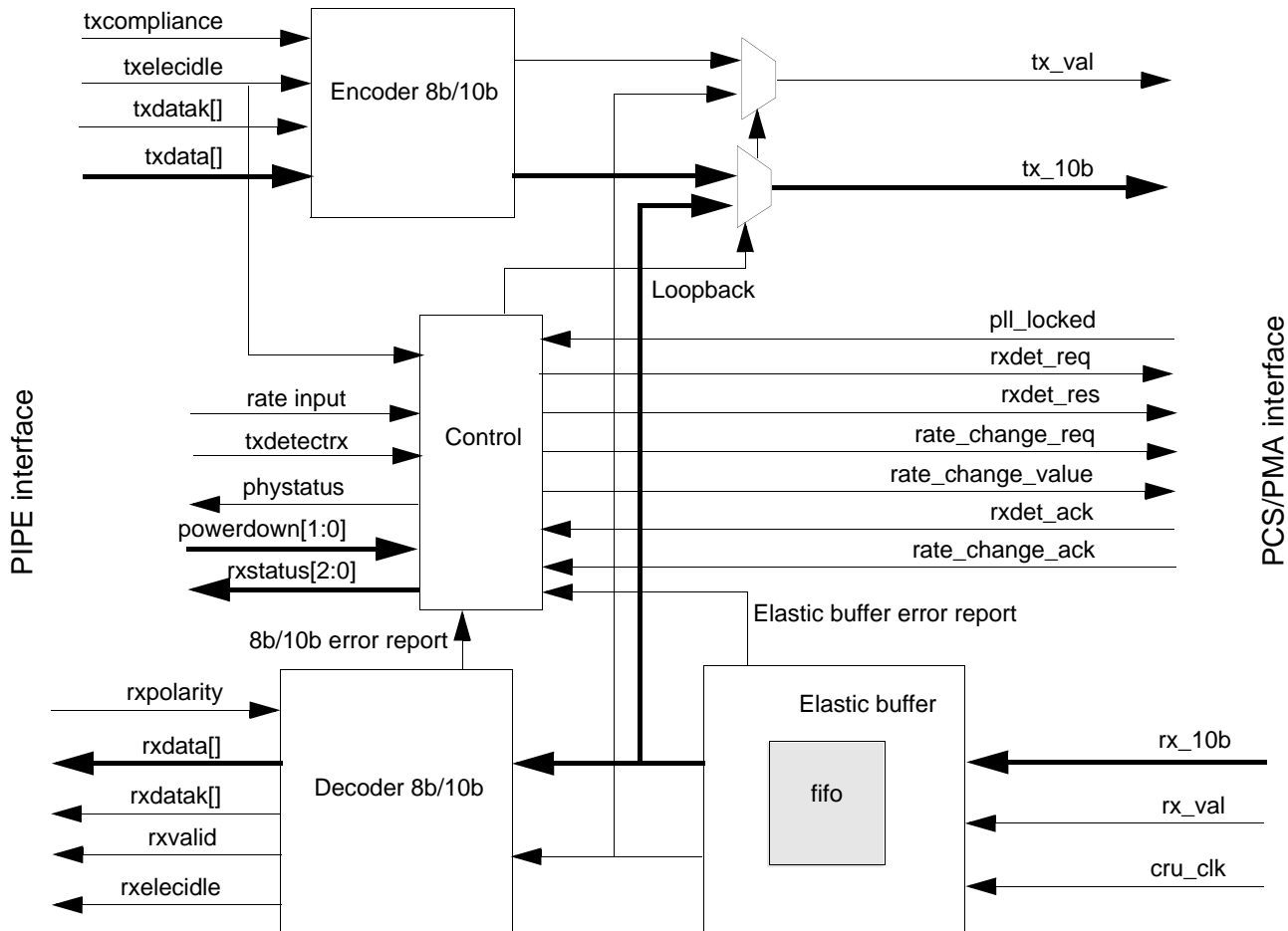


Figure 3: XpressPCS architecture

Each of the above signals are described in [Chapter 2: XpressPCS Interfaces and Signals](#).

The XpressPCS contains the following sub-blocks:

- **Elastic buffer:** The elastic buffer compensates for differences in frequencies between bit rates at the two ends of a link. It synchronizes the received data generated on the Clock Data Recovery clock (`cru_clk`) to the PHY clock (`clk`). Both clocks have the same frequency, but may have different PPMs.

To compensate for the PPM difference, the elastic buffer implements a dual-clock fifo to manage the transmission of SKP ordered sets sent by transmitters to maintain a continuous data flow.

A minimum (almost empty) value and a maximum (almost full) value is defined for the fifo, which is initially filled to its medium value. If the fifo reaches its almost full value and a SKP ordered set is detected at the elastic buffer input, the first SKP symbol is not written to the fifo and is discarded. However, if the fifo is almost empty and a SKP ordered set is detected at the output of the fifo, the fifo read request is deasserted for one clock cycle and a SKP symbol with the same disparity is added to the data flow.

Whenever a SKP symbol is added or removed, the event is reported to the MAC using the `RxStatus[2:0]` signals. The elastic buffer also reports elastic buffer underflow or overflow, as well as the addition or removal of SKP symbols to the Control sub-block, which sends the data to the 8B/10B decoder. The decoder then adds latency clock cycles to the data (the number of clock cycles added depends on the status of the fifo).

Note: For the 16-bit PIPE version of XpressPCS:

When the elastic buffer is almost empty and a SKP ordered set is received, the core stops reading from

the fifo and adds 2 SKP symbols to the data flow (as 2 symbols are received per clock cycle). In this respect, the XpressPCS differs from the PIPE specifications. This functionality is not used when the fifo is almost full, however, as only one SKP symbol can be received in the SKP ordered set, and the COM symbol can be the first or second SKP symbol.

- **8B/10B decoder:** This block performs 8B/10B decoding. It also replaces invalid data with the EDB symbol when errors are detected (disparity, or data or elastic buffer underflow). When the `RECEIVE_POLARITY: RXPOLARITY` signal is set, the 10B data is first inverted before being converted. This block adds one clock cycle latency.
- **8B/10B encoder:** This block performs the 8B/10B encoding of data. When the `TRANSMIT_COMPLIANCE: TXCOMPLIANCE` signal is set, the corresponding data is encoded with negative disparity. This block adds one clock cycle latency.
- **Control:** This block executes control functions between the MAC sub-layer (using the PIPE protocol) and the PMA sub-layer (receiver detection operation). It also reports any errors detected to the MAC and sets the PCS to specific modes such as loopback, or low-power.

1.4 Physical Media Attachment (PMA) Sub-layer

Note: The PMA block is not part of the XpressPCS. The following description is given as information only. For FPGA devices, the PMA block corresponds to the transceiver embedded in the device. Asic customers must create their own PMA or obtain a third-party PMA.

The Physical Media Attachment (PMA) sub-layer of the Physical layer corresponds to the SERDES, PLL and Receiver detection logic. Its main functions are:

The main analog functions are:

- **TX PLL:** The Transmit PLL receives the 100 MHz reference clock and generates the data sent on the link at 2.5/5.0 Gbps as well as the internal 125 MHz, 250 MHz, or 500 MHz core clock 'clk'.
- **Serializer:** this block is located on the transmit side and serializes the 10-bit parallel data at 250 MHz in a 2.5 Gbps link, or 500 MHz in a 5.0 Gbps link.
- **RX PLL:** A Receive PLL is implemented for each lane in order to recover the clock from the received data.
- **Deserializer:** A de-serializer is implemented for each lane in order to provide low-frequency 10-bit parallel data at 250 MHz from the 2.5 Gbps serial input data, or 500 MHz from the 5.0 Gbps data.
- **Electrical Idle detection:** Each lane detects when a differential serial input is in Electrical Idle and provides this information to the PCS sub-layer.
- **Electrical Idle generation:** Each transmitter can force Electrical Idle on the link (no differential output).
- **Beacon generation:** Each transmitter can transmit differential patterns (30 kHz to 300 MHz) to 'wake up' the system.
- **Beacon detection:** Each receiver can detect a differential regular pattern. Auxiliary power is required in the D3 cold state.
- **Receiver detection:** Each transmitter can generate a ramp on its DC common voltage and detecting from the response of the link whether a receiver (100 ohm) is attached or not (high-Z).
- **Powerdown mode:** this function switches some or all the core processes to power-down mode.

Note: Asic customers:

The XpressPCS block does not implement functionality corresponding to beacon generation/detection or to power down states. If you require these features, you should modify the XpressPCS as necessary.

The COMMA alignment function is also considered as being part of the PMA sub-block, so you should check that this function is effectively implemented in your design.

Chapter 2 XpressPCS Interfaces and Signals

2.1 Introduction

The following diagram shows all of the XpressPCS IP signals:

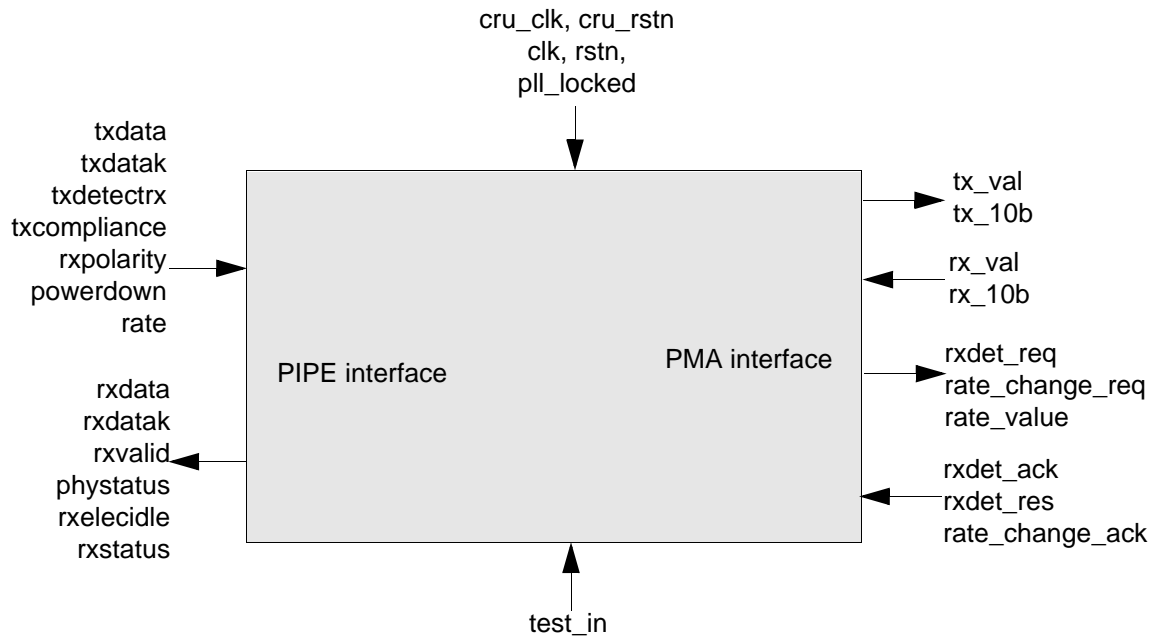


Figure 4: All XpressPCS signals

2.2 Global Signals

The following table describes global clock and reset signals for the XpressPCS:

Table 2: Global clock and reset signals

Signal	I/O	Description
clk	in	PHY Clock: This signal is the PHY Clock used by the elastic buffer. See Section 1.3: XpressPCS and the PCS Sub-Layer for more information.
rstn	in	PHY Clock Reset: This signal is the active-low reset signal associated with CLK : PHY CLOCK. This signal is used in conjunction with CRU_RSTN : DATA RECOVERY CLOCK RESET to reset the Electric Buffer. Both CRU_RSTN and RSTN must be active simultaneously in order to reset the buffer. Deassertion of this signal must be synchronous with CLK : PHY CLOCK. .
cru_clk	in	Clock Data Recovery Clock: This signal is the Clock Data Recovery Clock used by the elastic buffer. See Section 1.3: XpressPCS and the PCS Sub-Layer for more information.
cru_rstn	in	Clock Data Recovery Clock Reset: This signal is the active-low reset signal associated with CLOCK DATA RECOVERY CLOCK : CRU_CLK. This signal is used in conjunction with PHY CLOCK RESET : RSTN to reset the Electric Buffer. Both CRU_RSTN and RSTN must be active simultaneously in order to reset the buffer. Deassertion of this signal must be synchronous with the CLOCK DATA RECOVERY CLOCK : CRU_CLK.

Table 2: Global clock and reset signals

Signal	I/O	Description
pll_locked	in	PLL Locked: Indicates that the PLL is locked, that is, that the PHY clock (clk) is stable. This active-high signal is used to generate the PHY Status signal when in low-power mode.

2.3 PIPE Interface Signals

The XpressPCS is compliant with the 8-bit or 16-bit versions of the PIPE interface:.

Table 3: PIPE interface signals

Signal	I/O	Description
RxData[15:0] for 16-bit interface RxData[7:0] for 8-bit interface]	out	Receive Data: PCI Express data output bus. For the 16-bit interface, 16 bits represent 2 symbols of receive data.
RxDataK[1:0] for 16-bit interface RxDataK for 8-bit interface	out	Receive Data Control: This signal is used for separating control and data symbols <code>RECEIVE_DATA: RXDATA</code> . A value of zero indicates a data byte, a value of 1 indicates a control byte.
rxvalid	out	Receive Valid: This symbol indicates symbol lock and valid data on <code>RECEIVE_DATA: RXDATA</code> and <code>RECEIVE_DATA_CONTROL: RXDATAK</code> .
rxstatus[2:0]	out	Receive Status: This signal encodes receive status and error codes for the receive data stream and receiver detection. See the PIPE specifications for details.
rxpolarity	out	Receive Polarity: This signal instructs the PHY layer to perform a polarity inversion on the 8b/10b receiver decoding block.
rxelecidle	out	Receive Electrical Idle: This signal indicates receiver detection of an electrical idle.
TxData[15:0] for 16-bit interface TxData[7:0] for 8-bit interface	in	Transmit Data: PCI Express data input bus. For the 16-bit interface, 16 bits represent 2 symbols of transmit data.
TxDataK[1:0] for 16-bit interface TxDataK for 8-bit interface	in	Transmit Data Control: This signal is used for separating control and data symbols in <code>TRANSMIT_DATA: TXDATA</code> . A value of zero indicates a data byte, a value of 1 indicates a control byte.
txdetectrx	in	Transmit Detect Receive: This signal is used to instruct the PMA to start a receive detection operation or to begin loopback.
txelecidle0	in	Transmit Electrical Idle: This signal forces the transmit output to electrical idle in all power states. See the PIPE specifications for details.
txcompliance	in	Transmit Compliance: This signal forces the running disparity to negative in Compliance mode (negative COM character). This signal is also used in order to turn off Lanes which are not initialized. In this case, the signal is asserted at the same time as <code>TRANSMIT_ELECTRICAL_IDLE: TXELECIDLE</code> . This signal is set for one clock cycle at the start of the compliance pattern.
powerdown[1:0]	in	Power Down: This signal is responsible for determining the power phase of a transaction (P0, P0s, P1, or P2).

Table 3: PIPE interface signals

Signal	I/O	Description
rate	in	Control the link signaling rate: <ul style="list-style-type: none"> • 0: Use 2.5 GT/s signaling rate • 1: Use 5.0 GT/s signaling rate PIPE implementations that only support the 2.5GT/s signaling rate should hardwire this signal to '0'.
phystatus	out	PHY Status: This signal is used to communicate completion of several PHY requests received from the MAC, including: <ul style="list-style-type: none"> • power down requests • receiver detection requests • rate change requests It is also used to report when the PCLK is locked (stable) after a reset (or P2 entry/exit). See Section 3.1.2 for more information.

Note: With multi-lane links, some PIPE signals are shared across all lanes, and some are generated by the MAC on a per-lane basis. See [Section 3.1: Implementing XpressPCS on a Multi-Lane Link](#) for more information.

2.4 PMA Interface Signals

The XpressPCS implements a receiver detection interface between the PCS and PMA sub-blocks in order to determine if there is a receiver on the link. This interface enables the PCS to signal to the PMA that it has received a detection request from the MAC (using the signal `rxdet_req`). The PMA responds by asserting an acknowledgement signal (`rxdet_ack`) for one clock cycle, then transmitting the result of the receiver detection operation (`rxdet_res`) to the PCS, which transfers both signals to the MAC.

The following table describes all PMA interface signals:

Table 4: PMA interface signals

Signal	I/O	Description
rx_10b[9:0]	in	Receive Data: This signal contains the 10-bit data received from the MAC. It is valid only if <code>RECEIVE_VALID: RX_VAL</code> is asserted.
rx_val	in	Receive Valid: This signal is an active-high signal asserted by the PMA when the data in <code>RECEIVE_DATA: RX_10B</code> is valid. If Electrical Idle is detected, this signal is not asserted.
tx_10b[9:0]	out	Transmit Data: This signal contains the 10-bit data to transmit to the link. It is valid only if <code>TRANSMIT_VALID: TX_VAL</code> is asserted.
tx_val	out	Transmit Valid: This active-high signal is asserted by the PCS when the data in <code>TX_10B: TRANSMIT_DATA</code> is valid. If the data is not valid, the PCS does not assert this signal in order to force the PMA to transmit Electrical Idle.
rxdet_req	out	Receive Detection Request: The PCS asserts this signal when it receives a detection request from the MAC. The signal instructs the PMA to perform a receiver detection operation. This signal is deasserted one clock cycle after the PMA asserts the <code>RECEIVE_DETECTION_ACKNOWLEDGE: RXDET_ACK</code> signal.
rxdet_ack	in	Receive Detection Acknowledge: The PMA asserts this signal for one clock cycle when the receiver detection operation requested by <code>RECEIVE_DETECTION_REQUEST: RXDET_REQ</code> is complete.

Table 4: PMA interface signals

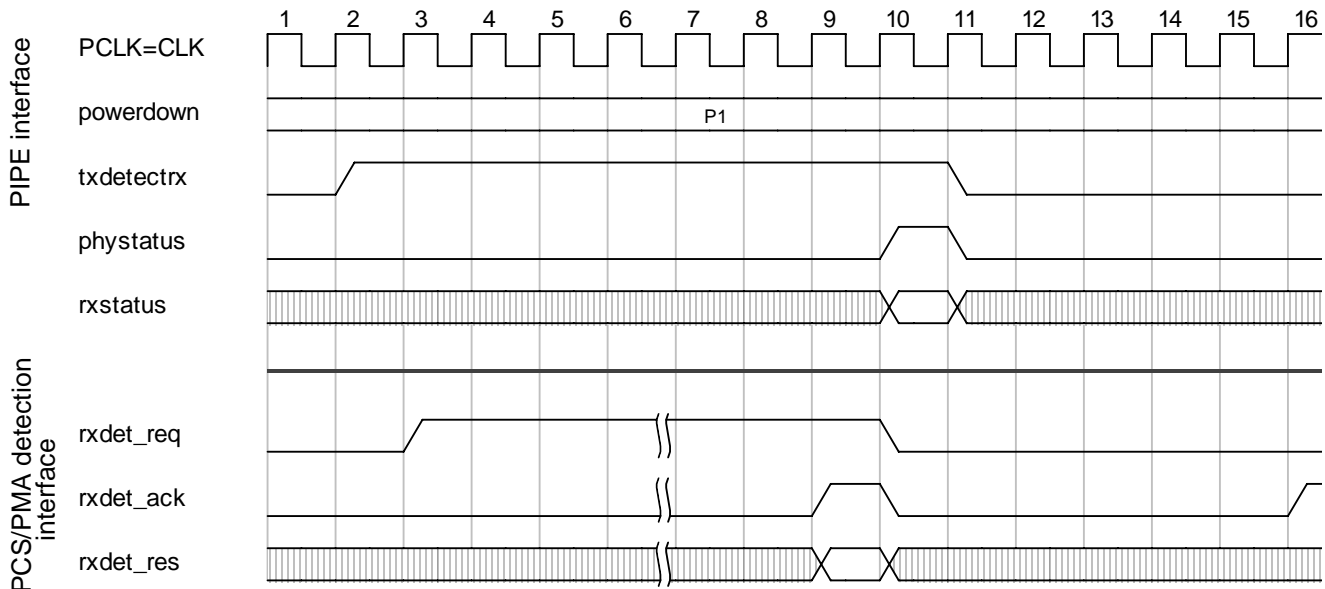
Signal	I/O	Description
rxdet_res	in	Receive Detection Result: This signal reports the result of the receiver detection operation requested by RECEIVE DETECTION REQUEST: RXDET_REQ. The PMA asserts this signal when the RECEIVE DETECTION ACKNOWLEDGE: RXDET_ACK signal is asserted to indicate that a receiver has been detected. If no receiver has been detected, this signal remains low.
rate_change_req	out	Rate Change Request: The PCS asserts this signal when it receives a rate change request from the MAC. This signal instructs the PMA to perform a rate change to the speed indicated by RATE CHANGE VALUE: RATE_VALUE.
rate_value	out	Rate Change Value: Valid when rate_change_req is high. '0' signifies a request to change the speed to 2.5 Gbps, '1' signifies a request to change the speed to 5.0 Gbps.
rate_change_ack	in	Rate Change Acknowledge: The PMA asserts this signal for one clock cycle when the speed change operation requested by RATE CHANGE REQUEST: RATE_CHANGE_REQ is complete and pclk is stable.

The diagram below shows the receiver detection sequence.

Note in particular:

- **clock cycle 2 and 3:** receiver detection request is transmitted and received.
- **clock cycle 9:** receiver detection request is acknowledged and the result provided.

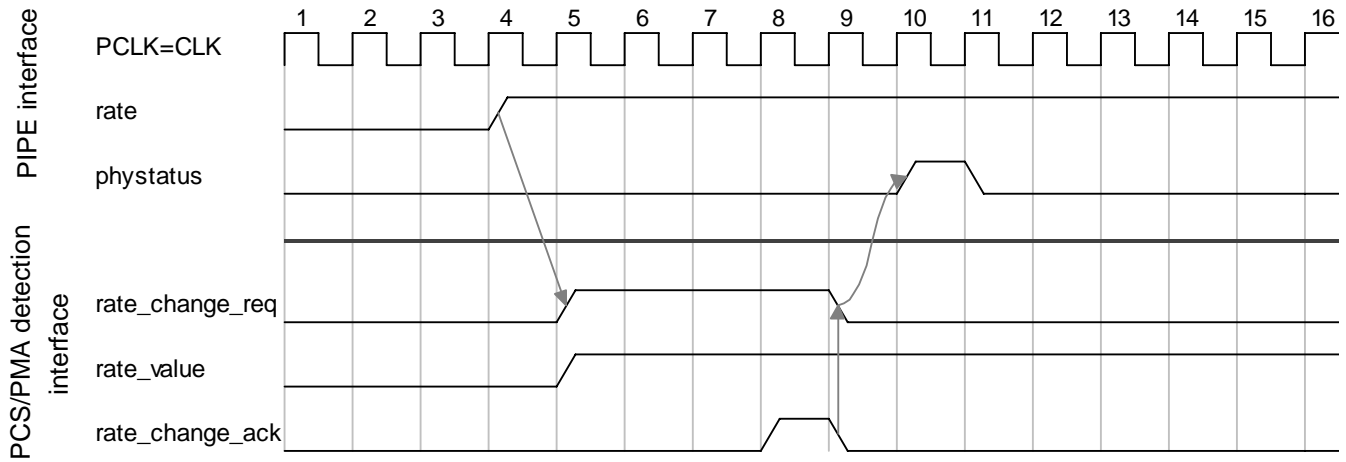
clock cycle 10: phystatus is asserted. The diagram below shows the receiver detection sequence.



The diagram below shows the rate change sequence for a rate change from 2.5 to 5.0 Gbps.

Note in particular:

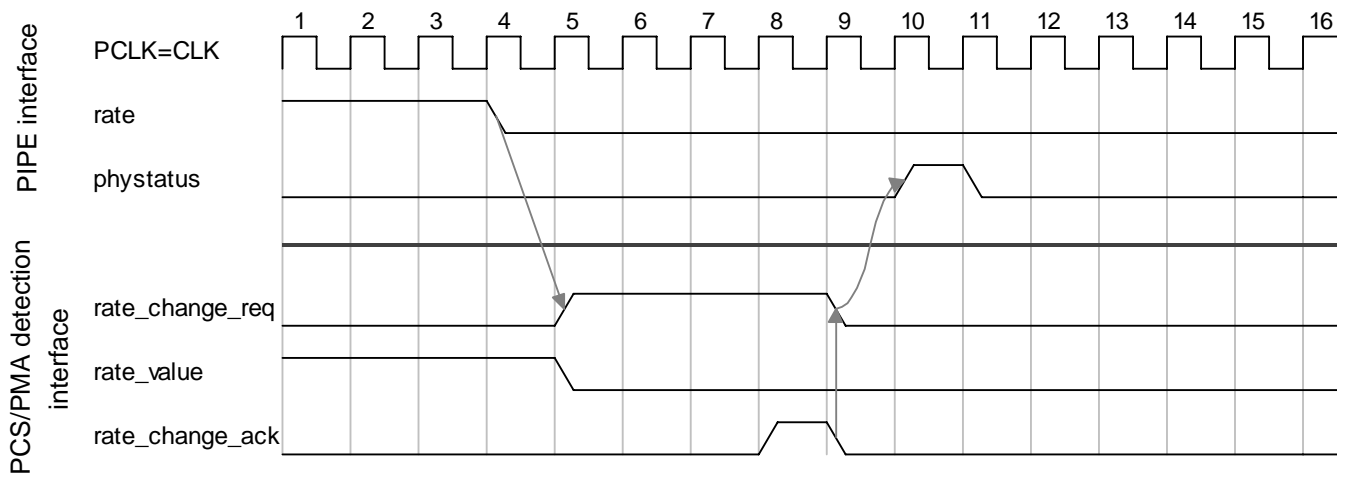
- **clock cycle 5:** rate change request and rate value is transmitted.
- **clock cycle 8:** rate change request is acknowledged.
- **clock cycle 10:** phystatus is asserted.



The diagram below shows the rate change sequence for a rate change from 5.0 to 2.5 Gbps.

Note in particular:

- **clock cycle 5:** rate change request and rate value is transmitted.
- **clock cycle 8:** rate change request is acknowledged.
- **clock cycle 10:** phystatus is asserted.



2.5 Test and Control Signals

The following table describes test and control signals for the XpressPCS:

Table 5: Test and control signals

Signal	I/O	Description
test_in[2:0]	in	<p>Test input port: This signal is used to generate errors in the XpressPCS:</p> <ul style="list-style-type: none"> • test_in[2:0]: 000: normal mode • test_in[2:0]: 001: inject data error • test_in[2:0]: 010: inject disparity error • test_in[2:0]: 011: inject different data • test_in[2:0]: 100: inject SDP instead of END • test_in[2:0]: 101: inject STP instead of END • test_in[2:0]: 110: inject END instead of data • test_in[2:0]: 111: inject EDB instead of END <p>Note: This signal does not exist in the 125 MHz version.</p>

Chapter 3 Implementing XpressPCS

This section describes how to implement XpressPCS for a multi-lane link, and provides an example of an implementation of a x4 PCI Express PHY using XpressPCS.

3.1 Implementing XpressPCS on a Multi-Lane Link

To implement a multi-lane PCI Express PHY, you should implement an instance of the XpressPCS for each lane.

The following table shows which PIPE signals are shared between each lane of a multi-lane implementation, and which signals are generated on a per-lane basis.

Table 6: Shared and per-lane PIPE signals

Shared Signals	Per-Lane PIPE Signals
• CLK	• RxData[], RxDataK[]
• PCLK	• TxEleIdle
• Reset#	• TxCompliance
• Txdetectrx	• RxPolarity
• powerdown	• RxValid
• phystatus	• RxEleIdle
• rate	• RxStatus[2:0]

3.1.1 Connecting Per-Lane Signals

Connect each per-lane PIPE signal to the XpressPCS instance for that lane.

3.1.2 Connecting Clock and Reset Signals

The following diagram shows the connection of external clock and reset signals for multi-lane PHY:

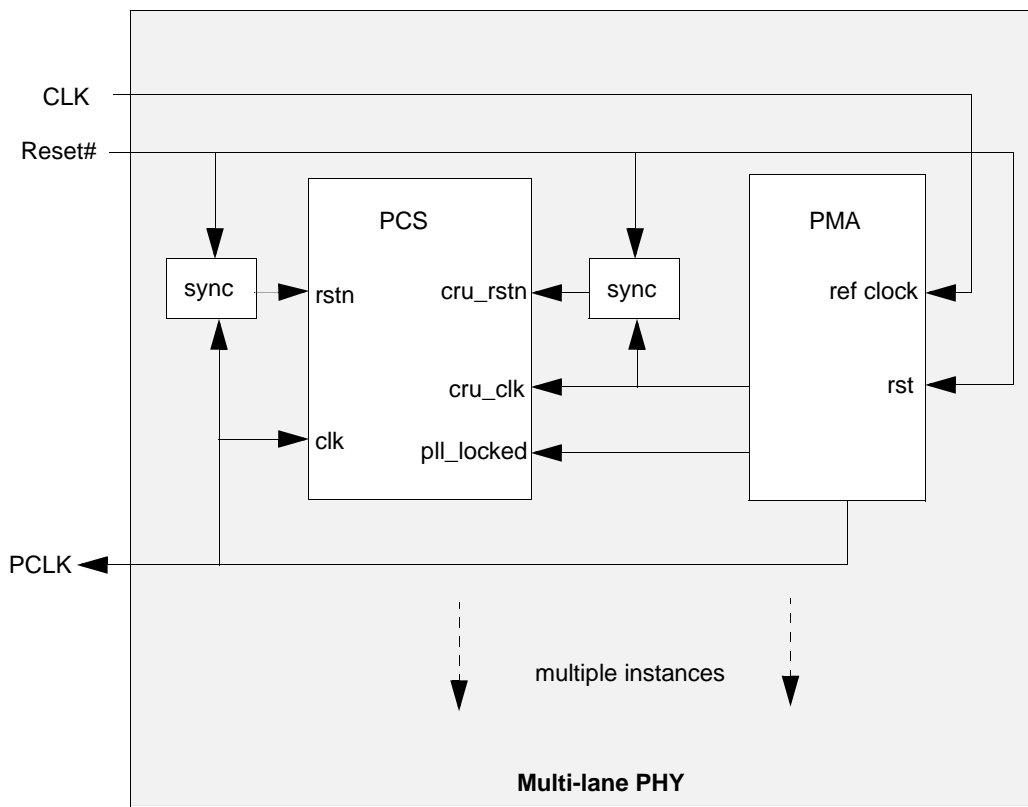


Figure 5: External clock and reset signals

The following table describes each of the external signals:

Table 7: Clock and reset signals

Signal	I/O	Description
CLK	In	CLK: This reference clock is used by all PMA instances to generate internal bit rate clocks for transmitting and receiving PCI Express data. The specifications for this clock signal (such as frequency and jitter) are implementation-dependent and must be specified for each implementation.
PCLK	Out	PCLK: Pipe Parallel clock. PCLK and the TX serial data rate should be derived from the same source (no ppm difference). This signal is usually generated by the PMA and is common to all lanes. For multi-lane implementations, you must select and connect one of the potential PCLK signals as the common signal (the pclk signals for other lanes should not be connected). The phystatus signal generated by the PCS on the same lane should also be selected as the common phystatus signal.
Reset#		Reset#: Active low reset. Used to generate XpressPCS active low (asynchronous falling edge / synchronous rising edge rstn) reset signals. You can also use this signal to generate PMA reset signals (additional logic may be required for PMA specifications).

3.1.3 Connecting MAC to PHY Signals

The PIPE signals `POWERDOWN` and `TXDETECTRX` are generated by the MAC and are common to all lanes. Connect these signals to each XpressPCS instance.

3.1.4 Connecting PHY to MAC Signals

The phystatus signal is used to communicate completion of PHY requests received from the MAC, and to indicate when the PCLK is stable. The PCLK signal and the phystatus signal should be generated on the same lane. For example, if the PCLK on lane 0 is chosen as the common PCLK signal, the phystatus signal on lane 0 should also be used as the common signal.

Generating the PCLK and the phystatus signals on the same lane ensures that PCLK stability will be correctly reported.

Note: Because all XpressPCS instances use the same PCLK, they receive MAC detection requests simultaneously, and transmit these requests to the PMA on the same PCLK cycle.

The MAC also samples the detection result for all lanes simultaneously, so the XpressPCS must provide this result when the common phystatus signal is asserted. This means that all PMAs must provide the receiver detection acknowledge and result signals on the same PCLK cycle. As a result, additional PMA logic may be required to synchronize these signals, depending on PMA behavior.

For multi-lane implementations, the `RATE_CHANGE_ACK` signal should be asserted for all PCS modules at the same PCLK cycle, when the rate change is complete and PCLK is stable.

3.2 Multi-Lane Implementation Example

The following example describes the implementation of a x4 PCIe PHY (16-bit at 125 MHz) using XpressPCS and Xilinx Virtex4 SERDES as PMA.

The source code for this example is provided in the full XpressPCS package files:

- `XpressPCS/src/vhdl/xilinx/v4fx125_v2/v4fx125_phyx4_v2.vhd`, or
- `XpressPCS/src/vlog/xilinx/v4fx125_v2/v4fx125_phyx4_v2.v`

The following diagram shows the `v4fx125_phyx4_v2` architecture:

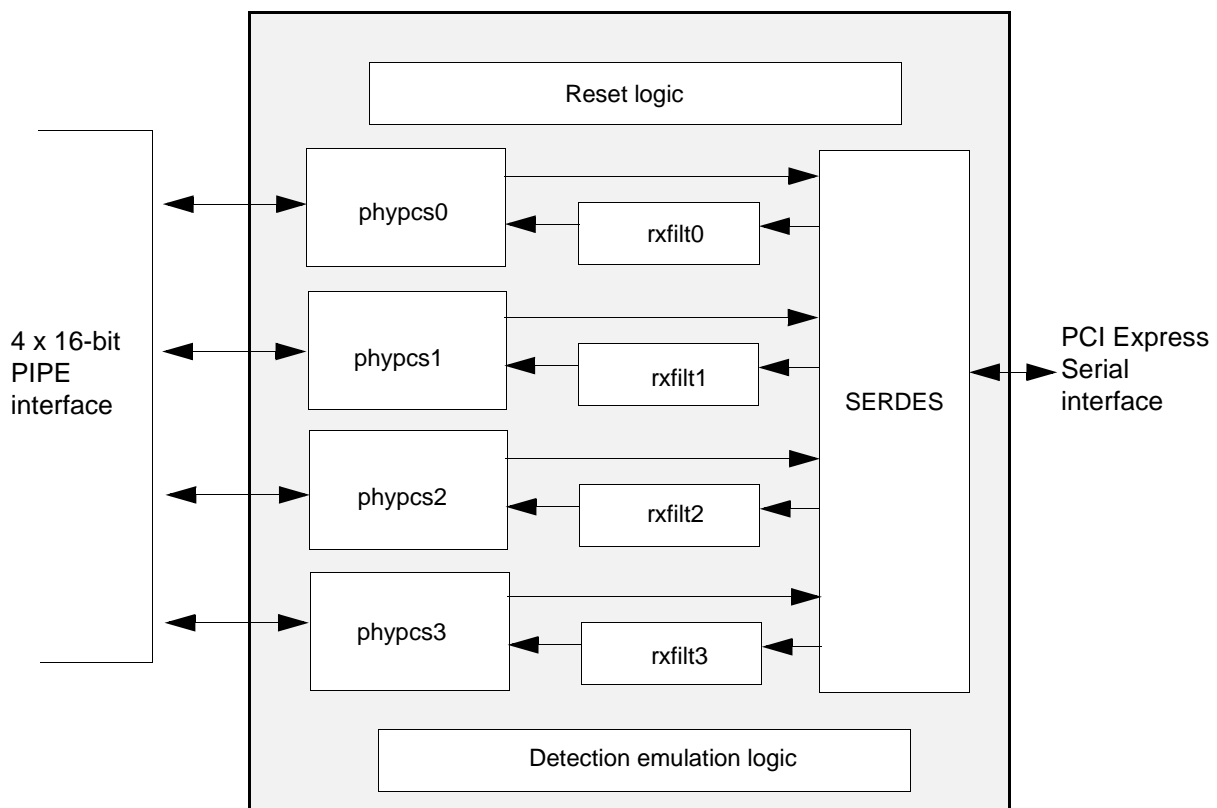


Figure 6: `v4fx125_phyx4_v2` architecture

Each sub-block in [Figure 6](#) is described in the table below:

Table 8: Description of v4fx125_phyx4_v2 architecture

Sub-block	Type	Description
Reset logic	Set of processes	Reset logic required by both the PMA and PCS.
phypcs0...3	pciexp125_phypcs instances	Multiple 16-bit XpressPCS instances
rxfilt0...3	v4fx125_rxfilter_v2 instances	Additional logic used to generate 10b received flow valid signal (rx_val).
SERDES	v4fx125_serdesx4_v2 instance	Wrapper for the Xilinx SERDES configured as PCI Express PMA. This block uses Xilinx MGT technology for clocking and calibration.
Detection emulation logic	Set of processes	Additional logic that emulates receiver detection. In this example, receiver detection is emulated because the Xilinx SERDES does not support receiver detection.

This example uses additional logic to meet PMA requirements and has the following characteristics:

- The PCLK (called 'clk125out' in the source code) is directly generated by the x4 PMA.
- The TX PLL on lane 0 is defined as the PCLK driver in the SERDES wrapper. As a result, the phystatus signal on lane 0 is shared between all lanes.
- Each 10b rx data flow is connected to each PCS instance with the associated Clock Data Recovery clock, enabling clock compensation in the elastic buffer of each PCS.
- The receiver detection emulation logic processes detection requests from the MAC on lane 0. Because the PCLK and phystatus is common to all lanes, detection requests from all lanes are received simultaneously by the PCS. Detection request acknowledge and results signals for all lanes are then transmitted simultaneously to the PCS by the PMA, which forwards this data to the PIPE interface using the phystatus signal.