Multi-Protocol 32G PCS for the DesignWare® Cores Multi-Protocol 32G PHY x4 for TSMC 7FF

PHY/PCS Wrapper Databook

PCS Version 1.54a
PHY Version: 2.03a
November 10, 2022

Copyright Notice and Proprietary Information

© 2022 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.

www.synopsys.com

Contents

Revision History 7

Preface 9
PHY/PCS Wrapper Databook Organization 9
STAR on the Web (SotW) 9
Reference Documentation 9
Web Resources 10
Synopsys Statement on Inclusivity and Diversity 10
Customer Support 11

Chapter 1
Product Overview 13
1.1 General Product Description 14
1.2 Features 16
1.2.1 Power States 16
1.2.2 Power Gating 17
1.2.3 Protocol Support 17
1.2.4 Bifurcation and Aggregation 17
1.2.5 Limitations When Operating in SerDes PIPE Architecture 18
1.2.6 Limitations When Operating in CCIX 18
1.2.7 Device Configuration 18
1.2.8 Device Parameters 25
1.3 Applications 27
1.4 Standards Compliance 28

Chapter 2
Architecture 31
2.1 PIPE Architecture 32
2.2 Resets 32
2.3 Clocks 32
2.3.1 PCLK as PHY Input Mode 33
2.3.2 PCLK as PHY Output Mode 36
2.3.3 PHY-to-PCS Clock Inputs 38
2.4 Transmit and Receive Datapaths 46
2.4.1 Original PIPE Architecture 47
2.4.2 SerDes PIPE Architecture 47

Chapter 3

PCS Signal Description Overview **49**
3.1 Signal Naming Conventions 50
3.2 Signal Descriptions per Control Parameters 50
3.3 Signals Controllable by SoC Registers 51
3.4 Scan Insertion Signals 51

Chapter 4
Signal Descriptions **53**
4.1 PCS Configuration Signals (top to phy/pcs) 55
4.2 PCS Configuration Signals (top to/from pcs) 56
4.3 PIPE Interface Signals (top to/from pcs) (for X = 0; X <= Nlanes-1) 66
4.4 MPLL Control Signals (top to/from pcs) (for N = 0; N <= Nphy-1) 88
4.5 MPLL Control Signals (top to/from phy) (for N = 0; N <= Nphy-1) 89
4.6 Reference Clock Control Signals (top to/from phy) 95
4.7 Power Related Signals (top to/from pcs) 102
4.8 Power Related Signals (top to/from pcs/phy) 103
4.9 Power Related Signals (top to/from phy) (for N = 0; N <= Nphy-1) 104
4.10 Analog I/O Signals (top to/from phy) 106
4.11 Receiver Signals (top to/from phy) (for X = 0; X <= Nlanes-1) 109
4.12 Transmitter Signals (top to/from phy) (for X = 0; X <= Nlanes-1) 111
4.13 PCS Scan Interface Signals (top to/from pcs) 113
4.14 PHY Scan Interface Signals (top to/from phy) (for N = 0; N <= Nphy-1) 115
4.15 JTAG Port Signals (top to/from phy) (for N = 0; N <= Nphy-1) 120
4.16 Boundary Scan Port Signals (top to/from phy) (for N = 0; N <= Nphy-1) 121
4.17 External Protocol Override Signals (top to/from pcs) 124
4.18 Parallel Control Register Port Signals (top to/from phy) (for N = 0; N <= Nphy-1) 328
4.19 External ROM Control Signals (top to/from phy) (for N = 0; N <= Nphy-1) 330
4.20 SRAM Control Signals (top to/from phy) (for N = 0; N <= Nphy-1) 331
4.21 PHY Test Signals (top to/from phy) 334
4.22 Resistor Tune Signals (Common Block Group) (top to/from phy) 335
4.23 Ethernet Clock Signals (top to/from phy) (for X = 0; X <= Nlanes-1) 337
4.24 Context Select Interface Signals (top to/from pcs) (for X = 0; X <= Nlanes-1) 340
4.25 Context Select Interface Signals (top to/from phy) (for X = 0; X <= Nlanes-1) 345
4.26 Scan Clock Interface Signals (top to/from pcs) 346
4.27 SerDes PIPE Signals (top to/from pcs) (for X = 0; X <= Nlanes-1) 347

Chapter 5
Usage Model **351**
5.1 Power-Up Sequence 353
5.2 Lane Reset Sequence 355
5.3 Clocks 355
5.3.1 PCS Output Clocks 355
5.3.2 PHY Output Clocks 361
5.3.3 Reference Clock Input 363
5.3.4 PLL Mode Selection 367
5.4 Datapath 368
5.4.1 Loopback Functions 370
5.5 Data Throttling Control for Original PIPE 371
5.6 Data Rate Change 372

5.7 Power State Transitions 372
5.7.1 Power State Transition Timing Diagrams 372
5.7.2 Power State Transition Times 385
5.8 Per-Lane ropll_clk Source Select 386
5.9 Clock Architecture and Link Selection 389
5.10 Enabling SRIS 390
5.11 Estore Configuration 391
5.12 Recalibration 391
5.13 RX Adaptation 392
5.14 PHY Bifurcation Support 392
5.15 PHY Aggregation Support 393
5.16 Lane Disabling and Up/Down-Configuration 394
5.16.1 RX Standby Mode 394
5.16.2 PIPE Lane Disable 394
5.16.3 P1.2 State 394
5.17 PHY Configuration Overrides 395
5.18 Transmitter Equalization Settings 396
5.19 Lane Margining at Receiver 397
5.20 ESM Rate Change Sequence 399
5.20.1 High-Level ESM Rate Change Sequence 399
5.21 EIEOS Format 401
5.22 Message Bus Support 402
5.22.1 Message Bus Registers for PCIE 402
5.22.2 Message Bus Registers for CCIX Specification 403
5.23 SATA BIST Loopback Mode 404
5.24 SATA: COM Align and Misalign Mechanism 404
5.25 SATA - Disconnect Detection 405
5.26 SATA - Nominal Empty Buffer Mode Operation 405
5.27 Power Gating Support 406
5.28 Tie-Off Values for Unused Lanes 408

Chapter 6
Simulation 411

Chapter 7
Synthesis 413
7.1 Synthesis Database 414
7.2 PIPE (PCS and PHY) Constraints 415
7.2.1 Hand-Placed Logic Constraints 416
7.2.2 Synchronizers 417

Revision History

PCS Version	PHY Version	Date	Description
1.54a	2.03a	November 10, 2022	Initial release

Preface

This document describes the Synopsys DesignWare® Cores Multi-Protocol 32G PCS core (PCS), which provides a PIPE 5-compliant interface between a MAC and a Synopsys DesignWare® Cores Multi-Protocol 32G PHY x4 for TSMC 7FF. The PCS, in conjunction with the DesignWare® Cores Multi-Protocol 32G PHY x4 for TSMC 7FF, provides interface connectivity in a System-on-Chip (SoC).

PHY/PCS Wrapper Databook Organization

This databook is organized as follows:

- Chapter 1, "Product Overview", introduces the Multi-Protocol 32G PCS core and its features.
- Chapter 2, "Architecture", describes the Multi-Protocol 32G PCS core's architecture.
- Chapter 3, "PCS Signal Description Overview", describes the organization of the Signal Description chapter of the Multi-Protocol 32G PCS core.
- Chapter 4, "Signal Descriptions", describes the top-level interface signals when DWC_PCLK_AS_-PHY_INPUT is defined.
- Chapter 5, "Usage Model", provides simulation requirements and usage information for the Multi-Protocol 32G PCS core.
- Chapter 6, "Simulation", lists modules that must be processed correctly during simulation to avoid problems related to the ASIC interface. In addition, this chapter includes ways to ensure that the simulation process is correct.
- Chapter 7, "Synthesis", describes the synthesis database and the PIPE constraints.

STAR on the Web (SotW)

You must review all STARs on the Web (SotWs) associated with your product. SotWs are considered a part of the Synopsys documentation suite, and show critical information related to your product. To review product SotWs, refer to the DesignWare IP product information:

https://www.synopsys.com/designware-ip.html

Reference Documentation

- *PCI Express® Base Specification*, Revision 5.0, Version 0.7, PCI-SIG Draft Specification 26, April 2018
- *PHY Interface For the PCI Express, SATA, and USB 3.1 Architectures,* Version 5.2, February 8, 2019, Intel Corporation

- PHY Interface For the PCI Express, SATA, and USB 3.1 Architectures**, Version 4.4.1, January 2017, Intel Corporation**
- PHY Interface For the PCI Express, SATA, and USB 3.1 Architectures**, Version 4.3, January 2014, Intel Corporation**
- *Serial ATA*, Revision 3.2, August 2013, Serial ATA International Organization
- *CCIX*, Revision **1.0, Version 0.9, March 23, 2018**
- **DesignWare Cores** *Multi-Protocol 32G PHY x4 for TSMC 7FF* **Databook**, Synopsys, Inc.

Web Resources

For more information, check the following Synopsys online resources:

- DesignWare IP product information: https://www.synopsys.com/designware-ip.html
- Your custom DesignWare IP page: https://www.synopsys.com/dw/mydesignware.php
- Documentation through SolvNetPlus: https://solvnetplus.synopsys.com (Synopsys password required)
- Synopsys Common Licensing (SCL): https://www.synopsys.com/support/licensing-installation-computeplatforms/licensing.html

Synopsys Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

Customer Support

To obtain support for your product, contact Support Center using one of the following methods:

- For the fastest response, enter a case through SolvNetPlus:
  a. https://solvnetplus.synopsys.com

  Note SolvNetPlus does not support Internet Explorer.

  b. Click the Cases menu and then click Create a New Case (below the list of cases).
  c. Complete the mandatory fields that are marked with an asterisk and click Save.
     Make sure to include the following:
     - Product L1: DesignWare Cores
     - Product L2: SERDES_32G_PHY

  For more information about general usage information, refer to the following article in SolvNetPlus:
  https://solvnetplus.synopsys.com/s/article/SolvNetPlus-Usage-Help-Resources
- Or, send an e-mail message to support_center@synopsys.com(your email will be queued and then, on a first-come, first-served basis, manually routed to the correct support engineer):
  - Include the Product L1 and Product L2 names, and Version number in your e-mail so it can be routed correctly.
  - For simulation issues, include the timestamp of any signals or locations in waveforms that are not understood.
  - Attach any debug files you created.
- Or, telephone your local support center:
  - North America:
    Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
  - All other countries:
    https://www.synopsys.com/support/global-support-centers.html

1 Product Overview

The DesignWare® Cores Multi-Protocol 32G PCS core is a configurable Physical Coding Sublayer (PCS) that complies with the PIPE 5.2, 4.4.1, and 4.3 specifications and supports the PCIe 5.0 protocols.

This chapter includes the following sections:

- "General Product Description" on page 14
- "Features" on page 16
- "Applications" on page 27
- "Standards Compliance" on page 28

1.1 General Product Description

The Multi-Protocol 32G PCS core provides many of the functions outlined in the PIPE 5.2 specification. The following modes (while not all combinations) are supported:

- Original PIPE Architecture
- SerDes PIPE Architecture
- "PCLK as PHY Output"
- "PCLK as PHY Input with PHY owned PLL"
- "PCLK as PHY Input with External PLL and PHY PLL"

The Multi-Protocol 32G PCS core can be used in conjunction with the Synopsys DesignWare Cores Multi-Protocol 32G PHY x4 for TSMC 7FF to create a PIPE5-compliant PHY solution. The following figure shows how the Multi-Protocol 32G PCS core fits in a PIPE5 PHY subsystem.

Figure 1-1 Multi-Protocol 32G PCS as Part of a PIPE5 PHY Subsystem

The top level—referred to as the PIPE wrapper— includes instantiation of the Multi-Protocol 32G PCS core and the Multi-Protocol 32G PHY x4 for TSMC 7FF. This databook describes the soft IP block highlighted (in blue) in Figure 1-1.

The top-level inputs and outputs of the PIPE wrapper directly connect to the PCS and the PHY.

"PCS Signal Description Overview" on page 49 provides the detailed mapping of these connections. However, the interface between the PCS and PHY is not described in this document.

The Multi-Protocol 32G PCS provides the following features, but not all features are supported in each mode:

- Support for "Original PIPE Architecture"
- Support for "SerDes Architecture", which shifts most of the protocol-specific logic into the controller, such as the Elastic Buffer
- Support for "PCLK as PHY Output"
- Support for "PCLK as PHY Input with PHY owned PLL"
- Support for "PCLK as PHY Input with External PLL and PHY PLL".

- Support for “Low Pin Count Interface”, which removes pins that do not have “critical timing requirements” and maps them into message bus registers.
- Standard PIPE interface to upper layers of a protocol stack (MAC layer and above)
- PCIe: 8b/10b (Gen 1/2) and 128b/130b (Gen3/4/5) encoding and decoding
- SATA: 8b/10b encoding and decoding
- Translation of PIPE power states (P0, P1, and so on) into the appropriate PHY controls, including any PHY-specific intermediate state sequencing
- Clock-rate compensation by SKP ordered sets and an elasticity buffer (Original PIPE Architecture)
- Error reporting
- Optional bifurcation of multi-lane PHYs into multiple independent PIPE 5 interfaces
- Optional aggregation to connect up to four PHYs

1.2 Features

This section summarizes the functions that the Multi-Protocol 32G PCS supports.

1.2.1 Power States

The following table lists the supported power states and the impact of each state.

Table 1-1 Power States Supported by the PCS

Power State	Disabled Elements	Active Elements	Notes
PCIe5 Operation			
P0	N/A	All datapath and control	Fully-powered state
P0s	■ PHY CDR disabled if EIOS received or pipe_rxX_standby is asserted ■ PHY RX clocks	■ PHY TX Electrical Idle ■ PHY reference clock	N/A
P1	■ PHY RX clocks ■ PHY TX Electrical Idle and TX clocks	■ MPLLs and MPLL clocks ■ PHY reference clock	N/A
P2	■ PHY RX clocks ■ PHY TX clocks ■ PHY TX common-mode ■ PCLK	■ PHY reference clock	N/A
P1.CPM	■ PHY RX clocks ■ PHY TX clocks ■ PCLK ■ PHY reference clock can be disabled	■ PHY TX common mode	Equivalent to L1 Clock Power Management power state
P1.1[a]	■ PHY RX clocks ■ RX Electrical Idle Exit (EIE) Detector ■ PHY TX clocks ■ PCLK ■ PHY reference clock can be disabled	■ PHY TX common mode	Equivalent to PCIe L1.1 power state
P1.2[b]	■ PHY RX clocks ■ RX Electrical Idle Exit (EIE) Detector ■ PHY TX clocks ■ PHY TX common-mode ■ PCLK pipe_laneX_pclk (pipe_laneX-_in_pclk) ■ PHY reference clock can be disabled If power gating is enabled, the switchable core supply can be disabled in this state (for more information, see "Power Gating" on page 17).	N/A	Equivalent to PCIe L1.2 power state

Power State	Disabled Elements	Active Elements	Notes
P2.CPM	■ PHY RX clocks ■ PHY TX clocks ■ PHY TX common-mode ■ PCLK ■ PHY reference clock can be disabled .	N/A	■ Equivalent to PCIe L2 state with reference clock disabled ■ Beacon detection is not supported because the reference clock is disabled
SATA6G Operation			
Active	N/A	All data path and control	Fully-powered state
Partial	■ PHY CDR ■ PHY RX Clocks	■ PHY TX Electrical Idle ■ PHY reference clock	N/A
Slumber	■ PHY RX clocks ■ PHY TX clocks ■ PHY TX common-mode ■ PCLK (pcs_laneX_pclk)	■ PHY reference clock	N/A
DevSleep	■ PHY RX clocks ■ PHY TX clocks ■ PHY TX common-mode ■ PCLK (pcs_laneX_pclk)	■ PHY reference clock	N/A

a. For PIPE specification 4.4.1, P1.1 is supported by setting the power state to P1.CPM. For more information about power states, see signal “pipe_laneX_powerdown” description at “PCS Signal Description Overview” on page 49.
b. For PIPE specification 4.4.1, P1.2 is supported by setting the power state to P1.CPM. For more information about power states, see signal “pipe_laneX_powerdown” description at “PCS Signal Description Overview” on page 49.

1.2.2 Power Gating

The Multi-Protocol 32G PCS supports gating of the core supply in a PCIe P1.2 state, where most of the logic is power-gated. A small portion of the design is in an always-on power domain and is needed to enter/exit the power-gated state. The Multi-Protocol 32G PCS also generates controls for the DesignWare® Cores Multi-Protocol 32G PHY x4 for TSMC 7FF to gate its digital core voltage supplies.

1.2.3 Protocol Support

Each PCS lane can be independently configured to operate as follows:

- PCIe (1.1, 2.1, 3.0, 4.0 and 5.0) modes
- SATA (1.5G, 3.5G, and 6G) modes

However, the following restrictions exist:

- The pipe_laneX_protocol[1:0] input for configuring the protocol for each lane can be changed only when the Multi-Protocol 32G PCS is in Reset mode.

1.2.4 Bifurcation and Aggregation

You can configure the Multi-Protocol 32G PCS lanes to operate as part of one or multiple PCIe links (bifurcation). In addition, the Multi-Protocol 32G PCS can support connectivity for up to four PHYs

(aggregation). Multiple options are available to configure the Multi-Protocol 32G PCS for aggregation and PCIe bifurcation. For more information, see "PHY Bifurcation Support" and "PHY Aggregation Support".

Note Multiple PCS instances cannot be aggregated to create a single link.

1.2.5 Limitations When Operating in SerDes PIPE Architecture

SATA is not supported in SerDes mode.

RxWidth[1:0] must be always the same as Width[1:0].

1.2.6 Limitations When Operating in CCIX

For CCIX operation, set the ext_pclk_req input signal to 1'b0 and the upcs_pipe_config[0] input to 1'b1. Set all lanes in the same link to the same ESM-related request in the PHY's Message Bus register, even for inactive lanes.

1.2.7 Device Configuration

This release requires you to manually edit the upcs/include/*e32_defaults.v file to change the configuration of the Multi-Protocol 32G PCS. The following table describes the RTL parameters available to change the configuration; you are permitted to use only these parameters. To enable the SerDes PIPE mode, refer to example in upcs/include/*e32_defaults.v.

Note If using the Synopsys PCIe Controller, configure the controller feature "Gen3/Gen4/Gen5 EQ Presets to Coefficients Mapping Modes" to use the programmable table option by setting "CX_GEN3_EQ_PSET_COEF_MAP_MODE" to "Programmable Table". For more information, refer to the DesignWare Cores PCIe Controller databook for further information. The Programmable Table option allows for full flexibility of mapping presets to coefficients.

Table 1-2 RTL Parameters Available for PCS Configuration

RTL Parameter	Description
DWC_E32MP_X4_TOP_PG_PINS	Power Gating Pins Support ■ When DWC_E32MP_X4_TOP_PG_PINS is defined, top-level pins for power are included. ■ When DWC_E32MP_X4_TOP_PG_PINS is undefined, no pins for power are included but may be inserted by the UPF flow. Default: Undefined
DWC_E32MP_X4_SEP_RESREF	Separate Reference Resistor Connections ■ When DWC_E32MP_X4_SEP_RESREF is defined, each PHY has a separate resref_pin, phyX_resref. ■ When DWC_E32MP_X4_SEP_RESREF is undefined, each PHY shares the same resref_pin, phy_resref. Default: Undefined

RTL Parameter	Description
DWC_E32MP_X4_SEP_RESREF_CALIB	Separate Reference Resistor Calibration Connections ■ When DWC_E32MP_X4_SEP_RESREF_CALIB is defined, connections for res_req_in/out and res_ack_in/out for each PHY are brought out to the top level. You are responsible for these connections. ■ When DWC_E32MP_X4_SEP_RESREF_CALIB is undefined (default), the UPCS daisy chains res_req_in/out and res_ack_in/out connections. For more information, see the “Reference Resistor” section in the PHY databook.
DWC_E32MP_X4_ALT_SRC_CLK	Alternate Source Clock Connections This is a preliminary feature. Contact Synopsys for more details. ■ When DWC_E32MP_X4_ALT_SRC_CLK is defined, connections exist to connect multiple UCPS instances together to form a single link. ■ When DWC_E32MP_X4_ALT_SRC_CLK is undefined (default), no connections exist to connect multiple UCPS instances together to form a single link.
DWC_PIPE_VER_4_4_1 DWC_PIPE_VER_5	PIPE Version Support Select one of these macros. ■ When the DWC_PIPE_VER_4_4_1 macro is defined, the UPCS supports Pipe 4.4.1. ■ When DWC_PIPE_VER_5 macro is defined: ❑ UPCS supports Pipe 5.2. ❑ The Low Pin Count feature is enabled. ❑ PCLK as PHY input is supported. ❑ DWC_UPCS_MULTIPROTOCOL_SUPPORT must be undefined. PCLK as PHY input: ■ Two phase-aligned synchronous clocks, pclk and pcs_clk are provided by MAC. ■ In Original PIPE Architecture, these two clocks can be of different frequencies. ■ In SerDes Architecture, pclk and pcs_clk must be the same frequency. The following two handshake sequences are required in this mode: ■ pipe_laneX_pclkchangeok and pipe_laneX_pclkchangeack ■ pipe_laneX_maxpclkreq_n and pipe_laneX_maxpclkack_n

RTL Parameter	Description
DWC_PCIE_PIPE_SERDES	SerDes PIPE Support When DWC_PCIE_PIPE_SERDES is defined: ■ SerDes PIPE Architecture is supported. ■ DWC_PIPE_VER_5 must also be defined. ■ One of the following macros must be defined: ❑ DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S ❑ DWC_UPCS_PIPEWIDTH_2S_2S_2S_2S_4S ❑ DWC_UPCS_PIPEWIDTH_4S_4S_8S_8S_8S ❑ DWC_UPCS_PIPEWIDTH_4S_4S_4S_8S_8S ■ DWC_UPCS_PIPEWIDTH_1S_2S_2S_2S_4S_PCLK1G must be undefined. ■ When DWC_PCIE_PIPE_SERDES is undefined, Original PIPE Architecture is supported.

RTL Parameter	Description
DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S DWC_UPCS_PIPEWIDTH_2S_2S_2S_2S_4S DWC_UPCS_PIPEWIDTH_4S_4S_8S_8S_8S DWC_UPCS_PIPEWIDTH_4S_4S_4S_8S_8S DWC_UPCS_PIPEWIDTH_1S_2S_2S_2S_4S_PCLK1G	Predetermined PIPE Width Support. Select none or one of the following macros. ■ When no macro is defined, PIPE Width is requested with pipe_laneX_if_width signal. (No clock optimized) ■ When DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S macro is defined, ❑ PIPE widths for PCIe Gen1/2/3/4/5 are 2S/2S/4S/4S/4S(S=symbol). ❑ ESM20/25 are the same as PCIe Gen5 (Clock optimized). ❑ DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S can be defined, if DWC_PCIE_PIPE_SERDES is undefined. ■ When DWC_UPCS_PIPEWIDTH_2S_2S_2S_2S_4S is defined, ❑ PIPE widths for PCIe Gen1/2/3/4/5 are 2S/2S/2S/2S/4S(S=symbol). ❑ ESM20/25 are the same as PCIe Gen5. (Clock optimized) ❑ DWC_PCIE_PIPE_SERDES must also be defined ■ When DWC_UPCS_PIPEWIDTH_4S_4S_8S_8S_8S macro is defined, PIPE widths for PCIe Gen1/2/3/4/5 are 4S/4S/8S/8S/8S (S=symbol). The original PIPE mode is unavailable and pipe_laneX_serdes_mode must be tied to 1. If this mode is used, include the -serdes_with_orig_mode argument when running synthesis. ■ When DWC_UPCS_PIPEWIDTH_4S_4S_4S_8S_8S macro is defined: ❑ PIPE widths for PCIe Gen1/2/3/4/5 are 4S/4S/4S/8S/8S (S=symbol). ❑ ESM20/25 are the same as PCIe Gen5 (Clock optimized). ❑ DWC_PCIE_PIPE_SERDES must also be defined ❑ The Original PIPE Architecture Mode is unavailable. (pipe_laneX_serdes_mode must be tied to 1.) ■ When DWC_UPCS_PIPEWIDTH_1S_2S_2S_2S_4S_PCLK1G macro is defined, ❑ PIPE widths for PCIe Gen1/2/3/4/5 are 1S/2S/2S/2S/4S (S=symbol) (Clock optimized). ❑ DWC_PIPE_VER_5 must be defined. ❑ DWC_PCIE_PIPE_SERDES must be undefined.

RTL Parameter	Description
DWC_UPCS_MULTIPROTOCOL_SUPPORT	Protocol Support ■ When DWC_UPCS_MULTIPROTOCOL_SUPPORT is defined: ❑ Support for PCIe and SATA protocols is included. ❑ DWC_PIPE_VER_4_4_1 must also be defined. ❑ DWC_PIPE_VER_5 must be undefined. ■ When DWC_UPCS_MULTIPROTOCOL_SUPPORT is undefined, only support for the PCIe protocol is included.
DWC_PCIE5_SUPPORT	PCIe Gen5 Support ■ When DWC_PCIE5_SUPPORT is defined, PCIe Gen5 is supported and 1000 MHz (4S for PCIe Gen5) is the highest frequency of PCLK. ■ When DWC_PCIE5_SUPPORT is undefined: ❑ PCIe Gen5 is not supported, and 781.25 MHz (4S for ESM25) is the highest frequency of PCLK. ❑ PIPE Width for PCIe Gen4 must be 4S with 50 MHz. ❑ “CCIX25 - MPLLA Cascaded Common Clock” must be used. ❑ DWC_PIPE_VER_4_4_1 must also be defined.
DWC_UPCS_PERLINK_CLK_ARCH	Per-link Clock Architecture Support ■ When DWC_UPCS_PERLINK_CLK_ARCH is defined, the per-link clock architecture is selected. ■ When DWC_UPCS_PERLINK_CLK_ARCH is undefined, the per-lane clock architecture is selected. Per-Link clock architecture enables more efficient implementation of CTS in the PCS. **Default:** Not defined **Note:** Not supported for x6 packages.
DWC_UPCS_NLINKS	Max Number of Links Supported ■ DWC_UPCS_NLINKS is only valid when DWC_UPCS_PERLINK_CLK_ARCH is defined. ■ DWC_UPCS_NLINKS defines the maximum number of links that can be supported. ■ The max number of links must be in the set {1, 2, 4} and must be no greater than the number of lanes. **Note:** Not supported for x6 packages.
DWC_UPCS_NLINKS_GTR1	■ DWC_UPCS_NLINKS_GTR1 is only valid when DWC_UPCS_PERLINK_CLK_ARCH is defined. ■ DWC_UPCS_NLINKS_GTR1 is a derived macro that must be defined when DWC_UPCS_NLINKS is greater than 1. **Note:** Not supported for x6 packages

RTL Parameter	Description
DWC_UPCS_NLINKS_GTR2	■ DWC_UPCS_NLINKS_GTR2 is only valid when DWC_UPCS_PERLINK_CLK_ARCH is defined. ■ DWC_UPCS_NLINKS_GTR2 is a derived macro that must be defined when DWC_UPCS_NLINKS is greater than 2. **Note:** Not supported for the x6 package
Examples of how to define *NLINKS* macros	For a system with only one link: //`define DWC_UPCS_PERLINK_CLK_ARCH //`define DWC_UPCS_NLINKS 1 For a system with only two links: //`define DWC_UPCS_PERLINK_CLK_ARCH //`define DWC_UPCS_NLINKS 2 //`define DWC_UPCS_NLINKS_GTR1 // For a system with only four links: //`define DWC_UPCS_PERLINK_CLK_ARCH //`define DWC_UPCS_NLINKS 4 //`define DWC_UPCS_NLINKS_GTR1 //`define DWC_UPCS_NLINKS_GTR2
DWC_PCIE_ESTORE_MPS_OPT_EN	Max Payload Size Support Estore Max Payload Size Optimizations Enable ■ When DWC_PCIE_ESTORE_MPS_OPT_EN is defined, max payload size-related area and latency optimizations in ESTORE are enabled. ■ When DWC_PCIE_ESTORE_MPS_OPT_EN is undefined, (default) ESTORE is sized to support max payload size packets of 4 KB.
DWC_PCIE_ESTORE_MPS_VALUE	Estore Max Payload Size Supported This feature can significantly reduce the area (and power) consumed by the IP within systems that are not designed for and will never take advantage of or use the larger max payload sizes (1024 bytes and larger). ■ If DWC_PCIE_ESTORE_MPS_OPT_EN is defined, DWC_PCIE_ESTORE_MPS_VALUE defines the max payload size (in bytes) supported in ESTORE. ■ If DWC_PCIE_ESTORE_MPS_OPT_EN is undefined, the max payload size supported in ESTORE is 4 KB, regardless of this macro. Use only one of the following macros: ■ DWC_PCIE_ESTORE_MPS_VALUE 128 ■ DWC_PCIE_ESTORE_MPS_VALUE 256 ■ DWC_PCIE_ESTORE_MPS_VALUE 512 ■ DWC_PCIE_ESTORE_MPS_VALUE 1024 ■ define DWC_PCIE_ESTORE_MPS_VALUE 2048 ■ define DWC_PCIE_ESTORE_MPS_VALUE 4096 Default value: 4,096

RTL Parameter	Description
DWC_E32MP_X4_ESTORE_DEPTH_CONTROL_SUPPORT	■ Elastic Buffer Depth Control is supported when DWC_E32MP_X4_ESTORE_DEPTH_CONTROL_SUPPORT is defined. ❑ DWC_PCIE_ESTORE_MPS_OPT_EN cannot be set at the same time. ❑ DWC_PIPE_VER_5 must also be defined. ■ Elastic Buffer Depth Control is not supported when DWC_E32MP_X4_ESTORE_DEPTH_CONTROL_SUPPORT is undefined.
DWC_E32MP_X4_INT_SRAM_SUPPORT	Internal versus External Instantiation of SRAM (required for PHY operation) ■ When DWC_E32MP_X4_INT_SRAM_SUPPORT is defined, the PHY instantiates internal SRAM (requires Synopsys SRAM license; contact Synopsys). ■ When DWC_E32MP_X4_INT_SRAM_SUPPORT is undefined, you instantiate SRAM outside the PHY. This is the commonly used mode.
DWC_E32MP_X4_REMOVE_CNTX_RSTR_INPUTS	PHY Pin Reduction Support ■ When DWC_E32MP_X4_REMOVE_CNTX_RSTR_INPUTS is defined: ❑ PHY configuration inputs pins are removed. ❑ PHY configuration is managed internally by the PHY. ■ When DWC_E32MP_X4_REMOVE_CNTX_RSTR_INPUTS is undefined: ❑ PHY configuration input signals are present. ❑ PHY configuration can be managed internally by the PHY or by SoC configuration registers.

RTL Parameter	Description
DWC_PCLK_AS_PHY_INPUT	Note: The description of this parameter is included in this table to describe the different modes of operation for PCLK as the PHY input. This parameter enables PCLK as the PHY input. These are the following source clock options in PCLK as PHY input. ■ PHY owned PLL and IN_PCLK SOURCE MAX_PCLK MODE ■ PHY owned PLL and IN_PCLK SOURCE MPLL_CLK MODE ■ External PLL and PHY PLL In any mode: ■ pipe_laneX_pclkchangeok and pipe_laneX_pclkchangeack handshake is required. PHY owned PLL and IN_PCLK SOURCE MAX_PCLK MODE: ■ pipe_laneX_max_pclk is used as a source clock of pipe_laneX_in_pclk ■ pipe_laneX_maxpclkreq_n and pipe_laneX_maxpclkack_n handshake is required. See these signals description. ■ pipe_laneX_ext_pll_mode=0 PHY owned PLL and IN_PCLK SOURCE MPLL_CLK MODE: ■ phyN_mpll(a/b)_*_clk is used as a source clock of pipe_laneX_in_pclk ■ phyN_mpll{a/b}_force_en and phyN_mpll{a/b}_force_ack handshake is required. See these signals description. ■ pipe_laneX_ext_pll_mode=1 External PLL and PHY PLL: ■ External PLL is a source clock of pipe_laneX_in_pclk ■ pipe_laneX_ext_pll_mode=1 **Note:** DWC_PCLK_AS_PHY_INPUT is automatically defined if DWC_PIPE_VER_5 is defined; otherwise, DWC_PCLK_AS_PHY_INPUT is not changeable.

1.2.8 Device Parameters

The following table describes system parameters that are required for configuring non-Synopsys controllers.

Table 1-3 Device Parameters

Parameter	Description
NFTS	This parameter is used only for PCI Express and represents the number of Fast Training Sequences (FTSs) required to be advertised to the remote partner to exit L0s/P0s successfully. NFTS = 200.

1.3 Applications

The Multi-Protocol 32G PCS is designed to be used with the DesignWare® Cores Multi-Protocol 32G PHY x4 for TSMC 7FF IP and to interface with the following controllers:

- **PCIe**: PIPE 5.2/4.4.1/4.3/4.2-compliant PCIe 1.1, 2.1, 3.0, 4.0 or 5.0 controllers
- **SATA6G**: PIPE4.3-compliant SATA 1.5G, 3G, or 6G controllers

The Multi-Protocol 32G PCS is designed to maximize scalability and configurability on a per-lane basis while minimizing the integration complexity with PHY and controller IP.

In addition, the Multi-Protocol 32G PCS supports a power-gating option for PCIe operation that is integrated with the PHY and is suitable for extremely low-power applications with low-exit latencies.

1.4 Standards Compliance

The Multi-Protocol 32G PCS is compliant to the PIPE 5.2/4.4.1/4.3 specification with the following exceptions:

- PIPE 4.3, Section 7.17: "…To support lane polarity inversion, the PHY must invert received data when RxPolarity is asserted. Inverted data must begin showing up on RxData[] within 20 PCLKs of when RxPolarity is asserted…."

  The Multi-Protocol 32G PCS requires more than 20 PCLK cycles for certain interface widths and data rates.

- PIPE 4.3, Section 6.1: "…PhyStatus… For "PCLK as PHY Output mode, when this signal transitions during entry and exit from any PHY state where PCLK is not provided, then the signaling is asynchronous…"

  The Multi-Protocol 32G PCS does not generate the PhyStatus output when transitioning between two power states where the reference clock input to the PHY can be turned off. For example, when transitioning between P1.CPM and P1.2, PhyStatus is not generated because the PCS does not have any clock available when the reference clock is disabled.

- PIPE 4.3, Section 6.1: "…AsyncPowerChangeAck…Only used when transitioning between two power states without PCLK..." (supported in PIPE 4.4.1)

  The Multi-Protocol 32G PCS does not support this handshake because, in states where PCLK is disabled, the reference clock to the PHY can also be disabled. In the absence of the reference clock, the Multi-Protocol 32G PCS cannot implement a handshake sequence.

- PIPE 4.4.1, Section 8.14 states rules for operating in Nominal Empty buffer mode. The rule stating, "All SKP symbols of SOS are removed(8b/10b SKP)," is not supported.

- PIPE 4.3, Section 7.2: The diagram in this section shows the following requirement for PCLK when Reset# is asserted: "PCLK running at any frequency less than or equal to final operational frequency."

  The Multi-Protocol 32G PCS does not generate pcs_laneX_pclk when pcs_laneX_reset_n is asserted. If the SATA controller requires pcs_laneX_pclk to be running at a specific frequency for the de-assertion of pcs_laneX_reset_n, an external, anti-glitch clock MUX circuit is required. When pcs_laneX_reset_n is asserted, the external clock MUX should select an 'always-on' clock, and when it is de-asserted, the MUX must select pcs_laneX_pclk coming from the PCS. The output of this clock MUX must be connected to the controller's per-lane PCLK input.

- PIPE 4.3, Section 7.5: "...Power state transitions between two power states that do not provide PCLK are not allowed."

  When SATA mode is selected, both Slumber and DevSleep states do not provide PCLK. The transition between these two states is allowed by the SATA specification and is supported by the Multi-Protocol 32G PCS. In SATA mode, the reference clock is never turned off; therefore, PhyStatus is generated for all the power-mode transitions.

- PIPE 4.3, Section 7.15: "...If more than one error should happen to occur on a received byte (or set of bytes transferred across a 16-bit interface or 32-bit interface), the errors should be signaled with the priority shown below."

  Misalign (SATA mode only) is shown in the fifth position on the priority list in the PIPE 4.3 specification. In SATA mode, the Multi-Protocol 32G PCS gives Misalign the highest priority among all the errors reported on the RxStatus bus.

- PIPE 4.3, Table 6.3: "TxDetectRx/Loopback: ...Sata Mode: Loopback support is optional for SATA PHYs...."

  The Multi-Protocol 32G PCS does not support the optional Loopback mode mentioned in the PIPE specification. The BIST Loopback modes required by the SATA specification are supported through

non-PIPE sideband signals. Near-end analog Loopback mode is supported using the pcs_laneX_tx2rx-_loopbk pin, and the Far-end analog Loopback mode is supported using the pcs_laneX_rx2tx_loopbk pin.

In addition, note that the Multi-Protocol 32G PCS is backward-compatible to the PIPE 4.2 specification with proprietary signal support for PCIe L1 substate entry and exit sequences.

- In PCLK as PHY Output mode, if a rate change is requested while PowerDown=P1 state, PCLK may change frequency to a new rate after some lane's PhyStatus response but must be complete before all lane's PhyStatus response for the link.

2 Architecture

This chapter contains the following sections:

- "PIPE Architecture" on page 32
- "Resets" on page 32
- "Clocks" on page 32
- "Transmit and Receive Datapaths" on page 46

2.1 PIPE Architecture

The Multi-Protocol 32G PCS supports the Original PIPE Architecture and SerDes PIPE Architecture. When DWC_PIPE_VER_4_4_1 is defined, only Original PIPE architecture and "PCLK as PHY Output" is supported. When DWC_PCIE_PIPE_SERDES is defined, the PCS supports both architectures and the pipe_laneX_serdes_mode input is used to select which architecture is currently active. When DWC_PCIE_PIPE_SERDES is defined, "PCLK as PHY Input" is the only option for clocking. For more information about these clocking modes, refer to "Clocks".

2.2 Resets

The Multi-Protocol 32G PCS provides the following reset inputs:

- Global reset: phy_reset
- Per-lane reset: pipe_laneX_reset_n (where X represents lane number)
- Per-lane PERST#: pipe_laneX_perst_n (where X represents lane number)

The active-high phy_reset input connects to the phy_reset input of each PHY connected to the Multi-Protocol 32G PCS. Therefore, if phy_reset is asserted, any PHY control register workarounds in place need to be re-applied after phy_reset de-assertion. Also, if phyN_sram_bootload_bypass input = 0 and any PHY firmware updates are made in SRAM, those updates would have to be re-applied after phy_reset de-assertion.

The per-lane reset, pipe_laneX_reset_n, is an active-low signal that resets all registers in lane X. This reset is equivalent in functionality to the PIPE interface reset signal (Reset#) as defined in the PIPE specification. For lanes belonging to the same PIPE link, pipe_laneX_reset_n for the corresponding lanes must be driven from the common PIPE reset signal from the MAC. The pipe_laneX_reset_n must be asserted whenever phy_reset is asserted.

The per-lane reset, pipe_laneX_perst_n, is an active-low signal that resets all registers in lane X and connects to per-lane PHYs input, laneX_perst_n. This reset corresponds to the PCI Express CEM specification signal PERST#. This reset must also be asserted whenever phy_reset is asserted and whenever a glitch is introduced in REFCLK.

All internal registers are reset asynchronously with the assertion of reset inputs. To reset the internal registers, clock inputs to the Multi-Protocol 32G PCS do not have to be active.

In addition to the primary reset inputs, the Multi-Protocol 32G PCS generates internal resets when Power-Gating mode is enabled (pg_mode_en = 1) and when the lanes are in a P1.2 power state. In this mode, when lane X enters a P1.2 power state, internal registers in lane X are reset in preparation for power to be disabled. All registers in lane X, except the always-ON logic (upcs_pipe_ctl_* and upcs_clk_ctl_*), are reset in this mode. If all Multi-Protocol 32G PCS lanes connected to a particular PHY (PHY*N*, where N represents PHY number) are in a P1.2 power-gating state, the phyN_pg_reset output to PHY*N* is asserted, resetting the internal registers in the PHY. This action is performed in preparation for power to be disabled in the PHY.

2.3 Clocks

This section discusses PCLK in PHY input and output modes, as well as PHY-to-PCS clock inputs.

2.3.1 PCLK as PHY Input Mode

In this mode, pclk and PCS datapath clocks are provided as inputs pipe_laneX_in_pclk and pipe_laneX_in_pcs_clk. The pipe_laneX_in_pclk signal corresponds to the PCLK signal in the PIPE specification. The pipe_laneX_in_pclk inputs from all lanes configured to be part of the same link are phase-synchronous to each other. The reference clock input is used for power state and rate control logic in the Multi-Protocol 32G PCS. In SerDes Architecture mode, the Multi-Protocol 32G PCS provides per-lane output PIPE RxClk, pipe_rxX_clk.

For more information about these signals, refer to the "PIPE Interface Signals" section within the links available at "PCS Signal Description Overview" on page 49.

Figures describing rate changes and power state entry/exit can be found in "Data Rate Change" on page 372.

Figure 2-1 shows the transmit and receive datapaths within each lane.

Figure 2-1 PCLK as PHY Input: Clock Architecture

Flop-based clock divider with Clock Gating Cell

Glitchless Clock Mux

Scan Clock Mux

Lane_rx ICG logic:

- Ratio is always 1 except for PCIE Gen4 in 32bit mode (DWC_PCIE_GEN4_32B_MODE is set)
- This logic is always present although it is unnecessary for PCIE Gen4 in 16b mode (DWC_PCIE_GEN4_16B_MODE is set)

Lane_clk_ctl ICG logic:

- Ratios are a function of protocol, rate, PIPE interface width

For clock source options when "PCLK as PHY Input" is enabled (DWC_PIPE_VER_5 is defined), refer to the following sections.

2.3.1.1 PHY-Owned PLL and IN_PCLK SOURCE MAX_PCLK MODE

In this mode, pipe_laneX_in_pclk is sourced from pipe_laneX_max_pclk, which corresponds to the MaxPCLK signal in the PIPE specification. The pipe_laneX_max_pclk outputs of all lanes that are part of the same link are phase-synchronous to each other. The pipe_laneX_max_pclk output is a flop-based divide of the PHY output txX_ropll_word_clk, which rate is based on pipe_laneX_rate and pipe_laneX_max_pclk_div_ratio_g*. The pipe_laneX_max_pclk_div_ratio_g* inputs are top-level inputs which need to be set to a value before assertion of phy_reset.

For a description of power states in which max pclk output stops, refer to Section 5.3.1, "PCS Output Clocks" and Table 5-3: "PCS Clock Dependency (PIPE 4.4 or later)".

Note pipe_laneX_max_pclk stops during rate change.

Requirements

- pipe_laneX_maxpclkreq_n and pipe_laneX_maxpclkack_n handshake is required.
- pipe_laneX_ropll_freq_mode can be used for the special case when the controller needs 2000 MHz pipe_laneX_max_pclk.
- upcs_pipe_config[0] must be set to 1'b1 (to allow for lane reversal).
- pipe_laneX_ext_pll_mode must be set to 1'b0.
- pipe_laneX_lanepll_src_sel can be used to select the source clock from per-lane ropll_clk. For more details, see Section 5.8, "Per-Lane ropll_clk Source Select".

2.3.1.2 PHY-Owned PLL and IN_PCLK SOURCE MPLL_CLK MODE

In this mode, pipe_laneX_in_pclk is sourced from phyN_mplla_word_clk and phyN_mpllb_word_clk. The phyN_mpll{a/b}_force_en and phyN_mpll{a/b}_force_ack interfaces are required to force the clock to continue toggling even when it would normally be turned off. In direct clocking mode, phyN_mplla_word_clk and phyN_mpllb_word_clk are fixed to 1 GHz. In cascaded clocking mode, phyN_mplla_word_clk and phyN_mpllb_word_clk are less than 1 GHz.

Requirements:

- upcs_pipe_config[0] can be set to 1'b1 or 1'b0.
- pipe_laneX_ext_pll_mode must be set to 1'b1.
- pipe_laneX_lanepll_src_sel can be used to select the source clock from per-lane ropll_clk. For more details, see Section 5.8, "Per-Lane ropll_clk Source Select".

2.3.1.3 External PLL and PHY PLL

In this mode, pipe_laneX_in_pclk is sourced from an external PLL. This mode is useful when there are system requirements to have the PLL always running or when the PHY output clock frequencies do not match the system requirements.

Requirements:

- upcs_pipe_config[0] can be set to 1'b1 or 1'b0.
- pipe_laneX_ext_pll_mode must be set to 1'b1.
- pipe_laneX_lanepll_src_sel can be used to select the source clock from per-lane ropll_clk. For more details, see Section 5.8, "Per-Lane ropll_clk Source Select".

2.3.2 PCLK as PHY Output Mode

If the DWC_UPCS_PERLINK_CLK_ARCH parameter is not defined, the clock selection logic is implemented per lane. First, each lane selects clock sources from the appropriate PHY lane, then the lane_pclk, lane_pcs_clk, and lane_pma_clk signals are derived from the MPLL clock sources of the selected PHY.

If DWC_UPCS_PERLINK_CLK_ARCH is defined, the clock selection logic is implemented per link. First, each link selects clocks sources from the appropriate PHY lane, then the link clocks are broadcast to all lanes that belong to that link as a function of the bifurcation use mode selected by the application on pipe_laneX_lanepll_src_sel[3:0]. For more information about the use modes, see "Clock Architecture and Link Selection" on page 389.

The Multi-Protocol 32G PCS provides the following clock inputs:

- MPLL clock inputs from PHYs: phy_txX_ropll_word_clk
- Reference clock inputs from PHYs: phyN_ref_dig_clk
- Receiver recovered clock input from PHY lanes: phy_rxX_clk

The MPLL clock inputs are used to derive the PCS datapath clocks and the pipe clock outputs (pipe_laneX_pclk, pipe_laneX_max_pclk) to the MAC. The duty cycle for these clocks is based on PHYs MPLL clocks. Refer to the PHY databook for Duty Cycle. The reference clock input is used for power state and rate control logic in the Multi-Protocol 32G PCS.

The Multi-Protocol 32G PCS provides the following clocks outputs:

- Per-lane PIPE PCLK: pipe_laneX_pclk
- Per-lane PIPE max PCLK: pipe_laneX_max_pclk

The **pipe_laneX_pclk** signal corresponds to the PCLK signal in the PIPE specification. The pipe_laneX_pclk outputs from all lanes configured to be part of the same link are phase-synchronous to each other. Therefore, pipe_laneX_pclk from one lane can be used to sample/launch synchronous PIPE interface signals for other lanes that are part of the same link. (For information about configuring for bifurcation, see "PHY Bifurcation Support" on page 392.)

Figure 2-2 shows the transmit and receive datapaths within each lane.

Figure 2-2 PCLK as PHY Output: Clock Architecture

Flop-based clock divider with Clock Gating Cell

Glitchless Clock Mux

Scan Clock Mux

Lane_rx ICG logic:

- Ratio is always 1 except for PCIE Gen4 in 32bit mode (DWC_PCIE_GEN4_32B_MODE is set)
- This logic is always present although it is unnecessary for PCIE Gen4 in 16b mode (DWC_PCIE_GEN4_16B_MODE is set)

Lane_clk_ctl ICG logic:

- Ratios are a function of protocol, rate, PIPE interface width

The following tables list the supported frequencies from data throttling.

Table 2-1 Supported Data Throttling Frequencies for PCIe

Rate	PIPE		PCS		Throttling
Gen1	1s	1000 MHz	2s	125 MHz	1 datavalid in 4 PCLKs
Gen2	2s	1000 MHz	2s	250 MHz	1 datavalid in 4 PCLKs
Gen3	2s	1000 MHz	2s	500 MHz	1 datavalid in 2 PCLKs
Gen4	2s	1000 MHz	2s	1000 MHz	None
Gen5	4s	1000 MHz	4s	1000 MHz	None
Note: Gen2/2s/1000 MHz/1 in four PCLks is not a PIPE-compliant combination.					

Table 2-2 Supported Data Throttling Frequencies for SATA

Rate	PIPE		PCS		Throttling
SATA 1	1s	150 MHz	2s	75 MHz	None
SATA 2	1s	300 MHz	2s	150 MHz	None
SATA 3	2s	300 MHz	2s	300 MHz	None

2.3.3 PHY-to-PCS Clock Inputs

The Multi-Protocol 32G PCS receives the following clock inputs from the PHY:

- MPLL clock inputs from PHYs: phy_txX_ropll_word_clk
- Reference clock inputs from PHYs: phyN_ref_dig_clk
- Receiver recovered clock input from PHY lanes: phy_rxX_clk

In PIPE 4.4.1, lane X can be disabled due to down-configuration by asserting pipe_txX_elecidle and pipe_txX_compliance. In a lane-disabled state, the pipe_laneX_pclk output can remain enabled by setting the upcs_pipe_config[0] input to 1. Doing so enables pipe_laneX_pclk from the disabled lane X to be used for sampling and launching PIPE signals for other lanes in the same link. When DWC_UPCS_PERLINK_CLK_ARCH is defined, set upcs_pipe_config[0] to 1. Setting this signal allows per-link clock architecture to support lane reversal with broken lanes, which means that physical lane0 is not detected.

For scenarios where the PIPE rate and PIPE interface width require PCLK frequencies lower than the source MPLL clocks, the PCLK is generated by pulse-gating the MPLL source clocks. As a result, the duty cycle for the divided-down PCLK is not 50% in these cases. However, the clock definitions in the synthesis constraints use a 50% duty cycle with the highest supported clock frequency, due to implementation constraints and because the design is constrained to the highest frequency (that is, the highest rate and smallest interface width).

Note When lanes are disabled, assert pipe_laneX_reset_n for that lane.

Table 2-3 lists the supported pipe_laneX_clk frequencies for PIPE 4.4.1.

Table 2-3 PIPE 4.4.1 pipe_laneX_pclk Frequencies

Protocol/Rate	pipe_laneX_pclk Frequency (MHz)		
	PIPE Interface Width: 8-bit	PIPE Interface Width: 16-bit	PIPE Interface Width: 32-bit
DWC_UPCS_PIPEWIDTH_* is Not Defined			
PCIe Gen1	250	125	62.5
PCIe Gen2	500	250	125
PCIe Gen3	N/A	500	250
PCIe Gen4	N/A	1,000	500
PCIe Gen5	N/A	N/A	1000
CCIX 20 GT/s	N/A	1,250	625
CCIX 25 GT/s	N/A	1,562.5	781.25
SATA Gen1	150	75	37.5
SATA Gen2	300	150	75
SATA Gen3	600	300	150
DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S is Defined			
PCIe Gen1	N/A	125	N/A
PCIe Gen2	N/A	250	N/A
PCIe Gen3	N/A	N/A	250
PCIe Gen4	N/A	N/A	500
PCIe Gen5	N/A	N/A	1000
CCIX 20 GT/s	N/A	N/A	625
CCIX 25 GT/s	N/A	N/A	781.25
SATA Gen1	N/A	75	N/A
SATA Gen2	N/A	150	N/A
SATA Gen3	N/A	N/A	150

Table 2-4 lists the supported pipe_laneX_in_clk frequencies for the PIPE 5 non-SerDes mode.

Table 2-4 PIPE 5 Non-SerDes Mode pipe_laneX_in_pclk Frequencies

Protocol/Rate	pipe_laneX_in_pclk Frequency (MHz)		
	PIPE Interface Width: 8-bit	PIPE Interface Width: 16-bit	PIPE Interface Width: 32-bit
DWC_UPCS_PIPEWIDTH_1S_2S_2S_2S_4S_PCLK1G is Defined			
PCIe Gen1	1,000	N/A	N/A

Protocol/Rate	pipe_laneX_in_pclk Frequency (MHz)		
	PIPE Interface Width: 8-bit	PIPE Interface Width: 16-bit	PIPE Interface Width: 32-bit
PCIe Gen2	N/A	1,000	N/A
PCIe Gen3	N/A	1,000	N/A
PCIe Gen4	N/A	1,000	N/A
PCIe Gen5	N/A	N/A	1000
CCIX 20 GT/s	N/A	N/A	N/A
CCIX 25 GT/s	N/A	N/A	N/A
SATA Gen1	N/A	N/A	N/A
SATA Gen2	N/A	N/A	N/A
SATA Gen3	N/A	N/A	N/A
DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S is Defined			
PCIe Gen1	N/A	125	N/A
PCIe Gen2	N/A	250	N/A
PCIe Gen3	N/A	N/A	250
PCIe Gen4	N/A	N/A	500
PCIe Gen5	N/A	N/A	1000
CCIX 20 GT/s	N/A	N/A	625
CCIX 25 GT/s	N/A	N/A	781.25
SATA Gen1	N/A	N/A	N/A
SATA Gen2	N/A	N/A	N/A
SATA Gen3	N/A	N/A	N/A

Table 2-5 lists the supported pipe_laneX_in_clk frequencies for PIPE 5 SerDes mode.

Table 2-5 PIPE 5 SerDes Mode pipe_laneX_in_pclk Frequencies

Protocol/Rate	pipe_laneX_in_pclk Frequency (MHz)			
	PIPE Interface Width: 8-bit	PIPE Interface Width: 16-bit	PIPE Interface Width: 32-bit	PIPE Interface Width: 64-bit
DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S is Defined				
PCIe Gen1	N/A	125	N/A	N/A
PCIe Gen2	N/A	250	N/A	N/A
PCIe Gen3	N/A	N/A	250	N/A
PCIe Gen4	N/A	N/A	500	N/A
PCIe Gen5	N/A	N/A	1000	N/A
CCIX 20 GT/s	N/A	N/A	625	N/A
CCIX 25 GT/s	N/A	N/A	781.25	N/A
SATA Gen1	N/A	N/A	N/A	N/A
SATA Gen2	N/A	N/A	N/A	N/A
SATA Gen3	N/A	N/A	N/A	N/A
DWC_UPCS_PIPEWIDTH_2S_2S_2S_2S_4S is Defined				
PCIe Gen1	N/A	125	N/A	N/A
PCIe Gen2	N/A	250	N/A	N/A
PCIe Gen3	N/A	500	N/A	N/A
PCIe Gen4	N/A	1000	N/A	N/A
PCIe Gen5	N/A	N/A	1000	N/A
CCIX 20 GT/s	N/A	N/A	N/A	N/A
CCIX 25 GT/s	N/A	N/A	N/A	N/A
SATA Gen1	N/A	N/A	N/A	N/A
SATA Gen2	N/A	N/A	N/A	N/A

Protocol/Rate	pipe_laneX_in_pclk Frequency (MHz)			
	PIPE Interface Width: 8-bit	PIPE Interface Width: 16-bit	PIPE Interface Width: 32-bit	PIPE Interface Width: 64-bit
SATA Gen3	N/A	N/A	N/A	N/A
DWC_UPCS_PIPEWIDTH_4S_4S_8S_8S_8S is Defined				
PCIe Gen1	N/A	N/A	62.5	N/A
PCIe Gen2	N/A	N/A	125	N/A
PCIe Gen3	N/A	N/A	N/A	125
PCIe Gen4	N/A	N/A	N/A	250
PCIe Gen5	N/A	N/A	N/A	500
CCIX 20 GT/s	N/A	N/A	N/A	N/A
CCIX 25 GT/s	N/A	N/A	N/A	N/A
SATA Gen1	N/A	N/A	N/A	N/A
SATA Gen2	N/A	N/A	N/A	N/A
SATA Gen3	N/A	N/A	N/A	N/A
DWC_UPCS_PIPEWIDTH_4S_4S_4S_8S_8S is Defined				
PCIe Gen1	N/A	N/A	62.5	N/A
PCIe Gen2	N/A	N/A	125	N/A
PCIe Gen3	N/A	N/A	250	N/A
PCIe Gen4	N/A	N/A	N/A	250
PCIe Gen5	N/A	N/A	N/A	500
CCIX 20 GT/s	N/A	N/A	N/A	N/A
CCIX 25 GT/s	N/A	N/A	N/A	N/A
SATA Gen1	N/A	N/A	N/A	N/A
SATA Gen2	N/A	N/A	N/A	N/A

Protocol/Rate	pipe_laneX_in_pclk Frequency (MHz)			
	PIPE Interface Width: 8-bit	PIPE Interface Width: 16-bit	PIPE Interface Width: 32-bit	PIPE Interface Width: 64-bit
SATA Gen3	N/A	N/A	N/A	N/A

Table 2-6 lists the supported pipe_laneX_in_clk and pipe_laneX_pcs_clk frequencies when the DWC_PIPE_VER_5 is defined.

Table 2-6 pipe_laneX_in_pclk and pipe_laneX_in_pcs_clk Frequencies When DWC_PIPE_VER_5 is Defined

PIPE Interface Width (MHz)						
DWC_PIPE_VER_5 is defined and DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S is not defined and DWC_UPCS_PIPEWIDTH_1S_2S_2S_2S_PCLK1G is not defined and DWC_UPCS_PIPEWIDTH_1S_1S_4S_4S_4S_PCLK1G is not defined and DWC_UPCS_PIPEWIDTH_1S_2S_2S_2S_4S_PCLK1G is not defined and DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S_PCLK1G is not defined and DWC_UPCS_PIPEWIDTH_4S_4S_8S_8S_8S is not defined and DWC_UPCS_PIPEWIDTH_4S_4S_4S_8S_8S is not defined						
Protocol/Rate	8-bit		16-bit		32-bit	
	in_pclk	in_pcs_clk	in_pclk	in_pcs_clk	in_pclk	in_pcs_clk
PCIe Gen1	250	125	125	125	62.5	125
PCIe Gen2	500	250	250	250	125	250
PCIe Gen3	N/A	N/A	500	500	250	500
PCIe Gen4	N/A	N/A	1000	1000	500	1000
PCIe Gen5	N/A	N/A	N/A	N/A	1000	1000
CCIX 20 GT/s	N/A	N/A	N/A	N/A	N/A	N/A
CCIX 25 GT/s	N/A	N/A	N/A	N/A	N/A	N/A

PIPE Interface Width (MHz)						
DWC_PIPE_VER_5 is defined **and DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S is not defined** **and DWC_UPCS_PIPEWIDTH_1S_2S_2S_2S_PCLK1G is not defined** **and DWC_UPCS_PIPEWIDTH_1S_1S_4S_4S_4S_PCLK1G is not defined** **and DWC_UPCS_PIPEWIDTH_1S_2S_2S_2S_4S_PCLK1G is not defined** **and DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S_PCLK1G is not defined** **and DWC_UPCS_PIPEWIDTH_4S_4S_8S_8S_8S is not defined** **and DWC_UPCS_PIPEWIDTH_4S_4S_4S_8S_8S is not defined**						
Protocol/Rate	**8-bit**		**16-bit**		**32-bit**	
	in_pclk	**in_pcs_clk**	**in_pclk**	**in_pcs_clk**	**in_pclk**	**in_pcs_clk**
SATA Gen1	N/A	N/A	N/A	N/A	N/A	N/A
SATA Gen2	N/A	N/A	N/A	N/A	N/A	N/A
SATA Gen3	N/A	N/A	N/A	N/A	N/A	N/A

Note

SerDes Mode supports the following modes of operation.

- When DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S macro is defined, PIPE Width for PCIe Gen1/2/3/4/5 are 2S/2S/4S/4S/4S(S=symbol).
- When DWC_UPCS_PIPEWIDTH_2S_2S_2S_2S_4S macro is defined, PIPE Width for PCIe Gen1/2/3/4/5 are 2S/2S/2S/2S/4S(S=symbol).
- When DWC_UPCS_PIPEWIDTH_4S_4S_8S_8S_8S macro is defined, PIPE Widths for PCIe Gen1/2/3/4/5 are 4S/4S/8S/8S/8S(S=symbol).
- When DWC_UPCS_PIPEWIDTH_4S_4S_4S_8S_8S macro is defined, PIPE Widths for PCIe Gen1/2/3/4/5 are 4S/4S/4S/8S/8S(S=symbol).

See full descriptions of these macros in “Device Configuration” on page 18.

The pipe_laneX_max_pclk signal corresponds to the MaxPCLK signal in the PIPE specification. The pipe_laneX_max_pclk output of all lanes that are part of the same link are phase-synchronous to each other. Table 2-7 provides the pipe_laneX_max_pclk frequencies for each protocol. When pipe_laneX_ext_pll_mode is set to 0 and DWC_PIPE_VER_5 is defined (DWC_PCLK_AS_PHY_INPUT), pipe_laneX_max_pclk_div_ratio_* will be used to divide the rate. When pipe_laneX_ext_pll_mode is set to 1, pipe_laneX_max_pclk is not used.

Table 2-7 pipe_laneX_max_pclk Frequencies

Protocol	Rate (MHz) with DWC_PCLK_AS_PHY_INPUT=OFF	Rate (MHz) with DWC_PCLK_AS_PHY_INPUT=ON and pipe_laneX_ext_pll_mode=0
PCIe Gen1	1,000	1,000/ pipe_laneX_max_pclk_div_ratio_g1
PCIe Gen2	1,000	1,000/ pipe_laneX_max_pclk_div_ratio_g2

Protocol	Rate (MHz) with DWC_PCLK_AS_PHY_INPUT=OFF	Rate (MHz) with DWC_PCLK_AS_PHY_INPUT=ON and pipe_laneX_ext_pll_mode=0
PCIe Gen3	1,000	1,000/ pipe_laneX_max_pclk_div_ratio_g3
PCIe Gen4	1,000	1,000/ pipe_laneX_max_pclk_div_ratio_g4
PCIe Gen5	1,000	1,000/ pipe_laneX_max_pclk_div_ratio_g5
CCIX 20 GT/s	625	625/ pipe_laneX_max_pclk_div_ratio_esm20
CCIX 25 GT/s	781.25	781.25/ pipe_laneX_max_pclk_div_ratio_esm25
SATA Gen1	600	
SATA Gen2	600	
SATA Gen3	600	

2.4 Transmit and Receive Datapaths

The following figure shows the transmit and receive datapaths within each lane.

Figure 2-3 Transmit and Receive Datapaths

2.4.1 Original PIPE Architecture

The pipe_txX_data transmit data and controls are converted from the PIPE interface clock to the internal pcs_clk in the TX gasket. The fixed data-path-width data at the output of the TX gasket is encoded as follows:

- 8b10b-encoded for 8b10b protocols
- 128b-130b-encoded for PCIe Gen3/4/5

To support RX-to-TX loopback mode as specified in the PIPE specification, the RX recovered data is multiplexed into the TX datapath. Note that for 8b10b mode, the non-decoded 8b10b received data is sent to the transmitter. For non-8b10b data, the received data is decoded first; that is, the header is extracted from the data, then looped back into the TX datapath, where it is encoded before being transmitted to the PHY.

Data received from the PHY is processed to find symbol alignment for 8b10b data and block alignment for non-8b10b data. The aligned data is monitored for EIOS ordered sets (PCIe) to detect end-of-packets and to infer entry into electrical idle condition. After detecting EIOSs, the Multi-Protocol 32G PCS de-asserts phy_rxX_data_en to the PHY lane, causing the lane CDR to stop monitoring input data transitions and to switch to reference clock tracking mode.

The CDR is also disabled in any of the following conditions:

- pipe_rxX_standby = 1,
- pipe_laneX_powerdown not in P0 or P0s states,
- During a pipe rate or pipe width change, or
- phy_rx_valid = 0

The aligned data is passed to the elastic buffer, where compensation for frequency drift is performed by adding or removing:

- SKP symbols (PCIe)
- ALIGNp primitives (SATA)

The frequency-compensated data is 8b10b-decoded for 8b10b data, then passed through the PIPE RX processing block where data and status signals are cycle-aligned. The RX data and status signals pass through the RX gasket that maps the internal fixed-width data to variable interface widths supported by the PIPE interface.

2.4.2 SerDes PIPE Architecture

When pipe_laneX_serdes_mode=1, the majority of the datapath, including estore, is moved to the controller. In this mode, the phy_txX_data is driven from the SerDes logic within lane_tx, and the non-Serdes datapath is not used. In this mode, pipe_rxX_data is held in reset, and pipe_rxX_serdes_data, pipe_rxX_serdes_elecidle, pipe_rxX_serdes_valid, and so on, are driven from the lane_rx_serdes module. There is a one-stage pipeline in RX and TX datapath. In SerDes mode, the RX and TX datapath has optional pipeline stages. For ease of timing closure, the optional pipeline stages are necessary. If the latency needs to be reduced, these stages can be removed.

The PHY CDR is disabled in any of the following conditions:

- pipe_laneX_powerdown is not in P0 or P0s power state
- During a pipe rate or pipe width change
- phy_rx_valid = 0, or

- pipe_rxX_standby is asserted

See how to configure SerDes PIPE Architecture in "Device Configuration" on page 18.

3
PCS Signal Description Overview

This chapter provides addition information about the PCS signals. The following topics are discussed:

- "Signal Naming Conventions"
- "Signal Descriptions per Control Parameters"
- "Signals Controllable by SoC Registers"
- "Scan Insertion Signals"

3.1 Signal Naming Conventions

Abstract signal names are used in this databook for convenience. Special notation enables one abstract signal name to represent a set of literal signals.

As an example, consider the signal names, pipe_laneX_protocol and phyN_mplla_div_clk:

- X denotes lane number (0,1,2,3...15).

  This range is represented as (for X=0; X <=Nlanes-1).

  Per-lane pins are replicated for each lane in the Multi-Protocol 32G PCS.
- N denotes PHY number (0,1,2,3).

  This range is represented as (for N=0; N <=Nphy-1).
- P denotes consistency and reduces redundancy for signals with the protocol prefix.

  This range is represented as (for P=0; P <=2).
- IPNAME referenced in signal descriptions is a placeholder for the module name, such as DWC_E32MP_X4.

3.2 Signal Descriptions per Control Parameters

Signal descriptions are identical across all Signal Description chapters. However, specific signals will be available, depending on the state of the Control Parameters.

3.3 Signals Controllable by SoC Registers

In general, inputs in the following sections should be controllable by SoC registers. However, some of the inputs (for example, pipe_laneX_protocol) may be tied to a certain value based on usage. Contact Synopsys support for further assistance.

- PCS Configuration Signals
- MPLL Control Signals
- Reference Clock Control Signals
- External Protocol Override Signals

  For protocolP_esm_* signals:

  - If Extended Speed Modes (ESM) for CCIX specification are not used, tie all protocolP_esm* inputs to 0.
  - If Extended Speed Modes (ESM) for CCIX specification are used, tie all protocol0_esm*_r0 inputs to 0.
  - When ESM is not enabled and overrides are enabled, protocol*_g1/2/3/4 overrides are used (non-ESM ones).
  - When ESM is enabled and overrides are enabled, AND

    When pipe_laneX_rate=Gen1, protocol0_*_g1 inputs are used.

    When pipe_laneX_rate=Gen2, protocol0_*_g2 inputs are used.

    When pipe_laneX_rate=Gen3 and esm_rate0 = 8G, protocol0_*_g3 inputs are used.

    When pipe_laneX_rate=Gen3 and esm_rate0 = 16G, protocol0_*_g4 inputs are used.

    When pipe_laneX_rate=Gen4 and esm_rate1 = 20G or 25G, protocol0_esm_*_r1 inputs are used.

    Default settings for ESM overrides reside in the `upcs/include/*e32_pcie_esm_ref_100m.v` file. Settings for 20G rate end with 20G and settings for 25G rate end with 25G.

- Context Select Interface Signals

  If macro DWC_E32MP_X4_REMOVE_CNTX_RSTR_INPUTS is defined, all protocolP_* inputs can be tied to 0, except for the following list. See description of DWC_E32MP_X4_REMOVE_CNTX_RSTR_INPUTS in Table 1-2, for more information.

  - protocol_ext_mplla_recal_bank_sel_ovrd_en
  - protocol_ext_mplla_recal_bank_sel
  - protocol_ext_mpllb_recal_bank_sel_ovrd_en
  - protocol_ext_mpllb_recal_bank_sel
  - protocolP_ext_tx_eq_ovrd_g*
  - protocolP_ext_tx_eq_pre_g*
  - protocolP_ext_tx_eq_main_g*
  - protocolP_ext_tx_eq_post_g*

3.4 Scan Insertion Signals

Some input/output ports are created during scan insertion. The number of added ports depends on the number of scan chains. Refer to DFT insert script (upcs/synth/scripts/upcs_dft_insert.tcl) for complete details.

4

Signal Descriptions

This chapter details all I/O signals in the IP. Inputs are on the left of the signal diagrams; outputs are on the right. In addition to describing the function of each signal, the signal descriptions in this chapter might include the following information:

- **Active State:** Indicates whether the signal is active high or active low. When a signal is not intended to be used in a particular application, then this signal needs to be tied or driven to the inactive state (opposite of the active state).
- **Registered:** Indicates whether or not the signal is registered directly inside the IP boundary without intervening logic (excluding simple buffers). A value of *No* does not imply that the signal is not synchronous, only that there is some combinatorial logic between the signal's origin or destination register and the boundary of the IP. A value of N/A indicates that this information is not provided for this IP title.
- **Synchronous to:** Indicates which clocks in the IP sample this input (drive for an output). This clock might not be the same as the clock that your application logic should use to clock (sample/drive) this pin. For more details, consult the clock section in the databook.

The I/O signals are grouped as follows:

- "PCS Configuration Signals (top to phy/pcs)" on page 55
- "PCS Configuration Signals (top to/from pcs)" on page 57
- "PIPE Interface Signals (top to/from pcs) (for X = 0; X <= Nlanes-1)" on page 67
- "MPLL Control Signals (top to/from pcs) (for N = 0; N <= Nphy-1)" on page 88
- "MPLL Control Signals (top to/from phy) (for N = 0; N <= Nphy-1)" on page 89
- "Reference Clock Control Signals (top to/from phy)" on page 95
- "Power Related Signals (top to/from pcs)" on page 102
- "Power Related Signals (top to/from pcs/phy)" on page 103
- "Power Related Signals (top to/from phy) (for N = 0; N <= Nphy-1)" on page 104
- "Analog I/O Signals (top to/from phy)" on page 106
- "Receiver Signals (top to/from phy) (for X = 0; X <= Nlanes-1)" on page 109
- "Transmitter Signals (top to/from phy) (for X = 0; X <= Nlanes-1)" on page 111
- "PCS Scan Interface Signals (top to/from pcs)" on page 113

- "PHY Scan Interface Signals (top to/from phy) (for N = 0; N <= Nphy-1)" on page 115
- "JTAG Port Signals (top to/from phy) (for N = 0; N <= Nphy-1)" on page 120
- "Boundary Scan Port Signals (top to/from phy) (for N = 0; N <= Nphy-1)" on page 121
- "External Protocol Override Signals (top to/from pcs)" on page 132
- "Parallel Control Register Port Signals (top to/from phy) (for N = 0; N <= Nphy-1)" on page 328
- "External ROM Control Signals (top to/from phy) (for N = 0; N <= Nphy-1)" on page 330
- "SRAM Control Signals (top to/from phy) (for N = 0; N <= Nphy-1)" on page 331
- "PHY Test Signals (top to/from phy)" on page 334
- "Resistor Tune Signals (Common Block Group) (top to/from phy)" on page 335
- "Ethernet Clock Signals (top to/from phy) (for X = 0; X <= Nlanes-1)" on page 337
- "Context Select Interface Signals (top to/from pcs) (for X = 0; X <= Nlanes-1)" on page 340
- "Context Select Interface Signals (top to/from phy) (for X = 0; X <= Nlanes-1)" on page 345
- "Scan Clock Interface Signals (top to/from pcs)" on page 346
- "SerDes PIPE Signals (top to/from pcs) (for X = 0; X <= Nlanes-1)" on page 347

4.1 PCS Configuration Signals (top to phy/pcs)

phy_laneX_rx2tx_par_lb_en (for X = 0; X <= Nlanes-1) -
phy_reset -

Table 4-1 PCS Configuration Signals (top to phy/pcs)

Port Name	I/O	Description
phy_laneX_rx2tx_par_lb_en (for X = 0; X <= Nlanes-1)	I	phy_laneX_rx2tx_par_lb_en is a PHY interface, per-lane configuration signal. For signal description, refer to the DesignWare Cores PHY Databook. **Note:** ■ Signal descriptions in the PHY databook do not have the prefix phyN_ . **Synchronous To:** None **Active State:** N/A
phy_reset	I	Power-on reset for PHY and PCS ■ This signal must be asserted after power is enabled for the PCS and PHY. ■ When phy_reset is de-asserted, the PHY goes through POR sequencing. ■ After phy_reset is deasserted, wait 10 ref_dig_clk cycles before accessing CREG interface. **Synchronous To:** Asynchronous **Active State:** High

4.2 PCS Configuration Signals (top to/from pcs)

ext_pclk_req -
pipe_laneX_cascaded_mpllb_sel (for X = 0; X <= Nlanes-1) -
pipe_laneX_cmn_refclk_mode (for X = 0; X <= Nlanes-1) -
pipe_laneX_destruct_rxmargin_mode (for X = 0; X <= Nlanes-1) -
pipe_laneX_ext_pll_mode (for X = 0; X <= Nlanes-1) -
pipe_laneX_lanepll_bypass_mode (for X = 0; X <= Nlanes-1) -
pipe_laneX_lanepll_src_sel (for X = 0; X <= Nlanes-1) -
pipe_laneX_legacy_pin_mode (for X = 0; X <= Nlanes-1) -
pipe_laneX_max_pclk_div_ratio_esm20 (for X = 0; X <= Nlanes-1) -
pipe_laneX_max_pclk_div_ratio_esm25 (for X = 0; X <= Nlanes-1) -
pipe_laneX_max_pclk_div_ratio_g1 (for X = 0; X <= Nlanes-1) -
pipe_laneX_max_pclk_div_ratio_g2 (for X = 0; X <= Nlanes-1) -
pipe_laneX_max_pclk_div_ratio_g3 (for X = 0; X <= Nlanes-1) -
pipe_laneX_max_pclk_div_ratio_g4 (for X = 0; X <= Nlanes-1) -
pipe_laneX_max_pclk_div_ratio_g5 (for X = 0; X <= Nlanes-1) -
pipe_laneX_protocol (for X = 0; X <= Nlanes-1) -
pipe_laneX_ropll_freq_mode (for X = 0; X <= Nlanes-1) -
pipe_rxX_es0_cmn_refclk_mode (for X = 0; X <= Nlanes-1) -
pipe_rxX_es_mode (for X = 0; X <= Nlanes-1) -
pipe_rxX_rec_ovrd_8b10b_decerr (for X = 0; X <= Nlanes-1) -
pipe_rxX_rec_ovrd_8b10b_uoflow (for X = 0; X <= Nlanes-1) -
pipe_rxX_rec_ovrd_en (for X = 0; X <= Nlanes-1) -
pipe_rxX_rec_ovrd_non8b10b_decerr (for X = 0; X <= Nlanes-1) -
pipe_rxX_rec_ovrd_non8b10b_uoflow (for X = 0; X <= Nlanes-1) -
upcs_pipe_config -

- upcs_max_payload_size_supt

Table 4-2 PCS Configuration Signals (top to/from pcs)

Port Name	I/O	Description
ext_pclk_req	I	External PCLK request. When asserted, the MPLL clock sources in the PHY are powered up and pipe_laneX_pclk outputs stay active, regardless of the pipe_laneX_powerdown(*) inputs. See Table "PCS Clock Dependency" for futher information. **Note:** Set ext_pclk_req to 1'b0. **Synchronous To:** Asynchronous **Active State:** High
pipe_laneX_cascaded_mpllb_sel (for X = 0; X <= Nlanes-1)	I	Per-lane cascaded mode mpllb_sel for lane X. When pipe_laneX_lanepll_bypass_mode is set to 0b, this pipe_laneX_cascaded_mpllb_sel signal is used. ■ 0: MPLLA ■ 1: MPLLB **Synchronous To:** Asynchronous **Active State:** High
pipe_laneX_cmn_refclk_mode (for X = 0; X <= Nlanes-1)	I	Elastic buffer mode for Nominal Half Full Buffer mode (Elasticity Buffer Mode is 0). This signal is "Don't care" during Nominal Empty Buffer mode (Elasticity Buffer Mode is 1). All lanes in a link must be in the same mode. ■ 1'b0: Not common Reference Clock mode ■ 1'b1: Common Reference Clock mode **Note:** ■ Any change to this input must be followed by phy_reset assertion. ■ When Common Reference Clock mode is enabled, SKIP add/rm is disabled in the Elastic Buffer. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
pipe_laneX_destruct_rxmargin_mode (for X = 0; X <= Nlanes-1)	I	Rx Margining mode for laneX; All lanes in a link must be in the same mode. ■ 0: Non-destructive margining (=Independent sampler) ❑ Changing margin offset does not impact to rx data path. ❑ Error is detected by PHY ❑ Error count is passed to MAC through message bus ■ 1: Destructive margining (=Dependent sampler) ❑ Changing margin offset impacts rx data path ❑ Error is detected by MAC (receiver error) ❑ MAC has error counter that counts transitions from state L0 to Recovery **Note:**This signal can be changed when phy_reset or pipe_laneX_reset_n is asserted. This signal exists when macro DWC_UPCS_DESTRUCTIVE_MARGINING is defined. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_laneX_ext_pll_mode (for X = 0; X <= Nlanes-1)	I	PCLK as PHY input External PLL mode. ■ Set this signal to 0b when "PHY owned PLL and IN_PCLK SOURCE MAX_PCLK MODE" is used. ■ Set this signal to 1b when "PHY owned PLL and IN_PCLK SOURCE MPLL_CLK MODE" or "External PLL and PHY PLL" is used. This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_laneX_lanepll_bypass_mode[4:0] (for X = 0; X <= Nlanes-1)	I	Per-lane and per-data rate lane PLL bypass mode for lane X. ■ 0: cascaded mode (MPLLA/B and lane ROPLL used) ■ 1: bypass mode (MPLLA/B is used and lane ROPLL is bypassed) Bit assignment: ■ Bit 0: Gen1 ■ Bit 1: Gen2 ■ Bit 2: Gen3 ■ Bit 3: Gen4 ■ Bit 4: Gen5, esm20, esm25 **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
pipe_laneX_lanepll_src_sel[3:0] (for X = 0; X <= Nlanes-1)	I	Per-lane ropllclk source select for lane X. ■ When using u2u3 config, see chapter U2/U3 for the setting. ■ When DWC_PIPE_VER_5 is defined and using External PLL and PHY PLL, this signal should be set to each lane num (e.g. pipe_lane0_lanepll_src_sel=0, pipe_lane_lanepll_src_sel=1, ...) ■ Otherwise, Lanes which are in the same link should select the source clock from the lowest lane of the link (e.g. for 2x2 links pipe_lane0/1_lanepll_src_sel=0, pipe_lane2/3_lanepll_src_sel=2) ■ **Note:**This signal is used to determine the bifurcation use mode. ■ For more details, see "Per-Lane ropll_clk source select" section. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_laneX_legacy_pin_mode (for X = 0; X <= Nlanes-1)	I	Legacy Pin interface mode. Same value must be used for all lanes in a link. ■ 0: Low Pin Count(LPC) interface mode ■ 1: Legacy Pin interface mode This signals exists if RTL macro DWC_PIPE_VER_5 is defined. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** High
pipe_laneX_max_pclk_div_ratio_esm20[4:0] (for X = 0; X <= Nlanes-1)	I	Max PCLK DIV_RATIO for ESM20 data rate. ■ This signal is used for DIV_RATIO for Max PCLK for the data rate. ■ Max PCLK frequency = ropll_word_clk frequency / DIV_RATIO. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** None **Active State:** N/A
pipe_laneX_max_pclk_div_ratio_esm25[4:0] (for X = 0; X <= Nlanes-1)	I	Max PCLK DIV_RATIO for ESM25 data rate. ■ This signal is used for DIV_RATIO for Max PCLK for the data rate. ■ Max PCLK frequency = ropll_word_clk frequency / DIV_RATIO. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
pipe_laneX_max_pclk_div_ratio_g1[4:0] (for X = 0; X <= Nlanes-1)	I	Max PCLK DIV_RATIO for gen1 data rate. ■ This signal is used for DIV_RATIO for Max PCLK for the data rate. ■ Max PCLK frequency = ropll_word_clk frequency / DIV_RATIO. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** None **Active State:** N/A
pipe_laneX_max_pclk_div_ratio_g2[4:0] (for X = 0; X <= Nlanes-1)	I	Max PCLK DIV_RATIO for gen2 data rate. ■ This signal is used for DIV_RATIO for Max PCLK for the data rate. ■ Max PCLK frequency = ropll_word_clk frequency / DIV_RATIO. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** None **Active State:** N/A
pipe_laneX_max_pclk_div_ratio_g3[4:0] (for X = 0; X <= Nlanes-1)	I	Max PCLK DIV_RATIO for gen3 data rate. ■ This signal is used for DIV_RATIO for Max PCLK for the data rate. ■ Max PCLK frequency = ropll_word_clk frequency / DIV_RATIO. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** None **Active State:** N/A
pipe_laneX_max_pclk_div_ratio_g4[4:0] (for X = 0; X <= Nlanes-1)	I	Max PCLK DIV_RATIO for gen4 data rate. ■ This signal is used for DIV_RATIO for Max PCLK for the data rate. ■ Max PCLK frequency = ropll_word_clk frequency / DIV_RATIO. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** None **Active State:** N/A
pipe_laneX_max_pclk_div_ratio_g5[4:0] (for X = 0; X <= Nlanes-1)	I	Max PCLK DIV_RATIO for gen5 data rate. ■ This signal is used for DIV_RATIO for Max PCLK for the data rate. ■ Max PCLK frequency = ropll_word_clk frequency / DIV_RATIO. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
pipe_laneX_protocol[1:0] (for X = 0; X <= Nlanes-1)	I	Protocol/mode for lane X; PIPE signal "PHY Mode(1:0)" (Sec 6.1, PIPE 4.3 or later). ■ Selects the PHY operating mode. ■ For PCIe operation, set to 2'b00. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_laneX_ropll_freq_mode (for X = 0; X <= Nlanes-1)	I	Ropll Clock Frequency Mode. ■ This signal is used for special case that controller needs 2000 MHz pipe_laneX_max_pclk. ■ This signal controls ropll_word_clk frequency for PCIe gen3, gen4 and gen5 which is source clock of pipe_laneX_max_pclk. ■ If set to 0b, ropll_word_clk frequency is 1000MHz for PCIe gen3, gen4 and gen5. ■ If set to 1b, ropll_word_clk frequency is 2000MHz for PCIe gen3, gen4 and gen5. ■ **Note:** ❑ ropll_word_clk frequency is always 1000MHz for PCIe gen1 and gen2 regardless of this signal. ❑ ropll_word_clk frequency is always 625 MHz / 781.25 MHz for esm20 and esm25 regardless of this signal. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** None **Active State:** N/A
pipe_rxX_es0_cmn_refclk_mode (for X = 0; X <= Nlanes-1)	I	Elastic buffer mode for Nominal Half Full Buffer mode (Elasticity Buffer Mode is 0). This signal is "Don't care" during Nominal Empty Buffer mode (Elasticity Buffer Mode is 1). All lanes in a link must be in the same mode. ■ 1'b0: Nominal Half Full Buffer mode ■ 1'b1: Common Reference Clock mode **Note:** ■ Any change to this input must be followed by phy_reset assertion. ■ When Common Reference Clock mode is enabled, SKIP add/rm is disabled in the Elastic Buffer. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
pipe_rxX_es_mode (for X = 0; X <= Nlanes-1)	I	Elasticity buffer mode for lane X; PIPE 4.3 signal "Elasticity Buffer Mode" (Sec 6.1, PIPE 4.3). ■ If macro DWC_PCIE_PIPE5_LPC is defined, and pipe_laneX_legacy_pin_mode = 1'b1, this signal controls the Elasticity Buffer operating mode, otherwise PHY Registers in the Message Bus Address space control the mode. ■ If macro DWC_PCIE_PIPE5_LPC is not defined, this signal controls the Elasticity Buffer operating mode. Selects Elasticity Buffer operating mode. ■ 1'b0: Nominal Half Full Buffer mode ■ 1'b1: Nominal Empty Buffer mode **Note:** This signal can be changed when pipe_laneX_reset_n is asserted or when pipe_laneX_rate is changed. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_rxX_rec_ovrd_8b10b_decerr[1:0] (for X = 0; X <= Nlanes-1)	I	RX data path error recovery mode for 8b10b decode error or disparity error. ■ 0: continue working (same as default behavior) ■ 1: RX data path recover without CDR reset ■ 2: RX data path recover with CDR reset ■ 3: Reserved **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_rxX_rec_ovrd_8b10b_uoflow[1:0] (for X = 0; X <= Nlanes-1)	I	RX data path error recovery mode for 8b10b underflow or overflow. ■ 0: continue working (same as default behavior) ■ 1: RX data path recover without CDR reset ■ 2: RX data path recover with CDR reset ■ 3: Reserved **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
pipe_rxX_rec_ovrd_en (for X = 0; X <= Nlanes-1)	I	RX data path error recovery mode Override Enable. ■ When this signal is asserted, pipe_rxX_rec_ovrd_8b10b_decerr(*), pipe_rxX_rec_ovrd_8b10b_uoflow(*), pipe_rxX-_rec_ovrd_non8b10b_decerr(*) and pipe_rxX-_rec_ovrd_non8b10b_uoflow(*) override signals are enabled. ■ When this signal is de-asserted, "default behavior" of description of pipe_rxX_rec_ovrd_8b10b_decerr(*), pipe_rxX-_rec_ovrd_8b10b_uoflow(*), pipe_rxX_rec_ovrd_non8b10b_de-cerr(*) and pipe_rxX_rec_ovrd_non8b10b_uoflow(*) signals are valid. ■ In loopback slave state, RX data path always continue working regardless of these signals setting. **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_rxX_rec_ovrd_non8b10b_decerr[1:0] (for X = 0; X <= Nlanes-1)	I	RX data path error recovery mode for non-8b10b sync header error / skip error. ■ 0: continue working ■ 1: RX data path recover without CDR reset (same as default behavior) ■ 2: RX data path recover with CDR reset ■ 3: Reserved **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_rxX_rec_ovrd_non8b10b_uoflow[1:0] (for X = 0; X <= Nlanes-1)	I	RX data path error recovery mode for non-8b10b underflow or overflow. ■ 0: continue working ■ 1: RX data path recover without CDR reset (same as default behavior) ■ 2: RX data path recover with CDR reset ■ 3: Reserved **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
upcs_max_payload_size_supt[2:0]	O	Maximum payload size supported. ■ This signal encodes the maximum payload size that is supported. ■ Currently, the PCIe 5.0 PCS supports a maximum 4,096 bytes. ■ For more information about supported payload sizes, see "Device Configuration" section). ■ 3'b000: 128 bytes ■ 3'b001: 256 bytes ■ 3'b010: 512 bytes ■ 3'b011: 1,024 bytes ■ 3'b100: 2,046 bytes ■ 3'b101: 4,096 bytes **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
upcs_pipe_config[15:0]	I	PCS PIPE configuration. ■ When upcs_pipe_config[0] is set to 1 and the lane is disabled via the PIPE specification method (TxElecIdle = 1 and TxCompliance = 1), the PCS will keep pipe_laneX_pclk alive (pclk as phy output) and pipe_laneX_phystatus will respond to power-down/rate/width requests. Otherwise, until the MAC de-asserts the "turned off" signaling, the PCS ignores any commands to change power-down/rate/width after being turned off. ■ upcs_pipe_config(3:1) are reserved, tie these bits to 0. ■ When upcs_pipe_config[4]=1 and operating at gen1 rate, disable PHY CDR when PHY detects Loss of Signal. When upcs_pipe_-config[4]=0, disable PHY CDR when UPCS detects an EIOS. ■ When upcs_pipe_config(5)=0 (default), the PCS isolates cells in P1.2 (For PIPE 4.4.1/5/2 - P1.CPM + pipe_rxX_disable=1 + pipe_txX_disable=1). Otherwise, the rx los signals are isolated in P1.2 and the rest of the cells are isolated in P1.CPM. ■ When upcs_pipe_config[6]=0 (default), the normal 6us recalibration timeout applies. When upcs_pipe_config[6]=1, the recalibration timeout mechanism is disabled. See section "Recalibration" for more details ■ When upcs_pipe_config[7]=0 (default) and in SATA mode, the UPCS takes 9 back-to-back COMs to achieve COM alignment. When upcs_pipe_config[7]=1 and in SATA mode, the 9 back-to-back COM check is disabled in order to support local PHY in Far End Analog Loopback mode. This bit is only supported when DWC_UPCS_MULTIPROTOCOL_SUPPORT is defined. ■ When upcs_pipe_config(8) is set to 0 (default) and IPNA-ME_SWITCH_MAXPCLK is defined, clock switching functionality is disabled. - When upcs_pipe_config(8) is set to 1 and IPNA-ME_SWITCH_MAXPCLK is defined, clock switching functionality is enabled. When If IPNAME_SWITCH_MAXPCLK is not defined this bit is unused and should be tied to 1'b0. ■ upcs_pipe_config(15:9) are reserved, tie these bits to 0. **Note:** ■ If DWC_UPCS_PERLINK_CLK_ARCH is defined, set upcs_pipe_-config(0) to 1'b1. ■ If DWC_PIPE_VER_5 is not defined, set upcs_pipe_config(0) to 1'b1. ■ If DWC_PIPE_VER_5 is defined and pipe_laneX_ext_pll_mode signal is 1'b0, set upcs_pipe_config(0) to 1'b1.
upcs_pipe_config[15:0]...(cont.)	I.	■ If DWC_PIPE_VER_5 is defined and pipe_laneX_ext_pll_mode signal is 1'b1, upcs_pipe_config[0] can be set to 1'b0 or 1'b1. ■ If Modified Compliance Pattern (MCP) Testing is used, it may be necessary to set upcs_pipe_config[6]=1, this disables the 6us reca-libration timer and allows the upcs to wait for longer periods before receiving the EIEOSs in the MCP. ■ Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** N/A

4.3 PIPE Interface Signals (top to/from pcs) (for X = 0; X <= Nlanes-1)

pipe_laneX_asyncpowerchangeack -
pipe_laneX_clkreq_n -
pipe_laneX_encdec_bypass -
pipe_laneX_if_width -
pipe_laneX_in_pclk -
pipe_laneX_in_pcs_clk -
pipe_laneX_m2p_messagebus -
pipe_laneX_maxpclkreq_n -
pipe_laneX_pclkchangeack -
pipe_laneX_perst_n -
pipe_laneX_powerdown -
pipe_laneX_rate -
pipe_laneX_reset_n -
pipe_laneX_tx2rx_loopbk -
pipe_rxX_blk_align_ctl -
pipe_rxX_disable -
pipe_rxX_eq_eval -
pipe_rxX_eq_in_prog -
pipe_rxX_eq_invld_req -
pipe_rxX_eq_training -
pipe_rxX_polarity -
pipe_rxX_sris_mode_en -
pipe_rxX_standby -
pipe_rxX_termination -
pipe_txX_compliance -
pipe_txX_data -
pipe_txX_datak -
pipe_txX_datavalid -
pipe_txX_deemph -
pipe_txX_detectrx -
pipe_txX_disable -
pipe_txX_elecidle -
pipe_txX_eq_preset -
pipe_txX_eq_preset_coeff_req -
pipe_txX_margin -
pipe_txX_ones_zeros -
pipe_txX_pattern -
pipe_txX_startblock -
pipe_txX_swing -
pipe_txX_syncheader -

- pipe_laneX_clkack_n
- pipe_laneX_databuswidth
- pipe_laneX_max_pclk
- pipe_laneX_maxpclkack_n
- pipe_laneX_p2m_messagebus
- pipe_laneX_pclk
- pipe_laneX_pclkchangeok
- pipe_laneX_phystatus
- pipe_laneX_power_present
- pipe_laneX_ref_clk_req_n
- pipe_rxX_align_detect
- pipe_rxX_data
- pipe_rxX_datak
- pipe_rxX_datavalid
- pipe_rxX_ebuff_location
- pipe_rxX_elecidle
- pipe_rxX_eq_dir_change
- pipe_rxX_eq_fig_merit
- pipe_rxX_standby_status
- pipe_rxX_startblock
- pipe_rxX_status
- pipe_rxX_syncheader
- pipe_rxX_valid
- pipe_txX_eq_preset_coeff
- pipe_txX_eq_preset_coeff_vld
- pipe_txX_eq_fs
- pipe_txX_eq_lf

Table 4-3 PIPE Interface Signals (top to/from pcs) (for X = 0; X <= Nlanes-1)

Port Name	I/O	Description
pipe_laneX_asyncpowerchangeack	I	Power change acknowledgment for lane X; PIPE signal "AsyncPowerChangeAck" (Sec 6.1, PIPE 4.3 or later). PCIe: ■ For PIPE 4.4.1 or later, used when transitioning between two power states without PCLK. ■ After the PHY asserts PhyStatus to acknowledge the power state change, the MAC responds by asserting pipe_laneX_asyncpower-changeack until the MAC samples PhyStatus deasserted. ■ For PIPE 4.3, reserved; tie to 1'b0. **Synchronous To:** Asynchronous **Active State:** High
pipe_laneX_clkack_n	O	Clock acknowledge for lane X. PCIe: ■ This is a side-band signal needed for the PIPE 4.2 controller to enter and exit P1.CPM, P1.1, and P1.2 power states. ■ For information about using this signal, see "PCIe L1 Substate Tran-sitions" section. ■ When integrating with a PIPE 4.3 or later controller, this output can be ignored. **Synchronous To:** Asynchronous **Active State:** Low
pipe_laneX_clkreq_n	I	Clock request for lane X. PCIe: ■ This is a side-band signal that a PIPE 4.2 controller needs to enter and exit P1.CPM, P1.1, and P1.2 power states. ■ When integrating with a PIPE 4.4.1/4.3-compliant controller, set this input to an asserted value of 0. ■ For information about using this signal, see "PCIe L1 Substate Tran-sitions" section. **Synchronous To:** Asynchronous **Active State:** Low
pipe_laneX_databuswidth[1:0]	O	Bus width configuration for lane X; PIPE signal "DataBusWidth(*)" (Sec 6.1,PIPE 4.3 or later). This field reports the width of the data bus configured for the PHY. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
pipe_laneX_encdec_bypass	I	Encode and decode bypass for lane X; PIPE signal "EncodeDecodeBypass" (Sec 6.1, PIPE 4.3 or later). Bypasses the 8b10 and block encoding/decoding in the PCIe 5.0 PCS. **Note:** ■ If macro DWC_PCIE_PIPE5_LPC is defined, tie this signal to 0. ■ In PCIe operation, the current PCS implementation does not support pipe_laneX_encdec_bypass in Gen2 CP patterns; therefore, tie this signal to 0. ■ Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** High
pipe_laneX_if_width[1:0]	I	PIPE interface width for lane X; PIPE signal "Width" (Sec 6.1, PIPE 4.3 or later). ■ Controls the data width of pipe_txX_data and pipe_rxX_data buses. ■ The width change must be simultaneously performed with the rate change when operating in SerDes PIPE. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_laneX_in_pclk	I	Parallel clock for lane X; PIPE signal "PCLK" as PHY input (Sec 6.1, PIPE 4.4.1/4.3). ■ This signal is the parallel interface data clock. ■ The pipe_laneX_in_pclk, pipe_laneX_in_pcs_clk and pipe_laneX_p-ma_in_clk inputs for all lanes that are part of the same link must be phase-synchronous. ■ This signal is Scan Clock Mux-ed clock. ■ For a list of pclk frequencies, see table "pipe_laneX_pclk / pipe_laneX_in_pclk Frequencies". This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_laneX_in_pcs_clk	I	PCS clock as PHY input. ■ This is a UPCS internal clock which is phase-synchronous to pipe_laneX_in_pclk. ■ This signal is Scan Clock Mux-ed clock. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined and DWC_UPCS_PIPEWIDTH_2S_2S_4S_4S_4S is not defined and DWC_UPCS_PIPEWIDTH_4S_4S_8S_8S_8S is not defined and DWC_UPCS_PIPEWIDTH_4S_4S_4S_8S_8S is not defined and DWC_UPCS_PIPEWIDTH_1S_2S_2S_2S_4S_PCLK1G is not defined. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
pipe_laneX_m2p_messagebus[7:0]	I	Mac-to-PHY Message Bus for lane X PIPE signal "M2P_MessageBus" (Sec 6.1, PIPE 4.3 or later). ■ MAC-driven Message Bus interface. ■ PIPE 4.4.1 or later, supports all message bus commands. For PIPE 4.3, the supported command mapping is as follows: ■ NOP ■ Write-committed, non-posted ■ write_ack For more information, see "Lane Margining at Receiver" section. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_laneX_max_pclk	O	Maximum PCLK; PIPE signal "Max PCLK" (Sec 6.1, PIPE 4.3 or later). ■ The frequency of this output clock is the maximum frequency supported for that protocol. ■ Contrary to what is indicated in the PIPE 4.4.1 or later, specification, the frequency does not change with rate. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_laneX_maxpclkack_n	O	PIPE spec high level interface "PLL Control" (PIPE version 5.2). This signal can be used only when DWC_PIPE_VER_5 is defined and PHY owned PLL and IN_PCLK SOURCE MAX_PCLK MODE. DWC_PIPE_VER_5 is defined and PHY owned PLL and IN_PCLK SOURCE MAX_PCLK MODE: ■ MAC per-link input MAXPCLKACK_N signal should connect to a master lane of pipe_laneX_maxpclkreq_n UPCS output signal which is the lowest lane of the link. ■ For the other non-master lanes, UPCS output signal can be opened. (DWC_PIPE_VER_5 is defined and PHY owned PLL and IN_PCLK SOURCE MPLL_CLK MODE) || (DWC_PIPE_VER_5 is defined and External PLL and PHY PLL): ■ UPCS output signal can be opened. This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
pipe_laneX_maxpclkreq_n	I	PIPE spec high level interface "PLL Control" (PIPE version 5.2). This signal can be used only when DWC_PIPE_VER_5 is defined and PHY owned PLL and IN_PCLK SOURCE MAX_PCLK MODE. DWC_PIPE_VER_5 is defined and PHY owned PLL and IN_PCLK SOURCE MAX_PCLK MODE: ■ MAC per-link output MAXPCLKREQ_N signal should connect to a master lane of pipe_laneX_maxpclkreq_n UPCS input signal which is the lowest lane of the link. ■ Default value of this signal must be 0b. ■ This signal must be updated from 0b to 1b after controller receives pulsed PhyStatus for PCLK-OFF powerdown state transition(PCIe: P1_CPM, P2, P2_CPM, P2_NOBEACON). ■ This signal must be updated from 1b to 0b before controller exit from PCLK-OFF powerdown state transition(PCIe: P1_CPM, P2, P2_CPM, P2_NOBEACON). ■ A four-way request-ack handshake must be followed between pipe_laneX_maxpclkreq_n and pipe_laneX_maxpclkack_n. ■ When this signal is changed and until the handshake completes, any other PIPE input signals cannot be changed. ■ For the other non-master lanes, pipe_laneX_maxpclkreq_n must be tied to 1b. (DWC_PIPE_VER_5 is defined and PHY owned PLL and IN_PCLK SOURCE MPLL_CLK MODE) || (DWC_PIPE_VER_5 is defined and External PLL and PHY PLL): ■ pipe_laneX_maxpclkreq_n for all lanes must be tied to 1b. This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_laneX_p2m_messagebus[7:0]	O	PHY-to-Mac Message Bus for lane X PIPE signal "P2M_MessageBus" (Sec 6.1, PIPE 4.3 or later). ■ PHY-driven Message Bus interface. ■ PIPE 4.4.1 or later, supports all message bus commands. For PIPE 4.3, the supported command mapping is as follows: ■ NOP ■ Write-committed, non-posted ■ write_ack For more information, see "Lane Margining at Receiver" section. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A

Port Name	I/O	Description
pipe_laneX_pclk	O	Parallel clock for lane X; PIPE signal "PCLK" (Sec 6.1, PIPE 4.3 or later). ■ This signal is the parallel interface differential data clock. ■ The pipe_laneX_clk outputs from all lanes that are part of the same link are phase-synchronous; that is, pipe_laneX_clk from one lane can be used to sample and drive signals on the other lanes. ■ For a list of pclk frequencies, see table "pipe_laneX_pclk / pipe_laneX_in_pclk Frequencies". This signal exists if RTL macro DWC_PIPE_VER_5 is not defined. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_laneX_pclkchangeack	I	PIPE signal "PclkChangeAck" (PIPE version 5.2). ■ MAC asserts this signal when a PCLK rate change or rate change is complete and stable. ■ UPCS waits pipe_laneX_pclkchangeack==1 before deassertng pipe_laneX_pclkchangeok. **Attention:** UPCS does NOT wait pipe_laneX_pclkchangeack==0 before asserting pipe_laneX_pclkchangeok. It is the requirement for MAC to deassert pipe_laneX_pclkchangeack before new Rate Change. ■ For the case MAC does not do the handshake like "PCLK as PHY input w/PHY owned PLL(Dynamic MAX PCLK frequency)" mode, pipe_laneX_pclkchangeack input signal must be tied to 1b. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Synchronous To:** pipe_laneX_in_pclk **Active State:** N/A
pipe_laneX_pclkchangeok	O	PIPE signal "PclkCHangeOk" (PIPE version 5.2). ■ UPCS asserts this signal when it is ready for the MAC to change the PCLK rate. ■ For the case MAC does not do the handshake like "PCLK as PHY input w/PHY owned PLL(Dynamic MAX PCLK frequency)" mode, this signal can be opened. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined. **Synchronous To:** pipe_laneX_in_pclk **Active State:** N/A

Port Name	I/O	Description
pipe_laneX_perst_n	I	PERST# for lane X; PCI Express CEM specification signal PERST#. ■ When phy_reset is asserted, all pipe_laneX_perst_n signals must be asserted. ■ When all lane pipe_laneX_perst_n signals which belong to phyN are asserted, the signal is treated as phy_reset for the phyN, and the PHY restarts from initial POR sequence. ■ After pipe_laneX_perst_n is deasserted, wait 10 ref_dig_clk cycles before accessing CREG interface. ■ If DWC_DEEP_PM_SUP is defined, this signal is not fully supported: pipe_laneX_perst_n = ~(Power-on reset) ■ If DWC_DEEP_PM_SUP is NOT defined, then the following rules apply. ■ If input phyN_multi_perst_en port does not exist, then PERST# is supported on a per-hard macro (PMA) basis ❑ All pipe_laneX_perst_n signals which belong to the same phy macro must be asserted/de-asserted at the same time. ❑ All pipe_laneX_perst_n signals which are in the same link must be asserted or de-asserted at the same time. ■ If input phyN_multi_perst_en port does exist, then PERST# is supported on a per-lane hard macro (PMA) basis ❑ pipe_laneX_perst_n signals which are in the same link must be asserted or de-asserted at the same time. **Synchronous To:** Asynchronous **Active State:** Low
pipe_laneX_phystatus	O	PHY status for lane X; PIPE signal "PhyStatus" (Sec 6.1, PIPE 4.3 or later). Acknowledgment signal for the following control inputs: ■ pipe_laneX_if_width(*) ■ pipe_laneX_powerdown(*) ■ pipe_laneX_rate(*) ■ pipe_rxX_eq_eval ■ pipe_txX_detectrx (when pipe_laneX_powerdown(*) = P1) **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_laneX_power_present	O	Power present for lane X; PIPE signal "PowerPresent" (Sec 6.1, PIPE 4.3 or later). ■ This signal is only used for USB which is not supported by this design. ■ This output is directly connected to input phy_laneX_power_present. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High

Port Name	I/O	Description
pipe_laneX_powerdown[3:0]	I	Power state for lane X; PIPE signal "PowerDown(*)" (Sec 6.1, PIPE 4.3 or later). Controls the PHY power states. The power-state mapping is as follows: ■ **PIPE 5.2/4.4.1- PIPE 4.3** ■ P0: 4'b0000 - P0: 4'b0000 ■ P0s: 4'b0001 - P0s: 4'b0001 ■ P1: 4'b0010 - P1: 4'b0010 ■ P2: 4'b0011 - P2: 4'b0011 ■ P1.CPM: 4'b0100a - P1.CPM: 4'b0100b ■ P1.1: 4'b0100a - P1.1: 4'b0101 ■ P1.2: 4'b0100a - P1.2: 4'b1100 ■ P2.CPM: 4'b1101 - P2.CPM: 4'b1101 ❑ RxEIDetectDisable and TxCommonModeDisable signals were added in PIPE 4.4. ❑ The P1.CPM power-down encoding is used along with RxEIDetectDisable and TxCommonModeDisable to encode P1.1 and P1.2. For information about power state transitions, see "Power State Transitions" section. **Synchronous To:** Synchronous to pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk unless exiting a powerdown state where pclk has been removed; then treat this signal as asynchronous. **Active State:** N/A
pipe_laneX_rate[2:0]	I	Protocol/mode for lane X; PIPE signal "Rate(*)" (Sec 6.1, PIPE 4.3 or later). Controls the link signaling rate. PCIe: ■ 3'b000: 2.5 Gbps ■ 3'b001: 5.0 Gbps ■ 3'b010: 8.0 Gbps ■ 3'b011: 16.0 Gbps ■ 3'b100: 32.0 Gbps ESM: ■ 3'b000: 2.5 GT/s ■ 3'b001: 5.0 GT/s ■ 3'b010: ESM_DATA_RATE0 (from PHY Message Bus register) ■ 3'b011: ESM_DATA_RATE1 (from PHY Message Bus register) **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A

Port Name	I/O	Description
pipe_laneX_ref_clk_req_n	O	Reference clock request for lane X. This is a side-band signal that the following PIPE controllers need to disable the reference clock input to the PHY: PCIe: 4.2 or later ■ The reference clock input to the PHY can be disabled when pipe_laneX_clkreq_n = 1 and pipe_laneX_ref_clk_req_n = 1 (acknowledge signal) for all lanes connected to the PHY. ■ For information about using this signal, see "Reference Clock Input" and "PCIe L1 Substate Transitions" sections. If DWC_DEEP_PM_SUP is defined, see "Deep Pm Support" section. **Synchronous To:** Asynchronous **Active State:** Low
pipe_laneX_reset_n	I	Reset for lane X; PIPE signal "Reset#" (Sec 6.1, PIPE 4.3 or later). ■ pipe_laneX_reset_n = ~(Power-on reset | ~PERST# | Controller Specific reset) ■ When phy_reset or pipe_laneX_perst_n is asserted, pipe_laneX_reset_n must be asserted. ■ pipe_laneX_reset_n signals which are in the same link must be asserted or de-asserted at the same time. ■ Resets the transmitter and receiver. ■ The PCS uses this reset. The pipe_laneX_phystatus output from the PCS is asserted asynchronously with pipe_laneX_reset_n assertion and is de-asserted after POR sequencing is complete. **Synchronous To:** Asynchronous **Active State:** Low
pipe_laneX_tx2rx_loopbk	I	TX-to-RX loopback enable for lane X. ■ When asserted, this input turns on the TX-to-RX serial loopback within the PHY. ■ This signal is for debug purposes only. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_rxX_align_detect	O	RX ALIGN symbol detected for lane X; PIPE 4.3 signal "AlignDetect" (Sec 6.1, PIPE 4.3 or later). ■ Indicates receiver detection of an Align. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ If only SerDes Pipe Architecture is used, this output can be left unconnected. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High

Port Name	I/O	Description
pipe_rxX_blk_align_ctl	I	RX block align control for lane X; PIPE signal "BlockAlignControl" (Sec 6.1, PIPE 4.3 or later). Controls whether the PHY performs block alignment. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, tie this signal to 0. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_rxX_data[39:0]	O	RX data for lane X; PIPE signal "RxData(31:0)" (Sec 6.1, PIPE 4.3 or later). RX parallel data. The valid data bits for pipe_rx_data(39:0) are as follows: **pipe_laneX_encdec_bypass - pipe_laneX_if_width(1:0) - Valid bits of pipe_rxX_data(39:0)** ■ 0 - 00 - (7:0) ■ 0 - 01 - (15:0) ■ 0 - 10 - (31:0) ■ 1 - 00 - (9:0) ■ 1 - 01 - (19:0) ■ 1 - 10 - (39:0) This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ If only SerDes Pipe Architecture is used, this output can be left unconnected. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_rxX_datak[3:0]	O	RX data control for lane X; PIPE signal "RxDataK(*)" (Sec 6.1, PIPE 4.3 or later). ■ Data/control for the symbols of received data. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ If only SerDes Pipe Architecture is used, this output can be left unconnected. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A

Port Name	I/O	Description
pipe_rxX_datavalid	O	RX data valid; PIPE signal "RxDataValid" (Sec 6.1, PIPE 4.3 or later). ■ RX data valid to throttle RX data for non-8b10b encoding modes. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ If only SerDes Pipe Architecture is used, this output can be left unconnected. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_rxX_disable	I	RX disable control for lane X. ■ This is a side-band signal that a PIPE 5.2/4.4.1/4.2 controller needs to enter and exit P1.1 and P1.2 power states. ■ When asserted, all RX lane circuitry (including RX Electrical Idle Exit Detection) for lane X is disabled. ■ When integrating with a PIPE 4.3 controller, pipe_rxX_disable must be set to 0. ■ When integrating with a PIPE 5.2/4.4.1 controller, pipe_rxX_disable must be asserted only in L1.1 or L1.2 idle ■ For P1.1 and P1.2 entry and exit sequences, see "PCIe L1 Substate Transitions" section. **Synchronous To:** Asynchronous **Active State:** High
pipe_rxX_ebuff_location[8:0]	O	Entries in elastic buffer for lane X; PIPE signal "ElasticBufferLocation(*)" (Sec 6.1, PIPE 4.3 or later). Encodes the number of entries currently in the elastic buffer. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, ignore these outputs. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_rxX_elecidle	O	RX electrical idle detection for lane X; PIPE signal "RxElecIdle" (Sec 6.1, PIPE 4.3 or later). ■ Indicates receiver detection of an electrical idle. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ If only SerDes Pipe Architecture is used, this output can be left unconnected. **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
pipe_rxX_eq_dir_change[5:0]	O	Direction change commands for link partner's TX equalization coefficients for lane X; PIPE signal "LinkEvaluationFeedbackDirectionChange(*)" (Sec 6.1, PIPE 4.3 or later). Provides link equalization evaluation feedback in the direction change format. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, ignore these outputs. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_rxX_eq_eval	I	RX equalization enable for lane X; PIPE signal "RxEqEval" (Sec 6.1, PIPE 4.3 or later). When the MAC asserts this signal, the PHY starts evaluation of the far-end transmitter TX EQ settings. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, tie this signal to 0. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_rxX_eq_fig_merit[7:0]	O	Figure of Merit for equalization settings of lane X; PIPE signal "LinkEvaluationFeedbackFigureMerit(*)" (Sec 6.1, PIPE 4.3 or later). Provides the PHY link equalization evaluation Figure of Merit value. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, ignore these outputs. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_rxX_eq_in_prog	I	RX equalization in progress for lane X; PIPE signal "RxEqInProgress" (Sec 6.1, PIPE 4.3 or later). ■ This input drives phy input rxX_adapt_in_prog. ■ This input should be connected to controller. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, tie this signal to 0. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High

Port Name	I/O	Description
pipe_rxX_eq_invld_req	I	RX equalization invalid request; PIPE signal "InvalidRequest" (Sec 6.1, PIPE 4.3 or later). ■ This input should be connected to controller. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, tie this signal to 0. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_rxX_eq_training	I	RX equalization training mode enable for lane X; PIPE signal "RxEqTraining" (Sec 6.1, PIPE 4.3 or later). Reserved. Set to 0. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_rxX_polarity	I	RX polarity inversion enable for lane X; PIPE signal "RxPolarity" (Sec 6.1, PIPE 4.3 or later). Requests the PHY to perform a polarity inversion on the received data. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, tie this signal to 0. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_rxX_sris_mode_en	I	RX SRIS ECM mode enable for lane X. If IPNAME_ESTORE_DEPTH_CONTROL_SUPPORT is NOT defined and this signal is asserted, the PHY CDR and the elastic buffer are configured to recover Independent Spread Spectrum Data. If IPNAME_ESTORE_DEPTH_CONTROL_SUPPORT is defined, only the PHY CDR is configured to recover Independent Spread Spectrum Data and the elastic buffer depth is controlled by message bus register "Elastic Buffer Control". **Note:** Any change to this input must be followed by phy_reset assertion. **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
pipe_rxX_standby	I	RX standby enable for lane X; PIPE signal "RxStandby" (Sec 6.1, PIPE 4.3 or later). ■ When asserted, the RX CDR for lane X is disabled. ■ PHY relies on RxStandby to de-assert RxValid when operating in SerDes PIPE. ■ The controllers must assert RxStandby to disable the PHY Rx during P0/P0s. ■ RxStandby must be de-asserted to activate the PHY Rx. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_rxX_standby_status	O	RX standby status for lane X; PIPE signal "RxStandbyStatus" (Sec 6.1, PIPE 4.3 or later). When asserted, indicates that the RX has entered Standby state in response to pipe_rxX_standby assertion. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High

Port Name	I/O	Description
pipe_rxX_startblock	O	RX start block indicator for lane X; PIPE signal "RxStartBlock" (Sec 6.1, PIPE 4.3 or later). ■ Enables PHY to provide MAC with the starting byte for a 128b block. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ If only SerDes Pipe Architecture is used, this output can be left unconnected. ■ If DWC_PCIE_PIPE_SERDES is defined ❑ If only SerDes Pipe Architecture is used, this output can be left unconnected. ❑ If pipe_laneX_serdes_mode=1, this signal is unused ❑ If pipe_laneX_serdes_mode=0, and pipe_laneX_sync_hdr_byp_en=1, this output indicates Start Block in128b encoding mode. This signal can be asserted at incorrect byte boundary for PCIe Gen5 due to wrong block alignment inside PHY (see note below) -- If pipe_laneX_serdes_mode=0, and pipe_laneX_sync_hdr_byp_en=0, this output indicates Start Block in non-8b10b encoding modes ■ If DWC_PCIE_PIPE_SERDES is NOT defined ❑ This output indicates Start Block in non-8b10b encoding modes **NOTE when sync header bypass is enabled at PCIe Gen5:** There is a possibility the UPCS will align to the incorrect byte boundary. This only occurs when the UPCS starts receiving valid data from the PHY in the middle of two consecutive Electrical Idle Exit Ordered Sets (EIEOS). The controller should keep the UPCS in block alignment phase until the Start of Data Stream is received. The UPCS will detect correct alignment once the next two consecutive EIEOSs are received. < br> ■ When block align enable =1 ❑ UPCS detects block alignment (EIEOS detection) ❑ UPCS detects EIOS to turn off CDR and to de-assert rx_valid / rx_serdes_valid ■ When block align enable =0 ❑ UPCS does NOT detect block alignment ❑ UPCS detects EIOS to turn off CDR and to de-assert rx_valid / rx_serdes_valid **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_rxX_status[2:0]	O	Receive status and error codes for lane X; PIPE signal "RxStatus(*)" (Sec 6.1, PIPE 4.3 or later). Encodes receiver status and error codes for the received datastream when receiving data. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A

Port Name	I/O	Description
pipe_rxX_syncheader[3:0]	O	RX SYNC header for lane X; PIPE signal "RxSyncHeader(*)" (Sec 6.1, PIPE 4.3 or later). ■ Provides the SYNC header for the PHY to use in the next block. ■ The pipe_rxX_syncheader(*) bits are valid for PCIe. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ If only SerDes Pipe Architecture is used, this output can be left unconnected. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_rxX_termination	I	RX termination enable for lane X PIPE signal "RX Termination" (Sec 6.1, PIPE 4.3 or later) This signal is only supported in PCIe mode, If not using PCIe, this signal can be tied to 1'b0 or 1'b1. PHY rx termination is normally controlled by the UPCS. When DWC_INC_RETIMER_PINS is defined (see "Device Configuration"), top-level input pipe_rxX_termination can be used to override the termination. The following is a prioritized list (high to low) of how phyN.rxX_term_en is driven when DWC_INC_RETIMER_PINS is defined: ■ If pg_mode_en=1 and pipe_laneX_powerdown=P1.CPM + pipe_rxX_disable=1 + pipe_txX_disable=1 (P1.2 state), phyN.rxX_term_en will be driven to 1'b1 (ISOLATED) ■ If pipe_rxX_termination=0, phyN.rxX_term_en will be driven to 1'b0 ■ If pipe_laneX_reset_n=0, phyN.rxX_term_en will be driven to 1'b0 ■ If phy_reset=1, phyN.rxX_term_en will be driven to 1'b0 ■ If pg_mode_en=1 and upcs_pwr_stable=0, phyN.rxX_term_en will be driven to 1'b0 ■ Otherwise, phyN.rxX_term_en will be driven to 1'b1 The following is a prioritized list (high to low) of how phyN.rxX_term_en is driven when DWC_INC_RETIMER_PINS is not defined: ■ If pg_mode_en=1 and pipe_laneX_powerdown=P1.CPM + pipe_rxX_disable=1 + pipe_txX_disable=1 (P1.2 state), phyN.rxX_term_en will be driven to 1'b1 (ISOLATED) ■ If pipe_laneX_reset_n=0, phyN.rxX_term_en will be driven to 1'b0 ■ If phy_reset=1, phyN.rxX_term_en will be driven to 1'b0 ■ If pg_mode_en=1 and upcs_pwr_stable=0, phyN.rxX_term_en will be driven to 1'b0 ■ Otherwise, phyN.rxX_term_en will be driven to 1'b1 **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
pipe_rxX_valid	O	RX valid for lane X; PIPE signal "RxValid" (Sec 6.1, PIPE 4.3 or later). ■ Indicates symbol lock and valid data on pipe_rxX_data and pipe_rxX_datak. ■ De-asserted on loss of symbol lock, loss of block alignment, sync header error, 8b/10b decode error, 128/130 decode error, overflow or underflow of elastic buffer, reception of EIOS, or RxStandby assertion. ❑ Note: No de-assertion by 8b10b decode error, overflow, underflow, in gen1 and gen2 ■ If DWC_PCIE_PIPE_SERDES is defined ❑ If only SerDes Pipe Architecture is used, this output can be left unconnected. ❑ If pipe_laneX_serdes_mode=1, this signal is unused ❑ If pipe_laneX_serdes_mode=0, and pipe_laneX_sync_hdr_bypass_en=1, this signal can be asserted at incorrect byte boundary for PCIe Gen5 due to wrong block alignment inside PHY. See NOTE for signal pipe_rxX_datavalid for more information ❑ If pipe_laneX_serdes_mode=0, and pipe_laneX_sync_hdr_bypass_en=0, this output indicates symbol lock and valid data ■ If DWC_PCIE_PIPE_SERDES is NOT defined ❑ This output indicates symbol lock and valid data **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_txX_compliance	I	TX compliance for lane X; PIPE signal "TxCompliance" (Sec 6.1, PIPE 4.3 or later). ■ Sets the running disparity to negative. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ This input signal must be 0 when operating in SerDes PIPE. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High

Port Name	I/O	Description
pipe_txX_data[39:0]	I	TX data for lane X; PIPE signal "TxData(31:0)" (Sec 6.1, PIPE 4.3 or later). TX parallel data. The valid data bits for pipe_tx_data(39:0) are as follows: **pipe_laneX_encdec_bypass - pipe_laneX_if_width(1:0) - Valid bits of pipe_txX_data(39:0)** ■ 0 - 00 - (7:0) ■ 0 - 01 - (15:0) ■ 0 - 10 - (31:0) ■ 1 - 00 - (9:0) ■ 1 - 01 - (19:0) ■ 1 - 10 - (39:0) ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ This input signal must be 0 when operating in SerDes PIPE. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_txX_datak[3:0]	I	TX data control for lane X; PIPE signal "TxDataK(*)" (Sec 6.1, PIPE 4.3 or later). ■ Data/control for the symbols of transmit data. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ This input signal must be 0 when operating in SerDes PIPE. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_txX_datavalid	I	TX data valid; PIPE signal "TxDataValid" (Sec 6.1, PIPE 4.3 or later). ■ TX data valid to throttle TX data for non-8b10b encoding modes. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ This input signal must be 0 when operating in SerDes PIPE. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High

Port Name	I/O	Description
pipe_txX_deemph[17:0]	I	TX equalization settings; PIPE signal "TxDeemph(*)" (Sec 6.1, PIPE 4.3 or later). ■ Selects TX de-emphasis. ■ For information about TX de-emphasis, see "Transmitter Equalization Settings" section. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, tie this signal to 0. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_txX_detectrx	I	Receiver detect enable; PIPE signal "TxDetectRx/Loopback" (Sec 6.1, PIPE 4.3 or later). Instructs the PHY to either begin a receiver detection operation, to begin loopback, or to signal LFPS during P0 for USB Polling state. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_txX_disable	I	Transmitter disable. ■ This is a side-band signal that a PIPE 5.2/4.4.1/4.2 controller needs to enter and exit P1.2 power state. ■ When asserted, the TX common mode for lane X is disabled. ■ When integrating with a PIPE 4.3 controller, pipe_txX_disable must be set to 0. ■ When integrating with a PIPE 5.2/4.4.1 controller, pipe_txX_disable must be asserted only in L1.2 idle. ■ For P1.2 entry and exit sequences, see "PCIe L1 Substate Transitions" section. **Synchronous To:** Asynchronous **Active State:** High
pipe_txX_elecidle	I	TX electrical idle enable for lane X; PIPE signal "TxElecIdle" (Sec 6.1, PIPE 4.3 or later). ■ When asserted (except during loopback), forces TX output to electrical idle. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ This input signal must be 1 when operating in SerDes PIPE. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High

Port Name	I/O	Description
pipe_txX_eq_preset_coeff[17:0]	O	TX equalization preset coefficients for lane X; PIPE signal "LocalTxPresetCoefficients(*)" (Sec 6.1, PIPE 4.3 or later). These signals are TX coefficients that correspond to the pipe_txX_eq_preset(*) input and are valid when pipe_txX_eq_preset_coeff_vld is asserted. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, ignore these outputs. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_txX_eq_preset_coeff_vld	O	TX equalization preset coefficient valid for lane X; PIPE signal "LocalTxCoefficientsValid" (Sec 6.1, PIPE 4.3 or later). When this signal is asserted, pipe_txX_eq_preset_coeff(*) is valid. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, ignore these outputs. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_txX_eq_fs[5:0]	O	TX equalization FS value for lane X; PIPE signal "LocalFS(*)" (Sec 6.1, PIPE 4.3 or later). Provides the FS value for the PHY. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, ignore these outputs. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_txX_eq_lf[5:0]	O	TX equalization LF value for lane X; PIPE signal "LocalLF(*)" (Sec 6.1, PIPE 4.3 or later). Provides the LF value for the PHY. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, ignore these outputs. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A

Port Name	I/O	Description
pipe_txX_eq_preset[5:0]	I	TX equalization preset for lane X; PIPE signal "LocalPresetIndex(*)" (Sec 6.1, PIPE 4.3 or later). ■ Index for local PHY preset coefficients requested by the MAC. Bit(4) is reserved; tie this bit to 1'b0 for PIPE 4.3. PCI Express mode (ESM Enable = 0): ■ 6'h00 - 6'h0A: 8.0 GT/s Preset P0-P10 ■ 6'h0B - 6'h15: 16.0 GT/s Preset P0-P10 ■ 6'h16 - 6'h20: 32.0 GT/s Preset P0-P10 ESM mode (ESM Enable = 1): ■ 6'h00 - 6'h0A: ESM Data Rate0 Preset ■ 6'h0B - 6'h15 ESM Data Rate1 Preset **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_txX_eq_preset_coeff_req	I	TX equalization preset mapping request for lane X; PIPE signal "GetLocalPresetCoeffcients" (Sec 6.1, PIPE 4.3 or later). A MAC holds this signal high for one PCLK cycle when requesting a preset to co-efficient mapping for the preset on pipe_txX_eq_preset(*) to coefficients on pipe_txX_eq_preset_coeff(*). **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, tie this signal to 0. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_txX_margin[2:0]	I	TX margin setting for lane X; PIPE signal "TxMargin(*)" (Sec 6.1, PIPE 4.3 or later). Selects the transmitter voltage levels as defined in the PIPE 4.3 specification. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, tie this signal to 0. **Note:** If a PCIe application needs to change transmit margin and update FS/LF/coefficient for new value, the following flow must be executed while running at a speed greater than PCIe Gen2. ■ 1. Change rate from PCIe Gen3/4/5 to PCIe Gen1 speed. ■ 2. Change "Transmit Margin" register in PCIe controller (Link Control 2 Register in PCI Express base specification). ■ 3. Set "Perform Equalization" bit in PCIe controller (Link Control 3 register in PCI Express base specification). ■ 4. Initiate speed change request in PCIe controller. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A

Port Name	I/O	Description
pipe_txX_ones_zeros	I	USB TX compliance pattern enable for lane X; PIPE 4.3 signal "TxOnesZeros" (Sec 6.1, PIPE 4.3). PCIe: Reserved **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_txX_pattern[1:0]	I	TX pattern for SATA OOB for lane X; PIPE 4.3 signal "TX Pattern(*)" (Sec 6.1, PIPE 4.3). PCIe: Reserved **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A
pipe_txX_startblock	I	TX start block for lane X; PIPE signal "TxStartBlock" (Sec 6.1, PIPE 4.3 or later). ■ Enables MAC to provide the PHY with the starting byte for a block using non-8b10b data rates. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ This input signal must be 0 when operating in SerDes PIPE. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_txX_swing	I	TX swing control; PIPE signal "TxSwing" (Sec 6.1, PIPE 4.3 or later). Not supported. Reserved. **Note:** If macro DWC_PCIE_PIPE5_LPC is defined, tie this signal to 0. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** High
pipe_txX_syncheader[3:0]	I	TX sync header for lane X; PIPE signal "TxSyncHeader(*)" (Sec 6.1, PIPE 4.3 or later). ■ Provides the SYNC header for the PHY to use in the next block. ■ The bits in pipe_txX_syncheader(*) are valid for PCIe operation. ■ This signal is only used when pipe_laneX_serdes_mode=0 (Original Pipe Architecture). ■ This input signal must be 0 when operating in SerDes PIPE. **Synchronous To:** pipe_laneX_in_pclk, when macro DWC_PIPE_VER_5 is defined, otherwise pipe_laneX_pclk **Active State:** N/A

4.4 MPLL Control Signals (top to/from pcs) (for N = 0; N <= Nphy-1)

phyN_mplla_ssc_en -
phyN_mpllb_ssc_en -

Table 4-4 MPLL Control Signals (top to/from pcs) (for N = 0; N <= Nphy-1)

Port Name	I/O	Description
phyN_mplla_ssc_en	I	Spread spectrum enable (mplla_ssc_en). ■ Enables spread-spectrum clock (SSC) generation on the MPLL clocks outputs (mpll(a,b)_word_clk, mpll(a,b)_div_clk). ■ If the reference clock already has spread spectrum applied, mpll(a,b)_ssc_en must be deasserted. ■ These inputs can be changed only when the txX_mpll_en input for all lanes is de-asserted. **Note:** ■ Use of built-in SSC generation results in increased deterministic jitter (DJ). ■ The SSC profile is not synchronized across aggregated PHYs. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_mpllb_ssc_en	I	Spread spectrum enable (mpllb_ssc_en). ■ Enables spread-spectrum clock (SSC) generation on the MPLL clocks outputs (mpll(a,b)_word_clk, mpll(a,b)_div_clk). ■ If the reference clock already has spread spectrum applied, mpll(a,b)_ssc_en must be deasserted. ■ These inputs can be changed only when the txX_mpll_en input for all lanes is de-asserted. **Note:** ■ Use of built-in SSC generation results in increased deterministic jitter (DJ). ■ The SSC profile is not synchronized across aggregated PHYs. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

4.5 MPLL Control Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Table 4-5 MPLL Control Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Port Name	I/O	Description
phyN_mplla_dword_clk	O	Double-Rate word clock (mplla_dword_clk). Double-symbol rate (x2) clock derived from MPLLA. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_mplla_force_ack	O	MPLLA Force Acknowledge ■ Acknowledge output for mpll(a,b)_force_en. ■ This output asserts when the MPLL is powered up in response to mpll(a,b)_force_en assertion. ■ This output de-asserts as a response to mpll(a,b)_force_en de-assertion, and at that point the MPLL is powered down if txX_mpll_en of other lanes is de-asserted. **Note:** A four-way level handshake must be followed between mpll(a,b)_force_en and mpll(a,b)_force_ack. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
phyN_mplla_force_en	I	MPLLA force enable. When asserted, the corresponding MPLL is forced to be powered up, irrespective of the txX_mpll_en input. (DWC_DEEP_PM_SUP and DWC_PIPE_VER_5 are defined and PHY owned PLL and IN_PCLK SOURCE MPLL_CLK MODE and phyN_mplla_*_clk is used as a source of IN_PCLK): ■ Default value of this signal must be 0b. ■ Initial sequence (contact Synopsys for details): This signal can be asserted when phy_reset is de-asserted and pipe_laneX_reset_n is asserted, with writing to PHY register. Once this signal is asserted, pipe_laneX_reset_n can be de-asserted only after phyN_mplla_force_ack is asserted. ■ The signal must be kept 1b when any links which use the MPLL clock as Input PCLK are in PCLK-ON powerdown states. ■ This signal must be updated from 1b to 0b only after all links which use the MPLL clock as source of Input PCLK complete PCLK-OFF powerdown state transition(PCIe: P1_CPM, P2, P2_CPM, P2_NOBEACON/ SATA: Slumber, DevSleep, DevSleep.PG). Phystatus are 1-shot pulse with synchronous to PCLK, and PCLK can be disable only after that. ■ This signal must be updated from 0b to 1b before any links which use the MPLL clock as Input PCLK exit from PCLK-OFF powerdown states (PCIe: P1_CPM, P2, P2_CPM, P2_NOBEACON. SATA: Slumber, DevSleep, DevSleep.PG). PowerDown is synchronous to PCLK, and PCLK must be active before changing the signal. ■ A four-way request-ack handshake must be followed with phyN_mplla_force_ack signal. This signal can be changed from 0b to 1b only when phyN_mplla_force_ack==0b. This signal can be changed from 1b to 0b only when phyN_mplla_force_ack==1b. (DWC_DEEP_PM_SUP is not defined) || (DWC_PIPE_VER_5 is not defined) || (DWC_PIPE_VER_5 is defined and PHY owned PLL and and IN_PCLK SOURCE MAX_PCLK MODE) || (DWC_PIPE_VER_5 is defined and External PLL and PHY PLL) || (DWC_DEEP_PM_SUP is defined and DWC_PIPE_VER_5 is defined and PHY owned PLL and IN_PCLK SOURCE MPLL_CLK MODE and phyN_mplla_*_clk is NOT used as a source of IN_PCLK): ■ This signal must be tied to 0b. Example: For x4_x2x2 config, and if only phy0_mplla_*_clk is used as a source of IN_PCLK, only phy0_mplla_force_en is used, and phy1_mplla_force_en, phy0_mpllb_force_en and phy1_mpllb_force_en are tied to 0b. **Voltage Range:**0-vpdig **Synchronous To:**
phyN_mplla_force_en...(cont.)	I.	Asynchronous **Active State:** High

Port Name	I/O	Description
phyN_mplla_oword_clk	O	Octuple-Rate word clock (mplla_oword_clk). Octuple-symbol rate (x8) clock derived from MPLLA. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_mplla_qword_clk	O	Quadruple-Rate word clock (mplla_qword_clk). Quadruple-symbol rate (x4) clock derived from MPLLA. **Voltage Range:**0-vpdig **Synchronous To:** N/A **Active State:** N/A
phyN_mplla_short_lock_en	I	MPLLA short lock enable. ■ This input indicates that a shared reference clock configuration is being used. ■ When asserted, the MPLL(A,B) locking time will be reduced. **Note:** This signal can be changed only when the txX_mpll_en input for all lanes is de-asserted. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_mplla_word_clk	O	Word clock (mplla_word_clk). Symbol rate clock derived from MPLLA. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_mplla_word_clk_en	I	MPLLA word clock output enable ■ In functional mode and if phyN_mplla_word_clk (for example, phy0) is used by the controller like "PHY owned PLL and IN_PCLK SOURCE MPLL_CLK MODE", this signal must be set to 1 for phyN (for example, only for phy0). ■ In functional mode and if phyN_mplla_word_clk is not used, this input can be set to 0. ■ In scan mode, set this signal to 1 if the PMA clock output is required for at-speed scan. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phyN_mpllb_dword_clk	O	Double-Rate word clock (mpllb_dword_clk). Double-symbol rate (x2) clock derived from MPLLB. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
phyN_mpllb_force_ack	O	MPLLB Force Acknowledge. ■ Acknowledge output for mpll(a,b)_force_en. ■ This output asserts when the MPLL is powered up in response to mpll(a,b)_force_en assertion. ■ This output de-asserts as a response to mpll(a,b)_force_en de-assertion, and at that point the MPLL is powered down if txX_mpll_en of other lanes is de-asserted. **Note:** A four-way level handshake must be followed between mpll(a,b)_force_en and mpll(a,b)_force_ack. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
phyN_mpllb_force_en	I	MPLLB force enable. When asserted, the corresponding MPLL is forced to be powered up, irrespective of the txX_mpll_en input. (DWC_DEEP_PM_SUPis defined and DWC_PIPE_VER_5 is defined and PHY owned PLL and IN_PCLK SOURCE MPLL_CLK MODE and phyN_mpllb_*_clk is used as a source of IN_PCLK): ■ Default value of this signal must be 0b. ■ Initial sequence (contact Synopsys for details): This signal can be asserted when phy_reset is de-asserted and pipe_laneX_reset_n is asserted, with writing to PHY register. Once this signal is asserted, pipe_laneX_reset_n can be de-asserted only after phyN_mpllb_force_ack is asserted. ■ The signal must be kept 1b when any links which use the MPLL clock as Input PCLK is in PCLK-ON powerdown states. ■ This signal must be updated from 1b to 0b only after all links which use the MPLL clock as source of Input PCLK complete PCLK-OFF powerdown state transition(PCIe: P1_CPM, P2, P2_CPM, P2_NOBEACON/ SATA: Slumber, DevSleep, DevSleep.PG). Phystatus are 1-shot pulse with synchronous to PCLK, and PCLK can be disable only after that. ■ This signal must be updated from 0b to 1b before any links which use the MPLL clock as Input PCLK exit from PCLK-OFF powerdown states (PCIe: P1_CPM, P2, P2_CPM, P2_NOBEACON/SATA: Slumber, DevSleep, DevSleep.PG). PowerDown is synchronous to PCLK, and PCLK must be active before changing the signal. ■ A four-way request-ack handshake must be followed with phyN_mpllb_force_ack signal. This signal can be changed from 0b to 1b only when phyN_mpllb_force_ack==0b. This signal can be changed from 1b to 0b only when phyN_mpllb_force_ack==1b. (DWC_DEEP_PM_SUPis not defined) || (DWC_PIPE_VER_5 is not defined) || (DWC_PIPE_VER_5 is defined and PHY owned PLL and and IN_PCLK SOURCE MAX_PCLK MODE) || (DWC_PIPE_VER_5 is defined and External PLL and PHY PLL) || (DWC_DEEP_PM_SUPis defined and DWC_PIPE_VER_5 is defined and PHY owned PLL and IN_PCLK SOURCE MPLL_CLK MODE and phyN_mplla_*_clk is NOT used as a source of IN_PCLK): ■ This signal must be tied to 0b. Example: For x4_x2x2 config, and if only phy0_mplla_*_clk is used as a source of IN_PCLK, only phy0_mplla_force_en is used, and phy1_mplla_force_en, phy0_mpllb_force_en and phy1_mpllb_force_en are tied to 0b. **Voltage Range:**0-vpdig **Synchronous To:**
phyN_mpllb_force_en...(cont.)	I.	Asynchronous **Active State:** High

Port Name	I/O	Description
phyN_mpllb_oword_clk	O	Octuple-Rate word clock (mpllb_oword_clk). Octuple-symbol rate (x8) clock derived from MPLLBs. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_mpllb_qword_clk	O	Quadruple-Rate word clock (mpllb_qword_clk). Quadruple-symbol rate (x4) clock derived from MPLLB. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_mpllb_short_lock_en	I	MPLLB short lock enable. ■ This input indicates that a shared reference clock configuration is being used. ■ When asserted, the MPLL(A,B) locking time will be reduced. **Note:** This signal can be changed only when the txX_mpll_en input for all lanes is de-asserted. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_mpllb_word_clk	O	Word clock (mpllb_word_clk). Symbol rate clock derived from MPLLB. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_mpllb_word_clk_en	I	MPLLB word clock output enable. ■ This input can be set to 0 in functional mode. ■ In scan mode, set it to 1, if PMA clock output is required for at-speed scan. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phyN_sup_misc[7:0]	I	Support miscellaneous controls. This input must be set to the values provided in the "PHY Usage and Configuration" chapter of the DesignWare Cores PHY Databook. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High

4.6 Reference Clock Control Signals (top to/from phy)

phyN_ref_repeat_clk_en (for N = 0; N <= Nphy-1) -
phyN_ref_repeat_clk_sel (for N = 0; N <= Nphy-1) -
phyN_refa_clk_div2_en (for N = 0; N <= Nphy-1) -
phyN_refa_clk_sel (for N = 0; N <= Nphy-1) -
phyN_refa_clkdet_en (for N = 0; N <= Nphy-1) -
phyN_refa_lane_clk_en (for N = 0; N <= Nphy-1) -
phyN_refa_range (for N = 0; N <= Nphy-1) -
phyN_refa_raw_clk_div2_en (for N = 0; N <= Nphy-1) -
phyN_refb_clk_div2_en (for N = 0; N <= Nphy-1) -
phyN_refb_clk_sel (for N = 0; N <= Nphy-1) -
phyN_refb_clkdet_en (for N = 0; N <= Nphy-1) -
phyN_refb_lane_clk_en (for N = 0; N <= Nphy-1) -
phyN_refb_range (for N = 0; N <= Nphy-1) -
phyN_refb_raw_clk_div2_en (for N = 0; N <= Nphy-1) -
phy_laneX_ref_sel (for X = 0; X <= Nlanes-1) -
phy_laneX_refa_clk_en (for X = 0; X <= Nlanes-1) -
phy_laneX_refb_clk_en (for X = 0; X <= Nlanes-1) -

- phyN_mplla_word_fr_clk (for N = 0; N <= Nphy-1)
- phyN_mpllb_word_fr_clk (for N = 0; N <= Nphy-1)
- phyN_ref_dig_clk (for N = 0; N <= Nphy-1)
- phyN_ref_repeat_clk_m (for N = 0; N <= Nphy-1)
- phyN_ref_repeat_clk_p (for N = 0; N <= Nphy-1)
- phyN_refa_clkdet_result (for N = 0; N <= Nphy-1)
- phyN_refa_dig_fr_clk (for N = 0; N <= Nphy-1)
- phyN_refb_clkdet_result (for N = 0; N <= Nphy-1)
- phyN_refb_dig_fr_clk (for N = 0; N <= Nphy-1)

Table 4-6 Reference Clock Control Signals (top to/from phy)

Port Name	I/O	Description
phyN_mplla_word_fr_clk (for N = 0; N <= Nphy-1)	O	Free running mplla_word_clk. This clock is a free-running version of the mplla_word_clk and is intended for use during at-speed scan capture when operating in on-chip clocking (OCC) mode. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_mpllb_word_fr_clk (for N = 0; N <= Nphy-1)	O	Free running mpllb_word_clk. This clock is a free-running version of the mpllb_word_clk and is intended for use during at-speed scan capture when operating in on-chip clocking (OCC) mode. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_ref_dig_clk (for N = 0; N <= Nphy-1)	O	phyN_ref_dig_clk is a signal that connects to output of the PHY. For signal description, refer to the DesignWare Cores PHY Databook. **Note:** ■ Signal descriptions in the PHY databook do not have the prefix phyX_ . ■ For example, phyN_mplla_word_clk is called mplla_word_clk in the PHY databook. **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
phyN_ref_repeat_clk_en (for N = 0; N <= Nphy-1)	I	Repeat reference clock enable. ■ Enables the CML output clocks ref_repeat_clk_(p,m). ■ This pair of clocks can be used as reference clocks for other on-chip PHYs. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_ref_repeat_clk_m (for N = 0; N <= Nphy-1)	O	Current-mode logic (CML) buffered version of input reference clock (ref_repeat_clk_m). ■ This is a CML-buffered version of either ref_pad_clk_p/ref_pad_-clk_m or ref_alt_clk_p/ref_alt_clk_m, depending on the settings of refa/b_clk_sel. ■ This pair of clocks can be used as reference clocks for other PHYs on chip. **Note:** This signal may be undriven when power gating is enabled. For more information on power gating, see the "PHY Usage and Configuration" chapter. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_ref_repeat_clk_p (for N = 0; N <= Nphy-1)	O	Current-mode logic (CML) buffered version of input reference clock (ref_repeat_clk_p). ■ This is a CML-buffered version of either ref_pad_clk_p/ref_pad_-clk_m or ref_alt_clk_p/ref_alt_clk_m, depending on the settings of refa/b_clk_sel. ■ This pair of clocks can be used as reference clocks for other PHYs on chip. **Note:** This signal may be undriven when power gating is enabled. For more information on power gating, see the "PHY Usage and Configuration" chapter. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_ref_repeat_clk_sel[1:0] (for N = 0; N <= Nphy-1)	I	Repeat reference clock select. Function: Selects the input reference clock for the CML output clocks ref_repeat_clk_(p,m). ■ 2'b00: ref_alt0_clk_(p/m) ■ 2'b01: ref_pad_clk_(p/m) ■ 2'b10: ref_alt1_clk_(p/m) ■ 2'b11: Reserved **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
phyN_refa_clk_div2_en (for N = 0; N <= Nphy-1)	I	Input reference clock divider control for refa. This input must be set to the values provided in the "PHY Usage and Configuration" chapter of the DesignWare Cores PHY Databook. See ref_clk_div2_en signal. **Note:** Any change to this input must be followed by phy_reset assertion. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phyN_refa_clk_sel[1:0] (for N = 0; N <= Nphy-1)	I	Select reference clock source from ref_pad_clk_p/m or ref_alt0/1_clk_p/m. **Note:** A transition on this input must be followed by the assertion of phy_reset. ■ 2'b00: ref_alt0_clk_(p/m) ■ 2'b01: ref_pad_clk_(p/m) ■ 2'b10: ref_alt1_clk_(p/m) ■ 2'b11: Reserved **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phyN_refa_clkdet_en (for N = 0; N <= Nphy-1)	I	Enable Reference Clock Detection for refa. Enables detection of the reference clock on either the pads input or the alt0/1 input (depending on refa_clk_sel). **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_refa_clkdet_result (for N = 0; N <= Nphy-1)	O	Reference Clock Detection result for refa. Indicates the presence of the reference clock on either the pads input or the alt0/1 input (depending on refa_clk_sel). ■ The result is only valid when ref_clkdet_en is asserted. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
phyN_refa_dig_fr_clk (for N = 0; N <= Nphy-1)	O	Free-running buffered version of input reference clock refa. ■ This is a buffered, single-ended version of either ref_pad_-clk_p/ref_pad_clk_m or ref_alt0/1_clk_p/ref_alt0/1_clk_m, depending on the settings of refa_clk_sel. ■ The frequency of this clock matches that of the input reference clock and is not affected by the setting of ref_clk_div2_en. **Note:** ■ Do not use this signal as a precision clock source for other PHY instantiations. It is not appropriate for such use. ■ This signal may be undriven when power gating is enabled. For more information on power gating, see the "PHY Usage and Configura-tion" chapter in the PHY databook. ■ This signal toggles only when ref_clk_en is asserted. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_refa_lane_clk_en (for N = 0; N <= Nphy-1)	I	Ungates refa_clk from prescaler to lane blocks in analog. ■ Must be set to 1 when txX_ropll_refsel=2'b10. ■ Any change in this input must be followed by phy_reset assertion. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_refa_range[3:0] (for N = 0; N <= Nphy-1)	I	Input reference clock frequency range for refa. This input must be set to the values provided in the "PHY Usage and Configuration" chapter of the DesignWare Cores PHY Databook. See ref_range(3:0) signal. **Note:** Any change to this input must be followed by phy_reset assertion. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phyN_refa_raw_clk_div2_en (for N = 0; N <= Nphy-1)	I	Input reference clock divider control for refa clock in RAW PCS. This input must be set to the values provided in the "PHY Usage and Configuration" chapter of the DesignWare Cores PHY Databook. See ref_raw_clk_div2_en signal. **Note:** Any change to this input must be followed by phy_reset assertion. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High

Port Name	I/O	Description
phyN_refb_clk_div2_en (for N = 0; N <= Nphy-1)	I	Input reference clock divider control for refb. This input must be set to the values provided in the "PHY Usage and Configuration" chapter of the DesignWare Cores PHY Databook. See ref_clk_div2_en signal. **Note:** Any change to this input must be followed by phy_reset assertion. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phyN_refb_clk_sel[1:0] (for N = 0; N <= Nphy-1)	I	Select reference clock source from ref_pad_clk_p/m, ref_alt0/1_clk_p/m. **Note:** Assert phy_reset after a transition on this input. ■ 2'b00: ref_alt0_clk_(p/m) ■ 2'b01: ref_pad_clk_(p/m) ■ 2'b10: ref_alt1_clk_(p/m) ■ 2'b11: Reserved **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phyN_refb_clkdet_en (for N = 0; N <= Nphy-1)	I	Enable Reference Clock Detection for refb. Enables detection of the reference clock on either the pads input or the alt0/1 input (depending on refb_clk_sel). **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_refb_clkdet_result (for N = 0; N <= Nphy-1)	O	Reference Clock Detection result for refb. Indicates the presence of the reference clock on either the pads input or the alt0/1 input (depending on refb_clk_sel). The result is only valid when ref_clkdet_en is asserted. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
phyN_refb_dig_fr_clk (for N = 0; N <= Nphy-1)	O	Free-running buffered version of input reference clock refb. ■ This is a buffered, single-ended version of either ref_pad_-clk_p/ref_pad_clk_m or ref_alt0/1_clk_p/ref_alt_clk_m, depending on the settings of refb_clk_sel. ■ The frequency of this clock matches that of the input reference clock and is not affected by the setting of ref_clk_div2_en. **Note:** ■ Do not use this signal as a precision clock source for other PHY instantiations. It is not appropriate for such use. ■ This signal may be undriven when power gating is enabled. For more information on power gating, see the "PHY Usage and Configura-tion" chapter in the PHY databook. ■ This signal toggles only when ref_clk_en is asserted. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_refb_lane_clk_en (for N = 0; N <= Nphy-1)	I	Ungates refb_clk from prescaler to lane blocks in analog. ■ Must be set to 1 when txX_ropll_refsel=2'b11. ■ Any change in this input must be followed by phy_reset assertion. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_refb_range[3:0] (for N = 0; N <= Nphy-1)	I	Input reference clock frequency range for refb. This input must be set to the values provided in the "PHY Usage and Configuration" chapter of the DesignWare Cores PHY Databook. See ref_range(3:0) signal. **Note:** Any change to this input must be followed by phy_reset assertion. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phyN_refb_raw_clk_div2_en (for N = 0; N <= Nphy-1)	I	Input reference clock divider control for refb clock in RAW PCS. This input must be set to the values provided in the "PHY Usage and Configuration" chapter of the DesignWare Cores PHY Databook. See ref_raw_clk_div2_en signal. **Note:** Any change to this input must be followed by phy_reset assertion. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High

Port Name	I/O	Description
phy_laneX_ref_sel (for X = 0; X <= Nlanes-1)	I	Lane reference clock selection. This input must be set to the values provided in the "PHY Usage and Configuration" chapter of DesignWare Cores PHY Databook. **Note:** Any change to this input must be followed by phy_reset assertion. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phy_laneX_refa_clk_en (for X = 0; X <= Nlanes-1)	I	Reference clock enable for refa. This input must be set to the values provided in the "PHY Usage and Configuration" chapter of the DesignWare Cores PHY Databook. See refa_clk_en signal. **Note:** Any change to this input must be followed by phy_reset assertion. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phy_laneX_refb_clk_en (for X = 0; X <= Nlanes-1)	I	Reference clock enable for refb. This input must be set to the values provided in the "PHY Usage and Configuration" chapter of the DesignWare Cores PHY Databook. See refb_clk_en signal. **Note:** Any change to this input must be followed by phy_reset assertion. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High

4.7 Power Related Signals (top to/from pcs)

Table 4-7 Power Related Signals (top to/from pcs)

Port Name	I/O	Description
phy_laneX_power_present (for X = 0; X <= Nlanes-1)	I	phy_laneX_power_present is a PHY interface, per-lane configuration signal. ■ This signal is only used for USB and is not supported by this design. ■ Tie this signal to 1'b0. **Synchronous To:** None **Active State:** N/A
upcs_pwr_en	O	Power enable for PCS power switch(es). ■ Enable signal for external switch(es) to supply power to the power-gated logic in the PCS. ■ For information about power gating, see "Power-Gating Support" section. **Synchronous To:** Asynchronous **Active State:** High
upcs_pwr_stable	I	Power stable to PCS. ■ Status signal indicating that power for the PCS is stable. ■ For information about power gating, see "Power-Gating Support" section. **Synchronous To:** Asynchronous **Active State:** High

4.8 Power Related Signals (top to/from pcs/phy)

pg_mode_en -

Table 4-8 Power Related Signals (top to/from pcs/phy)

Port Name	I/O	Description
pg_mode_en	I	Power gating support enable. ■ Control input to enable the power gating support. ■ When de-asserted, the control inputs related to power gating are ignored. ■ For power gating sequencing, see the "PHY Usage and Configuration" chapter. ■ This input is meant to be a tie-off and can be transitioned only during the PHY reset sequence. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

4.9 Power Related Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Table 4-9 Power Related Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Port Name	I/O	Description
phyN_nominal_vp_sel[1:0]	I	VP Nominal Selection. All settings of this pin are reserved for future use. For this version, tie the signal low to 2'b00. This signal indicates the supplied VP voltage level: ■ 2'b00: Reserved for future use ■ 2'b01: Reserved for future use ■ 2'b10: Reserved for future use ■ 2'b11: Reserved for future use **Note:** This input can be changed only when phy_reset is asserted. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A
phyN_nominal_vph_sel[1:0]	I	VPH Nominal Selection. All settings of this pin are reserved for future use. For this version, tie the signal low to 2'b00. This signal indicates the supplied VP voltage level: ■ 2'b00: Reserved for future use ■ 2'b01: Reserved for future use ■ 2'b10: Reserved for future use ■ 2'b11: Reserved for future use **Note:** This input can be changed only when phy_reset is asserted. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A
phyN_pcs_pwr_en	O	Power enable for Raw PCS power switches. ■ Enable signal for external switches to supply power to the power gated logic in Raw PCS. ■ For power gating sequencing, see the "PHY Usage and Configuration" chapter. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
phyN_pcs_pwr_stable	I	Power stable for Raw PCS. ■ Status signal indicating that the power for the Raw PCS is stable. ■ The pcs_pwr_stable signal should only be asserted if the supply is 90% of nominal or higher. ■ For power gating sequencing, see the "PHY Usage and Configuration" chapter. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_pma_pwr_en	O	Power enable for PMA power switch. ■ Enable signal for PMA power switch (external) to supply power to the PMA. ■ For power gating sequencing, see the "PHY Usage and Configuration" chapter. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_pma_pwr_stable	I	Power stable for PMA. ■ Status signal indicating that the power for the PMA is stable. ■ The pma_pwr_stable signal should only be asserted if the supply is 90% of nominal or higher. For power gating sequencing, see the "PHY Usage and Configuration" chapter. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

4.10 Analog I/O Signals (top to/from phy)

Table 4-10 Analog I/O Signals (top to/from phy)

Port Name	I/O	Description
phyN_ref_alt0_clk_m (for N = 0; N <= Nphy-1)	I	Low-swing differential input clock pair internal pin. Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, ref_alt0_clk_m description. ■ Either ref_pad_clk_(p,m) or ref_alt0/1_clk_(p,m) is available on a core, but both must not be active at the same time. ■ The selection is based on whether the reference clock is provided to the core via an on-die distribution network or directly to the core from off- die. ■ The pair that is selected depends on the value of the refa/b/repeat_-clk_sel input. **Voltage Range:**0-vp **Synchronous To:** None **Active State:** N/A
phyN_ref_alt0_clk_p (for N = 0; N <= Nphy-1)	I	Low-swing differential input clock pair internal pin. Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, ref_alt0_clk_p description. ■ Either ref_pad_clk_(p,m) or ref_alt0/1_clk_(p,m) is available on a core, but both must not be active at the same time. ■ The selection is based on whether the reference clock is provided to the core via an on-die distribution network or directly to the core from off- die. ■ The pair that is selected depends on the value of the refa/b/repeat_-clk_sel input. **Voltage Range:**0-vp **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
phyN_ref_alt1_clk_m (for N = 0; N <= Nphy-1)	I	Low-Swing differential input clock pair internal pin. Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, ref_alt1_clk_m description. ■ Either ref_pad_clk_(p,m) or ref_alt0/1_clk_(p,m) is available on a core, but both must not be active at the same time. ■ The selection is based on whether the reference clock is provided to the core via an on-die distribution network or directly to the core from off- die. ■ The pair that is selected depends on the value of the refa/b/repeat_-clk_sel input. **Voltage Range:**0-vp **Synchronous To:** None **Active State:** N/A
phyN_ref_alt1_clk_p (for N = 0; N <= Nphy-1)	I	Low-Swing differential input clock pair internal pin. Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, ref_alt1_clk_p description. ■ Either ref_pad_clk_(p,m) or ref_alt0/1_clk_(p,m) is available on a core, but both must not be active at the same time. ■ The selection is based on whether the reference clock is provided to the core via an on-die distribution network or directly to the core from off- die. ■ The pair that is selected depends on the value of the refa/b/repeat_-clk_sel input. **Voltage Range:**0-vp **Synchronous To:** None **Active State:** N/A
phyN_ref_pad_clk_m (for N = 0; N <= Nphy-1)	IO	Low-Swing differential input clock pair with pad. Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, ref_pad_clk_m description. ■ Either ref_pad_clk_(p,m) or ref_alt_clk_(p,m) is available on a core, but both must not be active at the same time. ■ The selection is based on whether the reference clock is provided to the core through an on-die distribution network or directly to the core from off-die. ■ The pair that is selected depends on the value of the refa/b_clk_sel inputs. **Voltage Range:**0-vp **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
phyN_ref_pad_clk_p (for N = 0; N <= Nphy-1)	IO	Low-Swing differential input clock pair with pad Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, ref_pad_clk_p description. ■ Either ref_pad_clk_(p,m) or ref_alt_clk_(p,m) is available on a core, but both must not be active at the same time. ■ The selection is based on whether the reference clock is provided to the core through an on-die distribution network or directly to the core from off-die. ■ The pair that is selected depends on the value of the refa/b_clk_sel inputs. **Voltage Range:**0-vp **Synchronous To:** None **Active State:** N/A
phyN_resref (for N = 0; N <= Nphy-1)	IO	If IPNAME_SEP_RESREF macro is not defined, only one resref pin, phy_resref, is provided. For the signal description, refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, resref description. **Synchronous To:** None **Active State:** N/A
phy_rxX_m (for X = 0; X <= Nlanes-1)	IO	High-Speed differential receive pair Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_m description. Receive differential pair for lane X **Voltage Range:**0-vp **Synchronous To:** None **Active State:** N/A
phy_rxX_p (for X = 0; X <= Nlanes-1)	IO	High-Speed differential receive pair. Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_p description. Receive differential pair for lane X. **Voltage Range:**0-vp **Synchronous To:** None **Active State:** N/A
phy_txX_m (for X = 0; X <= Nlanes-1)	IO	High-Speed differential transmit pair. Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_m description. Transmit differential pair for lane X . **Voltage Range:**0-vptxX **Synchronous To:** None **Active State:** N/A
phy_txX_p (for X = 0; X <= Nlanes-1)	IO	High-Speed differential transmit pair. Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_p description. Transmit differential pair for lane X . **Voltage Range:**0-vptxX **Synchronous To:** None **Active State:** FALSE

4.11 Receiver Signals (top to/from phy) (for X = 0; X <= Nlanes-1)

Table 4-11 Receiver Signals (top to/from phy) (for X = 0; X <= Nlanes-1)

Port Name	I/O	Description
phy_rxX_flyover_data_m	O	Receive Flyover Data Outputs (rxX_flyover_data_m). Outputs are directly connected to RX bump pads when test_flyover_en is enabled. **Voltage Range:**0-vp **Synchronous To:** Asynchronous **Active State:** High
phy_rxX_flyover_data_p	O	Receive Flyover Data Outputs (rxX_flyover_data_p). Outputs are directly connected to RX bump pads when test_flyover_en is enabled. **Voltage Range:**0-vp **Synchronous To:** Asynchronous **Active State:** High
phy_rxX_hs_clk_m	O	For the signal description, refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_hs_clk_m description. **Synchronous To:** None **Active State:** N/A
phy_rxX_hs_clk_p	O	For the signal description, refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_hs_clk_p description. **Synchronous To:** None **Active State:** N/A
phy_rxX_ppm_drift[5:0]	O	RX CDR PPM Drift. ■ This value represents the amount of ppm on the rxX_clk with respect to the ideal desired frequency. ■ For more information, see the "PHY Usage and Configuration" chapter. ■ This output is valid when rxX_ppm_drift_vld is asserted. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
phy_rxX_ppm_drift_vld	O	RX CDR PPM Drift Valid. ■ Indicates when the rxX_ppm_drift(*) output is valid and can be sampled. ■ For more information, see the "PHY Usage and Configuration" chapter. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phy_rxX_slat_clk	O	For the signal description, refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_slat_clk description. **Synchronous To:** None **Active State:** N/A
phy_rxX_slat_data[19:0]	O	For the signal description, refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_slat_data description. **Synchronous To:** None **Active State:** N/A
phy_rxX_term_acdc	I	Receiver termination control. Controls the terminations of the RX. Selector for floating versus grounded RX termination: ■ 0: Floating RX termination (DC coupled PHY) ■ 1: Grounded RX termination (AC coupled PHY) **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

4.12 Transmitter Signals (top to/from phy) (for X = 0; X <= Nlanes-1)

Table 4-12 Transmitter Signals (top to/from phy) (for X = 0; X <= Nlanes-1)

Port Name	I/O	Description
phy_txX_flyover_data_m	I	Transmit Flyover Data inputs (txX_flyover_data_m). Inputs are directly connected to TX bump pads when test_flyover_en is enabled. **Voltage Range:**0-vp **Synchronous To:** Asynchronous **Active State:** High
phy_txX_flyover_data_p	I	Transmit Flyover Data inputs (txX_flyover_data_p). Inputs are directly connected to TX bump pads when test_flyover_en is enabled. **Voltage Range:**0-vp **Synchronous To:** Asynchronous **Active State:** High
phy_txX_ropll_div_clk	O	TX ROPLL division clock output. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phy_txX_ropll_dword_clk	O	TX ROPLL dword clock output. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phy_txX_ropll_oword_clk	O	TX ROPLL oword clock output. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phy_txX_ropll_qword_clk	O	TX ROPLL qword clock output. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
phy_txX_ropll_word_clk	O	TX ROPLL word clock output. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phy_txX_slat_clk	O	For the signal description, refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_slat_clk description. **Synchronous To:** None **Active State:** N/A
phy_txX_slat_data[19:0]	I	For the signal description, refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_slat_data description. **Synchronous To:** None **Active State:** N/A
phy_txX_slat_data_en	I	For the signal description, refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_slat_data_en description. **Synchronous To:** None **Active State:** N/A
phy_txX_slat_xfer_sel	I	For the signal description, refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_slat_xfer_sel description. **Synchronous To:** None **Active State:** N/A

4.13 PCS Scan Interface Signals (top to/from pcs)

Table 4-13 PCS Scan Interface Signals (top to/from pcs)

Port Name	I/O	Description
pcs_scan_clk	I	PCS scan clock. Scan clock for scan shift and Stuck-At. **Synchronous To:** None **Active State:** N/A
pcs_scan_clk_sel	I	PCS scan clock. ■ pcs_scan_clk selector. ■ If this signal is set to 1b, pcs_scan_clk is used as a scan clock for PCS functional clocks. ■ If this signal is set to 0b, pcs_scan_pcs_clk is used as a scan clock for PCS functional clock. ■ If this signal is set to 0b and SerDes Mode is Not defined, pcs_s-can_pclk is also used. ■ If this signal is set to 0b and SerDes Mode is defined, pcs_scan_-max_pclk is also used. **Synchronous To:** None **Active State:** N/A
pcs_scan_mode	I	PCS scan mode enable. Scan mode enable input for the PCS. **Synchronous To:** None **Active State:** High
pcs_scan_pclk	I	PCS ASST scan clock for PCLK. ■ Scan clock input for laneX_pclk functional clock domain. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is not defined. **Synchronous To:** Asynchronous **Active State:** N/A
pcs_scan_ropll_clk	I	PCS ASST scan clock for ROPLLCLK. ■ ASST scan clock input for laneX_ropll_clk internal, functional clock domain. **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
pcs_scan_rst	I	PCS scan reset. Scan reset input for the PCS; must be de-asserted when pipe_scan_mode is de-asserted. **Synchronous To:** None **Active State:** High
pcs_scan_shift	I	Scan enable port for PCS. Dedicated scan enable for PCS. **Synchronous To:** None **Active State:** High
pcs_scan_shift_cg	I	Scan enable port for PCS clock gates. Dedicated scan enable for clock gating logic in PCS. **Synchronous To:** None **Active State:** High

4.14 PHY Scan Interface Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Table 4-14 PHY Scan Interface Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Port Name	I/O	Description
phyN_scan_clk	I	Stuck-at scan clock. This input is selected when scan_clk_sel input is 1. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_scan_clk_sel	I	This input selects the scan clock mode of the PHY. ■ 0: At speed scan clock ■ 1: Stuck-at scan clock (scan_clk input) **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_scan_cr_clk	I	Scan clock per clock domain (scan_cr_clk). ■ Test Clock muxed into the clock pin of registers during Scan operation. ■ All scan clock inputs are assumed to be asynchronous to each other. ■ All scan clock inputs can be active at the same time during scan shift operation; however, only one scan clock can be active at any time during the scan capture cycle. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
phyN_scan_cr_in[518:0]	I	Scan chain inputs per clock domain (scan_cr_in). **Note:** scan_txX_in(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_cr_out[518:0]	O	Scan chain outputs per clock domain (scan_cr_out). **Note:** scan_txX_out(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_jtag_in[2:0]	I	Scan chain inputs per clock domain (scan_jtag_in) **Note:** scan_txX_in(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_jtag_out[2:0]	O	Scan chain outputs per clock domain (scan_jtag_out). **Note:** scan_txX_out(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_mode	I	Scan mode enable. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_scan_mpll_dword_clk	I	Scan clock per clock domain (scan_mpll_dword_clk). ■ Test Clock muxed into the clock pin of registers during Scan operation. ■ All scan clock inputs are assumed to be asynchronous to each other. ■ All scan clock inputs can be active at the same time during scan shift operation; however, only one scan clock can be active at any time during the scan capture cycle. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_scan_mpll_dword_in[167:0]	I	Scan chain inputs per clock domain (scan_mpll_dword_in). **Note:** scan_txX_in(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High

Port Name	I/O	Description
phyN_scan_mpll_dword_out[167:0]	O	Scan chain outputs per clock domain (scan_mpll_dword_out). **Note:** scan_txX_in(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_mpll_word_clk	I	Scan clock per clock domain (scan_mpll_word_clk). ■ Test Clock muxed into the clock pin of registers during Scan operation. ■ All scan clock inputs are assumed to be asynchronous to each other. ■ All scan clock inputs can be active at the same time during scan shift operation; however, only one scan clock can be active at any time during the scan capture cycle. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_scan_mpll_word_in[12:0]	I	Scan chain inputs per clock domain (scan_mpll_word_in). **Note:** scan_txX_in(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_mpll_word_out[12:0]	O	Scan chain outputs per clock domain (scan_mpll_word_out). **Note:** scan_txX_in(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_occ_clk	I	Scan clock per clock domain (scan_occ_clk). ■ Test clock muxed into the clock pin of registers during scan operation. ■ All scan clock inputs are assumed to be asynchronous to each other. ■ All scan clock inputs can be active at the same time during scan shift operation; however, only one scan clock can be active at any time during the scan capture cycle. **Voltage Range:**0-vpdig **Synchronous To:** N/A **Active State:** N/A
phyN_scan_occ_in[2:0]	I	Scan chain inputs per clock domain (scan_occ_in). **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_occ_out[2:0]	O	Scan chain outputs per clock domain (scan_occ_out). **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High

Port Name	I/O	Description
phyN_scan_pma_occ_en	I	Scan OCC enable. This signal enables the on-chip clocking (OCC) latch and hold circuit to latch the control signals going to the prescaler and MPLL's. **Note:** Assert this signal only if scan_mode is 1'b0 and the prescaler and MPLLs are powered up. For more information about OCC, see the OCC section. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_ref_clk	I	Scan clock per clock domain (scan_ref_clk). ■ Test Clock muxed into the clock pin of registers during Scan operation. ■ All scan clock inputs are assumed to be asynchronous to each other. ■ All scan clock inputs can be active at the same time during scan shift operation; however, only one scan clock can be active at any time during the scan capture cycle. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_scan_ref_in[92:0]	I	Scan chain inputs per clock domain (scan_ref_in). **Note:** scan_txX_in(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_ref_out[92:0]	O	Scan chain outputs per clock domain (scan_ref_out). **Note:** scan_txX_out(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_set_rst	I	Scan set reset. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_shift	I	Scan shift. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_shift_cg	I	Scan shift for clock gates. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High

Port Name	I/O	Description
phyN_scan_tx_in[27:0]	I	Scan chain inputs per clock domain (scan_tx_in). **Note:** scan_txX_in(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High
phyN_scan_tx_out[27:0]	O	Scan chain outputs per clock domain (scan_tx_out). **Note:** scan_txX_in(*) is synchronous to txX_clk. **Voltage Range:**0-vpdig **Synchronous To:** scan_CLKDOMAIN_clk **Active State:** High

4.15 JTAG Port Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Table 4-15 JTAG Port Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Port Name	I/O	Description
phyN_jtag_tck	I	JTAG clock. JTAG state machine clock signal. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_jtag_tdi	I	JTAG input port. JTAG test data input signal. **Voltage Range:**0-vpdig **Synchronous To:** phyN_jtag_tck **Active State:** N/A
phyN_jtag_tdo	O	JTAG output port. JTAG test data output signal. **Voltage Range:**0-vpdig **Synchronous To:** phyN_jtag_tck **Active State:** N/A
phyN_jtag_tdo_en	O	JTAG output enable. JTAG test data output enable signal. **Voltage Range:**0-vpdig **Synchronous To:** phyN_jtag_tck **Active State:** High
phyN_jtag_tms	I	JTAG state machine control. JTAG test mode select signal. **Voltage Range:**0-vpdig **Synchronous To:** phyN_jtag_tck **Active State:** High
phyN_jtag_trst_n	I	JTAG reset. JTAG state machine reset signal. While this signal is held low, the state machine is held in test logic reset state. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** Low

4.16 Boundary Scan Port Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Table 4-16 Boundary Scan Port Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Port Name	I/O	Description
phyN_bs_acmode	I	Boundary Scan AC Mode (Figure 51, IEEE 1149.6) (bs_acmode). Enables Boundary Scan AC mode. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_bs_actest	I	Boundary Scan AC Test Signal (Figure 51, IEEE 1149.6) (bs_actest). The AC Test Signal is generated by the user logic when AC EXTEST instruction is decoded. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_bs_cdr	I	Boundary Scan Clock-DR (IEEE 1149.1) (bs_cdr). Clocks the Boundary Scan capture registers, in either Scan or Capture modes. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_bs_ce	I	Boundary scan compliance enable signal (bs_ce). ■ Enables the Boundary Scan Transmitter and Receiver register cells on all lanes. ■ This input must be driven from a sequential element and not from any combinatorial logic. ■ Any glitch on this input causes the PHY to go into reset state. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
phyN_bs_rx_bigswing	I	RX Boundary Scan big swing (bs_tx_bigswing). Enables RX ACJTAG big-swing mode. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_bs_rx_init	I	RX Boundary Scan Initialize Signal (bs_rx_init). ■ This signal is similar to the 'Init Clk' referenced in test receiver diagrams in the ACJTAG specification (Figure 48, IEEE 1149.6). ■ While this signal is high, the hysteresis in the receiver comparator is reset. ■ The falling edge of this signal must occur at appropriate times, as specified in the ACJTAG specification. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_bs_rx_level[4:0]	I	ACJTAG Receiver Sensitivity Level Control (bs_rx_level). ■ Sets the internal thresholds for the ACJTAG receiver. ■ The ACJTAG threshold level must be set to trigger for the expected amplitude for the given protocol. ■ For threshold levels, see the "PHY Usage and Configuration" chapter. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A
phyN_bs_sdr	I	Boundary Scan Shift-DR (IEEE 1149.1) (bs_sdr). Enables shifting of the Boundary Scan capture registers. **Voltage Range:**0-vpdig **Synchronous To:** phyN_bs_cdr **Active State:** High
phyN_bs_tdi	I	Boundary Scan Test Data In (bs_tdi). Boundary scan shift register input signal. **Voltage Range:**0-vpdig **Synchronous To:** phyN_bs_cdr **Active State:** High
phyN_bs_tdo	O	Boundary Scan Test Data Out (bs_tdo). Boundary scan shift register output signal. **Voltage Range:**0-vpdig **Synchronous To:** phyN_bs_cdr **Active State:** High

Port Name	I/O	Description
phyN_bs_tx_lowswing	I	TX Boundary Scan low swing (bs_tx_lowswing). Enables TX JTAG low-swing mode (otherwise, it is in full- swing mode). **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_bs_udr	I	Boundary Scan Update-DR (IEEE 1149.1) (bs_udr). Clocks the Boundary Scan update registers. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A

4.17 External Protocol Override Signals (top to/from pcs)

phy_ext_ctrl_sel -
protocol0_esm_ext_rx_adapt_mode_r0 -
protocol0_esm_ext_rx_adapt_mode_r1 -
protocol0_esm_ext_rx_cdr_ppm_max_r0 -
protocol0_esm_ext_rx_cdr_ppm_max_r1 -
protocol0_esm_ext_rx_cdr_vco_config_r0 -
protocol0_esm_ext_rx_cdr_vco_config_r1 -
protocol0_esm_ext_rx_dcc_ctrl_cm_range_r0 -
protocol0_esm_ext_rx_dcc_ctrl_cm_range_r1 -
protocol0_esm_ext_rx_dcc_ctrl_diff_range_r0 -
protocol0_esm_ext_rx_dcc_ctrl_diff_range_r1 -
protocol0_esm_ext_rx_delta_iq_r0 -
protocol0_esm_ext_rx_delta_iq_r1 -
protocol0_esm_ext_rx_dfe_bypass_r0 -
protocol0_esm_ext_rx_dfe_bypass_r1 -
protocol0_esm_ext_rx_eq_afe_config_r0 -
protocol0_esm_ext_rx_eq_afe_config_r1 -
protocol0_esm_ext_rx_eq_afe_rate_r0 -
protocol0_esm_ext_rx_eq_afe_rate_r1 -
protocol0_esm_ext_rx_eq_att_lvl_r0 -
protocol0_esm_ext_rx_eq_att_lvl_r1 -
protocol0_esm_ext_rx_eq_ctle_boost_r0 -
protocol0_esm_ext_rx_eq_ctle_boost_r1 -
protocol0_esm_ext_rx_eq_ctle_pole_r0 -
protocol0_esm_ext_rx_eq_ctle_pole_r1 -
protocol0_esm_ext_rx_eq_dfe_float_en_r0 -
protocol0_esm_ext_rx_eq_dfe_float_en_r1 -
protocol0_esm_ext_rx_eq_dfe_tap1_r0 -
protocol0_esm_ext_rx_eq_dfe_tap1_r1 -
protocol0_esm_ext_rx_eq_dfe_tap2_r0 -
protocol0_esm_ext_rx_eq_dfe_tap2_r1 -
protocol0_esm_ext_rx_eq_vga_gain_r0 -
protocol0_esm_ext_rx_eq_vga_gain_r1 -
protocol0_esm_ext_rx_misc_r0 -
protocol0_esm_ext_rx_misc_r1 -
protocol0_esm_ext_rx_ref_ld_val_r0 -
protocol0_esm_ext_rx_ref_ld_val_r1 -
protocol0_esm_ext_rx_sigdet_lf_threshold_r0 -
protocol0_esm_ext_rx_sigdet_lf_threshold_r1 -
protocol0_esm_ext_rx_sigdet_lfps_filter_en_r0 -
protocol0_esm_ext_rx_sigdet_lfps_filter_en_r1 -
protocol0_esm_ext_rx_term_ctrl_r0 -
protocol0_esm_ext_rx_term_ctrl_r1 -
protocol0_esm_ext_rx_vco_ld_val_r0 -
protocol0_esm_ext_rx_vco_ld_val_r1 -
protocol0_esm_ext_tx_dcc_ctrl_cm_range_r0 -
protocol0_esm_ext_tx_dcc_ctrl_cm_range_r1 -
protocol0_esm_ext_tx_dcc_ctrl_diff_range_r0 -
protocol0_esm_ext_tx_dcc_ctrl_diff_range_r1 -
protocol0_esm_ext_tx_eq_main_r0 -
protocol0_esm_ext_tx_eq_main_r1 -
protocol0_esm_ext_tx_eq_ovrd_r0 -
protocol0_esm_ext_tx_eq_ovrd_r1 -
protocol0_esm_ext_tx_eq_post_r0 -
protocol0_esm_ext_tx_eq_post_r1 -
protocol0_esm_ext_tx_eq_pre_r0 -
protocol0_esm_ext_tx_eq_pre_r1 -
protocol0_esm_ext_tx_fastedge_en_r0 -
protocol0_esm_ext_tx_fastedge_en_r1 -

protocol0_esm_ext_tx_misc_r0 -
protocol0_esm_ext_tx_misc_r1 -
protocol0_esm_ext_tx_ropll_cp_ctl_intg_r0 -
protocol0_esm_ext_tx_ropll_cp_ctl_intg_r1 -
protocol0_esm_ext_tx_ropll_cp_ctl_prop_r0 -
protocol0_esm_ext_tx_ropll_cp_ctl_prop_r1 -
protocol0_esm_ext_tx_ropll_fbdiv_r0 -
protocol0_esm_ext_tx_ropll_fbdiv_r1 -
protocol0_esm_ext_tx_ropll_postdiv_r0 -
protocol0_esm_ext_tx_ropll_postdiv_r1 -
protocol0_esm_ext_tx_ropll_rc_filter_r0 -
protocol0_esm_ext_tx_ropll_rc_filter_r1 -
protocol0_esm_ext_tx_ropll_refdiv_r0 -
protocol0_esm_ext_tx_ropll_refdiv_r1 -
protocol0_esm_ext_tx_ropll_refsel_r0 -
protocol0_esm_ext_tx_ropll_refsel_r1 -
protocol0_esm_ext_tx_ropll_v2i_mode_r0 -
protocol0_esm_ext_tx_ropll_v2i_mode_r1 -
protocol0_esm_ext_tx_ropll_vco_low_freq_r0 -
protocol0_esm_ext_tx_ropll_vco_low_freq_r1 -
protocol0_esm_ext_tx_ropll_word_clk_div_sel_r0 -
protocol0_esm_ext_tx_ropll_word_clk_div_sel_r1 -
protocol0_esm_ext_tx_term_ctrl_r0 -
protocol0_esm_ext_tx_term_ctrl_r1 -
protocolP_ext_mplla_fb_clk_div4_en (for P = 0; P <= 2) -
protocolP_ext_mplla_frac_den (for P = 0; P <= 2) -
protocolP_ext_mplla_frac_en (for P = 0; P <= 2) -
protocolP_ext_mplla_frac_quot (for P = 0; P <= 2) -
protocolP_ext_mplla_frac_rem (for P = 0; P <= 2) -
protocolP_ext_mplla_multiplier (for P = 0; P <= 2) -
protocolP_ext_mplla_ssc_peak (for P = 0; P <= 2) -
protocolP_ext_mplla_ssc_step_size (for P = 0; P <= 2) -
protocolP_ext_mplla_ssc_up_spread (for P = 0; P <= 2) -
protocolP_ext_mplla_word_clk_div (for P = 0; P <= 2) -
protocolP_ext_mpllb_fb_clk_div4_en (for P = 0; P <= 2) -
protocolP_ext_mpllb_frac_den (for P = 0; P <= 2) -
protocolP_ext_mpllb_frac_en (for P = 0; P <= 2) -
protocolP_ext_mpllb_frac_quot (for P = 0; P <= 2) -
protocolP_ext_mpllb_frac_rem (for P = 0; P <= 2) -
protocolP_ext_mpllb_multiplier (for P = 0; P <= 2) -
protocolP_ext_mpllb_ssc_peak (for P = 0; P <= 2) -
protocolP_ext_mpllb_ssc_step_size (for P = 0; P <= 2) -
protocolP_ext_mpllb_ssc_up_spread (for P = 0; P <= 2) -
protocolP_ext_mpllb_word_clk_div (for P = 0; P <= 2) -
protocolP_ext_ref_clk_mplla_div (for P = 0; P <= 2) -
protocolP_ext_ref_clk_mpllb_div (for P = 0; P <= 2) -
protocolP_ext_rx_adapt_mode_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_adapt_mode_g2 (for P = 0; P <= 2) -

protocolP_ext_rx_adapt_mode_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_adapt_mode_g4 -
protocol0_ext_rx_adapt_mode_g5 -
protocolP_ext_rx_adapt_sel_piperate0 (for P = 0; P <= 2) -
protocolP_ext_rx_adapt_sel_piperate1 (for P = 0; P <= 2) -
protocolP_ext_rx_adapt_sel_piperate2 (for P = 0; P <= 2) -
protocolP_ext_rx_adapt_sel_piperate3 (for P = 0; P <= 2) -
protocol0_ext_rx_adapt_sel_piperate4 -
protocolP_ext_rx_cdr_ppm_max_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_cdr_ppm_max_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_cdr_ppm_max_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_cdr_ppm_max_g4 -
protocol0_ext_rx_cdr_ppm_max_g5 -
protocolP_ext_rx_cdr_vco_config_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_cdr_vco_config_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_cdr_vco_config_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_cdr_vco_config_g4 -
protocol0_ext_rx_cdr_vco_config_g5 -
protocolP_ext_rx_dcc_ctrl_cm_range_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_dcc_ctrl_cm_range_g2 (for P = 0; P <= 2) -
protocol0_ext_rx_dcc_ctrl_cm_range_g3 -
protocol0_ext_rx_dcc_ctrl_cm_range_g4 -
protocol0_ext_rx_dcc_ctrl_cm_range_g5 -
protocolP_ext_rx_dcc_ctrl_diff_range_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_dcc_ctrl_diff_range_g2 (for P = 0; P <= 2) -
protocol0_ext_rx_dcc_ctrl_diff_range_g3 -
protocol0_ext_rx_dcc_ctrl_diff_range_g4 -
protocol0_ext_rx_dcc_ctrl_diff_range_g5 -
protocolP_ext_rx_delta_iq_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_delta_iq_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_delta_iq_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_delta_iq_g4 -
protocol0_ext_rx_delta_iq_g5 -
protocolP_ext_rx_dfe_bypass_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_dfe_bypass_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_dfe_bypass_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_dfe_bypass_g4 -
protocol0_ext_rx_dfe_bypass_g5 -
protocolP_ext_rx_eq_afe_config_g1 (for P = 0; P <= 2) -

protocolP_ext_rx_eq_afe_config_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_afe_config_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_eq_afe_config_g4 -
protocol0_ext_rx_eq_afe_config_g5 -
protocolP_ext_rx_eq_afe_rate_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_afe_rate_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_afe_rate_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_eq_afe_rate_g4 -
protocol0_ext_rx_eq_afe_rate_g5 -
protocolP_ext_rx_eq_att_lvl_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_att_lvl_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_att_lvl_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_eq_att_lvl_g4 -
protocol0_ext_rx_eq_att_lvl_g5 -
protocolP_ext_rx_eq_ctle_boost_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_ctle_boost_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_ctle_boost_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_eq_ctle_boost_g4 -
protocol0_ext_rx_eq_ctle_boost_g5 -
protocolP_ext_rx_eq_ctle_pole_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_ctle_pole_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_ctle_pole_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_eq_ctle_pole_g4 -
protocol0_ext_rx_eq_ctle_pole_g5 -
protocolP_ext_rx_eq_dfe_float_en_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_dfe_float_en_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_dfe_float_en_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_eq_dfe_float_en_g4 -
protocol0_ext_rx_eq_dfe_float_en_g5 -
protocolP_ext_rx_eq_dfe_tap1_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_dfe_tap1_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_dfe_tap1_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_eq_dfe_tap1_g4 -
protocol0_ext_rx_eq_dfe_tap1_g5 -
protocolP_ext_rx_eq_dfe_tap2_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_dfe_tap2_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_dfe_tap2_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_eq_dfe_tap2_g4 -
protocol0_ext_rx_eq_dfe_tap2_g5 -

protocolP_ext_rx_eq_vga_gain_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_vga_gain_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_eq_vga_gain_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_eq_vga_gain_g4 -
protocol0_ext_rx_eq_vga_gain_g5 -
protocolP_ext_rx_los_pwr_up_cnt (for P = 0; P <= 2) -
protocolP_ext_rx_misc_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_misc_g2 (for P = 0; P <= 2) -
protocol0_ext_rx_misc_g3 -
protocol0_ext_rx_misc_g4 -
protocol0_ext_rx_misc_g5 -
protocolP_ext_rx_ref_ld_val_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_ref_ld_val_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_ref_ld_val_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_ref_ld_val_g4 -
protocol0_ext_rx_ref_ld_val_g5 -
protocolP_ext_rx_sigdet_lf_threshold_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_sigdet_lf_threshold_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_sigdet_lf_threshold_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_sigdet_lf_threshold_g4 -
protocol0_ext_rx_sigdet_lf_threshold_g5 -
protocolP_ext_rx_sigdet_lfps_filter_en_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_sigdet_lfps_filter_en_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_sigdet_lfps_filter_en_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_sigdet_lfps_filter_en_g4 -
protocol0_ext_rx_sigdet_lfps_filter_en_g5 -
protocolP_ext_rx_term_ctrl_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_term_ctrl_g2 (for P = 0; P <= 2) -
protocol0_ext_rx_term_ctrl_g3 -
protocol0_ext_rx_term_ctrl_g4 -
protocol0_ext_rx_term_ctrl_g5 -
protocolP_ext_rx_vco_ld_val_g1 (for P = 0; P <= 2) -
protocolP_ext_rx_vco_ld_val_g2 (for P = 0; P <= 2) -
protocolP_ext_rx_vco_ld_val_g3 (for P = 0; P <= 2) -
protocol0_ext_rx_vco_ld_val_g4 -
protocol0_ext_rx_vco_ld_val_g5 -
protocolP_ext_rx_vref_ctrl (for P = 0; P <= 2) -
protocolP_ext_tx_dcc_ctrl_cm_range_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_dcc_ctrl_cm_range_g2 (for P = 0; P <= 2) -
protocol0_ext_tx_dcc_ctrl_cm_range_g3 -
protocol0_ext_tx_dcc_ctrl_cm_range_g4 -
protocol0_ext_tx_dcc_ctrl_cm_range_g5 -
protocolP_ext_tx_dcc_ctrl_diff_range_g1 (for P = 0; P <= 2) -

protocolP_ext_tx_dcc_ctrl_diff_range_g2 (for P = 0; P <= 2) -
protocol0_ext_tx_dcc_ctrl_diff_range_g3 -
protocol0_ext_tx_dcc_ctrl_diff_range_g4 -
protocol0_ext_tx_dcc_ctrl_diff_range_g5 -
protocolP_ext_tx_eq_main_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_eq_main_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_eq_main_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_eq_main_g4 -
protocol0_ext_tx_eq_main_g5 -
protocolP_ext_tx_eq_ovrd_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_eq_ovrd_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_eq_ovrd_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_eq_ovrd_g4 -
protocol0_ext_tx_eq_ovrd_g5 -
protocolP_ext_tx_eq_post_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_eq_post_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_eq_post_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_eq_post_g4 -
protocol0_ext_tx_eq_post_g5 -
protocolP_ext_tx_eq_pre_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_eq_pre_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_eq_pre_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_eq_pre_g4 -
protocol0_ext_tx_eq_pre_g5 -
protocolP_ext_tx_fastedge_en_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_fastedge_en_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_fastedge_en_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_fastedge_en_g4 -
protocol0_ext_tx_fastedge_en_g5 -
protocolP_ext_tx_misc_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_misc_g2 (for P = 0; P <= 2) -
protocol0_ext_tx_misc_g3 -
protocol0_ext_tx_misc_g4 -
protocol0_ext_tx_misc_g5 -
protocolP_ext_tx_ropll_cp_ctl_intg_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_cp_ctl_intg_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_cp_ctl_intg_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_ropll_cp_ctl_intg_g4 -
protocol0_ext_tx_ropll_cp_ctl_intg_g5 -
protocolP_ext_tx_ropll_cp_ctl_prop_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_cp_ctl_prop_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_cp_ctl_prop_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_ropll_cp_ctl_prop_g4 -
protocolP_ext_tx_ropll_cp_ctl_prop_g5 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_fbdiv_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_fbdiv_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_fbdiv_g3 (for P = 0; P <= 2) -

protocol0_ext_tx_ropll_fbdiv_g4 -
protocol0_ext_tx_ropll_fbdiv_g5 -
protocolP_ext_tx_ropll_postdiv_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_postdiv_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_postdiv_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_ropll_postdiv_g4 -
protocol0_ext_tx_ropll_postdiv_g5 -
protocolP_ext_tx_ropll_rc_filter_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_rc_filter_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_rc_filter_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_ropll_rc_filter_g4 -
protocol0_ext_tx_ropll_rc_filter_g5 -
protocolP_ext_tx_ropll_refdiv_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_refdiv_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_refdiv_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_ropll_refdiv_g4 -
protocol0_ext_tx_ropll_refdiv_g5 -
protocolP_ext_tx_ropll_refsel_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_refsel_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_refsel_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_ropll_refsel_g4 -
protocol0_ext_tx_ropll_refsel_g5 -
protocolP_ext_tx_ropll_v2i_mode_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_v2i_mode_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_v2i_mode_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_ropll_v2i_mode_g4 -
protocol0_ext_tx_ropll_v2i_mode_g5 -
protocolP_ext_tx_ropll_vco_low_freq_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_vco_low_freq_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_vco_low_freq_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_ropll_vco_low_freq_g4 -
protocol0_ext_tx_ropll_vco_low_freq_g5 -
protocolP_ext_tx_ropll_word_clk_div_sel_g1 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_word_clk_div_sel_g2 (for P = 0; P <= 2) -
protocolP_ext_tx_ropll_word_clk_div_sel_g3 (for P = 0; P <= 2) -
protocol0_ext_tx_ropll_word_clk_div_sel_g4 -
protocol0_ext_tx_ropll_word_clk_div_sel_g5 -
protocolP_ext_tx_term_ctrl_g1 (for P = 0; P <= 2) -

protocolP_ext_tx_term_ctrl_g2 (for P = 0; P <= 2) -
protocol0_ext_tx_term_ctrl_g3 -
protocol0_ext_tx_term_ctrl_g4 -
protocol0_ext_tx_term_ctrl_g5 -
protocol2_ext_rx_dcc_ctrl_cm_range_g3 -
protocol2_ext_rx_dcc_ctrl_diff_range_g3 -
protocol2_ext_rx_term_ctrl_g3 -
protocol2_ext_tx_dcc_ctrl_cm_range_g3 -
protocol2_ext_tx_dcc_ctrl_diff_range_g3 -
protocol2_ext_tx_misc_g2 -
protocol2_ext_tx_misc_g3 -
protocol2_ext_tx_term_ctrl_g3 -
protocol_ext_mplla_recal_bank_sel -
protocol_ext_mplla_recal_bank_sel_ovrd_en -
protocol_ext_mpllb_recal_bank_sel -
protocol_ext_mpllb_recal_bank_sel_ovrd_en -
protocol_ext_phyN_mplla_bw_high (for N = 0; N <= Nphy-1) -
protocol_ext_phyN_mplla_bw_low (for N = 0; N <= Nphy-1) -
protocol_ext_phyN_mplla_bw_threshold (for N = 0; N <= Nphy-1) -
protocol_ext_phyN_mplla_tx_clk_div (for N = 0; N <= Nphy-1) -
protocol_ext_phyN_mplla_tx_clk_div_ovrd_en (for N = 0; N <= Nphy-1) -
protocol_ext_phyN_mpllb_bw_high (for N = 0; N <= Nphy-1) -
protocol_ext_phyN_mpllb_bw_low (for N = 0; N <= Nphy-1) -
protocol_ext_phyN_mpllb_bw_threshold (for N = 0; N <= Nphy-1) -
protocol_ext_phyN_mpllb_tx_clk_div (for N = 0; N <= Nphy-1) -
protocol_ext_phyN_mpllb_tx_clk_div_ovrd_en (for N = 0; N <= Nphy-1) -
protocol_ext_rx_eq_afe_ovrd_en -
protocol_ext_rx_misc_ovrd_en -
protocol_ext_tx_ropll_ovrd_en -
protocol_ext_tx_ropll_postdiv_ovrd_en -
protocol_ext_tx_ropll_refdiv_ovrd_en -
protocol_ext_tx_ropll_word_clk_div_sel_ovrd_en -

Table 4-17 External Protocol Override Signals (top to/from pcs)

Port Name	I/O	Description
phy_ext_ctrl_sel	I	phy_ext_ctrl_sel is a PHY configuration setting per protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_adapt_mode_r0[11:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_adapt_mode_r1[11:0]	I	protocol0_esm_ext_rx_adapt_mode_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_cdr_ppm_max_r0[19:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_cdr_ppm_max_r1[19:0]	I	protocol0_esm_ext_rx_cdr_ppm_max_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_cdr_vco_config_r0[47:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_cdr_vco_config_r1[47:0]	I	protocol0_esm_ext_rx_cdr_vco_config_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_dcc_ctrl_cm_range_r0[3:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_dcc_ctrl_cm_range_r1[3:0]	I	protocol0_esm_ext_rx_dcc_ctrl_cm_range_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_dcc_ctrl_diff_range_r0[3:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_dcc_ctrl_diff_range_r1[3:0]	I	protocol0_esm_ext_rx_dcc_ctrl_diff_range_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_delta_iq_r0[15:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_delta_iq_r1[15:0]	I	protocol0_esm_ext_rx_delta_iq_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_dfe_bypass_r0[3:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_dfe_bypass_r1[3:0]	I	protocol0_esm_ext_rx_dfe_bypass_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_eq_afe_config_r0[47:0]	I	protocol0_esm_*_r0 is unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_eq_afe_config_r1[47:0]	I	protocol0_esm_ext_rx_eq_afe_config_r1 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_eq_afe_rate_r0[11:0]	I	protocol0_esm_*_r0 is unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_eq_afe_rate_r1[11:0]	I	protocol0_esm_ext_rx_eq_afe_rate_r1 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_eq_att_lvl_r0[11:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_eq_att_lvl_r1[11:0]	I	protocol0_esm_ext_rx_eq_att_lvl_r1 is a lane-based PHY configuration setting override per-protocol. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_eq_ctle_boost_r0[19:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_eq_ctle_boost_r1[19:0]	I	protocol0_esm_ext_rx_eq_ctle_boost_r1 is a lane-based PHY configuration setting override per-protocol. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_eq_ctle_pole_r0[7:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_eq_ctle_pole_r1[7:0]	I	protocol0_esm_ext_rx_eq_ctle_pole_r1 is a lane-based PHY configuration setting override per-protocol. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_eq_dfe_float_en_r0[3:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_eq_dfe_float_en_r1[3:0]	I	protocol0_esm_ext_rx_eq_dfe_float_en_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_eq_dfe_tap1_r0[35:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_eq_dfe_tap1_r1[35:0]	I	protocol0_esm_ext_rx_eq_dfe_tap1_r1 is a lane-based receiver equalization setting override. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_eq_dfe_tap2_r0[7:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_eq_dfe_tap2_r1[7:0]	I	protocol0_esm_ext_rx_eq_dfe_tap2_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_eq_vga_gain_r0[4:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_eq_vga_gain_r1[4:0]	I	protocol0_esm_ext_rx_eq_vga_gain_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_misc_r0[31:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_misc_r1[31:0]	I	protocol0_esm_ext_rx_misc_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_misc_ovrd_en, protocol*_*ext_rx_misc_r* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_ref_ld_val_r0[6:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_ref_ld_val_r1[6:0]	I	protocol0_esm_ext_rx_ref_ld_val_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_sigdet_lf_threshold_r0[2:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_sigdet_lf_threshold_r1[2:0]	I	protocol0_esm_ext_rx_sigdet_lf_threshold_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_sigdet_lfps_filter_en_r0	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_sigdet_lfps_filter_en_r1	I	protocol0_esm_ext_rx_sigdet_lfps_filter_en_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_term_ctrl_r0[3:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_term_ctrl_r1[3:0]	I	protocol0_esm_ext_rx_term_ctrl_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_rx_vco_ld_val_r0[12:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_rx_vco_ld_val_r1[12:0]	I	protocol0_esm_ext_rx_vco_ld_val_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_dcc_ctrl_cm_range_r0[3:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_dcc_ctrl_cm_range_r1[3:0]	I	protocol0_esm_ext_tx_dcc_ctrl_cm_range_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_dcc_ctrl_diff_range_r0[3:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_dcc_ctrl_diff_range_r1[3:0]	I	protocol0_esm_ext_tx_dcc_ctrl_diff_range_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_eq_main_r0[23:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_eq_main_r1[23:0]	I	protocol0_esm_ext_tx_eq_main_r1 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol0_esm_ext_tx_eq_main_r1, a per-protocol basis from these top-level pins when the corresponding protocol0_esm_ext_tx_eq_ovrd_r1 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_eq_ovrd_r0[3:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_eq_ovrd_r1[3:0]	I	protocol0_esm_ext_tx_eq_ovrd_r1 is a lane-based transmitter equalization setting override enable ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_(*) input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_eq_post_r0[19:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_eq_post_r1[19:0]	I	protocol0_esm_ext_tx_eq_post_r1 is a lane-based transmitter equalization setting override. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol0_esm_ext_tx_eq_post_r1, a per-protocol basis from these top-level pins when the corresponding protocol0_esm_ext_tx_eq_ovrd_r1 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_eq_pre_r0[19:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_eq_pre_r1[19:0]	I	protocol0_esm_ext_tx_eq_pre_r1 is a lane-based transmitter equalization setting override. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol0_esm_ext_tx_eq_pre_r1, a per-protocol basis from these top-level pins when the corresponding protocol0_esm_ext_tx_eq_ovrd_r1 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_fastedge_en_r0	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_fastedge_en_r1	I	protocol0_esm_ext_tx_fastedge_en_r1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_misc_r0[31:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_misc_r1[31:0]	I	protocol0_esm_ext_tx_misc_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When phy_ext_ctrl_sel=1, protocol*_*ext_tx_misc_r* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ External overrides protocol(0,1,2)_*ext_sup_misc(g1,g2,g3,g4,g5) are enabled by protocol_ext_sup_misc_ovrd_en. For more information, contact the "Support Center". ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_ropll_cp_ctl_intg_r0[27:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_ropll_cp_ctl_intg_r1[27:0]	I	protocol0_esm_ext_tx_ropll_cp_ctl_intg_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_ropll_cp_ctl_prop_r0[27:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_ropll_cp_ctl_prop_r1[27:0]	I	protocol0_esm_ext_tx_ropll_cp_ctl_prop_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_ropll_fbdiv_r0[27:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_ropll_fbdiv_r1[27:0]	I	protocol0_esm_ext_tx_ropll_fbdiv_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_ropll_postdiv_r0[11:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_ropll_postdiv_r1[11:0]	I	protocol0_esm_ext_tx_ropll_postdiv_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_postdiv_ovrd_en = 1, protocol_ext_*tx_ropll_postdiv* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_ropll_rc_filter_r0[11:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_ropll_rc_filter_r1[11:0]	I	protocol0_esm_ext_tx_ropll_rc_filter_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_ropll_refdiv_r0[15:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_ropll_refdiv_r1[15:0]	I	protocol0_esm_ext_tx_ropll_refdiv_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_refdiv_ovrd_en = 1, protocol_ext_*tx_ropll_refdiv_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_ropll_refsel_r0[7:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_ropll_refsel_r1[7:0]	I	protocol0_esm_ext_tx_ropll_refsel_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_ropll_v2i_mode_r0[7:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_ropll_v2i_mode_r1[7:0]	I	protocol0_esm_ext_tx_ropll_v2i_mode_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_ropll_vco_low_freq_r0[7:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_ropll_vco_low_freq_r1[7:0]	I	protocol0_esm_ext_tx_ropll_vco_low_freq_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_ropll_word_clk_div_sel_r0[7:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_ropll_word_clk_div_sel_r1[7:0]	I	protocol0_esm_ext_tx_ropll_word_clk_div_sel_r1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_word_clk_div_sel_ovrd_en = 1, protocol_ext_*tx_ropll_word_clk_div_sel* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_esm_ext_tx_term_ctrl_r0[3:0]	I	protocol0_esm_*_r0 are unused ■ These inputs can be tied to 'h0 **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_esm_ext_tx_term_ctrl_r1[3:0]	I	protocol0_esm_ext_rx_eq_buf_gain_r1 is a device-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_mplla_fb_clk_div4_en (for P = 0; P <= 2)	I	protocolP_ext_mplla_fb_clk_div4_en is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_mplla_frac_den[15:0] (for P = 0; P <= 2)	I	protocolP_ext_mplla_frac_den is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_mplla_frac_en (for P = 0; P <= 2)	I	protocolP_ext_mplla_frac_en is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_mplla_frac_quot[15:0] (for P = 0; P <= 2)	I	protocolP_ext_mplla_frac_quot is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_mplla_frac_rem[15:0] (for P = 0; P <= 2)	I	protocolP_ext_mplla_frac_rem is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_mplla_multiplier[11:0] (for P = 0; P <= 2)	I	protocolP_ext_mplla_multiplier is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_mplla_ssc_peak[19:0] (for P = 0; P <= 2)	I	protocolP_ext_mplla_ssc_peak is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_mplla_ssc_step_size[20:0] (for P = 0; P <= 2)	I	protocolP_ext_mplla_ssc_step_size is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_mplla_ssc_up_spread (for P = 0; P <= 2)	I	protocolP_ext_mplla_ssc_up_spread is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_mplla_word_clk_div[1:0] (for P = 0; P <= 2)	I	protocolP_ext_mplla_word_clk_div is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_mpllb_fb_clk_div4_en (for P = 0; P <= 2)	I	protocolP_ext_mpllb_fb_clk_div4_en is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_mpllb_frac_den[15:0] (for P = 0; P <= 2)	I	protocolP_ext_mpllb_frac_den is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_mpllb_frac_en (for P = 0; P <= 2)	I	protocolP_ext_mpllb_frac_en is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_mpllb_frac_quot[15:0] (for P = 0; P <= 2)	I	protocolP_ext_mpllb_frac_quot is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_mpllb_frac_rem[15:0] (for P = 0; P <= 2)	I	protocolP_ext_mpllb_frac_rem is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_mpllb_multiplier[11:0] (for P = 0; P <= 2)	I	protocolP_ext_mpllb_multiplier is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided.. **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_mpllb_ssc_peak[19:0] (for P = 0; P <= 2)	I	protocolP_ext_mpllb_ssc_peak is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_mpllb_ssc_step_size[20:0] (for P = 0; P <= 2)	I	protocolP_ext_mpllb_ssc_step_size is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_mpllb_ssc_up_spread (for P = 0; P <= 2)	I	protocolP_ext_mpllb_ssc_up_spread is a device-based PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_mpllb_word_clk_div[1:0] (for P = 0; P <= 2)	I	protocolP_ext_mpllb_word_clk_div is a device-based per-PHY configuration setting per-protocol. ■ External overrides for the per-protocol settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hardcoded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_ref_clk_mplla_div[2:0] (for P = 0; P <= 2)	I	protocolP_ext_ref_clk_mplla_div is a per-protocol device-based override for settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Synchronous To:** None **Active State:** N/A
protocolP_ext_ref_clk_mpllb_div[2:0] (for P = 0; P <= 2)	I	protocolP_ext_ref_clk_mpllb_div is a per-protocol device-based override for settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_adapt_mode_g1[11:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_adapt_mode_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_adapt_mode_g2[11:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_adapt_mode_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_adapt_mode_g3[11:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_adapt_mode_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_adapt_mode_g4[11:0]	I	protocol0_ext_rx_adapt_mode_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_adapt_mode_g5[11:0]	I	protocol0_ext_rx_adapt_mode_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_adapt_sel_piperate0[7:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_adapt_sel_piperate0 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_adapt_sel_piperate1[7:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_adapt_sel_piperate1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_adapt_sel_piperate2[7:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_adapt_sel_piperate2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_adapt_sel_piperate3[7:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_adapt_sel_piperate3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_adapt_sel_piperate4[7:0]	I	protocol0_ext_rx_adapt_sel_piperate4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_cdr_ppm_max_g1[19:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_cdr_ppm_max_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_cdr_ppm_max_g2[19:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_cdr_ppm_max_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_cdr_ppm_max_g3[19:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_cdr_ppm_max_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_cdr_ppm_max_g4[19:0]	I	protocol0_ext_rx_cdr_ppm_max_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_cdr_ppm_max_g5[19:0]	I	protocol0_ext_rx_cdr_ppm_max_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_cdr_vco_config_g1[47:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_cdr_vco_config_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_cdr_vco_config_g2[47:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_cdr_vco_config_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_cdr_vco_config_g3[47:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_cdr_vco_config_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_cdr_vco_config_g4[47:0]	I	protocol0_ext_rx_cdr_vco_config_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_cdr_vco_config_g5[47:0]	I	protocol0_ext_rx_cdr_vco_config_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_dcc_ctrl_cm_range_g1[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_dcc_ctrl_cm_range_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_dcc_ctrl_cm_range_g2[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_dcc_ctrl_cm_range_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_dcc_ctrl_cm_range_g3[3:0]	I	protocol0_ext_rx_dcc_ctrl_cm_range_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_dcc_ctrl_cm_range_g4[3:0]	I	protocol0_ext_rx_dcc_ctrl_cm_range_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_dcc_ctrl_cm_range_g5[3:0]	I	protocol0_ext_rx_dcc_ctrl_cm_range_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_dcc_ctrl_diff_range_g1[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_dcc_ctrl_diff_range_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_dcc_ctrl_diff_range_g2[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_dcc_ctrl_diff_range_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_dcc_ctrl_diff_range_g3[3:0]	I	protocol0_ext_rx_dcc_ctrl_diff_range_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_dcc_ctrl_diff_range_g4[3:0]	I	protocol0_ext_rx_dcc_ctrl_diff_range_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_dcc_ctrl_diff_range_g5[3:0]	I	protocol0_ext_rx_dcc_ctrl_diff_range_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_delta_iq_g1[15:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_delta_iq_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_delta_iq_g2[15:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_delta_iq_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_delta_iq_g3[15:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_delta_iq_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_delta_iq_g4[15:0]	I	protocol0_ext_rx_delta_iq_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_delta_iq_g5[15:0]	I	protocol0_ext_rx_delta_iq_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_dfe_bypass_g1[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_dfe_bypass_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_dfe_bypass_g2[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_dfe_bypass_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_dfe_bypass_g3[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_dfe_bypass_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_dfe_bypass_g4[3:0]	I	protocol0_ext_rx_dfe_bypass_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_dfe_bypass_g5[3:0]	I	protocol0_ext_rx_dfe_bypass_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:**he protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_afe_config_g1[47:0] (for P = 0; P <= 2)	I	protocol0_ext_rx_eq_afe_config_g1 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_afe_config_g2[47:0] (for P = 0; P <= 2)	I	protocol0_ext_rx_eq_afe_config_g2 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_afe_config_g3[47:0] (for P = 0; P <= 2)	I	protocol0_ext_rx_eq_afe_config_g3 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_afe_config_g4[47:0]	I	protocol0_ext_rx_eq_afe_config_g4 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_afe_config_g5[47:0]	I	protocol0_ext_rx_eq_afe_config_g5 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_afe_rate_g1[11:0] (for P = 0; P <= 2)	I	protocol0_ext_rx_eq_afe_rate_g1 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_afe_rate_g2[11:0] (for P = 0; P <= 2)	I	protocol0_ext_rx_eq_afe_rate_g2 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_afe_rate_g3[11:0] (for P = 0; P <= 2)	I	protocol0_ext_rx_eq_afe_rate_g3 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_afe_rate_g4[11:0]	I	protocol0_ext_rx_eq_afe_rate_g4 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_afe_rate_g5[11:0]	I	protocol0_ext_rx_eq_afe_rate_g5 is a lane-based PHY configuration setting override per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_att_lvl_g1[11:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_att_lvl_g1 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_att_lvl_g2[11:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_att_lvl_g2 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_att_lvl_g3[11:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_att_lvl_g3 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_att_lvl_g4[11:0]	I	protocol0_ext_rx_eq_att_lvl_g4 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_att_lvl_g5[11:0]	I	protocol0_ext_rx_eq_att_lvl_g5 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_ctle_boost_g1[19:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_ctle_boost_g1 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_ctle_boost_g2[19:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_ctle_boost_g2 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_ctle_boost_g3[19:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_ctle_boost_g3 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_ctle_boost_g4[19:0]	I	protocol0_ext_rx_eq_ctle_boost_g4 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_ctle_boost_g5[19:0]	I	protocol0_ext_rx_eq_ctle_boost_g5 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_ctle_pole_g1[7:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_ctle_pole_g1 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_ctle_pole_g2[7:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_ctle_pole_g2 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_ctle_pole_g3[7:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_ctle_pole_g3 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_ctle_pole_g4[7:0]	I	protocol0_ext_rx_eq_ctle_pole_g4 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_ctle_pole_g5[7:0]	I	protocol0_ext_rx_eq_ctle_pole_g5 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_dfe_float_en_g1[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_dfe_float_en_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_eq_dfe_float_en_g2[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_dfe_float_en_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_dfe_float_en_g3[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_dfe_float_en_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_eq_dfe_float_en_g4[3:0]	I	protocol0_ext_rx_eq_dfe_float_en_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_dfe_float_en_g5[3:0]	I	protocol0_ext_rx_eq_dfe_float_en_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_dfe_tap1_g1[35:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_dfe_tap1_g1 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_dfe_tap1_g2[35:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_dfe_tap1_g2 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_dfe_tap1_g3[35:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_dfe_tap1_g3 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_dfe_tap1_g4[35:0]	I	protocol0_ext_rx_eq_dfe_tap1_g4 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_dfe_tap1_g5[35:0]	I	protocol0_ext_rx_eq_dfe_tap1_g5 is a lane-based receiver equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's RX equalization inputs. ■ The PCIe 5.0 PCS internally determines the hard-coded optimal RX equalization settings for each protocol and rate. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Note:** ■ Because protocol1 supports only two rates, protocol1_ext_*_g3 inputs are not provided. ■ Because only protocol0 supports four rates, protocol(*)_ext_*_g4 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_rx_eq_dfe_tap1_({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_dfe_tap2_g1[7:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_dfe_tap2_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_eq_dfe_tap2_g2[7:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_dfe_tap2_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_dfe_tap2_g3[7:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_dfe_tap2_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_eq_dfe_tap2_g4[7:0]	I	protocol0_ext_rx_eq_dfe_tap2_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_dfe_tap2_g5[7:0]	I	protocol0_ext_rx_eq_dfe_tap2_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_eq_vga_gain_g1[4:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_vga_gain_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_eq_vga_gain_g2[4:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_vga_gain_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_eq_vga_gain_g3[4:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_eq_vga_gain_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_eq_vga_gain_g4[4:0]	I	protocol0_ext_rx_eq_vga_gain_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_eq_vga_gain_g5[4:0]	I	protocol0_ext_rx_eq_vga_gain_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_los_pwr_up_cnt[10:0] (for P = 0; P <= 2)	I	Unused. To be removed in future update. **Synchronous To:** None **Active State:** N/A
protocolP_ext_rx_misc_g1[31:0] (for P = 0; P <= 2)	I	protocolP_ext_ rx_misc_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_misc_ovrd_en, protocol_ext_rx_misc_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ External overrides protocol(0,1,2)_ext_rx_misc(g1,g2,g3,g4,g5) are enabled by protocol_ext_rx_misc_ovrd_en. For more information, contact the "Support Center". ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_misc_g2[31:0] (for P = 0; P <= 2)	I	protocolP_ext_ rx_misc_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_misc_ovrd_en, protocol_ext_rx_misc_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ External overrides protocol(0,1,2)_ext_rx_misc(g1,g2,g3,g4,g5) are enabled by protocol_ext_rx_misc_ovrd_en. For more information, contact the "Support Center". ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_misc_g3[31:0]	I	protocolP_ext_ rx_misc_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_misc_ovrd_en, protocol_ext_rx_misc_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ External overrides protocol(0,1,2)_ext_rx_misc(g1,g2,g3,g4,g5) are enabled by protocol_ext_rx_misc_ovrd_en. For more information, contact the "Support Center". ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_misc_g4[31:0]	I	protocol0_ext_ rx_misc_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_misc_ovrd_en, protocol_ext_rx_misc_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ External overrides protocol(0,1,2)_ext_rx_misc(g1,g2,g3,g4,g5) are enabled by protocol_ext_rx_misc_ovrd_en. For more information, contact the "Support Center". ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_misc_g5[31:0]	I	protocol0_ext_ rx_misc_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_rx_misc_ovrd_en, protocol_ext_rx_misc_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ External overrides protocol(0,1,2)_ext_rx_misc(g1,g2,g3,g4,g5) are enabled by protocol_ext_rx_misc_ovrd_en. For more information, contact the "Support Center". ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_ref_ld_val_g1[6:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_ref_ld_val_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_ref_ld_val_g2[6:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_ref_ld_val_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_ref_ld_val_g3[6:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_ref_ld_val_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_ref_ld_val_g4[6:0]	I	protocol0_ext_rx_ref_ld_val_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_ref_ld_val_g5[6:0]	I	protocolP_ext_rx_ref_ld_val_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_sigdet_lf_threshold_g1[2:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_sigdet_lf_threshold_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_sigdet_lf_threshold_g2[2:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_sigdet_lf_threshold_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_sigdet_lf_threshold_g3[2:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_sigdet_lf_threshold_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_sigdet_lf_threshold_g4[2:0]	I	protocolP_ext_rx_sigdet_lf_threshold_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_sigdet_lf_threshold_g5[2:0]	I	protocol0_ext_rx_sigdet_lf_threshold_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_sigdet_lfps_filter_en_g1 (for P = 0; P <= 2)	I	protocolP_ext_rx_sigdet_lfps_filter_en_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_sigdet_lfps_filter_en_g2 (for P = 0; P <= 2)	I	protocolP_ext_rx_sigdet_lfps_filter_en_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_sigdet_lfps_filter_en_g3 (for P = 0; P <= 2)	I	protocolP_ext_rx_sigdet_lfps_filter_en_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_sigdet_lfps_filter_en_g4	I	protocol0_ext_rx_sigdet_lfps_filter_en_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_sigdet_lfps_filter_en_g5	I	protocol0_ext_rx_sigdet_lfps_filter_en_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_term_ctrl_g1[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_term_ctrl_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_term_ctrl_g2[3:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_term_ctrl_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_term_ctrl_g3[3:0]	I	protocol0_ext_rx_term_ctrl_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_term_ctrl_g4[3:0]	I	protocol0_ext_rx_term_ctrl_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_term_ctrl_g5[3:0]	I	protocol0_ext_rx_term_ctrl_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_vco_ld_val_g1[12:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_vco_ld_val_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_vco_ld_val_g2[12:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_vco_ld_val_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_rx_vco_ld_val_g3[12:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_vco_ld_val_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_rx_vco_ld_val_g4[12:0]	I	protocol0_ext_rx_vco_ld_val_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_rx_vco_ld_val_g5[12:0]	I	protocol0_ext_rx_vco_ld_val_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_rx_vref_ctrl[4:0] (for P = 0; P <= 2)	I	protocolP_ext_rx_vref_ctrl is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_dcc_ctrl_cm_range_g1[3:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_dcc_ctrl_cm_range_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_dcc_ctrl_cm_range_g2[3:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_dcc_ctrl_cm_range_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_dcc_ctrl_cm_range_g3[3:0]	I	protocol0_ext_tx_dcc_ctrl_cm_range_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_dcc_ctrl_cm_range_g4[3:0]	I	protocol0_ext_tx_dcc_ctrl_cm_range_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_dcc_ctrl_cm_range_g5[3:0]	I	protocol0_ext_tx_dcc_ctrl_cm_range_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_dcc_ctrl_diff_range_g1[3:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_dcc_ctrl_diff_range_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_dcc_ctrl_diff_range_g2[3:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_dcc_ctrl_diff_range_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_dcc_ctrl_diff_range_g3[3:0]	I	protocol0_ext_tx_dcc_ctrl_diff_range_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_dcc_ctrl_diff_range_g4[3:0]	I	protocol0_ext_tx_dcc_ctrl_diff_range_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_dcc_ctrl_diff_range_g5[3:0]	I	protocol0_ext_tx_dcc_ctrl_diff_range_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_main_g1[23:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_main_g1 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g1 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_main_g2[23:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_main_g2 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g2 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_main_g3[23:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_main_g3 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g3 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_eq_main_g4[23:0]	I	protocol0_ext_tx_eq_main_g4 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g4 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_eq_main_g5[23:0]	I	protocol0_ext_tx_eq_main_g5 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g5 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_ovrd_g1[3:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_ovrd_g1 is a lane-based transmitter equalization setting override enable ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g1 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_ovrd_g2[3:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_ovrd_g2 is a lane-based transmitter equalization setting override enable ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g2 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_ovrd_g3[3:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_ovrd_g3 is a lane-based transmitter equalization setting override enable ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g3 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_eq_ovrd_g4[3:0]	I	protocol0_ext_tx_eq_ovrd_g4 is a lane-based transmitter equalization setting override enable ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g4 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_eq_ovrd_g5[3:0]	I	protocol0_ext_tx_eq_ovrd_g5 is a lane-based transmitter equalization setting override enable ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g5 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_post_g1[19:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_post_g1 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g1 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_post_g2[19:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_post_g2 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g2 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_post_g3[19:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_post_g3 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g3 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_eq_post_g4[19:0]	I	protocol0_ext_tx_eq_post_g4 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g4 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_eq_post_g5[19:0]	I	protocol0_ext_tx_eq_post_g5 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g5 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_pre_g1[19:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_pre_g1 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g1 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: ■ .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}), Example with a single programmable register driving all lanes to a common value: ■ .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}), **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_pre_g2[19:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_pre_g2 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g2 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_eq_pre_g3[19:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_eq_pre_g3 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g3 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_eq_pre_g4[19:0]	I	protocol0_ext_tx_eq_pre_g4 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g4 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_eq_pre_g5[19:0]	I	protocol0_ext_tx_eq_pre_g5 is a lane-based transmitter equalization setting overrides. ■ External overrides for the per-protocol settings of the PHY's TX equalization inputs. The PCIe 5.0 PCS internally determines the hard-coded optimal TX equalization settings for each protocol and rate. However, these settings can be overwritten on protocol(*)_ext_tx_eq_(*)_(*), a per-protocol basis from these top-level pins when the corresponding protocol(*)_ext_tx-_eq_ovrd_g5 input is asserted. ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,10, respectively. **Note:** ■ Because only protocol0 supports five rates, protocol1/2_ext_*_g4/5 inputs are not provided. ■ Because protocol1 supports only two rates, protocol1_ext_*_g3/4/5 inputs are not provided. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_eq_main_g1({NLANES{5'd0}}). Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_eq_main_g1({NLANES{<REG_A>}}). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_fastedge_en_g1 (for P = 0; P <= 2)	I	protocolP_ext_tx_fastedge_en_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_fastedge_en_g2 (for P = 0; P <= 2)	I	protocolP_ext_tx_fastedge_en_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_fastedge_en_g3 (for P = 0; P <= 2)	I	protocolP_ext_tx_fastedge_en_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_fastedge_en_g4	I	protocol0_ext_tx_fastedge_en_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_fastedge_en_g5	I	protocol0_ext_tx_fastedge_en_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_misc_g1[31:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_misc_g1 is a lane-based per-protocol override for settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_misc_g2[31:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_misc_g2 is a lane-based per-protocol override for settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Synchronous To:** None **Active State:** N/A
protocol0_ext_tx_misc_g3[31:0]	I	protocol0_ext_tx_misc_g3 is a lane-based per-protocol override for settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Synchronous To:** None **Active State:** N/A
protocol0_ext_tx_misc_g4[31:0]	I	protocol0_ext_tx_misc_g4 is a lane-based per-protocol override for settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_misc_g5[31:0]	I	protocol0_ext_tx_misc_g5 is a lane-based per-protocol override for settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Synchronous To:** None **Active State:** N/A
protocolP_ext_tx_ropll_cp_ctl_intg_g1[27:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_cp_ctl_intg_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_cp_ctl_intg_g2[27:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_cp_ctl_intg_g2(6:0) is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_cp_ctl_intg_g3[27:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_cp_ctl_intg_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_ropll_cp_ctl_intg_g4[27:0]	I	protocol0_ext_tx_ropll_cp_ctl_intg_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_ropll_cp_ctl_intg_g5[27:0]	I	protocol0_ext_tx_ropll_cp_ctl_intg_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_cp_ctl_prop_g1[27:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_cp_ctl_prop_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_cp_ctl_prop_g2[27:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_cp_ctl_prop_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_cp_ctl_prop_g3[27:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_cp_ctl_prop_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_ropll_cp_ctl_prop_g4[27:0]	I	protocol0_ext_tx_ropll_cp_ctl_prop_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_cp_ctl_prop_g5[27:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_cp_ctl_prop_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_fbdiv_g1[27:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_fbdiv_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_fbdiv_g2[27:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_fbdiv_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_fbdiv_g3[27:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_fbdiv_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_ropll_fbdiv_g4[27:0]	I	protocol0_ext_tx_ropll_fbdiv_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_ropll_fbdiv_g5[27:0]	I	protocolP_ext_tx_ropll_fbdiv_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_postdiv_g1[11:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_postdiv_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_postdiv_ovrd_en = 1, protocol_ext_*postdiv_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_postdiv_g2[11:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_postdiv_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_postdiv_ovrd_en = 1, protocol_ext_*postdiv_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_postdiv_g3[11:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_postdiv_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_postdiv_ovrd_en = 1, protocol_ext_*postdiv_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_ropll_postdiv_g4[11:0]	I	protocol0_ext_tx_ropll_postdiv_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_postdiv_ovrd_en = 1, protocol_ext_*postdiv_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_ropll_postdiv_g5[11:0]	I	protocol0_ext_tx_ropll_postdiv_g5 is a lane-based PHY configuration setting per protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_postdiv_ovrd_en = 1, protocol_ext_*postdiv_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) **Note:** External override protocol0_esm_ext_sup_misc_r1(*) is enabled by protocol_ext_sup_misc_ovrd_en. For more information, contact the "Support Center". Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_rc_filter_g1[11:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_rc_filter_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_rc_filter_g2[11:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_rc_filter_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_rc_filter_g3[11:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_rc_filter_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_ropll_rc_filter_g4[11:0]	I	protocol0_ext_tx_ropll_rc_filter_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_ropll_rc_filter_g5[11:0]	I	protocol0_ext_tx_ropll_rc_filter_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_refdiv_g1[15:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_refdiv_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_refdiv_ovrd_en = 1, protocol_ext_*tx_ropll_refdiv_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_refdiv_g2[15:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_refdiv_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_refdiv_ovrd_en = 1, protocol_ext_*tx_ropll_refdiv_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_refdiv_g3[15:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_refdiv_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_refdiv_ovrd_en = 1, protocol_ext_*tx_ropll_refdiv_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_ropll_refdiv_g4[15:0]	I	protocol0_ext_tx_ropll_refdiv_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_refdiv_ovrd_en = 1, protocol_ext_*tx_ropll_refdiv_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_ropll_refdiv_g5[15:0]	I	protocol0_ext_tx_ropll_refdiv_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_refdiv_ovrd_en = 1, protocol_ext_*tx_ropll_refdiv_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_refsel_g1[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_refsel_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_refsel_g2[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_refsel_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_refsel_g3[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_refsel_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_ropll_refsel_g4[7:0]	I	protocol0_ext_tx_ropll_refsel_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_ropll_refsel_g5[7:0]	I	protocol0_ext_tx_ropll_refsel_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_v2i_mode_g1[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_v2i_mode_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_v2i_mode_g2[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_v2i_mode_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_v2i_mode_g3[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_v2i_mode_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_ropll_v2i_mode_g4[7:0]	I	protocol0_ext_tx_ropll_v2i_mode_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_ropll_v2i_mode_g5[7:0]	I	protocol0_ext_tx_ropll_v2i_mode_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_vco_low_freq_g1[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_vco_low_freq_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_vco_low_freq_g2[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_vco_low_freq_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_vco_low_freq_g3[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_vco_low_freq_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_ropll_vco_low_freq_g4[7:0]	I	protocol0_ext_tx_ropll_vco_low_freq_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_ropll_vco_low_freq_g5[7:0]	I	protocol0_ext_tx_ropll_vco_low_freq_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ When protocol_ext_tx_ropll_ovrd_en = 1, protocol_ext_*tx_ropll_* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_word_clk_div_sel_g1[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_word_clk_div_sel_g1 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the protocol_ext_tx_ropll_word_-clk_div_sel_ovrd_en input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_ropll_word_clk_div_sel_g2[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_word_clk_div_sel_g2 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the protocol_ext_tx_ropll_word_-clk_div_sel_ovrd_en input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_ropll_word_clk_div_sel_g3[7:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_ropll_word_clk_div_sel_g3 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the protocol_ext_tx_ropll_word_-clk_div_sel_ovrd_en input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_ropll_word_clk_div_sel_g4[7:0]	I	protocol0_ext_tx_ropll_word_clk_div_sel_g4 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the protocol_ext_tx_ropll_word_-clk_div_sel_ovrd_en input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_ropll_word_clk_div_sel_g5[7:0]	I	protocol0_ext_tx_ropll_word_clk_div_sel_g5 is a lane-based PHY configuration setting per-protocol. ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the protocol_ext_tx_ropll_word_-clk_div_sel_ovrd_en input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocolP_ext_tx_term_ctrl_g1[3:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_term_ctrl_g1 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocolP_ext_tx_term_ctrl_g2[3:0] (for P = 0; P <= 2)	I	protocolP_ext_tx_term_ctrl_g2 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_term_ctrl_g3[3:0]	I	protocol0_ext_tx_term_ctrl_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol0_ext_tx_term_ctrl_g4[3:0]	I	protocol0_ext_tx_term_ctrl_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol0_ext_tx_term_ctrl_g5[3:0]	I	protocol0_ext_tx_term_ctrl_g5 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol2_ext_rx_dcc_ctrl_cm_range_g3[3:0]	I	protocol2_ext_rx_dcc_ctrl_cm_range_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol2_ext_rx_dcc_ctrl_diff_range_g3[3:0]	I	protocol2_ext_rx_dcc_ctrl_diff_range_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol2_ext_rx_term_ctrl_g3[3:0]	I	protocol2_ext_rx_term_ctrl_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol2_ext_tx_dcc_ctrl_cm_range_g3[3:0]	I	protocol2_ext_tx_dcc_ctrl_cm_range_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol2_ext_tx_dcc_ctrl_diff_range_g3[3:0]	I	protocol2_ext_tx_dcc_ctrl_diff_range_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol2_ext_tx_misc_g2[31:0]	I	protocolP_ext_rx_sigdet_lf_threshold_g4 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol2_ext_tx_misc_g3[31:0]	I	protocol0_ext_tx_misc_g3 is a lane-based per-protocol override for settings of the PHY configuration inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01, and 10, respectively. **Synchronous To:** Asynchronous **Active State:** N/A
protocol2_ext_tx_term_ctrl_g3[3:0]	I	protocol2_ext_tx_term_ctrl_g3 is a device-based PHY configuration setting per-protocol. ■ Device-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol_ext_mplla_recal_bank_sel[1:0]	I	protocol_ext_mplla_recal_bank_sel is a device-based PHY configuration settings. ■ When protocol_ext_mplla_recal_bank_sel_ovrd_en = 1, protocol_ext_mplla_recal_bank_sel is used to drive the mplla_recal_bank_sel for each PHY in the device; otherwise, the PCS drives this signal. ■ For more information about mplla/b_recal_bank_sel, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter. **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_mplla_recal_bank_sel_ovrd_en	I	protocol_ext_mplla_recal_bank_sel_ovrd_en is a device-based PHY configuration settings override enable. ■ When protocol_ext_mplla_recal_bank_sel_ovrd_en = 1, protocol_ext_mplla_recal_bank_sel is used to drive the mplla_recal_bank_sel for each PHY in the device; otherwise, the PCS drives this signal. ■ For more information about mplla/b_recal_bank_sel, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter. **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_mpllb_recal_bank_sel[1:0]	I	protocol_ext_mpllb_recal_bank_sel(1:0) is a device-based PHY configuration settings. ■ When protocol_ext_mpllb_recal_bank_sel_ovrd_en = 1, protocol_ext_mpllb_recal_bank_sel is used to drive the mpllb_recal_bank_sel for each PHY in the device; otherwise, the PCS drives this signal. ■ For more information about mplla/b_recal_bank_sel, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter. **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_mpllb_recal_bank_sel_ovrd_en	I	protocol_ext_mpllb_recal_bank_sel_ovrd_en is a device-based PHY configuration settings override enable. ■ When protocol_ext_mpllb_recal_bank_sel_ovrd_en = 1, protocol_ext_mpllb_recal_bank_sel is used to drive the mpllb_recal_bank_sel for each PHY in the device; otherwise, the PCS drives this signal. ■ For more information about mplla/b_recal_bank_sel, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter. **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol_ext_phyN_mplla_bw_high[15:0] (for N = 0; N <= Nphy-1)	I	protocol_ext_phyN_mplla_bw_high is a per-PHY PHY configuration setting per-protocol. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_phyN_mplla_bw_low[15:0] (for N = 0; N <= Nphy-1)	I	protocol_ext_phyN_mplla_bw_low is a per-PHY PHY configuration setting per-protocol. (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol_ext_phyN_mplla_bw_threshold[7:0] (for N = 0; N <= Nphy-1)	I	protocol_ext_phyN_mplla_bw_threshold is a per-PHY PHY configuration setting per-protocol. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_phyN_mplla_tx_clk_div[3:0] (for N = 0; N <= Nphy-1)	I	protocol_ext_phy0_mplla_tx_clk_div is a per-PHY configuration settings override ■ When protocol_ext_phy0_mplla_tx_clk_div_ovrd_en = 1, protocol_ext_phy0_mplla_tx_clk_div is used to drive the mplla_tx_clk_div input to the PHY; otherwise, the PCS drives this signal. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_phyN_mplla_tx_clk_div_ovrd_en (for N = 0; N <= Nphy-1)	I	protocol_ext_phy0_mplla_tx_clk_div_ovrd_en is a per-PHY configuration settings override enable. ■ When protocol_ext_phy0_mplla_tx_clk_div_ovrd_en = 1, protocol_ext_phy0_mplla_tx_clk_div is used to drive the mplla_recal_bank_sel input to the PHY; otherwise, the PCS drives this signal. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol_ext_phyN_mpllb_bw_high[15:0] (for N = 0; N <= Nphy-1)	I	protocol_ext_phyN_mpllb_bw_high is a per-PHY PHY configuration setting per-protocol. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_phyN_mpllb_bw_low[15:0] (for N = 0; N <= Nphy-1)	I	protocol_ext_phyN_mpllb_bw_low is a per-PHY configuration setting. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol_ext_phyN_mpllb_bw_threshold[7:0] (for N = 0; N <= Nphy-1)	I	protocol_ext_phyN_mpllb_bw_threshold is a per-PHY configuration setting. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_phyN_mpllb_tx_clk_div[3:0] (for N = 0; N <= Nphy-1)	I	protocol_ext_phy0_mpllb_tx_clk_div is a per-PHY configuration settings override ■ When protocol_ext_phy0_mpllb_tx_clk_div_ovrd_en = 1, protocol_ext_phy0_mpllb_tx_clk_div is used to drive the mpllb_tx_clk_div input to the PHY; otherwise, the PCS drives this signal. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_phyN_mpllb_tx_clk_div_ovrd_en (for N = 0; N <= Nphy-1)	I	protocol_ext_phy0_mpllb_tx_clk_div_ovrd_en is a per-PHY configuration settings override enable. ■ When protocol_ext_phy0_mpllb_tx_clk_div_ovrd_en = 1, protocol_ext_phy0_mpllb_tx_clk_div is used to drive the mpllb_recal_bank_sel input to the PHY; otherwise, the PCS drives this signal. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol_ext_rx_eq_afe_ovrd_en	I	protocol_ext_rx_eq_afe_ovrd_en is a device-based PHY configuration settings override enable ■ When protocol_ext_rx_eq_afe_ovrd_en = 1, protocol_ext_*afe* inputs are used to drive the phy inputs; otherwise, the PCS drives these inputs. ■ (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). **Note:** ■ The protocol_* inputs are not protocol specific. Example tie-off to constant value: .protocol_ext_rx_eq_afe_ovrd_en(1'b0) **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_rx_misc_ovrd_en	I	This signal (protocol_ext_rx_misc_ovrd_en) enables protocol*_ext_rx_misc_(g1,g2,g3,g4,g5) and protocol0_esm_ext_rx_misc_rX. **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_tx_ropll_ovrd_en	I	protocol_ext_tx_ropll_ovrd_en is a PHY configuration override enable ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ External overrides protocol(0,1,2)_ext_sup_misc(g1,g2,g3,g4,g5) are enabled by protocol_ext_sup_misc_ovrd_en. For more information, contact the "Support Center". ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol_ext_tx_ropll_postdiv_ovrd_en[3:0]	I	protocol_ext_tx_ropll_postdiv_ovrd_en is a lane-based PHY configuration setting override enable ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A
protocol_ext_tx_ropll_refdiv_ovrd_en[3:0]	I	protocol_ext_tx_ropll_refdiv_ovrd_en is a lane-based PHY configuration setting override enable**Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
protocol_ext_tx_ropll_word_clk_div_sel_ovrd_en[3:0]	I	protocol_ext_tx_ropll_word_clk_div_sel_ovrd_en is a lane-based PHY configuration override enable ■ Lane-based external overrides for the per-protocol settings of the PHY configuration inputs (For per-protocol settings to configure the PHY, refer to the DesignWare Cores PHY databook, "PHY Usage and Configuration" chapter). ■ The PCS internally determines the hard-coded optimal settings for each protocol. ■ However, these settings can be overwritten on a per-protocol basis from these top-level pins when the phy_ext_ctrl_sel input is asserted. **Note:** ■ The protocol0, protocol1, and protocol2 signals correspond to the pipe_laneX_protocol(*) value of 00, 01,and 10, respectively. ■ These signals are concatenated buses whose width is dependent on the number of lanes in the PCS (NLANES). The buses are ordered from highest lane in the MSBs to lane0 in the LSBs. Example tie-off to constant value: .protocol0_ext_tx_vboost_lvl({NLANES{5'd0}}) Example with a single programmable register driving all lanes to a common value: .protocol0_ext_tx_vboost_lvl({NLANES{<REG_A>}}) **Synchronous To:** Asynchronous **Active State:** N/A

4.18 Parallel Control Register Port Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Table 4-18 Parallel Control Register Port Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Port Name	I/O	Description
phyN_cr_para_ack	O	CR Read/Write acknowledgement. ■ Acknowledge signal indicating completion of the CR parallel read/write access. ■ For more information about operation during SRAM update, see the "PHY Usage and Configuration" chapter. **Voltage Range:**0-vpdig **Synchronous To:** phyN_cr_para_clk **Active State:** High
phyN_cr_para_addr[15:0]	I	CR address. **Voltage Range:**0-vpdig **Synchronous To:** phyN_cr_para_clk **Active State:** N/A
phyN_cr_para_clk	I	CR clock. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_cr_para_rd_data[15:0]	O	CR read data. **Voltage Range:**0-vpdig **Synchronous To:** phyN_cr_para_clk **Active State:** N/A
phyN_cr_para_rd_en	I	CR read enable. Reads from the referenced address on cr_para_addr and provides the data on cr_para_rd_data. **Voltage Range:**0-vpdig **Synchronous To:** phyN_cr_para_clk **Active State:** High

Port Name	I/O	Description
phyN_cr_para_sel	I	Control Register (CR) parallel interface select. Controls selection between JTAG and CR interfaces: ■ 0: JTAG ■ 1: Control Register (CR) This input can only be changed when the cr_para_clk and jtag_tck clock inputs are disabled. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_cr_para_wr_data[15:0]	I	CR write data. **Voltage Range:**0-vpdig **Synchronous To:** phyN_cr_para_clk **Active State:** N/A
phyN_cr_para_wr_en	I	CR write enable. Writes the data on cr_para_wr_data to the referenced address on cr_para_addr. **Voltage Range:**0-vpdig **Synchronous To:** phyN_cr_para_clk **Active State:** High

4.19 External ROM Control Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Table 4-19 External ROM Control Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Port Name	I/O	Description
phyN_rom_addr[14:0]	O	ROM address bus (rom_addr). Address bus for the external ROM. **Voltage Range:**0-vpdig **Synchronous To:** phyN_rom_clk **Active State:** N/A
phyN_rom_clk	O	ROM clock (rom_clk). Clock used to drive external ROM interface. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A
phyN_rom_rd_data[15:0]	I	ROM read data bus. Read data bus for the external ROM. **Voltage Range:**0-vpdig **Synchronous To:** phyN_rom_clk **Active State:** N/A

4.20 SRAM Control Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Table 4-20 SRAM Control Signals (top to/from phy) (for N = 0; N <= Nphy-1)

Port Name	I/O	Description
phyN_sram_addr[14:0]	O	SRAM address bus. ■ Address port for the SRAM. ■ Selected if macro *_SRAM_SUPPORT is defined. **Voltage Range:**0-vpdig **Synchronous To:** phyN_sram_clk **Active State:** N/A
phyN_sram_bootload_bypass	I	SRAM bootload bypass. ■ Control signal when asserted, bypasses the SRAM bootloading function. ■ This signal can be used to skip bootloading when coming out of phy_reset. ■ For more information, see the "PHY Usage and Configuration" chapter. **Note:** This signal should only be changed during phy_reset and must not change after phy_reset is de-asserted. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_sram_bypass	I	SRAM bypass. ■ Control signal when asserted, bypasses the SRAM interface. ■ In this case, the adaptation and calibration algorithms are executed from the hard wired values within the Raw PCS. ■ If SRAM is not bypassed, the internal algorithms are first loaded by Raw PCS into the SRAM at which point user can change the contents of the SRAM. ■ The updated SRAM contents are used for the adaptation and calibration routines. **Note:** This signal should only be changed during phy_reset and must not change after phy_reset is de-asserted. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
phyN_sram_clk	O	SRAM Clock. ■ Clock used to drive SRAM interface. ■ Selected if macro *_SRAM_SUPPORT is defined. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_sram_ecc_en	I	SRAM ECC enable. Control signal when asserted, enable SRAM ECC inside RAW PCS. **Note:**This signal should only be changed during phy_reset and must not change after phy_reset is de-asserted. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_sram_ext_ld_done	I	SRAM external load done. Signal asserted by user after any updates to the SRAM have been loaded. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_sram_init_done	O	SRAM Initialization done. ■ Signal indicating that the SRAM has been initialized by the boot loader in the Raw PCS. ■ This signal does not assert if sram_bypass is asserted. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phyN_sram_rd_data[21:0]	I	SRAM read bus (sram_rd_data). ■ Read data port for the SRAM. ■ Selected if macro *_SRAM_SUPPORT is defined. **Voltage Range:**0-vpdig **Synchronous To:** phyN_sram_clk **Active State:** N/A
phyN_sram_rd_en	O	SRAM read enable. ■ Read enable control to read data from SRAM. ■ Selected if macro *_SRAM_SUPPORT is defined. **Voltage Range:**0-vpdig **Synchronous To:** phyN_sram_clk **Active State:** High

Port Name	I/O	Description
phyN_sram_wr_data[21:0]	O	SRAM read bus (sram_wr_data). ■ Write data port for the SRAM. ■ Selected if macro *_SRAM_SUPPORT is defined. **Voltage Range:**0-vpdig **Synchronous To:** phyN_sram_clk **Active State:** N/A
phyN_sram_wr_en	O	SRAM write enable. ■ Write enable control to write data to SRAM. ■ Selected if macro *_SRAM_SUPPORT is defined. **Voltage Range:**0-vpdig **Synchronous To:** phyN_sram_clk **Active State:** High

4.21 PHY Test Signals (top to/from phy)

Table 4-21 PHY Test Signals (top to/from phy)

Port Name	I/O	Description
phyN_dtb_out[1:0] (for N = 0; N <= Nphy-1)	O	Digital test bus output. Used in debug mode for probing the values of various internal signals in the PHY. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A
phyN_test_flyover_en (for N = 0; N <= Nphy-1)	I	Enable flyover test mode. ■ Flyover test mode enables direct access to the TX and RX bump pads, from the txX_flyover_data_p/m and rxX_flyover_data_p/m I/Os. ■ This mode can be enabled in either normal mode or scan mode. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phy_test_burnin	I	All circuits activator. ■ Activates all circuitry in the PHY for burn-in testing. ■ This signal must not be asserted while test_powerdown signal is set to 1'b1. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phy_test_powerdown	I	All circuits Power-Down control. Powers down all circuitry in the PHY for IDDQ testing. **Note:** The PHY is not functional in this mode and must be reset after this signal is de-asserted. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phy_test_stop_clk_en	I	Stop clock test mode enable. Reserved **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

4.22 Resistor Tune Signals (Common Block Group) (top to/from phy)

Table 4-22 Resistor Tune Signals (Common Block Group) (top to/from phy)

Port Name	I/O	Description
phyN_rx_term_offset[4:0] (for N = 0; N <= Nphy-1)	I	Offset for RX Termination. ■ Specifies an additional fixed offset to calibrated RX termination value. ■ This is a signed input with 2's complement encoding. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A
phyN_txdn_term_offset[8:0] (for N = 0; N <= Nphy-1)	I	Offset for TX Down Termination. ■ Specifies an additional fixed offset to calibrated TX down termination value. ■ This is a signed input with 2's complement encoding. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A
phyN_txup_term_offset[8:0] (for N = 0; N <= Nphy-1)	I	Offset for TX Up Termination. ■ Specifies an additional fixed offset to calibrated TX up termination value. ■ This is a signed input with 2's complement encoding. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A
phyN_res_ack_in (for N = 0; N <= Nphy-1)	I	For a description of res_ack_in, refer to the PHY databook. **Note:** ■ This I/O is included when IPNAME_SEP_RESREF_CALIB is defined. ■ If IPNAME_SEP_RESREF_CALIB is not defined, then there exists a single input phy_res_ack_in. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A

Port Name	I/O	Description
phyN_res_ack_out (for N = 0; N <= Nphy-1)	O	For a description of res_ack_out, refer to the PHY databook. **Note:** ■ This I/O is included when IPNAME_SEP_RESREF_CALIB is defined. ■ If IPNAME_SEP_RESREF_CALIB is not defined, then there exists a single output phy_res_ack_out. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A
phyN_res_req_in (for N = 0; N <= Nphy-1)	I	For a description of res_req_in, refer to the PHY databook. **Note:** ■ This I/O is included when IPNAME_SEP_RESREF_CALIB is defined. ■ If IPNAME_SEP_RESREF_CALIB is not defined, then there exists a single input phy_res_req_in. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A
phyN_res_req_out (for N = 0; N <= Nphy-1)	O	For a description of res_req_out, refer to the PHY databook. **Note:** ■ This I/O is included when IPNAME_SEP_RESREF_CALIB is defined. ■ If IPNAME_SEP_RESREF_CALIB is not defined, then there exists a single output phy_res_req_out. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** N/A
phy_rtune_ack	O	Resistor tune acknowledge. Indicates that a resistor tune has completed. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phy_rtune_req	I	Resistor tune request. Assertion triggers a resistor tune request (if one is not already in progress). **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

4.23 Ethernet Clock Signals (top to/from phy) (for X = 0; X <= Nlanes-1)

- phy_rxX_125mhz_clk
- phy_rxX_dig_div_clk
- phy_rxX_div33_clk
- phy_rxX_div66_clk
- phy_rxX_div_clk
- phy_txX_dig_ropll_div_clk
- phy_txX_ropll_125mhz_clk
- phy_txX_ropll_div33_clk
- phy_txX_ropll_div66_clk

Table 4-23 Ethernet Clock Signals (top to/from phy) (for X = 0; X <= Nlanes-1)

Port Name	I/O	Description
phy_rxX_125mhz_clk	O	Per-lane RX 125 MHz clock output for some Ethernet protocols (used by Synopsys xPCS IP) (rx0_125mhz_clk). ■ Generated by pulse-gating rxX_clk. ■ As a result, duty cycle of this clock will not be 50/50. ■ Also there will be period jitter, so synthesis and timing closure frequency will be greater than 125 MHz. ■ Refer to Synopsys synthesis constraints in the release package for details. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phy_rxX_dig_div_clk	O	Per-lane RX divided clock output for some Ethernet protocols. ■ Generated by pulse-gating rxX_clk. As a result, duty cycle of this clock will not be 50/50. ■ Also there will be period jitter, so synthesis and timing closure frequency will be greater than selected division ratio of rxX_clk. ■ Refer to Synopsys synthesis constraints in the release package for details. ■ Set this input as per the tables provided in the "Support Configuration Settings" appendix and the "Lane Configuration Settings" appendix of the DesignWare Cores PHY Databook. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
phy_rxX_div33_clk	O	Per-lane RX div33 clock output for some Ethernet protocols (used by Synopsys xPCS IP) (rx0_div33_clk). ■ Generated by pulse-gating rxX_clk. ■ As a result, duty cycle of this clock will not be 50/50. ■ Also there will be period jitter, so synthesis and timing closure frequency will be greater than div33 of rxX_clk. Refer to Synopsys synthesis constraints in the release package for details. ■ Set this input as per the tables provided in "Support Configuration Settings" appendix and the "Lane Configuration Settings" appendix. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phy_rxX_div66_clk	O	Per-lane RX div66 clock output for some Ethernet protocols (used by Synopsys xPCS IP) (rx0_div66_clk). ■ Generated by pulse-gating rxX_clk. ■ As a result, duty cycle of this clock will not be 50/50. ■ Also there will be period jitter, so synthesis and timing closure frequency will be greater than div66 of rxX_clk. Refer to Synopsys synthesis constraints in the release package for details. ■ Set this input as per the tables provided in"Support Configuration Settings" appendix and the "Lane Configuration Settings" appendix. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phy_rxX_div_clk	O	phy_rx0_div_clk **Synchronous To:** N/A **Active State:** N/A
phy_txX_dig_ropll_div_clk	O	Per-lane TX divided clock output for some Ethernet protocols. ■ Generated by pulse-gating txX_ropll_dword_clk. As a result, duty cycle of this clock will not be 50/50. ■ Also there will be period jitter, so synthesis and timing closure frequency will be greater than selected division ratio of txX_ropll_d-word_clk. ■ Refer to Synopsys synthesis constraints in the release package for details. ■ Set this input as per the tables provided in the "Support Configura-tion Settings" appendix and the "Lane Configuration Settings" appendix of the DesignWare Cores PHY Databook. **Voltage Range:**0-vpdig **Synchronous To:** N/A **Active State:** N/A

Port Name	I/O	Description
phy_txX_ropll_125mhz_clk	O	Per-lane TX 125 MHz clock enable signal for some Ethernet protocols (used by Synopsys xPCS IP) (tx0_ropll_125mhz_clk). Set this input as per the tables provided in "Support Configuration Settings" appendix and the "Lane Configuration Settings" appendix. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phy_txX_ropll_div33_clk	O	Per-lane TX div33 clock output for some Ethernet protocols (used by Synopsys xPCS IP) (tx0_ropll_div33_clk). ■ Generated by pulse-gating txX_ropll_dword_clk output. ■ As a result, duty cycle of this clock will not be 50/50. ■ Also there will be period jitter, so synthesis and timing closure frequency will be greater than div33. Refer to Synopsys synthesis constraints in the release package for details. ■ Set this input as per the tables provided in "Support Configuration Settings" appendix and the "Lane Configuration Settings" appendix. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High
phy_txX_ropll_div66_clk	O	Per-lane TX div66 clock output for some Ethernet protocols (used by Synopsys xPCS IP) (tx0_ropll_div66_clk). ■ Generated by pulse-gating txX_ropll_dword_clk output. ■ As a result, duty cycle of this clock will not be 50/50. ■ Also there will be period jitter, so synthesis and timing closure frequency will be greater than div66. Refer to Synopsys synthesis constraints in the release package for details. ■ Set this input as per the tables provided in "Support Configuration Settings" appendix and the "Lane Configuration Settings" appendix. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High

4.24 Context Select Interface Signals (top to/from pcs) (for X = 0; X <= Nlanes-1)

Table 4-24 Context Select Interface Signals (top to/from pcs) (for X = 0; X <= Nlanes-1)

Port Name	I/O	Description
phy_rxX_cntx_sel_esm20[7:0]	I	Per-lane Rx context select for esm 20 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_cntx_sel description. ■ This input must be controlled using a programmable register. ■ If not using ESM rates, this input must be tied to 0. **Synchronous To:** None **Active State:** N/A
phy_rxX_cntx_sel_esm25[7:0]	I	Per-lane Rx context select for esm 25 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_cntx_sel description. ■ This input must be controlled using a programmable register. ■ If not using ESM rates, this input must be tied to 0. **Synchronous To:** None **Active State:** N/A
phy_rxX_cntx_sel_g1[7:0]	I	Per-lane Rx context select for pcie gen1 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_cntx_sel description. ■ This input must be controlled using a programmable register. **Synchronous To:** None **Active State:** High

Port Name	I/O	Description
phy_rxX_cntx_sel_g2[7:0]	I	Per-lane Rx context select for pcie gen2 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_cntx_sel description. ■ This input must be controlled using a programmable register. **Synchronous To:** None **Active State:** High
phy_rxX_cntx_sel_g3[7:0]	I	Per-lane Rx context select for pcie gen3 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_cntx_sel description. ■ This input must be controlled using a programmable register. **Synchronous To:** None **Active State:** High
phy_rxX_cntx_sel_g4[7:0]	I	Per-lane Rx context select for pcie gen4 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_cntx_sel description. ■ This input must be controlled using a programmable register. **Synchronous To:** None **Active State:** High
phy_rxX_cntx_sel_g5[7:0]	I	Per-lane Rx context select for pcie gen5 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, rxX_cntx_sel description. ■ This input must be controlled using a programmable register. **Synchronous To:** None **Active State:** High
phy_txX_cntx_sel_esm20[7:0]	I	Per-lane Tx context select for esm 20 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_cntx_sel description. ■ This input must be controlled using a programmable register. ■ If not using ESM rates, this input must be tied to 0. **Synchronous To:** None **Active State:** N/A
phy_txX_cntx_sel_esm25[7:0]	I	Per-lane Tx context select for esm 25 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_cntx_sel description. ■ This input must be controlled using a programmable register. ■ If not using ESM rates, this input must be tied to 0. **Synchronous To:** None **Active State:** N/A

Port Name	I/O	Description
phy_txX_cntx_sel_g1[7:0]	I	Per-lane Tx context select for pcie gen1 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_cntx_sel description. ■ This input must be controlled using a programmable register. **Synchronous To:** None **Active State:** High
phy_txX_cntx_sel_g2[7:0]	I	Per-lane Tx context select for pcie gen2 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_cntx_sel description. ■ This input must be controlled using a programmable register. **Synchronous To:** None **Active State:** High
phy_txX_cntx_sel_g3[7:0]	I	Per-lane Tx context select for pcie gen3 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_cntx_sel description. ■ This input must be controlled using a programmable register. **Synchronous To:** None **Active State:** High
phy_txX_cntx_sel_g4[7:0]	I	Per-lane Tx context select for pcie gen4 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_cntx_sel description. ■ This input must be controlled using a programmable register. **Synchronous To:** None **Active State:** High
phy_txX_cntx_sel_g5[7:0]	I	Per-lane Tx context select for pcie gen5 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, txX_cntx_sel description. ■ This input must be controlled using a programmable register. **Synchronous To:** None **Active State:** High
phy_tx_cmnX_cntx_sel_esm20[7:0]	I	Per-lane TX cmn context select for esm20 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, tx_cmnX_cntx_sel description. ■ This input must be controlled using a programmable register. ■ If not using ESM rates, this input must be tied to 0. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
phy_tx_cmnX_cntx_sel_esm25[7:0]	I	Per-lane TX cmn context select for esm25 rate. ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, tx_cmnX_cntx_sel description. ■ This input must be controlled using a programmable register. ■ If not using ESM rates, this input must be tied to 0. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phy_tx_cmnX_cntx_sel_g1[7:0]	I	Per-lane TX cmn context select for pcie gen1 rate ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, tx_cmnX_cntx_sel description. ■ This input must be controlled using a programmable register. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phy_tx_cmnX_cntx_sel_g2[7:0]	I	Per-lane TX cmn context select for pcie gen2 rate ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, tx_cmnX_cntx_sel description. ■ This input must be controlled using a programmable register. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phy_tx_cmnX_cntx_sel_g3[7:0]	I	Per-lane TX cmn context select for pcie gen3 rate ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, tx_cmnX_cntx_sel description. ■ This input must be controlled using a programmable register. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High
phy_tx_cmnX_cntx_sel_g4[7:0]	I	Per-lane TX cmn context select for pcie gen4 rate ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, tx_cmnX_cntx_sel description. ■ This input must be controlled using a programmable register. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

Port Name	I/O	Description
phy_tx_cmnX_cntx_sel_g5[7:0]	I	Per-lane TX cmn context select for pcie gen5 rate ■ Refer to the DesignWare Cores PHY databook, "Signal Descriptions" chapter, tx_cmnX_cntx_sel description. ■ This input must be controlled using a programmable register. **Voltage Range:**0-vpdig **Synchronous To:** Asynchronous **Active State:** High

4.25 Context Select Interface Signals (top to/from phy) (for X = 0; X <= Nlanes-1)

phy_laneX_cntx_en -

Table 4-25 Context Select Interface Signals (top to/from phy) (for X = 0; X <= Nlanes-1)

Port Name	I/O	Description
phy_laneX_cntx_en	I	Per-lane context enable input (lane0_cntx_en). ■ Change this signal only during phy_reset. ■ This input must be controlled using a programmable register. **Voltage Range:**0-vpdig **Synchronous To:** None **Active State:** High

4.26 Scan Clock Interface Signals (top to/from pcs)

Table 4-26 Scan Clock Interface Signals (top to/from pcs)

Port Name	I/O	Description
pcs_scan_max_pclk	I	PCS ASST scan clock for MAX_PCLK. ■ Scan clock input for laneX_max_pclk functional clock domain. ■ This signal exists if RTL macro DWC_PIPE_VER_5 is defined **Synchronous To:** None **Active State:** N/A
pcs_scan_pipe_rx_clk	I	Scan for PIPE RX clock of SerDes PIPE. ■ Scan clock input for the SerDes PIPE RxClk (pipe_rxX_clk) domain. ■ This signal exists if RTL macro DWC_PCIE_PIPE_SERDES is defined. **Synchronous To:** None **Active State:** N/A

4.27 SerDes PIPE Signals (top to/from pcs) (for X = 0; X <= Nlanes-1)

Table 4-27 SerDes PIPE Signals (top to/from pcs) (for X = 0; X <= Nlanes-1)

Port Name	I/O	Description
pipe_laneX_rxready	I	Indicates symbol lock and block alignment have been achieved. ■ This signal must be asserted and de-asserted as it is RxValid of Original PIPE. ■ The timer inside PHY starts when CDR is locked and stops when pipe_laneX_rxready is asserted. The recalibration is performed after 6 us timeout. ■ This signal exists if RTL macro DWC_PCIE_PIPE_SERDES is defined. **Synchronous To:** pipe_laneX_in_pclk **Active State:** High
pipe_laneX_serdes_mode	I	SerDesPIPE mode. This signal must be tied to 1 when DWC_PCIE_PIPE_SERDES is enabled. If this signal needs to be switched, contact Synopsys. (synthesis option: serdes_with_orig_mode). ■ 0: Original PIPE mode ■ 1: SerDes PIPE mode This signal exists if the RTL macro DWC_PCIE_PIPE_SERDES is defined. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_laneX_turnoff	I	Turn off a lane when this value is set to 1. ■ pipe_tx_serdes_elecidle must be asserted when this signal is 1. ■ This signal must be controlled if the controller does not control the powerdown signals per lane and needs to turn off each lane. ■ This signal is available only when operating in the SerDes PIPE mode. ■ This value must be 0 when operating in the Original PIPE mode or the Turn-Off is controlled by the powerdown control per lane. ■ See section "PIPE Lane Disable". ■ This signal exists if RTL macro DWC_PCIE_PIPE_SERDES is defined. **Synchronous To:** pipe_laneX_in_pclk **Active State:** High

Port Name	I/O	Description
pipe_rxX_clk	O	RxClk for laneX. ■ Indicates the recovered clock. ■ This signal exists if RTL macro DWC_PCIE_PIPE_SERDES is defined. **Synchronous To:** Asynchronous **Active State:** N/A
pipe_rxX_if_width[1:0]	I	RxWidth for laneX. ■ This value must be always same as pipe_laneX_if_width (used for TxWidth). ■ This signal exists if RTL macro DWC_PCIE_PIPE_SERDES is defined. ■ This input signal must be 0 when operating in OriginalPIPE. **Synchronous To:** pipe_laneX_pclk **Active State:** N/A
pipe_rxX_serdes_data[39:0]	O	RxData for laneX. ■ Indicates the parallel data output bus. ■ This signal exists if RTL macro DWC_PCIE_PIPE_SERDES is defined. ■ The bit width is 80 bits if DWC_UPCS_PIPE-WIDTH_4S_4S_8S_8S_8S or DWC_UPCS_PIPE-WIDTH_4S_4S_4S_8S_8S is defined. **Synchronous To:** pipe_rxX_clk **Active State:** N/A
pipe_rxX_serdes_elecidle	O	RxElecIdle for laneX. ■ Indicates receiver detection of an electrical idle. ■ This signal exists if RTL macro DWC_PCIE_PIPE_SERDES is defined. **Synchronous To:** Asynchronous **Active State:** High
pipe_rxX_serdes_valid	O	RxValid for laneX. ■ Indicates the recovered clock is stable. ■ This signal exists if RTL macro DWC_PCIE_PIPE_SERDES is defined. **Synchronous To:** pipe_rxX_clk **Active State:** N/A

Port Name	I/O	Description
pipe_txX_serdes_data[39:0]	I	TxData for laneX. ■ Indicates the parallel data input bus. ■ This signal exists if RTL macro DWC_PCIE_PIPE_SERDES is defined. ■ This input signal must be 0 when operating in OriginalPIPE. ■ The bit width is 80 bits if DWC_UPCS_PIPE-WIDTH_4S_4S_8S_8S_8S or DWC_UPCS_PIPE-WIDTH_4S_4S_4S_8S_8S is defined. **Synchronous To:** pipe_laneX_in_pclk **Active State:** N/A
pipe_txX_serdes_datavalid	I	TxValid for laneX. ■ Indicates whether PHY will use TxData or not. ■ This signal exists if RTL macro DWC_PCIE_PIPE_SERDES is defined. ■ This input signal must be 0 when operating in OriginalPIPE. **Synchronous To:** pipe_laneX_in_pclk **Active State:** N/A
pipe_txX_serdes_elecidle[3:0]	I	TxElecIdle for laneX. ■ Indicates the Tx electrical idle for laneX. ■ This signal exists if RTL macro DWC_PCIE_PIPE_SERDES is defined. ■ This input signal must be 0xF when operating in OriginalPIPE. **Synchronous To:** pipe_laneX_in_pclk **Active State:** N/A

5
Usage Model

This chapter includes the following sections:

- "Power-Up Sequence" on page 353
- "Lane Reset Sequence" on page 355
- "Clocks" on page 355
- "Datapath" on page 368
- "Data Throttling Control for Original PIPE" on page 371
- "Data Rate Change" on page 372
- "Power State Transitions" on page 372
- "Per-Lane ropll_clk Source Select" on page 386
- "Clock Architecture and Link Selection" on page 389
- "Enabling SRIS" on page 390
- "Estore Configuration" on page 391
- "Recalibration" on page 391
- "RX Adaptation" on page 392
- "PHY Bifurcation Support" on page 392
- "PHY Aggregation Support" on page 393
- "Lane Disabling and Up/Down-Configuration" on page 394
- "PHY Configuration Overrides" on page 395
- "Transmitter Equalization Settings" on page 396
- "Lane Margining at Receiver" on page 397
- "ESM Rate Change Sequence" on page 399
- "EIEOS Format" on page 401
- "Message Bus Support" on page 402
- "SATA BIST Loopback Mode" on page 404
- "SATA: COM Align and Misalign Mechanism" on page 404
- "SATA - Disconnect Detection" on page 405
- "SATA - Nominal Empty Buffer Mode Operation" on page 405
- "Power Gating Support" on page 406

■ "Tie-Off Values for Unused Lanes" on page 408

5.1 Power-Up Sequence

Figure 5-1 on page 354 shows the Multi-Protocol 32G PCS power-up reset sequence along with the power supply ramp-up. The figure shows signals on the PIPE wrapper module encapsulating the PCS and the PHY. (For a description of the PHY signals, refer to DesignWare Cores Multi-Protocol 32G PHY x4 for TSMC 7FF databook, “Signal Descriptions” chapter.)

The PCS and the Multi-Protocol 32G PHY x4 for TSMC 7FF support power gating. The Figure 5-1 also shows the state of the power switch enable signals during the reset sequence.

VPH, VP, and VPDIG are power supply inputs of the PMA macro in the PHY. VDD_SOC is the always-ON core supply for the Raw PCS (in the PHY) and the PCS.

Note VPH, VP, and VPDIG power supplies in the PCS map to vph, vp, and vpdig in the PHY, respectively.

VDD_SOCSW refers to the switched power supply for the Raw PCS and the PCS, which is controlled by the power enable signals, upcs_pwr_en (UPCS) and phyN_pcs_pwr_en/phyN_pma_pwr_en (PHY). For information about implementing power gating in the PHY and PCS, see “Power Gating Support” on page 406.

To support debug of the design, you might need to update control registers in the Multi-Protocol 32G PHY x4 for TSMC 7FF before running any operation using the PCS and the Multi-Protocol 32G PHY x4 for TSMC 7FF. You can program control registers in the PHY (using the PHY’s CR parallel or JTAG interface) by de-asserting the phy_reset input while keeping the lane resets (pipe_laneX_reset_n) asserted.

Note Before accessing any registers in the PHY, a register reset time of 10 phyN_ref_dig_clk cycles is required.

Figure 5-1 Power-up Reset Timing

5.2 Lane Reset Sequence

Each lane can be independently reset using the pipe_laneX_reset_n signal. In addition, for a multi-lane link, a common PIPE reset signal from the MAC can be connected to all lanes on the PCS operating as part of the same link. Figure 5-1 on page 354 shows the PIPE interface signals involved in lane reset assertion and exit.

5.3 Clocks

For details on clock architecture and frequencies, see "Clocks" on page 32. The clock outputs from the PCS and the PHYs are dependent on both PIPE signals and some side-band signals.

5.3.1 PCS Output Clocks

The following tables provide the conditions under which the PCS clock outputs are available.

Table 5-1 PCS Clock Dependency (PIPE 4.2)

I/O	Signals	Conditions: X=Don't care, D=Disabled, E=Enabled								
Inputs	Reference clock	D	E	E	E	E or D	E	E	E	E
	phy_reset	X	1 (reset)	0	0	0	0	0	0	0
	pipe_laneX_reset_n	X	X	0 (reset)	1	1	1	1	1	1
	ext_pclk_req	X	X	X	1	0	0	0	0	0
	pipe_laneX_clkreq_n (pipe_42)	X	X	X	X	1	0	0	0	0
	pipe_laneX_powerdown	X	(controller reset value) PCIe: P1 SATA: Slumber or more active states	(controller reset value) PCIe: P1 SATA: Slumber or more active states	X	PCIe: P2 SATA: DevSleep	PCIe: P2 SATA: ■ Slumber ■ DevSleep	PCIe: ■ P0 ■ P0s ■ P1 SATA: ■ Active ■ Partial	PCIe: ■ P0 ■ P0s ■ P1 SATA: ■ Active ■ Partial	PCIe: ■ P0 ■ P0s ■ P1 SATA: ■ Active ■ Partial
	LaneX (Disabled/Enabled)	X	X	X	X	X	X	E	D	D
	upcs_pipe_config[0]	X	X	X	X	X	X	X	0	1
Outputs	pipe_laneX_pclk pipe_laneX_max_pclk	D	D	D	E	D	D	E	D	E
	pcs_laneX_ref_clk_en (to PIPE_GLUE)	X	1	1	1	0	1	1	1	1

Note

- LaneX is disabled when pipe_txX_elecidle = 1 and pipe_txX_compliance = 1.
- Reference clock = phyN_ref_[alt, pad]_clk_[p,m]
- When the DWC_UPCS_PERLINK_CLK_ARCH parameter is defined, the MAC must drive the same value for each laneX input for all lanes belonging to the same link.
- pcs_laneX_ref_clk_en is a per-lane signal which is aggregated to create phyN_ref_clk_en. When any lane associated with phyN asserts pcs_laneX_ref_clk_en, phyN_ref_clk_en will assert. When all lanes associated with phyN are de-asserted, pcs_laneX_ref_clk_en and phyN_ref_clk_en will de-assert.
- When ext_pclk_req is 1, the PCS asserts pcs_laneX_ref_clk_en to 1 to keep ref_clk and pipe_laneX_pclk alive where ref_clk and pclk are normally removed.

Table 5-2 PCS Clock Dependency (PIPE 4.3)

I/O	Signals	Conditions: X=Don't care, D=Disabled, E=Enabled								
Inputs	Reference clock	D	E	E	E	E or D	E	E	E	E
	phy_reset	X	1 (reset)	0	0	0	0	0	0	0
	pipe_laneX_reset_n	X	X	0 (reset)	1	1	1	1	1	1
	ext_pclk_req	X	X	X	1	0	0	0	0	0
	pipe_laneX_powerdown	X	(controller reset value) PCIe: P1 SATA: Slumber or more active states	(controller reset value) PCIe: P1 SATA: Slumber or more active states	X	PCIe: ■ P1.CPM ■ P1.1 (pipe43) ■ P1.2 (pipe43) ■ P2.CPM SATA: N/A	PCIe: P2 SATA: ■ Slumber ■ DevSleep	PCIe: ■ P0 ■ P0s ■ P1 SATA: ActivePartial	PCIe: ■ P0 ■ P0s ■ P1 SATA: ■ Active ■ Partial	PCIe: ■ P0 ■ P0s ■ P1 SATA: ■ Active ■ Partial
	LaneX (Disabled/Enabled)	X	X	X	X	X	X	E	D	D
	upcs_pipe_config[0]	X	X	X	X	X	X	X	0	1

I/O	Signals	Conditions: X=Don't care, D=Disabled, E=Enabled								
Outputs	pipe_laneX_pclk pipe_laneX_max_pclk	D	D	D	E	D	D	E	D	E
	pcs_laneX_ref_clk_en (to PIPE_GLUE)	X	1	1	1	0	1	1	1	1

Note

- LaneX is disabled when pipe_txX_elecidle = 1 and pipe_txX_compliance = 1.
- Reference clock = phyN_ref_[alt, pad]_clk_[p,m]
- When the DWC_UPCS_PERLINK_CLK_ARCH parameter is defined, the MAC must drive each laneX input for all lanes that belong to the same link to the same identical value in each lane.
- pcs_laneX_ref_clk_en is a per-lane signal which is aggregated to create phyN_ref_clk_en. When any lane associated with phyN asserts pcs_laneX_ref_clk_en, phyN_ref_clk_en will assert. When all lanes associated with phyN are de-asserted, pcs_laneX_ref_clk_en, phyN_ref_clk_en will de-assert.
- When ext_pclk_req is 1, the PCS asserts pcs_laneX_ref_clk_en to 1 to keep ref_clk and pipe_laneX_pclk and pipe_laneX_in_pclk (when DWC_PIPE_VER_5 is defined) alive where ref_clk and pclk is normally removed.

Table 5-3 PCS Clock Dependency (PIPE 4.4 or later)

I/O	Signals	Conditions: X=Don't care, D=Disabled, E=Enabled											
Inputs	Reference clock	D	E	E or D	E	E	E	E or D	E or D	E	E	E	E
	phy_reset	X	1 (reset)	0	0	0	0	0	0	0	0	0	0
	pipe_laneX_reset_n	X	X	0 (reset)	0 (reset)	1	1	1	1	1	1	1	1
	ext_pclk_req (pclk as PHY output) pipe_laneX_maxpclkreq_n (pclk as PHY input)	X	X	0 1	X	ext_pclk_req=1 or pipe_laneX_maxpclkreq_n=0	0 1	0 1	0 1	0 1	0 1	0 1	0 1
	pipe_laneX_powerdown	X	(controller reset value) PCIe: P1 SATA: Slumber or more active states	(controller reset value) PCIe: P1 SATA: Slumber	(controller reset value) PCIe: P1 SATA: Slumber or more active states	X	PCIe: ■ P1.CPM ■ P2.CPM ■ P2NOBEACON SATA: P2.PG	PCIe: ■ P1.CPM ■ P2.CPM ■ P2NOBEACON SATA: P2.PG	SATA: ■ **DevSleep** X	PCIe: P2SATA: ■ Slumber ■ DevSleep	PCIe: ■ P0 ■ P0s ■ P1 SATA: ■ Active-Partial	PCIe: ■ P0 ■ P0s ■ P1 SATA: ■ Active ■ Partial	PCIe: ■ P0 ■ P0s ■ P1 SATA: ■ Active ■ Partial
	pipe_rxX_disable	x	0	0	0	0	pipe_rxX_disable has been changed from 1 to 0 in P1.CPM	pipe_rxX_disable has not been changed from 1 to 0 in P1.CPM	0	0	0	0	0
	LaneX (Disabled/Enabled)	X	X	X	X	X	X	X	X	X	E	D	D
	upcs_pipe_config[0]	X	X	X	X	X	X	X	X	X	X	0	1

I/O	Signals	Conditions: X=Don't care, D=Disabled, E=Enabled											
Outputs	pipe_laneX_pclk pipe_laneX_max_pclk	D	D	D	D	E ***a)**	D	D	D	D	E ***a)**	D	E ***a)**
	pcs_laneX_ref_clk_en (to PIPE_GLUE)	X	1	0	1	1	1	0	0	1	1	1	1

Note

- LaneX is disabled when pipe_txX_elecidle = 1 and pipe_txX_compliance = 1.
- Reference clock = phyN_ref_[alt, pad]_clk_[p,m]
- When the DWC_UPCS_PERLINK_CLK_ARCH parameter is defined, the MAC must drive the same value for each laneX input for all lanes belonging to the same link.
- In “PCLK as PHY Input Mode,” PCLK is an input to the UPCS as pipe_laneX_in_pclk; the controller can decide to turn this mode on or off but must align with this table.
- pcs_laneX_ref_clk_en is a per-lane signal which is aggregated to create phyN_ref_clk_en. When any lane associated with phyN asserts pcs_laneX_ref_clk_en, phyN_ref_clk_en will assert. When all lanes associated with phyN are de-asserted, pcs_laneX_ref_clk_en, phyN_ref_clk_en will de-assert.
- When ext_pclk_req is 1, the PCS asserts pcs_laneX_ref_clk_en to 1 to keep ref_clk and pipe_laneX_pclk and pipe_laneX_in_pclk (when DWC_PIPE_VER_5 is defined) alive where ref_clk and pclk is normally removed.
- When PIPE_4_4_1 is defined and pipe_laneX_powerdown=P1.CPM, pipe_txX_disable=1, and pipe_rxX_disable=1 followed by de-assertion of pipe_rxX_disable, pcs_laneX_ref_clk_en is asserted.

Clock availability is indicated on the PIPE interface using the pipe_laneX_phystatus signal. When the pipe_laneX_pclk/pipe_laneX_max_pclk clocks transition from Disabled (D) state to Enabled (E) state, these clocks are available before the de-assertion (0) of pipe_laneX_phystatus if pipe_laneX_phystatus is already asserted (1) either in the Clock Disabled state or before the assertion (1) of pipe_laneX_phystatus if pipe_laneX_phystatus is de-asserted (0) in the Clock Disabled state.

In “PCLK as PHY Output Mode”, note that the PCS clock outputs (pipe_laneX_pclk and pipe_laneX_max_pclk) can be used by the MAC to sample or launch the synchronous signals on the PIPE interface for the lane. This option requires independent clock architectures in the MAC on a per-lane basis. To ease this implementation in the MAC, the PCS clock outputs for each lane are designed to be phase-synchronous to other lanes configured to be part of the same link (for information about lane configuration, see “PHY Configuration Overrides” on page 395). Therefore, the PCS clocks from one lane can be used to sample or launch the synchronous signals on the PIPE interface for other lanes that are part of the same link. If the lane’s Down-Configure feature is required, it is recommended that you set the upcs_pipe_config[0] signal to 1. Doing so enables the PCS clocks for lane X to

be enabled during the following states even when lane X is disabled (for clock availability conditions, see Table 5-1) as a result of down-configuration:

- PCIe: P0, P0s, and P1
- SATA: Active and Partial

When upcs_pipe_config[0]=0 and the DWC_UPCS_PERLINK_CLK_ARCH parameter is not defined, the PIPE interface signals are ignored by the PCS, and the PCS clock from lane X might get disabled for the down-configured lane. Consequently, the MAC cannot use the clock from lane X as a common clock source to sample/launch the synchronous PIPE interface signals for other active lanes on the same link.

When DWC_UPCS_PERLINK_CLK_ARCH is defined, set upcs_pipe_config[0] to 1. Setting this signal allows per-link clock architecture to support lane reversal with broken lanes, which means that physical lane0 is not detected. PCS clocks for laneX are enabled during the following states, even when laneX is disabled due to down-configuration.

- PCIe: P0, P0s, and P1
- SATA: Active and Partial

5.3.2 PHY Output Clocks

This section describes the output clocks from the PHY.

5.3.2.1 MPLL Output Clocks

In addition to the PCS output clocks, pipe_laneX_pclk (only for "PCLK as PHY Output Mode")/pipe_laneX_max_pclk, the PHY's MPLL output clocks can also be used by logic external to the PCS and can be kept alive irrespective of the state of the per-lane PIPE control signals. These clock outputs are tapped from the PHY interface as shown in the PIPE wrapper included in the package release (./macro/rtl/dwc_*_upcs_*_pipe.v).

Table 5-4 provides the conditions under which the MPLL clocks are available. As indicated in the table, the phyN_mpll[a,b]_force_en input forces the MPLL clocks to be enabled—assuming phy_reset is de-asserted and a reference clock input to the PHY is available.

Table 5-4 PHY MPLL Output Clock Dependency

I/O	Signal	Conditions X = Do not care, D = Disabled, E = Enabled									
Inputs	phyN_mpll[a,b]_force_en	X	X	X	1	0	0	0	0	0	0
	phy_reset	X	1	0	0	0	0	0	0	0	0
	pipe_laneX_reset_n	X	X	0	1	1	1	1	1	1	1
	Reference clock	D	E	**E**	E	E	E	E	E	E	E
	ext_pclk_req(pclk as PHY output) pipe_laneX_maxpclkreq_n(pclk as PHY input)	X	X	X	X	For the remaining conditions, see Table 5-1 on page 356. Because MPLL clocks are required to generate PCS clock outputs, all conditions from Table 5-1 and the corresponding states of pipe_laneX_pclk/ pipe_laneX_max_pclk are also applicable to phyN_mpll[a,b,]_*_clk with the following additional requirements: ■ phy_txX_ropll_word_clk signals are Enabled (E) when **any** lane Xs using PLL clocks from PHY*N*-LaneX meet the conditions required to enable pipe_laneX_pclk/pipe_laneX-_max_pclk as specified in Table 5-1. ■ phy_txX_ropll_word_clk signals are Disabled (D) when **all** the lane Xs using PLL clocks from PHY*N*-LaneX meet the conditions required to disable pipe_laneX-_pclk/pipe_laneX_max_pclk as specified in Table 5-1.						
	upcs_pipe_config[0]	X	X	X	X						
	LaneX (Disabled/Enabled)	X	X	X	X						
	pipe_laneX_powerdown	X	(controller reset value) PCIe: P1	(controller reset value) PCIe: P1	X						
Output Clocks	phy_txX_ropll_word_clk	D	D	D	E						

Note
- LaneX is disabled when pipe_txX_elecidle = 1 and pipe_txX_compliance = 1. (original pipe architecture) or pipe_lane_turnoff (serdes pipe architecture)
- Reference clock = phyN_ref_[alt0,alt1, pad]_clk_[p,m]

5.3.2.2 Reference Clock Outputs

The reference clock outputs from the PHY, phyN_ref_dig_clk and phyN_ref_dig_fr_clk, can be used by logic external to the PCS.

When any pcs_laneX_ref_clk_en in Table 5-1 is 1b for laneX using MPLLs clocks from PHYN, the phyN_ref_clk_en signals are 1b.

When all pcs_laneX_ref_clk_en in Table 5-1 are 0b for laneX using MPLLs clocks from PHYN, the phyN_ref_clk_en signals are 0b.

Refer to the PHY databook for conditions for enabling phyN_ref_dig_clk and phyN_ref_dig_fr_clk.

5.3.3 Reference Clock Input

The PCS generates the clock configuration settings for the PHY based on the frequency of the reference clock supplied to the PHYs. The reference clock frequency supplied to the PHY must be defined as a compile macro to the PCS. The following table lists the supported macros corresponding to various reference clock frequencies.

Table 5-5 PCIe Reference Clock Macros

Reference clock Frequency Macro (`define)	Reference Clock Frequency (MHz)
Default	100

Table 5-6 SATA Reference Clock Macros

Reference clock Frequency Macro (`define)	Reference Clock Frequency (MHz)
REF_24M	24
REF_25M	25
Default	100

The PCS supports low power states where the reference clock input to the PHY (phyN_ref_[alt0/1, pad]_clk_[p,m]) can be turned off; for example, P1.CPM, P1.1, P1.2, P2.CPM. To support this feature, handshake signals are provided on the PCS to send and acknowledge requests to gate the reference clock. Gate the reference clock to the PHY only if all PCS lanes connected to the PHY are in a state where the reference clock can be turned off.

The following figure shows the reference clock topology for a single PHY and one or more (bifurcation) links on the PCS.

Figure 5-2 Reference Clock Topology for One PHY and One or More Links

The following figure shows the reference clock topology for two or more PHYs and one link on the PCS.

Figure 5-3 Reference Clock Topology for Two or More PHYs and One Link

The generation of the common CLKREQ# signal is the same as in Figure 5-2.

The following figure shows the reference clock topology for two or more PHYs and two or more links on the PCS. In this figure, it is assumed that all lanes for link 0 are connected to PHY0, and all lanes for link 1 are connected to PHY1.

Figure 5-4 Reference Clock Topology for Two or More Links and Two or More PHYs

5.3.4 PLL Mode Selection

The PHY has two independent PLLs (MPLLA and MPLLB) and a PLL per lane. Refer to the PHY databook for specific architecture and details of these PLLs. Direct clocking or Bypass mode for PCIE provides clocks directly from the MPLLA and MPLLB PLLs. Cascaded clocking, which uses MPLLA and MPLLB configured in series with the per-lane PLL, can be used for advanced PCIe bifurcation cases and for other protocols.

A single PHY supports one or two independent reference clocks.

- If a single PHY is bifurcated into two or more links and a single reference clock is used as source for all links, **Direct Clocking** mode can be used.
- If a single PHY is bifurcated into two links and each link uses a separate reference clock, **Cascaded Clocking** must be used.

Table 5-7 shows some of the use-cases supported by the PCS. For additional assistance, contact Synopsys Customer Support.

Table 5-7 PLL Modes

PLL Mode in Databook	pipe_laneX_lanepll_bypass_mode[4:0]	pipe_laneX_cascaded_mpllb_sel
PCIe CC: MPLLB Cascaded Clocking (Common Clock)	5’h0	1’b1
CCIX: MPLLA Cascaded Clocking)	5’h0	1’b1
PCIe SRIS/SRNS: MPLLB Cascaded Clocking	5’h0	1’b1
PCIe CC: MPLLA+B Direct Clocking (Common Clock)	5’h1F	1’b0
CXL: MPLLA+B Direct Clocking Common Clock	5’h1F	1’b0
PCIe SRIS/SRNS: MPLLA+B Direct Clocking	5’h1F	1’b0

5.4 Datapath

The datapath is active when the lane is in a P0 state (pipe_laneX_powerdown[3:0]=4'h0), pipe_txX_elecidle (Original PIPE), pipe_txX_serdes_elecidle (SerDes PIPE) is de-asserted, and pipe_rxX_valid (Original PIPE), pipe_rxX_serdes_valid (SerDes PIPE) is asserted. Table 5-8 provides Original PIPE Architecture TX datapath latencies for the PCIe PCS, and Table 5-10 provides Original PIPE Architecture RX datapath latencies for the PCIe PCS in the various supported modes. In all cases, the latency is shown for pipe_laneX_if_width[1:0] = 2'b01 (16-bit mode). For SerDes PIPE Architecture, the latency is only 1-clock cycle for RX and 1 clock cycle for TX. as shown in the following tables.

The RX datapath may introduce skew on incoming data relative to other lanes due to CDC between the recovered clock and pclk. Skew may also be introduced because of skip add or remove. The maximum skew added is 30 UI at gen1/2 rates and 24 UI at gen3/4/5 rates.

Table 5-8 Original PIPE Architecture TX Datapath Latency for PCIe PCS

PCIe UPCS-only: TX Data Path Latency (in ns) 16-bit PIPE interface, 16-bit UPCS Datapath Gen1/2/3/4 32-bit PIPE interface, 32-bit UPCS Datapath Gen5	
Rate	**UPCS**
PCIe Gen1	24.0
PCIe Gen2	12.0
PCIe Gen3	10.0
PCIe Gen4	5.0
PCIe Gen5	5.0

Note TX and RX datapath latencies are for PCLK as PHY output mode.

Table 5-9 SerDes PIPE Architecture TX Datapath Latency for PCIe PCS

PCIe UPCS-only: TX Data Path Latency (in ns) 16-bit PIPE interface, 16-bit UPCS Datapath Gen1/2/3/4 32-bit PIPE interface, 32-bit UPCS Datapath Gen5	
Rate	**UPCS**
PCIe Gen1	8
PCIe Gen2	4
PCIe Gen3	4
PCIe Gen4	2
PCIe Gen5	1

Table 5-10 Original PIPE Architecture RX Datapath Latency for PCIe PCS

PCIe UPCS-only: RX Data Path Latency (in ns) 16-bit PIPE interface, 16-bit UPCS Datapath Gen1/2/3/4 32-bit PIPE interface, 32-bit UPCS Datapath Gen5						
4096 Byte Payload	**pipe_rxX_es_mode = HalfFull**		**pipe_rxX_es_mode = NomEmpty**		**pipe_rxX_es_mode = comRefClk**	
Data Rate	**Avg Latency**	**Max PPM Latency**	**Avg Latency**	**Max PPM Latency**	**Avg Latency**	**Max PPM Latency**
PCIe Gen1, CMN	155.2	187.2	65.2	97.2	96.0	96.0
PCIe Gen2, CMN	76.7	92.7	28.7	44.7	49.3	49.3
PCIe Gen3, CMN	43.4	57.4	21.9	35.9	40.3	40.3
PCIe Gen4, CMN	22.3	29.3	10.8	17.8	14.9	14.9
PCIe Gen5, CMN	14.8	18.3	11.2	14.7	13.2	13.2
PCIe Gen1, SRNS	155.2	187.2	65.2	97.2	N/A	N/A
PCIe Gen2, SRNS	76.7	92.7	28.7	44.7	N/A	N/A
PCIe Gen3, SRNS	43.4	57.4	21.9	35.9	N/A	N/A
PCIe Gen4, SRNS	22.3	29.3	10.8	17.8	N/A	N/A
PCIe Gen5, SRNS	14.8	18.3	11.2	14.7	N/A	N/A
PCIe Gen1, SRIS	223.6	423.6	65.2	265.2	N/A	N/A
PCIe Gen2, SRIS	107.8	207.8	28.7	128.7	N/A	N/A
PCIe Gen3, SRIS	59.4	113.4	21.9	75.9	N/A	N/A
PCIe Gen4, SRIS	30.2	57.2	10.8	37.8	N/A	N/A
PCIe Gen5, SRIS	18.8	32.3	11.2	24.7	N/A	N/A

Note

- TX and RX datapath latencies are for PCLK as PHY output mode.
- Average latency is the latency with no PPM offset.
- Maximum latency is obtained when remote refclk is faster than local ref clk.
- Minimum latency (not shown) is obtained when remote refclk is slower than local ref clk (Avg Latency - (Max-Avg)).

Table 5-11 SerDes PIPE Architecture RX Datapath Latency for PCIe PCS

PCIe UPCS-only: RX Data Path Latency (in ns) 16-bit PIPE interface, 16-bit UPCS Datapath Gen1/2/3/4 32-bit PIPE interface, 32-bit UPCS Datapath Gen5	
Rate	**UPCS**
PCIe Gen1	8
PCIe Gen2	4
PCIe Gen3	4
PCIe Gen4	2
PCIe Gen5	1

5.4.1 Loopback Functions

Note Refer to the "Loopback Functions" and "BERT" sections in the PHY databook.

- **SATA**: Active, Partial, Slumber, and DevSleep

  The transition to and from the Active, Partial, Slumber, and DevSleep power states are indicated using the pcs_laneX_phystatus output from the PCS.

5.5 Data Throttling Control for Original PIPE

The PCS supports Data Throttling.

- If the DWC_UPCS_PIPEWIDTH_1S_2S_2S_2S_PCLK1G parameter is not defined, Data Throttling is not enabled. The pipe_txX_datavalid and pipe_rxX_datavalid signals are used only for inserting dead cycle every N Blocks for 128b/130b encoding.
- If the DWC_UPCS_PIPEWIDTH_1S_2S_2S_2S_PCLK1G parameter is defined, Data Throttling is enabled. In addition to inserting a dead cycle described previously, the pipe_txX_datavalid and pipe_rxX_datavalid signals are used to indicate a valid cycle of data-related PIPE signals.

Figure 5-5 TX Data Throttling in 128b/130b Encoding

Figure 5-6 RX Data Throttling in 128b/130b Encoding

Follow these requirements when using Data Throttling mode:

- Regardless of the value of the pipe_txX_datavalid signal, pipe_txX_compliance and pipe_txX_elecidle cannot be 1b at the same PCLK clock cycle except for signaling turn-off.
- Regardless of the value of the pipe_txX_datavalid signal, pipe_txX_compliance and pipe_txX_elecidle must be kept 1b for signaling turn-off.
- Regardless of the value of the pipe_txX_datavalid signal, pipe_txX_elecilde must be kept 0b while transmitting Data or Beacon.
- Regardless of the value of the pipe_txX_datavalid signal, pipe_txX_elecilde must be kept 1b while TX is in electrical idle.

5.6 Data Rate Change

Figure 5-7 PCLK as PHY Input Rate Change

5.7 Power State Transitions

The Multi-Protocol 32G PCS and Multi-Protocol 32G PHY x4 for TSMC 7FF support the following power states:

- **PCIe**: P0, P0s, P1, P2, P1.CPM, P1.1, P1.2, and P2.CPM

  The transition to and from the P0, P0s, P1, and P2 power states are indicated using the pipe_laneX_-phystatus output from the PCS. The transition to and from the other power states is indicated using either the pipe_laneX_phystatus output (PIPE 4.4.1/4.3) or the side-band signals (PIPE 4.2).

  PIPE 4.4.1 added support for the following transitions:

  - P0 -> P1.CPM
  - P1.CPM -> P0
  - P0 -> P2.CPM
  - P2.CPM -> P1
  - P1 -> P2.CPM
  - P2.CPM -> P0

5.7.1 Power State Transition Timing Diagrams

The timing diagram for PIPE interface signals matches the PIPE specification.

Note The power state and rate change transitions take effect after the TX datapath pipeline is internally cleared, enabling the MAC to change the TX controls in the same clock cycle as changing the power state and rate inputs.

The *PHY Interface For the PCI Express, SATA, and USB 3.1 Architectures*, version greater than or equal to 4.3 states: "Some PHY architectures may allow a speed change and a power state change to occur at the same time as a rate and/or width and/or PCLK rate change." The Multi-Protocol 32G PHY x4 for TSMC 7FF does not support speed change or PCLK rate change and a power state change simultaneously.

The following figures show an example waveform illustrating entry to and exit from a P0 power state along with a rate transition.

Figure 5-8 PCIe: P0 Entry and Exit Timing

The pipe_laneX_pclk signal is disabled (0) when in a P2 state, assuming other conditions do not enable the signal (for PCS clock dependencies, see Table 5-1). In this case, when transitioning into P2, the PCS is guaranteed to generate at least one rising clock edge on pipe_laneX_pclk after the assertion of pipe_laneX_phystatus.

Figure 5-9 PCLK as PHY Input: PCLK Off/RefClk Off Power State Entry (IN_PCLK SOURCE MAX_PCLK MODE)

Note If max_pclk is not used, max_pclkreq_n can be tied to 1'b1.

Figure 5-10 PCLK as PHY Input: PCLK Off/RefClk Off Power State Exit (IN_PCLK SOURCE MAX_PCLK MODE)

Note If max_pclk is not used, max_pclkreq_n can be tied to 1'b1.

Figure 5-11 PCLK as PHY Input: PCLK Off/RefClk On Power State Entry (IN_PCLK SOURCE MAX_PCLK MODE)

Note If max_pclk is not used, max_pclkreq_n can be tied to 1'b1.

Figure 5-12 PCLK as PHY Input: PCLK Off/RefClk On Power State Exit (IN_PCLK SOURCE MAX_PCLK MODE)

Note If max_pclk is not used, max_pclkreq_n can be tied to 1'b1.

Figure 5-13 PCLK as PHY Input: PCLK Off/RefClk Off Power State Entry (IN_PCLK SOURCE MPLL CLK MODE)

Note The phyN_mpll{a/b}_force_en signal must be updated from 1b to 0b only after all links which use the MPLL clock as the source of the PCLK input complete the PCLK off power-down state transition. Phystatus is a 1-shot pulse synchronous to PCLK. PCLK can only be disabled after the pulse.

Figure 5-14 PCLK as PHY Input: PCLK Off/RefClk Off Power State Exit (IN_PCLK SOURCE MPLL CLK MODE)

Note phyN_mpll{a/b}_force_en must be updated from 0b to 1b before any links which use the MPLL clock as the source of Input PCLK exit from PCLK off power-down state. PowerDown is synchronous to PCLK, and PCLK must be active before changing the signal.

Figure 5-15 PCLK as PHY Input: PCLK Off/RefClk On Power State Entry (IN_PCLK SOURCE MPLL CLK MODE)

Note The phyN_mpll{a/b}_force_en signal must be updated from 1b to 0b only after all links which use the MPLL clock as the source of the PCLK input complete the PCLK off power-down state transition. Phystatus is a 1-shot pulse synchronous to PCLK. PCLK can only be disabled after the pulse.

Figure 5-16 PCLK as PHY Input: PCLK Off/RefClk On Power State Exit (IN_PCLK SOURCE MPLL CLK MODE)

Note phyN_mpll{a/b}_force_en must be updated from 0b to 1b before any links which use the MPLL clock as the source of Input PCLK exit from PCLK off power-down state. PowerDown is synchronous to PCLK, and PCLK must be active before changing the signal.

5.7.1.1 PCIe L1 Substate Transitions

The PCS has two interfaces to support the L1 substate for PCIe4 applications: PIPE4.2 (using side-band signals) and PIPE4.3. The states on the PCS are referred to as P1.1 and P1.2. P1.2 is the lowest power state in which a particular lane can be placed.

For details on the generation of reference clock control CLKREQ# for PCIe applications, see "Reference Clock Input" on page 363.

For a MAC that supports PIPE 5.2/4.4.1/4.2, P1 substate entry and exit are supported using side-band signals. Figure 5-17 and Figure 5-18 show the timings for PIPE 4.2 P1.2 entry and exit using side-band signals. Figure 5-21 on page 385 shows the timing for PIPE 4.4.1 P1.1/P1.2 entry and exit using side-band signals. The timings for a P1.1 entry and exit are similar to a P1.2 entry and exit, except that in a P1.1 case, pipe_txX_disable remains at 0.

Figure 5-17 PCIe P1.2 Entry and Locally Initiated Exit Using Side-Band Signals (PIPE 4.2)

Figure 5-18 P1.2 Entry and Remotely Initiated Exit Using Side-Band Signals (PIPE 4.2)

Figure 5-19 and Figure 5-20 show the timings for P1.2 entry and exit using a PIPE 4.3 interface.

Figure 5-19 P1.2 Entry and Locally Initiated Exit Using PIPE 4.3 Interface

Figure 5-20 P1.2 Entry and Remotely Initiated Exit Using PIPE 4.3 Interface

The timings for a P1.1 entry and exit are similar to the P1.2 entry and exit, except that in a P1.1 case, pipe_laneX_powerdown transitions from P1.CPM to P1.1.

Note The timings in Figure 5-17 on page 381 through Figure 5-22 on page 385 are valid for both cases of power gating; that is, when power gating in the Multi-Protocol 32G PCS and Multi-Protocol 32G PHY x4 for TSMC 7FF is enabled (pg_mode_en = 1) and when power gating is disabled (pg_mode_en = 0).

Figure 5-21 P1.1/P1.2 Entry and Locally Initiated Exit Using PIPE 5.2/4.4.1 Interface

Figure 5-22 P2 to P2.CPM Transition Using PIPE 5.2/4.4.1 Interface

5.7.2 Power State Transition Times

The amount of time that the PCIe PCS takes to acknowledge completion (through pipe_laneX_phystatus) in response to a pipe_laneX_powerdown[3:0] transition is referred to as the power state transition time. Power state transition times are dependent on the reference clock frequency and the data rate.

The PCIe PCS transition times are the PHY's transition time plus 0.2us.

5.8 Per-Lane ropll_clk Source Select

Each PHY has a per-lane ropll_clk, and the UPCS has a source select to choose which ropll_clk is used for each lane of the UPCS. The pipe_laneX_lanepll_src_sel[3:0] signal must be set according to the link combination used. Table 5-12 lists the options for this source select (refer to rows included with the top-level wrappers in the package).

Table 5-12 Link Options Obtainable with lanepll_src_sel

Top-Level Wrapper	Link Options: pipe_laneX_lanepll_src_sel value for each lane																Description	Mnemonic	Supported Bifurcation: Per Lane Clock	Per Link Clock: DWC_UPCS_NLINKS		
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				1	2	4
upcs_pipe_x1_x1	0																1 link	x1	Y	N/A	N/A	N/A
upcs_pipe_x2_x2 upcs_pipe_x2_x1x1	0	0															1 link	x2	Y	Y	Y	N/A
	0	1															2 links	x1_x1	Y		Y	N/A
upcs_pipe_x4_x4 upcs_pipe_x4_x2x2 upcs_pipe_x4_x1x1x1x1	0	0	0	0													1 link	x4	Y	Y	Y	Y
	0	0	2	2													2 links	x2_x2	Y		Y	Y
	0	1	2	2													3 links	x1_x1_x2	Y			Y
	0	0	2	3													3 links	x2_x1_x1	Y			Y
	0	1	2	3													4 links	x1_x1_x1_x1	Y			Y

Top-Level Wrapper	Link Options: pipe_laneX_lanepll_src_sel value for each lane																Description	Mnemonic	Supported Bifurcation: Per Lane Clock	Per Link Clock: DWC_UPCS_NLINKS		
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				1	2	4
upcs_pipe_x8_x8 upcs_pipe_x8_x4x4 upcs_pipe_x8_x2x2x2x2	0	0	0	0	0	0	0	0									1 link	x8	Y	Y	Y	Y
	0	0	0	0	4	4	4	4									2 links	x4_x4	Y		Y	Y
	0	0	2	2	4	4	4	4									3 links	x2_x2_x4	Y			Y
	0	0	0	0	4	4	6	6									3 links	x4_x2_x2	Y			Y
	0	0	2	2	4	4	6	6									4 links	x2_x2_x2_x2	Y			Y
	0	1	2	3	4	5	6	7									8 links	x1_x1_x1_x1_x1_x1_x1_x1	Y			
	0	*	*	*	*	*	*	*									any combination of the above plus rule: (laneN_lanepll_src_sel must be equal to or larger than lane{N-1}_lanepll_src_sel)	any combination of the above x4_x2_x1_x1/ x4_x1_x1_x2 x2_x1_x1_x4 / x1_x1_x2_x4 / x4_x1_x1_x1_x1 / x1_x1_x1_x1_x4 / x2_x2_x2_x1_x1 / x2_x2_x1_x1_x2 / x2_x1_x1_x2_x2 / x1_x1_x2_x2_x2 / x2_x2_x1_x1_x1_x1 / x1_x1_x1_x1_x2_x2 / x2_x1_x1_x1_x1_x1_x1 / x1_x1_x2_x1_x1_x1_x1 / x1_x1_x1_x1_x2_x1_x1 / x1_x1_x1_x1_x1_x1_x2	Y			

Top-Level Wrapper	Link Options: pipe_laneX_lanepll_src_sel value for each lane																Description	Mnemonic	Supported Bifurcation: Per Lane Clock	Per Link Clock: DWC_UPCS_NLINKS		
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15				1	2	4
upcs_pipe_x16_x8x8 upcs_pipe_x16_x4x4x4x4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1 link	x16	Y	Y	Y	Y
	0	0	0	0	0	0	0	0	8	8	8	8	8	8	8	8	2 links	x8_x8	Y		Y	Y
	0	0	0	0	4	4	4	4	8	8	8	8	8	8	8	8	3 links	x4_x4_x8	Y			Y
	0	0	0	0	0	0	0	0	8	8	8	8	12	12	12	12	3 links	x8_x4_x4	Y			Y
	0	0	0	0	4	4	4	4	8	8	8	8	12	12	12	12	4 links	x4_x4_x4_x4	Y			Y
	0	0	2	2	4	4	6	6	8	8	10	10	12	12	14	14	8 links	x2_x2_x2_x2_x2_x2_x2_x2	Y			
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16 links	x1_x1_x1_x1_x1_x1_x1_x1_ x1_x1_x1_x1_x1_x1_x1_x1	Y			
	0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	any combination of the above plus rule: (laneN_lanepll_src_sel must be equal to or larger than lane{N-1}_lanepll_src_sel)	(all combinations are not provided in this table)	Y			

5.9 Clock Architecture and Link Selection

The PCS supports two clock architectures: per-lane and per-link. Per-link clock architecture is only supported in PCLK as PHY Output mode. SerDes Architecture only supports per-lane clock architecture.

- If the DWC_UPCS_PERLINK_CLK_ARCH parameter is not defined, the clock selection logic is implemented per lane.
- If DWC_UPCS_PERLINK_CLK_ARCH is defined, the clock selection logic is implemented per link.

If a design needs only one, two, three, or four links, it is best to choose the per-link clock architecture. This clock architecture simplifies Clock Tree Synthesis (CTS) by reducing the number of clocks in the design.

To use the per-link clock architecture, you must set DWC_UPCS_PERLINK_CLK_ARCH at compile time. DWC_UPCS_NLINKS = 1/2/4 must be set when DWC_UPCS_PERLINK_CLK_ARCH is defined, and the DWC_UPCS_NLINKS_GTR1 and DWC_UPCS_GTR2 parameters must be set when DWC_UPCS_NLINKS > 1 and DWC_UPCS_NLINKS > 2, respectively.

For a description of these parameters, see "Device Configuration" on page 18.

Note The supported link combinations are described in Table 5-12: "Link Options Obtainable with lanepll_src_sel".

Signal pipe_laneX_lanepll_src_sel must be constant at runtime and equal to the Table 5-12: "Link Options Obtainable with lanepll_src_sel".

When DWC_UPCS_NLINKS = 2, two use modes are possible at runtime.

- All lanes belong to the same link.
- Half of the lanes (lower lanes) belong to one link, while the other half (upper lanes) belong to another link.

DWC_UPCS_NLINKS = 3 is not supported. DWC_UPCS_NLINKS must be set to 4 to use three links.

When DWC_UPCS_NLINKS = 4, three basic use modes are possible at runtime.

- All lanes belong to the same link.
- Half of the lanes (lower lanes) belong to one link, while the other half (upper lanes) belong to another link.
- A quarter of the lanes (lower) belongs to one link, while the next quarter belongs to another link, and so on, up to four links.

The "half of the lanes" and "quarter of the lanes" combination is allowed.

5.10 Enabling SRIS

Estore supports SRNS / SRIS when pipe_rxX_es_mode=0 (half-full) or 1 (nominal empty).

If SRIS is supported with spread on input reference clock, use the following settings to enable SRIS:

- phyN_mplla/b_ssc_en=0 (PHY not enabled to generate SSC on MPLL clocks)
- pipe_rxX_sris_mode_en=1 (configures the PHY CDR and **if** DWC_E32MP_X4_ESTORE_DEPTH_-CONTROL_SUPPORT is NOT defined, the elastic buffer to recover Independent Spread Spectrum Data)
- pipe_laneX_cmn_refclk_mode=0 (not common refclk mode)
- pipe_rxX_es0_cmn_refclk_mode=0 (not common refclk mode)
- phyN_mplla/b_short_lock_en = 0

If SRIS is supported, by PHY generating spread, use the following settings:

- phyN_mplla/b_ssc_en=1 (PHY enabled to generate SSC on MPLL clocks)
- pipe_rxX_sris_mode_en=1 (configures the PHY CDR and **if** DWC_E32MP_X4_ESTORE_DEPTH_-CONTROL_SUPPORT is NOT defined, the elastic buffer to recover Independent Spread Spectrum Data)
- pipe_laneX_cmn_refclk_mode=0 (not common refclk mode)
- pipe_rxX_es0_cmn_refclk_mode=0 (not common refclk mode)
- phyN_mplla/b_short_lock_en = 0

To support SRNS:

- phyN_mplla/b_ssc_en=0 (PHY not enabled to generate SSC on MPLL clocks)
- pipe_rxX_sris_mode_en=0 (not SRIS enabled)
- pipe_laneX_cmn_refclk_mode=0 (not common refclk mode)
- pipe_rxX_es0_cmn_refclk_mode=0 (not common refclk mode)
- phyN_mplla/b_short_lock_en = 0

To support common reference clock system:

- phyN_mplla/b_ssc_en=0 (PHY not enabled to generate SSC on MPLL clocks)
- pipe_rxX_sris_mode_en=0 (not SRIS enabled)
- pipe_laneX_cmn_refclk_mode=1 (common refclk mode)
- pipe_rxX_es0_cmn_refclk_mode=0 or 1
- phyN_mplla/b_short_lock_en = 1

5.11 Estore Configuration

The Elastic Buffer or Estore supports Nominal Half-Full and Nominal Empty buffer modes and supports common reference clock systems in these modes. The UPCS drives MPLL settings to the PHY based on protocol, rate, and common clock mode.

If the PHY is operating in a common refclk system, set pipe_laneX_cmn_refclk_mode to 1'b1; otherwise, set it to 1'b0. The Estore mode can be changed when pipe_laneX_reset_n is asserted or when pipe_laneX_rate is changed.

Estore is reset when the following occurs:

- pipe_rxX_standby is asserted.
- Estore modes are changed.
- Loopback is NOT enabled, and Skip Block Error is detected.
- phy_rx_valid is de-asserted.
- Read Data is not valid.
- Underflow condition occurs.
- Overflow condition occurs.

5.12 Recalibration

When Original Pipe Architecture is enabled (pipe_laneX_serdes_mode=0), the PCS supports a recalibration timer. Once the PCS receives valid data from the PHY (phy_rxX_valid=1), it looks for symbol alignment for PCIe Gen1/2 rates or block alignment for PCIe Gen3/4/5 rates. If alignment is not seen within 6 us, the PCS requests the PHY to recalibrate. This procedure ensures that if the CDR in the PHY rails-out, it can recover.

When SerDes Pipe Architecture is enabled (pipe_laneX_serdes_mode=1), the PCS supports a recalibration timer. Once the PCS receives valid data from the PHY (phy_rxX_valid=1), it starts a 6-us timer. When the controller detects alignment, pipe_laneX_rxready asserts, and the UPCS stops the timer.

To disable this timeout, set upcs_pipe_config[6]=1.

5.13 RX Adaptation

The PCS supports receiver adaptation in legacy pin mode (non LowPinCount) through the pipe_rxX_eq_eval signal. In LowPinCount mode, the PCS supports receiver adaptation through the message bus. In either case, once the EQ EVAL is initiated, the PCS asserts phy_rxX_adapt_req to the PHY and waits for phy_rxX_adapt_ack.

The PCS acknowledges the evaluation in legacy pin mode by asserting PhyStatus and pipe_rxX_eq_fig_merit. In LowPinCount mode, the PCS acknowledges the evaluation over the messagebus, as described in PIPE 5.2 Specification.

When EQ ABORT is issued, either through de-assertion of pipe_rxX_eq_eval in legacy pin mode or over the message bus, the PHY will continue the last requested evaluation until it is completed. Then, the PCS issues the Phystatus or messagebus transaction to complete the evaluation. This may lead to an additional EQ Eval time after the abort is issued until the EQ EVAL is completed. Refer to the PHY databook for EQ EVAL time.

Note

For suggested presets that must be requested to the remote transmitter during receiver adaptation, see `phy/ipxact/*snps_iip.config` file in the package and search for the following macros:

- DEFAULT_GEN3_EQ_PSET_REQ_VEC
- DEFAULT_GEN4_EQ_PSET_REQ_VEC
- DEFAULT_GEN5_EQ_PSET_REQ_VEC

5.14 PHY Bifurcation Support

Splitting different lanes in the same PHY across different links is referred to as PHY bifurcation. If you want to limit bifurcation, set the DWC_UPCS_PERLINK_CLK_ARCH macro (N/A for PCLK as PHY Input). Otherwise, each PCS lane can be independently configured as part of a particular link. For further details, see "Clock Architecture and Link Selection".

PHY bifurcation in totally independent links is not supported because the control inputs to the PHY for reset, reference clock enable, and power enable are shared within the PHY by all the lanes belonging to that PHY. For details on restrictions regarding bifurcation support in the PHY, refer to the Multi-Protocol 32G PHY x4 for TSMC 7FF databook, "Aggregation and Bifurcation Support" section.

Attention

To support multiple links:

- Reference clock must not be removed if any lane is requesting refclk (pipe_laneX-_ref_clk_req_n==0).
- Power must remain applied if any power enable is asserted (phyN_pcs_pwr_en==1, phyN_pma_pwr_en==1, upcs_pwr_en==1).
- For the description of how to choose clock sources for PCS lane clocks in synthesis, see "PIPE (PCS and PHY) Constraints".

5.15 PHY Aggregation Support

The PCS can be connected to a maximum of four PHYs so that the number of lanes on the PCS equals to the cumulative number of lanes on the PHYs. The PHY release package includes the PCS versions that can be generated from the PHY.

For example, the release package for a x2 PHY includes the following PIPE wrappers instantiating the PCS and the PHYs included in the package are as follows:

- x2 PIPE (./macro/rtl/dwc_*_upcs_x2_*_ x2_x2_pipe.v)
- x4 PIPE (./macro/rtl/dwc_*_upcs_x2_*_ x4_x2x2_pipe.v)
- x8 PIPE (./macro/rtl/dwc_*_upcs_x2_*_ x8_x2x2x2x2_pipe.v)

Combining different instances of the PHY to operate as a common link is referred to as PHY aggregation. The PCS supports PHY aggregation through the pcs_laneX_lanepll_src_sel input, which selects the PHY Lane to be used to source the MPLL and reference clock inputs for each laneX (for an illustration, see Table 2-1: "Supported Data Throttling Frequencies for PCIe"). Using pcs_laneX_lanepll_src_sel, you can configure the PCS lanes to operate as one or more links—with each link operating with an independent rate (pcs_laneX_rate) and protocol (pcs_laneX_protocol)—provided that all lanes sourced by the same PHY are part of the same link. For more details, refer to "Per-Lane ropll_clk Source Select". For information about the number of links supported, see "Clock Architecture and Link Selection".

Attention See "PIPE (PCS and PHY) Constraints" for the description of how to choose clock sources for PCS lane clocks in synthesis.

5.16 Lane Disabling and Up/Down-Configuration

The PCS supports active disabling and enabling lanes in a link.

5.16.1 RX Standby Mode

For PIPE 4.4.1 compatibility, a lane in an active link can be disabled by setting pipe_txX_elecidle = 1 and pipe_rxX_standby = 1 while keeping pipe_laneX_powerdown set to P0. This setting maintains the TX common mode on the lane while turning off the transmit driver and disabling the CDR—keeping the remaining receiver circuitry on. You can re-enable the disabled lanes with minimal supported wake-up time by setting pipe_txX_elecidle = 0 and pipe_rxX_standby = 0.

5.16.2 PIPE Lane Disable

When pipe_laneX_serdes_mode=0, a lane in an active link can be disabled by setting pipe_txX_elecidle = 1 and pipe_tx1_compliance = 1. In this state, the PHY's TX and RX are placed in the following equivalent state (TX and RX circuits off):

- PCIe: P2
- SATA: P2

This state is a significantly lower power state as compared to disabling the lane using pipe_txX_elecidle = 1 and pipe_rxX_standby = 1. The disabled lanes can be enabled again according to the requirements defined in the PIPE 4.4.1/4.3 specification, "Multi-lane PIPE – PCI Express Mode" section. Therefore, the MAC must either assert pipe_laneX_reset_n or set pipe_laneX_powerdown to P1 when it exits the disabled state.

When pipe_laneX_serdes_mode=1, a lane can be disabled by pipe_laneX_turnoff=1.

Exception

The PIPE 4.4.1/4.3 specification states that the PHY should ignore the PIPE signals when a lane is disabled using pipe_txX_elecidle = 1 and pipe_tx1_compliance = 1.

When in such a disabled state, the PCS supports the following two modes:

- **upcs_pipe_config[0] = 0**: The PCS ignores PIPE interface signals. In this mode, the pipe_laneX-_pcs_clk output is gated.
- **upcs_pipe_config[0] = 1**: The PCS monitors the PIPE interface signal, pipe_laneX_powerdown.

  Therefore, if pipe_laneX_powerdown is set to the following states, the pipe_laneX_pcs_clk output is available.

  - PCIe: P0, P0s, and P1
  - SATA: Active and Partial

  This mode enables the MAC to continue using the pipe_laneX_pcs_clk output from a particular lane in a link to sample/launch PIPE signals for other active lanes in the same link, resulting in a simpler clock architecture in the MAC. Without this mode, the MAC is required to use the pipe_laneX_pcs_clk output of each lane to sample/launch data of the same lane. For details on this requirement, see "PCS Output Clocks" on page 355.

5.16.3 P1.2 State

You can disable a lane by placing it in a P1.2 state (for information about P.1.2 entry and exit, see "PCIe L1 Substate Transitions" on page 381). If all other lanes in the link and all lanes on one PHY are in a P1.2 state,

this mode is the most power-efficient because, in this case, the reference clock for that PHY can be turned off.

5.17 PHY Configuration Overrides

Attention

- Protocol0 is used for PCIe.
- Protocol1 is not used. Tie all of these inputs to 0.
- Protocol2 is used for SATA. It is not used; tie all of these inputs to 0.
- The protocol_esm* inputs are used for CCIX rates. If CCIX is not used, tie all of these inputs to 0.

The PCS drives the configuration inputs to the PHY using predefined and prevalidated settings.

You can override these settings using primary inputs, protocol[0,1,2]_ext_*_[g1,g2,g3,g4,g5], when phy_ext_ctrl_sel = 1. Apply these settings when phy_reset is asserted.

- protocol[0,1,2] refer to PHY settings that correspond to pipe_laneX_protocol[1:0] = 0, 1, and 2, respectively.
- g[1,2,3,4,5] refer to PHY settings that correspond to (up to) five rates within the selected protocol.

Because protocol1 (pipe_laneX_protocol = 01) supports only two rates, protocol1_ext_*_[g3,g4] inputs do not exist. Similarly, because protocol2 (pipe_laneX_protocol = 10) supports only three rates, protocol2_ext_*_g4 inputs do not exist. These overrides provide you with the flexibility to change PHY configuration settings in the SoC implementation.

Some override settings are lane-specific, while others are specific to each PHY.

- For lane-specific configuration settings, the PCS uses the per-lane protocol (pipe_laneX_protocol) and rate (pipe_laneX_rate) inputs to select the protocol and rate-specific configuration settings for lane X.
- For PHY-specific configuration settings (for example, MPLL controls for the PHY), the PCS uses the protocol (pipe_laneX_protocol) and rate (pipe_laneX_rate) inputs for the lane with the lowest lane index to select the protocol and rate-specific configuration settings for that PHY.

  For example, for the aggregation option of {x4 PHY0, x2 PHY1, x2 PHY2}, for pipe_lane0_protocol, pipe_lane4_protocol, and pipe_lane6_protocol, select the MPLL settings for PHY0, PHY1, and PHY2, respectively.

Note

It is mandatory that you drive the inputs (protocol[0,1,2]_ext_*, phy_ext_ctrl_sel, protocol_ext_mpll{a,b}_recal_bank_sel and protocol_ext_mpll{a,b}_recal_bank_sel_ovrd_en) using external, programmable registers in your SoC and ASIC with the phy_ext_ctrl_sel and protocol_ext_mpll{a,b}_recal_bank_sel_ovrd_en default values set to 0.

5.18 Transmitter Equalization Settings

The PHY transmitter equalization settings are controlled by the pipe_txX_deemph, protocol[0,1,2]_ext_tx_eq_[pre,main,post]_[g1,g2,g3,g4], and protocol[0,1,2]_ext_tx_eq_ovrd_[g1,g2,g3,g4] inputs of the PCS. The pipe_txX_deemph settings corresponding to supported equalization settings can be set to the following table values or any legal value with FS = 48 / LF = 16 for rules a, b, c of the PCIe base specification.

Table 5-13 Preset-to-Coefficient Table

Preset Number	User Codes (Decimal)		
	pipe_txX_deemph[11:6] (C0)	pipe_txX_deemph[5:0] (C-1)	pipe_txX_deemph[17:12] (C+1)
P4	48	0	0
P1	40	0	8
P0	36	0	12
P9	40	8	0
P8	36	6	6
P7	34	5	9
P5	44	4	0
P6	42	6	0
P3	42	0	6
P2	38	0	10
P10	32	0	16

For PCIe3/4/5 operation, the PCS maps the pipe_txX_eq_preset/pipe_txX_eq_preset_coeff_req inputs to the pipe_txX_eq_preset_coeff/pipe_txX_eq_preset_coeff_vld outputs. The mapping is the same as in Table 5-13. Note that in this case, the MAC must apply the value on pipe_txX_eq_preset_coeff back to the PCS through the pipe_txX_deemph input.

The PCS converts PCS pipe_txX_deemph to PHY txX_eq_post, txX_eq_main, and txX_eq_pre using the “Stage 1” and “Stage 2” equations defined in the section “TX Equalization Coefficient Mapping” of the PHY databook.

5.19 Lane Margining at Receiver

The PCS supports lane margining at the receiver based on the definition in PCI_Express_Base_5.0 rev 1.0. The following table shows the specific Lane Margining features supported.

Table 5-14 Lane Margining Parameters

Parameter	Value	Description
RXMarginingVoltageSupported	1	The PHY indicates where it supports voltage margining, encoded as follows: ■ 0: No support ■ 1: Support
RXMarginingSamplingRateVoltage	31	Percentage of bits margined during Voltage Margining mode, calculated as 1 / 64 * (Sampling_Rate[5:0] + 1) Valid values: 0–63
RXMarginingSamplingRateTiming	31	Percentage of bits margined during timing margining mode, calculated as 1 / 64 * (Sampling_Rate[5:0] + 1) Valid values: 0–63
RXMarginingIndependentLeftRight	1	The PHY indicates whether it supports independent left and right time margining, encoded as follows: ■ 0: No support ■ 1: Support
RXMarginingIndependentUpDown	1	The PHY indicates whether it supports independent up and down voltage margining, encoded as follows: ■ 0: No support ■ 1: Support
RXMarginingIndependentErrorSampler	1	The PHY indicates whether it supports an error sampler independent from the main sampler to enable higher BERs to be measured, encoded as follows: ■ 0: No support ■ 1: Support
RXMarginingVoltageSteps	32	Total number of voltage steps, minimum range ± 50 mV A value of zero indicates that voltage margining is not supported. Valid non-zero values: 32–127
RXMarginingTimingSteps	8	Total number of timing steps, minimum range ± .2 UI Valid values: 8–63

Parameter	Value	Description
RXMarginingMaxVoltageOffset	5	Offset at maximum step value as a percentage of 1 volt Valid values: 5–50
RXMarginingMaxTimingOffset	20	Offset at maximum step value as a percentage of nominal UI Valid values: 20–50
RXMarginingMaxLanes	0-15	Maximum number of lanes that can be margined simultaneously Valid values:1–32 Recommended value: Number of lanes the PHY supports
RXMarginingSampleReportingMethod	1	Indicates whether a sample frequency or a sample count is reported, encoded as follows: ■ 0: Sample count reported. ■ 1: Sample frequency reported.
RXMarginingMaxTimingOffsetChange	8	Maximum number of steps margin offset that can be changed with one command during timing margining Valid values: 1–127
RXMarginingMaxVoltageOffsetChange	32	Maximum number of steps margin offset that can be changed with one command during voltage margining Valid values: 1–127
RXMessageBusWriteBufferDepth	5	Number of write buffer entries that the PHY has implemented to receive writes from the MAC, where one entry can hold the 3 bytes of information associated with each write transaction.
TXMessageBusMinWriteBufferDepth	5	Minimum number of write buffer entries that the PHY expects the MAC to implement to receive writes from the PHY. PHY transmits a series of "N" uncommitted write commands when TxMessageBusMinWriteBufferDepth is "N." A single committed write command is transmitted after them. Valid values: 0–8 The MAC can implement more than the minimum required by the PHY; however, there is little benefit in increasing the number of write buffer entries more than is required.

The Message Bus interface is used to communicate margining commands between the PHY and a controller. To set the PHY registers from controller, pipe_laneX_m2p_messagebus[7:0] is used. To set MAC registers

from the PCS, pipe_laneX_p2m_messagebus[7:0] is used. The PCS supports only “write committed non posted” and “write_ack” commands.

If “write committed non posted” is requested by controller through pipe_laneX_m2p_messagebus[7:0], the PCS responds “write_ack” through pipe_laneX_p2m_messagebus[7:0]. If “write committed non posted” is requested by the PCS through pipe_laneX_p2m_messagebus[7:0], the controller is expected to respond “write_ack” through pipe_laneX_m2p_messagebus[7:0].

The PCS does not support SampleCount in the RX Margin Status1 register in the PIPE 5.2 Specification.

Refer to “Lane Margining at the Receiver Sequences” in the PIPE 5.2 Specification; for example, margining sequences are supported.

Refer to “independent” examples as the PHY has an Independent Error Sampler.

5.20 ESM Rate Change Sequence

This section describes the ESM rate change sequence, including L1 calibration flow and Electrical Idle Exit Ordered Set format.

5.20.1 High-Level ESM Rate Change Sequence

Before entering ESM rates, the link initially transitions via normal PCIe mode to L0 at 16.0 GT/s date rate. After this transition, system software (SSW) probes configuration space to determine support for ESM rates. SSW transitions the link to 2.5 GT/s, then programs ESM Rate 0 and Rate 1. SSW then programs the link to enter L1/P1, where the PHY calibrates programmed data rates in ESM Rate 0 and Rate 1 (if these rates are other than 2.5/5/8/16 GT/s). After calibration is complete, the link is transitioned out of L1 and link-up proceeds through ESM Data Rate 0 and Rate 1. For more detailed steps involved in linking up to ESM rates, refer to *CCIX Transport*.

For Message Bus registers involved in ESM, see section “Message Bus Support” on page 402.

The following figure shows a high-level sequence when changing the ESM rate.

Figure 5-23 High-Level ESM Rate Change Sequence

Note

- Even for a second L1 calibration for which the same ESM rate 0/1 is requested, the PCS requests MPLL calibration.
- The minimum time needed in L1 is 6 ms. For information about the L1 Calibration flow, refer to the Multi-Protocol 32G PHY x4 for TSMC 7FF databook, “L1 Calibration Flow” section.
- During MPLL calibration, pclk is removed. MPLL calibration for 2.5/5/8/16 GT/s is performed during the initial power-up.

5.21 EIEOS Format

When operating at data rates greater than 16.0 GT/s, an Electrical Idle Exit Ordered Set (EIEOS) transmitted and received is as follows.

Table 5-15 Electrical Idle Exit Ordered Set for Data Rates Greater Than 16.0 GT/s

Symbol Number	Value	Description
0,1, 2, 3, 8, 9, 10, 11	00h	Symbol 0: EIEOS identifier A low-frequency pattern that alternates between 32 zeros and 32 ones
4, 5, 6, 7, 12, 13, 14, 15	FFh	A low-frequency pattern that alternates between 32 zeros and 32 ones

5.22 Message Bus Support

5.22.1 Message Bus Registers for PCIE

Registers in the PIPE 5.2 Specification are supported, with the following exceptions:

- If Legacy Pin Mode is active, PHY registers related to Low Pin Count must not be accessed through Message Bus Interface.
- PHY Register Address 2h: Elastic Buffer Control update does not complete within 16 PCLK cycles.
- PHY Register Address 8h: PHY RX Control4, ElasticBufferResetControl sequence does not complete with 16 PCLK cycles.

RxEqTraining and RxEqTrainDone bits are supported, but the PHY does not require support for these bits.

The following registers are not supported (they are optional):

- PHY Register Address 004h: PHY RX Control1
  - Bit [1] IORecal
- PHY Register Address 007h: Elastic Buffer Location Update Frequency
- PHY Register Address 400h: PHY TX Control0
- PHY Register Address 401h: PHY TX Control1
- MAC Register Address 001h: RX Margin Status1
- MAC Register Address 004h: Elastic Buffer Location (supported in Legacy Pin Mode through pipe_rxX_ebuff_location)
- MAC Register Address 005h: RX Status0
  - Bit [1] IORecalDone
- MAC Register Address 006h: RX Control0
  - Bit [0] PhyIORecalRequest

5.22.2 Message Bus Registers for CCIX Specification

The following tables describe the PHY Message Bus registers for the CCIX specification.

Table 5-16 PHY Message Bus Registers

Register Address	Register Name	Bit	Field Name	Attribute	Description	Allowed? Command
12'hf00	ESM Rate0	[7]	Reserved	N/A		WC
		[6:0]	ESM Data Rate0	Level	Rate for ESM operation for first ESM rate. ■ 000011b: 8.0 GT/s ■ 000110b: 16.0 GT/s	
12'hf01	ESM Rate1	[7]	Reserved	N/A		WC
		[6:0]	ESM Data Rate1	Level	Rate for ESM operation for second ESM rate. ■ 001010b: 20.0 GT/s ■ 001111b: 25.0 GT/s	
12'hf02	ESM Control	[7:2]	Reserved	N/A		WC
		[1]	ESM Calibration request	One-cycle	Controller is requesting PHY to calibrate for ESM.	
		[0]	ESM Enable	Level	Enable for ESM operation.	
12'hf03	ESM Link Reach Target	[7:1]	Reserved	N/A		WC
		[0]	ESM Long Reach	Level	Control ESM Extended EQ2/3 timeout values	

The following table describes the Message Bus interface MAC registers.

Table 5-17 Message Bus Interface MAC Registers

Register Address	Register Name	Bit	Field Name	Attribute	Description	Allowed Command
12'hf00	ESM Status	[7:1]	Reserved	N/A		WC
		[0]	ESM Calibration Complete	One-cycle	Calibration complete indication.	

5.23 SATA BIST Loopback Mode

The PCS does not support the optional SATA Loopback mode mentioned in the PIPE 4.3 specification. However, the PCS supports two BIST Loopback modes (as required in the SATA specification) through the use of sideband signals. These two modes are described as follows.

Far-End Analog Loopback Mode

The far-end analog loopback signal from the controller must be connected to the pcs_laneX_rx2tx_loopbk signal. In this mode, data received on the RX serial pins is looped back on the TX serial pins in the PHY. To enable this loopback to work correctly, when pcs_laneX_rx2tx_loopbk = 1, the PCS changes phy_txX_rate to 0b, phy_txX_data_en to 1b, and phy_rxX_data_en to 1b.

Near-End Analog Loopback Mode

The near-end analog loopback signal from the controller must be connected to the pcs_laneX_tx2rx_loopbk signal. The controller sends a data pattern on the TX datapath in this mode. This data is sent out by the PCS on the TX datapath and is looped back in the PHY analog transceiver. The data is then returned through the RX datapath in the PCS. For data to be returned on the RX path in the PCS, COM detection and locking must first occur. The PCS needs a minimum of four COMs—with any amount of gap between them—for the locking to occur. After the lock happens, any data pattern sent by the controller is looped back correctly onto the RX path.

5.24 SATA: COM Align and Misalign Mechanism

When the PCS receives data from the PHY on the RX input (phy_rxX_data[1939:0], phy_rxX_valid), the PCS looks for COM (K28.5) characters over any 10 bits, even crossing the clock cycle boundary. When a COM is detected, it is then aligned to the lower 10 bits of the RX databus to meet the 'Byte 0' alignment requirement of the PIPE specification. After a COM is aligned to 'Byte 0', the PCS looks for nine back-to-back COMs to ensure that the received data has been captured correctly at the correct speed. It is possible for the PHY to capture COM characters randomly at the wrong speed. By checking for nine back-to-back COMs, the chance of COM detection at an incorrect speed is eliminated. After nine back-to-back COMs are detected, pcs_rxX_valid is asserted, and the received data passes through the elastic buffer and to the PIPE RX datapath.

After a COM is aligned to 'Byte 0', its alignment must not change. When a COM is detected that is not aligned to 'Byte 0', misalign detection begins. When four such misaligned COMs are detected, the PCS sends the misaligned information over the PIPE interface to the controller (pcs_rxX_status = 3'b011).

5.25 SATA - Disconnect Detection

The SATA controller must implement logic to detect and distinguish between OOB signaling and physical disconnection of the serial bus because the latency involved in detecting the disconnection might be more than the first gap of OOB signaling.

Here is the pseudo code for logic that you can use in the SATA controller for disconnect detection. This pseudo code is just an example and should not be considered as verified:

((pcs_rxX_elecidle == low) && (12 back-to-back 8b/10b decode errors)) OR

((pcs_rxX_elecidle == low) && (pcs_rxX_valid == low))

5.26 SATA - Nominal Empty Buffer Mode Operation

The elastic buffer must be operated in 'Nominal-Empty' mode when used for SATA (pcs_rxX_es_mode == 2'b01) operation. There is a 20-DWORD latency requirement for SATA between HOLD and HOLDA primitives, and this requirement cannot be met when the elastic buffer is operated in either Nominal Half-Full Buffer mode or Common Reference Clock mode.

5.27 Power Gating Support

The PCS and RAW_PCS (in the PHY) support one always-on power domain, "VDD," along with multiple power-gated domains. The power switches for the PCS and RAW_PCS portion of the PHY are external to the IP and control the supply to the power-gated domains.

The Always-on (AON) hierarchy in the design (PCS + PHY) is as follows:

- dwc_*_upcs_*_pipe.dwc_*_upcs_pcs_x*.dwc_*_upcs_pipe_ctl_x*
- dwc_*_phy_x*.dwc_*_pcs_raw_x*.dwc_*_pcs_raw_aon_x*

Figure 5-24 on page 407 shows control signal connectivity for power gating the PCS and the PHY. The power switches for the Raw PCS and the PCS can be implemented either as single switches (as shown in the figure) or distributed switches. In either case, one power-stable signal must be generated by the single switch or daisy chained through distributed switches.

PCIe: The Raw PCS generates power-gating enables for the Raw PCS switches as well as the PMA switch. The phyN_pcs_pwr_stable and phyN_pma_pwr_stable inputs must be asserted only after the supplies ramp up. Power Gating mode is enabled when the pg_mode_en input to the PCS and the PHY is set to 1 and when the PCS is placed into a P1.2 power state (for information about P1.2 entry and exit, see "PCIe L1 Substate Transitions" on page 381). When pg_mode_en = 0, the enables for the power switches are held asserted to a 1.

Figure 5-24 Power Gating Controls for Multi-Protocol 32G PCS and PHY

The expectation for the power-gating methodology is that the top-level wrappers and the always-on (AON) blocks are connected to the ungated power domain, while the majority of sub-blocks are on “islands” of power-gated logic.

5.28 Tie-Off Values for Unused Lanes

If UPCS lanes are unused, they need to be tied off to specific values. This allows other lanes to fully enter L1/P1 substates and allows power to be removed, where applicable.

Table 5-18 Tie-Off Values for Unused Lanes

Signal Name	Tie-Off Value
pipe_laneX_encdec_bypass	1'b0
pipe_laneX_if_width	2'd0
pipe_laneX_lanepll_src_sel	Unused lanes must be set to the lane’s number. For example, for x4 configurations (as an example, 1x4 macro / 2x2 macros / 4x1 macros), and only lane0 is active: pipe_lane0_lanepll_src_sel = 0 pipe_lane<1,2,3>_lanepll_src_sel = <1,2,3> (unused lanes) For example, for x8 configurations (as an example. 2x4 macros / 4x2 macros), lanes0-3 are each independent links, lane4-5 is one link and lanes 6,7 are unused: pipe_lane<0,1,2,3>_lanepll_src_sel =<0,1,2,3> pipe_lane<4,5>_lanepll_src_sel =<4,4> pipe_lane<6,7>_lanepll_src_sel =<6,7> (unused lanes) Contact Customer Support for additional assistance.
pipe_laneX_powerdown	4’b0100
pipe_laneX_rate	3'd0
pipe_laneX_reset_n	1'b0
pipe_laneX_m2p_messagebus	8'b0
pipe_rxX_es_mode	2'b0
pipe_rxX_blk_align_ctl	1'b0
pipe_rxX_eq_eval	1'b0
pipe_rxX_eq_in_prog	1'b0
pipe_rxX_eq_invld_req	1'b0
pipe_rxX_polarity	1'b0
pipe_rxX_sris_mode_en	1'b0
pipe_rxX_standby	1'b0
pipe_rxX_disable	1’b1

Signal Name	Tie-Off Value
pipe_txX_disable	1’b1
pipe_txX_compliance	1'b1
pipe_txX_elecidle	1'b1
pipe_txX_data	40'h0
pipe_txX_datak	4'b0
pipe_txX_datavalid	1'b0
pipe_txX_deemph	18'd0
pipe_txX_detectrx	1'b0
pipe_txX_eq_preset	5'd0
pipe_txX_eq_preset_coeff_req	1'b0
pipe_txX_margin	3'd0
pipe_txX_startblock	1'b0
pipe_txX_swing	1'b0
pipe_txX_syncheader	4'b0
pipe_laneX_perst_n	1’b0

6 Simulation

The release package includes Verilog RTL for the Multi-Protocol 32G PCS, Raw PCS, and a Verilog behavioral model of the PMA. Contrary to the Verilog RTL for the PCS and Raw PCS—which can be synthesized into a gate-level netlist by a synthesis tool—the PMA's Verilog behavioral model includes a netlist of generic GTECH gates and a non-synthesizable behavioral model of the analog circuits.

In the release package, the Multi-Protocol 32G PCS, Raw PCS Verilog RTL, and PMA Verilog behavioral model reside in the following directories:

- UPCS: ./upcs/rtl/
- Raw PCS: ./phy/rtl/
- PMA Verilog model: ./pma/behavior/

For details on the features that are not accurately modeled in the PMA behavioral model, refer to the DesignWare Cores Multi-Protocol 32G PHY x4 for TSMC 7FF databook.

The Verilog model can be simulated using Synopsys VCS. The following table lists the release directories where the Multi-Protocol 32G PCS, Raw PCS, and PMA model are stored.

Table 6-1 Multi-Protocol 32G PCS and PHY Model Database

Directory	Description
./phy/include	Contains macros used in the Raw PCS, PMA, and simulation files.
./upcs/include	Contains macros used in the PCS and simulation files.
./macro/rtl	Contains the top-level PIPE wrappers instantiating the PCS and the PHY. ■ dwc_* upcs_xN_ns xL_[xN][xN][xN][xN]_pipe.v: Top-level PIPE wrapper ❑ N = number of lanes of the PMA ❑ L = total number of lanes of the PIPE and PCS ❑ [xN][xN][xN][xN] represents the aggregation option.
./upcs/rtl	Contains Verilog RTL files for the UPCS, which includes the following: ■ dwc_*_upcs_xN_ns_upcs_pcs_xL_[xN][xN][xN][xN].v: Top-level PCS module ❑ N = number of lanes of the PMA ❑ L = total number of lanes of the PCS ❑ [xN][xN][xN][xN] represents the aggregation option.

Directory	Description
./phy/rtl	Contains Verilog RTL for the Raw PCS files. ■ dwc_*_phy_xN_ns_pcs_raw_xN.v: Top-level RTL file for the xN Raw PCS ■ dwc_*_phy_xN_ns.v: PHY top-level wrapper instantiating PMA and Raw PCS
./pma/behavior	Contains Verilog RTL for the behavioral models for the PMA. ■ dwc_*_pma_xN_ns_gtech.v: Top-level file for the xN PMA instantiating GTECH model of the digital and behavioral model of analog sub-blocks.

For instructions on PHY-level simulations, refer to the DesignWare Cores Multi-Protocol 32G PHY x4 for TSMC 7FF databook.

Instructions for PIPE-level simulations are provided in the TESTBENCH.README file in the /upcs/testbench/ directory.

The TESTBENCH.README file includes the following sections:

- **Macro definitions**: These definitions are global variables that control all the testbench objects.
- **PLL configuration**: This section defines the PLL output clock.
- **Link definitions**: This section defines each link's sequence.
- **Example test_def.v**: This file is a sample test configuration file.
- **Porting the PIPE to another testbench**: This section describes various simulation modes and explains how to invoke them.
- **Receiver detect**: This section provides instructions on how to force the decision of receiver detection.
- **Running simulations**: This section provides instructions on launching simulations from the testbench provided in the package.

For more information about each section, see TESTBENCH.README in the release package.

7 Synthesis

The Multi-Protocol 32G PCS is delivered as soft RTL, meaning that the product is hardened along with the rest of the RTL of the Raw PCS, controller, and ASIC. Synthesizing the PCS requires using a Verilog-2001-compatible synthesis tool.

This chapter includes the following sections:

- "Synthesis Database" on page 414
- "PIPE (PCS and PHY) Constraints" on page 415

7.1 Synthesis Database

The release package includes the Design Compiler synthesis scripts in TCL format, which you can use to synthesize the Multi-Protocol 32G PCS and Raw PCS (PHY) logic using the timing model of the PMA (PHY). The following table describes the synthesis database structure of this PIPE-level wrapper.

Table 7-1 Synthesis Database Structure

Directory	Description
./upcs/synth	Synthesis directory for PIPE (Multi-Protocol 32G PCS, Raw PCS, and PMA): ■ **run_synth.sh**: Shell script to run synthesis using Design Compiler TCL scripts and to run logic equivalence checks between RTL and synthesized netlist of the PCS and Raw PCS using Formality ■ **readme_synth.txt**: Readme containing instructions to configure and run the synthesis scripts You must update the variables in the scripts to match the installed technology library location (TECH_DIR, STD_DB). These scripts must be run in the ./upcs/synth/ directory.
./upcs/synth/scripts	Synthesis and Formality scripts: ■ **verilog_files.tcl**: List of Verilog files used for synthesis ■ **upcs_dft_insert.tcl**: Sourced from dwc_*_upcs_xN_ns_synth.tcl and used to insert DFT into the design. ■ **dwc_*_upcs_xN_ns_synth.tcl**: Design Compiler synthesis script to synthesize the PCS and Raw PCS using timing .libs for the PMA ■ **upcs_con_e32.tcl**: Design Compiler synthesis constraint file for the PCS and Raw PCS in TCL format ■ **dwc_*_upcs_xN_ns_formal.tcl**: Formality script for running logic equivalence check on the synthesized netlist of the PCS and Raw PCS

7.2 PIPE (PCS and PHY) Constraints

The synthesis constraints are provided in the form of a TCL script used by Synopsys's Design Compiler. This file (upcs_con.tcl) describes the design intent and surrounding constraints for synthesis, clocking, timing, power, test, and environmental and operating conditions. The constraints include the input and output delays of interface pins, clock definitions, and any timing path exceptions in the design, such as false paths or multicycle paths.

When the DWC_UPCS_PERLINK_CLK_ARCH parameter is defined, the packaged constraints define one clock source per link.

When the DWC_UPCS_PERLINK_CLK_ARCH parameter is not defined, the packaged constraints define clock sources for every supported combination of PHY sources as described in "Per-Lane ropll_clk Source Select" on page 386.

There are generated clocks created for each of these combinations. If this flexibility is not needed, these combinations can be limited by the constraints found in upcs/synth/scripts/upcs_con.tcl (see tcl procedure "clksrc_sel").

Note

- UPCS constraints provided are for the highest frequency and data rate.
  - If DWC_UPCS_MULTIPROTOCOL_SUPPORT is defined in upcs/include/*e32_defaults.v, constraints are provided for both SATA and PCIe.
  - If DWC_UPCS_MULTIPROTOCOL_SUPPORT is not defined in upcs/include/*e32_defaults.v, constraints are provided for PCIe only.
  - If lower rate constraints are needed, contact Synopsys.
- The PHY requires the minimum high and low pulse width of txX_clk input to be more than 2UI + 100ps. This txX_clk pulse width requirement must be added to your backend flow as it cannot be properly constrained in front-end synthesis.

Examples

Note These examples are for x16 configuration.

When 1 link is supported:

- Each lane lists "0" as a source of pll

When 4 links are supported:

- Lanes 0 - 3: list "0" as PHY source
- Lanes 4 - 7: list "4" as PHY source
- Lanes 8 - 11: list "8" as PHY source
- Lanes 12 - 15: list "12" as PHY source

Contact Synopsys support for assistance with this feature.

7.2.1 Hand-Placed Logic Constraints

The following table lists the modules in the PCS that must be hand-instantiated as part of synthesis. These modules are typically used around asynchronous clock domain crossing (CDC) points within the design. It is essential that you instantiate these modules correctly, not just synthesize them as usual with the PCS logic. Failure to do so may create glitch-generating and reconvergent logic that causes problems within the PCS. For details on hand-instantiated modules used in the Raw PCS, refer to the associated DesignWare Cores PCIe 5.0 PHY x for databook.

To prevent these modules from optimizationed during synthesis so that only the correct cell is selected by the tool:

- replace the contents of these modules with an instantiation of the appropriate standard cell from a digital library, or
- synthesize each module standalone with a restricted library.

In both cases, you must apply a “set_size_only” attribute to prevent re-optimization of the instance during regular PCS synthesis.

The following table lists which cells require hand placement and the “set_size_only” attribute.

Table 7-2 Hand-Instantiated Modules in PCIe 5.0

Module Name	Description	Hand-Instantiated Cell Required	set_size_only Attribute Needed
pipe_gen_sync	Three-stage data signal synchronizer	Yes, see 7.2.2 for more information.	No
pipe_gen_bus_sync	Three-stage data-bus synchronizer Instantiates pipe_gen_sync	No	No
pipe_gen_rst_sync	Reset synchronizer	Yes, see 7.2.2.	No
pipe_SPECIAL_ani_clk_gate	Clock-gating cell	Yes	Yes
pipe_gen_and2	Two-input AND gate	Yes	Yes
pipe_gen_or2	Two-input OR gate	Yes	Yes
pipe_gen_mux	Two-input MUX	Yes	Yes
pipe_SPECIAL_clk_mux	Two-input clock multiplex	Yes	Yes
pipe_SPECIAL_dummy_scan_clk_mux	Two-input scan multiplex	Yes	Yes
pipe_SPECIAL_dummy_scan_rst_mux	Two-input scan reset multiplex	Yes	Yes
pipe_SPECIAL_clk_mux2	Instantiates pipe_SPECIAL_clk_mux	No	No
pipe_SPECIAL_clk_mux4	Instantiates pipe_SPECIAL_clk_mux2	No	No
pipe_SPECIAL_clk_gate	Instantiates pipe_gen_mux & pipe_SPECIAL_ani_clk_gate	No	No
pipe_gen_buf	Generic random delay for simulation	No	No

7.2.2 Synchronizers

The PCS instantiates synchronizer modules used for clock domain crossing protection. Two types of synchronizer modules are used in the design as follows.

7.2.2.1 Data Synchronizers

For single-bit signals that must be passed from one clock domain to another, an instance of the "dwc_*pipe_gen_sync" modules is used to synchronize the source signal into the receiving clock domain. The following figure shows the structure of a single-bit data synchronizer.

Figure 7-1 dwc_*_pipe_gen_sync module

The input to the first stage of the synchronizer can be considered completely asynchronous, so all paths to the d_s1 register can be considered false_paths.

The second and third registers are d_s2 and q. To ensure stable synchronization, constrain the maximum delay to 20% of the sampling clock frequency for these cells: from d_s1/Q to d_s2/D and from d_s2/Q to d_s3/D.

To ease timing constraint definition, this delay can be implemented as 200 ps plus the intended clock tree depth of the respective sampling clock for the associated module instances. If the paths are constrained explicitly from d_s1/Q to d_s2/D and from d_s2/Q to q/D, you can ignore the clock-tree delay.

7.2.2.2 Reset Synchronizers

An instance of the "dwc_*_pipe_gen_rst_sync" modules is used for synchronizing the asynchronous reset signal. As shown in the following figure, any time an external or internal reset condition is passed from one clock domain to another domain, the reset condition must pass through a dwc_*_pipe_gen_rst_sync module to do the following:

1. Asynchronously capture the reset signal.
2. Stretch the reset pulse to be active for at least one full clock cycle in the receiving domain.
3. Synchronously de-assert the reset signal with respect to the receiving domain.

Figure 7-2 dwc_*_pipe_gen_rst_sync module

Because assertion of a reset causes all flip-flops in the synchronizer module to capture the pulse, the de-assertion edge of the asynchronous reset input does not affect the final flip-flop that drives the reset into the receiving domain. Therefore, all asynchronous paths through the d_s3 register of this module can be considered false.

The second and third registers in the dwc_*_gen_rst_sync module are d_s2 and d_s3. To ensure stable synchronization, constrain the maximum delay to these cells, from d_s1/Q to d_s2/D and from d_s2/Q to d_s3/D, to 20% of the sampling clock frequency.

To ease timing constraint definition, this delay can be implemented as 200 ps plus the intended clock tree depth of the respective sampling clock for the associated module instance. If the paths are constrained explicitly from d_s1/Q to d_s2/D and from d_s2/Q to d_s3/D, you can ignore the clock-tree delay.

Metastability behavior is modeled in the synchronizers and is activated by defining the ANI_SIM_MODE macro during simulations.