Speedster7t SerDes User Guide (UG099)

Speedster FPGAs

Preliminary Data

Achronix®
Data Acceleration

Copyrights, Trademarks and Disclaimers

Preliminary Data

This document contains preliminary information and is subject to change without notice. Information provided herein is based on internal engineering specifications and/or initial characterization data.

Achronix Semiconductor Corporation

2903 Bunker Hill Lane
Santa Clara, CA 95054
USA

Website: www.achronix.com
E-mail : info@achronix.com

Table of Contents

Chapter - 1: Introduction

The Achronix 7nm Speedster®7t FPGA family is specifically designed to deliver extremely high performance for demanding applications including data-center workloads and networking infrastructure. The processing tasks associated with these high-performance applications, specifically those associated with artificial intelligence, machine learning (AI/ML) and high-speed networking, represent some of the most demanding processing workloads in the data center. The Achronix Speedster7t FPGA SerDes provides the foundation for high-speed, low power interconnects between Speedster7t FPGAs and external components.

This user guide describes the function and operation of the Achronix Speedster7t DSP SerDes PHY IP for multi-standard applications. The Speedster7t FPGA family incorporates 24 to 32 instances of high-performance, low-power, multi-standard SerDes PHY, which are highly configurable and support leading NRZ and PAM4 data center standards from 1 Gbps to 56 Gbps in raw mode. Protocols ranging from PCI-Express Gen1 to 100 Gbps Ethernet are supported.

Chapter - 2: Overview

The Speedster7t AC7t1500 incorporates thirty-two SerDes transmit/receive pairs. These high-performance, low-power, multi-standard SerDes PHYs are highly configurable and support all leading NRZ and PAM4 standards from 1 Gbps to 112 Gbps (even including standards which are currently in development).

Speedster7t FPGA SerDes PHY Highlights

- Digital DSP-based SerDes architecture, performance up to 56 Gbps while consuming ~6 mW/Gbps
- Innovative analog to digital converter (ADC) and digital signal processing (DSP) architecture supports long reach channel losses with near-end crosstalk (NEXT) noise
- Power efficient for multi-lane operation
- Configurable DSP solution supports multiple power states and loop-timed applications
- Wide tuning (1–56 Gbps), low-jitter clock generation
- Independent transmit and receive lane clocking
- DSP receiver
    - Low-power, dynamic-rate, time-interleaved ADC
    - Power scales with sampling rate and with necessary signal to noise ratio (SNR)
    - Digital clock to data recovery (CDR) quickly acquires optimum sampling times for incoming signals.
    - Blind adaptive equalizer
- Low-power transmitter
    - Low-power, voltage-mode driver able to generate 56 Gbps PAM4
    - Power scales with transmit launch swings to meet necessary channel conditions
    - Reconfigurable five-tap finite impulse response (FIR) filter
- Integrated Ethernet link training and auto-negotiation

Table 1: ***Supported Standards***

Standards	Notes
OIF Chip-Chip/Module Backplane	
• CEI-56G-USR/XSR/VSR/MR/LR-PAM4 (supports NRZ as well) • CEI-28G-VSR/SR/MR (19.6-28.1Gbps), CEI-25G-LR (19.9 – 25.8Gbps) • CEI-11G SR/MR/LR (9.95 – 11.1 Gbps) • CEI-6G SR/LR (4.976 – 6.375 Gbps)	
Ethernet Chip-Chip/Module Backplane	

Standards	Notes
• Ethernet (53.125 Gbps, 26.5625 Gbps, 25.78125 Gbps, 10.3125 Gbps, 3.125 Gbps, 1.25 Gbps)	
• 400 Gbps Ethernet — CDAUI-8	8×50 Gbps (802.3bs)
• 200 Gbps Ethernet — 200GBASE-KR4 (802.3ck)	4×50 Gbps (802.3cd)
• 100 Gbps Ethernet — 200GBASE-KR2 (802.3ck)	2×50 Gbps (802.3cd)
• 100 Gbps Ethernet — 100GBASE-KR4 & KP4	4×25 Gbps (802.3bj)
• 40 Gbps Ethernet — XLAUI, 10G-BASE-KR4	4×10.3125 Gbps (802.3ba)
• 10 Gbps Ethernet — XAUI, 10G-BASE-KR (3.125 Gbps, 10.3125 Gbps, 10.3125 Gbps)	802.3ap
• Gigabit Ethernet — 1000BASE-CX, SGMII (1.25 Gbps)	
• Synchronous Ethernet (ITU-T G.8262)	
Optical Chip-Module	
• Interlaken (3.125–12.5 Gbps, 25 Gbps) • Fiber Channel	
Computing	
• PCI Express Gen1, Gen2, Gen3, Gen4, Gen5 (2.5 Gbps, 5.0 Gbps, 8 Gbps, 16 Gbps, 32 Gbps) • USB 3.0/3.1 • Quickpath QPI / KTI / UPI	
PON	
• GPON (1.25 Gbps, 2.5 Gbps, 10.0 Gbps) • E-PON (1.25 Gbps, 2.5 Gbps, 10.0 Gbps)	
Wireless	
• CPRI (24.576 Gbps, 24.330 Gbps, 19.660 Gbps, 12.165 Gbps, 10.137 Gbps, 9.830 Gbps, 6.144 Gbps, 3.072 Gbps, 2.457 Gbps, 1.228 Gbps) • JESD204A/B/C (3.125–25 Gbps)	

SerDes Placement on Speedster7t Devices

The Speedster7t FPGAs have both PCIe and Ethernet interfaces that use the SerDes PHY. All SerDes lanes can also be used independently (raw mode). The figure below shows the placement of the interface subsystems of the Speedster7t AC7t1500 device.

Note

For raw SerDes applications (those not using Ethernet MAC or PCIe IP), bandwidth is limited to 1–56 Gbps.

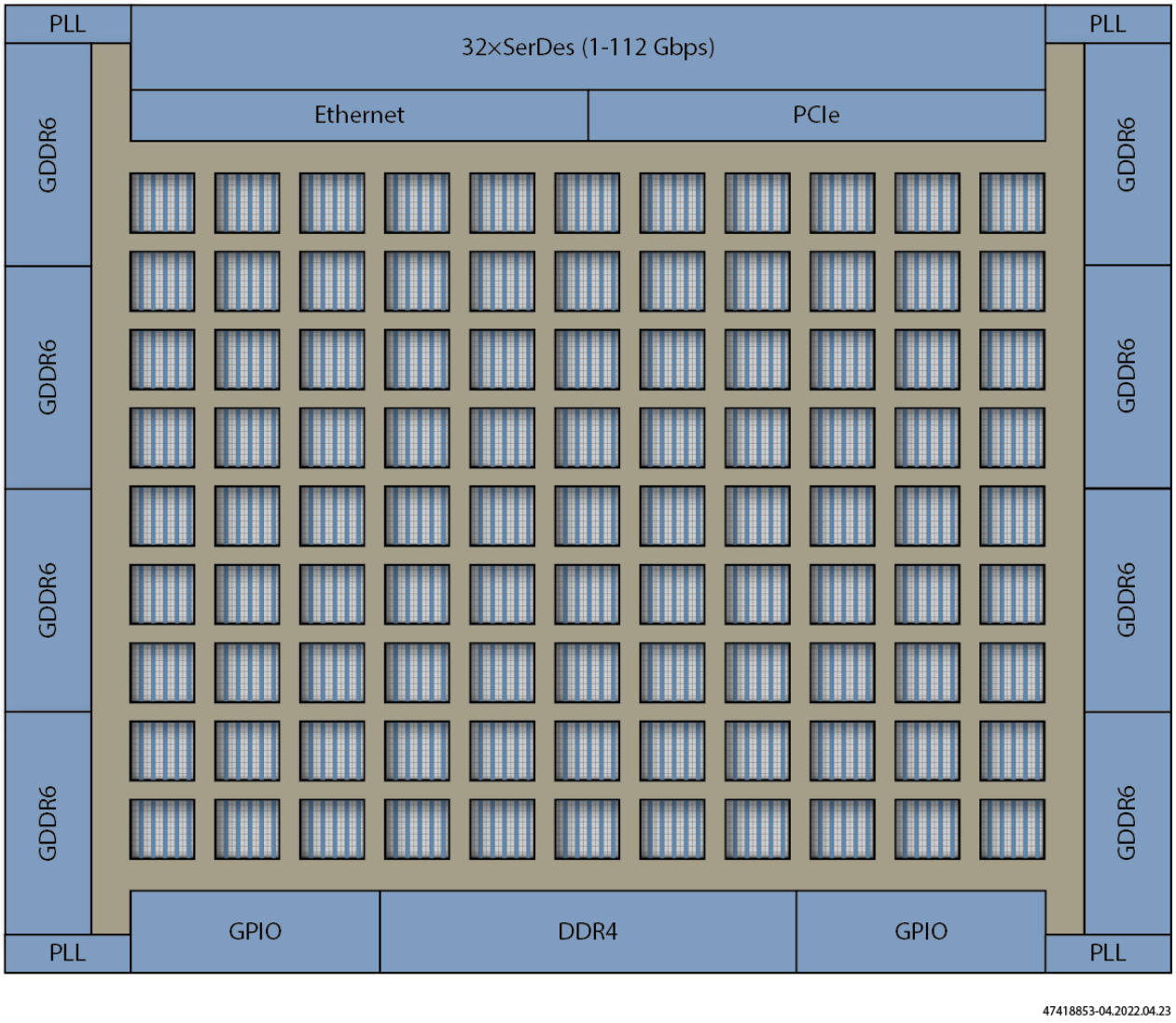


Figure 1: ***Speedster7t AC7t1500 SerDes Placement***

Eight quads of SerDes PHY are located at the north side of the chip as shown in the figure below.

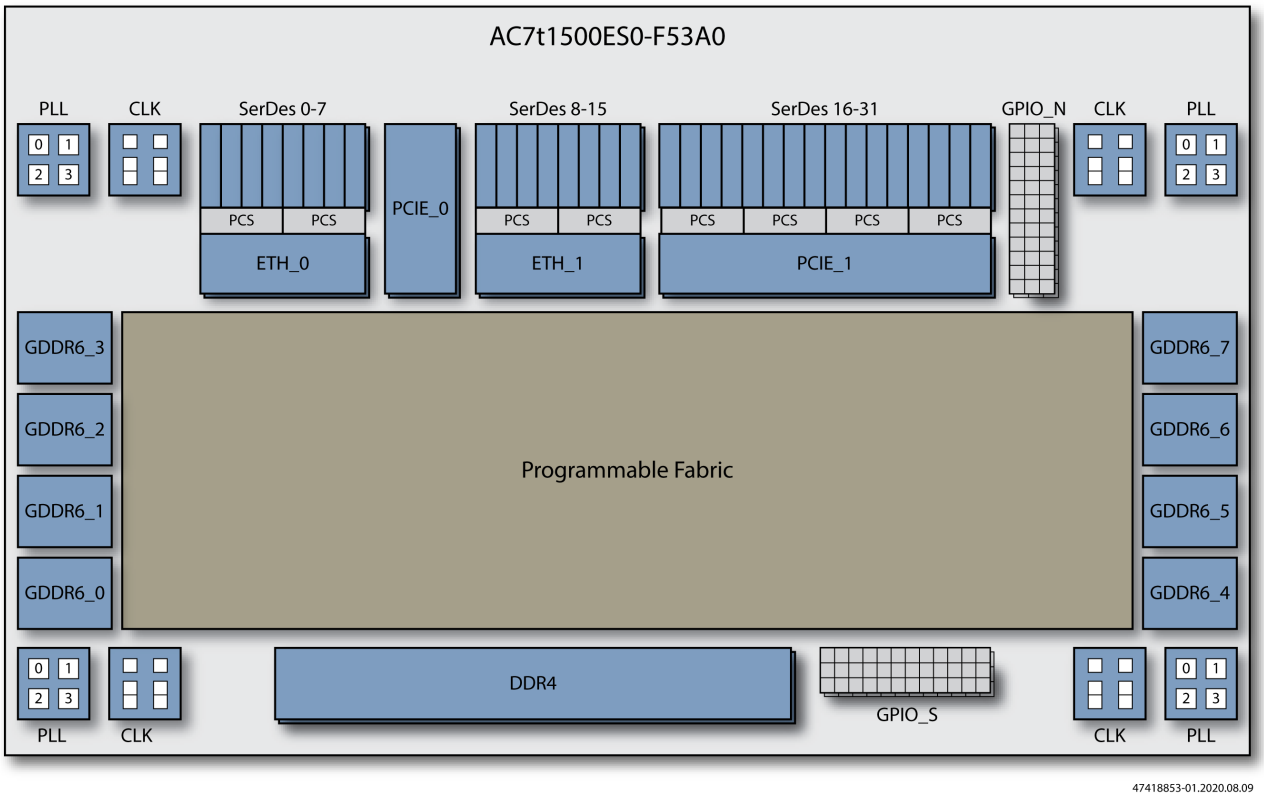


Figure 2: ***Speedster7t AC7t1500 Device Top-Level Block Diagram***

Each SerDes instance is named "Serdes [x]" and they are organized as follows:

- Sixteen lanes, Serdes [31:16], are associated with a hard PCIe controller
- Eight lanes, Serdes [15:8], are associated with a hard Ethernet controller
- Eight lanes, Serdes [7:0], are associated with both the Ethernet and PCIe controllers

When operating in conjunction with the associated hard controller, the operation of the SerDes is managed by the controller per the selected protocol. All 31 SerDes lanes can be used in raw mode. A single lane is the minimum number of lanes in a quad that can be selected for raw mode.

A quad can not be split. All lanes in a quad must be associated with an ethernet controller, PCIe controller, or raw mode. In raw mode the PCS block can be accessed directly by the user design in the FPGA fabric and the controllers are bypassed.

PMA

The SerDes PHYs are arranged as quads made up of four transmit/receive lanes and a common block. This common block is responsible for the following:

- Receiving the reference clock
- Generating clocks for the transmit and receive blocks (with 16-bit fractional multiplication factors and optional spread spectrum)
- Calibration

- Providing access to control/status registers

The transmit and receive portion of each lane contains an internal four-tap FIR filter. The transmitter provides a filter with output drivers that can deliver a 400–1200 mVp-p differential output voltage with adjustable slew rate.

The receive block receives the differential signal from the channel and applies equalization. It then determines level and symbol clock, captures the level and decodes per modulation into one or two bits per received symbol. The data is decoded, de-serialized, and driven out on data bus with a divided baud-clock.

The receiver supports AC and DC coupling with some restrictions on common-mode voltage. It incorporates robust automatically adjusted equalization including AGC, continuous-time linear equalizer (CTLE) with up to 20 dB gain tuned for key data rates, feed-forward equalization (FFE) and decision feedback equalizer (DFE) for robust recovery of data from impaired signals. Fast lock mode for EPON/GPON is also supported.

The SerDes includes debug features including multiple loopback modes and a non-destructive internal eye-scope in the receiver. The eye-scope can measure eye width, eye height and the bit error rate (BER).

Physical Coding Subsystem (PCS)

The Achronix PCS serves as the conduit for the SerDes datapath to and from the fabric. The PCS provides several options to encode, decode, bond lanes, and reorder bits. The PCS can be bypassed if none of the features are needed for the user design.

Three types of hard PCS controllers are offered within Speedster7t devices:

1. One for PCIe
2. One for Ethernet
3. One raw version for direct use by the fabric

The physical layout between the Speedster7t FPGA SerDes PHY and the PCS controllers are shown in the previous figure (see page 11).

ACE Support for the Speedster7t FPGA SerDes

ACE is the Achronix development tool for placing, routing, configuring, and debugging Achronix FPGAs. ACE works in conjunction with third-party synthesis and simulation tools to provide a complete design environment for Achronix FPGAs. The Speedster7t SerDes PHY can be conveniently configured and controlled through the ACE GUI.

For PCIe and Ethernet users, the ACE GUI has predefined options where the SerDes PHY can be configured easily to work seamlessly in PCIe/Ethernet mode. When the SerDes is used in raw mode, the SerDes PHY is configured via the ACE GUI. Configuration options are available by selecting the "Raw SerDes" option in the IP Library. The configuration can also be modified at runtime by writing to the control registers directly. The control and status registers are accessed through the JTAG port or by a NAP in the fabric. Refer to *Runtime Programming of Speedster FPGAs* (AN025) for details on how to access these registers through the JTAG port.

Chapter - 3: PMA Architecture

The Speedster7t SerDes PMA is composed of a common block and four transmit/receive blocks (lanes), and the combined assembly is referred to as a quad.

Speedster7t SerDes Block Diagram

Following is a block diagram of a single lane in a SerDes quad. A detailed sub-block-level diagram of the Speedster7t SerDes PHY can be found in Appendix II of the NDA version of this document.

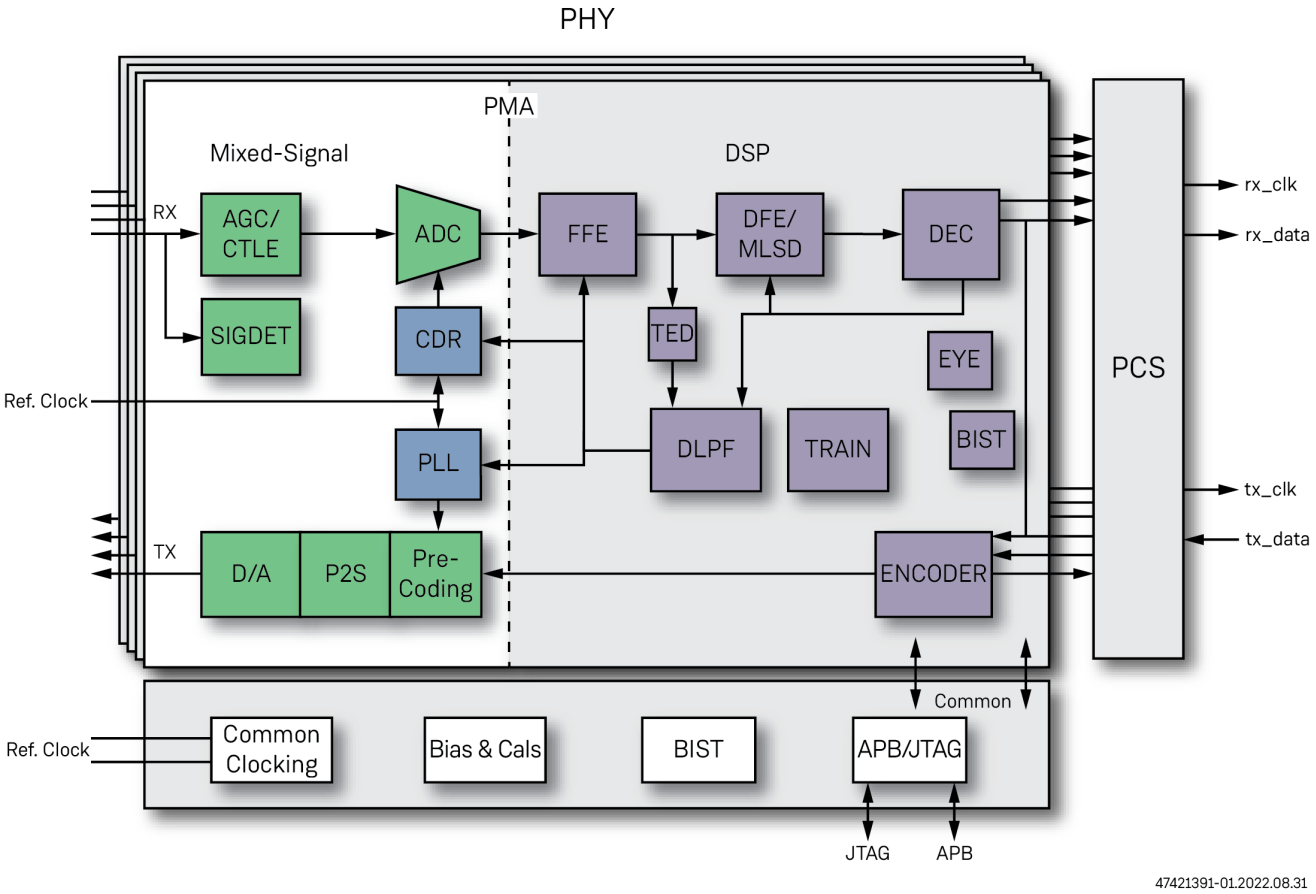


Figure 3: ***Speedster7t SerDes PHY Block Diagram***

Common Block

The common block performs calibration and contains control/status registers for the entire quad. The common block also receives the reference clock and generates a multiplied clock for the transmit and receive blocks (with 16-bit fractional multiplication factors and optional spread spectrum mode).

Calibration is performed automatically and can be invoked by register control. Details of the calibration control /status registers are provided in Control and Status Registers (see page 40).

The reference clock for a quad can be selectively sourced from external balls (SRDS_xx_REFCLK_N/P) or from a neighboring quad. The received reference clock can be selectively forwarded to neighboring quads. This option can be selected in the ACE IP configuration tool by selecting **Enable PMA Ref Clock Sharing**.

Clocking Architecture

The Speedster7t SerDes PHY clocking architecture provides a significant amount of configurability and flexibility. High quality clocks are generated and can be used directly by logic in the fabric. More details can be found in Appendix II *Speedster7t SerDes Architecture PMA*, available under NDA.

External Reference Clocks

The Speedster7t SerDes PHY reference clocks are used by the common block LC (Inductor capacitor) based PLL (LC PLL) to generate the high-speed clock, hsrefclk, that is distributed to the receive/transmit lane PLLs. The SerDes PHY includes the following external reference clock capabilities:

- Sourcing reference clock from the SRDS_N[7:0]_REFCLK external pins.
- Sourcing reference clocks from the adjacent quad for use in the local quad (limited to quads created within a single ACXIP file).

Note

More details can be found in Appendix II *Speedster7t SerDes Architecture PMA*, available under NDA.

Common LC PLL

The common LC PLL contains a low quantization noise fractional synthesizer which results in a high-degree flexibility with respect to supported data rates vs. reference clock rates. This fractional synthesis capability is also leveraged to enable spread spectrum clock generation. LC PLLs provide excellent jitter characteristics for a given power.

Lane PLLs

Each transmit and receive lane has its own local PLL to enable independent operation between transmit and receive blocks as well as between different lanes. The combination of a common LC PLL and the lane PLLs provide the benefit of flexibility across standards and configurations with the performance of an LC PLL. More details can be found in Appendix II *Speedster7t SerDes Architecture PMA*, available under NDA.

Speedster7t SerDes PHY Interface

This section provides a brief description of the signal interface between the SerDes PHY and the FPGA in PCS bypass mode. These signals are on a per lane basis. "IP-Name" refers to the name of the ACXIP file and "Lane" refers to the lane number. For example: `serdes_quad_2g5_lane0_o_eth_an_link_control`. More detailed information can be found in Appendix II *Speedster7t SerDes Architecture PMA*, available under NDA.

Table 2: *Serdes interface Ports*

Interface Signal Name	Direction	Width	Description
IP-Name_Lane[#]_o_eth_an_done	Output	1	When asserted, indicates auto-negotiation complete (link fully up, including LT and PCS). (1)
IP-Name_Lane[#]_o_eth_an_link_control[15:0]	Output	16	PCS enable, one-hot encoded. When auto-negotiation (AN) is complete, one of these signals is asserted to enable the attached PCS. (1)
IP-Name_Lane[#]_o_eth_an_link_good	Output	1	When asserted, indicates AN has negotiated a link speed and is attempting to start link training/PCS. (1)
IP-Name_Lane[#]_o_eth_an_newpage	Input	1	AN newpage received indicator. A positive edge on this signal indicates a new page has been received. The FPGA should read the associated registers, write extended page information, and assert the `next_page_loaded` bit, if needed. The FPGA might also poll the equivalent register. (1)
IP-Name_Lane[#]_i_eth_an_link_status[15:0]	Input	16	PCS link status. Must be asserted by the enabled PCS to indicate the link is up. (1)
IP-Name_Lane[#]_i_tx_elecidle[3:0]	Input	4	Pin control of the transmit electrical idle function. For PCIe applications with synchronous electrical idle requirement: 0 – I[#] transmitter in normal operating mode. 1 – I[#] transmitter placed into electrical idle. Each bit corresponds to 16 bits of transmit data: For widths up to 20 bits, only bit 0 is active. For 32 and 40 bits, bits 1:0 are active. For 64 bits, bits 3:0 are active. For all other applications including all PM-4 rates `ictl_tx_elecidle_ln[#]` should be operated as if all were a single control (i.e., logically tie all four bits together). (1)
IP-Name_Lane[#]_o_clk_rx_block	Output	1	Receive clock synchronous to the receive data.
IP-Name_Lane[#]_o_clk_tx_block	Output	1	Transmit clock synchronous to the transmit data.
IP-Name_Lane[#]_o_rx_data[127:0]	Output	128	Data output bus to the fabric. Software requires all 128 bits are connected to the fabric despite the RX data width chosen in the IP configuration.
IP-Name_Lane[#]_i_tx_data[127:0]	Input	128	Data input from the fabric. Software requires all 128 bits are connected to the fabric despite the TX data width chosen in the IP configuration.

Interface Signal Name	Direction	Width	Description
IP-Name_Lane[#]_o_rx_data_valid	Output	1	Pin output showing lock status of the receive CDR. Timed to `ck_rx_block`. 0 – CDR unlocked. 1 – CDR locked, data valid.
IP-Name_Lane[#]_o_rx_signal_detect	Output	1	Unfiltered output of the analog signal detection block.

Table Notes

1. Signal not used in PCS bypass mode.

Transmit/Receive Block

The transmit and receive portion of each lane contains an internal synthesizer which allows independent rate support for each lane.

Transmit Block

The transmit block is responsible for capturing the transmit data from the fabric using a transmit clock and sending it serially off chip. This data width is variable and is dependent upon the chosen protocol standard. Within the PMA, the data is modulated (NRZ/PAM4) and serialized before being transmitted. There is an optional 4 tap pre-emphasis filter to help improve the channel signal quality.

Transmit Equalization

The transmitter includes a five-tap digital FIR filter which boosts the relative amplitude of higher frequency data patterns by adjusting the amplitude of the bit/level being transmitted based on the values of preceding and succeeding bits.

In many cases, the PCIe/Ethernet subsystems attached to the SerDes properly adjust the transmit equalization during the establishment of the link. The transmit equalization can be adjusted manually to match channel requirements. The coefficient values can be overridden by writing new values to the TX FIR coefficients in the CSR registers space.

Further details can be found in Appendix II *Speedster7t SerDes Architecture PMA*, available under NDA.

Table 3: *Transmit Lane Features*

Parameter	Design Specification
Output Driver Voltage	• Up to 1200 mVdiff-pkpk • Programmable
Transmit Coupling	Supports: • External AC coupling • Floating DC coupled receive termination • DC coupling, for selected receive termination common modes. (1)

Parameter	Design Specification
On-chip Termination	Nominal value: 90Ω
Transmit Skew	• Skew between transmit Lanes < ± 2UI • Synchronization of transmit data ensured within multiple lanes of a By-2/By-4 as well as between multiple By-1, By-2 and By-4.
Beaconing	Output enable control for beaconing.
Receiver Detection	Fully supports PCI Express receiver detection.

Table Notes

1. More details can be found in Appendix I *Speedster7t SerDes Overview*, available under NDA.

Receiver Block

The receive channel accepts the serial differential signal from the channel and sends the data and recovered clock to the fabric. The PMA applies a continuous time linear equalization filter (CTLE) in the analog domain before the signal is digitized using a time interleaved analog-to-digital converter (TIADC). A discrete feed forward equalization filter (FFE) is used to reduce inter-symbol interference before the slicer and a decision feed back equalization (DFE) filter are used minimize decision errors. See the Receive Lane Features (see page 17) table for more details.

Table 4: *Receive Lane Features*

Parameter	Design Specification
Receive Input Range	Up to 1200 mVdiff-pkpk
Receive Equalization	• CTLE and DSP based equalization • Blind Adaptive Receiver and Equalizer (1)
Receive Coupling	Supports: • External AC coupling • DC coupled receive termination with internal AC coupling • DC coupling without internal AC coupling for selected common modes (1)
Signal Detection	• Receive signal envelope detection to support Beaconing and idle-exit detection • Digital signal detection
Spread spectrum	Fully supports recovery of signals with Spread Spectrum (0.5% down spread).

Parameter	Design Specification
Frequency Accuracy	Tolerates up to -5350 ppm to +350 ppm of data frequency inaccuracy from PHY reference clock.
On-Chip Eye Monitor	Non-destructive internal eye monitoring capability.
Lock Modes	• Supports both auto-lock to incoming data when valid data is present and manual external control of lock to data • Loss-of-Lock detector functionality
Oversampling	The receive CDR supports the oversampling mode of operation.
On-chip Termination	Nominal values: 90Ω. (1)
Reference Clock source	• Independently selectable reference clock input source per lane • PMA quad includes reference clock buffers/repeaters and distribution infrastructure

Table Notes

1. More details can be found in Appendix I *Speedster7t SerDes Overview*, available under NDA.

Clocking

Clocking frequencies and signal geometry are specified in the following table along with a summary of features of the Common Lane.

Table 5: *Common Lane Features and Requirements*

Parameter Description	Min	Typ	Max	Unit
General				
Reference clock frequency range	50		800	MHz
Frequency variation	-5350		350	ppm
Duty cycle	40	50	60	%
Rise/fall times (20-80%)			0.2	Refclk Period
Skew between positive and negative differential inputs			10	ps
Total integrated RMS phase noise for the band of frequency ranging from 12 kHz to 20 MHz			0.7	psRMS
External Reference Clock I/O Inputs				

Parameter Description		Min	Typ	Max	Unit
Input impedance	Terminated mode	40	50	60	Ω
	High-impedance mode	20k			Ω
Input Differential Voltage	PCIe	0.15			V
	LVDS	0.25		0.4	V
	LVPECL	0.525		0.95	V
Input Common Mode Voltage		0		1.2	V
Common Lane Features					
Fractional Synthesizer		• Fractional synthesizer support to eliminate integer reference clock requirements • Programmable spread spectrum generation			
Synthesizer Loop Timing		The transmit PLL supports loop timed synchronous operation.			
Synthesizer reference clock source		Independently selectable reference clock input source.			
Other clock generation		• Internal heartbeat clock for power up and beaconing. • Internal clock outputs (1)			
Reference Clock Sources		• Reference clock I/O buffer that supports a variety of input reference clock signaling (e.g., CML, LVPECL, PCIe) • On-chip CML distribution. (1)			

Table Notes

1. More details can be found in Appendix I *Speedster7t SerDes Overview*, available under NDA.

Speedster7t SerDes PMA Configuration Options

Table 6: *PMA Configurability*

Parameter	Description
Number of Lanes	1 to 16, lanes must be contiguous in a single IP.
PMA Rate Presets	Up to 8 presets.
Full Duplex Support	• Synchronized and non-synchronized full duplex operation • Independent operation of transmit and receive Lanes (power-down and reset)

Parameter	Description
Power States	Fully supports Energy Efficient Ethernet (EEE) low power states (deep sleep, fast wake). Fully supports PCIe power savings modes: • PD – Full power down on all blocks. • P2 – Lowest power/longest recovery time state. Everything powered down except Beaconing and receiver detection. • P1 – Low power/ Medium recovery time state. All P2 functionality with synthesizer powered up but transmit driver and receive paths powered down. • P0s– Some power saving / fast recovery state. All P1 functionality with receive path powered up and transmit driver powered down. • P0 – Normal full operation mode. Everything powered up.
transmit FIR	Each tap is independently programmable. (1) • C(-3) pre-cursor • C(-2) pre-cursor • C(-1) pre-cursor • C(1) post-cursor
Data Path Parallel Interface	Selectable 16/32/64/128 and 20/40 data and byte clock.

Table Notes

1. More details can be found in Appendix I *Speedster7t SerDes Overview*, available under NDA.

Speedster7t SerDes Test and Debug

Near End Serial Loopback

This procedure, operating on a quad, isolates the front end external signal and sets both the RX and TX into loopback mode.

Serial transmit to receive (max coverage of serial path) shown as path *one* in Figure 2 (see page 22). The TX data originates in the fabric and is routed back to the fabric. A TCL procedure is provided with the reference design to set the appropriate registers:

Table 7: *Near End Serial Transmit To Receive Loopback Procedure*

Item	Details
TCL library	`<DEVICE>_serdes_utils.tcl` Example: `AC7t1500ES0_serdes_utils.tcl`
Procedure	`nes_loopback <target> <quad> <enable>`
Targets	`ETHERNET_0, ETHERNET_1, PCIE_1`

Near End Parallel Loopback

This procedure, operating on a quad, sets both the RX and TX into near end parallel loopback mode and forces `rx_signal_detect` to be valid.

Parallel transmit to receive (confirms integrity of fabric-to-PMA interface) shown as path *two* in Figure 2 (see page 22). A TCL function is provided with the reference designs to set the appropriate registers:

Table 8: *Near End Parallel Transmit To Receive Procedure*

Item	Details
TCL library	`<DEVICE>_serdes_utils.tcl` Example: `AC7t1500ES0_serdes_utils.tcl`
Procedure	`nep_loopback <target> <quad> <enable>`
Targets	`ETHERNET_0, ETHERNET_1, PCIE_1`

Serdes RX Equalization

This procedure, operating on a quad or a single lane, checks for the absence of CDR lock on the lane. If CDR lock has not been asserted, the procedure activates receive equalization and checks the status bit until an acknowledge is set. The procedure times out if the acknowledge bit is never set.

RX equalization is recommended for rates above 10 Gbps. The procedure is not compatible with PCIE protocols under 10 Gbps when run in `pcs_bypass` mode. A TCL procedure is provided with the reference designs to set the appropriate registers:

Table 9: *Trigger SerDes RX Equalization Procedure*

Item	Details
TCL library	`<DEVICE>_serdes_utils.tcl` Example: `AC7t1500ES0_serdes_utils.tcl`
Procedure	`acx_pcs_trigger_rx_eq <target> <quad> <max_channel>`
Targets	`ETHERNET_0, ETHERNET_1, PCIE_1`
Max Channel	1–4

Clear RX TX State Request

This procedure, operating on a quad, clears the RX and TX request bits for the target and quad specified.

The RX and TX state request must be cleared before any state changes such as equalization. A TCL function is provided with the reference designs to set the appropriate registers:

Table 10: *Clear TX RX State Request Procedure*

Item	Details
TCL library	`<DEVICE>_serdes_utils.tcl`
Procedure	`tx_rx_state_req <target> <quad> <enable>`
Targets	`ETHERNET_0, ETHERNET_1, PCIE_1`

Loopback Modes

Four PHY Data loopback mode paths are available as detailed in the diagram below:

1. Internal serial TX to RX (maximum coverage of serial path) loopback. Data is generated in and looped back to the fabric. The CTLE data bypasses the FIR and IIR filters. Referred to as NES_loopback. Shown as path 1
2. Internal parallel TX to RX. Data is generated in and looped back to the fabric. Referred to as NEP_loopback. Shown as path 2.
3. External serial RX to TX. Untimed for continuity tests at low rates by external sources. Shown as path 3.
4. External parallel RX to TX. Transmits received data with respect to the device. Used to characterize the serdes for compliance and bit error rate. Shown as path 4.

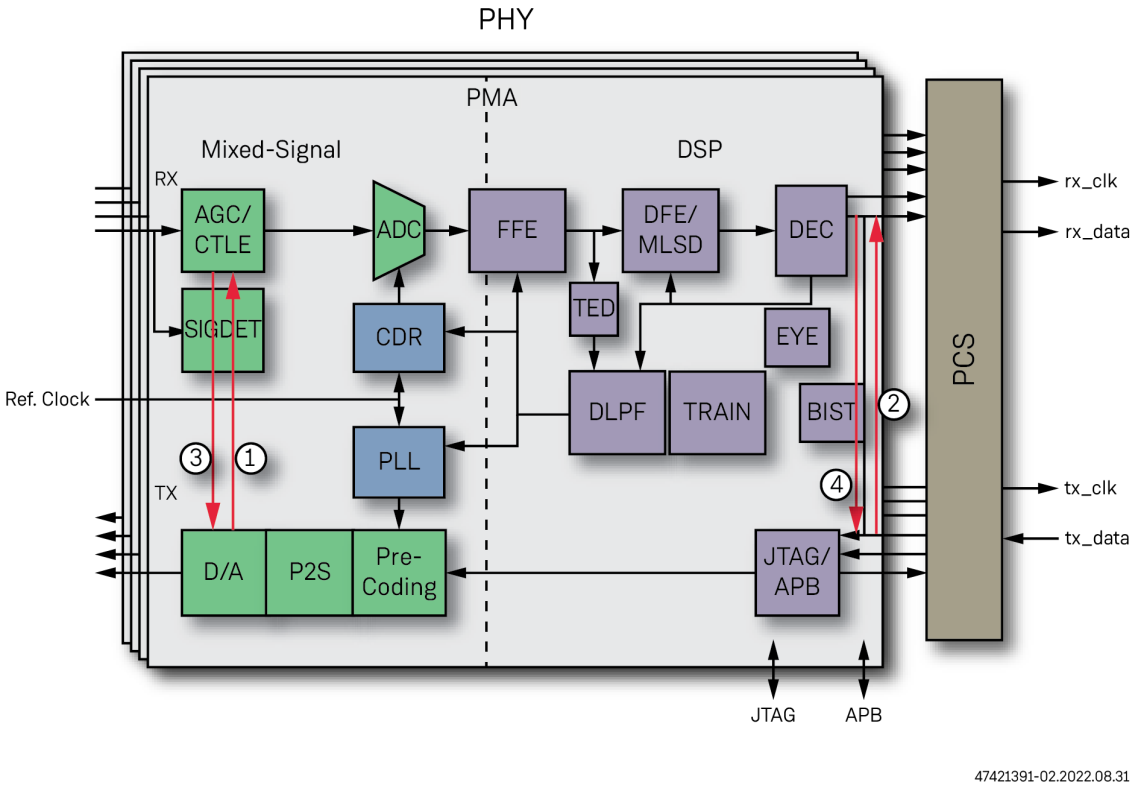


Figure 4: *Speedster7t SerDes PHY Data Loopback Modes*

Chapter - 4: Speedster7t PCS Block

Introduction

The Achronix PCS serves as the conduit for the SerDes datapath to and from the fabric. The PCS provides several options to encode, decode, bond lanes, and reorder bits. The PCS can be bypassed if none of the features are needed for the user design.

Three types of hard PCS controllers are offered within Speedster7t devices:

1. One for PCIe
2. One for Ethernet
3. One raw version for direct use by the fabric

The raw SerDes PCS controller is covered in this section.

The Achronix PCS can support the rates and performance requirements for other protocols including CPRI, JESD204x and Interlaken. A soft controller is required in the fabric for these protocols. The Achronix PCS and PMA can be thought of as a unit. Some features in the PMA are controlled through the Achronix PCS. All lanes are configured as a quad, but can be used as a single lane or combined with other quads for a configurations that spans 16 lanes. Four quads are used to provided 16 high-speed lanes, two quads for eight lanes, and one quad for a four lane solution as shown in the following figure.

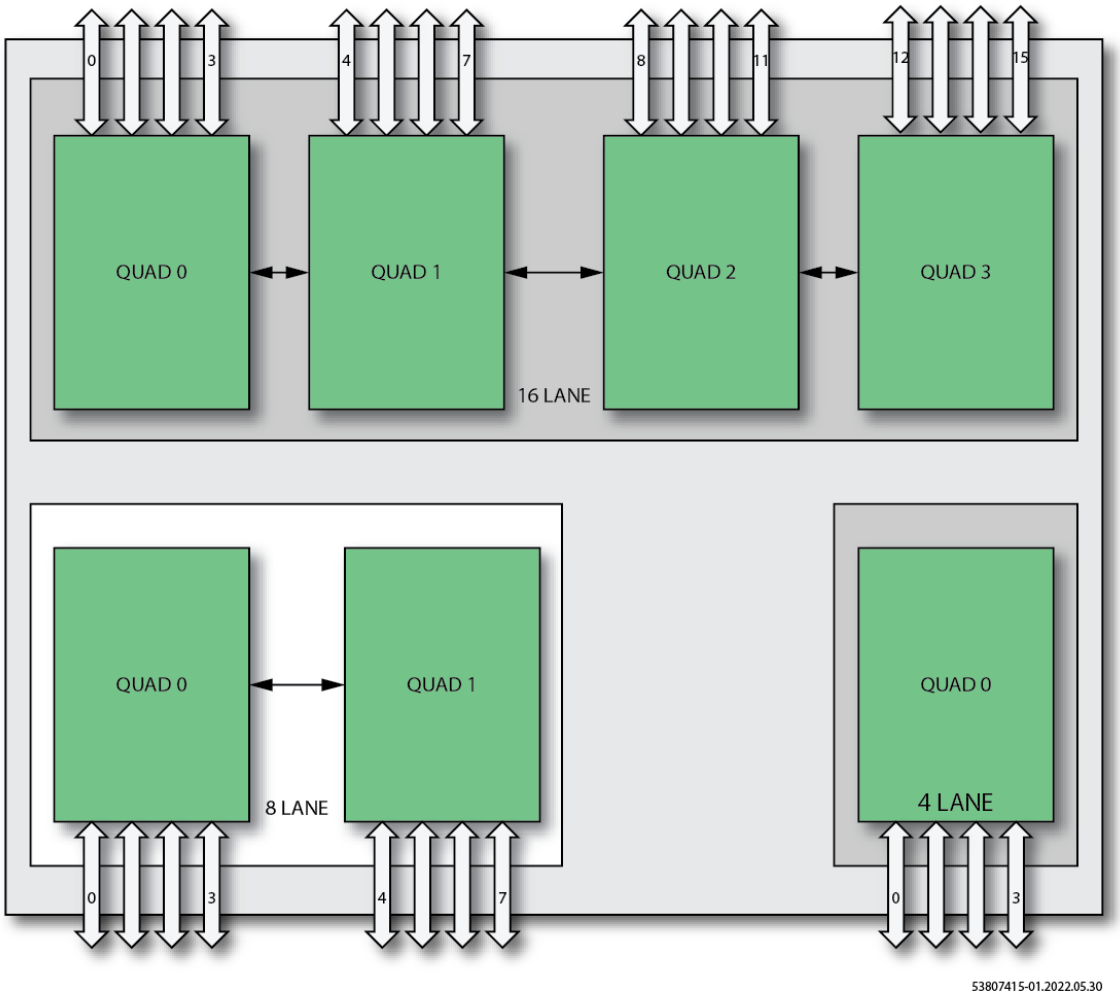


Figure 5: ***SerDes Configurations***

PCS Modes

The PCS has multiple modes:

- Ethernet Mode: The PCS is connected to a hard Ethernet MAC. This is covered in the *Speedster7t Ethernet User Guide* (UG097).
- PCIe Mode: The PCS is connected to the hard PCIe Gen5 subsystem. This is covered in the *Speedster7t PCIe User Guide* (UG098) (support account required).
- Raw SerDes Mode: The PCS routes data directly to and from the fabric. In Raw Mode, there are three options for the datapath. These options can be found in the ACE GUI under the **PCS Data Path** selector.
- Pipe Mode: The data is routed through a PCIe PIPE to enable soft controllers in the fabric, up to GEN4
- PCS Bypass Mode: The data is sent directly to/from the PMA to the fabric. All word alignment must be accounted for in the fabric.
- Gearbox Mode: The data is routed through the gearbox logic. The gearbox logic supports 66b/64b synchronous mode, 66b/64b asynchronous mode, and 67b/64b synchronous mode.

The following figure shows a high-level block diagram of the PCS and the RX/TX data multiplexing.

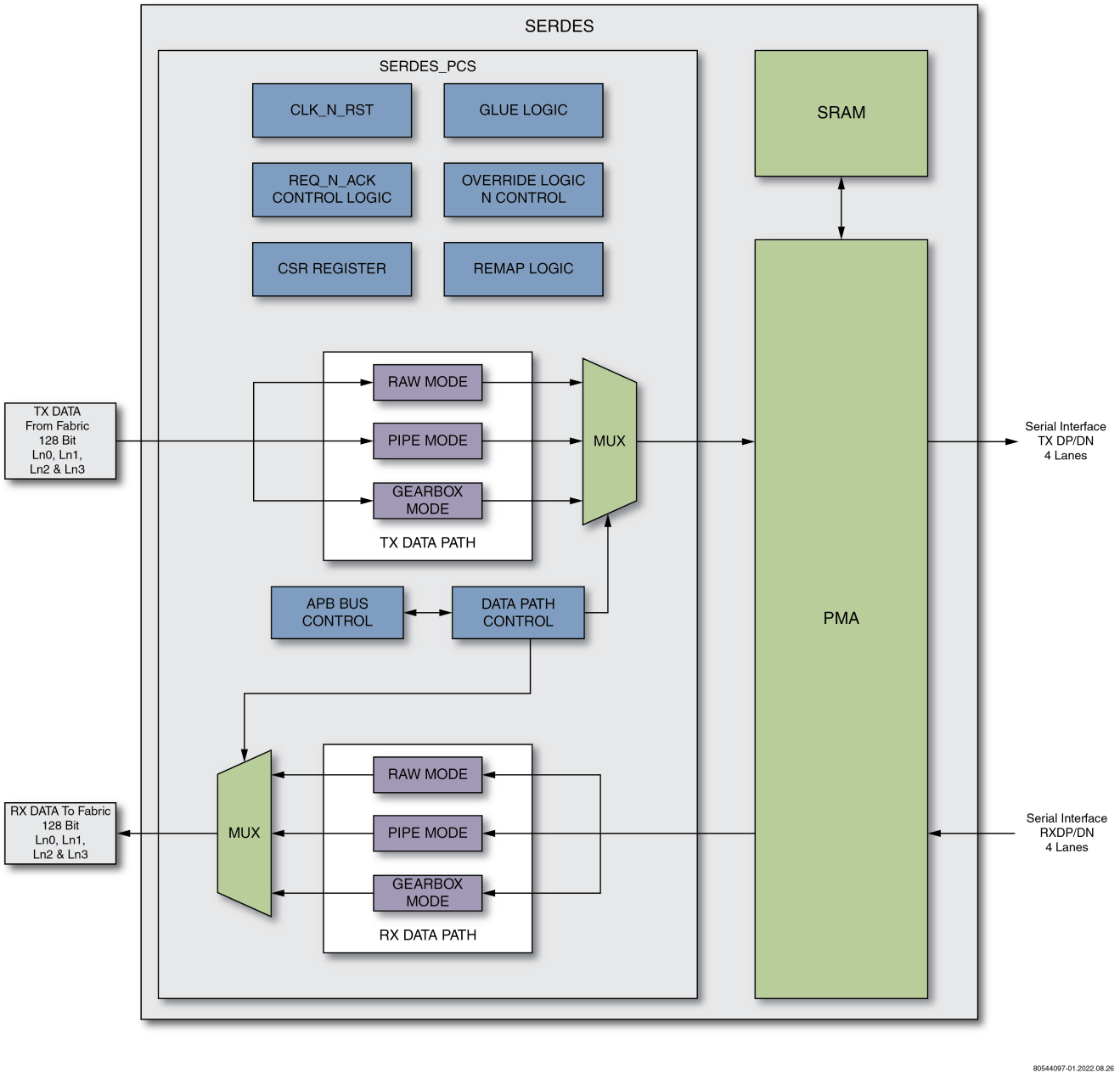


Figure 6: ***PCS Modes***

PCS Features

- 16/20/32/40/64/128 bit FPGA data path support
- PCIe protocol via the PIPE specification for implementation of a soft PCIE controller in the fabric:
  - 40-bit internal data path support for 8b/10b encoder path for PCIE Gen1/Gen2
  - Comma detection and byte/word alignment only for PCIe protocol 8b/10B alignment
  - 32-bit internal data path support for 128/130b encoder/decoder for PCIe Gen3/Gen4
  - Receive elastic FIFO is only for PIPE mode for clock compensation
- Support for bypassing the PCS for hard IP such as Ethernet Controller in the fabric
- Support for 66b/64b CAUI gearbox in both synchronous and asynchronous mode.
- Support for 67b/64b gearbox in synchronous mode only
- Protocols supported by the PMA via raw mode include:
  - Ethernet 1G/10G/25G/50G/100G/XAUI
  - CPRI
  - JESD204A/B/C
  - SyncE
  - Interlaken
- The PCS SerDes supports lane bonding up to 16 lanes using a quad as the basic lane bonding block

PCS SerDes Block Diagram

This section outlines the high-level hardware architecture block diagram of a quad lane SerDes (figure below). The PCS SerDes IP consists of the following main functional blocks:

- PCS (transmitter and receiver)
- PMA (quad SerDes)

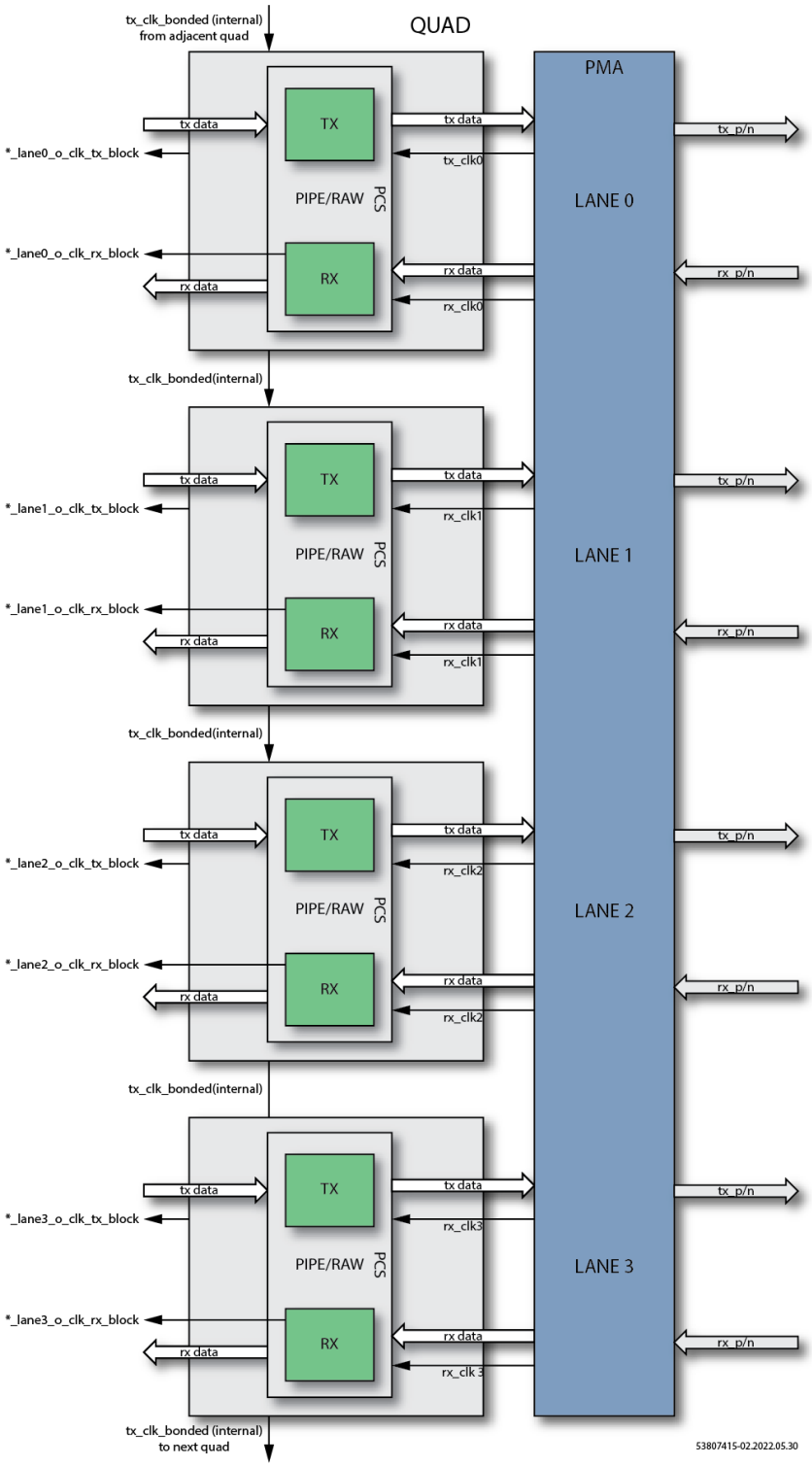


Figure 7: *Quad Transmit/Receive Path*

Reference Clock Sharing

The figure below details sharing the reference clock between quads. The reference clocks can only be shared within a single ACXIP and must be within a single subsystem (ETHERNET_0, ETHERNET_1, or PCIE_1). An AXCIP is a configuration file created in ACE defining an IP configuration such as SerDes.

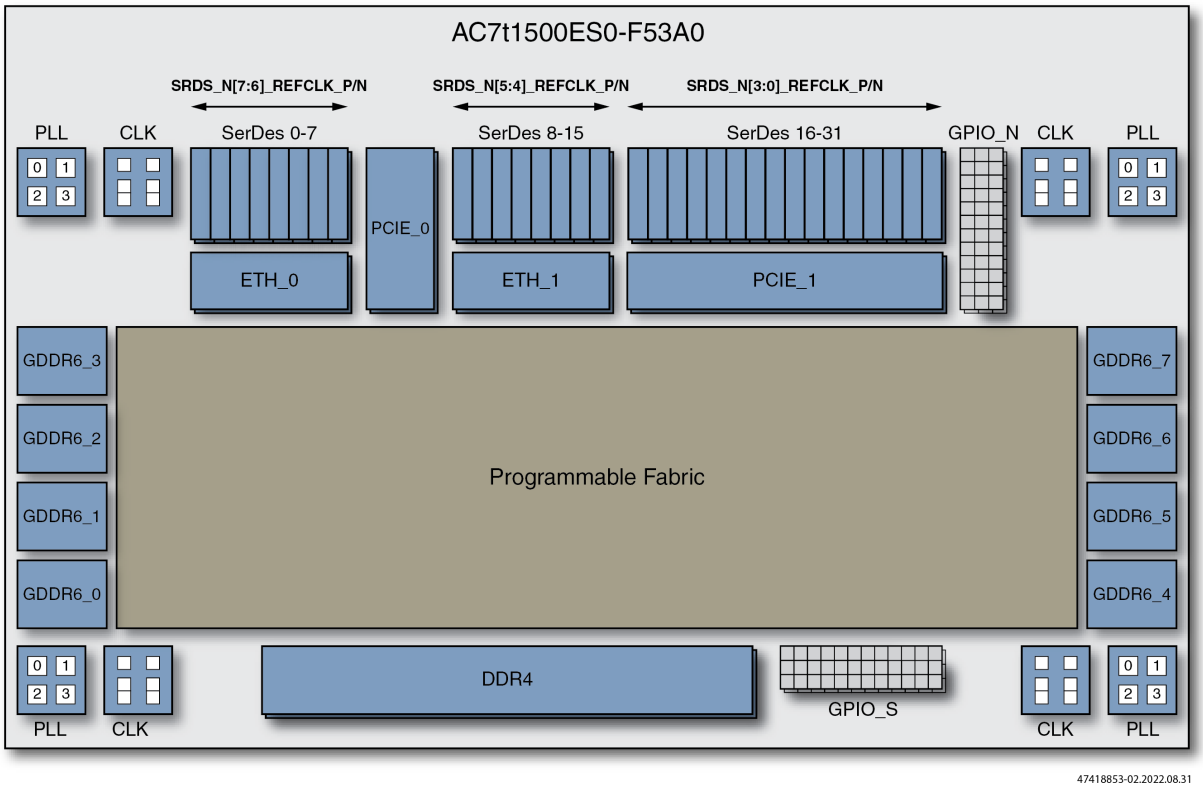


Figure 8: ***Reference Clock Sharing***

The following table details which lanes belong to each subsystem. Reference clock sharing is limited to lanes within a subsystem.

Table 11: ***Reference Clock Sharing Table***

Lanes	Subsystem	Reference Clock
3:0	ETHERNET_0	SRDS_N6_REFCLK_P
7:4	ETHERNET_0	SRDS_N7_REFCLK_P
11:8	ETHERNET_1	SRDS_N4_REFCLK_P
15:12	ETHERNET_1	SRDS_N5_REFCLK_P
19:16	PCIE_1	SRDS_N0_REFCLK_P
23:20	PCIE_1	SRDS_N1_REFCLK_P

Lanes	Subsystem	Reference Clock
27:24	PCIE_1	SRDS_N2_REFCLK_P
31:28	PCIE_1	SRDS_N3_REFCLK_P

PCS Transmit Architecture

The transmitter consists of the following four data paths. Each data path supports a particular encoding scheme which caters to a set of protocols.

- Transmit mux logic
- 8b/10b encoding path (PCIe Gen1 and Gen 2)
- 128/130b encoding path (PCIe Gen3 and Gen 4)
- Transmit gearboxing (66b/64b and 67b/64b)
- PCS transmit direct path (16/20/32/40/64/128 bit)

Transmit MUX Logic

The transmit MUX logic block performs the mode-dependent bit multiplexing. The supported modes are:

- Gearboxing
- PIPE (32 bit)
- Raw (128 bit)
- PRBS Generator

8b/10b Encoder

The 8b/10b encoder generates 10-bit code groups from 8-bit data and single-bit control input. The encoder uses the code group mapping specified in IEEE 802.3 clause 36. If the fabric interface uses a 32-bit data path, then four 8b/10b encoders are cascaded to produce a 40-bit code group output to the PMA for serialization. The 10-bit encoded output `tx_dataout[9:0]` maps to bits `{jhgf iedcba}` per the labeling used in IEEE 802.3-2005 clause 36. The encoder operates with four-byte data widths.

This block supports PCIe Gen1 and Gen2 in PIPE mode.

128b/130b Encoder Path

The 128b/130b encoder is specifically targeted at PCIe gen3/gen4. The interface is compliant with PIPE 5.1.

PCS Receive Architecture

The receiver consists of the following four data paths. Each data path supports a particular encoding scheme which caters to a set of protocols.

- Receive demux logic
- 8b/10b decoding (PCIe Gen1 and Gen 2)
- 128/130b decoding (PCIe Gen3-Gen5)
- 64b/66b and 64b/67b Gearboxing
- PCS receive direct path
- PRBS checker

Receive DeMUX logic

The receive deMUX logic block performs the mode-dependent bit demultiplexing. The supported modes are:

- Gearboxing (66b/67b)
- PIPE (32-bit)
- Raw (128 bit)

8b/10b Decoder Path

This path is used for implementing 8b/10b decoder-related receive path blocks. This path contains the following blocks:

- Comma detect and symbol alignment
- 8b/10b decoder
- Receive 8b/10b elastic buffer

Comma Detect and Symbol Alignment

The symbol alignment block can be configured to support a variety of standards by programming various characters. Currently supported standard are PCIe (Gen 1/Gen 2).

Symbol alignment is performed in automatic alignment mode only, using alignment and sequence characters for identifying the correct symbol boundary in the received data-stream. Attributes for alignment and sequence detect symbols are specified to be 10 bits wide. This block offers a programmable comma pattern with sequence options to fit any protocol or custom scenario.

8b/10b Decoder

The 8b/10b decoder generates 8-bit code groups and 1-bit control from 10-bit encoded data. It uses the code group mapping specified in IEEE 802.3 clause 36. If the fabric interface uses a 32-bit data path, then four 8B/10B decoders are cascaded to produce 32-bit data to the fabric. The decoder handles various error conditions reported on a per byte lane basis.

The running disparity should change polarity when a 6-bit or 4-bit code word with an unequal number of 0s and 1s is received. If reception of a 6-bit or 4-bit code word with an unequal number of 0s and 1s does not change the running disparity, a disparity error is asserted for the corresponding byte.

Any 10-bit code word that is not present in tables 36-1, 36-2 of the IEEE 802.3-2005 specification are considered as an invalid code word. In addition, the 11 code words corresponding to the K characters are programmable to be flagged as invalid code words. If the 10-bit code word happens to be present in tables 36-1 or 36-2, but corresponds to the wrong column (per the currently running disparity calculation), a wrong column indication is asserted.

Disparity and code errors are not mutually exclusive. However, code error and wrong column indication are mutually exclusive. For PCIe, if a code error and disparity error is detected on the same byte, the `pipe_rxstatus` is encoded to indicate a code error.

Receive 8b/10b Elastic Buffer

The 8b/10b elastic FIFO is used to synchronize the received data from the PMA-recovered clock to a system clock or local clock (typically transmit clock) in the 8b/10b encoding environment. This block operates in a 40-bit mode and provides a very generic programmable skip character detection.

The elastic FIFO compensates for the frequency offset by adding or deleting pre-configured skip characters from the received data stream. The elastic FIFO provides an indication that skip characters were added or deleted to the downstream logic. For PCIe, the elastic FIFO also includes the appropriate status encoding to indicate add /delete operation.

128b/130b Decoder Path

The 128b/130b decoder path is specifically targeted at PCIe Gen3-Gen4. The interface is compliant to PIPE 5.0 and consists of a block synchronizer and receives the Gen3-Gen4 elastic buffer.

Block Synchronizer

PMA parallelization occurs at arbitrary word boundaries. Consequently, the parallel data from the receive PMA CDR must be realigned to meaningful character boundaries. The PCI Express 4.0 base specification outlines that the data is formed using 130-bit blocks, with the exception of SKP blocks. A SKP ordered set can be 66, 98, 130, 162, or 194 bits long. The block synchronizer searches for the electrical idle exit sequence ordered set or skips (SKP) ordered set to identify the correct boundary for the incoming stream and to achieve block alignment. The block is realigned to the new block boundary following the receipt of a SKP ordered set, as it can be of variable length.

Receive Gen3-Gen5 Elastic Buffer

This elastic buffer compensates for small clock frequency differences between the recovered clock and the local clock domains by inserting or removing SKP symbols in the data stream to keep the FIFO from going empty or full respectively for PCIe Gen3/4/5 systems.

The PCI Express 4.0 base specification defines that the SKP ordered set (OS) can be 66, 98, 130, 162, or 194 bits long. The SKP OS has the following fixed bits: 2-bit Sync, 8-bit SKP END, and a 24-bit LFSR for a total of 34 bits. This block adds or deletes the 4 SKP characters (32 bits) to keep the FIFO from going empty or full, respectively. If the FIFO is nearly full, it deletes the 4 SKP characters (32 bits) by disabling write whenever a SKP is found. If the FIFO is nearly empty, the design waits for a SKP OS to start, then stops reading the data from the FIFO, and inserts a SKP in the outgoing data.

64b/66b and 64b/67b Bit Gearboxing

The receive gearbox is used to adapt the PMA data width to the width of the PCS-fabric interface. It supports different ratios (PMA interface width: fabric interface width) such as 64b/66b and 64b/67b. There are two modes supported:

- Synchronous mode – Supported for both 66 and 67 bit mode. The gearbox stalls the data throughput by de-asserting `data_valid` for one clock cycle to match the ratio. The fabric receives a `data_valid` signal along with the data which must be monitored.
- Asynchronous mode – Supported only for 66 bit mode. The gearbox provides the data on each clock cycle of the receive clock. Data is valid on every clock cycle in asynchronous mode.

Gearbox Mode

The transmit gearbox is used to adapt the PCS data width to the width of the PCS-PMA interface. The gearbox supports different ratios (fabric interface width: PMA interface width) such as 66b/64b (synchronous and asynchronous) and 67b/64b (synchronous). This block can either operate seamlessly based on two clocks with different rates 66b/64b, or operate on a single clock with wait states added during the gearbox wrap around time for 67b/64b (synchronous mode).

Gearboxing Implementation

Clocking and Usage

The following figure shows the clock and data path for a gearboxing implementation in the PCS for a quad.

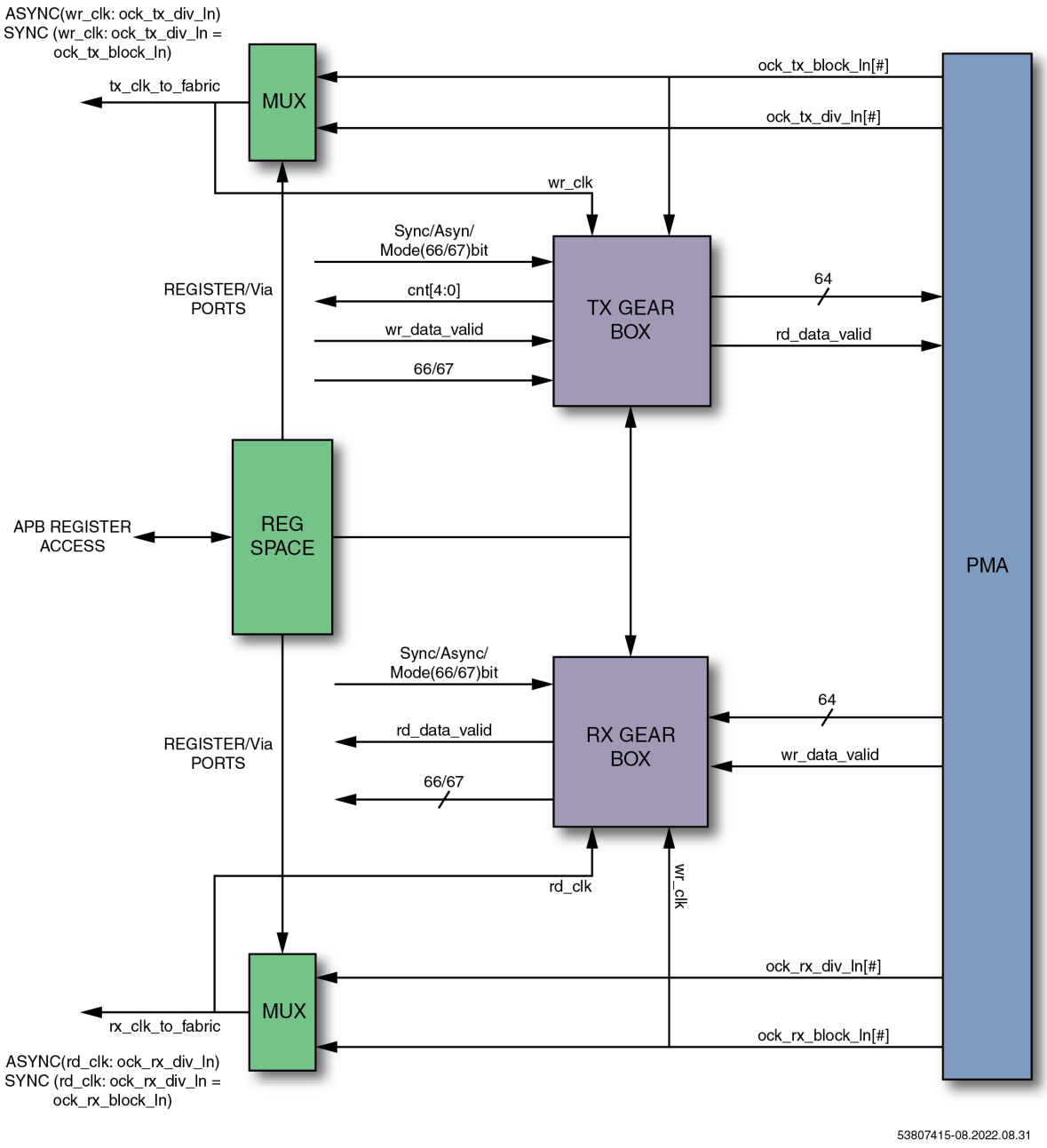


Figure 9: *Gearbox Implementation*

Gearbox Ports Mapping

Table 12: *Gearbox Transmit Data Port Mapping*

Port	Gearbox Port Name/Usage	Width
i_tx_data[66:0]	67/66 bit data input.	67
i_tx_data[67]	Enable 66-bit asynchronous gearbox.	1
i_tx_data[68]	Enable 66-bit synchronous gearbox.	1
i_tx_data[69]	data valid.	1
i_tx_data[70]	Enable 67-bit synchronous gearbox.	1
i_tx_data[77:71]	7-bit synchronous counter.	7

Table 13: *Gearbox Receive Data Port Mapping*

Port	Gearbox Port Name/Usage	Width
o_rx_data[66:0]	67/66 bit output data.	67
o_rx_data[67]	Indicates skip missed for synchronous gearbox.	1
o_rx_data[68]	Rx data valid.	1

PIPE Mode

The PIPE mode enables access to the PCS PIPE 5.1-compliant interface. The PCS layer supports PCIe Gen4 in PIPE mode using a soft controller in the fabric. The following diagram details the various supported combinations of clock and data width when the PCS supports PCIe.

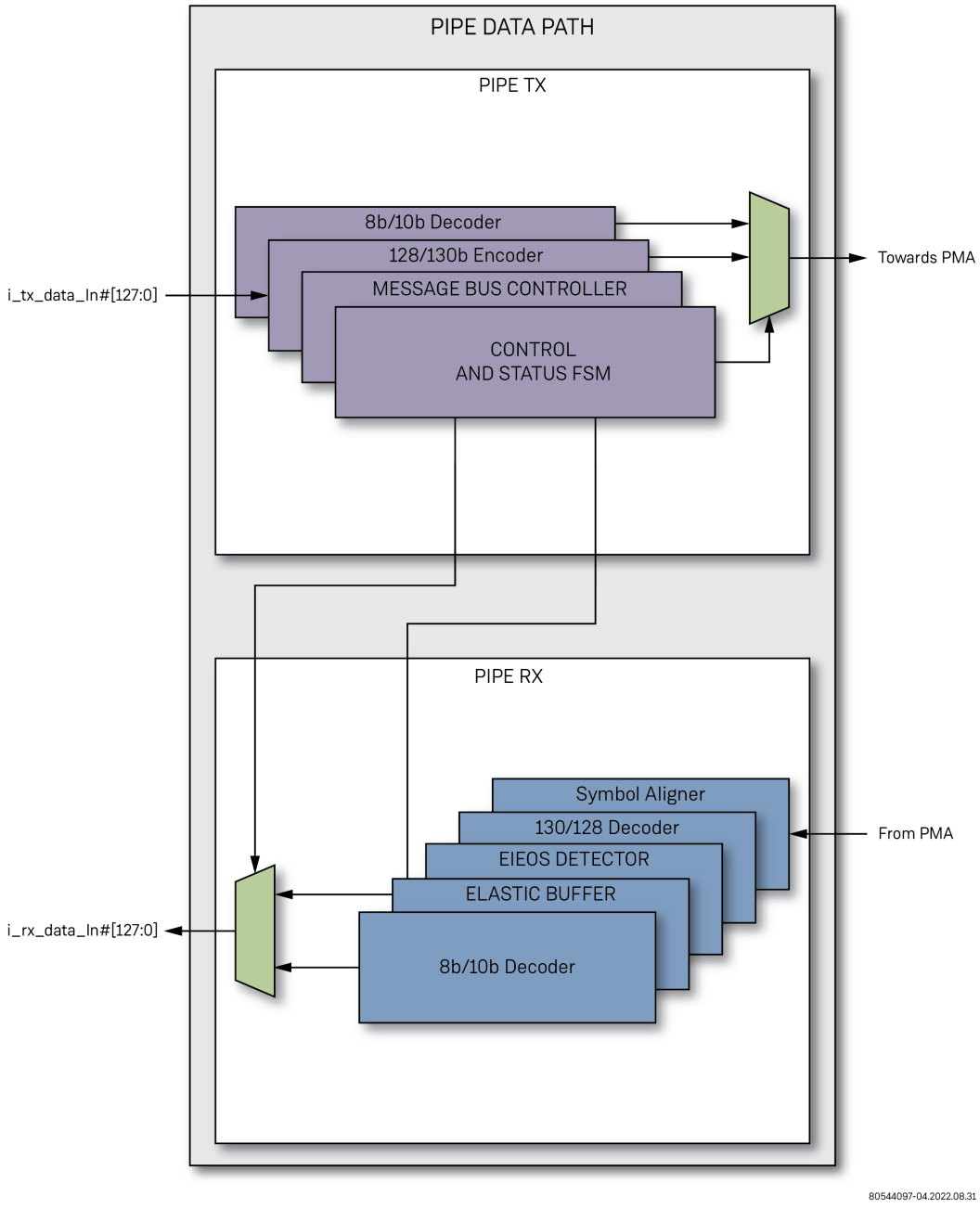


Figure 10: ***PCIe PIPE Mode Clock and Data Width***

Rates

The PCS layer includes the glue logic to switch the PMA data width to 32-bit mode and program the final rate bits for PCIe gen3/gen4 operation. The following table lists the various supported combinations of clock and data width when the PCS supports PCIe.

Table 14: *PCIE Generation with PCLK Rates and Supported Data Widths*

PCIe Mode	PCLK	PMA Data Width
2.5 Gbps Gen1	62.5 MHz	40 bits
5.0 Gbps Gen2	125 MHz	40 bits
8.0 Gbps Gen3	250 MHz	32 bits
16.0 Gbps Gen4	500 MHz	

PIPE Ports Mapping

Refer to PIPE spec 5.1 for further details.

Table 15: *PIPE Mode Transmit Data Port Mapping*

Port	PIPE Port Name	Width	Description
`i_tx_data[31:0]`	`TxData`	32	Transmit PIPE Data.
`i_tx_data[70:67]`	`TxElecidle`	4	Transmit electrical idle forces tx_p/n to idle state.
`i_tx_data[72:71]`	`TxSubsample`	2	Not Supported.
`i_tx_data[73]`	`TxBeaconEnable`	1	Enable transmit of a Beacon. 1 Beacon transmit is enabled, 0 Beacon transmit is disabled.

`i_tx_data[77:74]`	`TxDataK`	4	These signals indicate the type of characters on the TxData bus. A 1'b0 indicates a data character; while a 1'b1 indicates a control character.
			`i_tx_data[74]` – `TxData[7:0]`
			`i_tx_data[75]` – `TxData[15:8]`
			`i_tx_data[76]` – `TxData[23:16]`
			`i_tx_data[77]` – `TxData[31:24]`

`i_tx_data[78]`	`TxDataValid`	1	TxDataValid qualifies the data on the TxData bus and is active high.
`i_tx_data[80:79]`	`TxSyncHeader`	2	For Gen3 operation. Specifies the receive block type: `2'b01`: Ordered Set Block `2'b10`: Data Block These signals can be grounded if the design is not targeting Gen3.

Table 16: ***PIPE Mode Transmit Data Port Mapping (cont.)***

Port	PIPE Port Name	Width	Description
i_tx_data[81]	TxStartBlock	1	Allows the MAC to tell the PHY the starting byte for the 128b block, active high. The starting byte for a 128b block must always start with byte 0 of the data interface.
i_tx_data[82]	TxCompliance	1	Used when transmitting the PCI Express compliance pattern, active high. This signal is sampled on TxDataValid.
i_tx_data[83]	SRISEnable	1	Not Supported.
i_tx_data[91:84]	M2PMessageBus	8	The MAC multiplexes command, and required address, and any required data for sending read and write requests to access PHY PIPE registers and for sending read completion responses and write ack responses to PHY initiated requests. Refer PIPE spec 5.1 message bus section for full definition.
i_tx_data[92]	RXEiDetectDisable	1	Not Supported.
i_tx_data[93]	TxCommonModeDisable	1	Not Supported.
i_tx_data[94]	PclkChangeAck	1	Not Supported.
i_tx_data[95]	AsyncPowerChangeAck	1	Not Supported.
i_tx_data[96]	TxDetectrx/Loopback	1	Used to tell the PHY to begin a receiver detection operation or to begin loopback, active high.
i_tx_data[100:97]	PowerDown	4	Power the transceiver up or down. Power states: 4'b0000 – P0 normal operation. 4'b0001 – P0 low recovery time latency, power saving state. 4'b0010 – P1 longer recovery time latency, lower power state. 4'b0011 – P2 lowest power state. 4'b0100 – PD (full power down).
i_tx_data[103:101]	TxRate	3	Control the link symbol rate: 3'b000 – 2.5 GB/s symbol rate. 3'b001 – 5.0 GB/s symbol rate. 3'b010 – 8.0 GB/s symbol rate. 3'b011 – 16.0 GB/s symbol rate.
i_tx_data[106:105]	Width	2	Fixed to 32 bit.

Port	PIPE Port Name	Width	Description
i_tx_data[111:107]	PclkRate	5	Not Supported.
i_tx_data[112]	RxStandby	1	Not Supported.
i_tx_data[113]	MsgbusRstn	1	Reset the message bus, active low.

Table 17: ***PIPE Mode Receive Data Port Mapping***

Port	PIPE Port Name	Width	Description
o_rx_data[31:0]	RxData	32	Received PIPE data.
o_rx_data[67-32]	Unused.	36	Unused.
o_rx_data[68]	RxValid	1	Indicates symbol lock and valid data on RxData and RxDataK and further qualifies RxDataValid when used. Active high.
o_rx_data[69]	RxElecidle	1	Receiver detection of an electrical idle. Indicates detection of a beacon in PCIe mode. Active high.
o_rx_data[70]	PHYStatus	1	Communicates the completion of a stable clock after any change in state (power state transition, rate change, reset, or receiver connection). Active high.
o_rx_data[74:71]	RxDataK	4	Data/Control bit for the symbols of receive data. A value of 1 indicates a control byte. o_rx_data[71] = RxData[7:0] o_rx_data[72] = RxData[15:8] o_rx_data[73] = RxData[23:16] o_rx_data[74] = RxData[31:24]
o_rx_data[75]	RxDataValid	1	Indicates RxData is valid, active high.
o_rx_data[78:76]	RxStatus	3	Receive status and error from the PHY to the MAC. 3'b000 – Receive Data Ok. 3'b001 – SKP added. 3'b010 – SKP removed. 3'b011 – Receiver detected. 3'b100 – 8b/10b, 128b/130b decode error. 3'b101 – Elastic buffer overflow. 3'b110 – Elastic buffer underflow. 3'b111 – Receive disparity error.
o_rx_data[79]	RxStandbyStatus	1	Not Supported.
o_rx_data[81:80]	RxSyncHeader	2	Sync header for the MAC to use with the next 128b block. The MAC reads this value when the RxStartBlock signal is asserted.

Port	PIPE Port Name	Width	Description
o_rx_data[82]	RxStartBlock	1	Used at the 8.0 GB/s and 16 Gb/s PCI Express signaling rates, active high. The signal allows the PHY to tell the MAX the starting byte for a 128b block. The starting byte for a 128b block must always start with byte 0 of the data interface.
o_rx_data[90:83]	P2MMessagebus	8	The PHY multiplexes command, required address, and any required data for sending read and write requests to access MAX PIPE registers and for sending read completion responses and write ack responses to MAX-initiated requests. Refer to the PIPE 5.1 spec for message bus definitions.
o_rx_data[92:91]	DataBusWidth	2	Fixed to 32 bit.
o_rx_data[93]	RefClkRequired	1	Not Supported.
o_rx_data[94]	PclkChangeOk	1	Not Supported.

Note

It is the responsibility of the user to implement the deskew and CDC-related implementation in the fabric.

Chapter - 5: Speedster7t Serdes Register Map

Introduction

In order to support configuration and status monitoring of the SerDes subsystem, the control and status registers (CSR) are located within the overall device memory map. The memory area can be accessed through a number of mechanisms, including:

- PCIe subsystem
- JTAG port
- `NAP_AXI_MASTER` instantiated in the programmable fabric

For more general information about the device register space, see *Runtime Programming of Speedster FPGAs (AN025)*.

Control and Status Registers

The Speedster7t FPGA control and status registers (CSR) are programmable and observable. This includes registers in the SerDes PHY. The key organizational principal of the register space is the subdivision of registers into common and lane-specific registers. Common registers apply to both the common lane of the quads and SerDes PHY functions that are shared across all of the lanes. Lane-specific registers contain the configuration and status information for a specific lane (e.g., Lane0)

CSRs serve a number of different functions in the design, including:

- Configuration registers, where the register provides the only means to set up advanced functionality or behaviors within the SerDes PHY
- Status registers, where the register provides the only means of observing a specific status signal
- SerDes PHY/Fabric interface to override registers, allowing the SerDes PHY interface outputs and/or inputs to be overridden to permit system-level debug
- Observability of registers for debug (additional observability is possible via the SerDes test/trace bus)

Global Address Space

CSR Addressing

The Speedster7t device supports a 42-bit address region. Within that address region, bits [41:34] define the top level spaces. The CSR space is defined as shown in the table below.

Table 18: *Speedster7t FPGA Global Address Map*

Address Bit	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	…	0
CSR Space	0	0	1	0	0	0	0	0	Target ID						IP ID				Register Address			

Target ID Addressing

The raw SerDes lanes are a subset of the Ethernet and PCIE target IDs. Each SerDes quad is associated with one of the following Target IDs (address bits [33:28]):

Table 19: *SerDes Target ID Address Map*

Target ID							Description
CSR Space	**33**	**32**	**31**	**30**	**29**	**28**	
Ethernet 0	0	1	1	0	1	1	Ethernet 0, SerDes 0 (lanes[3:0]) Ethernet 0, SerDes 1 (lanes[7:4])
Ethernet 1	0	1	1	1	0	0	Ethernet 1, SerDes 0 (lanes[11:8]) Ethernet 1, SerDes 1 (lanes[15:12])
PCIE 1	0	1	1	0	0	1	PCIE 1, SerDes 0 (lanes[19:16]) PCIE 1, SerDes 1 (lanes[23:20]) PCIE 1, SerDes 2 (lanes[27:24]) PCIE 1, SerDes 3 (lanes[31:28])

IP ID Addressing

Each SerDes quad has a unique IP ID under the quad target ID. The `IP_ID` address field (address bits [27:24]) is defined as follows:

Table 20: *Ethernet IP ID Address Map*

IP ID					Description
CSR Space	**27**	**26**	**25**	**24**	
SerDes Quad 0	1	0	0	0	Ethernet 0, SerDes Quad 0 (lanes 3-0). Ethernet 1, SerDes Quad 0 (lanes 11-8). PCIE 1, SerDes Quad 0 (lanes 19-16).
SerDes Quad 1	1	0	0	1	Ethernet 0, SerDes Quad 1 (lanes 7-4). Ethernet 1, SerDes Quad 1 (lanes 15-12). PCIE 1, SerDes Quad 1 (lanes 23-20).
SerDes Quad 2	1	0	1	0	PCIE 1, SerDes Quad 2 (lanes 27-24).
SerDes Quad 3	1	0	1	1	PCIE 1, SerDes Quad 3 (lanes 31-28).

Address Dictionary Tokens

To simplify access to the device memory space, there is an address dictionary which divides the memory areas into tokens. This dictionary is used to access memory areas and individual registers using names, rather than direct addresses. This token approach results in more readable code while reducing the need to remember individual addresses.

The address dictionary is included as part of ACE, accessed using the Tcl console, and used to access the memory space via the JTAG port. The tokens for the SerDes subsystems address areas listed in the above table are as follows:

Table 21: *Address Dictionary Tokens*

Top Level	Target ID	IP ID	SerDes Lanes
CSR_SPACE	ETHERNET_0	SERDES_0	[3:0]
	ETHERNET_0	SERDES_1	[7:4]
	ETHERNET_1	SERDES_0	[11:8]
	ETHERNET_1	SERDES_1	[15:12]
	PCIE_1	SERDES_0	[19:16]
	PCIE_1	SERDES_1	[23:20]
	PCIE_1	SERDES_2	[27:24]
	PCIE_1	SERDES_3	[31:28]

For more information on how to use the Address dictionary tokens refer to *Runtime Programming of Speedster FPGAs* (AN025).

Register Address Within the SerDes

The Register Address, bits [23:0], of each SerDes quad is broken down into three subsystems: PMA, PCS Common, and PCS lane-specific registers. Within each of those subsystems, bits [19:0] can be used to address registers within those subsytems.

Table 22: *SerDes Subsystem Register Addressing*

Level	Subsystem	Register Address[23:20]
SerDes PHY	PMA	4'b0000
	PCS Common	4'b1011
	PCS Lane[0]	4'b1100
	PCS Lane[1]	4'b1101
	PCS Lane[2]	4'b1110
	PCS Lane[3]	4'b1111

Register Address, PMA Subsystem

The Register Address space of the PMA system can be broken down further than that shown above. If the PMA subsystem is selected (bits[23:20] = 4'b0000), the remaining bits are defined as shown in the following table.

Table 23: *SerDes PMA Subsystem Register Addressing*

Address Bit	23	22	21	20	19	18	16	15	14	13	…	0
PMA Subsystem Address	0	0	0	0	SRAM	Broadcast	PMA lane			Register offset		

SRAM: Bit[19]

Bit 19 selects the static RAM of the PMA quad. The PMA requires loading of the instruction set into the SRAM upon the device bringup. The SRAM space holds the customer-defined preset rate configurations. These presets are typically part of the IP setup in ACE and written at initial configuration.

Broadcast: Bit[18]

Bit 18 selects broadcast mode for the PMA quad. In broadcast mode, the register offset location of each of the four lanes is written with the desired value.

Lane: Bit[16:14]

When not in broadcast mode, Bits [16:14] select the lane of the PMA quad being written to or read from.

Table 24: *SerDes PMA Subsystem Quad Lane Addressing*

Level	PMA Lane	Bits [16:14]
PMA Lane	Common	3'b000
	Lane 0	3'b001
	Lane 1	3'b010
	Lane 2	3'b011
	Lane 3	3'b100

Chapter - 6: ACE Support for Speedster7t SerDes

Introduction

The ACE GUI supports configuration of the Speedster7t SerDes in raw mode. Within ACE, the IP can be configured and added to the project. ACE also supports generating the timing files, pin constraints, bitstream files, and simulation files. These files are necessary for simulation and place/route. The following sections describe how to configure the SerDes subsystem in Raw Mode and generate the necessary files within the ACE environment.

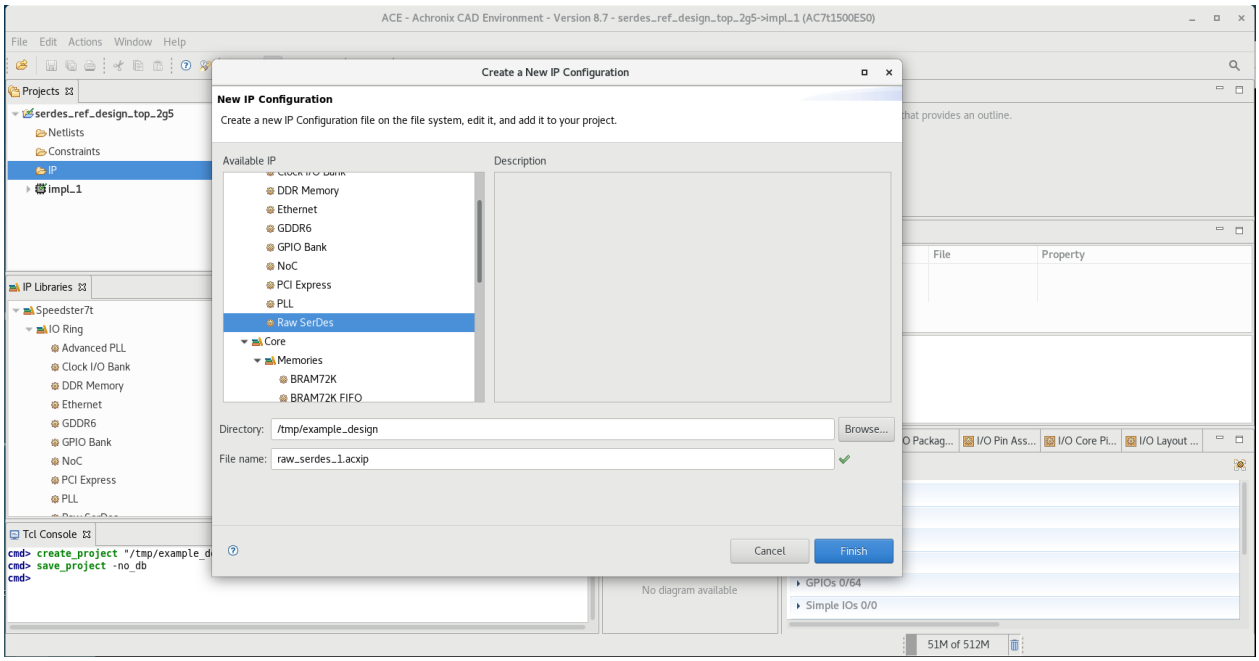
More details on the usage of the ACE tool can be found in *ACE User Guide* (UG070).

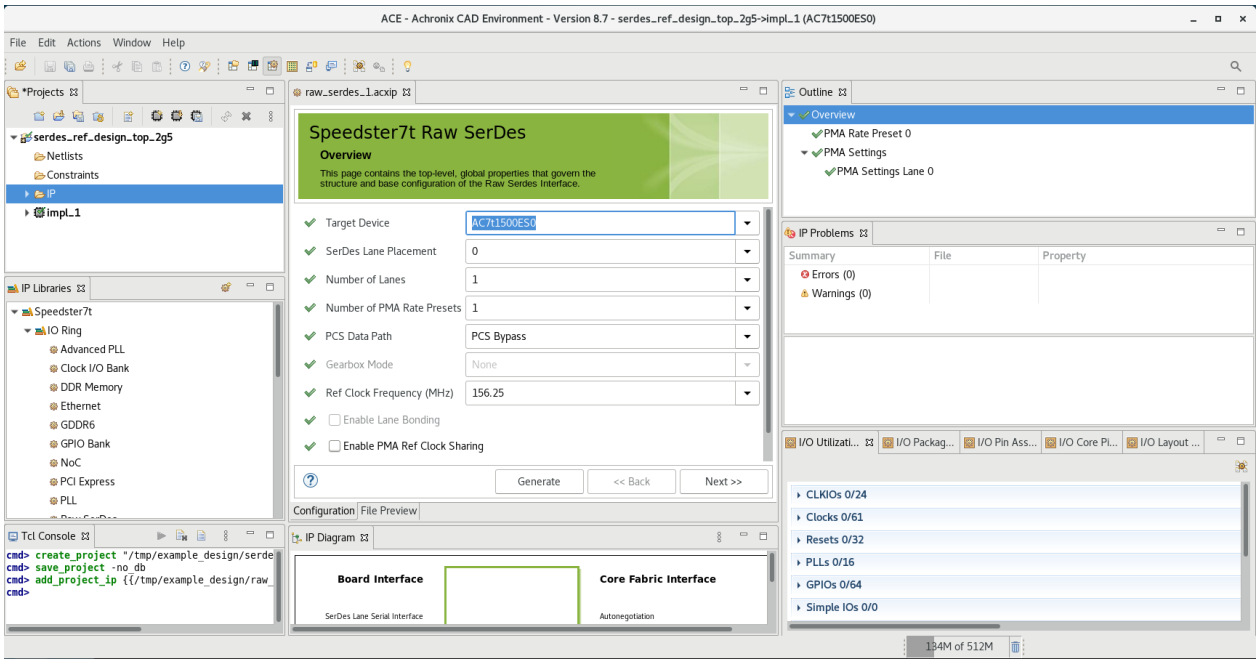
SerDes IP Configuration

SerDes IP configuration consists of the following steps:

New SerDes IP Instantiation

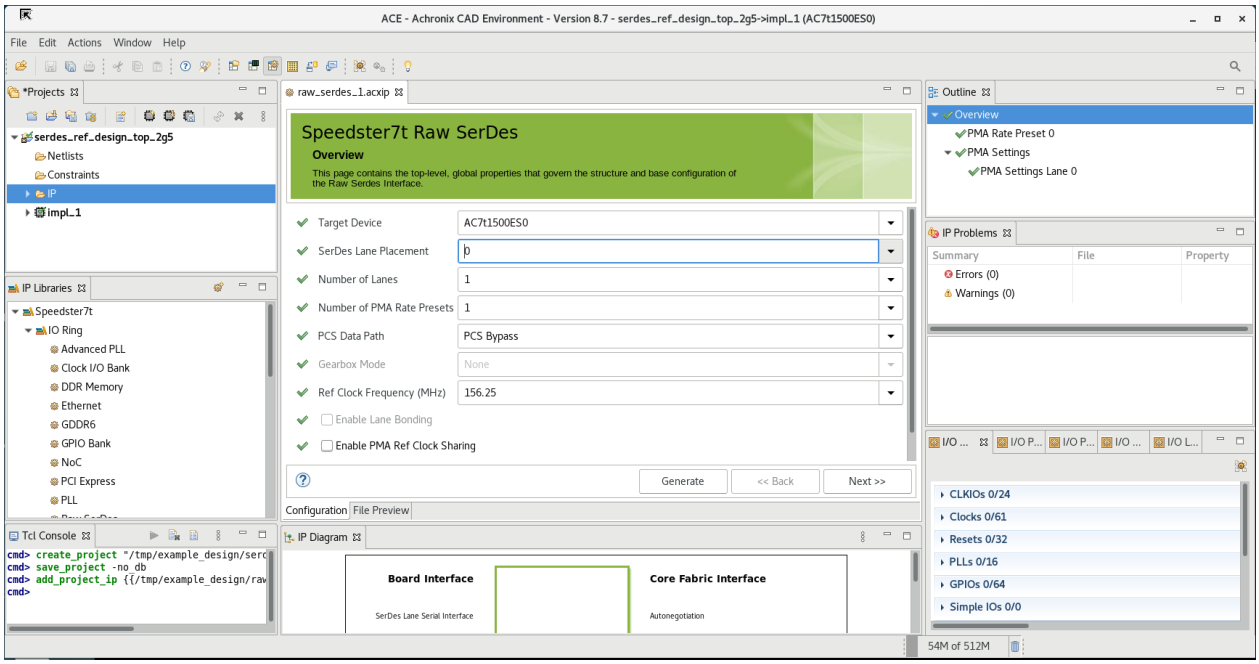
1. From an open project, right click the **IP** folder under the project name and select **New IP Configuration**.
2. In the **New IP Configuration** dialog, select **Raw SerDes** and assign a proper file name for the new IP (i.e., `raw_serdes_1.acxip`). Click **Finish** to complete the process.
3. A new IP with the given name `raw_serdes_1.acxip` appears in the IP folder.



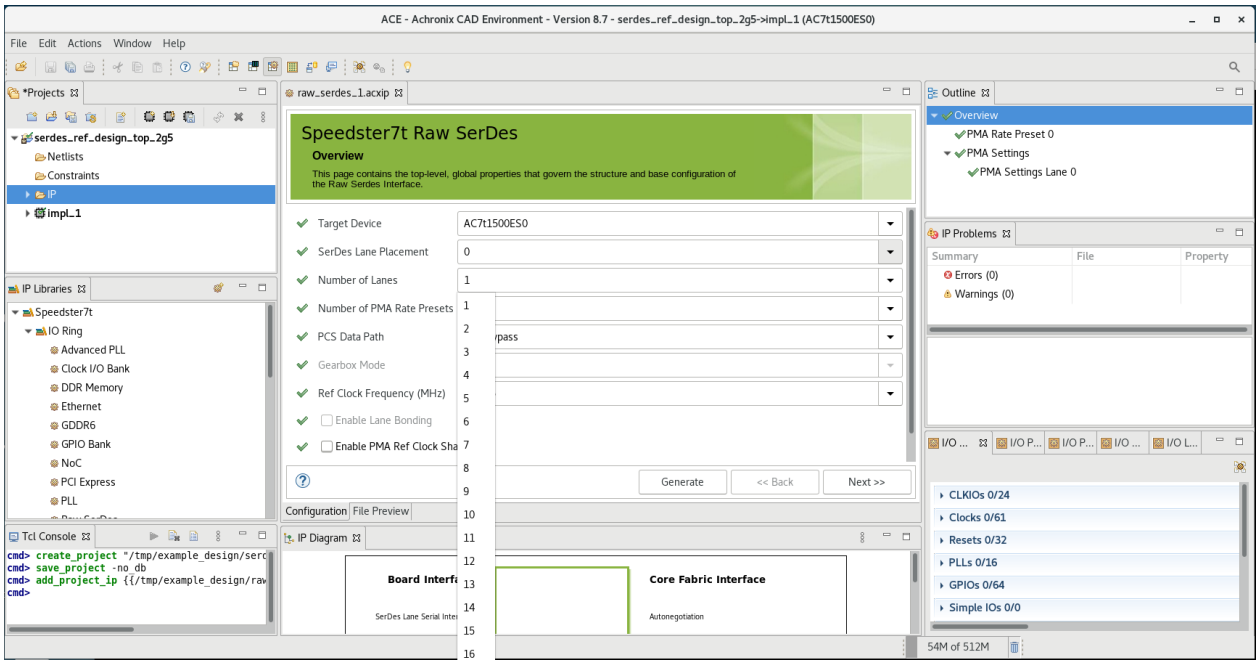


New SerDes IP Configuration

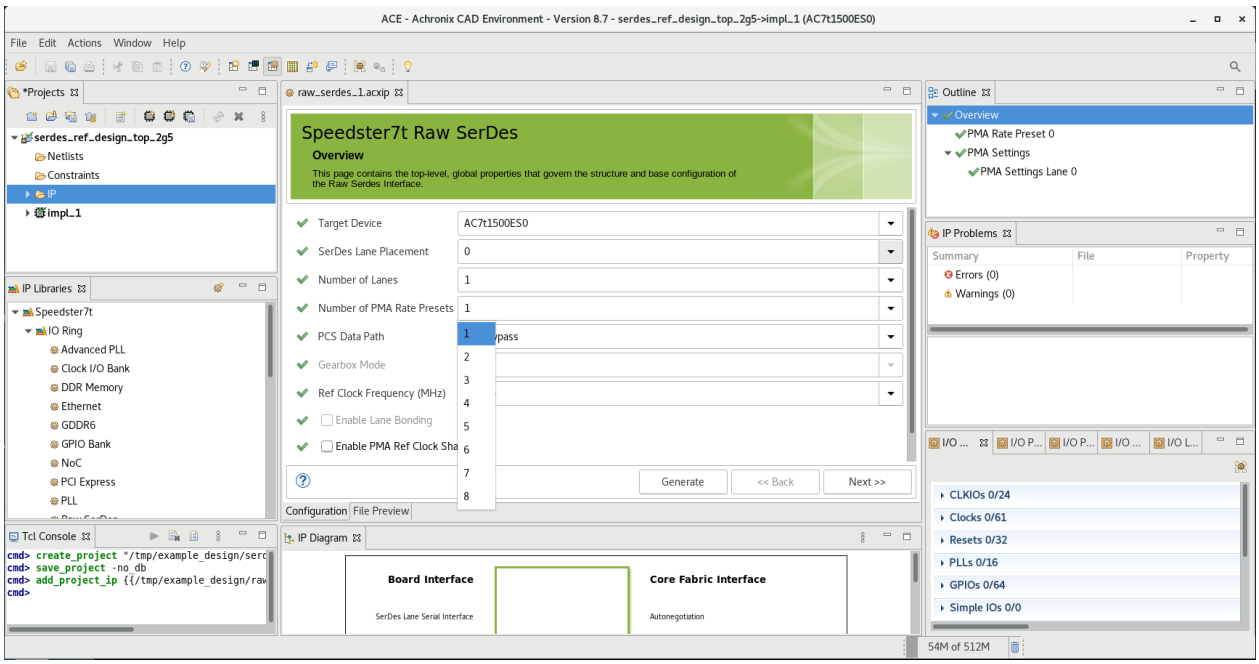
1. In **IP Configuration Perspective** view, click the new SerDes IP, `raw_serdes_1.acxip` under the IP folder. The Speedster7t **Raw SerDes Overview** window opens. This window is where the Raw SerDes properties and parameters are configured. In this example, a single lane IP with 100MHz reference clock and a symbol rate of 2.5G (PCIe Gen1) is configured.
2. Set the Target Device to **AC7t1500ES0**.
3. Open the **SerDes Lane Placement** listbox and select the lane number where the new SerDes IP will be placed. In this example, Lane 0 is selected.



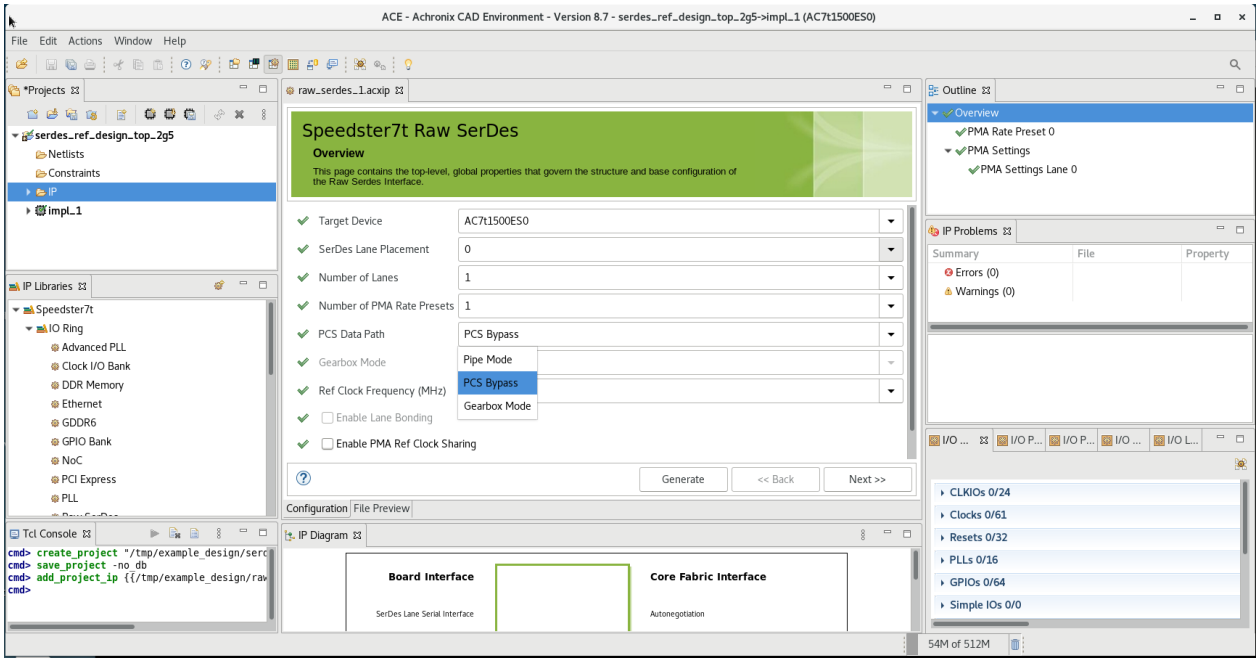
4. Open the **Number of Lanes** listbox and choose the number of lanes for the SerDes IP. The maximum number of lanes allowed is 16. In this example, the IP will be one lane wide.



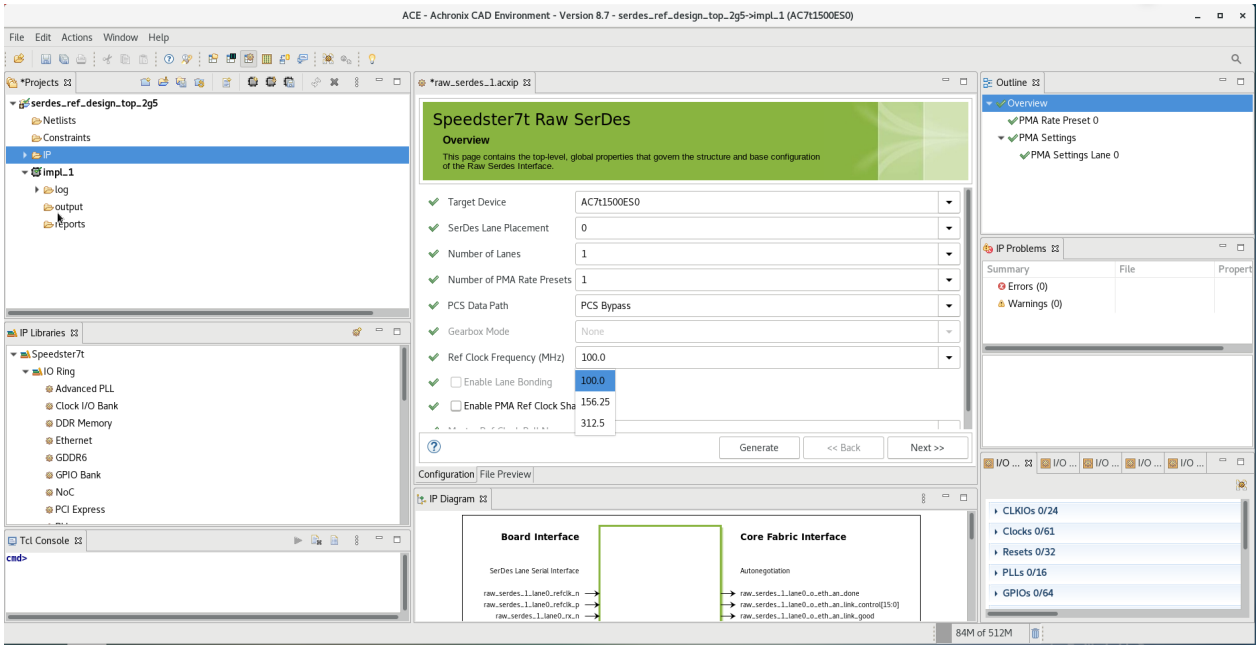
5. Open the **Number of PMA Rate Presets** listbox and select the number of preset data rates for the new SerDes IP. This selection determines the number of rate configurations that are preprogrammed into the device at power up. The maximum number of preset rates is 8. In this example, one preset rate is selected. If multiple rates are selected, the device powers up to "PMA Rate Preset 0". Details on how to change rates during runtime can be found in the PMA section (see page 13).



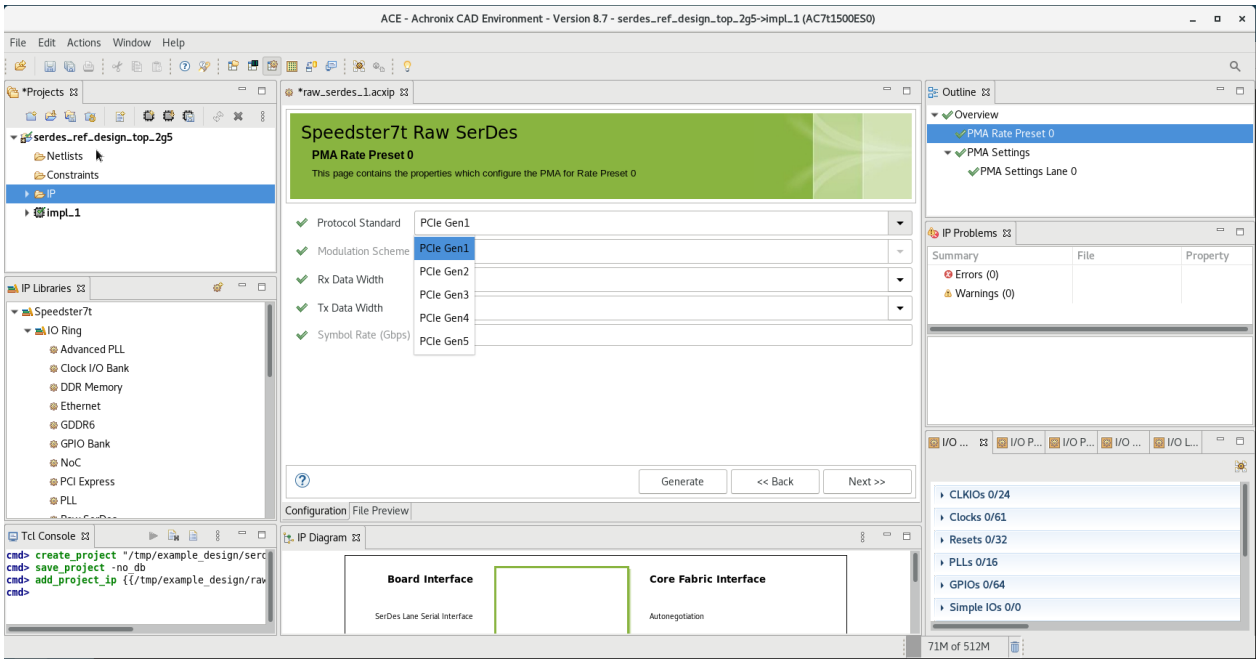
6. In the **PCS Data Path** listbox, select **PCS Bypass mode** for the new SerDes IP. This allows a direct connection between the SerDes and the logic in the fabric. **Pipe Mode** selects the PHY interface for he PCI Express and **Gearbox Mode** provides access to the gearbox modes.



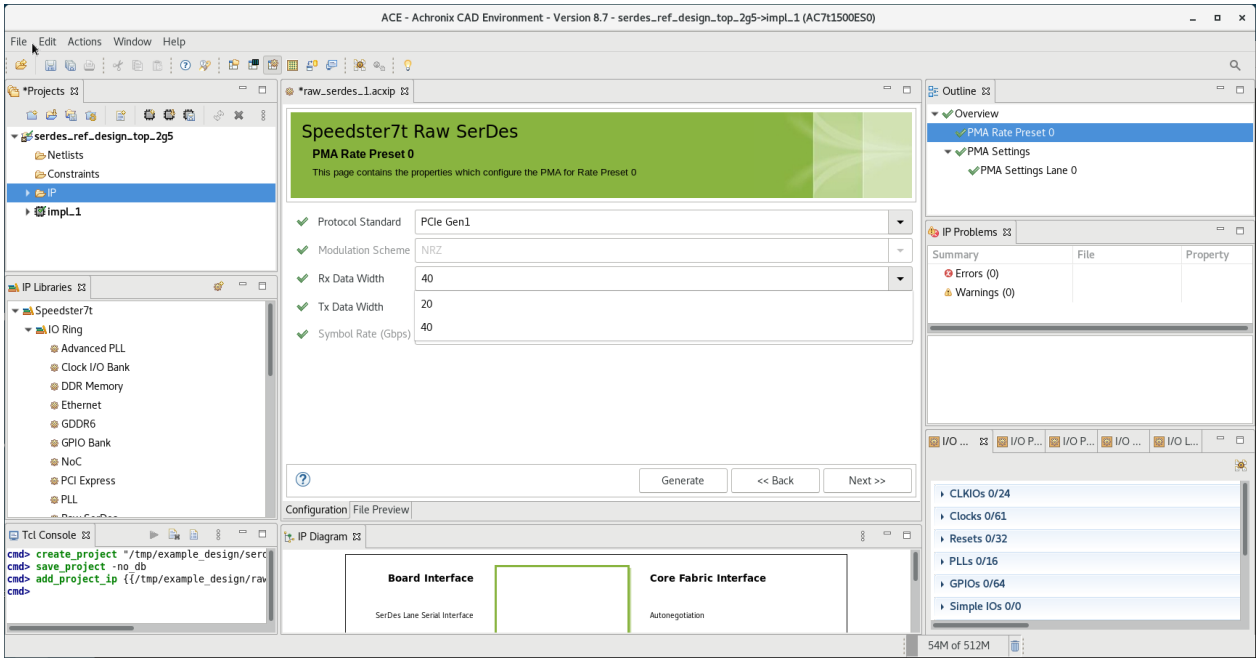
7. Select the clock frequency that is being driven to the reference clock pin. This example is using lane zero, so reference clock SRDS_N6_REFCLK_P/N is applicable. This example assumes 100MHz is being driven to that pin.



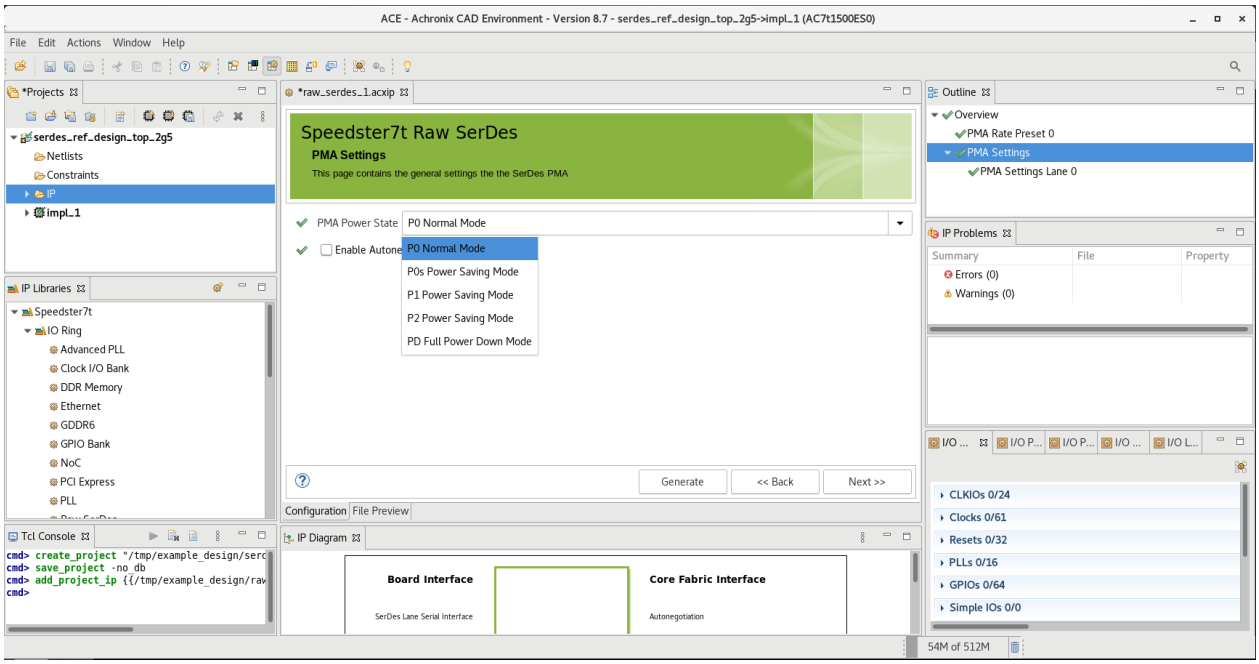
8. An external reference clock is required for the SerDes IP. The PMA Ref Clock Sharing function is not enabled for this design because it is a single lane. If the ACXIP spans multiple quads, the Ref Clock Sharing Function can be enabled and the ref clock can be shared across the quads. Click **Next** to continue the configuration process.
9. The necessary protocol can be selected to meet system requirements. The **Ref Clock Frequency** dictates which protocols can be selected. The **Protocol Standard**, **PCIe Gen1**, is used in this example. After selection, the resulting Symbol Rate is shown.



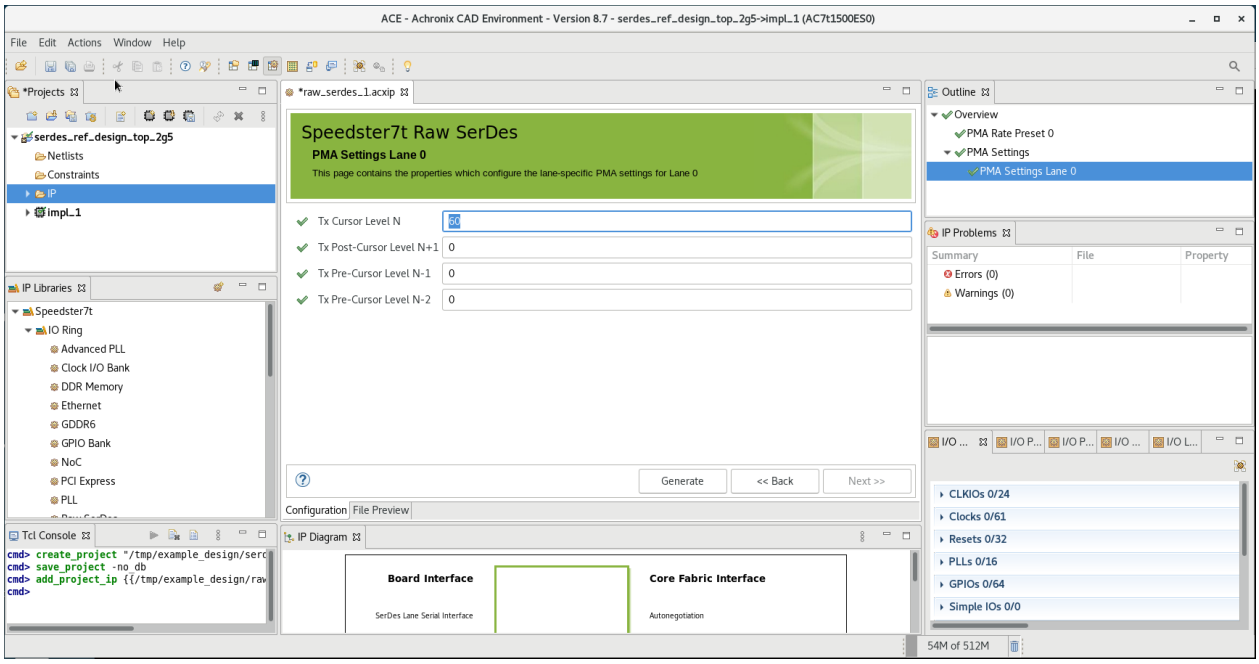
10. The data width chosen affects the speed at which the fabric/IORING interface needs to run to meet the protocol symbol rate. The **Rx Data Width** and **Tx Data Width** are both selected to be 40 in this example. A selection of 20 would require the IORING fabric interface to run at twice the rate.



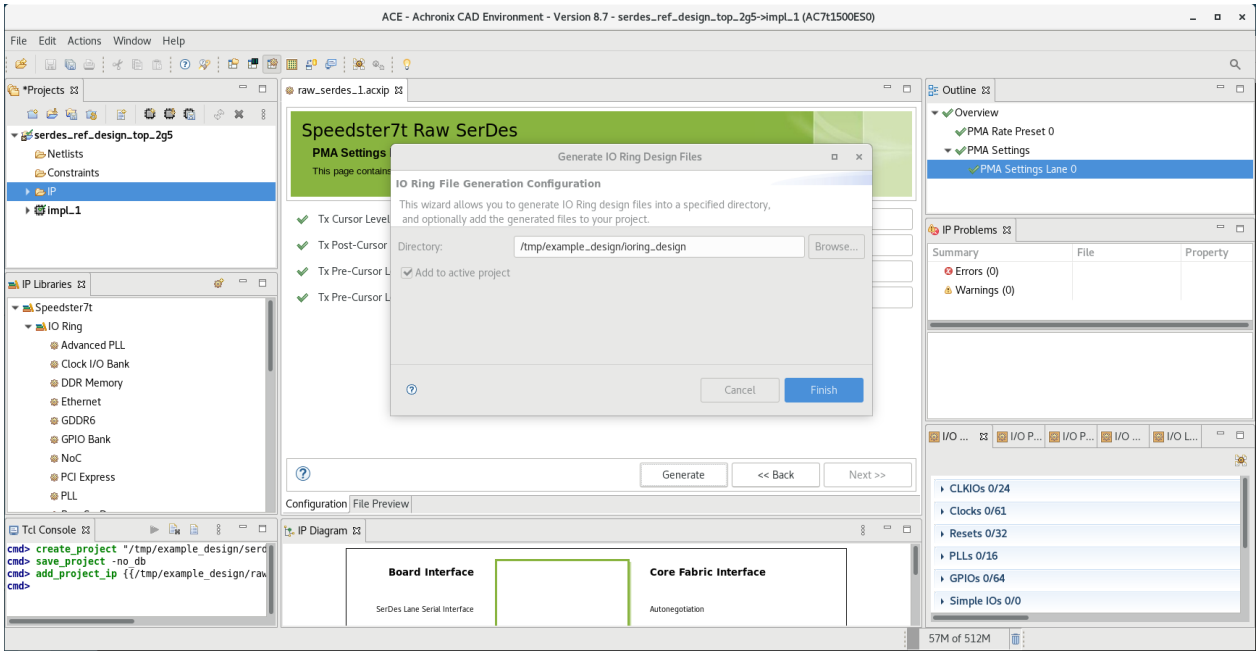
11. **PMA power state** selects the mode in which the PMA is initialized. **P0 Normal Mode** is the normal, full operational mode and is selected in this example. Details on the other available power states can be found in the PMA section (see page 13).

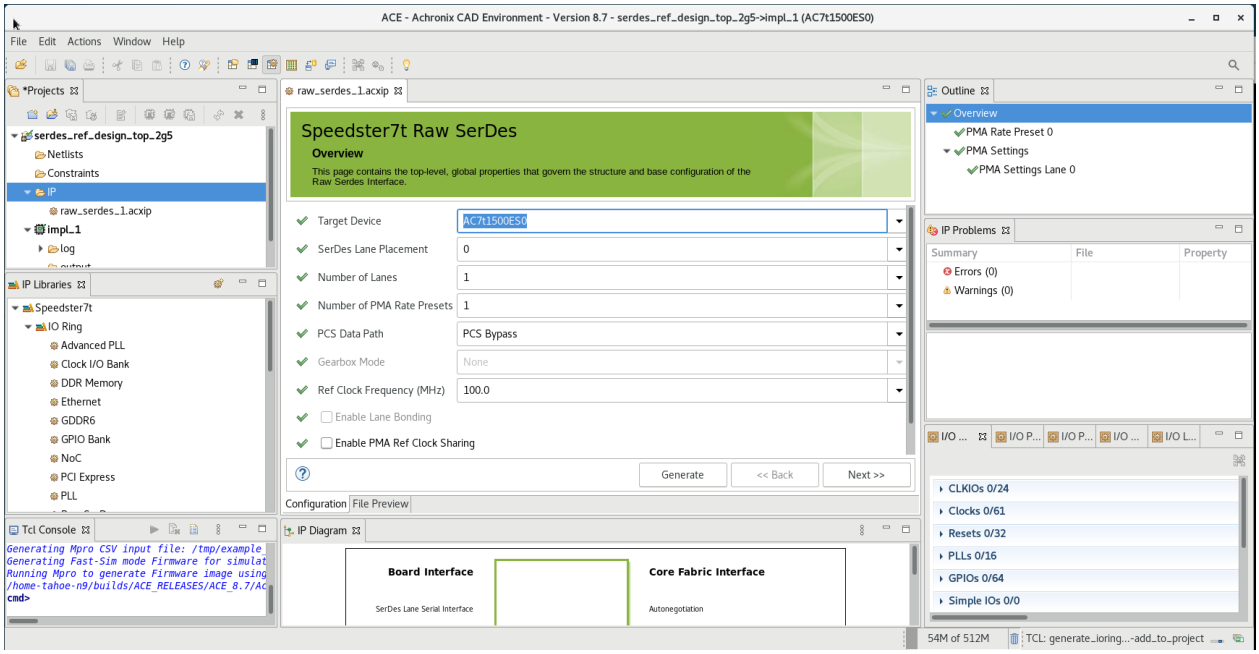


12. The TX driver settings are preset for each protocol. It is recommended to keep these values at their default settings. Details on the available driver options can be found in the PMA section (see page 13).

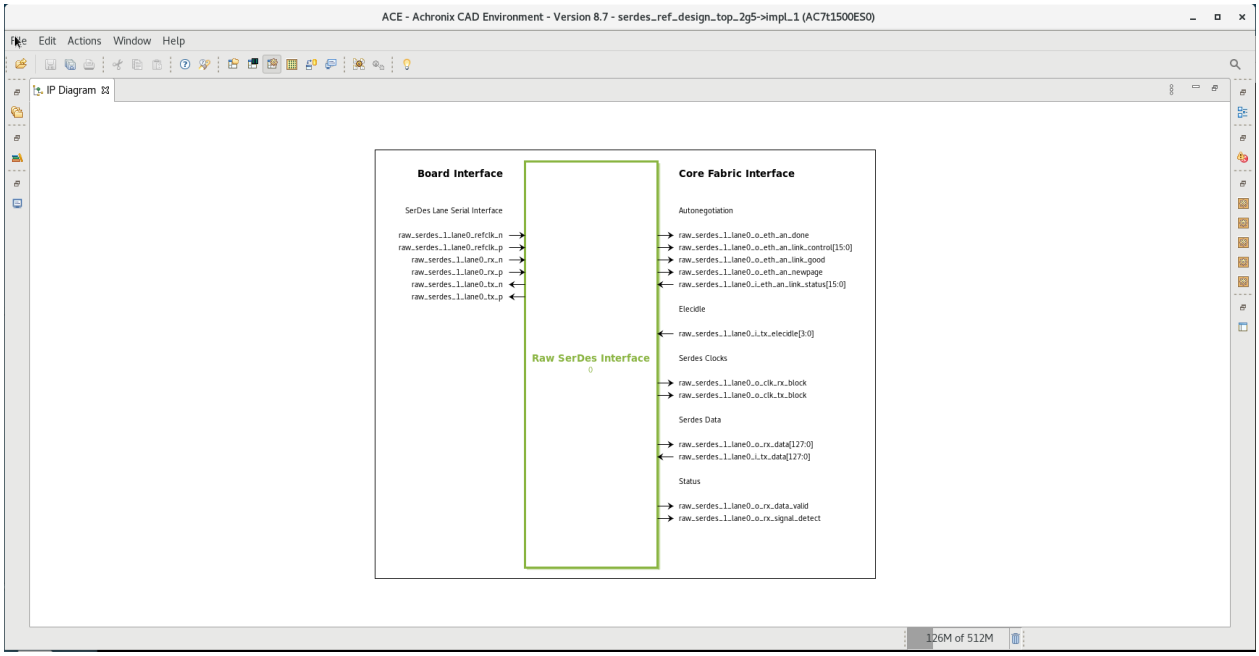


13. Click the **Generate** button. This action starts the process that generates the collateral needed by simulation and by ACE for the hardware implementation. The files are generated in a folder chosen by the user. The default location, `<implementation_directory>/ioring_design`, is shown in the example as `tmp/example_designs/ioring_design`. The constraint files and bitstream files needed by ACE are added to the project automatically if **Add to active project** is selected.





14. The interface pins and signals for the new SerDes IP are displayed In the IP Diagram widow. These signals are written to a text file during the generate step. This file is called `<project_name>_user_design_port_list.svh`. In this example, the files name is `serdes_ref_design_top_2g5_user_design_port_list.svh`. The port list in the top-level RTL must match this list exactly.



This concludes the creation of a raw serdes IP. A detailed example can be found in the *Speedster7t SerDes Reference Design Guide* (RD029).

Chapter - 7: Simulation of Speedster7t SerDes

The SerDes has full simulation support. For general information on how to simulate designs that include Achronix hard IP, such as the SerDes, see the *Simulation User Guide* (UG072). This guide contains specific details on simulating the SerDes IP.

Device Simulation Model

Designs that include the SerDes, and other interface subsystems, require the Device Simulation Model (DSM). The DSM includes the full SerDes Register Transfer Level (RTL) Model and Bus Functional Model (BFM). The BFM provides a functionally accurate simulation model with significantly faster simulation times.

Selecting the Required DSM

DSM Utility Package

There is a DSM for each device, with each DSM representing the specific features of that device. It is therefore necessary to select the correct DSM within a simulation testbench. Selection of the correct DSM is achieved by including the appropriate DSM utility package. The package creates macros and functions to access the appropriate DSM. The utility package defines the macro `ACX_DEVICE_NAME`, which is used to instantiate and refer to the DSM. The SystemVerilog header file is named `<device>_utils.svh` (i.e., `ac7t1500_utils.svh`). *This file must be included in your testbench.*

Header File Include Example

```
`include "ac7t1500_utils.svh"
```

DSM Compile Files

The following files must be compiled in your simulator so the DSM Utility package, `<device>_utils.svh`, and other necessary libraries can be found:

- $(ACX_DEVICE_INSTALL_DIR)
  `$(ACE_INSTALL_DIR)/system/data/$(DEVICE)`

  ACX_DEVICE_INSTALL_DIR Include Example

  ```
  ACE_8.8/Achronix-linux/system/data/AC7t1500ES0
  ```

- Includes Defines

  Includes Defines Example

  ```
  $(ACX_DEVICE_INSTALL_DIR)/sim/ac7t1550_dsm_incdirs.f
  ```

- Device Simulation Model

Device Simulation Model Example

```
$(ACX_DEVICE_INSTALL_DIR)/sim/ac7t1550_dsm_filelist.v
```

IO Ring Connection

The DSM and the fabric must be instantiated in the testbench. The ports of both must be connected together.

Fabric Logic Instantiation

The macro, `ACX_DEVICE_NAME, is defined in the DSM utility file for the selected device. Use this macro to instantiate the fabric logic. It is also necessary to connect the chip_ready signal. The chip_ready signal indicates when configuration is complete and the fabric is now in user mode.

ACX_DEVICE_NAME Macro Example

```
`ACX_DEVICE_NAME `ACX_DEVICE_NAME (
  .FCU_CONFIG_USER_MODE (chip_ready)
)
```

Port Binding

A binding file is necessary to tie the SerDes IO fabric connections to the appropriate connections in the DSM, which includes the SerDes BFM/RTL models. The file must be included in the user testbench and is written into the ioring directory when the ACXIP files are generated.

Port Binding Include File Example

```
<acx project_name>_user_design_port_bindings.svh
```

Simulation Defines

The *_sim_defines.svh file must be included in your compilation prior to the DSM being compiled. It is written into the ioring directory when the ACXIP files are generated.

Simulation Defines Header File Example

```
`define SIM_DEFINES_FILENAME "../../src/ioring/serdes_ref_design_top_2g5_sim_defines.svh"
```

Simulation Configuration

The DSM requires configuration. The *_sim_config.svh file writes the SerDes configuration registers in the RTL/BFM model to match how the SerDes was configured in the GUI. This configuration file is written out during the IP generation stage and must be included in your testbench.

Simulation Configuration Header File Example

```
<acx_project_name>_sim_config.svh
```

Special SerDes Configuration

In addition to the simulation configuration written by ACE in the ACXIP generation, it is necessary to configure the SerDes data mux to switch the SerDes signals to the fabric during simulation. This mux defaults to driving the SerDes signals to their respective Ethernet subsystems. Thus, it is required to configure this mux to route the SerDes data signals to the user design. This is demonstrated in the SerDes Reference Design. The mux is configured using the `rawmode_config.txt` configuration file. It is sourced in the testbench when in BFM simulation mode. The same command file is used for both Ethernet subsystems since the relative address of the SerDes data mux is the same within both subsystems. No special configuration is necessary for lanes associated with subsystem `PCIE_1`. The sequence below was generated for a BFM simulation run on the IP `ETHERNET_0` and `ETHERNET_1`.

Command

A function is called in the testbench to read and write the registers that configure the SerDes data mux during BFM simulation. This function is available when including the DSM Utility package.

Data MUX Configuration Example

```
`ACX_DEVICE_NAME.fcu.configure( ramode_config.txt, "full" );
```

Contents of the Configuration File

Reads and writes to the DSM addresses which set the SerDes data Mux.

Data MUX Configuration File Example

```
# CSR_VERIFY from CSR_SPACE ETHERNET_0 CFG_EIU CONFIG : expect 0x0
v 081b0000010 00000000
# CSR_READ from CSR_SPACE ETHERNET_0 CFG_EIU CONFIG
r 081b0000010 00000000
# CSR_WRITE to CSR_SPACE ETHERNET_0 CFG_EIU CONFIG : value 0x20000000
w 081b0000010 20000000
# CSR_VERIFY from CSR_SPACE ETHERNET_0 CFG_EIU CONFIG : expect 0x20000000
v 081b0000010 20000000
# CSR_VERIFY from CSR_SPACE ETHERNET_1 CFG_EIU CONFIG : expect 0x0
v 081c0000010 00000000
# CSR_READ from CSR_SPACE ETHERNET_1 CFG_EIU CONFIG
r 081c0000010 00000000
# CSR_WRITE to CSR_SPACE ETHERNET_1 CFG_EIU CONFIG : value 0x20000000
w 081c0000010 20000000
# CSR_VERIFY from CSR_SPACE ETHERNET_1 CFG_EIU CONFIG : expect 0x20000000
v 081c0000010 20000000
```

Revision History

Version	Date	Description
1.0	09 Sep 2022	• Initial Release.