JESD204C v4.2

LogiCORE IP Product Guide

Vivado Design Suite

PG242 March 25, 2021

XILINX

Table of Contents

Introduction

The Xilinx® LogiCORE™ IP JESD204C core implements a JESD204C [Ref 9] compatible interface supporting line rates from 1 Gb/s to 32 Gb/s (the maximum line rate supported is dependent on the transceiver type and speed grade of the selected device). The JESD204C core can be configured to transmit or receive using either a 64B66B or 8B10B link layer. The JESD204C core is fully backwards compatible with JESD204B.

Features

- Designed to JEDEC® JESD204C Standard
- Supports GTY transceivers on Versal™ ACAPs
- Supports GTH and GTY transceivers on UltraScale+™ and UltraScale™ devices
- Supports up to eight lanes per core and greater number of lanes using multiple cores
- Supports 64B66B and 8B10B link layers
- Supports FEC Encoding (TX) and Decoding (RX) on the 64B66B link layer
- Supports CRC-12, CMD and FEC meta data modes on the 64B66B link layer
- Supports subclass 0 and 1 on the 64B66B link layer and Subclass 0,1, and 2 on the 8B10B link layer
- Provides physical and data link layer functions when used in conjunction with the Versal ACAP Transceiver Wizard for Versal ACAPs and JESD204_PHY core for UltraScale and UltraScale+ devices

  Note: The Versal ACAP Transceiver Wizard is used directly by the JESD204C core and the JESD204_PHY is no longer required.
- AXI4-Lite configuration interface
- AXI4-Stream Data and Command interfaces
- Supports Transceiver sharing between TX and RX cores using the JESD204_PHY core or Versal ACAP Transceiver Wizard

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	Versal ACAP, UltraScale+, UltraScale
Supported User Interfaces	AXI4-Lite, AXI4-Stream
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Encrypted RTL
Example Design	Verilog
Test Bench	Verilog
Constraints File	XDC
Simulation Model	Verilog
Supported S/W Driver	N/A
Tested Design Flows(2)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 68804
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Xilinx Support web page	

Notes:

1. For a complete listing of supported devices, see the Vivado IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. This document covers the following design processes:

- **Hardware, IP, and Platform Development**: Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado timing, resource and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Port Descriptions
  - Register Space
  - Clocking
  - Resets
  - Customizing and Generating the Core
  - Example Design

Core Overview

The LogiCORE™ IP JESD204C core implements a JESD204C link layer. When used in conjunction with the LogiCORE™ IP Versal ACAP Transceiver Wizard or JESD204_PHY core (to provide the physical layer), a JESD204C system can be created supporting line rates between 1 and 32 Gb/s on 1 to 8 lanes using Versal GTY transceivers or UltraScale+ and UltraScale transceivers (both GTH and GTY).

See the device data sheets for maximum line rates supported by each device and family. The JESD204C core can be configured as transmit or receive, using either 64B66B or 8B10B linecoding, and multiple cores can be used to realize links requiring more than eight lanes.

The JESD204C core is delivered by using the Xilinx Vivado® Design Suite. In addition, an example design is provided in Verilog.

Unsupported Features

Data Converter sample data mapping is not supported by this IP as the data mapping varies for different devices. For more information see applicable converter data sheets. A simple example sample data mapper and demapper is provided for reference in the example design that can be generated for the core.

Licensing and Ordering

License Checkers

If the IP requires a license key, the key must be verified. The Vivado design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with error. License checkpoints are enforced by the following tools:

- Vivado synthesis
- Vivado implementation
- write_bitstream (Tcl command)

IMPORTANT: *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

License Type

This Xilinx LogiCORE IP module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the Vivado Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. Contact your local Xilinx sales representative for information about pricing and availability.

For more information, visit the JESD204 product web page.

Information about other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

A free evaluation version of the core is provided with the Xilinx Vivado Design Suite which lets you assess the core functionality and demonstrates the various interfaces of the core in simulation. To access the evaluation version visit the JESD204 IP Evaluation page.

License Options

The JESD204C IP core license is provided as part of the JESD204 core license (no separate license is required). The JESD204 core license provides three options. After installing the Vivado Design Suite and the required IP Service Packs, choose a licensing option.

Simulation Only

The Simulation Only Evaluation license key is provided with the Xilinx Vivado Design Suite. This key lets you assess core functionality with either the example design provided with the JESD204C core, or alongside your own design and demonstrates the various interfaces to the core in simulation. (Functional simulation is supported by a dynamically generated HDL structural model.)

Full System Hardware Evaluation

The Full System Hardware Evaluation license is available at no cost and lets you fully integrate the core into an FPGA design, place-and-route the design, evaluate timing, and perform functional simulation of the JESD204C core using the example design and demonstration test bench provided with the core.

In addition, the license key lets you generate a bitstream from the placed and routed design, which can then be downloaded to a supported device and tested in hardware. The core can be tested in the target device for a limited time before timing out (ceasing to function), at which time it can be reactivated by reconfiguring the device.

Full

The Full license key is available when you purchase the core and provides full access to all core functionality both in simulation and in hardware, including:

- Gate-level functional simulation support
- Back annotated gate-level simulation support
- Functional simulation support
- Full-implementation support including place and route and bitstream generation
- Full functionality in the programmed device with no time-outs

Obtaining Your License Key

This section contains information about obtaining a simulation, full system hardware, and full license keys.

Simulation License

No action is required to obtain the Simulation Only Evaluation license key; it is provided by default with the Xilinx Vivado Design Suite.

Full System Hardware Evaluation License

To obtain a Full System Hardware Evaluation license, perform these steps:

1. Navigate to the JESD204 product page for this core.
2. Click Evaluate.
3. Follow the instructions on the page.

Obtaining a Full License

To obtain a Full license key, you must purchase a license for the core. After doing so, click the **Access Core** link on the xilinx.com IP core product page for further instructions.

Installing Your License File

The Simulation only Evaluation license key is provided with the Vivado Design Suite and does not require installation of an additional license file. For the Full System Hardware Evaluation license and the Full license, an email will be sent to you containing instructions for installing your license file. Additional details about IP license key installation can be found in the Vivado Design Suite Installation, Licensing and Release Notes document.

Product Specification

The JESD204C core is used in conjunction with the Versal™ ACAP Transceiver Wizard or JESD204_PHY core to support the JESD204C physical layer link layer specification.

Standards

JEDEC® Serial interface for Data Converters JESD204C [Ref 9].

Performance and Resource Utilization

For details about performance and resource utilization, visit the Performance and Resource Utilization web page.

Port Descriptions

The port descriptions for the JESD204C core are described in the following sections.

TX Core

Table 2-1: **TX Core: System Signals**

Signal Name	Interface	I/O	Description
tx_core_clk	System s_axis_tx s_axis_tx_cmd	I	Core logic clock input. Frequency: Serial line rate / 66 (for 64B66B linecoding) or Serial line rate / 40 (for 8B10B linecoding)
tx_core_reset	System	I	Core asynchronous logic reset active High.
tx_aresetn	s_axis_tx s_axis_tx_cmd	O	AXI4-Stream interface reset. Active low. Associated with both data and command interfaces.
s_axi_aclk	s_axi	I	AXI4-Lite clock input.

Table 2-1: **TX Core: System Signals *(Cont'd)***

Signal Name	Interface	I/O	Description
s_axi_aresetn	s_axi	I	AXI4-Lite reset input. Active low.
s_axi*	s_axi	I	See Appendix A of the *Vivado AXI Reference Guide* (UG1037) [Ref 10] for a description of AXI4 signals.
irq	System	O	System interrupt output.
tx_sysref	System	I	SYSREF input. When Subclass 1 mode is selected, this signal is required and used by the core to set the phase of the local extended multi-block clock. This SYSREF signal must be generated synchronous to the core clock. This input should be driven from an external device generating SYSREF for both TX and RX on a link.
tx_sync	System	I	Sync signal. The sync signal is defined as an active-Low sync request signal by JESD204 so this signal is Low until comma alignment is completed and High to request ILA and normal data. This signal is only available when the core is generated with 8B10B linecoding selected.
Versal ACAPs			
gt_powergood(1)	System	I	This active-High signal from Versal ACAP Transceiver indicates when the GT clocking resources have completed power up.
gt_loopback[2:0](1)	System	O	This signal to the Versal ACAP Transceiver controls the various loopback modes. This must be connected to the chN_loopback[2:0] port in the CHn_DEBUG interface of the GT Quad.
UltraScale+/UltraScale Devices			
tx_reset_gt	System	O	JESD204_PHY TX datapath reset. Core output to reset the transmit datapath in a connected JESD204_PHY. This must be connected to a JESD204_PHY.
tx_reset_done	System	I	JESD204_PHY TX reset done input. Indicates the JESD204_PHY has completed the transmit reset process.

Notes:

1. For an example of how to connect this port to the GT Quad, generate the example design.

Table 2-2: **TX Core: Versal GT TX Interface Ports**

Signal Name	Interface	I/O	Description
chN_txctrl0[15:0]	GT_TX	O	TX Disparity control to the Versal ACAP Transceiver. N = Lanes -1
chN_txctrl1[15:0]	GT_TX	O	TX Disparity polarity control to the Versal ACAP Transceiver. N = Lanes -1

Table 2-2: **TX Core: Versal GT TX Interface Ports *(Cont'd)***

Signal Name	Interface	I/O	Description
chN_txctrl2[7:0]	GT_TX	O	TX Char is K to the Versal ACAP Transceiver. N = Lanes -1
chN_txdata[127:0]	GT_TX	O	TX Data to the Versal ACAP Transceiver. N = Lanes - 1
chN_txdiffctrl[4:0]	GT_TX	O	TX diffctrl to the Versal ACAP Transceiver. N = Lanes - 1
chN_txelecidle	GT_TX	O	TX txelecidle to the Versal ACAP Transceiver. N = Lanes - 1
chN_txheader[5:0]	GT_TX	O	TX Header to the Versal ACAP Transceiver. N = Lanes - 1
chN_txinhibit	GT_TX	O	TX Inhibit to the Versal ACAP Transceiver. N = Lanes - 1
chN_txmstreset	GT_TX	O	TX Master Reset sequence start to the Versal ACAP Transceiver. N = Lanes - 1
chN_txmstresetdone	GT_TX	I	TX Master Reset sequence Done to the Versal ACAP Transceiver. N = Lanes - 1
chN_txpd[1:0]	GT_TX	O	TX Power Down to the Versal ACAP Transceiver. N = Lanes - 1
chN_txpmaresetdone	GT_TX	I	TX PMA Reset Done from the Versal ACAP Transceiver. N = Lanes - 1
chN_txpolarity	GT_TX	O	TX Polarity control to the Versal ACAP Transceiver. N = Lanes - 1
chN_txpostcursor[4:0]	GT_TX	O	TX Post-cursor pre-emphasis control to the Versal ACAP Transceiver. N = Lanes - 1
chN_txprbssel[3:0]	GT_TX	O	TX PRBS Select to the Versal ACAP Transceiver. N = Lanes - 1
chN_txprecursor[4:0]	GT_TX	O	TX Pre-cursor pre-emphasis control to the Versal ACAP Transceiver. N = Lanes - 1
chN_txrate[7:0]	GT_TX	O	TX Line Rate Control to the Versal ACAP Transceiver. N = Lanes - 1
chN_txsequence[6:0]	GT_TX	O	TX Sequence Counter to the Versal ACAP Transceiver. N = Lanes - 1
chN_txuserrdy	GT_TX	O	TX User clocks stable to the Versal ACAP Transceiver. N = Lanes - 1

Notes:

1. JESD204C only uses the above subset of the available signals on the Versal ACAP Transceiver Tx_GT_IP_Interface. For further details see *Versal ACAP Transceiver Wizard Product Guide* [Ref 12].
2. The signals in this table are connected between the JESD204C core and the Versal ACAP Transceiver using Block Automation, see Chapter 4 for details.

Table 2-3: **TX Core: JESD204_PHY Interface Ports for UltraScale+/UltraScale Devices**

Signal Name	Interface	I/O	Description
gtN_txdata[63:0]	PHY	O	TX data to JESD204 PHY. N = Lanes - 1
gtN_txheader[1:0]	PHY	O	TX header to JESD204 PHY. N = Lanes - 1
gtN_txcharisk[3:0]	PHY	O	TX Char is K to JESD204 PHY. N = Lanes -1

Table 2-4: **TX Core: Transmit Interface (64B66B Linecoding Only)**

Signal Name	Interface	I/O	Description
tx_tdata[(64*N)-1:0]	s_axis_tx	I	Transmit data input. N = Lanes
tx_tready	s_axis_tx	O	AXI4-Stream tready.
tx_soemb	s_axis_tx	O	Start of extended multi-block boundary indication. Set to 1 to indicate tx_tdata in the following clock cycle is the start of an extended multi-block.
tx_cmd_tdata[(32*N)-1:0]	s_axis_tx_cmd	I	Transmit Cmd interface N = Lanes For Meta mode = CRC, Cmd payload is bits [6:0] with bits [31:7] set to Zero. For Meta mode = CMD, Cmd payload is [18:0] with bits [31:19] set to Zero
tx_cmd_tvalid	s_axis_tx_cmd	I	AXI4-Stream tvalid.
tx_cmd_tready	s_axis_tx_cmd	O	AXI4-Stream tready. tx_cmd_tready will be set for 1 cycle every multi-block to control the Cmd word flow.

Transmit Interface (8B10B Linecoding Only)

Table 2-5: **Transmit Interface Port Descriptions**

Signal Name	Interface	I/O	Description
tx_tdata [(32*N)-1:0]	s_axis_tx	I	Transmit data input. N = Lanes
tx_tready	s_axis_tx	O	AXI4-Stream tready

Table 2-5: **Transmit Interface Port Descriptions *(Cont'd)***

Signal Name	Interface	I/O	Description
tx_sof [3:0]	s_axis_tx	O	Start of frame boundary indication. The signal is four bits to indicate the byte position of the first byte of a frame in tdata in the following clock cycle. • When tx_sof = 0001, the first byte of a frame is in bits [7:0] of the tdata word with the next 3 bytes in bits[31:8]. • When tx_sof = 0010, the first byte is in bits [15:8] of the tdata word with the next 2 bytes in bits[31:16]; bits [7:0] contain the end of the previous frame. • When tx_sof = 0100, the first byte is in bits [23:16] of the tdata word with the next byte in bits[31:24]; bits [15:0] contain the end of the previous frame. • When tx_sof = 1000, tdata contains the last 3 bytes of the previous frame in bits [23:0] and the first byte of a new frame in bits [31:24]. ***Note:*** Multiple bits of tx_sof can be asserted in the same cycle, depending on the number of octets per frame. For example, for a frame size of 2 octets and tx_sof = 0101, the first and third bytes (bits[7:0] and bits[23:16]) of the tdata word contain the first bytes of frames.
tx_somf [3:0]	s_axis_tx	O	Start of multi-frame boundary indication. The position of the first byte of each multi-frame is encoded in the same way as tx_sof.

RX Core

Table 2-6: **RX Core: System Signals**

Signal Name	Interface	I/O	Description
rx_core_clk	System s_axis_rx s_axis_rx_cmd	I	Core logic clock input. Frequency: Serial line rate / 66 (for 64B66B linecoding) or Serial line rate / 40 (for 8B10B linecoding)
rx_core_reset	System	I	Core asynchronous logic reset active-High.
rx_aresetn	s_axis_rx s_axis_rx_cmd	O	AXI4-Stream interface reset. Active-Low. Associated with both data and command interfaces.
s_axi_aclk	s_axi	I	AXI4-Lite clock input.
s_axi_aresetn	s_axi	I	AXI4-Lite reset input. Active-Low.
s_axi*	s_axi	I	See Appendix A of the *Vivado AXI Reference Guide* (UG1037) [Ref 10] for a description of AXI4 signals.
Irq	System	O	System interrupt output.

Table 2-6: **RX Core: System Signals *(Cont'd)***

Signal Name	Interface	I/O	Description
rx_sysref	System	I	SYSREF input. When Subclass 1 mode is selected, this signal is required and used by the core to set the phase of the local extended multi-block clock. This SYSREF signal must be generated synchronous to the core clock. This input should be driven from an external device generating SYSREF for both TX and RX on a link.
rx_sync	System	O	Sync signal. The sync signal is defined as an active-Low sync request signal by JESD204, so this signal is Low until comma alignment is completed and High to indicate the receiver is ready for ILA and normal data. This signal is only available when the core is generated with 8B10B linecoding selected.
encommaalign	System	O	Enable/disable 8B10B comma alignment logic within the JESD204_PHY or the Versal ACAP. Transceiver GT Quad: this port should be connected to the Versal GT Quad as follows: JESD Channel 0: GPI[8] JESD Channel 1: GPI[9] JESD Channel 2: GPI[10] JESD Channel 3: GPI[11] This signal is only available when the core is generated with 8B10B linecoding selected.
Versal ACAPs			
gt_powergood(Notes:)	System	I	This active High signal from Versal ACAP Transceiver indicates when the GT clocking resources have completed power up.
UltraScale+/UltraScale Devices			
rx_reset_gt	System	O	JESD204_PHY RX datapath reset. Core output to reset the receive datapath in a connected JESD204_PHY. This must be connected to a JESD204_PHY.
rx_reset_done	System	I	JESD204_PHY RX reset done input. Indicates the JESD204_PHY has completed the receive reset process.

Notes:
1. For an example of how to connect this port to the JESD204_PHY or Versal GT Quad, generate the example design.

Table 2-7: **RX Core: Versal GT RX Interface Ports**

Signal Name	Interface	I/O	Description
chN_rxctrl0[15:0]	GT_RX	I	RX Char is K from the Versal ACAP Transceiver. N = Lanes -1
chN_rxctrl1[15:0]	GT_RX	I	RX Disparity Error from the Versal ACAP Transceiver. N = Lanes -1

Table 2-7: **RX Core: Versal GT RX Interface Ports *(Cont'd)***

Signal Name	Interface	I/O	Description
chN_rxctrl2[7:0]	GT_RX	I	RX Char is Comma from the Versal ACAP Transceiver. N = Lanes -1
chN_rxctrl3[7:0]	GT_RX	I	RX Not In Table Error from the Versal ACAP Transceiver. N = Lanes -1
chN_rxdata[127:0]	GT_RX	I	RX Data from the Versal ACAP Transceiver. N = Lanes - 1
chN_rxdatavalid[1:0]	GT_RX	I	RX Data Valid from the Versal ACAP Transceiver. N = Lanes - 1
chN_rxgearboxslip	GT_RX	O	RX Gearbox Slip to the Versal ACAP Transceiver. N = Lanes - 1
chN_rxheader[5:0]	GT_RX	I	RX Header from the Versal ACAP Transceiver. N = Lanes - 1
chN_rxheadervalid[1:0]	GT_RX	I	RX Header Valid from the Versal ACAP Transceiver. N = Lanes - 1
chN_rxlpmen	GT_RX	O	RX LPM Mode Enable to the Versal ACAP Transceiver. N = Lanes - 1
chN_rxmstreset	GT_RX	O	RX Master Reset sequence start to the Versal ACAP Transceiver. N = Lanes - 1
chN_rxmstresetdone	GT_RX	I	RX Master Reset sequence Done to the Versal ACAP Transceiver. N = Lanes - 1
chN_rxpd[1:0]	GT_RX	O	RX Power Down to the Versal ACAP Transceiver. N = Lanes - 1
chN_rxpmaresetdone	GT_RX	I	RX PMA Reset Done from the Versal ACAP Transceiver. N = Lanes - 1
chN_rxpolarity	GT_RX	O	RX Polarity control to the Versal ACAP Transceiver. N = Lanes - 1
chN_rxprbscntreset	GT_RX	O	RX PRBS Error Count Reset to the Versal ACAP Transceiver. N = Lanes - 1
chN_rxprbserr	GT_RX	I	RX PRBS Error from the Versal ACAP Transceiver. N = Lanes - 1
chN_rxprbslocked	GT_RX	I	RX PRBS Locked from the Versal ACAP Transceiver. N = Lanes - 1
chN_rxprbssel[3:0]	GT_RX	O	RX PRBS Select to the Versal ACAP Transceiver. N = Lanes - 1
chN_rxrate[7:0]	GT_RX	O	RX Line Rate Control to the Versal ACAP Transceiver. N = Lanes - 1
chN_rxuserrdy	GT_RX	O	RX User clocks stable to the Versal ACAP Transceiver. N = Lanes - 1

Notes:

1. JESD204C only uses the above subset of the available signals on the Versal ACAP Transceiver Rx_GT_IP_Interface. For further details see Versal ACAP Transceiver Wizard Product Guide.
2. The signals in Table 2-7 are connected between the JESD204C core and the Versal ACAP Transceiver using Block Automation, please see Chapter 4 for details.

Table 2-8: **RX Core: JESD204_PHY Interface Ports for UltraScale+/UltraScale Devices**

Signal Name	Interface	I/O	Description
gtN_rxdata[63:0]	PHY	I	RX data from JESD204 PHY. N = Lanes - 1
gtN_rxheader[1:0]	PHY	I	RX header from JESD204 PHY. N = Lanes - 1
gtN_misalign	PHY	I	Signal from JESD204 PHY to indicate a misaligned sync header was detected. N = Lanes - 1
gtN_block_sync	PHY	I	Signal from JESD204 PHY to indicate block sync status. N = Lanes -1
gtN_rxcharisk[3:0]	PHY	I	RX Char is K from JESD204 PHY. N = Lanes -1
gtN_rxdisperr[3:0]	PHY	I	RX Disparity Error from JESD204 PHY. N = Lanes -1
gtN_notintable[3:0]	PHY		RX Not In Table Error from JESD204 PHY. N = Lanes -1

Table 2-9: **RX Core: Receive Interface (64B66B Linecoding Only)**

Signal Name	Interface	I/O	Description
rx_tdata [(64*N)-1:0]	m_axis_rx	O	Receive data output. N = Lanes
rx_tvalid	m_axis_rx	O	AXI4-Stream tvalid.
rx_soemb	m_axis_rx	O	Start of extended multi-block boundary indication. Set to 1 to indicate tx_tdata in the following clock cycle is the start of an extended multi-block.
rx_emb_err	m_axis_rx	O	Extended Multi-block Error. Set to 1 on the last block of an extended multi-block if a multi-block alignment error was detected.
rx_crc_err	m_axis_rx	O	CRC error. Set to 1 on the last block of an multi-block if a CRC or Uncorrectable FEC error was detected within the multi-block
rx_cmd_tdata[(32*N)-1:0]	m_axis_rx_cmd	O	Transmit Cmd interface N = Lanes For Meta mode = CRC, Cmd payload is bits [6:0] with bits [31:7] set to Zero. For Meta mode = CMD, Cmd payload is [18:0] with bits [31:19] set to Zero
rx_cmd_tvalid	m_axis_rx_cmd	O	AXI4-Stream tvalid. rx_cmd_tvalid will be set for 1 cycle every multi-block to control the Cmd word flow.
rx_cmd_tready	m_axis_rx_cmd	I	AXI4-Stream tready.
rx_cmd_tuser[N:0]	m_axis_rx_cmd	O	AXI4-Stream tuser. N = Lanes - 1. The tuser data bits are used to signal that a Multiblock Alignment error was detected in the previous multiblock of the associated lane. This may mean the CMD data is invalid.

Receive Interface (8B10B Linecoding Only)

Table 2-10: **Receive Interface Port Descriptions**

Signal Name	Interface	I/O	Description
rx_tdata [(32*N)-1:0]	m_axis_rx	O	Receive data output. N = Lanes
rx_tvalid	m_axis_rx	O	AXI4-Stream tvalid
rx_sof[3:0]	m_axis_rx	O	Start of frame boundary indication. The position of the first byte in a frame is encoded in the same way as tx_sof [3:0]. This signal is asserted one cycle before the AXI4-Stream data. The alignment of the first valid byte is always in byte 0 if the multi-frame size is a multiple of 4 and rx_buffer_delay is a multiple of 4.
rx_somf[3:0]	m_axis_rx	O	Start of multi-frame boundary indication. The position of the first byte of each multi-frame is encoded in the same way as rx_sof.
rx_frm_err[3:0]	m_axis_rx	O	Error in byte. JESD204 specifies that data must be replicated from the previous frame if certain errors occur. The core does not buffer the previous frame. You can choose to implement a frame buffer or use a buffer elsewhere in the system to perform this function if required. The rx_frm_err signal indicates that a single byte error exists in the data stream. There is one bit for each byte of each AXI stream. For example, a four lane interface has four 32-bit AXI streams, the error signal is 16 bits wide with bit 15 of the error signal corresponding to the most significant byte of lane 4 and bit 0 of the error signal corresponding to the least significant byte of lane 1. This signal is synchronous to rx_core_clk and output in the cycle before the data in the same way as rx_sof.

Register Space

The JESD204C core is configured using an AXI4-Lite Register Interface. The register map is shown in the following table.

The RX and TX cores share a common address map and register definitions where possible, exceptions are highlighted.

RECOMMENDED: *Xilinx recommends that if significant configuration changes are made using the control registers (in particular, changes to framing parameters), the core should be reset to ensure that the link is resynchronized using the updated parameters.*

Table 2-11: **Register Address Map**

AXI4-Lite Address	Register Name	64B66B		8B10B	
		TX Access Type	RX Access Type	TX Access Type	RX Access Type
0x000	IP_VERSION	R	R	R	R
0x004	IP_CONFIG	R	R	R	R
0x020	RESET	RW	RW	RW	RW
0x024	CTRL_ENABLE	RW	RW	N/A	N/A
0x028	CTRL_TX_SYNC	N/A	N/A	RW	N/A
0x030	CTRL_MB_IN_EMB	RW	RW	N/A	N/A
0x034	CTRL_SUB_CLASS	RW	RW	RW	RW
0x038	CTRL_META_MODE	RW	RW	N/A	N/A
0x03C	CTRL_8B10B_CFG	N/A	N/A	RW	RW
0x040	CTRL_LANE_ENA	RW	RW	RW	RW
0x044	CTRL_RX_BUF_ADV	N/A	RW	N/A	RW
0x048	CTRL_TEST_MODE	N/A	N/A	RW	RW
0x04C	CTRL_RX_MBLOCK_TH	N/A	RW	N/A	N/A
0x050	CTRL_SYSREF	RW	RW	RW	RW
0x054	STAT_LOCK_DEBUG	N/A	R	N/A	N/A
0x058	STAT_RX_ERR	N/A	N/A	N/A	R
0x05C	STAT_RX_DEBUG	N/A	N/A	N/A	R
0x060	STAT_STATUS	R	R	R	R
0x064	CTRL_IRQ	RW	RW	RW	RW
0x068	STAT_IRQ	R	R	R	R
0x070	CTRL_TX_ILA_CFG0	N/A	N/A	RW	N/A
0x074	CTRL_TX_ILA_CFG1	N/A	N/A	RW	N/A
0x078	CTRL_TX_ILA_CFG2	N/A	N/A	RW	N/A
0x07C	CTRL_TX_ILA_CFG3	N/A	N/A	RW	N/A
0x080	CTRL_TX_ILA_CFG4	N/A	N/A	RW	N/A

Table 2-11: **Register Address Map *(Cont'd)***

AXI4-Lite Address	Register Name	64B66B		8B10B	
		TX Access Type	RX Access Type	TX Access Type	RX Access Type
0x400[1]	(Lane 0) STAT_RX_BUF_LVL	N/A	R	N/A	R
0x404[1]	(Lane 0) CTRL_TX_ILA_LID	N/A	N/A	RW	N/A
0x410[1]	(Lane 0) STAT_RX_ERROR_CNT0	N/A	R	N/A	N/A
0x414[1]	(Lane 0) STAT_RX_ERROR_CNT1	N/A	R	N/A	N/A
0x420[1]	(Lane 0) STAT_LINK_ERR_CNT	N/A	N/A	N/A	R
0x424[1]	(Lane 0) STAT_TEST_ERR_CNT	N/A	N/A	N/A	R
0x428[1]	(Lane 0) STAT_TEST_ILA_CNT	N/A	N/A	N/A	R
0x42C[1]	(Lane 0) STAT_TEST_MF_CNT	N/A	N/A	N/A	R
0x430[1]	(Lane 0) CTRL_RX_ILA_CFG0	N/A	N/A	N/A	R
0x434[1]	(Lane 0) CTRL_RX_ILA_CFG1	N/A	N/A	N/A	R
0x438[1]	(Lane 0) CTRL_RX_ILA_CFG2	N/A	N/A	N/A	R
0x43C[1]	(Lane 0) CTRL_RX_ILA_CFG3	N/A	N/A	N/A	R
0x440[1]	(Lane 0) CTRL_RX_ILA_CFG4	N/A	N/A	N/A	R
0x444[1]	(Lane 0) CTRL_RX_ILA_CFG5	N/A	N/A	N/A	R
0x448[1]	(Lane 0) CTRL_RX_ILA_CFG6	N/A	N/A	N/A	R
0x44C[1]	(Lane 0) CTRL_RX_ILA_CFG7	N/A	N/A	N/A	R
0x460[1]	(Lane 0) CTRL_TX_GT	RW	RW	RW	RW
0X464[1]	(Lane 0) CTRL_RX_GT	RW	RW	RW	RW

Notes:

1. As shown, lane 0 registers start at 0x400. Lane 1 registers occupy the equivalent space starting at 0x480, lanes 2-7 follow the same pattern (i.e., lane 2 = 0x500, lane 3 = 0x580, etc.).

Table 2-12: **IP_VERSION**

Bits	Default Value	Description
31:24	–	Version: Major
23:16	–	Version: Minor
15:8	–	Version: Revision
7:0	–	Reserved (read 0x00)

Register Address Map

Table 2-13: **IP_CONFIG**

Bits	Default Value	Description
18	-	1= FEC Included 0=FEC not included (FEC is only available when configured for 64B66B)
17	-	1 = Core is 64B66B 0 = Core is 8B10B
16	–	1 = Core is TX 0 = Core is RX
3:0	–	Number of lanes in core.

Register Address Map

Table 2-14: **RESET**

Bits	Default Value	Description
31:24[1]	–	gt_mst_reset_busy[7:0] 1-bit per enabled GT lane. These bits show the inversion of the mstresetdone inputs from the Versal GTs.
23:16[1]	–	gt_pma_reset_busy[7:0] 1-bit per enabled GT lane. These bits show the inversion of the mstresetdone inputs from the Versal GTs.
7[1]	–	gt_reset_busy 1 = GTs are still in reset 0 = GTs have finished reset
6[1]	–	gt_powergood input state 1 = gt_powergood asserted 0 = gt_powergood deasserted
5[1]	–	Core register Reset Register State 1 = tx/rx_reset asserted 0 = tx/rx_reset deasserted
4[1]	–	Core External Reset Pin State 1 = tx/rx_core_reset asserted 0 = tx/rx_core_reset deasserted
0	0	Reset. (not self-clearing) 1 = put core into reset 0 = Release core from reset After setting this bit to 1 and clearing back to 0, this bit will read back 1 until the reset process has completed.

Notes:
1. For Versal ACAPs only.

Register Address Map

Table 2-15: **CTRL_ENABLE**

Bits	Default Value	Description
1	0	Enable Data Interface. 1 = Enables the AXI4-Stream Data interface and transmits/receives data on the link. 0 = The link will be transmitting/receiving scrambled 0s
0	0	Enable Cmd interface. 1 = Enables the AXI4-Stream Cmd interface and the associated processing of the sync header meta data. 0 = Cmd words will be zeroed.

Register Address Map

Table 2-16: **CTRL_TX_SYNC**

Bits	Default Value	Description
0	0	tx_sync_force. Force on 8B10B transmitter. When set to 1, this register overrides the value on the tx_sync pin.

Register Address Map

Table 2-17: **CTRL_MB_IN_EMB**

Bits	Default Value	Description
7:0	0x1	Number of multi-blocks in an extended multi-block. Program this register with the actual value. **Note**: 0 is Not valid.

Register Address Map

Table 2-18: **CTRL_SUB_CLASS**

Bits	Default Value	Description
1:0	0x1	Sub Class: 0 = Subclass 0 1 = Subclass 1 2 = Subclass 2 (8B10B only)

Register Address Map

Table 2-19: **CTRL_META_MODE**

Bits	Default Value	Description
1:0	0x0	Meta Mode: 0 = CRC12 1 = N/A 2 = CMD 3 = FEC

Register Address Map

Table 2-20: **CTRL_8B10B_CFG**

Bits	Default Value	Description
31:24	0x3	ILA multi-frames. Multiframes in the Transmitted Initial Lane Alignment Sequence. Parameter Range: 4–256; program the register with required value minus 1.
21:20	0x0	Reserved. Must be set to zero.
19	0	Link Error Counters Enable: 1 = Enable Link Error counters (Link errors are counted and reported using Link Error Count registers per lane) 0 = Disable Link Error counters
18	0	Error Reporting via sync: 1 = Error reporting using SYNC interface Enabled 0 = Error reporting using SYNC interface Disabled
17	1	ILA Required: 1 = Enable ILA Support 0 = Disable ILA Support
16	1	Scrambling: 1 = Enable Scrambling 0 = Disable Scrambling
12:8	0xF	Frames per Multiframe (K) Parameter range 1–32; Program register with required value minus 1 (for example, for K = 16, 0x0F should be programmed)
7:0	0x1	Octets per Frame (F) Parameter range 1–256; Program register with required value minus 1 (for example, for F = 4, 0x03 should be programmed)

Register Address Map

Table 2-21: **CTRL_LANE_ENA**

Bits	Default Value	Description
7:0	-	Lane enable register. Default is all lanes enabled. Set 1 bit per lane (bit 0 = lane 0, bit 1 = lane 1 etc.) ***Note:*** If any lanes are disabled, the lane ID and the number of lanes per link registers should be reprogrammed accordingly.

Register Address Map

Table 2-22: **CTRL_RX_BUF_ADV**

Bits	Default Value	Description
9:0	0x0	Advance the release of the receiver buffer: For 64B66B linecoding, advance the release of the buffer by N 64 bit words. For 8B10B linecoding, advance the release of the buffer by N octets.

Register Address Map

Table 2-23: **CTRL_TEST_MODE**

Bits	Default Value	Description
30:28	0x0	GT loopback: 00: Normal operation 10: Near-end physical medium attachment (PMA) loopback 01,11: Reserved
27:3	-	Reserved
2:0	0x0	Test mode select (8B10B mode): 000 = Normal operation 001 = Transmit receive /K28.5/ indefinitely(2) 010 = Synchronize as normal then transmit/receive repeated ILA sequences.(2) 101 = Transmit Modified Random Pattern RPAT (TX Only)(1) 111 = Transmit Scrambled Jitter Pattern JSPAT (TX Only)(1)

Notes:

1. These test modes are only applicable to the JESD204C 8B10B transmitter IP. They are used to set the transceiver to output specific patterns that may be used to evaluate the electrical characteristics of a link using tools such as IBERT. A JESD204 8B10B receiver core will not synchronize or function if these test patterns are received.
2. Physical layer test modes are made available through the Versal ACAP Transceiver or JESD204 PHY register interface.

Register Address Map

Table 2-24: **CTRL_RX_MBLOCK_TH**

Bits	Default Value	Description
2:0	0x0	MB lock threshold. How many correct/incorrect multi-block alignment markers are required to achieve/lose multi-block lock. The actual value used is 1 plus the number in this register.

Register Address Map

Table 2-25: **CTRL_SYSREF**

Bits	Default Value	Description
23:16	0x0	(For 64b66b cores only) SYSREF Delay: Add additional delay to SYSREF alignment LEMC. 0xFF = 255 core_clk cycles delay .... 0x00 = 0 core_clk cycles delay. This register is used to retard the phase of the LEMC.
19:16	0x0	SYSREF Delay: Add additional delay to SYSREF alignment of LMFC. 1111 = 15 core_clk cycles delay .... 0000 = 0 core_clk cycles delay This register is used to retard the phase of the LMFC.
10:8	0x0	SYSREF Tolerance: Specify a tolerance value in core_clk cycles for SYSREF detection when in SYSREF Always mode. If a SYSREF event is detected within +/- tolerance core_clk cycles from its expected position, no error signal will be issued.
1	0	SYSREF Required on Re-Sync 1 = Following a Link Re-Sync event, a SYSREF event is required to re-align the local LMFC/LEMC before the link will operate. 0 = No SYSREF is required to restart a link after a Re-sync event.
0	0	SYSREF Always 1 = The core will align the LMFC/LEMC counter on all SYSREF events. 0 = The core will only align the LMFC/LEMC counter on the first SYSREF event following a reset, all subsequent SYSREF events will be ignored.

Register Address Map

Table 2-26: **STAT_LOCK_DEBUG**

Bits	Default Value	Description
23:16	-	Lane indicator multi-block aligned. 1 bit per lane. Set to 1 when multi-block alignment is achieved. 0 otherwise.
7:0	-	Lane indicator 64B66B sync header aligned. 1 bit per lane. Set to 1 when sync header alignment is achieved. 0 otherwise.

Register Address Map

Table 2-27: **STAT_RX_ERR**

Bits	Default Value	Description
31:28	-	RX Error status lane 7
27:24	-	RX Error status lane 6
23:20	-	RX Error status lane 5
19:16	-	RX Error status lane 4
15:12	-	RX Error status lane 3
11:8	-	RX Error status lane 2
7:4	-	RX Error status lane 1
3:0	-	RX Error status lane 0 Bit 3: unused Bit 2: Unexpected K-character(s) received Bit 1: Disparity Error(s) received Bit 0: Not in table Error(s) received Each bit indicates that 1 or more errors of that type have been received in Lane 0 since the register was last read. All status bits are cleared to 0 on read of this register.

Register Address Map

Table 2-28: **STAT_RX_DEBUG**

Bits	Default Value	Description
31:28	-	Link Debug status Lane 7 as per lane 0
27:24	-	Link Debug status Lane 6 as per lane 0
23:20	-	Link Debug status Lane 5 as per lane 0
19:16	-	Link Debug status Lane 4 as per lane 0
15:12	-	Link Debug status Lane 3 as per lane 0
11:8	-	Link Debug status Lane 2 as per lane 0
7:4	-	Link Debug status Lane 1 as per lane 0

Table 2-28: **STAT_RX_DEBUG *(Cont'd)***

Bits	Default Value	Description
3:0	-	Link Debug status Lane 0 Bit 3: 1 = Start of Data was Detected(1) Bit 2: 1 = Start of ILA was Detected(1) Bit 1: 1 = Lane has Code Group Sync(2) Bit 0: 1 = Lane is currently receiving K28.5's (BC alignment characters)(2)

Notes:

1. The status bits 3:2 latch when set and are cleared on read or when the core is reset. If the core is streaming data when these bits are cleared, they are instantly set again. The purpose of these bits is to detect whether these conditions have occurred since SYNC was asserted.
2. The status bits 1:0 show instantaneous status.

Register Address Map

Table 2-29: **STAT_STATUS**

Bits	Default Value	Description
15	-	8B10B Alignment Error: 1= An 8B10B RX misalignment has been detected. Misalignment is determined by monitoring the Multi-frame framing characters. If eight consecutive framing characters are detected in misaligned positions, then this bit is asserted.
14	-	8B10B RX started: 1 = The link has started outputting data on the AXI4-Stream port. This bit is applicable to an 8B10B RX only.
13	-	8B10B CGS status: 1 = The link has achieved Code Group Sync. This bit is applicable to an 8B10B RX only.
12	-	8B10B SYNC status: 1 = The receiver has signaled SYNC has been achieved. This bit is applicable to an 8B10B link only.
10	-	Buffer Overflow error. 1 = The receiver buffer has overflowed.
5	-	64B66B Multi-block Lock Status: 1 = Multi-block lock achieved on all lanes This bit is a logical OR of the individual lane status bits.
4	-	64B66B Sync Header Lock Status: 1 = Sync Header lock achieved on all lanes. This bit is a logical OR of the individual lane status bits
2	-	SYSREF error. A sysref was detected out of phase with the local extended multi-block clock.
1	-	SYSREF captured.
0	-	Interrupt pending.

Table 2-30: **CTRL_IRQ**

Bits	Default Value	Description
14	0	1 = Enable interrupt on 8B10B RX AXI4-Stream data start.
13	0	1 = Enable interrupt on 8B10B RX Resync request.
12	0	1 = Enable interrupt on 8B10B SYNC assertion.
10	0	1 = Enable Interrupt on overflow Error.
9	0	1 = Enable Interrupt on 64B66B FEC Error.
8	0	1 = Enable Interrupt on 64B66B CRC Error.
7	0	1 = Enable Interrupt on 64B66B Multi-block Error.
6	0	1 = Enable Interrupt on 64B66B Block Sync Error.
5	0	1 = Enable Interrupt on Loss of 64B66B Multi-block Lock.
4	0	1 = Enable Interrupt on Loss of 64B66B Sync Header Lock.
2	0	1 = Enable Interrupt on SYSREF Error.
1	0	1 = Enable Interrupt on SYSREF Received.
0	0	Global Interrupt Enable: Must be set for any interrupt to function.

Register Address Map

Table 2-31: **STAT_IRQ**

Bits	Default Value	Description
14	-	1 = 8B10B RX AXI4-Stream data start interrupt triggered.
13	-	1 = 8B10B RX Resync request interrupt triggered.
12	-	1 = 8B10B SYNC assertion interrupt triggered.
10	-	1 = Overflow Error Interrupt triggered.
9	-	1 = 64B66B FEC Error detected Interrupt triggered.
8	-	1 = 64B66B CRC Error detected Interrupt triggered.
7	-	1 = 64B66B Multi-block Error detected Interrupt triggered.
6	-	1 = 64B66B Block Sync Error detected Interrupt triggered.
5	-	1 = 64B66B Multi-block Lock Status Interrupt triggered.
4	-	1 = 64B66B Sync Header Lock Status Interrupt triggered.
2	-	1 = SYSREF Error Interrupt triggered.
1	-	1 = SYSREF Received Interrupt triggered.

Register Address Map

Table 2-32: **CTRL_TX_ILA_CFG0**

Bits	Default Value	Description
11:8	0x0	BID (Bank ID). Binary value.
7:0	0x0	DID (Device ID). Binary value.

Register Address Map

Table 2-33: **CTRL_TX_ILA_CFG1**

Bits	Default Value	Description
31:26	-	Reserved
25:24	0x0	CS (Control bits per Sample). Binary value.
23:21	-	Reserved
20:16	0x0	N' (Totals bits per Sample). Binary value minus 1.
15:13	-	Reserved
12:8	0x0	N (Converter Resolution). Binary value minus 1.
7:0	0x0	M (Converters per Device). Binary value minus 1.

Register Address Map

Table 2-34: **CTRL_TX_ILA_CFG2**

Bits	Default Value	Description
28:24	0x0	CF (Control Words per Frame). Binary value.
16	0	HD (High Density format)
12:8	0x0	S (Samples per Converter per Frame). Binary value minus 1.

Register Address Map

Table 2-35: **CTRL_TX_ILA_CFG3**

Bits	Default Value	Description
31:17	-	Reserved
16	0	ADJDIR (Adjust Direction) [Subclass 2 Only]. Binary value.
15:9	-	Reserved
8	0	PHADJ (Phase Adjust Request) [Subclass 2 Only]. Binary value.
7:4	-	Reserved
3:0	0x0	ADJCNT (Phase Adjust Count) [Subclass 2 Only]. Binary value. RX: captured configuration data from the ILA sequence (per lane). TX: Sets the values to be transmitted in the ILA sequence for all lanes.

Register Address Map

Table 2-36: **CTRL_TX_ILA_CFG4**

Bits	Default Value	Description
15:8	0x0	RES2 (Reserved Field 2)
7:0	0x0	RES1 (Reserved Field 1)

Register Address Map

Table 2-37: **STAT_RX_BUF_LVL**

Note: This is a *Per Lane* Register

Bits	Default Value	Description
9:0	-	Buffer fill level. The amount of data in the receiver buffer for lane 0. For 64B66B linecoding: The value returned is the number of 64-bit words in the buffer. For 8B10B Linecoding: The value returned is the number of bytes in the buffer.

Register Address Map

Table 2-38: **CTRL_TX_ILA_LID**

Note: This is a *Per Lane* Register

Bits	Default Value	Description
31:21	-	Reserved
20:16	L	Number of lanes per link (binary value minus 1). The default value for all lanes is L (total number of lanes minus 1). These values should be programmed when: • Not all lanes in the generated IP core are enabled. • A single TX core is used to drive multiple DAC devices. • Multiple TX cores are combined to create links with more than eight lanes.
15:5	-	Reserved
4:0	N	ID of lane N. Value can be anywhere between 0 and 31. The default value N is set to the lane number. For interfaces using more than 8 lanes and hence multiple JESD204 cores, this register should be programmed to ensure each lane has the correct identifier.

Register Address Map

Table 2-39: **STAT_RX_ERROR_CNT0**

Note: This is a *Per Lane* Register. The counts are cumulative and are cleared on read or reset. The counts should be cleared with a register read when a link has been successfully established.

Bits	Default Value	Description
31:16	-	CRC error counter.

Table 2-39: **STAT_RX_ERROR_CNT0**

Note: This is a *Per Lane* Register. The counts are cumulative and are cleared on read or reset. The counts should be cleared with a register read when a link has been successfully established.

Bits	Default Value	Description
15:8	-	64B66B Multi-block alignment error counter.
7:0	-	64B66B Sync Header alignment error counter.

Register Address Map

Table 2-40: **STAT_RX_ERROR_CNT1**

Note: This is a *Per Lane* Register. The counts are cumulative and are cleared on read or reset. The counts should be cleared with a register read when a link has been successfully established.

Bits	Default Value	Description
31:16	-	64B66B FEC uncorrected errors counter.
15:0	-	64B66B FEC corrected errors counter.

Register Address Map

Table 2-41: **STAT_LINK_ERR_CNT**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:0	-	Link Error Count Count of total received link errors (per lane) when Link Error Counters is Enabled. Errors counted are Disparity or Not In Table errors indicated by the lane. The error counter can be reset by disabling and re-enabling the Link Error Counters Enable, bit 19 in the CTRL_8B10B_CFG register.

Register Address Map

Table 2-42: **STAT_TEST_ERR_CNT**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:0	-	Test Mode Error Count Count of Errors received in Data link Layer test modes. Test Mode = 001 (Continuous K28.5): counts any non K28.5 characters received Test Mode = 010 (Continuous ILA): counts any unexpected characters received This count resets to zero on transition to an active test mode and retains any count value on transition out of an active test mode.

Register Address Map

Table 2-43: **STAT_TEST_ILA_CNT**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:0	-	Test Mode ILA Count Count of total ILA Sequences received when Test Mode = 010 (Continuous ILA) This count resets to zero on transition to Test Mode = 010, and retains any count value on transition out of test mode.

Register Address Map

Table 2-44: **STAT_TEST_MF_CNT**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:0	-	Test Mode Multiframe Count Count of total ILA Multiframes received when Test Mode = 010 (Continuous ILA) This count resets to zero on transition to Test Mode = 010 and retains any count value on transition out of test mode.

Register Address Map

Table 2-45: **CTRL_RX_ILA_CFG0**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:11	-	Reserved
10:8	-	JESDV (JESD204 version): 000=JESD204A 001=JESD204B 010 = JESD204C
7:3	-	Reserved
2:0	-	SUBCLASS: 000=Subclass0 001=Subclass1 010=Subclass2

Register Address Map

Table 2-46: **CTRL_RX_ILA_CFG1**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:8	-	Reserved
7:0	-	F (Octets per Frame). Binary value minus 1.

Register Address Map

Table 2-47: **CTRL_RX_ILA_CFG2**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:5	-	Reserved
4:0	-	K (Frames per Multiframe). Binary value minus 1.

Register Address Map

Table 2-48: **CTRL_RX_ILA_CFG3**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:29	-	Reserved
28:24	-	L (Lanes per Link). Binary value minus 1.
23:21	-	Reserved
20:16	0x0	LID (Lane ID). Binary value.
15:12	-	Reserved
11:8	0x0	BID (Bank ID). Binary value.
7:0	0x0	DID (Device ID). Binary value.

Register Address Map

Table 2-49: **CTRL_RX_ILA_CFG4**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:26	-	Reserved
25:24		CS (Control bits per Sample). Binary value.
23:21	-	Reserved
20:16		N' (Totals bits per Sample). Binary value minus 1.
15:13	-	Reserved

Table 2-49: **CTRL_RX_ILA_CFG4**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
12:8		N (Converter Resolution). Binary value minus 1.
7:0		M (Converters per Device). Binary value minus 1.

Register Address Map

Table 2-50: **CTRL_RX_ILA_CFG5**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:29	-	Reserved
28:24	0x0	CF (Control Words per Frame). Binary value.
23:17	-	Reserved
16	0	HD (High Density format)
15:13	-	Reserved
12:8	0x0	S (Samples per Converter per Frame). Binary value minus 1.
7:1	-	Reserved
0	-	SCR (Scrambling Enable) [RX only, not writeable for TX] 1 = enabled

Register Address Map

Table 2-51: **CTRL_RX_ILA_CFG6**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:17	-	Reserved
16	0	ADJDIR (Adjust Direction) [Subclass 2 Only]. Binary value.
15:9	-	Reserved
8	-	PHADJ (Phase Adjust Request) [Subclass 2 Only]. Binary value.
7:4	-	Reserved
3:0	0x0	ADJCNT (Phase Adjust Count) [Subclass 2 Only]. Binary value. RX: captured configuration data from the ILA sequence (per lane). TX: Sets the values to be transmitted in the ILA sequence for all lanes.

Register Address Map

Table 2-52: **CTRL_RX_ILA_CFG7**

Note: This is a *Per Lane* Register.

Bits	Default Value	Description
31:24	-	Reserved
23:16	0x0	FCHK (Checksum) [RX only, not writeable for TX]. Binary value.
15:8	0x0	RES2 (Reserved Field 2)
7:0	0x0	RES1 (Reserved Field 1)

Register Address Map

Table 2-53: **CTRL_TX_GT**

Note: This is a *Per Lane* Register for Versal ACAPs only.

Bits	Default Value	Description
31:29	-	Reserved
28:24	0x0	TXPRECURSOR Transmitter pre-cursor pre-emphasis control
20:16	0x0	TXPOSTCURSOR Transmitter post-cursor pre-emphasis control
12:8	0x0	TXDIFFCTRL Driver swing control
4	0	TXINHIBIT Set High to inhibit transmission of TX data
3	0	TXELECIDLE Set High to force idle signal on transmitter output
2:1	0x0	TXPD TX power down
0	0	TXPOLARITY Set High to invert the polarity of the outgoing TX data

Register Address Map

Table 2-54: **CTRL_RX_GT**

Note: This is a *Per Lane* Register for Versal ACAPs only.

Bits	Default Value	Description
31:3	-	Reserved
2:1	0x0	RXPD RX power down
0	0	RXPOLARITY Set High to invert the polarity of the incoming RX data

Register Address Map

Designing with the Core

This chapter provides a general description of how to use the JESD204C core in your designs and should be used in conjunction with Chapter 2, Product Specification, which describes specific core interfaces.

General Design Guidelines

This section describes the steps required to turn a JESD204C core into a fully-functioning design with user-application logic. It is important to know that not all implementations require all of the design steps listed in this chapter. Follow the logic design guidelines in this manual carefully.

Use the Example Design as a Starting Point

Each instance of the JESD204C core created by the Vivado® Design Suite is delivered with an example design that can be implemented in an FPGA and simulated. This design can be used as a starting point for your own design or can be used to troubleshoot your application, if necessary.

See Example Design for information about using and customizing the example designs for the JESD204C core.

Know the Degree of Difficulty

JESD204C designs are challenging to implement in any technology, and the degree of difficulty is further influenced by:

- Maximum system clock frequency
- Targeted device architecture
- Nature of your application

All JESD204C implementations require careful consideration of system performance requirements. Pipelining, logic mapping, placement constraints, and logic duplication are all methods that help boost system performance.

Keep it Registered

To simplify timing and increase systems performance in an FPGA design, keep all inputs and outputs between your application and the core registered. This means that all inputs and outputs from your application should come from, or connect to, a flip-flop. While registering signals may not be possible for all paths, it simplifies timing analysis and makes it easier for the Xilinx tools to place-and-route the design.

Recognize Time-Critical Signals

The XDC provided with the Example Design for the core identifies the critical signals and the timing constraints that should be applied. See Constraining the Core for further information.

Use Supported Design Flows

The core is synthesized in the Vivado IDE and is delivered as Verilog. The example implementation scripts currently provided use Vivado synthesis as the synthesis tool for the IP integrator example design that is delivered with the core. Other synthesis tools can be used.

Make Only Allowed Modifications

The JESD204C core is not user-modifiable. Any modifications can have adverse effects on system timing and protocol compliance. Supported user configurations of the JESD204C core can only be made by selecting the options from within the Vivado Customize IP dialog box and using the top-level parameters described in this document. See Design Flow Steps for more information.

Recommended Design Experience

Although the JESD204C core is a fully-verified solution. The challenges associated with implementing a complete design vary depending on the configuration and functionality of the application. For best results, previous experience in building high-performance, pipelined FPGA designs using Xilinx implementation tools and the XDC is recommended.

Contact your local Xilinx representative for a closer review and estimation for your specific requirements.

Subclass Mode

The JESD204C core supports operation in two JESD204C Subclass modes (0 and 1) for 64B66B linecoding, and three Subclass modes (0, 1 and 2) for 8B10B linecoding.

This is controlled by a register setting. By default the core operates in Subclass 1 mode.

The core pinout for 64B66B supports both subclass modes of operation, however an externally generated SYSREF is required for Subclass 1 operation. For Subclass 0, the SYSREF input signal is not required and can be tied off.

The core pinout for 8B10B supports all three subclass modes of operation, however an externally generated SYSREF is required for Subclass 1 operation. For Subclasses 0 and 2, the SYSREF input signal is not required and can be tied off.

Subclass 0

Subclass 0 is supported for both 64B66B and 8B10B linecoding. Subclass 0 does not support Deterministic Latency and the SYSREF input is not required.

Subclass 1

Subclass 1 is supported for both 64B66B and 8B10B linecoding. Subclass 1 supports deterministic latency through the use of a common SYSREF signal between the converter and logic device. The SYSREF signal is generated external to the core, and is distributed to all devices within a system. SYSREF is permitted by the JESD204C standard to be either a *one-shot*, *periodic*, or *gapped periodic*. The JESD204C core is capable of operating with any of these selections. The timing and clocking requirements for the reliable capture of SYSREF are key to achieving reliable deterministic latency.

Subclass 2

Subclass 2 is only supported for 8B10B linecoding. Subclass 2 supports deterministic latency using only the SYNC signal. The timing and clocking requirements for the launch (by an RX core), and capture (by a TX core) of the SYNC signal are key to achieving reliable deterministic latency. Care must be taken to ensure the timing of this signal is met.

Programming the Core

Run time operation of the JESD204C core is configured through an AXI4-Lite register interface. See Register Space for details of the register map and available configuration registers.

For correct operation and bring-up of a JESD204C link, it is important that the major framing and link operation parameters match at both ends of the link. These parameters are determined by the configurations available in the ADC/DAC converter device to which the core is interfacing.

For 64B66B Linecoding, these are:

- Meta Mode
- Multi-blocks in Extended Multi-block

- Subclass mode
- SYSREF handling (for subclass 1 mode)

For 8B10B Linecoding, these are:

- Octets per frame
- Frames per Multi-frame
- Scrambling On/Off
- Subclass mode
- SYSREF handling (for subclass 1 mode)

For 8B10B transmitter cores, in addition to the above parameters, some of the additional content of the configuration data which is transmitted in the ILA sequence at link start-up is also programmed through the register interface. The data values transmitted in the ILA configuration data are not normally critical to the operation of the link, but this is dependent on the behavior of the receiving device.

For 8B10B receive cores, the configuration data received in the ILA sequence is captured for each lane and can be examined using the register interface.

After programming the link parameters the JESD204C core must be reset to restart the link using the newly programmed values. If the JESD204C core is not reset after programming, the new parameters will not be used.

Clocking

IMPORTANT: *It is strongly recommended that you use one of the clocking schemes presented in this section. Use of alternative clocking schemes may lead to design failure.*

The JESD204C specification [Ref 9] does not define specific serial line rates for any JESD204C link, but a valid range of line rates from 312.5 Mb/s to 32 Gb/s. The JESD204C core supports 8B10B linecoding at line rates from 1 Gb/s to 16,375 Gb/s (depending on the part and speed grade selection) and 64B66B linecoding at line rates from 1 Gb/s to 32 Gb/s (depending on the part and speed grade selection). In most instances, the serial line rate selection is governed by the specifications of the ADC/DAC device(s) to which the core is interfaced. The required operating serial line rate directly relates to the clock rate at which the core logic operates (core clock); the serial line rate also governs the selection of the reference clock required by the transceiver(s).

Core Clock 64B66B Linecoding

The JESD204C 64B66B core operates using a 64-bit (8-byte) datapath. The core clock frequency is always the line rate divided by 66. For example, for a serial line rate of 16.5 Gb/s, the core clock frequency is 250 MHz.

The AXI4-Stream RX / TX Data and Cmd interfaces operate at this core clock frequency. TX and RX core clock should be used as the clock source for these interfaces.

Core Clock 8B10B Linecoding

The JESD204C 8B10B core operates using a 32-bit (4-byte) datapath. The core clock frequency is always the line rate divided by 40. For example, for a serial line rate of 12.5 Gb/s, the core clock frequency is 312.5 MHz.

The AXI4-Stream RX and TX Data interfaces operate at this core clock frequency. TX and RX core clock should be used as the clock source for these interfaces.

Reference Clock

In Versal™ ACAP, the GTY serial transceivers in the Versal ACAP Transceivers require a stable, low-jitter reference clock which has a device and speed grade-dependent range. For UltraScale+™ and UltraScale™ devices, the GTH/GTY serial transceivers in the JESD204_PHY require a stable, low-jitter reference clock which has a device and speed grade-dependent range.

In some circumstances, it can be advantageous to use the same clock frequency for both core clock and reference clock. However this might not always be practical. It is important to understand the limitations imposed on the reference clock and core clock, together with system-level implications such as the synchronous capture of SYSREF for Subclass 1.

JESD204_PHY Outclocks for UltraScale+/UltraScale Devices

The JESD204_PHY IP core generates the clocks and `rxoutclk` at the correct frequency to use as `core_clk`. However the output phase of these clocks varies from reset to reset. This means that for JESD204 interfaces the JESD204_PHY IP core `outclock` ports may only be used to drive `core_clk` when Subclass 0 is used and deterministic latency is not required. For systems that require deterministic latency and therefore use Subclass 1 or Subclass 2 the JESD204_PHY IP core `outclocks` should not be used.

Versal ACAP Transceiver Outclocks

The Versal ACAP Transceiver Wizard generates `txoutclk` and `rxoutclk` at the correct frequency to use as `core_clk`. However the output phase of these clocks varies from reset to reset. This means that for JESD204 interfaces the Versal ACAP Transceiver `outclock` ports may only be used to drive `core_clk` when Subclass 0 is used and deterministic latency is not required. For systems that require deterministic latency and therefore use Subclass 1 or Subclass 2 the Versal ACAP Transceiver `outclocks` should not be used.

The rest of this section details the valid clocking architecture options.

AXI4-Lite Interface Clock

The JESD204C core is configured and monitored through an AXI4-Lite processor interface. The clock for this interface is separate and independent from the core and reference clocks.

DRP Clock for UltraScale+/UltraScale Devices

JESD204C system implementation requires the use of a JESD204 PHY core. The JESD204_PHY core must be supplied with a DRP clock (see *JESD204 PHY LogiCORE IP Product Guide* PG198 [Ref 11]).

Separate Transceiver Reference and Core Clocks

For JESD204C, the most generic and flexible clocking scheme uses separate transceiver reference and JESD204C core clocks supplied to the FPGA. In this configuration, the reference and core clocks are physically separate and can be run at independent, but related, frequencies, without additional constraints.

The reference clock can be run at any frequency within the limitations of the transceiver for the selected line rate. The core clock always runs at the required rate (1/66th or 1/40th of the serial line rate). This configuration is shown in the following two figures for 64B66B and 8B10B linecoding.

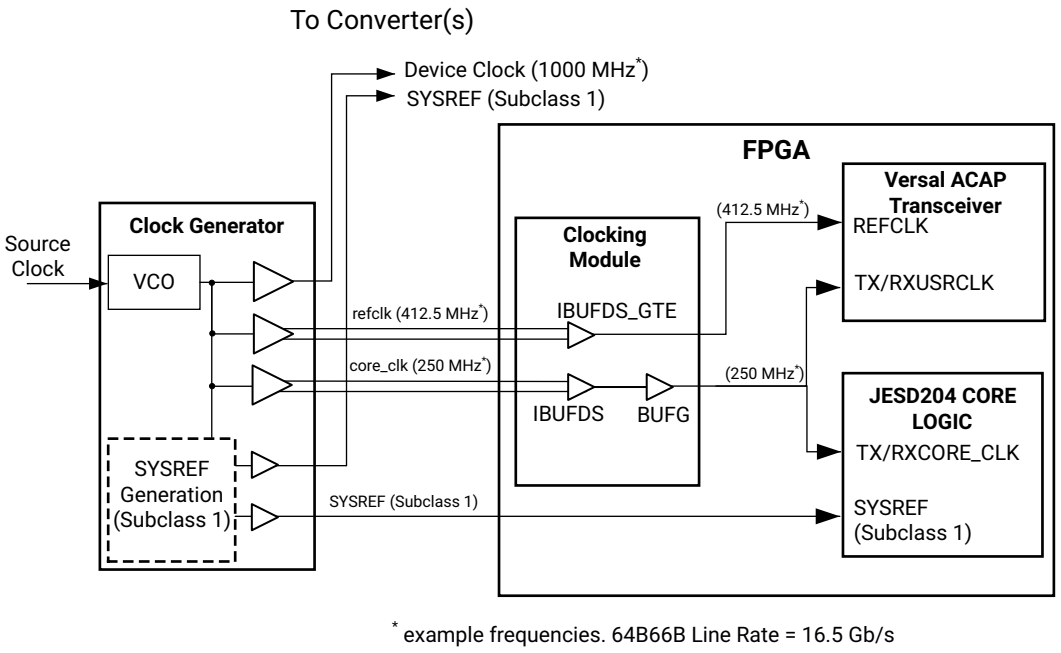

Figure 3-1: **Separate Transceiver Reference and Core Clocks: 64B66B Example for Versal ACAPs**

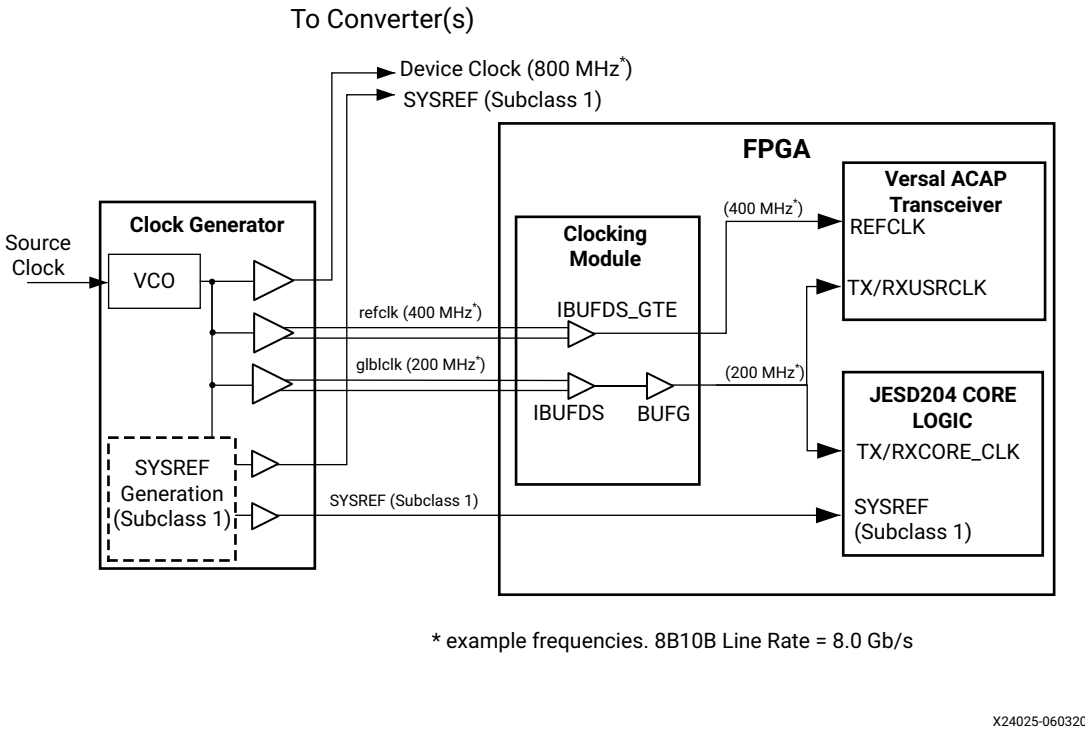

Figure 3-2: **Separate Transceiver Reference Clock and Core Clock: 8B10B Example for Versal ACAPs**

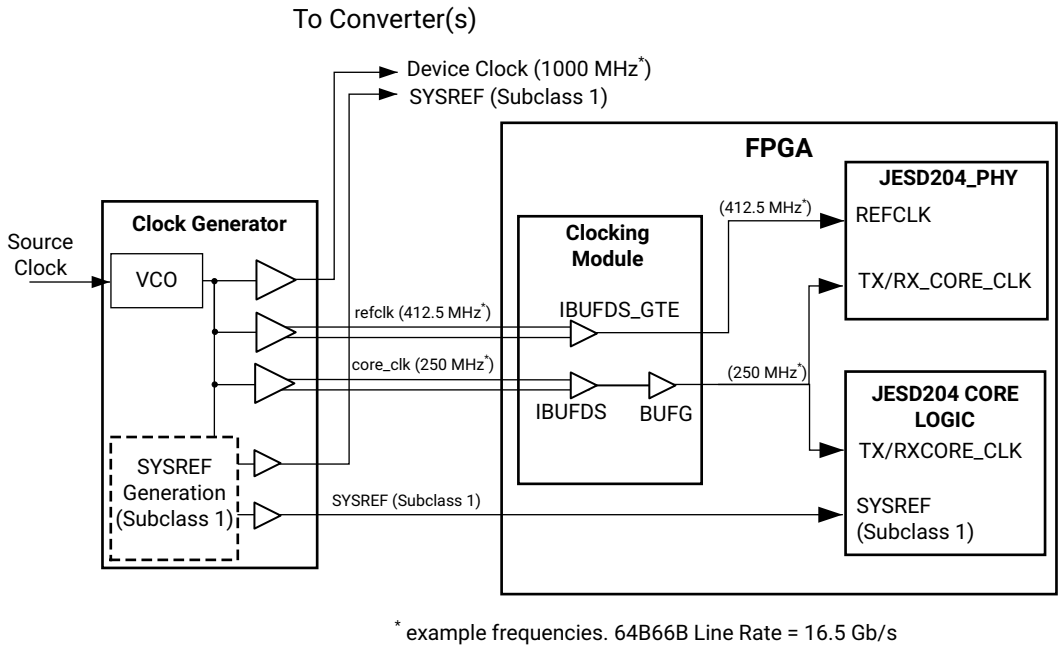

Figure 3-3: **Separate Transceiver Reference and Core Clocks: 64B66B Example for UltraScale+/ UltraScale Devices**

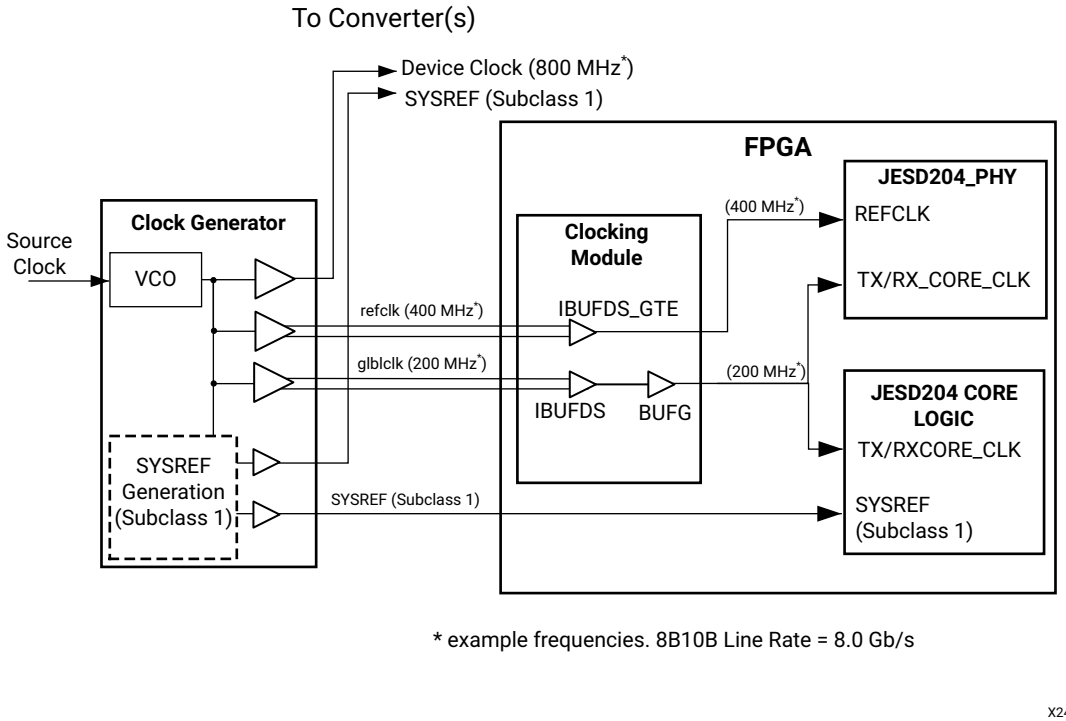

Figure 3-4: **Separate Transceiver Reference Clock and Core Clock: 8B10B Example for UltraScale+/UltraScale Devices**

Transceiver Reference Clock Used as Core Clock

For some systems it is possible to run a single clock input which acts as both the transceiver reference clock and the JESD204C core clock. While this configuration can sometimes simplify a system design, it is not always compatible.

For 64B66B systems in Versal ACAPs, the required core clock frequency is usually not suitable for use as a reference clock for a Versal ACAP Transceiver configured to use the RPLL. Therefore, the LCPLL should be used if a single clock is required. In this configuration, the input transceiver reference clock must always be the required rate (1/66th of the serial line rate for 64B66B systems or 1/40th of the serial line rate for 8B10B systems).

For 64B66B systems in UltraScale+/UltraScale devices, the required core clock frequency is not suitable for use as a reference clock for a JESD204 PHY configured to use the CPLL (therefore QPLL 0 or 1 must be used if a single clock is required, or two clocks must be supplied if the CPLL must be used). In this configuration, the input transceiver reference clock must always be the required rate (1/66th of the serial line rate for 64B66B systems or 1/40th of the serial line rate for 8B10B systems).

Note: When using this clocking scheme in UltraScale+/UltraScale devices, the signal GT_POWERGOOD output from the JESD204_PHY must be connected to the CE pin on the BUFG_GT used to source `core_clk` from `refclk`.

The configurations are shown in the following figures.

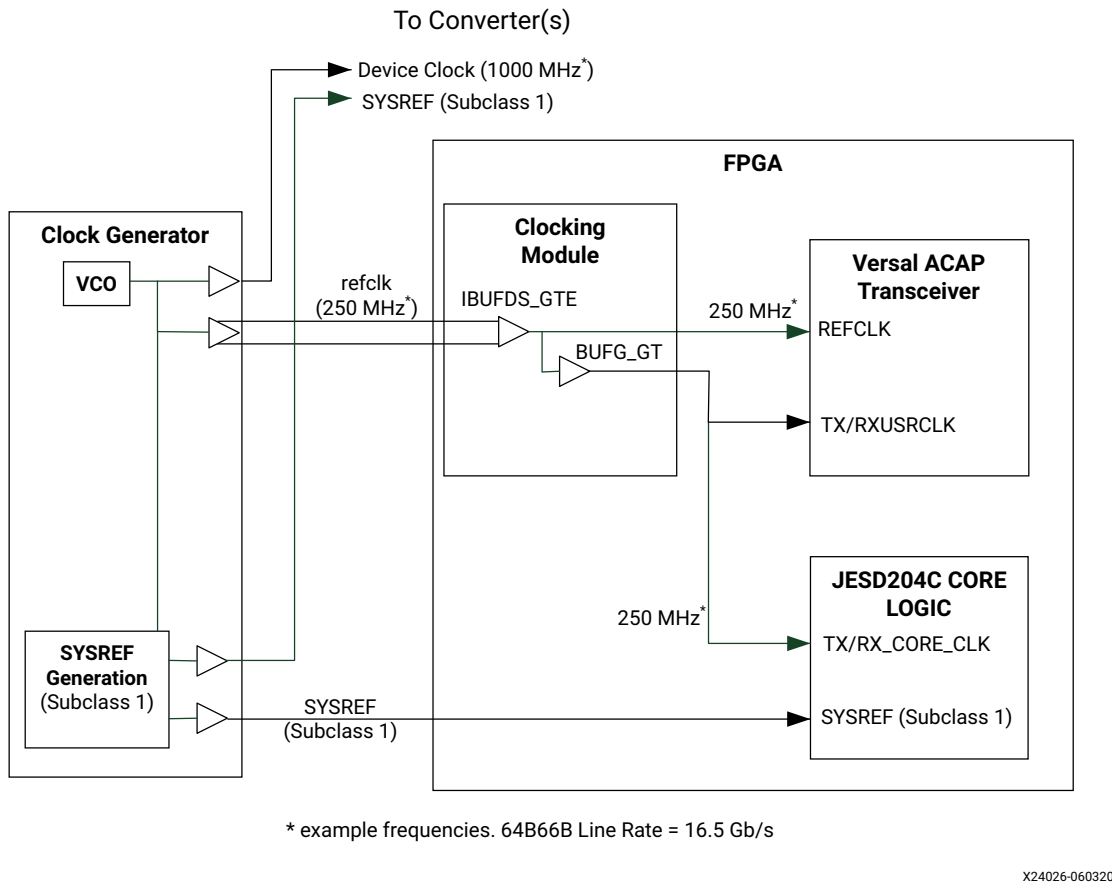

Figure 3-5: **Transceiver Reference Clock Used as Core Clock 64B66B Example for Versal ACAPs**

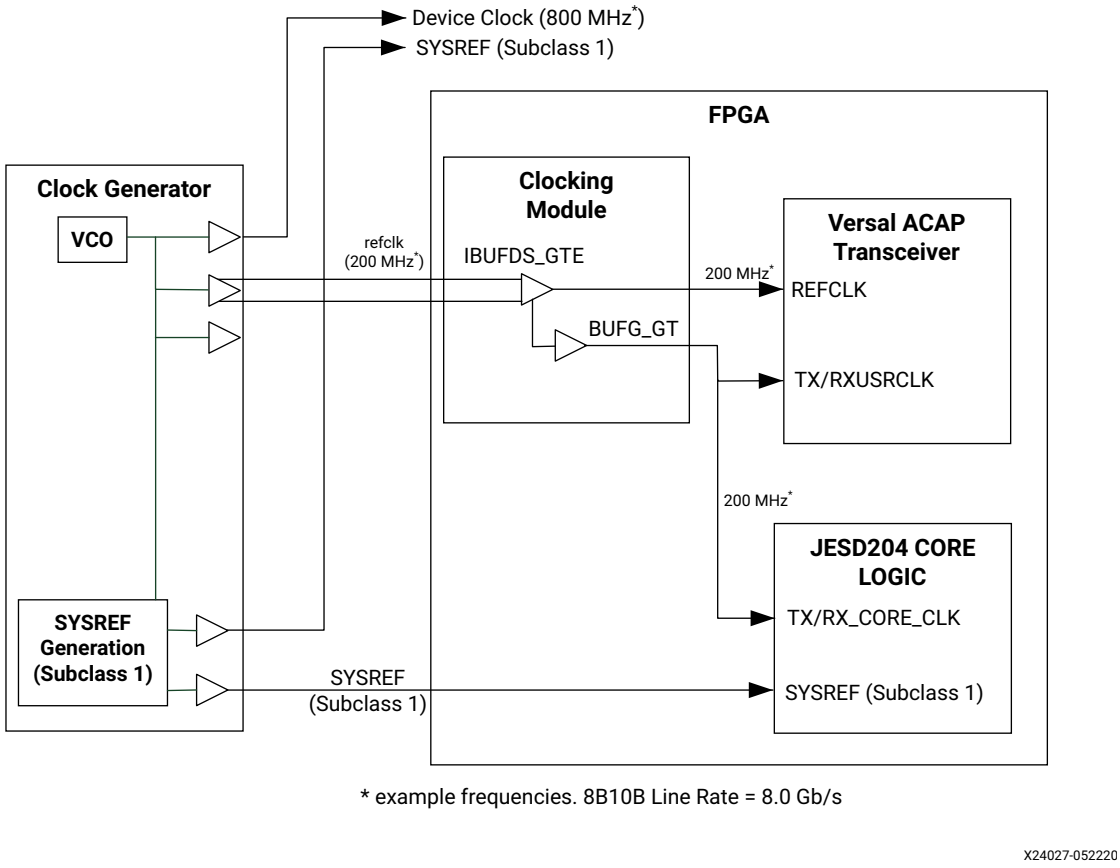

Figure 3-6: **Transceiver Reference Clock Used as Core Clock 8B10B Example for Versal ACAPs**

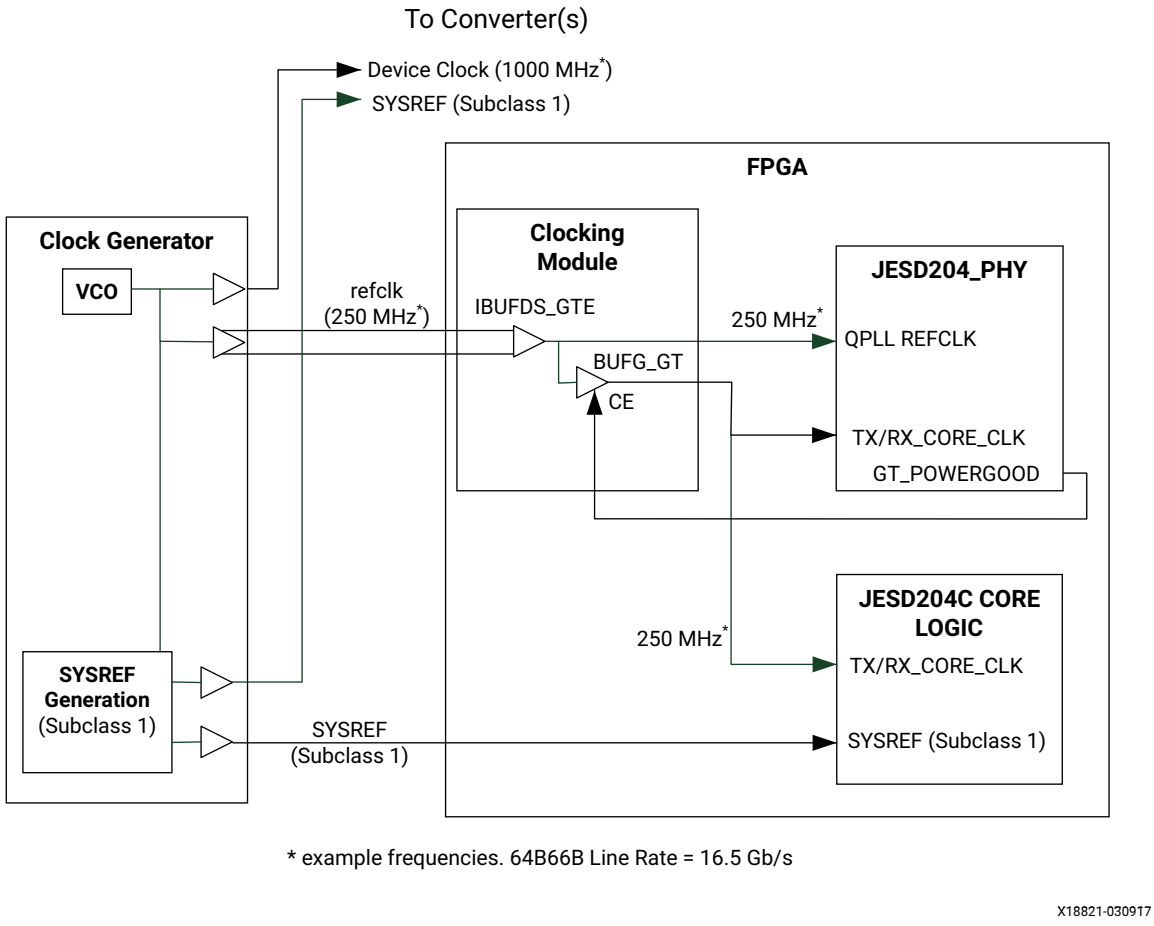

Figure 3-7: **Transceiver Reference Clock Used as Core Clock 64B66B Example for UltraScale+/ UltraScale Devices**

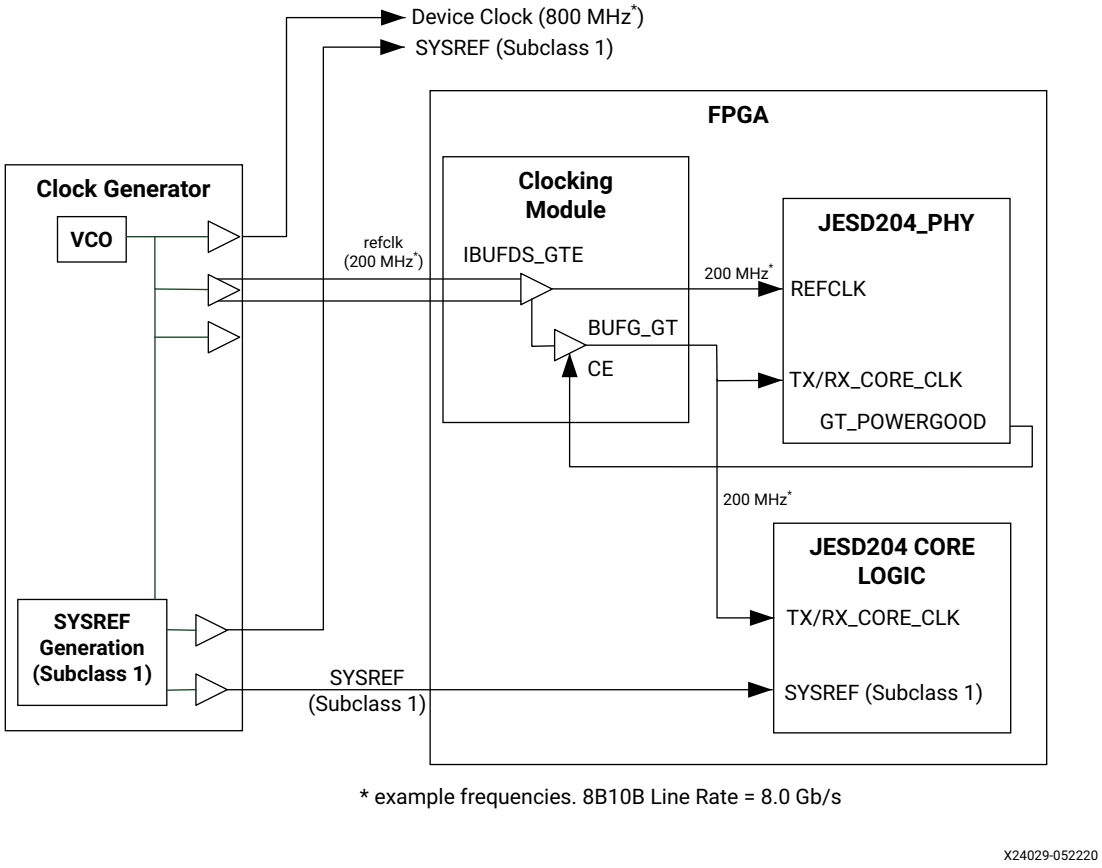

Figure 3-8: **Transceiver Reference Clock Used as Core Clock 8B10B Example for UltraScale+/ UltraScale Devices**

Transceiver Output Clock Used as Core Clock (Subclass 0)

For Subclass 0 only operation, the timing limitations imposed to support deterministic latency are removed, and a simplified clocking arrangement can be used which requires only a reference clock input. In this case, the transceiver PLL is used to generate the core clock signal. In this configuration any clock frequency that is suitable to use as the transceiver reference cock is acceptable.

The Versal ACAP and UltraScale+/UltraScale device configurations are shown in the following figures.

Note: This configuration is not suitable for subclass 1 or 2 operation because the output phase of the transceiver PLL is unknown and therefore this clock cannot be used to reliably sample SYSREF or SYNC.

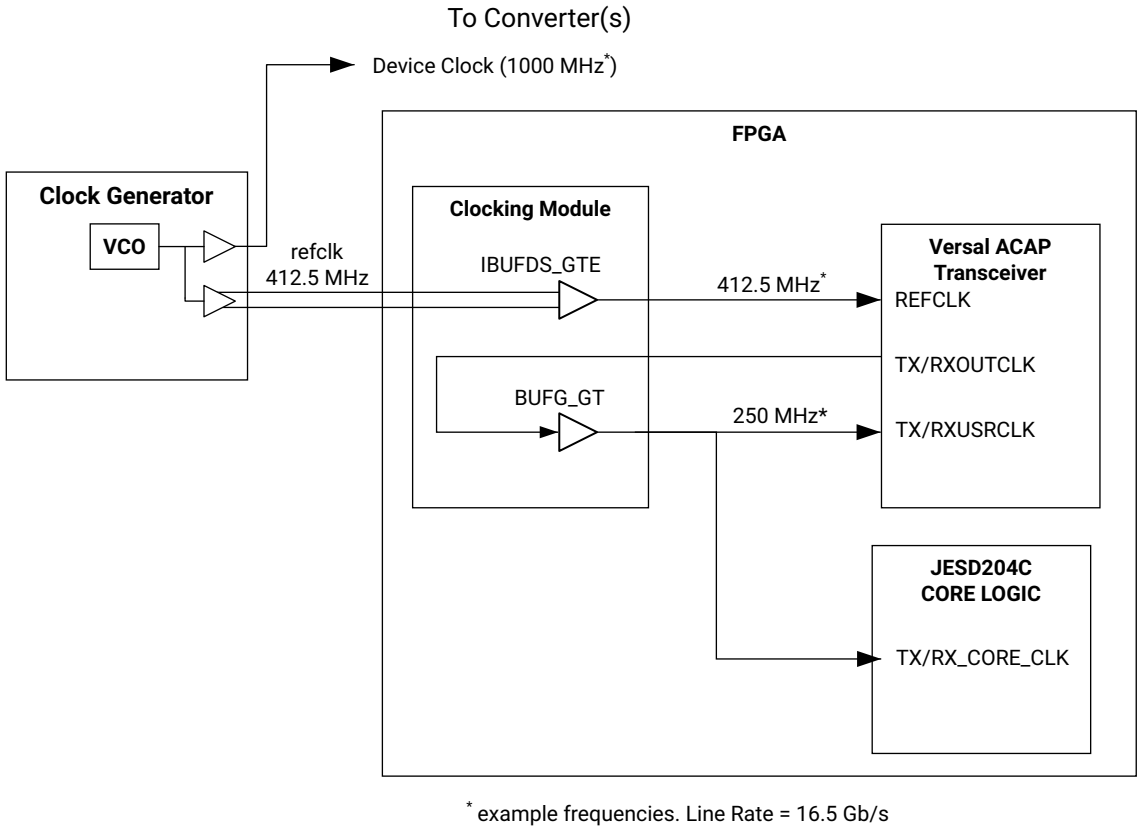

Figure 3-9: **Transceiver Output Clock Used as Core Clock (Subclass 0) for Versal ACAPs**

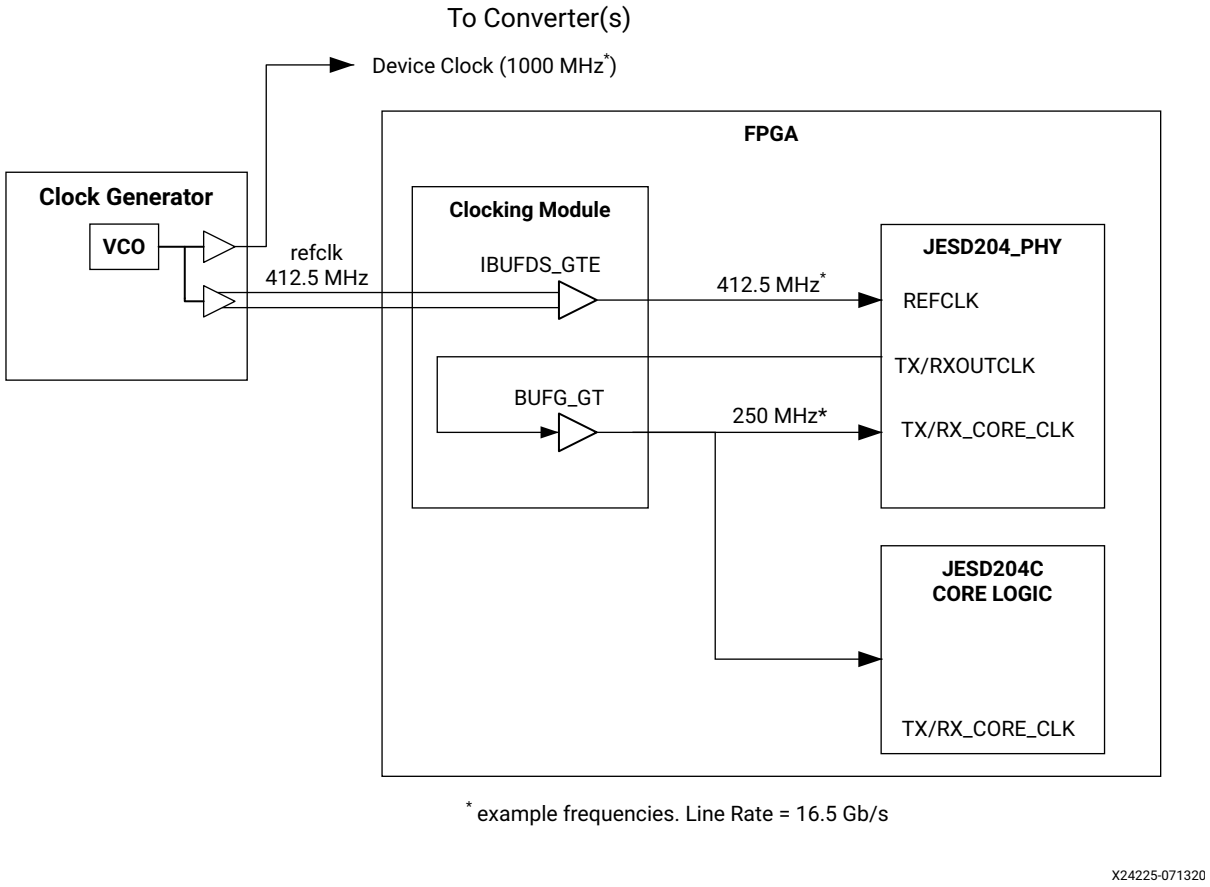

Figure 3-10: **Transceiver Output Clock Used as Core Clock (Subclass 0) for UltraScale+/ UltraScale Devices**

Clocking Considerations

- Always refer to the device data sheet for the chosen part and speed grade to confirm which PLLs are available for a required line rate - PLL selection for a particular rate might not be arbitrary.

  Versal ACAPs include:

  - The RPLL supports a maximum line rate in any device of 16.0 Gb/s.
  - The LCPLL supports a maximum line rate in any device of 32.0 Gb/s.
  - If the RPLL is required, the transceiver reference clock cannot be used as the core clock when the core is configured for 64B66B linecoding because the acceptable reference clock input frequencies to the RPLL do not cover the required Line Rate/ 66 ratio. This restriction does not apply when the core is configured for 8B10B linecoding.

UltraScale+/UltraScale devices include:

- The CPLL supports a maximum line rate in any device of 12.5 Gb/s (UltraScale+/ UltraScale devices only).
- If the CPLL is required, the transceiver reference clock cannot be used as the core clock when the core is configured for 64B66B linecoding because the acceptable reference clock input frequencies to the CPLL do not cover the required Line Rate/ 66 ratio. This restriction does not apply when the core is configured for 8B10B linecoding.
- For Line rates above 16.375 Gb/s, ensure only port MGTREFCLK0 is used to drive QPLL0, and MGTREFCLK1 to drive QPLL1.

Resets

The reset inputs and outputs on the JESD204C core are as shown in the following table.

Table 3-1: **JESD204C Resets**

Reset	Description
tx/rx_core_reset	This reset input is asynchronous and active High. This reset input will reset the JESD204C core logic but does not reset the AXI4-Lite register interface - so all programmed register values will be maintained.
s_axi_aresetn	This reset input must be synchronized with the AXI4-Lite interface clock. This reset input will reset the AXI4-Lite register interface.
tx/rx_aresetn	This reset output is synchronous to tx/rx_core_clk. This output is an AXI4-Stream interface reset signal to be used with the AXI4-Stream RX/TX Data and Cmd interfaces.
Versal ACAPs	
chN_txmstreset	This active-High reset output is connected to the Versal ACAP Transceiver using Block Automation. The output is driven by internal logic in the JESD204C core. N = Lanes - 1
chN_txpmaresetdone	This active-High reset input is connected to the Versal ACAP Transceiver using Block Automation. The input holds the JESD204C core in reset until it asserts High. N = Lanes - 1
chN_txuserrdy	This active-High reset output is connected to the Versal ACAP Transceiver using Block Automation. The output is driven by internal logic in the JESD204C core. N = Lanes - 1
chN_txmstresetdone	This active-High reset input is connected to the Versal ACAP Transceiver using Block Automation. The input holds the JESD204C core in reset until it asserts High. N = Lanes - 1
chN_rxmstreset	This active-High reset output is connected to the Versal ACAP Transceiver using Block Automation. The output is driven by internal logic in the JESD204C core. N = Lanes - 1

Table 3-1: **JESD204C Resets *(Cont'd)***

Reset	Description
chN_rxpmaresetdone	This active-High reset input is connected to the Versal ACAP Transceiver using Block Automation. The input holds the JESD204C core in reset until it asserts High. N = Lanes - 1
chN_rxuserrdy	This active-High reset output is connected to the Versal ACAP Transceiver using Block Automation. The output is driven by internal logic in the JESD204C core. N = Lanes - 1
chN_rxmstresetdone	This active-High reset input is connected to the Versal ACAP Transceiver using Block Automation. The input holds the JESD204C core in reset until it asserts High. N = Lanes - 1
UltraScale+/UltraScale Devices	
tx/rx_reset_gt	This reset output must be connected to the JESD204_PHY core. This signal is used to initiate a JESD204_PHY GT reset sequence.
tx/rx_reset_done	This input must be connected to the JESD204_PHY core. This signal is used to hold the JESD204C core in reset until completion of the JESD204_PHY GT reset sequence. **Note**: A Low input on this port will force the JESD204C core into a reset state.

Data and Command Interfaces

The transmitter and Receiver cores incorporate AXI4-Stream interfaces for data ingress and egress. These AXI4-Stream interfaces include data and flow control signals only. In addition, there are supplementary control signals that are used to signal the timing of the data on the AXI4-Stream interface.

Note: The AXI4-Stream interfaces transfer the JESD204C transport layer - not raw converter samples. Refer to the appropriate converter data sheet for information on correctly mapping samples into the transport layer.

For a 64B66B transmitter, the following figure shows the timing of the `tx_soemb` (Start Of Extended Multiblock) signal relative to the AXI4-Stream data `tx_tdata` and `tx_cmd_tdata`. The `tx_soemb` signal is a single bit and it is set High in the cycle preceding the first data block of an extended multi-block. The data interface will transfer one 64-bit block B every core clock cycle. The command interface will transfer one command word (either 19-bit or 7-bit) every multi-block. If data is not available on the command interface (`tx_cmd_tvalid` = 1) then an IDLE command will be transmitted. The command interface has been padded out to 32 bits per lane. Only bits [18:0] or [6:0] are actually used and all unused bits are set to 0.

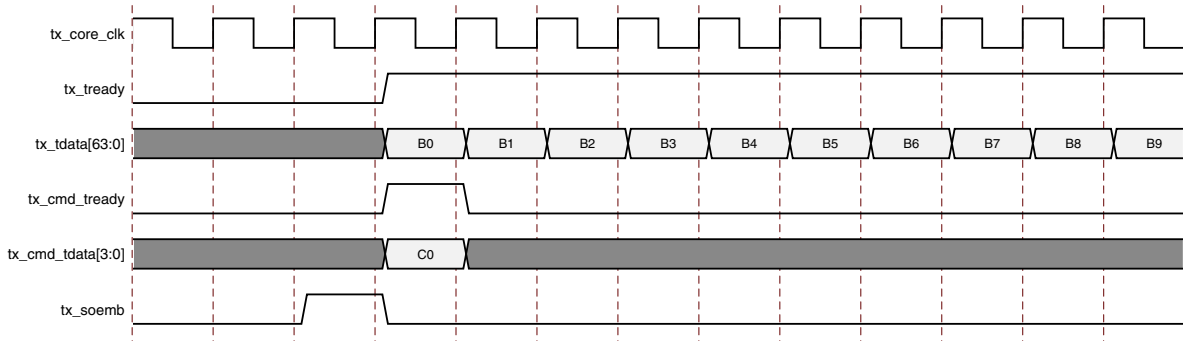

Figure 3-11: **64B66B Transmit Data Interface Timing**

For a 64B66B receiver, Figure 3-12 shows the timing of the `rx_soemb` signal relative to the AXI4-Stream data `rx_tdata`. The `rx_soemb` signal is a single bit and it is set High in the cycle preceding the first data block of an extended multi-block. The command interface will transfer one word every multi-block.

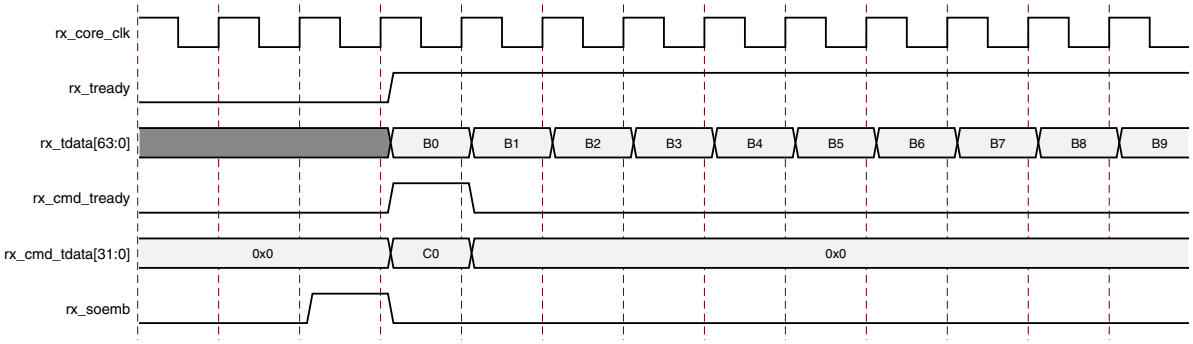

Figure 3-12: **64B66B Receive Data Interface Timing**

For an 8B10B transmitter, Figure 3-13 shows the timing of `tx_sof` (Start Of Frame) and `tx_somf` (Start Of Multiframe) signals relative to the AXI data `tx_tdata`. `tx_sof` and `tx_somf` are fixed at four bits wide because the internal data width of each lane is 32 bits and the start of frame (or multiframe) can occur in any of the 4 byte positions of the 32-bit word. For multi-lane configurations, the start of frame (or multiframe) signal indicates the byte position of the first byte of a frame in `tx_tdata[31:0]`, `tx_tdata[63:32]`, `tx_tdata[95:64]`, etc. For example, in a four lane configuration when `tx_sof` = 0001 the first byte of four new frames appears in `tx_tdata` in a single cycle, `tx_tdata[7:0]`, `tx_tdata[39:32]`, `tx_tdata[71:64]`, and `tx_tdata[103:96]`.

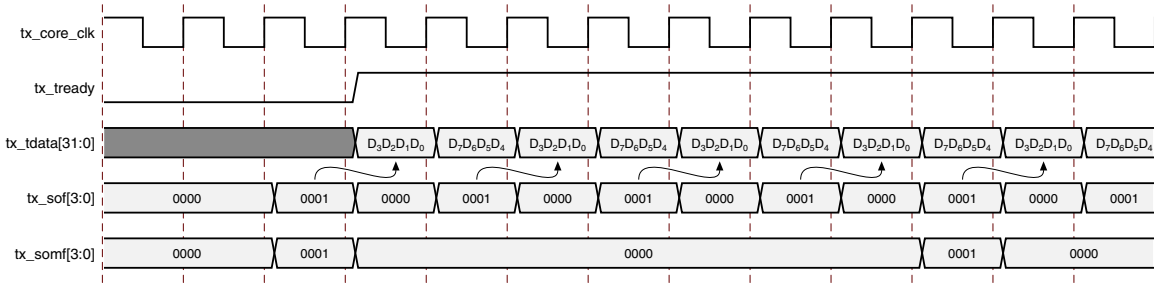

Figure 3-13: **8B10B Transmit Data Interface Timing for F = 8 and K = 4**

For an 8B10B receiver, Figure 3-14 shows the timing of `rx_sof` (Start Of Frame) and `rx_somf` (Start Of Multiframe) signals relative to the AXI data `rx_tdata`. `rx_sof` and `rx_somf` are fixed at four bits wide because the internal data width of each lane is 32 bits and the start of frame (or multiframe) can occur in any of the 4 byte positions of the 32-bit word. For multi-lane configurations, the start of frame (or multiframe) signal indicates the byte position of the first byte of a frame in `rx_tdata[31:0]`, `rx_tdata[63:32]`, `rx_tdata[95:64]`, etc. For example, in a four lane configuration when `rx_sof` = 0001 the first byte of four new frames appears in `rx_tdata` in a single cycle, `rx_tdata[7:0]`, `rx_tdata[39:32]`, `rx_tdata[71:64]`, and `rx_tdata[103:96]`.

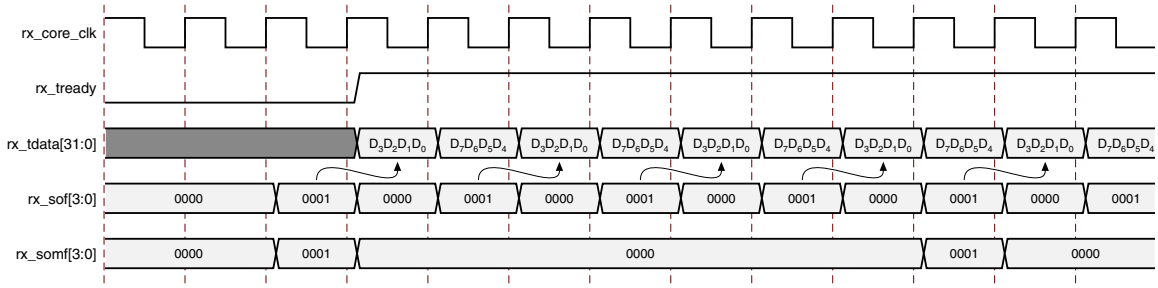

Figure 3-14: **8B10B Receive Data Interface Timing for F = 8 and K = 4**

SYSREF

SYSREF Timing

When the JESD204C is used in Subclass 1, the SYSREF signal is the master timing reference for the system. To achieve accurate deterministic latency, the SYSREF signal must be captured synchronously to the core clock. To achieve this, the SYSREF period must be a multiple of 4-byte clock periods for 8B10B linecoding and 8-byte clock periods for 64B66B linecoding. This is because the core uses a 4-byte or 8-byte internal datapath for 8B10B and 64B66B respectively.

SYSREF Type

The accurate capture of SYSREF is critical in Subclass 1 operation. The JESD204C Specification allows SYSREF to be generated in any of the following ways:

- Periodic
- One-shot
- *Gapped* Periodic

For maximum flexibility, the JESD204C core provides several options for how SYSREF is handled for Subclass 1 operation.

It is important to note that because the JESD204 core operates using a 32-bit (4-byte) or 64-bit (8-byte) datapath, if a periodic or gapped periodic SYSREF is used in the system, the following conditions must be met:

- For 64B66B linecoding, the period must be an integer multiple of the Extended Multi-block period.
- For 8B10B linecoding, the period must be an integer multiple of the multi-frame period. It must also be a multiple of 4-byte clocks. Care must be taken to ensure both of these conditions are met if the multi-frame period is not a multiple of 4-byte clocks.

SYSREF Handling

SYSREF Delay

The SYREF Delay bits in the CTRL_SYSREF register can be used to add delay to the SYSREF signal after it is captured (see CTRL_SYSREF). This allows the effective phase of the local multi frame clock (LMFC)/ local extended multiblock clock (LEMC) to be adjusted. The value programmed into the SYSREF delay register equates to the number of core clock cycles that SYSREF will be delayed by.

For 8B10B linecoding, the deterministic latency mechanism as defined in the JESD204C standard requires that the multi-frame size be larger than the maximum possible delay across the link. In practice, this can be difficult to achieve, particularly with small frame sizes. However, as long as the multiframe size is greater than the maximum variation between lanes in delay across the link, then deterministic latency can be achieved. A potential issue occurs when the maximum lane delay variation causes the overall latency to straddle the boundary between two adjacent LMFC periods. In such a case, latency variations of exactly one LMFC period can be observed between system restarts. In this case the SYSREF may be delayed to adjust the LMFC boundary position to alleviate the problem.

For 64B66B linecoding, the deterministic latency mechanism defined in the JESD204C standard requires that the maximum variation between lanes in delay across the link be less than the Extended Multi-block size. However the alignment buffer in the 64B66B RX IP core

is fixed at a size of two multiblocks. Care should be taken when designing systems where the number of Multiblocks in an Extended Multiblock (MB_IN_EMB) is greater than two because this can cause the buffers to overflow if more than two Multiblocks of data need to be stored before the next LEMC buffer release point. The SYSREF Delay register can be used to offset the phase of the LEMC clock from SYSREF such that even for large values of MB_IN_EMB the RX IP core LEMC can be adjusted such that the required amount of data to be buffered is less than two Multiblocks. The TX IP core also has this register to aid system latency optimization.

SYSREF Always

The SYSREF Always bit in the CTRL_SYSREF register provides the JESD204C core with a programmable option allowing the choice of how a periodic SYSREF is used internally (see CTRL_SYSREF).

When **SYSREF Always** is set to 0, only an initial SYSREF event seen after reset (or on link resynchronization) is used to align the internal LMFC counter. All subsequent SYSREF events will be ignored.

When **SYSREF Always** is set to 1, all SYSREF events are used to (re)align the LMFC counter. This setting requires that the SYSREF period be a correct multiple of the Multiframe / Extended Multiblock periods.

SYSREF Required

The SYSREF required bit in the CTRL_SYSREF register provides the JESD204C core with a programmable option allowing the choice of whether a SYSREF event is required or not for the link to restart after a resync request (see CTRL_SYSREF).

When **SYSREF Required** is set to 0, a resync request will automatically restart the link.

When **SYSREF Required** is set to 1, a resync request will stall until a new SYSREF event has been detected.

Capturing SYSREF

The synchronous capture of SYSREF is critical to the deterministic latency mechanism of JESD204C. By default, no constraints are applied to the SYSREF input. However, the required timing of the SYSREF input can be checked using the `report_datasheet` command in the Vivado Design Suite.

An example timing diagram is shown in the following figure. This example uses the following settings:

`core_clk` period = 6.4 ns (6.25 Gb/s line rate with 8B10B linecoding)

In this example, the `report_datasheet` command gives a setup of 4.6 ns and hold of -1.5 ns for the SYSREF pin.

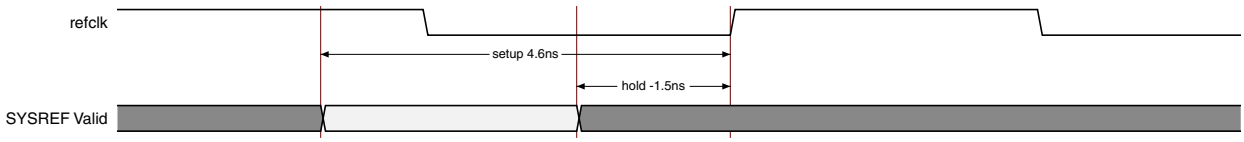

Figure 3-15: **SYSREF Timing example**

The easiest method to ensure a design will be able to reliably capture SYSREF is to use a programmable clock generator chip that allows fine delay adjustment of its outputs, to generate all the JESD204C clocks and SYSREF signals in the system. This will allow for the delay between core clock and SYSREF to be adjusted to meet the setup and hold requirements achieved by your design.

It is also possible to use an MMCM to adjust the phase of core clock internally to align with the setup and hold requirements.

SYSREF on Initial Link Bring-Up with 8B10B Linecoding

After a reset, a JESD204C core configured for subclass 1 operation requires at least one SYSREF event to align the internal LMFC counter, and bring up the link:

- A receive core requires an initial SYSREF event to align the LMFC, and then asserts SYNC on the next LMFC boundary when code group sync has been achieved. The core does not assert SYNC until an initial SYSREF event is detected.
- A transmit core requires a SYSREF event to align the LMFC. The core begins ILA transmission on an LMFC boundary after SYNC is asserted. The core does not begin ILA transmission until an initial SYSREF event is detected.

The system must ensure that SYSREF to the JESD204 core is generated after the core has completed reset. This is of particular importance if the system is operating a One-shot SYSREF.

SYSREF on Link Resynchronization with 8B10B Linecoding

After initial bring-up, if a link re-synchronization is requested (by the deassertion of SYNC by the receiving device), the desired core behavior relative to SYSREF can be controlled using the **SYSREF Required** control bit in the CTRL_SYSREF Handling register.

When **SYSREF Required** is set to 0, no SYSREF event is required for the link to re-synchronize (the assumption is that LMFC counters continue to free-run and remain valid).

- A receive core asserts SYNC on the next LMFC boundary after code group sync.
- A transmit core transmits the ILA sequence on the next LMFC boundary after SYNC is asserted.

When **SYSREF Required** is set to 1, a SYSREF event is required for the link to re-establish SYNC following a re-sync request. In this case the behavior is the same as the initial link bring up detailed earlier.

This setting is particularly important in systems where a One-Shot SYSREF is used, or where SYSREF is periodic, but SYSREF Always is set to 0.

SYSREF on Initial Link Bring-Up with 64B66B Linecoding

After a reset, a JESD204C core configured for subclass 1 operation requires at least one SYSREF event to align the internal LMBC counter, and bring up the link:

- A receiver core requires an initial SYSREF event to align the LMBC, The core does not start the LEMC counter until an initial SYSREF event is detected.
- A transmitter core requires a SYSREF event to align the LMBC. The core does not begin transmission of multi-blocks until an initial SYSREF event is detected. Therefore a link cannot achieve multi-block lock until after a SYSREF event has been seen by both a transmitter and a receiver.

The system must ensure that SYSREF to the JESD204C core is generated after the core has completed reset. This is of particular importance if the system is operating a One-shot SYSREF.

SYSREF on Link Resynchronization with 64B66B Linecoding

After initial bring-up, if a link re-synchronization is requested by the receiving device, the desired core behavior relative to SYSREF can be controlled using the **SYSREF Required** control bit in the CTRL_SYSREF Handling register.

When **SYSREF Required** is set to 0, no SYSREF event is required for the link to re-synchronize (the assumption is that LMBC counters continue to free-run and remain valid).

- A *receive* core will re-acquire multi-block lock and output received data on the next LMBC boundary.
- A *transmit* core will continue to transmit multi-block data.

When **SYSREF Required** is set to 1, a SYSREF event is required for the link to re-establish SYNC following a re-sync request. In this case, the behavior is the same as the initial link bring-up detailed earlier.

This setting is particularly important in systems where a One-shot SYSREF is used, or where SYSREF is periodic but **SYSREF Always** is set to 0.

Subclass 2 Operation (8B10B Linecoding Only)

The operation of the JESD204C circuit in Subclass 2 mode is similar to a device in Subclass 1 mode. In this case, deterministic latency across the link is achieved using the SYNC~ interface. When the receiver has aligned to the incoming idle characters from the transmitter, it asserts the SYNC signal. The transmitter detects this and waits until the next LMFC crossing before transmitting data or the ILA sequence (depending on the setting of the enable link synchronization control bit). The receiver buffers the data at its input until the next LMFC crossing at its side of the link, before sending the received data to the client.

Subclass 2 operation can be difficult to achieve at medium to high line rates due to the critical need to accurately capture the SYNC signal without violating the setup or hold requirements of the capturing flip flop. It is recommended that careful design validation is performed early to ensure this can be achieved.

Number of Lanes per Link

The maximum number of lanes per link is eight. For interfaces which require more than eight lanes, simply create multiple cores with a maximum of eight lanes each.

For 8B10B *transmit* interfaces, the lane ID and number of lanes per link ILAS parameters for each lane can be independently programmed using the Lane ID registers.

For 8B10B *receive* interfaces, the lane ID and number of lanes per link parameters for each lane can be read from the LID and L fields of the RX_ILA_CFG3 register for each lane.

This programmable Lane ID and number of lanes per link feature for 8B10B cores can also be utilized to share a single JESD204C core with multiple synchronous converters. For example, eight one lane converters may be connected to a single JESD204C core. If this is an 8B10B transmitter core, the Lane IDs can all be programmed to be lane 0 with 1 lane per link (value = 0). For 64B66B cores, there are no Lane IDs so this is not applicable.

Design Flow Steps

This chapter describes customizing and generating the JESD204C core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the Vivado IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 1]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4]

Customizing and Generating the Core

This section includes information on using Xilinx tools to customize and generate the core in the Vivado Design Suite.

For Versal™ ACAPs, the core should be used in IP integrator, because this is the only supported flow to allow IP integrator Block Automation to generate and correctly configure Versal ACAP Transceiver cores as required. This is also required to enable transceiver sharing with other JESD cores (TX and RX) and other IP cores.

For UltraScale+™/UltraScale™ devices, if you are customizing and generating the core in the IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 1] for detailed information.

IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value you can run the `validate_bd_design` command in the Tcl Console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the Vivado IP catalog.
2. Double-click the selected IP or select the **Customize IP** command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3].

Note: Figures in this chapter are illustrations of the JESD204C GUI in the Vivado Integrated Design Environment (IDE). This layout might vary from the current version.

Configuration Tab

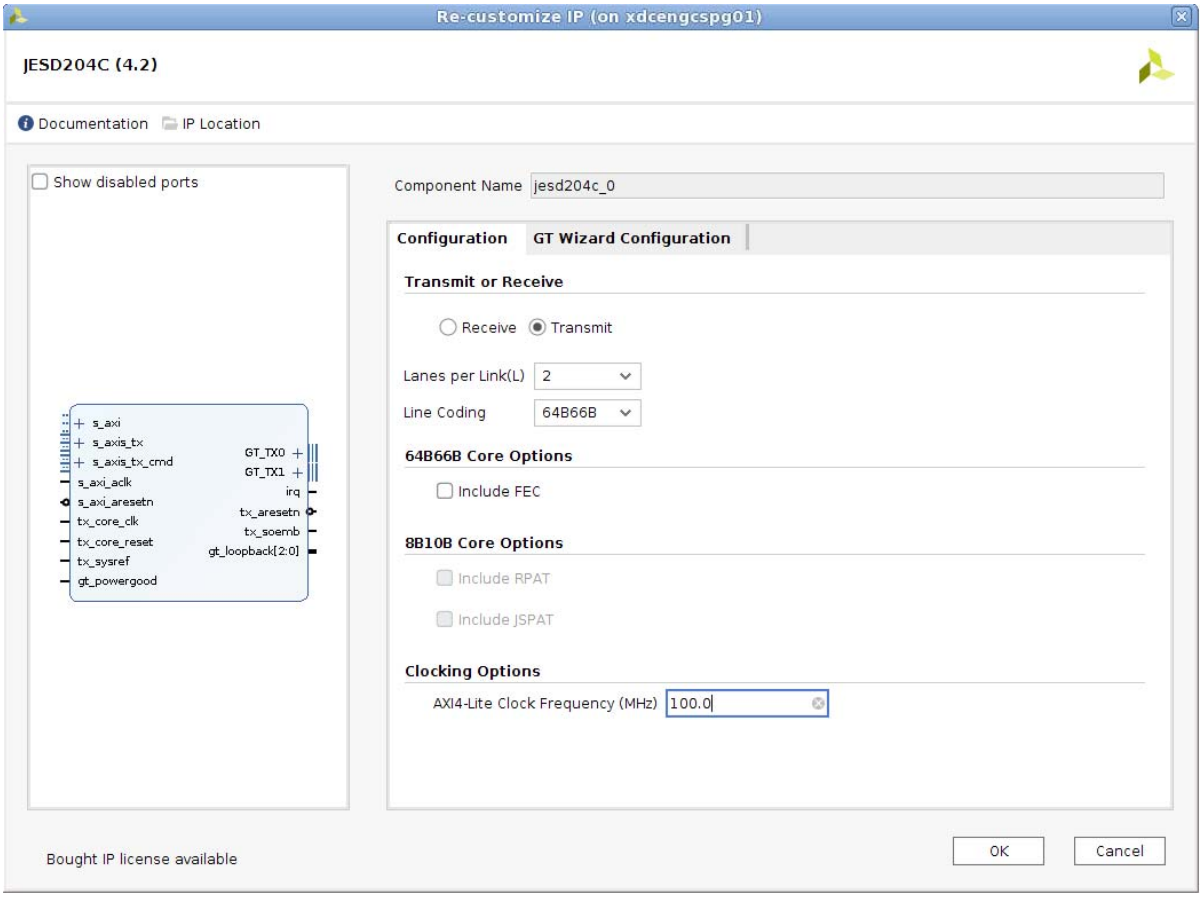

Figure 4-1: **Configuration Tab for Versal ACAPs**

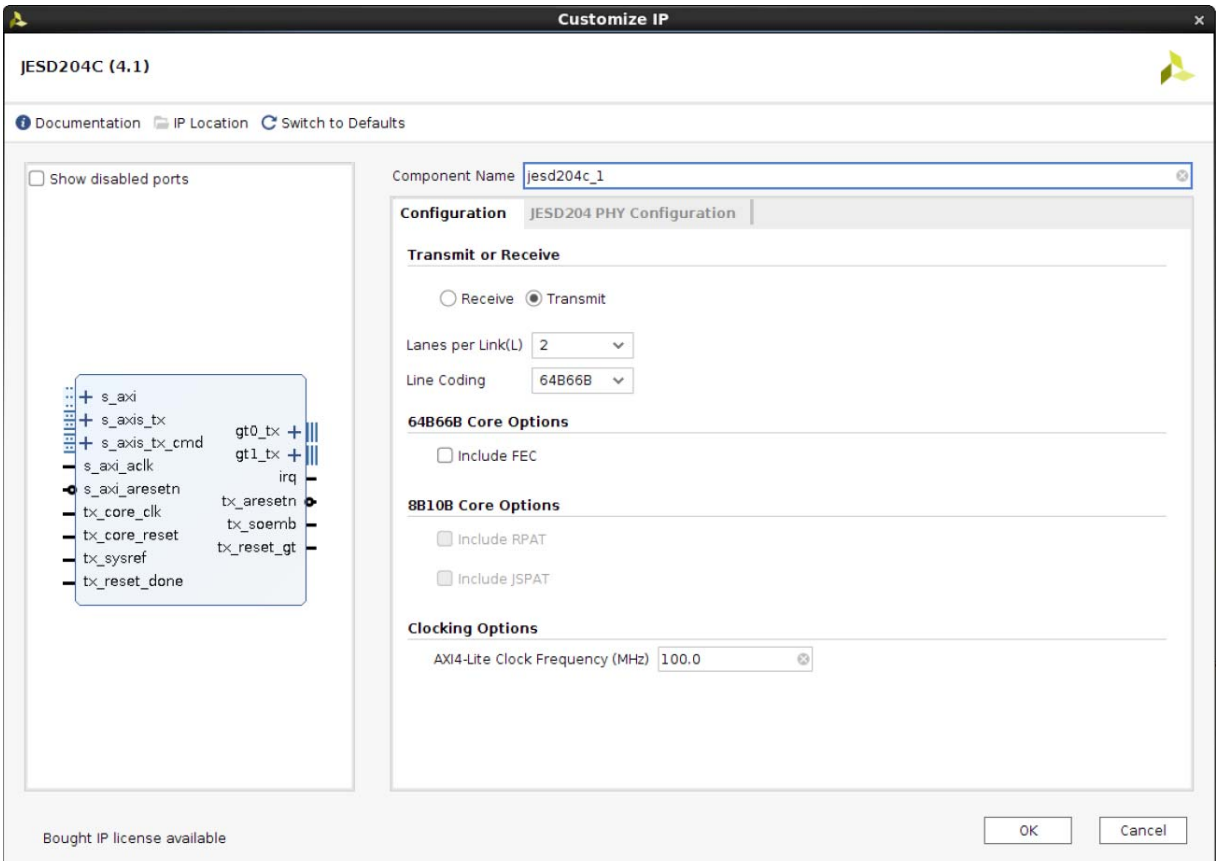

Figure 4-2: **Configuration Tab for UltraScale+/UltraScale Devices**

- **Component Name** – The component name is used as the base name of the output files generated for the core. Names must begin with a letter and must be composed from these characters: a through z, 0 through 9 and "_" (underscore).
- **Transmit or Receive** – The core can be configured as a transmitter, for connection to DAC devices, or receiver, for connection to ADC devices.
- **Lanes per Link (L)** – The core supports 1 to 8 lanes. For interfaces requiring more than 8 lanes, multiple core must be used.
- **Line Coding** – The core supports 64B66B and 8B10B linecoding modes.
- **Include FEC** – (64B66B only.) Check this option to include the FEC Encoder (TX) or Decoder (RX) in the core. Including the FEC core will increase the resources required by the core.
- **AXI4-Lite Clock Frequency** – The frequency of the clock connected to the AXI4-Lite Management Interface.
- **Include RPAT** – (8B10B Transmitter only.) Check this option to include the logic required to generate a Modified Random test pattern. This option will increase the resources required by the core.

- **Include JSPAT** – (8B10B Transmitter only.) Check this option to include the logic required to generate a Scrambled Jitter test pattern. This option will increase the resources required by the core.

GT Wizard Configuration Tab for Versal ACAPs

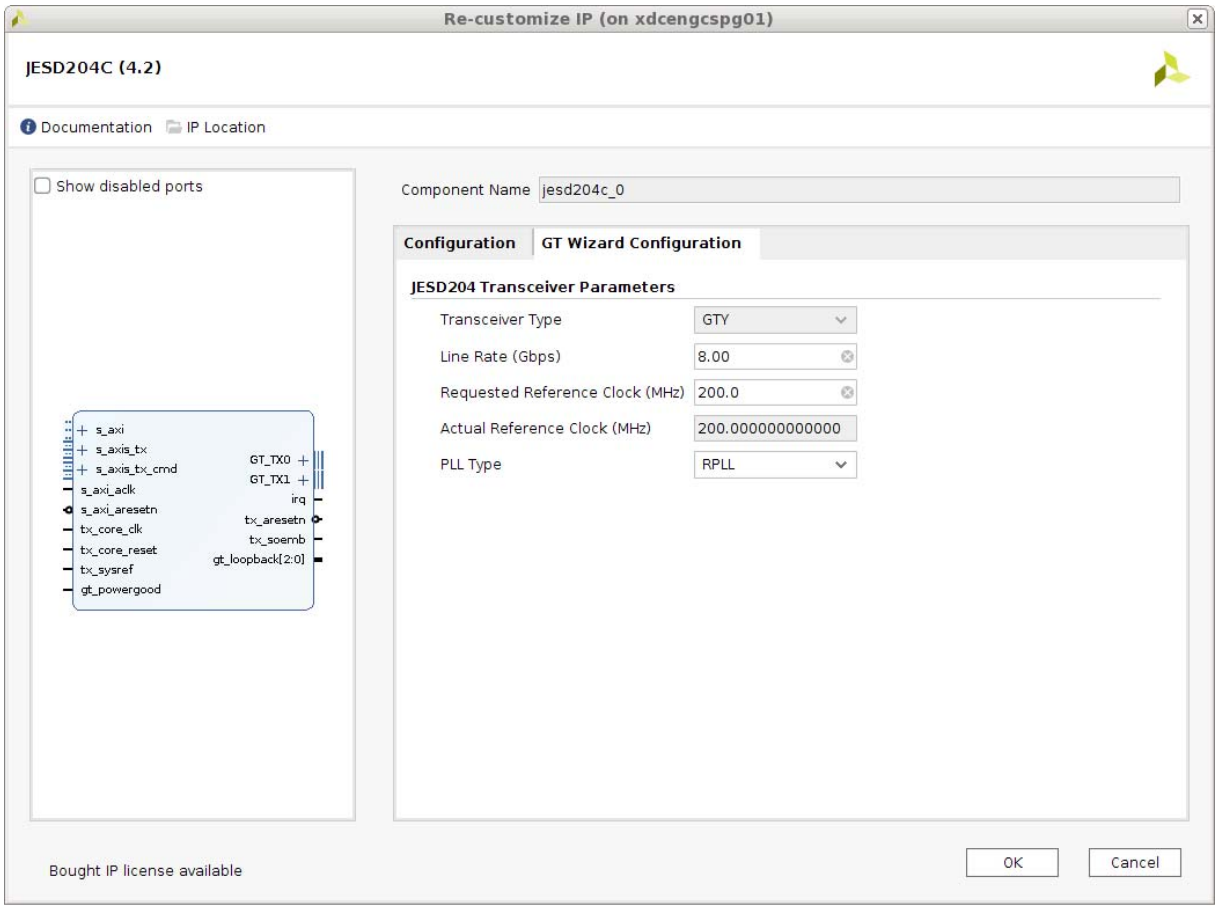

Figure 4-3: **GT Wizard Configuration Tab**

- **Transceiver Parameters** – For any selected Line Rate and PLL Type, valid Reference Clock frequencies can be calculated by first entering the requested reference clock frequency.

JESD204 PHY Configuration Tab for UltraScale+/UltraScale Devices

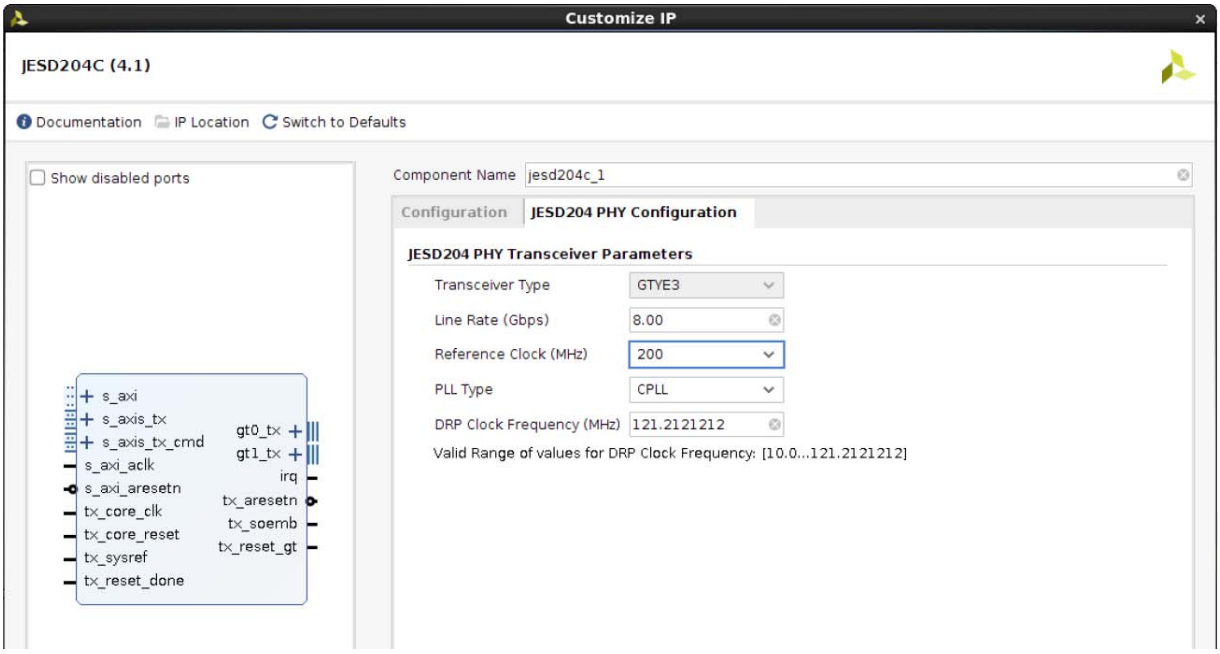

Figure 4-4: **JESD204 PHY Tab**

- **Transceiver Parameters** – For any selected Line Rate and PLL Type, valid Reference Clock frequencies can be selected from a drop-down list. A free-running DRP clock must be supplied, and the frequency (within the displayed valid range) must be entered in the DRP Clock Frequency box.

User Parameters

Table 4-1 shows the relationship between the GUI fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
Transmit or Receive	C_NODE_IS_TRANSMIT	1 (= Transmit)
Lanes per Link	C_LANES	2
AXI4-Lite Clock Frequency	AXICLK_FREQ	100.00
Transceiver Parameters		
Transceiver Type[3]	Transceiver	GTY
Line Rate[2]	GT_Line_Rate	8.0
Ref Clock Frequency[2][4]	GT_REFCLK_FREQ	200.0
DRP Clock Frequency[4]	DRPCLK_FREQ	121.2121212

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter/Value[1]	User Parameter/Value[1]	Default Value
Requested Reference Clock[3]	GT_REFCLK_FREQ_REQUEST	200.0
Actual Reference Clock[3]	GT_REFCLK_FREQ_ACTUAL	200.0
PLL Type	C_PLL_SELECTION	0 (=CPLL)[4] 3 (=RPLL)[3]
Line Coding	C_ENCODING	1 (1 = 64B66B. 0 = 8B10B)
Include FEC	C_USE_FEC	0 (= not included)
Include RPAT	C_USE_RPAT	0 (= not included)
Include JSPAT	C_USE_JSPAT	0 (= not included)

Notes:
1. Parameter values are listed in the table where the Vivado IDE parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.
2. Varies depending on device.
3. For Versal ACAPs only.
4. For UltraScale+/UltraScale devices only.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Transceiver Sharing for UltraScale+/UltraScale Devices

The JESD204_PHY core (see *JESD204 PHY LogiCORE IP Product Guide* (PG198) [Ref 11]) provides a simple way to share transceivers between JESD204/JESD204C cores. Any number of JESD204_PHY cores can be connected to any number of JESD204/JESD204C cores to cater for any combination of ADCs and DACs using different line rates, lane counts, linecoding, and versions of the JESD204 standard.

An example of a two lane 64B66B TX and two lane 8B10B RX sharing a JESD204 PHY is shown in Figure 4-5. The transmitter and the receiver are configured for different line rates. Separate `refclk` inputs are provided for each PLL and separate core clocks are provided for TX and RX to support subclass 1 (see Figure 3-7).

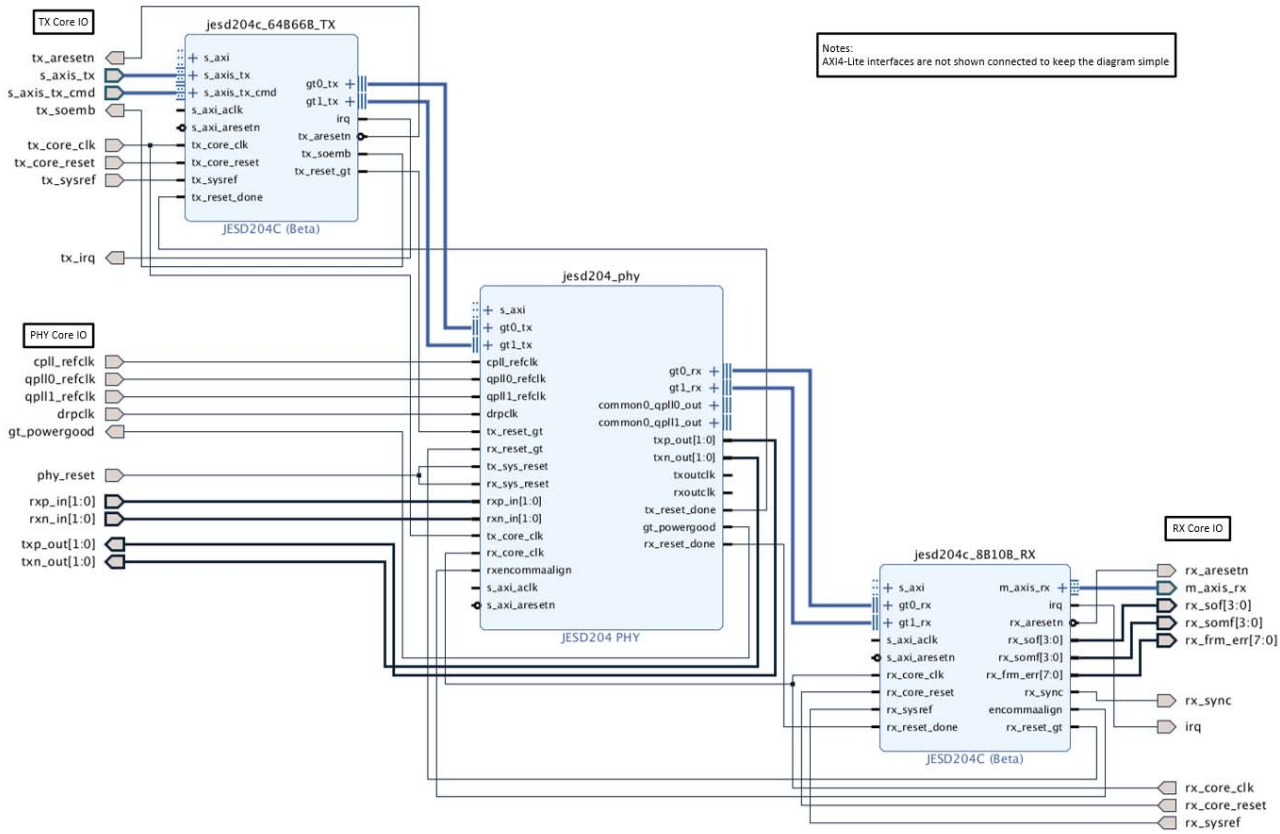

Figure 4-5: **Transceiver Sharing for UltraScale+/UltraScale Devices**

Connecting to Transceivers Using Block Automation for Versal ACAPs

Block Automation is the methodology in IP integrator which is used to connect the JESD204C core to a Versal ACAP Transceiver. This is the only supported flow to correctly generate and configure the transceiver. Block Automation is also required to enable transceiver sharing with other JESD cores (TX and RX) and other IPs. The following figure shows the Run Block Automation link in the IP integrator window.

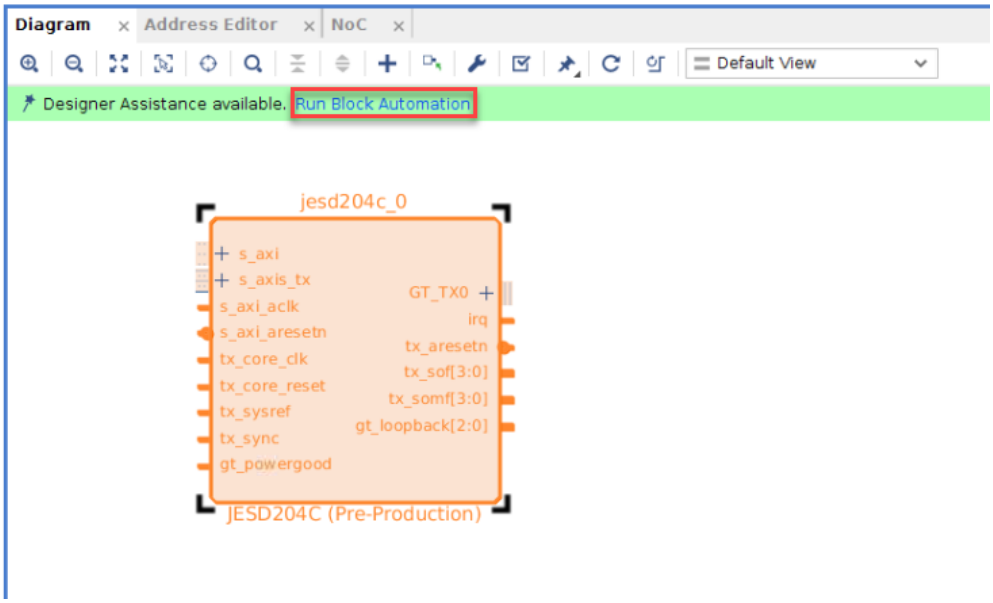

Figure 4-6: **Block Automation Link for Versal ACAPs**

The following figure shows the Block Automation dialog box. In this case two JESD204C IP cores have been placed onto the IP integrator canvas. The dialog box gives you the choice of selecting one or both cores for the application of Block Automation. There is also the option of **Auto**, **Start_With_New_Quad**, or **Customized Connections**.

When a core has been selected to have Block Automation applied, then the Block Automation control option of **Auto**, **Start_With_New_Quad**, or **Customized Connections** for that core is available.

If **Auto** is selected, the Block Automation will sequentially place the selected cores in the first available GT Quad locations that are suitable. If no existing Versal ACAP Transceiver GT Quad is available or suitable, then a new GT Quad will be created. Block Automation will try to pack TX and RX cores (and other IP) into as few GT resources as possible. TX and RX cores will share PLL resources when they have common settings (line rate, ref clock, and PLL) or use different resources when they do not. This is shown in Example 2.

If **Start_With_New_Quad** is selected, then Block Automation will sequentially place the selected cores in new Versal ACAP Transceiver GT Quad IPs.

If **Customized Connections** is selected, then the Block Automation will allow the user to choose the specific GT locations to use for each lane on the cores. This option will not add new GT Quads to the IPI canvas so requires the GT Quads to be added to the IPI canvas prior to running Block Automation. This option gives the maximum flexibility around the wiring of JESD lanes to GT quad locations.

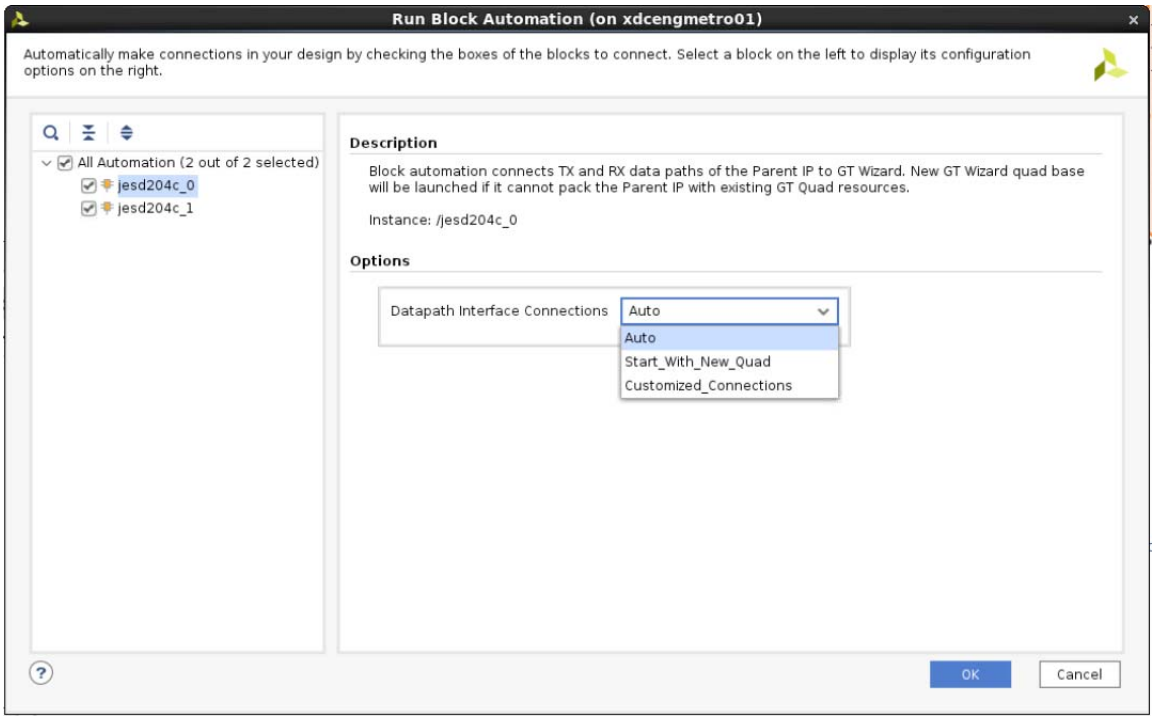

Figure 4-7: **Block Automation Dialog Box**

Block Automation Output

The following two examples show what results from running Block Automation.

- Versal ACAP Transceiver GT Quad will be created with the correct settings required for the selected JESD204C IP cores
- JESD204C IP cores will be connected to the Versal ACAP Transceiver GT Quad

The remaining unconnected ports such as `ref_clock`, `core_clk`, resets and sysref etc. then need to be connected up manually. For an example of how to connect up the remaining ports, generate the IP example design (see Chapter 5, Example Design) in Vivado and use this as a reference.

Example 1

The following figure shows the result of running Block Automation on a single JESD204C core with default settings.

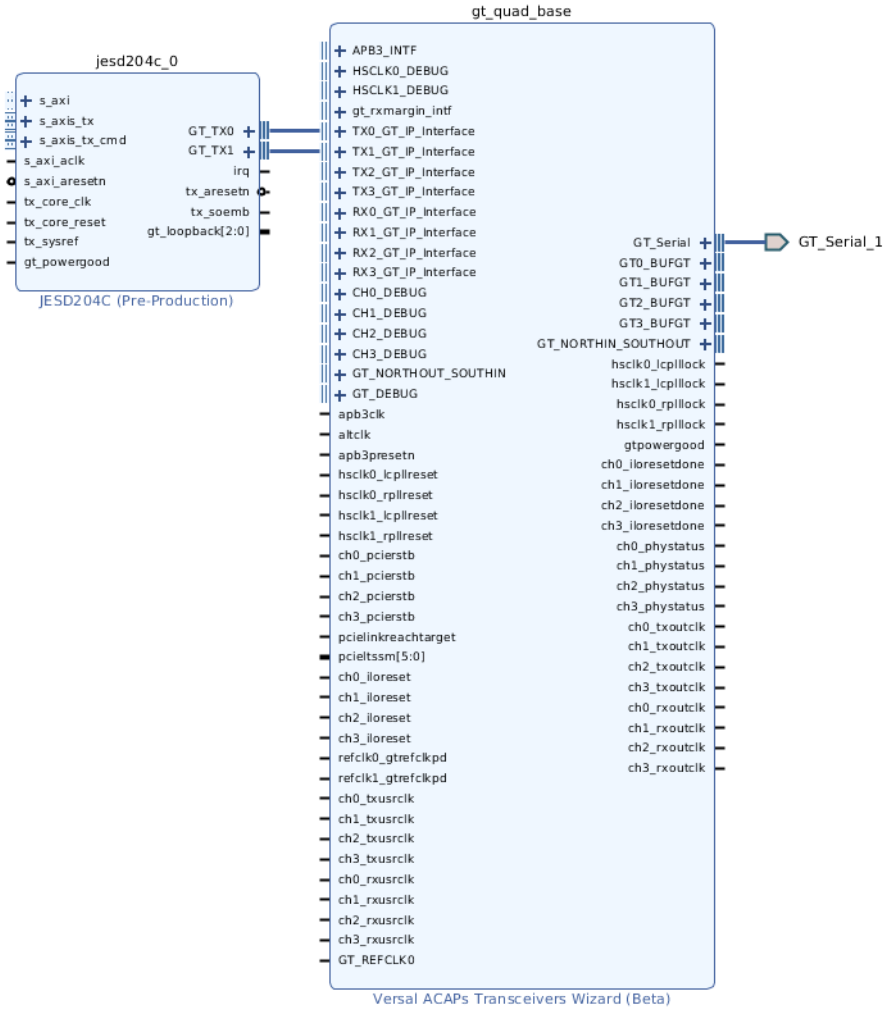

Figure 4-8: **Block Automation on a Single JESD204C Core with Default Settings**

Example 2

The following figure shows the result of running Block Automation on two JESD204C cores, one TX and one RX. The JESD204C cores are running at different line rates and therefore have been configured so that one uses the RPLL and the other the LCPLL.

Note: To see the case where both TX and RX cores sharing a Transceiver are running at the same line rate, generate the IP example design (see Chapter 5, Example Design) in Vivado.

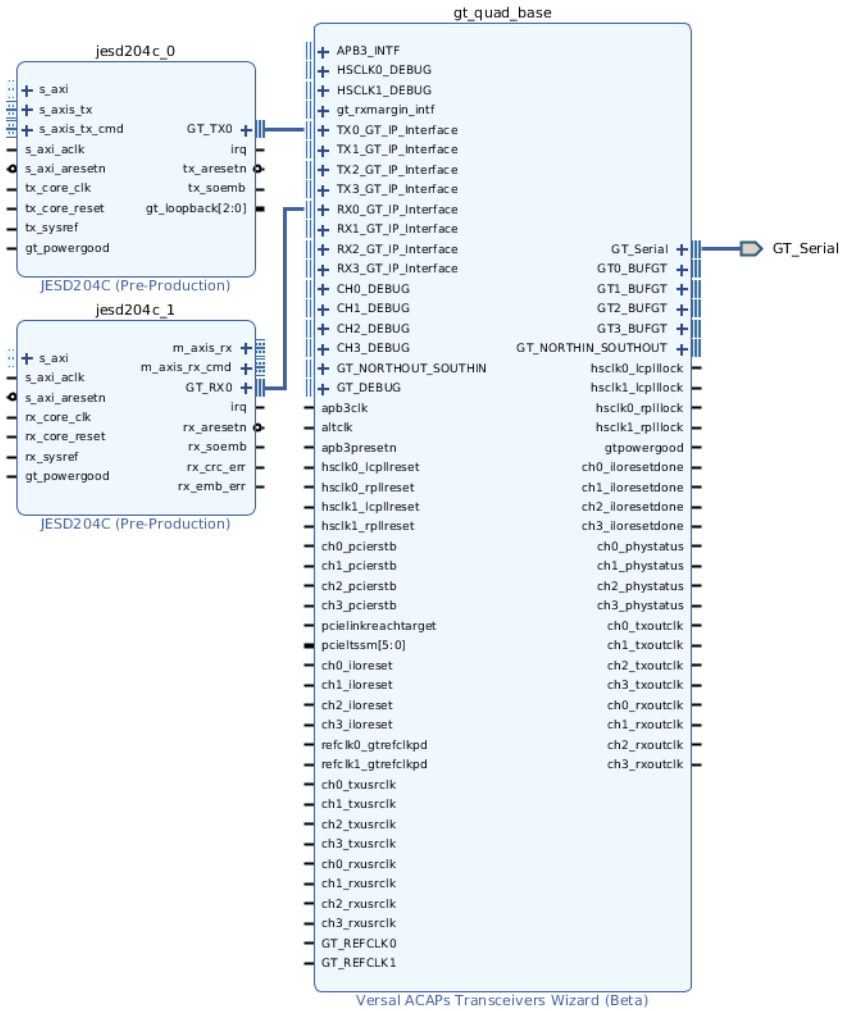

Figure 4-9: **Block Automation on a TX JESD204C Core and RX JESD204C Core Running at Different Line Rates**

Configuring the JESD204 PHY in IP Integrator for UltraScale+/UltraScale Devices

The example design that can be generated for the JESD204C core in Vivado (see Chapter 5) delivers a JESD204 PHY core with the settings used in the JESD204C GUI. When configuring a JESD204 PHY core for use with a JESD204C core, the following values must be set:

- The transceiver type must be set to GTHE3, GTHE4, GTYE3, or GTYE4.
- The JESD204 Version must be set to JESD204C.

As highlighted in Figure 4-10, **Line Coding** must be selected (64B66B or 8B10B).

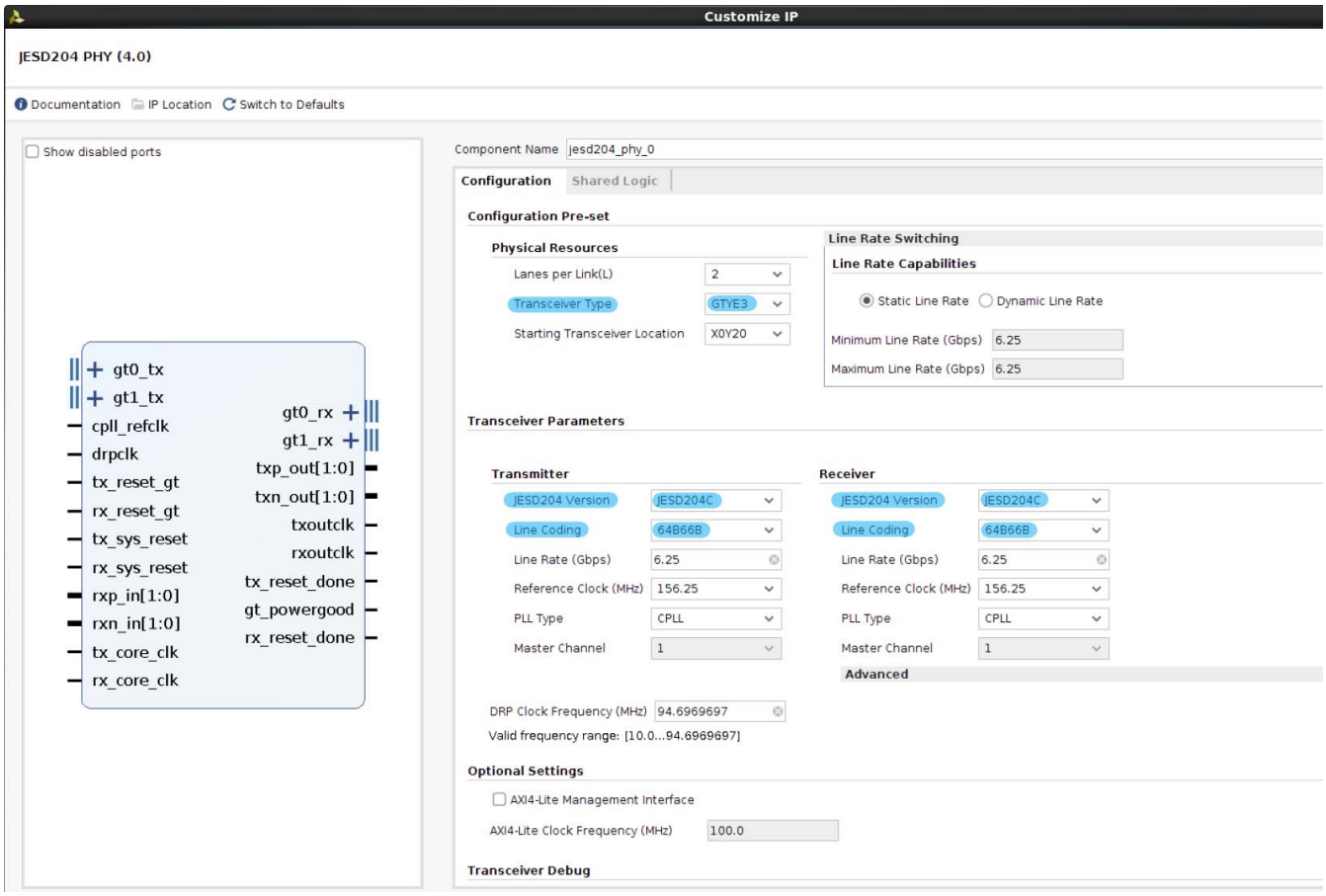

Figure 4-10: **JESD204 PHY GUI**

It is possible to share a JESD204 PHY between instances of JESD204B and JESD204C cores because the JESD204 Version parameter is set independently for each direction (TX and RX). The value must be set to JESD204C to connect to a JESD204C core, and JESD204B for a JESD204 core. Both JESD204 and JESD204C cores support 8B10B linecoding, however the interfaces to the JESD204_PHY are not identical. So the correct version of the standard must be chosen based on which core you are intending to interface with.

Constraining the Core

This section describes how to constrain a design containing the JESD204C core. This is accomplished by using the XDC delivered with the core at generation time. An additional XDC file is generated with the IP example design; only the core XDC file should be used in user designs.

Required Constraints

This section defines the constraint requirements for the core. Constraints are provided in several XDC files which are delivered with the core and the example design to give a starting point for constraints for the user design.

There are four XDC constraint files associated with this core:

- `<corename>_example_design.xdc`
- `<corename>_ooc.xdc`
- `<corename>.xdc`
- `<corename>_clocks.xdc`

The first is used only by the example design; the second file is used for Out Of Context support where this core can be synthesized without any wrappers; the third file is the main XDC file for this core. The last file defines constraints which depend on clock period definition, either those defined by other XDC files or those generated automatically by the Xilinx tools, and this XDC file is marked for automatic late processing within the Vivado design tools to ensure that definitions exist.

Device, Package, and Speed Grade Selections

See the appropriate device data sheet listed in References to determine the maximum line rate supported. Not all devices, packages and speed grades can operate at the maximum line rate supported by the IP.

Clock Frequencies

The reference clock and core clock frequency constraints vary depending on the selected line rate and reference clock when generating the core. See the generated XDC for details.

Clock Domains

There are also several paths where clock domains are crossed. These include the management interface. See the generated XDC file for details.

Clock Management

Reference clock and core clock resources require location constraints appropriate to your top-level design.

Clock Placement

Reference clock input should be given location constraints appropriate to your top-level design and to the placement of the transceivers.

Core clock input (if required) should be given location constraints appropriate to your top-level design.

Banking

All ports should be given location constraints appropriate to your top-level design within banking limits.

Transceiver Placement

Transceivers should be given location constraints appropriate to your design.

I/O Standard and Placement

All ports should be given I/O standard and location constraints appropriate to your top-level design.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4].

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 2].

Example Design

The JESD204C IP can be generated as a TX or RX configuration with either 64B66B or 8B10B linecoding. All selections include a lightweight test harness to enable familiarization with the design and signal interface.

To create the example design for Versal™ ACAPs:

1. In Vivado®, create a new block design in IP integrator.
2. Add the JESD204C IP core to the canvas.
3. Select the **JESD204C** IP core and configure exactly as required.
4. Right-click the JESD204C IP and select **Open IP Example Design**, from the drop-down menu as shown in the following figure. This opens a new Vivado project containing the complete RX or TX design example.

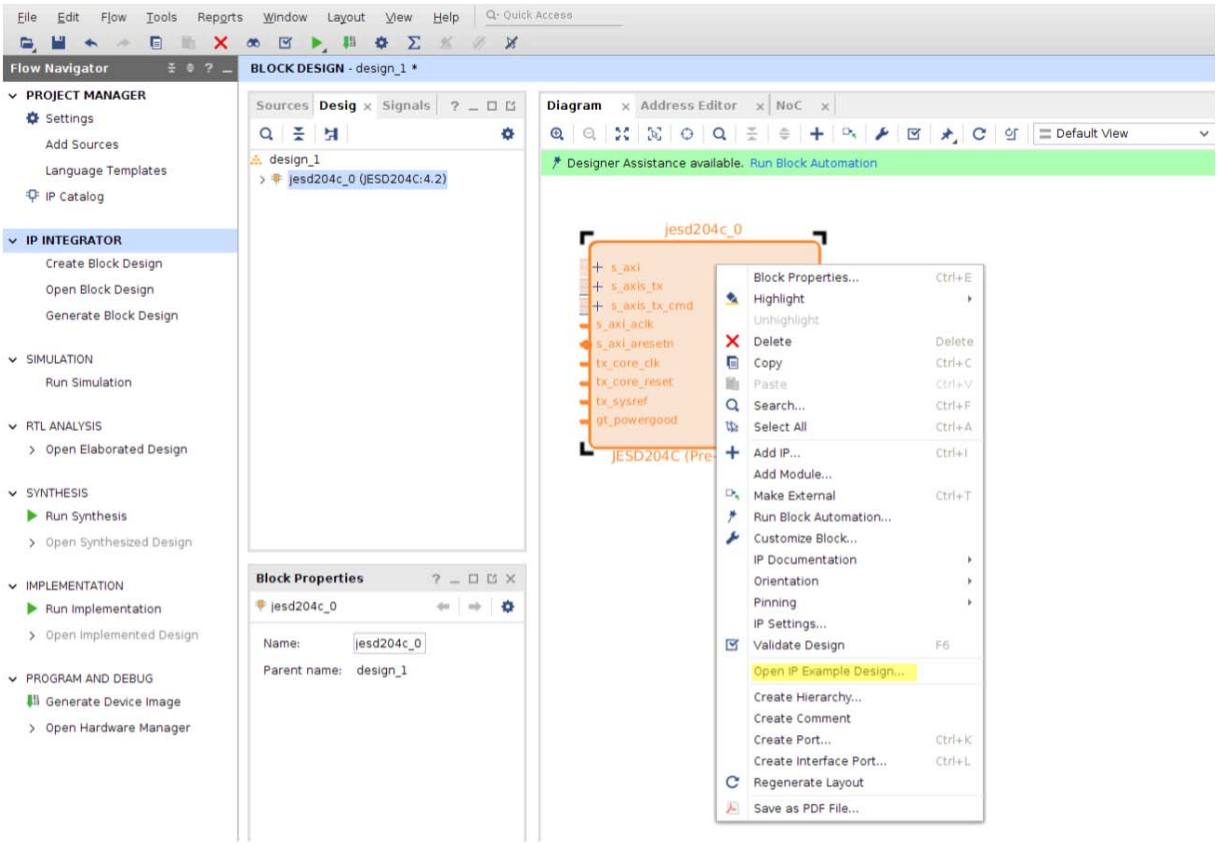

Figure 5-1: **Opening the Example Design**

To create the example design for UltraScale+™ and UltraScale™ devices:

1. In Vivado, create a new empty project.
2. Select the FPGA part that you wish to use.
3. Using the Vivado IP catalog, select the **JESD204C** IP core and configure exactly as required.
4. Right-click the block under **Design Sources** and select **Open IP Example Design**, from the drop-down menu as shown in the following figure. This opens a new Vivado project containing the complete RX or TX design example.

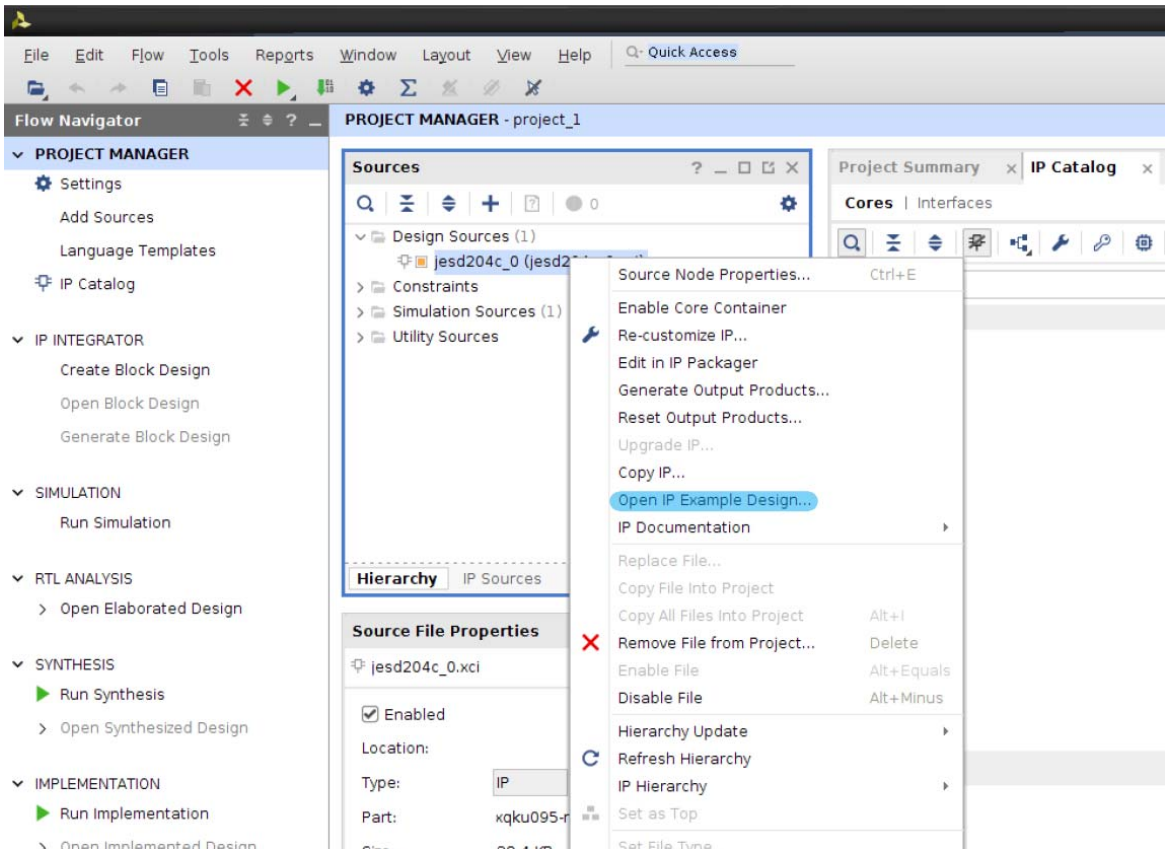

Figure 5-2: **Opening the Example Design**

The following figures show an overview of the example design created for both a JESD204C TX and an JESD204C RX Core. The design contains a full transmit and receive path example sharing a Versal ACAP Transceiver or JESD204_PHY core. The design will generate the JESD204C TX or RX core as configured in the IP GUI. A matching JESD204C TX or RX core will also be generated with settings to complement.

The design is composed of the following main blocks:

- An AXI4-Lite interconnect block to provide multiplexed access to the TX, RX and PHY AXI4-Lite interfaces.
- A simple pattern generator that generates analog sample data and control bits.
- An example mapper that demonstrates mapping the analog sample data and control words into the JESD204C transport layer on the AXI4-Stream interface to drive the TX core.
- (Only in 64b66b cores.) A simple command word generator that passes commands to the TX core. These commands are aligned with the values from the data generator.
- A JESD204C TX core (configuration set in the JESD204C core IP GUI).
- A Versal ACAP transceiver core (configuration set in the JESD204C core GUI GT Wizard Configuration tab) (Figure 5-3).

- A JESD204_PHY core (configuration set in the JESD204C core GUI PHY Configuration tab) (Figure 5-4).
- A JESD204C RX core (configuration set in the JESD204C core IP GUI).
- An example de-mapper that demonstrates the AXI4-Stream interface and the JESD204C transport layer back to analog samples and control words.
- A simple pattern checker that checks the received sample and control words for correctness.
- (Only in 64b66b cores.) A simple command word checker that checks the received command word for correctness.

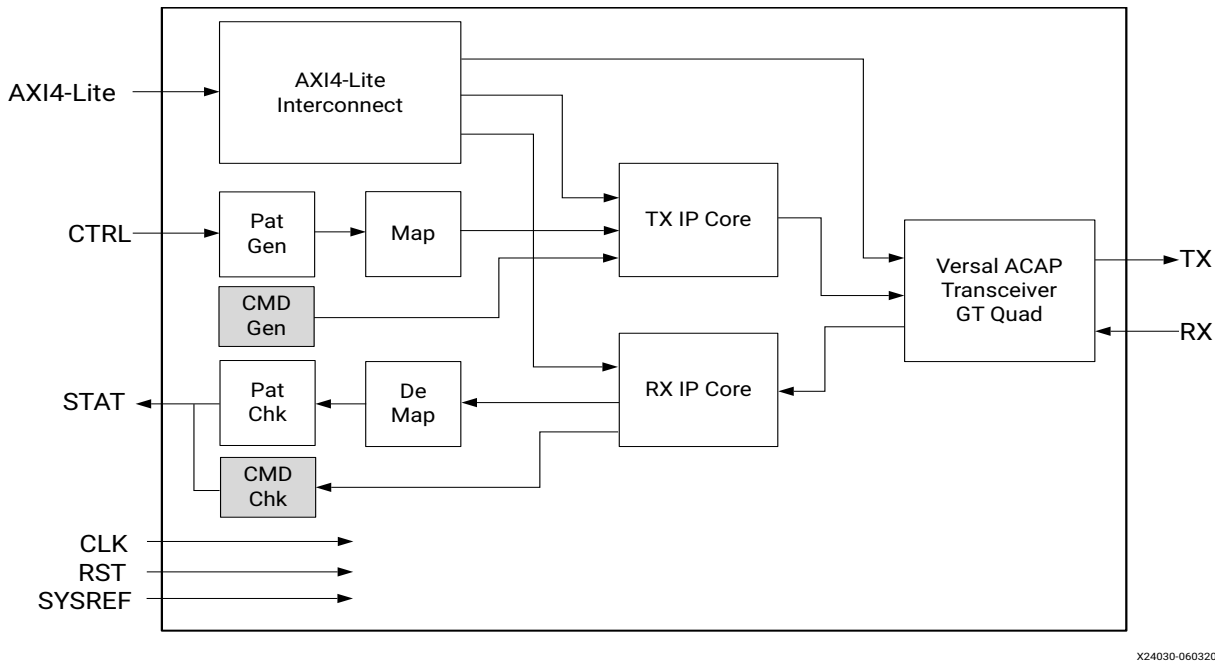

Figure 5-3: **Core Example Design for Versal ACAPs**

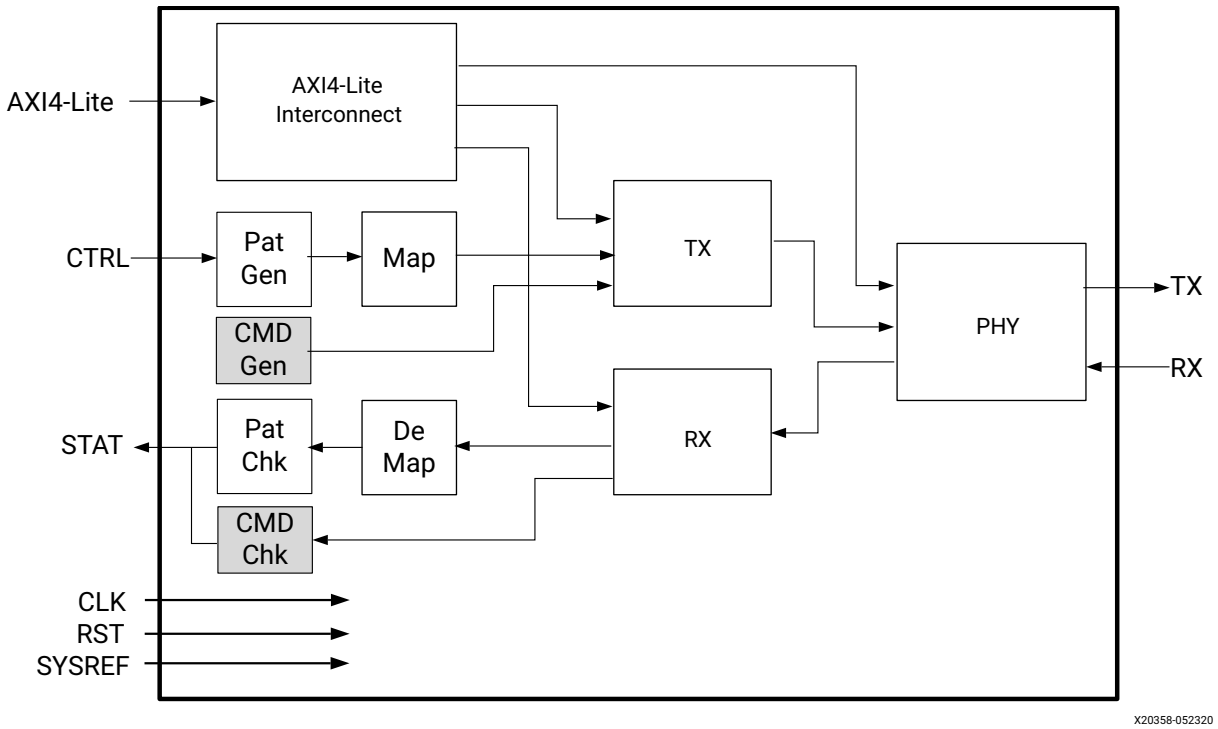

Figure 5-4: **Core Example Design for UltraScale+/UltraScale Devices**

The example design is intended to be used as a starting point for your custom design.

Note: The IO clock buffers necessary for the example design to function as a complete FPGA design are instantiated in the top level wrapper that encapsulates this example design.

Test Bench

The example design supplied with the JESD204C core provides a complete simulation environment including a demonstration test bench that allows you to simulate the core and view the inputs and outputs using the Vivado® Design Suite.

The test bench instantiates the example design described in Chapter 5, and provides the necessary stimulus to show the example design functioning. The test bench can be run at all stages of the design process from behavioral simulation of the RTL code through full post-implementation timing simulation.

Figure 6-1 shows an overview of the test bench delivered with the example design.

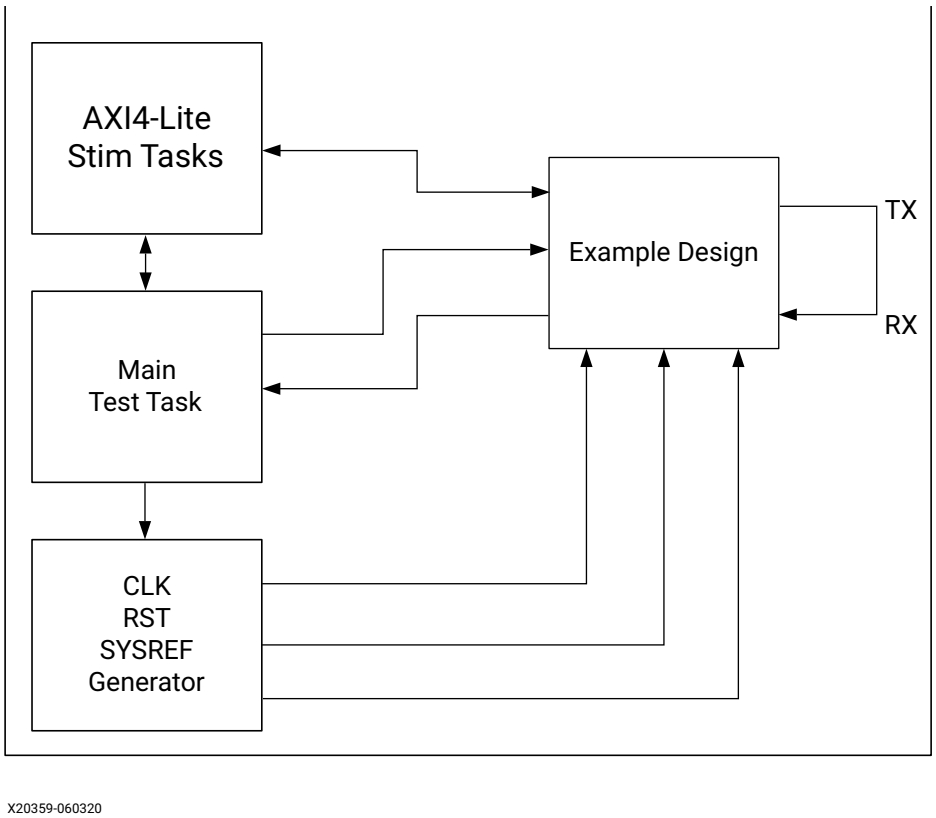

Figure 6-1: **Core Demo Test Bench**

Upgrading

Upgrading from v3.0 to v4.0

The TX and RX Command interface AXI4-Stream tdata ports have been increased in width to 32 bits per lane. Bits [31:19] are unused.

Upgrading from v2.0 to v3.0

No action is required.

Upgrading from v1.0 to v2.0

New ports have been added to the JESD204 PHY interface for TX and RX as follows:

- gtN_txcharisk[3:0]
- gtN_rxcharisk[3:0]
- gtN_rxdisperr[3:0]
- gtN_rxnotintable[3:0]

These ports must be wired to the corresponding ports on the JESD204 PHY core.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

TIP: *If the IP generation halts with an error, there might be a license issue. See License Checkers in Chapter 1 for more details.*

Finding Help on Xilinx.com

To help in the design and debug process when using the JESD204C, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the JESD204C. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as:

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the JESD204C Core

AR: 68804

Technical Support

Xilinx provides technical support at the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Debug Tools

There are many tools available to address JESD204C design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allow you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used with the logic debug LogiCORE IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 6].

Reference Boards

The Xilinx VCU118 evaluation board boards support the JESD204CC. This board can be used to prototype designs and establish that the core can communicate with the system.

Simulation Debug

The simulation debug flow for Questa Advanced Simulator is illustrated in Figure B-1. A similar approach can be used with other simulators.

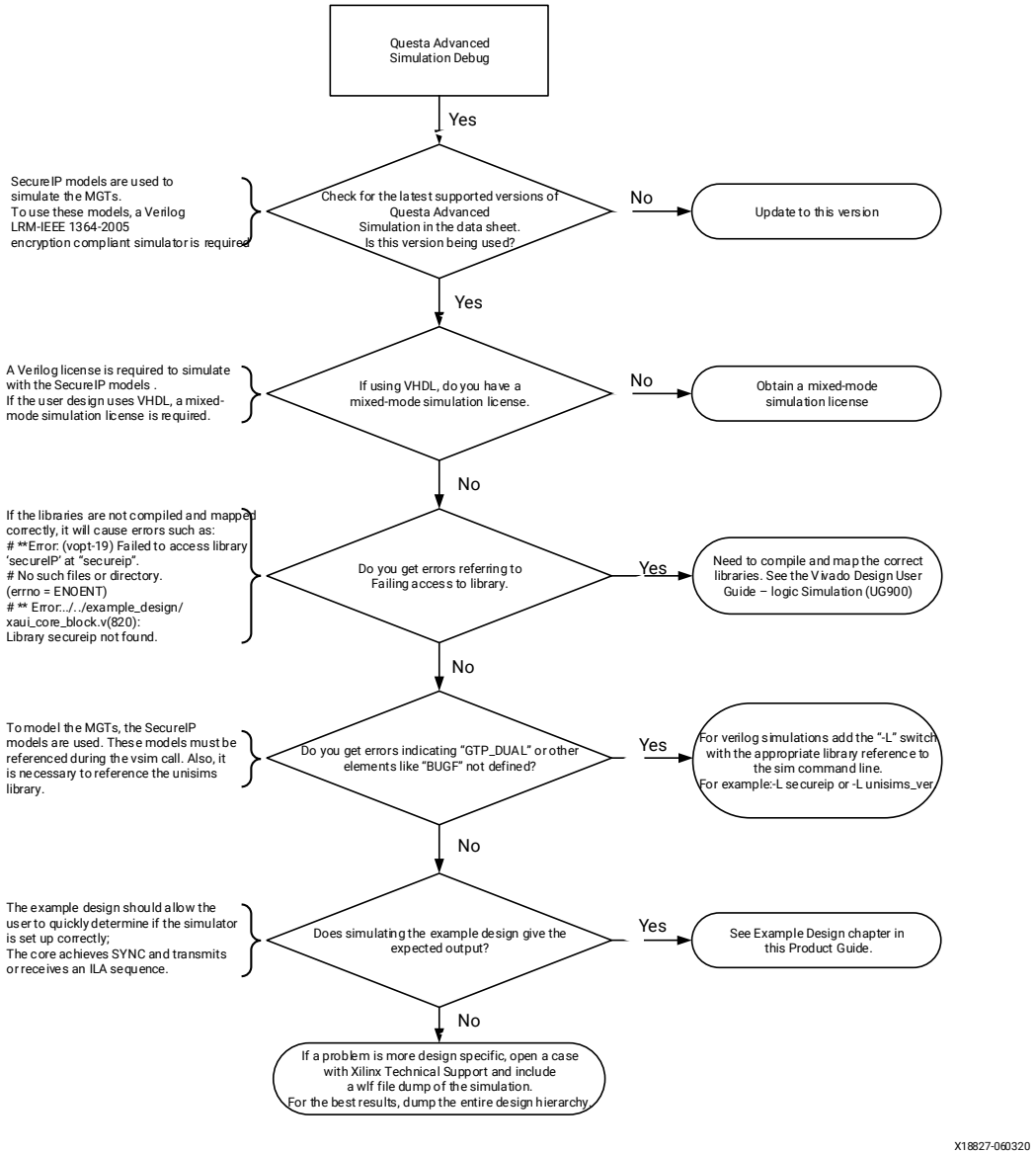

Figure B-1: **Questa Advanced Simulator Debug Flow Diagram**

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the debug feature for debugging the specific problems.

General Checks

- For Versal™ ACAPs, ensure that the core is correctly wired up and that Block Automation has been used to connect up the Versal ACAP Transceiver.
- For UltraScale+™ and UltraScale™ devices, ensure that the core is correctly wired up and that the lane based signals are wired to the correct location on the JESD204_PHY.
- Ensure that all the timing constraints for the core were met during implementation.
- Ensure that all clock sources are clean and in particular that the transceiver reference clocks meet the transceiver requirements from the appropriate FPGA Data Sheet.
- Ensure all clock sources are stable before deasserting the external reset signal to the core.
- For UltraScale+ and UltraScale devices, ensure that all transceiver PLLs have obtained lock by monitoring the QPLLLOCK_OUT and/or CPLLLOCK_OUT port either using the debug feature or by routing the signals to a spare pin.

Issues Obtaining Lane Synchronization

- Ensure that the AXI4-Lite registers have been programmed with the correct value for multi-blocks per extended multi-block.

Interface Debug

AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. Output `s_axi_arready` asserts when the read address is valid, and output `s_axi_rvalid` asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The `S_AXI_ACLK and ACLK` inputs are connected and toggling.
- The interface is not being held in reset, and `S_AXI_ARESET` is an active-Low reset.

- The interface is enabled, and `s_axi_aclken` is active-High (if used).
- The main core clocks are toggling and that the enables are also asserted.
- If the simulation has been run, verify in simulation and/or the Vivado Design Suite debug feature capture that the waveform is correct for accessing the AXI4-Lite interface.

AXI4-Stream Interfaces

If data is not being transmitted or received, check the following conditions:

- If transmit `tready` is stuck Low following the tvalid input being asserted, the transmit core cannot send data.
- If the receive `tvalid` is stuck Low following the `tready` input being asserted, the core is not receiving data.
- Check that the `core_clk` signals are connected to the TX core AXI4-Stream data source or the RX core AXI4-Stream data sink.
- Check that the AXI4-Stream waveforms are being followed (see *Vivado AXI Reference Guide* [Ref 10]).
- Check core configuration.

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support web page.

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on Documentation Navigator, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product guide:

1. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite User Guide: Getting Started* (UG910)
4. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
5. *ISE to Vivado Design Suite Migration Guide* (UG911)
6. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
7. *Vivado Design Suite User Guide: Implementation* (UG904)
8. *AXI Interconnect LogiCORE IP Product Guide* (PG059)
9. JESD204C Standard www.jedec.org
10. *Vivado AXI Reference Guide* (UG1037)
11. *JESD204 PHY LogiCORE IP Product Guide* (PG198)
12. *Versal ACAP Transceivers Wizard LogiCORE IP Product Guide* (PG331)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/25/2021	4.2	Removed the encommaalign_AXI4l block from the Versal ACAP example designs. Added Customized Connections option information for block automation.
07/16/2020	4.2	Added support for Versal devices.
06/03/2020	4.2	• Added support for GTHE3 and GTHE4 transceivers.
05/22/2019	4.1	• Added tolerance radius for SYSREF in SYSREF always mode. • Added configurable lanes per link register.
11/14/2018	4.0	• Modified tx and rx CMD AXI4-Stream tdata ports to be 32-bits per lane. • Modified example design. • Added configuration bits to identify if core was generated with FEC included.
04/04/2018	3.0	Added 8B10B linecoding mode.

Date	Version	Revision
10/4/2017	2.0	• Added ports: gtN_txcharisk[3:0], gtN_rxcharisk[3:0], gtN_rxdisperr[3:0], gtN_rxnotintable[3:0] • Removed reference to individual parts of JESD204C specification.
06/07/2017	1.0	Added GT_POWERGOOD from JESD204_PHY to clocking example description and figure 3-2.
04/05/2017	1.0	Initial Xilinx Release.

Please Read: Important Legal Notices