JESD204 v6.2

LogiCORE IP Product Guide

Vivado Design Suite

PG066 November 18, 2015

XILINX
ALL PROGRAMMABLE™

Table of Contents

Introduction

The Xilinx® LogiCORE™ IP JESD204 core implements a JESD204B interface supporting line rates from 1 Gb/s to 12.5 Gb/s. The JESD204 core can be configured as a transmitter or receiver.

Features

- Designed to JEDEC® JESD204B [Ref 1]
- Supports 1 to 12 lane configurations
- Supports Initial Lane Alignment
- Supports scrambling
- Supports 1–256 octets per frame(1)
- Supports 1–32 frames per multiframe(1)
- Supports Subclass 0, 1, and 2
- Physical and Data Link Layer functions provided
- AXI4-Lite configuration interface [Ref 2]
- AXI4-Stream data interface [Ref 3]
- Supports transceiver sharing between TX and RX cores

1. The maximum supported multiframe size is 1000 octets and the minimum is 20 octets.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	UltraScale+™ Families UltraScale™ Architecture Zynq®-7000 All Programmable SoC 7 Series
Supported User Interfaces	AXI4-Stream, AXI4-Lite Control/Status
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	
Provided by Xilinx at the 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

The LogiCORE™ IP JESD204 core implements a JESD204B interface supporting line rates between 1 and 12.5 Gb/s on 1 to 12 lanes using GTX, GTH, GTP or GTY (UltraScale only) transceivers. See the device data sheets for maximum line rates supported by each device and family. The JESD204 core can be configured as transmit or receive.

The JESD204 core is a fully-verified solution design delivered by using the Xilinx® Vivado® Design Suite. In addition, an example design is provided in Verilog.

Transmitter

Figure 1-1 shows an overview block diagram for the transmitter of the JESD204 core.

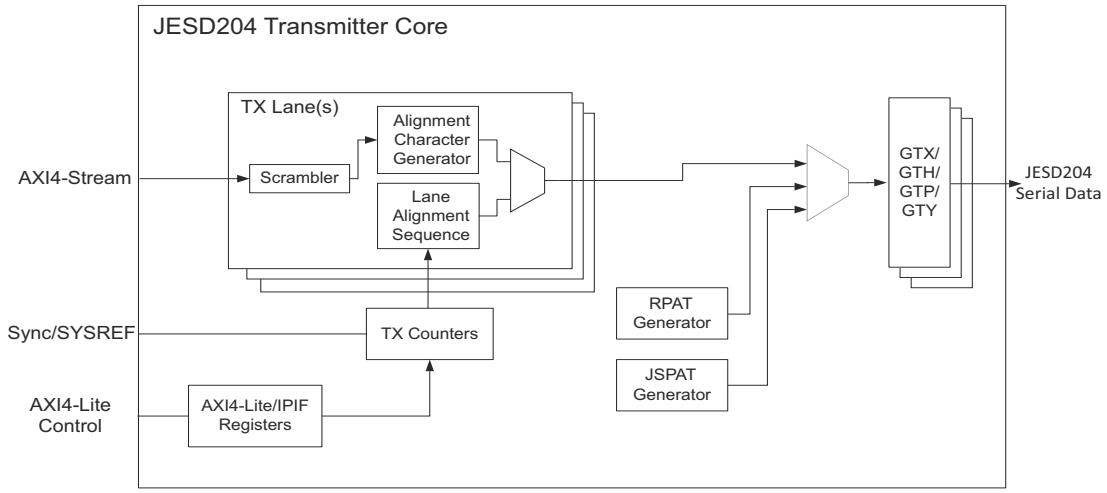


Figure 1-1: **Transmitter Core Overview**

The main blocks are:

- Single AXI4-Stream interface for all lanes
- TX lane logic, per lane, contains:
  - Scrambling
  - Alignment character insertion logic

  - Initial Lane Alignment (ILA) sequence generation
- TX Counters – control, state machine and SYNC/SYSREF interface
- Transceiver wrapper logic
- RPAT generator
- JSPAT generator
- AXI4-Lite Management interface and control/status registers

Receiver

Figure 1-2 shows an overview block diagram for the receiver of the JESD204 core.

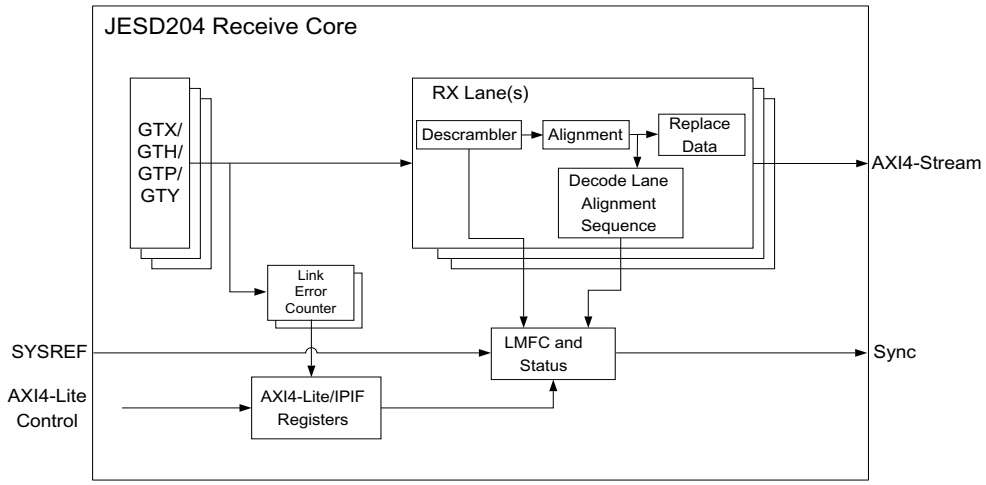


Figure 1-2: **Receiver Core Overview**

The main blocks are:

- Single AXI4-Stream interface for all lanes
- RX lane logic, per lane, contains:
  - ILA capture
  - Descrambling
  - Alignment character detection and replacement logic
- Local Multiframe Clock (LMFC) state machine and SYNC/SYSREF interface
- Transceiver wrapper logic
- Error counters for each lane
- AXI4-Lite Management interface and control/status registers

Core Level Architecture

The JESD204 core is delivered by the Vivado Design Suite with supporting wrapper files. Either a JESD204B transmitter core or a JESD204B receiver core can be selected for generation using the Vivado IDE.

Core-level Verilog wrappers are provided to instantiate the JESD204 IP, the clock/reset logic, Management block, the GTX/GTH/GTP/GTY transceiver, the JSPAT and RPAT pattern generator blocks, and the Error Counting blocks depending on whether the core is a transmitter or a receiver. The core support layer, delivered with the example design, is intended to be instantiated in simple unidirectional designs.

The Management block provides core Control and Status registers with a standard AXI4-Lite interface. The RPAT and JSPAT blocks are optional test pattern generators which can be included in a TX core. Link Error counter blocks are included in a receiver core to support datalink layer test modes and link status monitoring.

A Verilog example design is provided which instantiates the core-level wrapper, together with example interface modules. This is a device-level design and can be used to run the core through the Xilinx tool flow, but is not intended to be used directly in customer designs.

The transmit and receive logic is completely independent; a core can be generated as a transmitter or a receiver. The core can be generated with the transceiver in the example design to allow it to be shared by multiple cores (see Shared Logic Tab).

Applications

JESD204 is a high-speed serial interface designed to connect Analog-to-Digital Converter (ADCs) and Digital-to-Analog Converter (DACs) to logic devices. The JESD204 interface is specified in the *JEDEC® JESD204B Specification* [Ref 1]. Figure 1-3 and Figure 1-4 show how the JESD204 provides the interface between an ADC/DAC and user logic over an example four lane interface.

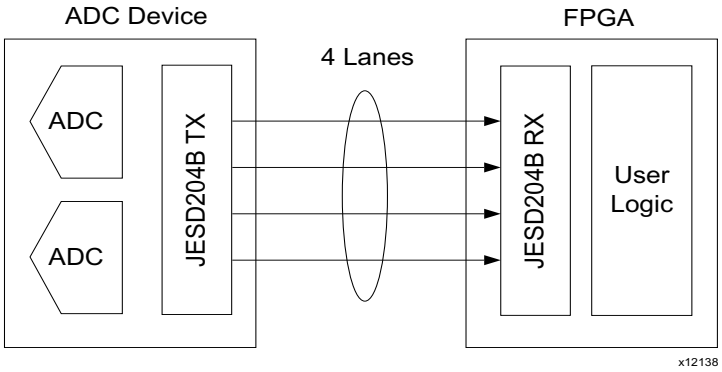


Figure 1-3: **Example ADC Application**

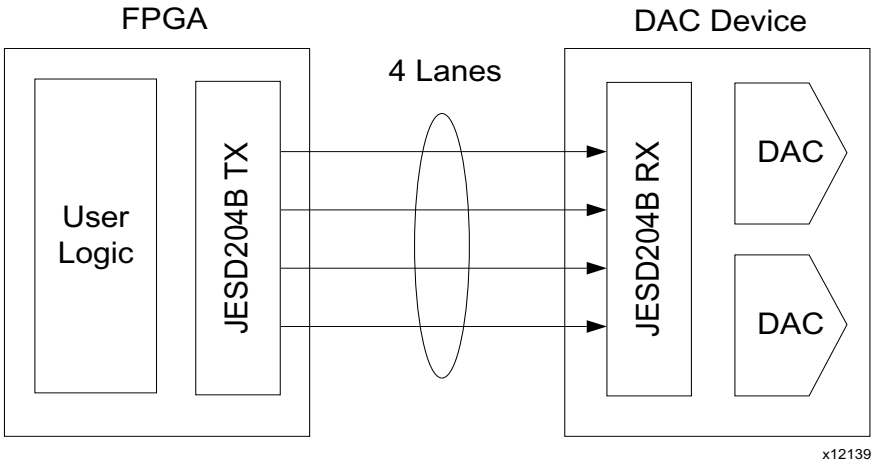


Figure 1-4: **Example DAC Application**

Unsupported Features

Sample data mapping/demapping is not provided by the core, because of the requirement that it be customized for different converter devices. For more information, see Interfacing to the AXI4-Stream Data Interface.

Licensing and Ordering Information

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 JESD204 core provides three licensing options. After installing the Vivado Design Suite and the required IP Service Packs, choose a license 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 JESD204 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 JESD204 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 JESD204 core supports JESD204B. The original JESD204 specification defined a serial link between one data converter and a logic device. The link was made up of one lane. Revision A and B extend this to cover multiple converters, each linked to the logic device using multiple lanes. See Figure 2-1.

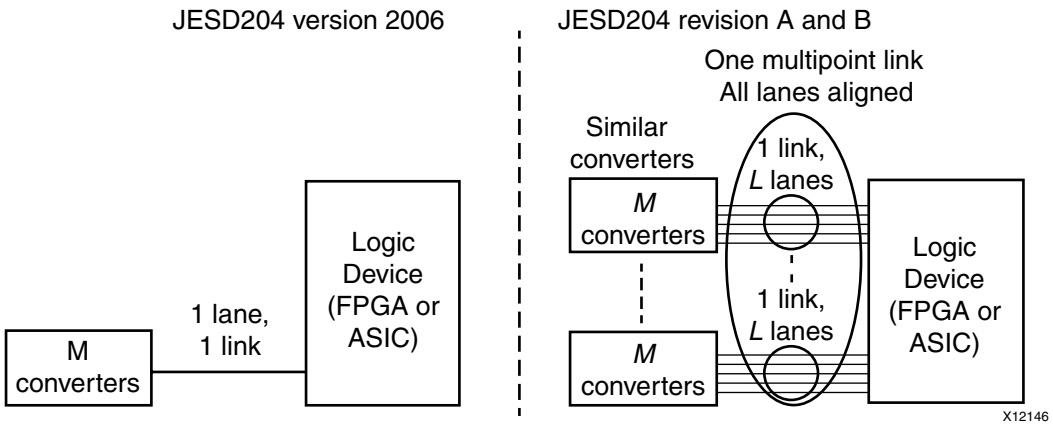


Figure 2-1: **System Overview**

Standards

JEDEC® Serial Interface for Data Converters (JESD204B) available from www.jedec.org.

Performance

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

Resource Utilization

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

Port Descriptions

The port descriptions for the JESD204 core are described in the following sections.

Clock and Reset Ports – TX Core

The clock and reset ports available on the delivered core component depend on the **Shared Logic** selection when customizing the core; see Table 2-1 or Table 2-2.

Table 2-1: **TX Core: Clock and Reset Ports – Shared Logic in Example Design**

Signal Name	Direction	Description
tx_core_clock	In	Core logic clock input. Frequency = serial line rate/40
tx_reset	In	Core asynchronous logic reset.
tx_reset_gt	Out	Transceiver reset. Core output to reset connected transceiver(s)
tx_aclk	Out	AXI4-Stream clock. Associated with the transmit data interface. Runs at the same frequency as tx_core_clock.(1)
tx_aresetn	Out	AXI4-Stream reset. Active-Low. Associated with the transmit data interface.
s_axi_aclk	In	AXI4-Lite clock. Associated with the management interface.
s_axi_aresetn	In	AXI4-Lite reset. Active-Low. Associated with the management interface.

Notes:

1. This signal may be removed in subsequent versions of the core. tx_core_clock should be used instead.

Table 2-2: **TX Core: Clock and Reset Ports – Shared Logic in Core**

Signal Name	Direction	Description
refclk_p/refclk_n	In	Differential transceiver reference clock input Reference clock for the transceiver(s) and Quad Common PLL(s)
glblclk_p/glblclk_n	In	Differential core logic clock input. Additional global logic clock required for Subclass 1 or Subclass 2 operation where the reference clock cannot be used for the synchronous capture of SYSREF or SYNC. Frequency = serial line rate/40. (See Clocking)
drpclk	In	Dynamic Reconfiguration Port (DRP) clock. A free-running DRP clock is required for UltraScale architecture-based devices and optional for 7 series devices.
common0_pll_clk_out	Out	Clock output from the QPLL (Quad 0) associated with serial lanes 0–3. This port is only present when using QPLL.
common0_pll_refclk_out	Out	Reference Clock output from the QPLL (Quad 0) associated with serial lanes 0–3. This port is only present when using QPLL.

Table 2-2: **TX Core: Clock and Reset Ports – Shared Logic in Core *(Cont'd)***

Signal Name	Direction	Description
common0_pll_lock_out	Out	Clock Lock output from the QPLL (Quad 0) associated with serial lanes 0–3. This port is only present when using QPLL. • 1 = Indicates that the QPLL is locked
common1_pll_clk_out	Out	Clock output from the QPLL (Quad 1) associated with serial lanes 4–7. This port is only present for configurations with 5 to 12 lanes and QPLL is selected.
common1_pll_refclk_out	Out	Reference Clock output from the QPLL (Quad 1) associated with serial lanes 4–7. This port is only present for configurations with 5 to 12 lanes.
common1_pll_lock_out	Out	Clock Lock output from the QPLL (Quad 1) associated with serial lanes 4–7. This port is only present for configurations with 5 to 12 lanes and QPLL is selected. • 1 = Indicates that the QPLL is locked.
common2_pll_clk_out	Out	Clock output from the QPLL (Quad 2) associated with serial lanes 8–11. This port is only present for configurations with 9 to 12 lanes and QPLL selected.
common2_pll_refclk_out	Out	Reference clock output from the QPLL (Quad 2) associated with serial lanes 8–11. This port is only present for configurations with 9 to 12 lanes and QPLL selected.
common2_pll_lock_out	Out	Clock lock output from the QPLL (Quad 2) associated with serial lanes 8–11. This port is only present for configurations with 9 to 12 lanes and QPLL selected. • 1 = Indicates that the QPLL is locked.
tx_core_clk_out	Out	Core logic clock output. Frequency = serial line rate/40
tx_aclk	Out	AXI4-Stream clock. Associated with the transmit data interface. This runs at the same frequency as tx_core_clock.(1)
tx_aresetn	Out	AXI4-Stream reset. Active-Low. Associated with the transmit data interface.
s_axi_aclk	In	AXI4-Lite clock. Associated with the management interface.
s_axi_aresetn	In	AXI4-Lite reset. Active-Low. Associated with the management interface.

1. This signal may be removed in subsequent versions of the core. tx_core_clock should be used instead.

Clock and Reset Ports – RX Core

The clock and reset ports available on the delivered core component depend on the **Shared Logic** selection when customizing the core; see Table 2-3 or Table 2-4.

Table 2-3: **RX Core: Clock and Reset Ports – Shared Logic in Example Design**

Signal Name	Direction	Description
rx_core_clock	In	Core logic clock. Frequency = serial line rate/40
rx_reset	In	Core asynchronous logic reset.
rx_reset_gt	Out	Transceiver reset. Core output to reset connected transceiver(s).
rx_aclk	Out	AXI4-Stream clock. Associated with the receive data interface. (1) This runs at the same frequency as rx_core_clock.
rx_aresetn	Out	AXI4-Stream reset. Active-Low. Associated with the receive data interface.
s_axi_aclk	In	AXI4-Lite clock. Associated with the management interface.
s_axi_aresetn	In	AXI4-Lite reset. Active-Low. Associated with the management interface.

Notes:

1. This signal may be removed in subsequent versions of the core. rx_core_clock should be used instead.

Table 2-4: **RX Core: Clock and Reset Ports – Shared Logic in Core**

Signal Name	Direction	Description
refclk_p/refclk_n	In	Differential transceiver reference clock input. Reference clock for the transceiver(s) and Quad Common PLL(s).
glblclk_p/glblclk_n	In	Differential core logic clock input. Additional global logic clock required for Subclass 1 or Subclass 2 operation where the reference clock cannot be used for the Synchronous capture of SYSREF or SYNC. Frequency = serial line rate/40. (See Clocking).
drpclk	In	DRP clock. A free-running DRP clock is required for UltraScale architecture-based devices and optional 7 series devices.
common0_pll_clk_out	Out	Clock output from the QPLL (Quad 0) associated with serial lanes 0–3. This port is present only when QPLL is selected.
common0_pll_refclk_out	Out	Reference Clock output from the QPLL (Quad 0) associated with serial lanes 0–3. This port is present only when QPLL is selected.
common0_pll_lock_out	Out	Clock lock output from the QPLL (Quad 0) associated with serial lanes 0–3. This port is present only when QPLL is selected. • 1 = Indicates that the QPLL is locked
common1_pll_clk_out	Out	Clock output from the QPLL (Quad 1) associated with serial lanes 4–7. This port is only present for configurations with 5 to 12 lanes and QPLL is selected.
common1_pll_refclk_out	Out	Reference Clock output from the QPLL (Quad 1) associated with serial lanes 4–7. This port is only present for configurations with 5 to 12 lanes and QPLL is selected.

Table 2-4: **RX Core: Clock and Reset Ports – Shared Logic in Core *(Cont'd)***

Signal Name	Direction	Description
common1_pll_lock_out	Out	Clock Lock output from the QPLL (Quad 1) associated with serial lanes 4–7. 1 = Indicates that the QPLL is locked. This port is only present for configurations with 5 to 12 lanes and QPLL is selected.
common2_pll_clk_out	Out	Clock output from the QPLL (Quad 2) associated with serial lanes 8–11. This port is only present for configurations with 9 to 12 lanes and QPLL is selected.
common2_pll_refclk_out	Out	Reference Clock output from the QPLL (Quad 2) associated with serial lanes 8–11 and QPLL is selected.
common2_pll_lock_out	Out	Clock Lock output from the QPLL (Quad 2) associated with serial lanes 8–11. 1 = Indicates that the QPLL is locked. This port is only present for configurations with 9 to 12 lanes and QPLL is selected.
rx_core_clk_out	Out	Core logic clock output. Frequency = serial line rate/40
rx_aclk	Out	AXI4-Stream clock. Associated with the RX data interface. This runs at the same frequency as rx_core_clock.(1)
rx_aresetn	Out	AXI4-Stream reset. Active-Low. Associated with the RX data interface.
s_axi_aclk	In	AXI4-Lite clock. Associated with the management interface.
s_axi_aresetn	In	AXI4-Lite reset. Active-Low. Associated with the management interface.

1. This signal may be removed in subsequent versions of the core. rx_core_clock should be used instead.

Transceiver Interface Ports – TX Core

The transceiver ports available on the delivered core component depend on the **Shared Logic** selection when customizing the core; see Table 2-5 or Table 2-6.

Table 2-5: **TX Core: Transceiver Interface Ports – Shared Logic in Example Design**

Signal Name	Direction	Description
gt*N*_txdata[31:0]	Out	TX data to transceiver. *N* = Lanes - 1
gt*N*_txcharisk[3:0]	Out	TX Char is K to transceiver. *N* = Lanes - 1

Table 2-6: **TX Core: Transceiver Interface Ports – Shared Logic in Core**

Signal Name	Direction	Description
txp[N:0]	Out	Positive differential serial data output N = (Lanes - 1)
txn[N:0]	Out	Negative differential serial data output N = (Lanes - 1)

Transceiver Interface Ports – RX Core

The transceiver ports available on the delivered core component depend on the **Shared Logic** selection when customizing the core; see Table 2-7 or Table 2-8.

Table 2-7: **RX Core: Transceiver Interface Ports – Shared Logic in Example Design**

Signal Name	Direction	Description
gt*N*_rxdata[31:0]	In	RX data from transceiver. *N* = 0 to (Lanes - 1)
gt*N*_rxcharisk[3:0]	In	RX Char is K from transceiver. *N* = Lanes - 1
gt*N*_rxdisperr[3:0]	In	RX disparity error from transceiver. *N* = Lanes - 1
gt*N*_rxnotintable[3:0]	In	RX Not In Table from transceiver. *N* = Lanes - 1

Table 2-8: **RX Core: Transceiver Interface Ports – Shared Logic in Core**

Signal Name	Direction	Description
rxp[N:0]	In	Positive differential serial data input N = (Lanes - 1)
rxn[N:0]	In	Negative differential serial data input N = (Lanes - 1)

Transmit Data Interface – TX Core

Note: This interface is clocked by the port tx_core_clock.

Table 2-9: **Transmit Data Interface**

Signal Name	Direction	Description
AXI4-Stream Interface Signals (Transmit Only)		
tx_aresetn	Out	Active-Low reset (shared by all lanes)
tx_tdata[(32*N)-1:0]	In	AXI transmit data (samples and control words); transmitted least significant byte first. Data for Serial Lane 0 on tx_tdata[31:0] Data for Serial Lane 1 on tx_tdata[63:32] ... Data for Serial Lane N on tx_tdata[((N + 1) × 32) - 1:(N × 32))]
tx_tready	Out	AXI slave ready for data
Non-AXI Data Interface Signals (Transmit Only)		

Table 2-9: **Transmit Data Interface *(Cont'd)***

Signal Name	Direction	Description
tx_start_of_frame[3:0]	Out	Frame boundary indication. The signal is 4 bits to indicate the byte position of the first byte of a frame in tdata in the following clock cycle. • When start_of_frame = 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 start_of_frame = 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 start_of_frame = 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 start_of_frame = 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_start_of_frame can be asserted in the same cycle, depending on the number of octets per frame (for example, for F = 1, tx_start_of_frame = 1111)
tx_start_of_multiframe [3:0]	Out	Multi-frame boundary indication. The position of the first byte of each multiframe is encoded in the same way as start_of_frame.
tx_sysref	In	SYSREF input. When Subclass 1 mode is selected, this signal is required and used by the core. JESD204B specifies a SYSREF signal must be generated synchronous to the core clock (see Clocking for details). This input should be driven from an external device generating SYSREF for both TX and RX.
tx_sync(1)	In	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.

Notes:

1. See the *JEDEC JESD204 specifications* [Ref 1] for details of this signal.

Figure 2-2 shows the timing of `tx_start_of_frame` and `tx_start_of_multiframe` relative to the AXI data. `tx_start_of_frame` and `tx_start_of_multiframe` 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_start_of_frame` = 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]`.

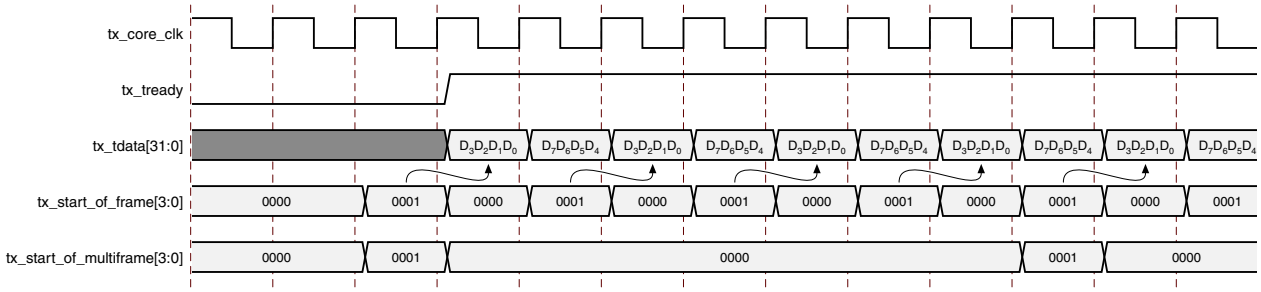


Figure 2-2: **Transmit Data Interface Timing for F = 8 and K = 4**

Receive Data Interface – RX Core

Note: This interface is clocked by the port rx_core_clock.

Table 2-10: **Receive Data Interface**

Signal Name	Direction	Description
AXI4-Stream Interface Signals (RX Only)		
rx_aresetn	Out	Active-Low reset (shared by all lanes)
rx_tdata[(32*N)-1:0]	Out	AXI receive data (samples and control words). Data in least significant byte was received first. Data from Serial Lane 0 on rx_tdata[31:0] Data from Serial Lane 1 on rx_tdata[63:32] ... Data from Serial Lane N on rx_tdata[((N + 1) × 32) - 1:(N × 32))]
rx_tvalid	Out	AXI receive data valid
Non-AXI Data Interface Signals (RX Only)		
rx_start_of_frame[3:0]	Out	Frame boundary indication. The position of the first byte in a frame is encoded in the same way as tx_start_of_frame[3:0]. This signal is asserted one cycle before the AXI4-Stream data. The alignment of the very first valid byte is always in byte 0 if the multiframe size is a multiple of 4 and rx_buffer_delay is a multiple of 4.

Table 2-10: **Receive Data Interface *(Cont'd)***

Signal Name	Direction	Description
rx_end_of_frame[3:0]	Out	Frame boundary indication. The position of the last byte in a frame is encoded in the same way as start_of_frame.
rx_frame_error [(LANES*4)-1:0]	Out	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_frame_error 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_aclk and output in the cycle before the data in the same way as rx_start_of_frame.
rx_sync	Out	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.
rx_sysref	In	SYSREF Input. When Subclass 1 mode is selected, this signal is required and used by the core. JESD204B specifies that a SYSREF signal must be generated synchronous to the core clock (see Clocking for details). This input should be driven from an external device generating SYSREF for both TX and RX.

Figure 2-3 and Figure 2-4 show the timing of `rx_start_of_frame` and `rx_end_of_frame` relative to the AXI data. `rx_start_of_frame` and `rx_start_of_frame` are fixed at 4 bits wide because the internal data width of each lane is 32 bits and the start (or end) of frame can occur in any of the 4-byte positions of the 32-bit word. For multi-lane configurations the start (or end) of frame 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_start_of_frame` = 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]`.

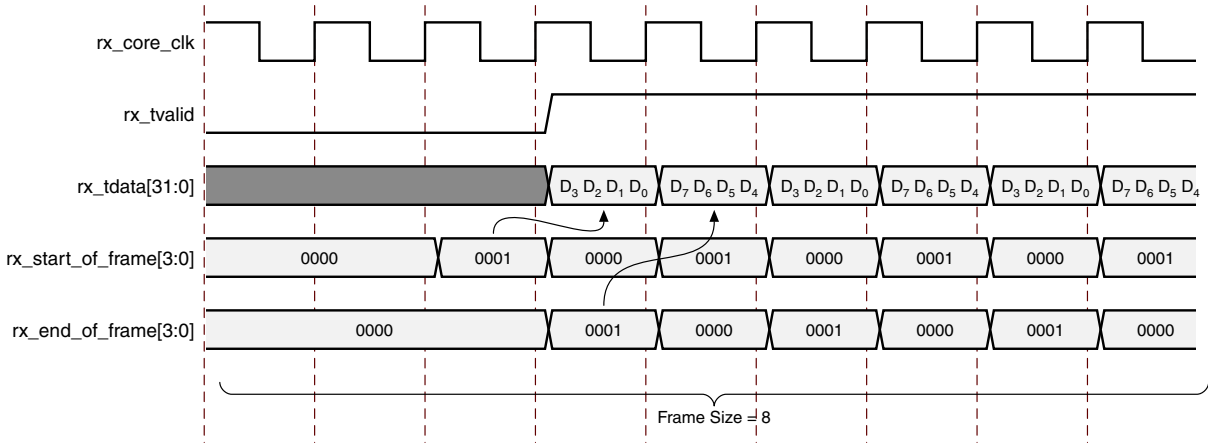


Figure 2-3: **Receive Data Interface Timing for F = 8**

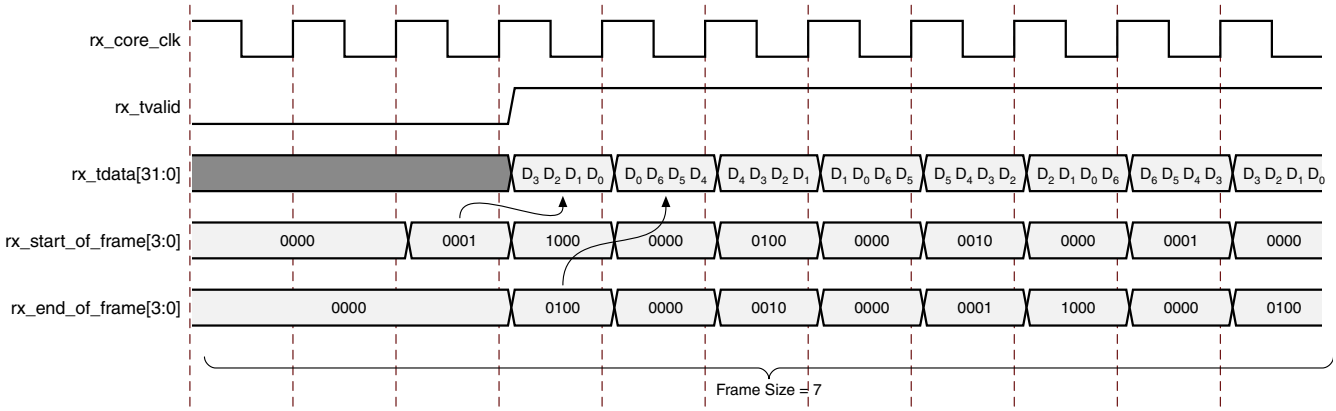


Figure 2-4: **Receive Data Interface Timing for F = 7**

Management Interface (AXI4-Lite)

Also, see the *Xilinx Vivado AXI Reference Guide* (UG1037) [Ref 15] for AXI4-Lite interface information.

Table 2-11: **Management Interface (AXI4-Lite)**

Signal Name	Direction	Description
s_axi_aclk	In	Clock
s_axi_aresetn	In	Active-Low reset
s_axi_awaddr[11:0]	In	Write Address
s_axi_awvalid	In	Write Address Valid
s_axi_awready	Out	Write Address Ready
s_axi_wdata[31:0]	In	Write Data

Table 2-11: **Management Interface (AXI4-Lite) *(Cont'd)***

Signal Name	Direction	Description
s_axi_wstrb[3:0]	In	Write Data Byte Strobe
s_axi_wvalid	In	Write Data Valid
s_axi_wready	Out	Write Data Ready
s_axi_bresp[1:0]	Out	Write Response (Always = 00 = OK)
s_axi_bvalid	Out	Write Response Valid
s_axi_bready	In	Write Response Ready
s_axi_araddr[11:0]	In	Read Address
s_axi_arvalid	In	Read Address Valid
s_axi_arready	Out	Read Address Ready
s_axi_rdata[31:0]	Out	Read Data
s_axi_rresp[1:0]	Out	Read Response (Always = 00 = OK)
s_axi_rvalid	Out	Read Data Valid
s_axi_rready	In	Read Data Ready

Transceiver Debug Interface

IMPORTANT: *The ports in the Transceiver Control and Status Interface must be driven in accordance with the appropriate GT user guide. Using the input signals listed in Table 2-12 and Table 2-13 might result in unpredictable behavior of the IP core.*

The transceiver debug interface (when present) provides access to transceiver control and status pins for debug purposes. See the appropriate transceiver user guide (*UltraScale Architecture GTH Transceivers User Guide* (UG576) [Ref 6], *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476) [Ref 7], or *7 Series FPGAs GTP Transceivers User Guide* (UG482) [Ref 8]) for a detailed description of these pins. This interface is only present on the core when "Include Shared Logic in core" and "Additional transceiver control and status ports" options are selected when generating the core.

Table 2-12: **Optional Transceiver Debug Ports (7 Series Devices)**

Signal Name[(1)(2)]	Direction	Clock Domain	Description
gtN_drpaddr [8:0]	In	drp_clk	DRP Address Bus
gtN_drpdi [15:0]	In	drp_clk	Data bus for writing configuration data from the FPGA logic resources to the transceiver
gtN_drpen	In	drp_clk	DRP Enable Signal 0: No read or write operation performed 1: Enables a read or write operation
gtN_drpwe	In	drp_clk	DRP Write Enable 0: Read operation when DEN is 1 1: Write operation when DEN is 1

Table 2-12: **Optional Transceiver Debug Ports (7 Series Devices) *(Cont'd)***

Signal Name[1][2]	Direction	Clock Domain	Description
gtN_drpdo [15:0]	Out	drp_clk	Data bus for reading configuration data from the GTX/GTH transceiver to the FPGA logic resources
gtN_drprdy	Out	drp_clk	Indicates operation is complete for write operations and data is valid for read operations
loopback[2:0]	In	Async	Transceiver Loopback: • 000: No loopback • 001: Near-End PCS Loopback • 010: Near-End PMA Loopback • 100: Far-End PMA Loopback • 110: Far-End PCS Loopback
gtN_txpostcursor[4:0]	In	tx_core_clock	Transmit Differential Driver control. (TX only)
gtN_txpresursor[4:0]	In	tx_core_clock	Transmit Differential Driver control. (TX only)
gtN_txdiffctrl[3:0]	In	tx_core_clock	Transmit Differential Driver control. (TX only)
gtN_txpolarity	In	tx_core_clock	Transmit polarity control. (TX only)
gt_txdata[X:0]	Out	tx_core_clock	Copy of raw transceiver data bus between core and transceiver, width depends on number of lanes. (TX only)
gt_txcharisk[Y:0]	Out	tx_core_clock	Copy of raw transceiver charisk bus between core and transceiver, width depends on number of lanes. (TX only)
gtN_rxpolarity	In	rx_core_clock	Receive polarity control. (RX only)
gt_rxdata[X:0]	Out	rx_core_clock	Copy of raw transceiver data bus between core and transceiver, width depends on number of lanes. (RX only)
gt_rxcharisk[Y:0]	Out	rx_core_clock	Copy of raw transceiver charisk bus between core and transceiver, width depends on number of lanes. (RX only)
gt_rxdisperr[Y:0]	Out	rx_core_clock	Copy of raw transceiver disparity error bus between core and transceiver, width depends on number of lanes. (RX only)
gt_rxnotintable[Y:0]	Out	rx_core_clock	Copy of raw transceiver not-in-table error bus between core and transceiver, width depends on number of lanes. (RX only)
gtN_cplllock_out	Out	Async	Active-High signal indicating that the channel PLL has locked to the input reference clock
gtN_eyescandataerror_out	Out	Async	Asserted when an EYESCAN error occurs
gtN_eyescanreset_in	In	Async	This port is pulsed High to initiate the EYESCAN reset process
gtN_eyescantrigger_in	In	rx_core_clock	A High on this port causes an EYESCAN trigger event
gtN_rxbufreset_in	In	Async	This port is driven High and then deasserted to start the RX elastic buffer reset process.
gtN_rxbufstatus_out[2:0]	Out	rx_core_clock	RX Elastic Buffer Status
gtN_rxbyteisaligned_out	Out	rx_core_clock	RX Byte Alignment Status
gtN_rxbyterealign_out	Out	rx_core_clock	RX Byte Alignment has changed

Table 2-12: **Optional Transceiver Debug Ports (7 Series Devices) *(Cont'd)***

Signal Name[1][2]	Direction	Clock Domain	Description
gtN_rxcdrhold_in	In	Async	Hold the CDR control loop frozen
gtN_rxcommadet_out	Out	rx_core_clock	RX Comma Detect Out
gtN_rxdfelpmreset_in	In	Async	DFE Reset
gtN_rxlpmen_in	In	rx_core_clock	LPM Mode Enable
gtN_rxmonitorout_out	Out	Async	RX Monitor Out
gtN_rxmonitorsel_in	In	Async	RX Monitor Out Mode Select
gtN_rxpcsreset_in	In	Async	PCS Reset
gtN_rxpd_in[1:0]	In	Async	RX Power Down
gtN_rxpmareset_in	In	Async	PMA Reset
gtN_rxprbscntreset_in	In	rx_core_clock	RX PRBS Counter Reset
gtN_rxprbserr_out	Out	rx_core_clock	RX PRBS Error Detect
gtN_rxprbssel_in	In	rx_core_clock	RX PRBS Select
gtN_rxresetdone_out	Out	rx_core_clock	RX Reset Done
gtN_rxstatus_out[2:0]	Out	rx_core_clock	Encodes RX Status and Error Codes
gtN_txbufstatus_out[1:0]	Out	tx_core_clock	TX Elastic Buffer Status
gtN_txpcsreset_in	In	Async	TX PCS Reset
gtN_txinhibit	In	tx_core_clock	TX Inhibit
gtN_txpd_in	In	tx_core_clock	TX Power Down
gtN_txpmareset_in	In	Async	TX PMA Reset
gtN_txprbsforceerr_in	In	tx_core_clock	TX PRBS Force Error
gtN_txresetdone_out	Out	tx_core_clock	TX Reset Done
gtN_rxlpmhfhold_in	In	rx_core_clock	(GTP Only) LPM Mode Control
gtN_rxlpmhfoverden_in	In	rx_core_clock	(GTP Only) LPM Mode Control
gtN_rxlpmlfhold_in	In	rx_core_clock	(GTP Only) LPM Mode Control

Notes:

1. N is the number of the transceiver channel.
2. If you are migrating from a 7 series to an UltraScale architecture-based device, the prefixes of the optional transceiver debug ports for single-lane cores are changed from `gt0`, `gt1` to `gt`, and the postfix `_in` and `_out` are dropped. For multi-lane cores, the prefixes of the optional transceiver debug ports `gt(n)` are aggregated into a single port (see Table 2-13).

Table 2-13: **Optional Transceiver Debug Ports (UltraScale Architecture-Based Devices)**

Signal Name[1]	Direction	Clock Domain	Description
gtN_drpaddr [8:0]	In	drp_clk	DRP Address Bus ***Note:*** GTH=[8:0], GTY=[9:0]
gtN_drpdi [15:0]	In	drp_clk	Data bus for writing configuration data from the FPGA logic resources to the transceiver

Table 2-13: **Optional Transceiver Debug Ports (UltraScale Architecture-Based Devices) *(Cont'd)***

Signal Name[(1)]	Direction	Clock Domain	Description
gtN_drpen	In	drp_clk	DRP Enable Signal 0: No read or write operation performed 1: Enables a read or write operation
gtN_drpwe	In	drp_clk	DRP Write Enable 0: Read operation when DEN is 1 1: Write operation when DEN is 1
gtN_drpdo [15:0]	Out	drp_clk	Data bus for reading configuration data from the GTX/GTH transceiver to the FPGA logic resources
gtN_drprdy	Out	drp_clk	Indicates operation is complete for write operations and data is valid for read operations
gt_txpmareset [(LANES-1):0]	In	Async	Port is pulsed High to start the TX PMA reset process
gt_txpcsreset [(LANES-1):0]	In	Async	Port is pulsed High to start the TX PCS reset process
gt_txresetdone [(LANES-1):0]	Out	tx_core_clock	A High on this port indicates that the TX reset process has completed
gt_rxpmareset [(LANES-1):0]	In	Async	Port is pulsed High to start the RX PMA reset process
gt_rxpcsreset [(LANES-1):0]	In	Async	Port is pulsed High to start the RX PCS reset process
gt_rxbufreset [(LANES-1):0]	In	Async	Port is driven High and then deasserted to start the RX elastic buffer reset process
gt_rxpmaresetdone [(LANES-1):0]	Out	Async	A High on this port indicates that the RX PMA reset process has completed
gt_rxresetdone [(LANES-1):0]	Out	rx_core_clock	A High on this port indicates that the RX reset process has completed
gt_txbufstatus [(LANES*2)-1:0]	Out	tx_core_clock	Elastic Buffer Status
gt_rxbufstatus [(LANES*3)-1:0]	Out	rx_core_clock	RX Elastic Buffer Status
gt_cplllock [(LANES-1):0]	Out	refclk	Active-High signal indicating that the channel PLL has locked to the input reference clock
gt_rxrate [(LANES*3)-1:0]	In	rx_core_clock	Link signaling rate control
gt_eyescantrigger [(LANES-1):0]	In	rx_core_clock	A High on this port causes an EYESCAN trigger event
gt_eyescanreset [(LANES-1):0]	In	Async	Port is pulsed High to initiate the EYESCAN reset process
gt_eyescandataerror [(LANES-1):0]	Out	Async	Asserted when an EYESCAN error occurs

Table 2-13: **Optional Transceiver Debug Ports (UltraScale Architecture-Based Devices) *(Cont'd)***

Signal Name[(1)]	Direction	Clock Domain	Description
gt_loopback [(LANES*3)-1:0]	In	Async	Transceiver Loopback: • 000: No loopback • 001: Near-End PCS Loopback • 010: Near-End PMA Loopback • 100: Far-End PMA Loopback • 110: Far-End PCS Loopback
gt_rxpolarity [(LANES-1):0]	In	rx_core_clock	Set High to invert the incoming serial data
gt_txpolarity [(LANES-1):0]	In	tx_core_clock	Set High to invert the outgoing serial data
gt_rxdfelpmreset [(LANES-1):0]	In	Async	Reset for the LPM and DFE datapath
gt_rxlpmen [(LANES-1):0]	In	rx_core_clock	Set to 1 to select the LPM datapath
gt_txprecursor [(LANES*5)-1:0]	In	tx_core_clock	Transmitter pre-cursor pre-emphasis control
gt_txpostcursor [(LANES*5)-1:0]	In	tx_core_clock	Transmitter post-cursor pre-emphasis control
gt_txdiffctrl [(LANES*4)-1:0]	In	tx_core_clock	Driver swing control
gt_txprbsforceerr [(LANES-1):0]	In	tx_core_clock	Set High to drive errors into the PRBS transmitter
gtN_txinhibit	In	tx_core_clock	TX Inhibit
gt_pcsrsvdin [(LANES*16)-1:0]	In	Async	DRP Reset (Bits 2, 18, 34,....)
gt_txprbssel [(LANES*4)-1:0]	In	tx_core_clock	Transmitter PRBS generator test pattern control
gt_rxprbssel [(LANES*4)-1:0]	In	rx_core_clock	Receiver PRBS checker test pattern control
gt_rxprbserr [(LANES-1):0]	In	rx_core_clock	A High on this port indicates that PRBS errors have occurred
gt_rxprbscntreset [(LANES-1):0]	In	rx_core_clock	Reset the PRBS error counter
gt_rxcdrhold [(LANES-1):0]	In	Async	Hold the CDR control loop frozen
gt_dmonitorout [(LANES*15-1):0]	Out	Async	Digital Monitor Output Bus
gt_rxdisperr [(LANES*4-1):0]	Out	rx_core_clock	Receiver disparity error indicator
gt_rxnotintable [(LANES*4-1):0]	Out	rx_core_clock	Receiver not in table error indicator

Table 2-13: **Optional Transceiver Debug Ports (UltraScale Architecture-Based Devices) *(Cont'd)***

Signal Name[1]	Direction	Clock Domain	Description
gt_rxcommadet [(LANES-1):0]	Out	rx_core_clock	A High on this port indicates that the comma alignment block has detected a valid comma
gt_rxpd [(LANES*2-1):0]	In	Async	RX Power Down 00=Normal Operation 11=Lowest power mode
gt_txpd [(LANES*2-1):0]	In	tx_core_clock	TX Power Down 00=Normal Operation 11=Lowest power mode

Notes:
1. N is the number of the transceiver channel.

Register Space

The JESD204 core is configured using an AXI4-Lite Register Interface. The register map is shown in Table 2-14.

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 2-14: **Register Address Map**

RX Core Registers			TX Core Registers	
Address Offset	Description	R/W	Description	R/W
0x000	Version	R	Version	R
0x004	Reset	R/W	Reset	R/W
0x008	ILA Support	R/W	ILA Support	R/W
0x00C	Scrambling	R/W	Scrambling	R/W
0x010	SYSREF Handling	R/W	SYSREF Handling	R/W
0x014	–	–	ILA Multiframes	R/W
0x018	Test Modes	R/W	Test Modes	R/W
0x01C	Link Error Status (Lanes 0 to 7)	–	–	–
0x020	Octets per Frame	R/W	Octets per Frame	R/W
0x024	Frames per Multiframe	R/W	Frames per Multiframe	R/W
0x028	Lanes in Use	R/W	Lanes in Use	R/W
0x02C	Subclass Mode	R/W	Subclass Mode	R/W

Table 2-14: **Register Address Map *(Cont'd)***

RX Core Registers			TX Core Registers	
Address Offset	**Description**	**R/W**	**Description**	**R/W**
0x030	RX Buffer Delay (RX Only)	R/W	–	–
0x034	Error Reporting (RX Only)	R/W	–	–
0x038	Sync Status	R	Sync Status	R
0x03C	Link Error Status (Lanes 8 to 11)	R	–	–
0x040–0x7FC	Reserved	–	Reserved	–
0x400	Reserved	R/W	Lane 0 ID	R/W
0x404	Reserved	R/W	Lane 1 ID	R/W
0x408	Reserved	R/W	Same as Lane 1 ID	R/W
0x40C	Reserved	R/W	Same as Lane 1 ID	R/W
0x410	Reserved	R/W	Same as Lane 1 ID	R/W
0x414	Reserved	R/W	Same as Lane 1 ID	R/W
0x418	Reserved	R/W	Same as Lane 1 ID	R/W
0x41C	Reserved	R/W	Same as Lane 1 ID	R/W
0x420	Reserved	R/W	Same as Lane 1 ID	R/W
0x424	Reserved	R/W	Same as Lane 1 ID	R/W
0x428	Reserved	R/W	Same as Lane 1 ID	R/W
0x42C	Reserved	R/W	Same as Lane 1 ID	R/W
0x430	Reserved	R/W	Same as Lane 1 ID	R/W
0x800	ILA Config Data 0	R	–	–
0x804	ILA Config Data 1	R	–	–
0x808	ILA Config Data 2	R	–	–
0x80C	ILA Config Data 3	R	ILA Config Data 3	R/W
0x810	ILA Config Data 4	R	ILA Config Data 4	R/W
0x814	ILA Config Data 5	R	ILA Config Data 5	R/W
0x818	ILA Config Data 6	R	ILA Config Data 6	R/W
0x81C	ILA Config Data 7	R	ILA Config Data 7	R/W
0x820	Test Mode Error Count	R	–	–
0x824	Link Error Count	R	–	–
0x828	Test Mode ILA Count	R	–	–
0x82C	Test Mode Multiframe Count	R	–	–
0x830	Buffer Adjust	R	–	–
0x834–0x83C	Reserved	–	–	–

Table 2-14: **Register Address Map *(Cont'd)***

RX Core Registers			TX Core Registers	
Address Offset	**Description**	**R/W**	**Description**	**R/W**
0x840–0x87C	Same as 0x800–0x83C for Lane 1	R	–	–
0x880–0x8BC	Same as 0x800–0x83C for Lane 2	R	–	–
0x8C0–0x8FC	Same as 0x800–0x83C for Lane 3	R	–	–
0x900–0x93C	Same as 0x800–0x83C for Lane 4	R	–	–
0x940–0x97C	Same as 0x800–0x83C for Lane 5	R	–	–
0x980–0x9BC	Same as 0x800–0x83C for Lane 6	R	–	–
0x9C0–0x9FC	Same as 0x800–0x83C for Lane 7	R	–	–
0xA00–0xA3C	Same as 0x800–0x83C for Lane 8	R	–	–
0xA40–0xA7C	Same as 0x800–0x83C for Lane 9	R	–	–
0xA80–0xABC	Same as 0x800–0x83C for Lane 10	R	–	–
0xAC0–0xAFC	Same as 0x800–0x83C for Lane 11	R	–	–

Table 2-15: **Version**

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

Table 2-16: **Reset**

Bits	Default Value	Description
31:17	–	Reserved
16	1	Watchdog Timer Disable(1): 1 = Disable GT watchdog timer
15:1	–	Reserved
0	1	Reset Write 1 to reset core Read: 1 = reset in progress; 0 = reset complete

Notes:

1. Applicable to GTXE2 transceivers only.

Table 2-17: **ILA Support**

Bits	Default Value	Description
31:1	–	Reserved

Table 2-17: **ILA Support *(Cont'd)***

Bits	Default Value	Description
0	1	1 = Enable ILA Support[(1)]; 0 = Disable ILA Support

Notes:

1. When Enable ILA Support is 1, a TX core transmits an Inter Lane Alignment (ILA) sequence; an RX core expects to receive an ILA sequence. Should always be enabled for Subclass 1 or 2 operation and any multi-lane configuration.

Table 2-18: **Scrambling**

Bits	Default Value	Description
31:1	–	Reserved
0	0	1 = Enable Scrambling; 0 = Disable Scrambling

Table 2-19: **SYSREF Handling**

Bits	Default Value	Description
31:17	–	Reserved
16	0	SYSREF Required on Re-Sync 1 = A SYSREF event is required following a Link Re-Sync event: TX core transmits K28.5 until a SYSREF re-aligns the LMFC; RX core does not deassert SYNC until a SYSREF event re-aligns the LMFC. 0 = No SYSREF event is required on a Link Re-Sync event: TX core transmits ILA sequence on the next LMFC. RX core deasserts SYNC on the next LMFC.
15:12	–	Reserved
11:8	0000	SYSREF delay: add additional delay to SYSREF re-alignment of LMFC counter 1111 = 15 core_clk cycles delay .... 0000 = 0 core_clk cycles delay
7:1	–	Reserved
0	0	SYSREF Always 1 = Core re-aligns LMFC counter on all SYSREF events 0 = Core only aligns LMFC counter on the first SYSREF event detected following reset, and ignores subsequent SYSREF events

Table 2-20: **ILA Multiframes**

Bits	Default Value	Description
31:8	–	Reserved
7:0	0x03	Multiframes in the Transmitted Initial Lane Alignment Sequence Parameter Range: 4–256; program the register with required value minus 1

Table 2-21: **Test Modes**

Bits	Default Value	Description
31:3	–	Reserved
2:0	000	Test Mode Select 000: Normal operation 001: Transmit/Receive /K28.5/ indefinitely 010: Synchronize as normal then transmit/receive repeated ILA sequences 011: Transmit/receive /D21.5/ indefinitely 101: Transmit Modified Random Pattern (RPAT) [TX ONLY] 111: Transmit Scrambled Jitter Pattern (JSPAT) [TX ONLY] Other bit combinations: unsupported

Table 2-22: **Link Error Status (Lanes 0 to 7)**

Bits	Default Value	Description
31	–	Lane Alignment Error Detected Alarm 1 = A Received Multiframe Framing character was detected in a mis-aligned location relative to the LMFC. Alignment error is asserted if any lane sees seven consecutive misaligned alignment characters.
30	–	SYSREF LMFC Alarm (Subclass 1 Only) 1 = A SYSREF event was detected, misaligned to current LMFC counter
29	–	RX Buffer Overflow Alarm 1 = RX Lane Alignment Buffer Overflow has occurred
28:24	–	Reserved (Read 00000)
23:21	–	Link Error Status, Lane 7
20:18	–	Link Error Status, Lane 6
17:15	–	Link Error Status, Lane 5
14:12	–	Link Error Status, Lane 4
11:9	–	Link Error Status, Lane 3
8:6	–	Link Error Status, Lane 2
5:3	–	Link Error Status, Lane 1; format as per lane 0
2:0	–	Link Error Status, Lane 0 bit 2: 1 = Unexpected K-character(s) received bit 1: 1 = Disparity Error(s) received bit 0: 1 = Not in Table Error(s) received Each bit indicates that 1 or more errors of that type have been received in Lane 0 because the register was last read. All status bits are auto-cleared on read of the register.

Notes:

1. The status bits can only be set when the core has completed initial lane synchronization, errors during synchronization and comma alignment are ignored. This register is cleared on read or when the core is reset. The contents might be cleared if the watchdog timer is enabled.

Table 2-23: **Octets per Frame**

Bits	Default Value	Description
31:8	–	Reserved
7: 0	0x01	Octets per Frame (F) Parameter range 1–256; Program register with required value minus 1 (for example, for F = 4, 0x03 should be programmed)

Table 2-24: **Frames per Multiframe**

Bits	Default Value	Description
31:5	–	Reserved
4:0	0x1F	Frames per Multiframe (K) Parameter range 1–32; Program register with required value minus 1 (for example, for K = 16, 0x0F should be programmed)

Table 2-25: **Lanes in Use**

Bits	Default Value	Description
31:12	–	Reserved
11:0	Varies depending on Number of Lanes which the core was generated for	Lanes in Use: Allows the number of active lanes to be set for test purposes. Each bit corresponds to a single lane, when set to “1” lane is active. Lanes 0 to X are active if bits 0 to X are set to 1. For example, for three active lanes (lanes 0 to 2 active), 0x07 is programmed. But if you wanted lane 2 and 0 active, program 0x05. ***Note:*** You cannot activate any lane > X which is set during customization.

Table 2-26: **Subclass Mode**

Bits	Default Value	Description
31:3	–	Reserved
2:0	01	Subclass Mode 11: Reserved 10: Subclass 2 01: Subclass 1 00: Subclass 0

Table 2-27: **RX Buffer Delay (RX Only)**

Bits	Default Value	Description
31:13	–	Reserved
12:0	0	RX Buffer Delay RX Buffer Delay can be programmed, in conjunction with the RX Buffer Adjust values read from the lanes, to minimize the overall RX Latency. See Minimum Deterministic Latency Support. An indication of the maximum allowable reduction of the latency is output on the rx_buffer_adjust register. This provides an indication of the difference between the write and read pointers of the receiver elastic buffer in each lane. The number of octets output in each 10-bit value give an indication of the buffer fill level in each lane. The lowest number given can be programmed to the rx_buffer_delay register to reduce the overall latency by that number of octets.

Table 2-28: **Error Reporting (RX Only)**

Bits	Default Value	Description
31:9	–	Reserved
8	0	Disable Error Reporting Using SYNC Interface 1 = Error reporting using SYNC interface Disabled 0 = Error reporting using SYNC interface Enabled
7:1	0	Reserved
0	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

Table 2-29: **Sync Status**

Bits	Default Value	Description
31:17	–	Reserved
16	0	SYSREF Captured (Subclass 1 Only) 1 = A SYSREF event has been captured 0 = No SYSREF event has been captured
15:1	–	Reserved
0	0	SYNC Status 1 = Link SYNC achieved 0 = Link SYNC not achieved

Table 2-30: **Link Error Status (Lanes 8 to 11)**

Bits	Default Value	Description
31:12	–	Reserved
11:9	–	Link Error Status, Lane 11

Table 2-30: **Link Error Status (Lanes 8 to 11) *(Cont'd)***

Bits	Default Value	Description
8:6	–	Link Error Status, Lane 10
5:3	–	Link Error Status, Lane 9; format as per lane 8
2:0	–	Link Error Status, Lane 9 bit 2: 1 = Unexpected K-character(s) received bit 1: 1 = Disparity Error(s) received bit 0: 1 = Not in Table Error(s) received Each bit indicates that 1 or more errors of that type have been received in Lane 0 because the register was last read. All status bits are auto-cleared on read of the register.

Notes:

1. The status bits can only be set when the core has completed initial lane synchronization, errors during synchronization and comma alignment are ignored. This register is cleared on read or when the core is reset. The contents might be cleared if the watchdog timer is enabled.

Table 2-31: **Lane 0 ID**

Bits	Default Value	Description
31:5	–	Reserved
4:0	0	ID of lane 0. Value can be anywhere between 0 and 31.

Table 2-32: **Lane 1 ID**

Bits	Default Value	Description
31:5	–	Reserved
4:0	1	ID of lane 1. Value can be anywhere between 0 and 31.

Notes:

1. Registers for remaining Lane IDs are similar to the one above except for the default value. This increases by 1 for each lane.

Table 2-33: **ILA Config Data 0**

Bits	Default Value	Description(1)
31:11	–	Reserved
10:8	–	JESDV (JESD204 version)
7:3	–	Reserved
2:0	–	SUBCLASS

Notes:

1. RX only: captured configuration data from the ILA sequence (per lane);
   TX: Not applicable, the values transmitted in the ILA sequence are generated automatically by the core based on the configuration.

Table 2-34: **ILA Config Data 1**

Bits	Default Value	Description[1]
31:8	–	Reserved
7:0	–	F (Octets per Frame)

Notes:

1. RX only: captured configuration data from the ILA sequence (per lane);
   TX: Not applicable, the values transmitted in the ILA sequence are generated automatically by the core based on the configuration.

Table 2-35: **ILA Config Data 2**

Bits	Default Value	Description[1]
31:5	–	Reserved
4:0	–	K (Frames per Multiframe)

Notes:

1. RX only: captured Configuration data from the ILA sequence (per lane);
   TX: Not Applicable, the values transmitted in the ILA sequence are generated automatically by the core based on the configuration.

Table 2-36: **ILA Config Data 3**

Bits	Default Value	Description[1]
31:29	–	Reserved
28:24	–	L (Lanes per Link) [RX only, not writeable for TX]
23:21	–	Reserved
20:16	0x0	LID (Lane ID) [RX only, not writeable for TX]
15:12	–	Reserved
11:8	0x0	BID (Bank ID)
7:0	0x00	DID (Device ID)

Notes:

1. RX: captured configuration data from the ILA sequence (per lane);
   TX: Sets the values to be transmitted in the ILA sequence for all lanes. LID and L values cannot be programmed, they are generated automatically by the core based on the configuration.

Table 2-37: **ILA Config Data 4**

Bits	Default Value	Description[1]
31:26	–	Reserved
25:24	00	CS (Control bits per Sample)
23:21	–	Reserved
20:16	00000	N' (Totals bits per Sample)
15:13	–	Reserved

Table 2-37: **ILA Config Data 4 *(Cont'd)***

Bits	Default Value	Description[1]
12:8	00000	N (Convertor Resolution)
7:0	0x00	M (Convertors per Device)

Notes:
1. RX: captured configuration data from the ILA sequence (per lane);
   TX: Sets the values to be transmitted in the ILA sequence for all lanes.

Table 2-38: **ILA Config Data 5**

Bits	Default Value	Description[1]
31:29	–	Reserved
28:24	00000	CF (Control Words per Frame)
23:17	–	Reserved
16	0	HD (High Density format)
15:13	–	Reserved
12:8	00000	S (Samples per Convertor per Frame)
7:1	–	Reserved
0	–	SCR (Scrambling Enable) [RX only, not writeable for TX]

Notes:
1. RX: captured configuration data from the ILA sequence (per lane);
   TX: Sets the values to be transmitted in the ILA sequence for all lanes. SCR value cannot be programmed, it is generated automatically by the core based on the configuration.

Table 2-39: **ILA Config Data 6**

Bits	Default Value	Description[1]
31:24	–	Reserved
23:16	0x00	FCHK (Checksum) [RX only, not writeable for TX]
15:8	0x00	RES2 (Reserved Field 2)
7:0	0x00	RES1 (Reserved Field 1)

Notes:
1. RX: captured configuration data from the ILA sequence (per lane);
   TX: Sets the values to be transmitted in the ILA sequence for all lanes. FCHK value cannot be programmed, it is calculated automatically by the core for each lane.

Table 2-40: **ILA Config Data 7**

Bits	Default Value	Description
31:17	–	Reserved
16	0	ADJDIR (Adjust Direction) [Subclass 2 Only]

Table 2-40: **ILA Config Data 7 *(Cont'd)***

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

Table 2-41: **Test Mode Error Count**

Bits	Default Value	Description
31:0	–	Test Mode Error Count Count of Errors received in Datalink 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.

Table 2-42: **Link Error Count**

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 using the control bit in the Error Reporting register.

Table 2-43: **Test Mode ILA Count**

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.

Table 2-44: **Test Mode Multiframe Count**

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.

Table 2-45: **Buffer Adjust**

Bits	Default Value	Description
31:10	–	Reserved
9:0	–	RX Buffer Adjust Indicates the RX Buffer fill level (per lane). This can be used to minimize overall latency. See Minimum Deterministic Latency Support.

Designing with the Core

This chapter provides a general description of how to use the JESD204 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 JESD204 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 JESD204 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 Chapter 5, Example Design for information about using and customizing the example designs for the JESD204 core.

Know the Degree of Difficulty

JESD204 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 JESD204 implementations require careful attention to 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 system performance in an FPGA design, keep all inputs and outputs registered between your application and the core. This means that all inputs and outputs from your application should come from or connect to a flip-flop. While registering signals might 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 Timing 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 provided currently use Vivado synthesis as the synthesis tool for the Verilog example design that is delivered with the core. Other synthesis tools can be used.

Make Only Allowed Modifications

The JESD204 core is not user-modifiable. Any modifications can have adverse effects on system timing and protocol compliance. Supported user configurations of the JESD204 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 Chapter 4, Design Flow Steps.

Recommended Design Experience

Although the JESD204 core is a fully-verified solution, the challenge associated with implementing a complete design varies depending on the configuration and functionality of the application. For best results, previous experience 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.

Core Overview and Getting Started

This section provides an overview of the core, its deliveries and delivery options, together with system-level decisions which must be taken account of in generating and using the core.

Serial Line Rate and Clocking

The JESD204B specification (JEDEC® Serial Interface for Data Converters (JESD204B) available from www.jedec.org) does not define specific serial line rates for any JESD204B link, but a valid range of line rates from 312.5 Mb/s to 12.5 Gb/s. The JESD204B core supports line rates from 1 Gb/s to 12.5 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 Convertor 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

The JESD204 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 6.25 Gbs, the core clock frequency is 156.25 MHz. The AXI-streaming RX and TX data interfaces operate at this core clock frequency.

Reference Clock

The GTP/GTX/GTH/GTY serial transceivers 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 or source 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/SYNC for Subclass 1 or 2 deterministic latency. See Clocking for more information.

AXI4-Lite Interface Clock

The core is configured and monitored through an AXI4-Lite processor interface. The clock for this interface is a separate clock from either core clock or reference clock. There are no dependencies between this clock and either core clock or reference clock.

Selecting the Operating Line Rate and Reference Clock

The serial line rate and a valid reference clock frequency can be selected when generating the core in the Vivado IDE, and customized transceiver wrapper files are delivered.

Core Delivery – Shared Logic Example Design

The JESD204B core is delivered through the Vivado IDE, either through the Vivado IP catalog, or through the Vivado IP integrator block design tool. The core customization Vivado IDE presents the options available in core generation. See Chapter 4, Design Flow Steps for more information.

As well as allowing the selection of the main core parameters, the Vivado IDE allows the selection of how the core and some of its supporting logic is delivered; this is using the Shared Logic Tab in the Vivado IDE. The default selection is **Include Shared Logic in Example Design** where the core is delivered without shared logic. In this case, the JESD204 PHY core is not delivered directly with the core.

To access the JESD204 PHY core and other supporting logic, it is necessary to open the IP example design (using the right-click menu). This opens a separate Vivado project which instantiates the IP within the example design. The example design instantiates the core support layer, which includes the JESD204 PHY core and other supporting logic such as clock modules. The example design and the support layer delivered in the example design project are intended to provide a useful example and starting point for you own design.

If **Include Shared Logic in Core** is selected, the core is delivered complete with a support layer including the JESD204 PHY core and other supporting logic such as clock buffers. For simple core implementations where there is no requirement for transceiver sharing, the line rate is configurable in the Vivado IDE. To perform this, select the **Include Shared Logic in Core** option.

For more complex core implementations, for example, transceiver sharing, PLL and/or clock sharing, select **Include Shared Logic in Example Design** and use the IP example design as a starting point for your own design.

Transceiver Sharing

Because JESD204B is defined as a unidirectional protocol, the JESD204 core is customized and delivered as either a transmitter or as a receiver. The Xilinx GTP/GTX/GTH/GTY transceivers are duplex, each transceiver supporting both transmit and receive directions. In a system implementation using both TX and RX JESD204 links, you might want to use both the TX and RX directions of a transceiver (or group of transceivers), and connect to both TX and RX JESD204 cores. Such transceiver sharing can be supported by the JESD204 core, within the limitations and constraints imposed by line rates and clock sources (see Sharing Transceivers between Transmit and Receive).

Subclass Mode

The core supports operation in the three JESD204B Subclass modes. This is controlled by a register setting. By default the core operates in Subclass 1 mode. The core pinout 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 backwards compatible with JESD204A, with extended line rate range support up to 12.5 Gb/s. There is no support for Deterministic Latency. SYSREF is not required.

Subclass 1

Subclass 1 supports deterministic latency through the use of a common SYSREF signal. The SYSREF signal is generated external to the core, and is distributed to all devices in a system. SYSREF is permitted by the JESD204B standard to be either a "one-shot", periodic, or gapped periodic. The 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 (see Clocking).

Subclass 2

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 (see Clocking).

Programming the Core

Run time operation of the JESD204 core is configured through an AXI4-Lite register interface. See Register Space for details of the register map.

For correct operation and bring-up of a JESD204 link, it is important that the major framing and link operation parameters match at both ends of the link:

- Octets per Frame
- Frames per Multiframe
- Scrambling On/Off
- Subclass mode
- SYSREF handling (for Subclass 1 mode)

These parameters are dictated by the configurations available in the ADC/DAC convertor device to which the core is interfacing.

For TX cores, in addition to these 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 RX cores, the configuration data received in the ILA sequence is captured, for each lane, and can be examined using the register interface.

Clocking

This section describes the options available for clocking the JESD204 core and the transceiver(s). The following clocks are used in the JESD204 core:

- **Device Clock –** The JESD204B specification defines a device clock which is distributed to each device in the system. The device clocks to different devices such as DAC/ADC converters and FPGAs can run at different rates to suit the individual devices, but must all be related in frequency and be generated from a common source. See the *JESD204B specification* [Ref 1], paragraphs 4.7 and 4.8, for definitions.
- **Byte Clock** – The frame and multiframe periods in each device are derived from the device clock. The frame/multiframe periods must match for the transmitting and receiving devices. From the frame period the octet (byte) clock rate can be directly inferred (F octets per frame).
- **Serial Line Rate** – The serial line rate for all lanes is common, and runs at 10 times the byte clock rate on each lane.
- **Core Clock –** The JESD204 core operates using a 32-bit (4-byte) datapath. The device clock for the core logic therefore runs at one quarter of the byte clock rate (1/40th of the serial line rate). For the JESD204 core, this is referred to as the core clock.
- **Reference Clock** – The GTP/GTX/GTH/GTY serial transceivers require a stable, low-jitter reference clock which has a device and speed grade dependent range. In some circumstances, the same source clock can supply both the reference clock and core clock.

Depending on the **Shared Logic** selection made when generating the core, a clocking module is delivered as part of the core (**Shared Logic in Core**) or as part of the Core Support Layer delivered with the Example Design project (**Shared Logic in Example Design**). This clocking module contains example clocking resources relevant to the selected device and the core customization choices.

In either case the FPGA level input clocks are:

- `refclk` (p/n) – transceiver reference clock. This is always present.
- `glblclk` (p/n) – core clock. This is an optional input from the Vivado IDE. However, it is always present if `refclk` is greater than 165 MHz or less than 65 MHz, or if `refclk` is not equal to the core clock.

RECOMMENDED: *For the greatest flexibility in implementing system-level clocking, Xilinx recommends that the FPGA pinout and system-level clocking resources are designed to provide both of these clocks to the FPGA.*

AXI4-Lite Configuration Interface Clock (axi_clk) – This is asynchronous to any other clock and can be driven by the processor subsystem.

Supporting Subclass 1 and 2 Deterministic Latency

Supporting Subclass 1 or 2 deterministic latency requires careful consideration of the clocking scheme chosen, to ensure that SYSREF (for Subclass 1) or SYNC (for Subclass 2 TX cores) can be reliably captured by the core clock.

For Subclass 0, where there are no such constraints, the clocking requirements are simplified, and alternative clocking arrangements can be used.

Number of lanes per link

The maximum number of lanes per link is 12. For interfaces more than 12 lanes, Xilinx recommends creating multiple cores with a maximum of 8 lanes each.

For a transmit interface, the lane ID of each lane of a transmit core can be independently programmed using the Lane ID registers (see Table 2-31).

For a receive interface, the lane ID of each lane can be read from the LID field of the ILA Config Data 3 register for each lane (see Table 2-36).

This programmable Lane ID feature can also be utilized to share a single JESD core with multiple synchronous converters. For example, eight one lane converters may be connected to a single JESD204 core. If this is a transmitter core, the lanes can be programmed to be lane 0.

This has the advantage of potentially aligning the data from all the converter as well as reducing the logic overhead that would be generated if eight individual cores were used.

Note: The system must designed to support this functionality.

Basic Generic Clocking Schemes

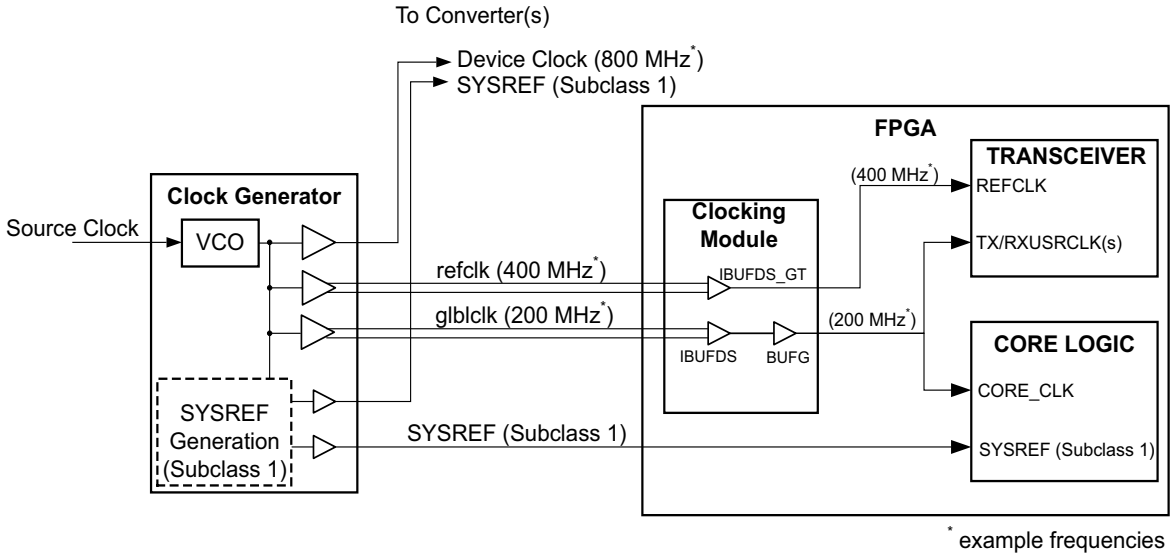


Figure 3-1: **Clock Configuration using Separate refclk and Core Clock**

Figure 3-1 shows the most generic and flexible clocking scheme, where separate `refclk` and `glblclk` inputs are used to provide the transceiver reference clock and the core clock, respectively. With this configuration, the reference clock and core clock are physically separate clocks and can be run at independent 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 (1/40th of the serial line rate).

This configuration must be used where the reference clock frequency is out of the range that can be used for the core clock and support the reliable capture of SYSREF/SYNC. This frequency range is governed by the particular device family and speed grade (see Tables 3-1 to Table 3-5).

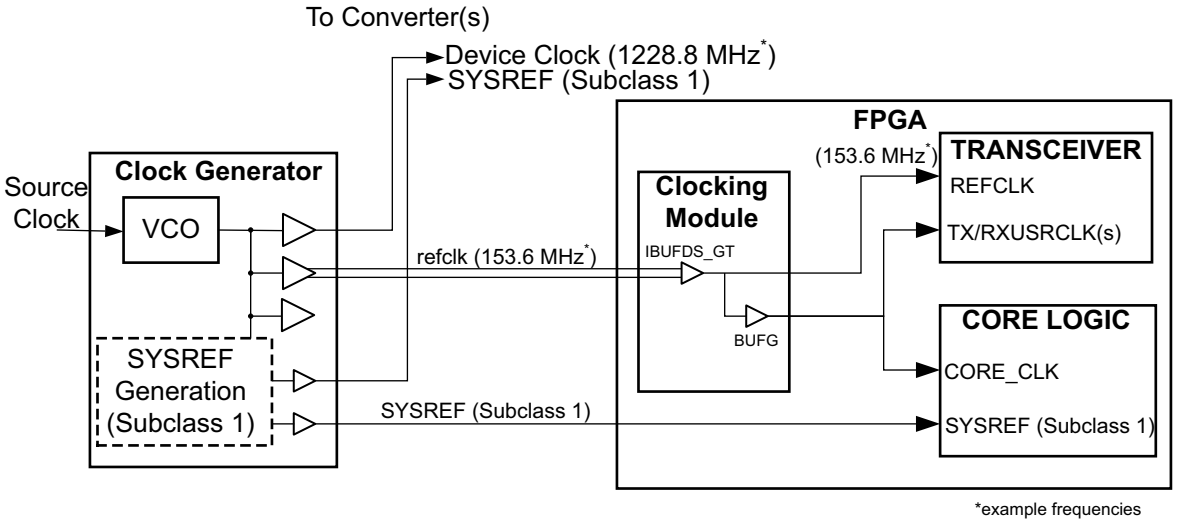


Figure 3-2: **Clock Configuration using refclk as Core Clock**

Figure 3-2 shows a clocking scheme that can be used when the reference clock frequency is in the range that can be used for the core clock and support the reliable capture of SYSREF/SYNC. This frequency range is governed by the particular device family and speed grade (see Tables 3-1 to Table 3-5).

Supported Clock Frequency Ranges

The following tables list the relevant parameters for each of the supported device families.

Table 3-1: **Frequency Ranges for Artix-7 Devices**

Device Family: Artix-7 and Zynq-7000 (GTP Transceiver)			
Parameter	**Speed Grade**		
	-3	**-2**	**-1(2)**
Line Rate MAX (Gb/s)(1)	6.6	6.6	3.75
Line Rate MIN (Gb/s)(1)	0.5	0.5	0.5
refclk MAX (MHz)(1)	660	660	660
refclk MIN (MHz)(1)	60	60	60
MAX frequency for refclk as core clock (MHz)(3)	165	165	93.75
MIN frequency for refclk as core clock (MHz)(3)	60	60	80

Notes:

1. Further limitations might apply: see the *Artix-7 FPGAs Data Sheet: DC and Switching Characteristics* (DS181) [Ref 9] and the *Zynq-7000 All Programmable SoC (Z-7010, Z-7015, and Z-7020): DC and AC Switching Characteristics* (DS187) [Ref 10] for full specifications.
2. For -1 speed grade devices, the line rate is limited to 3.75 Gb/s and the GTP PLL VCO lower limit prevents selecting a `refclk` lower than 80 MHz.
3. If these conditions are met, the clocking in Figure 3-2 can be used. Otherwise, the clocking in Figure 3-1 must be used.

Table 3-2: **Frequency Ranges for Kintex-7 Devices**

Device Family: Kintex-7, Virtex-7 and Zynq-7000 (GTXE2 Transceiver)			
Parameter	**Speed Grade**		
	-3	**-2**	**-1**
Line Rate MAX (Gb/s)(1)	12.5	10.3	8.0
Line Rate MIN (Gb/s)(1)	0.5	0.5	0.5
refclk MAX (MHz)(1)	700	670	670
refclk MIN (MHz)(1)	60	60	60
MAX frequency for refclk as core clock (MHz)(3)	165	165	165
MIN frequency for refclk as core clock (MHz)(3)	80	80	80

Notes:

1. Further limitations might apply—see the *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* (DS182) [Ref 11] and *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics* (DS183) [Ref 12] and *Zynq-7000 All Programmable SoC (Z-7030, Z-7045, and Z-7100): DC and AC Switching Characteristics* (DS191) [Ref 13] for full specifications.
2. The GTXE2 can operate with `refclk` as low as 60 MHz, but the operating band of the CPLL VCO prevents `refclk` and core clock sharing at frequencies less than 80 MHz (CPLL) or 75 MHz (QPLL).
3. If these conditions are met, the clocking in Figure 3-2 can be used. Otherwise, the clocking in Figure 3-1 must be used.

Table 3-3: **Frequency Ranges for Virtex-7 Devices Using GTHE2**

Device Family: Virtex-7 (GTHE2 Transceiver)			
Parameter	**Speed Grade**		
	-3	**-2**	**-1**
Line Rate MAX (Gb/s)(1)	12.5	11.3	8.5
Line Rate MIN (Gb/s)(1)	0.5	0.5	0.5
refclk MAX (MHz)(1)	820	820	820
refclk MIN (MHz)(1)	60	60	60
MAX frequency for refclk as core clock (MHz)(2)	165	165	165
MIN frequency for refclk as core clock (MHz)(2)	80	80	80

Notes:

1. Further limitations might apply —see the *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics* (DS183) [Ref 11] for full specifications.
2. If these conditions are met, the clocking in Figure 3-2 can be used. Otherwise, the clocking in Figure 3-1 must be used.

Table 3-4: **Frequency Ranges for Kintex UltraScale Architecture-Based Devices**

Device Family: Kintex UltraScale (GTHE3 Transceiver)			
Parameter	**Speed Grade**		
	-3	**-2**	**-1**
Line Rate MAX (Gb/s)(1)	12.5	12.5	12.5
Line Rate MIN (Gb/s)(1)	0.5	0.5	0.5
refclk MAX (MHz)(1)	820	820	820
refclk MIN (MHz)(1)	60	60	60
MAX frequency for refclk as core clock (MHz)(2)	165	165	165
MIN frequency for refclk as core clock (MHz)(2)	80	80	80

Notes:

1. Further limitations might apply—see the *Kintex UltraScale Architecture Data Sheet: DC and Switching Characteristics* (DS892) [Ref 14] for full specifications.
2. If these conditions are met, the clocking in Figure 3-2 can be used. Otherwise, the clocking in Figure 3-1 must be used.

Table 3-5: **Frequency Ranges for Virtex UltraScale Architecture-Based Devices**

Device Family: Virtex UltraScale (GTHE3 Transceiver)			
Parameter	**Speed Grade**		
	-3	**-2**	**-1**
Line Rate MAX (Gb/s)	TBD	TBD	TBD
Line Rate MIN (Gb/s)	TBD	TBD	TBD
refclk MAX (MHz)	TBD	TBD	TBD
refclk MIN (MHz)	TBD	TBD	TBD
MAX frequency for refclk as core clock (MHz)(1)	TBD	TBD	TBD
MIN frequency for refclk as core clock (MHz)(1)	TBD	TBD	TBD

Notes:

1. If these conditions are met, the clocking in Figure 3-2 can be used. Otherwise, the clocking in Figure 3-1 must be used.

Detailed Clocking

For Kintex-7, Virtex-7, Kintex UltraScale architecture and Virtex UltraScale architecture family devices (GTXE2, GTHE2, GTHE3 and GTYE3 transceivers), the clocking schemes described in Basic Generic Clocking Schemes can be used. Depending on the **Shared Logic** selection at core generation time, example clocking source code modules are delivered as part of the core, or as part of the core example design project.

For Artix-7 family devices, the GTP transceiver uses a 16-bit internal datapath. This imposes an extra requirement for TXUSRCLK/RXUSRCLK (supplied to the GTP transceiver) which runs at twice the core clock rate. TXUSRCLK2/RXUSRCLK2 runs at the same rate as the core clock. To support this, an MMCM is normally used as shown in Figure 3-3. For Artix-7 devices, the generated example clocking source code modules delivered include the relevant MMCM configuration.

For most Artix-7 devices, the line rate and reference clock limitations are such that the configuration shown in Figure 3-2 (`refclk` as core clock) external clocking configuration can be used. Figure 3-3 shows an example of the required clocking for this configuration.

For -1 speed grade devices only, and line rates below 3.2 Gb/s, it is necessary to provide a separate device clock because `refclk` is limited to 80 MHz. Figure 3-4 shows an example of the required clocking for this configuration.

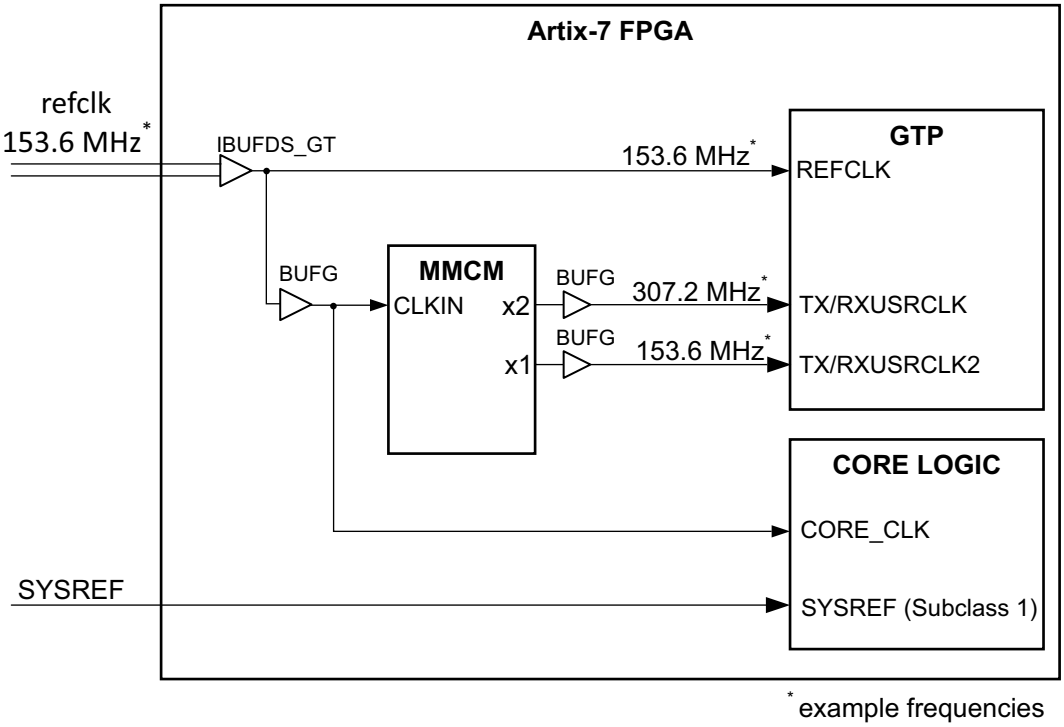


Figure 3-3: **Artix-7 Clock Configuration Using refclk as Core Clock**

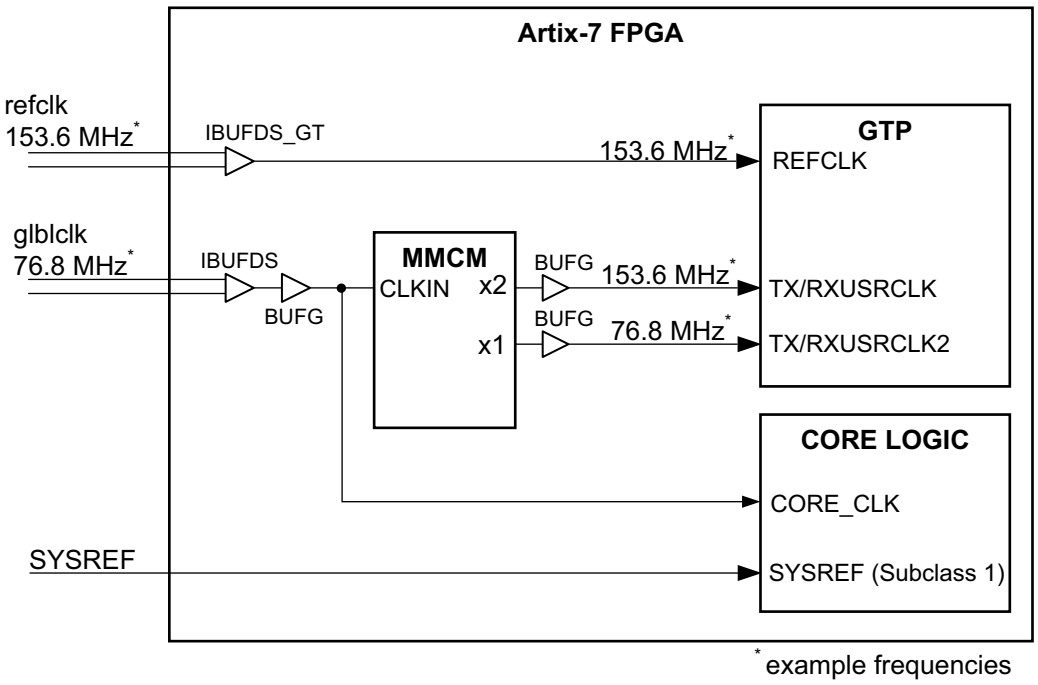


Figure 3-4: **Artix-7 Clock Configuration Using Separate refclk and Core Clock**

Clocking for Subclass 0 Mode

If Subclass 0 operation is required, the timing limitations imposed to support deterministic latency are removed, and simplified clocking arrangements can be used which require only a reference clock input.

In these cases it is possible to use the TXOUTCLK from the transceiver to generate the core clock where `glblclk` becomes optional and can be left open. Figure 3-5 shows the generic case for use with most devices. However, Artix-7 devices require an additional MMCM as shown in Figure 3-6 for generation of TXUSRCLK/RXUSRCLK at twice the core clock. The core clock is the same as TXUSRCLK2/RXUSRCLK2.

Note: No example of this clocking configuration is generated for any configuration of the JESD204 core. The IP is configured to use the Subclass 1 clocking scheme as seen above. If this clocking scheme is required, it is suggested that the standard clocking module generated with the core (or example design), is used as a starting point to create the required clocking configuration.

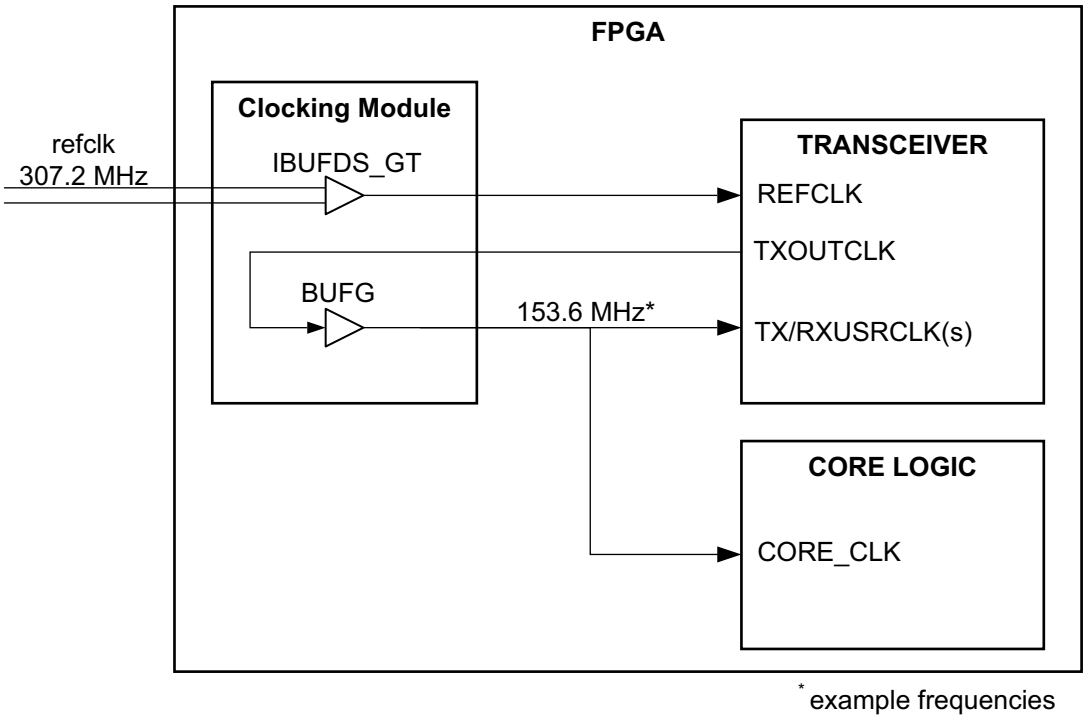


Figure 3-5: **Subclass 0 Clocking**

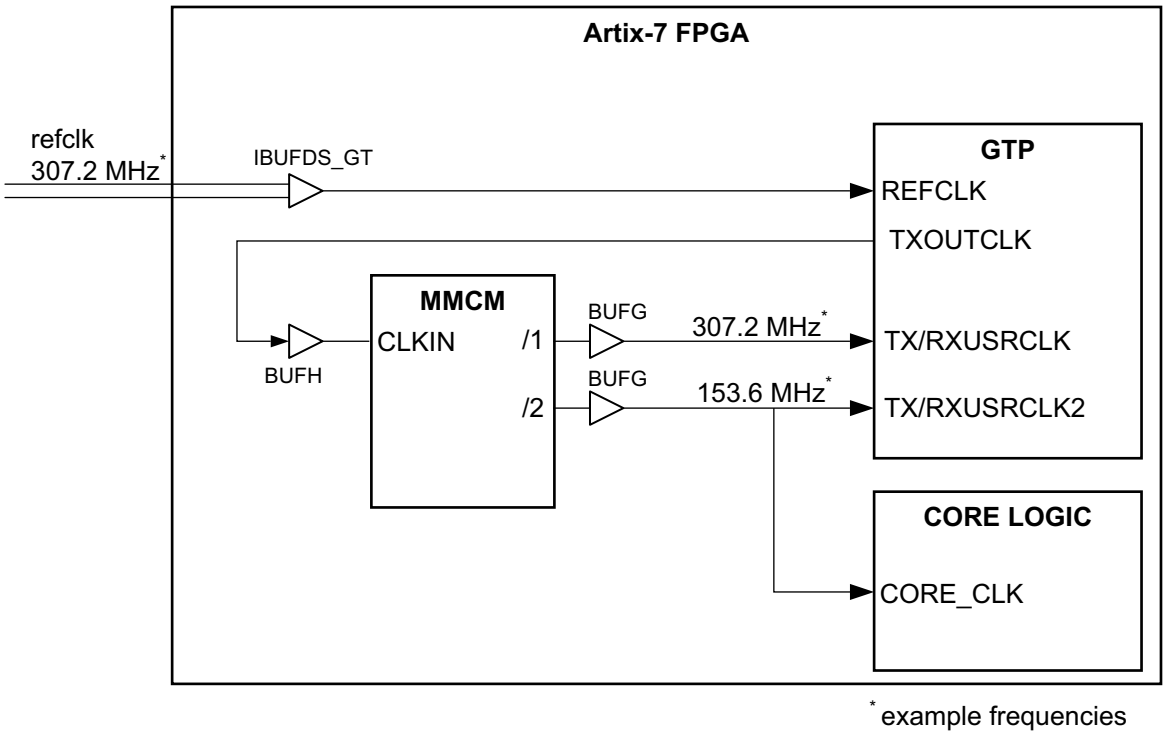


Figure 3-6: **Subclass 0 Clocking for Artix-7 Devices**

Resets

The JESD204 core uses the following resets.

System Reset

An asynchronous reset is provided to reset the complete system (core logic and transceiver). On a transmit core this signal is `tx_reset` and on a receiver the reset signal is `rx_reset`. The AXI4-Lite interface and configuration registers are unaffected by these reset signals. A separate reset signal (`s_axi_aresetn`) is provided for the AXI4-Lite interface which resets the configuration registers to the default values.

Software Reset

A register is provided through the AXI4-Lite interface which triggers a data path reset sequence for the transmit or receive logic data path under software control. The configuration registers are unaffected by this operation.

Note: The use of this reset does not reset the PLLs.

Watchdog Timer Reset

For devices using the GTXE2 transceiver, the core logic includes a "Watchdog" timer which automatically provides a reset if SYNC is not achieved, or is lost for a prolonged period. The Watchdog timer is implemented as a 20-bit counter which increments if SYNC remains deasserted. The counter is clocked by the AXI4-Lite clock, so the timeout is directly related to this clock frequency (for example, a 100 MHz clock provides a timeout of approximately 10 ms).

A Watchdog timeout reset acts in the same manner as the Software Reset.

The Watchdog timer can be disabled, if necessary, using a register access (see Reset register), for example, during system testing where a link can be held out of SYNC for test purposes. In normal system operation, Xilinx recommends that the Watchdog remain enabled.

AXI4-Stream Reset

When either a system reset or software reset is performed the `rx_aresetn` or `tx_aresetn` outputs are asserted Low until the reset cycle is completed.

Interfacing to the AXI4-Stream Data Interface

The AXI4-Stream data transmit and receive interfaces are used to pass JESD204 formatted data to and from the core. The AXI data input and output by the core contains 4-bytes per clock cycle per lane with the least significant byte position in each 32-bit block holding the first byte received from the ADC or transmitted to the DAC. Figure 3-7 shows an example of how the JESD204 data is mapped on to the AXI4-Stream interface.

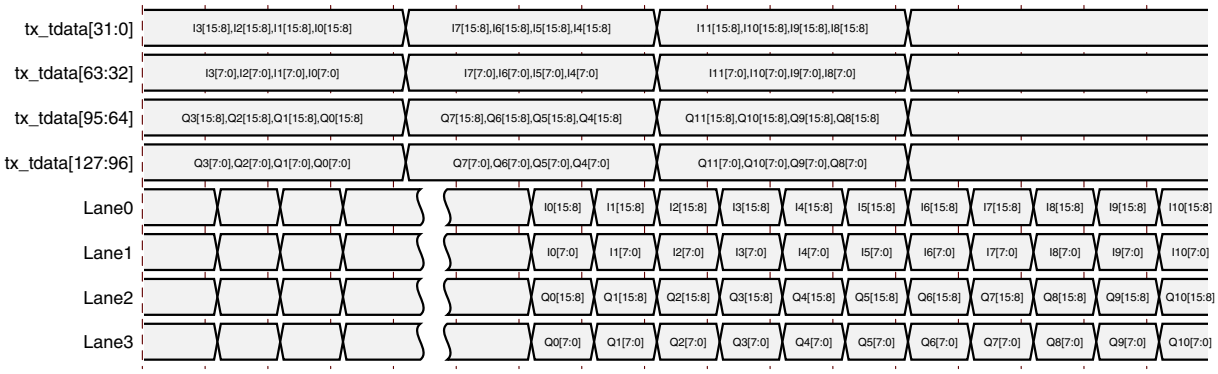


Figure 3-7: **Four Lane DAC, Frame Size (F) = 1, 16 Bit I and Q with I Sample in Lanes 0 & 1 and Q Sample in Lanes 3 & 4**

Transport Layer

The following examples show transport layer functions for some common ADC converters.

TI ADS42JB69 ADC 2 Converter, 4 Lane Mode

- 250 MHz sample clock, 4 serial lanes at 2.5 Gb/s
- Four 16 bit samples per convertor per 62.5 MHz core clock cycle

```
adc0_sample0 = {rx_tdata[7:0],   rx_tdata[39:32]};   // ADC0 sample N
adc0_sample1 = {rx_tdata[15:8],  rx_tdata[47:40]};   // ADC0 sample N+1
adc0_sample2 = {rx_tdata[23:16], rx_tdata[55:48]};   // ADC0 sample N+2
adc0_sample3 = {rx_tdata[31:24], rx_tdata[63:56]};   // ADC0 sample N+3
adc1_sample0 = {rx_tdata[71:64], rx_tdata[103:96]};  // ADC0 sample N
adc1_sample1 = {rx_tdata[79:72], rx_tdata[111:104]}; // ADC0 sample N+1
adc1_sample2 = {rx_tdata[87:80], rx_tdata[119:112]}; // ADC0 sample N+2
adc1_sample3 = {rx_tdata[95:88], rx_tdata[127:120]}; // ADC0 sample N+3
```

ADI AD9250 ADC 2 Convertor, 2 Lane Mode

- 250 MHz sample clock, 2 serial lanes at 5 Gb/s
- Two 14 bit samples per convertor per 125 MHz core clock cycle

```
adc0_sample0 = {rx_tdata[7:0],   rx_tdata[15:10]}; // ADC0 sample N
```

```
adc0_sample1 = {rx_tdata[23:16], rx_tdata[31:26]}; // ADC0 sample N+1
adc1_sample0 = {rx_tdata[39:32], rx_tdata[47:42]}; // ADC1 sample N
adc1_sample1 = {rx_tdata[55:48], rx_tdata[63:58]}; // ADC1 sample N+1
```

IDT ADC1443D ADC 2 Convertor, 2 Lane Mode

- 200 MHz sample clock, 2 serial lanes at 4 Gb/s
- Two 14 bit samples per convertor per 100 MHz core clock cycle

```
adc0_sample0 = {rx_tdata[7:0],   rx_tdata[15:10]}; // ADC0 sample N
adc0_sample1 = {rx_tdata[23:16], rx_tdata[31:26]}; // ADC0 sample N+1
adc1_sample0 = {rx_tdata[39:32], rx_tdata[47:42]}; // ADC1 sample N
adc1_sample1 = {rx_tdata[55:48], rx_tdata[63:58]}; // ADC1 sample N+1
```

AXI4-Lite Management Interface

For AXI information, go to www.xilinx.com/ipcenter/axi4.htm.

The JESD204 core is configured and monitored using control and status registers accessible through an AXI4-Lite interface.The AXI4-Lite address map is detailed in Register Space.

Subclass 1 Operation

Also to supporting lane operation at up to 12.5 Gb/s, devices of Subclass 1 support deterministic latency using the SYSREF interface. Figure 3-8 shows the timing relationships between the signals on a JESD204B link. Subclass 1 is recommended for designs requiring deterministic latency.

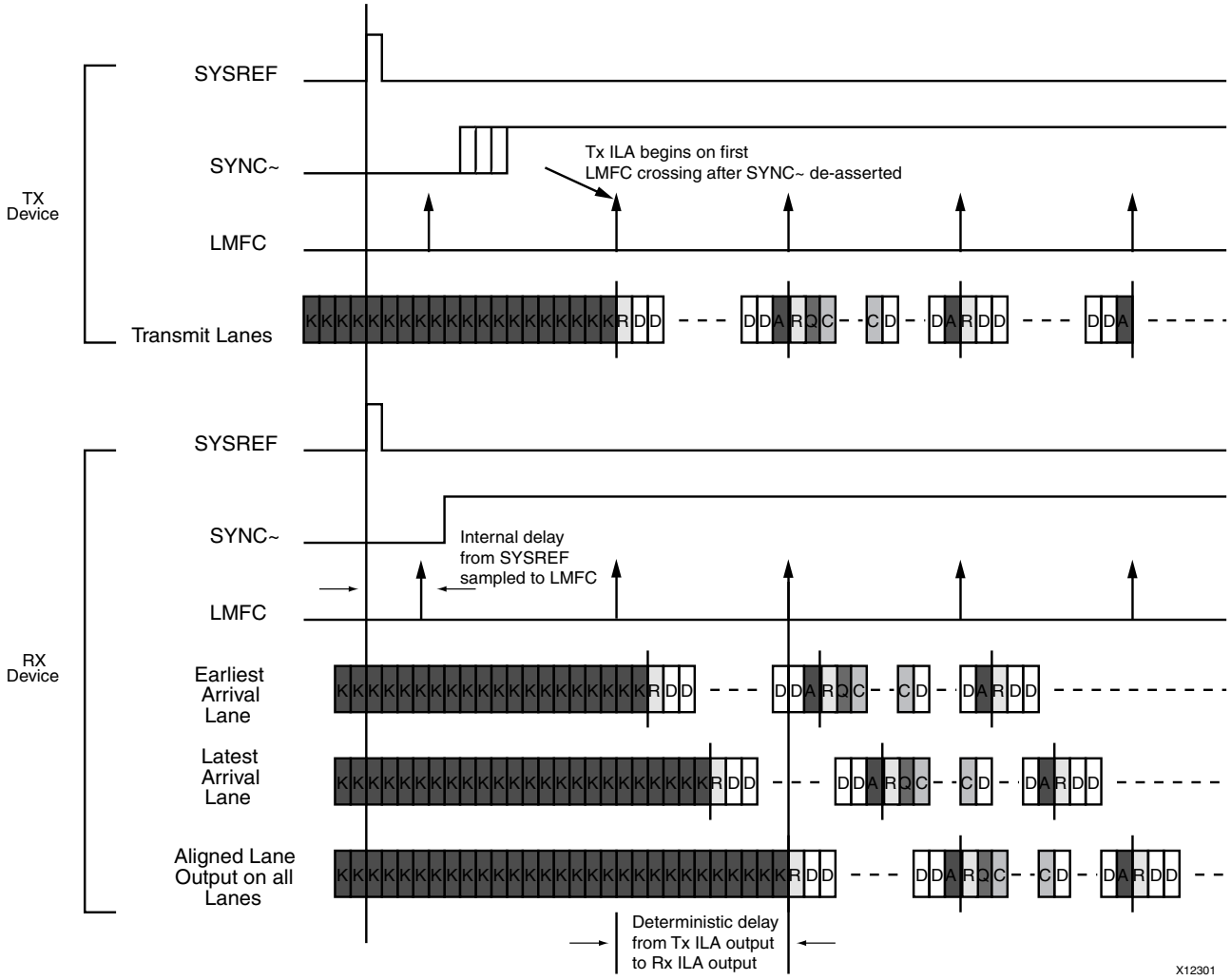


Figure 3-8: **Deterministic Latency across the JESD204B Link**

The SYSREF signal is distributed to all devices in a system. On assertion, the JESD204 receiver aligns its internal Local Multiframe Clock (LMFC) to the incoming SYSREF signal and deasserts SYNC~. The LMFC counts through the octets in a multiframe. A LMFC crossing occurs at the end of each multiframe. The device clock for the FPGA is the FPGA logic clock (tx_aclk or rx_aclk).

When the transmit device detects the SYNC~ signal going High, it waits until the next LMFC crossing and starts transmitting data. If ILA generation is supported, then this is the ILA sequence; otherwise it is normal frame data. When the receiver detects that all lanes have valid data, it buffers the data until the next LMFC crossing. Now the buffers are released and the data sent to the client interface. In this manner the latency between the data output and the SYSREF signal can be accurately ascertained and the end-to-end latency of the system is deterministic and repeatable.

The length of the multiframe is intended by the JESD204B specification to be larger than the longest possible delay across any lane in the JESD204B link. This might not be achievable in practice in real systems due to short multiframe sizes dictated by A/D and D/A devices.

SYSREF Timing

When JESD204B 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. The SYSREF period must also be a multiple of 4-byte clock periods because the core uses a 4-byte internal datapath. In the example provided with the core, by default the SYSREF signal is captured on the falling edge of the core clock to allow the rising edge of the clock and SYSREF to be aligned at the edge of the device; this is the case typically when a divided device clock is used for SYSREF. See Clocking for details.

SYSREF Handling

The correct handling of SYSREF is critical in Subclass 1 operation. The JESD204B specification allows SYSREF to be generated in any of the following ways:

- Periodic
- One-Shot
- "Gapped" Periodic

The JESD204 core provides several options for how SYSREF is handled for Subclass 1 operation to support maximum flexibility.

Note: The JESD204 core operates using a 32-bit (4-byte) datapath. If a periodic or gapped periodic SYSREF is used in the system, the SYSREF period must be an integer multiple of the multiframe period. But it must also be a multiple of 4-byte clocks if the multiframe period is not itself a multiple of 4-byte clocks.

SYSREF Sampling Clock Edge

The core can be configured to sample SYSREF on either the rising or falling edge of the core clock. This is a core generation time option on the customization in the Vivado IDE. This selection controls the clock edge used for the first flip-flop (normally an I/O flip-flop) in the SYSREF handling logic. Only this first flip-flop is affected by this setting; all other processing is done on the rising edge of the core clock.

The default setting is for SYSREF to be sampled on the negative edge of core clock. This is based on the assumption that an externally generated SYSREF is synchronous to the clock supplied to the core (depending on clocking options, refclk or glblclk—see Clocking), and thus arrives edge-aligned at the FPGA I/O. This is generally the case when SYSREF is generated as a divided version of the source clock. Negative edge clocking in this case provides the best timing margin.

To provide flexibility in the design, however, the core can be generated to use a positive edge sampling of SYSREF. This should be used where the system-level timing is suitable for such an approach.

Capturing SYSREF using the Falling Edge of the Clock

Capturing SYSREF using the negative edge of the clock allows SYSREF to be a divided clock with rising edges aligned to the device clock. In this case, a low skew clock divider chip is used to generate both the device clock and SYSREF.

Constraints can be applied in the example design to align the clock and SYSREF edges within 1 ns, for example, as follows.

TIP: *Higher line rates reduce the value.*

```
# Set +/- 0.5ns between the rising edges of the clock and sysref
set_input_delay -clock jesd204_0_refclk -max 0.5 [get_ports *x_sysref]
set_input_delay -clock jesd204_0_refclk -min -0.5 [get_ports *x_sysref]
```

The timing of SYSREF can be confirmed using the report_datasheet command in the Vivado Design Suite. An example is shown in Figure 3-9. This example uses the following settings:

- Negative edge clocking of SYSREF using REFCLK as the core clock.
- REFCLK period is 6.4 ns (6.25 Gb/s line rate).
- SYSREF input delay = ±0.5 ns.

The report_datasheet command gives a setup of 1.2 ns and hold of 1.8 ns for the SYSREF input. Figure 3-9 is a timing diagram based on these values.

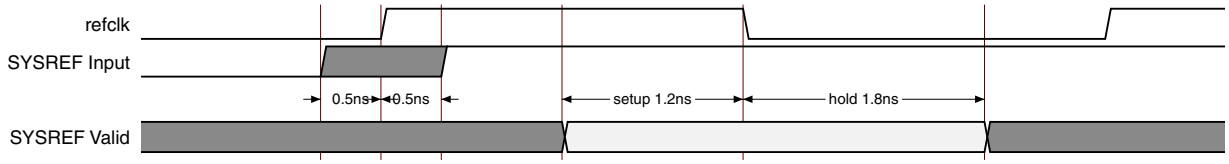


Figure 3-9: **SYSREF Timing using Negative Edge Sampling**

Capturing SYSREF using the Rising Edge of the Clock

Capturing SYSREF using the positive edge is an option in the core. When this option is selected, no constraint is applied to the SYSREF input. The timing of the SYSREF input can be checked using the report_datasheet command in the Vivado Design Suite. An example is shown in Figure 3-10. This example uses the following settings:

- Positive edge clocking of SYSREF using REFCLK as the core clock.
- REFCLK period is 6.4 ns (6.25 Gb/s line rate).

The report_datasheet command gives a setup of 4.6 ns and hold of -1.5 ns for the SYSREF input. Figure 3-10 is a timing diagram based on these values.

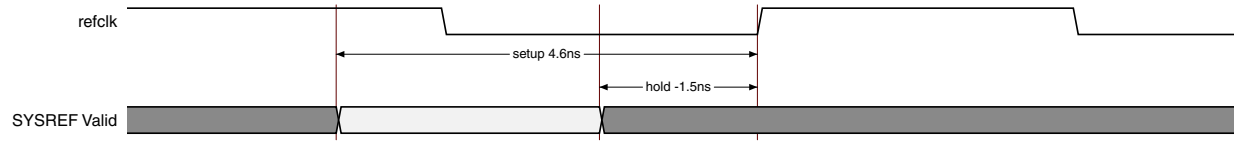


Figure 3-10: **SYSREF Timing using Positive Edge Sampling**

SYSREF Always

The core provides a programmable option allowing the choice of how a periodic SYSREF is used internally. This is selected using the **SYSREF Always** control bit in the SYSREF handling register (See Table 2-19).

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.

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 period.

SYSREF on Initial Link Bring-Up

After a reset, the JESD204 core requires at least one SYSREF event to align the internal LMFC counter, and bring up the link:

- An RX core requires an initial SYSREF event to align the LMFC, and then deasserts SYNC on the next LMFC boundary when code group sync has been achieved. The core does not deassert SYNC until an initial SYSREF event is detected.

- A TX core requires a SYSREF event to align the LMFC. The core begins ILA transmission on an LMFC boundary after SYNC is deasserted. 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

When a link has been initially brought up, if a link re-synchronisation is requested (by the deassertion of SYNC by the receiving device), then the desired core behavior relative to SYSREF can be controlled using the **SYSREF Required on Re-Sync** control bit in the SYSREF Handling register (See Table 2-19).

When **SYSREF Required on Re-Sync** 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).

- An RX core deasserts SYNC on the next LMFC boundary after code group sync.
- A TX core transmits ILA sequence on the next LMFC boundary after SYNC is deasserted.

When **SYSREF Required on Re-Sync** is set to 1, a SYSREF event is required for the link to re-establish SYNC following a re-sync request.

- An RX core waits for a SYSREF event to realign the LMFC counter, and only deasserts SYNC on the next LMFC boundary.
- A TX core waits for a SYSREF event to realign the LMFC counter, and only then begins ILA transmission on an LMFC boundary after SYNC is deasserted.

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 Delay

The deterministic latency mechanism as defined in the JESD204B standard requires that the multiframe size be larger than the maximum possible delay across the link. In practice, this is difficult to achieve, particularly with small frame sizes. However, as long as the multiframe size is greater than the maximum variation in delay across the link, then deterministic latency can be achieved.

A potential issue occurs when the maximum 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. Calculation of the overall system latency (see RX End to End Latency or TX End to End Latency) should be carried out to determine if this is a likely scenario.

In cases where this situation arises, the LMFC boundary in TX or RX devices can be moved relative to each other by adding additional delay to SYSREF in one of the devices. The JESD204 core supports this shifting of the internal LMFC by allowing an additional delay in the internal SYSREF handling logic. This is programmed using the **SYSREF Delay** field in the SYSREF handling register (see Table 2-19) which allows a delay of 0 to 15 core clock cycles to be inserted between the SYSREF event detection and LMFC counter reset. A change in value for SYSREF delay requires a core reset to force the link to realign.

Subclass 2 Operation

The operation of the JESD204B 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 deasserts 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.

In Subclass 2 devices the transmitter and receiver latencies are as listed in the previous Subclass 1 section.

JESD204B Receiver

This section covers implementation and system-level details for a JESD204B receiver core.

Elastic Buffer Implementation

In JESD204B devices, the receiver lane alignment elastic buffer can be implemented in block memory or distributed memory. To use the block RAM implementation the **Use BRAM** check box should be enabled. In addition, you can select the size of the buffer. The **Octets in Multiframe** option should be set to a size that holds a full multiframe of data.

Receive Latency

The latency variation is critical for JESD204B systems. The latency through the IP core is fixed (it can be varied under user control—see Minimum Deterministic Latency Support for details), but the transceiver introduces some variation as the internal receive elastic buffer of the transceiver is included in the datapath. The receive datapath latencies are shown in Table 3-6. The variation in latency is compensated automatically by the core to ensure that the overall end-to-end latency has no variation.

Table 3-6: **Receive Datapath Latencies**

	GTP	GTXE2	GTHE2	GTHE3	GTY
T_{RXLMFC}	32	32	32	32	32
T_{RXIN}	72±4	92±4	92±4	80±4	88
T_{RXOUT}	0	0	0	0	0

Notes:

1. Latency values are given in byte clocks. See RX End to End Latency for detailed use.

RX End to End Latency

Overall latency in a JESD204 system requires consideration of the various sources of fixed and variable latencies across the link.

ADC Timing

The key parameters of the ADC required to calculate end to end latency are:

1. The fixed delay from SYSREF to LMFC (T_{TXLMFC})
2. The fixed delay from analogue input to LMFC (T_{TXIN})
3. The delay from LMFC to JESD204 serial output (T_{TXOUT})

The delay from SYSREF to LMFC and analogue data in to LMFC must be fixed for a Subclass 1 device but the delay from LMFC to JESD204 serial data out can vary because it is compensated for in the receiver during alignment to LMFC. See Figure 3-11.

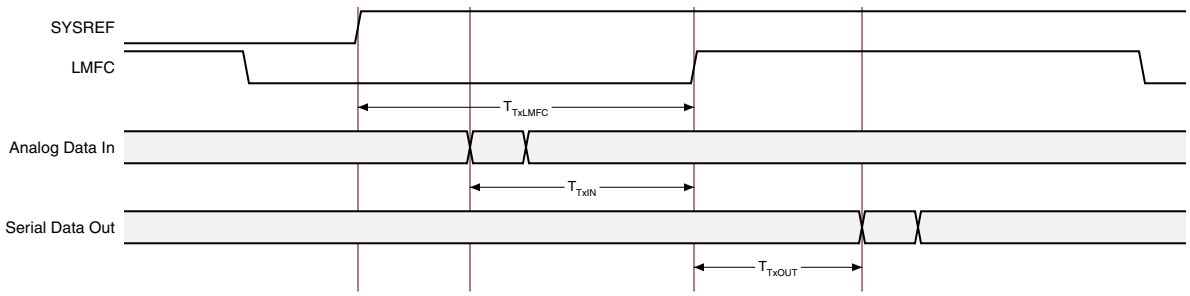


Figure 3-11: **ADC Timing**

Core Timing

The key parameters of the FPGA receiver required to calculate end to end latency are:

1. The fixed delay from SYSREF to LMFC (T_{RXLMFC})
2. The delay from JESD204 input to LMFC (T_{RXIN})
3. The fixed delay from LMFC to AXI Data output (T_{RXOUT})

The delay from SYSREF to LMFC and from LMFC to AXI Data output must be fixed for a Subclass 1 device but the delay from JESD204 serial data in to LMFC can vary because the receiver buffer compensates for variations in end to end latency. In the latency calculations T_{RXOUT} = 0 and all the fixed delays are included in T_{RXLMFC}. See Figure 3-12.

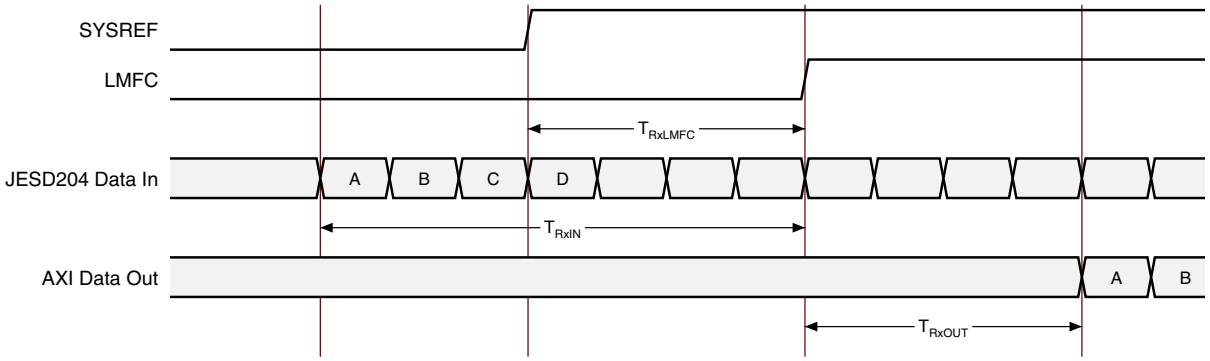


Figure 3-12: **FPGA Receive Timing**

Calculating End to End Latency

The end to end latency is always fixed and made up of a multiple number of LMFC periods plus the fixed TX and RX delays. To calculate the end to end latency, use the following steps, referring to Figure 3-13.

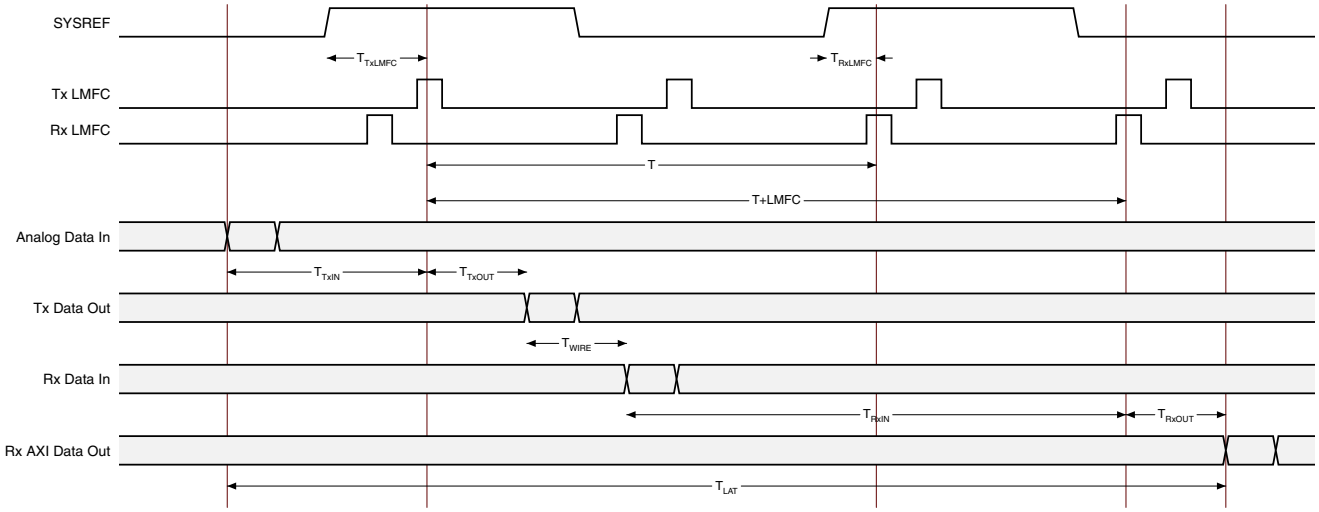


Figure 3-13: **End to End Latency**

Step 1 - Determine how many LMFC periods are required (N)

The delay between LMFC at TX and LMFC at RX (T) is an integer number (N) of LMFC periods adjusted by the internal delays from SYSREF to LMFC of TX and RX:

$$T = N*LMFC - T_{TXLMFC} + T_{RXLMFC}$$

To ensure the overall propagation delay is constant between system restarts, the maximum propagation delay must be less than T plus one LMFC period:

$$(T_{TXOUT} + T_{WIRE(max)} + T_{RXIN(max)}) < T + LMFC$$

The minimum propagation delay must also be greater than T:

$$(T_{TXOUT} + T_{WIRE(min)} + T_{RXIN(min)}) > T$$

Substituting $(N*LMFC - T_{TXLMFC} + T_{RXLMFC})$ for T gives:

$$(T_{TXOUT} + T_{WIRE(max)} + T_{RXIN(max)}) < ((N+1)*LMFC - T_{TXLMFC} + T_{RXLMFC})$$

$$(T_{TXOUT} + T_{WIRE(min)} + T_{RXIN(min)}) > (N*LMFC - T_{TXLMFC} + T_{RXLMFC})$$

It is possible that no valid value for N can be found if the received data arrives close to an RX LMFC boundary and the variation in the link causes it to fall just before or just after the boundary after resetting the system. This would be seen as a jump in latency of exactly one LMFC period between system restarts. If no valid value can be found for N then the SYSREF signals can be delayed relative to one another to move the RX LMFC boundary relative to the TX LMFC boundary. For each cycle of delay added to TX SYSREF add 4 to T_{TXLMFC}. Additional delay can be added to the SYSREF processing in the JESD204 core to accomplish this; see the SYSREF Handling register.

Step 2 - Calculate the end to end latency using N

The data is received after N LMFC periods and before N+1 LMFC periods so the minimum deterministic latency is T plus one LMFC plus the fixed input delay at the FPGA and the fixed output delay at the DAC:

$$T_{LAT} = (T + LMFC) + T_{TXIN}$$

Substituting $(N*LMFC - T_{TXLMFC} + T_{RXLMFC})$ for T gives:

$$T_{LAT} = ((N*LMFC - T_{TXLMFC} + T_{RXLMFC}) + LMFC) + T_{TXIN}$$

$$= (N+1)*LMFC - T_{TXLMFC} + T_{RXLMFC} + T_{TXIN}$$

Example

JESD204 v5.2 using GTX	Assume an ADC with Parameters)	Assume
$T_{RXLMFC} = 32$	$T_{TXIN} = 14$	LMFC = 32
$T_{RXIN} = 92 \pm 4$	$T_{TXLMFC} = 3$	$T_{WIRE} = 0$
$T_{RXOUT} = 0$	$T_{TXOUT} = 6$	

$$(T_{TXOUT(max)} + T_{WIRE(max)} + T_{RXIN(max)}) < ((N+1)*LMFC - T_{TXLMFC} + T_{RXLMFC})$$

$$(6 + 0 + 96) < ((N+1)*32 - 3 + 32)$$

$$102 < 125 \text{ (margin = 23)}$$

$$(T_{TXOUT(min)} + T_{WIRE(min)} + T_{RXIN(min)}) > (N*LMFC - T_{TXLMFC} + T_{RXLMFC})$$

$$(6 + 0 + 88) > (N*32 - 3 + 32)$$

$$94 > 93 \text{ (margin = 1)}$$

N=2 is OK, no need to skew SYSREF

The data is received after 2 LMFCs (N) and less than 3 LMFCs (N+1) so the latency is 3*LMFC plus fixed delays:

$$T_{LAT} = T + T_{TXIN}$$

$$T_{LAT} = (N+1)*LMFC - T_{TXLMFC} + T_{RXLMFC} + T_{TXIN}$$

$$= 96 - 3 + 32 + 14 = 139$$

Note that in this case the margin (how far after the RX LMFC the data arrives) is very small (1-byte periods) and any variation introduced by the ADC or connection between the ADC and FPGA could cause the data to be received early causing the data to be received before the LMFC boundary. There is plenty of margin before the third LMFC so, in this case, it is advisable to delay the TX LMFC by 1 or 2 to increase the margin and therefore the reliability of the system. If delaying the TX LMFC is not possible then the RX LMFC can be delayed by 5 cycles and N reduced by 1 as follows:

T_{RXLMFC} becomes 32 + 5*4 = 52

$(T_{TXOUT(max)} + T_{WIRE(max)} + T_{RXIN(max)}) < ((N+1)*LMFC - T_{TXLMFC} + T_{RXLMFC})$

$(6 + 0 + 96) < ((N+1)*32 - 3 + 52)$

102 < 113 (margin 11 byte periods)

$(T_{TXOUT(min)} + T_{WIRE(min)} + T_{RXIN(min)}) > (N*LMFC - T_{TXLMFC} + T_{RXLMFC})$

$(6 + 0 + 88) > (N*32 - 3 + 52)$

94 > 81 (margin 13 byte periods)

The margin numbers at min delay and max delay are closer in value which means the system is operating with the received multiframe boundary close to the middle of the RX LMFC period; this reduces the chances of the latency changing due to delay variations in the link.

The data is received after one LMFCs (N) and less than two LMFCs (N+1) so the latency is 2*LMFC plus fixed delays:

$T_{LAT} = T + T_{TXIN}$

$T_{LAT} = (N+1)*LMFC - T_{TXLMFC} + T_{RXLMFC} + T_{TXIN}$

$= 64 - 3 + 52 + 14 = 127$

Minimum Deterministic Latency Support

When the JESD204B link is first established in Subclass 1 and Subclass 2 devices, the receiver outputs data as shown in Figure 3-14. Data is output on the LMFC crossing after valid data is detected in all lanes. It is possible to support minimum latency by adjusting the number of octets input on the `rx_buffer_delay` register.

An indication of the maximum allowable reduction of the latency is output on the `rx_buffer_adjust` register. This provides an indication of the difference between the write and read pointers of the receiver elastic buffer in each lane. The number of octets output in each 10-bit value give an indication of the buffer fill level in each lane. The lowest number given can be programmed to the `rx_buffer_delay` register to reduce the overall latency by that number of octets. The `rx_buffer_adjust` value is only 10 bits; the maximum LMFC size should be chosen such that the difference between the maximum expected latency and the LMFC size is less than 1024 octets to prevent `rx_buffer_adjust` overflow.

A full link resynchronization cycle must take place before the modified latency setting is acted upon. A minimum latency example is shown in Figure 3-14.

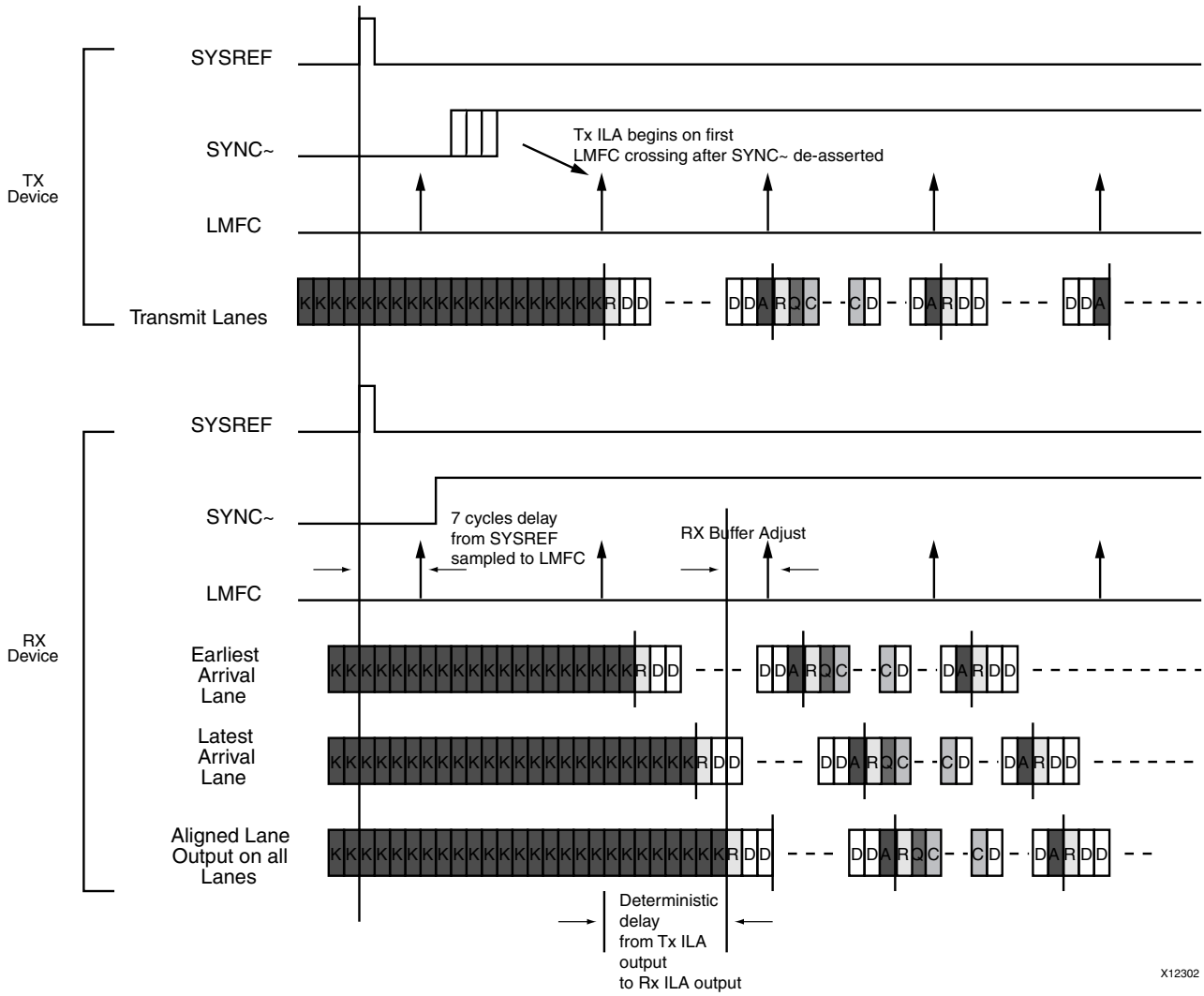


Figure 3-14: **Minimum Deterministic Latency**

Error Signaling Using the SYNC~ Interface

In the JESD204 core the `SYNC~` signal can be deasserted for two frame periods at the end of each multiframe to indicate an error. This behavior is enabled by the deassertion of the `disable_error_reporting` control bit.

The `SYNC~` signal is a 1-bit output; this enables the correct error output for a frame size of down to two octets. The signal is Low when the receiver has detected an error. An error occurs when an idle or an unexpected control character is received during normal frame transmission.

To give a frame-accurate SYNC~ output when the frame size is one octet the SYNC~ signal should be asserted for only half a cycle of the device clock. This is not implemented in the core as it would require a dedicated clock (either twice the device clock speed or an inverted device clock). If accurate timing of SYNC~ assertion for error reporting is required logic

must be added externally to generate a half cycle pulse on the external SYNC~ pin if the core SYNC~ signal is asserted for a single cycle and F = 0.

Link Re-initialization Using the SYNC~ Interface

To signal a link re-initialization request, the receiver deasserts SYNC~ for a period of five frame periods and nine octets. When this occurs, the transmitter should fall back out of data transmission mode and renegotiate the link as described in the previous device subclass sections.

JESD204B Transmitter

This section covers implementation and system-level details for a JESD204B transmitter core.

Transmit Latency

The latency variation is critical for JESD204B systems. The latency through the IP core is fixed but the transceiver introduces some variation as the internal transmit elastic buffer of the transceiver is included in the datapath.

The transmit datapath latencies are shown in Table 3-7.

Table 3-7: **Transmit Datapath Latencies**

	GTP	GTXE2	GTHE2	GTHE3	GTYE3
T_{TXLMFC}	36	36	36	36	36
T_{TXOUT}	41±2	56±2	56±2	52±2	56±2
T_{TXIN}	0	0	0	0	0

Notes:
1. Latency values are given in byte clocks. See TX End to End Latency for detailed use.

TX End to End Latency

Overall latency in a JESD204 system requires consideration of the various sources of fixed and variable latencies across the link.

Core Timing

The key parameters of the FPGA transmitter required to calculate end to end latency are:

1. The fixed delay from SYSREF to LMFC (T_{TXLMFC})
2. The fixed delay from AXI input to LMFC (T_{TXIN})

3. The delay from LMFC to JESD204 serial output (T_{TXOUT})

The delay from SYSREF to LMFC and AXI data in to LMFC must be fixed for a Subclass 1 device but the delay from LMFC to JESD204 serial data out can vary because it is compensated for in the receiver during alignment to LMFC. See Figure 3-15.

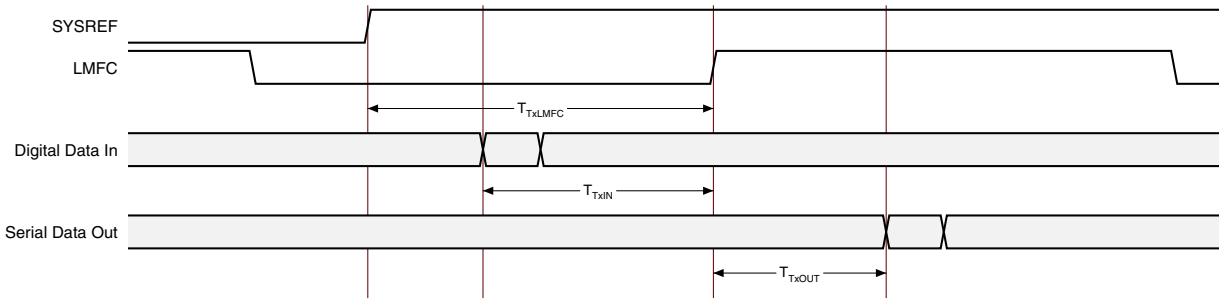


Figure 3-15: **FPGA TX Timing**

DAC Timing

The key parameters of the DAC receiver required to calculate end to end latency are:

1. The fixed delay from SYSREF to LMFC (T_{RXLMFC})
2. The delay from JESD204 input to LMFC (T_{RXIN})
3. The fixed delay from LMFC to analog output (T_{RXOUT})

The delay from SYSREF to LMFC and from LMFC to output must be fixed for a Subclass 1 device but the delay from JESD204 serial data in to LMFC can vary as the receiver buffer compensates for variations in end to end latency. See Figure 3-16.

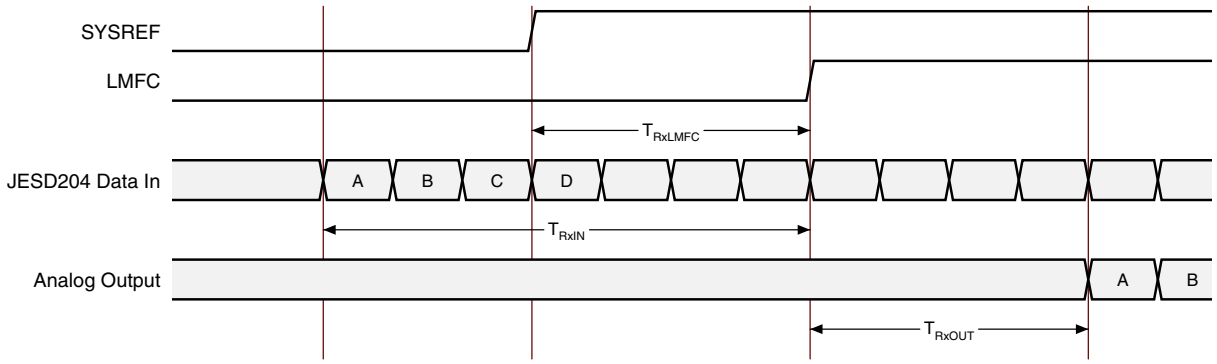


Figure 3-16: **DAC RX Timing**

Calculating End to End Latency

The end to end latency is always fixed and made up of a multiple number of LMFC periods plus the fixed TX and RX delays. To calculate the end to end latency, use the following steps, referring to Figure 3-17.

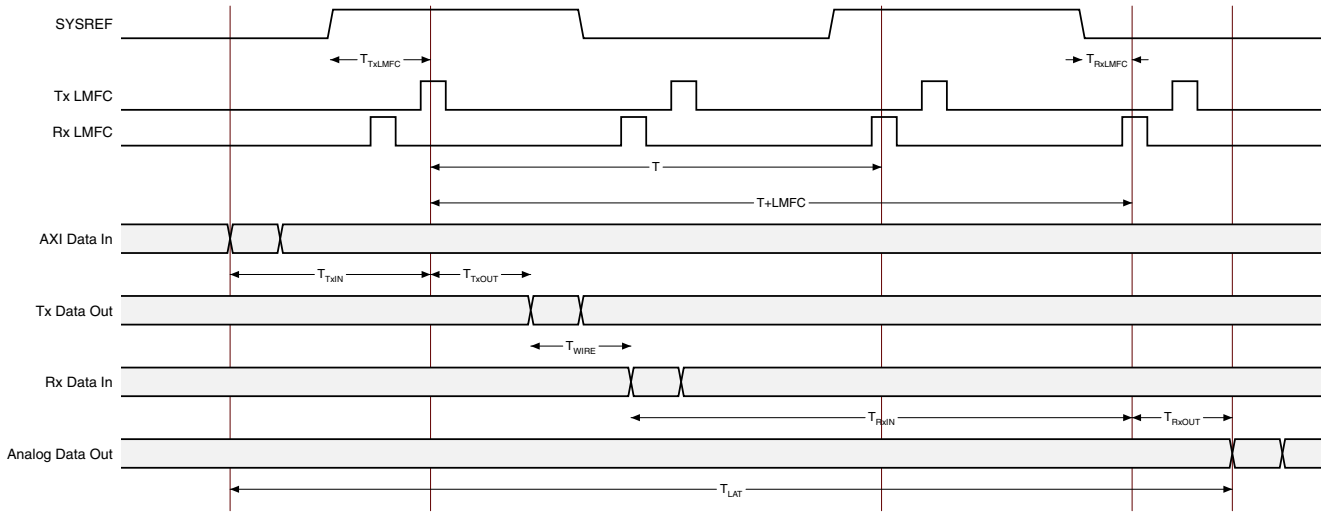


Figure 3-17: **End to End Latency**

Step 1 – Determine how many LMFC periods are required (N)

The delay between LMFC at TX and LMFC at RX (T) is an integer number (N) of LMFC periods adjusted by the internal delays from `SYSREF` to LMFC of TX and RX:

$$T = N*LMFC - T_{TXLMFC} + T_{RXLMFC}$$

To ensure the overall propagation delay is constant between system restarts, the maximum propagation delay must be less than T plus one LMFC period:

$$(T_{WIRE(max)} + T_{RXIN(max)} + T_{TXOUT(max)}) < T + LMFC$$

The minimum propagation delay must also be > T:

$$(T_{WIRE(min)} + T_{RXIN(min)} + T_{TXOUT(min)}) > T$$

Substituting $(N*LMFC - T_{TXLMFC} + T_{RXLMFC})$ for T gives:

$$(T_{WIRE(max)} + T_{RXIN(max)} + T_{TXOUT(max)}) < ((N+1)*LMFC - T_{TXLMFC} + T_{RXLMFC})$$

$$(T_{WIRE(min)} + T_{RXIN(min)} + T_{TXOUT(min)}) > (N*LMFC - T_{TXLMFC} + T_{RXLMFC})$$

It is possible that no valid value for N can be found if the received data arrives close to an RX LMFC boundary and the variation in the link causes it to fall just before or just after the boundary after resetting the system. This would be seen as a jump in latency of exactly 1

LMFC period between system restarts. If no valid value can be found for N then the SYSREF signals can be delayed relative to one another to move the RX LMFC boundary relative to the TX LMFC boundary. For each cycle of delay added to TX SYSREF add 4 to T_{TXLMFC}. Additional delay can be added to the SYSREF processing in the JESD204 core to accomplish this; see the SYSREF Handling register.

Step 2 – Calculate the end to end latency using N

The data is received after N LMFC periods and before N+1 LMFC periods so the min deterministic latency is T plus one LMFC plus the fixed input delay at the FPGA and the fixed output delay at the DAC:

$$T_{LAT} = (T + LMFC) + T_{TXIN} + T_{RXOUT}$$

Substituting $(N*LMFC - T_{TXLMFC} + T_{RXLMFC})$ for T gives:

$$TLAT = ((N*LMFC - T_{TXLMFC} + T_{RXLMFC}) + LMFC) + T_{RXOUT} + T_{TXIN}$$

$$= (N+1)*LMFC - T_{TXLMFC} + T_{RXLMFC} + T_{TXIN} + T_{RXOUT}$$

Example

JESD204 v5.2 using GTX	Assume a DAC with Parameters	Assume
T_{TXLMFC} = 36 bytes	T_{RXIN} = 50±2 bytes	T_{WIRE} = 0
T_{TXOUT} = 56±2 bytes	T_{RXLMFC} = 4 bytes	LMFC = 32 bytes
T_{TXIN} = 0	T_{RXOUT} = 20 bytes	

$$(T_{TXOUT(max)} + T_{WIRE(max)} + T_{RXIN(max)}) < ((N+1)*LMFC - T_{TXLMFC} + T_{RXLMFC})$$

$$(TT_{XOUT(min)} + T_{WIRE(min)} + T_{RXIN(min)}) > (N*LMFC - T_{TXLMFC} + T_{RXLMFC})$$

$$(58 + 0 + 52) < ((N+1)*32 - 36 + 4) \text{ and } (54 + 0 + 48) > (N*32 - 36 + 4)$$

Try N = 4

110 < 128 and 102 > 96

N = 4 is OK, no need to skew SYSREF

The data is received after 4 LMFCs (N) and less than 5 LMFCs (N+1) so the latency is 5*LMFC plus fixed delays:

$$T_{LAT} = T + T_{TXIN}$$

$$T_{LAT} = (N+1)*LMFC - T_{TXLMFC} + T_{RXLMFC} + T_{TXIN}$$

$$= 160 - 36 + 4 + 20 = 156 \text{ byte clock periods}$$

Transmitter Phase Adjustment for Subclass 2

The core provides ports to enable the alignment of the frame clock and LMFC internal to the FPGA to those of the DAC receiver. The required clock and LMFC phase adjustments are communicated to the DAC using the `tx_cfg_adjcnt`, `tx_cfg_adjdir` and `tx_cfg_phadj` registers as detailed in Table 2-14.

In Subclass 2 devices, the DAC receiver deasserts the `SYNC~` signal on its internal LMFC boundary. At the FPGA transmitter it is the responsibility of the client logic to detect the deassertion of the `SYNC~` signal and to compare its timing to that of the transmitter LMFC. If adjustment of the DAC LMFC phase is required, the client inputs the required adjustment step count to the `tx_cfg_adjcnt` register. The direction of the phase adjustment is input on `tx_cfg_adjdir` and an adjustment request is signaled by the assertion of the `tx_cfg_phadj` register.

The values on the phase adjustment ports are embedded in bytes 1 and 2 of the ILA sequence that is sent to the DAC during link initialization. On the reception of the ILA, the DAC adjusts its LMFC phase by the step count value and sends back an error report with the new LMFC phase information. This process can be repeated until the LMFC at the DAC and the FPGA are aligned.

Link Test Modes

The JESD204 transmitter core supports the following Datalink Layer test modes as described in JESD204B 5.3.3.8.2.

Continuous D21.5 Characters

This mode transmits a continuous sequence of /D21.5/ characters. This pattern is created within the GTX/GTH transceivers by setting PRBSSEL to 110. To enable this test mode, set **Test Mode Select** in the Configuration register to 011 or if the **Additional transceiver control and status ports** box is checked on the JESD204 GUI, set port, txprbssel[2:0], to 110 (see Table 2-12 for the debug port description).

Modified RPAT

This mode transmits a continuous sequence of a modified random pattern. This pattern is generated by the RPAT block and must be selected in the Vivado IDE. The block generates a 32-bit output containing four RPAT sequences which is then sent to each transmitter. To transmit this pattern, set **Test Mode Select** in the Configuration register to 101.

JSPAT

This mode transmits a continuous sequence of a scrambled jitter pattern. This pattern is generated by the JSPAT block and must be selected in the Vivado IDE. The block generates a 32-bit output containing four JSPAT sequences which is then sent to each transmitter. To transmit this pattern, set **Test Mode Select** in the Configuration register to 111.

On the receiver side, an Error Counting block is used to calculate the number of errors in the received data. The counter has been modified to calculate all the errors in the received data. To enable this block, signal `rx_cfg_link_test_enable` must be set High. A block is created for each transceiver and uses the `rx_disperr` and `rxnotintable` signals to calculate the number of errors. The block produces a 32-bit output which is then written to a register.

Sharing Transceivers between Transmit and Receive

RECOMMENDED: *Xilinx recommends using Vivado IP integrator to simplify the process of sharing transceivers using the JESD204_PHY IP.*

Sharing Transceivers in IP Integrator

The JESD204_PHY core (see the *JESD204 PHY LogiCORE IP Product Guide (PG198)* [Ref 4]) provides a simple way to instantiate transceivers for use with the JESD204 cores. Any number of JESD204_PHY cores can be connected to any number of JESD204 cores to cater for any combination of ADCs and DACs using different line rates and lane counts.

An example of a two lane TX and two lane RX sharing two transceivers is shown in Figure 3-18. The transmitter is configured for 3 Gb/s and the receiver configures for 6 Gb/

s. The `refclk` is 150 MHz and shared between TX and RX. Separate core clocks are provided for TX and RX to support subclass 1 (see Figure 3-1 in Clocking, page 44).

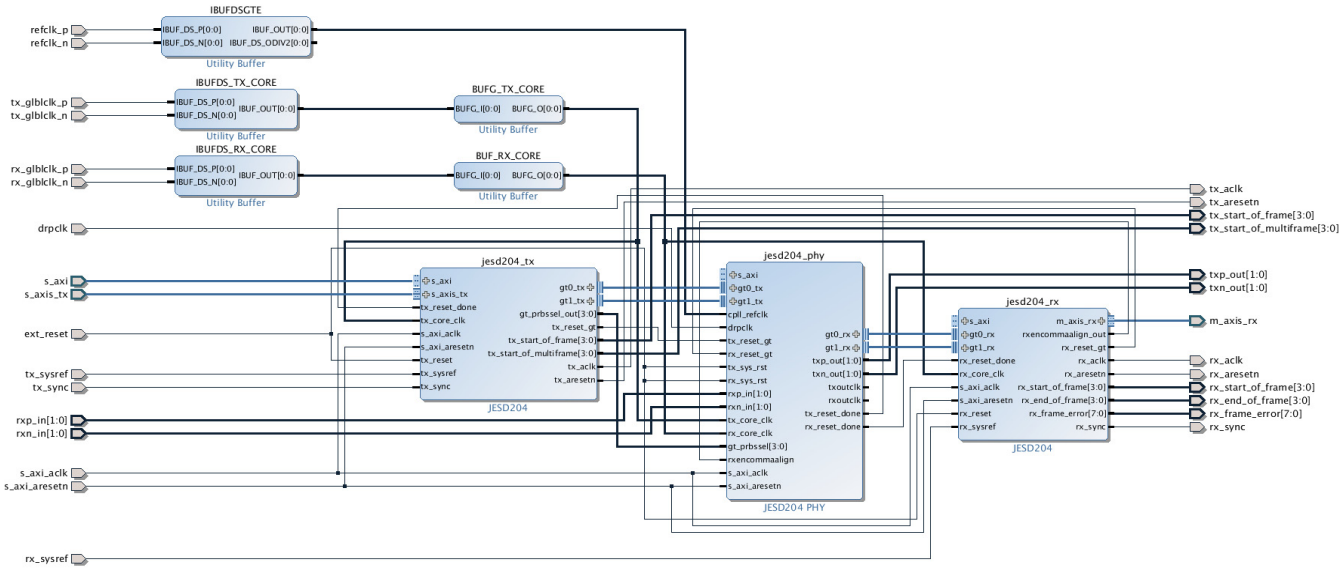


Figure 3-18: **Two Lane TX and Two Lane RX Sharing Transceivers**

The connections between the JESD204 cores and the JESD204_PHY can be seen more clearly in Figure 3-19, which has the external I/O removed.

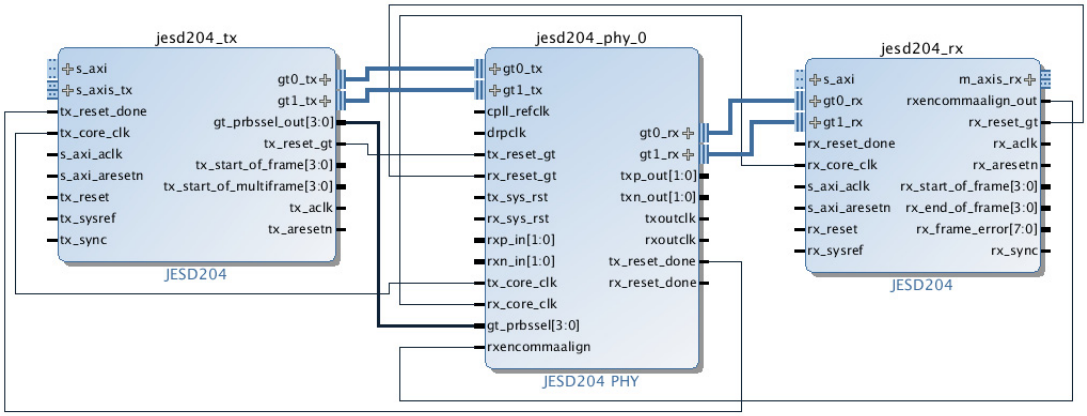


Figure 3-19: **Connections Required between JESD204 and JESD204_PHY**

When sharing transceivers between cores with different lane counts there should be multiple JESD204_PHY cores instantiated to ensure the JESD204 links can be reset independently. For an example, see Figure 3-20. In this example, three independent, two

lane transmitters are connected to three JESD204_PHYs and a single six lane receiver shares the transceivers.

All signals are connected directly between the JESD204 and JESD204_PHY as in the previous examples, except the `rx_reset_done` from each JESD204_PHY must be AND'd together to ensure all six transceivers have completed reset before the six lane receiver is enabled.

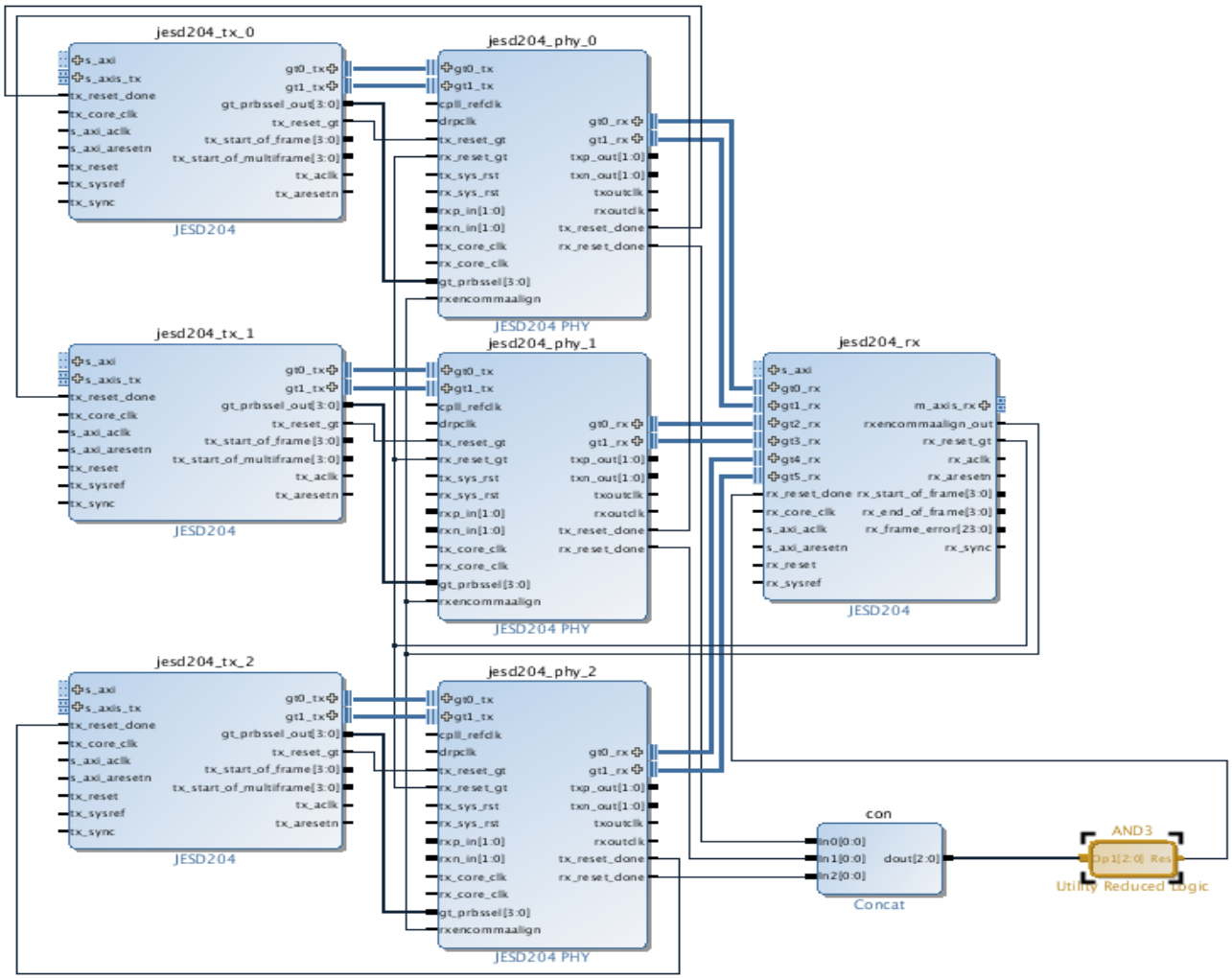


Figure 3-20: **Multiple JESD204_PHYs Instantiated for Independent Transmitters**

Sharing a QPLL

To share the QPLL between two JESD204_PHYs, one JESD204_PHY should be configured with shared logic in core (`jesd204_phy_1` in Figure 3-21) and the other with shared logic in the example design (`jesd204_phy_0` in Figure 3-21). The QPLL outputs are available on the core containing the shared logic and can be connected to the QPLL inputs on the core without shared logic as shown in Figure 3-21.

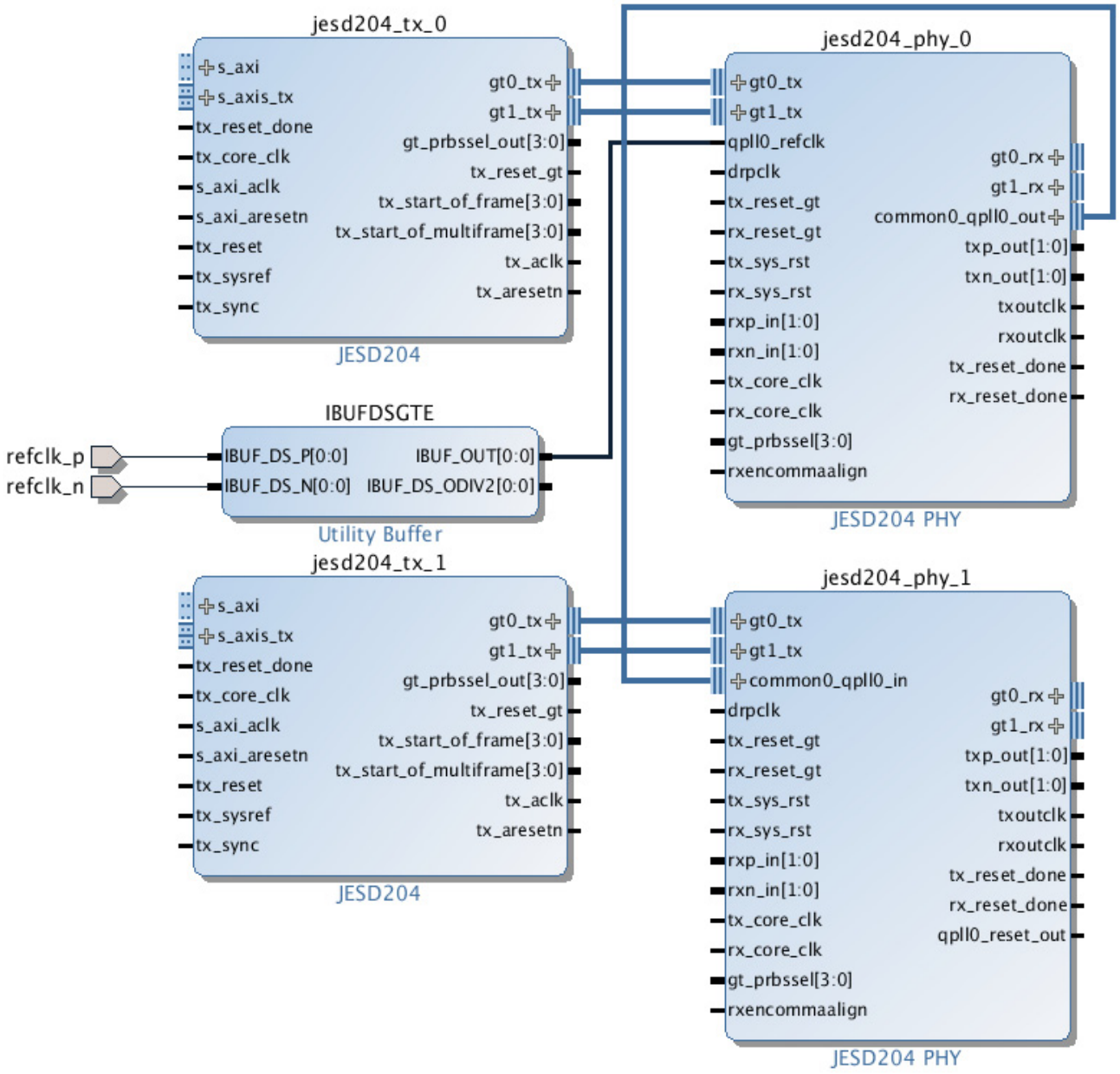


Figure 3-21: **Sharing QPLL between Two JESD204_PHYs**

Sharing Transceivers in RTL Designs

When sharing transceivers in an RTL design, the same JESD204 and JESD204_PHY combinations shown in the previous section should be used. The JESD204 IP and JESD204_PHY should be connected together using Verilog or VHDL in exactly the same way as shown in Figure 3-18 to Figure 3-20. Clock buffers for the `refclk` (IBUFDS_GTEn) and (IBUFDS and BUFG) should be added manually. The `<component_name>_support.v` and `<component_name>_clocking.v` files provided with the IP Example Design can be used as a reference for adding clock buffers to the design. When instantiating the JESD204 IP, the `.xci` should be directly used and not the support layer as in previous releases of the IP.

Powering down unused GT channels

For designs that are generated with shared logic in the core and *without* Transceiver Debug enabled, the opposite and unused TX or RX GT channels are powered down by default.

For designs that are generated with shared logic in the core *with* Transceiver Debug enabled, the user must drive the GT_PD (Power Down) bits of the unused GT channels, if required.

For designs that are generated with shared logic in the example design, the JESD204 PHY core controls the power down state of the individual GT channels. Powering down individual TX and RX lanes can be achieved using either the AXI register interface or the Transceiver debug pins.

Design Flow Steps

This chapter describes customizing and generating the 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 17]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 16]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 18]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 20]

Customizing and Generating the Core

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

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 17] 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 16] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 18].

Note: Figures in this chapter are illustrations of the JESD204 GUI in the Vivado Integrated Design Environment (IDE). This layout might vary from the current version.

Configuration Tab

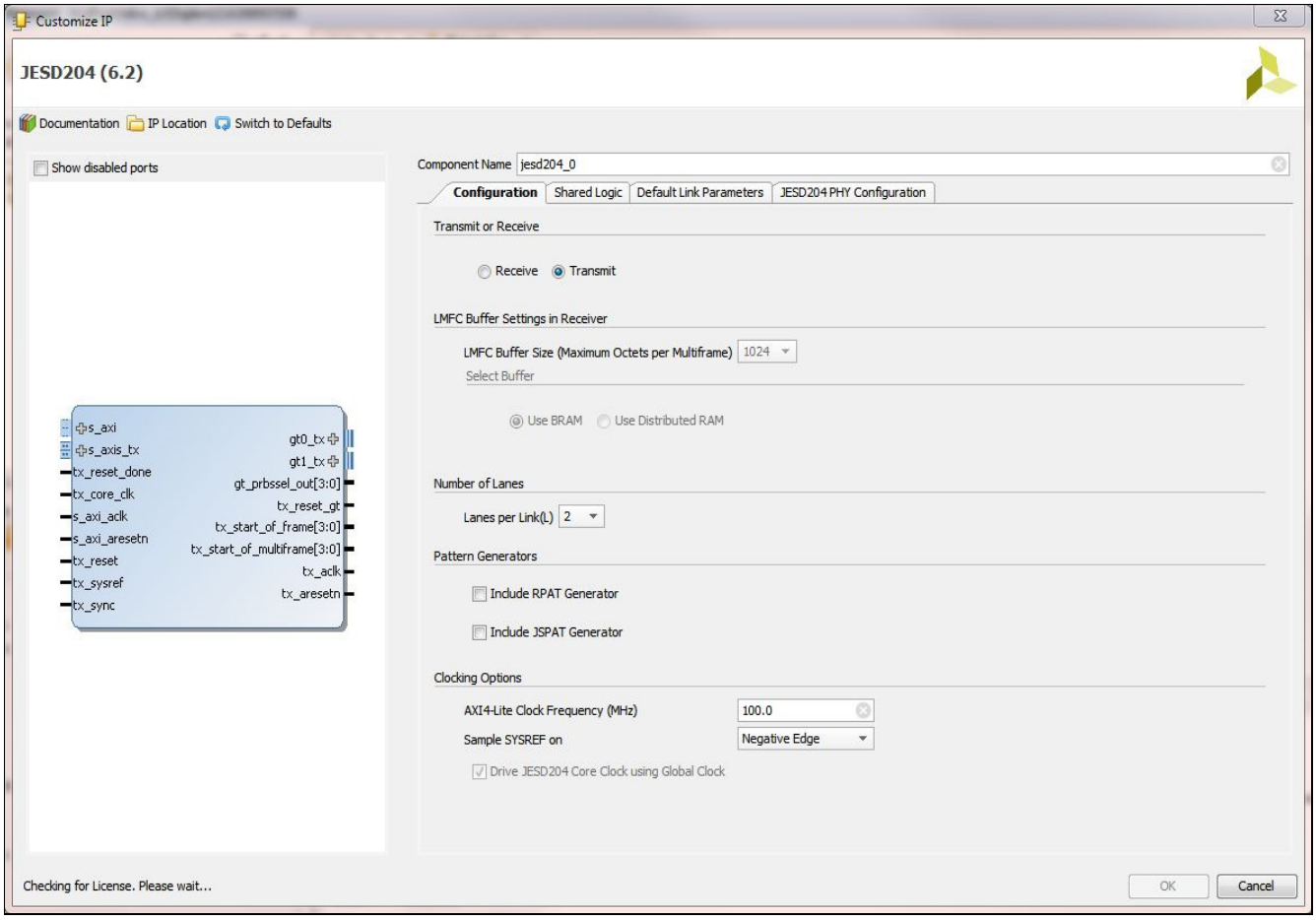


Figure 4-1: **Configuration Tab**

- **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. Separate Transmit and Receive cores can optionally share transceivers (see Sharing Transceivers between Transmit and Receive).
- **LMFC Buffer Settings in Receiver** – LMFC buffer (buffer for lane alignment and deterministic latency support). The size of this buffer (per serial lane) can be selected. In addition the LMFC buffer can be implemented using block RAM or distributed RAM.

IMPORTANT: *The LMFC buffer size option is included to allow minimal resource use where appropriate. The LMFC buffer size must be greater than the MAXIMUM multiframe size which is used in the implemented design (multiframe size is F*K, where F = octets per frame; K = frames per multiframe)*

- **Number of Lanes** – The core supports 1 to 12 lanes.

- **Pattern Generators** – Select "Include RPAT" and/or "Include JSPAT" to include the logic required to generate these test patterns if required. See Link Test Modes for more information.
- **AXI4-Lite Clock Frequency** – The frequency of the clock connected to the AXI4-Lite Management Interface.

  For UltraScale™ architecture devices the valid frequency range is from 10 MHz to 200 MHz.

  For 7 series devices, the valid frequency range available is dependent on the **Shared Logic Selection**:

  - When **Include Shared Logic in core** is selected, the AXI clock is common with the Transceiver DRP clock. The maximum frequency is limited to the maximum DRP clock frequency for the selected device.
  - When **Include Shared Logic in example design** is selected, the valid range is from 10 MHz to 200 MHz.
- **Clocking Options, SYSREF Sampling Edge** – The clock edge which is used to sample SYSREF can be selected. See SYSREF Handling for more information.

Link Configuration Tab

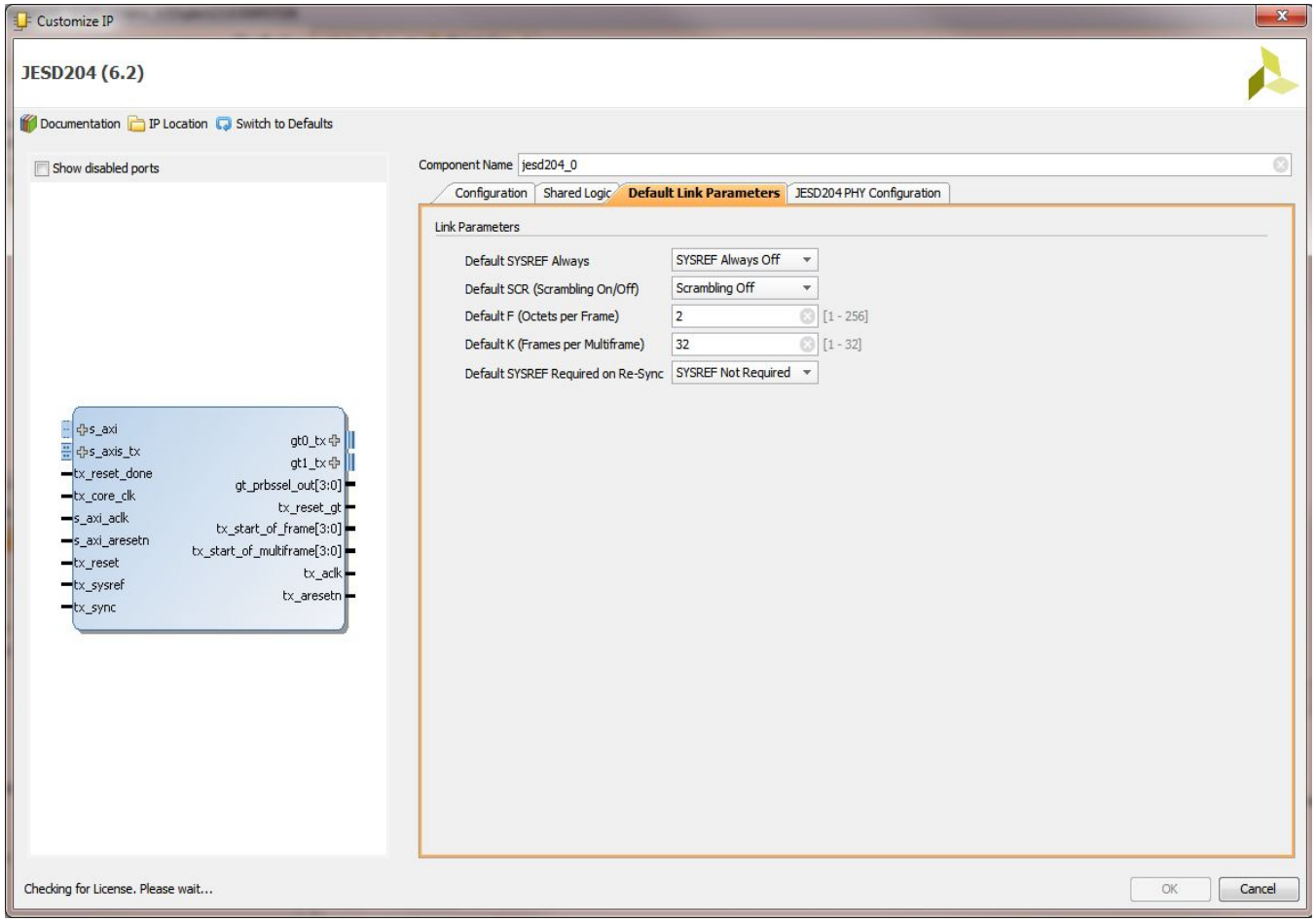


Figure 4-2: **Link Configuration Tab**

- **Default Link Parameters** – All of the following parameters set the default (reset) value stored in the equivalent configuration register, see Register Space in Chapter 2. The value set during core generation is overwritten if the register is written through the AXI4-Lite configuration bus. If software is used to configure the converter frame parameters, there is no need to configure values here. These configuration settings can be used to use the IP without an AXI4-Lite master connected. Only a small subset of the registers can be configured this way. To access all configuration settings and monitor link status, the AXI4-Lite interface should be connected.
  - **Octets per Frame (F)** – Sets the default (reset) value of register 0x020. See Table 2-23.
  - **Frames per Multiframe (K)** – Sets the default (reset) value of register 0x024. See Table 2-24.
  - **Scrambling Enabled** – Sets the default (reset) value of register 0x00C. See Table 2-18.

- **SYSREF Always** – Sets the default (reset) value of bit 0 of register 0x010. See Table 2-19.
- **SYSREF Required on Re-Sync** – Sets the default (reset) value of bit 16 of register 0x010. See Table 2-19.
- **Subclass** – Sets the default (reset) value of register 0x02C. See Table 2-26.

Shared Logic Tab

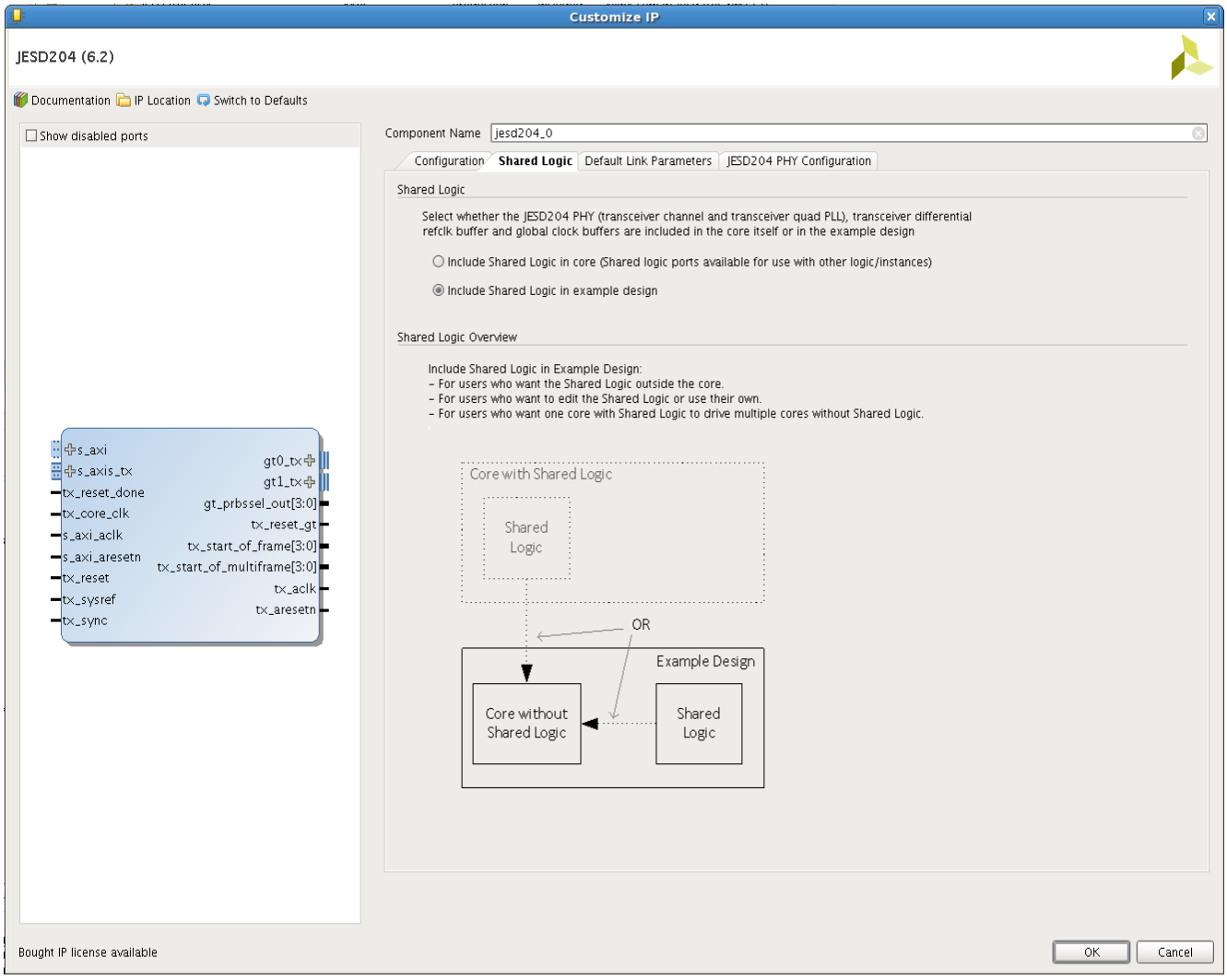


Figure 4-3: **Shared Logic Tab**

The JESD204 IP can be generated with the shared logic included in the core or with the shared logic included in the example design to enable the sharing of logic between multiple IP cores as shown in Figure 4-4. Select **Include Shared Logic in core** to include the JESD204 PHY core and clock buffers, if applicable in the core. Select **Include Shared Logic in example design** to instantiate the JESD204 core and clock buffers in the example design so they can be shared with other IP cores, if desired.

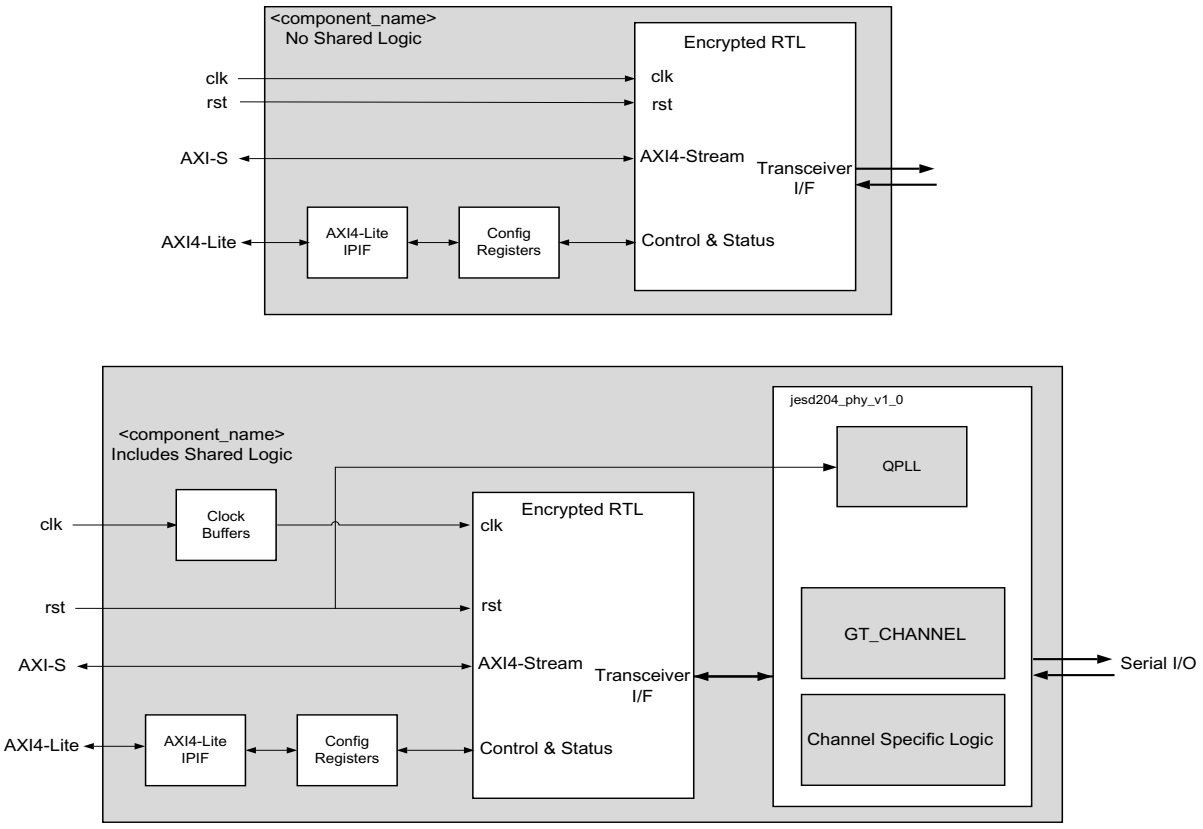


Figure 4-4: **Difference Between Cores Including and Excluding Shared Logic**

For simple unidirectional lanes, including the shared logic in the core is preferred. The QPLL output signals are available as top-level ports enabling the QPLL to be shared with other IP cores using the same transceiver quad. If you need to create a design where the transceiver is shared between a receive and transmit core, then generate cores with the shared logic in the example design and see Sharing Transceivers between Transmit and Receive, page 73.

JESD204 PHY Configuration Tab

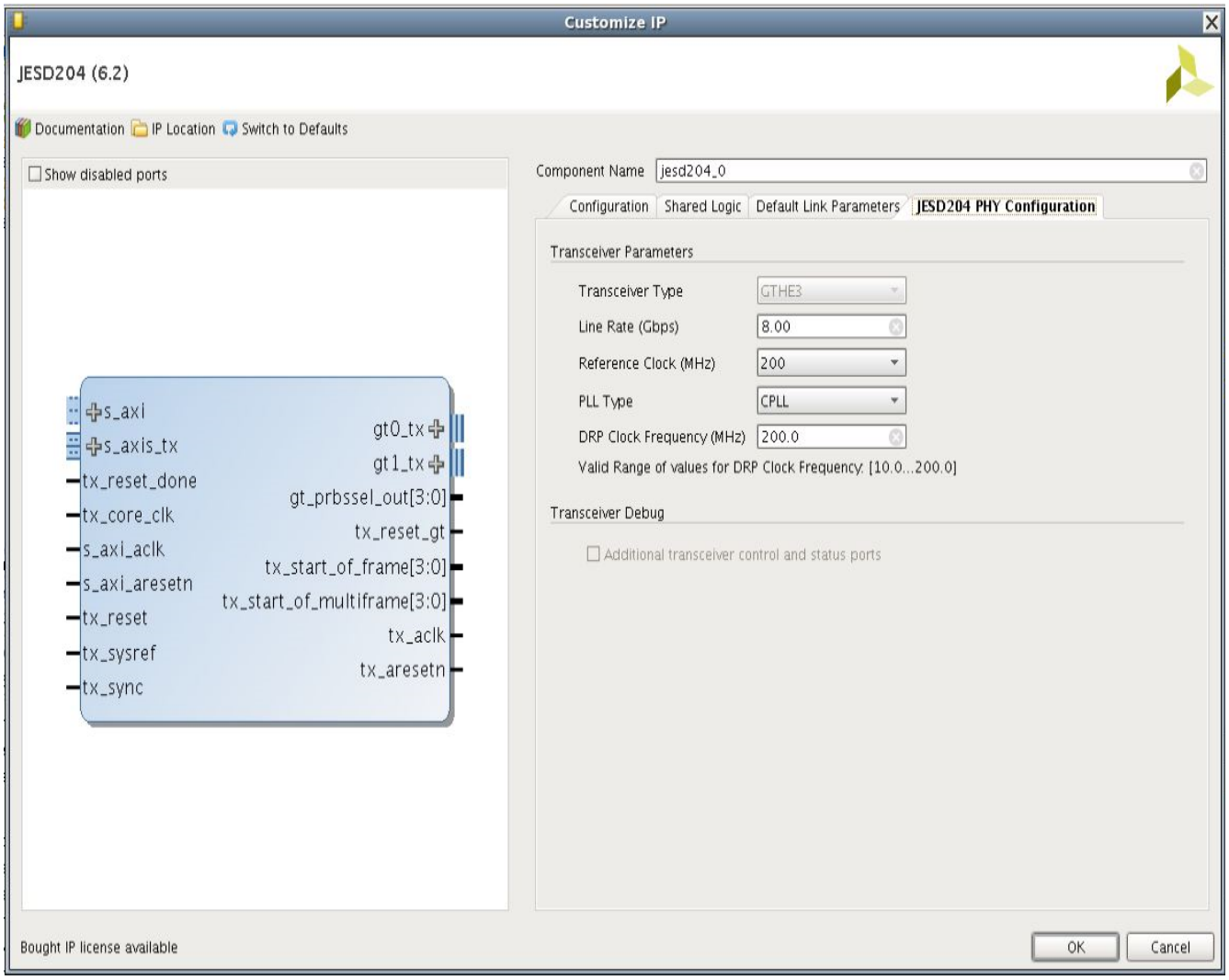


Figure 4-5: **JESD204 PHY Tab**

- **Transceiver Parameters** – The line rate and reference clock frequency can be selected using the Vivado IDE (see Figure 4-1). For any selected line rate, valid reference clock frequencies can be selected from a drop-down list.

  For UltraScale architecture devices, 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.

  Devices that support GTYE3 also allow the selection of the transceiver type (GTH or GTY).

  For 7 series devices, depending on the Shared Logic configuration, a DRP clock can be provided. When Shared Logic = 1 the DRP clock is tied to the AXI clock.

- **Additional Transceiver Control and Status Ports** – Select to include additional transceiver control and status ports for debug purposes. See Transceiver Debug Interface for more information. This selection is only available if the **Shared Logic** selection is set to **Shared Logic in Core**.

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	0 (= Transmit)
LMFC Buffer Size[2]	C_LMFC_BUFFER_SIZE	1024
Select Buffer[2]	C_USE_BRAM	1 (= Use BRAM)
Lanes per Link (L)	C_LANES	2
Include RPAT Generator[3]	USE_RPAT	FALSE
Include JSPAT Generator[3]	USE_JSPAT	FALSE
Sample SYSREF on	C_SYSREF_SAMPLING_EDGE	0 (= Negative Edge)
AXI4-Lite Clock Frequency	AXICLK_FREQ	100.00
Transceiver Parameters		
Line Rate[4]	GT_Line_Rate	8.0
Reference Clock[4]	GT_REFCLK_FREQ	200.0
DRP Clock Frequency	DRPCLK_FREQ	200.0 for UltraScale 100.0 for 7 series
PLL Type	C_PLL_SELECTION	0 (=CPLL)
Shared Logic	SupportLevel	0
Include Shared Logic in core	1	
Include Shared Logic in example design	0	
Additional Transceiver Control and Status ports	TransceiverControl	FALSE

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. Receiver only.
3. Transmitter only.
4. Varies depending on device.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 16]. The final step in generating IP is to select DCP. If you have included the shared logic in the core and need to change the transceiver configuration then ensure that DCP is not selected.

Constraining the Core

This section describes how to constrain a design containing the JESD204 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.

SYSREF Constraints

The example design provided with the JESD204 core has an example of a setup and hold window assuming the SYSREF and clock have aligned edges. The following example is for a transmit design with a 1 ns setup and hold window (the exact requirements for these constraints depend on the chosen method of generating SYSREF and the board layout, the constraints provided with the core are for example only):

```
set_input_delay -clock clk -max 0.5 [get_ports tx_sysref]

set_input_delay -clock clk -min -0.5 [get_ports tx_sysref]
```

If the SYSREF input sampling edge is changed, these constraints require modification. See SYSREF Sampling Clock Edge in Chapter 3 for more 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. In some cases, example transceiver location constraints can be found in the example design XDC file.

I/O Standard and Placement

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

Simulation

TFor 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 20].

IMPORTANT: *For cores targeting 7 series or Zynq-7000 AP SoC devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 16].

Example Design

The JESD204 IP can be generated as an IP in TX or RX configuration. Both selections come with a lightweight test harness to enable familiarization with the design and signal interface.

In Vivado®, create a new empty project. Be sure to select the FPGA part that you wish to use. Using the Vivado IP catalog, select the JESD204 IP core and configure as required. Right-click the block under **Design Sources** and select **Open IP Example Design**, as shown in Figure 5-1. This opens a new Vivado project containing the complete RX or TX design example.

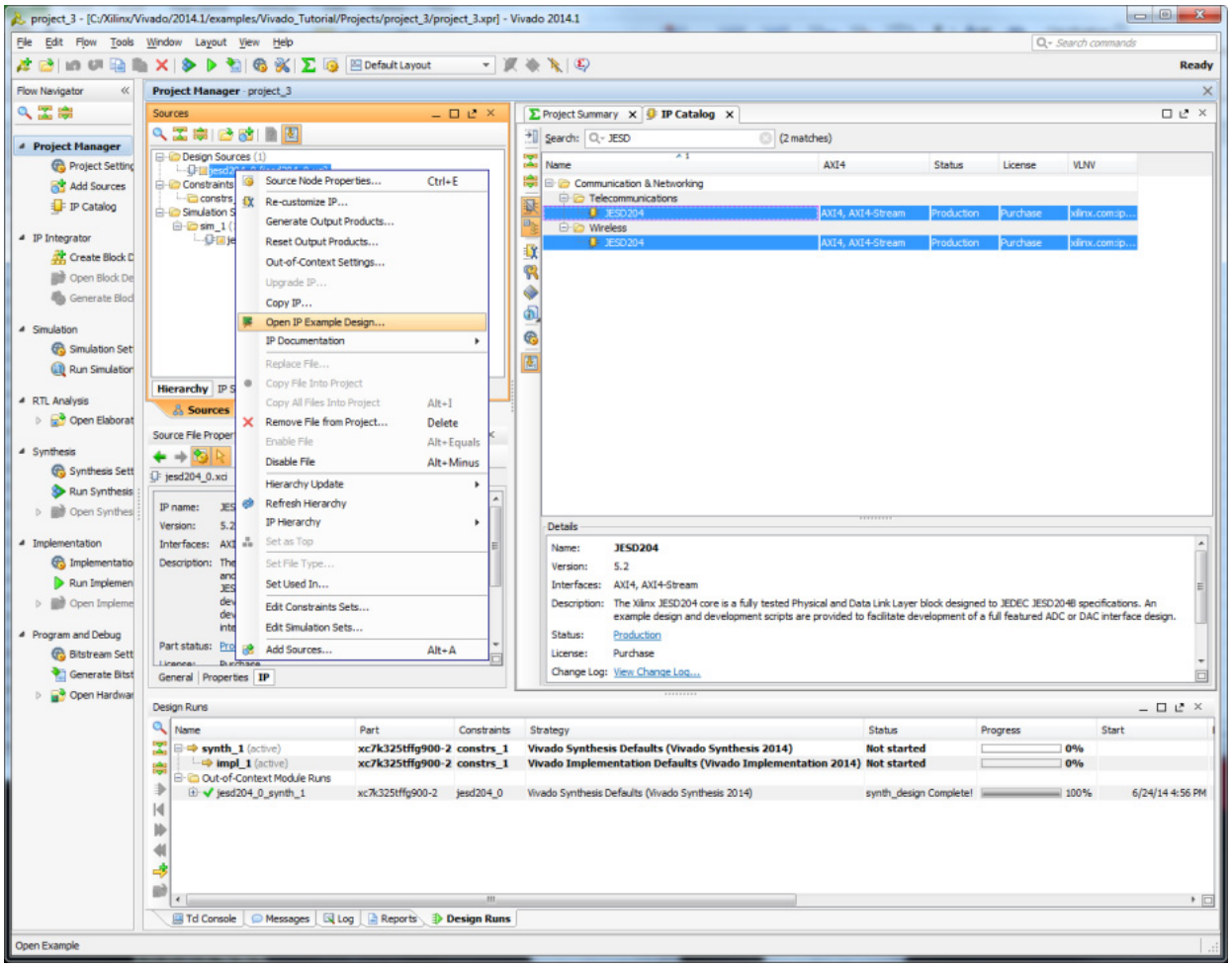


Figure 5-1: **Opening the Example Design**

Common Design Elements

The example design is configured to model one ADC per lane, and samples at twice the core clock rate. Both the RX and TX JESD204 IP blocks can be configured for 1 to 12 lanes. Your generated design example reflects this choice. For all lane configurations, F is set to 2, therefore for all examples LMF = LL2.

Valid configurations are 112, 222, 332, 442, 552, 662, 772, 882, 992, 10102, 11112 and 12122.

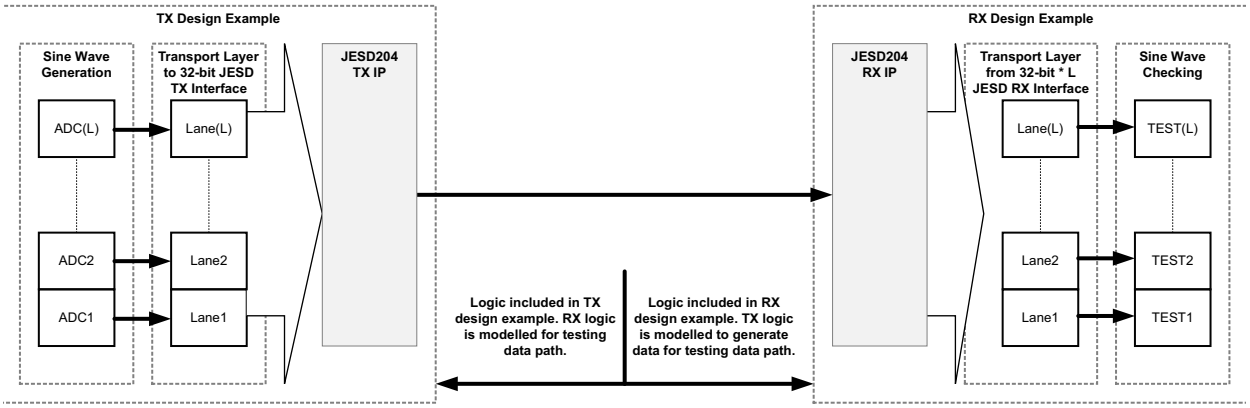


Figure 5-2: **TX to RX Data Path Summary**

In both example designs, the JESD204 interface parameters are the same and are listed in Table 5-1. The initialization sequences configure both designs. You can modify the line and core clock rates during IP configuration if using UltraScale™ devices.

Table 5-1: **Interface Parameter Description**

Parameter	Unit	Value	Comment
F (Octets per frame)		2	This can be modified as a multiple of two.
ADC Sample Width	Bits	14	Two control bits are added to increase this to 16-bits.

Transport Layer Mapping

The JESD204 per-lane data interface is 32 bits wide. The AXI4-Stream interface is *L**32-bits wide. Samples must be mapped into and out of this interface. The design example packages two samples per 32-bit data word passed to the AXI4-Stream interface. Each sample is made up of a 14-bit sine wave sample and two example control bits. The mapping is consistent across each lane. Samples are not distributed between lanes. There is one sample per frame, and the sample bit mapping is:

- **Sample (S)[15:14]** – Example ADC control bits
- **Sample (S)[13:0]** – Sine wave sample value

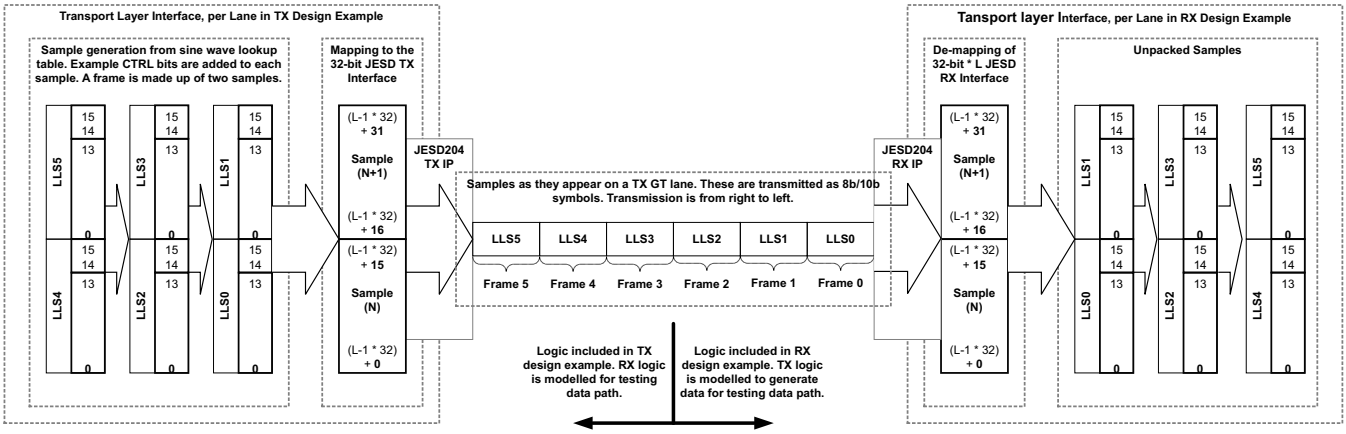


Figure 5-3: **Per-lane Data Generation and Transport Layer Mapping of Data**

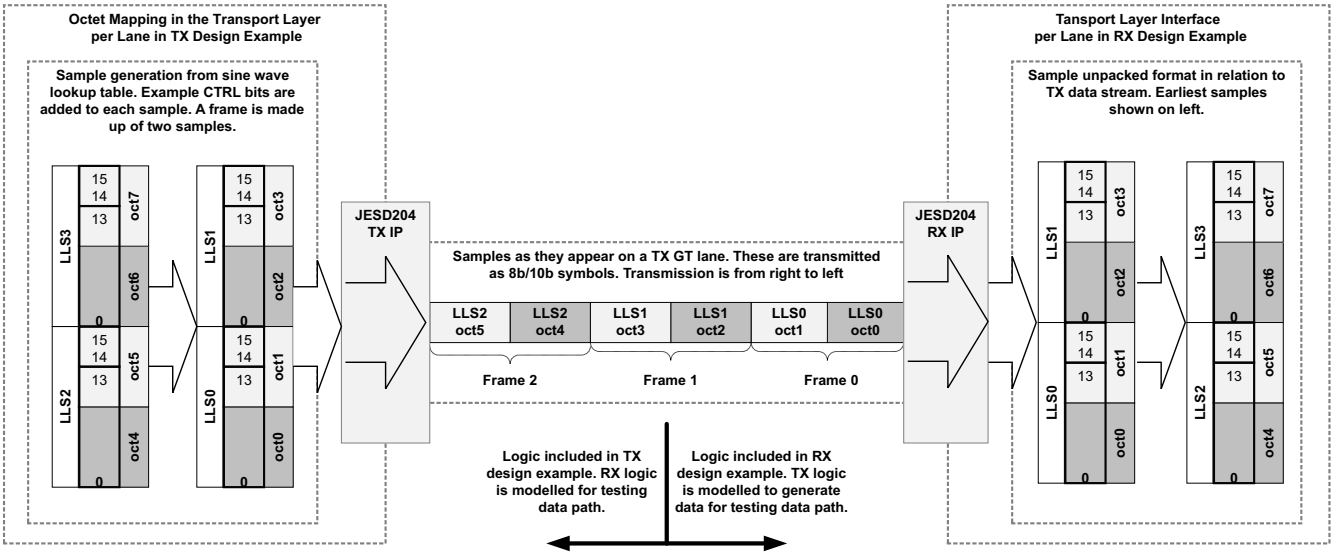


Figure 5-4: **Per-lane Transport Layer Octet/Frame Mapping**

sysref Generation

The `sysref` signal is generated in both the TX and RX test bench using `tx_aclk`/`rx_aclk`.

TX Block Layout

The design example is the same for both shared logic options. There is a subtle hierarchical name change denoting the top-level of the IP within the example design. In Figure 5-5 and Figure 5-6, the blocks included in the generated top-level IP are shown within the shaded box.

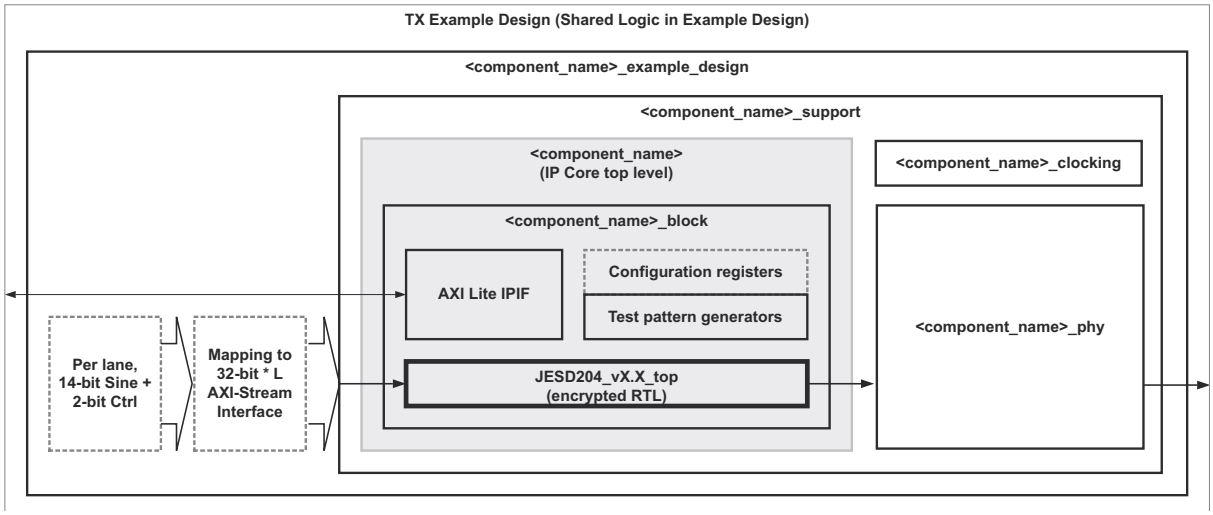


Figure 5-5: **TX Example Design with Shared Logic in Example Design**

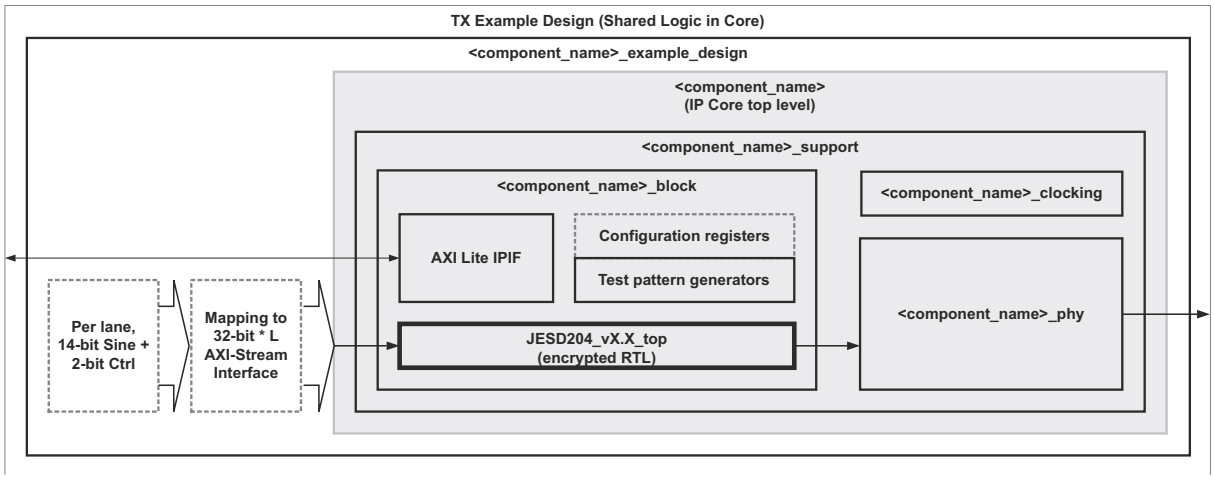


Figure 5-6: **TX Example Design with Shared Logic in Core**

TX Transport Layer Mapping

The JESD204 per-lane data interface is 32 bits wide. The AXI4-Stream interface is *L**32-bits wide. Samples must be mapped into and out of this interface. The design example packages two samples per 32-bit data word passed to the AXI4-Stream interface. Each sample is made up of a 14-bit sine wave sample and two example control bits. The mapping is consistent across each lane. Samples are not distributed between lanes. There is one sample per frame, and the sample bit mapping is:

- **Sample (S)[15:14]** – Example ADC control bits
- **Sample (S)[13:0]** – Sine wave sample value

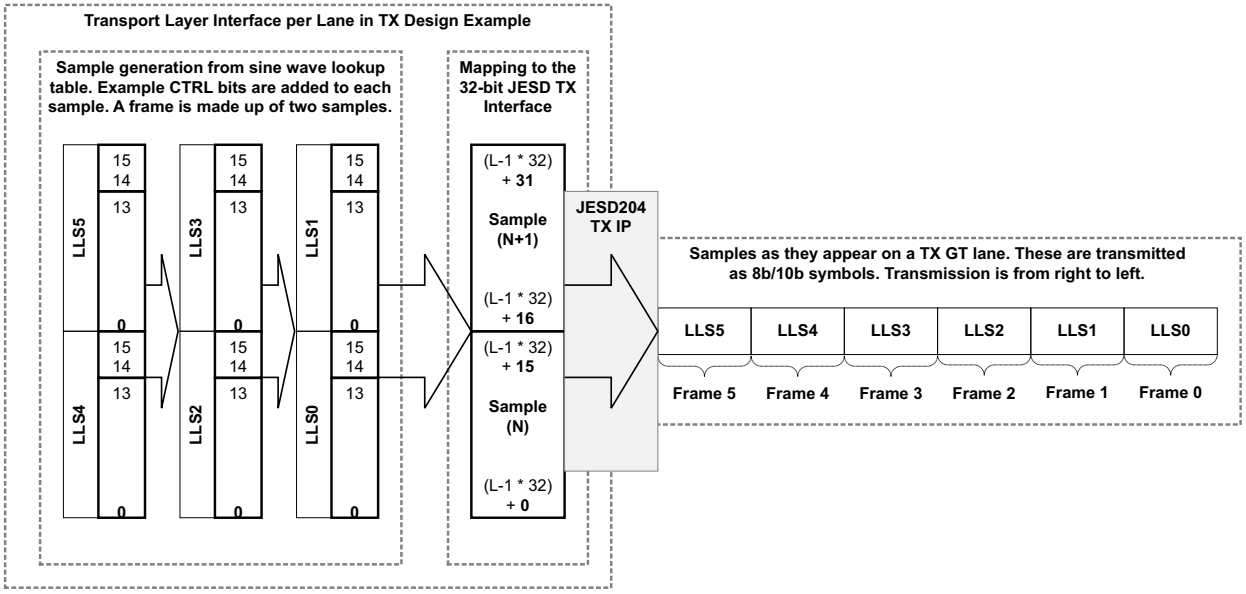


Figure 5-7: **Per-lane Generation and Transport Layer Mapping of Data for the TX Data Path**

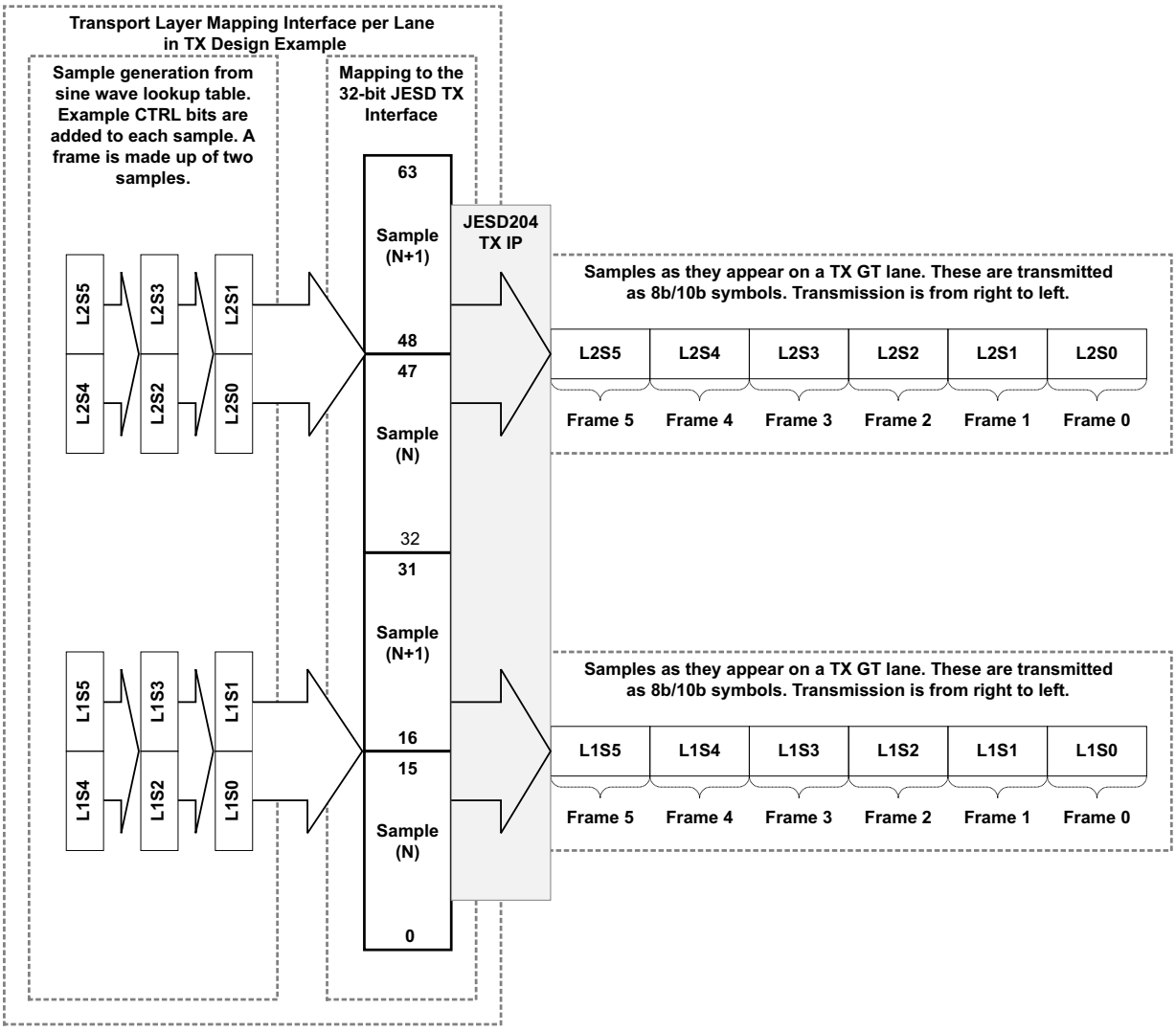


Figure 5-8: **Data Generation and Transport Layer Mapping for a 2-Lane Example, LMF= 222**

RX Block Layout

Similar to the TX, the design example is the same for both shared logic options. There is a subtle hierarchical name change denoting the top-level of the IP within the example design. In Figure 5-9 and Figure 5-10, the blocks included in the generated top-level IP are shown within the shaded box.

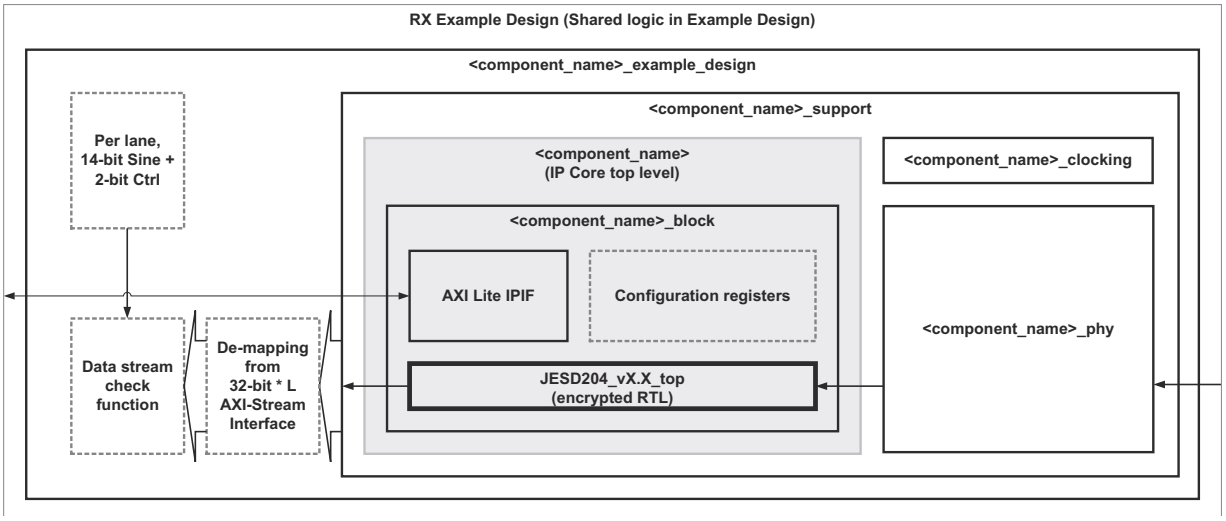


Figure 5-9: **RX Example Design Hierarchy, Non-Shared**

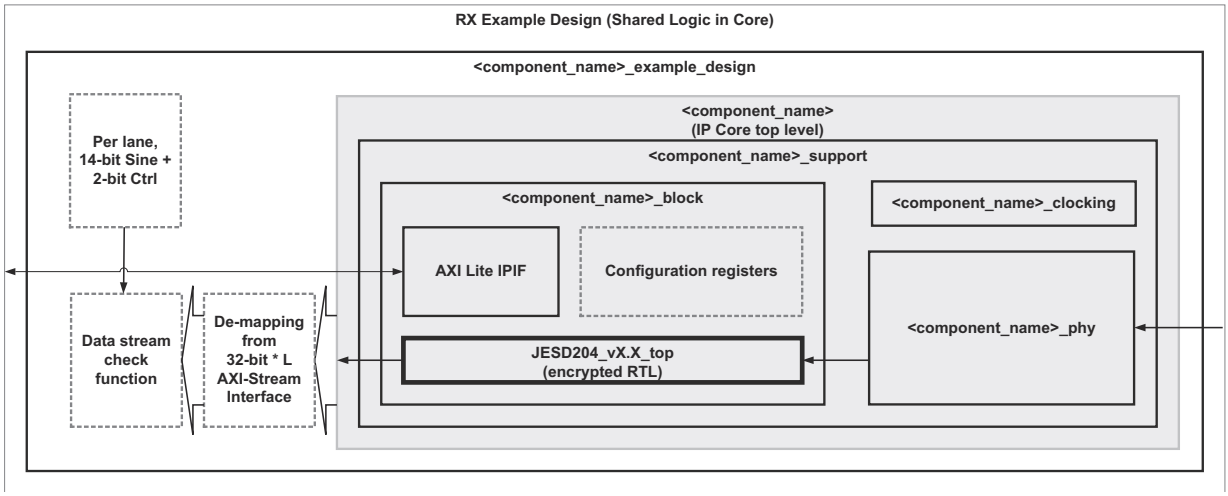


Figure 5-10: **RX Example Design Hierarchy, Shared**

RX Transport Layer Mapping

The JESD204 per-lane data interface is 32 bits wide. The AXI4-Stream interface is *L**32-bits wide. Samples must be mapped into and out of this interface. The design example packages two samples per 32-bit data word passed to the AXI4-Stream interface. Each sample is made up of a 14-bit sine wave sample and two example control bits. The mapping is consistent across each lane. Samples are not distributed between lanes. There is one sample per frame, and the sample bit mapping is:

- **Sample (S)[15:14]** – Example ADC control bits
- **Sample (S)[13:0]** – Sine wave sample value

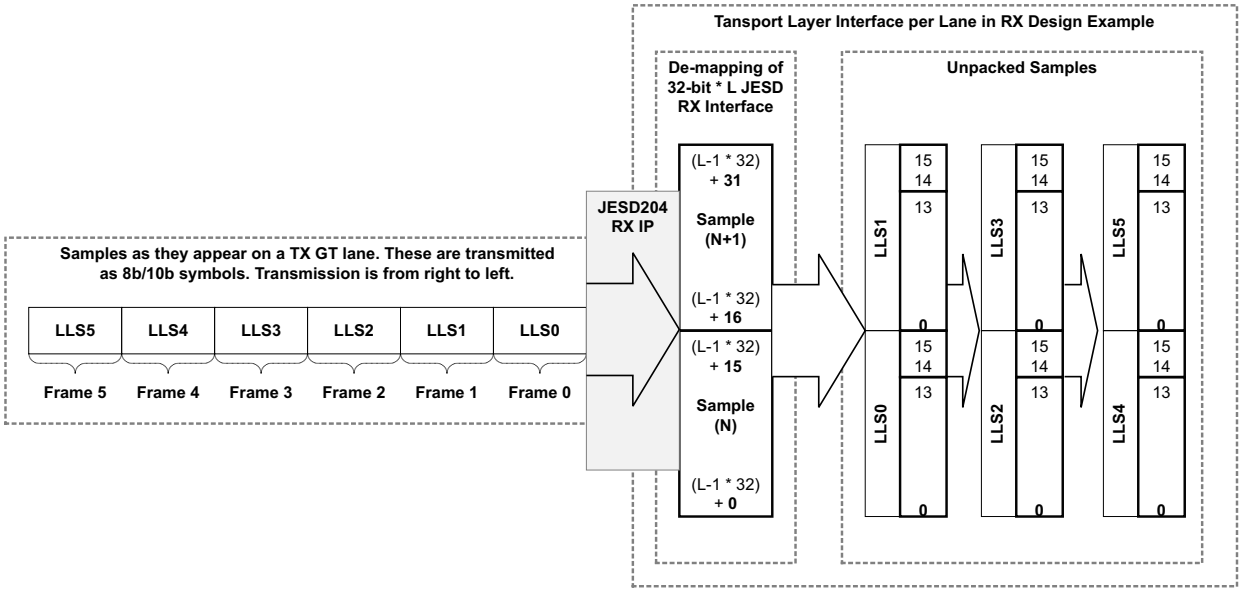


Figure 5-11: **Per-lane Generation and Transport Layer Mapping of Data for the RX Data Path**

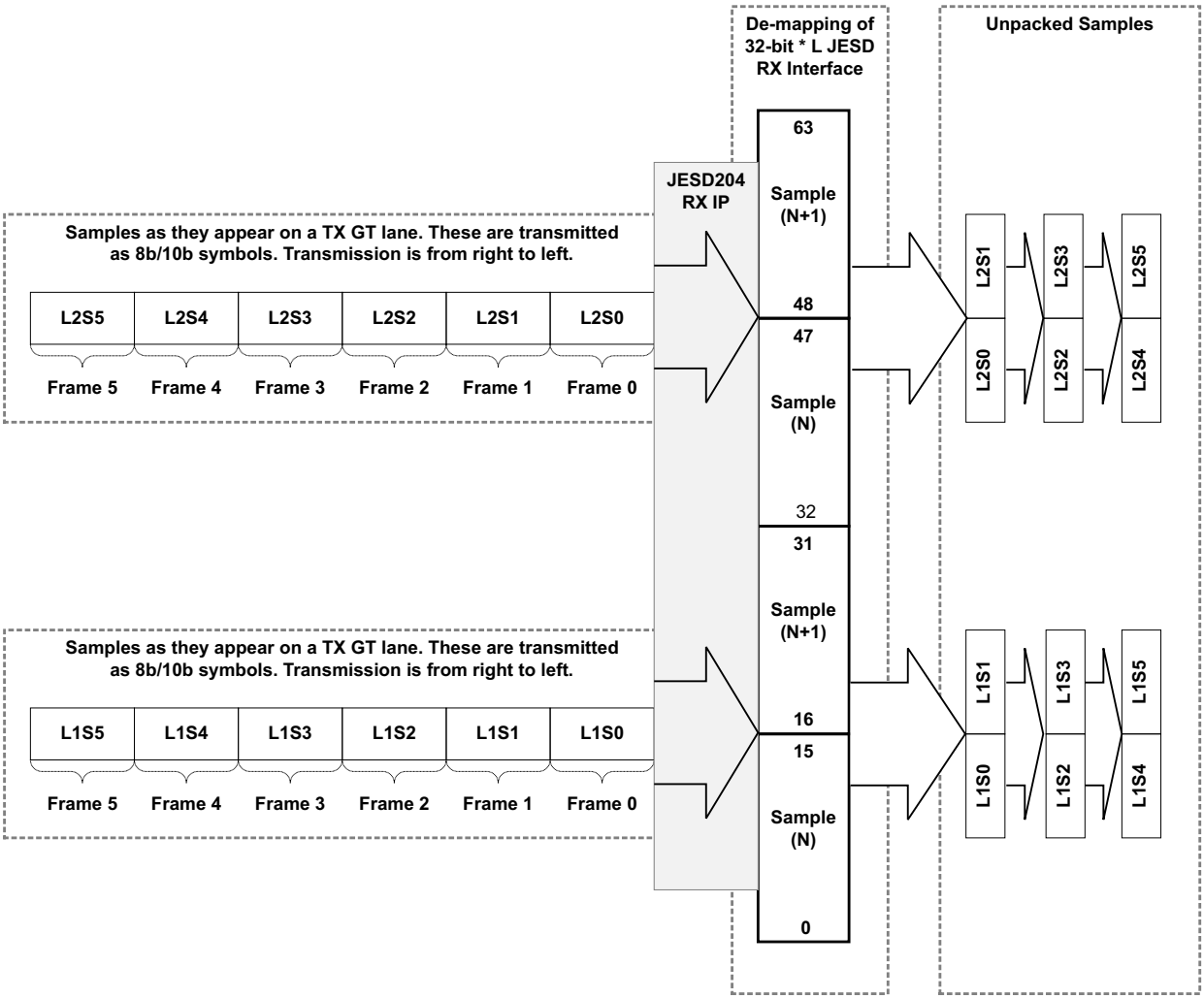


Figure 5-12: **Data Generation and Transport Layer Mapping for a 2-Lane Example, LMF= 222**

Test Bench

The example design supplied with the JESD204 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, Example Design 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.

The demonstration test bench is delivered, in one of two formats depending on whether you have created a transmitter or receiver core, as described in Chapter 5, Example Design. Figure 6-1 shows the transmitter core test bench and Figure 6-2 shows the receiver core test bench.

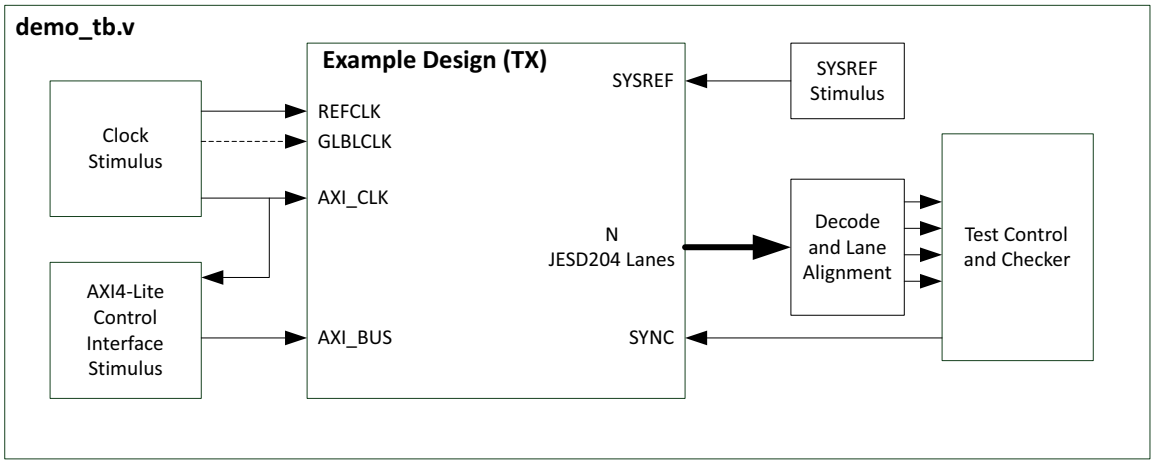


Figure 6-1: **JESD204 Transmitter Test Bench**

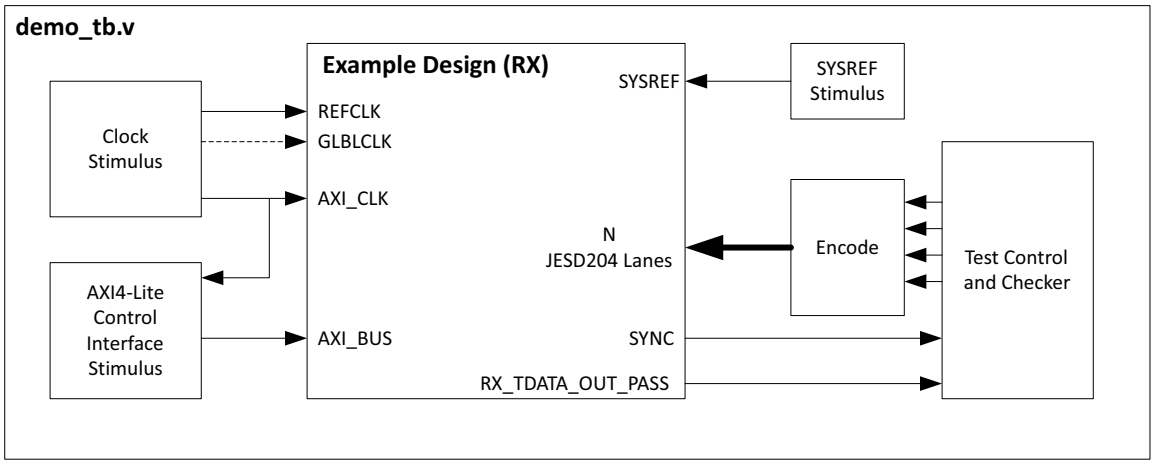


Figure 6-2: **JESD204 Receiver Test Bench**

Both the transmitter and receiver core test benches include the following common stimulus:

- **Clocking** – Includes clocking stimulus for `REFCLK`, `GLBLCLK` (if required by the selected configuration), and the AXI4-Lite interface clock `s_axi_aclk`. The generated clock frequencies reflect the frequency choices selected in the JESD204 core configuration GUI.
- **SYSREF** – The test bench generates a repeating `SYSREF` signal with a period of four multi-frames (equal to F*K frame clock periods).
- **Configuration** – The test bench configures the programmable core parameters using the core register interface over the AXI4-Lite bus interface (see Register Space in Chapter 2 for more details). The parameters are configured as shown in Table 6-1. The following procedure is followed when programming the core registers:
  a. Program the required core registers.
  b. Assert core reset by writing 0x1 to the core reset register.
  c. Poll the core reset register until the reset bit has cleared.

Table 6-1: **Parameter Descriptions**

Parameter	Value	Encoded Value Programmed	Description
F	2	1	Number of octets per frame
K	32	31	Number of frames per multi-frame
L	Lanes	Lanes-1	Number of lanes for which the core was generated
M	Lanes	Lanes-1	Number of lanes for which the core was generated (test bench simulates one converter per lane)
N	14	13	Converter resolution
N'	16	15	Total number of bits per sample
S	1	0	Number of samples per converter per frame cycle

Table 6-1: **Parameter Descriptions *(Cont'd)***

Parameter	Value	Encoded Value Programmed	Description
SCR	0	0	Scrambling disabled
CF	1	1	Control words per frame clock
CS	2	2	Number of control bits per sample
SUBCLASSV	1	1	JESD204 Subclass version = 1

The transmitter core test bench includes logic to:

- Decode the 8B/10B serialized data streams from the TX ports of the example design.
- Align the received data lanes.
- Check the data from the example design. The test bench expects 14-bit sine wave samples with 2-bit control information generated in the transmitter core example design.

The receiver core test bench includes logic to:

- Generate and map 14-bit sine wave samples with 2-bit control information as expected by the receiver core example design.
- Encode and serialize 8B/10B data into the RX ports of the example design.
- Check the `rx_tdata_out_pass` signal from the example design. This output bit confirms the example design has received the data as expected.

To run the test bench and simulate the example design, ensure you have first opened the example design and select **Run Simulation** under the Simulation flow in the Vivado IDE. For further details on setting up the simulation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 16].

After selecting **Run Simulation**, by default Vivado does the following:

- Compiles the test bench and example design including your generated core.
- Launches the simulator.
- Opens the waveforms widow and displays the signals in the test bench.
- Runs the simulation for 1 ms.

Now, the simulation has only just started. You are required to manually run the simulator until the simulation completes. You should see the following log output in the simulator console after a successful simulation run.

TX core log:

```
Resetting the core...
Test Started
** Reset Done, Wait for GT reset
Version = Major   6 Minor   0 Rev   0
** GT Reset Done
** Wait for K28.5
Reset complete..
** Assert Sync
** ILA Start
*** End of Multi Frame 1
*** End of Multi Frame 2
*** End of Multi Frame 3
*** End of Multi Frame 4
** ILA Complete
** Test Passed
** Test completed successfully
```

RX core log:

```
Resetting the core...
Test Started
** Reset Done
Version = Major   6 Minor   0 Rev   0
Reset complete..
** Rx sync asserted
Sync complete..
** Sysref asserted
** Init sequence complete. Sending sample data
** Test Passed
** Test completed successfully
```

To gain a better understanding of the operation of the signals into and out of your generated core, observe in the simulation what is happening inside the example design. To see this, right-click the DUT instance and select **Add To Wave Window**. This shows all the signals from the example design level of the hierarchy. You must rerun the simulation from the start to see the values for the added waveforms.

Appendix A

Verification, Compliance, and Interoperability

The JESD204 core has been verified using both simulation and hardware testing.

Simulation

A highly parameterizable transaction-based simulation test suite has been used to verify the core. Tests include:

- Scrambling and alignment
- Loss and regain of synchronization
- Frame transmission
- Frame reception
- Recovery from error conditions

Hardware Testing

The core has been used in many hardware test platforms within Xilinx and in interoperability testing with external hardware vendors.

Hardware Demonstration Design

For information on the Hardware Demonstration Design see the documentation (registration required).

Migrating and Upgrading

This appendix contains information about upgrading to a recent version of the IP core.

Migrating to the Vivado Design Suite

For information on migrating from the ISE® Design Suite to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 21].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations between core versions.

Device Migration

If you are migrating from a 7 series GTX or GTH device to an UltraScale GTH device, the prefixes of the optional transceiver debug ports for single-lane cores are changed from `gt0`, `gt1` to `gt`, and the postfix `_in` and `_out` are dropped. For multi-lane cores, the prefixes of the optional transceiver debug ports `gt(n)` are aggregated into a single port. For example, `gt0_gtrxreset` and `gt1_gtrxreset` now become `gt_gtrxreset[1:0]`. This is true for all ports, with the exception of the DRP buses which follow the convention of `gt(n)_drpxyz`.

It is important to update your design to use the new transceiver debug port names. For more information about migration to UltraScale architecture-based devices, see the *UltraScale Architecture Migration Methodology Guide* (UG1026) [Ref 22].

Upgrading from JESD204 v6.0

No changes are required.

Upgrading from JESD204 v6.1 or v6.0

Upgrading to the JESD204 v6.2 core: For designs the are built with shared logic in the core no changes are required. For designs that are built with shared logic in the example design, upgrading the JESD204 PHY will produce critical warnings about external port differences.

Upgrading from JESD204 v5.2

In addition to the detail above. The JESD204 PHY core has two new reset ports added (tx_sys_rst and rx_sys_rst). These ports should be connected to the reset source that drives the tx_reset and rx_reset input to your JESD204 cores. See [Ref 4] for further detail on the new JESD204 PHY ports.

Upgrading to the JESD204 v6.2 core depends on your use of the core. This section includes upgrade information for the following solutions:

- Transmitter with Shared Logic in Example Design
- Transmitter with Shared Logic in Example Design Converted to Include Shared Logic in Core
- Receiver with Shared Logic in Example Design
- Receiver with Shared Logic in Example Design Converted to Include Shared Logic in Core
- Transmitter with Shared Logic in Core
- Receiver with Shared Logic in Core

Transmitter with Shared Logic in Example Design

The interface between the core and the Transceiver has been changed to have separate buses for each lane in the link. An example of the changes is shown below for a 2-lane design. The ports:

- output [63:0] `gt_txdata_out`
- output [63:0] `gt_txcharisk_out`

have been replaced with:

- Lane 0
  - output [31:0] `gt0_txdata`
  - output [3:0] `gt0_txcharisk`
- Lane 1
  - output [31:0] `gt1_txdata`

- output [3:0] `gt1_txcharisk`

To change a design using v5.2 of the core to use v6.0, follow the changes shown in Figure C-1.

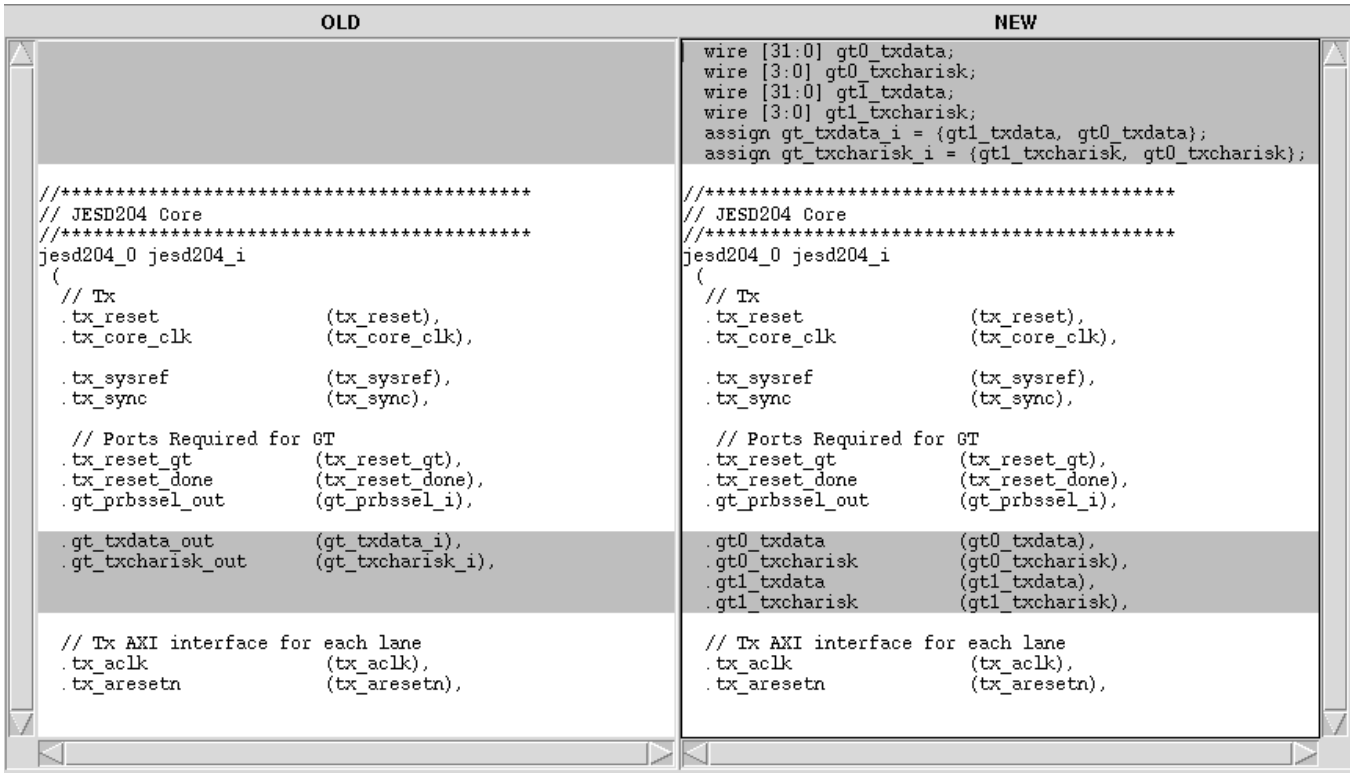


Figure C-1: **Upgrading Transmitter with Shared Logic in Example Design**

Transmitter with Shared Logic in Example Design Converted to Include Shared Logic in Core

IMPORTANT: *This is a more complicated upgrade, and if the transceiver logic has been modified from that provided with the example design, it is best to follow the steps for Transmitter with Shared Logic in Example Design.*

This scenario is now possible because the transceiver parameters can be updated using the JESD204 IP GUI. To perform the upgrade:

1. Upgrade the JESD204 IP and ignore the critical warnings about pin changes.
2. Double-click on the JESD204 XCI file to re-customize the IP.
3. Change from using shared logic in example design to shared logic in core in the Shared Logic Tab.
4. Check that the transceiver line rate and reference clock match the values used in your existing Transceiver Wizard instantiation.

5. Allow the IP to regenerate.
6. Modify the instantiation of jesd204_0_support in `jesd204_0_example_design.v` or your own RTL top-level that instantiates jesd204_0_support, as shown in Figure C-2.

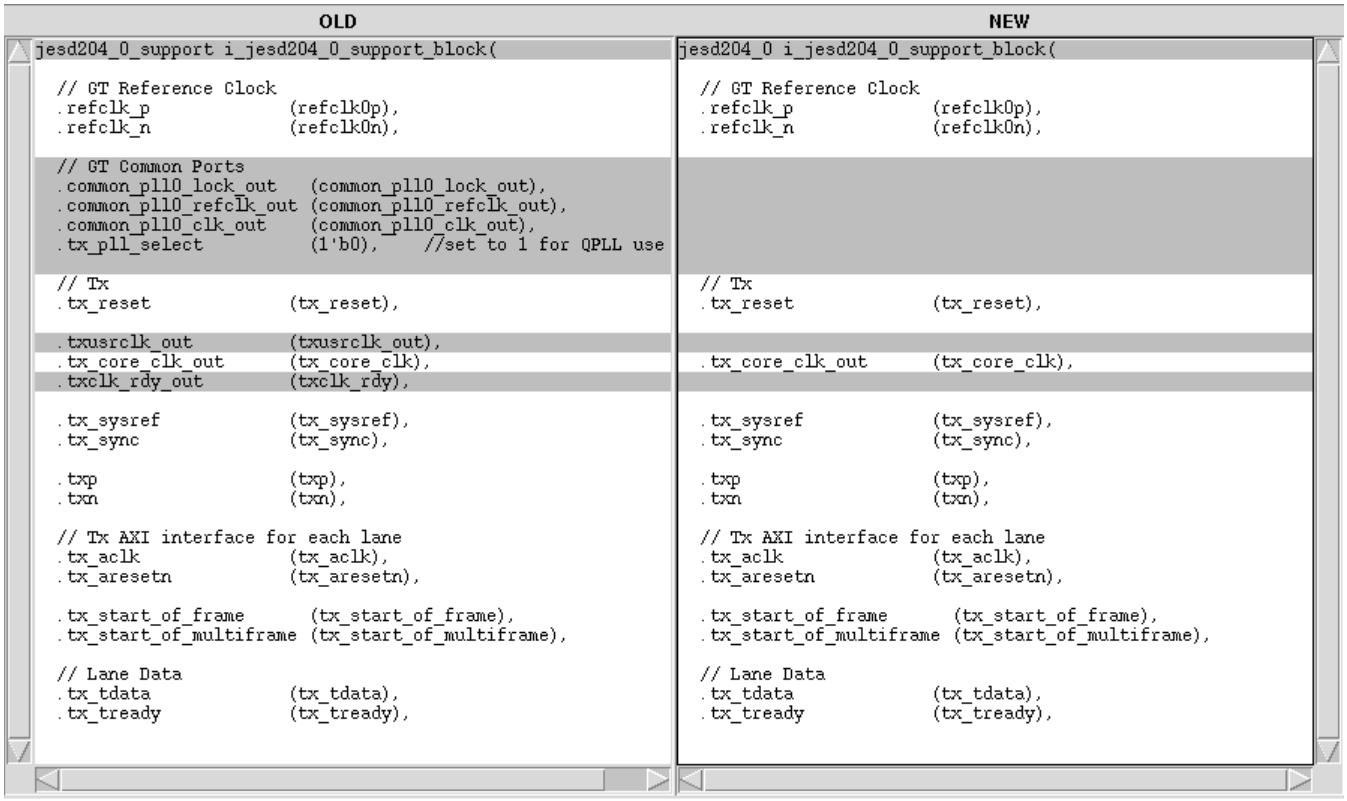


Figure C-2: **Upgrading Transmitter with Shared Logic in Ex. Des. Converted to Incl. Shared Logic in Core**

TIP: *There will be redundant files in the project which can be removed, these contain the old Transceiver Wizard and associated logic.*

Receiver with Shared Logic in Example Design

The interface between the core and the transceiver has been changed to have separate buses for each lane in the link. An example of the changes is shown below for a 2-lane design. The ports:

- input [63:0] `gt_rxdata_in`
- input [7:0] `gt_rxcharisk_in`
- input [7:0] `gt_rxdisperr_in`
- input [7:0] `gt_rxnotintable_in`

have been replaced with:

- input [31:0] `gt0_rxdata`
- input [3:0] `gt0_rxcharisk`
- input [3:0] `gt0_rxdisperr`
- input [3:0] `gt0_rxnotintable`
- input [31:0] `gt1_rxdata`
- input [3:0] `gt1_rxcharisk`
- input [3:0] `gt1_rxdisperr`
- input [3:0] `gt1_rxnotintable`

To change a design using v5.2 of the core to use v6.2, make the changes to the IP instantiation shown in Figure C-3.

OLD

```
//*****************************************
// JESD204 Core
//*****************************************
jesd204_0 jesd204_i
 (

  // Rx
 .rx_reset               (rx_reset),
 .rx_core_clk            (rx_core_clk),

 .rx_sysref              (rx_sysref),
 .rx_sync                (rx_sync),

 // Ports Required for GT
 .rx_reset_gt            (rx_reset_gt),
 .rxencommaalign_out     (rxencommaalign_i),
 .rx_reset_done          (rx_reset_done),

 .gt_rxdata_in           (gt_rxdata_i),
 .gt_rxcharisk_in        (gt_rxcharisk_i),
 .gt_rxdisperr_in        (gt_rxdisperr_i),
 .gt_rxnotintable_in     (gt_rxnotintable_i),

 // Rx AXI-S interface for each lane
 .rx_aclk                (rx_aclk),
 .rx_aresetn             (rx_aresetn),

 .rx_start_of_frame      (rx_start_of_frame),
 .rx_end_of_frame        (rx_end_of_frame),
 .rx_frame_error         (rx_frame_error),

 .rx_tdata               (rx_tdata),
 .rx_tvalid              (rx_tvalid),
```

NEW

```
 assign gt0_rxdata =       gt_rxdata_i[31:0];
 assign gt0_rxcharisk =    gt_rxcharisk_i[3:0];
 assign gt0_rxdisperr =    gt_rxdisperr_i[3:0];
 assign gt0_rxnotintable = gt_rxnotintable_i[3:0];

 assign gt1_rxdata =       gt_rxdata_i[63:32];
 assign gt1_rxcharisk =    gt_rxcharisk_i[7:4];
 assign gt1_rxdisperr =    gt_rxdisperr_i[7:4];
 assign gt1_rxnotintable = gt_rxnotintable_i[7:4];

//*****************************************
// JESD204 Core
//*****************************************
jesd204_0 jesd204_i
 (

  // Rx
 .rx_reset               (rx_reset),
 .rx_core_clk            (rx_core_clk),

 .rx_sysref              (rx_sysref),
 .rx_sync                (rx_sync),

 // Ports Required for GT
 .rx_reset_gt            (rx_reset_gt),
 .rxencommaalign_out     (rxencommaalign_i),
 .rx_reset_done          (rx_reset_done),

 .gt0_rxdata             (gt0_rxdata),
 .gt0_rxcharisk          (gt0_rxcharisk),
 .gt0_rxdisperr          (gt0_rxdisperr),
 .gt0_rxnotintable       (gt0_rxnotintable),

 .gt1_rxdata             (gt1_rxdata),
 .gt1_rxcharisk          (gt1_rxcharisk),
 .gt1_rxdisperr          (gt1_rxdisperr),
 .gt1_rxnotintable       (gt1_rxnotintable),

 // Rx AXI-S interface for each lane
 .rx_aclk                (rx_aclk),
 .rx_aresetn             (rx_aresetn),

 .rx_start_of_frame      (rx_start_of_frame),
 .rx_end_of_frame        (rx_end_of_frame),
 .rx_frame_error         (rx_frame_error),

 .rx_tdata               (rx_tdata),
 .rx_tvalid              (rx_tvalid),
```

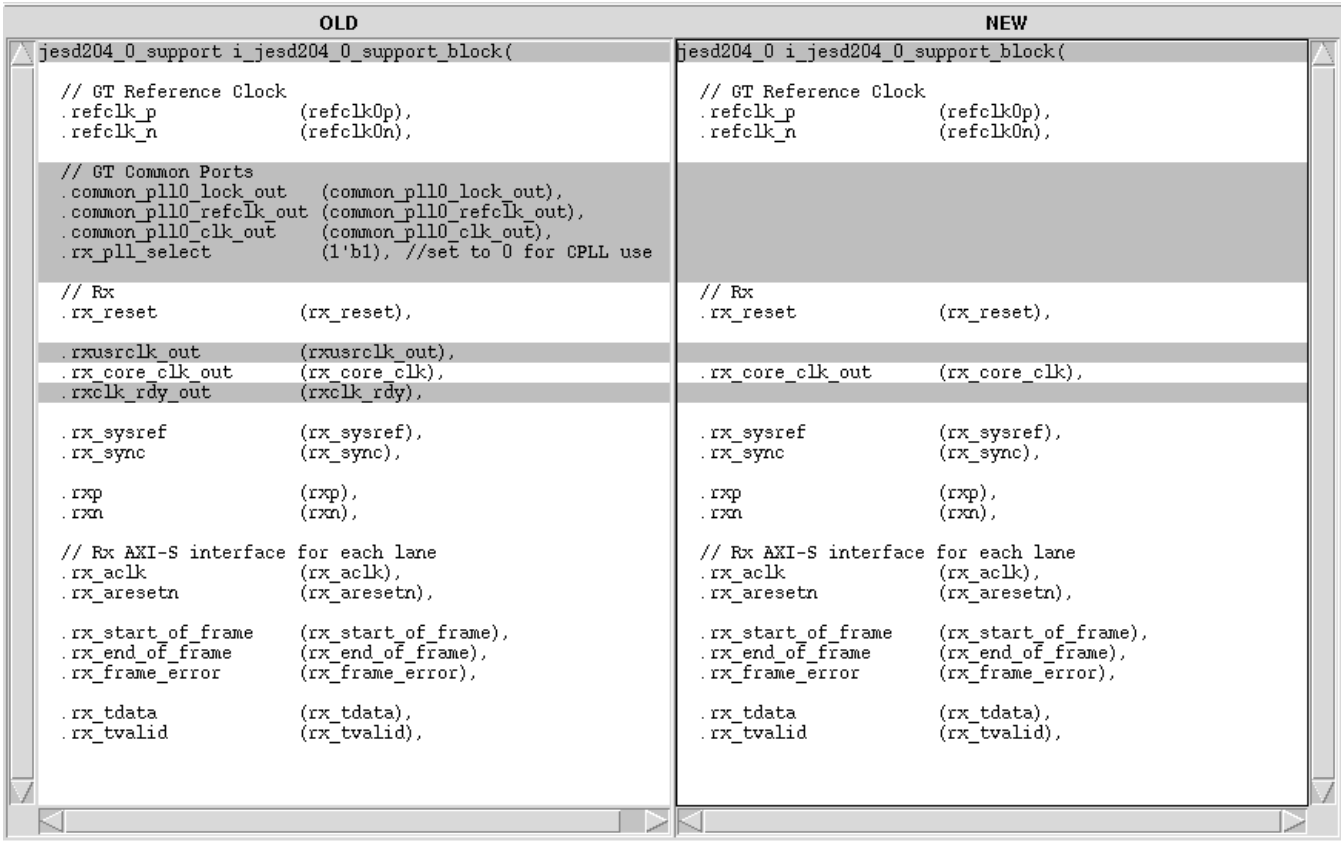
Figure C-3: **Upgrading Receiver with Shared Logic in Example Design**

Receiver with Shared Logic in Example Design Converted to Include Shared Logic in Core

IMPORTANT: *This is a more complicated upgrade, and if the transceiver logic has been modified from that provided with the example design, it is better to follow the steps for Transmitter with Shared Logic in Example Design.*

This is now possible because the transceiver parameters can be updated using the JESD204 IP GUI. To perform the upgrade:

1. Upgrade the JESD204 IP and ignore the critical warnings about pin changes.
2. Double-click the JESD204 XCI file to re-customize the IP.
3. Change from using shared logic in example design to shared logic in core in the Shared Logic Tab.
4. Check that the transceiver line rate and reference clock match the values used in your existing Transceiver Wizard instantiation.
5. Allow the IP to regenerate.
6. Modify the instantiation of jesd204_0_support in `jesd204_0_example_design.v` or your own RTL top-level that instantiates jesd204_0_support, as shown in Figure C-4.:



OLD

```
jesd204_0_support i_jesd204_0_support_block(

  // GT Reference Clock
  .refclk_p              (refclk0p),
  .refclk_n              (refclk0n),

  // GT Common Ports
  .common_pll0_lock_out   (common_pll0_lock_out),
  .common_pll0_refclk_out (common_pll0_refclk_out),
  .common_pll0_clk_out    (common_pll0_clk_out),
  .rx_pll_select          (1'b1), //set to 0 for CPLL use

  // Rx
  .rx_reset              (rx_reset),

  .rxusrclk_out          (rxusrclk_out),
  .rx_core_clk_out       (rx_core_clk),
  .rxclk_rdy_out         (rxclk_rdy),

  .rx_sysref             (rx_sysref),
  .rx_sync               (rx_sync),

  .rxp                   (rxp),
  .rxn                   (rxn),

  // Rx AXI-S interface for each lane
  .rx_aclk               (rx_aclk),
  .rx_aresetn            (rx_aresetn),

  .rx_start_of_frame     (rx_start_of_frame),
  .rx_end_of_frame       (rx_end_of_frame),
  .rx_frame_error        (rx_frame_error),

  .rx_tdata              (rx_tdata),
  .rx_tvalid             (rx_tvalid),
```

NEW

```
jesd204_0 i_jesd204_0_support_block(

  // GT Reference Clock
  .refclk_p              (refclk0p),
  .refclk_n              (refclk0n),

  // Rx
  .rx_reset              (rx_reset),

  .rx_core_clk_out       (rx_core_clk),

  .rx_sysref             (rx_sysref),
  .rx_sync               (rx_sync),

  .rxp                   (rxp),
  .rxn                   (rxn),

  // Rx AXI-S interface for each lane
  .rx_aclk               (rx_aclk),
  .rx_aresetn            (rx_aresetn),

  .rx_start_of_frame     (rx_start_of_frame),
  .rx_end_of_frame       (rx_end_of_frame),
  .rx_frame_error        (rx_frame_error),

  .rx_tdata              (rx_tdata),
  .rx_tvalid             (rx_tvalid),
```

Figure C-4: **Upgrading Receiver with Shared Logic in Ex. Des. Converted to Include Shared Logic in Core**

Transmitter with Shared Logic in Core

To upgrade this configuration to the v6.2 core, remove the following redundant pins from the core instantiation:

- common_pll0_lock_out
- common_pll0_refclk_out
- common_pll0_clk_out
- tx_pll_select
- txusrclk_out
- txclk_rdy_out

In addition, change the core instantiation as shown in Figure C-5.

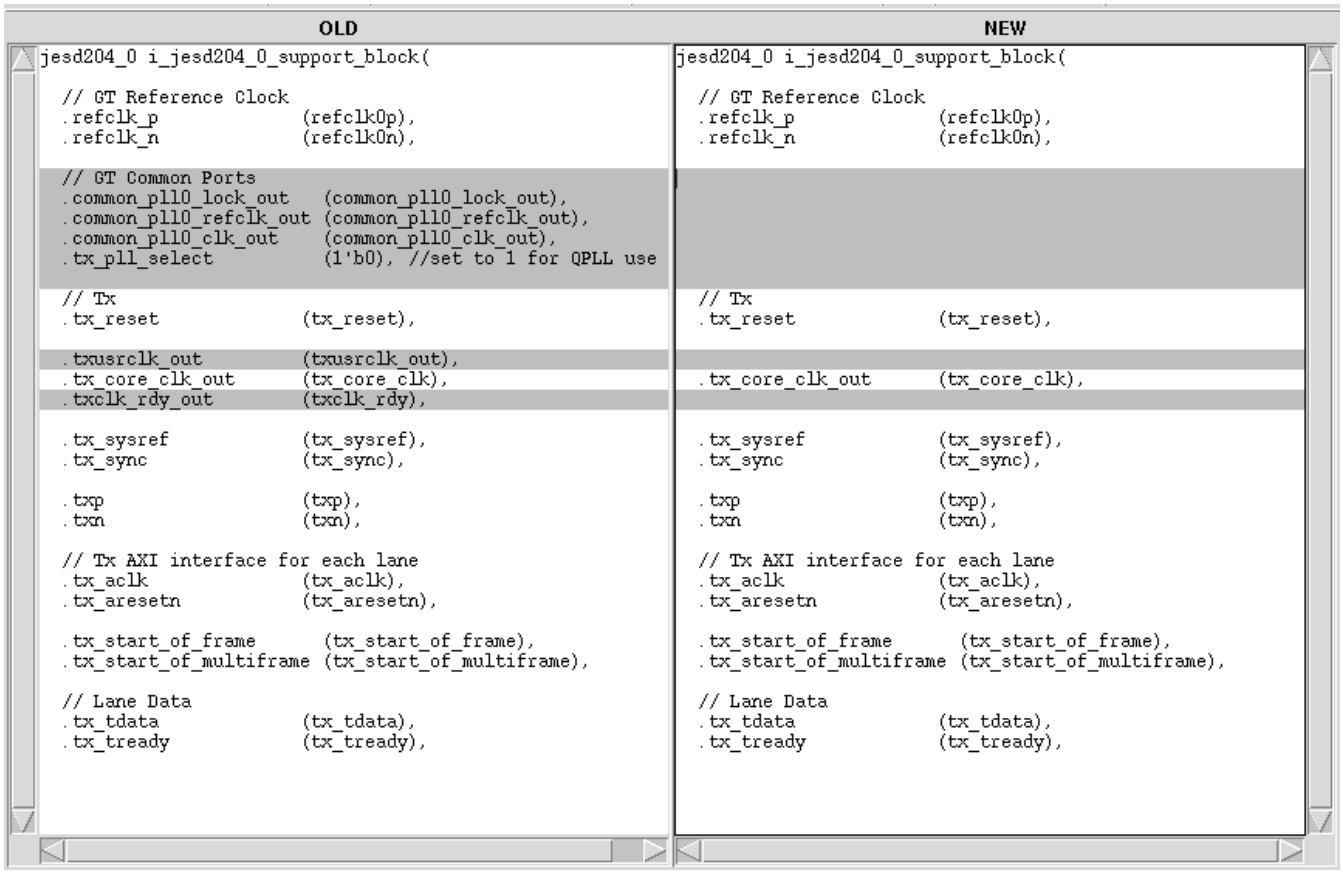


Figure C-5: **Upgrading Transmitter with Shared Logic in Core**

Receiver with Shared Logic in Core

To upgrade this configuration to the v6.2 core, remove the following redundant pins from the core instantiation:

- common_pll0_lock_out
- common_pll0_refclk_out
- common_pll0_clk_out
- rx_pll_select
- rxusrclk_out
- rxclk_rdy_out

In addition, change the core instantiation as shown in Figure C-6.

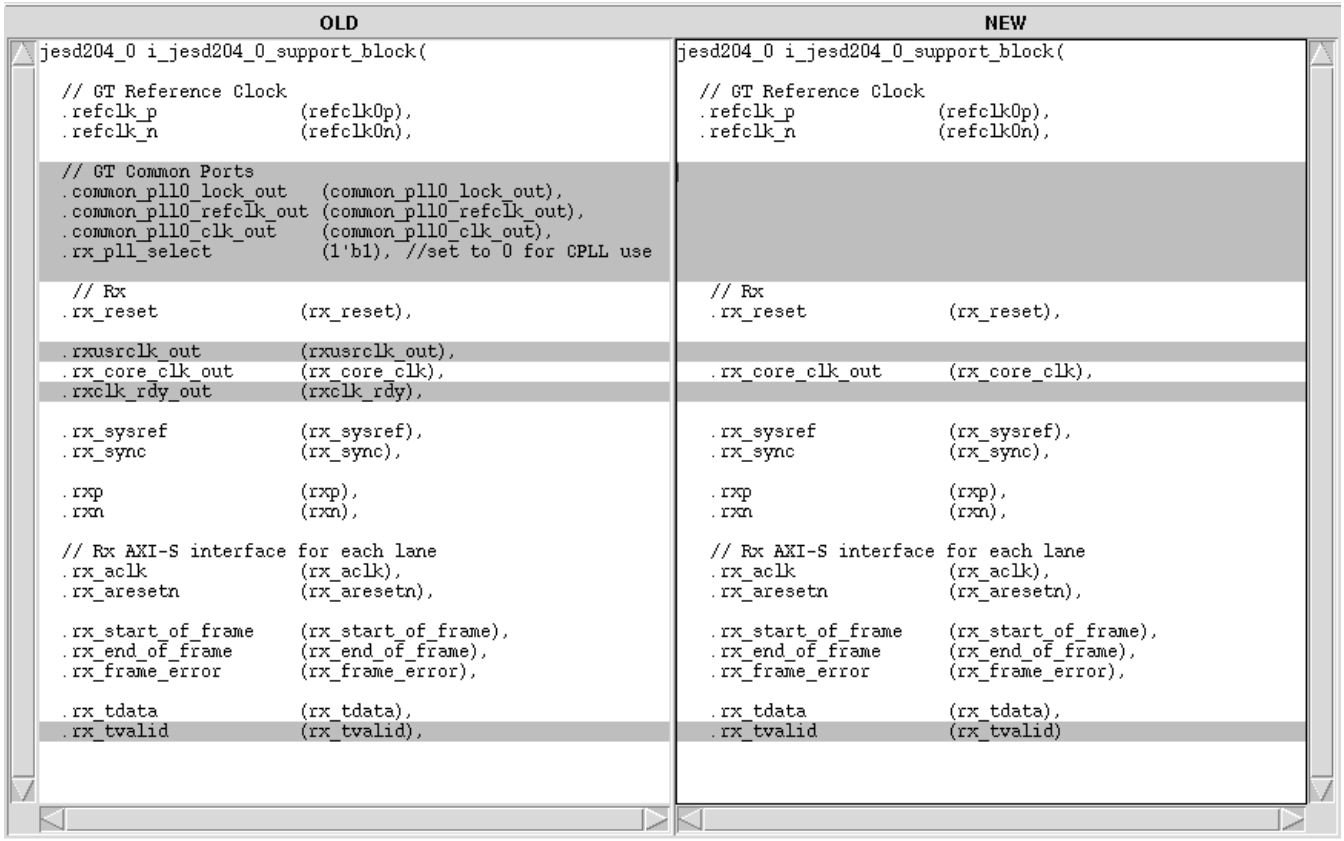


Figure C-6: **Upgrading Receiver with Shared Logic in Core**

Upgrading from JESD204 v5.1

- New Status registers have been added to support 8 to 12 lane configurations. See Register Space.
- New SYSREF handling control options for Subclass 1 operation have been added. See SYSREF Handling register.

Upgrading from JESD204 v5.0

Some additional transceiver control and status ports have been added if this option is selected in the IDE.

Upgrading from JESD204 v4.0

In the JESD204 v4.0 core the clocking logic was moved to the example design. This has been extended to include moving the transceiver modules into the example design in JESD204 v5.0. When "Include Shared Logic in example design" is selected when customizing the core the clocking, reset and transceiver logic is placed outside the core. Any modifications made to this example logic, to allow a single GTX transceiver to be used by both a transmit and receive JESD204 core, for example, are unaffected by future core upgrades.

When upgrading from a single JESD204 v4.0 core, select the option to "Include Shared Logic in core" to generate a JESD204 v5.0 core which most closely matches the previous version.

Register Map Changes

The register address map and register definitions have significantly changed from v4.0 core. See Register Space for the updated register map.

Receiver Port Changes

AXI4-Stream

- `rx0_tvalid`, `rx1_tvalid`, `rx2_tvalid`... have combined into a single pin called `rx_tvalid`
- `rx0_tdata`, `rx1_tdata`, `rx2_tdata`... have combined into a single bus called `rx_tdata` with width (32 * Number of Lanes)
- `rx0_tready` has been removed (it was not used in JESD204 v4.0 and only included on the core interface to comply with the AXI specification which has been updated to make `tready` optional

Clocks

The clock buffers are part of the shared logic which is now included in the core so the following changes must be made:

- `rxclk_rdy` was an input and is now an output (`rxclk_rdy_out`) when the core is generated with the shared logic; this is tied High inside the core unless the design uses an MMCM to generate the `userclk`.
- `rxuserclk` was an input and is now an output (`rxusrclk_out`) when the core is generated with the shared logic; in most cases this is the same as `rx_core_clk_out` (`rxusrclk2` on the GT) unless the design uses an MMCM to generate the `userclk` and `userclk2` at different frequencies.
- `rx_core_clk` was an input and is now an output (`rx_core_clk_out`); in most cases this is the same as `rxusrclk_out` (`rxusrclk` on the transceiver) unless the design uses an MMCM to generate the `userclk` and `userclk2` at different frequencies.
- `refclk` was a single ended signal driven by an IBUFDS_GTE2 primitive and is now a differential pair because the IBUFDS_GTE2 is included in the shared logic in the core.

Debug

`rxploarity` has been removed in the default configuration but can be included by enabling transceiver debug pins when customizing the core.

If transceiver debug signals were selected in the Vivado IDE then the pins listed in Table 2-12 are also new and can be left open if not required.

GT_COMMON

- `common_pll0_lock_out` has been added and can be left open unless the QPLL is shared with other cores using the same transceiver Quad Tile.
- `common_pll0_refclk_out` has been added and can be left open unless the QPLL is shared with other cores using the same transceiver Quad Tile.
- `common_pll0_clk_out` has been added and can be left open unless the QPLL is shared with other cores using the same transceiver Quad Tile.

PLL Selection

`rx_pll_select` has been added and should be driven High if the QPLL is being used and Low if the CPLL is being used. In JESD204 v4.0 this selection was made by modifying the core RTL directly after unlocking the core in the Vivado Design Suite.

Transmit Port Differences

AXI4-Stream

tx0_tvalid, tx1_tvalid, tx2_tvalid... have been removed as they were ignored by the core.

tx0_tdata, tx1_tdata, tx2_tdata... have combined into a single bus called tx_tdata with width (32 * Number of Lanes).

tx0_tready, tx0_tready, tx0_tready... have combined into a single pin called tx_tready.

Clocks

The clock buffers are part of the shared logic which is now included in the core so the following changes must be made:

- txclk_rdy was an input and is now an output (txclk_rdy_out) when the core is generated with shared logic; in most cases is tied High unless the design uses an MMCM to generate the userclk.
- txuserclk was an input and is now an output (txusrclk_out) when the core is generated with shared logic; in most cases this is the same as tx_core_clk_out (txusrclk2 on the transceiver) unless the design uses an MMCM to generate the userclk and userclk2 at different frequencies.
- tx_core_clk is an input when the core is generated without shared logic and an output (tx_core_clk_out) when the core is generated with shared logic; in most cases this is the same as txusrclk_out (txusrclk on the transceiver) unless the design uses an MMCM to generate the userclk and userclk2 at different frequencies.
- refclk was a single ended signal driven by an IBUFDS_GTE2 primitive and is now a differential pair because the IBUFDS_GTE2 is included in the shared logic in the core.

Debug

txpolarity, gt_txdata, gt_txcharisk, txpostcursor, txprecursor, txdiffctrl and txpolarity have been removed unless the optional transceiver debug signals are selected when generating the core. See Table 2-12.

If transceiver debug signals were selected in the GUI then the pins listed in Table 2-12 are also new and can be left open if not required.

GT_COMMON

- common_pll0_lock_out has been added and can be left open unless the QPLL is shared with other cores using the same transceiver Quad Tile.

- `common_pll0_refclk_out` has been added and can be left open unless the QPLL is shared with other cores using the same transceiver Quad Tile.
- `common_pll0_clk_out` has been added and can be left open unless the QPLL is shared with other cores using the same transceiver Quad Tile.

PLL Selection

`tx_pll_select` has been added when the core is generated with shared logic and should be driven High if the QPLL is being used and Low if the CPLL is being used. In JESD204 v4.0 this selection was made by modifying the core RTL directly after unlocking the core in the Vivado Design Suite.

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 JESD204, 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 JESD204. 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 JESD204 Core

AR: 54480

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 JESD204 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 23].

Reference Boards

Various Xilinx development boards support the JESD204. These boards can be used to prototype designs and establish that the core can communicate with the system.

- 7 series FPGA evaluation boards:
  - KC705
  - AC701
  - ZC706
  - VC709
- Kintex Ultrascale evaluation board:
  - KCU105

Simulation Debug

The simulation debug flow for QuestaSim is illustrated in Figure D-1. A similar approach can be used with other simulators.

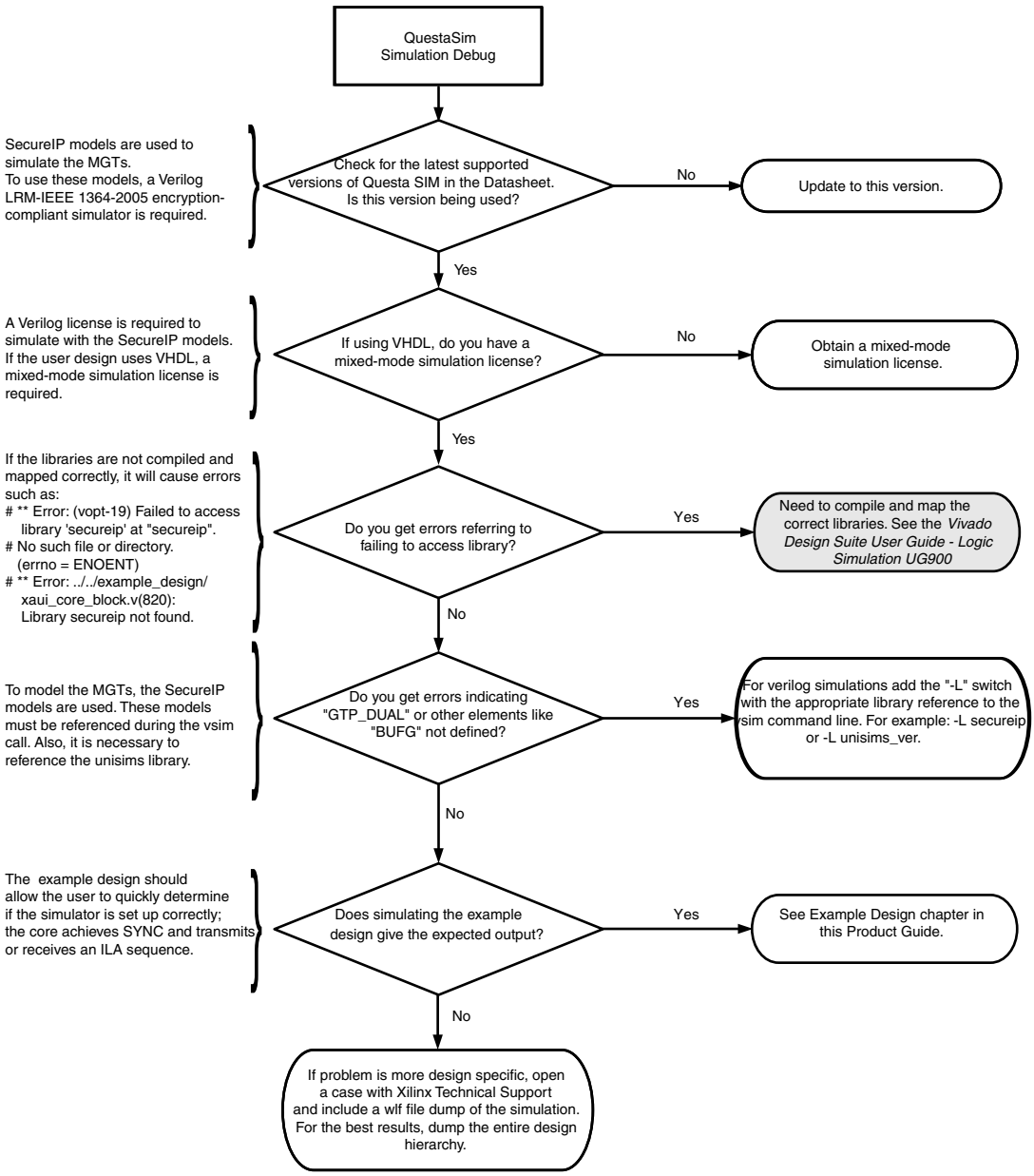


Figure D-1: **QuestaSim Debug Flow Diagram**

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

- 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 GTX/GTH/GTP transceiver requirements from the appropriate FPGA Data Sheet.
- Ensure that all GTX/GTH/GTP 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.
- Ensure that when regenerating a new GTX/GTH/GTP transceiver the reference clock of the new transceiver matches that of the design.

Issues Obtaining Lane Synchronization

- Ensure that the AXI4-Lite registers have been programmed with the correct values for the frame size parameters (octets per frame, frames per multiframe) and scrambling enable/disable so the transmitter matches the receiver.
- In the case of a receiver ensure that `SYNC` is deasserted (set to 1) when valid K28.5 (0xBC) characters are received from the GTX/GTH/GTP by monitoring `RXDATA` and `RXCHARISK` from the GTX/GTH/GTP using the debug feature.
- In the case of a transmitter ensure that core generates valid K28.5 (0xBC) characters to the GTX/GTH/GTP transceiver by monitoring `TXDATA` and `TXCHARISK` from the GTX/GTH/GTP transceiver using the debug feature until `SYNC` is deasserted (set to 1).

Issues Losing Synchronization Soon After Gaining Synchronization

- Ensure that the AXI4-Lite registers have been programmed with the correct values for F (Octets per Frame) and K (Frames per Multiframe).

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.

Additional Resources and Legal Notices

Xilinx Resources

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

References

These documents provide supplemental material useful with this product guide:

To search for JESD204 documentation, go to www.jedec.org

1. *Serial Interface for Data Converters* (JESD204B)
2. *AMBA AXI and ACE Protocol Specification* (ARM IHI 0022D)
3. *AMBA AXI4-Stream Protocol Specification* (ARM IHI 0051A)
4. *JESD204 PHY LogiCORE IP Product Guide* (PG198)
5. *7 Series FPGAs Configurable Logic Block User Guide* (UG474)
6. *UltraScale Architecture GTH Transceivers User Guide* (UG576)
7. *7 Series FPGAs GTX/GTH Transceivers User Guide* (UG476)
8. *7 Series FPGAs GTP Transceivers User Guide* (UG482)
9. *Artix-7 FPGAs Data Sheet: DC and Switching Characteristics* (DS181)
10. *Zynq-7000 All Programmable SoC (Z-7010, Z-7015, and Z-7020): DC and AC Switching Characteristics* (DS187)
11. *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* (DS182)
12. *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics* (DS183)
13. *Zynq-7000 All Programmable SoC (Z-7030, Z-7045, and Z-7100): DC and AC Switching Characteristics* (DS191)
14. *Kintex UltraScale Architecture Data Sheet: DC and Switching Characteristics* (DS892)

15. *Xilinx Vivado AXI Reference Guide* (UG1037)
16. *Vivado Design Suite User Guide: Designing with IP* (UG896)
17. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994)
18. *Vivado Design Suite User Guide: Getting Started* (UG910)
19. 7 Series data sheets
20. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
21. *ISE to Vivado Design Suite Migration Guide* (UG911)
22. *UltraScale Architecture Migration Methodology Guide* (UG1026)
23. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/18/2015	6.2	Added support for UltraScale+ families.
09/30/2015	6.2	• Added support for GTY transceivers. • Removed tx_aclk and rx_aclk from the AXI stream interface signals. Noted these signals may be removed in subsequent versions of the core. • Updated description on transceiver debug ports gt_rxpd and gt_txpd. • Updated description of lanes in use register. • Updated diagrams to show the new JESD PHY ports tx_sys_reset and rx_sys_reset. • Updated latency figures and calculations.

Date	Version	Revision
04/01/2015	6.1	• Added GT Port important note in Transceiver Control and Status Ports section. • Updated Fig. 2-2: Transmit Data Interface Timing for F = 8 and K = 4 for F = 8 to Fig. 2-4: Receive Data Interface Timing for F = 7. • Updated Table 2-14: Optional Transceiver Debug Ports (7 Series Devices) and Table 2-15: Optional Transceiver Debug Ports (UltraScale Architecture-Based Devices). • Added Lane 0 to 12 ID registers. • Updated Bit[16] default value to 1 in Table 2-18: Reset. • Updated Bit[31] description in Table 2-24: Link Error Status. • Updated Table 2-27: Lanes in Use. • Updated Bits[12:0] description in Table 2-29: RX Buffer Delay (RX Only). • Updated Table 2-30: Error Reporting (RX Only). • Updated description in Transceiver Sharing section. • Updated Table 3-4: Frequency Ranges for Kintex UltraScale Architecture-Based Devices. • Updated description in Clocking for Subclass 0 Mode section. • Added description in Receive Latency section. • Updated Table 3-6: Receive Datapath Latencies and Table 3-7: Transmit Datapath Latencies. • Updated the Sharing Transceivers between Transmit and Receive section. • Updated Figs. 4-1 to 4-3 in Design Flow Steps chapter. • Added JESD204 PHY Configuration Tab section. • Updated Table 4-1: Vivado IDE Parameter to User Parameter Relationship. • Added UNISIM important note in Simulation section. • Added Upgrading from JESD204 v6.0 section.
10/01/2014	6.0	Updated details about the example design and test bench.
06/04/2014	5.2	• Updated migration instructions. • Added Table 4-1 for Vivado IDE parameter to user parameter relationship.
04/02/2014	5.2	• Added core support for 12 lanes. • Updated Port Descriptions to clarify port differences for TX or RX cores and Shared Logic options. • Updated Register Space section to separate address map from register descriptions. • Restructured Chapter 3, Designing with the Core section and added introductory content. • Updated instructions for transceiver configuration update.
12/18/2013	5.1	Added UltraScale™ architecture support.

Date	Version	Revision
10/02/2013	5.0	• Revision number advanced to 5.0 to align with core version number. • Replaced option to generate shared core with option to include or exclude shareable logic resources in the core. • Added option to include or exclude RPAT and JSPAT modules. • Added optional transceiver control and status ports. • Removed GUI option for JESD204 subclass selection; subclass is now selected using a register. • AXI4-Lite address map has been updated. • A single AXI4-Stream bus is used for all txdata and rxdata lanes
03/20/2013	3.0	Updated to core version 4.0: • Hierarchy updated; block level now the default core top-level • AXI4-Lite address map corrections, including addition of byte write support • Added Artix®-7 support • Increase rx_buffer_adjust from 256 to 1024 • Pipeline stage added to receiver to improve timing • Zynq support added to HW demonstration platform
12/18/2012	2.0	Updated for 2012.4: • The core now supports 1, 2, 3, 4, 5, 6, 7 and 8 lane configurations in 7 series devices • Added 12.5 Gb/s line rate support • Removed JESD204A (new designs should use JESD204B Subclass 0) • Removed ISE • Added three new test modes • Added software lane select • Added a hardware demonstration design targeting the KC705 evaluation board, Appendix B, Hardware Demonstration Design • Updated Appendix D, Debugging
07/25/2012	1.0	Initial Xilinx release. This Product Guide is derived from DS814 and UG774.

Please Read: Important Legal Notices