Integrated Interlaken 150G v2.4

LogiCORE IP Product Guide

Vivado Design Suite

PG169 February 4, 2021

XILINX

Table of Contents

Introduction

The Xilinx® Integrated Interlaken LogiCORE™ IP is a scalable chip-to-chip interconnect protocol designed to enable the following for use in select UltraScale™ and UltraScale+™ architectures:

- The protocol logic supported in each integrated IP core scales up to 150 Gb/s.
- The protocol bypass (lane logic only) mode allows for 1-12 lanes up to 12.5G on UltraScale and up to 25.78125 Gb/s on UltraScale+ architecture on each serial transceiver to be used to build a fully featured Interlaken interface.

The integrated Interlaken IP core solution is designed to be compliant with *Interlaken Protocol Definition, Revision 1.2, October 7, 2008* [Ref 1]. This integrated IP core implements both the lane logic and protocol logic portions of the specification, which saves approximately 40 to 50k logic cells (LCs) per instantiation and uses about 1/8th the power of soft implementations.

Features

- A total bandwidth up to 150 Gb/s, available in the following configurations
  - 1 to 12 lanes x up to 12.5 Gb/s
  - 1 to 6 lanes x 12.5 Gb/s to 25.78125 Gb/s
- Data striping and de-striping across 1 to 12 lanes
- Retransmit protocol extension
- Lane decommissioning
- Supports both packet and burst interleaved modes

See Feature Summary in Chapter 1 for more features.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family(1)	Virtex® UltraScale+, Kintex® UltraScale+, Zynq® UltraScale+ Virtex UltraScale, Kintex UltraScale
Supported User Interfaces	Segmented local bus (LBUS)
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	Verilog
Example Design	Verilog
Test Bench	Verilog
Constraints File	Xilinx Design Constraints (XDC)
Simulation Model	Verilog
Supported S/W Driver	N/A
Tested Design Flows(2)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 58697
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list 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

This product guide describes the function and operation of the Xilinx® integrated IP core for Interlaken, including how to design, customize, and implement it. The integrated Interlaken IP core is a high-performance, low-power, flexible implementation of the Interlaken protocol, based on the Interlaken Protocol definition rev. 1.2. The Interlaken integrated IP core is a highly configurable integrated IP core that can support an overall bandwidth up to 150 Gb/s for protocol logic transmission.

The core instantiates the Interlaken integrated IP core found in UltraScale+™ and UltraScale™ devices. This core simplifies the design process and reduces time to market.

Using the latest serial transceiver technology and a flexible protocol layer, Interlaken minimizes the pin and power overhead of chip-to-chip interconnect and provides a scalable solution that can be used throughout an entire system. In addition, Interlaken uses two levels of cyclic redundancy check (CRC) and a synchronous data scrambler to ensure data integrity and link robustness. See Chapter 2, Product Specification for details on the core.

RECOMMENDED: *For the best results, previous experience building high performance, pipelined FPGA designs using Xilinx implementation software and constraint files is recommended.*

Navigating Content by Design Process

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

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

- Customizing and Generating the Core
- Example Design

Feature Summary

- Programmable BurstMax, BurstShort, and MetaFrameSize parameters
- 64B/67B encoding and decoding
- Automatic word and lane alignment
- Synchronous data scrambler
- Uses GTY or GTH transceivers for UltraScale+ and UltraScale devices
- 512-bit segmented LBUS user-side interface
- 64-bit interface to the serial transceiver
- CRC24 generation and checking for burst data integrity
- CRC32 generation and checking for lane data integrity
- Programmable rate limiting circuitry.
- Rate matching with a granularity of 1 Gb/s
- Robust error condition detection and recovery
- Channel-level and link-level flow control mechanism
- Support for up to 2,048 different logical channels
- BurstMax can be programmed up to 256 bytes
- Support for BurstShort minimum of 64 bytes and subsequent increments of 32 bytes
- Support for up to 256 different In-Band flow control channels and 2048 out-of-band flow control channels
- Support for link-level flow control
- Meta frame length programmable between 128 to 8K words
- Support for status messaging
- Dynamic reconfiguration port (DRP) interface for dynamic reconfiguration of the core
- Protocol bypass (lane logic only) mode. See IP Facts.

Applications

The integrated IP core for Interlaken offers system designers a risk-free and quick path for adopting Interlaken as their chip-to-chip interconnect protocol. Typical applications include:

- Media Access Control (MAC)-to-Interlaken bridging (for example, 100GE, nx40GE, nx10GE)
- Interlaken switch with 100GE granularity

Unsupported Features

The integrated IP core for Interlaken does not support Look-Aside mode.

Licensing and Ordering

This Xilinx® LogiCORE™ IP core is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License. Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page.

For more information on Interlaken 150G and to generate a no-charge license key, visit the UltraScale/UltraScale+ Interlaken page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

The integrated IP core for Interlaken is a single core capable of up to 150 Gb/s. The core connects to the serial transceivers at defined rates up to 12.5 Gb/s with GTH transceivers and up to 25.78125 Gb/s with GTY transceivers.

Table 2-1 defines the integrated IP core for Interlaken solutions.

Table 2-1: **Integrated IP Core for Interlaken Solutions**

Lane Width	Line Rate	SerDes	SerDes Width
x1 – x12	Up to 12.5 Gb/s	GTH/GTY	64-bit
x1 – x6	Up to 25.78125 Gb/s	GTY	64-bit

The Interlaken core internally instantiates the Interlaken integrated IP core (ILKN). The core also instantiates GTH/GTY and an example of how the two integrated IP cores are connected together, along with the reset and clocking for those integrated IP cores.

Figure 2-1 illustrates the following interfaces to the Interlaken integrated IP core.

- Serial transceiver interface
- User-side LBUS interface
- Lane logic bus interface
- Status/Control interface
- DRP interface used for configuration

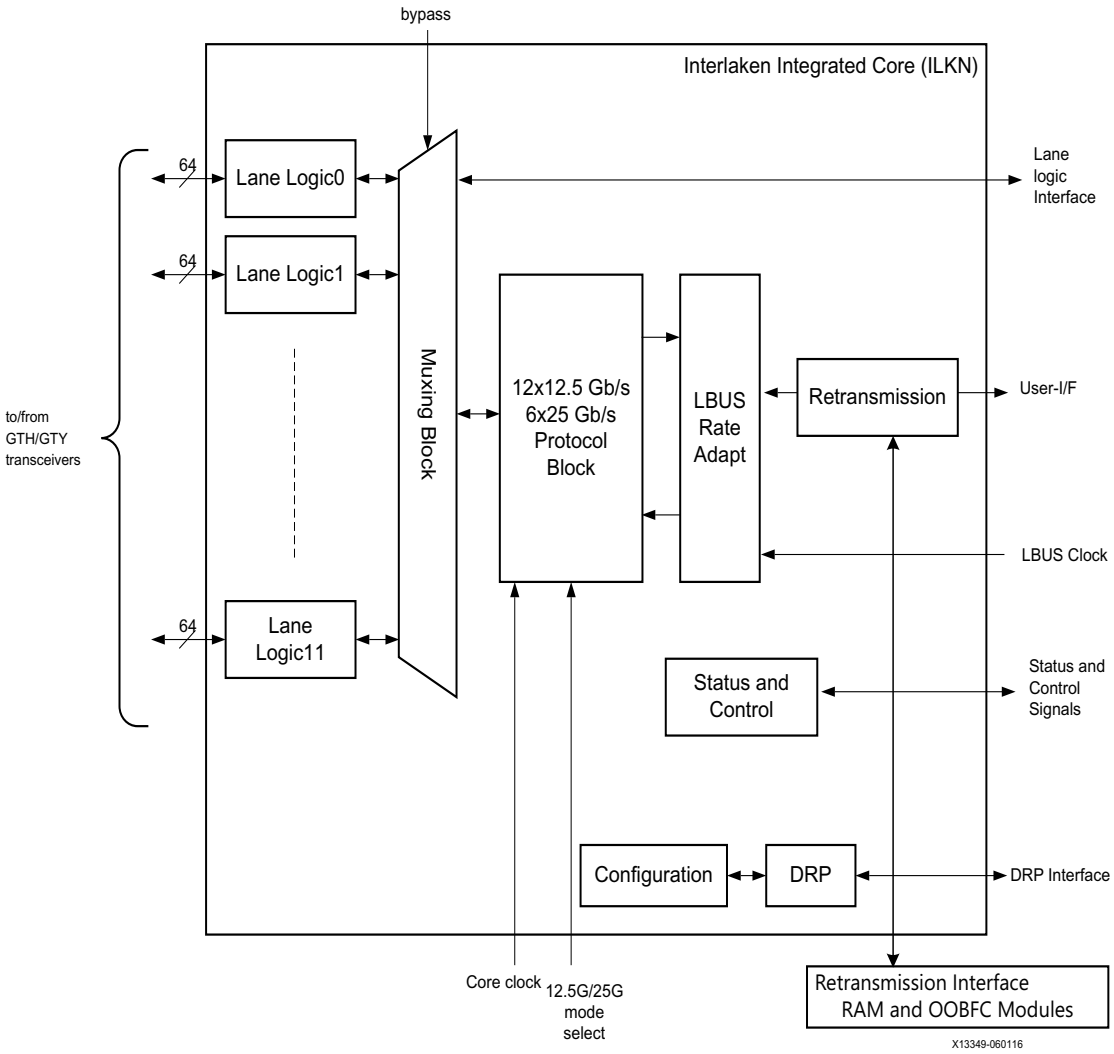


Figure 2-1: **Block Diagram of the Interlaken Integrated IP Core**

Typical Operation

The integrated IP core for Interlaken handles all protocol related functions to communicate to the other devices Interlaken interface. All handshaking, synchronizing, and error checking are handled by the Interlaken IP core. You can provide packet data through the local bus (LBUS) TX interface and receive it from the LBUS RX interface. The LBUS is designed to match packet bus protocols made common by the SPI4.2 protocol; a detailed description is provided in Chapter 3, Designing with the Core.

The Interlaken IP core is designed to be as flexible as possible and to be used in many different applications. As such, the Interlaken IP core provides all of the flexibility offered by the Interlaken protocol. Flow control information is automatically extracted by the RX path of the Interlaken IP core. You must monitor the flow control information and ensure proper data transmission through the core. Also, the Interlaken IP core TX path consists of a single pipeline with a single memory buffer. You must build the enhanced scheduling algorithm block external to the Interlaken IP core.

The following is an example:

The operation steps after the Interlaken IP core is powered up are as follows:

1. After the device is powered up and the reset procedure completed, the Interlaken IP core TX path starts transmitting Control/Idle words to align and synchronize the receiving device Interlaken interface. Similarly, the Interlaken IP core RX path receives Control/Idle words and completes its own synchronization procedure.
2. You must set all of the flow control inputs to the Interlaken IP core TX path to the XOFF state to prevent any real data transfer.
3. The RX path becomes synchronized and aligned, and signals the user logic that alignment is complete. You can then turn the flow control information from XOFF to XON for any of the channels that are ready to accept data.
4. When the other device is ready to receive data, it sends XON information to the Interlaken IP core. The Interlaken IP core signals the user logic which channels can be used for data transmission.

These steps provide a simple and easily implemented procedure for using the Interlaken IP core. You build a scheduler to multiplex data among the different logical channels and use the flow control information output by the Interlaken IP core to manage the scheduling function. You do not need to be concerned with any of the lower level Interlaken protocol details.

Standards

The Interlaken integrated IP core is compliant with the *Interlaken Protocol Definition, Revision 1.2, October 7, 2008* [Ref 1] and the Interlaken Retransmit Extension Protocol Definition, Revision 1.2.

Performance and Resource Utilization

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

Port Descriptions

Table 2-2 describes the UltraScale+™ and UltraScale™ device Interlaken (ILKN) primitive ports.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports**

Name	Direction	Clock Domain	Description
Transceiver I/O			
RX_SERDES_DATA0[63:0]	Input	rx_serdes_clk[0]	Data bus from the serial transceiver macros for lane0. There are 12 rx_serdes_data buses; one bus for each serial transceiver lane and each bus has 64 bits. By definition, bit [63] is the first bit received by the Interlaken core. Bit [0] is the last bit received.
RX_SERDES_DATA1[63:0]	Input	rx_serdes_clk[1]	Data bus from the serial transceiver macros for lane1.
RX_SERDES_DATA2[63:0]	Input	rx_serdes_clk[2]	Data bus from the serial transceiver macros for lane2.
RX_SERDES_DATA3[63:0]	Input	rx_serdes_clk[3]	Data bus from the serial transceiver macros for lane3.
RX_SERDES_DATA4[63:0]	Input	rx_serdes_clk[4]	Data bus from the serial transceiver macros for lane4.
RX_SERDES_DATA5[63:0]	Input	rx_serdes_clk[5]	Data bus from the serial transceiver macros for lane5.
RX_SERDES_DATA6[63:0]	Input	rx_serdes_clk[6]	Data bus from the serial transceiver macros for lane6.
RX_SERDES_DATA7[63:0]	Input	rx_serdes_clk[7]	Data bus from the serial transceiver macros for lane7.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
RX_SERDES_DATA8[63:0]	Input	rx_serdes_clk[8]	Data bus from the serial transceiver macros for lane8.
RX_SERDES_DATA9[63:0]	Input	rx_serdes_clk[9]	Data bus from the serial transceiver macros for lane9.
RX_SERDES_DATA10[63:0]	Input	rx_serdes_clk[10]	Data bus from the serial transceiver macros for lane10.
RX_SERDES_DATA11[63:0]	Input	rx_serdes_clk[11]	Data bus from the serial transceiver macros for lane11.
TX_SERDES_DATA0[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane0. There are 12 tx_serdes_data buses; one bus for each serial transceiver lane and each bus has 64 bits. By definition, bit [63] is the first bit transmitted by the Interlaken core. Bit [0] is the last bit transmitted.
TX_SERDES_DATA1[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane1.
TX_SERDES_DATA2[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane2.
TX_SERDES_DATA3[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane3.
TX_SERDES_DATA4[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane4.
TX_SERDES_DATA5[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane5.
TX_SERDES_DATA6[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane6.
TX_SERDES_DATA7[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane7.
TX_SERDES_DATA8[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane8.
TX_SERDES_DATA9[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane9.
TX_SERDES_DATA10[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane10.
TX_SERDES_DATA11[63:0]	Output	tx_serdes_refclk	Data bus to the serial transceiver macros for lane11.
RX_SERDES_CLK[11:0]	Input		Recovered clock of each serial transceiver lane. The rx_serdes_data bus for each lane is synchronized to the positive edge of the corresponding bit of this bus.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
RX_SERDES_RESET[11:0]	Input	Async	Reset for each RX serial transceiver lane. The recovered clock for each serial transceiver lane has associated with it an active-High reset. This signal should be 1 whenever the associated recovered clock is not operating at the correct frequency. The RX_SERDES_RESET signals should be held in reset until the GT (also known as serial transceiver) initialization is complete and the clocks are stable.
TX_SERDES_REFCLK	Input		Reference clock for the TX datapath. This clock must be frequency locked to the TX SERDES clock inputs. Typically, the same reference clock used to drive the TX serial transceiver is connected to this input.
TX_SERDES_REFCLK_RESET	Input	Async	Reset for TX Reference clock. This active-High signal should 1 whenever the tx_serdes_refclk input is not operating at the correct frequency as indicated by the transceiver phase-locked (PLL) lock signal.
LBUS Interface – Clock/Reset/Control Signals			
CORE_CLK	Input		300/412 MHz core clock. The minimum core clock frequency is 300 MHz for 12 x 12.5 Gb/s mode.
LBUS_CLK	Input		Rate-adapting FIFO clock for the user side logic. LBUS signals are synchronized to this clock.
RX_RESET	Input	Async	Asynchronous reset for the RX circuits. This signal is active-High (1= reset) and must be held High until all of the clocks for the RX path are fully active. These clocks are CORE_CLK, LBUS_CLK and rx_serdes_clk[11:0]. The Interlaken core handles synchronizing the rx_reset input to the appropriate clock domains within the core.
TX_RESET	Input	Async	Asynchronous reset for the TX circuits. This signal is active-High (1= reset) and must be held High until all of the clocks for the TX path are fully active. These clocks are CORE_CLK, LBUS_CLK, and tx_serdes_refclk. The Interlaken core handles synchronizing the tx_reset input to the appropriate clock domains within the core.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
RX_OVFOUT	Output	lbus_clk	Receive LBUS overflow. If this signal is asserted, it means that the LBUS clock is too slow for the incoming data stream. The LBUS bandwidth must be greater than the Interlaken bandwidth.
RX_DATAOUT0[127:0]	Output	lbus_clk	Receive segmented LBUS data for segment0. The value of the bus is only valid in cycles in which rx_enout0 is sampled as 1.
RX_DATAOUT1[127:0]	Output	lbus_clk	Receive segmented LBUS Data for segment1.
RX_DATAOUT2[127:0]	Output	lbus_clk	Receive segmented LBUS Data for segment2.
RX_DATAOUT3[127:0]	Output	lbus_clk	Receive segmented LBUS Data for segment3.
RX_CHANOUT0[10:0]	Output	lbus_clk	Receive channel number for segment0. The bus indicates the channel number of the in-flight packet and is only valid in cycles in which rx_enout0 is sampled as 1. The maximum number of channels is programmed by the ctl_rx_chan_ext pin. See that pin description for the encoding of that signal.
RX_CHANOUT1[10:0]	Output	lbus_clk	Receive channel number for segment1.
RX_CHANOUT2[10:0]	Output	lbus_clk	Receive channel number for segment2.
RX_CHANOUT3[10:0]	Output	lbus_clk	Receive channel number for segment3.
RX_ENAOUT0	Output	lbus_clk	Receive LBUS enable for segment0. This signal qualifies the other signals of the RX segmented LBUS Interface. Signals for segment0 of the RX LBUS interface are only valid in cycles in which rx_enaout0 is sampled as 1.
RX_ENAOUT1	Output	lbus_clk	Receive LBUS enable for segment1.
RX_ENAOUT2	Output	lbus_clk	Receive LBUS enable for segment2.
RX_ENAOUT3	Output	lbus_clk	Receive LBUS enable for segment3.
RX_SOPOUT0	Output	lbus_clk	Receive LBUS start of packet (SOP) for segment0. This signal indicates the SOP when it is sampled as a 1 and is only valid in cycles in which rx_enaout0 is sampled as a 1.
RX_SOPOUT1	Output	lbus_clk	Receive LBUS SOP for segment1.
RX_SOPOUT2	Output	lbus_clk	Receive LBUS SOP for segment2.
RX_SOPOUT3	Output	lbus_clk	Receive LBUS SOP for segment3.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
RX_EOPOUT0	Output	lbus_clk	Receive LBUS EOP for segment0. This signal indicates the end of packet (EOP) when it is sampled as a 1 and is only valid in cycles in which rx_enaout0 is sampled as a 1.
RX_EOPOUT1	Output	lbus_clk	Receive LBUS EOP for segment1.
RX_EOPOUT2	Output	lbus_clk	Receive LBUS EOP for segment2.
RX_EOPOUT3	Output	lbus_clk	Receive LBUS EOP for segment3.
RX_ERROUT0	Output	lbus_clk	Receive LBUS Error for segment0. This signal indicates that the current packet being received has an error when it is sampled as a 1. This signal is only valid in cycles when both rx_enaout0 and rx_eopout0 are sampled as a 1. When this signal is a value of 0, it indicates that there is no error in the packet being received.
RX_ERROUT1	Output	lbus_clk	Receive LBUS Error for segment1.
RX_ERROUT2	Output	lbus_clk	Receive LBUS Error for segment2.
RX_ERROUT3	Output	lbus_clk	Receive LBUS Error for segment3.
RX_MTYOUT0[3:0]	Output	lbus_clk	Receive LBUS Empty for segment0. This bus indicates how many bytes of the rx_dataout0 bus are empty or invalid for the last transfer of the current packet. This bus is only valid in cycles when both rx_enaout0 and rx_eopout0 are sampled as 1. When rx_errout0 and rx_enaout0 are sampled as 1, the value of rx_mtyout0[2:0] is always 000. Other bits of rx_mtyout0 are as usual.
RX_MTYOUT1[3:0]	Output	lbus_clk	Receive LBUS Empty for segment1.
RX_MTYOUT2[3:0]	Output	lbus_clk	Receive LBUS Empty for segment2.
RX_MTYOUT3[3:0]	Output	lbus_clk	Receive LBUS Empty for segment3.
LBUS Interface – TX Path Signals			
TX_RDYOUT	Output	lbus_clk	Transmit LBUS Ready. This signal indicates whether the Interlaken core TX path is ready to accept data and provides back-pressure to the user logic. A value of 1 means the user logic can pass data to the core. A value of 0 means the user logic must stop transferring data to the core. When tx_rdyout is asserted depends on a pre-determined value of FIFO fill.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
TX_OVFOUT	Output	lbus_clk	Transmit LBUS Overflow. This signal indicates whether you have violated the back pressure mechanism provided by the tx_rdyout signal. If tx_ovfout is sampled as a 1, a violation has occurred. You must design the rest of the user logic to prevent the overflow of the TX interface.
TX_DATAIN0[127:0]	Input	lbus_clk	Transmit segmented LBUS Data for segment0. This bus receives input data from the user logic. The value of the bus is captured in every cycle that tx_enain0 is sampled as 1.
TX_DATAIN1[127:0]	Input	lbus_clk	Transmit segmented LBUS Data for segment1.
TX_DATAIN2[127:0]	Input	lbus_clk	Transmit segmented LBUS Data for segment2.
TX_DATAIN3[127:0]	Input	lbus_clk	Transmit segmented LBUS Data for segment3.
TX_CHANIN0[10:0]	Input	lbus_clk	Transmit LBUS channel number for segment0. This bus receives the channel number for the packet being written. The value of the bus is captured in every cycle that tx_enain0 is sampled as 1. The maximum number of channels is programmed by the ctl_tx_chan_ext pin. See that pin description for the encoding of that signal.
TX_CHANIN1[10:0]	Input	lbus_clk	Transmit LBUS channel number for segment1.
TX_CHANIN2[10:0]	Input	lbus_clk	Transmit LBUS channel number for segment2.
TX_CHANIN3[10:0]	Input	lbus_clk	Transmit LBUS channel number for segment3.
TX_ENAIN0	Input	lbus_clk	Transmit LBUS enable for segment0. This signal is used to enable the TX LBUS Interface. Signals for segment0 of the TX LBUS interface are sampled only in cycles in which tx_enain0 is sampled as 1
TX_ENAIN1	Input	lbus_clk	Transmit LBUS enable for segment1.
TX_ENAIN2	Input	lbus_clk	Transmit LBUS enable for segment2.
TX_ENAIN3	Input	lbus_clk	Transmit LBUS enable for segment3.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
TX_SOPIN0	Input	lbus_clk	Transmit LBUS SOP for segment0. This signal is used to indicate the SOP when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles in which **`tx_enain0`** is sampled as a 1.
TX_SOPIN1	Input	lbus_clk	Transmit LBUS SOP for segment1.
TX_SOPIN2	Input	lbus_clk	Transmit LBUS SOP for segment2.
TX_SOPIN3	Input	lbus_clk	Transmit LBUS SOP for segment3.
TX_EOPIN0	Input	lbus_clk	Transmit LBUS EOP for segment0. This signal is used to indicate the EOP when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles in which tx_enain0 is sampled as a 1.
TX_EOPIN1	Input	lbus_clk	Transmit LBUS EOP for segment1.
TX_EOPIN2	Input	lbus_clk	Transmit LBUS EOP for segment2.
TX_EOPIN3	Input	lbus_clk	Transmit LBUS EOP for segment3.
TX_ERRIN0	Input	lbus_clk	Transmit LBUS Error for segment0. This signal is used to indicate a packet contains an error when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles in which tx_enain0 and tx_eopin0 are sampled as 1.
TX_ERRIN1	Input	lbus_clk	Transmit LBUS Error for segment1.
TX_ERRIN2	Input	lbus_clk	Transmit LBUS Error for segment2.
TX_ERRIN3	Input	lbus_clk	Transmit LBUS Error for segment3.
TX_MTYIN0[3:0]	Input	lbus_clk	Transmit LBUS Empty for segment0. This bus is used to indicate how many bytes of the tx_datain bus are *empty* or *invalid* for the last transfer of the current packet. This bus is sampled only in cycles that tx_enain0 and tx_eopin0 are sampled as 1. When tx_eopin0 and tx_errin0 are sampled as 1, the value of tx_mtyin0[2:0] is ignored and treated as if it was 000. The other bits of tx_mtyin0 are used as usual.
TX_MTYIN1[3:0]	Input	lbus_clk	Transmit LBUS Empty for segment1.
TX_MTYIN2[3:0]	Input	lbus_clk	Transmit LBUS Empty for segment2.
TX_MTYIN3[3:0]	Input	lbus_clk	Transmit LBUS Empty for segment3.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
TX_BCTLIN0	Input	lbus_clk	Transmit force insertion of Burst Control word for segment0. This input is used to force the insertion of a Burst Control Word. When tx_bctlin0 and tx_enain0, are sampled as 1, a Burst Control word is inserted before the data on the tx_datain0 bus is transmitted even if one is not required to observe the BurstMax parameter. This input is used by external schedulers that wish to reduce bandwidth lost due to observation of the BurstShort parameter. (See Use of TX_BCTLIN.) The use of an enhanced scheduling algorithm as described in the Interlaken Protocol Definition 1.2 is required.
TX_BCTLIN1	Input	lbus_clk	Transmit force insertion of Burst Control word for segment1.
TX_BCTLIN2	Input	lbus_clk	Transmit force insertion of Burst Control word for segment2.
TX_BCTLIN3	Input	lbus_clk	Transmit force insertion of Burst Control word for segment3.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
LBUS Interface - TX Path Control/Status Signals			
CTL_TX_ENABLE	Input	lbus_clk	TX Enable. This signal is used to enable the transmission of data when it is sampled as a 1. When sampled as a 0, only Idle Control Words (and the Meta Frame Words) are transmitted by the Interlaken core. This input should not be set to 1 until the receiver it is sending data to (the receiver in the other device) is fully aligned and ready to receive data. Otherwise, loss of data can occur. This input can be used for Link-Level flow control. For example, setting this input to 0 halts transmission of data and results in the entire link going into XOFF state.
CTL_TX_FC_STAT[255:0]	Input	lbus_clk	TX In-Band Flow Control Input. These signals are used to set the status for each calendar position in the in-band-flow control mechanism (see the Interlaken Protocol specification [Ref 1]). A value of 1 means XON, a value of 0 means XOFF. These bits are transmitted in the Interlaken Control Word bits [55:40]. Each bit of ctl_tx_fc_stat represents an entry in the flow control calendar with a length of 256 entries. When bit 56 of a Control Word is a value of 1, the first 16 calendar entries are output on bits 55-40. Specifically, Bit 55 of the first Control Word reflects the state of ctl_tx_fc_stat[0], bit 54 reflects the state of ctl_tx_fc_stat[1], bit 53 reflects the state of ctl_tx_fc_stat[2], etc. as explained in the example in section 5.3.4.1 of Interlaken spec 1.1. Subsequent Control Words contain the next 16 calendar entries and so forth. This input must be synchronous with LBUS_CLK.
CTL_TX_MUBITS[7:0]	Input	lbus_clk	TX Multiple-Use Control Bits. This bus contains the "Multi-Use" field of the Interlaken Control (see the Interlaken Protocol specification [Ref 1]). The value of the bus is transmitted in the Interlaken Control Word bits[31:24]. You must define the function of this bus. If the bus is not used, set all bits to 0.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
CTL_TX_RLIM_ENABLE	Input	lbus_clk	TX Rate Limiter Enable. This signal is used to enable the Rate Limiter. A value of 1 turns on the Rate Limiter and a value of 0 turns off the Rate Limiter.
CTL_TX_RLIM_MAX[11:0]	Input	lbus_clk	TX Rate Limiter Maximum Token Count. This bus is used to set the maximum number of tokens in the bucket. A token is equal to 1 byte.
CTL_TX_RLIM_DELTA[11:0]	Input	lbus_clk	TX Rate Limiter Delta. This bus is used to set the number of tokens to add to the bucket each interval. A token is equal to 1 byte.
CTL_TX_RLIM_INTV[7:0]	Input	lbus_clk	TX Rate Limiter Update Interval. This bus is used to set the number of LBUS clock cycles between additions of ctl_tx_rlim_delta tokens to the bucket.
CTL_TX_DIAGWORD_LANESTAT[11:0]	Input	Async	Lane Status messaging inputs. This bus sets bit 33 in the Diagnostic Word for the respective lane. See Appendix A in the Interlaken Revision 1.2 specification [Ref 1].
CTL_TX_DIAGWORD_INTFSTAT	Input	Async	Interface Status messaging inputs. This signal sets bit 32 in the Diagnostic Word of each lane. See Appendix A in the Interlaken Revision 1.2 specification [Ref 1].
STAT_TX_UNDERFLOW_ERR	Output	lbus_clk	TX Underflow. This signal indicates if the LBUS interface is being clocked too slowly to properly fill the link with data. In normal operation, this signal is always sampled as 0. If this signal is sampled as 1, the clocks are not set to proper frequencies and must be fixed.
STAT_TX_OVERFLOW_ERR	Output	lbus_clk	TX Overflow. This output should never be asserted and indicates a critical failure. The core needs to be reset. This output is synchronous with the LBUS_CLK.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
STAT_TX_BURST_ERR	Output	lbus_clk	TX BurstShort Error. When this signal is a value of 1, a burst (that is, a sequence of Data Words between two Control Words) was shorter than the value specified by ctl_tx_burstshort. This signal is only asserted if the final Control Word did not contain an EOP. This signal is provided to identify a poor scheduler design that results in reduced LBUS transaction errors. The TX core must be reset if this signal is asserted.
LBUS Interface - RX Path Control/Status Signals			
CTL_RX_FORCE_RESYNC	Input	Async	RX Resync input. This signal is used to force the RX lane logic to reset, re-synchronize, and realign. A value of 1 forces the reset operation. A value of 0 allows normal operation. This input should normally be Low and should only be pulsed (1 cycle minimum pulse) to force realignment. ***Note:*** CTL_RX_FORCE_RESYNC restarts the synchronization state machine but doesn't reset the GT logic. In most cases when there is an RX failure, the GT RX need to be reset.
STAT_RX_BURSTMAX_ERR	Output	lbus_clk	RX BurstMax Error. When this signal is a value of 1, a burst (that is, a sequence of Data Words between two Control Words) was detected that was longer than the value of BurstMax specified by ctl_rx_burstmax. This signal is informational only and can be optionally ignored.
STAT_RX_DIAGWORD_LANESTAT[11:0]	Output	lbus_clk	Lane Status messaging outputs. This bus reflects the most recent value in bit 33 of the Diagnostic Word received on the respective lane. These bits should only be considered valid if the respective bit in stat_rx_crc32_valid is a value of 1. See Appendix A in the Interlaken Revision 1.2 specification [Ref 1].

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
STAT_RX_DIAGWORD_INTFSTAT[11:0]	Output	lbus_clk	Lane Status messaging outputs. This bus reflects the most recent value in bit 32 of the Diagnostic Word received on the respective lane. These bits should only be considered valid if the respective bit in stat_rx_crc32_valid is a value of 1. See Appendix A in the Interlaken Revision 1.2 specification [Ref 1].
STAT_RX_CRC32_VALID[11:0]	Output	lbus_clk	Diagnostic Word CRC32 Valid. This bus reflects the validity of the CRC32 in the most recently received Diagnostic Word for the respective lane. A value of 1 indicated the CRC32 was valid and a value of 0 indicated the CRC32 was invalid. See section 5.4.6 of the Interlaken Revision 1.2 specification [Ref 1].
STAT_RX_CRC32_ERR[11:0]	Output	lbus_clk	Diagnostic Word CRC32 Error/Invalid. This bus provides indication of an invalid CRC32 in the Diagnostic Word for the respective lane. These signals are asserted with a value of 1 for one LBUS clock cycle each time an error is detected.
STAT_RX_FC_STAT[255:0]	Output	lbus_clk	RX Flow control Outputs. These signals indicate the flow control status for all of the calendar positions of the received data. A value of 1 means XON, a value of 0 means XOFF. These outputs reflect the information contained in bits 55-40 of the Control Words received by the RX. Only 256 In-Band flow control bits are supported. If a longer calendar is received, latter bits are ignored and are never output. If a shorter calendar is received, the bits of stat_rx_fc_stat that were not updated maintain their previous state. Each bit of stat_rx_fc_stat represents a received flow control calendar entry. Stat_rx_fc_stat[0] is the first received calendar entry, stat_rx_fc_stat[1] is the second received calendar entry, stat_rx_fc_stat[2] is the third received calendar entry, etc. as explained in the example in section 5.3.4.1 Interlaken spec 1.2. Whenever a CRC24 or a loss of lane alignment occurs, all bits of stat_rx_fc_stat are set to a value of 0. This output is synchronous with the LBUS_CLK.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
STAT_RX_MUBITS[7:0]	Output	lbus_clk: if retransmission is disabled. core_clk: if retransmission is enabled.	RX Multiple-Use Control Bits. This bus contains the "Multi-Use" field of the Interlaken Control (see the Interlaken Protocol specification [Ref 1]). The value of the bus are bits[31:24] of the most recently received Interlaken Control Word.
STAT_RX_SYNCED[11:0]	Output	lbus_clk	Word Boundary Synchronized. These signals indicate whether a lane is word boundary synchronized. A value of 1 indicates the corresponding lane has achieved word boundary synchronization as follows: a) 64B/67B Word Boundary Locked, b) Correctly receiving the Meta Frame Synchronization Word, and c) Correctly receiving the Scrambler State Control Word as described in sections 5.4.2, 5.4.3, and 5.4.4 of Interlaken spec 1.1. This output is synchronous with LBUS_CLK.
STAT_RX_SYNCED_ERR[11:0]	Output	lbus_clk	Word Boundary Synchronization Error. These signals indicate whether an error occurred during word boundary synchronization in the respective lane. A value of 1 indicates the corresponding lane had a word boundary synchronization error.
STAT_RX_MF_LEN_ERR[11:0]	Output	lbus_clk	Meta Frame Length Error. These signals indicate whether a Meta Frame length mismatch occurred in the respective lane. A value of 1 indicates the corresponding lane is receiving Meta Frame of wrong length.
STAT_RX_MF_REPEAT_ERR[11:0]	Output	lbus_clk	Meta Frame Consecutive Error. These signals indicate whether consecutive Meta Frame errors occurred in the respective lane. A value of 1 indicates an error in the corresponding lane.
STAT_RX_DESCRAM_ERR[11:0]	Output	lbus_clk	Scrambler State Control Word Error. These signals indicate a mismatch between the received Scrambler State Word and the expected value. A value of 1 indicates an error in the corresponding lane.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
STAT_RX_ALIGNED	Output	lbus_clk	All Lanes Aligned/De-Skewed. This signal indicates whether or not all lanes are aligned and de-skewed. A value of 1 indicates all lanes are aligned and de-skewed. When this signal is a 1, the RX path is aligned and can receive packet data.
STAT_RX_ALIGNED_ERR	Output	lbus_clk	Loss of Lane Alignment/De-Skew. This signal indicates an error occurred during lane alignment or lane alignment was lost. A value of 1 indicates an error occurred.
STAT_RX_CRC24_ERR	Output	lbus_clk	Control Word CRC24 Error. This signal indicates whether or not a mismatch occurred between the received and the expected CRC24 value. A value of 1 indicates a mismatch occurred.
STAT_RX_OVERFLOW_ERR	Output	lbus_clk	RX FIFO Overflow Error. This signal indicates if the LBUS interface is being clocked too slowly to properly receive the data being transmitted across the link. A value of 1 indicates an error occurred. In normal operation, this signal is always sampled as 0. If this signal is sampled as 1, the clocks are not set to proper frequencies and must be fixed.
STAT_RX_MF_ERR[11:0]	Output	lbus_clk	Meta Frame Synchronization Word Error. These signals indicate that an incorrectly formed Meta Frame Synchronization Word was detected in the respective lane. A value of 1 indicates an error occurred.
STAT_RX_FRAMING_ERR[11:0]	Output	lbus_clk	Framing Error. These signals indicate that an illegal framing pattern was detected in the respective lane. A value of 1 indicates an error occurred.
STAT_RX_MSOP_ERR	Output	lbus_clk	Missing SOP Error. This signal indicates that a Missing SOP was detected (and corrected).
STAT_RX_MEOP_ERR	Output	lbus_clk	Missing EOP Error. This signal indicates that a Missing EOP was detected (and corrected).
STAT_RX_BURST_ERR	Output	lbus_clk	Burst Error. This signal indicates that a BurstShort or a burst length error was detected.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
STAT_RX_MISALIGNED	Output	lbus_clk	Alignment Error. This signal indicates that the lane aligner did not receive the expected Meta Frame Synchronization Word across all (active) lanes. This signal can be used to collect the statistic "RX_Alignment_Error" as described in Table 5-9 of the Interlaken specification. This signal is not asserted until the Meta Frame Synchronization Word has been received at least once across all lanes. A value of 1 indicates the error occurred.
STAT_RX_BAD_TYPE_ERR[11:0]	Output	lbus_clk	Unexpected or Illegal Meta Frame Control Word Block Type. These signals indicate an unexpected or illegal Meta Frame Control Word Block Type was detected. These signals can be used to collect the statistic "RX_Bad_Control_Error" as described in Table 5-9 of the Interlaken specification. A value of 1 indicates an error in the corresponding lane.
STAT_RX_MUBITS_UPDATED	Output	lbus_clk: if retransmission is disabled. core_clk: if retransmission is enabled.	RX Multiple-Use/General Purpose Control Bits Updated. This output indicates that STAT_RX_MUBITS has been updated and is asserted for one clock cycle.
STAT_RX_WORD_SYNC[11:0]	Output	lbus_clk	64B/67B Word Boundary Locked. These signals indicate whether a lane is 64B/67B word boundary locked. A 64B/67B word boundary lock occurs if a lane detects 64 consecutive valid framing patterns on bits[65:64] as per the Interlaken Specification 1.2 Section 5.4.2. These signals are independent of both the Meta Frame Synchronization Word and Scrambler State Control Word. A value of 1 indicates the corresponding lane has achieved 64B/67B word boundary lock.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
Protocol Bypass (Lane Logic Only) Interface - RX Path Signals			
RX_BYPASS_RDIN	Input	core_clk	This signal initiates a read operation.
RX_BYPASS_FORCE_REALIGNIN	Input	core_clk	This signal causes the word synchronizer to sync again.
RX_BYPASS_IS_AVAILOUT[11:0]	Output	core_clk	This signal indicates whether the data can be read using the rx_bypass_rdin signal.
RX_BYPASS_DATAOUT0[65:0]	Output	core_clk	Data or control word output for bypass lane0.
RX_BYPASS_DATAOUT1[65:0]	Output	core_clk	Data or control word output for bypass lane1.
RX_BYPASS_DATAOUT2[65:0]	Output	core_clk	Data or control word output for bypass lane2.
RX_BYPASS_DATAOUT3[65:0]	Output	core_clk	Data or control word output for bypass lane3.
RX_BYPASS_DATAOUT4[65:0]	Output	core_clk	Data or control word output for bypass lane4.
RX_BYPASS_DATAOUT5[65:0]	Output	core_clk	Data or control word output for bypass lane5.
RX_BYPASS_DATAOUT6[65:0]	Output	core_clk	Data or control word output for bypass lane6.
RX_BYPASS_DATAOUT7[65:0]	Output	core_clk	Data or control word output for bypass lane7.
RX_BYPASS_DATAOUT8[65:0]	Output	core_clk	Data or control word output for bypass lane8.
RX_BYPASS_DATAOUT9[65:0]	Output	core_clk	Data or control word output for bypass lane9.
RX_BYPASS_DATAOUT10[65:0]	Output	core_clk	Data or control word output for bypass lane10.
RX_BYPASS_DATAOUT11[65:0]	Output	core_clk	Data or control word output for bypass lane11.
RX_BYPASS_ENAOUT[11:0]	Output	core_clk	This signal qualifies the corresponding rx_bypass_dataout bus.
RX_BYPASS_IS_BADLYFRAMEDOUT[11:0]	Output	core_clk	This signal identifies the metaframe words that must be discarded for each lane.
RX_BYPASS_IS_SYNCWORDOUT[11:0]	Output	core_clk	This signal identifies metaframe synchronization words.
RX_BYPASS_IS_OVERFLOWOUT[11:0]	Output	core_clk	This signal indicates whether the lane buffer has overflowed.
RX_BYPASS_IS_SYNCEDOUT[11:0]	Output	core_clk	This signal indicates that the corresponding lane is synced to metaframe and is ready for alignment.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
Protocol Bypass (Lane Logic Only) Interface - TX Path Signals			
TX_BYPASS_ENAIN	Input	tx_serdes_refclk	This signal qualifies the TX inputs.
TX_BYPASS_GEARBOX_SEQIN[7:0]	Input	tx_serdes_refclk	This signal determines the gearbox sequencing according to a pre-determined format.
TX_BYPASS_MFRAMER_STATEIN[3:0]	Input	tx_serdes_refclk	This signal determines the metaframe state according to a pre-determined format.
TX_BYPASS_CTRLIN[11:0]	Input	tx_serdes_refclk	This signal identifies a word as being either data or control for the corresponding lane.
TX_BYPASS_DATAIN0[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane0.
TX_BYPASS_DATAIN1[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane1.
TX_BYPASS_DATAIN2[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane2.
TX_BYPASS_DATAIN3[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane3.
TX_BYPASS_DATAIN4[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane4.
TX_BYPASS_DATAIN5[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane5.
TX_BYPASS_DATAIN6[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane6.
TX_BYPASS_DATAIN7[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane7.
TX_BYPASS_DATAIN8[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane8.
TX_BYPASS_DATAIN9[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane9.
TX_BYPASS_DATAIN10[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane10.
TX_BYPASS_DATAIN11[63:0]	Input	tx_serdes_refclk	This bus is the data (or control) words for lane11.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
Retransmission Interface			
CTL_TX_RETRANS_ENABLE	Input	Static	TX Retransmission Enable. This signal enables the retransmission feature for the Transmit path. When this input is a value of 1, the input CTL_TX_MUBITS is ignored. This input is static and only changed during reset.
CTL_TX_ERRINJ_BITERR_GO	Input	lbus_clk	When this input is assigned a value of 1 for a single Local bus clock cycle, it initiates the generation of a single bit error. The error will occur on the selected lane only once. Once CTL_TX_ERRINJ_BITERR_GO has been assigned a value of 1 for a single Local bus clock cycle, it should not be assigned a value of 1 again until STAT_TX_ERRINJ_BITERR_DONE has changed from a value of 1 to a value of 0 to a value of 1 again.
STAT_TX_ERRINJ_BITERR_DONE	Output	lbus_clk	Initially this output has a value of 1 to indicate that the error injector is ready to inject a new error. After CTL_TX_ERRINJ_BITERR_GO is asserted, this signal is negated and attains a value of 0. When all of the errors have been injected, this output attains a value of 1 again.
CTL_TX_ERRINJ_BITERR_LANE[3:0]	Input	lbus_clk	This input is used to select which lane the error will be injected on. A valid, commissioned lane must be selected. This input is pseudo-static and should only be changed when CTL_TX_ERRINJ_BITERR_GO has a value of 0 and STAT_TX_ERRINJ_BITERR_DONE has a value of 1.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
CTL_TX_RETRANS_REQ	Input	lbus_clk	Request Input. This input is used to request retransmission and is ignored if: • CTL_TX_RETRANS_ENABLE has a value of 0, • CTL_TX_RETRANS_REQ_VALID has a value of 0, or • There is insufficient data written into the buffer. Also, in order for a request for retransmission to be accepted, a clock cycle with CTL_TX_RETRANS_REQ_VALID set to a value of 1 with a request set to a value of 0 must occur before a clock cycle with CTL_TX_RETRANS_REQ_VALID set to a value of 1 with request set to a value of 1. This is in keeping with the requirement in section 2.2.1 of the Interlaken Retransmission Protocol Extension v1.2 which states: "The bit in the calendar must be deasserted, set to Xoff, before another retransmit request can be signaled." This input is intended to be connected to one of the rx_fc bits on the RX Out-of-Band flow control interface.
CTL_TX_RETRANS_REQ_VALID	Input	lbus_clk	Request Valid Input. This input is used to qualify the input request as described previously. This input is intended to be connected to the rx_update bit that corresponds to the selected rx_fc bit on the RX Out-of-Band flow control interface.
STAT_TX_RETRANS_RAM_WDATA[644-1:0]	Output	lbus_clk	TX Retransmission buffer write data
STAT_TX_RETRANS_RAM_WE_B0	Output	lbus_clk	TX Retransmission buffer write enable for RAM Bank 0
STAT_TX_RETRANS_RAM_WE_B1	Output	lbus_clk	TX Retransmission buffer write enable for RAM Bank 1
STAT_TX_RETRANS_RAM_WE_B2	Output	lbus_clk	TX Retransmission buffer write enable for RAM Bank 2
STAT_TX_RETRANS_RAM_WE_B3	Output	lbus_clk	TX Retransmission buffer write enable for RAM Bank 3
STAT_TX_RETRANS_RAM_WADDR[9-1:0]	Output	lbus_clk	TX Retransmission buffer write address
CTL_TX_RETRANS_RAM_RDATA[644-1:0]	Input	lbus_clk	TX Retransmission buffer read data
STAT_TX_RETRANS_RAM_RD_B0	Output	lbus_clk	TX Retransmission buffer read enable for RAM Bank 0

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
STAT_TX_RETRANS_RAM_RD_B1	Output	lbus_clk	TX Retransmission buffer read enable for RAM Bank 1
STAT_TX_RETRANS_RAM_RD_B2	Output	lbus_clk	TX Retransmission buffer read enable for RAM Bank 2
STAT_TX_RETRANS_RAM_RD_B3	Output	lbus_clk	TX Retransmission buffer read enable for RAM Bank 3
STAT_TX_RETRANS_RAM_RADDR[9-1:0]	Output	lbus_clk	TX Retransmission buffer read address
STAT_TX_RETRANS_RAM_RSEL[1:0]	Output	lbus_clk	TX Retransmission Read Data Select. This input is used to select the read data from the corresponding RAM bank and latch the data into CTL_RX_RETRANS_RAM_RDATA. This signal is valid two clock cycles after the read signal of the corresponding RAM bank is asserted. The following values are defined: 00 = Select read data from RAM Bank 0 01 = Select read data from RAM Bank 1 10 = Select read data from RAM Bank 2 11 = Select read data from RAM Bank 3
CTL_TX_RETRANS_RAM_PERRIN	Input	lbus_clk	Retransmission read data parity error input. This input is for external parity logic that checks the validity of data read from the retransmission buffer. When this input has a value of 1, it causes the forwarding of data to halt, and causes the STAT_TX_RETRANS_RAM_PERROUT signal to be asserted. When this input is unused, it must be tied to a value of 0.
STAT_TX_RETRANS_BURST_ERR	Output	lbus_clk	TX Retransmission BurstShort error. This output is asserted with a value of 1 if a burst, less than BurstShort, not ending with an EOP, is written into the TX when CTL_TX_RETRANS_ENABLE has a value of 1.
STAT_TX_RETRANS_BUSY	Output	lbus_clk	TX Retransmission Buffer Busy. When this output has a value of 1, it indicates the buffer is retransmitting data.
STAT_TX_RETRANS_RAM_PERROUT	Output	lbus_clk	Retransmission read data parity error output. This output is asserted whenever CTL_TX_RETRANS_RAM_PERRIN is asserted and indicates that a fatal error has occurred and the TX path has been halted. A reset is required to clear this signal.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
CTL_RX_RETRANS_ENABLE	Input	Static	RX Retransmission Enable. This signal enables the retransmission feature for the Receive path when asserted. This input should be static and only changed during reset.
STAT_RX_RETRANS_REQ	Output	lbus_clk	Interlaken RX Retransmission Request. When enabled, this output requests an external TX to retransmit recent bursts because a burst sequence error of some sort was detected. This output changes from 0 to a value of 1 and remains at 1 while the input CTL_RX_RETRANS_ACK has a value of 0. If CTL_RX_RETRANS_ACK is tied to a value of 1, this output will only be a value of 1 for a single LBUS cycle.
CTL_RX_RETRANS_ACK	Input	lbus_clk	Interlaken RX Retransmission Acknowledge. This input is used to clear the signal stat_rx_retrans_req. When both ctl_rx_retrans_ack and stat_rx_retrans_req have a value of 1, the output stat_rx_retrans_req is forced to a value of 0.
STAT_RX_RETRANS_STATE[2:0]	Output	core_clk	Interlaken RX Retransmission State. These outputs indicate the state of the RX Retransmission logic: • 000 = waiting for initial bursts • 001 = normal operating mode • 010 = an error was detected and the short timer is counting • 011 = waiting for a discontinuity • 100 = waiting for a new sequence • 101 = waiting for the expected sequence • 110 = error detected and waiting for assertion of CTL_RX_RETRANS_RESET • 111 = waiting for negation of CTL_RX_RETRANS_RESET
STAT_RX_RETRANS_SEQ[7:0]	Output	core_clk	Interlaken RX Retransmission Primary Sequence Number. When CTL_RX_RETRANS_ENABLE has a value of 1, these outputs indicate the value of bits 31:24 of the Burst Control Word most recently forwarded to the LBUS interface.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
STAT_RX_RETRANS_SUBSEQ[4:0]	Output	core_clk	Interlaken RX Retransmission Sub-Sequence Number. When CTL_RX_RETRANS_ENABLE has a value of 1, these outputs indicate the implied sub-sequence of the Burst Control Word most recently forwarded to the LBUS interface. ***Note:*** The values of these bits might not be correct when STAT_RX_RETRANS_STATE=000.
STAT_RX_RETRANS_SEQ_UPDATED	Output	core_clk	Interlaken RX Retransmission Sequence Number Updated. When CTL_RX_RETRANS_ENABLE has a value of 1, this output indicates that STAT_RX_RETRANS_SEQ has been updated and is asserted for one clock cycle.
CTL_RX_RETRANS_RESET	Input	lbus_clk	Interlaken RX Retransmission Reset. When this input has a value of 1, the RX retransmission logic is reset. This is not a general purpose input but only has an effect if a fatal retransmission error has been detected.
CTL_RX_RETRANS_RESET_MODE	Input	lbus_clk	Interlaken RX Retransmission Reset Mode. This input only has an effect with the negation of CTL_RX_RETRANS_RESET. If this input has a value of 0, with the negation CTL_RX_RETRANS_RESET, the RX will mark open packets as having an error and wait for the expected sequence. If this input has a value of 1, with the negation CTL_RX_RETRANS_RESET, the RX will mark open packets as having an error and establish a new sequence.
STAT_RX_RETRANS_RETRY_ERR	Output	core_clk	Interlaken RX Retransmission Retry Error. This output indicates too many re-requests for a particular sequence that occurred as defined by CTL_RX_RETRANS_RETRY. This output changes from a value of 0 to a value of 1 when the fatal error condition is detected and remains asserted until the input CTL_RX_RETRANS_RESET is asserted and negated.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
STAT_RX_RETRANS_WRAP_ERR	Output	core_clk	Interlaken RX Retransmission Wrap Around Error. This output indicates that after a request for retransmission was made, the expected sequence arrived but was preceded by too many other bursts suggesting that the "true" burst to be received was lost. This output changes from a value of 0 to a value of 1 when the error condition is detected and remains asserted until the input CTL_RX_RETRANS_RESET is asserted and negated.
STAT_RX_RETRANS_WDOG_ERR	Output	core_clk	Interlaken RX Retransmission Watchdog Timer Error. This output indicates the counter associated with CTL_RX_RETRANS_WDOG has reached its maximum value. This output changes from a value of 0 to a value of 1 when the fatal error condition is detected and remains asserted until the input CTL_RX_RETRANS_RESET is asserted and negated.
CTL_RX_RETRANS_ERRIN	Input	lbus_clk	Interlaken RX Retransmission Forced Error. This input is used to force a fatal error. When asserted, the input CTL_RX_RETRANS_RESET must be asserted and negated. If unused, this input must be tied to a value of 0.
STAT_RX_RETRANS_DISC	Output	core_clk	Interlaken RX Retransmission Discontinuity. When CTL_RX_RETRANS_ENABLE has a value of 1, this output indicates that a discontinuity was detected.
STAT_RX_RETRANS_CRC24_ERR	Output	core_clk	Interlaken RX Retransmission CRC24 Error. When CTL_RX_RETRANS_ENABLE has a value of 1, this output indicates that a CRC24 error was detected. T
CTL_RX_RETRANS_FORCE_REQ	Input	lbus_clk	Interlaken RX Retransmission Forced Request. This input is used to force a request for retransmission and only does two things: • Asserts STAT_RX_RETRANS_REQ and • Clears the timer associated with STAT_RX_RETRANS_LATENCY This input should only be asserted for one clock cycle.

Table 2-2: **UltraScale+ and UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Clock Domain	Description
STAT_RX_RETRANS_LATENCY[15:0]	Output	lbus_clk	Interlaken RX Retransmission Request to Discontinuity Latency. This timer is used to measure the latency from request for retransmission to the receipt of the associated discontinuity in terms of Interlaken Words. When STAT_RX_RETRANS_REQ is asserted, STAT_RX_RETRANS_LATENCY is set to 0. When the next discontinuity is received, STAT_RX_RETRANS_LATENCY is updated with the number of Interlaken Words that have been received. If no discontinuity is received, STAT_RX_RETRANS_LATENCY is set to 'hFFFF.
DRP Path/Control Signals			
DRP_DO[15:0]	Output	DRP_CLK	Data bus for reading configuration data from the ILKN to the FPGA logic resources.
DRP_RDY	Output	DRP_CLK	Indicates operation is complete for write operations and data is valid for read operations.
DRP_ADDR[9:0]	Input	DRP_CLK	DRP Address Bus.
DRP_CLK	Input		DRP Interface Clock. When DRP is not used, this can be tied to GND.
DRP_DI[15:0]	Input	DRP_CLK	Data bus for writing configuration data from the FPGA logic resources to the Interlaken core.
DRP_EN	Input	DRP_CLK	DRP Enable Signal. • 0: No read or write operations performed. • 1: Enables a read or write operation. For write operations, DRP_WE and DRP_EN should be driven High for one DRP_CLK cycle only.
DRP_WE	Input	DRP_CLK	DRP Write Enable. • 0: Read operation when DRP_EN is 1. • 1: Write operation when DRP_EN is 1. For write operations, DRP_WE and DRP_EN should be driven High for one DRP_CLK cycle only.

Attribute Descriptions

Table 2-3 provides a detailed description for the UltraScale+ and UltraScale device integrated Interlaken core attributes and their default values. These attributes are configured through the DRP interface. These attributes should be static and only changed during reset.

Table 2-3: **UltraScale+ and UltraScale Device Integrated Interlaken Core Attributes**

Name	Type	Description	Default Value
LBUS Interface – Clock/Reset/Control Attributes			
MODE	Boolean	This attribute selects between 6 x 25 Gb/s and 12 x 12.5 Gb/s operation of the protocol logic. TRUE: Selects 12x12.5 Gb/s. FALSE: Selects 6x25 Gb/s.	TRUE
BYPASS	Boolean	This attribute controls whether Interlaken is operating in protocol block bypass (lane logic only) mode or not. TRUE: Enables protocol bypass (lane logic only) mode. FALSE: Disables protocol bypass (lane logic only) mode.	FALSE
LBUS Interface – TX Path Control/Status Attributes			
CTL_TX_FC_CALLEN[7-1:0]	7-bit Hex	TX Flow Control Calendar Length Input. This input controls the number of bits of ctl_tx_fc_stat that are actually used. The settings (in decimal) for calendar length are as follows: 7'h0= 16 entries; 7'h1 = 32 entries; 7'h3 = 64 entries; 7'h7 = 128 entries; 7'hF = 256 entries. All other values are reserved and must not be used.	7'h00

Table 2-3: **UltraScale+ and UltraScale Device Integrated Interlaken Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_TX_BURSTMAX[1:0]	2-bit Hex	Interlaken TX BurstMax. This bus sets the BurstMax parameter for the TX as follows: 2'h0 = 64 bytes 2'h1 = 128 bytes 2'h2 = 192 bytes 2'h3 = 256 bytes The burst size selected by CTL_TX_BURSTMAX must be greater than or equal to the burst size selected by CTL_TX_BURSTSHORT.	2'h3
CTL_TX_BURSTSHORT[2:0]	3-bit Hex	Interlaken TX BurstShort. This bus sets the BurstShort parameter for the TX as follows: 3'h0 = not valid 3'h1 = 64 bytes 3'h2 = 96 bytes 3'h3 = 128 bytes 3'h4 = 160 bytes 3'h5 = 192 bytes 3'h6 = 224 bytes 3'h7 = 256 bytes The burst size selected by CTL_TX_BURSTSHORT must be less than or equal to the burst size selected by CTL_TX_BURSTMAX.	3'h1
CTL_TX_DISABLE_SKIPWORD	Boolean	Skip Word Injection Deletion. As required by the Interlaken specification, a skip word is inserted once per Meta Frame to permit clock compensation through a repeater function. If the Interlaken core is not transmitting through an intermediary device, but is simply transmitting to an "ultimate" receiver such as another Interlaken core, this attribute can be set to TRUE. See section 5.4.7 of the Interlaken specification, revision 1.2 [Ref 1].	TRUE
CTL_TX_MFRAMELEN_MINUS1[15:0]	16-bit Hex	TX Meta Frame Length minus one. This bus sets the MetaFrameLength parameter for the TX and should be set to the desired length minus 1. For example, to set the MetaFrame length to 2,048, set this bus to 2,047 (decimal). MetaFrame length is defined as the number of Interlaken words. Each Interlaken word is 8 bytes (64 bits). The minimum value for CTL_TX_MFRAMELEN_MINUS1 is 255 or 0xFF.	16'h07FF

Table 2-3: **UltraScale+ and UltraScale Device Integrated Interlaken Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_TX_LAST_LANE[3:0]	4-bit Hex	This attribute is used for lane decommissioning of the TX and indicates the last active lane.	4'hB
LBUS Interface – RX Path Control/Status Attributes			
CTL_RX_PACKET_MODE	Boolean	RX Packet Mode Error Handling. This attribute changes the way the error handler in the RX path processes errors. TRUE: Packet mode. Assumes packets are arriving as complete packets. FALSE: Burst Interleaved mode. Assumes packets are arriving in bursts interleaved from different channels. Use of this attribute ensures that packets delivered to the LBUS have the appropriate SOP and EOP pairing.	TRUE
CTL_RX_BURSTMAX[1:0]	2-bit Hex	Interlaken RX BurstMax. This bus set the BurstMax parameter for the RX as follows: 2'h0 = 64 bytes 2'h1 = 128 bytes 2'h2 = 192 bytes 2'h3 = 256 bytes These inputs are only used in conjunction with STAT_RX_BURSTMAX_ERR.	2'h3
CTL_RX_MFRAMELEN_MINUS1[15:0]	16-bit Hex	RX Meta Frame Length minus one. This bus sets the MetaFrameLength parameter for the RX and should be set to the desired length minus 1. For example, to set the MetaFrame length to 2,048, set this bus to 2,047 (decimal). MetaFrame length is defined as the number of Interlaken words. Each Interlaken word is 8 bytes (64 bits). The minimum value for CTL_RX_MFRAMELEN_MINUS1 is 255 or 0xFF.	16'h07FF
CTL_RX_LAST_LANE[3:0]	4-bit Hex	This attribute is used for lane decommissioning of the RX and indicates the last active lane.	4'hB

Table 2-3: **UltraScale+ and UltraScale Device Integrated Interlaken Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
Data Channel Extension Feature Interface Attributes			
CTL_TX_CHAN_EXT[1:0]	2-bit Hex	Selects the maximum number of TX data channels. Coding as follows: 2'h0 = selects 256 channels in Control Word bits 39:32 2'h1 = selects 512 channels in Control Word bits 39:31 2'h2 = selects 1,024 channels in Control Word bits 39:30 2'h3 = selects 2,048 channels in Control Word bits 39:29	2'h0
CTL_RX_CHAN_EXT[1:0]	2-bit Hex	Selects the maximum number of RX data channels. Coding as follows: 2'h0 = selects 256 channels in Control Word bits 39:32 2'h1 = selects 512 channels in Control Word bits 39:31 2'h2 = selects 1,024 channels in Control Word bits 39:30 2'h3 = selects 2,048 channels in Control Word bits 39:29	2'h0
Retransmission Attributes			
CTL_TX_RETRANS_RAM_BANKS[1:0]	2-bit Hex	TX Retransmission Buffer Size This attribute is used to set the total number of banks of RAM to be used by the buffer. The total RAM is divided into four banks. 2'h0 = Use Bank 0 only 2'h1 = Use Banks 0-1 only 2'h2 = Use Banks 0-2 only 2'h3 = Use all four banks Default Value: 2'h3	

Table 2-3: **UltraScale+ and UltraScale Device Integrated Interlaken Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_TX_RETRANS_MULT[2:0]	3-bit Hex	TX Retransmission Sequence Multiplier. This input selects the value of the sequence multiplier. The following values are defined: 3'h0 = x1 3'h1 = x2 3'h2 = x4 3'h3 = x8 3'h4 = x16 3'h5 = x32 3'h6 = reserved 3'h7 = reserved The TX Retransmission Sequence Multiplier must be the same as the link partner RX. The retransmission sequence number and the channel extension both use the Multiple-Use bits in a Burst Control Word. In order to ensure sufficient bursts before the retransmission sequence wraps around, the following should be observed: If the multiplier is 4 or less, then CTL_TX_CHAN_EXT should be equal to 0. If the multiplier is 8, then CTL_TX_CHAN_EXT can be 0 or 1. If the multiplier is 16, then CTL_TX_CHAN_EXT can be 0, 1 or 2. If the multiplier is 32, then CTL_TX_CHAN_EXT can be 0, 1, 2 or 3.	3'h0
CTL_TX_RETRANS_DEPTH[13:0]	14-bit Hex	TX Retransmission buffer depth. This input is used to set the maximum number of entries to be stored in the retransmission buffer. An entry is defined as a single write to the buffer or eight Interlaken words. The buffer depth needs to be set to a suitable value depending upon the latency, sequence multiplier and channel extension setting.	14'h0800

Table 2-3: **UltraScale+ and UltraScale Device Integrated Interlaken Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_RX_RETRANS_MULT[2:0]	3-bit Hex	RX Retransmission Sequence Multiplier. This input selects the value of the sequence multiplier. The following values are defined: 3'h0 = x1 3'h1 = x2 3'h2 = x4 3'h3 = x8 3'h4 = x16 3'h5 = x32 3'h6 = reserved 3'h7 = reserved The RX Retransmission Sequence Multiplier must be the same as the link partner TX. The retransmission sequence number and the channel extension both use the Multiple-Use bits in a Burst Control Word. In order to ensure sufficient bursts before the retransmission sequence wraps around, the following should be observed: If the multiplier is 4 or less, then CTL_RX_CHAN_EXT should be equal to 0. If the multiplier is 8, then CTL_RX_CHAN_EXT can be 0 or 1. If the multiplier is 16, then CTL_RX_CHAN_EXT can be 0, 1 or 2. If the multiplier is 32, then CTL_RX_CHAN_EXT can be 0, 1, 2 or 3.	3'h0

Table 2-3: **UltraScale+ and UltraScale Device Integrated Interlaken Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_RX_RETRANS_TIMER1[15:0]	16-bit Hex	Interlaken RX Retransmission Short Timer Value. The input sets the maximum value timer1 counts to in terms of Interlaken words received. It begins counting as soon as a CRC24 error causes a retransmission request. This counter has two modes of operation depending upon the setting of the input CTL_RX_RETRANS_TIMER2. If CTL_RX_RETRANS_TIMER2 has a non-zero value (Mode 1), then while timer1 is counting to its limit, all subsequent errors and valid sequence discontinuities are ignored. If CTL_RX_RETRANS_TIMER2 has a zero value (Mode 2), then while timer1 is counting to its limit, all subsequent errors are ignored but valid sequence discontinuities are accepted. If the timer expires without a valid sequence discontinuity being received, retransmission is re-requested. If used in this mode, CTL_RX_RETRANS_TIMER1 should be assigned a value greater than the expected request-to-discontinuity latency. CTL_RX_RETRANS_TIMER1 must always be assigned a value greater than eight.	16'h00800
CTL_RX_RETRANS_TIMER2[15:0]	16-bit Hex	Interlaken RX Retransmission Long Timer Value. The input sets the maximum value timer2 counts to in terms of Interlaken words received. It begins counting as soon as a CRC24 error causes a retransmission request. This input must be set to a value of zero (to enable Mode 2) or a value greater than the expected request-to-discontinuity latency (to enable Mode 1). If the timer expires without a valid discontinuity being received, retransmission is re-requested.	16'h0000
CTL_RX_RETRANS_RETRY[3:0]	4-bit Hex	Interlaken RX Retransmission Retry Timer Value. This input sets the maximum value for a counter that counts the number of times re-requests for a sequence are made before asserting STAT_RX_RETRANS_RETRY_ERR. For reliable operation, this input must have a value of at least 2.	4'h2

Table 2-3: **UltraScale+ and UltraScale Device Integrated Interlaken Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_RX_RETRANS_WRAP_TIMER[7:0]	8-bit Hex	Interlaken RX Retransmission Wrap Around Timer Value. This input sets the value for a timer that counts changes in the primary sequence number while waiting for an expected sequence after a request for retransmission. This counter has no effect if this input is programmed with a value of 0. If this counter reaches its maximum value, STAT_RX_RETRANS_WRAP_ERR is asserted and remains asserted until the input CTL_RX_RETRANS_RESET is asserted and negated.	8'h00
CTL_RX_RETRANS_WDOG[11:0]	12-bit Hex	Interlaken RX Retransmission Watchdog Timer Value. This input sets the most significant bits of a maximum value for a 32-bit watchdog timer that counts while waiting for an expected sequence after a request for retransmission. This counter has no effect if all bits of this bus are a value of 0. This counter is incremented in terms of incoming Interlaken words. If this counter reaches its maximum value, STAT_RX_RETRANS_WDOG_ERR is asserted and remains asserted until the input CTL_RX_RETRANS_RESET is asserted and negated.	12'h000

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

Clocking

The Interlaken IP core has the following major clock domains:

- LBUS_CLK

  The **LBUS_CLK** drives logic for both the RX and TX LBUS interfaces and the rate adapter. The **LBUS_CLK** is also the clock for most of the control and status signals. Exceptions are noted in the port descriptions. See the section on port description for more information.

- CORE_CLK

  The **CORE_CLK** is used to clock the protocol logic portion of the design.

- RX Serial Transceiver Domain

  Each serial transceiver lane has its own recovered clock. The **RX_SERDES_CLK[11:0]** is used for all of the logic for all serial transceiver receive lanes and the receive portion of Interlaken lane logic.

  When the GT RX buffer is enabled, all the **RX_SERDES_CLK[11:0]** clocks share a single common clock.

- TX Serial Transceiver Domain

  The TX_SERDES_REFCLK is used for all of the logic for all serial transceiver transmit lanes and the transmit portion of Interlaken lane logic.

- DRP_CLK

  This clock is optional and necessary only for DRP operations. A comfortable frequency up to 250 MHz can be used.

Table 3-1 shows the typical clock frequencies for each Interlaken configuration. See the following data sheets for minimum and maximum allowable clock frequencies across speed grades.

- *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS893) [Ref 5]
- *Virtex UltraScale+ Architecture Data Sheet: DC and AC Switching Characteristics* (DS923) [Ref 17]
- *Kintex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS892) [Ref 17]
- *Kintex UltraScale+ Architecture Data Sheet: DC and AC Switching Characteristics* (DS922) [Ref 19]

Table 3-1: **Typical Clock Frequencies for Each Interlaken Configuration**

ILKN Link Width	ILKN Line Rate	rx serdes_clk frequency (MHz)	tx serdes_clk frequency (MHz)	core_clk frequency (MHz)	lbus_clk frequency (MHz)
1 to 6	25.78125G	402.8320312	402.8320312	412	300
1 to 12	12.5G	195.312	195.312	300	300
1 to 12	10.3125G	161.1328125	161.134	300	300
1 to 12	6.25G	97.656	97.656	300	300
1 to 12	5G	78.125	78.125	300	300

Resets

The integrated IP core for Interlaken has separate reset inputs for the RX and TX paths that can be asserted independently. Within the RX and TX paths, there are resets for each of the various clock domains. The reset procedure is simple and the only requirement is that a reset must be asserted when the corresponding clock is stable. The Interlaken core takes care of ensuring the different resets properly interact with each other internally and the interface operates properly (that is, there is no order required for asserting/deasserting different resets). The core must be held in reset until the corresponding clock is fully stable.

The Interlaken IP core provides **`sys_reset`** input to reset the GTs and integrated Interlaken block and **`gtwiz_reset_tx_datapath`** and **`gtwiz_reset_rx_datapath`** to reset the GT and Interlaken RX and TX datapaths individually.

Note: Some of the attributes to the Interlaken core can only be modified while the core is held in reset. If one of these attributes needs changing, the appropriate RX or TX LBUS reset input (**`RX_RESET`** or **`TX_RESET`**) must be asserted until the control input is stabilized. The core has several reset inputs. All resets must be, asynchronously asserted, and synchronously deasserted.

The reset and clocking scheme selected for the core depends on how the TX and RX logic are clocked by the GT output clocks. In synchronous clock mode, both the TX and RX helper blocks are clocked with the same GT output clock, **`txoutclk_out[0]`**. In asynchronous clock mode, TX and RX helper blocks are clocked with **`txoutclk_out[0]`** and **`rxoutclk_out[0]`** clock outputs from the GT respectively.

The selection between these modes is made in the Vivado® Design Suite. See Chapter 4, Design Flow Steps for more information. Figure 3-1 and Figure 3-2 show the clocking reset diagrams for the asynchronous and synchronous clock modes of the Interlaken core.

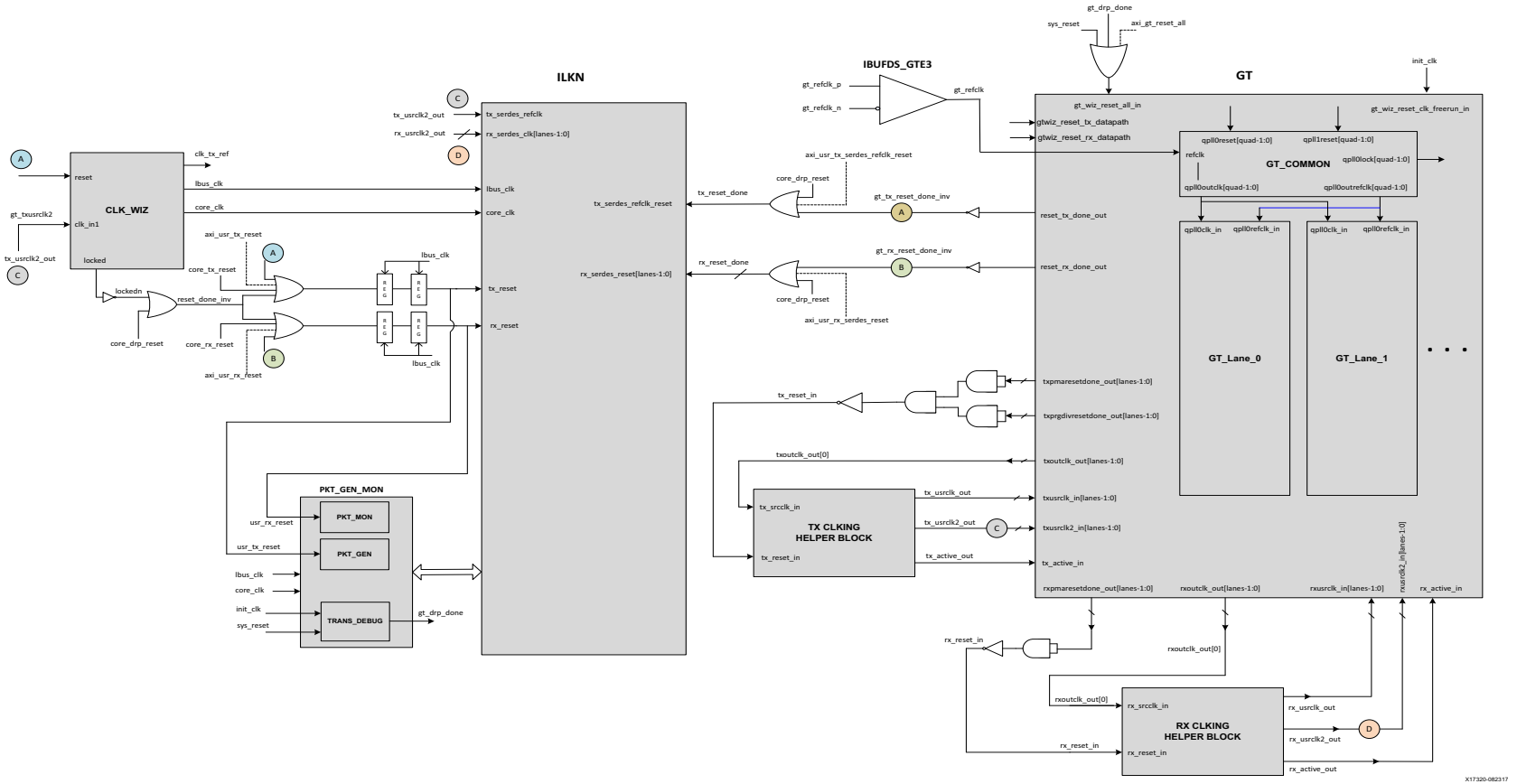


Figure 3-1: **Interlaken Core Clocking Reset Interface - Asynchronous Clock Mode**

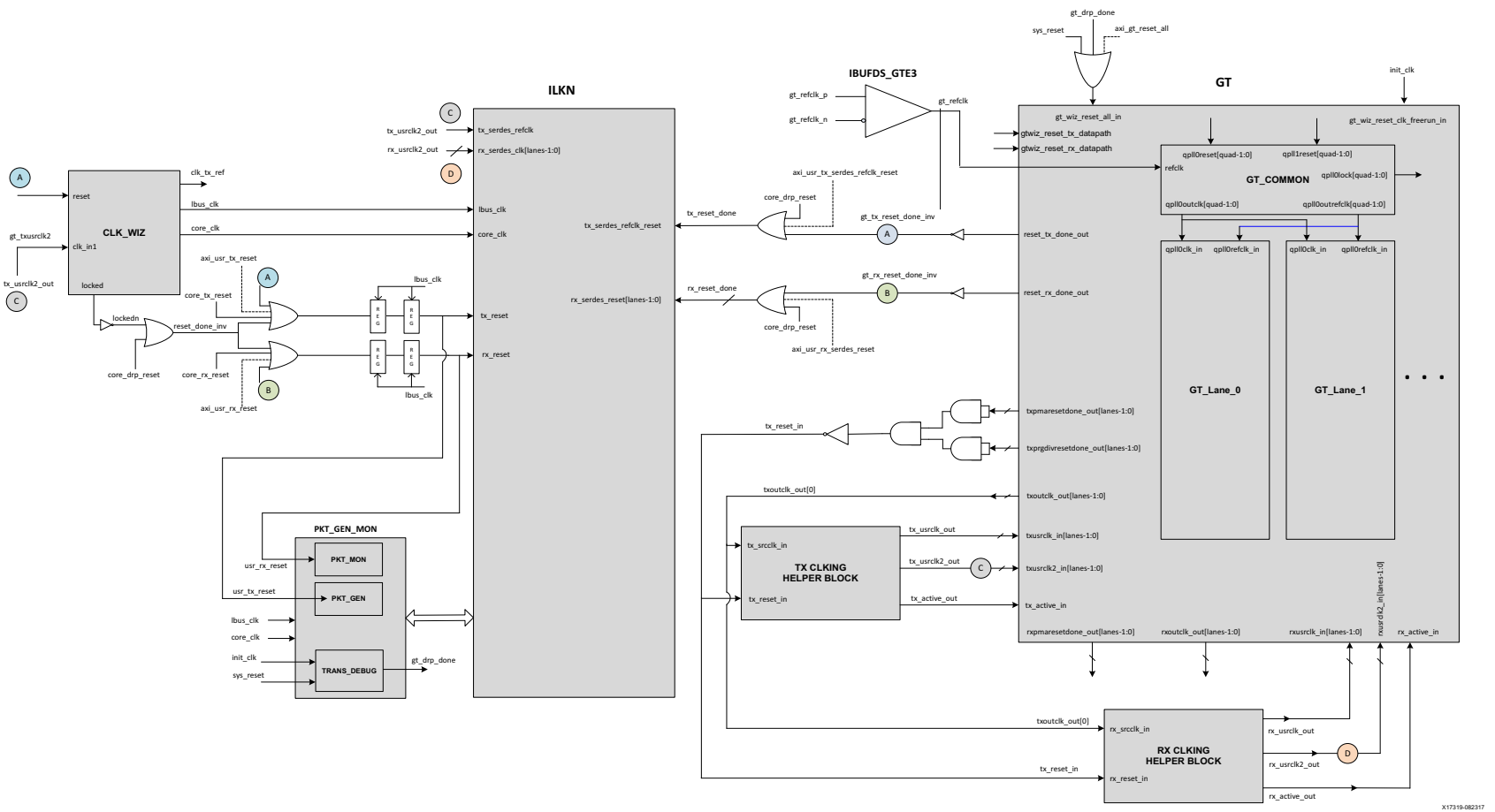


Figure 3-2: **Interlaken Core Clocking Reset Interface - Synchronous Clock Mode**

User Interface

The user interface is a simple packet interface designed to allow easy integration of the Interlaken IP core into a system.

The LBUS consists of the following separate interfaces:

- Transmitter (TX) interface

  The transmitter accepts packet-oriented data, packages the data in accordance with the Interlaken specification, and sends that packaged data to the serial transceiver macros. The transmitter has control/configuration inputs to shape the data packaging to meet specific user requirements.

- Receiver (RX) interface

  The receiver accepts Interlaken bitstreams from the serial transceiver, removes the Interlaken packaging, and provides packet oriented data.

- Status/Control interface

  The status/control interface sets the characteristics of the interface and monitors its operation.

This section describes the various LBUS interfaces and provides a detailed description of each individual port. The UltraScale™ device integrated core for Interlaken implements a 512-bit segmented LBUS.

Note: In this section, asserting means "assigning a value of 1", and negating means "assigning a value of 0".

Segmented LBUS Protocol

Overview

This section describes the segmented LBUS protocol for the Interlaken system side interface. The segmented LBUS consists of four segments, each one 128 bits wide, for a total of 512 bits.

Summary

The disadvantage of a wide non-segmented LBUS is the loss of potential bandwidth that occurs at the end of a packet when the size of the packet is not a multiple of the LBUS width. Therefore, the Interlaken hard block employs the segmented LBUS.

Conceptually, the segmented LBUS is a collection of narrower LBUSes, each 128 bits wide, with multiple transfers presented in parallel during the same clock cycle. Each segment has all the control signals associated with a complete 128-bit LBUS. The 512-bit segmented LBUS has four 128-bit segments with the signals for each segment listed in Table 3-2:

Table 3-2: **512-bit Segmented LBUS Signals**

Segment Number	TX Signals	RX Signals
0	tx_datain0[127:0] tx_chanin0[10:0] tx_enain0 tx_sopin0 tx_eopin0 tx_errin0 tx_mtyin0[3:0] tx_bctlin0	rx_dataout0[127:0] rx_chanout0[10:0] rx_enaout0 rx_sopout0 rx_eopout0 rx_errout0 rx_mtyout0[3:0]
1	tx_datain1[127:0] tx_chanin1[10:0] tx_enain1 tx_sopin1 tx_eopin1 tx_errin1 tx_mtyin1[3:0] tx_bctlin1	rx_dataout1[127:0] rx_chanout1[10:0] rx_enaout1 rx_sopout1 rx_eopout1 rx_errout1 rx_mtyout1[3:0]
2	tx_datain2[127:0] tx_chanin2[10:0] tx_enain2 tx_sopin2 tx_eopin2 tx_errin2 tx_mtyin2[3:0] tx_bctlin2	rx_dataout2[127:0] rx_chanout2[10:0] rx_enaout2 rx_sopout2 rx_eopout2 rx_errout2 rx_mtyout2[3:0]
3	tx_datain3[127:0] tx_chanin3[10:0] tx_enain3 tx_sopin3 tx_eopin3 tx_errin3 tx_mtyin3[3:0] tx_bctlin3	rx_dataout3[127:0] rx_chanout3[10:0] rx_enaout3 rx_sopout3 rx_eopout3 rx_errout3 rx_mtyout3[3:0]

Following is a detailed description of the signals associated with segment 0 of the TX and RX LBUS interfaces. Signals associated with other segments are defined similarly.

tx_datain0[127:0]

Transmit LBUS Data. This bus receives input data from the user logic. The value of the bus is captured in every cycle for which **tx_enain0** is sampled as 1.

tx_chanin0[10:0]

Transmit LBUS channel number. This bus receives the channel number for the packet being written. The value of the bus is captured in every cycle for which **tx_enain0** is sampled as 1.

In packet mode, the channel number remains the same for the duration of the packet transfer from SOP to EOP. In burst-interleaved mode, the channel number can change for each burst.

tx_enain0

Transmit LBUS enable. This signal is used to enable the TX LBUS Interface. All signals on the LBUS interface are sampled only in cycles during which **tx_enain0** is sampled as 1.

tx_sopin0

Transmit LBUS Start Of Packet. This signal is used to indicate the SOP when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles during which **tx_enain0** is sampled as 1.

tx_eopin0

Transmit LBUS EOP. This signal is used to indicate the EOP when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles during which **tx_enain0** is sampled as 1.

tx_errin0

Transmit LBUS Error. This signal is used to indicate a packet contains an error when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles during which **tx_enain0** and **tx_eopin0** are sampled as 1.

tx_mtyin0[3:0]

Transmit LBUS Empty. This bus is used to indicate how many bytes of the tx_datain0 bus are empty or invalid for the last transfer of the current packet. This bus is sampled only in cycles that **tx_enain0** and **tx_eopin0** are sampled as 1.

When **tx_eopin0** and **tx_err0 in** are sampled as 1, the value of **tx_mtyin0[2:0]** is ignored and treated as if it was 000. **tx_mtyin0[3]** is used as usual.

tx_bctlin0

Transmit force insertion of Burst Control word. This input is used to force the insertion of a Burst Control Word. When **tx_bctlin0** and **tx_enain0**, are sampled as 1, a Burst Control word is inserted before the data on the **tx_datain0** bus is transmitted even if one is not required to observe the BurstMax parameter.

This input is used by the enhanced scheduling algorithm, external to the Interlaken IP Core.

IMPORTANT: *Enhanced scheduling is required for the segmented LBUS.*

rx_dataout0[127:0]

Receive LBUS Data. The value of the bus is only valid in cycles during which **rx_enaout0** is sampled as 1.

rx_chanout0[10:0]

Receive channel number. The bus indicates the channel number of the in-flight packet and is only valid in cycles during which **rx_enaout0** is sampled as 1.

rx_enaout0

Receive LBUS enable. This signal qualifies the other signal of the RX LBUS Interface. The signals of the RX LBUS Interface are only valid in cycles during which **rx_enaout0** is sampled as 1.

rx_sopout0

Receive LBUS Start of Packet. This signal indicates the SOP when it is sampled as 1 and is only valid in cycles during which **rx_enaout0** is sampled as a 1.

rx_eopout0

Receive LBUS EOP. This signal indicates the EOP when it is sampled as 1 and is only valid in cycles during which **rx_enaout0** is sampled as a 1.

rx_errout0

Receive LBUS Error. This signal indicates that the current packet being received has an error when it is sampled as 1. This signal is only valid in cycles when both **rx_enaout0** and **rx_eopout0** are sampled as a 1. When this signal is a value of 0, it indicates that there is no error in the packet being received.

rx_mtyout0[3:0]

Receive LBUS Empty. This bus indicates how many bytes of the rx_dataout bus are empty or invalid for the last transfer of the current packet. This bus is only valid in cycles when both **rx_enaout0** and **rx_eopout0** are sampled as 1.

When **rx_errout0** and **rx_enaout0** are sampled as 1, the value of **rx_mtyout0[2:0]** is always 000. **rx_mtyout0[3]** is used as usual.

TX LBUS Interface

The synchronous TX Local bus interface accepts packet-oriented data of arbitrary length. All signals are synchronous relative to the rising-edge of the lbus_**clk** port. Figure 3-3 shows a sample waveform for data transactions for two consecutive 65-byte packets using a 512-bit segmented bus. Each of the 4 segments is 128 bits wide.

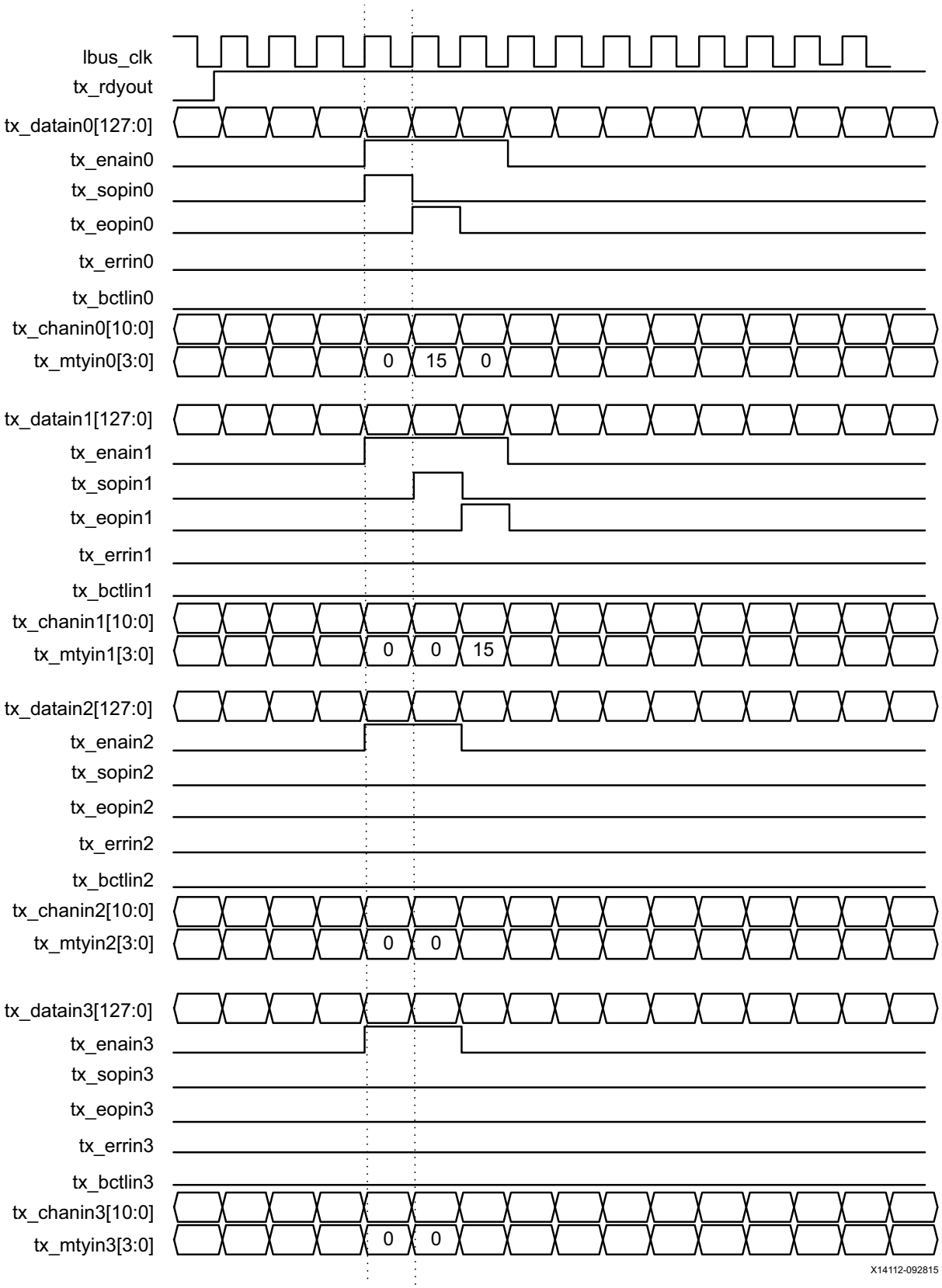


Figure 3-3: **Sample Waveform for TX LBUS Interface**

TX Transactions

Data is transferred on a given **tx_datain<N>** segment when the corresponding **tx_enain<N>** is asserted. The t**x_enain<N>** signal qualifies other inputs of segment <N> and must be valid every LBUS clock cycle. When **tx_enain<N>** is deasserted, other signals of segment <N> are ignored.

The start of a packet is identified by the assertion of **tx_sopin<N>** with the corresponding **tx_enain<N>**. Similarly, the end of a packet is identified by the assertion of **tx_eopin<N>** with the corresponding **tx_enain<N>**. Both **tx_sopin<N>** and **tx_eopin<N>** can be asserted on a given cycle. This occurs for packets that are less than or equal to the LBUS width. Furthermore, both **tx_sopin<N>** and **tx_eopin<N>** can be asserted for a given segment on a given cycle. This occurs for packets that are less than or equal to 16 bytes (the segment size).

The channel number for a packet is presented on the **tx_chanin<N>** input of the corresponding segment and must be valid for every segment where **tx_enain<N>** is asserted. After SOP has been asserted for a certain channel number, it cannot be asserted again with that channel number until EOP has been asserted for the same channel number.

The first 16 bytes of a packet must be presented on a given **tx_datain<N>** segment during the cycle that the corresponding **tx_sopin<N>** and **tx_enain<N>** are asserted. In other words, the SOP is segment aligned. Subsequent 16-byte chunks of data are transferred during segments that follow. For each of those segments, the corresponding **tx_sopin<N>** must be negated. The first byte of the packet is written on bits [127:120] of the segment, the second byte on bits [119:112], and so forth.

The last bytes of the packet are transferred on the **tx_datain<N>** segment whose corresponding **tx_eopin<N>** is asserted. Unless **tx_eopin<N>** is asserted, all 16 bytes of **tx_datain<N>** must contain valid data whenever **tx_enain<N>** is asserted. Note that if burst-interleaved mode is employed, then segments containing data from other packets can be interleaved with segments containing data for a given packet. The **tx_chanin<N>** input identifies the packets from different channels.

During the segment containing the last bytes of a packet, the **tx_mtyin<N>** port reflects how many bytes of the corresponding **tx_datain<N>** are invalid (or empty). A given **tx_mtyin<N>** port only has meaning during cycles when both the corresponding **tx_enain<N>** and **tx_eopin<N>** are asserted. If **tx_mtyin<N>** has a value of 0x0, there are no empty byte lanes (that is, all bits of the segment are valid). If **tx_mtyin<N>** has a value of 0x1, then one byte lane is empty, specifically **tx_datain<N>[7:0]** does not contain valid data. If **tx_mtyin<N>** has a value of 0x2, then two byte lanes are empty — specifically **tx_datain<N>[15:0]** does not contain valid data. If **tx_mtyin<N>** has a value of 0x3, then three byte lanes are empty — specifically **tx_datain<N>[23:0]** does not contain valid data. And so forth for other possible values of **tx_mtyin<N>**.

During the segment containing the last bytes of a packet, when **tx_eopin<N>** is asserted with **tx_enain<N>**, the corresponding **tx_errin<N>** can also be asserted. This marks the packet as being in error and this information is included in the final Interlaken Control Word associated with this packet. When **tx_eopin<N>** and **tx_errin<N>** are sampled as 1, the value of **tx_mtyin<N>[2:0]** is ignored and treated as equal to 000, while **tx_mtyin<N>[3]** is used as usual.

tx_rdyout

Data can be safely written, that is, **tx_enain0** asserted, whenever **tx_rdyout** is asserted. After **tx_rdyout** is negated, additional writes, using **tx_enain0**, can be safely performed provided **tx_ovfout** is never asserted. When **tx_rdyout** is asserted again, additional data can be written. If, at any time, the back-pressure mechanism is violated, the **tx_ovfout** is asserted to indicate the violation. Up to 8 write cycles can be safely performed after **tx_rdyout** is negated, but no more until **tx_rdyout** is asserted again.

IMPORTANT: *To maximize bandwidth on the TX Interlaken interface, data bursts should be packed on the TX LBUS interface. Under-utilizing the TX LBUS interface (that is, having idle segments on the TX LBUS) results in lost bandwidth on the TX Interlaken interface due to insertion of Idle Control Words.*

Data Formatting

Interlaken breaks packets into bursts as described in the Interlaken Revision 1.2 specification document. A burst is a sequence Data Word between two Control Words. The size of the bursts generated by the Interlaken IP core is controlled by these factors:

- Inputs **CTL_TX_BURSTMAX** and **CTL_TX_BURSTSHORT**
- How packets are written to the TX

The Interlaken IP core operates in one of two modes, depending on how the data is written to the TX:

- Packet Mode
- Burst Interleaved Mode

Packet Mode

Packet mode is when a packet with a certain channel number is written in its entirety without interruption by a packet for a different channel.

Unless **tx_bctlin<N>** is asserted (see Use of TX_BCTLIN) the size of the bursts (that is, the number of Data Words between Control Words) is **CTL_TX_BURSTMAX**. The EOP and EOP-1 bursts are to be determined by the value of BurstMin as calculated by the enhanced scheduling algorithm.

Burst Interleaved Mode

Interleaved mode is when packets with different channel addresses/identifiers are burst interleaved. Ensure that the following are strictly observed:

- When **tx_sopin<N>** has been asserted for a channel, it cannot be asserted again for that channel until a corresponding **tx_eopin<N>** for that channel has been written.
- Unless **tx_eopin<N>** is asserted, the full width of the segment, **tx_datain<N>** must contain valid data as discussed in TX LBUS Interface.

The size of the bursts generated in interleaved mode is governed by the **tx_bctlin<N>** inputs (see Use of TX_BCTLIN), **CTL_TX_BURSTSHORT**, **CTL_TX_BURSTMAX**, and the changing of the channel ID of packets.

IMPORTANT: *Xilinx requires implementing the Optional Scheduling Enhancement as described in section 5.3.2.1.1 of the Interlaken Revision 1.2 specification document.*

Use of TX_BCTLIN

The **tx_bctlin<N>** input operates in a similar manner to **tx_sopin<N>** or a change in **tx_chanin<N>**; they cause a Burst Control Word to be injected into the data stream before the data on segment <N>.

The purpose of the **tx_bctlin<N>** input is to permit the forcing of Burst Control Words that otherwise would not be transmitted. This is a necessary function for the creation of an external scheduler that implements the Optional Scheduling Enhancement described in section 5.3.2.1.1 of the Interlaken Revision 1.2 specification document.

The Interlaken IP core strictly observes the programmed values for **CTL_TX_BURSTMAX** and **CTL_TX_BURSTSHORT** and injects Burst and Idle Control Words where required. Consequently, the Interlaken IP core can inject Idle Control Words that otherwise would not be required, which results in reducing the effective bandwidth.

For example, assume the **CTL_TX_BURSTMAX** is set to 256 bytes and **CTL_TX_BURSTSHORT** is set to 64 bytes. A packet of 264 bytes written into the Interlaken IP core, without the use of **tx_bctlin<N>**, is followed by three undesirable Idle Control Words that are required to meet the **CTL_TX_BURSTSHORT** parameter. Specifically:

1. One Burst Control Word (with Start of Packet) is sent.
2. 32 Data Words are sent.
3. One Burst Control Word (without Start of Packet) is sent.
4. One Data Word is sent.
5. Seven Idle Control Words are sent (to satisfy BurstShort) for a total of 42 Words.

If the corresponding **tx_bctlin<N>** signal is asserted after 128 bytes are sent, the following occurs:

1. One Burst Control Word (with Start of Packet) is sent.
2. 16 Data Words are sent.
3. One Burst Control Word (without Start of Packet) is sent.
4. 17 Data Words are sent.
5. 0 Idle Control Words is sent for a total of 35 Words.

Ensure that all rules that govern Interlaken bursts, as defined in the Interlaken Specification document Revision 1.2, are followed when using **tx_bctlin<N>** signals. In particular, you must ensure that each burst on each channel which is not EOP or EOP-1 is equal to BurstMax.

RX LBUS Interface

The synchronous RX Local bus interface provides packet-oriented data much like the TX Local bus interface accepts. All signals are synchronous with the rising-edge of the Local bus clock. Figure 3-4 shows a sample waveform for two data transactions for 65-byte packets using a 512-bit segmented LBUS.

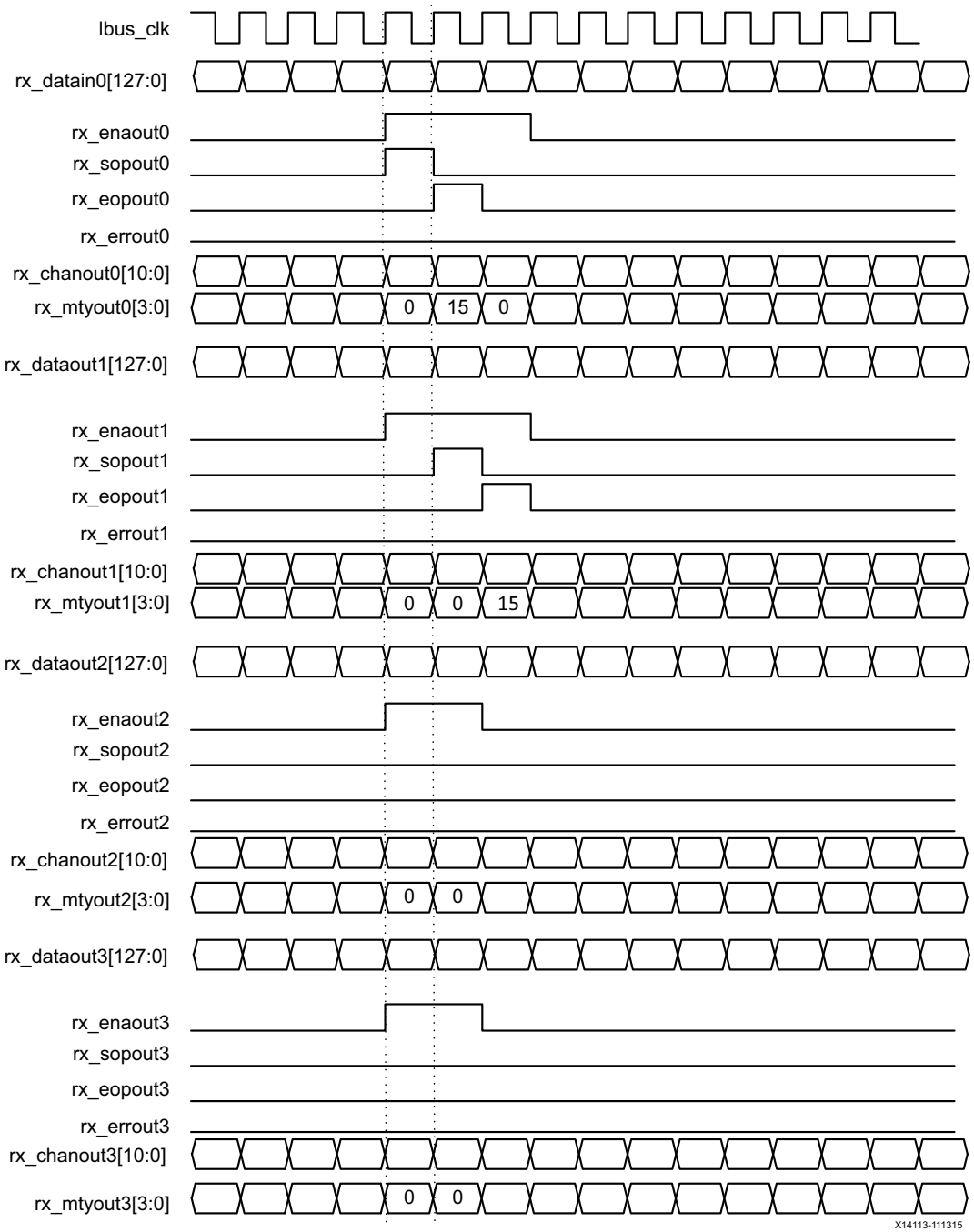


Figure 3-4: **Sample Waveform for Two Data Transactions**

Data is supplied on segment <N> by the Interlaken core on every LBUS clock cycle when **rx_enaout<N>** is asserted. This signal qualifies the other outputs of segment <N> of the RX Local bus interface.

The RX is similar to the TX, in that **rx_sopout<N>** identifies the start of a packet on segment <N> and **rx_eopout<N>** identifies the end of a packet on segment <N>. Both **rx_sopout<N>** and **rx_eopout<N>** are asserted during the same cycle for packets that are less than or equal to the segment width.

As in the TX, The first byte of a packet that starts on segment <N> is supplied on bits [127:120] of **rx_dataout<N>**, the second byte on bits [119:112], and so forth.

Portions of packets are written on the bus segments in the full width of the segment (16 bytes) unless for the last segment of the packet. When **rx_eopout<N>** is asserted on segment <N>, the **rx_mtyout<N>** bus indicates how many byte lanes in the segment are invalid. The encoding is the same as for **tx_mtyin<N>**.

During the last cycle of a packet, when **rx_eopout<N>** is asserted with **rx_enaout<N>**, **rx_errout<N>** can also be asserted to indicate an error in the packet ended on segment <N>.

There is no mechanism to back pressure the RX Local bus interface. The user logic must be capable of receiving data when **rx_enaout0** is asserted. The Interlaken flow control mechanism can be used to stop the flow of data, either using the inband or out of band protocol, or both.

The data provided by the RX Local bus interface is in the same sequence as it is received from the Interlaken bus. Packets can be interleaved and are distinguished using the channel number presented on **rx_chanout<N>**.

The RX segmented LBUS can contain more than one SOP and more than one EOP, in contrast to the way the TX must function. This is because there is no requirement to accommodate Burst Control Words on the receive side LBUS (they have already been processed) and therefore packets can be packed as efficiently as possible when they arrive.

Bus Rules

Several rules govern the successful use of the segmented LBUS protocol.

Segment Ordering

The 128-bit segments are ordered 0 to 3. The first of the 128-bit transfers occurs on segment 0, the second on segment 1, and so forth. During each local bus clock cycle that data is transferred on the segmented LBUS, segment 0 must be active. The segmented bus is aligned such that the first bit of the incoming data is placed at the MSB of segment 0.

Active Segments

Data is transferred in a segment on the TX interface when the corresponding **tx_enain<N>** is a value of 1. The TX interface buffers data and does not forward until it has a sufficient quantity. Therefore, it is acceptable to have clock cycles in which none of the **tx_enain<N>** signals are active. However, during a clock cycle with **tx_enain0** active, segments must be filled in sequence with no gaps between active segments. The following are some of the illegal combinations of **tx_enain<N>**:

tx_enain0=0, tx_enain1=1, tx_enain2=1, tx_enain3=1

tx_enain0=1, tx_enain1=0, tx_enain2=1, tx_enain3=1

tx_enain0=1, tx_enain1=1, tx_enain2=0, tx_enain3=1

Data is transferred in a segment on the RX interface when the corresponding **rx_enain<N>** is a value of 1. Similarly, the RX interface buffers data and does not forward until it has a sufficient quantity. Therefore, there will be clock cycles in which none of the **rx_enain<N>** signals are active.

TX Back-Pressure

The optimal use of bandwidth requires that TX Local bus data be able to be written at a rate faster than can be delivered on the serial interface. This means that there must be back pressure, or flow-control, on the TX segmented LBUS. The signals used to implement back-pressure are **tx_rdyout** and **tx_ovfout**. These signals are common for all segments and operate in the same manner as with the regular LBUS. When responding to back-pressure during a clock cycle, none of the **tx_enain<N>** can be active.

Gaps

The purpose of the segmented LBUS is to provide a means to optimally use the data bus. Therefore, as discussed in the section Active Segments, segments must be filled in sequence with no gaps between used segments. However, if a segment has an EOP, the following segments might be inactive. For example, the following are permitted during a single clock cycle:

tx_enain0=1 tx_eopin0=0 tx_enain1=1 tx_eopin1=0

tx_enain2=1 tx_eopin2=1 tx_enain3=0 tx_eopin3=0

or

tx_enain0=1 tx_eopin0=0 tx_enain1=1 tx_eopin1=1

tx_enain2=0 tx_eopin2=0 tx_enain3=0 tx_eopin3=0

or

tx_enain0=1 tx_eopin0=1 tx_enain1=0 tx_eopin1=0

tx_enain2=0 tx_eopin2=0 tx_enain3=0 tx_eopin3=0

Examples

The following examples illustrate segmented LBUS cycles covering various combinations of SoP, Dat (data in the middle of a packet), EOP, and idle (no data on the bus). Valid and invalid cycles are shown.

The segmented LBUS is assumed to be 512 bits wide and each segment is 128 bits wide (16 bytes). The TX direction is illustrated. The RX direction has analogous behavior but there will be no invalid cycles on the receive segmented LBUS. Unlike the TX, the RX is able to transfer two packets in one cycle, for example, with two SoP and two EOP, due to the removal of overhead.

It is assumed that Packet Mode is being used.

Valid Cycles

Table 3-3 and Table 3-4 show many possible valid TX segmented LBUS cycles. In these examples, BurstMax has been set to 256 and BurstShort to 64. The different shadings represent different bursts from different packets.

Table 3-3: **Segmented LBUS Valid Cycles (BurstMax = 256 and BurstMax = 64)**

Clock Cycle	1	2	3	4	5	6	7	8	9	10
seg0	SoP	idle	SoP	SoP	Dat	Dat	idle	Dat	SoP	idle
seg1	Dat	idle	Dat	Dat	EoP	Dat	idle	Dat	Dat	idle
seg2	Dat	idle	Dat	Dat	SoP	Dat	idle	Dat	Dat	idle
seg3	EoP	idle	EoP	Dat	Dat	Dat	idle	EoP	Dat	idle
tx_rdyout	1	1	1	1	1	1	0	1	0	0
tx_ovfout	0	0	0	0	0	0	0	0	0	0

Table 3-4: **Segmented LBUS Valid Cycles (BurstMax = 256 and BurstMax = 64)**

Clock Cycle	1	2	3	4	5	6	7	8	9	10
seg0	Dat	Dat	Dat	Dat	Dat	Dat	Dat	Dat	SoP	idle
seg1	Dat	Dat	Dat	Dat	Dat	Dat	idle	Dat	Dat	idle
seg2	Dat	Dat	Dat	Dat	Dat	Dat	idle	Dat	Dat	idle
seg3	Dat	Dat	Dat	Dat	Dat	Dat	idle	EoP	Dat	idle
tx_rdyout	1	1	1	1	1	1	0	1	0	0
tx_ovfout	0	0	0	0	0	0	0	0	0	0

Invalid Cycles

Table 3-5 shows several invalid TX segmented LBUS cycles as indicated by the asterisks. In these examples, BurstMax has been set to 256 and BurstShort to 64.

Table 3-5: **Invalid Segmented LBUS Cycles**

Clock Cycle	1	2	3*	4	5*	6*	7*	8	9*	10		18*	19*
seg0	SoP	idle	SoP	Dat	Dat	SoP	Dat	Dat	SoP	Dat		SoP	Dat
seg1	Dat	idle	Dat	Dat	Dat	Dat	idle	Dat	Dat	Dat		Dat	Dat
seg2	Dat	idle	Dat	Dat	Dat	Dat	idle	Dat	idle	Dat	--	Dat	Dat
seg3	EoP	idle	SoP	Dat	Dat	Dat	idle	EoP	Dat	EoP		Dat	Dat
tx_rdyout	1	1	1	1	1	1	1	1	1	0	0	0	0
tx_ovfout	0	0	0	0	0	0	0	0	0	0	0	0	1

- Cycle 3 is not valid because it contains two SoPs.
- Cycle 5 does not contain an EoP even though there is an SoP in the next cycle.

  Note: In Burst Interleaved mode this would be permitted because the current packet will be ended in a later cycle.
- Cycle 6 has an SoP even though the preceding packet was not closed with an EoP. This sequence is not permitted by the LBUS rules and results in undefined behavior.
- Cycle 7 contains idles even though there is no EoP or BurstMax.
- Cycle 9 contains an idle segment during a packet transfer which is not permitted by the segmented LBUS rules.
- Cycle 18 is not permitted because a data transfer is being performed even though **`tx_rdyout`** has been deasserted for eight consecutive cycle.
- Cycle 19 must never be performed because **`tx_ovfout`** has been asserted. In the event of **`tx_ovfout`** being asserted, the TX should be reset.

Burst Rules

The segmented LBUS requires that certain rules be followed to obtain the correct Interlaken burst behavior. These are described in the following subsections.

Burst Length

In Interlaken, a burst is defined as the number of 64-bit data words between two control words. Data for different channels can be interleaved between control words. The Segmented LBUS requires that bursts, not ending with an EOP, be multiples of the full width of the segmented LBUS. Consequently, for a Segmented LBUS with four segments, bursts, not ending with an EOP, must be 64-bytes, 128-bytes, 192-bytes, 256-bytes, and so forth.

The Interlaken specification describes an enhanced scheduling algorithm. The previous example is the same as a scheduler with the enhanced algorithm that has a BurstMin of 64-bytes.

Burst Control Words

Burst Control Words are either forced through a **tx_sopin<N>** and **tx_bctlin<N>** input, forced through a change of channel on **tx_chanin<N>**, or implied by the value of BurstMax. The Segmented LBUS requires that there be only one Burst Control Word per clock cycle. Consequently, two bursts, implied or forced, cannot begin in the same clock cycle. The signal **stat_tx_burst_err** is asserted if two Burst Control Words occur in the same cycle.

BurstShort

BurstShort must be at least equal to the total LBUS width AND a multiple of 32 bytes. For a 512-bit segmented LBUS, BurstShort can be 64 bytes, 96 bytes, 128 bytes, and so on up to 256 bytes. BurstShort must always be less than or equal to BurstMax.

The Segmented LBUS in the TX core never violates the BurstShort value set by the **tx_ctl_burstshort** input bus. However, more than the absolute minimum required number of idle control words can be injected depending on the burst size and which segment the burst ends. Xilinx requires designing the transmitting scheduler so that burst sizes are always equal to BurstMax except for the last two transfers of a packet (EOP and EOP-1).

The required value of BurstShort must be matched by the link partner.

BurstMax Requirements

BurstMax must be greater than or equal to BurstShort. BurstMax must be a multiple of 64 bytes. The required value of BurstMax must be matched by the link partner.

Enhanced Scheduling

The segmented LBUS must be used in conjunction with the enhanced scheduling algorithm described in the Interlaken Protocol Definition. Among the requirements of this algorithm are:

- All bursts except the last two must be equal to BurstMax.
- Bursts must be written to the LBUS in their entirety before changing channels or writing the next burst.
- The last two bursts of a packet are delineated using bctlin and by knowing the value of BurstMin.

BurstMin Requirements

BurstMin must be less than or equal to half BurstMax and a multiple of the LBUS width.

Channel Changes

Channel changes are only permitted after a burst has been fully written to the LBUS.

Status/Control Interface

The Status/Control interface allows you to set up the Interlaken IP core configuration and monitor the Interlaken IP core status. The following sections describe the various Status and Control signals.

Note: Most of the following status signal descriptions assume a good understanding of the Interlaken Protocol. See the Interlaken Protocol Definition Revision 1.2 document for more details.

RX Meta Frame Status

The Interlaken protocol requires that each lane align or synchronize to incoming words using the procedure described in the Interlaken specification. The Interlaken IP core provides status bits to indicate the state of word boundary synchronization and lane alignment. All signals are synchronous with the rising-edge of **`LBUS_CLK`** and a detailed description of each signal is included in this section.

STAT_RX_SYNCED[11:0]

When a bit of this bus is 0, it indicates that word boundary synchronization of the corresponding lane is not complete or that an error has occurred as identified by another status bit.

When a bit of this bus is 1, it indicates that the corresponding lane is word boundary synchronized and is receiving Meta Frame Synchronization Words and Scrambler State Control Words as expected.

STAT_RX_SYNCED_ERR[11:0]

When a bit of this bus is 1, it indicates one of several possible failures on the corresponding lane:

- Word boundary synchronization in the lane was not possible using Framing bits [65:64].
- After word boundary synchronization in the lane was achieved, errors were detected on Framing bits [65:64].
- After word boundary synchronization in the lane was achieved, a valid Meta Frame Synchronization Word was never received.

The bits of the bus remain asserted until word boundary synchronization occurs or until some other error or failure is signaled for the corresponding lane.

STAT_RX_MF_LEN_ERR[11:0]

When a bit of this bus is 1, it indicates that Meta Frame Synchronization Words are being received but not at the expected rate in the corresponding lane. The transmitter and receiver must be re-configured with the same Meta Frame length.

The bits of the bus remain asserted until word boundary synchronization occurs or until some other error or failure is signaled for the corresponding lane.

STAT_RX_MF_REPEAT_ERR[11:0]

After word boundary synchronization is achieved in a lane, if a bit of this bus is a 1, it indicates one of the following:

- Four consecutive invalid Meta Frame Synchronization Words were detected in the corresponding lane.
- Three consecutive invalid Scrambler State Control Words were detected in the corresponding lane.

The bits of the bus remain asserted until word boundary synchronization occurs or until some other error or failure is signaled for the corresponding lane.

STAT_RX_DESCRAM_ERR[11:0]

When a bit of this bus is 1, it indicates that a Scrambler State Control Word with an unexpected value was received on the corresponding lane. This bit is only asserted after word boundary synchronization is achieved. This output is asserted for one clock period each time a descrambler error is detected.

STAT_RX_MF_ERR[11:0]

When a bit of this bus is 1, it indicates that an invalid Meta Frame Synchronization Word was received on the corresponding lane. This bit is only asserted after word boundary synchronization is achieved. This output is asserted for one clock period each time an invalid Meta Frame Synchronization Word is detected.

STAT_RX_ALIGNED

When **STAT_RX_ALIGNED** is a value of 1, all of the lanes are aligned or de-skewed as explained in the Interlaken specification and the receiver is ready to receive packet data.

STAT_RX_ALIGNED_ERR

When **STAT_RX_ALIGNED_ERR** is a value of 1, one of the following occurs:

- Lane alignment fails after several attempts
- Lane alignment is lost (**STAT_RX_ALIGNED** is asserted and then it is negated)

STAT_RX_FRAMING_ERR[11:0]

When a bit of this bus is 1, an illegal framing pattern is detected on the corresponding lane after word boundary synchronization. If this error is detected after lane alignment, the error is treated like a CRC24 error.

This output is asserted for one clock period each time an illegal framing pattern is detected.

RX Error Status

The Interlaken IP core provides status signals to identify Interlaken data transmission protocol violations in sequences of Control and Data words. These are errors independent of the status of the Meta Frame. Generally, these signals do not indicate a failure on the part of the sending transmitter but some type of corruption during the transmission.

All signals are synchronous with the rising-edge of **LBUS_CLK** and a detailed description of each signal follows.

STAT_RX_CRC24_ERR

When this signal is a value of 1, it indicates that the error detection logic has identified a mismatch between the expected and received value of CRC24 in a Control Word.

Every time a CRC24 error is detected, all open packets are marked as containing errors as specified by the Interlaken Protocol specification. By definition, there is no mechanism provided by Interlaken to associate a CRC24 error with individual packets.

This signal is asserted for one clock period each time a CRC24 error is detected.

STAT_RX_MSOP_ERR

Packets received with a particular channel address must begin with a valid SOP. If data is detected for a particular channel without a valid SOP, this signal is asserted for a single Local bus clock cycle. Additionally, the required SOP is inserted before the data and an error is signaled in the EOP cycle by the **RX_ERROUT** signal.

This signal is available as a status signal to indicate that a missing SOP error condition occurred. No indication is provided on the Local bus as to which packet had the missing SOP. The packet is marked as containing an error. This is because a missing SOP is almost always associated with other errors that cannot be associated with a particular packet.

The purpose of SOP insertion is to ensure that packets for a particular channel are always delivered on the RX Local bus beginning with an SOP and ending with an EOP to remove the need for user logic to perform bus protocol checking. The **STAT_RX_MSOP_ERR** status signal indicates that this function is being performed and for most applications can be ignored.

STAT_RX_MEOP_ERR

Packets received with a particular channel address must begin with a valid SOP and end with a valid EOP. If an SOP is detected without receiving an EOP for the previous packet, this signal is asserted for a single Local bus clock cycle. Additionally, the extra SOP is deleted, the packets are merged together, and an error is signaled with the EOP by the **RX_ERROUT** signal.

This signal is available as a status signal to indicate a missing EOP error condition occurred and that SOP deletion occurred. No indication is provided on the Local bus which packet is actually a merged packet. The packet is marked as containing an error. This is because a missing EOP is almost always associated with other errors that cannot be associated with a particular packet.

The purpose of SOP deletion is to ensure that packets for a particular channel are always delivered on the RX Local bus beginning with an SOP and ending with an EOP to remove the need for user logic to perform bus protocol checking. The **STAT_RX_MEOP_ERR** status signal indicates that this function is being performed and for most applications can be ignored.

STAT_RX_BURST_ERR

This signal is asserted if:

- BurstShort violation is detected
- Burst length violation is detected

When this signal has a value of 1, it indicates one of the preceding burst errors has been detected. These errors are treated as CRC24 errors and all open packets are treated as being in error.

This signal is asserted for one clock period each time an error is detected.

A BurstShort error occurs when the spacing between Burst Control Words is less than the minimum RX BurstShort parameter which depends on the variant configuration. A burst length violation occurs when the length of a received burst, other than that ending with an EOP, is not a multiple of the LBUS width.

TX Rate Limiting

The Interlaken IP core rate limiter can be used to reduce the overall Data Word transmission rate. This is achieved by transmitting Idle Control Words in between packet bursts to limit the effective data transfer rate. The purpose of transmitter rate limiting is to reduce buffering requirements by the receiving device and reduce the amount of flow-control stalling that can otherwise be required.

Rate limiting is not a substitute for flow control but something that should be used in conjunction with flow control when a receiver cannot continuously accept Data Words at the full rate.

The rate limiter uses a token bucket scheme. A token represents a single byte. When the token bucket contains at least BurstMax number of tokens, up to BurstMax bytes are sent. When that has completed, the transmitter waits until there are at least BurstMax number of tokens in the bucket again before sending more data. The token count goes negative if it is necessary to send a burst of data that cannot be interrupted.

The token bucket is refilled at a specified interval with some number of tokens. This interval is specified in terms of Local bus clock **LBUS_CLK** cycles.

During each **LBUS_CLK** cycle, eight tokens are drained for each Interlaken Data Word that is forwarded. This is true even for EOP Data Words that contain less than eight valid bytes.

A description of the signals that set the characteristics of the rate limiter follows. All signals are synchronous with the rising-edge of **LBUS_CLK**.

CTL_TX_RLIM_ENABLE

When this input is a value of 1, the rate limiter is enabled. When this input is a value of 0, the rate limiter is disabled.

This input should only be changed from a 0 to a 1 after appropriate values have been put on **CTL_TX_RLIM_MAX**, **CTL_TX_RLIM_DELTA**, and **CTL_TX_RLIM_INTV**.

CTL_TX_RLIM_MAX[11:0]

This input defines the maximum number of tokens in the bucket in terms of bytes (a value of 1, means 1 byte). The number of tokens in the bucket never exceed this value. This value must be at least BurstMax. (For example, if BurstMax is set for 256 bytes, this value should be at least 256).

The value of this input should not be changed when **CTL_TX_RATE_ENABLE** is a value of 1.

Note: Xilinx has observed that rates closest to the expected rates are observed when CTL_TX_RLIM_MAX is set to a value between 1 and 2 times the value of BurstMax.

CTL_TX_RLIM_INTV[7:0]

This input specifies the update interval: the number of Local bus clock cycle between additions to the token bucket. The value of this input should not be changed when **CTL_TX_RLIM_ENABLE** is a value of 1.

Note: Xilinx recommends values between 8 and 32 for this input.

CTL_TX_RLIM_DELTA[11:0]

This input specifies how many tokens are to be added to the bucket after each interval. A token is equal to 1 byte. This value must be greater than 0. The value of this input should not be changed when **CTL_TX_RLIM_ENABLE** is a value of 1.

Note: This value should be calculated based on the desired rate and the value in CTL_TX_RLIM_INTV.

Example: Programming The Rate Limiter

Assuming the following:

- The Local bus clock frequency is 200 MHz
- BurstMax is 256 bytes
- The transmission rate is to be limited to 16 Gb/s (or 2 GB/s)
- The interval is arbitrarily chosen to be 16 Local bus clock cycles

The value for **CTL_TX_RLIM_DELTA** is:

= (Byte Rate * Interval) / Local bus Frequency

= $(2^9 * 16) / (200^6)$

= 160 bytes

The value for **CTL_TX_RLIM_MAX** must be BurstMax (typically 256B) or greater. Different values result in different shaping of traffic. Simulations must be done to select the proper value to get the desired packet rate.

CRC32 Diagnostics Checking

Interlaken implements a CRC32 check for each lane of the interface for monitoring the health of each lane. All signals are synchronous with the rising-edge of **LBUS_CLK**. The Interlaken IP core uses the following two signals for this function:

STAT_RX_CRC32_VALID[11:0]

When a bit of this bus is 1, it indicates:

- The CRC32 in the most recently received Diagnostic Word on the corresponding lane was valid
- The corresponding lane is word boundary synchronized

When this bit is a value of 0, it indicates that a CRC32 error was detected or the corresponding lane is not word boundary synchronized.

STAT_RX_CRC32_ERR[11:0]

When a bit in this bus is 1, it indicates that after the corresponding lane was word boundary synchronized, a CRC32 error was detected. This output is asserted for one clock period each time a CRC32 error is detected.

Note: CRC32 errors do not affect word boundary synchronized. They are only reported as status indicators. Keep a count of how many CRC32 errors were detected for each lane to examine the health of each individual lane over a period of time.

Note: The checking of the Diagnostic Word only checks the CRC32 and does not check whether or not the unused bits of the Diagnostic Word, bits[57:34], are 0s as described in the specification.

Interlaken Status Messaging for the Receiver

The Meta Frame Diagnostic words calculate a CRC32 over all the data within the Meta Frame in a lane to help diagnose errors. The Interlaken protocol provides for optional status messaging within these Diagnostic Words. This mechanism allows a Receiver to communicate, through the adjacent Transmitter or an out-of-band flow control interfaces, the health of each (received) lane and the overall health of the Receiver interface to the other device.

The results of received Diagnostic Words are described in the following subsections. All signals are synchronous with the rising-edge of **LBUS_CLK**.

STAT_RX_DIAGWORD_INTFSTAT[11:0]

Each bit of this bus reflects the value of bit[32], the interface health (Status Bit 0), in the most recently received Diagnostic Word on the corresponding lane. The value of this bit should be considered invalid and ignored if the corresponding bit in **STAT_RX_CRC32_VALID[11:0]** is a value of 0.

STAT_RX_DIAGWORD_LANESTAT[11:0]

Each bit of this bus reflects the value of bit[33], the lane health (Status Bit 1), in the most recently received Diagnostic Word on the corresponding lane. The value of this bit should be considered invalid and ignored if the corresponding bit in **STAT_RX_CRC32_VALID[11:0]** is a value of 0.

Interlaken Status Messaging for the Transmitter

The Transmitter is capable of inserting the Status Messaging as described in the Interlaken Protocol into the Meta Frame Diagnostics words. You should feed these inputs based on the health of the receiver.

All signals are synchronous with the rising-edge of **LBUS_CLK** and a detailed description of each signal follows.

CTL_TX_DIAGWORD_INTFSTAT

This input is transmitted on bit[32], the interface health (Status Bit 0), of every Diagnostic Word on all of the lanes. A value of 1 is defined as a healthy condition.

You must drive proper data for this input. In typical applications, connect this input to the **STAT_RX_ALIGNED** output of the receiver block.

CTL_TX_DIAGWORD_LANESTAT[11:0]

Each bit of this bus is transmitted on bit[33], the lane health (Status Bit 1), of every Diagnostic Word for the corresponding lane. A value of 1 is defined to mean a healthy condition.

You must drive proper data for this input. In typical applications, connect this input to the **STAT_RX_SYNCED[11:0]** output of the receiver block.

Transmitter Multiple-Use Bits

Interlaken defines an eight-bit field in each Control Word as "Multiple-Use" bits. These bits are transmitted with every Control Word that is sent and can be used to transmit any information. For example, one of the bits can be used to represent a link-level flow control status.

The Interlaken IP core provides a mechanism to set these bits to any desired value. All signals are synchronous with the rising-edge of **LBUS_CLK** and a detailed description of each signal follows.

CTL_TX_MUBITS[7:0]

These inputs control the information contained in bits [31:24] of the Control words generated by the Transmitter. The value of **CTL_TX_MUBITS[0]** appears in bit 24 of the next Control Word generated by the TX. The value of **CTL_TX_MUBITS[1]** appears in bit 25, and so forth.

The values on **CTL_TX_MUBITS** are completely ignored when retransmission is enabled by **CTL_TX_RETRANS_ENABLE**=1.

The value of **CTL_TX_MUBITS[7]** is ignored when **CTL_TX_CHAN_EXT[1:0]** =2'h1.

The values of **CTL_TX_MUBITS[7:6]** are ignored when **CTL_TX_CHAN_EXT[1:0]** =2'h2.

The values of **CTL_TX_MUBITS[7:5]** are ignored when **CTL_TX_CHAN_EXT[1:0]** =2'h3.

Receiver Multiple-Use Bits

Similar to the Transmitter, the Interlaken IP core extracts the "Multiple-Use" field from every received Control Word and outputs the information for your interpretation. All signals are synchronous with the rising-edge of **LBUS_CLK** and a detailed description of each signal follows.

STAT_RX_MUBITS[7:0]

When **CTL_RX_RETRANS_ENABLE**=0, these outputs contain the information in bits [31:24] of the most recently received Control word. The value of Control Word bit [24] appears on **STAT_RX_MUBITS[0]**. The value of Control Word bit [25] appears on **STAT_RX_MUBITS[1]**, and so forth.

When **CTL_RX_RETRANS_ENABLE**=1, these outputs contain the information in bits [31:24] of the most recently received Burst Control word, that is, the Retransmission Sequence Number. This is the sequence number received for a burst that might or might not be forwarded to the LBUS. (**STAT_RX_RETRANS_SEQ** reflects the sequence number of bursts being forwarded to the LBUS.) Like above, the value of Control Word bit [24] appears on **STAT_RX_MUBITS[0]**. The value of Control Word bit [25] appears on **STAT_RX_MUBITS[1]**, and so forth.

When **CTL_RX_CHAN_EXT[1:0]** =2'h1, **STAT_RX_MUBITS[7]** is always a value of 1'h0.
When **CTL_RX_CHAN_EXT[1:0]** =2'h2, **STAT_RX_MUBITS[7:6]** is always a value of 2'h0.
When **CTL_RX_CHAN_EXT[1:0]** =2'h3, **STAT_RX_MUBITS[7:5]** is always a value of 3'h0.

STAT_RX_MUBITS_UPDATED

This output indicates that **stat_rx_mubits** has been updated and is asserted for one clock cycle.

Transmitter Flow-Control Inputs

The Interlaken IP core implements the Interlaken in-band flow control mechanism as described in section 5.3.4 of the Interlaken Protocol Definition 1.2. This mechanism communicates XON/XOFF (for example, for different channels) using the In-Band Flow Control bits of Control words. Additionally, the Multiple-Use bits of Control Words can be used in a similar manner.

Inside each Interlaken Control Word are 16 bits of In-Band Flow Control information, bits[55:40], and a Reset Calendar bit, bit[56]. These bits are shared over the calendar length as described in the following subsections. The Interlaken core has a fixed calendar length and provides one transmit bit and one receive bit for each calendar entry.

By definition, XON is represented by 1, and XOFF is represented by 0 for both the Transmitter and the Receiver. All signals are synchronous with the rising-edge of **LBUS_CLK** and a detailed description of each signal follows.

CTL_TX_FC_STAT[255:0]

There is full flexibility to implement any mechanism to handle system-wide flow control. The Interlaken IP core Transmitter inputs the supplied calendar information and packs it into the Interlaken Control words and transmits it over the link.

This mechanism allows you to take the system wide parameters into account and optimize the buffering by implementing the most optimum flow control mechanism.

The operation is as described in the Interlaken specification. The first calendar entry, **CTL_TX_FC_STAT[0]**, is sent in bit[55] of a Control Word with the Reset Calendar bit, bit[56], set to a value of 1. The next calendar entry, **CTL_TX_FC_STAT[1]**, is sent in bit[54] of the same Control Word and so on to bit[40]. The 17th calendar entry, **CTL_TX_FC_STAT[16]**, is sent in bit[55] of the next Control Word that has the Reset Calendar bit, bit[56], set to a value of 0, and so forth.

CTL_TX_FC_CALLEN[3:0]

The flow control calendar length can be shorter than 256 bits. When **CTL_TX_FC_CALLEN** is a value of 0, the calendar length becomes 16 and only **CTL_TX_FC_STAT[15:0]** are used. When **CTL_TX_FC_CALLEN** is a value of 1, the calendar length becomes 32 only **CTL_TX_FC_STAT[31:0]** are used. And so forth. The valid settings for calendar length are as follows:

0x00 = 16 entries

0x01 = 32 entries

0x03 = 64 entries

0x07 = 128 entries

0x0F = 256 entries

All other values are reserved.

Note: This input should be static and must only be changed during reset.

Receiver Flow-Control Outputs

The Interlaken IP core Receiver automatically extracts the flow control information received over the link and outputs the information. You can then interpret the flow control status and take the appropriate action if required.

By definition, XON is represented by 1, and XOFF is represented by 0 for both the Transmitter and the Receiver. All signals are synchronous with the rising-edge of **LBUS_CLK** and a detailed description of each signal follows.

STAT_RX_FC_STAT[255:0]

The operation is as described in the Interlaken specification. The first calendar entry, **STAT_RX_FC_STAT[0]**, is received from bit[55] of a Control Word with the Reset Calendar bit, bit[56], set to a value of 1. The next calendar entry, **STAT_RX_FC_STAT[1]**, is received from bit[54] of the same Control Word and so on. The 17th calendar entry, **STAT_RX_FC_STAT[16]**, is received from bit[55] the next Control Word that has the Reset Calendar bit, bit[56], set to a value of 0, and so forth.

Transceiver Interface

The integrated IP core for Interlaken uses the transceiver in "RAW mode" by bypassing of the encoder/decoder and gearboxes; but can use the RX elastic buffer.

Also there are restrictions and limitations as to which serial transceivers you can use to implement Interlaken. Following are the rules:

- Interlaken GTs have to be contiguous
- Interlaken on the left column must map to GTs on the left column
- Interlaken on the right column must map to GTs on the right column
- Interlaken must be implemented within an SLR

- Each Interlaken block can only connect to the GT directly adjacent and two up/down quads.

RECOMMENDED: *For transceiver selections outside of these rules, contact Xilinx Technical Support or your local FAE.*

Protocol Bypass (Lane Logic Only) Interface

The protocol bypass (lane logic only) interface helps the Integrated Interlaken core bypass the protocol logic and enables direct user access to the lane logic function. This allows you to build a fully featured Interlaken core by leveraging existing integrated lane logic and soft protocol logic.

On UltraScale devices, the maximum rate for each lane through the protocol bypass interface is 12.5 Gb/s for a maximum aggregate throughput of 150 Gb/s (12 x 12.5 Gb/s). Whereas on UltraScale+™ devices, each lane supports up to 25.78125 Gb/s through the protocol bypass interface for a maximum aggregate throughput of 300 Gb/s (12 x 25.78125 Gb/s). The interface to the transceivers is a 64-bit "RAW" interface, with the gearbox function performed in the Interlaken IP core. All lanes support 64B/67B encoding/decoding, synchronization and scrambling required by the Interlaken Protocol.

On Virtex® UltraScale+ devices, the protocol bypass (lane logic only) interface can be used along with Xilinx soft protocol logic to build Interlaken cores with overall bandwidth of up to 300 Gb/s. For more details, see the *Interlaken 600G v1.2 LogiCORE IP Product Guide* (PG209) [Ref 20].

Retransmission Feature

The retransmission feature is used to automatically re-transmit bursts in the event of a CRC24 error. The retransmission feature of the Integrated Interlaken IP core implements retransmission as defined in the Interlaken Retransmit Extension v1.2 from the Interlaken alliance [Ref 2].

Bursts need to be stored in a buffer in case they need to be re-transmitted. Because the number of bursts to be maintained is implementation-specific, the memory to store the bursts is external to the Integrated Interlaken IP core.

Retransmission is signaled over the slower out-of-band flow control (OOBFC) interface which has its own independent CRC checking. At least one bit in the OOBFC calendar is assigned to request retransmission. The position of this bit must be known to both ends of the interface.

The retransmission feature only works in burst-interleaved mode, as controlled by **CTL_RX_PACKET_MODE**. Retransmission is enabled with the **CTL_TX_RETRANS_ENABLE** and **CTl_RX_RETRANS_ENABLE** signals.

IMPORTANT: *When the RX retransmission is enabled (CTL_RX_RETRANS_ENABLE = 1), the RX must operate in the burst-interleaved mode (CTL_RX_PACKET_MODE = 0).*

When the TX retransmission is enabled (**CTL_TX_RETRANS_ENABLE**=1), you should not stall in the middle of a burst on the LBUS if **tx_rdyout** is asserted. This is required for reliable operation of retransmission.

Multiplier

The Interlaken Retransmission Protocol Extension v1.2 defines a sequence number multiplier. The sequence number is contained in the Multiple-Use bits of a Burst Control Word. The multiplier causes the same sequence number to be transmitted for multiple bursts thus implying a sub-sequence number.

The Channel Extension bits can also be contained in the Multiple-Use bits of a Burst Control Word. This can limit the maximum number of sequence numbers. The multiplier extends the number of bursts in a sequence. For example, if the channel extension is set to 256 channels and the sequence multiplier to x32, the maximum sequence length is 8192 bursts. However, if the channel extension is set to 2048, the maximum sequence length is 1024 bursts.

The Interlaken integrated IP core supports sequence number multipliers of 1, 2, 4, 8, 16, and 32.

Note: The corresponding link partner TX and RX cores should be configured with the same multiplier value.

Sequence Discontinuity

Key to the robust operation of the retransmission protocol is the correct generation and detection of a valid sequence discontinuity. A valid sequence discontinuity is the mechanism the TX uses to indicate that it has begun retransmission and the mechanism the RX uses to identify the start of retransmission.

A valid sequence discontinuity is defined as follows:

"A sequence discontinuity is an unexpected change in primary sequence numbers between Burst Control Words without any CRC24 errors in between."

For example, suppose a burst with a primary sequence number of 157 is received, followed by numerous Idle Control Words without CRC24 errors, followed by another burst with a primary sequence number of 123. This is a sequence discontinuity. Zero or more Idle Control Words without CRC24 errors can appear between the pair of Burst Control Words.

If one or more of the Idle Control Words in between the pair of Burst Control Words has a CRC24 error, the pair is not recognized as a valid sequence discontinuity.

This definition of a valid sequence discontinuity is unambiguous and ensures robust operation. If the transmitter follows the preceding rule, it will generate sequence discontinuities that can be positively identified.

If a valid sequence discontinuity cannot be positively identified, the RX will again request retransmission if configured to do so.

Latency

The retransmission protocol relies on a circular buffer in the transmitter and possible re-requests in order to positively identify a discontinuity. Sizing the buffer and configuring the timer(s) depends on latency in the system — the time from request for retransmission to the receipt of a sequence discontinuity. This parameter is referred to as the request-to-discontinuity latency and is usually measured in terms of Interlaken words.

Rate Limiter

As correct operation of the retransmission protocol is dependent upon latency, Xilinx recommends that the rate limiter in the TX not be used when retransmission mode is enabled.

IMPORTANT: *Xilinx recommends that the rate limiter in the TX should not be used when retransmission mode is enabled.*

Transmit

TX retransmission logic contains a circular buffer and an optional error injector.

The TX retransmission buffer memory is separate and not integrated into the main IP. It connects to the Integrated Interlaken IP core via a set of RAM ports.

Connecting the TX Retransmission Buffer RAM

Data to be transmitted is written into the TX LBUS interface. When the **CTL_TX_RETRANS_ENABLE** is enabled, the TX retransmission logic writes the TX data into the retransmission buffer using the signals **STAT_TX_RETRANS_RAM_WDATA**, **STAT_TX_RETRANS_RAM_WE_Bx**, and **STAT_TX_RETRANS_RAM_WADDR**. Data is read from the buffer using the signals **CTL_TX_RETRANS_RAM_RDATA**, **STAT_TX_RETRANS_RAM_RD_Bx**, and **STAT_TX_RETRANS_RAM_RADDR**.

The buffer can contain up to four banks. Each bank consists of 11 parallel instances of 64x512 RAMB36 SDP block RAM to create a memory that is 644 bits wide and 512 entries deep. Each RAM is configured with DO_REG=1. The number of banks required depends upon latency and is selected using the inputs **CTL_TX_RETRANS_RAM_BANKS[1:0]**.

All banks share a common write bus and write address. Each bank has its own write enable signal: **STAT_TX_RETRANS_RAM_WE_Bx**.

All banks share a common read address. Each bank has its own read enable signal: **STAT_TX_RETRANS_RAM_RD_Bx**. The read output of each bank goes into a 4:1 multiplexer. Bank 0 is selected when the multiplexer select signals are a value of 0. Bank 1 is selected when the multiplexer select signals are a value of 1. And so forth. The signals **STAT_TX_RETRANS_RAM_RSEL** are used to control the multiplexer.

The output of the multiplexer goes to a flip-flop pipeline stage which need not be reset. The output of pipeline stage goes to **CTL_TX_RETRANS_RAM_RDATA**.

See Figure 3-5.

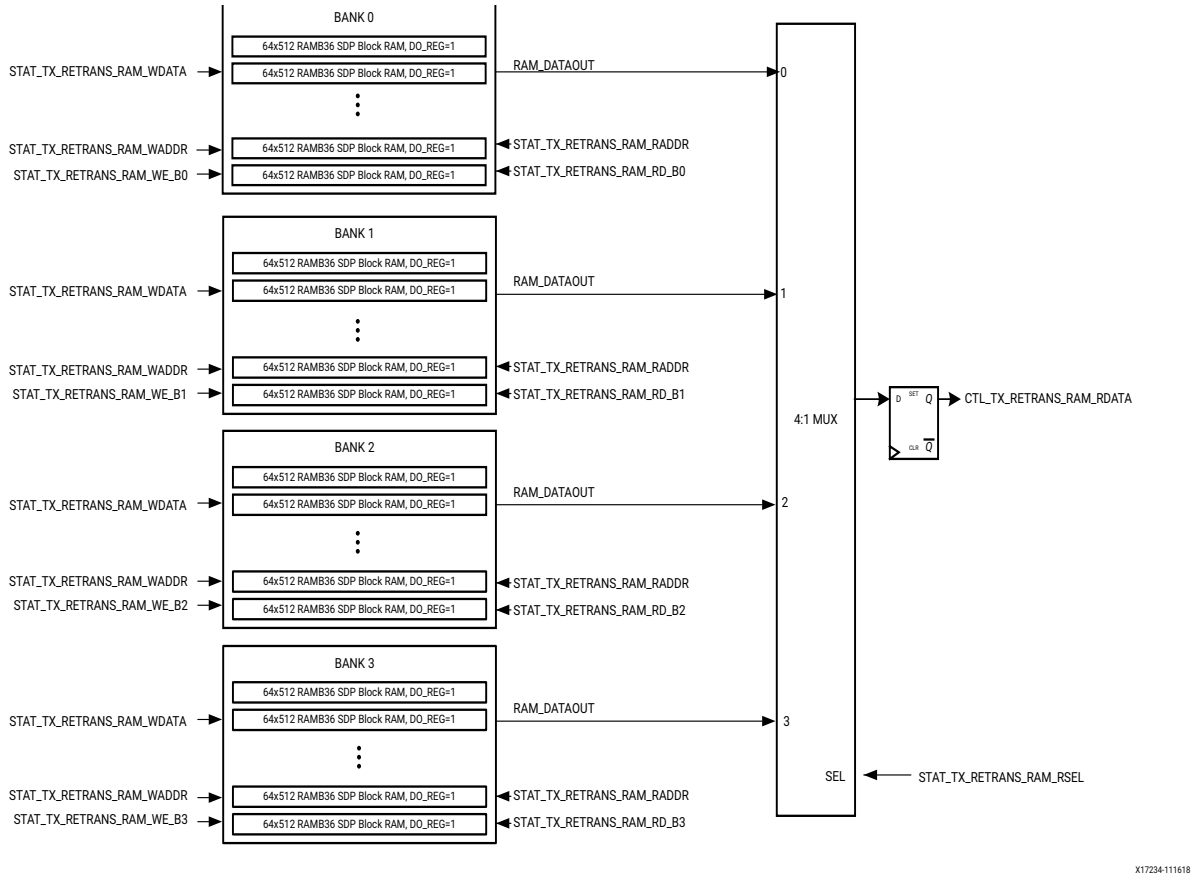


Figure 3-5: **TX Retransmission Buffer RAM Connections**

In UltraScale+ devices, the TX retransmission logic is enhanced to allow up to two stages of pipeline in the fabric for all signals on the write path to the RAM banks.

Configuring the TX Retransmission Buffer Depth

The input **CTL_TX_RETRANS_DEPTH** (which is set through the DRP interface) is used to set the maximum number of eight Interlaken word entries to be stored in the retransmission buffer. This value should be between two and three times the greater of the values assigned to **CTL_RX_RETRANS_TIMER1** and **CTL_RX_RETRANS_TIMER2** and then divided by eight.

For example, if **CTL_RX_RETRANS_TIMER1** has a value of 20 and **CTL_RX_RETRANS_TIMER2** has a value of 2000, the **CTL_TX_RETRANS_DEPTH** should be set to a value of about 625. Similarly, if **CTL_RX_RETRANS_TIMER1** has a value of 2500 and **CTL_RX_RETRANS_TIMER2** has a value of 0, the **CTL_TX_RETRANS_DEPTH** should be set to a value of about 782.

There must be sufficient depth in the TX retransmission buffer to contain at least five different sequence numbers when maximum sized bursts are stored. The total number of bursts required is affected by the multiplier in use. Thus, the minimum depth of the TX retransmission buffer can be found by satisfying the following equation:

CTL_TX_RETRANS_DEPTH > (multiplier * 33 * 5) / 8

For example, if the multiplier is 1, the minimum number of entries required for the retransmission buffer depth is 21. If the multiplier is 2, the minimum number of entries required for the retransmission buffer depth is 42. And so forth.

Generally, the minimum depth requirement is easily met by the latency in the system.

Selecting a Sequence Multiplier

The best value for the sequence multiplier is one that is no larger than it needs to be but ensures that there is no wrap-around of sequence numbers as contained in the available Multiple-Use bits of a Burst Control Word.

If the channel extension is set to 256 channels, the multiplier is found by satisfying the following equation:

multiplier * 126 < CTL_TX_RETRANS_DEPTH *AND*

multiplier * 252 >= CTL_TX_RETRANS_DEPTH

If the multiplier is 1, the buffer should be no deeper than 252 which ensures no sequence number appears more than once in the buffer when minimum sized bursts are stored. If the multiplier is 2, the buffer should be no deeper than 504 which ensures no sequence number appears more than twice in the buffer when minimum sized bursts are stored.

If the multiplier is 4, the buffer should be no deeper than 1008 which ensures no sequence number appears more than four times in the buffer when minimum sized bursts are stored. And so forth.

If the channel extension is set to something greater than 256 channels, the number of bits available for sequence numbers in the Multiple-Use bits of a Burst Control Word is reduced. Thus, if the channel extension is set to 512 channels, the multiplier is found by satisfying the following:

multiplier * 62 < CTL_TX_RETRANS_DEPTH *AND*

multiplier * 124 >= CTL_TX_RETRANS_DEPTH

If the channel extension is set to 1024 channels, the multiplier is found by satisfying the following:

multiplier * 30 < CTL_TX_RETRANS_DEPTH *AND*

multiplier * 60 >= CTL_TX_RETRANS_DEPTH

If the channel extension is set to 2048 channels, the multiplier is found by satisfying the following:

multiplier * 14 < CTL_TX_RETRANS_DEPTH *AND*

multiplier * 28 >= CTL_TX_RETRANS_DEPTH

Both the TX and RX link partners must be assigned the same multiplier.

IMPORTANT: *Sequence multipliers of 1 and 2 are aggressive settings that require careful analysis of the request-to-discontinuity latency in the system. You must use the technique described in the Measuring Request-to-Discontinuity Latency section to measure the request-to-discontinuity and set the retransmission parameters accordingly.*

Error Injector

The Integrated Interlaken IP core includes the ability to inject a bit error as recommended in section 2.6 of the Interlaken Retransmission Protocol Extension v1.2.

This error injector forces a pseudo-random single bit error on the selected lane without regard for regular Control Words, Data Words, or Meta Frame Words. Consequently, the injected error might not necessarily result in a CRC24 error that will cause a request for retransmission.

When the RX is aligned, at least one of the following signals will indicate that a bit error was detected:

- **STAT_RX_CRC32_ERR**
- **STAT_RX_FRAMING_ERR**
- **STAT_RX_DESCRAM_ERR**

For example, if a single bit error was injected on lane 3, at least one of:

STAT_RX_CRC32_ERR[3], or

STAT_RX_FRAMING_ERR[3], or

STAT_RX_DESCRAM_ERR[3]

on the RX will be asserted for a single clock cycle. Retransmission only occurs if a CRC24 error is detected in a regular Control or Data Word.

The following ports on the Interlaken core are associated with the single bit error injector feature:

- **CTL_TX_ERRINJ_BITERR_GO**
- **CTL_TX_ERRINJ_BITERR_DONE**
- **CTL_TX_ERRINJ_BITERR_LANE[3:0]**

A lane is selected via **CTL_TX_ERRINJ_BITERR_LANE**. **CTL_TX_ERRINJ_BITERR_GO** is asserted for a single clock cycle. When **CTL_TX_ERRINJ_BITERR_DONE** is a value of 1 again, you can change the value of **CTL_TX_ERRINJ_BITERR_LANE** and initiate another error with **CTL_TX_ERRINJ_BITERR_GO** again.

Connection to Out-Of-Band Flow Control Transmitter

The input **CTL_TX_RETRANS_REQ** should be connected to one of the **rx_fc** outputs on the RX Out-Of-Band Flow Control module. The corresponding **rx_update** signal should be connected to the **CTL_TX_RETRANS_REQ_VALID** input.

Retransmission is only initiated when retransmission is enabled, there is sufficient data in the buffer, and both **CTL_TX_RETRANS_REQ** and **CTL_TX_RETRANS_REQ_VALID** are a value of 1 in the same clock cycle. A subsequent request for retransmission will be ignored unless it is proceeded by a clock cycle in which **CTL_TX_RETRANS_REQ** is a value of 0 and **CTL_TX_RETRANS_REQ_VALID** is a value of 1. This is in keeping with the requirement in section 2.2.1 of the Interlaken Retransmission Protocol Extension v1.2 which states:

"The bit in the calendar must be deasserted, set to Xoff, before another retransmit request can be signaled."

Burst Error

When the TX begins switching from sending standard data to retransmitted data, the last burst of standard data might contain a burst length error. This will be signaled by the **STAT_TX_BURST_ERR** signal on the Interlaken core and can be ignored. However, should **STAT_TX_BURST_ERR** be asserted at any other time, it indicates a badly formed burst might be transmitted.

Receive

The RX retransmission buffer is integrated within the Integrated Interlaken IP block.

Start-up

After reset, the retransmission logic in the RX does not become active immediately. It must receive several bursts so it can correctly determine the implied sub-sequence numbers.

The RX is self-synchronizing and does not require that the first burst sequence number it receives be 0 or some other value.

Measuring Request-to-Discontinuity Latency

In order for the retransmission logic in the RX to operate correctly, the inputs **CTL_RX_RETRANS_TIMER1** and **CTL_RX_RETRANS_TIMER2** must be configured with suitable values based on the maximum request-to-discontinuity latency.

Each time the RX makes a request for retransmission, a counter begins counting from 0 and continues until a sequence discontinuity is detected or the counter reaches its limit. The result of this counter is available on the port **STAT_RX_RETRANS_LATENCY[15:0]**.

A request for retransmission can be forced by asserting the input **CTL_RX_RETRANS_FORCE_REQ** for a single clock cycle.

In order to measure the request-to-discontinuity latency, the TX core must be configured appropriately. Some RAM needs to be available for buffering and **CTL_TX_RETRANS_ENABLE** must be a value of 1. The TX must respond to **CTL_TX_RETRANS_REQ** with the assertion of **STAT_TX_RETRANS_BUSY**. It is not necessary to have the correct amount of RAM to determine the latency. All that has to happen is that the TX core generate a discontinuity in response to a request for retransmission.

The timers **CTL_RX_RETRANS_TIMER1** and **CTL_RX_RETRANS_TIMER2** need not be set in order to determine the latency. A retransmission multiplier of 16 or 32 can be chosen. When the latency is determined, these values and the TX buffer depth can be adjusted.

The request-to-discontinuity latency can be measured using the following technique:

- A retransmission capable TX continually sends packets.
- **CTL_RX_RETRANS_RETRY** is initialized to a value of 0.
- Assign **CTL_RX_RETRANS_RESET_MODE** a value of 0.
- Wait until **STAT_RX_RETRANS_STATE** equals **3'b001**.
- Assert **CTL_RX_RETRANS_FORCE_REQ** for a single clock cycle.
- Wait for **STAT_RX_RETRANS_RETRY_ERR** to be asserted.
- Assert **CTL_RX_RETRANS_RESET** for a single clock cycle.
- Read the resulting value from **STAT_RX_RETRANS_LATENCY**.
- Assign **CTL_RX_RETRANS_RESET_MODE** a value of 1.
- Assert **CTL_RX_RETRANS_ERRIN** for a single clock cycle.
- Assert **CTL_RX_RETRANS_RESET** for a single clock cycle.

When **STAT_RX_RETRANS_RETRY_ERR** is asserted, **STAT_RX_RETRANS_STATE** should equal **3'b110** and **STAT_RX_RETRANS_LATENCY** should be cleared to a value of **16'h0000**. With **CTL_RX_RETRANS_RESET_MODE** equal to 0, asserting **CTL_RX_RETRANS_RESET** a single cycle should cause **STAT_RX_RETRANS_STATE** to change to **3'b111** and then to **3'b100** and **STAT_RX_RETRANS_LATENCY** should be updated with the measured latency. When the latency is read, asserting **CTL_RX_RETRANS_ERRIN** should cause **STAT_RX_RETRANS_STATE** to change to **3'b110**. With **CTL_RX_RETRANS_RESET_MODE** equal to 1, asserting **CTL_RX_RETRANS_RESET** a single cycle should cause **STAT_RX_RETRANS_STATE** to change to **3'b111** and then to **3'b000**.

The resulting value of **STAT_RX_RETRANS_LATENCY** is the number of Interlaken Words received during the interval between the request for retransmission and the receipt of a valid sequence discontinuity or in other words, the request-to-discontinuity latency. The process should be repeated several times and the maximum value recorded to configure the timers.

Timer Modes and Configuration

There are two timers that govern the operation of the retransmission logic in the RX:

- timer1 (Short Timer), configured with the input **CTL_RX_RETRANS_TIMER1**, and
- timer2 (Long Timer), configured with the input **CTL_RX_RETRANS_TIMER2**

These timers are configured in terms of Interlaken words that are received during the desired period. For example, assume a configuration of 12 lanes at 5 Gb/s. Each Interlaken word received represents a period of:

= (67 * 200 ps) / 12

= 1.12 ns

Therefore to program a timer with a period of 224 ns, it should be configured with a value of 200.

There are two ways to use the timers.

Mode 1

This is the mode of operation if both timer1 and timer2 are configured with non-zero values. Both timers begin counting when a CRC24 error is detected and a request for retransmission is signaled.

In this mode, timer1 ignores "any possible short term secondary errors", including CRC24 errors, and also valid sequence discontinuities. If timer2 expires before a valid sequence discontinuity is detected, a re-request for retransmission immediately occurs. If a CRC24 error is detected after timer1 has expired but timer2 has not expired, another request for retransmission immediately occurs.

CTL_RX_RETRANS_TIMER1 should be set to a value much less than the minimum request-to-discontinuity latency. However, if it is too low, it will not serve its intended purpose.

CTL_RX_RETRANS_TIMER2 should be set to a value greater than the maximum request-to-discontinuity latency.

Mode 2

This is the mode of operation if **CTL_RX_RETRANS_TIMER1** is set to a non-zero value and **CTL_RX_RETRANS_TIMER2** is set to a value of zero.

In this mode, while timer1 is counting, all CRC24 errors are ignored but valid sequence discontinuities are handled. This mode of operation is simpler because a valid sequence discontinuity is unambiguous and never contains any "short term secondary errors".

In this mode, **CTL_RX_RETRANS_TIMER1** should be set to a value greater than the maximum request-to-discontinuity latency and **CTL_RX_RETRANS_TIMER2** to a value of zero.

Note: Xilinx strongly recommends configuring the timers in Mode 2.

CRC24 Errors

When aligned, every bit error received by the RX is identified by the assertion of one of the following:

- **STAT_RX_CRC32_ERR**
- **STAT_RX_FRAMING_ERR**
- **STAT_RX_DESCRAM_ERR**

However, only errors that occur in regular Control and Data Words will result in a request for retransmission and these are identified by the assertion of **STAT_RX_RETRANS_CRC24_ERR**.

The purpose of the retransmission logic is to reduce the number of packets lost due to CRC24 errors. However, CRC24 errors are forwarded to the LBUS and handled as they would be in non-retransmission mode under the following circumstances:

- a fatal error occurs (retry timeout, etc.)
- a CRC24 error occurs during initialization

In these cases the signal **STAT_RX_CRC24_ERR** is asserted to indicate the error.

Sequence Numbers

In retransmission mode, the port **STAT_RX_MUBITS** reflects the value of bits 31:24 of the most recently received Burst Control Word or in other words, the most recently received primary sequence number. The signal **STAT_RX_MUBITS_UPDATED** indicates an update on the port **STAT_RX_MUBITS**.

The outputs **STAT_RX_RETRANS_SEQ** and **STAT_RX_RETRANS_SUBSEQ** reflect the last good primary sequence and implied sub-sequence number forwarded to the RX LBUS.

The signal **STAT_RX_RETRANS_SEQ_UPDATED** indicates an update on the ports **STAT_RX_RETRANS_SEQ** and **STAT_RX_RETRANS_SUBSEQ**.

By comparing the values on these ports, the retransmission operation can be easily observed.

Note: The value of the implied sub-sequence number on the port **STAT_RX_RETRANS_SUBSEQ** will not be correct until after the sequence has been established.

States

When retransmission is enabled, the RX operates in eight different states. These can be monitored on the port **STAT_RX_RETRANS_STATE**.

- State 000: Waiting for the initial bursts, forwarding data to RX LBUS
- State 001: Normal mode, forwarding data to RX LBUS
- State 010: A CRC24 error was detected and timer1 is counting
- State 011: Waiting for a Discontinuity while timer2 is counting
- State 100: Waiting to re-establish the implied burst sub-sequence
- State 101: Waiting for the expected sequence
- State 110: Fatal Error, waiting for assertion of **CTL_RX_RETRANS_RESET**
- State 111: Fatal Error, waiting for negation of **CTL_RX_RETRANS_RESET**

Fatal Errors

There are several kinds of fatal errors from which the RX cannot recover from without intervention. These include loss of alignment, wrap around errors, and too many retries.

In order to recover from a fatal error, the signal **CTL_RX_RETRANS_ERRIN** must be negated and the signal **CTL_RX_RETRANS_RESET** must be asserted for several LBUS clock cycles and then negated. This action marks any open packets as having an error. Thereafter, the RX will continue to wait for the expected sequence or start as if it was reset depending upon the setting of **CTL_RX_RETRANS_RESET_MODE**.

Loss of Alignment

The retransmission protocol does not specify what should happen during a loss of alignment and there is no obvious action the RX should take in such an event. Therefore, RX treats this as a fatal error if the output **STAT_RX_ALIGNED_ERR** is connected to the input **CTL_RX_RETRANS_ERRIN**. Otherwise loss of alignment is treated as a CRC24 error.

Retry Timer

RX needs to perform retries in the event it cannot correctly identify a sequence discontinuity. Therefore, the input **CTL_RX_RETRANS_RETRY** must be configured with a value of at least 2 for a reliable operation.

RX will re-request retransmission for a single error up to the value configured by **CTL_RX_RETRANS_RETRY**. Thereafter **STAT_RX_RETRANS_RETRY_ERR** will be asserted and treated as a fatal error.

Wrap Around Timer

The only kind of wrap-around error the RX can consistently detect is one that occurs after having received a discontinuity and then too many changes in the primary sequence numbers occur before the expected sequence arrives.

The value that should be assigned to **CTL_RX_RETRANS_WRAP_TIMER** is a value greater than the total number of primary sequence numbers that can be stored in the TX retransmission buffer. A value should be chosen that satisfies the following equation:

multiplier * **CTL_RX_RETRANS_WRAP_TIMER** > **CTL_TX_RETRANS_DEPTH**

where multiplier is the sequence multiplier, that is, one of 1, 2, 4, 8, 16, or 32.

If the channel extension is set to 256 channels, then the value for **CTL_RX_RETRANS_WRAP_TIMER** should be a non-zero value less than 252.

If the channel extension is set to 512 channels, then the value for **CTL_RX_RETRANS_WRAP_TIMER** should be a non-zero value less than 124.

If the channel extension is set to 1024 channels, then the value for **CTL_RX_RETRANS_WRAP_TIMER** should be a non-zero value less than 60.

If the channel extension is set to 2048 channels, then the value for **CTL_RX_RETRANS_WRAP_TIMER** should be a non-zero value less than 28.

After the wrap-around condition is detected, **STAT_RX_RETRANS_WRAP_ERR** is asserted and treated as a fatal error.

If unused, **CTL_RX_RETRANS_WRAP_TIMER** should be programmed with a value of 0.

Watchdog Timer

The 32-bit watchdog timer is an optional timer which counts incoming Interlaken words while waiting for the expected sequence after a discontinuity has been detected. If the watchdog timer expires before the expected sequence is detected, **STAT_RX_RETRANS_WDOG_ERR** is asserted and treated as a fatal error. If unused, **CTL_RX_RETRANS_WDOG** should be programmed with a value of 0.

Connection to Out-Of-Band Flow Control Transmitter

The input **CTL_RX_RETRANS_ACK** should be connected to one of the **tx_update** outputs of the TX Out-Of-Band Flow Control module. The **tx_update** signal is used to clear the **STAT_RX_RETRANS_REQ** signal so the next message sent by the TX Out-Of-Band Flow Control module will not have the request bit set which is in keeping with the requirement in section 2.2.1 of the Interlaken Retransmission Protocol Extension v1.2 which states:

"The bit in the calendar must be deasserted, set to Xoff, before another retransmit request can be signaled."

The output **STAT_RX_RETRANS_REQ** should be connected to one of the **tx_fc** inputs of the TX Out-Of-Band Flow Control module that corresponds to the selected **tx_update** signal. The length of the calendar affects the request-to-discontinuity latency. It might be desirable to have a separate Out-Of-Band Flow Control interface for retransmission requests.

Burst Error

When the TX switches from sending standard data to retransmitted data, the last burst of standard data can contain a burst length error. This is signaled by the **STAT_RX_BURST_ERR** signal but ignored for all other purposes. Should **STAT_RX_BURST_ERR** be asserted at any other time, it indicates a burst length error that will be treated like a CRC24 error.

Dynamic Reconfiguration Port

The dynamic reconfiguration port (DRP) allows the dynamic change of attributes in the integrated IP core for Interlaken. The DRP interface is a processor-friendly synchronous interface with an address bus (**DRP_ADDR**) and separated data buses for reading (**DRP_DO**) and writing (**DRP_DI**) configuration data to the ILKN block. An enable signal (**DRP_EN**), a read/write signal (**DRP_WE**), and a ready/valid signal (**DRP_RDY**) are the control signals that implement read and write operations, indicate operation completion, or indicate the availability of data.

For the DRP to work, a clock must be provided to the **DRP_CLK** port. See the following data sheets for the maximum allowed clock frequencies.

- *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS893) [Ref 5]
- *Virtex UltraScale+ Architecture Data Sheet: DC and AC Switching Characteristics* (DS923) [Ref 17]
- *Kintex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS892) [Ref 17]
- *Kintex UltraScale+ Architecture Data Sheet: DC and AC Switching Characteristics* (DS922) [Ref 19]

The ILKN block must be held in reset when you want to dynamically change the attributes through the DRP. That is, **TX_RESET**, **RX_RESET**, **TX_SERDES_REFCLK_RESET**, and the **RX_SERDES_RESET[11:0]** signals need to be asserted High.

Dynamic Reconfiguration Port Interface

The DRP interface provides a means to change attribute values by over-writing memory cells without re-configuring the entire FPGA or using partial reconfiguration. The Interlaken core must be held in reset while the DRP port is used, and all external timing requirements must be met. The memory cells is defined as multi-cycle paths in core timing analysis, and the DRP requires several clock cycles before the new values can be read (indicated by **DRP_RDY** assertion).The new set of attribute values must contain a legal configuration of the Interlaken core before the core is brought out of the reset state.

The DRP clock can be any continuously running clock not exceeding 250 MHz.

DRP Write Operation

Figure 3-6 shows the DRP write operation timing diagram. New DRP operations can be initiated when the **DRP_RDY** signal is asserted.

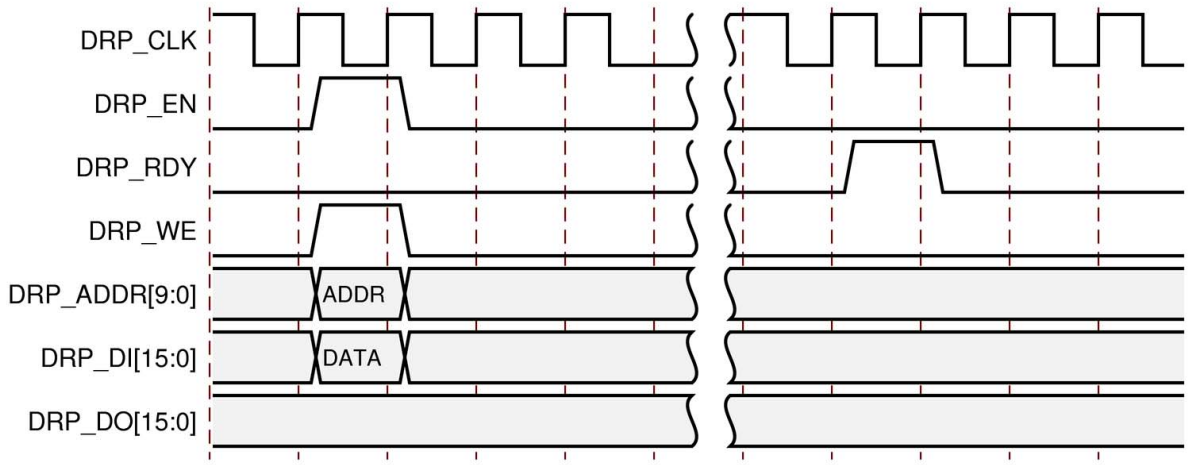


Figure 3-6: **DRP Write Operation Timing Diagram**

DRP Read Operation

Figure 3-7 shows the DRP read operation timing diagram. New DRP operations can be initiated when the **DRP_RDY** signal is asserted.

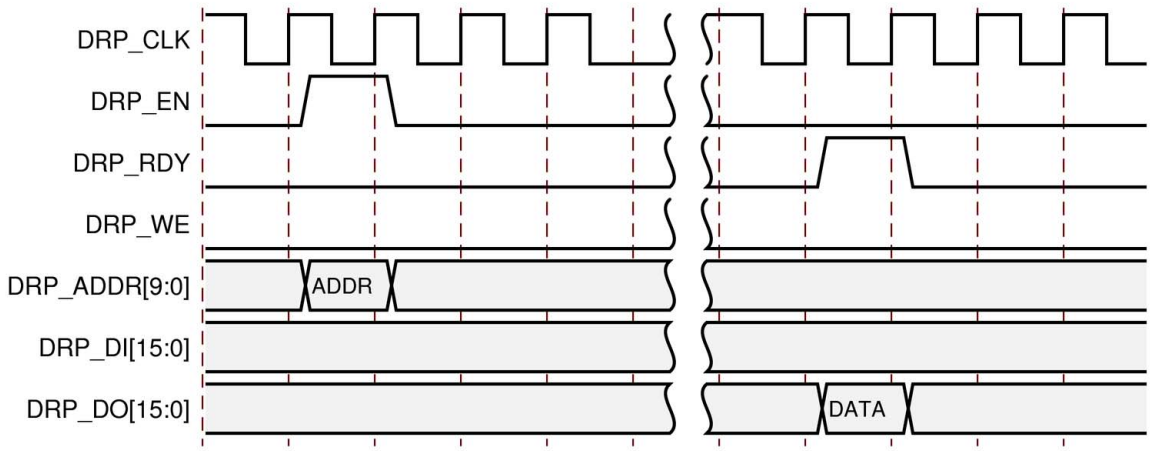


Figure 3-7: **DRP Read Operation Timing Diagram**

DRP Address Map of the ILKN Block in the UltraScale Device Architecture

Table 3-6 lists the DRP map of the ILKN block in UltraScale devices sorted by address. All other addresses are reserved.

Table 3-6: **DRP Map for UltraScale Devices**

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
8	[3:0]	R/W	CTL_TX_LAST_LANE[3:0]	0-B	0-B
10	[6:0]	R/W	CTL_TX_FC_CALLEN[6:0]	0-F	0-F
18	0	R/W	CTL_TX_DISABLE_SKIPWORD	FALSE	0
				TRUE	1
20	[15:0]	R/W	CTL_TX_MFRAMELEN_MINUS1[15:0]	FF-1FFF	FF-1FFF
28	[1:0]	R/W	CTL_TX_BURSTMAX[1:0]	0-3	0-3
30	[2:0]	R/W	CTL_TX_BURSTSHORT[2:0]	1-3	1-3
38	[3:0]	R/W	CTL_RX_LAST_LANE[3:0]	0-B	0-B
40	[15:0]	R/W	CTL_RX_MFRAMELEN_MINUS1[15:0]	FF-1FFF	FF-1FFF
48	[1:0]	R/W	CTL_RX_BURSTMAX[1:0]	0-3	0-3
50	0	R/W	MODE	FALSE	0
				TRUE	1

Table 3-6: **DRP Map for UltraScale Devices *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
58	0	R/W	BYPASS	FALSE	0
				TRUE	1
68	[13:0]	R/W	CTL_TX_RETRANS_DEPTH[13:0]	200-800	200-800
70	[2:0]	R/W	CTL_TX_RETRANS_MULT[2:0]	0-5	0-5
78	0	R/W	CTL_RX_PACKET_MODE	FALSE	0
				TRUE	1
80	[1:0]	R/W	CTL_RX_CHAN_EXT[1:0]	0-3	0-3
98	[2:0]	R/W	CTL_RX_RETRANS_MULT[2:0]	0-5	0-5
A0	[15:0]	R/W	CTL_RX_RETRANS_TIMER1[15:0]	9-FFFF	9-FFFF
A8	[15:0]	R/W	CTL_RX_RETRANS_TIMER2[15:0]	0-FFFF	0-FFFF
B0	[3:0]	R/W	CTL_RX_RETRANS_RETRY[3:0]	2-F	2-F
B8	[7:0]	R/W	CTL_RX_RETRANS_WRAP_TIMER[7:0]	0-FF	0-FF
C0	[11:0]	R/W	CTL_RX_RETRANS_WDOG[11:0]	0-FFF	0-FFF
C8	[1:0]	R/W	CTL_TX_CHAN_EXT[1:0]	0-3	0-3
D0	[1:0]	R/W	CTL_TX_RETRANS_RAM_BANKS[1:0]	0-3	0-3

Table 3-7 lists the DRP map of the ILKN block in UltraScale+ devices sorted by address. All other addresses are reserved.

Table 3-7: **DRP Address Map of the ILKN block in UltraScale+ Devices**

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
6	[3:0]	R/W	CTL_TX_LAST_LANE[3:0]	0-B	0-B
C	[3:0]	R/W	CTL_TX_FC_CALLEN[3:0]	0-F	0-F
12	0	R/W	CTL_TX_DISABLE_SKIPWORD	FALSE	0
				TRUE	1
18	[15:0]	R/W	CTL_TX_MFRAMELEN_MINUS1[15:0]	FF-1FFF	FF-1FFF
1E	[1:0]	R/W	CTL_TX_BURSTMAX[1:0]	0-3	0-3
24	[2:0]	R/W	CTL_TX_BURSTSHORT[2:0]	1-3	1-3
2A	[3:0]	R/W	CTL_RX_LAST_LANE[3:0]	0-B	0-B
30	[15:0]	R/W	CTL_RX_MFRAMELEN_MINUS1[15:0]	FF-1FFF	FF-1FFF
36	[1:0]	R/W	CTL_RX_BURSTMAX[1:0]	0-3	0-3

Table 3-7: **DRP Address Map of the ILKN block in UltraScale+ Devices *(Cont'd)***

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
3C	0	R/W	MODE	FALSE	0
				TRUE	1
42	0	R/W	BYPASS	FALSE	0
				TRUE	1
4E	[13:0]	R/W	CTL_TX_RETRANS_DEPTH[13:0]	200-800	200-800
54	[2:0]	R/W	CTL_TX_RETRANS_MULT[2:0]	0-5	0-5
5A	0	R/W	CTL_RX_PACKET_MODE	FALSE	0
				TRUE	1
60	[1:0]	R/W	CTL_RX_CHAN_EXT[1:0]	0-3	0-3
72	[2:0]	R/W	CTL_RX_RETRANS_MULT[2:0]	0-5	0-5
78	[15:0]	R/W	CTL_RX_RETRANS_TIMER1[15:0]	9-FFFF	9-FFFF
7E	[15:0]	R/W	CTL_RX_RETRANS_TIMER2[15:0]	0-FFFF	0-FFFF
84	[3:0]	R/W	CTL_RX_RETRANS_RETRY[3:0]	2-F	2-F
8A	[7:0]	R/W	CTL_RX_RETRANS_WRAP_TIMER[7:0]	0-FF	0-FF
90	[11:0]	R/W	CTL_RX_RETRANS_WDOG[11:0]	0-FFF	0-FFF
96	[1:0]	R/W	CTL_TX_CHAN_EXT[1:0]	0-3	0-3
9C	[1:0]	R/W	CTL_TX_RETRANS_RAM_BANKS[1:0]	0-3	0-3

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 the this IP core. More detailed information about the standard Vivado® design flows and the 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 8]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 10]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 11]

Customizing and Generating the Core

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

The integrated IP core for Interlaken is generated using the Vivado IP catalog. 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 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 9] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 10].

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

When the integrated IP core for Interlaken is selected from the IP catalog, a window displays showing the different available configurations. These are organized in various tabs for better readability and configuration purposes. The details related to these tabs are described in the next sections.

General Tab

Figure 4-1 shows the general customization options for the IP.

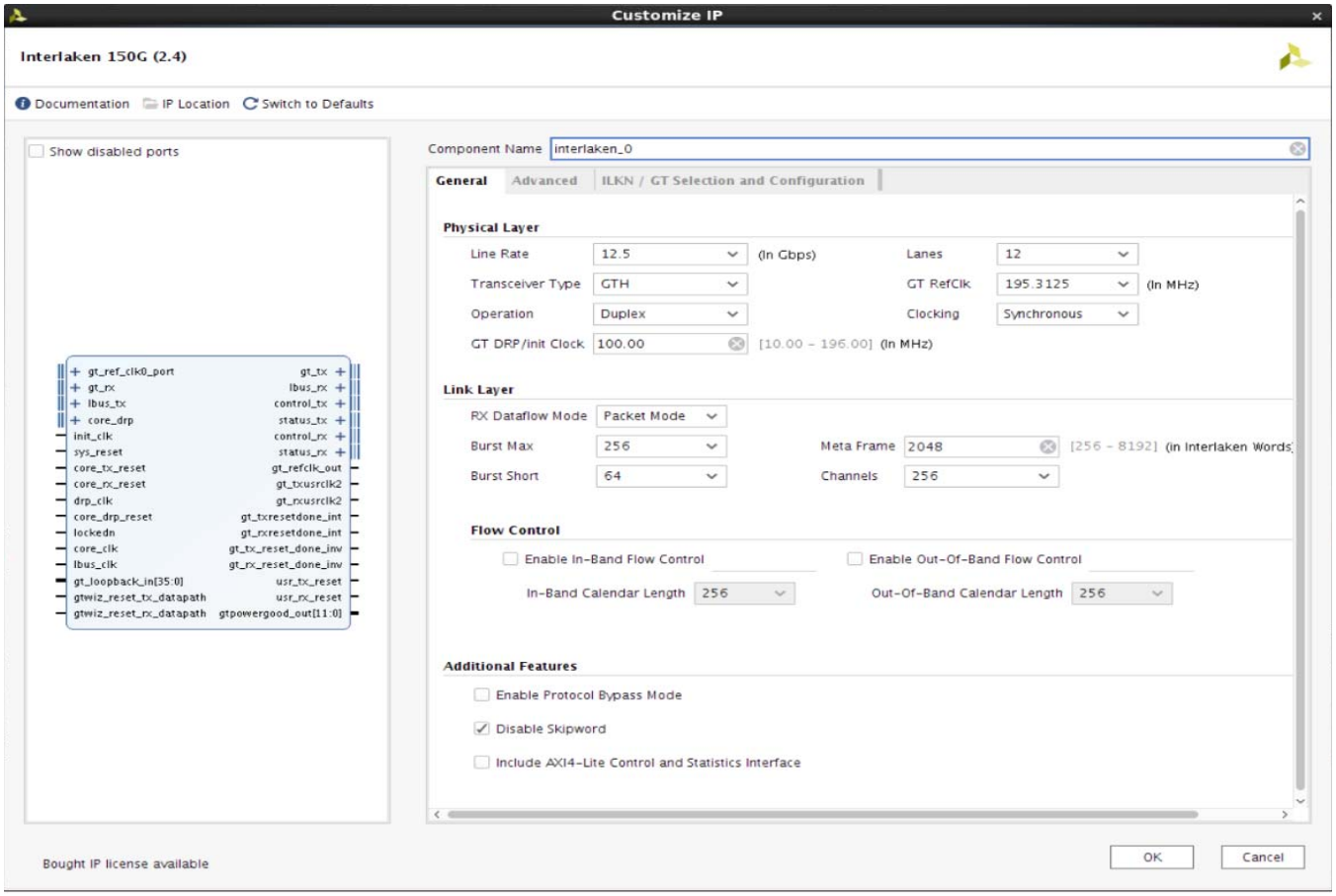


Figure 4-1: **General Tab**

Table 4-1: **General Configuration Options**

Parameter	Description	Default Value	Value Range
Physical Layer			
Line Rate	Line rate of the interface in Gb/s	12.5	3.125 5 6.25 10.3125 12.5 25.78125[(1)]
Lanes	Number of Lanes	12	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
Transceiver Type	Transceiver Type	GTH	GTH GTY

Table 4-1: **General Configuration Options *(Cont'd)***

Parameter	Description	Default Value	Value Range
GT Ref Clk	Reference Clock for the GTs used, in MHz. The default value is dependent on the transceiver configuration.	195.3125 (GTH at 12.5 Gb/s) 402.832 (GTY at 25.78125 Gb/s)	125.00 128.90 156.25 161.133 189.39 195.3125 201.41 250.00 257.81 312.50 322.26 386.71 390.62 402.832
Operation	Operating mode	Duplex	Duplex Simplex TX Simplex RX
Clocking [(2)]	Clocking mode	Synchronous	Synchronous Asynchronous
GT DRP/Init Clock	This specifies the frequency (in MHz) that is used to provide a free running clock to GT and also for the DRP operations.	100	Depends on the Line Rate used
Link Layer			
RX Dataflow Mode[(3)]	Packet Mode	Packet mode	Packet Mode Burst Mode
BurstMax	Burst Max in Bytes	256	64 128 192 256
BurstShort[(4)]	Burst Short in Bytes	64	64 96 128 160 192 224 256
Meta Frame	Meta Frame length in Interlaken words	2,048	256 to 8,192

Table 4-1: **General Configuration Options *(Cont'd)***

Parameter	Description	Default Value	Value Range
Channels	Number of Channels	256	256 512 1,024 2,048
Flow Control			
Enable In-band flow control	In-band flow control	0	0 – Disable 1 – Enable
In-Band Calendar Length	Entries for calendar length	256	16 32 64 128 256
Enable out-of-band flow control	Out-of-band flow control	0	0 – Disable 1 – Enable
Out-of-band Calendar Length	Entries for calendar length	256	32 64 128 256 512 1,024 2,048
Additional Features			
Enable Protocol Bypass Mode	Enables Interlaken IP to operate in Protocol Bypass Mode	0	0 – Disable 1 – Enable
Disable Skipword	Used to provide clock compensation in the Meta Frames if intermediate electrical repeater functions are used	1	0 – False 1 – True
Include AXI4-Lite Control and Statistics Interface	When you enable, the AXI4-Lite interface is provided in the core	0	0 – Disable 1 – Enable

Notes:

1. 25.78125 Gb/s line rate not supported for devices that have only GTH transceiver support or -1 speed grade.
2. Applicable only when the operating mode is **Duplex**. For Simplex TX / Simplex RX, this field displays **Asynchronous** and is grayed-out.
3. Applicable only for the RX. The operating mode must be set to **Duplex** or **Simplex RX**. For Simplex TX, this field displays as **NA** and is grayed-out.
4. BurstShort must be less than or equal to BurstMax.

Advanced Tab

Figure 4-2 shows the advanced parameters for this IP.

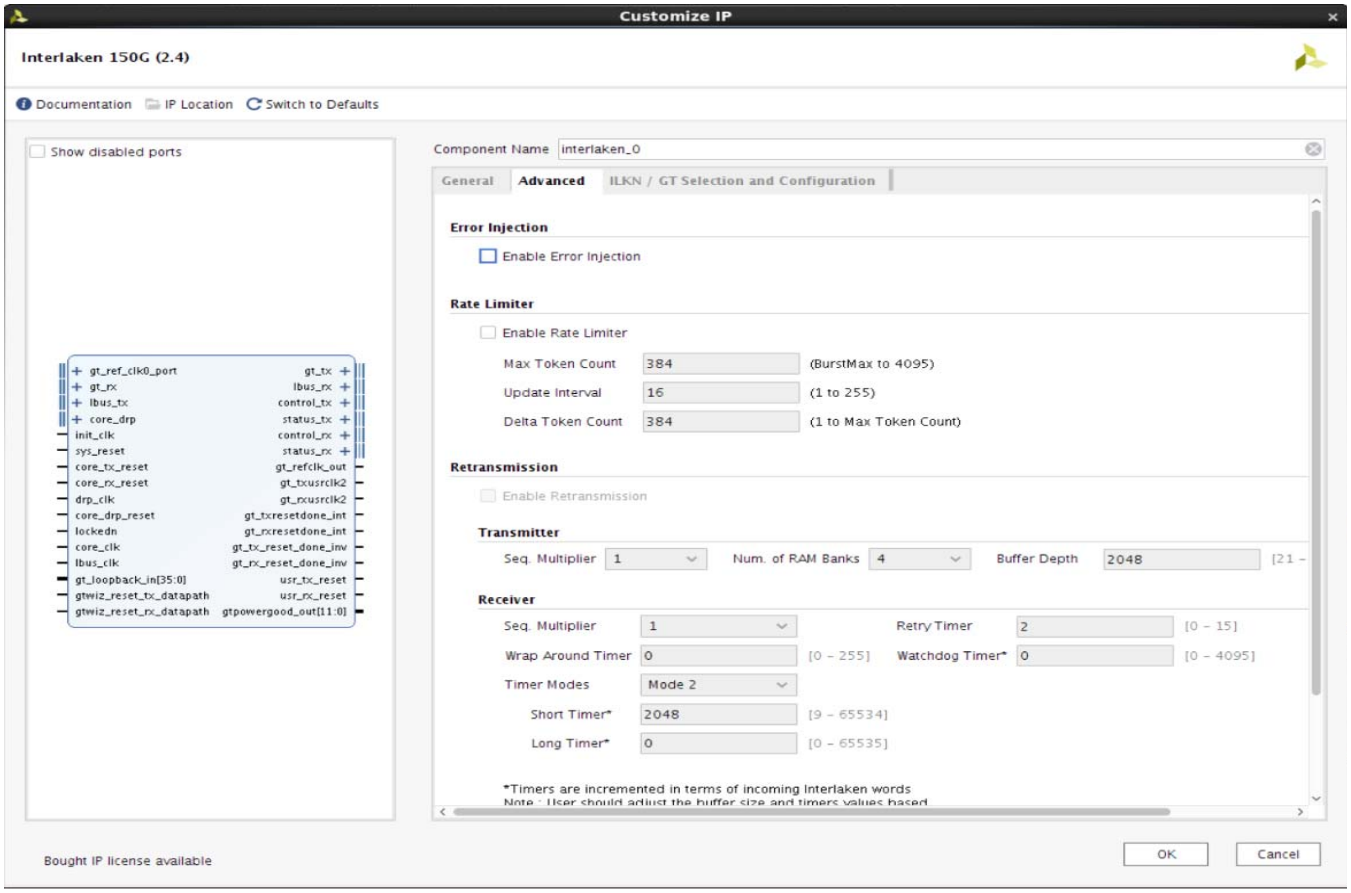


Figure 4-2: **Advanced Tab**

Table 4-2 describes the options available in the Advanced tab.

Table 4-2: **Advanced Configuration Options**

Parameter	Description	Default Value	Value Range
Error Injection			
Enable Error Injection	Enable Error Injection	0	0 – Disable 1 – Enable
Rate Limiter			
Enable Rate Limiter[(1)]	Enable Rate Limiter	0	0 – Disable 1 – Enable
Max Token Count[(2)]	This value sets the maximum number of tokens in the bucket in terms of Bytes. A value of 1, means 1 byte. This value must be equal to or greater than Burst Max. ***Note:*** Setting a value between 1 and 2 times the value of Burst Max can result in rates closest to the expected rates.	384	Burst Max to 4,095
Update Interval[(2)]	This value sets the number of LBUS clock cycles between additions of **Delta Token Count** tokens to the bucket. ***Note:*** Xilinx recommends setting a value between 8 and 32.	16	1 to 255
Delta Token Count[((2)]	This value specifies how many tokens are added to the bucket after each interval. A token here is equal to 1 byte. This value must be greater than 0 and less than or equal to **Max Token Count**.	384	1 to Max Token Count
Retransmission			
Enable Retransmission[(3)]	Enable Retransmission	0	0 – Disable 1 – Enable
Transmitter			
Seq. Multiplier[(4)]	Sequence Multiplier	1	1 2 4 8 16 32

Table 4-2: **Advanced Configuration Options *(Cont'd)***

Parameter	Description	Default Value	Value Range
Num. of RAM Banks[4]	Number of RAM Banks	4	1 2 3 4
Buffer Depth[4]	Buffer Depth	2048	8 to 512 (For Num. of RAM Banks = 1) 8 to 1,024 (For Num. of RAM Banks = 2) 8 to 1,536 (For Num. of RAM Banks = 3) 8 to 2,048 (For Num. of RAM Banks = 4)
Receiver			
Seq. Multiplier[4]	Sequence Multiplier	1	1 2 4 8 16 32
Retry Timer[4]	Retry Timer	2	2 to 15
Wrap Around Timer[4]	Wrap Around Timer	0	0 to 255
Watchdog Timer[4]	Watchdog Timer	0	0 to 4,095
Timer Modes[4]	Timer Modes	Mode 2	Mode 1 Mode 2
Short Timer[4]	Short Timer	2048	9 to 65,534 (In Mode 2) 9 to 65,534 (In Mode 1)
Long Timer[4]	Long Timer	0 (In Mode 2) 3,000 (In Mode 1)	0 (In Mode 2) Short Timer to 65,535 (In Mode 1)

Notes:

1. Enable Rate Limiter option is available only in a Duplex or Simplex TX operation.
2. Applicable only if retransmission is enabled.
3. Out-of-band flow control must be enabled to support retransmission
4. Applicable only if retransmission is enabled.

GT Selections and Configuration

Figure 4-3 shows the GT selection and configuration options for the IP.

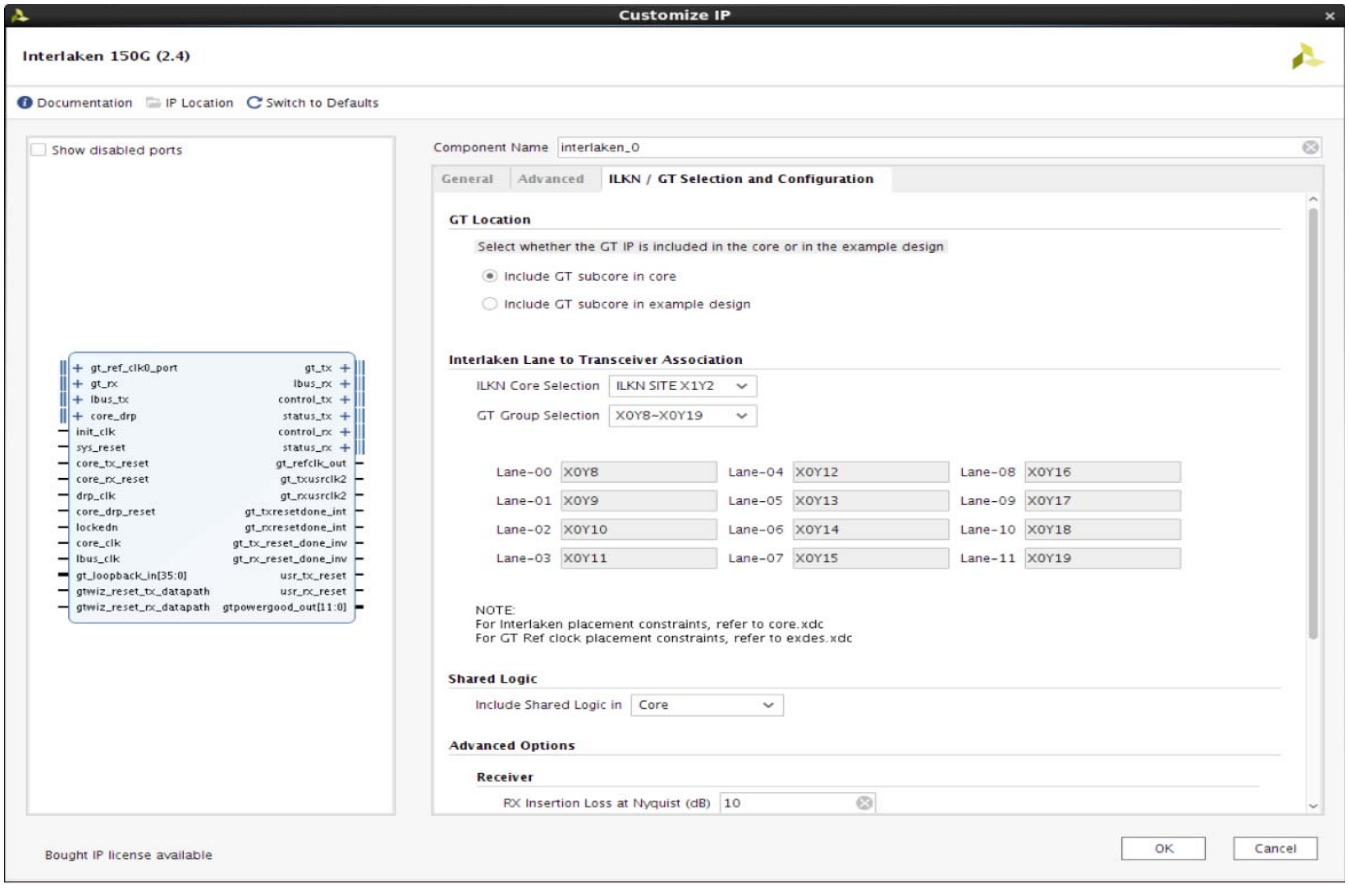


Figure 4-3: **GT Selections and Configuration Tab**

Table 4-3 describes the options available in the ILKN/GT Selection and Configuration tab.

Table 4-3: **ILKN/GT Selection and Configuration Options**

Parameter	Description	Default Value	Value Range
GT Location			
GT Location	Select whether the GT IP is included in the core or in the example design.	Include GT subcore in core	Include GT subcore in core Include GT subcore in example design
Interlaken Lane to Transceiver Association			
ILKN Core Selection	XY location of the transceiver connected to the Interlaken core	Based on the selected package/ device	Options based on the selected package/device
GT Group Selection	Based on the FPGA, part number, GT type selected, and number of lanes, the available ILKN cores for the particular device/package. The best and all possible GT groups are listed as per GT selection guidelines	Based on device, package and core location, the best possible GT combinations are listed	Possible combination based on the device, package, and core location selected
Lane 00 to Lane 11	Lanes will be auto filled, based on the selection of the ILKN core location and GT group selection.	Auto filled	
Shared Logic			
Include Shared Logic In	Determines the location of the transceiver shared logic	Core	Core Example Design
Advanced Options			
Receiver			
RX Insertion Loss at Nyquist (dB)	Specify the insertion loss of the channel between the transmitter and receiver at the Nyquist frequency in dB	10	Depends on GT
RX Equalization Mode	When **Auto** is specified, the equalization mode implemented by the Wizard depends on the value specified for insertion loss at Nyquist. Refer to Xilinx UG576 [Ref 15]/UG578 [Ref 16] to determine the appropriate equalization mode for your system.	Auto	Auto DFE LPM
GT QPLL			
PLL Type	GT PLL Type	QPLL0	QPLL0 QPLL1

Table 4-3: **ILKN/GT Selection and Configuration Options *(Cont'd)***

Parameter	Description	Default Value	Value Range
Others			
Enable Pipeline Registers	Selecting this option will include one stage pipeline register between ILKN core and the GT to ease timing	0	0-Disable 1-Enable
Enable Additional GT Control/ Status and DRP Ports	If selected, enables additional GT control, status and DRP ports	0	0-Disable 1-Enable

User Parameters

This section is not applicable for this core.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

The integrated IP core for Interlaken solution requires the specification of timing and other physical implementation constraints to meet the specified performance requirements. These constraints are provided in a Xilinx® Device Constraints (XDC) file. Pinouts and hierarchy names in the generated XDC correspond to the provided example design of the integrated IP core for Interlaken.

To achieve consistent implementation results, an XDC containing these original, unmodified constraints must be used when a design is run through the Xilinx design tools. For additional details on the definition and use of an XDC specific constraints, see the *Vivado Design Suite User Guide: Using Constraints* (UG903) [Ref 12].

Constraints provided in the integrated IP core for Interlaken have been verified through implementation and provide consistent results. Constraints can be modified, but modifications should only be made with a thorough understanding of the effect of each constraint.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 11].

For information regarding simulating the example design, see Simulating the Example Design in Chapter 5.

Simulation Speed Up

A `define SIM_SPEED_UP is available for faster simulation with reduced Meta Frame length.

VCS

Use the vlogn option +define+SIM_SPEED_UP

ModelSim/Mentor Graphics Questa Advanced Simulator

Use the vlog option +define+SIM_SPEED_UP

IES

Use the ncvlog option +define+SIM_SPEED_UP

Vivado Simulator

Use the xvlog option -d SIM_SPEED_UP

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 9].

For information regarding synthesizing and implementing the example design, see Synthesizing and Implementing the Example Design in Chapter 5.

Example Design

This chapter contains information about the example design provided in the Vivado® Design Suite.

Overview

This chapter describes the integrated IP core for Interlaken example design and the various test scenarios implemented within the example design.

Example Design Hierarchy

Figure 5-1 shows the instantiation of various modules and their hierarchy in the example design.

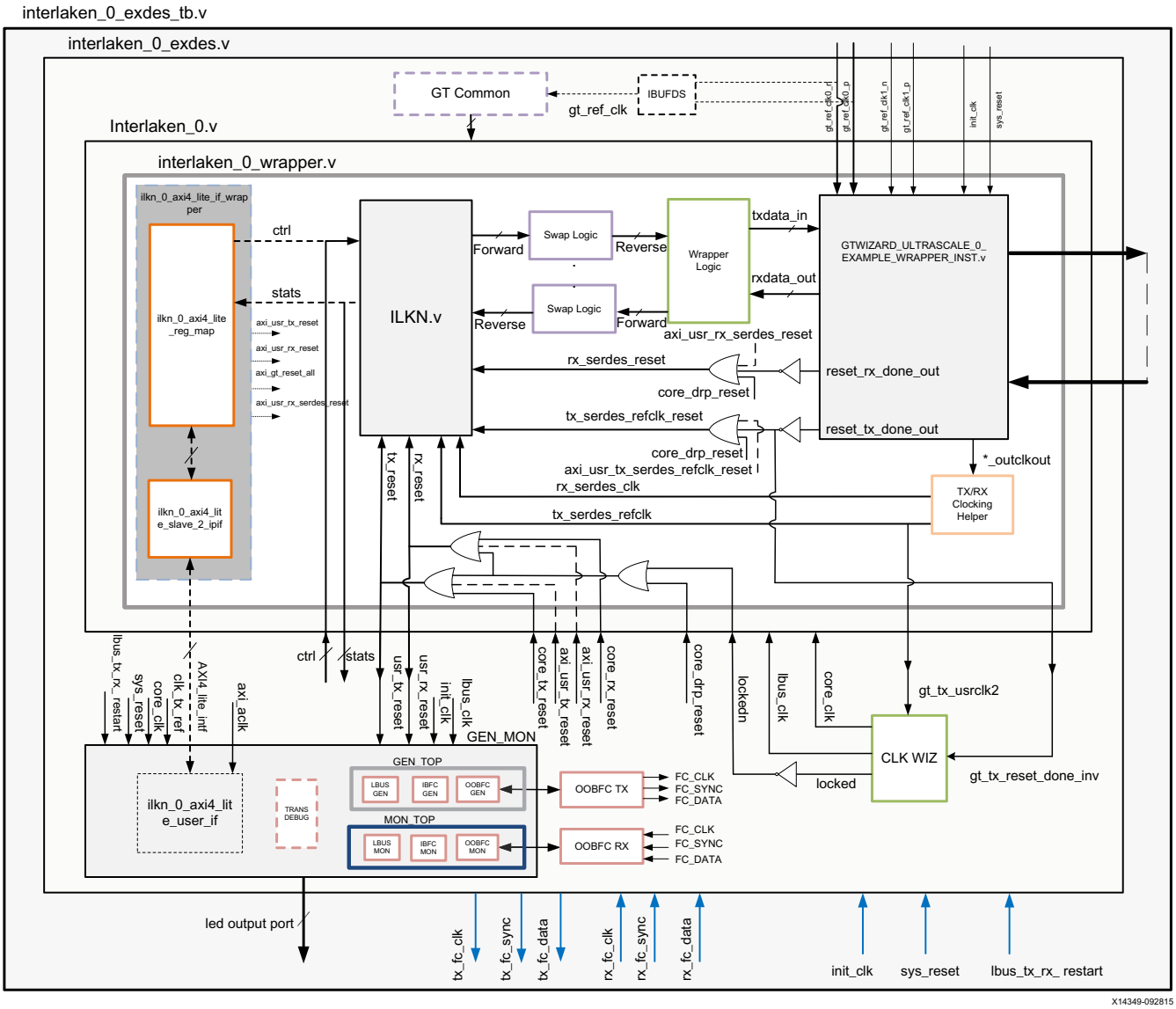


Figure 5-1: **Example Design Hierarchy**

Details of the example design hierarchy are as follows:

- The interlaken_0 module instantiates Interlaken and the serial transceiver (GT) along with various helper blocks.
- The interlaken_0_pkt_gen_mon module instantiates the interlaken_0_gen_wrapper and interlaken_0_mon_wrapper modules.
- The interlaken_0_gen_wrapper module instantiates interlaken_0_pkt_gen (packet generator), in-band flow control generator (if IBFC is enabled in the Vivado Integrated Design Environment (IDE)) and out-of-band flow control generator modules (if out-of-band flow control (OOBFC) is enabled in the Vivado IDE) based on ILKN Vivado IDE configuration.

- The interlaken_0_mon_wrapper module instantiates interlaken_0_pkt_mon (packet monitor), in-band flow control monitor (if IBFC is enabled in the Vivado IDE) and out of band flow control monitor (if OOBFC enabled in the Vivado IDE) modules based on ILKN Vivado IDE configuration.
- The interlaken_0_pkt_gen_mon and interlaken_0 modules handshake with each other using few signals: GT locked, RX alignment and data transfer signals as per LBUS protocol. You can change the number of packets in the **`interlaken_0_exedes.v`** file using a parameter. The same number is used by the generator and monitor to send and receive packets. The number of packets should be even and in the range of 2 to 65534.
- The interlaken_0_pkt_gen module is mainly responsible for the generation of LBUS packets and generating control signals to flow control (IBFC and OOBFC) generator modules and error injection test. It contains a state machine which monitors the GT and Interlaken status (that is, GT lock and RX alignment) and sends traffic to Interlaken.
- Similarly, the interlaken_0_pkt_mon module is mainly responsible for reception of packets from Interlaken and monitoring status signals from flow control (IBFC and OOBFC) monitor modules and error injection status signals from Interlaken core. It also contain a state machine which monitors the GT and Interlaken status (that is, GT lock and RX alignment) and receives traffic from Interlaken.
- The example design supports both Packet mode and Burst mode. The LBUS packet length is predefined to make sure that all segments SOP and EOP are toggled.
  - For the Packet mode, channel number starts from 0 and increments for each packet up to 255 and resets back to 0.
  - For Burst mode, channel number toggles between two predefined channel numbers.
- The Swap Logic does bit swapping on the data interface between the Interlaken core and the GT.
- The transceiver_debug module is available in the example design when you enable the **Additional transceiver control and status ports** from the **GT Selections and Configuration** tab of the Interlaken Vivado IDE. This module brings out all the DRP ports of the transceiver module out of the Interlaken core.
- The GT_common module is available in the example design when you select the **Include Shared Logic in example design** from the **GT Selections and Configuration** tab of the Interlaken Vivado IDE. This module brings out the transceiver common module out of the Interlaken core.
- The clocking wizard generates different clocks based on the core configuration in the Interlaken Vivado IDE. If you set **Line Rate** to 12.5 Gb/s, the clocking wizard generates one clock with 300 MHz and the same clock is used as **`core_clk`** and **`lbus_clk`**. If you set **Line Rate** to 25.78125 Gb/s, the clocking wizard generates two clocks, **`lbus_clk`** with a frequency of 300 MHz and **`core_clk`** with a frequency of 412 MHz.

- The AXI4-Lite interface wrapper is available when you select the **Include AXI4-Lite Control and Statistics Interface** option from the **General** tab. This module is included inside the ilkn_0_wrapper. This wrapper contains two modules: ilkn_0_axi4_lite_reg_map, and ilkn_0_axi4_lite_slave_2_ipif. These modules are described in detail in AXI4-Lite Interface Implementation.
- The AXI4-Lite user interface is available when you select the I**nclude AXI4-Lite Control and Statistics Interface** option from the **General** tab. This module is present in the ilkn_pkt_gen_mon. This module is described in detail in AXI4-Lite Interface Implementation.

IMPORTANT: *When Interlaken is operating at 25.78125 Gb/s with either 5 or 6 lanes, there are:*
- two IBUFDS instantiated to drive two different reference clocks to the two GT quads, and
- two pairs of `gt_ref_clk` *are needed to drive the two IBUFDS, as shown in Figure 5-2.*
From the UltraScale™ GTY Transceivers User Guide (UG578) [Ref 7], "For UltraScale FPGAs, channels operating above 16.375 Gb/s cannot source a reference clock from another Quad and must utilize one of the two local reference clock pin pairs in its own Quad." "For UltraScale+ FPGAs, channels operating from 16.375 Gb/s up to 28.21 Gb/s can source a reference clock from up to one Quad above and below."

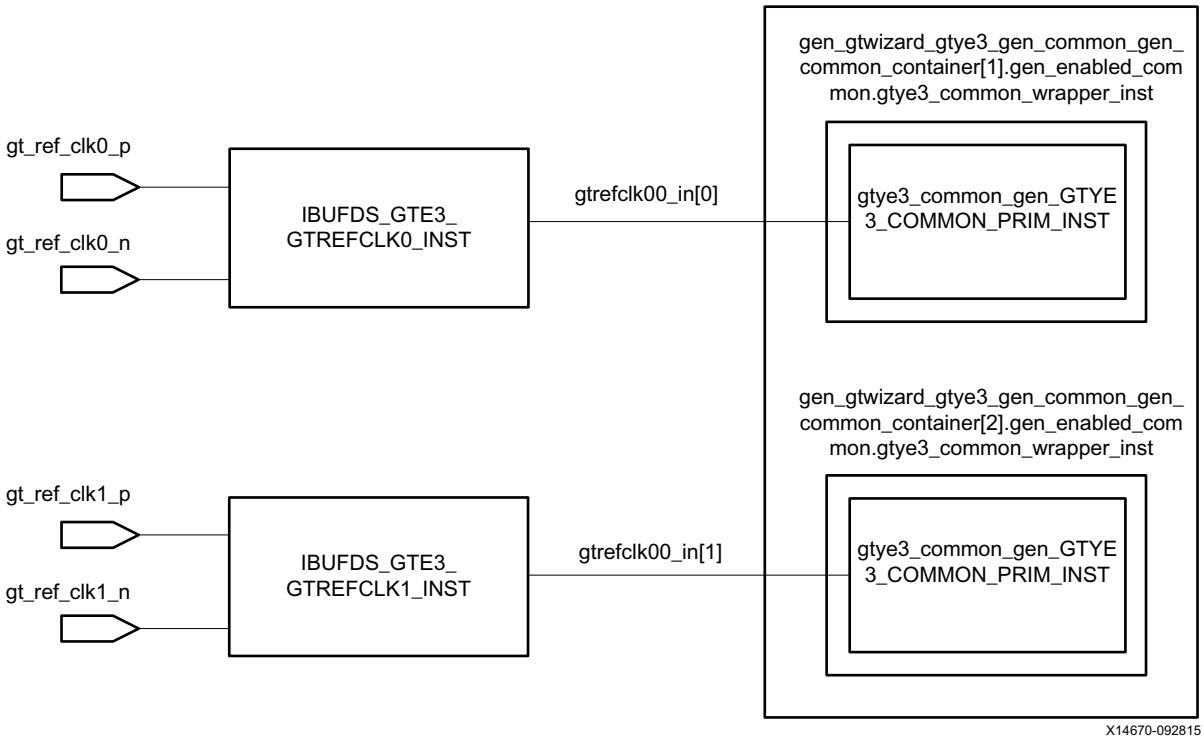
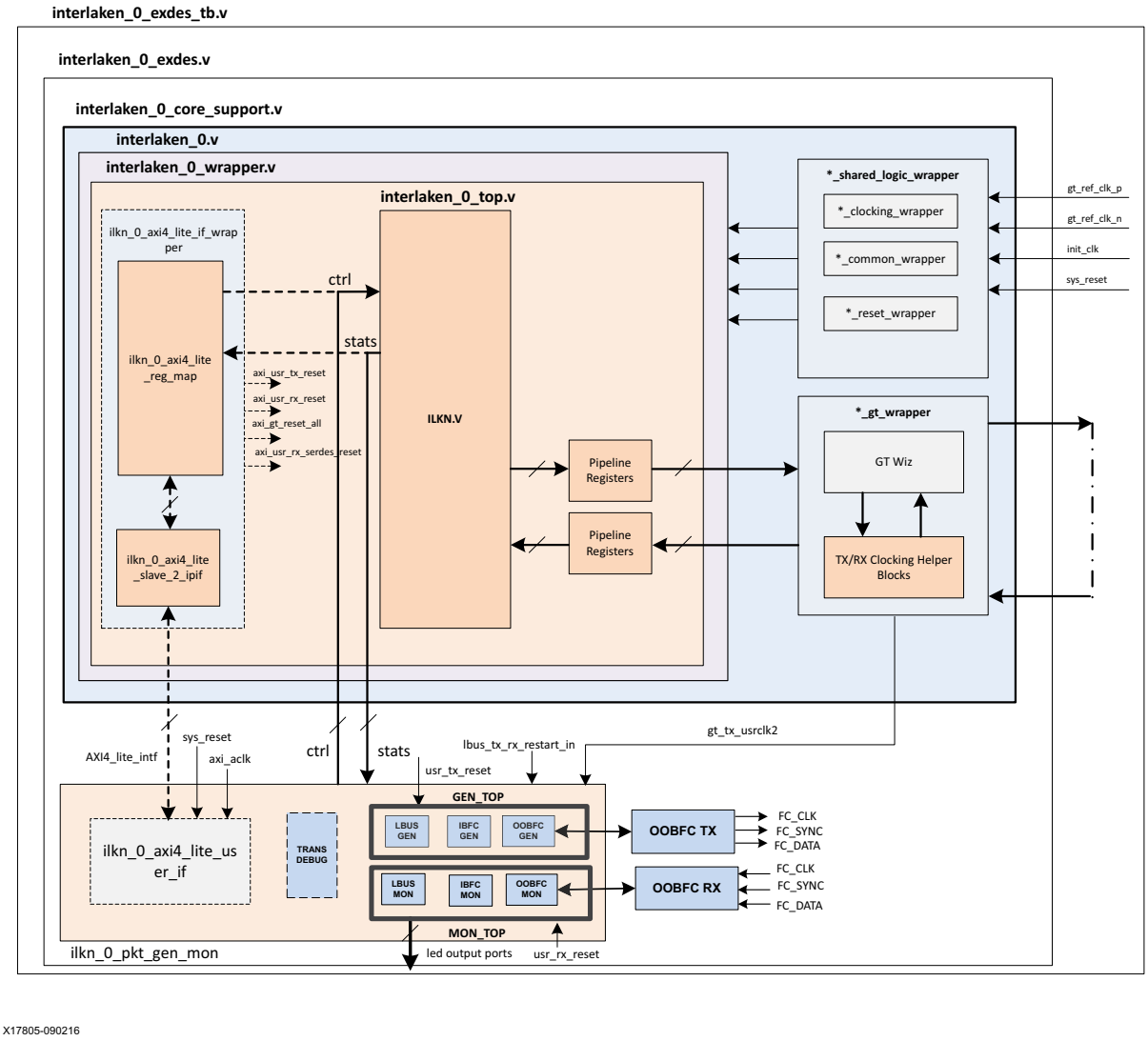


Figure 5-2: **25 Gb/s Reference Clock**

Table 5-1: **Interlaken Supported Modes and Configurations**

Number of Lanes	GT Type	Speed Grade
(1 to 6 Lanes) x25.78G	GTY Only	(-2 or higher silicon)
(1 to 12 Lanes) x12.5G	GTH and GTY	(-1 or higher silicon)
(1 to 12 Lanes) x10.3125G	GTH and GTY	(-1 or higher silicon)
(1 to 12 Lanes) x6.25G	GTH and GTY	(-1 or higher silicon)
(1 to 12 Lanes) x5G	GTH and GTY	(-1 or higher silicon)
(1 to 12 Lanes) x3.125G	GTH and GTY	(-1 or higher silicon)

Example Design Hierarchy (GT Subcore in Example Design)



X17805-090216

Figure 5-3: **Example Design Hierarchy (GT Subcore in Example Design)**

Figure 5-3 shows the instantiation of various modules and their hierarchy in the example design when the GT (serial transceiver) is outside the IP core, that is, in the example design. This hierarchical example design is delivered when you select the **Include GT subcore in example design** option from the ILKN/GT Selection and Configuration tab.

The `Interlaken_0_core_support.v` is present in the hierarchy when you select the **Include GT subcore in example design** option from the ILKN/GT Selection and Configuration tab or the **Include Shared Logic in example design** option from the ILKN/GT Selection and Configuration tab.

This instantiates the **`Interlaken_0_shared_logic_wrapper.v`** module and the **`Interlaken_0.v`** module for the **Include Shared Logic in example design** option. The **`Interlaken_0_gt_wrapper.v`** module will be present when you select the **GT subcore in example design** option.

The Interlaken_0 module instantiates the Interlaken_0_wrapper module that has the Interlaken and Sync registers along with the pipeline registers to synchronize the data between the Interlaken core and the GT subcore (in the example design). The GT sub-core generates the required clock frequencies with help of the clocking helper blocks for the Interlaken core. The **`Interlaken_0_pkt_gen_mon`** module instantiates the Interlaken_0_pkt_gen (packet generator) and Interlaken_0_pkt_mon (packet monitor). The Interlaken_0_pkt_gen_mon and Interlaken_0 handshake with each other using a few signals such as GT locked, RX alignment and data transfer signals as per the LBUS protocol (more on this will be described in later sections). The Interlaken_0_pkt_gen module is mainly responsible for the generation of packets. It contains a state machine that monitors the status of GT and Interlaken (that is, GT lock and RX alignment) and sends traffic to the core. Similarly the Interlaken_0_pkt_mon module is mainly responsible for reception and checking of packets from the core. It also contains a state machine that monitors the status of GT and INTERLAKEN (that is, GT lock and RX alignment) and receives traffic from the core.

Other optional modules instantiated in the example design are as follows:

- **Interlaken_0_shared_logic_wrapper**: When you select **Include GT subcore in example design** or **Include Shared Logic in example design** in the ILKN/GT Selection and Configuration tab of the Interlaken IP Vivado IDE, this module will be available in the example design. This wrapper contains three modules: Interlaken_0_clocking_wrapper, Interlaken_0_reset_wrapper, and Interlaken_0_common_wrapper. The Interlaken_0_clocking_wrapper has the instantiation of the IBUFDS for the **`gt_ref_clk`**, and the Interlaken_0_reset_wrapper brings out the reset architecture instantiated between the core and the GT. The Interlaken_0_common_wrapper brings the transceiver common module out of the Interlaken IP core.
- Interlaken_0_gt_wrapper: This module is present in the example design when you select the **Include GT subcore in example design** option from the ILKN/GT Selection and Configuration tab. This module instantiates the GT along with various helper blocks. The clocking helper blocks are used to generate the required clock frequency for the core.

User Interface

GPIO has been provided so that you can control the example design. The I/O ports follow:

Table 5-2: **User I/O Ports**

Name	Size	Direction	Description
sys_reset	1	Input	Reset for interlaken_0.
gt_ref_clk0_p	1	Input	Differential input clk to GT.
gt_ref_clk0_n	1	Input	Differential input clk to GT.
init_clk	1	Input	Stable input clk to GT.
s_axi_pm_tick	1	Input	PM tick input for AXI4-Lite read operations ***Note:*** This input is available when **Include AXI4-Lite Control and Statistics Interface** is selected in the General tab.
gt_ref_clk1_p	1	Input	Differential input clk to GT. Applicable only when 25G lane rate for 5 lanes or 6 lanes.
gt_ref_clk1_n	1	Input	Differential input clk to GT. Applicable only when 25G lane rate for 5 lanes or 6 lanes.
lbus_tx_rx_restart_in	1	Input	This signal is used to restart the packet generation and reception for the data sanity test, when the packet generator and the packet monitor are in idle state, that is, when **`tx_busy_led`** = 0 and **`rx_busy_led`** = 0.
simplex_mode_rx_aligned	1	Input	This signal is used to indicate the generator module that the Simplex RX module is aligned and generator can now start the packet generation. ***Note:*** This pin is available only for simplex TX mode.
tx_done_led	1	Output	Indicates that packet generator has sent all the packets.
tx_fail_led	1	Output	Indicates TX FIFO overflow / underflow error has occurred.
tx_busy_led	1	Output	Indicate that the generator busy, and not able to respond to the **`lbus_tx_rx_restart_in`** command.
rx_gt_locked_led	1	Output	Indicates that the GT has been locked.
rx_aligned_led	1	Output	Indicates RX alignment has been achieved.

Note: For all the input and output signals mentioned in Table 5-2, a three-stage registering is done internally.

Core XCI Top-Level Port List

The top level port list for the core XCI is listed in Table 5-3.

Table 5-3: **Core XCI Top-level Input and Output Ports**

Name	Size	Direction	Description
gt_ref_clk0_p	1	Input	Differential input clk to the GT.
gt_ref_clk0_n	1	Input	Differential input clk to the GT.
init_clk	1	Input	Stable input clk to the GT.
sys_reset	1	Input	Reset for interlaken_0.
gt_txusrclk2	1	Output	TX user clock output from the GT.
gt_rxusrclk2	1	Output	RX user clock output from the GT.
gt_tx_reset_done_inv	1	Output	TX user reset output from the GT.
gt_rx_reset_done_inv	1	Output	RX user reset output from the GT.
gt0_rxp_in	1	Input	Differential serial GT RX input for lane 0.
gt0_rxn_in	1	Input	Differential serial GT RX input for lane 0.
gt1_rxp_in	1	Input	Differential serial GT RX input for lane 1. ***Note:*** This port is available when number of lanes is more than 1.
gt1_rxn_in	1	Input	Differential serial GT RX input for lane 1. ***Note:*** This port is available when number of lanes is more than 1.
gt2_rxp_in	1	Input	Differential serial GT RX input for lane 2. ***Note:*** This port is available when number of lanes is more than 2.
gt2_rxn_in	1	Input	Differential serial GT RX input for lane 2. ***Note:*** This port is available when number of lanes is more than 2.
gt3_rxp_in	1	Input	Differential serial GT RX input for lane 3. ***Note:*** This port is available when number of lanes is more than 3.
gt3_rxn_in	1	Input	Differential serial GT RX input for lane 3. ***Note:*** This port is available when number of lanes is more than 3.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt4_rxp_in	1	Input	Differential serial GT RX input for lane 4. ***Note:*** This port is available when number of lanes is more than 4.
gt4_rxn_in	1	Input	Differential serial GT RX input for lane 4. ***Note:*** This port is available when number of lanes is more than 4.
gt5_rxp_in	1	Input	Differential serial GT RX input for lane 5. ***Note:*** This port is available when number of lanes is more than 5.
gt5_rxn_in	1	Input	Differential serial GT RX input for lane 5. ***Note:*** This port is available when number of lanes is more than 5.
gt6_rxp_in	1	Input	Differential serial GT RX input for lane 6. ***Note:*** This port is available when number of lanes is more than 6.
gt6_rxn_in	1	Input	Differential serial GT RX input for lane 6. ***Note:*** This port is available when number of lanes is more than 6.
gt7_rxp_in	1	Input	Differential serial GT RX input for lane 7. ***Note:*** This port is available when number of lanes is more than 7.
gt7_rxn_in	1	Input	Differential serial GT RX input for lane 7. ***Note:*** This port is available when number of lanes is more than 7.
gt8_rxp_in	1	Input	Differential serial GT RX input for lane 8. ***Note:*** This port is available when number of lanes is more than 8.
gt8_rxn_in	1	Input	Differential serial GT RX input for lane 8. ***Note:*** This port is available when number of lanes is more than 8.
gt9_rxp_in	1	Input	Differential serial GT RX input for lane 9. ***Note:*** This port is available when number of lanes is more than 9.
gt9_rxn_in	1	Input	Differential serial GT RX input for lane 9. ***Note:*** This port is available when number of lanes is more than 9.
gt10_rxp_in	1	Input	Differential serial GT RX input for lane 10. ***Note:*** This port is available when number of lanes is more than 10.
gt10_rxn_in	1	Input	Differential serial GT RX input for lane 10. ***Note:*** This port is available when number of lanes is more than 10.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt11_rxp_in	1	Input	Differential serial GT RX input for lane 11. ***Note:*** This port is available when number of lanes is more than 11.
gt11_rxn_in	1	Input	Differential serial GT RX input for lane 11. ***Note:*** This port is available when number of lanes is more than 11.
gt0_txn_out	1	Output	Differential serial GT TX output for lane 0.
gt0_txp_out	1	Output	Differential serial GT TX output for lane 0.
gt1_txn_out	1	Output	Differential serial GT TX output for lane 1. ***Note:*** This port is available when number of lanes is more than 1.
gt1_txp_out	1	Output	Differential serial GT TX output for lane 1. ***Note:*** This port is available when number of lanes is more than 1.
gt2_txn_out	1	Output	Differential serial GT TX output for lane 2. ***Note:*** This port is available when number of lanes is more than 2.
gt2_txp_out	1	Output	Differential serial GT TX output for lane 2. ***Note:*** This port is available when number of lanes is more than 2.
gt3_txn_out	1	Output	Differential serial GT TX output for lane 3. ***Note:*** This port is available when number of lanes is more than 3.
gt3_txp_out	1	Output	Differential serial GT TX output for lane 3. ***Note:*** This port is available when number of lanes is more than 3.
gt4_txn_out	1	Output	Differential serial GT TX output for lane 4. ***Note:*** This port is available when number of lanes is more than 4.
gt4_txp_out	1	Output	Differential serial GT TX output for lane 4. ***Note:*** This port is available when number of lanes is more than 4.
gt5_txn_out	1	Output	Differential serial GT TX output for lane 5. ***Note:*** This port is available when number of lanes is more than 5.
gt5_txp_out	1	Output	Differential serial GT TX output for lane 5. ***Note:*** This port is available when number of lanes is more than 5.
gt6_txn_out	1	Output	Differential serial GT TX output for lane 6. ***Note:*** This port is available when number of lanes is more than 6.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt6_txp_out	1	Output	Differential serial GT TX output for lane 6. ***Note:*** This port is available when number of lanes is more than 6.
gt7_txn_out	1	Output	Differential serial GT TX output for lane 7. ***Note:*** This port is available when number of lanes is more than 7.
gt7_txp_out	1	Output	Differential serial GT TX output for lane 7. ***Note:*** This port is available when number of lanes is more than 7.
gt8_txn_out	1	Output	Differential serial GT TX output for lane 8. ***Note:*** This port is available when number of lanes is more than 8.
gt8_txp_out	1	Output	Differential serial GT TX output for lane 8. ***Note:*** This port is available when number of lanes is more than 8.
gt9_txn_out	1	Output	Differential serial GT TX output for lane 9. ***Note:*** This port is available when number of lanes is more than 9.
gt9_txp_out	1	Output	Differential serial GT TX output for lane 9. ***Note:*** This port is available when number of lanes is more than 9.
gt10_txn_out	1	Output	Differential serial GT TX output for lane 10. ***Note:*** This port is available when number of lanes is more than 10.
gt10_txp_out	1	Output	Differential serial GT TX output for lane 10. ***Note:*** This port is available when number of lanes is more than 10.
gt11_txn_out	1	Output	Differential serial GT TX output for lane 11. ***Note:*** This port is available when number of lanes is more than 11.
gt11_txp_out	1	Output	Differential serial GT TX output for lane 11. ***Note:*** This port is available when number of lanes is more than 11.
tx_reset_done	1	Input	TX reset done input to the core from the reset wrapper logic. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** in the ILKN/GT Selection and Configuration tab.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
rx_reset_done	1	Input	RX reset done input to the core from the reset wrapper logic. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** in the ILKN/GT Selection and Configuration tab.
axi_usr_tx_reset	1	Output	User TX reset from the AXI4-Lite register map module. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** in the ILKN/GT Selection and Configuration tab and **Include AXI4-Lite Control and Statistics Interface** is selected in the General tab.
axi_usr_rx_reset	1	Output	User RX reset from the AXI4-Lite register map module. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** in the ILKN/GT Selection and Configuration tab and **Include AXI4-Lite Control and Statistics Interface** is selected in the General tab.
axi_usr_rx_serdes_reset	12	Output	User RX serdes reset from the AXI4-Lite register map module. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** in the ILKN/GT Selection and Configuration tab and **Include AXI4-Lite Control and Statistics Interface** is selected in the General tab.
axi_usr_tx_serdes_refclk_reset	1	Output	RX serdes clock out from the core to the reset wrapper. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** in the ILKN/GT Selection and Configuration tab.
qpll0clk_in	12	Input	QPLL0 clock input. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** in the ILKN/GT Selection and Configuration tab.
qpll0refclk_in	12	Input	QPLL0 ref clock input. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** in the ILKN/GT Selection and Configuration tab.
qpll1clk_in	12	Input	QPLL1 clock input. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** in the ILKN/GT Selection and Configuration tab.
qpll1refclk_in	12	Input	QPLL1 ref clock input. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** in the ILKN/GT Selection and Configuration tab.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gtwiz_reset_qpll0lock_in	3	Input	QPLL0 lock reset input to the GT. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** and **PLL Type** is selected as **QPLL0** in the ILKN/GT Selection and Configuration tab.
gtwiz_reset_qpll0reset_out	3	Output	QPLL0 lock reset output from the GT. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** and **PLL Type** is selected as **QPLL0** in the ILKN/GT Selection and Configuration tab.
gtwiz_reset_qpll1lock_in	3	Input	QPLL1 lock reset input to the GT. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** and **PLL Type** is selected as **QPLL0** in the ILKN/GT Selection and Configuration tab.
gtwiz_reset_qpll1reset_out	3	Output	QPLL1 lock reset output from the GT. ***Note:*** This port is available when the **Include Shared Logic in** option is selected as **Example Design** and **PLL Type** is selected as **QPLL0** in the ILKN/GT Selection and Configuration tab.
rx_ovfout	1	Output	Receive LBUS overflow. If this signal is asserted, it means that the LBUS clock is too slow for the incoming data stream. The LBUS bandwidth must be greater than the Interlaken bandwidth.
rx_dataout0	128	Output	Receive segmented LBUS Data for segment0. The value of the bus is only valid in cycles in which RX_ENAOUT is sampled as 1.
rx_chanout0	11	Output	Receive channel number for segment0. The bus indicates the channel number of the in-flight packet and is only valid in cycles in which RX_ENAOUT is sampled as 1. The maximum number of channels is programmed by the CTL_RX_CHAN_EXT pin. See that pin description for the encoding of that signal.
rx_enaout0	1	Output	Receive LBUS enable for segment0. This signal qualifies the other signals of the RX segmented LBUS Interface. Signals of the RX LBUS Interface are only valid in cycles in which RX_ENAOUT is sampled as a 1.
rx_sopout0	1	Output	Receive LBUS Start of Packet for segment0. This signal indicates the SOP when it is sampled as a 1 and is only valid in cycles in which RX_ENAOUT is sampled as a 1.
rx_eopout0	1	Output	Receive LBUS EOP for segment0. This signal indicates the EOP when it is sampled as a 1 and is only valid in cycles in which RX_ENAOUT is sampled as a 1.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
rx_errout0	1	Output	Receive LBUS error for segment0. This signal indicates that the current packet being received has an error when it is sampled as a 1. This signal is only valid in cycles when both rx_enaout and rx_eopout are sampled as a 1. When this signal is a value of 0, it indicates that there is no error in the packet being received.
rx_mtyout0	4	Output	Receive LBUS Empty for segment0. This bus indicates how many bytes of the RX_DATAOUT bus are empty or invalid for the last transfer of the current packet. This bus is only valid in cycles when both RX_ENAOUT and RX_EOPOUT are sampled as 1. When RX_ERROUT and RX_ENAOUT are sampled as 1, the value of RX_MTYOUT[3:0] is always 000. Other bits of RX_MTYOUT are as usual.
rx_dataout1	128	Output	Receive segmented LBUS Data for segment1.
rx_chanout1	11	Output	Receive channel number for segment1.
rx_enaout1	1	Output	Receive LBUS enable for segment1.
rx_sopout1	1	Output	Receive LBUS Start of Packet for segment1.
rx_eopout1	1	Output	Receive LBUS EOP for segment1.
rx_errout1	1	Output	Receive LBUS Error for segment1.
rx_mtyout1	4	Output	Receive LBUS Empty for segment1.
rx_dataout2	128	Output	Receive segmented LBUS Data for segment2.
rx_chanout2	11	Output	Receive channel number for segment2.
rx_enaout2	1	Output	Receive LBUS enable for segment2.
rx_sopout2	1	Output	Receive LBUS Start of Packet for segment2.
rx_eopout2	1	Output	Receive LBUS EOP for segment2.
rx_errout2	1	Output	Receive LBUS Error for segment2.
rx_mtyout2	4	Output	Receive LBUS Empty for segment2.
rx_dataout3	128	Output	Receive segmented LBUS Data for segment3.
rx_chanout3	11	Output	Receive channel number for segment3.
rx_enaout3	1	Output	Receive LBUS enable for segment3.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
rx_sopout3	1	Output	Receive LBUS Start of Packet for segment3.
rx_eopout3	1	Output	Receive LBUS EOP for segment3.
rx_errout3	1	Output	Receive LBUS Error for segment3.
rx_mtyout3	4	Output	Receive LBUS Empty for segment3.
tx_rdyout	1	Output	Transmit LBUS Ready. This signal indicates whether the Interlaken core TX path is ready to accept data and provides back-pressure to the user logic. A value of 1 means the user logic can pass data to the core. A value of 0 means the user logic must stop transferring data to the core. When TX_RDYOUT is asserted depends on a pre-determined value of FIFO fill.
tx_ovfout	1	Output	Transmit LBUS Overflow. This signal indicates whether you have violated the back pressure mechanism provided by the TX_RDYOUT signal. If TX_OVFOUT is sampled as a 1, a violation has occurred. You must design the rest of the user logic to prevent the overflow of the TX interface.
tx_datain0	128	Input	Transmit segmented LBUS Data for segment0. This bus receives input data from the user logic. The value of the bus is captured in every cycle that TX_ENAIN is sampled as 1.
tx_chanin0	11	Input	Transmit LBUS channel number for segment0. This bus receives the channel number for the packet being written. The value of the bus is captured in every cycle that TX_ENAIN is sampled as 1.The maximum number of channels is programmed by the CTL_TX_CHAN_EXT pin. See that pin description for the encoding of that signal.
tx_enain0	1	Input	Transmit LBUS enable for segment0. This signal is used to enable the TX LBUS Interface. All signals on this interface are sampled only in cycles in which TX_ENAIN is sampled as a 1.
tx_sopin0	1	Input	Transmit LBUS Start Of Packet for segment0. This signal is used to indicate the SOP when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles in which TX_ENAIN is sampled as a 1.
tx_eopin0	1	Input	Transmit LBUS EOP for segment0. This signal is used to indicate the EOP when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles in which TX_ENAIN is sampled as a 1.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
tx_errin0	1	Input	Transmit LBUS Error for segment0. This signal is used to indicate a packet contains an error when it is sampled as a 1 and is 0 for all other transfers of the packet. This signal is sampled only in cycles in which TX_ENAIN and TX_EOPIN are sampled as 1.
tx_mtyin0	4	Input	Transmit LBUS Empty for segment0. This bus is used to indicate how many bytes of the TX_DATAIN bus are empty or invalid for the last transfer of the current packet. This bus is sampled only in cycles that TX_ENAIN and TX_EOPIN are sampled as 1. When TX_EOPIN and TX_ERRIN are sampled as 1, the value of TX_MTYIN[2:0] is ignored as treated as if it was 000. The other bits of TX_MTYIN are used as usual.
tx_bctlin0	1	Input	Transmit force insertion of Burst Control word for segment0. This input is used to force the insertion of a Burst Control Word. When TX_BCTLIN and TX_ENAIN, are sampled as 1, a Burst Control word is inserted before the data on the TX_DATAIN bus is transmitted even if one is not required to observe the BurstMax parameter. This input is used by external schedulers that wish to reduce bandwidth lost due to observation of the BurstShort parameter. The use of an enhanced scheduling algorithm as described in the Interlaken Protocol Definition 1.2 is required.
tx_datain1	128	Input	Transmit segmented LBUS Data for segment1.
tx_chanin1	11	Input	Transmit LBUS channel number for segment1.
tx_enain1	1	Input	Transmit LBUS enable for segment1.
tx_sopin1	1	Input	Transmit LBUS Start Of Packet for segment1.
tx_eopin1	1	Input	Transmit LBUS EOP for segment1.
tx_errin1	1	Input	Transmit LBUS Error for segment1.
tx_mtyin1	4	Input	Transmit LBUS Empty for segment1.
tx_bctlin1	1	Input	Transmit force insertion of Burst Control word for segment1.
tx_datain2	128	Input	Transmit segmented LBUS Data for segment2
tx_chanin2	11	Input	Transmit LBUS channel number for segment2.
tx_enain2	1	Input	Transmit LBUS enable for segment2.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
tx_sopin2	1	Input	Transmit LBUS enable for segment2.
tx_eopin2	1	Input	Transmit LBUS EOP for segment2.
tx_errin2	1	Input	Transmit LBUS Error for segment2.
tx_mtyin2	4	Input	Transmit LBUS Empty for segment2.
tx_bctlin2	1	Input	Transmit force insertion of Burst Control word for Segment2.
tx_datain3	128	Input	Transmit segmented LBUS Data for segment3.
tx_chanin3	11	Input	Transmit LBUS channel number for segment3.
tx_enain3	1	Input	Transmit LBUS enable for segment3.
tx_sopin3	1	Input	Transmit LBUS Start Of Packet for segment3.
tx_eopin3	1	Input	Transmit LBUS EOP for segment3.
tx_errin3	1	Input	Transmit LBUS Error for segment3.
core_tx_reset	1	Input	Asynchronous reset for the TX circuits. This signal is active-High (1= reset) and must be held High until all of the clocks for the TX path are fully active. These clocks are CORE_CLK, LBUS_CLK, and TX_SERDES_REFCLK. The Interlaken core handles synchronizing the TX_RESET input to the appropriate clock domains within the core.
core_rx_reset	1	Input	Asynchronous reset for the RX circuits. This signal is active-High (1= reset) and must be held High until all of the clocks for the RX path are fully active. These clocks are CORE_CLK, LBUS_CLK and RX_SERDES_CLK[11:0]. The Interlaken core handles synchronizing the RX_RESET input to the appropriate clock domains within the core.
tx_mtyin3	4	Input	Transmit LBUS Empty for segment3.
tx_bctlin3	1	Input	Transmit force insertion of Burst Control word for Segment3.
drp_clk	1	Input	DRP interface clock. When DRP is not used, this can be tied to GND.
core_drp_reset	1	Input	Core DRP reset.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt_drp_done	1	Input	GT DRP done signal is asserted High for at least two clock cycles (drp_clk). This is to reset the GT after any GT DRP write operations are performed.
lockedn	1	Input	User output to drive tx_reset and rx_reset of Interlaken.
drp_en	1	Input	DRP enable signal. 0: No read or write operations performed. 1: Enables a read or write operation. For write operations, DRP_WE and DRP_EN should be driven High for one DRP_CLK cycle only.
drp_we	1	Input	DRP write enable. 0: Read operation when DRP_EN is 1. 1: Write operation when DRP_EN is 1. For write operations, DRP_WE and DRP_EN should be driven High for one DRP_CLK cycle only.
drp_addr	10	Input	DRP address bus.
drp_di	16	Input	Data bus for writing configuration data from the FPGA logic resources to the Interlaken core.
drp_do	16	Output	Data bus for reading configuration data from the ILKN to the FPGA logic resources.
usr_tx_reset	1	Output	TX reset output for the user.
usr_rx_reset	1	Output	RX reset output for the user.
drp_rdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
core_clk	1	Input	300/412 MHz core clock. The minimum core clock frequency is 300 MHz for 12 x 12.5 Gb/s mode.
lbus_clk	1	Input	Rate-adapting FIFO clock for the user side logic. LBUS signals are synchronized to this clock.
gt_loopback_in	36	Input	GT loopback input signal. See the *UltraScale Architecture GTY Transceivers User Guide (UG578)* [Ref 7].
gt_eyescanreset	12	Input	For the port description, see the *UltraScale Architecture GTY Transceivers User Guide (UG578)*. ***Note:*** This port is available when **Enable Additional GT Control and Status Ports** is selected from the **GT Selection and Configuration** tab.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt_rxdfelpmreset	12	Input	For the port description, see the *UltraScale Architecture GTY Transceivers User Guide (UG578)*. ***Note:*** This port is available when **Enable Additional GT Control and Status Ports** is selected from the **GT Selection and Configuration** tab.
gt_rxlpmen	12	Input	For the port description, see the *UltraScale Architecture GTY Transceivers User Guide (UG578)*. ***Note:*** This port is available when **Enable Additional GT Control and Status Ports** is selected from the **GT Selection and Configuration** tab. Port width: 10-bit for 100 Gigabit Attachment Unit Interface 10 (CAUI10) or Run time Selectable case and 4-bit width for CAUI4 mode)
gt_rxprbscntreset	12	Input	For the port description, see the *UltraScale Architecture GTY Transceivers User Guide (UG578)*. ***Note:*** This port is available when **Enable Additional GT Control and Status Ports** is selected from the **GT Selection and Configuration** tab. Port width: 10-bit for CAUI10 or Run time Selectable case and 4-bit width for CAUI4 mode)
gt_rxprbserr	12	Output	For the port description, see the *UltraScale Architecture GTY Transceivers User Guide (UG578)*. ***Note:*** This port is available when **Enable Additional GT Control and Status Ports** is selected from the **GT Selection and Configuration** tab.
gt_rxprbssel	48	Input	Refer GT user guide for the port description. ***Note:*** This port is available when **Enable Additional GT Control and Status Ports** is selected from the **GT Selection and Configuration** tab.
gt_rxresetdone	12	Output	For the port description, see the *UltraScale Architecture GTY Transceivers User Guide (UG578)*. ***Note:*** This port is available when **Enable Additional GT Control and Status Ports** is selected from the **GT Selection and Configuration** tab.
gt_txprbssel	48	Input	For the port description, see the *UltraScale Architecture GTY Transceivers User Guide (UG578)*. ***Note:*** This port is available when **Enable Additional GT Control and Status Ports** is selected from the **GT Selection and Configuration** tab.
gt_txresetdone	12	Output	For the port description, see the *UltraScale Architecture GTY Transceivers User Guide (UG578)*. ***Note:*** This port is available when **Enable Additional GT Control and Status Ports** is selected from the **GT Selection and Configuration** tab.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt_rxbufstatus	36	Output	For the port description, see the *UltraScale Architecture GTY Transceivers User Guide (UG578)*. ***Note:*** This port is available when **Enable Additional GT Control and Status Ports** is selected from the **GT Selection and Configuration** tab.
gt_drpclk	1	Input	DRP interface clock.
gt0_drpdo	16	Output	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.
gt0_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
gt0_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt0_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt0_drpaddr	10	Input	DRP address bus.
gt0_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt1_drpdo	16	Output	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.
gt1_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
gt1_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt1_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt1_drpaddr	10	Input	DRP address bus.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt1_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt2_drpdo	16	Output	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.
gt2_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
gt2_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt2_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt2_drpaddr	10	Input	DRP address bus.
gt2_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt3_drpdo	16	Output	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.
gt3_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
gt3_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt3_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt3_drpaddr	10	Input	DRP address bus.
gt3_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt4_drpdo	16	Output	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt4_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
gt4_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt4_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt4_drpaddr	10	Input	DRP address bus.
gt4_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt5_drpdo	16	Output	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.
gt5_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
gt5_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt5_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt5_drpaddr	10	Input	DRP address bus.
gt5_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt6_drpdo	16	Output	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.
gt6_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt6_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt6_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt6_drpaddr	10	Input	DRP address bus.
gt6_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt7_drpdo	16	Output	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.
gt7_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
gt7_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt7_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt7_drpaddr	10	Input	DRP address bus.
gt7_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt8_drpdo	16	Output	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.
gt8_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
gt8_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt8_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt8_drpaddr	10	Input	DRP address bus.
gt8_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt9_drpdo	16	Output	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.
gt9_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
gt9_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt9_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt9_drpaddr	10	Input	DRP address bus.
gt9_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt10_drpdo	16	Output	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.
gt10_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
gt10_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt10_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt10_drpaddr	10	Input	DRP address bus.
gt10_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt11_drpdo	16	Input	Data bus for reading configuration data from the GT transceiver to the FPGA logic resources.
gt11_drprdy	1	Output	Indicates operation is complete for write operations and data is valid for read operations.
gt11_drpen	1	Input	DRP enable signal. 0: No read or write operation performed. 1: Enables a read or write operation. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt11_drpwe	1	Input	DRP write enable. 0: Read operation when DRPEN is 1. 1: Write operation when DRPEN is 1. For write operations, DRPWE and DRPEN should be driven High for one DRPCLK cycle only.
gt11_drpaddr	10	Input	DRP address bus.
gt11_drpdi	16	Input	Data bus for writing configuration data from the FPGA logic resources to the transceiver.
gt_eyescantrigger	12	Input	Causes a trigger event.
gt_rxcdrhold	12	Input	Hold the clock data recovery (CDR) control loop frozen.
gt_rxpolarity	12	Input	The RXPOLARITY port can invert the polarity of incoming data: 0: Not inverted. RXP is positive and RXN is negative. 1: Inverted. RXP is negative and RXN is positive.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

<table>
<tr><th>Name</th><th>Size</th><th>Direction</th><th>Description</th></tr>
<tr><td>gt_rxrate</td><td>36</td><td>Input</td><td>Dynamic pins to automatically change effective PLL dividers in the GTH transceiver RX. These ports are used for PCI Express® and other standards.
000: Use RXOUT_DIV attributes
001: Divide by 1
010: Divide by 2
011: Divide by 4
100: Divide by 8
101: Divide by 16
110: Divide by 1
111: Divide by 1
RXBUF_RESET_ON_RATE_CHANGE attribute enables optional automatic reset.</td></tr>
<tr><td>gt_txdiffctrl</td><td>48</td><td>Input</td><td>Driver Swing Control. The default is user specified. All listed values are in mVPPD.
Note: The peak-to-peak differential voltage is defined when TXPOSTCURSOR = 5'b00000 and TXPRECURSOR = 5'b00000.
<table>
<tr><th>[3:0]</th><th>mVPPD</th></tr>
<tr><td>4'b0000</td><td>269</td></tr>
<tr><td>4'b0001</td><td>336</td></tr>
<tr><td>4'b0010</td><td>407</td></tr>
<tr><td>4'b0011</td><td>474</td></tr>
<tr><td>4'b0100</td><td>543</td></tr>
<tr><td>4'b0101</td><td>609</td></tr>
<tr><td>4'b0110</td><td>677</td></tr>
<tr><td>4'b0111</td><td>741</td></tr>
<tr><td>4'b1000</td><td>807</td></tr>
<tr><td>4'b1001</td><td>866</td></tr>
<tr><td>4'b1010</td><td>924</td></tr>
<tr><td>4'b1011</td><td>973</td></tr>
<tr><td>4'b1100</td><td>1018</td></tr>
<tr><td>4'b1101</td><td>1056</td></tr>
<tr><td>4'b1110</td><td>1092</td></tr>
<tr><td>4'b1111</td><td>1119</td></tr>
</table></td></tr>
</table>

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
gt_txpolarity	12	Input	The TXPOLARITY port is used to invert the polarity of outgoing data. 0: Not inverted. TXP is positive, and TXN is negative. 1: Inverted. TXP is negative, and TXN is positive.
gt_txpostcursor	60	Input	Transmitter post-cursor TX pre-emphasis control. The default is user specified. All listed values (dB) are typical. ***Note:*** The TXPOSTCURSOR values are defined when the TXPRECURSOR =5'b00000. Emphasis = 20log10(Vhigh/Vlow) = |20log10 (Vlow/Vhigh)|
gt_txprbsforceerr	12	Input	When this port is driven High, errors are forced in the pseudo-random binary sequence (PRBS) transmitter. While this port is asserted, the output data pattern contains errors. When TXPRBSSEL is set to 4'b0000, this port does not affect TXDATA.
gt_txprecursor	60	Input	Transmitter pre-cursor TX pre-emphasis control. The default is user specified. All listed values (dB) are typical. ***Note:*** The TXPRECURSOR values are defined when the TXPOSTCURSOR =5'b00000. Emphasis = 20log10(Vhigh/Vlow) = |20log10 (Vlow/Vhigh)|
gt_eyescandataerror	12	Output	Asserts high for one REC_CLK cycle when an (unmasked) error occurs while in the COUNT or ARMED state
gt_txbufstatus	24	Output	TXBUFSTATUS provides status for the TX Buffer or the TX asynchronous gearbox. When using the TX asynchronous gearbox, the port status is as follows. Bit 1: 0: No TX asynchronous gearbox FIFO overflow. 1: TX asynchronous gearbox FIFO overflow. Bit 0: 0: No TX asynchronous gearbox FIFO underflow. 1: TX asynchronous gearbox FIFO underflow. After the port is set High, it remains High until the TX asynchronous gearbox is reset.
gtpowergood_out	12	Output	Refer to the *UltraScale FPGAs Transceivers Wizard LogiCORE IP Product Guide* (PG182) [Ref 6] for the port description.
gt_refclk_out	1	Output	gt_refclk_out which is the same as gt_ref_clk to drive user fabric logic.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
ctl_tx_fc_stat	256	Input	TX In-Band Flow Control Input. These signals are used to set the status for each calendar position in the in-band-flow control mechanism (see the Interlaken Protocol specification [Ref 1]). A value of 1 means XON, a value of 0 means XOFF. These bits are transmitted in the Interlaken Control Word bits [55:40]. Each bit of CTL_TX_FC_STAT represents an entry in the flow control calendar with a length of 256 entries. When bit 56 of a Control Word is a value of 1, the first 16 calendar entries are output on bits 55-40. Specifically, Bit 55 of the first Control Word reflects the state of CTL_TX_FC_STAT[0], bit 54 reflects the state of CTL_TX_FC_STAT[1], bit 53 reflects the state of CTL_TX_FC_STAT[2], etc. as explained in the example in section 5.3.4.1 of Interlaken spec 1.1. Subsequent Control Words contain the next 16 calendar entries and so forth. This input must be synchronous with LBUS_CLK.
stat_tx_underflow_err	1	Output	TX Underflow. This signal indicates if the LBUS interface is being clocked too slowly to properly fill the link with data. In normal operation, this signal is always sampled as 0. If this signal is sampled as 1, the clocks are not set to proper frequencies and must be fixed.
stat_tx_burst_err	1	Output	TX BurstShort Error. When this signal is a value of 1, a burst (that is, a sequence of Data Words between two Control Words) was shorter than the value specified by CTL_TX_BURSTSHORT. This signal is only asserted if the final Control Word did not contain an EOP. This signal is provided to identify a poor scheduler design that results in reduced LBUS transaction errors. The TX core must be reset if this signal is asserted.
stat_tx_overflow_err	1	Output	TX Overflow. This output should never be asserted and indicates a critical failure. The core needs to be reset. This output is synchronous with the LBUS_CLK.
stat_rx_diagword_lanestat	12	Output	Lane Status messaging outputs. This bus reflects the most recent value in bit 33 of the Diagnostic Word received on the respective lane. These bits should only be considered valid if the respective bit in STAT_RX_CRC32_VALID is a value of 1. See Appendix A in the Interlaken Revision 1.2 specification.
stat_rx_diagword_intfstat	12	Output	Lane Status messaging outputs. This bus reflects the most recent value in bit 32 of the Diagnostic Word received on the respective lane. These bits should only be considered valid if the respective bit in STAT_RX_CRC32_VALID is a value of 1. See Appendix A in the Interlaken Revision 1.2 specification.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
stat_rx_crc32_valid	12	Output	Diagnostic Word CRC32 Valid. This bus reflects the validity of the CRC32 in the most recently received Diagnostic Word for the respective lane. A value of 1 indicated the CRC32 was valid and a value of 0 indicated the CRC32 was invalid. See section 5.4.6 of the Interlaken Revision 1.2 specification.
stat_rx_crc32_err	12	Output	Diagnostic Word CRC32 Error/Invalid. This bus provides indication of an invalid CRC32 in the Diagnostic Word for the respective lane. These signals are asserted with a value of 1 for one LBUS clock cycle each time an error is detected.
stat_rx_fc_stat	256	Output	RX Flow control Outputs. These signals indicate the flow control status for all of the calendar positions of the received data. A value of 1 means XON, a value of 0 means XOFF. These outputs reflect the information contained in bits 55-40 of the Control Words received by the RX. Only 256 In-Band flow control bits are supported. If a longer calendar is received, latter bits are ignored and are never output. If a shorter calendar is received, the bits of STAT_RX_FC_STAT that were not updated maintain their previous state. Each bit of STAT_RX_FC_STAT represents a received flow control calendar entry. STAT_RX_FC_STAT[0] is the first received calendar entry, STAT_RX_FC_STAT[1] is the second received calendar entry, STAT_RX_FC_STAT[2] is the third received calendar entry, etc. as explained in the example in section 5.3.4.1 Interlaken spec 1.2. Whenever a CRC24 or a loss of lane alignment occurs, all bits of STAT_RX_FC_STAT are set to a value of 0. This output is synchronous with the LBUS_CLK.
stat_rx_mubits	8	Output	RX Multiple-Use Control Bits. This bus contains the "Multi-Use" field of the Interlaken Control (see the Interlaken Protocol specification). The value of the bus are bits[31:24] of the most recently received Interlaken Control Word.
stat_rx_mubits_updated	1	Output	RX Multiple-Use/General Purpose Control Bits Updated. This output indicates that STAT_RX_MUBITS has been updated and is asserted for one clock cycle.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
stat_rx_word_sync	12	Output	64B/67B Word Boundary Locked. These signals indicate whether a lane is 64B/67B word boundary locked. A 64B/67B word boundary lock occurs if a lane detects 64 consecutive valid framing patterns on bits[65:64] as per the Interlaken Specification 1.2 Section 5.4.2. These signals are independent of both the Meta Frame Synchronization Word and Scrambler State Control Word. A value of 1 indicates the corresponding lane has achieved 64B/67B word boundary lock.
stat_rx_synced	12	Output	Word Boundary Synchronized. These signals indicate whether a lane is word boundary synchronized. A value of 1 indicates the corresponding lane has achieved word boundary synchronization as follows: a) 64B/67B Word Boundary Locked, b) Correctly receiving the Meta Frame Synchronization Word, and c) Correctly receiving the Scrambler State Control Word as described in sections 5.4.2, 5.4.3, and 5.4.4 of Interlaken spec 1.1. This output is synchronous with LBUS_CLK.
stat_rx_synced_err	12	Output	Word Boundary Synchronization Error. These signals indicate whether an error occurred during word boundary synchronization in the respective lane. A value of 1 indicates the corresponding lane had a word boundary synchronization error.
stat_rx_framing_err	12	Output	Framing Error. These signals indicate that an illegal framing pattern was detected in the respective lane. A value of 1 indicates an error occurred.
stat_rx_bad_type_err	12	Output	Unexpected or Illegal Meta Frame Control Word Block Type. These signals indicate an unexpected or illegal Meta Frame Control Word Block Type was detected. These signals can be used to collect the statistic "RX_Bad_Control_Error" as described in Table 5-9 of the Interlaken specification. A value of 1 indicates an error in the corresponding lane.
stat_rx_mf_err	12	Output	Meta Frame Synchronization Word Error. These signals indicate that an incorrectly formed Meta Frame Synchronization Word was detected in the respective lane. A value of 1 indicates an error occurred.
stat_rx_descram_err	12	Output	Scrambler State Control Word Error. These signals indicate a mismatch between the received Scrambler State Word and the expected value. A value of 1 indicates an error in the corresponding lane.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
stat_rx_mf_len_err	12	Output	Meta Frame Length Error. These signals indicate whether a Meta Frame length mismatch occurred in the respective lane. A value of 1 indicates the corresponding lane is receiving Meta Frame of wrong length.
stat_rx_mf_repeat_err	12	Output	Meta Frame Consecutive Error. These signals indicate whether consecutive Meta Frame errors occurred in the respective lane. A value of 1 indicates an error in the corresponding lane.
stat_rx_aligned	1	Output	All Lanes Aligned/De-Skewed. This signal indicates whether or not all lanes are aligned and de-skewed. A value of 1 indicates all lanes are aligned and de-skewed. When this signal is a 1, the RX path is aligned and can receive packet data.
stat_rx_misaligned	1	Output	Alignment Error. This signal indicates that the lane aligner did not receive the expected Meta Frame Synchronization Word across all (active) lanes. This signal can be used to collect the statistic "RX_Alignment_Error" as described in Table 5-9 of the Interlaken specification. This signal is not asserted until the Meta Frame Synchronization Word has been received at least once across all lanes. A value of 1 indicates the error occurred.
stat_rx_aligned_err	1	Output	Loss of Lane Alignment/De-Skew. This signal indicates an error occurred during lane alignment or lane alignment was lost. A value of 1 indicates an error occurred.
stat_rx_crc24_err	1	Output	Control Word CRC24 Error. This signal indicates whether or not a mismatch occurred between the received and the expected CRC24 value. A value of 1 indicates a mismatch occurred.
stat_rx_msop_err	1	Output	Missing Start of Packet Error. This signal indicates that a Missing Start of Packet was detected (and corrected).
stat_rx_meop_err	1	Output	Missing EOP Error. This signal indicates that a Missing EOP was detected (and corrected).
stat_rx_overflow_err	1	Output	RX FIFO Overflow Error. This signal indicates if the LBUS interface is being clocked too slowly to properly receive the data being transmitted across the link. A value of 1 indicates an error occurred. In normal operation, this signal is always sampled as 0. If this signal is sampled as 1, the clocks are not set to proper frequencies and must be fixed.

Table 5-3: **Core XCI Top-level Input and Output Ports *(Cont'd)***

Name	Size	Direction	Description
stat_rx_burstmax_err	1	Output	Interlaken RX BurstMax. This bus set the BurstMax parameter for the RX as follows: 0x0 = 64 bytes 0x1 = 128 bytes 0x2 = 192 bytes 0x3 = 256 bytes These inputs are only used in conjunction with STAT_RX_BURSTMAX_ERR.
stat_rx_burst_err	1	Output	Burst Error. This signal indicates that a BurstShort or a burst length error was detected.
txdata_in	768	Output	Core TX user data in when GT is present in the example design ***Note:*** This port is available for configurations when **Include GT subcore in example design** is selected from the ILKN/ GT Selection and Configuration tab.
rxdata_out	768	Input	Core RX user data in when GT is present in example design ***Note:*** This port is available for configurations when **Include GT subcore in example design** is selected from the ILKN/ GT Selection and Configuration tab.
tx_clk	1	Input	TX user clock ***Note:*** This port is available for configurations when **Include GT subcore in example design** is selected from the ILKN/GT Selection and Configuration tab.
rx_clk	1	Input	RX user clock ***Note:*** This port is available for configurations when **Include GT subcore in example design** is selected from the ILKN/GT Selection and Configuration tab.
axi_gt_reset_all	1	Output	User reset from the AXI4-Lite register map module ***Note:*** This port is available for configurations when **Include GT subcore in example design** is selected from the ILKN/GT Selection and Configuration tab.
axi_gt_loopback	1	Output	GT loopback signal to GT driven through AXI ***Note:*** This port is available for configurations when **Include GT subcore in example design** is selected from the ILKN/GT Selection and Configuration tab.

Note: AXI 4-Lite interface ports are visible only when you select **Include AXI4-Lite Control and Statistics Interface** option from **General** tab. For the AXI4-Lite interface port list and description, see AXI4-Lite User Interface Ports.

Modes of Operation

Three modes of operations are supported for this example design which are:

- Duplex Mode
- Simplex TX Mode
- Simplex RX Mode

Duplex Mode

In this mode of operation both Interlaken transmitter and receiver are active and loopback is provided at the GT output interface, that is, output is fed back as input. Packet generation and monitor are also active in this mode.

To enable this mode of operation, select **Duplex mode** from the parameters. Figure 5-4 shows the duplex mode of operation.

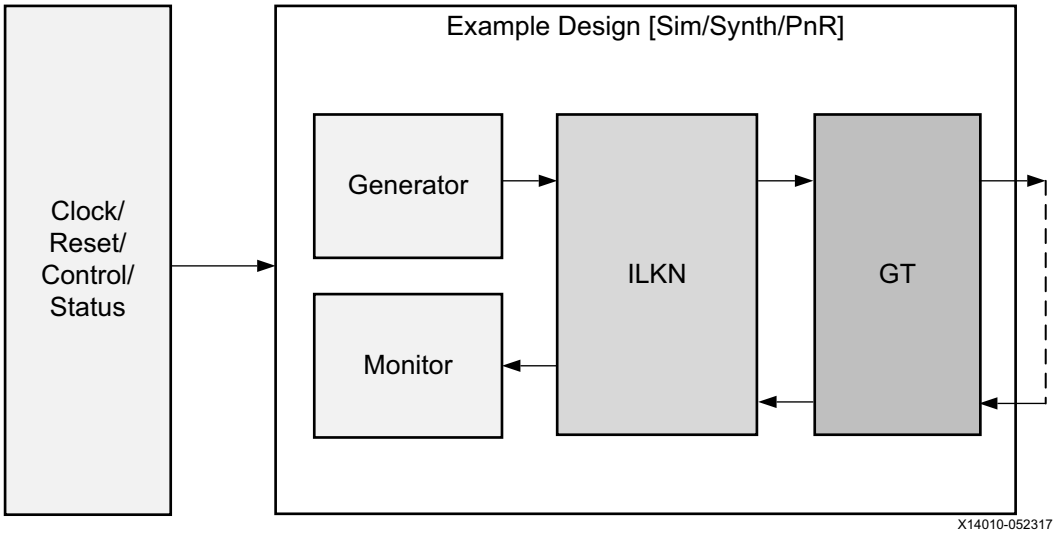


Figure 5-4: **Duplex Mode of Operation**

Simplex TX Mode

In this mode of operation only the Interlaken transmitter is enabled as shown in the diagram. Also, only packet generator is enabled for the generation of packets.

To enable this mode of operation, select **Simplex TX mode** from the parameters. Figure 5-5 shows the Simplex TX mode of operation.

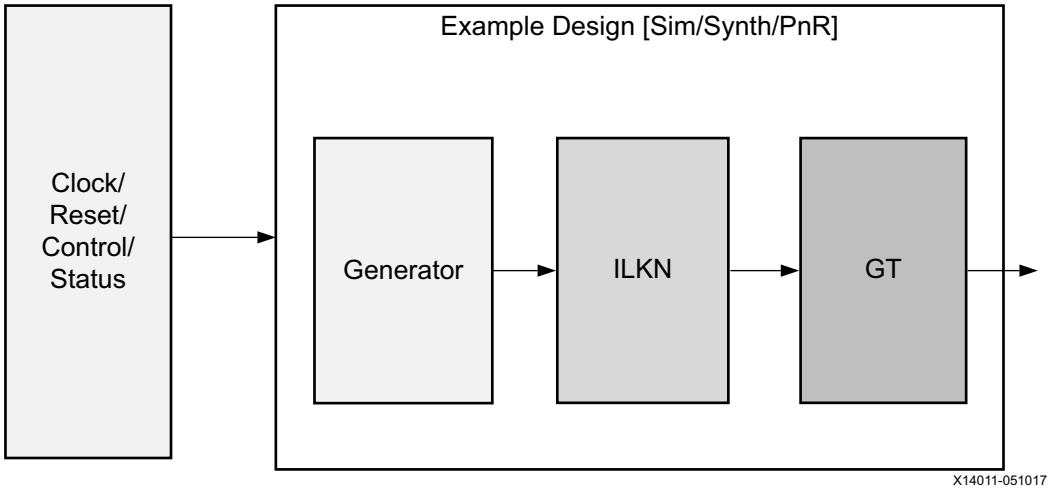


Figure 5-5: **Simplex TX Mode of Operation**

Simplex TX Mode Simulation

As shown in Figure 5-5, only the ILKN Transmitter is enabled and packet generator enabled for the generation of packets in this mode of operation. For simulation, a partner test bench is instantiated to perform the functionality of the ILKN Receiver. This partner test bench has an ILKN receiver and a packet monitor to verify the received data form the generator.

Example design Packet generator supports both packet mode and burst mode in Simplex TX. However, you must change the DATA_FLOW_MODE parameter value in **`interlaken_*_segmented_lbus_pkt_gen.v`** according to the Receiver mode (e.g., packet or burst). For example:

- if Simplex RX is configured as packet mode, set DATA_FLOW_MODE to 1, or
- if Simplex RX is configured as burst mode, set DATA_FLOW_MODE to 0.

Figure 5-6 shows the Simplex TX mode for simulation block diagram.

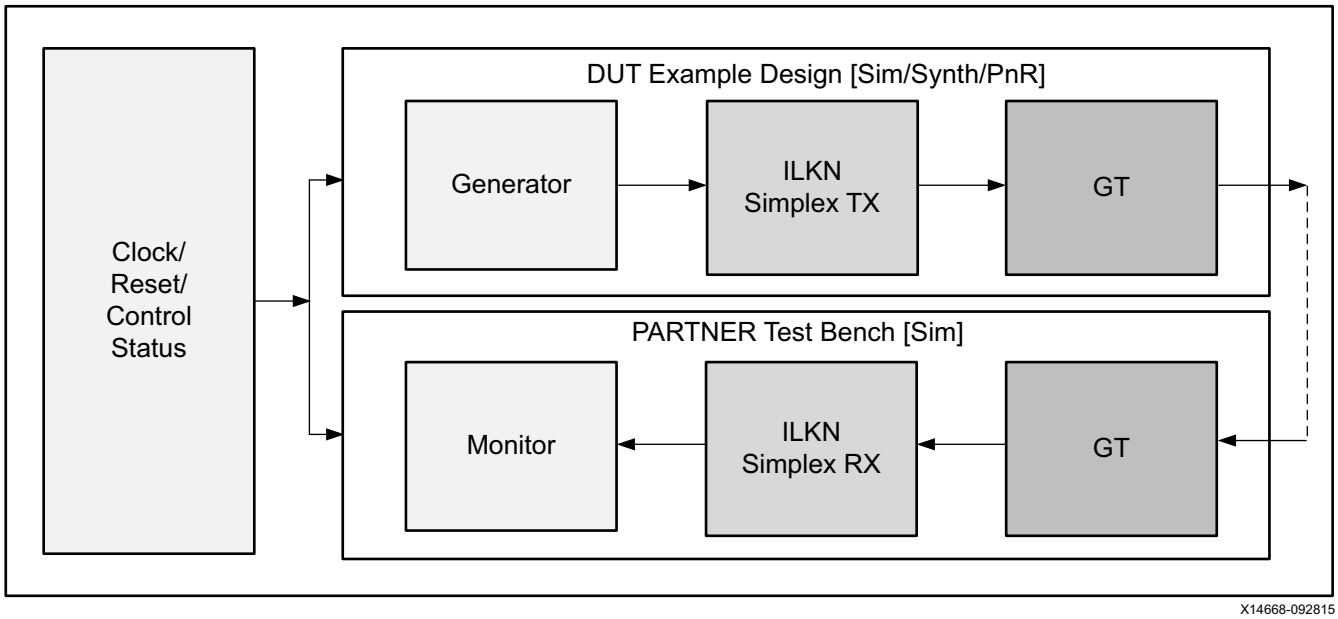


Figure 5-6: **Simplex TX Mode Simulation Block Diagram**

Simplex RX Mode

In this mode of operation only the Interlaken receiver is enabled as shown in Figure 5-7. Also only the packet monitor is enabled for the reception of packets.

To enable this mode of operation, select **Simplex RX mode** from the Vivado IDE parameters. Figure 5-7 shows the Simplex RX mode of operation.

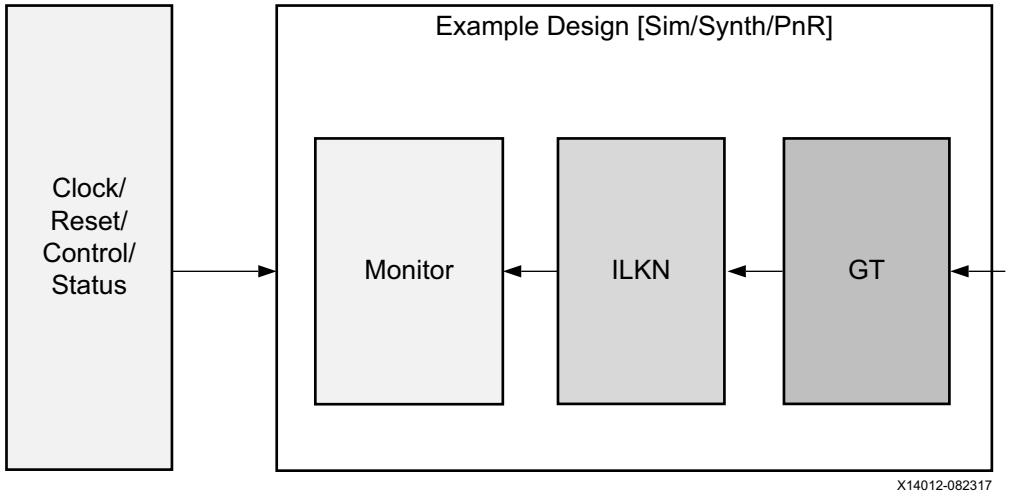


Figure 5-7: **Simplex RX Mode of Operation**

Simplex RX Mode Simulation

As shown in Figure 5-7, only the ILKN Receiver is enabled and packet monitor enabled to verify the received data in this mode of operation. For simulation, a partner test bench is instantiated to perform the functionality of the ILKN Transmitter. This partner test bench will have an ILKN Transmitter and a packet generator to generate the test data.

Packet generator supports both packet mode and burst mode in Simplex RX. However, you must change the DATA_FLOW_MODE parameter value in the **`interlaken_*_segmented_lbus_pkt_gen.v`** file according to the Receiver mode (e.g., packet or burst). For example:

- if Simplex RX is configured as packet mode, set DATA_FLOW_MODE as 1, or
- if Simplex RX is configured as burst mode, set DATA_FLOW_MODE as 0.

Figure 5-8 shows the Simplex RX mode for simulation block diagram.

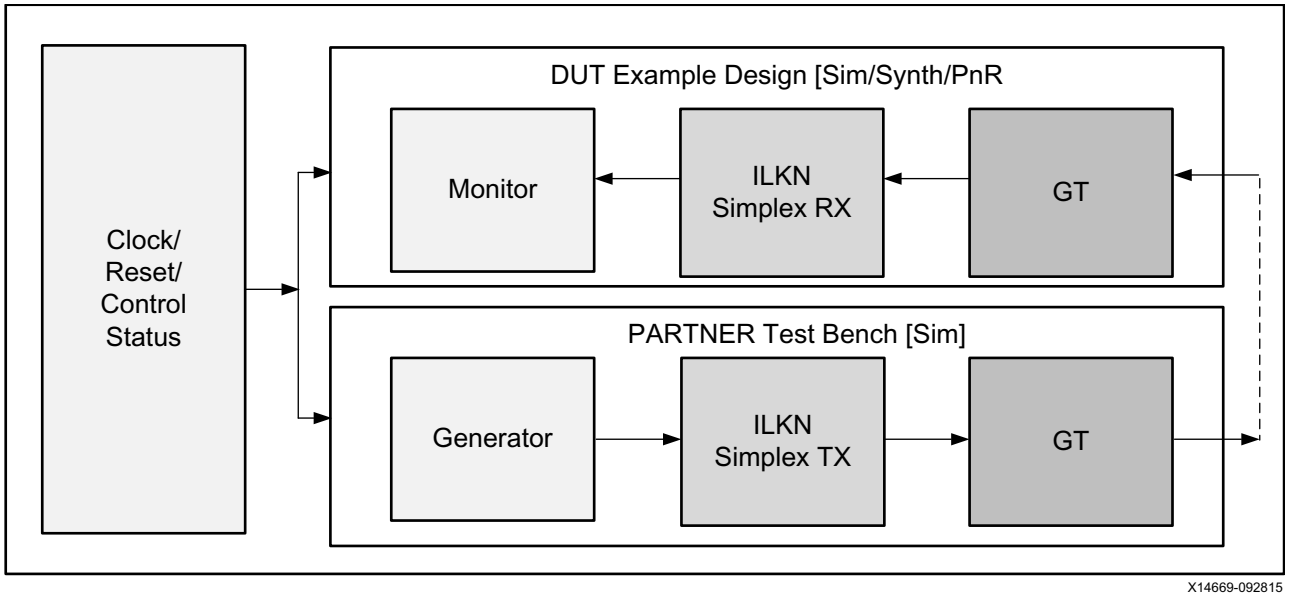


Figure 5-8: **Simplex RX Mode Simulation Block Diagram**

Transaction Flow

This section describes the flow of data between interlaken_0_pkt_gen_mon and interlaken_0 and various state transitions that happens within interlaken_0_pkt_gen and interlaken_0_pkt_mon.

Packet Generation

As mentioned earlier interlaken_0_pkt_gen is responsible for the generation of packets. Typically packet generator waits for the GT to get locked and Interlaken RX to get aligned. After alignment, the packet generator generates a predefined number of packets. The finite state machine (FSM) description of each state follows, and the state transition that occurs during this process is shown in Figure 5-9.

- **TOP_IDLE_STATE**: By default the controller will be in this state. When **reset_done** signal becomes active-High, it moves to GT_LOCK_STATE.
- **GT_LOCK_STATE**: Sets:

  ```
  ctl_tx_enable = 1`b0
  ctl_tx_mubits = 8`d0
  ctl_tx_diagword_lanestat = 12'hFFF
  ctl_tx_diagword_intfstat = 1'b1
  ```

  - **init_done = 1'b0**
  - and moves to WAIT_RX_ALIGN_STATE.
- **WAIT_RX_ALIGN_STATE**: Waits for the Interlaken core **stat_rx_aligned** or **simplex_mode_rx_aligned** in simplex TX mode, which indicates that the ILKN RX core is aligned. After that, FSM moves to ENABLE_PKT_TRANS_STATE state.
- **ENABLE_PKT_TRANS_STATE**: Set the **ctl_tx_enable=1`b1** and moves to PKT_TX_INIT_STATE.
- **PKT_TX_INIT_STATE**: Initialize all signals to start LBUS packet generation. When **init_done** and **tx_rdyout** is set, move to TRANSMIT_STATE_0.
- **TRANSMIT_STATE_0**: Checks for the number of packets to be generated, and generates predefined size of LBUS packets.
  - Moves to TRANSMIT_STATE_1 after sending respective SOP and EOP, and other LBUS control signals using the send_packet function.
  - After sending all the packets, the FSM moves to the DONE_STATE.
  - During transmission of the packets if ready =0, the FSM controller moves to HOLD_STATE.

- **TRANSMIT_STATE_1**: Sends remaining LBUS packet.
  - Once pkt_end reached, FSM moves to TRANSMIT_STATE_0 to end the current LBUS packet and start new packet
  - During transmission of the packets if **`ready =0`**, FSM controller moves to HOLD_STATE.
- **DONE_STATE**: Sets **`tx_done_int = 1`b1`** and moves to PKT_RESTART_STATE.
- **HOLD_STATE**: if **`ready =1`**, the FSM controller moves to TRANSMIT_STATE_0 or TRANSMIT_STATE_1 depending on which state it came from.
- **PKT_RESTART_STATE**: Sets **`gen_busy=0`** and moves to TOP_IDLE_STATE.

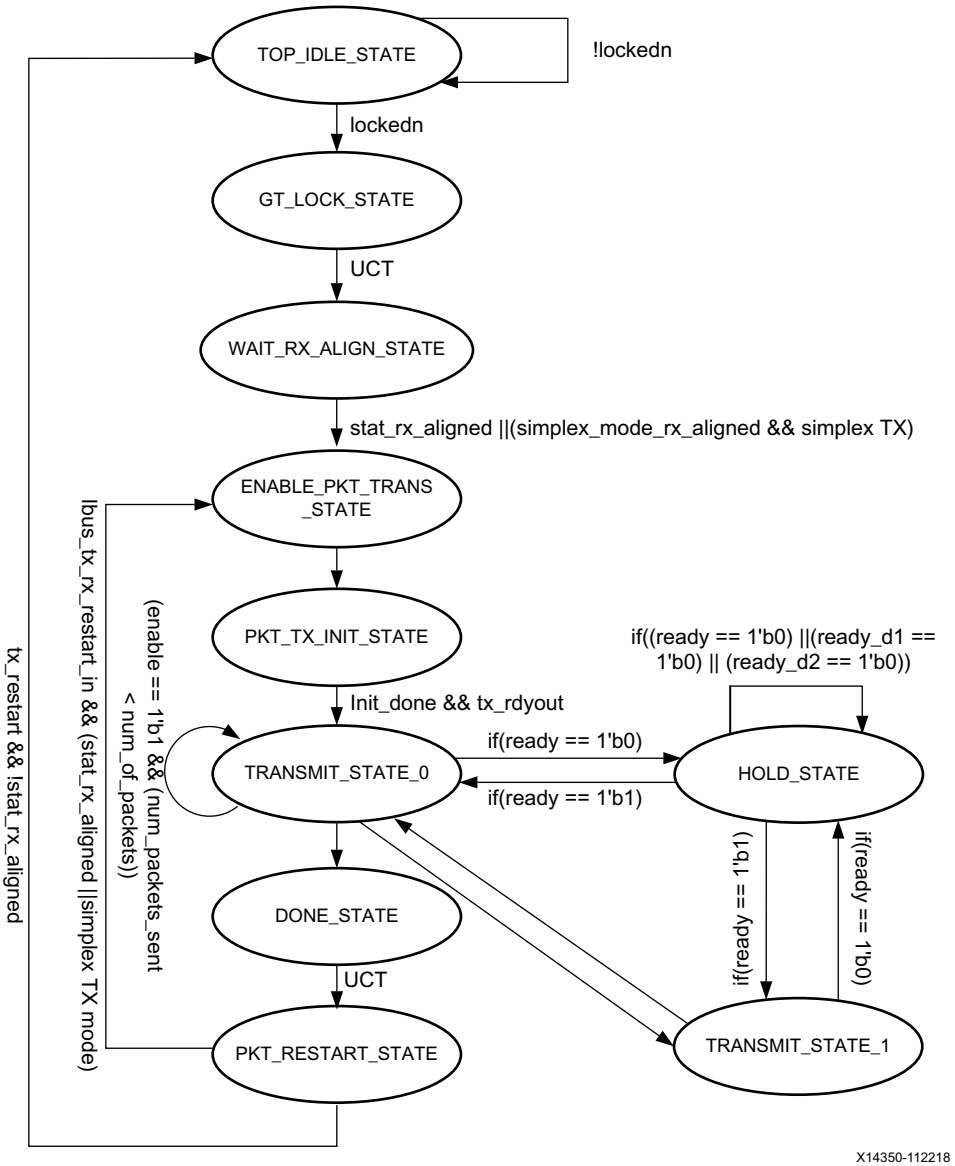


Figure 5-9: **State Transition Diagram for Packet Generator**

In the Simple TX mode of operation because RX alignment information is not available, the state machine waits for you to input **simplex_mode_rx_aligned**. After you assert this input, the packet transmission starts.

Packet Reception

The interlaken_0_pkt_mon is responsible for the reception of packets. Typically the packet monitor waits for the GT to get locked and the Interlaken RX to get aligned. After this is done the packet monitor receives a predefined number of packets. The FSM description of each state follows, and the state transition that occurs during this process is shown in Figure 5-9.

- **TOP_IDLE_STATE**: By default the FSM is in IDLE state. When the **reset_done** signal becomes active-high, the FSM moves to GT_LOCK_STATE.
- **GT_LOCK_STATE**: Sets the **mon_gt_locked=1** and **rx_busy=1**, and moves to WAIT_RX_ALIGN_STATE.
- **WAIT_RX_ALIGN_STATE**: Wait for the **rx_aligned=1**, which indicates the ILKN RX core is aligned, then the FSM moves to ENABLE_PKT_RECV_STATE.
- **ENABLE_PKT_RECV_STATE**: Sets the **enable=1** and moves to PKT_RX_INIT_STATE.
- **PKT_RX_INIT_STATE**: Initializes all signals related to LBUS packet reception, and the FSM moves to RECEIVE_STATE
- **RECEIVE_STATE**: Receives the LBUS packets and compares them with the expected packets. If there is any mismatch, sets the respective error signal.

  Data mismatch error signal for segment 0: **error_rx_data_1**.

  Data mismatch error signal for segment 1: **error_rx_data_2**.

  Data mismatch error signal for segment 2: **error_rx_data_3**.

  Data mismatch error signal for segment 3: **error_rx_data_4**.

  Channel mismatch error signal for segment 0: **error_rx_channel_1**.

  Channel mismatch error signal for segment 1: **error_rx_channel_2**.

  Channel mismatch error signal for segment 2: **error_rx_channel_3**.

  Channel mismatch error signal for segment 3: **error_rx_channel_4**.

  mty mismatch error signal: **error_rx_mty**.

  Any assertion of above mentioned error signal will set **rx_error_int = 1** and this flag is de-asserted only on reset. After receiving all the packets, the FSM moves to DONE_STATE.

- **DONE_STATE**: Sets **`rx_done_int = 1`**, and moves to PKT_RECV_MODE_STATE.
- **PKT_RECV_MODE_STATE**: Sets **`mon_rx_done = 1`** and **`mon_rx_failed`** to 0 or 1, depending on the **`rx_error_int`** signal, and moves to PKT_RESTART_STATE.
- **PKT_RESTART_STATE**: Resets all the signals related to LBUS packet monitor, and resets the **`rx_busy=0`**. Waits for the **`rx_restart_3d=1`**, and moves to the TOP_IDLE_STATE.

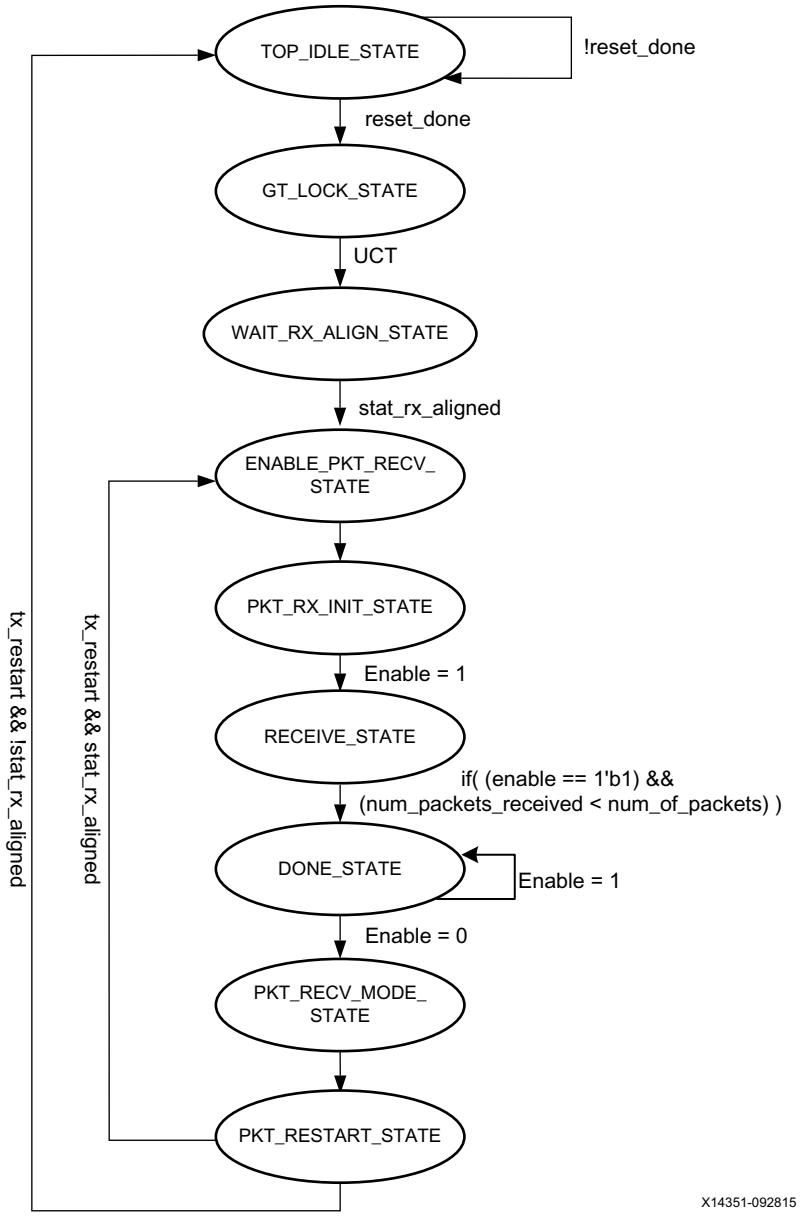


Figure 5-10: **State Transition Diagram for Packet Monitor**

Shared Logic Implementation

Shared logic includes the GT common module which can be present as part of the core or in the example design.

By default shared logic is present inside the core. If you want to instantiate shared logic in the example design, you must select the **Include Shared logic in example design** parameter in the Vivado IDE. Figure 5-11 shows the implementation when shared logic is instantiated in the Example design.

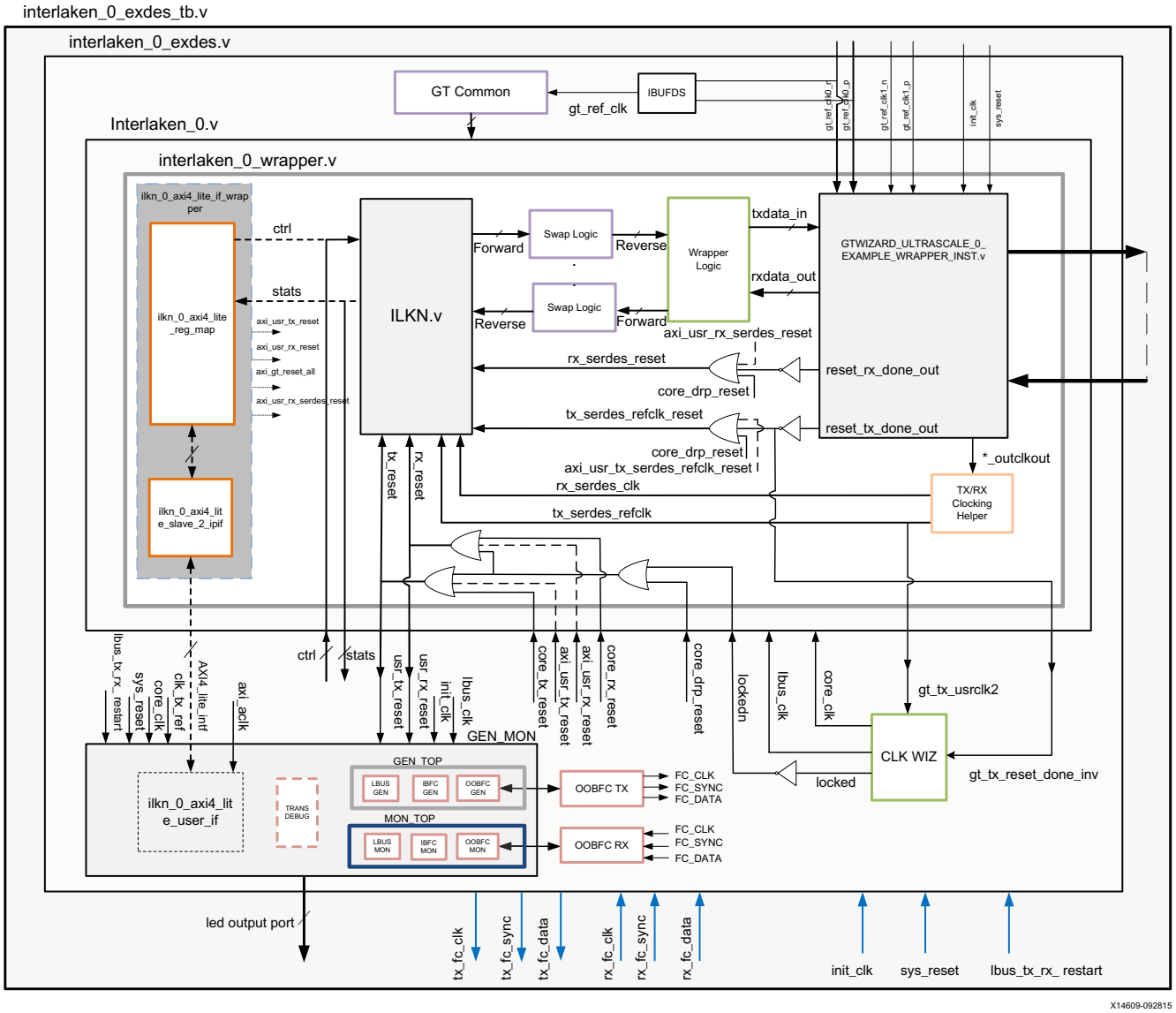


Figure 5-11: **Example Design Hierarchy with Shared Logic Implementation**

CORE DRP Operation

1. Make `core_drp_reset` signal High.
2. Perform the DRP write / read operation.
3. After the DRP operation, make the `core_drp_reset` signal Low.
4. Wait for core alignment.

AXI4-Lite Interface Implementation

If you want to instantiate AXI4-Lite interface to access the control and status registers of the ILKN core, they have to tick mark the Include **AXI4-Lite Control and Statistics Interface** check box in the **General** tab. It enables ilkn_0_axi4_lite_if_wrapper module (that contains ilkn_0_axi4_lite_reg_map along with the ilkn_0_axi4_lite_slave_2_ipif module) in the ilkn_0_wrapper. The user interface logic (ilkn_0_axi4_lite_user_if) used for accessing the registers (control, status and statistics) is present in ilkn_0_pkt_gen_mon module.

This mode enables the following features:

- **Configure all the CTL ports of the core through AXI4-Lite interface**. This operation is performed by writing to a set of address locations with the required data to the register map interface. The address location with the configuration register list is found in Table 5-6.
- **Access all the status and statistics registers from the core through AXI4-Lite interface**. This is performed by reading the address locations for the status and statistics registers through register map. Table 5-7 shows the address with the corresponding register descriptions.

The following diagram shows the implementation when the **Include AXI4-Lite Control and Statistics Interface** option is selected.

.h Header File

AXI4 register information such as register address, register name with bit position, mask value, access type and their default values are provided in the header (.h) file format when the IP core is generated with **Include AXI4-lite** enabled in the Vivado Design Suite and the header file can be found under the folder `header_files` of the project path.

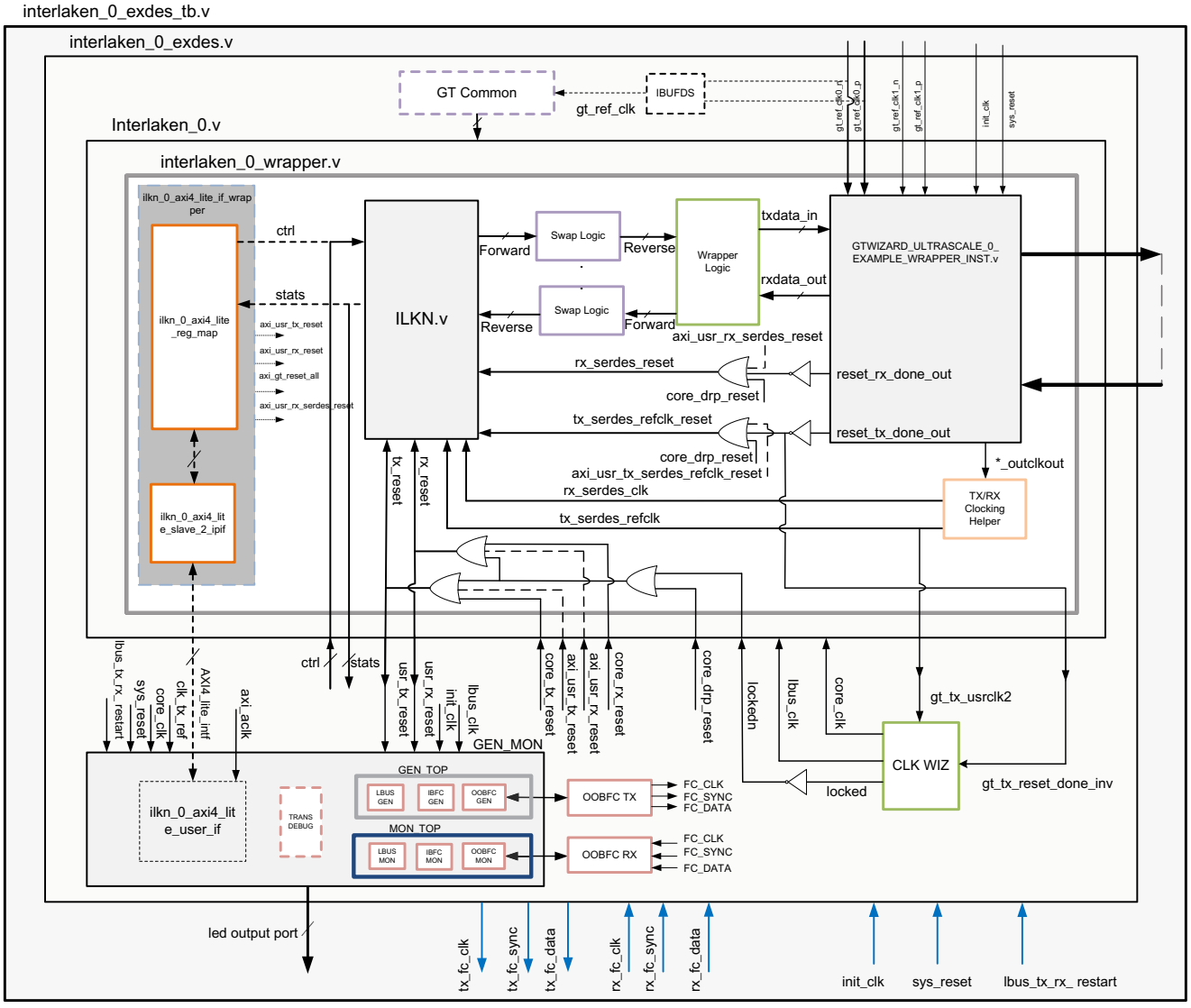


Figure 5-13: **Example Design Hierarchy with AXI4-Lite Interface**

AXI4-Lite Interface User Logic

The following sections provide the AXI4-Lite interface state machine control and ports.

User State Machine

The read and write through the AXI4-Lite slave module interface is controlled by a state machine as shown in Figure 5-14.

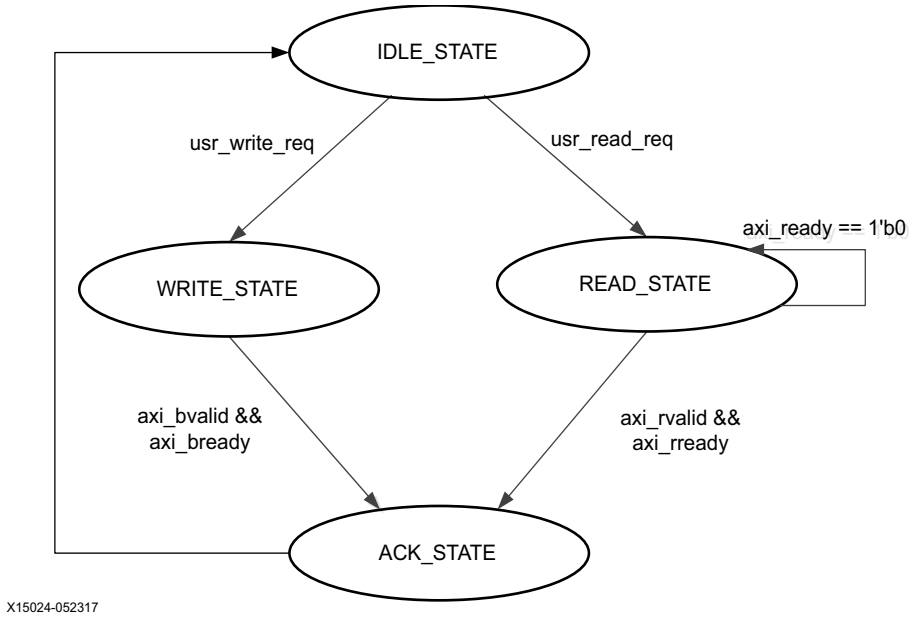


Figure 5-15: **State Machine for AXI4-Lite Interface**

The functional description of each state is as follows:

- **IDLE_STATE**: By default the FSM is in IDLE_STATE. When the **user_read_req** signal becomes High, then the state moves to READ_STATE. Otherwise, when the **user_write_req** signal is High, it moves to WRITE_STATE.
- **WRITE_STATE**: The user logic provide **S_AXI_AWVALID**, **S_AXI_AWADDR**, **S_AXI_WVALID**, **S_AXI_WDATA** and **S_AXI_WSTRB** in this state to write to the register map through the AXI4-Lite interface. When **S_AXI_BVALID** and **S_AXI_BREADY** from the AXI slave are High, then the state moves to ACK_STATE. If there is any write operation happens in any illegal addresses, the **S_AXI_BRESP[1:0]** indicates **2'b10** that asserts the write error signal to the user logic.
- **READ_STATE**: The user logic provides **S_AXI_ARVALID** and **S_AXI_ARADDR** in this state to read from the register map through AXI4-Lite interface. When **S_AXI_RVALID** and **S_AXI_RREADY** are High, then the state moves to ACK_STATE. If there is any read operation happens from any illegal addresses, the **S_AXI_RRESP[1:0]** indicates **2'b10** that asserts the read error signal to the user logic.
- **ACK_STATE**: The state moves to IDLE_STATE.

AXI4-Lite User Interface Ports

The interface ports are listed and described in Figure 5-5.

Table 5-4: **User Input / Output ports for the AXI4-Lite Interface**

Name	Size	Direction	Description
S_AXI_ACLK	1	Input	AXI clock signal.
S_AXI_SRESET	1	Input	AXI active-High synchronous reset.
S_AXI_PM_TICK	1	Input	PM tick user input.
S_AXI_AWADDR	32	Input	AXI write address.
S_AXI_AWVALID	1	Input	AXI write address valid.
S_AXI_AWREADY	1	Output	AXI write address ready.
S_AXI_WDATA	32	Input	AXI write data.
S_AXI_WSTRB	4	Input	AXI write strobe. This signal indicates which byte lanes hold valid data.
S_AXI_WVALID	1	Input	AXI write data valid. This signal indicates that valid write data and strobes are available.
S_AXI_WREADY	1	Output	AXI write data ready.
S_AXI_BRESP	2	Output	AXI write response. This signal indicates the status of the write transaction. 'b00 = OKAY 'b01 = EXOKAY 'b10 = SLVERR 'b11 = DECERR
S_AXI_BVALID	1	Output	AXI write response valid. This signal indicates that the channel is signaling a valid write response.
S_AXI_BREADY	1	Input	AXI write response ready.
S_AXI_ARADDR	32	Input	AXI read address.
S_AXI_ARVALID	1	Input	AXI read address valid.
S_AXI_ARREADY	1	Output	AXI read address ready.
S_AXI_RDATA	32	Output	AXI read data issued by slave.

Table 5-4: **User Input / Output ports for the AXI4-Lite Interface *(Cont'd)***

Name	Size	Direction	Description
S_AXI_RRESP	2	Output	AXI read response. This signal indicates the status of the read transfer. 'b00 = OKAY 'b01 = EXOKAY 'b10 = SLVERR 'b11 = DECERR
S_AXI_RVALID	1	Output	AXI read data valid.
S_AXI_RREADY	1	Input	AXI read ready. This signal indicates the user/master can accept the read data and response information.

User Side AXI4-Lite Write / Read Transactions

The timing diagram waveforms for the AXI4-Lite interface are shown as follows:

- Valid Write transactions (Figure 5-16)
- Invalid Write transactions (Figure 5-17)
- Valid Read transactions (Figure 5-18)
- Invalid Read transactions (Figure 5-19)

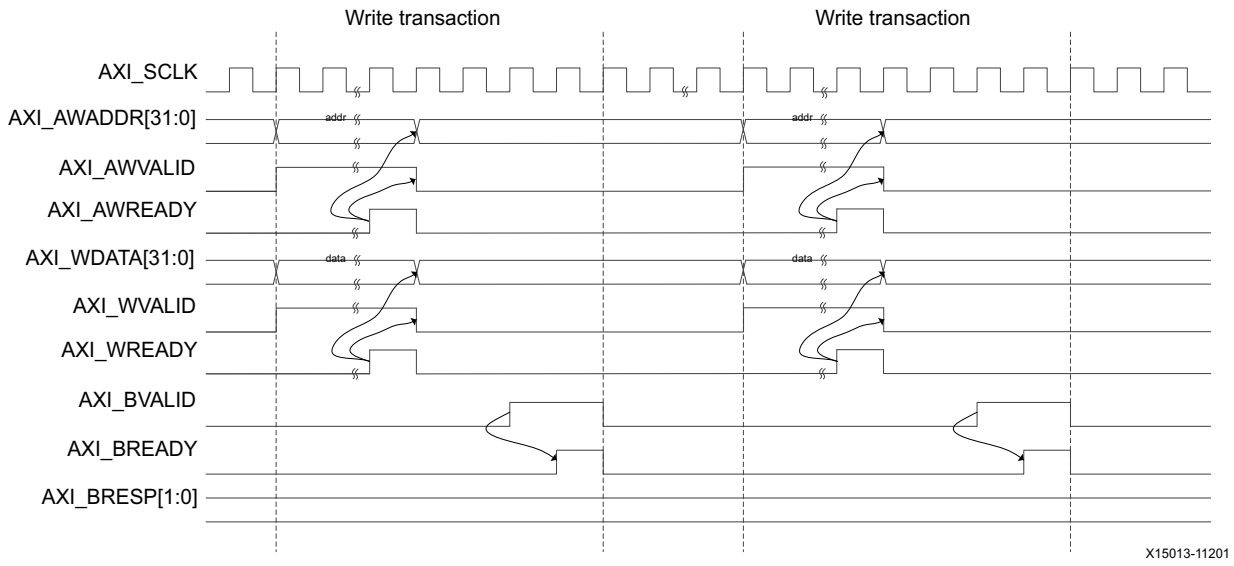


Figure 5-16: **AXI4-Lite User Side Valid Write Transaction**

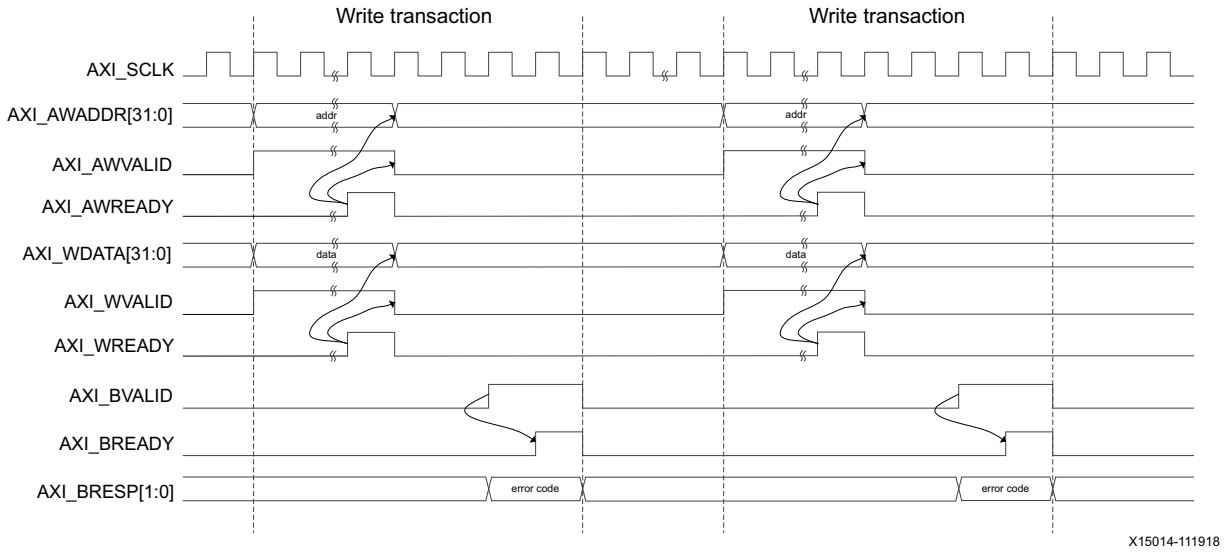


Figure 5-17: **AXI4-Lite User Side Write Transaction with Invalid Write Address**

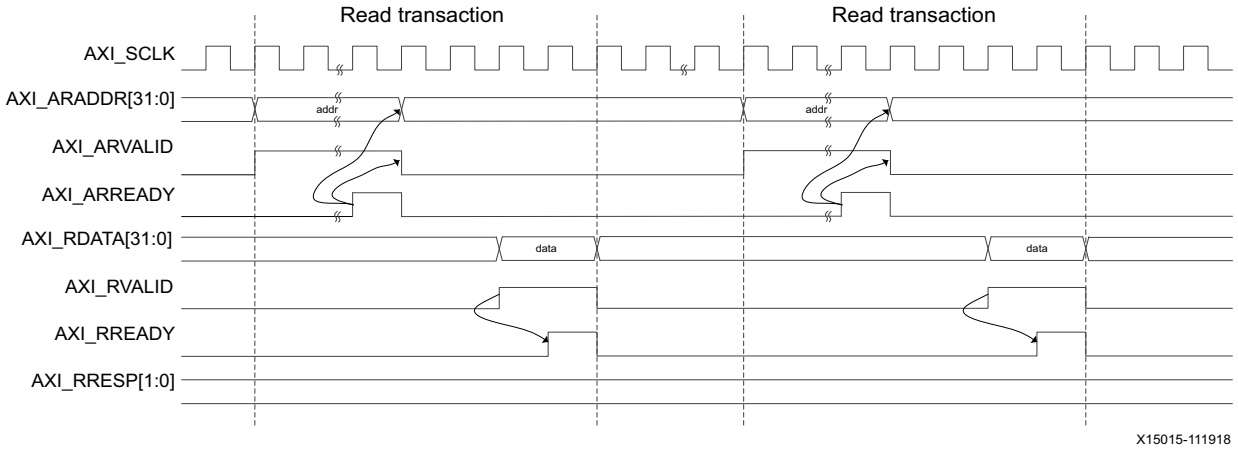


Figure 5-18: **AXI4-Lite User Side Valid Read Transaction**

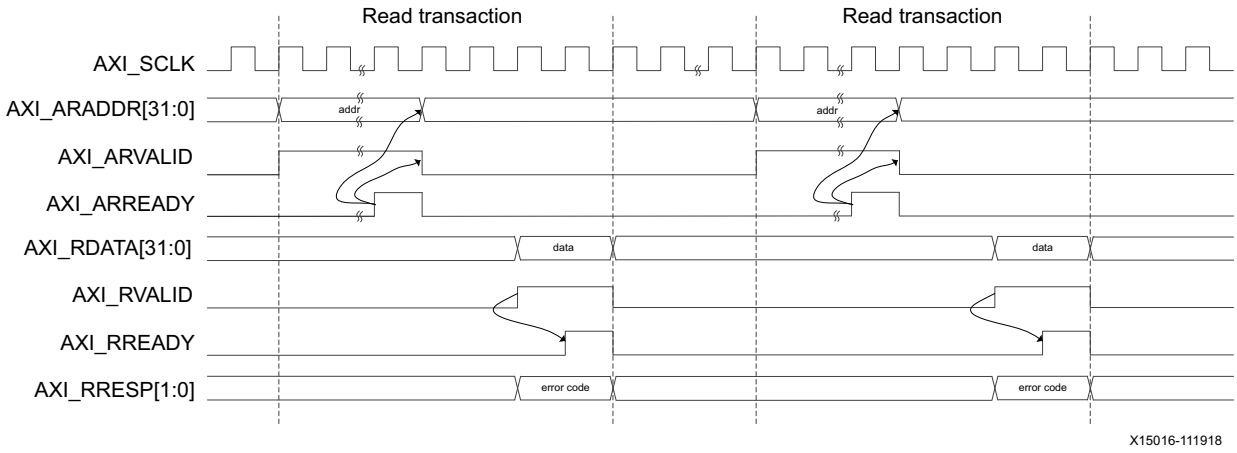


Figure 5-19: **AXI4-Lite User Side Read Transaction with Invalid Read Address**

Register Map

The following sections provide the register map and register descriptions for the core.

Base Pages

The register map is broken into two 512 base address pages to allow for future development and expansion, as shown in Table 5-5.

Table 5-5: **Register Base Addresses**

Base Address	Space Name
0x0000 0000	IP Configuration Registers
0x0000 0200	Status and Statistics Registers

All registers are 32 bytes in size, and aligned on 32-byte addressing. In the following register space maps, any holes in the address space should be considered RESERVED and might cause the AXI Control interface IP to respond with an error if accessed.

Configuration Register Space

The configuration space provides the software with the ability to configure the IP for various use-cases.

The integrated Interlaken IP makes use of a dynamic reconfiguration port (DRP) to provide you with the ability to configure aspects of the IP without the need for fabric logic connections. In this case those configuration bits in the soft AXI Control register set will become RESERVED (unused) and the software should use the DRP operation registers to configure those attributes of the IP. See Table 5-6 for the DRP address map.

Table 5-6: **Configuration Register Map**

Address	Register Name
0x0000	CORE_VERSION_REG
0x0004	GT_RESET_REG
0x0008	RESET_REG
0x000C	CONFIGURATION_TX_REG
0x0010	CONFIGURATION_TX_DIAGWORD_REG
0x0014	CONFIGURATION_RX_REG
0x0018	GT_LOOPBACK_REG
0x001C – 0x01FF	Reserved

Status and Statistics Register Space

The Status and Statistics registers provide an indication of the health of the link and histogram counters to provide classification of the traffic, and error counts.

The status and counters are all read-only. Some bits are sticky, that is, latching their values High or Low once set. This is indicated by the suffix LH (Latched High) or LL (Latched Low). Status registers are clear on read, counters controlled by a "tick" mechanism.

The counters accumulate their counts in an internal accumulator. A write to the TICK_REG register causes the accumulated counts to be pushed to the readable STAT_*_MSB/LSB registers and simultaneously clears the accumulators. The STAT_*_MSB/LSB registers can then be read. In this way all values stored in the statistics counters represent a snapshot over the same time-interval.

The STAT_CYCLE_COUNT_MSB/LSB register contains a count of the number of SerDes clock cycles between TICK_REG register writes. This allows for easy time-interval based statistics. The counters have a default width of 48 bits. The counters saturate to 1s. The values in the counters are held until the next write to the TICK_REG register.

- R/LL: Register bit defaults to 1; upon error condition this bit latches to 0; the bit is set back to its default state after each read.
- R/LH: Register bit defaults to 0; upon error condition this bit latches to 1; the bit is set back to its default state after each read

If the register bit is not defaulting to its respective value after each read, the error state is ongoing.

The addresses shown in Table 5-7 for the counters are the addresses of the LSB register, or bits 31:0 of the count. The MSB bits 47:32 of the counter are located at +0x4 from the LSB.

Table 5-7: **Status and Statistics Register Map**

Address	Register Name
0x0200	STAT_TX_STATUS_REG
0x0204	STAT_RX_STATUS_REG
0x0208	STAT_RX_DIAGWORD_REG
0x020C	STAT_RX_MUBITS_REG
0x0210	STAT_RX_SYNCED_REG
0x0214	STAT_RX_SYNCED_ERR_REG
0x0218	STAT_RX_MF_ERR_REG
0x021C	STAT_RX_MF_LEN_ERR_REG
0x0220	STAT_RX_MF_REPEAT_ERR_REG
0x0224	STAT_RX_DESCRAM_ERR_REG
0x0228 – 0x02AF	Reserved
Histogram / Counter Registers	
0x02B0	TICK_REG
0x02B8	STAT_CYCLE_COUNT
0x02C0	STAT_RX_CRC32_ERR_LANE0
0x02C8	STAT_RX_CRC32_ERR_LANE1
0x02D0	STAT_RX_CRC32_ERR_LANE2
0x02D8	STAT_RX_CRC32_ERR_LANE3
0x02E0	STAT_RX_CRC32_ERR_LANE4
0x02E8	STAT_RX_CRC32_ERR_LANE5
0x02F0	STAT_RX_CRC32_ERR_LANE6
0x02F8	STAT_RX_CRC32_ERR_LANE7
0x0300	STAT_RX_CRC32_ERR_LANE8
0x0308	STAT_RX_CRC32_ERR_LANE9
0x0310	STAT_RX_CRC32_ERR_LANE10
0x0318	STAT_RX_CRC32_ERR_LANE11
0x0320	STAT_RX_CRC24_ERR
0x0328	STAT_RX_BAD_TYPE_ERR_LANE0

Table 5-7: **Status and Statistics Register Map *(Cont'd)***

Address	Register Name
0x0330	STAT_RX_BAD_TYPE_ERR_LANE1
0x0338	STAT_RX_BAD_TYPE_ERR_LANE2
0x0340	STAT_RX_BAD_TYPE_ERR_LANE3
0x0348	STAT_RX_BAD_TYPE_ERR_LANE4
0x0350	STAT_RX_BAD_TYPE_ERR_LANE5
0x0358	STAT_RX_BAD_TYPE_ERR_LANE6
0x0360	STAT_RX_BAD_TYPE_ERR_LANE7
0x0368	STAT_RX_BAD_TYPE_ERR_LANE8
0x0370	STAT_RX_BAD_TYPE_ERR_LANE9
0x0378	STAT_RX_BAD_TYPE_ERR_LANE10
0x0380	STAT_RX_BAD_TYPE_ERR_LANE11
0x0388	STAT_RX_FRAMING_ERR_LANE0_LSB
0x038C	STAT_RX_FRAMING_ERR_LANE0_MSB
0x0390	STAT_RX_FRAMING_ERR_LANE1_LSB
0x0394	STAT_RX_FRAMING_ERR_LANE1_MSB
0x0398	STAT_RX_FRAMING_ERR_LANE2_LSB
0x039C	STAT_RX_FRAMING_ERR_LANE2_MSB
0x03A0	STAT_RX_FRAMING_ERR_LANE3_LSB
0x03A4	STAT_RX_FRAMING_ERR_LANE3_MSB
0x03A8	STAT_RX_FRAMING_ERR_LANE4_LSB
0x03AC	STAT_RX_FRAMING_ERR_LANE4_MSB
0x03B0	STAT_RX_FRAMING_ERR_LANE5_LSB
0x03B4	STAT_RX_FRAMING_ERR_LANE5_MSB
0x03B8	STAT_RX_FRAMING_ERR_LANE6_LSB
0x03BC	STAT_RX_FRAMING_ERR_LANE6_MSB
0x03C0	STAT_RX_FRAMING_ERR_LANE7_LSB
0x03C4	STAT_RX_FRAMING_ERR_LANE7_MSB
0x03C8	STAT_RX_FRAMING_ERR_LANE8_LSB
0x03CC	STAT_RX_FRAMING_ERR_LANE8_MSB
0x03D0	STAT_RX_FRAMING_ERR_LANE9_LSB

Table 5-7: **Status and Statistics Register Map *(Cont'd)***

Address	Register Name
0x03D4	STAT_RX_FRAMING_ERR_LANE9_MSB
0x03D8	STAT_RX_FRAMING_ERR_LANE10_LSB
0x03DC	STAT_RX_FRAMING_ERR_LANE10_MSB
0x03E0	STAT_RX_FRAMING_ERR_LANE11_LSB
0x03E4	STAT_RX_FRAMING_ERR_LANE11_MSB
0x3E4 – 0x07FF	Reserved

Register Descriptions

Table 5-8: **CORE_VERSION_REG**

Address	Bits	Default	Type	Description
0x0000	7:0	minor	R	Current version of the core in the format "major.minor". For example, core version 1.9.
	15:8	major	R	Bits [7:0] represents minor version that is 9. Bits [15:8] represents major version that is 1.
	31:16	0	NA	Reserved

Table 5-9: **GT_RESET_REG**

Address	Bits	Default	Type	Description
0x0004	0	0	RW	gt_reset_all. A write of 1 issues a RESET to the GTs. This is a clear on write register.
	31:1	0	NA	Reserved

Table 5-10: **RESET_REG**

Address	Bits	Default	Type	Description
0x0008	11:0	0	RW	usr_rx_serdes_reset. Unused PCS bits are RESERVED. A write of 1 in a given bit location puts that PCS lane logic into reset.
	12	0	RW	usr_tx_serdes_refclk_reset. A write of 1 puts the TX Serdes Refclk path in reset.
	29:13	0	NA	Reserved
	30	0	RW	usr_rx_reset. RX core reset. A write of 1 puts the RX path in reset.
	31	0	RW	usr_tx_reset. TX core reset. A write of 1 puts the TX path in reset.

Table 5-11: **CONFIGURATION_TX_REG**

Address	Bits	Default	Type	Description
0x000C	0	0	RW	ctl_tx_enable
	8:1	0	RW	ctl_tx_mubits
	31:9	0	NA	Reserved

Table 5-12: **CONFIGURATION_TX_DIAGWORD_REG**

Address	Bits	Default	Type	Description
0x0010	0	0	RW	ctl_tx_diagword_intfstat
	12:1	0	RW	ctl_tx_diagword_lanestat
	31:13	0	NA	Reserved

Table 5-13: **CONFIGURATION_RX_REG**

Address	Bits	Default	Type	Description
0x0014	0	0	RW	ctl_rx_force_resync
	31:1	0	NA	Reserved

Table 5-14: **GT_LOOPBACK_REG**

Address	Bits	Default	Type	Description
0x0018	0	0	RW	axi_gt_loopback 0 is for Normal operation 1 is for Near End PMA loopback
	31:1	0	NA	Reserved

Table 5-15: **STAT_TX_STATUS_REG**

Address	Bits	Default	Type	Description
0x0200	0	0	R/LH	stat_tx_underflow_err
	1	0	R/LH	stat_tx_overflow_err
	2	0	R/LH	stat_tx_burst_err
	3	1	R/LL	stat_tx_errinj_biterr_done
	31:4	0	NA	Reserved

Table 5-16: **STAT_RX_STATUS_REG**

Address	Bits	Default	Type	Description
0x0204	0	1	R/LL	stat_rx_aligned. Default value is the value after reset.
	1	0	R/LH	stat_rx_aligned_err
	2	0	R/LH	stat_rx_misaligned
	3	0	R/LH	stat_rx_msop_err
	4	0	R/LH	stat_rx_meop_err
	5	0	R/LH	stat_rx_burst_err
	6	0	R/LH	stat_rx_burstmax_err
	7	0	R/LH	stat_rx_overflow_err
	19:8	12'hFFF	R/LL	stat_rx_word_sync
	31:20	0	NA	Reserved

Table 5-17: **STAT_RX_DIAGWORD_REG**

Address	Bits	Default	Type	Description
0x0208	11:0	12'hFFF	R/LL	stat_rx_diagword_lanestat
	23:12	12'hFFF	R/LL	stat_rx_diagword_intfstat
	31:24	0	NA	Reserved

Table 5-18: **STAT_RX_MUBITS_REG**

Address	Bits	Default	Type	Description
0x020C	7:0	0	R/LH	stat_rx_mubits
	31:8	0	NA	Reserved

Table 5-19: **STAT_RX_SYNCED_REG**

Address	Bits	Default	Type	Description
0x0210	11:0	12'hFFF	R/LL	stat_rx_synced
	31:12	0	NA	Reserved

Table 5-20: **STAT_RX_SYNCED_ERR_REG**

Address	Bits	Default	Type	Description
0x0214	11:0	0	R/LH	stat_rx_synced_err
	31:12	0	NA	Reserved

Table 5-21: **STAT_RX_MF_ERR_REG**

Address	Bits	Default	Type	Description
0x0218	11:0	0	R/LH	stat_rx_mf_err
	31:12	0	NA	Reserved

Table 5-22: **STAT_RX_MF_LEN_ERR_REG**

Address	Bits	Default	Type	Description
0x021C	11:0	0	R/LH	stat_rx_mf_len_err
	31:12	0	NA	Reserved

Table 5-23: **STAT_RX_MF_REPEAT_ERR_REG**

Address	Bits	Default	Type	Description
0x0220	11:0	0	R/LH	stat_rx_mf_repeat_err
	31:12	0	NA	Reserved

Table 5-24: **STAT_RX_FRAMING_ERR_REG**

Address	Bits	Default	Type	Description
0x0224	11:0	0	R/LH	stat_rx_framing_err
	31:12	0	NA	Reserved

Table 5-25: **STAT_RX_DESCRAM_ERR_REG**

Address	Bits	Default	Type	Description
0x0228	11:0	0	R/LH	stat_rx_descram_err
	31:12	0	NA	Reserved

Table 5-26: **TICK_REG**

Address	Bits	Default	Type	Description
0x02B0	0	0	WO/SC	tick_reg. Writing a 1 to the Tick bit triggers a snapshot of all the Statistics Counters into their readable registers. The bit self-clears, thus only a single write is required by the user logic.
	31:1	0	NA	Reserved

Sample Statistics Counter

A sample statistics counter is illustrated in Table 5-27. The format for the counters is the same for all counter types.

After the 'tick' is issued, the counters will contain their updated value and can be read multiple times without destruction of this data.

Table 5-27: **STAT_RX_CRC32_ERR_LANE0[47:0]**

Address	Bits	Default	Type	Description
0x02C0	32	0	R	stat_rx_crc32_err_lane0_lsb[31:0]
0x02C4	16	0	R	stat_rx_crc32_err_lane0_msb[15:0]

Use Case for Different Modes

This section describes the use case for different modes of operation of the Interlaken core.

Simulation — Duplex/Simplex RX Mode

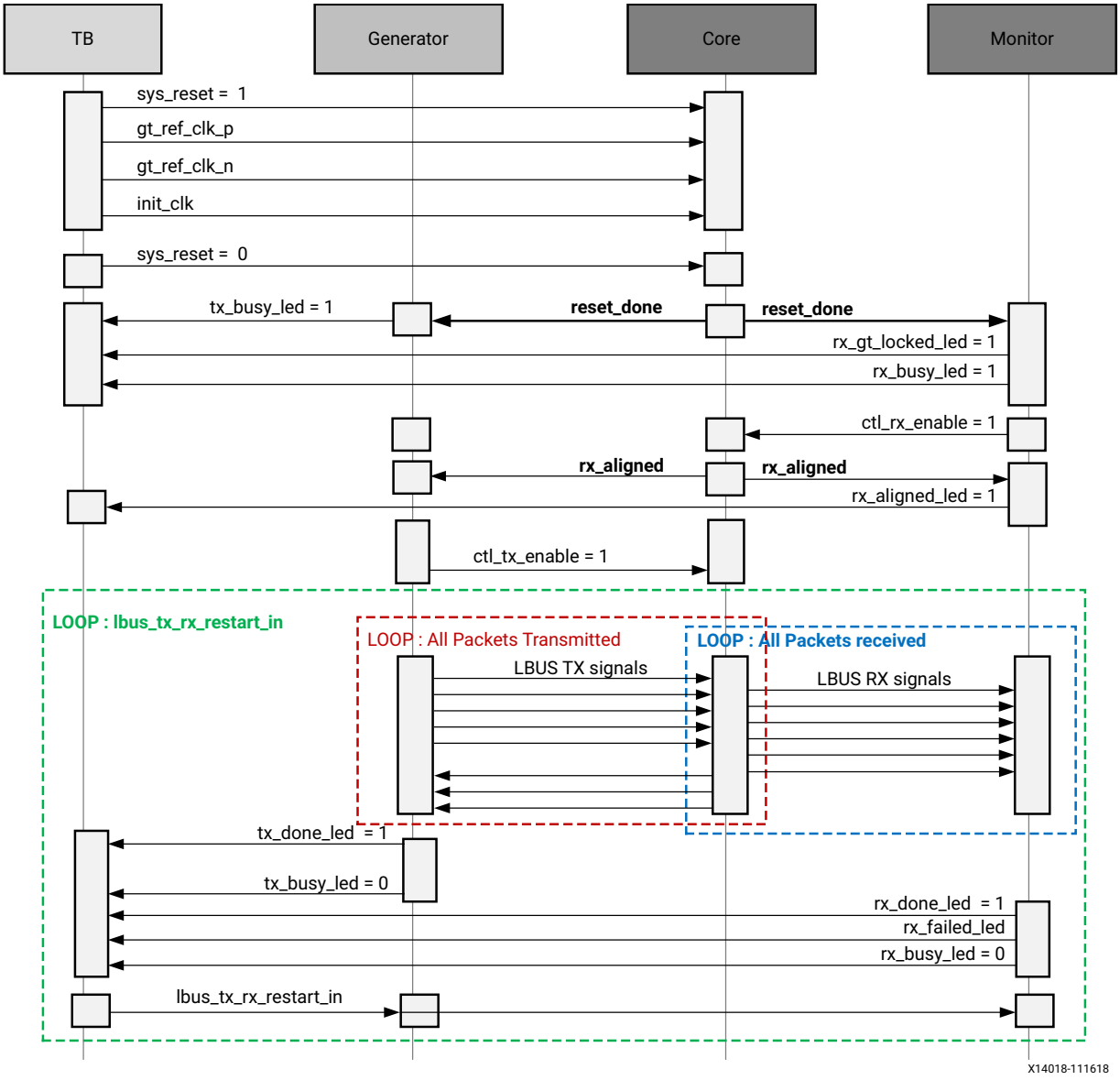


Figure 5-20: **Simulation Use Case for Duplex/Simplex RX Configuration**

Simulation — Simplex TX Mode

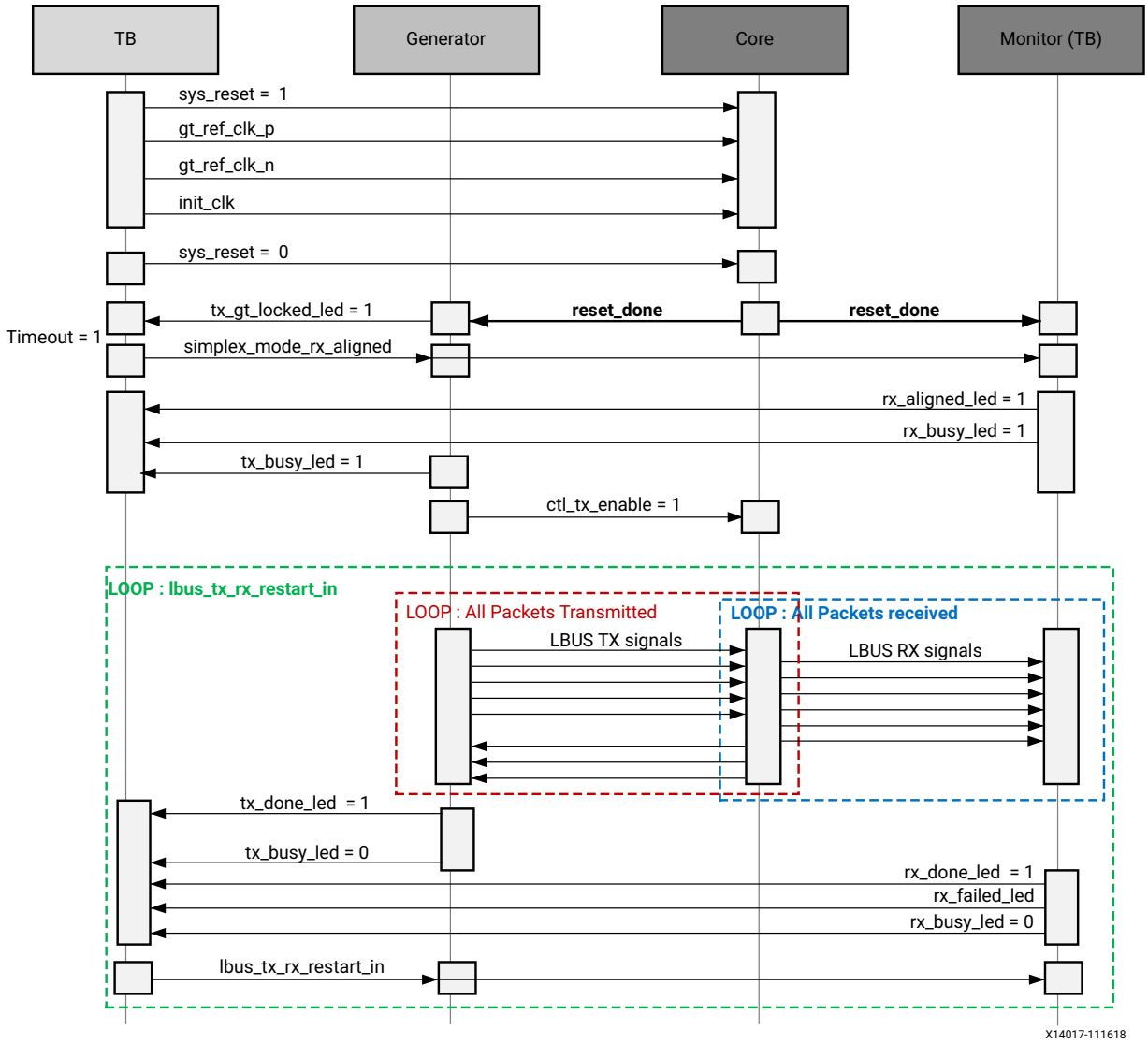


Figure 5-21: **Simulation Use Case for Simplex TX Configuration**

Validation — Duplex/Simplex RX mode

The following table describes the LED behavior and input switch condition for the validation of the Interlaken core onboard for Duplex/Simplex RX mode configuration.

Note: Green color indicates the successful completion of the respective test. Red color indicates the current process is busy or respective test failed.

Validation — Passing Scenario For Duplex/Simplex RX Mode

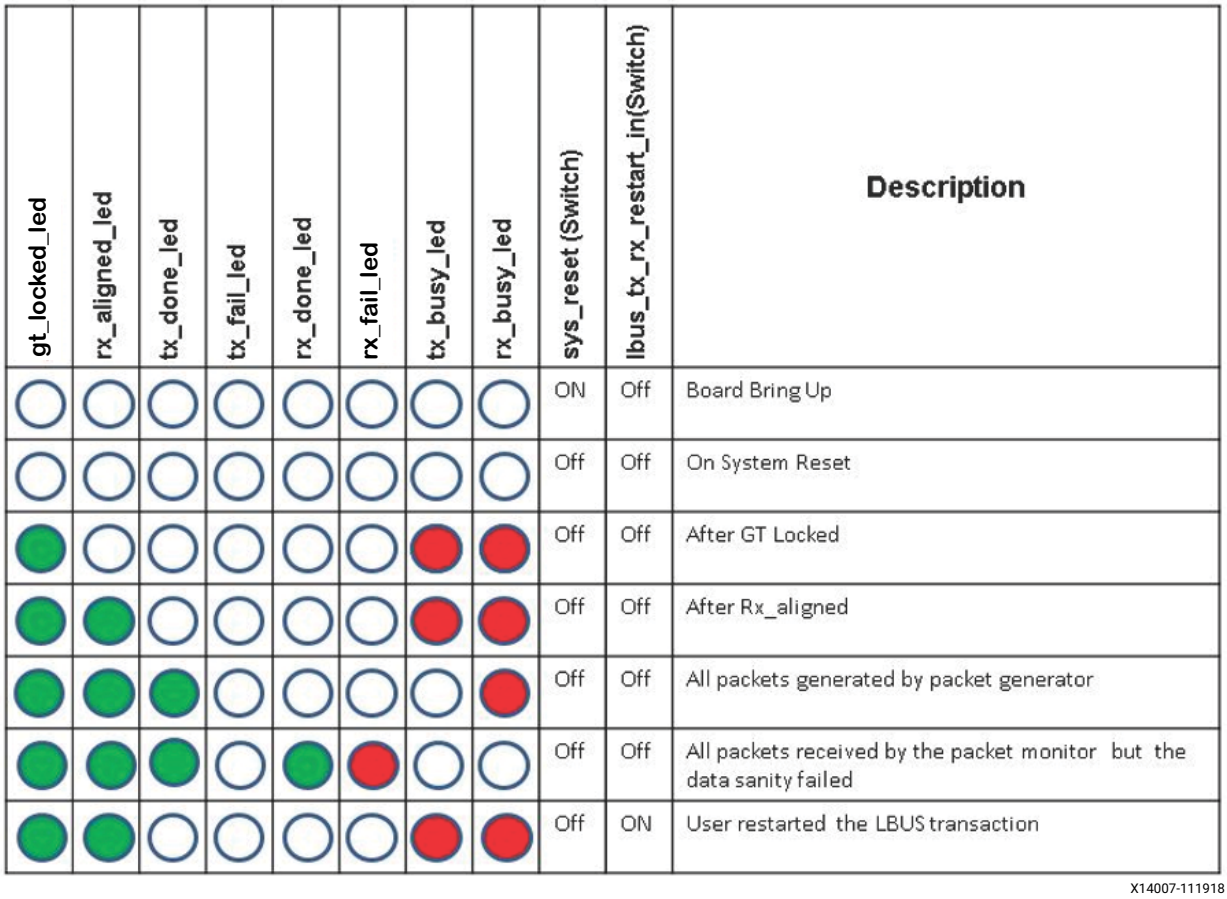
Figure 5-22 describes the LED behavior and input switch condition for the validation of the Interlaken core for passing scenario. Where the packet generator and monitor are active onboard for Duplex/Simplex RX mode configuration.

gt_locked_led	rx_aligned_led	tx_done_led	tx_fail_led	rx_done_led	rx_fail_led	tx_busy_led	rx_busy_led	sys_reset (Switch)	lbus_tx_rx_restart_in(Switch)	Description
○	○	○	○	○	○	○	○	ON	Off	Board Bring Up
○	○	○	○	○	○	○	○	Off	Off	On System Reset
●	○	○	○	○	○	●	●	Off	Off	After GT Locked
●	●	○	○	○	○	●	●	Off	Off	After Rx_aligned
●	●	●	○	○	○	○	●	Off	Off	All packets generated by packet generator
●	●	●	○	●	○	○	○	Off	Off	All packets received by the packet monitor without error
●	●	○	○	○	○	●	●	Off	ON	User restarted the LBUS transaction

X14006-092815

Figure 5-22: **Board Validation for Duplex/Simplex RX Configuration - Passing Scenario**

Validation — Failing Scenario For Duplex/Simplex RX Mode



gt_locked_led	rx_aligned_led	tx_done_led	tx_fail_led	rx_done_led	rx_fail_led	tx_busy_led	rx_busy_led	sys_reset (Switch)	lbus_tx_rx_restart_in(Switch)	Description
○	○	○	○	○	○	○	○	ON	Off	Board Bring Up
○	○	○	○	○	○	○	○	Off	Off	On System Reset
●	○	○	○	○	○	●	●	Off	Off	After GT Locked
●	●	○	○	○	○	●	●	Off	Off	After Rx_aligned
●	●	●	○	○	○	○	●	Off	Off	All packets generated by packet generator
●	●	●	○	●	●	○	○	Off	Off	All packets received by the packet monitor but the data sanity failed
●	●	○	○	○	○	●	●	Off	ON	User restarted the LBUS transaction

X14007-111918

Figure 5-23: **Board Validation for Duplex/Simplex RX Configuration - Failing Scenario**

Validation — Simplex TX Mode

Figure 5-24 and Figure 5-25 describes the LED behavior and input switch condition for the validation of the Interlaken core onboard for Simplex TX mode configuration.

Validation — Passing Scenario For Simplex TX Mode

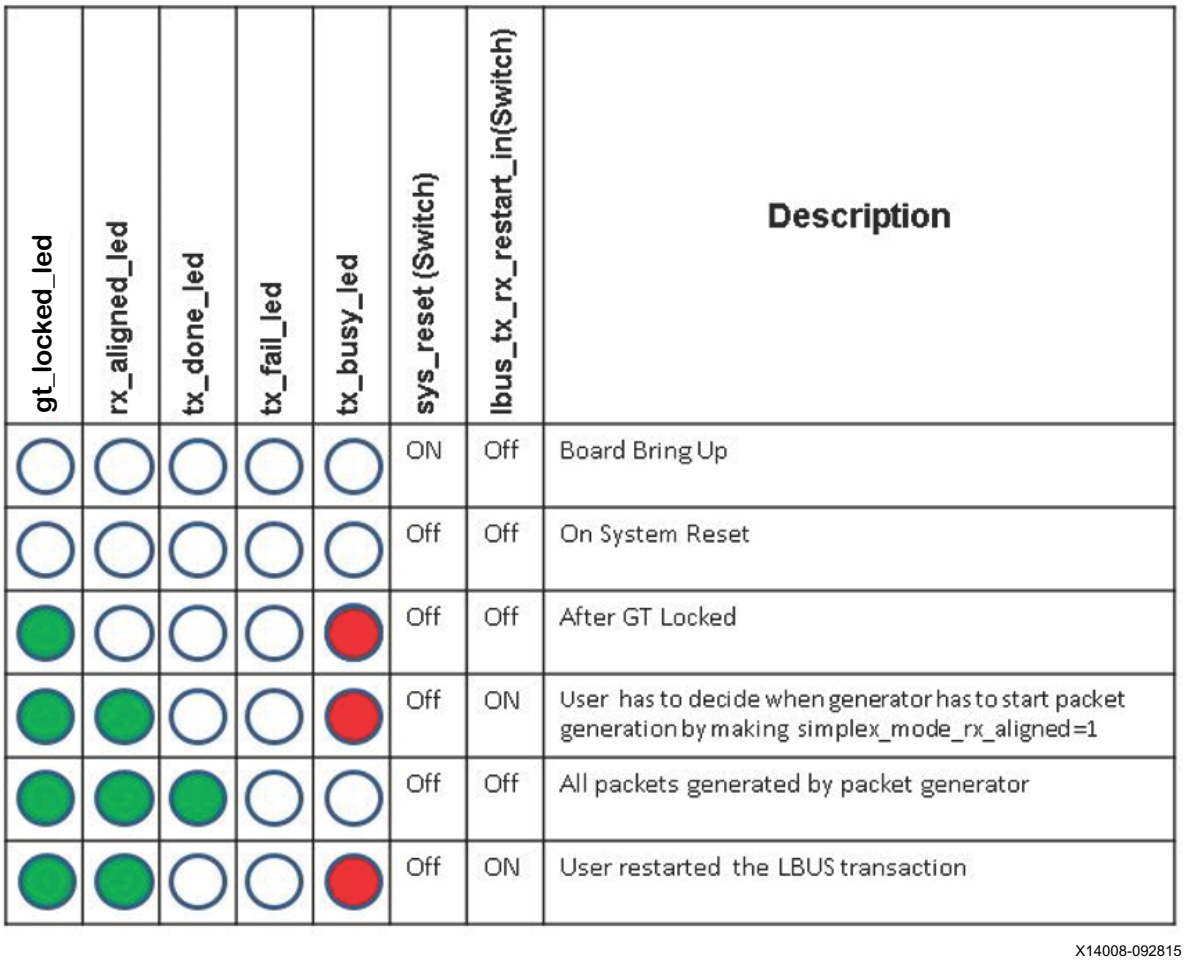


Figure 5-24: **Board Validation for Simplex TX configuration - Passing Scenario**

Validation — Failing Scenario For Simplex TX Mode

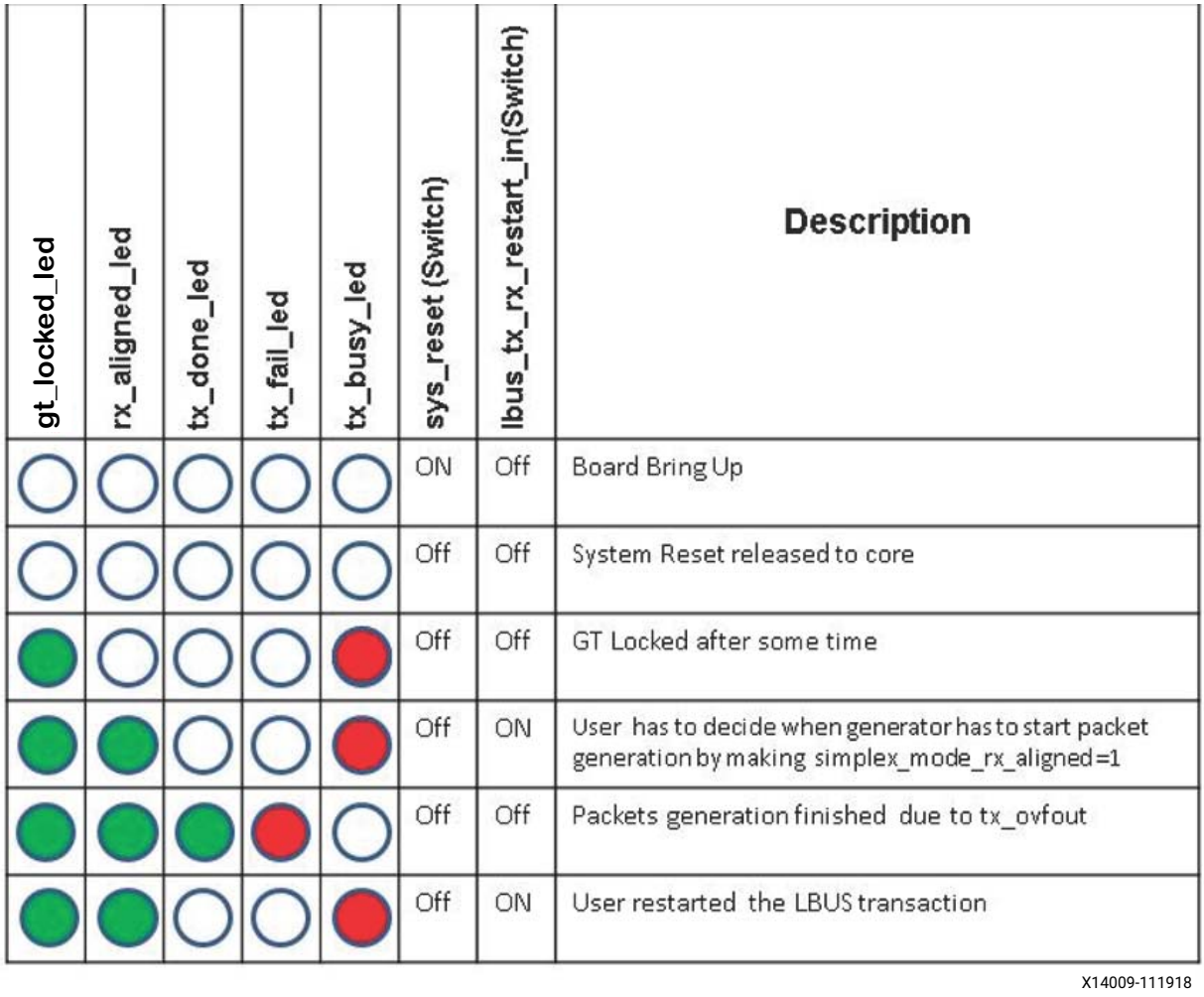


Figure 5-25: **Board Validation for Simplex TX Configuration - Failing Scenario**

Simulating the Example Design

The example design provides a quick way to simulate and observe the behavior of the Interlaken core example design projects generated using the Vivado Design Suite.

The currently supported simulators are:

- Vivado simulator (default)
- Mentor Graphics Questa Advanced Simulator (integrated in the Vivado IDE)
- Cadence Incisive Enterprise Simulator (IES)
- Synopsys VCS and VCS MX

The simulator uses the example design test bench and test cases provided along with the example design.

For any project (Interlaken core) generated out of the box, the simulations can be run as follows:

1. In the Sources Window, right-click the example project file (.xci), and select **Open IP Example Design**. The example project is created.
2. In the Flow Navigator (left-hand pane), under Simulation, right-click **Run Simulation** and select **Run Behavioral Simulation**.

   Note: The post-synthesis and post-implementation simulation options are not supported for the Interlaken core.

   After the Run Behavioral Simulation Option is running, you can observe the compilation and elaboration phase through the activity in the **Tcl Console**, and in the **Simulation** tab of the **Log** Window.
3. In **Tcl Console**, type the run all command and press **Enter**. This runs the complete simulation as per the test case provided in example design test bench.

   After the simulation is complete, the result can be viewed in the **Tcl Console**.

To change the simulators:

1. In the Flow Navigator, under Simulation, select **Simulation Settings**.
2. In the Project Settings for Simulation dialog box, change the Target Simulator to **QuestaSim/ModelSim**.
3. When prompted, click **Yes** to change and then run the simulator.

Synthesizing and Implementing the Example Design

To run synthesis and implementation on the example design in the Vivado Design Suite:

1. Go to the XCI file, right-click, and select **Open IP Example Design**.

   A new Vivado tool window opens with the project name "example_project" within the project directory.
2. In the Flow Navigator, click **Run Synthesis** and **Run Implementation**.

TIP: *Click **Run Implementation** first to run both synthesis and implementation. Click **Generate Bitstream** to run synthesis, implementation, and then bitstream.*

Appendix A

Out-of-Band Flow Control

Interlaken is a scalable chip-to-chip interconnect protocol designed to enable transmission speeds from 10 Gb/s to 100 Gb/s and beyond. Using the latest SerDes technology and a flexible protocol layer, Interlaken minimizes the pin and power overhead of chip-to-chip interconnect and provides a scalable solution that can be used throughout an entire system. In addition, Interlaken uses two levels of CRC checking and a self-synchronizing data scrambler to ensure data integrity and link robustness.

Interlaken defines two mechanisms for handling flow control across the interface:

- In-band flow control
- Out-of-band flow control

The choice of using in-band or out-of-band mechanisms depends on system considerations that are beyond the scope of this document. For more information or feedback, contact Xilinx technical support.

This appendix describes the out-of-band flow control (OOBFC) implemented with the integrated IP core for Interlaken.

Overview

Interlaken employs a simple and flexible XON/XOFF mechanism to communicate flow control information across the interface. The XON/XOFF information is transmitted for all supported logical channels and provides you with a flexible means for implementing any scheduling algorithm.

Xilinx provides two independent modules for handling out-of-band flow control:

- TX_OOBFC
- RX_OOBFC

The module TX_OOBFC transmits out-of-band flow control information and the RX_OOBFC module receives out-of-band flow control information.

Logically, these modules should be instantiated at the same level of hierarchy as the Interlaken Core and operate on their own to transmit and receive flow control information from the corresponding interface.

Out-of-Band Flow Control is enabled in the **General** tab of the Vivado® IDE. The maximum calendar length (MAX_CAL) supported by the core is also selected there. The maximum calendar length can be set to 32, 64, 128, 256, 512, 1024, or 2048.

Figure A-1 shows a typical instantiation of the Interlaken core and the OOBFC modules. You handle packet scheduling and respond to flow control information; it is represented by the user logic block in this diagram.

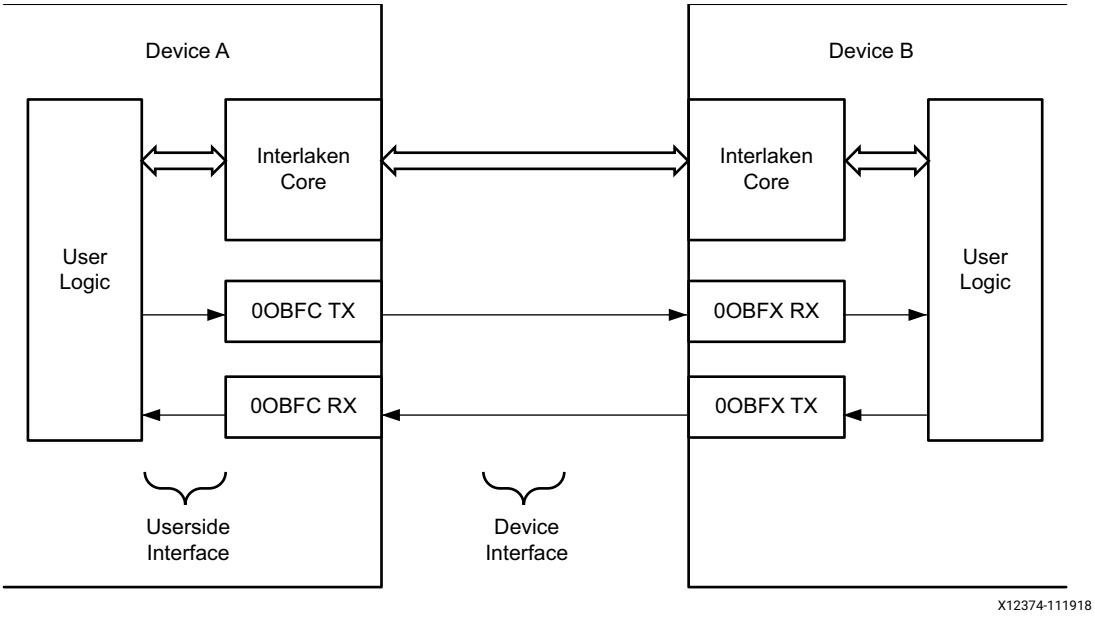


Figure A-1: **Typical Instantiation of the Interlaken and OOBFC Modules**

Port List

Table A-1 and Table A-2 show the port list of the TX_OOBFC and RX_OOBFC modules along with a description of each pin respectively. Note that the MAX_CAL parameter is the maximum number of calendar entries supported by the core, up to a maximum value of 2048.

Table A-1: **TX_OOBFC Pin List**

Name	Direction	Description
Clocking and Resets		
clk	Input	All signals between the TX_OOBFC and the user logic are synchronized to the positive edge of clk. In typical applications, this clock should be tied to the same clock that is used to run the user-side interface of the Interlaken core.
clk_tx_ref	Input	This clock is used to generate the flow control signals. You are required to supply a clock with a maximum frequency of 200 MHz. The 200 MHz limitation (which is twice the OOBFC clock) is defined by the Interlaken Specification, revision 1.2.
reset	Input	Active-High, asynchronous reset input. This signal is automatically synchronized to the appropriate clock domain by the TX_OOBFC. All circuits in the module are reset while this input has a value of 1. This signal must remain asserted until after several clock cycles on both the clk and clk_tx_ref inputs.

Table A-1: **TX_OOBFC Pin List *(Cont'd)***

Name	Direction	Description
User-Side Interface - TX_OOBFC Signals		
tx_fc[MAX_CAL-1:0]	Input	Input data bus containing the flow control information to be transmitted by the TX_OOBFC. The width of the bus is set by the MAX_CAL parameter. Unused bits, as defined by tx_callen_minus1[7:0], must be tied to 0. Interlaken defines a value of 1 as XON for the corresponding channel, and a value of 0 as XOFF for the corresponding channel. This bus is synchronized to the positive edge of clk.
tx_callen_minus1[10|9|8|7:0]	Input	This input sets the calendar length for flow control information for the TX_OOBFC. Allowed values are 0 to MAX_CAL-1. For example, when MAX_CAL is set to 32, tx_callen_minus1[7:0] can be set to any value between 0 and 31. It is up to you to ensure that this input is set correctly. Incorrect setting results in undetermined behavior. This input should only be changed when reset is asserted (that is, set to 1). Changing the value on this input when not in reset can result in incorrectly transmitted flow-control or status information. This bus is synchronized to the positive edge of clk.
tx_lanes_minus1[3:0]	Input	This input sets the number of lanes to be included in the Status Message. Allowed values are 0 to 11 (decimal). This value should be set to the number of lanes (minus 1) expected by the receiver. An incorrect setting can result in undetermined behavior. This input should only be changed when reset is asserted (that is, set to 1). This bus is synchronized to the positive edge of clk.
tx_intf_status	Input	Indicates the health of the interface to be transmitted to the other device as described in the Interlaken specification. A value of 1 indicates the interface is healthy. If unused, this input should be a set to a value of 1. This signal is synchronized to the positive edge of clk.
tx_lane_status[11:0]	Input	Indicates the health of each lane to be transmitted to the other device as described in the Interlaken specification. A value of 1 indicates the corresponding lane is healthy. Bit 0 corresponds to lane 0, bit 1 corresponds to lane 1, and so on. If unused, all inputs should be a set to a value of 1. This bus is synchronized to the positive edge of clk.
tx_err	Output	In the event that the clock rates are incorrect for the selected calendar length, the signal tx_err can be asserted to notify the user of this configuration error.

Table A-1: **TX_OOBFC Pin List *(Cont'd)***

Name	Direction	Description
tx_status_enable	Input	This input can be used to enable and disable the transmission of the interface and lane status. When this input is a value of 1, tx_intf_status and tx_lane_status operate according to the descriptions. When this input is a value of 0, the values on tx_intf_status and tx_lane_status are ignored, and the interface and lane status are not transmitted. If the transmission of the interface and lane status are not required, this value should be tied to a value of 0.
tx_update[31|15|7|3|1|0:0]	Output	This output bus indicates when status and flow control information are latched for transmission. If tx_update[0] is asserted, tx_fc[63:0], tx_lane_status, and tx_intf_status bits will be latched at the completion of the current clock cycle; if tx_update[1] is asserted, tx_fc[127:64] will be latched at the end of the current clock cycle; if tx_update[2] is asserted, tx_fc[191:128] will be latched at the end of the current clock cycle and so on. This output bus is synchronized to the edge of clk.
Device Interface - TX_OOBFC Signals(1)		
TX_FC_CLK	Output (LVCMOS)	This is the source synchronous clock generated by TX_OOBFC as defined by the Interlaken protocol. The frequency of this clock is one-half the frequency of clk_tx_ref. For example, if clk_tx_ref is 200 MHz, this clock is 100 MHz. This signal must be connected directly to a device output pin.
TX_FC_DATA	Output (LVCMOS)	The flow control information is transmitted by the TX_OOBFC with this signal as defined by the Interlaken protocol. This signal must be connected directly to a device output pin.
TX_FC_SYNC	Output (LVCMOS)	This signal is used to synchronize the transmitted flow control information as defined by the Interlaken Protocol. This signal must be connected directly to a device output pin.

Notes:

1. For further details, the electrical and timing specifications of these signals, see the Interlaken Protocol Definition, Revision 1.2 [Ref 1].

Table A-2: **RX_OOBFC Pin List**

Name	Direction	Description
Clocking and Resets		
clk	Input	All signals between the RX_OOBFC and the user logic are synchronized to the positive edge of clk. In typical applications, this clock should be tied to the same clock used to run the user-side interface of the Interlaken Core. In order for correct operation in the RX_OOBFC, the frequency of this clock must be at least 33% faster than the frequency of RX_FC_CLK. For example, if the frequency of RX_FC_CLK is 100 MHz, the frequency of clk must be at least 133 MHz.
reset	Input	Active-High, asynchronous reset input. This signal is automatically synchronized to the appropriate clock domain by the RX_OOBFC. All circuits in the module are reset while this input is a value of 1. This signal must remain asserted until after several clock cycles on both the clk and RX_FC_CLK inputs.
User-Side Interface - RX_OOBFC Signals		
rx_fc[MAX_CAL-1:0]	Output	Output data bus to the user logic containing the flow control information received by the RX_OOBFC. The width of the bus is set by the MAX_CAL parameter. Bit 0 corresponds to channel 0, bit 1 corresponds to channel 1, and so on. Interlaken defines a value of 1 as XON and a value of 0 as XOFF. If the calendar length for the received flow control information has less than MAX_CAL entries, the unused bits are undefined and should be ignored. It is up to you to monitor this bus and take appropriate action as flow control information is changed. When an unhealthy interface status is received, as indicated by rx_intf_status being a value of 0, or an unhealthy lane status is received, as indicated by a bit of rx_lane_status being a value of 0, then all bits of rx_fc are XOFF or a value of 0. When a CRC error is detected, the outputs rx_fc are unchanged. This bus is synchronized to the positive edge of clk.
rx_crcerr	Output	Indicates if an error was observed in the CRC field of the incoming status or flow control information. A value of 1 indicates a CRC error was detected. When this bit is a value of 1, the outputs rx_fc, rx_intf_status, and rx_lane_status are not updated and can be ignored. The cause of a CRC error can be due to clk not being sufficiently faster than RX_FC_CLK as required for correct operation. This signal is synchronized to the positive edge of clk.
rx_overflow	Output	Indicates that clk is not faster than RX_FC_CLK as required for correct operation. This signal is synchronized to the positive edge of clk.
rx_intf_status	Output	Indicates the health of the receive interface as described in the Interlaken specification. When this output has a value of 0, all flow-control bits in the rx_fc bus will be XOFF (that is, 0). This signal is synchronized to the positive edge of clk.

Table A-2: **RX_OOBFC Pin List *(Cont'd)***

Name	Direction	Description
rx_lane_status[11:0]	Output	Indicates the health of the corresponding receive lanes as described in the Interlaken specification. Bit 0 corresponds to lane 0, bit 1 corresponds to lane 1, and so on. When any bit of this bus has a value of 0, all flow-control bits in the rx_fc bus will be XOFF (that is, 0). This bus is synchronized to the positive edge of clk.
rx_update [31|15|7|3|1|0:0]	Output	Indicates a valid CRC4 was detected and rx_fc bits were updated. If rx_update[0] is asserted, the CRC4 associated with rx_fc[63:0] was valid and rx_fc[63:0] were updated; if rx_update[1] is asserted, the CRC4 associated with rx_fc[127:64] was valid and rx_fc[127:64] were updated, and so on. This bus is synchronized to the positive edge of clk.
rx_callen_minus1[10|9|8|7:0]	Output	Indicates the length of the most recently received calendar. This bus is synchronized to the positive edge of clk.
rx_force_xoff_if_crcerr	Input	When this input is a value of 1, all bits of rx_fc are forced to XOFF if a CRC error is detected.
Device Interface - RX_OOBFC Signals(1)		
RX_FC_CLK	Input (LVCMOS)	This the source-synchronous clock used by RX_OOBFC as defined by the Interlaken protocol. This signal must be connected directly to a device input pin.
RX_FC_DATA	Input (LVCMOS)	The flow control information is received by the RX_OOBFC with this signal as defined by the Interlaken protocol. This signal must be connected directly to a device input pin.
RX_FC_SYNC	Input (LVCMOS)	This signal is used to synchronize the received flow control information as defined by the Interlaken Protocol. This signal must be connected directly to a device input pin.

Notes:

1. Further details of the electrical and timing specifications for these signals are found in the Interlaken Protocol Definition, Revision 1.2 [Ref 1].

General Operation

The OOBFC implements the protocol as described in the Interlaken Protocol Revision 1.2 document.

There are 3 line signals: **clock**, **data**, and **sync**. Data is double rate (transfers on the rising and falling edges of the clock). The maximum clock frequency is 100 MHz.

Each bit can represent on/off for one channel or any other mapping you select. A 4-bit CRC is added for every 64 data bits.

Figure A-2 shows the relative timing between the signals.

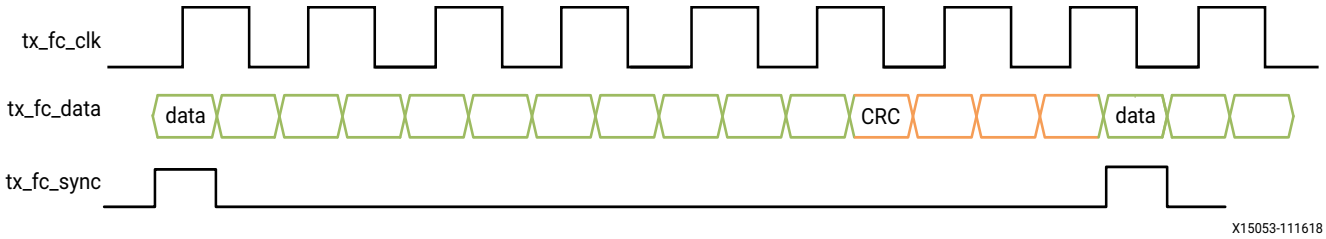


Figure A-2: **OOBFC Signals**

The following sections describe the operation in more detail and are intended to augment the Interlaken specification.

TX_OOBFC

The TX portion of the OOBFC starts transmitting information as soon as the **reset** input is deasserted. The **clk_tx_ref** input must be stable and running correctly before the **reset** input is deasserted.

The health status inputs (**tx_intf_status** and **tx_lane_status**) are transmitted only when an unhealthy condition exists. An unhealthy condition occurs when **tx_intf_status** is 0, or any bit of **tx_lane_status** is 0. The transmission of the status inputs is alternated with the transmission of the flow control information when an unhealthy status exists.

If the health status inputs (**tx_intf_status** and **tx_lane_status**) are all 1, meaning the interface is healthy, only flow control information is transmitted.

The length of the flow control calendar is programmed through **tx_callen_minus1**. The calendar is transmitted with the value of **tx_fc[0]** sent first, **tx_fc[1]** sent second, **tx_fc[2]** sent third and so on.

The **tx_fc** input is latched for transmission in 64-bit groups. The input **tx_fc[63:0]**, **tx_intf_status**, and **tx_lane_status** are latched at the completion of the clock cycle when **tx_update[0]** is asserted; the **input tx_fc[127:64]** is latched at the completion of the clock cycle when **tx_update[1]** is asserted; the **input tx_fc[191:128]** is latched at the completion of the clock when **tx_update[2]** is asserted and so on.

RX_OOBFC

For correct operation of the RX_OOBFC to occur, two things are essential:

- The user-side clock, **clk**, must be faster than the receiving clock, **RX_FC_CLK**, by at least 33%. For example, if the frequency of **RX_FC_CLK** is 100 MHz, then the frequency of **clk** must be at least 133 MHz.
- The reset input, **reset**, must be asserted until after several cycles of both input clocks, **clk** and **RX_FC_CLK**.

Reception of flow control information starts as soon as the **reset** input is deasserted. However, the RX_OOBFC does not consider itself initialized and does not supply the received information until it has seen several correct transitions on the **RX_FC_SYNC** input. During this initialization procedure, the **rx_fc** output bus is set to XOFF (that is, 0). After the initialization procedure is completed, the RX_OOBFC supplies status and flow-control information as received through the interface.

The Interlaken Protocol does not require the transmission of status information if the interface is healthy. As a result, RX_OOBFC assumes the interface is healthy if it receives two back-to-back valid calendars of flow control information. In healthy conditions, the **rx_intf_status** and **rx_lane_status** outputs are all set to 1.

Additionally, in healthy conditions, the received flow control information is presented on the **rx_fc** output bus. The first calendar value received is presented on **rx_fc[0]**, the second on **rx_fc[1]**, the third on **rx_fc[2]**, and so on. If the received calendar has fewer entries than the total width of **rx_fc** bus, the "unreceived" bits in **rx_fc** will be undetermined and must be ignored by the user logic.

Whenever unhealthy status information is received, all the bits of **rx_fc** are forced to XOFF (that is, 0). When healthy status information is determined, **rx_fc** is updated to reflect the most recently received values of flow control information.

If a CRC4 error is ever detected during the receipt of a calendar of flow control information or during the receipt of status information, the **rx_crcerr** output is asserted (set to 1). Whenever **rx_crcerr** has a value of 1, the **rx_fc**, **rx_intf_status**, and **rx_lane_status** outputs are not updated and should be considered invalid (the Interlaken Protocol requires you to take appropriate action when **rx_crcerr** is asserted). The **rx_fc**, **rx_intf_status**, and **rx_lane_status** outputs should be considered valid only when **rx_crcerr** is negated (that is, 0).

The **rx_overflow** output is an aid to help identify incorrect clock rates. It is asserted (set to 1) when the **clk** input is not fast enough relative to **RX_FC_CLK**.

The RX_OOBFC accepts any calendar length up to MAX_CAL. The most recently received calendar length is reported on the output **rx_callen_minus1**.

The **rx_fc** output is updated in groups of 64-bits. When the first group is updated, **rx_update[0]** is asserted. Then the second group of 64-bits is updated, **rx_update[1]** is asserted, and so forth.

Appendix B

UltraScale to UltraScale+ FPGA Enhancements

The UltraScale+™ Integrated Interlaken IP core is derived from UltraScale™ Integrated Interlaken IP core with a few enhancements and minor modifications, as documented in this appendix.

Feature Enhancements in the UltraScale+ Integrated Interlaken IP

- The maximum rate of the protocol bypass interface (lane logic only) in the UltraScale+ Integrated Interlaken IP is increased to 300Gb/s (12 x 25.78125 Gb/s).

  Note: The protocol logic of the Integrated Interlaken IP remains the same and has a maximum rate of 150 Gb/s (12 x 12.5 Gb/s or 6 x 25.78125 Gb/s configurations).

- Registers were added on the input for better timing of attributes, TX LBUS, Retransmission, TX/RX bypass and RX SerDes.
- The TX retransmission logic is enhanced to allow up to two stages of pipeline in the fabric on the write path to retransmission block RAM.

Modifications

DRP addresses are different between the architectures. See Table 3-6 and Table 3-7 for more information on the DRP address maps for the UltraScale and UltraScale+ devices.

Upgrading

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

Migrating to the Vivado Design Suite

This section does not apply to this core.

Upgrading in the Vivado Design Suite

This section provides information about any updates to the user logic or port designations that take place when you upgrade to a more current version of this IP core.

Changes from v2.3 to v2.4

Register Changes

Updated the following registers in Table 5-7.

- Updated STAT_RX_DESCRAM_ERR_REG address to 0x0224.
- Removed register address 0x0228.
- Updated Reserved register address to 0x0228 - 0x02AF that was 0x022C - 0x02AF.
- Updated Reserved register address to 0x3888 to 0x3E4 that was 0x0388 – 0x07FF

Registers Added

Added the following registers to the end of Table 5-7.

- STAT_RX_FRAMING_ERR_LANE0_LSB: 0x3888
- STAT_RX_FRAMING_ERR_LANE0_MSB: 0x038C
- STAT_RX_FRAMING_ERR_LANE1_LSB: 0x0390

- STAT_RX_FRAMING_ERR_LANE1_MSB: 0x0394
- STAT_RX_FRAMING_ERR_LANE2_LSB: 0x0398
- STAT_RX_FRAMING_ERR_LANE2_MSB: 0x039C
- STAT_RX_FRAMING_ERR_LANE3_LSB: 0x03A0
- STAT_RX_FRAMING_ERR_LANE3_MSB: 0x03A4
- STAT_RX_FRAMING_ERR_LANE4_LSB: 0x03A8
- STAT_RX_FRAMING_ERR_LANE4_MSB: 0x03AC
- STAT_RX_FRAMING_ERR_LANE5_LSB: 0x03B0
- STAT_RX_FRAMING_ERR_LANE5_MSB: 0x03B4
- STAT_RX_FRAMING_ERR_LANE6_LSB: 0x03B8
- STAT_RX_FRAMING_ERR_LANE6_MSB: 0x03BC
- STAT_RX_FRAMING_ERR_LANE7_LSB: 0x03C0
- STAT_RX_FRAMING_ERR_LANE7_MSB: 0x03C4
- STAT_RX_FRAMING_ERR_LANE8_LSB: 0x03C8
- STAT_RX_FRAMING_ERR_LANE8_MSB: 0x03CC
- STAT_RX_FRAMING_ERR_LANE9_LSB: 0x03D0
- STAT_RX_FRAMING_ERR_LANE9_MSB: 0x03D4
- STAT_RX_FRAMING_ERR_LANE10_LSB: 0x03D8
- STAT_RX_FRAMING_ERR_LANE10_MSB: 0x03DC
- STAT_RX_FRAMING_ERR_LANE11_LSB: 0x03E0
- STAT_RX_FRAMING_ERR_LANE11_MSB: 0x03E4

Changes from v2.2 to v2.3

Added VCU118 IP Integrator board support.

Changes from v2.1 to v2.2

Port Changes

Added the **gtpowergood_out** and **gt_refclk_out** ports to Table 5-3.

Attribute Changes

- Updated Attribute Encoding and DRP Encoding for CTL_TX_LAST_LANE[3:0] and CTL_RX_LAST_LANE[3:0] in Table 3-6.
- Updated DRP Address, Attribute Encoding and DRP Encoding for CTL_TX_LAST_LANE[3:0] and CTL_RX_LAST_LANE[3:0] in Table 3-7.

Port Changes from v2.0 to v2.1

Removed **tx_gt_locked_led** from Table 5-2.

Port Changes from v1.10 to v2.0

Added six new ports to Table 5-3: **txdata_in**, **rxdata_out**, **tx_clk**, **rx_clk**. **axi_gt_reset_all**, and **axi_gt_loopback** for the **Include GT subcore in example design** option.

Port Changes from v1.9 to v1.10

The Protocol Bypass (Lane Logic Only) Mode feature is now available in the wizard and thus the ports and attributes related to Protocol Bypass (Lane Logic Only) Mode are added. Also the GT DRP Done port is added.

Port Changes from v1.8 to v1.9

The retransmission feature is now available in the wizard and thus the ports and attributes on the retransmission interface are no longer reserved.

Debugging

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

Finding Help on Xilinx.com

To help in the design and debug process when using the integrated IP core for Interlaken, 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 integrated IP core for Interlaken.

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 integrated IP core for Interlaken

AR: 58697

Technical Support

Xilinx provides technical support in 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.

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 allows 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 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 13].

Simulation Debug

Slow Simulation

Simulations might appear to run slowly under some circumstances. If a simulation is unacceptably slow, the following suggestions might improve the run time performance.

1. Use a faster computer with more memory.
2. Make use of a Platform LSF (Load Sharing Facility) if available in your organization.
3. Bypass the Xilinx transceiver (this might require that you create your own test bench)

4. Send fewer packets.
5. Specify a shorter metaframe length. This should result in a shorter lane alignment phase, at the expense of more overhead. However, when the Interlaken IP core is finally implemented in a system, the metaframe length should follow the specification recommendations. This can be done by setting SIM_SPEED_UP, see Simulation Speed Up in Chapter 4.

Simulation Fails Before Completion

If the sample simulation fails or hangs before successfully completing, it is possible that a timeout has occurred. Ensure that the simulator timeouts are long enough to accommodate the waiting periods in the simulation, for example, during the lane alignment phase.

Simulation Completes But Fails

In the event that the sample simulation completes with a failure, contact Xilinx technical support. Tests normally complete successfully. Consult the sample simulation log file for the expected behavior.

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 Hardware Manager is a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the Hardware Manager for debugging the specific problems.

Many of these common issues can also be applied to debugging design simulations, as described in the following sections.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the **LOCKED** port.

Interlaken Specific Checks

Many issues can commonly occur during the first hardware test of an integrated IP core for Interlaken. These should be checked as indicated below.

The usual sequence of debugging is to proceed in the following sequence:

1. Clean up signal integrity.
2. Ensure that each SerDes achieves CDR lock.
3. Check that each lane has achieved word alignment.
4. Check that lane alignment has been achieved.
5. Proceed to Interface Debug and Protocol Debug.

Signal Integrity

When bringing up a board for the first time, if the integrated IP core for Interlaken does not seem to be achieving lane alignment, the most likely problem is related to signal integrity. Signal integrity issues must be addressed before any other debugging can take place.

Even if lane alignment is achieved, if there are periodic CRC32 errors, then signal integrity issues are indicated. Check the **`stat_rx_crc32_err`** signals to assist with debug.

Signal integrity should be debugged independently from the integrated IP core for Interlaken. The following procedures should be carried out:

- Transceiver Settings
- Checking For Noise
- Bit Error Rate Testing

Note: It assumed that the PCB itself has been designed and manufactured in accordance with the required trace impedances and trace lengths.)

If assistance is required for transceiver and signal integrity debugging, contact Xilinx technical support.

Lane Swapping

Unlike Ethernet, Interlaken requires that TX and RX lanes maintain the correct ordering.

IMPORTANT: *Lane alignment can still be achieved even if lanes are swapped.*

The reason is that the alignment marker is the same for each lane. Furthermore, even with lanes swapped, the CRC32 might be correct provided that signal integrity is good.

Lane swapping is often indicated by a CRC24 error while data is being sent.

N/P Swapping

If the positive and negative signals of a differential pair are swapped, no data will be correctly received on that lane. You should verify that each link has the correct polarity of each differential pair.

Clocking and Resets

See Clocking and Resets in Chapter 3 for these requirements.

Ensure that the clock frequencies for both the integrated IP core for Interlaken and the Xilinx transceiver reference clock are set correctly according to the configured mode of the core. Typical frequencies for each Interlaken configuration mode are listed in Table 3-1.

The first thing to verify during debugging is to ensure that resets remain asserted until the clock is stable. It must be frequency-stable and free from glitches before the integrated IP core for Interlaken is taken out of reset. This applies to both the SerDes clock and the IP core clock.

If any subsequent instability is detected in a clock, the integrated IP core for Interlaken must be reset. One example of such instability is a loss of CDR lock. The user logic should determine all external conditions which would require a reset, for example, clock glitches, loss of CDR lock, power supply glitches, etc. The GT requires a GTRXRESET after the serial data becomes valid to ensure correct CDR lock to the data. This is required after powering on, resetting or reconnecting the link partner. At the core level to avoid interruption on the TX side of the link, the reset can be triggered using **`gtwiz_reset_rx_datapath`**.

Configuration changes cannot be made unless the IP core is reset. An example of a configuration change is a change in BurstMax. Check the description for the particular signal on the port list to determine if this requirement applies to the parameter that is being changed.

Interface Debug

LBUS Interface

The integrated IP core for Interlaken user interface is called the segmented LBUS (Local bus). For proper operation of the Interlaken interface, it is important that the segmented LBUS protocol is followed correctly.

Note: The segmented LBUS requires the use of the enhanced scheduling algorithm.

TX Debug

TX debug is assisted by means of several diagnostic signals.

Buffer Errors

Data must be written to the TX LBUS such that there are no overflow or underflow conditions. LBUS bandwidth must always be greater than the Interlaken bandwidth in order to guarantee that bursts can be sent without interruption.

When writing data to the LBUS, the **tx_rdyout** signal must always be observed. This signal indicates whether the fill level of the TX buffer is within an acceptable range or not. If this signal is ever deasserted, you must stop writing to the TX LBUS until the signal is asserted. Since the TX LBUS has greater bandwidth than the TX Interlaken interface, it is not unusual to see this signal being frequently asserted and this is not a cause for concern.

IMPORTANT: *You must ensure that TX writes are stopped when `tx_rdyout` is deasserted.*

If **tx_rdyout** is ignored, the signal **tx_ovfout** might be asserted, indicating a buffer overflow. This must not be allowed to occur.

RECOMMENDED: *It is recommended that the Interlaken IP core be reset if `tx_ovfout` is ever asserted.*

Do not attempt to continue debugging after **tx_ovfout** is asserted until the cause of the overflow has been addressed.

If **stat_tx_underflow_err** is ever asserted, debugging must stop until the condition which caused the underflow is addressed. This can happen if the core clock is not fast enough to supply the transceiver with data, and you should ensure that the minimum core clock frequency is being observed.

Burst Errors

The TX must observe the LBUS rules which are required for compliance to the *Interlaken Protocol Definition, Revision 1.2* [Ref 1] and the Xilinx Integrated IP core for Interlaken requirements.

If a burst rule violation occurs, the **stat_tx_burst_err** signal is asserted. You must ensure that the rules governing the scheduling of bursts are observed. In particular, the segmented LBUS protocol should be given careful reading.

One common issue is that you have written data in such a way that more than one Burst Control Word is generated during one clock cycle. Carefully review the segmented LBUS rules. Take care that an SOP or a channel change does not occur in the same cycle as an implied Burst Control Word resulting from BurstMax having been reached.

This can be avoided by implementing the enhanced scheduling algorithm, which uses the **tx_bctlin<N>** signals to force Burst Control Words in such a way as to avoid two in one clock cycle.

In summary, Burst Control Words can be implied or generated by the following cases:

- BurstMax reached (implied)
- EOP/SOP (implied)
- Channel Change (implied)
- bctlin assertion (forced)

IMPORTANT: *You must ensure that there is only one Burst Control Word (implied or forced) during a given cycle.*

RECOMMENDED: *Xilinx strongly recommends that you implement Enhanced Scheduling, as described in Enhanced Scheduling, page 62 and Interlaken Protocol Definition [Ref 1], as a means to avoid some of the implied Burst Control Words so that two BCWs in one clock cycle do not occur.*

RX Debug

The RX User Side interface indicates an error condition using the **rx_errout<N>** signal.

If **rx_errout<N>** is asserted, it usually indicates that the packet being received contains an error. One possible error condition that must be examined carefully is a missing SOP or EOP. In this event, the integrated IP core for Interlaken attempts to recover the data by merging packets and closing them, presenting the data on the RX LBUS with an **rx_errout<N>** indication. You should ensure that the transmitting device is sending proper SOP and EOP indications.

Ensure that the **stat_rx_overflow_err** signal is not asserted. If it is asserted, the RX LBUS is not being clocked fast enough to empty the RX buffer. The LBUS must have more bandwidth than the Interlaken interface.

Protocol Debug

To achieve error-free data transfers with the Iintegrated IP core for Interlaken, the protocol parameters need to be set correctly. This section details some common protocol problems which might occur. It assumed that the number of lanes and the bit rates are matched. It is also assumed that the signal integrity and lane ordering has been verified.

Configuration Match

In order for the Interlaken protocol to function correctly, both devices must have matching Interlaken Protocol parameters. Ensure that the following parameters are the same for each end of the link:

- Metaframe Length
- BurstMax
- BurstShort

Performance

In the event that performance (throughput) does not match expectations, ensure that the LBUS protocol is being properly followed. For greatest efficiency, all bursts should follow the requirements of the enhanced scheduling algorithm. Ensure that all bursts except the last two for a packet are BurstMax.

Diagnostic Signals

There are many error indicators available to check for protocol violations. Carefully read the description of each one to see if it is useful for a particular debugging issue.

The following is a suggested debug sequence:

1. Ensure that Word sync has been achieved.
2. Ensure that Lane sync has been achieved.
3. Verify that the Metaframe indicators are clean.
4. Make sure there are no descrambler state errors.
5. Eliminate CRC24 errors, if any.
6. Make sure there are no burst errors (BurstMax, BurstShort mismatch).
7. Look for SOP and EOP errors and eliminate those, if any occur.

Appendix E

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

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

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

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

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

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

References

These documents provide supplemental material useful with this product guide:

1. *Interlaken Protocol Definition* (Revision 1.2, October 7, 2008)
2. *Interlaken Retransmit extension Protocol Definition,* Revision 1.2, June 28, 2012
3. Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator (UG994)
4. Vivado Design Suite User Guide: Logic Simulation (UG900)
5. *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS893)
6. *UltraScale FPGAs Transceiver Wizards* (PG182)
7. *UltraScale Architecture GTY Transceivers User Guide* (UG578)
8. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
9. *Vivado Design Suite User Guide: Designing with IP* (UG896)
10. *Vivado Design Suite User Guide: Getting Started* (UG910)
11. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
12. *Vivado Design Suite User Guide: Using Constraints* (UG903)
13. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
14. *Vivado Design Suite User Guide: Implementation* (UG904)
15. *UltraScale Architecture GTH Transceivers User Guide* (UG576)
16. *UltraScale Architecture GTY Transceivers User Guide* (UG578)
17. *Virtex UltraScale+ Architecture Data Sheet: DC and AC Switching Characteristics* (DS923)
18. *Kintex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS892)
19. *Kintex UltraScale+ Architecture Data Sheet: DC and AC Switching Characteristics* (DS922)
20. *Interlaken 600G LogiCORE IP Product Guide* (PG209). Only available in a lounge.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
02/04/2021	2.4	• Updated Interface Debug.
12/05/2018	2.4	• Added a note in the CTL_RX_FORCE_RESYNC port description in Table 2-2. • Updated IMPORTANT note in the Example Design Hierarchy section in Chapter 5.
04/04/2018	2.4	• Added IMPORTANT note about sequence multipliers to the Selecting a Sequence Multiplier section in Chapter 3. • Updated Figures 4-1, 4-2, and 4-3. • Added the .h Header File subsection to the AXI4-Lite Interface Implementation section in Chapter 5. • Updated and added many registers n Table 5-7. See Changes from v2.3 to v2.4 in Appendix C for a complete list.
10/04/2017	2.3	• Updated figures in Chapter 4. • Added VCU118 IP Integrator board support. • Added R/LL and R/LH information to the Status and Statistics Register Space section in Chapter 5, Example Design. • Added gtwiz_reset_tx_datapath and gtwiz_reset_rx_datapath ports in Figures 3-1 and 3-2 and text in Reset section.
06/07/2017	2.2	• Updated screen displays in Chapter 4. • Added Simulation Speed Up section to Chapter 4. • For port changes. See Port Changes in Appendix C. • For port changes. See Attribute Changes in Appendix C
04/05/2017	2.1	• Updated Licensing and Ordering Information in Chapter 1. • Updated screen displays in Chapter 4. • Removed tx_gt_locked_led from Table 5-2.

Date	Version	Revision
11/30/2016	2.0	• Removed “up to 150” and “Higher speed” from the IP Facts page Introduction. • Added a bulleted item to the second paragraph of the Transceiver Interface section of Chapter 3. • Updated much of the text in the Protocol Bypass Interface section of Chapter 3. • Modified the first three paragraphs of the Protocol Bypass Interface section in Chapter 3. • Added text to the Important note in the Burst Interleaved Mode section in Chapter 3. • Added Important note to the tx_rdyout section in Chapter 3. • Changed the default value to 1 for Bit 3 in Table 5-15, STAT_TX_STATUS_REG. • Modified the default values for Bit 0 to 1 and Bits 19:8 to 12’hFFF in Table 5-16, STAT_RX_STATUS_REG. • Modified the default values for Bits 11:0 and Bits 23:1‘2 to 12’hFFF in Table 5-17, STAT_RX_DIAGWORD_REG. • Modified the default values for Bits 11:0 to 12’hFFF in Table 5-19, STAT_RX_SYNCED_REG. • Modified the text in the Feature Enhancements in the UltraScale+ Integrated Interlaken IP section of Appendix B. • Added a reference to PG209 in the References section of Appendix E. • Changed “protocol bypass” to “Protocol bypass (lane logic only)” throughout.
10/05/2016	2.0	• Change X signal notation to <N> throughout. • Updated Figures 4-1through 4-3. • Added GT Location section to Table 4-3. • Updated the Licensing and Ordering Information section in Chapter 1. • Added Example Design Hierarchy (GT Subcore in Example Design) section to Chapter 5. Example Design. • Added text to the Retransmission Feature section in Chapter 3. • Updated the description in the Clocking and Resets section in Appendix D: Debugging. • Added six new ports to Table 5-3: txdata_in, rxdata_out, tx_clk, rx_clk. axi_gt_reset_all, and axi_gt_loopback. • Updated the four timing diagrams Figures 5-15 through 5-18. • Added

Date	Version	Revision
06/08/2016	1.10	• Updated the following Figures: 3-1, 3-2, 4-1, 4-2, and 4-3. • Added gt_drp_done signal to Table 5-3. • Added the ports and attributes related to Protocol Bypass Mode.
04/06/2016	1.9	• Added support for UltraScale+ devices. • Added support for retransmission feature. • Added references to DS892, DS922, and DS923 data sheets. • Added DRP_CLK to the Clock Domain column for the DRP Path/Control Signals in Table 2-2. • Changed reserved port descriptions in Table 2-2 to be retransmission interface ports. See Port Changes in Appendix C, Migrating and Updating. • Changed reserved attribute descriptions to be retransmission attributes in Table 2-3. See Attribute Changes in Appendix C, Migrating and Updating. • Updated Figure 3-1, Figure 3-2, Figure 4-1, Figure 4-2, Figure 4-3. • Added the Protocol Bypass Interface and Retransmission Interface sections to Chapter 3. • Added Table 3-7, DRP Address Map of the ILKN block in UltraScale+ Devices • Modified Packet Mode and Burst Interleaved Mode sections in Chapter 3. • Completely BurstMin Requirements, Configuring the TX Retransmission Buffer Depth, Selecting a Sequence Multiplier, and Measuring Request-to-Discontinuity Latency sections. • Completely revised TX Transactions and RX LBUS Interface sections in Chapter 3. • Changed CORE_CLK to LBUS_CLK in CRC32 Diagnostics Checking and Transmitter Multiple-Use Bits sections. • Added rx_mubits_updated description. • Changed CTL_TX_FC_STAT[MAX_CALLEN-1:0] to CTL_TX_FC_STAT[MAX_CALLEN-1:255] • Updated CTL_TX_FC_CALLEN[3:0] description • In Table 4-1, added GT DRP/Init Clock parameter and removed AXI4-Lite Interface section. • Updated most all of Table 4-2, Table 4-3 and Table 5-3. • Added s_axi_pm_tick to Table 5-2. • Added S_AXI_PM_TICK to Table 5-4. • Added some text to Status and Statistics Register Space section. • Updated defaults in Table 5-8 and Table 5-9.

Date	Version	Revision
11/18/2015	1.8	Designing with the Core Chapter: • Added description to examples section that illustrates LBUS cycles. • Updated Figure 3-4. • Added additional description to RX LBUS Interface section. Design Flow Steps Chapter: • Updated Vivado IDE core customization screen captures. • Added Include AXI4-Lite Control and Statistics Interface option to the Configuration Options tab. Out-of-Band Flow Control Appendix: • Changed resetn to reset. • In the TX_OOBFC Pin List table: ◦ Changed tx_lanes_minus1[LANES_LOG2-1:0] to tx_lanes_minus1[3:0]. ◦ Changed tx_lane_status[LANES-1:0] to tx_lane_status[11:0]. ◦ Added tx_status_enable. • In the RX_OOBFC Pin List table: ◦ Changed rx_lane_status[LANES-1:0] to rx_lane_status[11:0]. ◦ Added rx_force_xoff_if_crcerr. ◦ Removed resetn_RX_FC_CLK. Debugging Appendix: • Added information about performance to Protocol Debug section.
09/30/2015	1.7	• Added domain information to port description table. • Updated the Clocking Reset Interface tables for Synchronous Mode and Asynchronous Mode. • Updates to the Vivado IDE core customization options. • Removed User Parameter section content. • Example Design chapter: ◦ Updated the example design hierarchy diagram, and design details. ◦ Added the Core XCI Top-Level Ports section. ◦ Added the AXI4-Lite Interface Implementation section.

Date	Version	Revision
06/24/2015	1.6	• Updated document title. • Moved performance and resource utilization data to www.xilinx.com. • Updated the Clocking Reset Interface tables for Synchronous Mode and Asynchronous Mode. • Updated Vivado IP catalog options/User Parameters for core v1.6: ◦ added new Protocol Bypass Mode and Disable Skipword parameters. ◦ added new Line Rate value. ◦ indicated that Out-of-Band Flow Control is now grayed out. ◦ added Enable Rate Limiter description. ◦ added Max Token Count description, default range. ◦ added Delta Token Count description, default range; updated values. ◦ added Update Interval description, default range. ◦ added Enable Error Injection description. • Updated the Invalid Segment LBUS Cycles table. • Updated the Example Design Hierarchy, and Example Design Hierarchy with Shared Logic Implementation tables. • Updated the Interlaken Support Modes and Configurations table. • Updated Vivado Lab Edition to Vivado Design Suite debug feature.
04/01/2015	1.5	• Updated resource utilization information. • Vivado IDE parameter additions, and updates. • Updated example design details, added new user interface ports, and added Simplex TX and RX mode simulation information. • Updated Vivado lab tools to Vivado Lab Edition.
02/12/2015	1.4	• Updated to v1.4 of the core. • Updated *Typical Clock Frequencies for Each Interlaken Configuration* table with new line rate and refclk rates. • Vivado IDE parameter additions, and updates. • Included information regarding retransmission not being supported in this core version.
10/01/2014	1.3	• Updated to v1.3 of the core. • Added clocking reset diagrams. • Added new parameters: Low Latency Mode, Enable Retransmission, Number of RAM Banks, Buffer Depth, Sequence Multiplier (TX), Sequence Multiplier (RX), Watchdog Timer, Long Timer, Retry Timer, Wrap Around Timer, Lane 00 to Lane 11. • Added values for existing parameters: Line Rate, GT Ref Clk, Channel Topology. • Updated user parameters table. • Updated example design information. • Updated out-of-band flow control information. • Added simulation, hardware, interface, and protocol debug information.

Date	Version	Revision
06/04/2014	1.2	• Updated to v1.2 of the core. • Added out-of-band flow control information. • Added user parameters table.
04/02/2014	1.1	• Added DRP information. • Updated Figures 2-1, 4-1, 4-2, 4-3, 5-1, 5-5, and 5-6. • Added performance and resource material. • Updated transceiver interface rules. • Updated the constraints information. • Added new transceiver_debug and GT_common module text. • Added shared logic implementation section. • Added information about simulating, synthesizing, and implementing the example design.
01/20/2014	1.0	Initial release.

Please Read: Important Legal Notices