UltraScale Architecture Integrated IP Core for Interlaken v1.3

LogiCORE IP Product Guide

Vivado Design Suite

PG169 October 1, 2014

XILINX
ALL PROGRAMMABLE™

Table of Contents

Introduction

The Xilinx® UltraScale™ architecture integrated IP core for Interlaken is a scalable chip-to-chip interconnect protocol designed to enable the following for use in select UltraScale architectures:

- The lane logic only mode allows each serial transceiver to be used to build a fully featured Interlaken interface. In devices with 48 serial transceivers, up to 600 Gb/s of total throughput can be sustained.
- The protocol logic supported in each integrated IP core scales up to 150 Gb/s.

The Interlaken integrated 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
- 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
Supported User Interfaces	Segmented LBUS
Resources	Table 2-2
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	
Provided by Xilinx @ www.xilinx.com/support	

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® UltraScale™ architecture 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 the 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.*

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 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
- DRP interface for dynamic reconfiguration of the core
- Lane logic only mode. See IP Facts.

Applications

The UltraScale architecture 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:

- 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 lookaside mode.

Licensing and Ordering Information

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 information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

The UltraScale™ architecture 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

Note: The line rate of 25.78125 Gb/s is available on select devices. (Virtex® UltraScale FPGAs in typical speed grades.)

The UltraScale architecture integrated 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

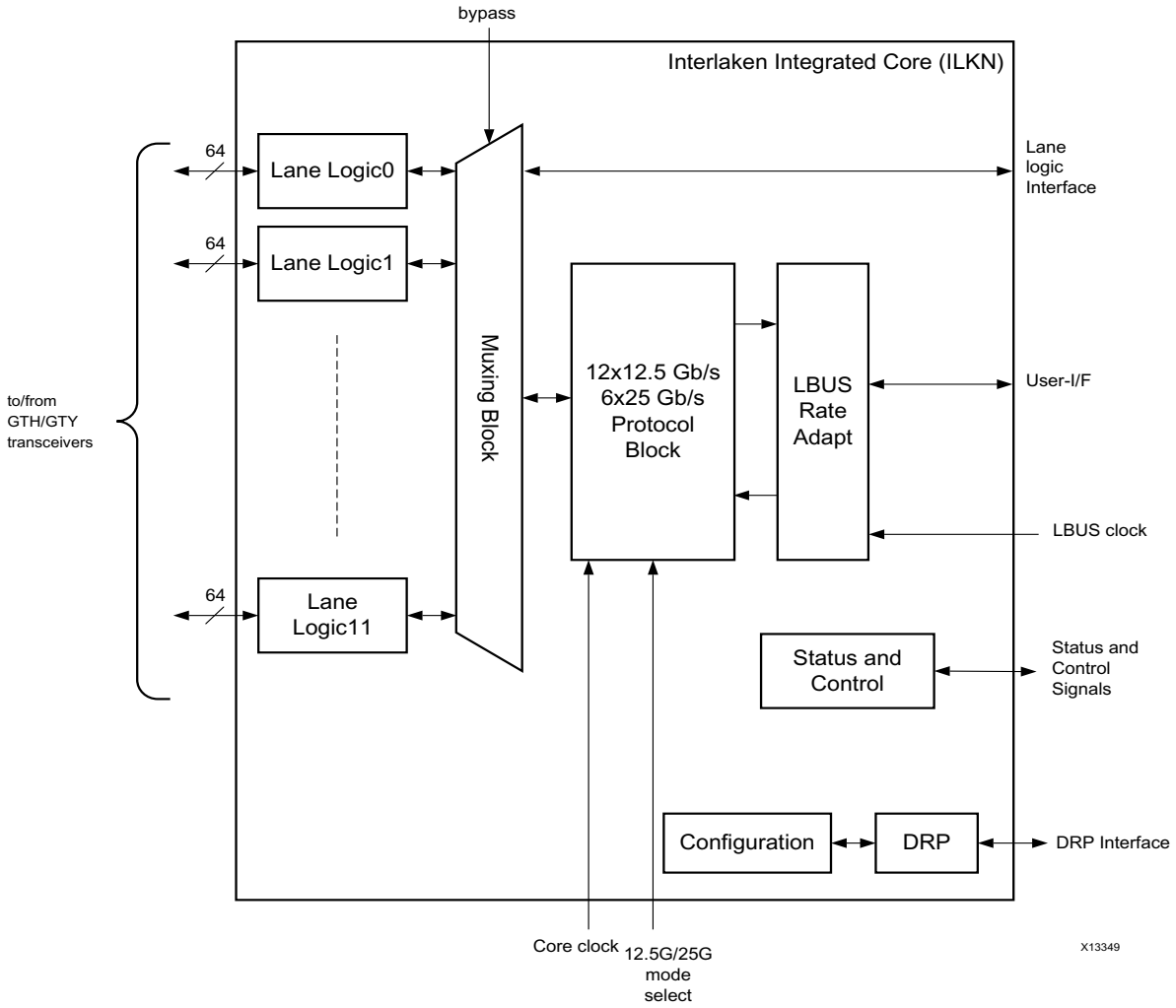

Figure 2-1: **Block Diagram of the Interlaken Integrated IP Core**

Typical Operation

The UltraScale architecture 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.

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 is eventually aligned and synchronized and signals the user logic that synchronization 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].

Performance

The integrated IP core for Interlaken is designed to operate with the performance characteristics of the ILKN primitive it instantiates.

Maximum Frequencies

See the *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS893) [Ref 2] for the maximum frequencies allowed on the Interlaken core specified by speed grade.

IMPORTANT: *A free-running clock input, `init_clk`, is required for the transceiver portion of the integrated IP core for Interlaken. See the UltraScale FPGAs Transceiver Wizards (PG182) [Ref 3] for more information on the `gtwiz_reset_clk_freerun_in` input port.*

Resource Utilization

Resources required for the UltraScale architecture integrated Interlaken core have been estimated for the Virtex UltraScale devices. These values were generated using the Vivado Design Suite. The resources listed in Table 2-2 are for the default core configuration.

Table 2-2: **UltraScale Resource Numbers**

Mode	GT	Only Core (Example Design as Black Box)			
		Flip Flops	LUTs (Logic + Memory)	Clock Resources	Carry8
12 x 12.5	GTH	4844	94 (94 + 0)	BUFGCE = 3 BUFG_GT = 4 BUFG_GT_SYNC = 1 MMCME3_ADV = 1	3
12 x 12.5	GTY	4834	85 (85 + 0)	BUFGCE = 3 BUFG_GT = 4 BUFG_GT_SYNC = 1 MMCME3_ADV = 1	3
6 x 25.78125	GTY	2530	76 (76 + 0)	BUFGCE = 4 BUFG_GT = 4 BUFG_GT_SYNC = 1 MMCME3_ADV = 1	3

Port Descriptions

Table 2-3 describes the UltraScale device Interlaken (ILKN) primitive ports.

Table 2-3: **UltraScale Device Interlaken Primitive Ports**

Name	Direction	Description
Transceiver I/O		
RX_SERDES_DATA0[63:0]	Input	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	Data bus from the serial transceiver macros for lane1.
RX_SERDES_DATA2[63:0]	Input	Data bus from the serial transceiver macros for lane2.
RX_SERDES_DATA3[63:0]	Input	Data bus from the serial transceiver macros for lane3.
RX_SERDES_DATA4[63:0]	Input	Data bus from the serial transceiver macros for lane4.
RX_SERDES_DATA5[63:0]	Input	Data bus from the serial transceiver macros for lane5.
RX_SERDES_DATA6[63:0]	Input	Data bus from the serial transceiver macros for lane6.
RX_SERDES_DATA7[63:0]	Input	Data bus from the serial transceiver macros for lane7.
RX_SERDES_DATA8[63:0]	Input	Data bus from the serial transceiver macros for lane8.
RX_SERDES_DATA9[63:0]	Input	Data bus from the serial transceiver macros for lane9.
RX_SERDES_DATA10[63:0]	Input	Data bus from the serial transceiver macros for lane10.
RX_SERDES_DATA11[63:0]	Input	Data bus from the serial transceiver macros for lane11.
TX_SERDES_DATA0[63:0]	Output	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	Data bus to the serial transceiver macros for lane1.
TX_SERDES_DATA2[63:0]	Output	Data bus to the serial transceiver macros for lane2.
TX_SERDES_DATA3[63:0]	Output	Data bus to the serial transceiver macros for lane3.
TX_SERDES_DATA4[63:0]	Output	Data bus to the serial transceiver macros for lane4.
TX_SERDES_DATA5[63:0]	Output	Data bus to the serial transceiver macros for lane5.
TX_SERDES_DATA6[63:0]	Output	Data bus to the serial transceiver macros for lane6.
TX_SERDES_DATA7[63:0]	Output	Data bus to the serial transceiver macros for lane7.
TX_SERDES_DATA8[63:0]	Output	Data bus to the serial transceiver macros for lane8.
TX_SERDES_DATA9[63:0]	Output	Data bus to the serial transceiver macros for lane9.
TX_SERDES_DATA10[63:0]	Output	Data bus to the serial transceiver macros for lane10.
TX_SERDES_DATA11[63:0]	Output	Data bus to the serial transceiver macros for lane11.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
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.
RX_SERDES_RESET[11:0]	Input	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 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	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 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	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	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.
RX_OVFOUT	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[127:0]	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_DATAOUT1[127:0]	Output	Receive segmented LBUS Data for segment1.
RX_DATAOUT2[127:0]	Output	Receive segmented LBUS Data for segment2.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
RX_DATAOUT3[127:0]	Output	Receive segmented LBUS Data for segment3.
RX_CHANOUT0[10:0]	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_CHANOUT1[10:0]	Output	Receive Channel Number for segment1.
RX_CHANOUT2[10:0]	Output	Receive Channel Number for segment2.
RX_CHANOUT3[10:0]	Output	Receive Channel Number for segment3.
RX_ENAOUT0	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_ENAOUT1	Output	Receive LBUS Enable for segment1.
RX_ENAOUT2	Output	Receive LBUS Enable for segment2.
RX_ENAOUT3	Output	Receive LBUS Enable for segment3.
RX_SOPOUT0	Output	Receive LBUS Start-Of-Packet for segment0. This signal indicates the Start Of Packet (SOP) when it is sampled as a 1 and is only valid in cycles in which RX_ENAOUT is sampled as a 1.
RX_SOPOUT1	Output	Receive LBUS Start-Of-Packet for segment1.
RX_SOPOUT2	Output	Receive LBUS Start-Of-Packet for segment2.
RX_SOPOUT3	Output	Receive LBUS Start-Of-Packet for segment3.
RX_EOPOUT0	Output	Receive LBUS End-Of-Packet 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_ENAOUT is sampled as a 1.
RX_EOPOUT1	Output	Receive LBUS End-Of-Packet for segment1.
RX_EOPOUT2	Output	Receive LBUS End-Of-Packet for segment2.
RX_EOPOUT3	Output	Receive LBUS End-Of-Packet for segment3.
RX_ERROUT0	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_ERROUT1	Output	Receive LBUS Error for segment1.
RX_ERROUT2	Output	Receive LBUS Error for segment2.
RX_ERROUT3	Output	Receive LBUS Error for segment3.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
RX_MTYOUT0[3:0]	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_MTYOUT1[3:0]	Output	Receive LBUS Empty for segment1.
RX_MTYOUT2[3:0]	Output	Receive LBUS Empty for segment2.
RX_MTYOUT3[3:0]	Output	Receive LBUS Empty for segment3.
LBUS Interface – TX Path Signals		
TX_RDYOUT	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	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[127:0]	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_DATAIN1[127:0]	Input	Transmit segmented LBUS Data for segment1.
TX_DATAIN2[127:0]	Input	Transmit segmented LBUS Data for segment2.
TX_DATAIN3[127:0]	Input	Transmit segmented LBUS Data for segment3.
TX_CHANIN0[10:0]	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_CHANIN1[10:0]	Input	Transmit LBUS Channel Number for segment1.
TX_CHANIN2[10:0]	Input	Transmit LBUS Channel Number for segment2.
TX_CHANIN3[10:0]	Input	Transmit LBUS Channel Number for segment3.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
TX_ENAIN0	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_ENAIN1	Input	Transmit LBUS Enable for segment1.
TX_ENAIN2	Input	Transmit LBUS Enable for segment2.
TX_ENAIN3	Input	Transmit LBUS Enable for segment3.
TX_SOPIN0	Input	Transmit LBUS Start Of Packet for segment0. This signal is used to indicate the Start Of Packet (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_SOPIN1	Input	Transmit LBUS Start Of Packet for segment1.
TX_SOPIN2	Input	Transmit LBUS Start Of Packet for segment2.
TX_SOPIN3	Input	Transmit LBUS Start Of Packet for segment3.
TX_EOPIN0	Input	Transmit LBUS End Of Packet for segment0. This signal is used to indicate the End Of Packet (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.
TX_EOPIN1	Input	Transmit LBUS End Of Packet for segment1.
TX_EOPIN2	Input	Transmit LBUS End Of Packet for segment2.
TX_EOPIN3	Input	Transmit LBUS End Of Packet for segment3.
TX_ERRIN0	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_ERRIN1	Input	Transmit LBUS Error for segment1.
TX_ERRIN2	Input	Transmit LBUS Error for segment2.
TX_ERRIN3	Input	Transmit LBUS Error for segment3.
TX_MTYIN0[3:0]	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_MTYIN1[3:0]	Input	Transmit LBUS Empty for segment1.
TX_MTYIN2[3:0]	Input	Transmit LBUS Empty for segment2.
TX_MTYIN3[3:0]	Input	Transmit LBUS Empty for segment3.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
TX_BCTLIN0	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. (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	Transmit force insertion of Burst Control word for segment1.
TX_BCTLIN2	Input	Transmit force insertion of Burst Control word for segment2.
TX_BCTLIN3	Input	Transmit force insertion of Burst Control word for segment3.
LBUS Interface - TX Path Control/Status Signals		
CTL_TX_ENABLE	Input	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	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.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
CTL_TX_MUBITS[7:0]	Input	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.
CTL_TX_RLIM_ENABLE	Input	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	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	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	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	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	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	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	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_TX_BURST_ERR	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.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
LBUS Interface - RX Path Control/Status Signals		
CTL_RX_FORCE_RESYNC	Input	RX Resync input. This signal is used to force the RX path 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.
STAT_RX_BURSTMAX_ERR	Output	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	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].
STAT_RX_DIAGWORD_INTFSTAT[11:0]	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 [Ref 1].
STAT_RX_CRC32_VALID[11:0]	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 [Ref 1].
STAT_RX_CRC32_ERR[11:0]	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.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
STAT_RX_FC_STAT[255:0]	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[7:0]	Output	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	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	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	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	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.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
STAT_RX_DESCRAM_ERR[11:0]	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.
STAT_RX_ALIGNED	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_ALIGNED_ERR	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	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_OVERFLOW_ERR	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.
STAT_RX_MF_ERR[11:0]	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_FRAMING_ERR[11:0]	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_MSOP_ERR	Output	Missing Start-of-Packet (SOP) Error. This signal indicates that a Missing Start-of-Packet was detected (and corrected).
STAT_RX_MEOP_ERR	Output	Missing End-of-Packet (EOP) Error. This signal indicates that a Missing End-of-Packet was detected (and corrected).
STAT_RX_BURST_ERR	Output	Burst Error. This signal indicates that a BurstShort or a burst length error was detected.
STAT_RX_MISALIGNED	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.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
STAT_RX_BAD_TYPE_ERR[11:0]	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_MUBITS_UPDATED	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.
STAT_RX_WORD_SYNC[11:0]	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.
Protocol Bypass Interface - RX Path Signals		
RX_BYPASS_RDIN	Input	This signal initiates a read operation.
RX_BYPASS_FORCE_REALIGNIN	Input	This signal causes the word synchronizer to sync again.
RX_BYPASS_IS_AVAILOUT[11:0]	Output	This signal indicates whether the data can be read using the RX_BYPASS_RDIN signal.
RX_BYPASS_DATAOUT0[65:0]	Output	Data or control word output for bypass lane0.
RX_BYPASS_DATAOUT1[65:0]	Output	Data or control word output for bypass lane1.
RX_BYPASS_DATAOUT2[65:0]	Output	Data or control word output for bypass lane2.
RX_BYPASS_DATAOUT3[65:0]	Output	Data or control word output for bypass lane3.
RX_BYPASS_DATAOUT4[65:0]	Output	Data or control word output for bypass lane4.
RX_BYPASS_DATAOUT5[65:0]	Output	Data or control word output for bypass lane5.
RX_BYPASS_DATAOUT6[65:0]	Output	Data or control word output for bypass lane6.
RX_BYPASS_DATAOUT7[65:0]	Output	Data or control word output for bypass lane7.
RX_BYPASS_DATAOUT8[65:0]	Output	Data or control word output for bypass lane8.
RX_BYPASS_DATAOUT9[65:0]	Output	Data or control word output for bypass lane9.
RX_BYPASS_DATAOUT10[65:0]	Output	Data or control word output for bypass lane10.
RX_BYPASS_DATAOUT11[65:0]	Output	Data or control word output for bypass lane11.
RX_BYPASS_ENAOUT[11:0]	Output	This signal qualifies the corresponding RX_BYPASS_DATAOUT bus.
RX_BYPASS_IS_BADLYFRAMEDOUT[11:0]	Output	This signal indicates badly framed data or control words (not valid for metaframe words).
RX_BYPASS_IS_SYNCWORDOUT[11:0]	Output	This signal identifies metaframe framing words.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
RX_BYPASS_IS_OVERFLOWOUT[11:0]	Output	This signal indicates whether the lane buffer has overflowed.
RX_BYPASS_IS_SYNCEDOUT[11:0]	Output	This signal indicates that the corresponding lane is ready for alignment.
Protocol Bypass Interface - TX Path Signals		
TX_BYPASS_ENAIN	Input	This signal qualifies the TX inputs.
TX_BYPASS_GEARBOX_SEQIN[7:0]	Input	This signal determines the gearbox sequencing according to a pre-determined format.
TX_BYPASS_MFRAMER_STATEIN[3:0]	Input	This signal determines the metaframe state according to a pre-determined format.
TX_BYPASS_CTRLIN[11:0]	Input	This signal identifies a word as being either data or control for the corresponding lane.
TX_BYPASS_DATAIN0[63:0]	Input	This bus is the data (or control) words for lane0.
TX_BYPASS_DATAIN1[63:0]	Input	This bus is the data (or control) words for lane1.
TX_BYPASS_DATAIN2[63:0]	Input	This bus is the data (or control) words for lane2.
TX_BYPASS_DATAIN3[63:0]	Input	This bus is the data (or control) words for lane3.
TX_BYPASS_DATAIN4[63:0]	Input	This bus is the data (or control) words for lane4.
TX_BYPASS_DATAIN5[63:0]	Input	This bus is the data (or control) words for lane5.
TX_BYPASS_DATAIN6[63:0]	Input	This bus is the data (or control) words for lane6.
TX_BYPASS_DATAIN7[63:0]	Input	This bus is the data (or control) words for lane7.
TX_BYPASS_DATAIN8[63:0]	Input	This bus is the data (or control) words for lane8.
TX_BYPASS_DATAIN9[63:0]	Input	This bus is the data (or control) words for lane9.
TX_BYPASS_DATAIN10[63:0]	Input	This bus is the data (or control) words for lane10.
TX_BYPASS_DATAIN11[63:0]	Input	This bus is the data (or control) words for lane11.
Miscellaneous Signals		
CTL_TX_RETRANS_ENABLE	Input	Reserved
CTL_TX_ERRINJ_BITERR_GO	Input	Reserved
STAT_TX_ERRINJ_BITERR_DONE	Output	Reserved
CTL_TX_ERRINJ_BITERR_LANE[3:0]	Input	Reserved
CTL_TX_RETRANS_REQ	Input	Reserved
CTL_TX_RETRANS_REQ_VALID	Input	Reserved
STAT_TX_RETRANS_RAM_WDATA[644-1:0]	Output	Reserved
STAT_TX_RETRANS_RAM_WE_B0	Output	Reserved
STAT_TX_RETRANS_RAM_WE_B1	Output	Reserved
STAT_TX_RETRANS_RAM_WE_B2	Output	Reserved
STAT_TX_RETRANS_RAM_WE_B3	Output	Reserved

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
STAT_TX_RETRANS_RAM_WADDR[9-1:0]	Output	Reserved
CTL_TX_RETRANS_RAM_RDATA[644-1:0]	Input	Reserved
STAT_TX_RETRANS_RAM_RD_B0	Output	Reserved
STAT_TX_RETRANS_RAM_RD_B1	Output	Reserved
STAT_TX_RETRANS_RAM_RD_B2	Output	Reserved
STAT_TX_RETRANS_RAM_RD_B3	Output	Reserved
STAT_TX_RETRANS_RAM_RADDR[9-1:0]	Output	Reserved
STAT_TX_RETRANS_RAM_RSEL[1:0]	Output	Reserved
CTL_TX_RETRANS_RAM_PERRIN	Input	Reserved
CTL_TX_RETRANS_RAM_BANKS[1:0]	Input	Reserved
STAT_TX_RETRANS_BURST_ERR	Output	Reserved
STAT_TX_RETRANS_BUSY	Output	Reserved
STAT_TX_RETRANS_RAM_PERROUT	Output	Reserved
CTL_RX_RETRANS_ENABLE	Input	Reserved
STAT_RX_RETRANS_REQ	Output	Reserved
CTL_RX_RETRANS_ACK	Input	Reserved
STAT_RX_RETRANS_STATE[2:0]	Output	Reserved
STAT_RX_RETRANS_SEQ[7:0]	Output	Reserved
STAT_RX_RETRANS_SUBSEQ[4:0]	Output	Reserved
STAT_RX_RETRANS_SEQ_UPDATED	Output	Reserved
CTL_RX_RETRANS_RESET	Input	Reserved
CTL_RX_RETRANS_RESET_MODE	Input	Reserved
STAT_RX_RETRANS_RETRY_ERR	Output	Reserved
STAT_RX_RETRANS_WRAP_ERR	Output	Reserved
STAT_RX_RETRANS_WDOG_ERR	Output	Reserved
CTL_RX_RETRANS_ERRIN	Input	Reserved
STAT_RX_RETRANS_DISC	Output	Reserved
STAT_RX_RETRANS_CRC24_ERR	Output	Reserved
CTL_RX_RETRANS_FORCE_REQ	Input	Reserved
STAT_RX_RETRANS_LATENCY[15:0]	Output	Reserved
DRP Path/Control Signals		
DRP_DO[15:0]	Output	Data bus for reading configuration data from the CMAC to the FPGA logic resources.
DRP_RDY	Output	Indicates operation is complete for write operations and data is valid for read operations.

Table 2-3: **UltraScale Device Interlaken Primitive Ports *(Cont'd)***

Name	Direction	Description
DRP_ADDR[9:0]	Input	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	Data bus for writing configuration data from the FPGA logic resources to the Interlaken core.
DRP_EN	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	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.

Attribute Descriptions

Table 2-4 provides a detailed description for the UltraScale device integrated Interlaken core attributes and their default values.

Table 2-4: **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 mode or not. TRUE: Enables protocol bypass mode. FALSE: Disables protocol bypass 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: 0x0 = 16 entries; 0x1 = 32 entries; 0x3 = 64 entries; 0x7 = 128 entries; 0xF = 256 entries. All other values are reserved and must not be used. This attribute should only be changed by DRP when `TX_RESET` is asserted.	7'h00
CTL_TX_BURSTMAX[1:0]	2-bit Hex	Interlaken TX BurstMax. This bus sets the BurstMax parameter for the TX as follows: 0x0 = 64 bytes 0x1 = 128 bytes 0x2 = 192 bytes 0x3 = 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

Table 2-4: **UltraScale Device Integrated Interlaken Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
CTL_TX_BURSTSHORT[2:0]	3-bit Hex	Interlaken TX BurstShort. This bus sets the BurstShort parameter for the TX as follows: 0x0 = not valid 0x1 = 64 bytes 0x2 = 96 bytes 0x3 = 128 bytes 0x4 = 160 bytes 0x5 = 192 bytes 0x6 = 224 bytes 0x7 = 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]. This attribute should only be changed through DRP when `TX_RESET` is asserted.	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_MINUS 1 is 255 or 0xFF. This attribute should only be changed when `TX_RESET` is asserted.	16'h07FF
CTL_TX_LAST_LANE[3:0]	4-bit Hex	This attribute is used for lane decommissioning of the TX.	4'hB

Table 2-4: **UltraScale Device Integrated Interlaken Core Attributes *(Cont'd)***

Name	Type	Description	Default Value
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: 0x0 = 64 bytes 0x1 = 128 bytes 0x2 = 192 bytes 0x3 = 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. This attribute should only be changed when RX_RESET or RX_FORCE_RESYNC are asserted.	16'h07FF
CTL_RX_LAST_LANE[3:0]	4-bit Hex	This attribute is used for lane decommissioning of the RX.	4'hB

Table 2-4: **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
Miscellaneous Attributes			
CTL_TX_RETRANS_MULT[2:0]	3-bit Hex	Reserved	3'h0
CTL_TX_RETRANS_DEPTH[13:0]	14-bit Hex	Reserved	14'h0800
CTL_RX_RETRANS_MULT[2:0]	3-bit Hex	Reserved	3'h0
CTL_RX_RETRANS_TIMER1[15:0]	16-bit Hex	Reserved	16'h0000
CTL_RX_RETRANS_TIMER2[15:0]	16-bit Hex	Reserved	16'h0008
CTL_RX_RETRANS_RETRY[3:0]	4-bit Hex	Reserved	4'h2
CTL_RX_RETRANS_WRAP_TIMER[7:0]	8-bit Hex	Reserved	8'h00
CTL_RX_RETRANS_WDOG[11:0]	12-bit Hex	Reserved	12'h000
Testing Attributes			
CTL_TEST_MODE_PIN_CHAR	Boolean	Reserved. Set this attribute to FALSE.	FALSE
TEST_MODE_PIN_CHAR	Boolean	Reserved. Set this attribute to FALSE.	FALSE

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[(LANES-1):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[(LANES-1):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 *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS893) [Ref 2], for the maximum allowable clock frequency across speed grades.

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

Interlaken Configuration	TX_SERDES_REFCLK	RX_SERDES_CLK	CORE_CLK	LBUS_CLK
12x12.5 Gb/s	195.3125 MHz	195.3125 MHz	300 MHz to 349.52 MHz	300 MHz to 349.52 MHz
6x25.78 Gb/s	402.8125 MHz	402.8125 MHz	414.25 MHz	300 MHz to 349.52 MHz

Resets

The UltraScale™ architecture 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/de-asserting different resets). The core must be held in reset until the corresponding clock is fully stable.

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.

Figure 3-1 shows the clocking reset diagrams for Interlaken core.

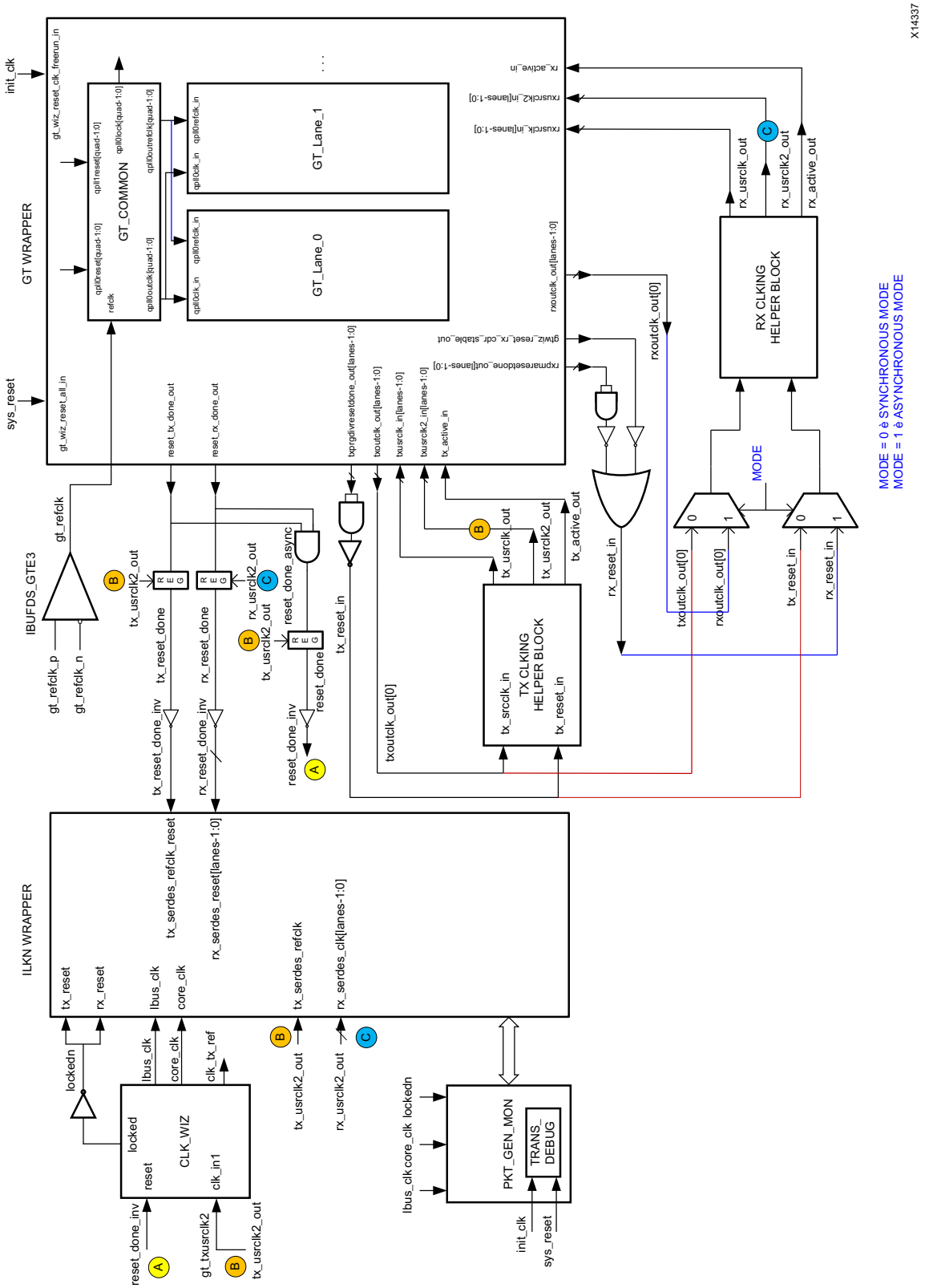

Figure 3-1: **Interlaken Core Clocking Reset Interface**

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 each segment.

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_enain` 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_enain` 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_enain` is sampled as 1.

tx_sopin0

Transmit LBUS Start Of Packet. This signal is used to indicate the Start Of Packet (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_enain` is sampled as 1.

tx_eopin0

Transmit LBUS End Of Packet. This signal is used to indicate the End Of Packet (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_enain` 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_enain` and `tx_eopin` are sampled as 1.

tx_mtyin0[3:0]

Transmit LBUS Empty. 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 and treated as if it was 000. The other bits of `tx_mtyin` are 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_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 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_enaout` 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_enaout` 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_enaout` is sampled as 1.

rx_sopout0

Receive LBUS Start-Of-Packet. This signal indicates the Start Of Packet (SOP) when it is sampled as 1 and is only valid in cycles during which `rx_enaout` is sampled as a 1.

rx_eopout0

Receive LBUS End-Of-Packet. This signal indicates the End Of Packet (EOP) when it is sampled as 1 and is only valid in cycles during which `rx_enaout` 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_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[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_enaout` and `rx_eopout` are sampled as 1.

When `rx_errout` and `rx_enaout` are sampled as 1, the value of `rx_mtyout[2:0]` is always 000. Other bits of `rx_mtyout` are 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 `clk` port. Figure 3-2 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.

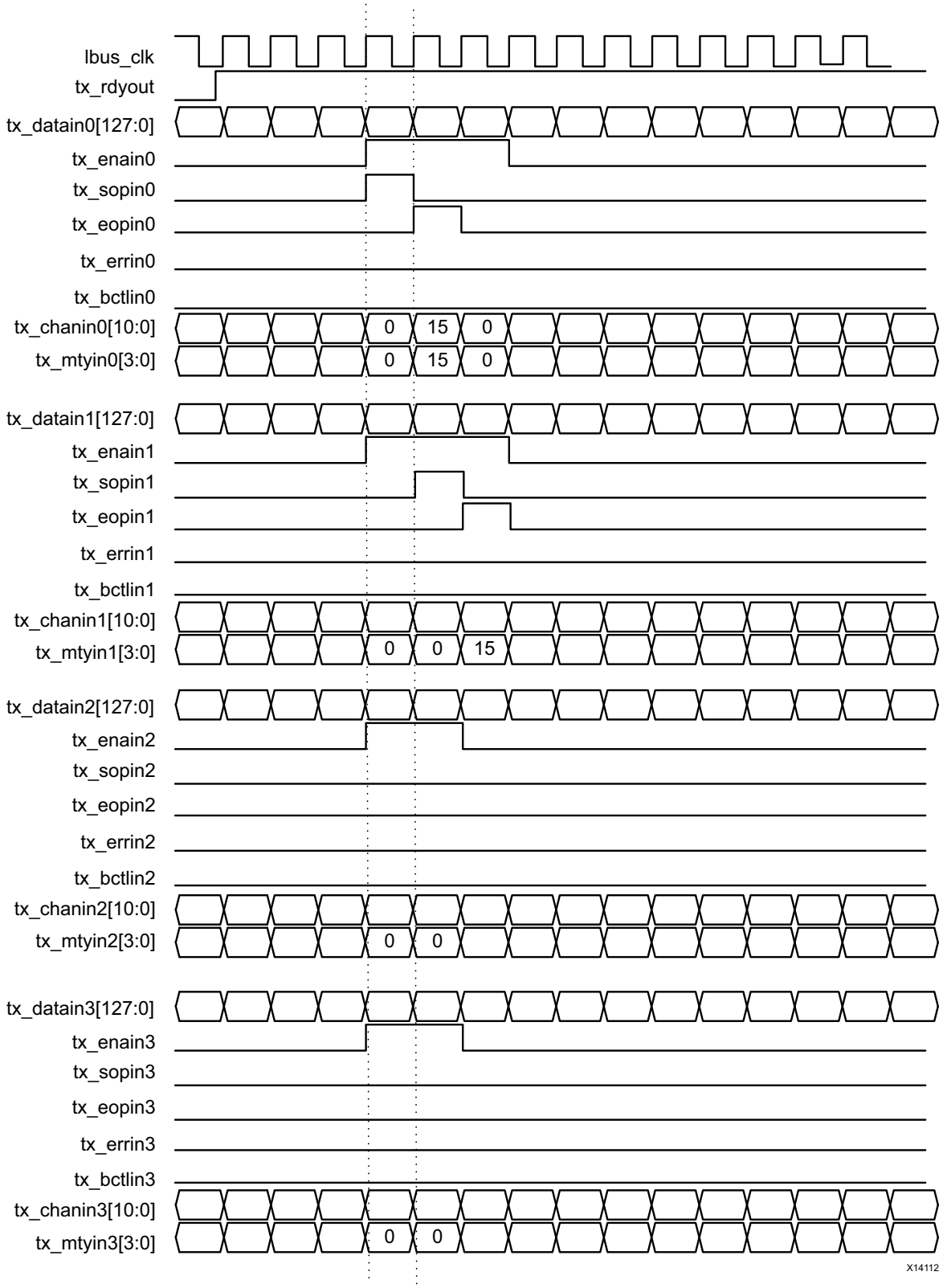

Figure 3-2: **Sample Waveform for TX LBUS Interface**

TX Transactions

Data is written into the interface on every clock cycle when tx_enain is asserted. This signal qualifies the other inputs of the TX Local bus interface. This signal must be valid every clock cycle. When tx_enain is deasserted, data on the other buses is ignored.

The start of a packet is identified by asserting tx_sopin with tx_enain. The end of a packet is identified by asserting tx_eopin with tx_enain. Both tx_sopin and tx_eopin can be asserted during the same cycle provided there are no empty segments between them. This is done for packets that are less than or equal to the bus width.

Data is presented on the tx_datain inputs. For a given segment, the first byte of the packet is written on bits [127:120], the second byte on bits [119:112], and so forth.

For a 128-bit segment, the first 16 bytes of a packet are presented on the bus during the cycle that tx_sopin and tx_enain are asserted. Subsequent 16-byte chunks are written during successive cycles with tx_sopin negated. The last bytes of the packet are written with tx_eopin asserted. Unless tx_eopin is asserted, all 128 bits must be presented with valid data whenever tx_enain is asserted.

During the last cycle of a packet the tx_mtyin signals might be asserted. The value of tx_mtyin must be 0 for all but the last cycle. The tx_mtyin signals indicate how many byte lanes in the data bus are invalid (or empty). The tx_mtyin signals only have meaning during cycles when both tx_enain and tx_eopin are asserted. For a 128-bit wide segment, tx_mtyin is 4 bits wide.

If tx_mtyin has a value of 0x0, there are no empty byte lanes, or in other words, all bits of the data bus are valid. If tx_mtyin has a value of 0x1, then the 1-byte lane is empty, specifically bits [7:0] of tx_datain do not contain valid data. If tx_mtyin has a value of 0x2, then the 2-byte lanes are empty, specifically bits [15:0] do not contain valid data. If tx_mtyin has a value of 0x3, then the 3-byte lanes are empty, specifically bits [23:0] do not contain valid data, and this pattern continues until 15 of 16 bytes are invalid or empty.

During the last cycle of a packet, when tx_eopin is asserted with tx_enain, tx_errin might also be asserted. This marks the packet as being in error. When tx_errin is asserted, the value of tx_mtyin is ignored.

tx_rdyout

Data can be safely written, that is, tx_enain asserted, whenever tx_rdyout is asserted. After tx_rdyout is negated, additional writes, using tx_enain, 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. The threshold for an overflow indication is fixed at a value of 11. Up to 8 write cycles can be safely performed after tx_rdyout is negated, but no more until tx_rdyout is asserted again.

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. The first data written for a packet begins with TX_SOPIN asserted and the final write operation is TX_EOPIN asserted.

Unless TX_BCTLIN 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 has been asserted for a channel, it cannot be asserted again for that channel until a corresponding TX_EOPIN for that channel has been written.
- Unless TX_EOPIN is asserted, the full width of the data bus, TX_DATAIN, 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 input (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 input operates in a similar manner to TX_SOPIN or a change in TX_CHANIN; both signals cause a Burst Control Word to be injected into the data stream.

The purpose of the TX_BCTLIN 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, 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 TX_BCTLIN 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. 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 3-3 shows a sample waveform for two data transactions for 65-byte packets using a 512-bit segmented LBUS.

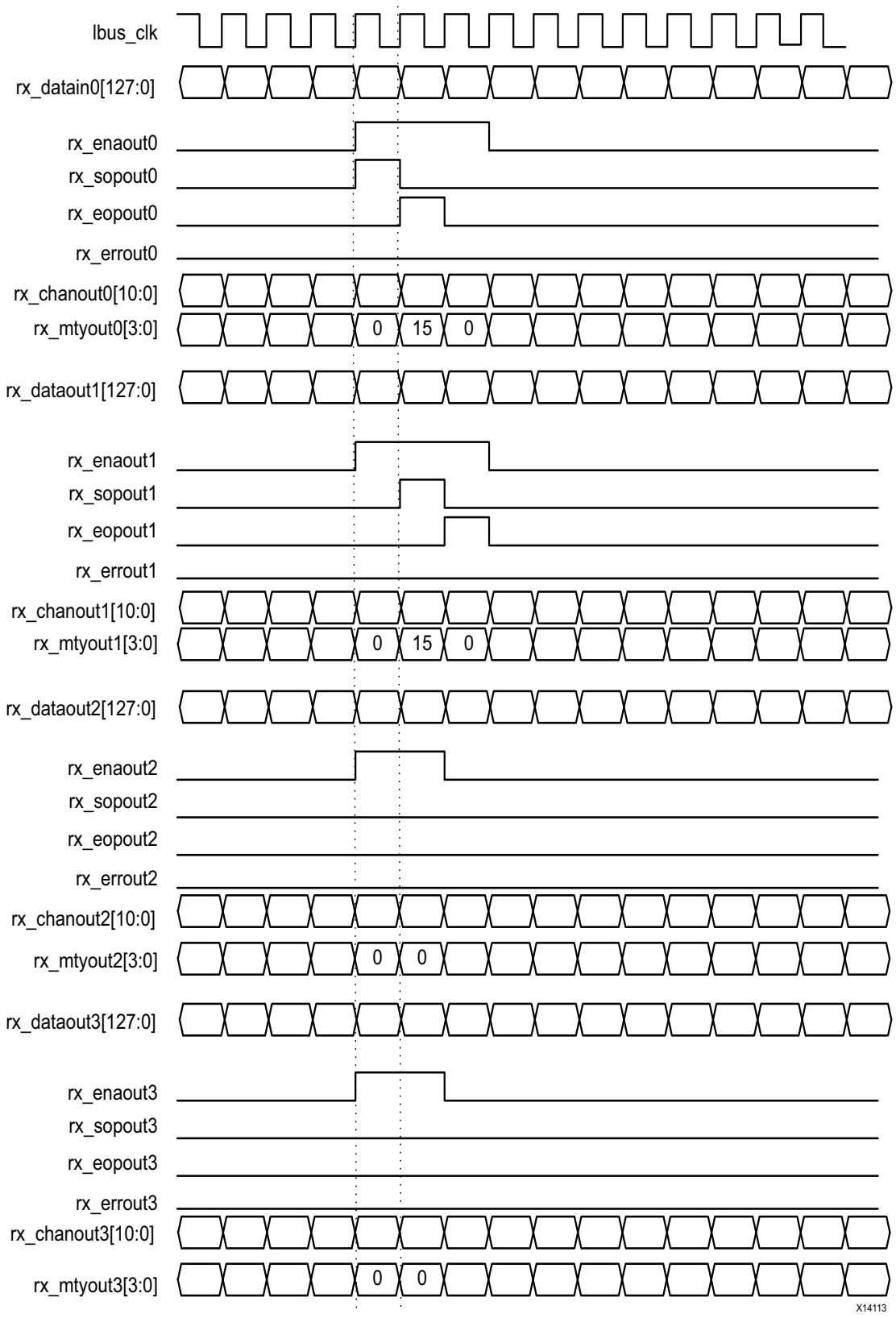

Figure 3-3: **Sample Waveform for Two Data Transactions**

Data is supplied by the Interlaken core on every clk clock cycle when `rx_enaout` is asserted. This signal qualifies the other outputs of the RX Local bus interface.

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

As in the TX, the first byte of a packet is supplied on the most significant bits of rx_dataout. For a 128-bit wide segment, the first byte of the packet is written on bits [127:120], the second byte on bits [119:112], and so forth.

As in the TX, portions of packets are written on the bus in the full width of the bus unless `rx_eopout` is asserted. When `rx_eopout` is asserted, the `rx_mtyout` bus indicates how many byte lanes in the data bus are invalid. The encoding is the same as for tx_mtyin.

During the last cycle of a packet, when rx_eopout is asserted with `rx_enaout`, `rx_errout` can also be asserted. This indicates the packet received either

- had an FCS error such as CRC24 or
- was sent with the "error flag" set

There is no mechanism to back pressure the RX Local bus interface. The user logic must be capable of receiving data when `rx_enaout` 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`.

Bus Rules

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

Segment Ordering

The 128-bit segments are ordered 0 to 3 (for a 512-bit segmented LBUS). 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_enainX` 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_enainX` 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_enainX`:

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_enainX` 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_enainX` 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_enainX` 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 (Start of Packet), Dat (data in the middle of a packet), EoP (End of Packet), 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.

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	EoP	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.
- 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_sopinX` and `tx_bctlinX` input, forced through a change of channel on `tx_chanin`, 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.

Note: It is possible to set BurstShort = 64 Bytes and BurstMax = 64 Bytes. In cases such as those, the calculation for BurstMin does not apply, because bursts are not allowed to be less than BurstShort. This means that the enhanced scheduling algorithm needs to be modified such that all bursts are 64 Bytes. Also there will be bandwidth wasted due to idle byte insertion.

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[LANES-1:0]

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[LANES-1:0]

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[LANES-1:0]

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[LANES-1:0]

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[LANES-1:0]

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[LANES-1:0]

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[LANES-1:0]

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 Start of Packet (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 End of Packet (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 Start of Packet (SOP) and end with a valid End of Packet (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 End of Packet (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 BurstShort parameter. A burst length violation occurs when the length of a received burst, other than that ending with an End of Packet, is not a multiple of the RX 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

= (2e9 * 16) / (200e6)

= 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 CORE_CLK. The Interlaken IP core uses the following two signals for this function:

STAT_RX_CRC32_VALID[LANES-1:0]

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[LANES-1:0]

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, 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[LANES-1:0]

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[LANES-1:0] is a value of 0.

STAT_RX_DIAGWORD_LANESTAT[LANES-1:0]

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[LANES-1:0]` 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[LANES-1:0]

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[LANES-1:0]` 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 `CORE_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.

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]

These outputs contain the information in bits [31:24] of the Control words received by the Receiver. 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.

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 `CORE_CLK` and a detailed description of each signal follows.

Note: The MAX_CALLEN parameter represents the supported calendar length for each specific Interlaken IP core configuration.

CTL_TX_FC_STAT[MAX_CALLEN-1:0]

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 the MAX_CALLEN of the Interlaken IP core configuration. 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: The value selected for CTL_TX_FC_CALLEN must be less than or equal to the value of MAX_CALLEN. 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.

Note: The MAX_CALLEN parameter represents the support calendar length for each specific Interlaken IP core configuration.

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[MAX_CALLEN -1:0]

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 UltraScale architecture integrated 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

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

Dynamic Reconfiguration Port

The dynamic reconfiguration port (DRP) allows the dynamic change of attributes in the UltraScale Interlaken core. 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 *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS893) [Ref 2], for the maximum allowed clock frequency.

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 (DRP) 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 multicycle 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 3-4 shows the DRP write operation timing diagram. New DRP operations can be initiated when the DRP_RDY signal is asserted.

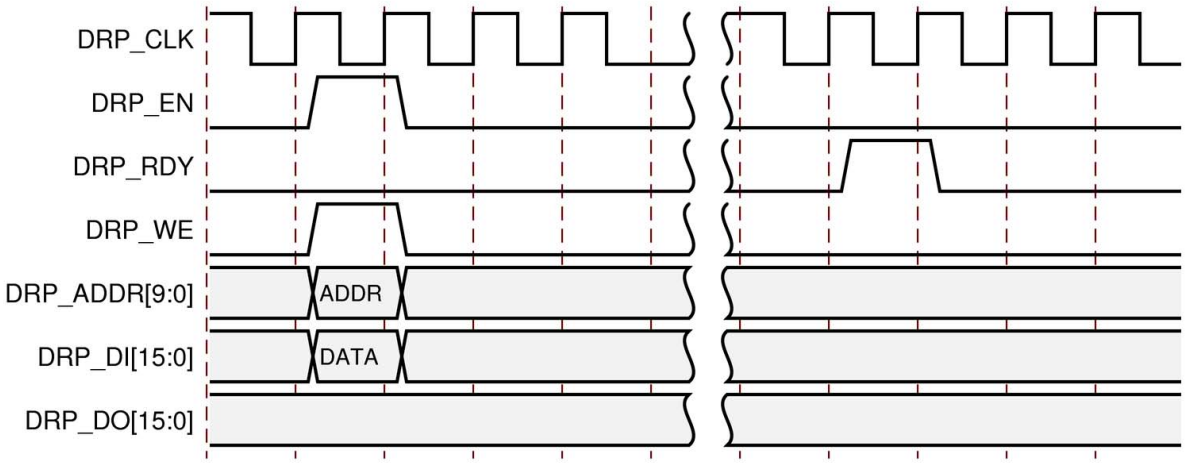

Figure 3-4: **DRP Write Operation Timing Diagram**

DRP Read Operation

Figure 3-5 shows the DRP read operation timing diagram. New DRP operations can be initiated when the DRP_RDY signal is asserted.

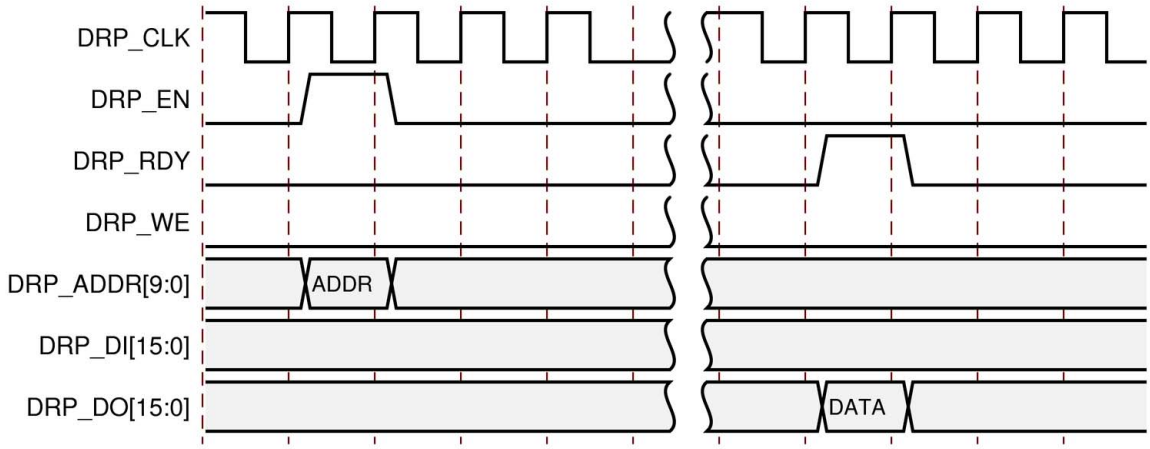

Figure 3-5: **DRP Read Operation Timing Diagram**

DRP Address Map of the ILKN Block

Table 3-6 lists the DRP map of the ILKN block sorted by address.

Table 3-6: **DRP Map**

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]	1-B	1-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]	1-B	1-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
58	0	R/W	BYPASS	FALSE	0
				TRUE	1

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

DRP Address (Hex)	DRP Bits	R/W	Attribute Name	Attribute Encoding (Hex)	DRP Encoding (Hex)
60	0	R/W	TEST_MODE_PIN_CHAR	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]	0-FFFF	0-FFFF
A8	[15:0]	R/W	CTL_RX_RETRANS_TIMER2[15:0]	8-FFFF	8-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
C1	0	R/W	CTL_TEST_MODE_PIN_CHAR	FALSE	0
				TRUE	1
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

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 in 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 4]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 6]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 7]

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 UltraScale™ architecture 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 5] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 6].

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.

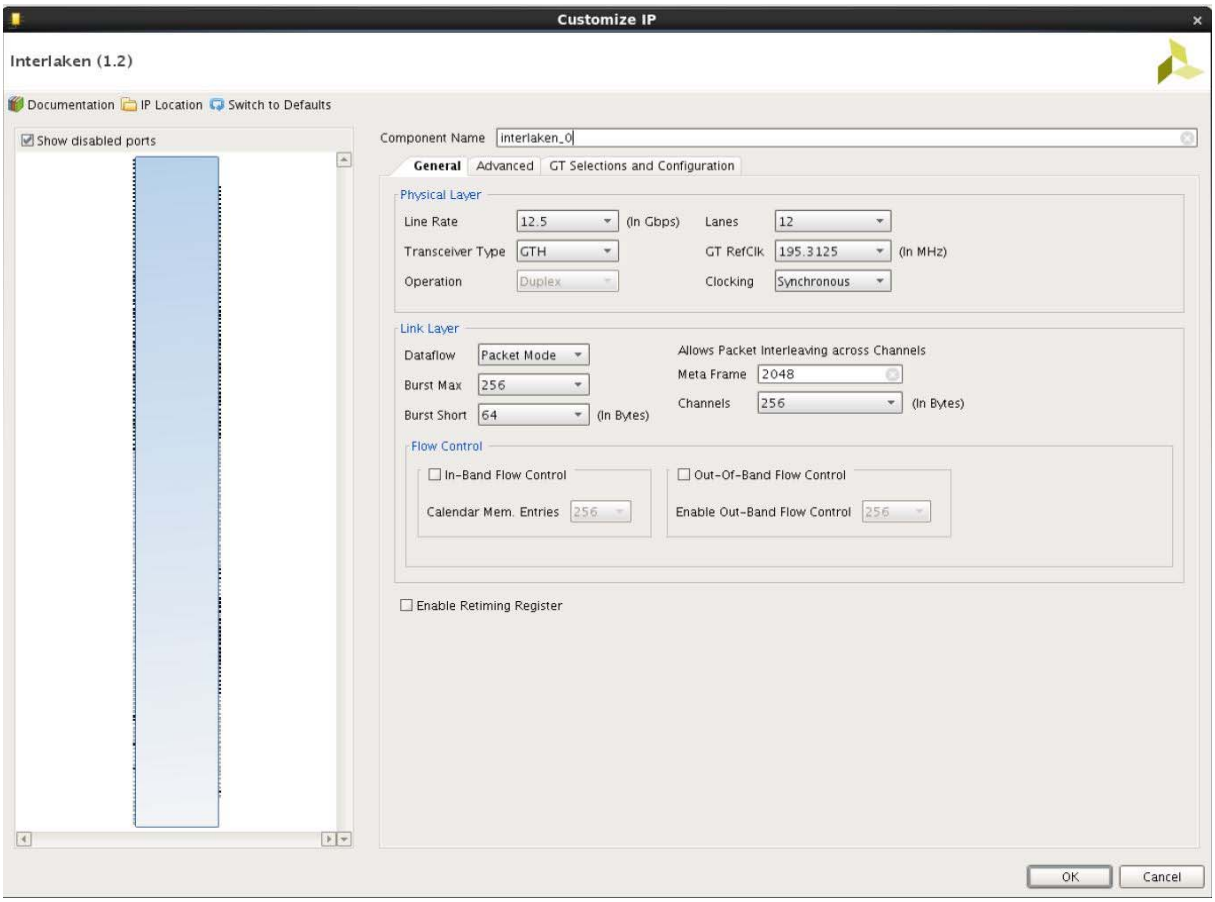

Figure 4-1: **General Tab**

Table 4-1 describes the General tab options.

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

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

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

GUI 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(3)	Operating mode	Duplex	Duplex
Clocking (4)	Clocking mode	Synchronous	Synchronous Asynchronous
Low Latency Mode(5)	Low Latency	1	0 – Disable 1 – Enable
Link Layer			
Data Flow(6)	Packet Mode	Packet mode	Packet Mode Burst Interleaved Mode
Channels	Number of Channels	256	256 512 1,024 2,048
Meta Frame	Meta Frame length in Interlaken words	2,048	256 to 8192
BurstMax(7)	Burst Max in Bytes	256	64 128 192 256
BurstShort(7)(8)	Burst Short in Bytes	64	64 96 128
Flow Control			
In-Band Flow Control	Enable In-band flow control	0	0 – Disable 1 – Enable

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

GUI Parameter	Description	Default Value	Value Range
In-Band Flow Control Calendar Length	Entries for calendar length	256	16 32 64 128 256
Out-of-Band Flow Control	Enable out-of-band flow control	0	0 – Disable 1 – Enable
Out-of-band Flow Control Calendar Length	Entries for calendar length	256	128 256 512 1024
Miscellaneous			
Enable Retiming Register	Enable insertion of pipeline registers in the FPGA fabric between the Interlaken core and GTH/GTY transceiver interface to ease timing	0	0 – Disable 1 – Enable

1. This release only supports 12 L x12.5 Gb/s, and 6 L x25.78125 Gb/s.
2. 25.78125 Gb/s line rate not supported for devices that have only GTH support or -1 speed grade.
3. Simplex TX and Simplex RX are not supported in this core version.
4. Applicable only when the operating mode is duplex.
5. Available only when all the GTs are within an SLR (GT buffer bypass is not supported across SLRs). This will drive the following user parameters of the GT Wizard: Enable (GT buffer bypass, that is, TX_BUFFER_MODE=0 and RX_BUFFER_MODE=0), and Disable (GT buffer enable, that is, TX_BUFFER_MODE=1 and RX_BUFFER_MODE=1).
6. Applicable only for the RX. The operating mode must be set to Duplex or Simplex Rx.
7. Applicable only for the TX. The operating mode must be set to Duplex or Simplex Tx.
8. BurstShort must be less than or equal to BurstMax.

Advanced Tab

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

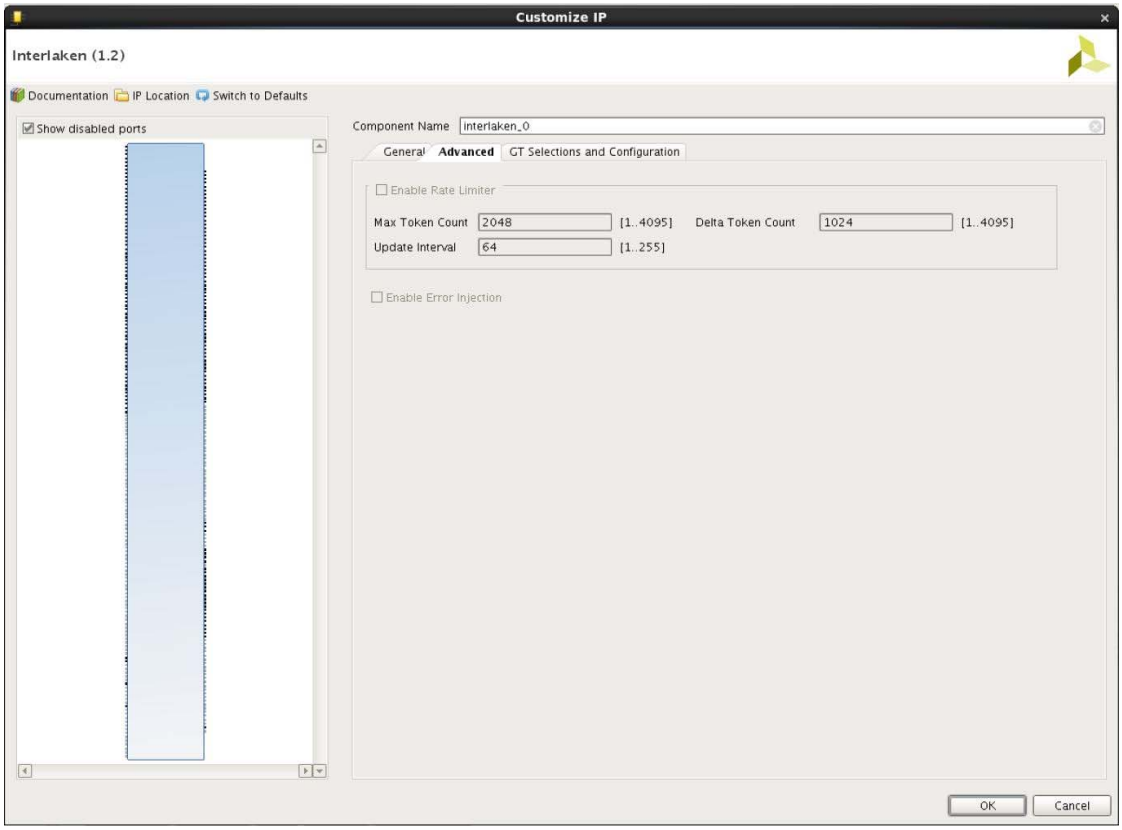

Figure 4-2: **Advanced Tab**

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

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

GUI Parameter	Description	Default Value	Value Range
Enable Retransmission[1]		0	0 – Disable 1 – Enable
Transmitter			
Number of RAM Banks[2]		4	1 2 3 4
Buffer Depth[2]		2048	512 1024 2048 4096 8192
Sequence Multiplier[2]		1	1 2 4 8 16 32
Receiver			
Sequence Multiplier[2]		1	1 2 4 8 16 32
Watchdog Timer[2]		0	0 to 4095
Long Timer[2]	Long timer in Interlaken words	3000	8 to 65535
Short Timer[2]		2048	8 to 65535
Retry Timer[2]		2	2 to 15
Wrap Around Timer[2]	Wrap Around Timer	0	0 to 255
Rate Limiter			
Enable Rate Limiter[3]		0	0 – Disable 1 – Enable
Max Token Count[4][5]		2048	Burst Max to 4095
Delta Token Count[4][6]	Delta token count in Bytes	1024	1 to 4095
Update Interval[4][7]	Update Interval Cycles for core clk cycles	64	1 to 255

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

GUI Parameter	Description	Default Value	Value Range
Miscellaneous			
Enable Error Injection[8]		0	0 – Disable 1 – Enable

Notes:

1. Out-of-band flow control must be enabled to support retransmission, Operation should be Duplex.
2. Applicable only if retransmission is enabled.
3. Enabling rate limiter is not supported in this core version.
4. Applicable only if the Rate Limiter is enabled
5. This value must be equal to or greater than BurstMax
6. This value must be less than Max Token Count.
7. Recommended value is greater than or equal to 8.
8. Enabling error injection is not supported in this core version.

GT Selections and Configuration

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

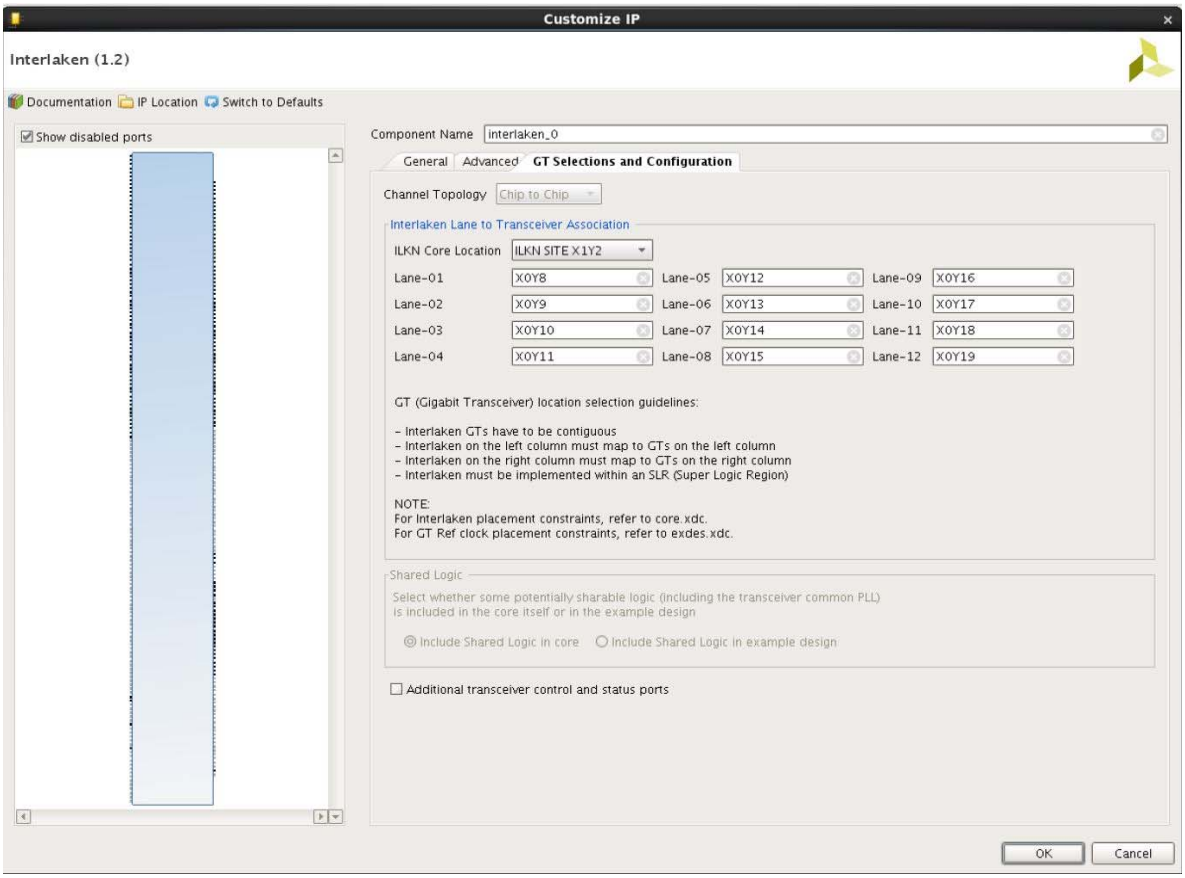

Figure 4-3: **GT Selections and Configuration Tab**

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

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

GUI Parameter	Description	Default Value	Value Range
Channel Topology	Type of channel used	Chip to Chip	Chip to Chip Backplane
Interlaken Core Location	XY location of the transceiver connected to the Interlaken core	Based on the selected package/ device	Options based on the selected package/device
Lane 00 to Lane 11	Enter a value based on selected package/device	X0Y11	X0Y0 to X0Y11
Shared Logic	Determines the location of the transceiver shared logic	Include Shared Logic in Core	Include Shared Logic in Core Include Shared Logic in Example Design
Additional transceiver control and status ports	Enables additional transceiver control and status ports	0	0 – Disable 1 – Enable

User Parameters

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

Table 4-4: **GUI Parameter to User Parameter Relationship**

GUI Parameter/Value[(1)]	User Parameter/Value[(1)]	Default Value
Line Rate	LINE_RATE	12.5
Lanes	NUM_LANES	12
Transceiver Type	GT_TYPE	GTH
GT Ref Clk	GT_REF_CLK_FREQ	195.3125
Operation Duplex	OPERATING_MODE duplex	duplex
Clocking Synchronous Asynchronous	CLOCKING_MODE synchronous asynchronous	synchronous
Low Latency Mode	LOW_LATENCY_MODE	1 (enable)
Data Flow Packet Mode Burst Mode	PACKET_MODE packet_mode burst_mode	packet_mode
Channels	CHAN_EXT	256
MetaFrame	MFRAMELEN	2048
Burst Max	BURSTMAX	256

Table 4-4: **GUI Parameter to User Parameter Relationship *(Cont'd)***

GUI Parameter/Value[1]	User Parameter/Value[1]	Default Value
Burst Short	BURSTSHORT	64
In-Band Flow Control	ENABLE_IN_BAND_FC	0 (disable)
In-Band Flow Control Calendar Length	IN_BAND_FC_CALLEN	256
Out-of-Band Flow Control	ENABLE_OUT_BAND_FC	0 (disable)
Out-of-band Flow Control Calendar Length	OUT_BAND_FC_CALLEN	256
Enable Retiming Register	ENABLE_RETIMING_REGISTER	0 (disable)
Enable Retransmission	EN_RETRANS	0 (disable)
Number of RAM Banks	TX_RETRANS_RAM_BANKS	4
Buffer Depth	TX_RETRANS_DEPTH	2048
Sequence Multiplier	TX_RETRANS_MULT	1
Sequence Multiplier	RX_RETRANS_MULT	1
Watchdog Timer	RX_RETRANS_WDOG	0
Long Timer	RX_RETRANS_TIMER1	3000
Short Timer	RX_RETRANS_TIMER2	2048
Retry Timer	RX_RETRANS_RETRY	2
Wrap Around Timer	RX_RETRANS_WRAP_TIMER	0
Enable Rate Limiter	TX_RLIM_ENABLE	0
Max Token Count	TX_RLIM_MAX	2048
Delta Token Count	TX_RLIM_DELTA	1024
Update Interval	TX_RLIM_INTV	64
Enable Error Injection	ENABLE_ERR_INJ	0
Channel Topology Chip to Chip Backplane	CHANNEL_TOPOLOGY chip_to_chip backplane	chip_to_chip
Interlaken Core Location	ILKN_CORE_LOC	Based on selected device/ package
Lane 00 to Lane 11	LANE0_GT_LOC to LANE11_GT_LOC	to X0Y11
Shared Logic Core Example design	INCLUDE_SHARED_LOGIC 2 1	2
Additional transceiver control and status ports	ADD_GT_CNRL_STS_PORTS	0 (disable)

1. Parameter values are listed in the table where the GUI parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Constraining the Core

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

This section can also contain board layout information, if applicable.

Required Constraints

The UltraScale architecture integrated LogiCORE™ IP 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 8].

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 7].

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

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

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.

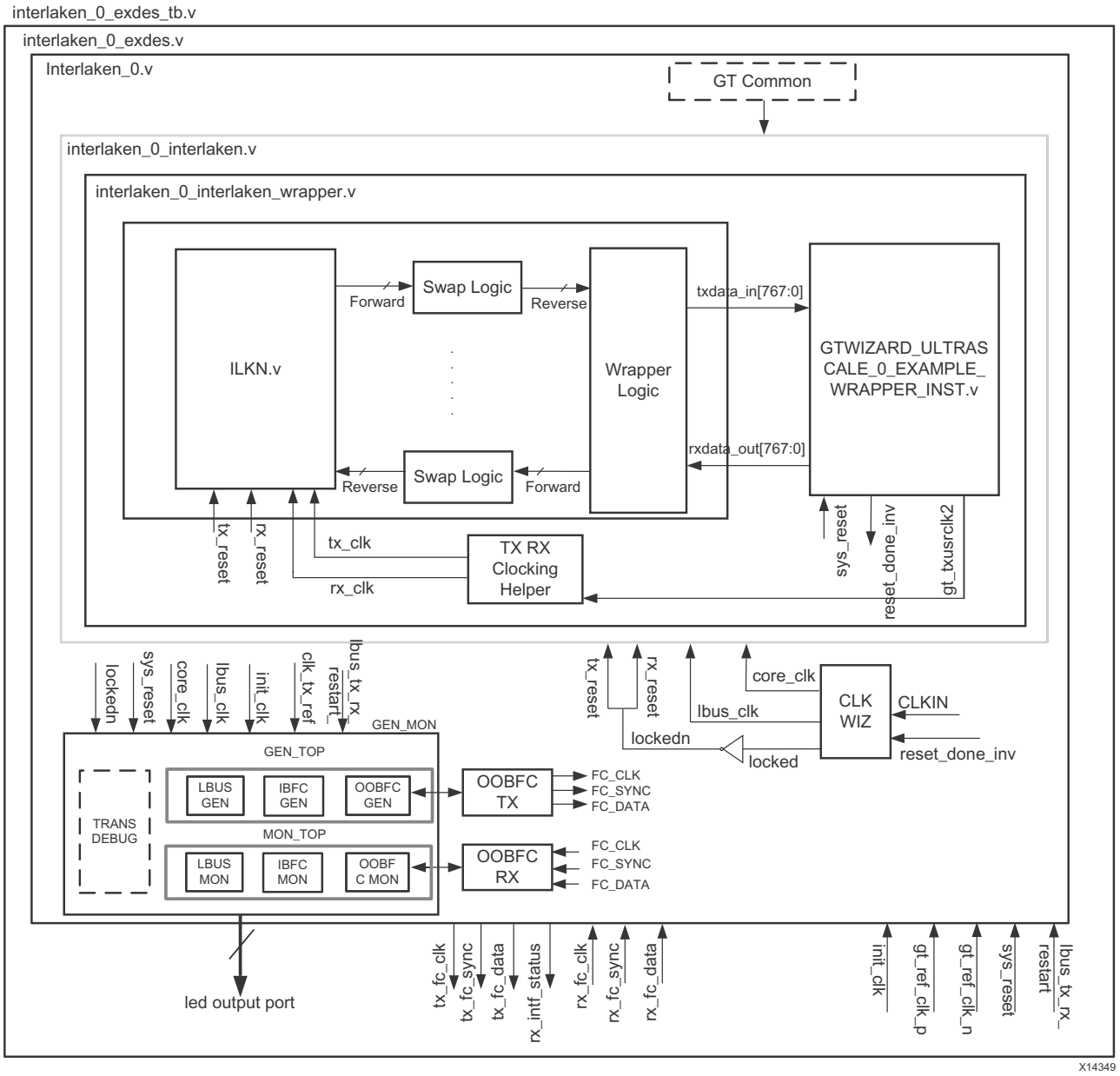

Figure 5-1: **Example Design Hierarchy**

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 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. The number of packets can be changed 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 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 300MHz 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 300MHz and `core_clk` with a frequency of 412 MHz.

User Interface

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

Table 5-1: **User I/O Ports**

Name	Size	Direction	Description
sys_reset	1	Input	Reset for interlaken_0.
gt_ref_clk_p	1	Input	Differential input clk to GT.
gt_ref_clk_n	1	Input	Differential input clk to GT.
init_clk	1	Input	Stable input clk to GT.
lbus_tx_rx_restart_in	1	Input	This signal initiates the state transaction in packet generator and monitor module. This signal is used to enable the packet generation and reception when the packet generator and the packet monitor are idle (that is, `tx_busy_led` = 0 and `rx_busy_led` = 0).
tx_gt_locked_led	1	Output	Indicates that GT has been locked. 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 the generator busy, and not able to respond to the `lbus_tx_rx_restart_in` command.
rx_gt_locked_led	1	Output	Indicates that 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 5-1, a three-stage data registering is done internally.

Modes of Operation

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

- Duplex Mode
- Simplex TX Mode
- Simplex RX Mode

IMPORTANT: *Simplex TX Mode and Simplex RX Mode are not supported in this release.*

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 5-2 shows the duplex mode of operation.

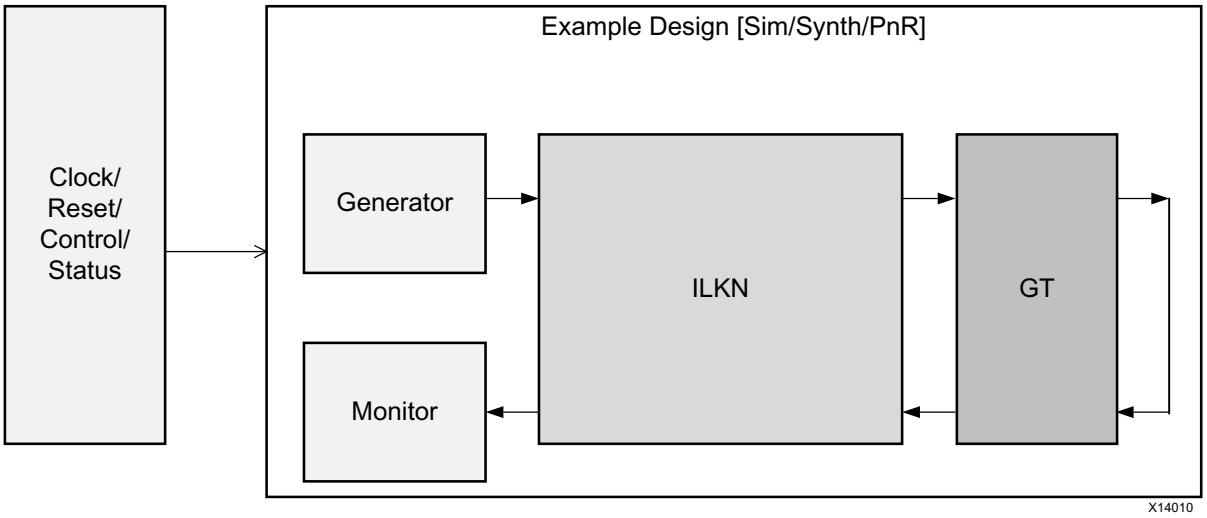

Figure 5-2: **Duplex Mode of Operation**

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 5-3 shows the Simplex TX mode of operation.

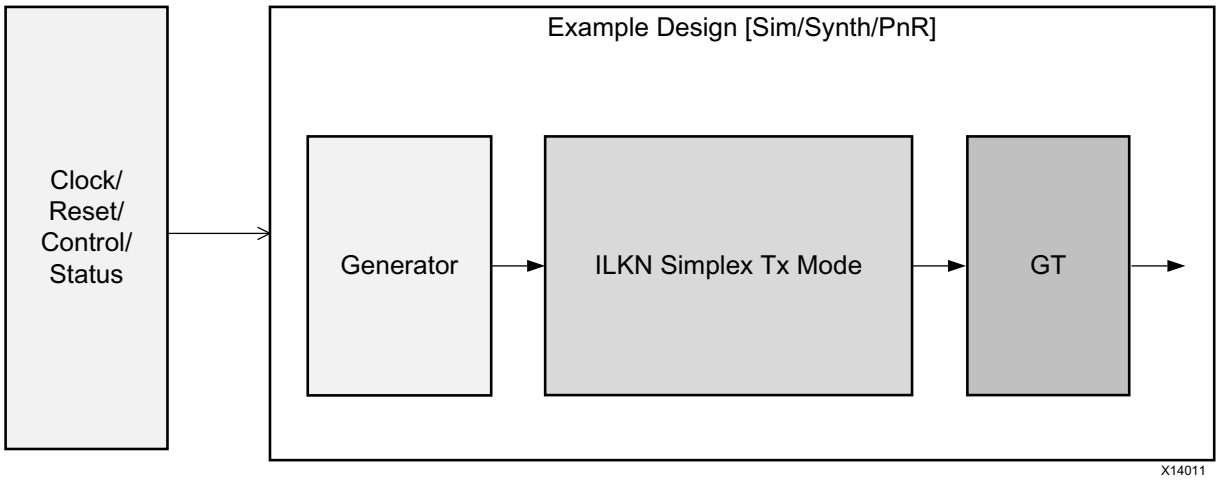

Figure 5-3: **Simplex TX Mode of Operation**

Simplex RX Mode

In this mode of operation only the Interlaken receiver is enabled as shown in Figure 5-4. 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 5-4 shows the Simplex RX mode of operation.

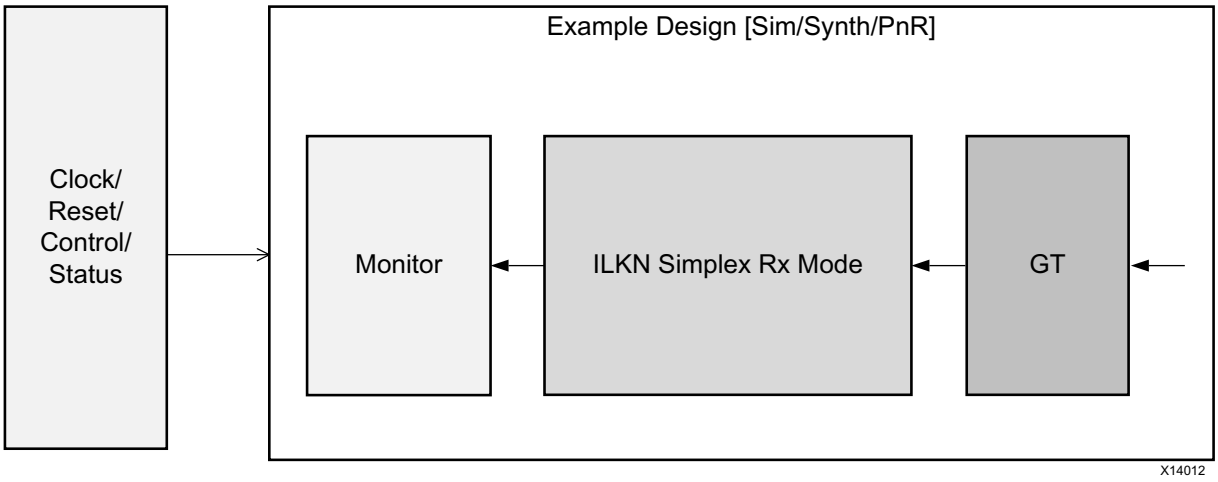

Figure 5-4: **Simplex RX Mode of Operation**

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 this is done 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 5-5.

- **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 the `ctl_tx_enable=1'b0`, `ctl_tx_mubits=8'd0`, `ctl_tx_diagword_lanestat = 12'hFFF`, `gt_loopback_in = 36'd0`, `ctl_tx_diagword_intfstat = 1'b1` and `init_done = 1'b0`, and moves to WAIT_RX_ALIGN_STATE.
- **WAIT_RX_ALIGN_STATE**: Waits for the integrated IP core for Interlaken (ILKN) `stat_rx_aligned` or (`lbus_tx_rx_restart_in` and Simplex TX mode), which indicates the ILKN RX core is aligned, after that FSM will move to STATE_PKT_TRANSFER_INIT 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.

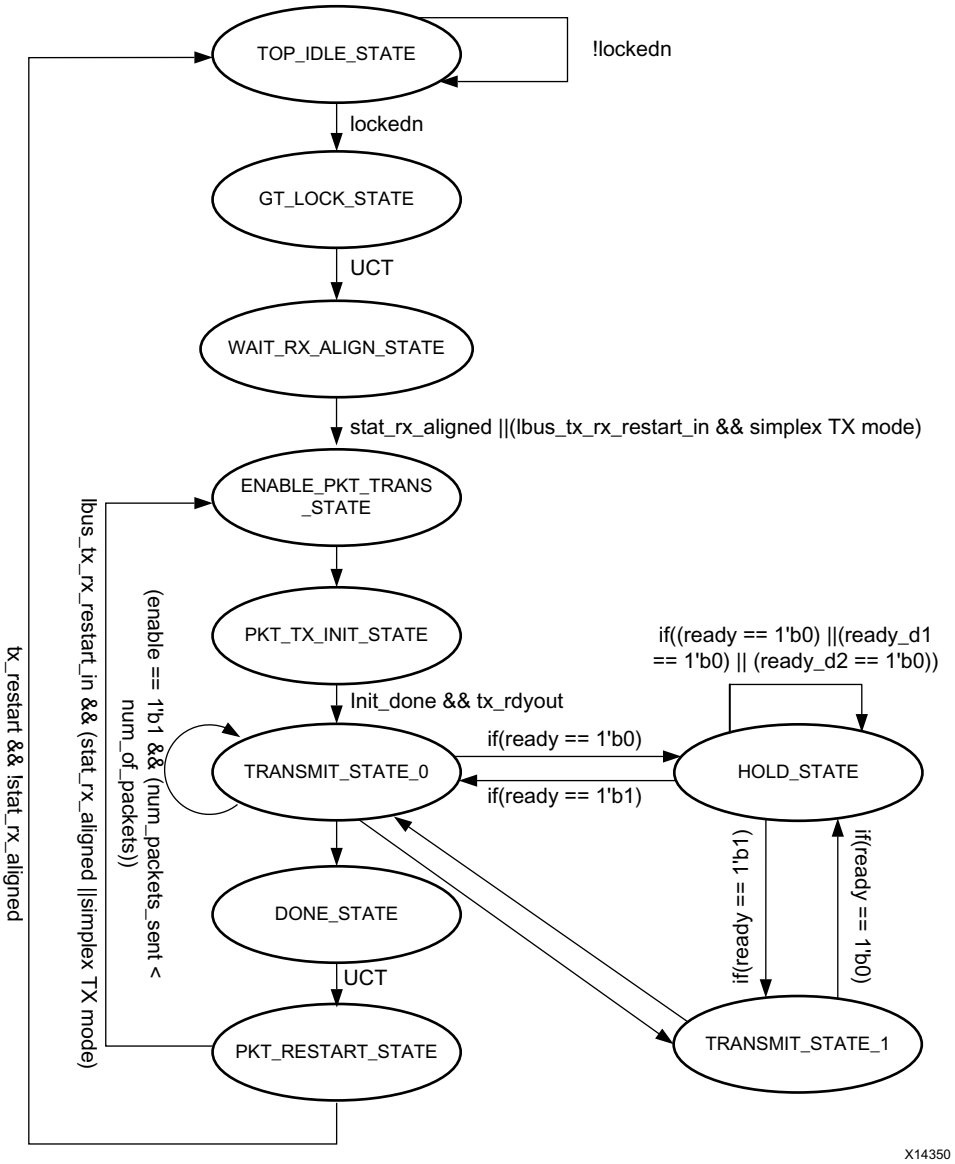

Figure 5-5: **State Transition Diagram for Packet Generator**

In the Simple TX mode of operation because RX alignment information is not available, the state machine waits for you to input `lbus_tx_rx_restart_in`. 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 5-5.

- **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_done_init = 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 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.

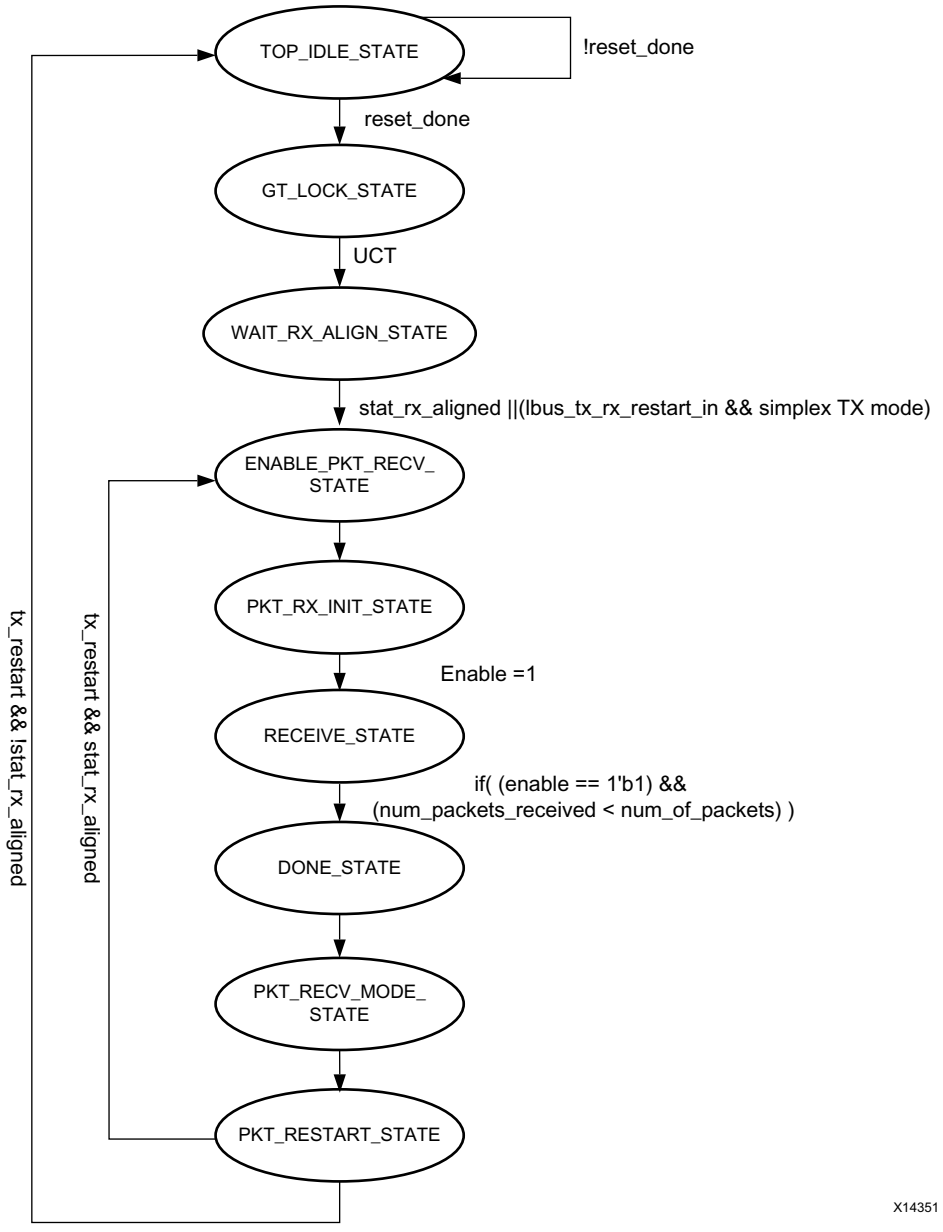

Figure 5-6: **State Transition Diagram for Packet Monitor**

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.

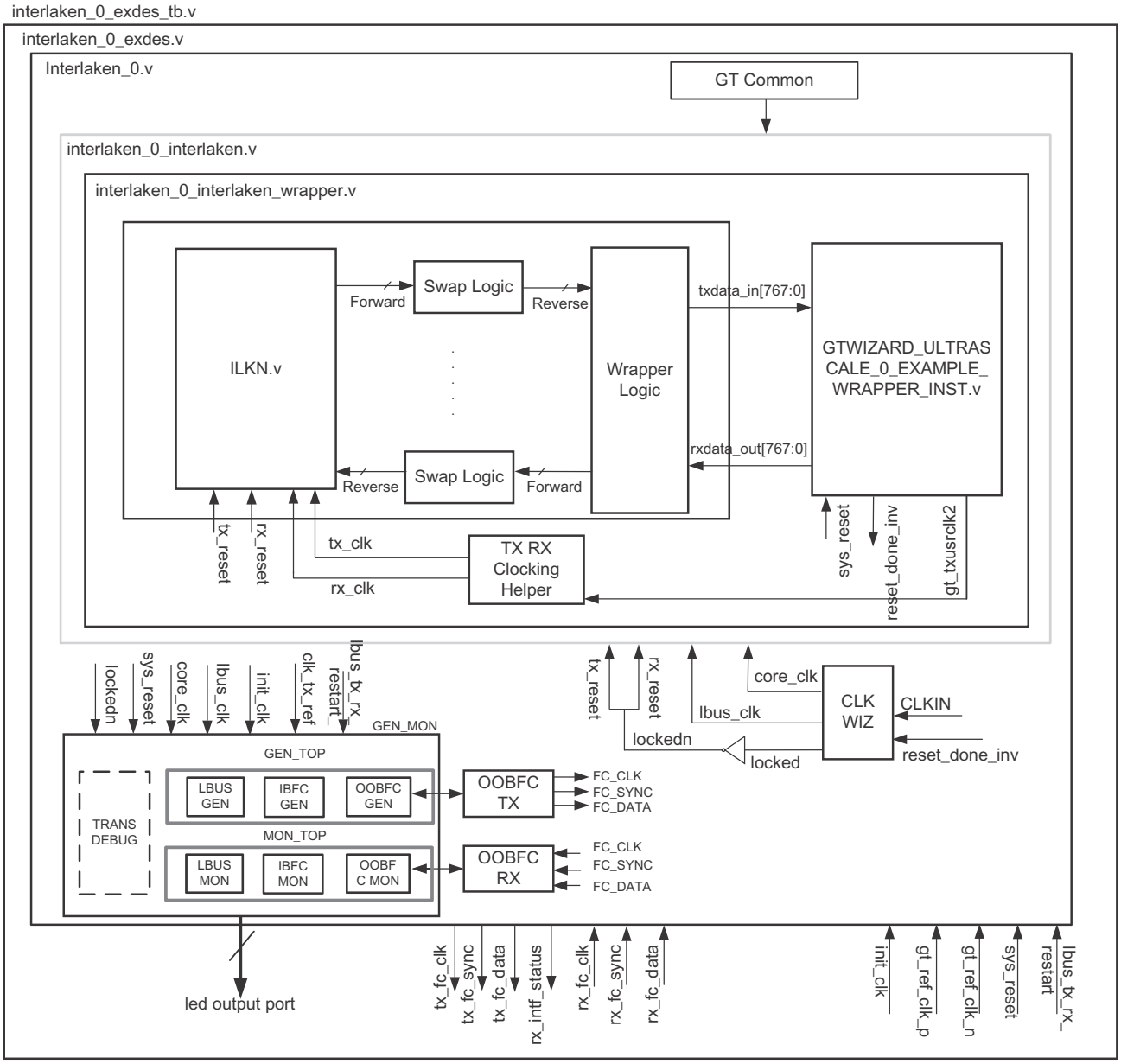

Figure 5-7: **Example Design Hierarchy with Shared Logic Implementation**

This feature provides the following advantages:

- Allow customers to share common blocks across multiple instantiations.
- Minimize the amount of HDL modifications required by customers.
- Still allow customizations beyond the normal use cases.
- Allow core upgrade without overwriting customer work.

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

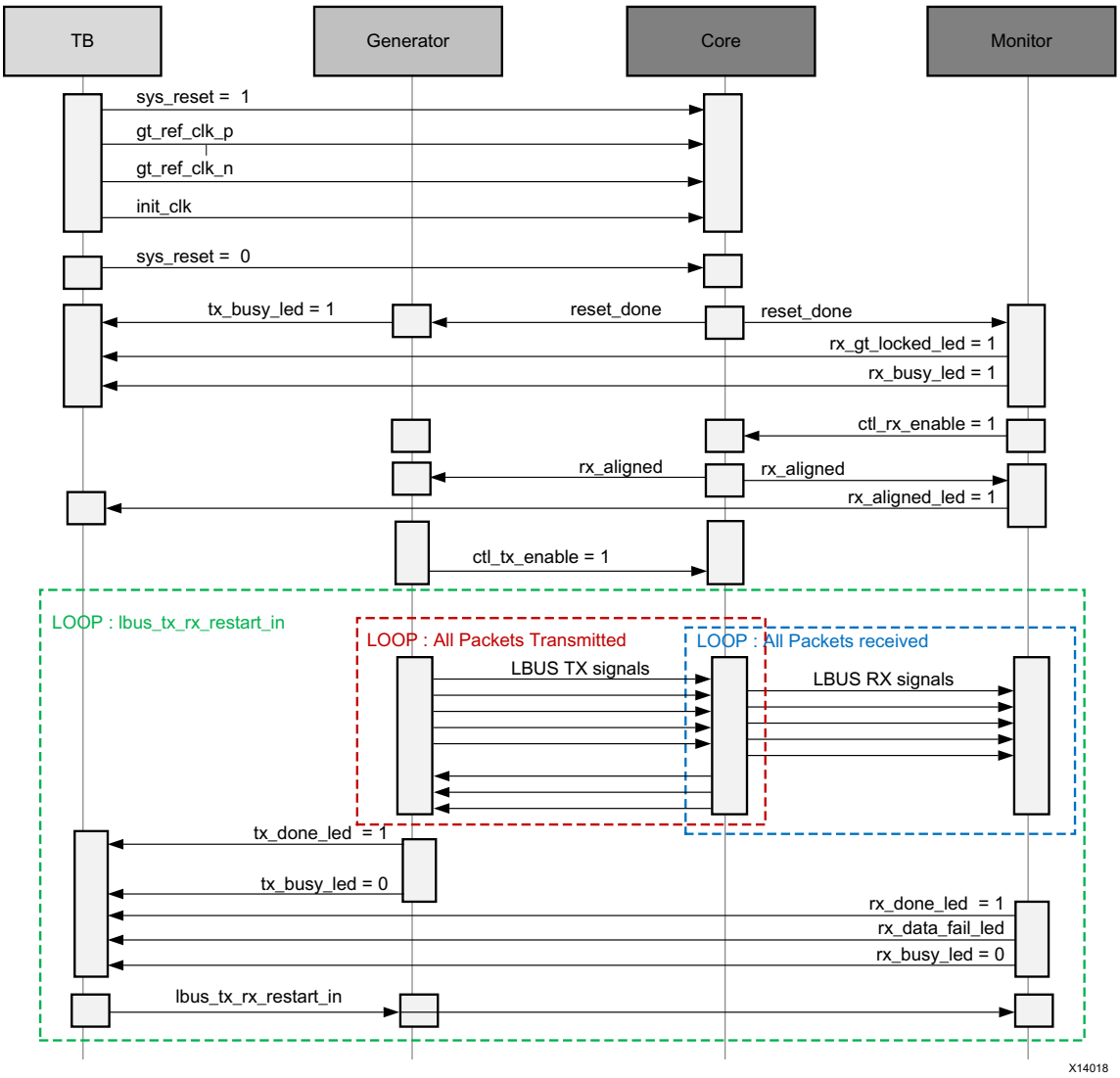

Figure 5-8: **Simulation Use Case for Duplex/Simplex Rx Configuration**

Simulation — Simplex TX Mode

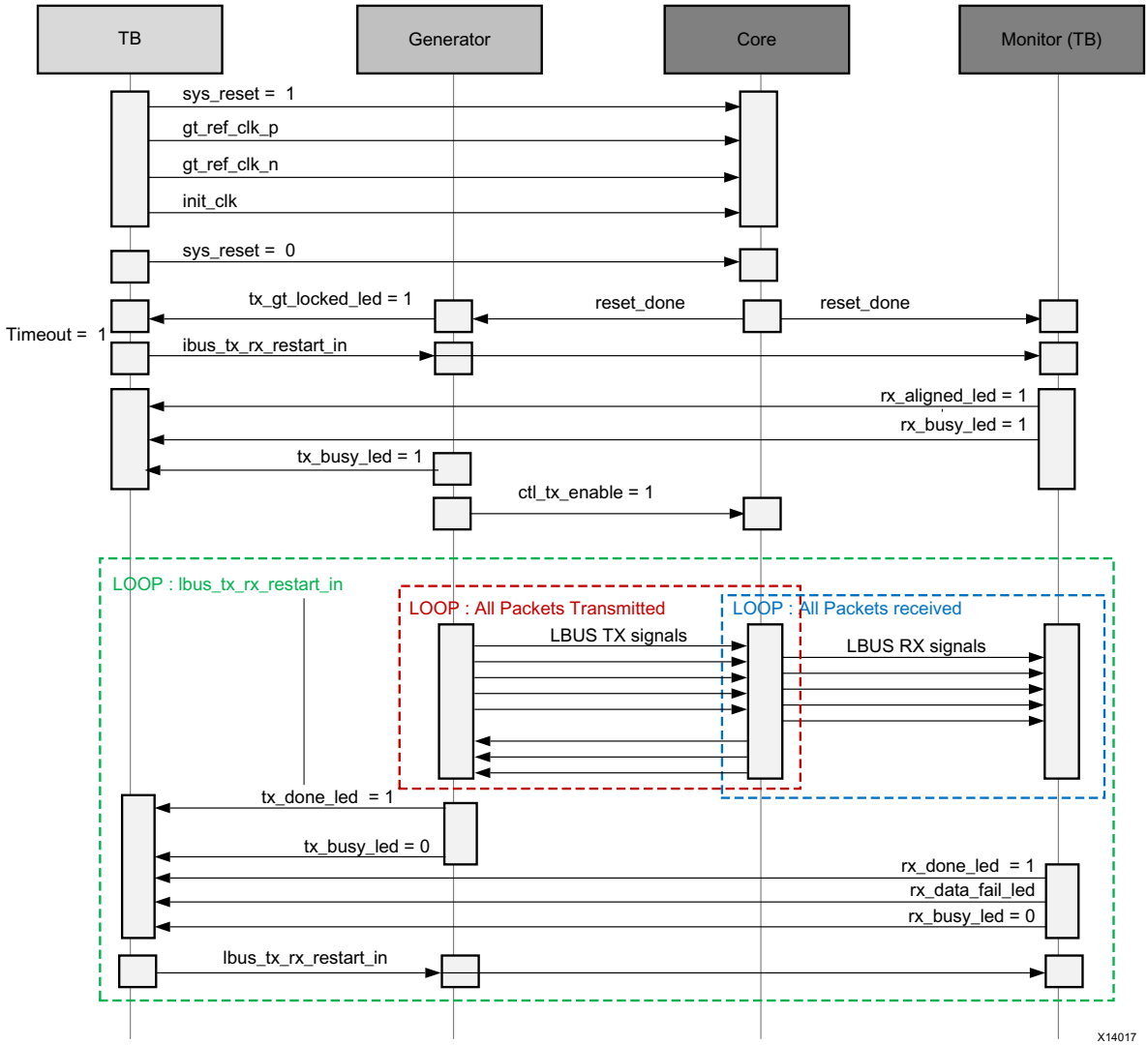

Figure 5-9: **Simulation Use Case for Simplex Tx Configuration**

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 Duplex/Simplex RX Mode

Figure 5-10 describes the LED behavior and input switch condition for the validation of the Interlaken core for passing scenario. Where the Generator generates and Monitor onboard for Duplex/Simplex RX mode configuration.

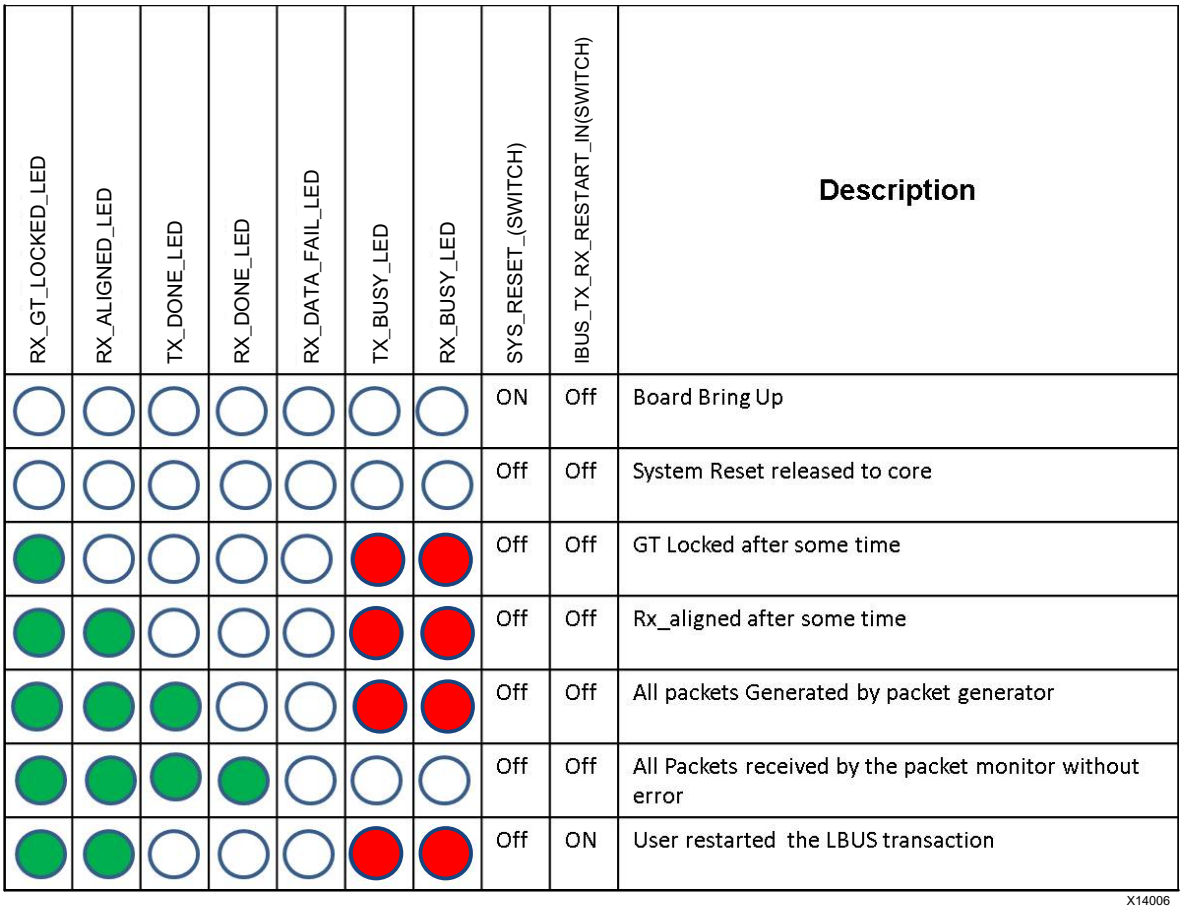

Figure 5-10: **Board Validation for Duplex/Simplex Rx Configuration - Passing Scenario**

Validation — Failing Scenario Duplex/Simplex RX Mode

RX_GT_LOCKED_LED	RX_ALIGNED_LED	TX_DONE_LED	RX_DONE_LED	RX_DATA_FAIL_LED	TX_BUSY_LED	RX_BUSY_LED	SYS_RESET_(SWITCH)	IBUS_TX_RX_RESTART_IN(SWITCH)	Description
○	○	○	○	○	○	○	ON	Off	Board Bring Up
○	○	○	○	○	○	○	Off	Off	System Reset released to core
●	○	○	○	○	●	●	Off	Off	GT Locked after some time
●	●	○	○	○	●	●	Off	Off	Rx_aligned after some time
●	●	●	○	○	●	●	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

Figure 5-11: **Board Validation for Duplex/Simplex Rx Configuration - Failing Scenario**

Validation — Simplex TX Mode

Figure 5-12 and Figure 5-13 describes the LED behavior and input switch condition for the validation of the Interlaken core onboard for Simplex TX mode configuration.

Validation — Passing Scenario Simplex TX Mode

tx_GT_LOCKED_LED	rx_ALIGNED_LED	tx_DONE_LED	tx_BUSY_LED	sys_RESET_(SWITCH)	lbus_tx_rx_restart_in(Switch)	Description
○	○	○	○	ON	Off	Board Bring Up
○	○	○	○	Off	Off	System Reset released to core
●	○	○	●	Off	Off	GT Locked after some time
●	●	○	●	Off	ON	User has to decide when the generator has to start packet generation
●	●	●	○	Off	Off	All packets Generated by packet generator
●	●	○	●	Off	ON	User restarted the LBUS transaction

X14008

Figure 5-12: **Board Validation for Simplex TX configuration - Passing Scenario**

Validation — Failing Scenario Simplex TX Mode

TX_GT_LOCKED_LED	RX_ALIGNED_LED	TX_DONE_LED	TX_BUSY_LED	SYS_RESET_(SWITCH)	IBUS_TX_RX_RESTART_IN(SWITCH)	Description
○	○	○	○	ON	Off	Board Bring Up
○	○	○	○	Off	Off	System Reset released to core
●	○	○	●	Off	Off	GT Locked after some time
●	●	○	●	Off	ON	User has to decide when the generator has to start packet generation
●	●	●	○	Off	Off	Packets generation finished due to tx_ovfout / tx_unfout
●	●	○	●	Off	ON	User restarted the LBUS transaction

X14009

Figure 5-13: **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 SIM (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 **Questa Sim/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.*

Migrating and Upgrading

This appendix contains information about migrating a design from the ISE® Design Suite to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information about migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 5].

Upgrading in the Vivado Design Suite

There is no upgrade information for this release of the core.

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 IP Catalog. The calendar length is also selected there.

Figure B-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.

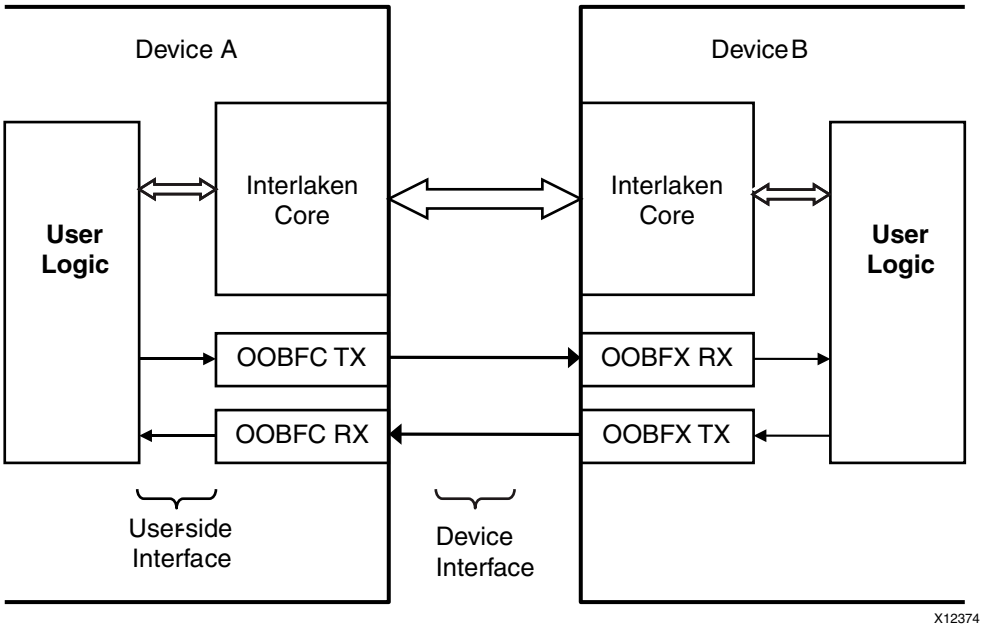

Figure B-1: **Typical Instantiation of the Interlaken and OOBFC Modules**

Port List

Table B-1 and Table B-2 show the port list of the TX_OOBFC and RX_OOBFC modules along with a description of each pin respectively. The LANES parameter is equal to the number of SerDes lanes. The MAX_CAL parameter is set to the maximum number of calendar entries supported by the core, up to a maximum value of 1024.

Table B-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.
resetn	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 B-1: **TX_OOBFC Pin List *(Cont'd)***

Name	Direction	Description
User-Side Interface - TX_OOBFC Signals		
tx_fc[MAX_CALLEN-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_CALLEN 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[7:0]	Input	This asynchronous input sets the calendar length for flow control information for the TX_OOBFC. Allowed values are 0 to MAX_CALLEN-1. For example, when MAX_CALLEN is set to 16, `tx_callen_minus1[7:0]` can be set to any value between 0 and 15. 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 `resetn` 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[LANES_LOG2-1:0]	Input	This input sets the number of lanes to be included in the Status Message. Allowed values are 0 to LANES-1. 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 `resetn` 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[LANES-1:0]	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	This signal is asserted if the OOBFC clocks are set incorrectly.
tx_update	Output	Indicates that the inputs `tx_fc`, `tx_callen_minus1`, `tx_lanes_minus1`, `tx_intf_status`, and `tx_lane_status` will be latched at the completion of the current clock cycle. This output is synchronized to the positive edge of `clk`.

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

Name	Direction	Description
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.

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

Table B-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.
resetn		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.
resetn _RX_FC_CLK	Input	Active-High, synchronous reset input. This signal only appears on the port list if the RX_OOBFC is implemented with flip-flips with synchronous reset inputs. This signal must remain asserted until after several clock cycles the `RX_FC_CLK` input.

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

Name	Direction	Description
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_CALLEN 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_CALLEN 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`.
rx_lane_status[LANES-1:0]	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 [3|2|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[127:8]` were updated, and so on. This bus is synchronized to the positive edge of `clk`.
rx_callen_minus1[7:0]	Output	Indicates the length of the most recently received calendar. This bus is synchronized to the positive edge of `clk`.

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

Name	Direction	Description
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.

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 B-2 shows the relative timing between the signals

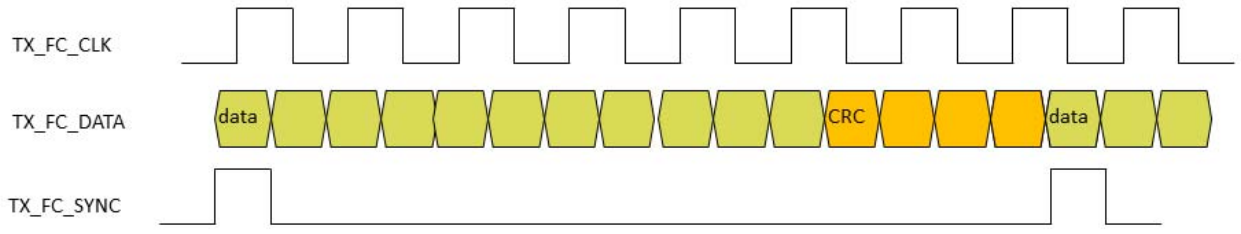

Figure B-2: **OOBFC Signals**

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 resetn input is deasserted. The clk_tx_ref input must be stable and running correctly before the resetn 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 inputs tx_fc, tx_callen_minus1, tx_lanes_minus1, tx_intf_status, and tx_lane_status are latched at the completion of the current clock cycle when tx_update is asserted.

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, resetn, 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 resetn 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_CALLEN. 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.

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 (www.xilinx.com/support) 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 Design Tools tab on the Downloads page (www.xilinx.com/download). 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

Contacting Technical Support

Xilinx provides technical support at www.xilinx.com/support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Additional Resources.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file(s) to include with the WebCase.

Note: Access to WebCase is not available in all cases. Log in to the WebCase tool to see your specific support options.

Vivado Lab Tools

Vivado® lab tools insert logic analyzer and virtual I/O cores directly into your design. Vivado lab tools also allow you to set trigger conditions to capture application and integrated core 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 9].

Simulation Debug

Slow Simulation

Simulations may appear to run slowly under some circumstances. If a simulation is unacceptably slow, the following suggestions may 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 may 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.

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.
- If your outputs go to 0, check your licensing.

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.

It is assumed that the integrated IP core for Interlaken has already passed all simulation testing that are implemented in hardware. This is a pre-requisite for any kind of hardware debug.

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 `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: *Note that 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 may 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 match the configuration requested when the IP core was ordered. The core clock will have a minimum frequency associated with it. The maximum core clock frequency will be determined by timing constraints. The minimum core clock frequency is derived from the required Interlaken bandwidth plus some margin reserved for clock tolerance, wander, and jitter.

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.

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 asserted, you must stop writing to the TX LBUS until the signal is de-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 simply ensure that TX writes are stopped when `tx_rdyout` is asserted.*

The level at which `tx_rdyout` becomes asserted is determined by the `ctl_tx_rdyout_thresh` bus. It may be necessary to adjust this bus as required in order to ensure that your transmitting protocol is able to handle the `tx_rdyout` requirement.

In the event that `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.

When a burst transaction has begun in the TX direction, it must continue until completion or there may be a buffer underflow as indicated by the `stat_tx_underflow_err` signal. This must not be allowed to occur.

IMPORTANT: *Bursts must be written on the TX LBUS without interruption.*

If `stat_tx_underflow_err` is ever asserted, debugging must stop until the condition which caused the underflow is addressed.

Burst Errors

The TX must observe the LBUS rules which are required for compliance to the Interlaken Protocol Definition and the Xilinx implementation-specific requirements.

In the event that a burst rule violation has occurred, the `stat_tx_burst_err` signal will be asserted. You must ensure that the rules governing the scheduling of bursts are observed according to the LBUS width in the particular implementation. In particular, the segmented LBUS protocol should be given careful reading.

One common problem 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 EoP 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` signal 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 these cases do not occur during the same clock cycle.*

RECOMMENDED: *It is recommended (and required for segmented LBUS) that you implement Enhanced Scheduling, as described in Enhanced Scheduling, page 47 and Interlaken Protocol Definition [Ref 1], as a means to avoid some of the implied Burst Count Words so that two BCWs in one clock cycle do not occur.*

RX Debug

The RX User Side interface indicates an error condition by means of the `rx_errout` signal.

If `rx_errout` 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` indication. The user 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

In order 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 may 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

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 problem.

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 D

Additional Resources and Legal Notices

Xilinx Resources

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

References

These documents provide supplemental material useful with this product guide:

1. *Interlaken Protocol Definition* (Revision 1.2, October 7, 2008)
2. *Virtex UltraScale Architecture Data Sheet: DC and AC Switching Characteristics* (DS893)
3. *UltraScale FPGAs Transceiver Wizards* (PG182)
4. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
5. *Vivado Design Suite User Guide: Designing with IP* (UG896)
6. *Vivado Design Suite User Guide: Getting Started* (UG910)
7. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
8. *Vivado Design Suite User Guide: Using Constraints* (UG903)
9. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
10. *ISE to Vivado Design Suite Migration Guide* (UG911)
11. *Vivado Design Suite User Guide: Implementation* (UG904)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
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.
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

application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.