LogiCORE IP SPI-3 Link Layer v7.3

DS504 December 14, 2010 **Product Specification**

Introduction

The Xilinx LogiCORE™ IP SPI-3 Link Layer core provides a complete, pre-engineered solution that is fully compatible with the *OIF-SPI3-01.0 System Packet Interface Level-3* implementation agreement. This fully verified solution implements the SPI-3 Link Layer interface, which interconnects with SPI-3 Physical (PHY) Layer devices.

Features

- Configurable interface data widths: 32-bit, 16-bit, and 8-bit
- Byte-level and packet-level transmit flow control configuration options
- Supports 1 to 256 addressable channels
- Fully parameterizable internal FIFO: Depth ranges from 16 to 4096 entries implemented in user-selectable block RAM or distributed memory
- LocalLink user interface allows easy interconnection to other LocalLink compliant interfaces
- Configurable Transmit Calendar implementation supporting up to 512 locations
- Greater than 125 MHz supported in Virtex® families; greater than 104 MHz supported in Spartan® families

LogiCORE IP Facts					
Core Specifics					
Supported Device Family	Virtex-6 (1), Spartan-6 (2),Virtex-5, Virtex-4, Spartan-3, Spartan-3A/3AN, Spartan-3E				
Resources Used (3)	I/O	LUTs	FFs	Block RAMs	LUT/ Flop Pair
Tx Core (8-bit)	24	220	306	1 (36k BRAM) 1 (18k BRAM)	398
Tx Core (32-bit)	50	229	390	1 (36k BRAM) 2 (18k BRAM)	468
Rx Core (8-bit)	16	117	213	1 (36k BRAM)	245
Rx Core (32-bit)	42	128	252	1 (36k BRAM) 1 (18k BRAM)	292
Provided with Core					
Documentation	Product Specification Getting Started Guide				
Design File Formats	VHDL and Verilog				
Constraints File	.ucf (user constraints file)				
Verification	VHDL and Verilog Test Bench				
Instantiation Template	VHDL and Verilog Wrapper				
Design Tool Requirements					
Xilinx Implementation Tools	ISE 12.4				
Verification	Mentor Graphics ModelSim v6.5c				
Simulation	Mentor Graphics ModelSim v6.5c Cadence Incisive Simulator (IES) v9.2 Synopsys VCS and VCS MX 2009.12				
Synthesis	XST Synplicity Synplify Pro				
Support					
Provided by Xilinx, Inc.					

1. Includes Virtex-6-1L devices
2. -3 speed grade and faster devices
3. Resources statistics are for a core configured with packet-level transfer control and a 512-deep block RAM FIFO.

Applications

The SPI-3 Link Layer core enables the interconnection of *link* layer devices to *physical* layer devices. Figure 1 illustrates the SPI-3 Link Layer core in a typical link layer application.

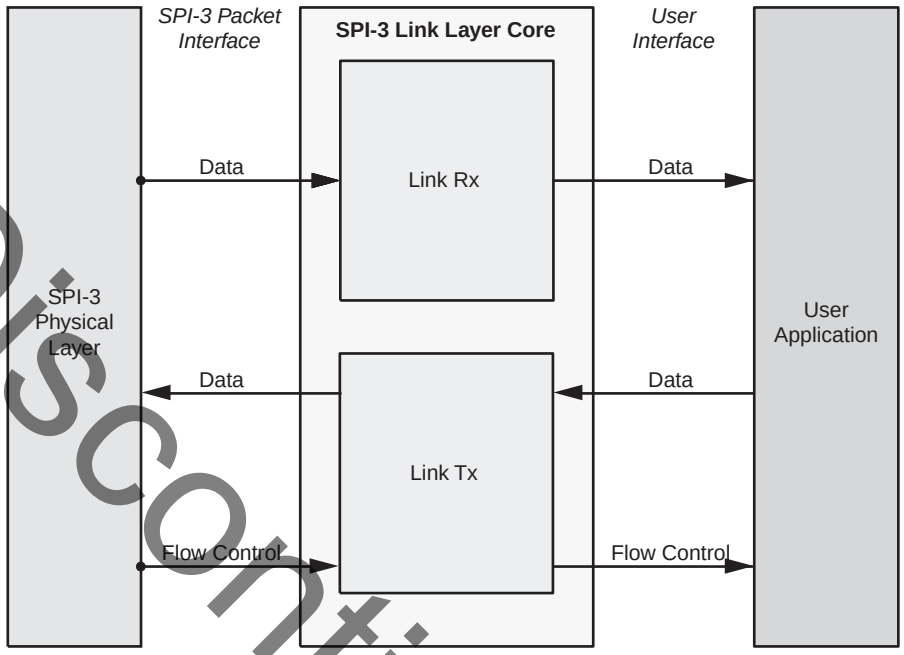

Figure 1: **SPI-3 Link Layer Block Diagram**

Feature Summary

The SPI-3 Link Layer Receive (Rx) and Transmit (Tx) core features are defined below.

Receive Core

- **Data Bus Width**: The data bus width supported by the SPI-3 Link Layer core includes an 8-bit and 32-bit data path implementation for *OIF-SPI3-01.0* compliance. An additional 16-bit data path implementation is provided for compatibility beyond the *OIF-SPI3-01.0* standard.
- **Parity Error Checking**: A parity check is performed independently on the address and the data received on the SPI-3 Receive Packet Interface. Any parity mismatches are detected and flagged on an error bus on the Receive User Interface.
- **Optionally Independent Clock Domains**: The Receive core is provided with the option of using independent clock domains on the SPI-3 Receive Packet Interface and the Receive User Interface.

Transmit Core

- **Data Bus Width**: The data bus width supported by the SPI-3 Link Layer core includes an 8-bit and 32-bit data path implementation for *OIF-SPI3-01.0* compliance. An additional 16-bit data path implementation is provided for compatibility beyond the *OIF-SPI3-01.0* standard.
- **Byte-level and Packet-level Transmit Control Modes**: A transfer control interface is provided on the Transmit User Interface. Byte-level mode lets users have direct access to the Transmit Packet Available status of all supported port addresses. In packet-level mode, the Transmit Packet

Available status is transmitted over a single signal. The order in which the flow control information appears is determined by programming a calendar.

- **1 to 256 Supported Port Addresses**: 1 to 256 port addresses are supported through the CORE Generator™ graphical user interface (GUI), enabling a wide range of flow control information to be reported. The generated core is optimized based on the number of addresses required.
- **Optionally Independent Clock Domains**: The Transmit core is provided with the option of using independent clock domains on the SPI-3 Transmit Packet Interface and the Transmit User Interface.
- **Maximum Burst Size**: The Transmit core implements a maximum burst size of 256 bytes in a manner transparent to the User Interface. The Transmit core breaks up transfers larger than 256 bytes by asserting the start of transfer signal and reasserting the address.

Functional Overview

The SPI-3 Link Layer core consists of two separate modules: the Receive core and the Transmit core. The SPI-3 Link Layer *Receive* core implements the SPI-3 Receive Packet Interface and is responsible for processing data received on the RDAT signal; the SPI-3 Link Layer *Transmit* core implements the SPI-3 Transmit Packet Interface and is responsible for generating data transmitted on the TDAT signal.

Receive Core

The Receive (Rx) core receives 32-bit, 16-bit, or 8-bit data from the SPI-3 Receive Packet Interface (SPI-3 Packet Interface) and forwards these data words onto the Receive User Interface (User Interface). In addition to data processing and formatting, the Receive core is responsible for error detection.

The Receive core has two primary interfaces: the SPI-3 Packet Interface and the User Interface. Figure 2 illustrates the Receive core input and output signals, each of which is defined in Receive Core Interfaces, page 6.

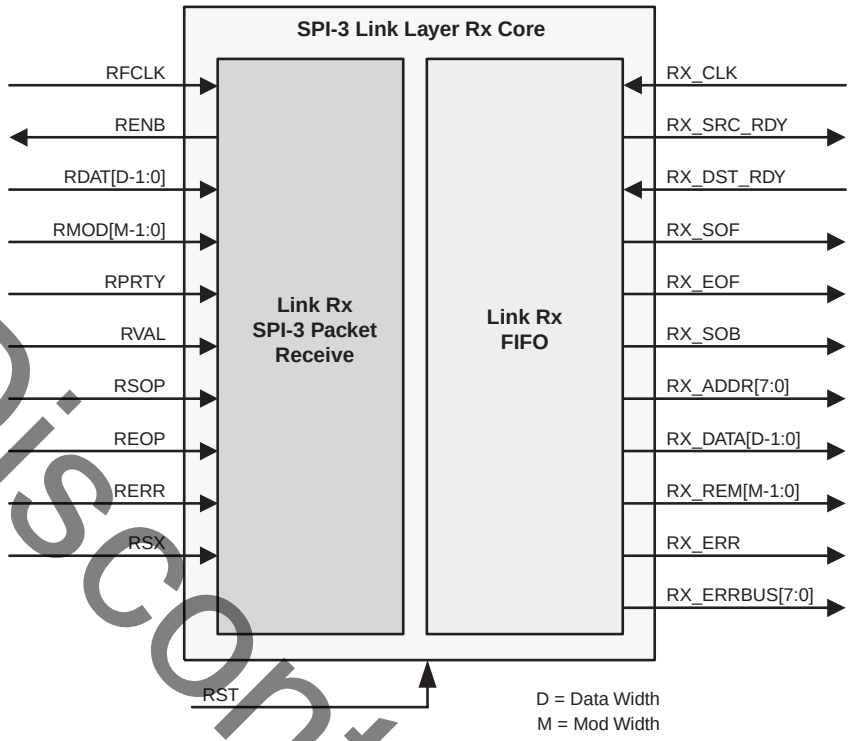

Figure 2: **SPI-3 Link Layer Receive Core**

Transmit Core

The Transmit (Tx) core transmits 32-bit, 16-bit, or 8-bit data on the SPI-3 Transmit Packet Interface (SPI-3 Packet Interface) by processing and formatting 32-bit, 16-bit, or 8-bit data words from the Transmit User Interface (User Interface). In addition to data processing and formatting, the Transmit core is responsible for monitoring flow control signals.

The Transmit core has two primary interfaces: the SPI-3 Packet Interface and the User Interface. In addition to data processing interfaces, the Transmit core provides an additional Transmit Packet Available Interface on the SPI-3 Packet Interface, as well as Flow Control and a Calendar Interface on the User Interface. Figure 3 displays the Transmit core input and output signals, each of which is defined in Transmit Core Interfaces, page 12.

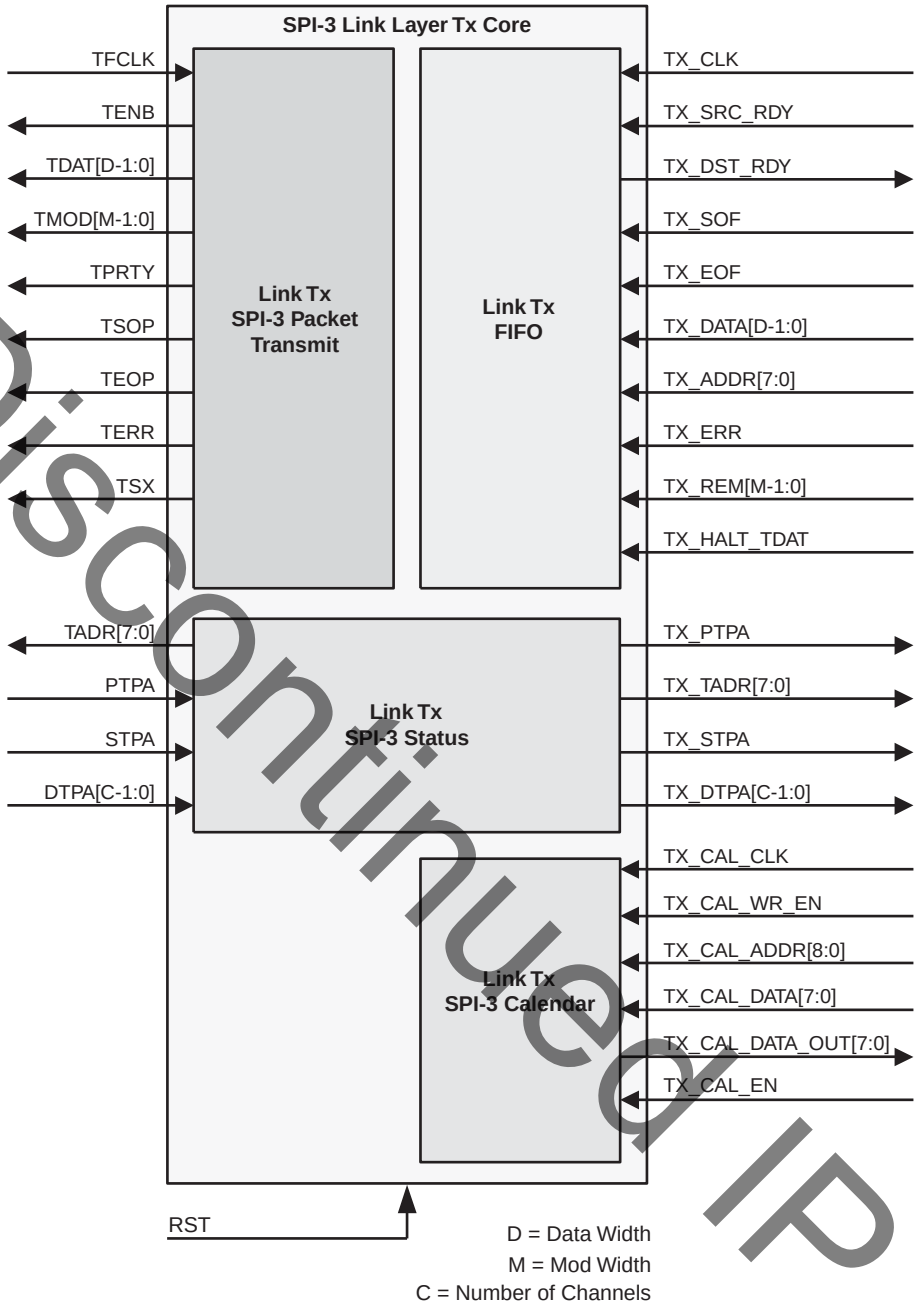

Figure 3: **SPI-3 Link Layer Transmit Core**

Core Interfaces

This section describes the SPI-3 architecture and defines the interface signals on the Receive and Transmit cores.

- **User Interface**: On both the Receive and Transmit cores, the User Interface implements the Xilinx LocalLink standard, providing a simple, flexible way to transmit and receive packets. The User Interface consists of a unidirectional data bus with control signals that delineate packet boundaries and allow the user application and the core to stall a data transfer.
- **SPI-3 Packet Interface**: On both the Receive and Transmit cores, the SPI-3 Packet Interface connects to the reciprocal SPI-3 Packet Interface on the PHY Layer device.

Receive Core Interfaces

The Receive core receives data from the SPI-3 Packet Interface and processes the data for transmission on the User Interface. The SPI-3 Packet Interface and the User Interface operate on optionally independent clock domains.

SPI-3 Packet Interface

Table 1 defines the SPI-3 Packet Interface signals on the Receive core.

Table 1: **Receive SPI-3 Packet Interface Signals**

Signal	Direction	Description
RFCLK	IN	**Receive Clock**: Used to synchronize data transfer transactions between the Link Layer device and the PHY Layer device. All SPI-3 Packet Interface signals are synchronous to the rising edge of this clock.
RVAL	IN	**Receive Data Valid**: Indicates the validity of the receive data signals. RVAL is deasserted between transfers and when RSX is asserted; and also when the PHY pauses a transfer. When a transfer is paused by deasserting RENB, RVAL holds its value unchanged, although no new data will be present on RDAT until the transfer resumes. • When RVAL is asserted, RDAT, RMOD, RSOP, REOP, and RERR are valid. • When RVAL is deasserted, RDAT, RMOD, RSOP, REOP, and RERR are invalid and disregarded. RSX is valid when RVAL is deasserted.
RENB	OUT	**Receive Read Enable**: Used to control the flow of data to the Receive core. The RENB signal may be deasserted at any time if the Link Receive core is unable to accept data from the SPI-3 Packet Interface. • When RENB is asserted, RDAT, RPRTY, RMOD, RSOP, REOP, RERR, RSX, and RVAL are updated on the following rising edge of RFCLK. • When RENB is deasserted RDAT, RPRTY, RMOD, RSOP, REOP, RERR, RSX, and RVAL remain unchanged on the following rising edge of RFCLK. Note that RENB asserts low.
RDAT[D-1:0]	IN	**Receive Packet Data Bus**: The RDAT bus carries the packet data as well as the in-band port address of the packet data. Data will be transmitted in big endian order on RDAT. For example, when the width of the data bus is 32-bits, bit 31 is transmitted first and bit 0 is transmitted last.
RPRTY	IN	**Receive Parity**: Indicates the parity calculated over the RDAT bus. The Receive core checks for odd parity.
RMOD[M-1:0]	IN	**Receive Word Modulo**: Indicates the number of valid bytes of data presented on RDAT. The RMOD bus should always be all zero, except during the last transfer of a packet on RDAT. When REOP is asserted, the number of valid packet data bytes on RDAT is specified by RMOD. When an 8-bit interface is used, the RMOD signal is not present.

Table 1: **Receive SPI-3 Packet Interface Signals *(Cont'd)***

Signal	Direction	Description
RSOP	IN	**Receive Start of Packet**: Used to delineate the packet boundaries on the RDAT bus. When RSOP is asserted, the start of the packet is present on the RDAT bus.
REOP	IN	**Receive End of Packet**: Used to delineate the packet boundaries on the RDAT bus. When REOP is asserted, the end of the packet is present on the RDAT bus. When a 32-bit or 16-bit interface is used, RMOD indicates the number of valid packet bytes on RDAT when REOP is asserted.
RERR	IN	**Receive Error Indicator**: Indicates that the current packet is in error. RERR must only be asserted when REOP is asserted.
RSX	IN	**Receive Start of Transfer**: Indicates when the in-band port address is present on the RDAT bus. When RSX is asserted, the value of RDAT[7:0] is the address of the packet data. Subsequent data transfers on the RDAT bus will be from the port specified by this in-band address. RSX will be asserted at the beginning of each transfer. When RSX is asserted, RVAL must be deasserted.

Figure 4 illustrates a sample operation of the SPI-3 Packet Interface receiving two complete packets. The first packet is initiated on the second rising edge of RFCLK, when the PHY Layer logic deasserts RVAL, and asserts RSX while providing the in-band port address of the current packet, A0, on RDAT[7:0]. For the following three clock cycles, packet data is transmitted by the PHY Layer to the Receive core by asserting RVAL and driving RDAT with valid packet data, D0, D1, and D2.

After the fifth clock cycle, the PHY Layer asserts REOP and signals RMOD appropriately to indicate the remaining valid data, D3, for packet in progress. The following cycle, a transfer to a new port address is initiated with the PHY Layer providing an in-band address of A1 and completing seven valid cycles of data transfer.

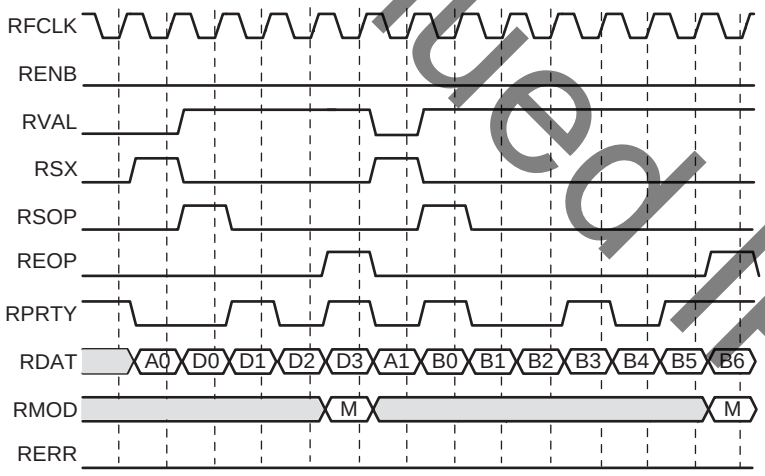

Figure 4: **SPI-3 Packet Interface: Receive Two Full Packets**

Figure 5 illustrates a sample operation of the SPI-3 Packet Interface receiving a complete packet utilizing the flow control signals. After RENB is asserted by the Receive core, the PHY Layer transmits 3 valid cycles on the SPI-3 Packet Interface. After transmitting data word D1, the PHY Layer stalls the SPI-3 Packet Interface for one cycle by deasserting the RVAL signal. The following clock cycle, RVAL is reasserted, and the Receive core continues receiving data. Following the successful transmission of D4,

the Receive core deasserts RENB due to an almost-full internal FIFO, which halts the PHY Layer from transmitting additional data. Two clock cycles later, RENB asserts again, and data transmission resumes one cycle later and continues until the end of the packet is transmitted.

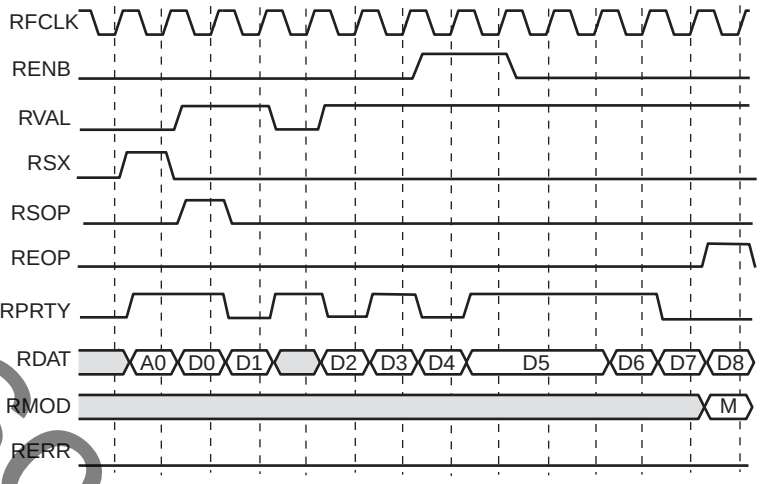

Figure 5: **SPI-3 Packet Interface: Receive Packet with Flow Control**

User Interface

Table 2 defines the User Interface signals on the Receive core. The User Interface connects to the user side of the system and implements the Xilinx LocalLink standard, providing a simple, flexible way to transmit packets.

Table 2: **Receive User Interface Signals**

Signal Name	Direction	Description
RX_CLK	IN	**Receive Clock**: All User Interface signals are synchronous to the rising edge of this clock.
RX_SRC_RDY	OUT	**Receive Source Ready**: Indicates a word presented by the Receive core is valid (not accepted until RX_DST_RDY is also asserted).
RX_DST_RDY	IN	**Receive Destination Ready**: Indicates a word presented by the Receive core will be accepted (if RX_SRC_RDY is also asserted).
RX_SOB	OUT	**Receive Start of Burst**: Indicates that the data on RX_DATA represented a new burst of data, determined by a change of the in-band address.
RX_SOF	OUT	**Receive Start of Frame**: Indicates that the data on RX_DATA represents the beginning of a frame. A RX_SOF indication is passed through as RSOP from the SPI-3 Packet Interface.
RX_EOF	OUT	**Receive End of Frame**: Indicates that the data on RX_DATA represents the end of a frame. A RX_EOF indication is passed through as REOP from the SPI-3 Packet Interface.
RX_ADDR[7:0]	OUT	**Receive Address**: Indicates the port address associated with the packet.
RX_DATA[D-1:0]	OUT	**Receive Data**: Packet data read from the SPI-3 Packet Interface.

Table 2: **Receive User Interface Signals *(Cont'd)***

Signal Name	Direction	Description
RX_REM[M-1:0]	OUT	**Receive Remainder**: Binary-encoded count of valid bytes in RX_DATA when RX_EOF is asserted; valid only when RX_EOF is asserted. Valid bytes begin with the most-significant byte of RX_DATA. The actual number of valid bytes is (RX_REM+1). When an 8-bit interface is used, the RX_REM bus is not present.
RX_ERR	OUT	**Receive Error**: When asserted, indicates that the packet contained an error. Valid only when RX_EOF is asserted.
RX_ERRBUS[7:0]	OUT	**Receive Error Bus**: Provides real-time error notification. Errors are held until the in-band address is changed, or a REOP was received on the SPI-3 Packet Interface. • [7:3] = reserved (tied to 0) • [2] = in-band address parity error • [1] = data parity error • [0] = RERR was indicated

Figure 6 illustrates a sample operation of the User Interface transmitting two complete packets. The first packet starts on the second rising edge of RX_CLK, when RX_SRC_RDY and RX_DST_RDY are asserted. The Receive core asserts RX_SOB to indicate that either the port address is changing or a new packet is starting. This signal can be used as an indicator to the user logic to switch FIFOs external to the Receive core.

The Receive core logic drives RX_SOF, RX_EOF, RX_ADDR, RX_DATA, and RX_REM appropriately during the five-clock cycle transfer. RX_SRC_RDY is not deasserted by the core in this example, allowing the user logic to run at full rate. The packet begins with an assertion of RX_SOF and ends four clock cycles later with an assertion of RX_EOF and a valid RX_REM. The value of RX_REM is translated appropriately from the RMOD signal on the SPI-3 Packet Interface. A second packet transfer then begins, with the Receive core reporting a port address change on RX_SOB. At the 10th and 11th rising clock edges RX_DST_RDY is deasserted by the user logic, which stalls the data transfer for those two cycles. Data transfer then resumes and the packet completes.

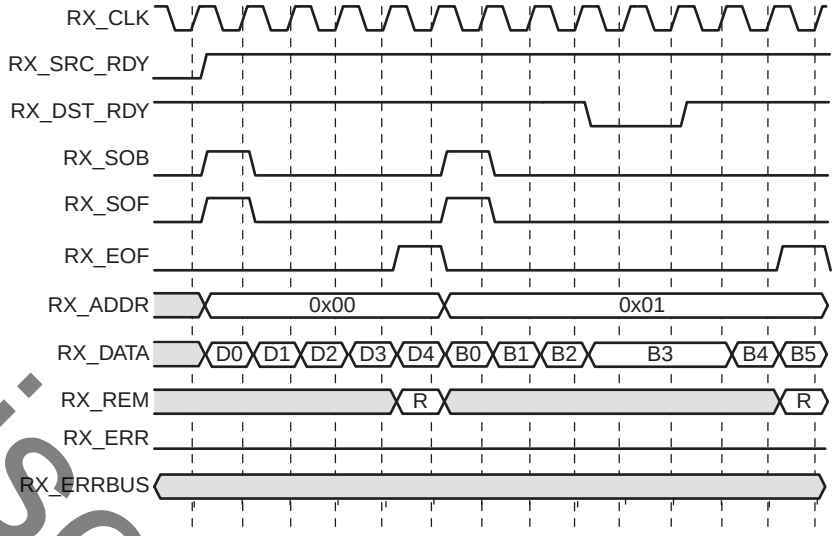

Figure 6: **User Interface: Transmit Two Full Packets**

Figure 7 illustrates a sample operation of the User Interface presenting two interwoven packets. The first packet (port address 0x00), starts and transmits for three clock cycles, after which the user logic switches to port address 0x01 and transmits for four cycles. The assertion of RX_SOB is used to indicate that the port address has changed. To finish the packets, the user logic switches back to the first packet, port address 0x00 and finishes the packet in two clock cycles; then switches to port address 0x01 and transmits for an additional four clock cycles, finishing the second packet.

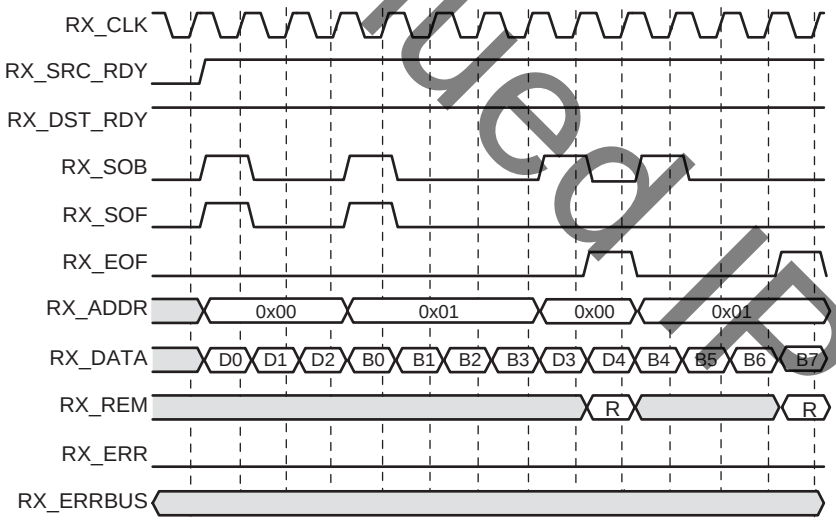

Figure 7: **User Interface: Transmitting Two Interwoven Packets**

Figure 8 illustrates a sample operation of the User Interface presenting two complete packets. The first packet indicates that RERR was flagged on the SPI-3 Packet Interface. The RX_ERRBUS is driven to 0x01 to indicate that the current packet received a RERR indication. RX_ERR is also asserted indicating that

RX_ERRBUS contains an error. When a channel changes, or after an assertion of RX_EOF, the value of RX_ERRBUS is reset to 0x00 and RX_ERR is deasserted.

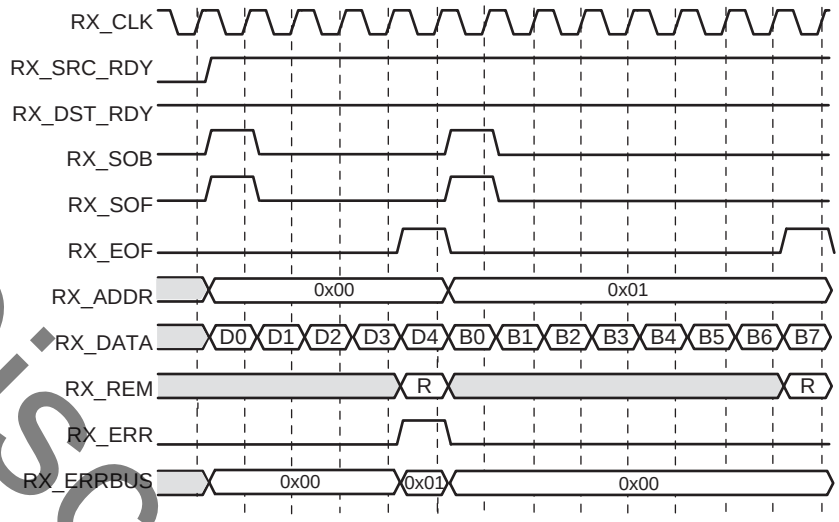

Figure 8: **User Interface: RERR indicated on RX_ERRBUS**

Figure 9 illustrates a sample operation of the User Interface presenting two complete packets. The first packet indicates that the packet had a data parity error. This error is indicated on the RX_ERRBUS as 0x02 and can be asserted at any point in time in a packet. After the parity error is detected, the error is held until after RX_EOF is asserted or the in-band port address has changed. The second packet further illustrates that the data parity error is held until after RX_EOF asserts or RX_ADDR changes.

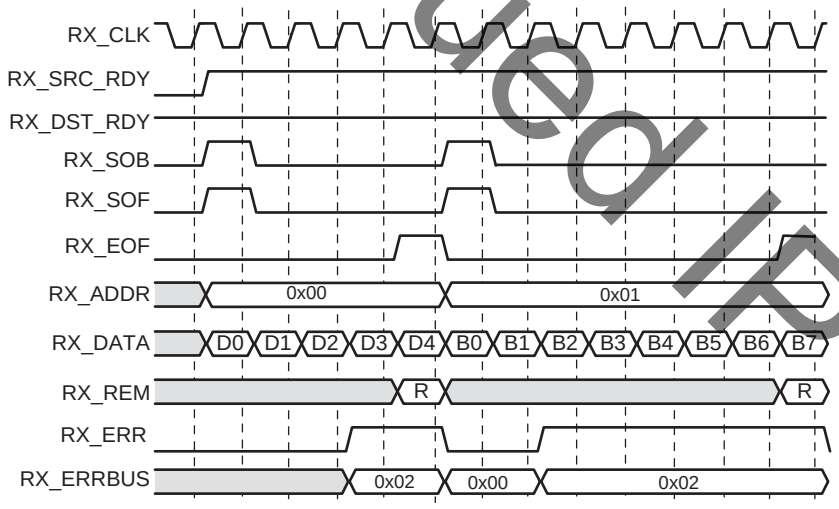

Figure 9: **User Interface: Data Parity Error Indicated on RX_ERRBUS**

Figure 10 illustrates a sample operation of the User Interface presenting two complete packets. The first packet indicates that the packet had an in-band port address parity error. This error is indicated on the RX_ERRBUS as 0x04 and is asserted for the length of the packet until either RX_EOF asserts or RX_ADDR changes.

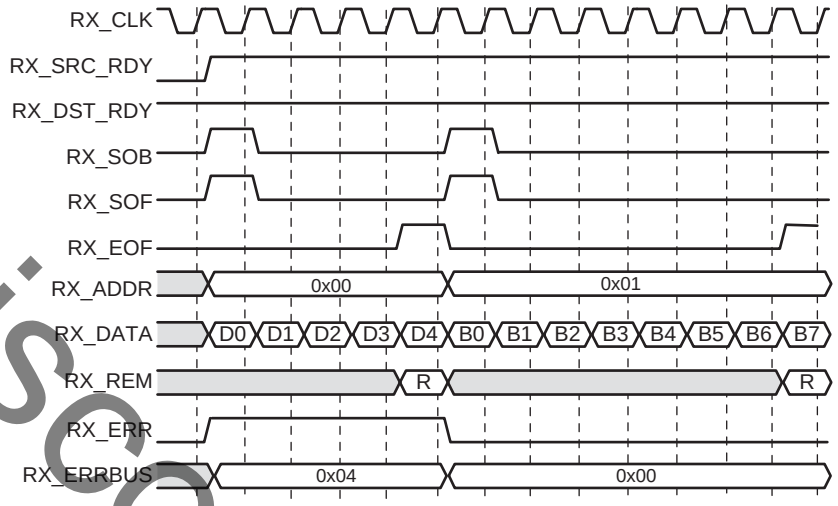

Figure 10: **User Interface: Address Parity Indicated on RX_ERRBUS**

Transmit Core Interfaces

The Transmit core receives data from the User Interface and processes the data for transmission on the SPI-3 Packet Interface. The SPI-3 Packet Interface and the User Interface operate on optionally independent clock domains.

SPI-3 Packet Interface

Table 3 defines the SPI-3 Packet Interface signals on the Transmit core.

Table 3: **Transmit SPI-3 Packet Interface Signals**

Signal Name	Direction	Description
TFCLK	IN	**Transmit Clock**: Used to synchronize data transfer transactions between the Link Layer device and the PHY Layer device. All SPI-3 Packet Interface signals are synchronous to the rising edge of this clock.
TENB	OUT	**Transmit Write Enable**: Indicates the validity of the transmit data signals. When TENB is deasserted, TDAT, TMOD, TSOP, TEOP, and TERR are invalid and should be ignored by the Physical Layer device. TSX is valid when TENB is deasserted. When TENB is asserted, TDAT, TMOD, TSOP, TEOP, and TERR are valid. When TENB is asserted, TSX should be ignored by the PHY Layer device. Note that TENB asserts low.
TDAT[D-1:0]	OUT	**Transmit Packet Data Bus**: The TDAT bus carries the packet data presented on the User Interface, as well as the in-band port address of the packet data. Data is transmitted in big endian order on TDAT. For example, when the width of the data bus is 32-bits, bit 31 is transmitted first and bit 0 is transmitted last.
TPRTY	OUT	**Transmit Parity**: Indicates the parity calculated over the TDAT bus. TPRTY is considered valid only when TENB or TSX is asserted. The Transmit core supports odd parity.

Table 3: **Transmit SPI-3 Packet Interface Signals *(Cont'd)***

Signal Name	Direction	Description
TMOD[M-1:0]	OUT	**Transmit Word Modulo**: Indicates the number of valid bytes of the data presented on TDAT. When TEOP is asserted, the number of valid packet data bytes on TDAT is specified by TMOD. When an 8-bit interface is used, the TMOD bus is not present.
TSOP	OUT	**Transmit Start of Packet**: Used to delineate the packet boundaries on the TDAT bus. When TSOP is asserted, the start of the packet is present on the TDAT bus.
TEOP	OUT	**Transmit End of Packet**: Used to delineate the packet boundaries on the TDAT bus. When TEOP is asserted, the end of the packet is present on the TDAT bus. When a 32-bit or 16-bit interface is used, TMOD indicates the number of valid packet bytes on TDAT when TEOP is asserted.
TERR	OUT	**Transmit Error Indicator**: Indicates that the current packet is in error. TERR will only be asserted when TEOP is asserted.
TSX	OUT	**Transmit Start of Transfer**: Indicates when the in-band port address is present on the TDAT bus. When TSX is asserted, the value of TDAT[7:0] is the address of the packet data. Subsequent data transfers on the TDAT bus will be from the port specified by this in-band address. If the address changes, TSX will be asserted at the beginning of the transfer. TSX will also interrupt a transfer and assert if the 256 byte maximum is reached, and it will then resume the transfer. TSX is considered valid only when TENB is deasserted.

Figure 11 illustrates a sample operation of the SPI-3 Packet Interface transmitting two complete packets. On the first rising clock edge, TENB is deasserted, halting the PHY Layer Transmit core from receiving any data. On the next clock cycle, TSX is asserted and TENB is deasserted, which indicates that the value of TDAT[7:0] is the value of the in-band port address of A0. In the following four clock cycles a complete packet is transmitted, and demarcated with a TSOP and TEOP. TENB is asserted throughout the packet transfer, indicating data is valid on each clock cycle. On the seventh clock cycle a new packet is started with an in-band port address of A1. The following seven clock cycles illustrate an additional data packet being transmitted.

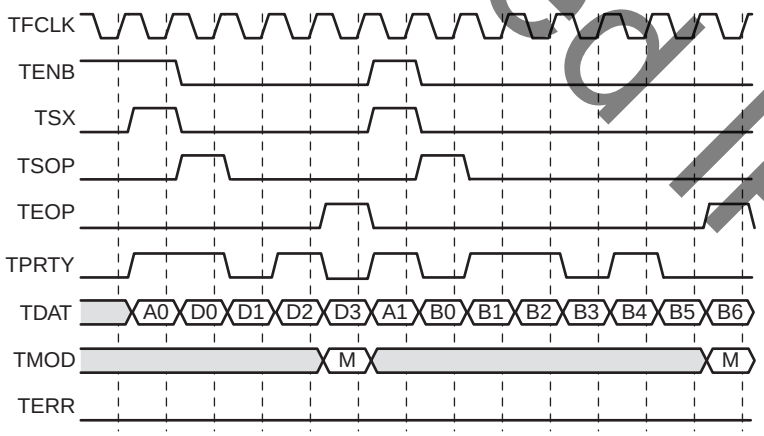

Figure 11: **SPI-3 Packet Interface: Transmitting Two Complete Packets**

Figure 12 illustrates a sample operation of the SPI-3 Packet Interface transmitting a complete packet utilizing the TENB flow control signal. A packet is sent out of the Transmit core similar to the sample operation shown in Figure 11; however, Figure 12 shows TENB deasserting during the eighth and ninth

rising clock edge, stalling the data transfer. On the 10th clock cycle, TENB asserts and continues transmission until the end of the packet is transmitted.

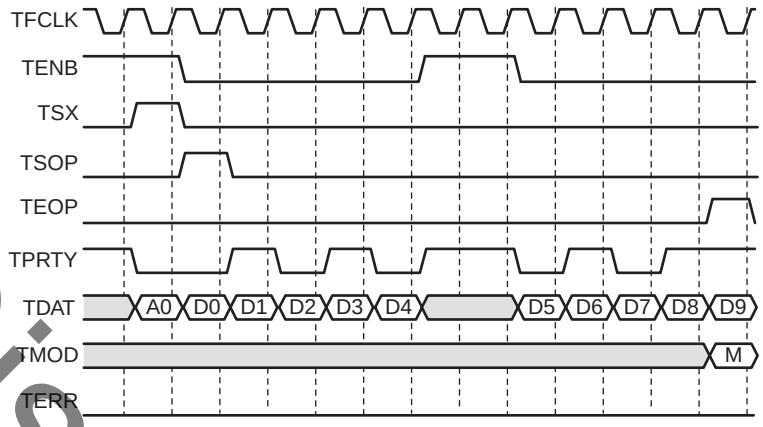

Figure 12: **SPI-3 Packet Interface: Transmitting Complete Packet with Flow Control**

User Interface

Table 4 defines the User Interface signals on the Transmit core. The User Interface connects to the link side of the system and implements the Xilinx LocalLink standard, providing a simple flexible way to transmit packets.

Table 4: **Transmit Core User Interface Signals**

Signal Name	Direction	Description
TX_CLK	IN	**Transmit Clock**. All User Interface signals are synchronous to the rising edge of this clock.
TX_SRC_RDY	IN	**Transmit Source Ready**: Indicates a word presented by the user is valid (not accepted until TX_DST_RDY is also asserted).
TX_DST_RDY	OUT	**Transmit Destination Ready**: Indicates a word presented by the user will be accepted (if TX_SRC_RDY is also asserted).
TX_SOF	IN	**Transmit Start of Frame**: Indicates that the data on TX_DATA represents the beginning of a frame. A TX_SOF indication is passed through as TSOP to the SPI-3 Packet Interface.
TX_EOF	IN	**Transmit End of Frame**: Indicates that the data on TX_DATA represents the end of a frame. A TX_EOF indication is passed through as TEOP to the SPI-3 Packet Interface.
TX_ADDR[7:0]	IN	**Transmit Address**: Indicates the port address associated with the packet. A change in TX_ADDR induces an in-band port address to be transmitted.
TX_DATA[D-1:0]	IN	**Transmit Data**: Carries the packet data associated with the transmit address.

Table 4: **Transmit Core User Interface Signals *(Cont'd)***

Signal Name	Direction	Description
TX_REM[M-1:0]	IN	**Transmit Remainder**: Binary-encoded count of valid bytes in TX_DATA when TX_EOF is asserted; valid only when TX_EOF is asserted. Valid bytes begin with the most-significant byte of TX_DATA. The actual number of valid bytes is (TX_REM+1). When an 8-bit interface is used, the TX_REM bus is not present.
TX_ERR	IN	**Transmit Error**: When asserted, indicates that the packet contained an error. Valid only when TX_EOF is asserted.
TX_HALT_TDAT	IN	**Transmit Halt Data:** When asserted, indicates that the Transmit core should halt data transfer on the SPI-3 Packet Interface. Data transfer is halted by deassertion of the TENB signal.

Figure 13 illustrates a sample operation of the User Interface sending two complete packets to the SPI-3 Packet Interface. Starting on the second rising clock edge, the TX_SOF is asserted to start a new packet, and TX_ADDR and TX_DATA are driven to the appropriate address and data values for use on the SPI-3 Packet Interface. Because TX_SRC_RDY and TX_DST_RDY are asserted by the user and the Transmit core, respectively, the cycle is accepted as valid. Clock cycles three through six show the user logic completing the packet and asserting TX_EOF with a remainder on TX_REM during the sixth clock cycle. On the following clock cycle, TX_SOF is asserted to begin a new packet, and TX_ADDR changes to a new port address. For rising clock edges 10 and 11, TX_SRC_RDY is deasserted by the user logic, to stall the SPI-3 Packet Interface, and data transfer resumes on the 12th rising clock edge.

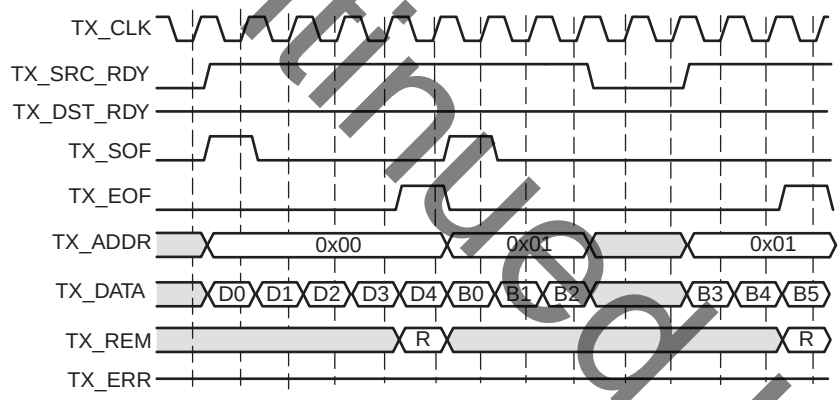

Figure 13: **User Interface: Transmitting Two Complete Packets**

Figure 14 illustrates a sample operation of the User Interface sending two interwoven packets to the SPI-3 Packet Interface. On the second rising clock edge, TX_SOF indicates the start of a new packet and TX_ADDR indicates a port address of 0x00. Three clock cycles later, TX_SOF is asserted again and TX_ADDR changes to indicate a port address of 0x01 Note that the first packet to port address 0x00 has not been terminated yet with a TX_EOF; there are now two outstanding un-terminated packets. On the ninth rising clock edge, TX_ADDR changes again to indicate that the Transmit core is to return to port address 0x00, and on the 10th cycle the remainder of this frame is transmitted and terminated by asserting TX_EOF with a remainder presented on TX_REM. On the eleventh rising clock edge, TX_ADDR changes one more time to indicate a port address of 0x01. The remainder of this frame is sent by the user logic in the following three clock cycles.

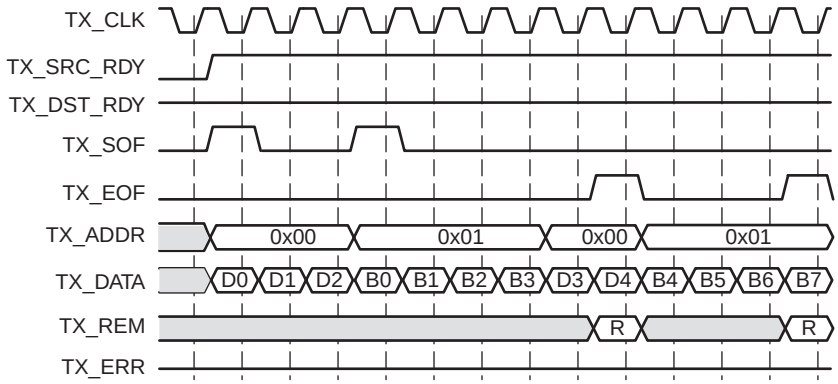

Figure 14: **User Interface: Transmitting Two Interwoven Packets**

Transmit Packet Available Interface

Table 5 defines the Transmit Packet Available Interface signals. The Transmit Packet Available Interface connects to the reciprocal Transmit Packet Available Interface on the PHY Layer device.

Table 5: **Transmit Packet Available Interface**

Signal Name	Direction	Mode	Description
TADR[7:0]	OUT	Packet	**Transmit Address**: Used with the PTPA signal to poll the Physical Layer FIFO's packet available status. When TADR is sampled on the rising edge of the TFCLK by the PHY device, the polled packet available indication PTPA signal is updated with the status of the port specified by the TADR address on the following rising edge of TFCLK.
PTPA	IN	Packet	**Polled-PHY Transmit Packet Available**: PTPA asserts when a pre-defined minimum number of bytes are available in the polled transmit FIFO. After assertion, PTPA indicates that the transmit FIFO is not full. When PTPA deasserts, the transmit FIFO is full or near full. PTPA allows the polling of the PHY selected by the TADR address bus. The port which PTPA reports is updated on the following rising edge of TFCLK after the PHY address on TADR is sampled by the PHY device.
DTPA[C-1:0]	IN	Byte	**Direct Transmit Packet Available**: Provides direct status indication for the corresponding ports in the PHY device. DTPA asserts when a predefined minimum number of bytes are available in its transmit FIFO. After assertion, the DTPA signal indicates that the corresponding transmit FIFO is not full. When DTPA deasserts, the transmit FIFO is full or near full.
STPA	IN	Byte	**Selected-PHY Transmit Packet Available**: STPA asserts when a predefined minimum number of bytes is available in the transmit FIFO specified by the in-band address on TDAT. After assertion, the STPA signal indicates that the corresponding transmit FIFO is not full. When STPA deasserts, the transmit FIFO is full or near full. STPA always provides status indication for the selected port of the PHY device in order to avoid FIFO overflows while polling is performed. The port which STPA reports is updated on the following rising edge of TFCLK after the PHY address on TDAT is sampled by the PHY device.

Transmit Flow Control Interface

Table 6 defines the Transmit Flow Control Interface signals. The Transmit core samples the status of the PHY device and reports back to the User Interface.

Table 6: **Transmit Flow Control Interface**

Signal Name	Direction	Mode	Description
TX_TADR[7:0]	OUT	Both	**Transmit Address**: Used in conjunction with TX_PTPA or TX_STPA to indicate the corresponding address of the value of TX_PTPA or TX_STPA. In the case of TX_PTPA, the value on TX_TADR is the address that was sent on TADR. In the case of TX_STPA, the value on TX_TADR is the in-band address present on TDAT.
TX_PTPA	OUT	Packet	**Polled-PHY Transmit Packet Available**: Used in conjunction with the TX_TADR signal. The TX_PTPA signal is updated with the status received on PTPA. TX_PTPA asserts when a pre-defined minimum number of bytes are available in the PHY device's polled transmit FIFO. After assertion, TX_PTPA indicates that the PHY device's transmit FIFO is not full. When TX_PTPA deasserts, the PHY device's transmit FIFO is full or near full. Note that there are several cycles of latency between PTPA and TX_PTPA.
TX_DTPA[C-1:0]	OUT	Byte	**Direct Transmit Packet Available**: Provides a direct status indication for the corresponding ports in the PHY device. A bit in TX_DTPA asserts when a predefined minimum number of bytes are available in the PHY device's corresponding transmit FIFO for that channel. After assertion, a bit in TX_DTPA indicates that the PHY device's transmit FIFO is not full for that channel. When a bit in TX_DTPA deasserts, the PHY device's corresponding transmit FIFO for that channel is full or near full. Note that there are several cycles of latency between DTPA and TX_DTPA.
TX_STPA	OUT	Byte	**Selected-PHY Transmit Packet Available**: Used in conjunction with the TX_TADR signal. TX_STPA always provides status indication for the selected port of the PHY device in order to avoid FIFO overflows while writes are performed. TX_STPA asserts when a predefined minimum number of bytes are available in the PHY device's transmit FIFO. After assertion, TX_STPA indicates that the PHY device's transmit FIFO is not full. When STPA deasserts, the PHY device's transmit FIFO is full or near full. Note that there are several cycles of latency between STPA and TX_STPA.

Table 7 defines the additional Transmit Calendar Interface. This interface is only present when the Transmit core is configured to use polled flow-control mode.

Table 7: **Transmit Calendar Interface Signals**

Signal Name	Direction	Description
TX_CAL_CLK	IN	**Transmit Calendar Clock**: All Transmit Calendar Interface signals are synchronous to the rising edge of this clock.
TX_CAL_EN	IN	**Transmit Calendar Enable**: When asserted, the Transmit Calendar is enabled to generate an address on TADR. When deasserted, the Transmit Calendar holds its output on TADR. While programming (writing) the calendar, deassert this signal to avoid spurious TADR values from being generated.
TX_CAL_WR_EN	IN	**Transmit Calendar Write Enable**: When asserted, the Transmit Calendar is loaded with the data on the TX_CAL_DATA bus on the rising edge of TX_CAL_CLK. When deasserted, the Transmit Calendar is read on the TX_CAL_DATA_OUT bus.
TX_CAL_ADDR[8:0]	IN	**Transmit Calendar Address**: This bus contains the address to write the calender number indicated on the TX_CAL_DATA bus when TX_CAL_WR_EN is asserted. When TX_CAL_WR_EN is deasserted, TX_CAL_ADDR indicates the address to read from for the TX_CAL_DATA_OUT bus.
TX_CAL_DATA[7:0]	IN	**Transmit Calendar Data**: This bus contains the port address to write into the calendar buffer when TX_CAL_WR_EN is asserted.
TX_CAL_DATA_OUT[7:0]	OUT	**Transmit Calendar Data Out**: This bus contains the port address that is read from the calendar buffer when TX_CAL_WR_EN is deasserted.

Figure 15 illustrates a sample operation of the Transmit Packet Available Interface in Byte-level Transfer mode. The current status of the PHY Layer's Transmit FIFOs are sampled using the signals on the Transmit Packet Available Interface (DTPA and STPA). Approximately four clock cycles later, the TX_DTPA and TX_STPA signals are updated to reflect the latest status information. While TX_DTPA reflects the status of every port address, TX_STPA reflects the pipeline status of only the current port address being written to (TX_TADR) on the Transmit Packet Available Interface.

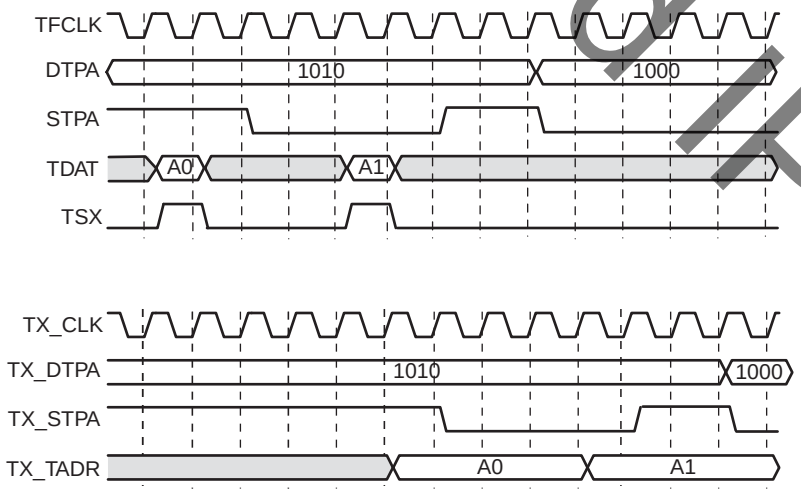

Figure 15: **Transmit Packet Available Interface: Byte-level Mode**

Figure 16 illustrates a sample operation of the Transmit Packet Available Interface in Packet-Level Transfer mode. The current status of the PHY Layer's Transmit FIFOs are sent using the PTPA signal on the Transmit Packet Available Interface. The Transmit core provides a port address on the TADR signal (the address sequence is determined by the Calendar, discussed below). The PHY Layer device or core generates the appropriate status on the PTPA signal, which is sampled and provided on TX_PTPA. In Figure 16, when port address A0 is driven on TADR on the second clock cycle, the corresponding PTPA is deasserted on the third clock cycle. This process continues for each address in the Calendar. On the User Interface, the 1-cycle offset between TADR and PTPA is removed when TX_TADR and TX_PTPA is provided.

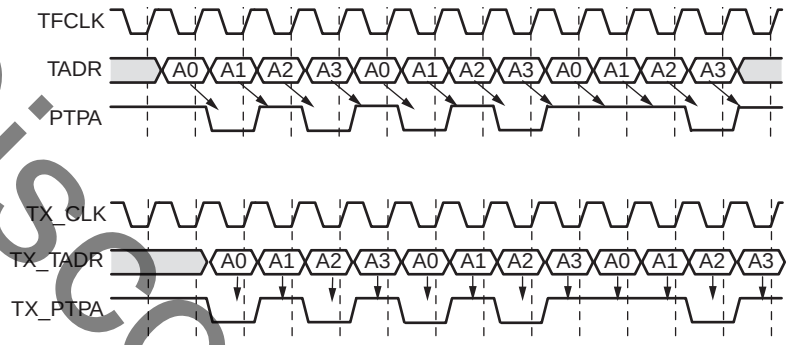

Figure 16: **Transmit Packet Available Interface: Packet-level Mode**

Figure 17 illustrates a sample operation of the Transmit Calendar Interface in Packet-Level Transfer mode. In Packet mode, the values sent on TADR are not necessarily all the port addresses in sequence, and the user may select any sequence desired via the Transmit Calendar Interface. In the figure, starting on the second clock cycle, the TX_CAL_EN signal is deasserted to freeze the output TADR bus to the current value. This is done to prevent spurious values from appearing on TADR as the Calendar is being written; if this is not a concern, then TX_CAL_EN need not be deasserted. Also note that TX_CAL_EN should be deasserted a few cycles before a Calendar write begins, since the signal must be synchronized into the TFCLK domain.

Starting on the fourth rising clock edge, the TX_CAL_WR_EN signal is asserted for five cycles. For those five cycles, the port addresses provided on TX_CAL_DATA are written into the Calendar at the addresses provided on TX_CAL_ADDR. When TX_CAL_WR_EN deasserts, the Calendar goes into read mode and the contents of the Calendar at address TX_CAL_ADDR are provided on TX_CAL_DATA_OUT, with one cycle of delay. When TX_CAL_EN is asserted again, the TADR bus resumes rotating through the Calendar values.

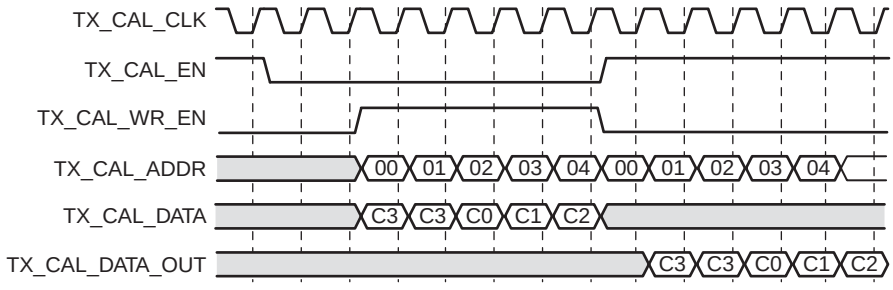

Figure 17: **Read and Write to the Calendar Interface**

Signal Widths

The Receive and Transmit cores can transmit and receive data on 8-bit, 16-bit, or 32-bit interface widths, depending on the core configuration. As a result, the remainder/modulo widths are related to the data path width. Additionally, the flow control interface signals depend on the number of supported port addresses in the selected configuration.

Table 8 defines the relationship between the data bus width and the LocalLink remainder (TX_REM, RX_REM) and SPI-3 modulo (TMOD, RMOD) signals. When the core is configured in 8-bit mode, the remainder and modulo signals are not present.

Table 8: **Data Bus Width Dependencies**

Data Bus Width (D)	Remainder/Modulo Width (M)
8	N/A
16	1
32	2

The number of supported port addresses is used to determine the width of the signals DTPA and TX_DTPA. The number of supported port addresses is configured through the CORE Generator GUI.

Remainder and Modulo Signaling

When an 8-bit interface is used, the REM signal on the User Interface and MOD signal on the Packet Interface are not present. The User Interface on the Receive and Transmit cores implement the Xilinx LocalLink standard. The Xilinx LocalLink standard specifies that the remainder signal indicate the number of valid bytes in the data bus as REM + 1, with the valid bytes MSB justified. This signal is only valid when EOF is asserted. Table 9 shows the possible REM signaling and the valid DATA signal bits.

Table 9: **Example REM Signaling**

32-bit		16-bit	
REM[1:0]	DATA valid	REM[0]	DATA valid
"00"	[31:24]	"0"	[15:8]
"01"	[31:16]	"1"	[15:0]
"10"	[31:8]		
"11"	[31:0]		

The SPI-3 Packet Interfaces on both the Receive and Transmit cores implement the *OIF SPI3-01.0* standard. The SPI-3 standard uses a modulo signal to indicate valid bytes on the last cycle of a transfer. Table 10 shows the MOD signaling and the valid bits on the DAT signal.

Table 10: **Example MOD Signaling**

32-bit		16-bit	
MOD[1:0]	**DAT valid**	**MOD[0]**	**DAT valid**
"00"	[31:0]	"0"	[15:0]
"01"	[31:8]	"1"	[15:8]
"10"	[31:16]		
"11"	[31:24]		

Verification

The Xilinx SPI-3 Link Layer core has been verified in a proprietary test environment using an internally developed bus functional model (BFM). Thousands of test vectors have been generated and verified, including both correct and errored frames.

References

OIF SPI3-01.0 System Packet Interface Level 3 Implementation Agreement

Support

Xilinx provides technical support for this LogiCORE 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*.

Ordering Information

This Xilinx LogiCORE module is provided under the SignOnce IP Site License. A free evaluation version of the module is provided with the Xilinx CORE Generator, which lets you assess the core functionality and demonstrates the various interfaces of the core in simulation. After purchase, the core may be downloaded from the Xilinx IP Center for use with the CORE Generator v12.4. The CORE Generator is bundled with ISE Foundation v12.4 software at no additional charge.

Please contact your local Xilinx sales representative for more information. Information about additional Xilinx LogiCORE modules is available on the Xilinx IP Center.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
8/31/05	4.0	Initial Xilinx release.
1/18/06	4.1	Updated to ISE tools v8.1i.
1/31/06	4.2	Spartan-3E added to supported device family in Facts table.
8/08/07	4.3	Added support for Spartan 3A/3AN/3A DSP devices, tool support for IP1 Jade Minor release.
4/24/09	4.4	Updated to ISE tools v11.1. Removed support for Spartan-3A DSP devices.
9/16/09	5.0	Updated ISE tools to v11.3 and core to v6.1. Added support for Virtex-6 (including -1L) devices.
12/02/09	6.0	Updated ISE tools to v11.4 and core to v7.1. Added support for Spartan-6 -4 speed grade devices.
4/19/10	7.0	Updated ISE tools to v12.1 and core to v7.2; added support for Spartan-6 -3 speed grade devices.
12/14/10	8.0	Updated ISE tools to v12.4 and core to v7.3.

Notice of Disclaimer