LogiCORE IP SPI-4.2 Lite v5.2

Introduction

The Xilinx LogiCORE™ IP SPI-4.2 (PL4) Lite core implements, and is functionally compliant with, the *OIF-SPI4-02.1 System Packet Interface Phase 2* standard. This fully verified solution interconnects physical layer devices to link layer devices in 2.5 Gbps POS, ATM, and Ethernet applications.

Features

- Up to 275+ MHz DDR on SPI-4.2 interface supporting 550 Mbps pin pair total bandwidth
- Supports Static Phase Alignment
- Bandwidth optimized Source core achieves optimal bus throughput without additional FPGA resources
- Flexible pin assignment: maximum core performance achievable with user defined pinouts
- Configurable 32-bit or 64-bit user interface
- Multiple core support—more than 4 cores can be implemented in a single device
- Sink and Source cores configured through Xilinx CORE Generator™ software for easy customization
- Delivers Sink and Source cores as independent solutions, enabling flexible implementation
- Supports 1—256 addressable channels with fully configurable SPI-4.2 calendar interface
- Supports LVTTL or LVDS Status FIFO path operating at 1/4 or 1/8 of the data rate
- Provides DIP-4 and DIP-2 parity generation and verification
- Provides Sink and Source FIFO controls for flushing the contents of the FIFO without restarting the interface

LogiCORE IP Facts	
Core Specifics	
Supported Device Family	Virtex®-6 [1], Spartan®-6, Virtex-5, Virtex-4, Spartan-3, Spartan-3E Spartan-3A/3AN/3A DSP
Resources Used	
See Table 17 through Table 21.	
Performance	
See Table 16 for performance metric details.	
Provided with Core	
Documentation	Product Specification Getting Started Guide User Guide Release Notes
Design File Formats	NGC File
Constraints File	Example UCE Embedded RPMs
Verification	VHDL and Verilog Test Bench
Design Tool Requirements	
Xilinx Implementation Tools	ISE® 12.1
Simulation	Mentor Graphics ModelSim® v6.5c and above Synopsys VCS and VCS MX 2009.12 and above Cadence Incisive Enterprise Simulator (IES) v9.2 and above
Synthesis	XST, ISE
Support	
Provided by Xilinx, Inc.	

1. Includes Virtex-6-1L devices.

Applications

The SPI-4.2 (PL4) Lite interface core enables the interconnection of physical layer devices to link-layer devices in 2.5 Gbps POS, ATM, and Ethernet applications. The symmetric interface can be used to implement both the PHY and Link layer. Figure 1 illustrates the core in a typical link-layer application.

Driven by the improved efficiencies and lower cost-per-Mbit of Packet-over-SONET/SDH, the core is ideally suited for line cards in gigabit routers, terabit and optical cross-connect switches, and a wide range of multi-service DWDM and SONET/SDH-based transmission systems.
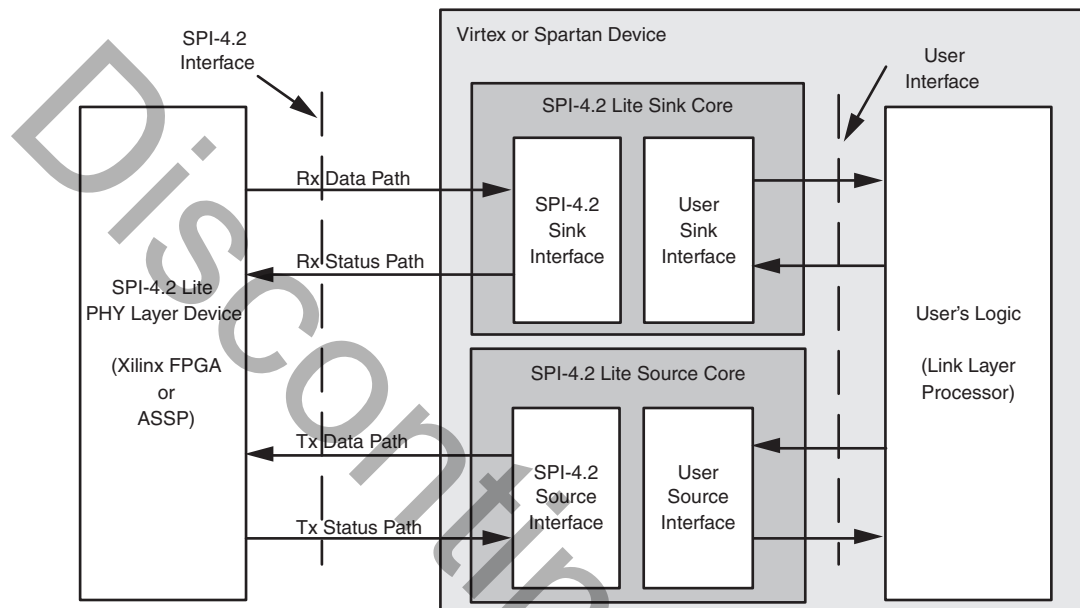

Figure 1: **SPI-4.2 Lite Core in a Typical Link Layer Application**

Functional Overview

The SPI-4.2 Lite solution consists of two separate modules: the Sink core and the Source core. The Sink core receives data and sends status on its SPI-4.2 interface; the Source core transmits data and receives status on its SPI-4.2 interface.

Sink Core

The Sink core receives 16-bit Source synchronous data on the SPI-4.2 interface and combines these bits into 32-bit or 64-bit data words on the user interface. The core also processes 2-bit status information (for each channel) from the user interface and transmits it in sequence on the SPI-4.2 interface with the appropriate framing and DIP2 information. In addition to data, other signals associated with the core operational state and received packets are also available. These signals include FIFO status and SPI-4.2 protocol violations.

The Sink core has two primary interfaces, the SPI-4.2 interface and the user interface. The input and output signals and the functional blocks of the Sink core are illustrated in Figure 2. The interface signals to each of the functional modules are described in detail in the "Core Interfaces," page 5 of this document.
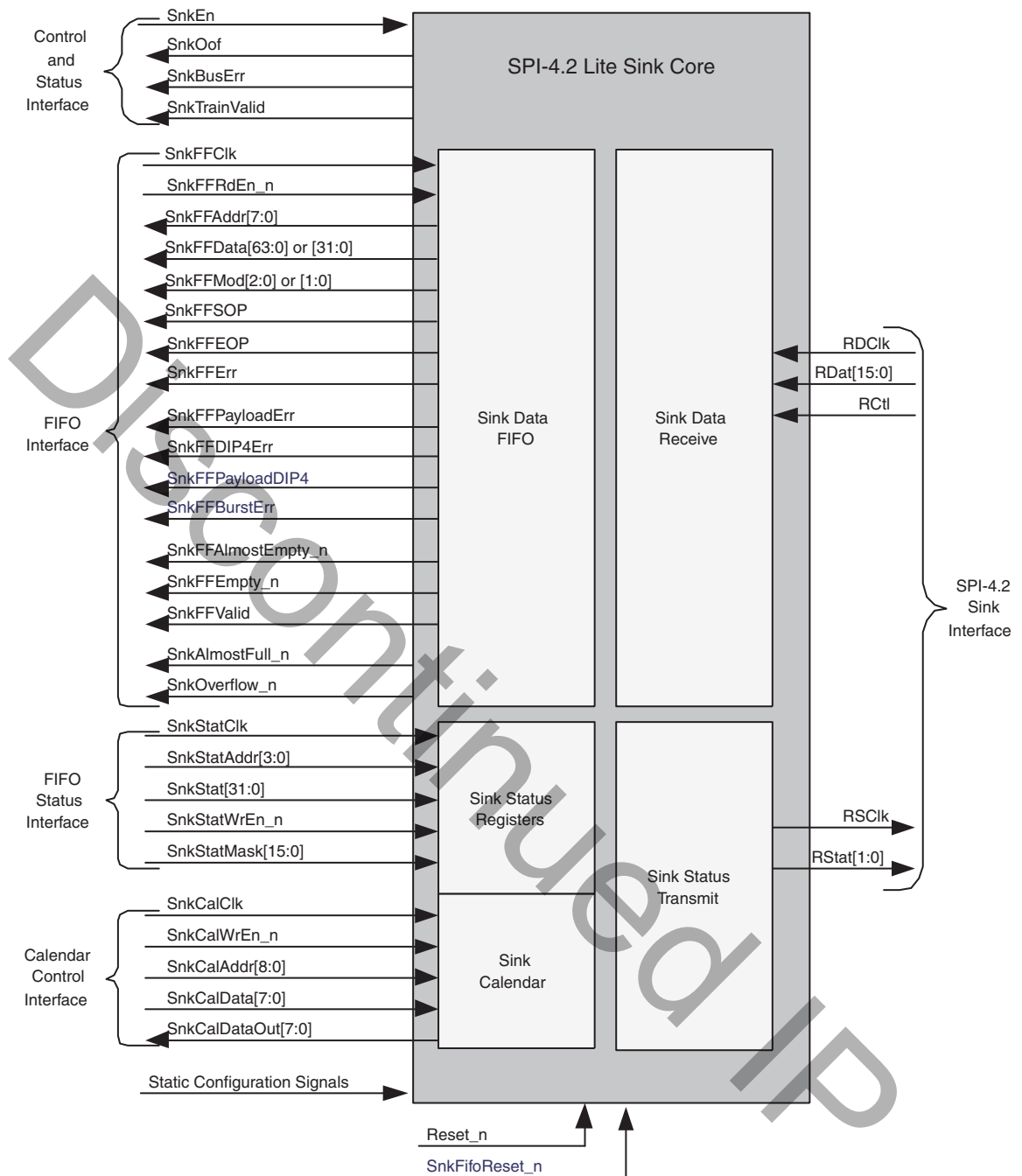
Figure 2: **Sink Core Block Diagram and I/O Interface Signals**

Source Core

The Source core transmits 16-bit Source synchronous data on its SPI-4.2 interface by processing and formatting 32-bit or 64-bit data words from its user interface. The core also processes 2-bit status (for each channel) on the SPI-4.2 interface and presents it on its user interface. In addition to data, other signals associated with the operational state of the core and transmitted packets are also available. These signals include FIFO status and SPI-4.2 protocol violations.

The Source core has two primary interfaces, the SPI-4.2 interface and the user interface. The input and output signals and the functional blocks of the Source core are illustrated in Figure 3. The interface signals to each of the functional modules are described in "Core Interfaces," page 5.
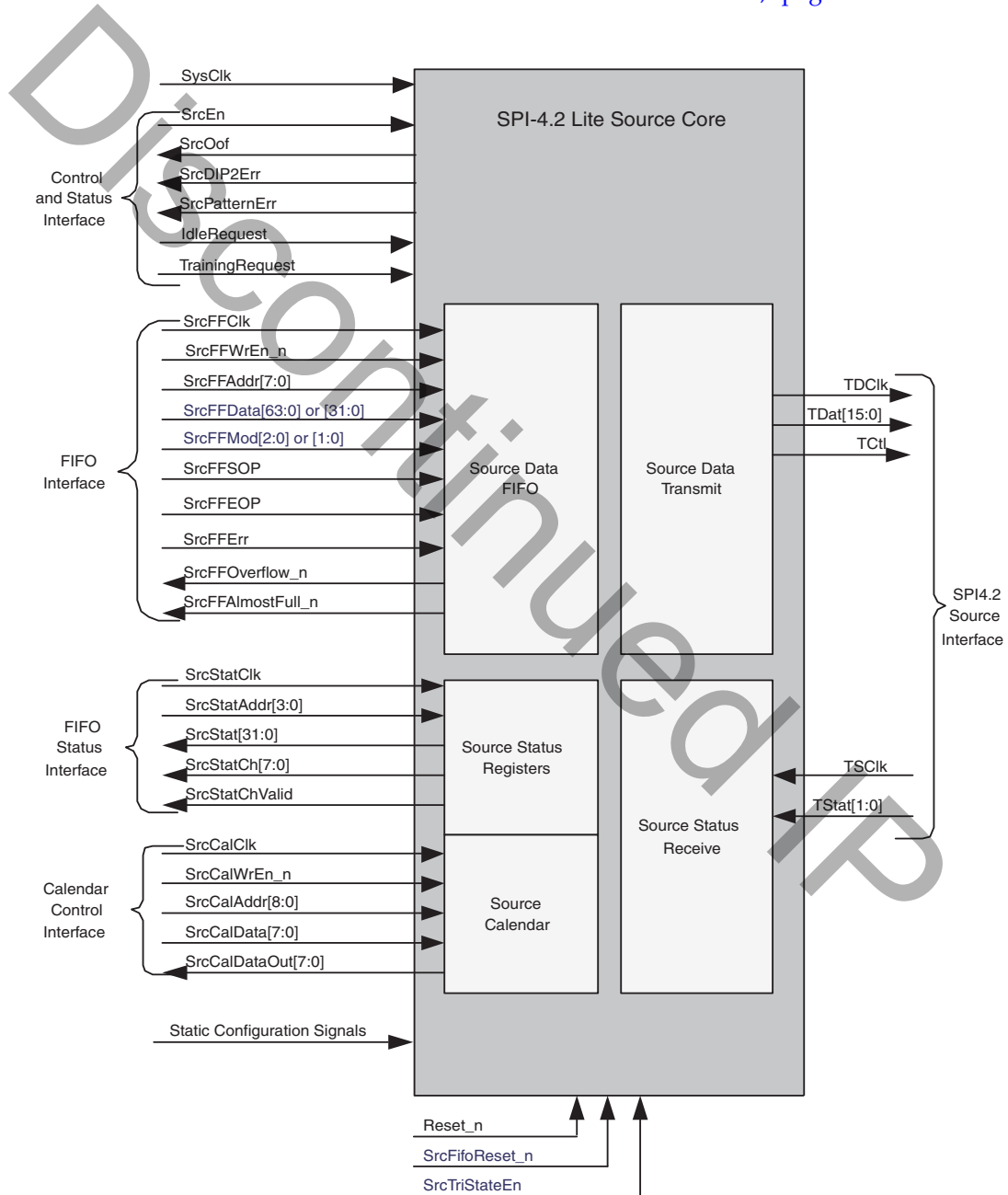

Figure 3: **Source Core Block Diagram and I/O Interface Signals**

Core Interfaces

This section provides definitions of the interface signals of the Sink and Source cores.

Sink Interfaces

The Sink core has two primary interfaces, the SPI-4.2 interface and the user interface.

Sink SPI-4.2 Interface

Table 1 defines the signals on the Sink SPI-4.2 interface.

Table 1: **Sink SPI-4.2 Interface Signals**

Name	Direction	Clock Domain	Description
RDClk_P RDClk_N	Input	n/a	**SPI-4.2 Receive Data Clock** (LVDS): Source synchronous clock received with RDat and RCtl. The rising and falling edges of this clock (DDR) are used to clock RDat and RCtl.
RDat_P[15:0] RDat_N[15:0]	Input	RDClk	**SPI-4.2 Receive Data Bus** (LVDS): The 16-bit data bus used to receive SPI-4.2 data and control information.
RCtl_P RCtl_N	Input	RDClk	**SPI-4.2 Receive Control** (LVDS): SPI-4.2 Interface signal that indicates whether data or control information is present on the RDat bus. When RCtl is deasserted, data is present on RDat. When RCtl is asserted, control information is present on RDat.
RSClk	Output	n/a	**SPI-4.2 Receive Status Clock**: Source synchronous clock transmitted with RStat at 1/4 or 1/8 rate of the RDClk. The rate of the status clock is controlled by the static configuration signal RSClkDiv. The user can select this signal to be transmitted as LVTTL or LVDS.
RStat[1:0]	Output	RSClk	**SPI-4.2 Receive FIFO Status**: FIFO Status Channel flow control interface. The user can select this bus to be transmitted as LVTTL or LVDS.

Sink User Interface

The Sink User Interface can be divided into several subgroups based on function:

- Control and Status Interface
- FIFO Interface
- Status and Flow Control Interface
  - Calendar Control Interface
  - Status FIFO Interface
- Configuration Interface

Sink Control and Status Interface

The Sink core control and status interface signals control the operation of the Sink core and provide status information that is not associated with a particular channel (port) or packet. Table 2 defines the Sink control and status signals.

Table 2: **Sink Control and Status Signals**

Name	Direction	Clock Domain	Description
Reset_n	Input	n/a	**Reset**: Active low signal that asynchronously initializes internal flip-flops, registers, and counters. When Reset_n is asserted, the Sink core will go out of frame and the entire data path is cleared (including the FIFO). The Sink core will also assert SnkOof, and deassert SnkBusErr and SnkTrainValid. When Reset_n is asserted, the Sink core will transmit framing "11" on RStat and continue to drive RSClk. Following the deassertion of Reset_n, the Sink calendar should be programmed if the calendar is initialized in-circuit.
SnkFifoReset_n	Input	SnkFFClk	**Sink FIFO Reset:** This active low signal enables the user to reset the Sink FIFO and the associated data path logic. This enables the FIFO to be cleared while remaining in-frame. Coming out of SnkFifoReset_n, the Sink core will discard all data on the SPI-4.2 interface until a valid SOP control word is received.
SnkEn	Input	SnkStatClk	**Sink Enable:** Active high signal that enables the Sink core. When SnkEn is deasserted, the Sink core will go out of frame and will not store any additional data in the FIFO. The current contents of the FIFO remain intact. The Sink core will also assert SnkOof, and deassert SnkBusErr and SnkTrainValid. When SnkEn is deasserted, the Sink core will transmit framing "11" on RStat and continue to drive RSClk.
SnkOof	Output	SnkFFClk	**Sink Out-of-Frame**: Active high signal that indicates that the Sink core is not in-frame. This signal is asserted when SnkEn is deasserted or the Sink core loses synchronization with the data received on the SPI-4.2 interface. This signal is deasserted once the Sink core reacquires synchronization with the received SPI-4.2 data.
SnkBusErr	Output	SnkFFClk	**Sink Bus Error:** Active high signal that indicates SPI-4.2 protocol violations or bus errors that are not associated with a particular packet. Information on the specific error condition that caused the SnkBusErr assertion is provided on SnkBusErrStat

Table 2: **Sink Control and Status Signals** *(Cont'd)*

Name	Direction	Clock Domain	Description
SnkBusErrStat[7:0]	Output	SnkFFClk	**Sink Bus Error Status**: Each bit of this bus corresponds to a specific Sink Bus Error condition and is asserted concurrently with SnkBusErr. The error conditions detected are reported as follows: • SnkBusErrStat [0]: Minimum SOP spacing violation • SnkBusErrStat [1]: Control word with EOP not preceded by a data word • SnkBusErrStat [2]: Payload control word not followed by a data word • SnkBusErrStat [3]: DIP4 error received during training or on idles • SnkBusErrStat [4]: Reserved control words received • SnkBusErrStat [5]: Non-zero address bits on control words received (except on payload and training control words) • SnkBusErrStat [6:7]: Reserved bits (tied low)
SnkTrainValid	Output	SnkFFClk	**Sink Training Valid:** Active high signal that indicates that a valid training pattern has been received. This signal is asserted for the duration of the training pattern (20 SPI-4.2 Lite bus clock cycles or 5 RDClk0_GP clock cycles), if the training pattern received is successfully decoded.

Sink FIFO Interface

The Sink core FIFO interface provides data received on the SPI-4.2 interface to the user's logic. In addition to the 32-bit or 64-bit data word, there are control and status signals (including error signals) associated with a particular channel or packet. These include signals to flag improper packet format, DIP4 error, FIFO Status, and so forth. Table 3 defines the Sink FIFO signals.

Table 3: **Sink FIFO Signals**

Name	Direction	Description
SnkFFClk	Input	**Sink FIFO Clock**: All Sink FIFO Interface signals are synchronous to the rising edge of this clock.
SnkFFRdEn_n	Input	**Sink FIFO Read-Enable**: When detected low at the rising edge of SnkFFClk, data and status information is available from the FIFO on the next rising edge of SnkFFClk.
SnkFFAddr[7:0]	Output	**Sink FIFO Channel Address**: Channel number associated with the data on SnkFFData.
SnkFFData[63:0] or SnkFFData[31:0]	Output	**Sink FIFO Data Out**: The Sink FIFO data bus. Bit 0 is the LSB. The core can be configured to have a 32-bit or 64-bit Interface. The 64-bit interface enables the user to run at half the clock rate required for a 32-bit interface.
SnkFFMod[1:0] or SnkFFMod[2:0]	Output	**Sink FIFO Modulo**: This signal indicates which bytes on the SnkFFData bus are valid when the SnkFFEOP signal is asserted. SnkFFMod[1:0] is used with a 32-bit interface. SnkFFMod[2:0] is used with a 64-bit interface.
SnkFFSOP	Output	**Sink FIFO Start of Packet**: When asserted (active high), this signal indicates the start of a packet is being read out of the Sink FIFO.
SnkFFEOP	Output	**Sink FIFO End of Packet**: When asserted (active high), this signal indicates that the end of a packet is being read out of the Sink FIFO.

Table 3: **Sink FIFO Signals** *(Cont'd)*

Name	Direction	Description
SnkFFErr	Output	**Sink FIFO Error**: When asserted (active high), this signal indicates that the current packet is terminated with an EOP abort condition. This signal is only asserted when SnkFFEOP is asserted.
SnkFFEmpty_n	Output	**Sink FIFO Empty**: When asserted (active low), this signal indicates that the Sink FIFO is empty. No data can be read until this signal is deasserted. This signal is asserted with the last data word read out of the FIFO.
SnkFFAlmostEmpty_n	Output	**Sink FIFO Almost Empty**: When this signal is asserted (active low), it indicates that one word remains in the FIFO, and the user should deassert the read enable signal on the next clock cycle. The user's read logic should evaluate the SnkFFEmpty_n signal to verify that there is no data in the FIFO in case an additional word was simultaneously written into the FIFO. An example of the behavior of this interface signal is provided with the SPI-4.2 Lite core in the Example Design. (See the pl4_lite_fifo_loopback_read.v/vhd file.)
SnkFFValid	Output	**Sink FIFO Read Valid**: When asserted (active high), this signal indicates that the information on SnkFFData, SnkFFAddr, SnkFFSOP, SnkFFEOP, SnkFFBurstErr, SnkFFMod, SnkFFErr, SnkFFDIP4Err, and SnkFFPayloadErr is valid.
SnkFFDIP4Err	Output	**Sink FIFO DIP-4 Error:** When asserted (active high), this signal indicates that a DIP-4 parity error was detected with the SPI-4.2 control word ending a packet or burst of data. This signal is asserted at the end of that packet or burst of data.
SnkFFPayloadDIP4	Output	**Sink FIFO Payload DIP-4 Error:** When asserted (active high), this signal indicates that a DIP-4 parity error was detected with the SPI-4.2 control word starting a packet or burst of data. This signal is asserted at the end of that packet or burst of data.
SnkFFBurstErr	Output	**Sink FIFO Burst Error:** When asserted (active high), this signal indicates that the Sink has received data that was terminated on a non-credit boundary without an EOP. SnkFFBurstErr may be used by the user's logic to indicate missing EOPs, or incorrectly terminated bursts. In this case the Sink core does not assert SnkFFEOP or SnkFFErr.
SnkFFPayloadErr	Output	**Sink FIFO Payload Error:** When asserted (active high), this signal indicates that the received data was not preceded by a valid payload control word. Since it is not clear what the packet Address and SOP should be, it is flagged as an error. This is asserted with each data word coming out of the FIFO, and will remain asserted until a valid payload control word is followed by data.
SnkAlmostFull_n	Output	**Sink Almost Full**: When asserted (active low), this signal indicates that the Sink core is approaching full (as defined by the parameter SnkAFThresAssert), and that immediate action should be taken to prevent overflow.
SnkOverflow_n	Output	**Sink Overflow:** When asserted (active low), this signal indicates that the Sink core has overflowed and is in an error condition. Data will be lost if SnkOverflow_n is asserted, since no data is written into the FIFO when the overflow signal is asserted.

Sink Status and Flow Control Interface

Sink Calendar Control Interface

The Sink core calendar control interface determines the status channel order and frequency. Through this interface, the user can program the calendar buffer that determines the order and frequency in which channel status is sent on the SPI-4.2 interface. This interface can force DIP-2 parity error insertions for use in system testing and diagnostics. Table 4 defines the calendar control interface signals.

Table 4: **Sink Calendar Control Signals**

Name	Direction	Clock Domain	Description
SnkCalClk	Input	n/a	**Sink Calendar Clock:** All Sink calendar signals are synchronous to this clock.
SnkCalWrEn_n	Input	SnkCalClk	**Sink Calendar Write Enable:** When this signal is asserted (active low), the Sink Calendar is written with the data on the SnkCalData bus on the rising edge of SnkCalClk. When the signal is deasserted, the Sink Calendar data can be read on SnkCalDataOut.
SnkCalAddr[8:0]	Input	SnkCalClk	**Sink Calendar Address**: When SnkCalWrEn_n is asserted, this bus indicates the calendar address to which the data on SnkCalData is written. When SnkCalWrEn_n is deasserted, this bus indicates the calendar address from which the channel number on SnkCalDataOut is driven.
SnkCalData[7:0]	Input	SnkCalClk	**Sink Calendar Data**: This bus contains the channel number to write into the calendar buffer when SnkCalWrEn_n is enabled. The channel numbers written into the calendar indicate the order that status is sent on RStat.
SnkCalDataOut[7:0]	Output	SnkCalClk	**Sink Calendar Data Output**: This bus contains the channel number that is read from the calendar buffer when SnkCalWrEn_n is disabled. The channel numbers read from the calendar indicate the order that status is sent on RStat.

Sink Status FIFO Interface

The Sink core Status FIFO interface enables the user to send flow control data to the transmitting device. Flow control can be implemented either automatically or manually. Table 5 defines the status FIFO interface signals.

Table 5: **Sink Status FIFO Signals**

Name	Direction	Clock Domain	Description
SnkStatClk	Input	n/s	**Sink Status Clock:** All Sink Status write signals are synchronous to this clock.
SnkStat[31:0]	Input	SnkStatClk	**Sink Status Bus:** This 32-bit bus is used to write status information into the Status FIFO. The user can write the status for 16 channels each clock cycle. The 16-channel status that are accessed simultaneously are grouped in the following manner: channels 15 to 0, channels 31 to 16, channels 47 to 32, . . . , channels 255 to 239.
SnkDIP2ErrRequest	Input	SnkStatClk	**Sink DIP-2 Error Request:** This is an active high signal that requests an incorrect DIP-2 to be sent out of the RStat bus. When this signal is asserted, the Sink Status FIFO responds by inverting the next DIP-2 value that it transmits.
SnkStatAddr[3:0]	Input	SnkStatClk	**Sink Status Address bus**: The Sink Status Address determines the group of 16-channel status that SnkStat will be updating. Bank 0: SnkStatAddr=0 channels 15 to 0 Bank 1: SnkStatAddr=1, channels 31 to 16 Bank 2: SnkStatAddr=2, channels 47 to 32 . . . Bank 15: SnkStatAddr=15 channels 255 to 239
SnkStatWr_n	Input	SnkStatClk	**Sink Status Write**: The Sink Status Write qualifies the SnkStatMask signal. When SnkStatWr_n is asserted, status for the different channels is updated. When SnkStatWr_n is deasserted, the SnkStat input is ignored.
SnkStatMask[15:0]	Input	SnkStatClk	**Sink Status Mask Bus**: The Sink Status Mask indicates which portions of the SnkStat bus are valid when SnkStatWr_n is asserted. This allows the user to update status for a subset of the 16 channels represented by SnkStat. When SnkStatMask[x] = 1, status for channel x will be updated. When SnkStatMask[y] = 0, status for channel y will not be updated. For example, if SnkStatMask[15] = 0, then SnkStat[31:30] will be disregarded and the channel it represents will not have its status value updated. If SnkStatMask are all zeros, none of the sixteen 2-bit status values are updated. If SnkStatMask are all ones, all sixteen of the 2-bit status values are updated.

Sink Static Configuration Interface

The Sink core static configuration signals enable users to customize the core based on their system requirements. These input signals are statically driven by setting them to a constant value in the top-level wrapper file. Two of the Sink static configuration signals can be changed in circuit: SnkCalendar_M and SnkCalendar_Len. Both signals are static registers that are synchronous to SnkStatClk. Table 6 defines the static configuration signals.

Table 6: **Sink Static Configuration Signals**

Name	Direction	Range	Description
NumDip4Errors[3:0]	Static Input	1-15 Value of 0 is set to 1	**Number of DIP-4 Errors**: The Sink Interface will go out-of-frame (assert SnkOof) and will stop accepting data from the SPI-4.2 bus after receiving NumDip4Errors consecutive DIP-4 errors.
NumTrainSequences[3:0]	Static Input	1-15 Value of 0 is set to 1	**Number of Complete Training Sequences**: A complete training pattern consists of 10 training control words and 10 training data words. The Sink interface requires NumTrainSequences consecutive training patterns before going in-frame (deasserting SnkOof) and accepting data from the SPI-4.2 bus.
SnkCalendar_M[7:0]	Input	0-255 (effective range 1-256)	**Sink Calendar Period**: The SnkCalendar_M parameter sets the number of repetitions of the calendar sequence before the DIP-2 parity and framing words are inserted. The core implements this parameter as a static register synchronous to SnkStatClk, and it can be updated in circuit by first deasserting SnkEn. Note that the Sink Calendar Period equals SnkCalendar_M + 1. For example, if SnkCalendar_M=22, the Sink Calendar Period will be equal to 23.
SnkCalendar_Len[8:0]	Input	0-511 (effective range 1-512)	**Sink Calendar Length**: The SnkCalendar_Len parameter sets the length of the calendar sequence. The core implements this parameter as a static register synchronous to SnkStatClk, and it can be updated in circuit by first deasserting SnkEn. Note that the Sink Calendar Length equals SnkCalendar_Len + 1. For example, if SnkCalendar_Len=15, the Sink Calendar Length will be equal to 16.

Table 6: **Sink Static Configuration Signals** *(Cont'd)*

Name	Direction	Range	Description
SnkAFThresAssert[8:0]	Static Input	1–508 Values less than 1 are set to 1. Values greater than 508 are set to 508.	**Sink Almost Full Threshold Assert**: The SnkAFThresAssert parameter defines the minimum number of empty FIFO locations that exist when SnkAlmostFull_n is asserted. Note that the assert threshold must be less than or equal to the negate threshold (SnkAFThresNegate). When SnkAlmostFull_n is asserted, the core initiates the flow control mechanism selected by the parameter FifoAFMode. The FifoAFMode defines when the interface stops sending valid FIFO status levels and begins sending flow control information on RStat. This indicates to the transmitting device that the core is almost full and additional data cannot be sent.
SnkAFThresNegate[8:0]	Static Input	SnkAFThresAssert to 508 Values less than SnkAFThresAssert are set to SnkAFThresAssert. Values greater than 508 are set to 508.	**Sink Almost Full Threshold Negate**: The SnkAFThresNegate parameter defines the minimum number of empty FIFO locations that exist when SnkAlmostFull_n is deasserted. Note that the negate threshold must be greater or equal to the assert threshold (SnkAFThresAssert). When SnkAlmostFull_n is deasserted, the core stops sending flow control and resumes transmission of valid FIFO status levels. This indicates to the transmitting device that additional data can be sent.
RSClkDiv	Static Input	n/a	**Sink Status Clock Divide**: This static input is used to determine if the RSClk is 1/4 of the data rate, which is compliant with the OIF specification, or 1/8 of the data rate, which is required by some PHY ASSPs: 0: RSClkDiv = 1/4 rate (default value) 1: RSClkDiv = 1/8 rate
RSClkPhase	Static Input	n/a	**Sink Status Clock Phase**: This static input determines whether the *FIFO Status* Channel data (RStat[1:0]) changes on the rising edge of RSClk or the falling edge of RSClk: 0: RSClkPhase = rising edge of RSClk (default value) 1: RSClkPhase = falling edge of RSClk

Table 6: **Sink Static Configuration Signals** *(Cont'd)*

Name	Direction	Range	Description
FifoAFMode[1:0]	Static Input	n/a	**Sink Almost Full Mode**: Selects the mode of operation for the Sink interface when the Sink core reaches the Almost Full threshold (SnkAFThresAssert). If FifoAFMode is set to "00," the Sink interface goes out-of-frame when the core is almost full, and the Sink Status logic sends the framing sequence "11" until Sink core is not almost full. If FifoAFMode is set to "01," the Sink interface remains in-frame (SnkOof deasserted), and the Sink Status logic sends satisfied "10" on all channels until SnkAlmostFull_n is deasserted. If FifoAFMode is set to "10" or "11," the Sink interface will remain in-frame (SnkOof deasserted), and the Sink Status logic continues to drive out the user's status information (*i.e.,* continues in normal operation). In this case, the user should take immediate action to prevent overflow and loss of data.

Source Interfaces

The Source core has two primary interfaces, the SPI-4.2 interface and the user interface.

Source SPI-4.2 Lite Interface

Table 7 defines the signals on the Source SPI-4.2 Interface.

Table 7: **Source SPI-4.2 Lite Interface Signals**

Name	Direction	Clock Domain	Description
TDClk_P TDClk_N	Output	n/a	**SPI-4.2 Transmit Data Clock (LVDS)**: Source synchronous clock transmitted with TDat. The rising and falling edges of this clock (DDR) are used to clock TDat and TCtl.
TDat_P[15:0] TDat_N[15:0]	Output	TDClk	**SPI-4.2 Transmit Data Bus (LVDS)**: The 16-bit data bus is used to transmit SPI-4.2 data and control information.
TCtl_P TCtl_N	Output	TDClk	**SPI-4.2 Transmit Control (LVDS)**: SPI-4.2 Interface signal that defines whether data or control information is present on the TDat bus. When TCtl is Low, data is present on TDat. When TCtl is High, control information is present on TDat.
TSClk	Input	n/a	**SPI-4.2 Transmit Status Clock**: Source synchronous clock that is received by the Source core with TStat at 1/4 rate (or 1/8 rate) of TDClk. The user can select this signal to be transmitted as LVTTL or LVDS.
TStat[1:0]	Input	TSClk	**SPI-4.2 Transmit FIFO Status**: FIFO-Status-Channel flow control interface. The user can select this bus to be transmitted as LVTTL or LVDS.

Source User Interface

The Source User Interface can be divided into several subgroups based on function:

- Control and Status Interface
- Data FIFO Interface
- Status and Flow Control Interface
  - Calendar Control Interface
  - Status FIFO Interface
- Configuration Interface

Source Control and Status Interface

The Source core control and status interface signals control the operation of the Source core and provide status information that is not associated with a particular channel (port) or packet. Table 8 defines the Source control and status signals.

Table 8: **Source Control and Status Signals**

Name	Direction	Clock Domain	Description
Reset_n	Input	n/a	**Reset_n**: This active low, asynchronous control signal enables the user to restart the entire Source core. This means that the core will go out-of-frame. While Reset_n is asserted, the Source core transmits idles cycles on TDat. Coming out of Reset_n, the Source core transmits training patterns. Following the release of Reset_n, the Source Calendar should be programmed if the calendar is to be initialized in-circuit.
SrcFifoReset_n	Input	SrcFFClk	**SrcFifoReset_n**: This active low control signal enables the user to reset the Source FIFO and the associated data path logic. This enables the FIFO to be cleared while remaining in-frame. Upon Source FIFO Reset, the Source core sends idle cycles until the user writes data into the FIFO.
SrcEn	Input	SrcStatClk	**Source Enable**: An active high signal that enables the Source core. When SrcEn is deasserted, the Source core will not store or verify received status information. The Source core will also assert SrcOof, and deassert SrcDIP2Err and SrcPatternErr. When SrcEn is deasserted, the Source core will transmit training patterns on TDat.
SrcOof	Output	SrcFFClk	**Source Out-of-Frame:** When this signal is asserted (active high), it indicates that the Source core is not in-frame. This signal is asserted when the Source core loses synchronization on the transmit FIFO status interface. This is caused by the receipt of consecutive DIP-2 parity errors (determined by the parameter NumDip2Errors), invalid received status frame sequence (of four consecutive frame words "11"), or when SrcEn is deasserted. This signal is deasserted once the Source core reacquires synchronization with the SPI-4.2 transmit Status Channel. Synchronization occurs when consecutive valid DIP2 words (determined by the Static Configuration signal NumDip2Matches) are received and SrcEn is asserted.

Table 8: **Source Control and Status Signals** *(Cont'd)*

Name	Direction	Clock Domain	Description
SrcDIP2Err	Output	SrcFFClk	Source DIP-2 Parity Error: When this signal is asserted (active high), it indicates that a DIP-2 parity error was detected on TStat. This signal is asserted for one clock cycle each time a parity error is detected.
SrcStatFrameErr	Output	SrcFFClk	**Source Status Frame Error**: When this signal is asserted (active high) it indicates that a non "11" frame word was received after DIP-2 on TStat. This signal is asserted for one clock cycle each time an error frame word is detected.
SrcOofOverride	Input	SrcFFClk	**Source Out-of-Frame Override**: When this signal is asserted, the Source core behaves as if in-frame, and sends data on TDat regardless of the status received on TStat. This signal is used for system testing and debugging.
SrcPatternErr	Output	SrcFFClk	**Source Data Pattern Error:** When this signal is asserted (active high), it indicates that the data pattern written into the Source FIFO is illegal. Illegal patterns include the following: Burst of data terminating on a non-credit boundary (not a multiple of 16 bytes) with no EOP Non-zero value on SrcFFMod when SrcFFEOP is deasserted This signal is asserted for one clock cycle each time an illegal data pattern is written into the Source FIFO.
IdleRequest	Input	SrcFFClk	**Idle Request**: This is an active high signal that requests idle control words be sent out of the Source SPI-4.2 interface. The Source core responds by sending out idle control words at the next burst boundary. This signal overrides normal SPI-4.2 data transfer requests, but it does not override training sequence requests (TrainingRequest). Activating the request for idle cycles does not affect the Source FIFO contents or the user side operation.
TrainingRequest	Input	SrcFFClk	**Training Pattern Request**: This is an active high signal that requests training patterns be sent out of the Source SPI-4.2 interface. The Source core responds by sending out training patterns at the next burst boundary. This signal overrides idle requests (IdleRequest) and normal SPI-4.2 data transfers. Activating the request for training cycles does not affect the Source FIFO contents or the user side operation.
SrcTriStateEn	Input	SrcFFClk	**SrcTriStateEn**: This is an active high control signal that enables the user to tri-state the IOB drivers for the following Source core outputs: TDClk, TDat[15:0], and TCtl. When SrcTriStateEn=0 the outputs are not tri-stated. When SrcTriStateEn=1 the outputs are tri-stated. Default setting for this signal is disabled (SrcTriStateEn=0.)

Source FIFO Interface

The Source core FIFO interface stores data from the user's logic which will be transmitted on the SPI-4.2 interface. In addition to the 32-bit or 64-bit data word, there are control and status signals (including error signals) associated with a particular channel or packet. These include signals to insert DIP-4 errors, idles, training patterns, etc. Table 9 defines the Source FIFO signals.

Table 9: **Source FIFO Signals**

Name	Direction	Clock Domain	Description
SrcFFClk	Input	n/a	**Source FIFO Clock**: All Source FIFO Interface signals are synchronous to the rising edge of this clock.
SrcFFWrEn_n	Input	SrcFFClk	**Source FIFO Write-Enable**: When asserted (active low) at the rising edge of SrcFFClk, data and packet information is written into the FIFO.
SrcFFAddr[7:0]	Input	SrcFFClk	**Source FIFO Channel Address**: Channel number associated with the data on SrcFFData.
SrcFFData[63:0] or SrcFFData[31:0]	Input	SrcFFClk	**Source FIFO Data**: The Source FIFO data bus. Bit 0 is the LSB. The core can be configured to have a 32-bit or a 64-bit interface. The 64-bit interface enables the user to run at half the clock rate required for a 32-bit interface.
SrcFFMod[1:0] or SrcFFMod[2:0]	Input	SrcFFClk	**Source FIFO Modulo:** This signal indicates which bytes on the SrcFFData bus are valid when the SrcFFEOP or SrcFFErr signal is asserted. When SrcFFEOP is deasserted, SrcFFMod should always be zero. SrcFFMod[1:0] is used with a 32-bit interface. SrcFFMod[2:0] is used with a 64-bit interface.
SrcFFSOP	Input	SrcFFClk	**Source FIFO Start of Packet**: When asserted (active high), this signal indicates that the start of a packet is being written into the Source FIFO.
SrcFFEOP	Input	SrcFFClk	**Source FIFO End of Packet**: When asserted (active high), this signal indicates that the end of a packet is being written into the Source FIFO. May be concurrent with SrcFFSOP.
SrcFFErr	Input	SrcFFClk	**Source FIFO Error**: When asserted (active high) simultaneously with the SrcFFEOP flag, the current packet written into the FIFO contains an error. This causes an EOP abort to be sent on the SPI-4.2 Interface. SrcFFErr can be used in combination with SrcFFEOP to insert erroneous DIP-4 values for testing purposes. When SrcFFErr is asserted and SrcFFEOP is not asserted, the core inserts an EOP (1 or 2 bytes depending on the SrcFFMod value) with an erroneous DIP-4 value. The erroneous DIP-4 value is an inversion of the correctly calculated value.
SrcFFAlmostFull_n	Output	SrcFFClk	**Source FIFO Almost Full**: When asserted (active low), this signal indicates that the FIFO is approaching full, and no more data should be written.
SrcFFOverflow_n	Output	SrcFFClk	**Source FIFO Overflow:** When asserted (active low), this signal indicates that the FIFO has overflowed and is in an error condition. No more data can be written until it is deasserted. SrcFFWrEn_n is ignored if SrcFFOverflow_n is asserted.

Source Status and Flow Control Interface

Source Calendar Control Interface

The Source core calendar control interface determines the received status channel order and frequency. Through this interface, the user can program the calendar buffer that determines the order and frequency in which a channel status is received on the SPI-4.2 interface. This interface can verify received DIP-2 parity and flag errors. Table 10 defines the calendar control interface signals.

Table 10: **Source Calendar Control Signals**

Name	Direction	Clock Domain	Description
SrcCalClk	Input	n/a	**Source Calendar Clock:** All Source calendar signals are synchronous to this clock.
SrcCalWrEn_n	Input	SrcCalClk	**Source Calendar Write Enable:** When this signal is asserted (Active Low), the Source Calendar is loaded with the data on the SrcCalData bus on the rising edge of SrcCalClk.
SrcCalAddr[8:0]	Input	SrcCalClk	**Source Calendar Address:** When SrcCalWrEn_n is asserted, this bus indicates the calendar address to which the data on SrcCalData is written. When SrcCalWrEn_n is deasserted, this bus indicates the calendar address from which the data on SrcCalDataOut is driven.
SrcCalData[7:0]	Input	SrcCalClk	**Source Calendar Data:** This bus contains the channel number to write into the calendar buffer when SrcCalWrEn_n is enabled. The channel numbers written into the calendar indicate the order that status is updated on the SrcStat bus.
SrcCalDataOut[7:0]	Output	SrcCalClk	**Source Calendar Data Output:** This Source Calendar Data Output bus contains the channel number that is read from the calendar buffer when SrcCalWrEn_n is disabled. The channel numbers read from the calendar indicates the order that status is updated on SrcStat bus.

Source Status FIFO Interface

The Source core Status FIFO interface enables the user to receive flow control data from the SPI-4.2 interface and provides the option to present status information on the FIFO interface in one of two ways:

- **Addressable Status Interface**. This interface allows the user to access the status of 16 channels in one clock cycle. Status is processed and stored in the Source core.

- **Transparent Status Interface**. This interface presents the status to the user interface as it is received on TStat[1:0], with minimal latency. It provides the ideal interface to customize the processing of received status information.

Table 11 defines the Status FIFO interface signals.

Table 11: **Source Status FIFO Signals**

Name	Direction	Clock Domain	Description
SrcStatClk (Addressable I/F Only)	Input	n/a	**Source Status Clock:** For the *Addressable Interface*, all Source Status read signals are synchronous to this clock. For the *Transparent Interface*, this clock signal is not present. For this interface, all signals are synchronous to TSClk_GP.
SrcStat[31:0] (Addressable I/F Only) SrcStat[1:0] (Transparent I/F Only)	Output	SrcStatClk (Addressable I/F only) TSClk_GP (Transparent I/F only)	**Source Status**: For the *Addressable Interface*, the 32-bit Source Status bus is the dedicated 16-channel interface. The user can read the status for 16-channels each clock cycle. The 16-channel status that are accessed simultaneously are grouped in the following manner: channel 15 to 0, channel 31 to 16, channel 47 to 32, ..., channel 255 to 240. For the *Transparent Interface,* this Source Status bus is two bits wide and represents the last status received.
SrcStatAddr[3:0] (Addressable I/F Only)	Input	SrcStatClk	**Source Status Address**: For the *Addressable Interface*, the Source Status Address determines which group of 16-channels gets its status driven onto SrcStat on the following clock cycle. The address bus is associated with banks of channels as follows: Bank 0: SrcStatAddr=0 channel 15-0 Bank 1: SrcStatAddr=1, channel 31-16 Bank 2: SrcStatAddr=2, channel 47-32 ... Bank 15: SrcStatAddr=15 channel 255-240 For the *Transparent Interface*, this signal is not present.
SrcStatCh[7:0]	Output	TSClk_GP	**Source Status Channel**: The Source Status Channel is an 8-bit bus containing the channel address that is being updated on the SrcStatAddr bus in the current clock cycle.
SrcStatChValid	Output	TSClk_GP	**Source Status Channel Valid:** When asserted, Source Status Channel Valid indicates that the value on SrcStatCh is valid. When the core is processing DIP-2 or frame words, SrcStatChValid is deasserted. Note that a transition of the SrcStatChValid from 0 to 1 indicates that the core has started a new calendar sequence.

Source Static Configuration Interface

The Source core static configuration signals enables customization of the core based on individual system requirements. These input signals are statically driven by setting them to a constant value in the top-level wrapper file. Three of the Source static configuration signals can be changed in circuit: `SrcBurstLen`, `SrcCalendar_M,` and `SrcCalendar_Len`. `SrcBurstLen` is a static register synchronous to `SnkFFClk`; `SrcCalendar_M` and `SrcCalendar_Len` are static registers that are synchronous to `SrcStatClk`. Table 12 defines the static configuration signals.

Table 12: **Source Static Configuration Signals**

Name	Direction	Range	Description
SrcBurstMode	Static Input	0 or 1	**Source Burst Mode**: When SrcBurstMode is set to 0, the Source core transmits data in the FIFO if the data in the FIFO is terminated by an EOP or if there is a complete credit of data. When SrcBurstMode is set to 1, the Source core only transmits data that is terminated by an EOP or when there is data in the FIFO equal to the maximum burst length defined by SrcBurstLen.
SrcBurstLen[5:0]	Input	1-63 Values equal to 0 are set to 1.	**Source Burst Length**: The Source core automatically segments packets larger than this parameter into multiple bursts, which are each SrcBurstLen in length. This parameter is defined in credits (16 bytes). The core implements this parameter as a static register synchronous to SrcFFClk, and it can be updated in circuit by first deasserting SrcEn.
SrcAFThresAssert[8:0]	Static Input	If SrcBurstMode = 0 1–508 Values less than 1 are set to 1. Values greater than 508 are set to 508. If SrcBurstMode = 1 SrcBurstLen to 508. Values less than SrcBurstLen get set to SrcBurstLen. Values greater than 508 are set to 508.	**Source Almost Full Threshold Assert**: The SrcAFThresAssert parameter specifies the minimum number of empty FIFO locations that exist in the Source FIFO before the Almost Full signal (SrcFFAlmostFull_n) is asserted. If SrcBurstMode = 0, then SrcAFThresNegate is greater than or equal to SrcAFThresAssert. If SrcBurstMode = 1, then: (1) SrcAFThresNegate is greater than or equal to SrcAFThresAssert (2) SrcAFThresNegate and SrcAFThresAssert are greater than or equal to SrcBurstLen.

Table 12: **Source Static Configuration Signals** *(Cont'd)*

Name	Direction	Range	Description
SrcAFThresNegate[8:0]	Static Input	SrcAFThresAssert to 508 Values less than SrcAFThresAssert are set to SrcAFThresAssert. Values greater than 508 are set to 508.	**Source Almost Full Threshold Negate**: The SrcAFThresNegate parameter specifies the minimum number of empty FIFO locations that exist in the Source FIFO before the Almost Full signal (SrcFFAlmostFull_n) is deasserted. If SrcBurstMode=0, then: SrcAFThresNegate is greater than or equal to SrcAFThresAssert. If SrcBurstMode=1, then: (1) SrcAFThresNegate is greater than or equal to SrcAFThresAssert (2) SrcAFThresNegate and SrcAFThresAssert are greater than or equal to SrcBurstLen
SrcCalendar_M[7:0]	Input	0–255 (effective range 1-256)	**Source Calendar Period**: The SrcCalendar_M parameter sets the number of repetitions of the calendar sequence before the DIP-2 parity and framing words are received. The Source core implements this parameter as a static register synchronous to SrcStatClk, and it can be updated in circuit by first deasserting SrcEn. Note that the Source Calendar Period equals SrcCalendar_M + 1. For example, if SrcCalendar_M=22, the Source Calendar Period will be equal to 23.
SrcCalendar_Len[8:0]	Input	0–511 (effective range 1–512)	**Source Calendar Length**: The SrcCalendar_Len parameter sets the length of the calendar sequence. The Source core implements this parameter as a static register synchronous to SrcStatClk, and it can be updated in circuit by first deasserting SrcEn. Note that the Source Calendar Length equals SrcCalendar_Len + 1. For example, if SrcCalendar_Len=15, the Source Calendar Length will be equal to 16.
DataMaxT[15:0]	Static Input	0, 16–65535	**Maximum Data-Training Interval:** Maximum interval between scheduling of training sequences on the SPI-4.2 data path (in SPI-4.2 bus cycles). Note that setting DataMaxT to zero configures the core to never send periodic training.

Table 12: **Source Static Configuration Signals** *(Cont'd)*

Name	Direction	Range	Description
AlphaData[7:0]	Static Input	0–255	**Data Training Pattern Repetitions**: Number of repetitions of the 20-word data training pattern. Note that setting AlphaData to zero configures the core to not periodically send training patterns. In this case, the user can manually send training patterns by asserting the TrainingRequest command.
NumDip2Errors[3:0]	Static Input	1–15 Value equal to 0 are set to 1	**Number of DIP-2 Errors**: The Source Interface will go out-of-frame (SrcOof asserted) and stop transmitting SPI-4.2 data across the data bus after receiving NumDip2Errors consecutive DIP-2 errors.
NumDip2Matches[3:0]	Static Input	1–15 Value equal to 0 are set to 1	**Number of DIP-2 Matches**: The Source Interface requires NumDip2Matches consecutive DIP-2 matches before going in-frame and beginning to transfer SPI-4.2 data across the SPI-4.2 data bus.

Sink Data Capture Implementation

The Sink Core performs static alignment by shifting the clock relative to the 16-bit data such that the incoming clock edge is centered to the data eye of RDat/RCtl. Static alignment can be implemented using a DCM, MMCM, or the Virtex device ChipSync™ IDELAY function.

Static Alignment Using DCM or MMCM

For all Virtex and Spartan families, a DCM or MMCM can be used to phase-shift the RDClk. This DCM- or MMCM-based static alignment is only supported for global clocking distribution. The ability of the DCM or MMCM to shift the internal clock in small increments enables the RDClk to be shifted relative to the sampled data. For statically-aligned systems, the DCM or MMCM output clock phase offset is a critical part of the system. The static alignment solution using the DCM or MMCM assumes that the PCB is designed with precise delay and impedance matching for all LVDS differential pairs of the data bus. This assumption is critical as the DCM or MMCM does not compensate for deviations in delay between bits.

Static Alignment Using ChipSync IDELAY

For Virtex-6, Virtex-5, and Virtex-4 FPGA designs, static alignment can also be performed using the ChipSync IDELAY function of the Virtex-6, Virtex-5 or Virtex-4 FPGA ISERDES. This configuration is available when Sink user clocking mode with regional clocking distribution is selected. The ability of the IDELAY function to delay its input by small increments (75 ps), enables the internal RDClk to be shifted relative to the sampled data. For statically aligned systems, the delay chain length is a critical path of the system. The static alignment solution assumes that the PCB is designed with precise delay and impedance matching for all LVDS differential pairs of the data bus. This assumption is critical because time-shifting the internal RDClk relative to the data bits using the IDELAY function does not compensate for deviations in delay between bits. In a similar fashion, small timing variations across the

bus can be manually compensated for using the IDELAY function on a bit by bit basis. When IDELAY is used to perform static alignment, a 200 MHz reference clock input to the user clocking module is required to ensure the 75 ps IDELAY tap delay.

Clocking Options

The SPI-4.2 Lite solution provides several clocking options for the Sink and Source cores, providing users with the flexibility to choose the most suitable option for their system. In addition to regional and global clocking distribution, the user also has the option to choose between embedded clocking and slave (user) clocking. Depending on the chosen clocking distribution, different clock resources are used. The embedded clocking resource count for each clock distribution option, not including the user interface clocks, is detailed in Table 13, Table 14, and Table 15.

Table 13: **Sink Core Clocking Distribution Resources**

Clocking Distribution	BUFR	BUFG	DCM/MMCM
Global clocking for Virtex-6 devices	0	2	1
Global clocking for devices other than Virtex-6	0	1	1
Regional clocking	1	1 [1]	0

1. The Sink core requires the SnkIdelayRefClk clock (200 MHZ reference clock) to be driven by a global clock buffer. This reference clock provides a time reference to IDELAYCTRL modules to calibrate all the individual delay elements (IDELAY) in the region. Multiple cores need only one global clock buffer to distribute the SnkIdelayRefClk clock.

Table 14: **SysClk Clocking Distribution Resources**

Clocking Distribution	BUFR	BUFG	DCM/MMCM
Global Clocking for Virtex-6	0	2	1
Global Clocking for Devices Other Than Virtex-6	0	1	1
Regional Clocking	1	0	0

Table 15: **TSClk Clocking Option Resources**

Clocking Distribution	BUFR	BUFG	DCM/MMCM
Global Clocking	0	1	0
Regional Clocking	1	0	0

Embedded Clocking

The SPI-4.2 Lite core, targeting the Virtex-4 or Virtex-5 families, supports embedded global and regional clocking distribution for the Source core, and embedded global clocking distribution for the Sink core. Global clocking uses dedicated global (chip) routing; regional clocking uses clock-region specific resources. All other families provide embedded global clocking distribution for both Sink and Source cores.

Source Slave Clocking

In addition to the embedded clocking modes described above, the SPI-4.2 Lite solution includes the option to generate the Source core in slave clocking mode for all architectures with the exception of Virtex-6 and Spartan-6 devices. This enables the user to implement an external clocking module to generate the Source core clocks. This clocking module can be used to customize the clocking logic for the user application. The module can also be used to drive the clock inputs for multiple Source cores configured in slave clocking mode.

Source User Clocking

When targeting the Spartan-6 and Virtex-6 families, the SPI-4.2 Lite solution generates the source in user clocking mode. Like the slave clocking mode, the user clocking mode enables the user to implement an external clocking module to generate the Source core clocks. This clocking module can be used to customize the clocking logic for the user application and drive clock inputs for multiple Source cores.

Sink User Clocking

The SPI-4.2 Lite solution also provides the option to generate the Sink core in user clocking mode. Like the Source core, this mode enables the user to implement an external clocking module to generate the Sink core clocks. This clocking module can be used to customize the clocking logic for the user application and also supports regional clocking in Virtex-6, Virtex-5, and Virtex-4 devices. For designs targeting Spartan-6 and Virtex-6 families, only sink user clocking option is available. See the *SPI-4.2 Lite User Guide* for more information regarding the Sink user clocking option.

Multiple Core Instantiations

The user has flexibility in implementing multiple SPI-4.2 Lite cores in a single target device. Larger devices can support more than four SPI-4.2 Lite cores. See the *SPI-4.2 Lite User Guide* for information.

Verification

Extensive software testing with an internally developed verification platform is performed for each SPI-4.2 Lite release.

Using the in-house verification environment, the SPI-4.2 Lite core was tested in three stages:

- Functional (RTL) verification
- Gate-level (post ngdbuild back-annotation HDL) verification
- Gate-level with back-annotated Timing (with SDF file) verification targeting the following device/frequency combinations:
  - Virtex-6 devices up to 550 Mbps on the SPI-4.2 interface and 275 MHz on the user interface (SrcFFClk and SnkFFClk) clocks
  - Spartan-6 devices up to 210 Mbps on the SPI-4.2 interface and 105 MHz on the user interface (SrcFFClk and SnkFFClk) clocks
  - Virtex-5 devices up to 550 Mbps on the SPI-4.2 Interface and 275 MHz on the user interface (SrcFFClk and SnkFFClk) clocks
  - Virtex-4 devices up to 380 Mbps on the SPI-4.2 interface and 190 MHz on the user interface (SrcFFClk and SnkFFCClk) clocks

- ♦ Spartan-3E devices up to 180 Mbps on the SPI-4.2 interface and 90 MHz on the user interface (`SrcFFClk` and `SnkFFCClk`) clocks
- ♦ Spartan-3A/3AN/3A DSP devices up to 210 Mbps on the SPI-4.2 interface and 105 MHz on the user interface (`SrcFFClk` and `SnkFFClk`) clocks.
- ♦ Spartan-3 devices up to 230 Mbps on the SPI-4.2 interface and 115 MHz on the user interface (`SrcFFClk` and `SnkFFCClk`) clocks

Performance

Table 16 provides core performance metrics for all supported devices.

Table 16: **Core Performance**

Device	Speedgrade	Performance
Virtex-6	-1	450 Mbps
	-1L	450 Mbps
	-2	550 Mbps
	-3	550 Mbps
Virtex-5	-1	450 Mbps
	-2	550 Mbps
	-3	550 Mbps
Virtex-4	-10	340 Mbps
	-11	380 Mbps
Spartan-6	-2	210 Mbps
Spartan-3A/3AN/3A DSP	-4	210 Mbps
Spartan-3E	-4	180 Mbps
Spartan-3	-4	210 Mbps
	-5	230 Mbps

Device Utilization

Table 17 through Table 21 contain the Block RAM, LUTs, FFs and Slice or LUT/Flop pairs counts for the SPI-4.2 Lite Sink and Source core targeting different architectures.

Table 17: **Virtex-6 FPGA Core Utilization**

Core Type	I/O	LUTs	FFs	Block RAM	LUT/Flop Pair
64-bit Sink	17	909	940	3 (36k BRAM) 1 (18k BRAM)	1268
32-bit Sink	17	720	770	2 (36k BRAM) 1 (18k BRAM)	1032
64-bit Source	18	1134	1490	3 (36k BRAM) 1 (18k BRAM)	1873
32-bit Source	18	1024	1352	2 (36k BRAM) 1 (18k BRAM)	1710

Table 18: **Virtex-5 FPGA Core Utilization**

Core Type	I/O	LUTs	FFs	Block RAM	LUT/Flop Pair
64-bit Sink	18	900	950	4	1100
32-bit Sink	18	750	750	3	950
64-bit Source	18	1200	1500	4	1750
32-bit Source	18	1200	1380	3	1650

Table 19: **Virtex-4 FPGA Core Utilization**

Core Type	I/O	LUTs	FFs	Block RAM	Slices
64-bit Sink	18	1100	950	4	700
32-bit Sink	18	900	800	3	600
64-bit Source	18	1450	1500	4	1100
32-bit Source	18	1350	1380	3	1000

Table 20: **Spartan-6 FPGA Core Utilization**

Core Type	I/O	LUTs	FFs	Block RAM	Slices
64-bit Sink	18	820	940	4	1250
32-bit Sink	18	770	770	3	1080
64-bit Source	18	1105	1490	4	1880
32-bit Source	18	1020	1360	3	1730

Table 21: **Spartan-3, Spartan-3E, and Spartan-3A/3AN/3A DSP FPGA Core Utilization**

Core Type	I/O	LUTs	FFs	Block RAM	Slices
64-bit Sink	18	1100	1000	4	750
32-bit Sink	18	900	800	3	650
64-bit Source	18	1250	1500	4	950
32-bit Source	18	1200	1380	3	1000

References

1. PMC-Sierra, Inc., POS-PHY™ Level-4, *A Saturn Packet and Cell Interface Specification for OC-192 SONET/SDH and 10 Gb/s Ethernet Applications*, Issue 5: June 2000.

2. Optical Internetworking Forum (OIF), *OIF-SPI4-02.1 System Packet Interface Level-4 (SPI-4) Phase 2 Revision 1: OC-192 System Interface for Physical and Link Layer Devices.*

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

The SPI-4.2 Lite core is provided under the SignOnce IP Site License and can be generated using the Xilinx CORE Generator system v12.1 or higher. The CORE Generator system is shipped with Xilinx ISE Foundation™ Series Development software.

A simulation evaluation license for the core is shipped with the CORE Generator system. To access the full functionality of the core, including FPGA bitstream generation, a full license must be obtained from Xilinx. For more information, please visit the SPI-4.2 Product Page.

Please contact your local Xilinx sales representative for pricing and availability of additional Xilinx cores and software. 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	1.0	Initial Xilinx release.
1/18/06	1.1	Updated the release date, product version and Xilinx tools version.
7/13/06	2.0	Added support for Virtex-5 devices, updated product version to 4.1, release date, ISE to v8.2i
2/15/07	3.0	Added support for Spartan-3A/3AN devices, product version advanced to 4.2, ISE to 9.1i.
4/07/07	3.1	Added support for Spartan-3A DSP devices.
6/27/08	4.0	Updated for ISE v10.1 release.
4/24/09	4.4	Updated core to v4.4 and ISE to v11.1.
6/24/09	4.5	Added support for Virtex-6 and Spartan-6 devices. Removed support for Virtex-II and Virtex-II Pro devices. Updated core to v5.1 and ISE to v11.2.
4/19/10	5.0	Updated core to v5.2 and ISE to v12.1.

Notice of Disclaimer

Xilinx is providing this product documentation, hereinafter "Information," to you "AS IS" with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.