LogiCORE IP SPI-4.2 v10.5

DS209 June 22, 2011 **Product Specification**

Introduction

The Xilinx SPI-4.2 (PL4) core implements and is fully 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 10 Gbps POS, ATM and Ethernet applications. The core leverages SelectIO™ features to achieve both smaller and faster SPI-4.2 products, which enables higher-level functions such as switches, bridges, and NPU interfaces.

Features

- Up to 600 MHz DDR on SPI-4.2 interface supporting 1.2 Gbps pin pair total bandwidth
- Supports Static and Dynamic Phase Alignment utilizing ChipSync™ technology
- Bandwidth optimized Source core achieves optimal bus throughput without additional FPGA resources
- Flexible clocking options utilizing DCM, PMCD, global, and regional clocking resources
- SelectIO technology supports flexible pin assignment
- Configurable 64-bit or 128-bit user interface, both supporting full bandwidth capabilities
- Supports unsegmented burst sizes up to 16K
- Faster Dynamic-Phase Alignment (DPA) algorithm
- Optional continuous DPA window monitoring
- Optional advanced DPA diagnostics
- Lower power consumption than Virtex-II FPGA solutions
- Multiple core support: more than 4 cores can be implemented in a single device
- Sink and Source cores configured through Xilinx CORE Generator™ system for easy customization
- Supports all Virtex®-6, Virtex-5 and Virtex-4 device and package configurations
- Delivers Sink and Source cores as independent solutions — enabling flexible implementation
- Supports 1 to 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
- DIP-4 and DIP-2 parity generation and verification

LogiCORE IP Facts				
Core Specifics				
Device Family		Virtex-6 (1), Virtex-5, Virtex-4		
Resources Used (2)				
Alignment Type		**Performance (Mbps)/ Speed**	**LUT /Flop Pairs**	**Block RAM**
FPGAs	64-bit static	622–700/-1L, -1, -2, -3	Rx: 2200 Tx: 2780	Rx: 1 (36k BRAM) 5 (18k BRAM) Tx: 6 (18k BRAM)
	128-bit static	622–700/-1L, -1, -2, -3	Rx: 3075 Tx: 2750	Rx: 1 (36k BRAM) 9 (18k BRAM) Tx: 6 (18k BRAM)
	64-bit dynamic (3)	622-900/ -1L 622-1.1 Gbps/ -1 622-1.25 Gbps/ -2 622-1.4 Gbps/ -3	Rx: 2650 Tx: 2780	Rx: 1(36k BRAM) 5 (18k BRAM) Tx: 6 (18k BRAM
	128-bit dynamic (3)	622-900 Mbps/-1L 622-1 Gbps/-1 622-1.1 Gbps/-2 622-1.2 Gbps/-3	Rx: 3800 Tx: 2750	Rx: 1 (36k BRAM) 9 (18k BRAM) Tx: 6 (18k BRAM)
Provided with Core				
Documentation		User Guide Release Notes		
Design File Formats		NGC file		
Constraints		Example UCF and Embedded RPMs		
Verification Test Bench		VHDL and Verilog		
Reference Design		SPI-4.2 to Quad SPI-3		
Design Tool Requirements				
Implementation		Xilinx ISE v13.2		
Simulation(4)		Mentor Graphics ModelSim Cadence IES Synopsys VCS and VCS MX		
Synthesis(4)		XST, ISE, Synplicity Synplify Pro		
Provided by Xilinx, Inc.				

1. Includes Virtex-6-1L devices.
2. Numbers are for default configurations in Virtex-6 devices. See Table 20 through Table 29 for a complete description of resource utilization and performance by configuration and device family.
3. Listed DPA resources include optional continuous DPA logic.
4. For the supported versions of the tools, see the ISE Design Suite 13: Release Notes Guide.

Applications

The SPI-4.2 (PL4) interface core enables the connection of physical-layer devices to link-layer devices in 10 Gbps POS, ATM, and Ethernet applications. The symmetric interface can be used to implement both the PHY and Link layer.

Figure 1 shows 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 for a wide range of multi-service DWDM and SONET/SDH-based transmission systems.

The OIF SPI4-02.1 interface is widely used to connect network processors (such as the Intel IXP2800) with OC-192 framers and mappers, as well as Gigabit and 10-Gigabit Ethernet data link MACs. The Xilinx SPI-4.2 core is an implementation of this high-performance, low-pin-count data transfer protocol that is ideally suited for these applications.

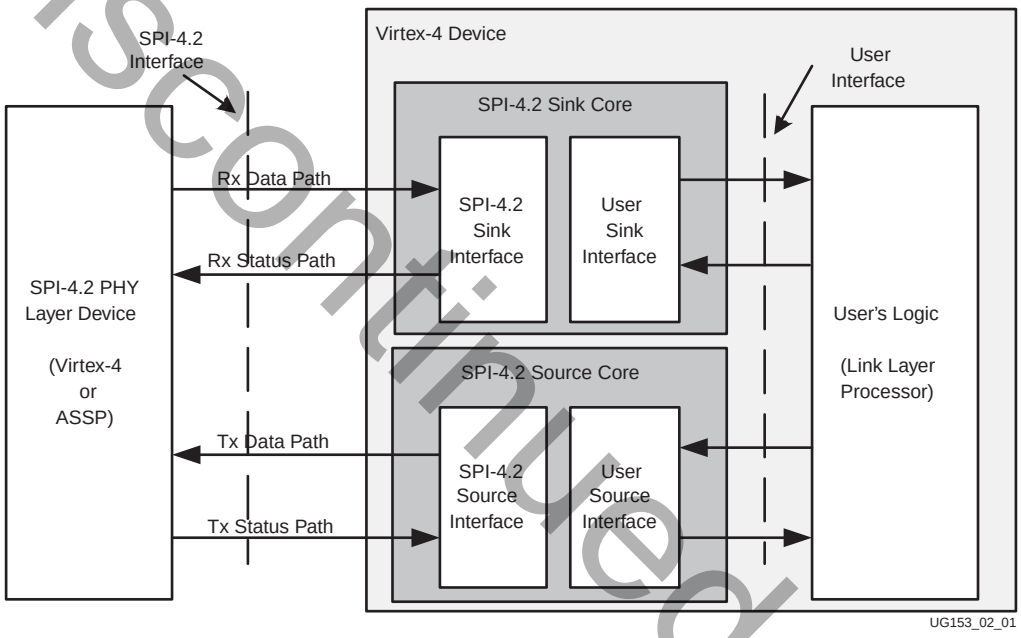

Figure 1: **SPI-4.2 Core in a Typical Link-Layer Application**

Functional Overview

The SPI-4.2 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 64-bit or 128-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

operational state of the core 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.

Figure 2 shows input and output signals and the functional blocks of the Sink core. The interface signals to each of the functional modules are described in detail in Core Interfaces, page 5..

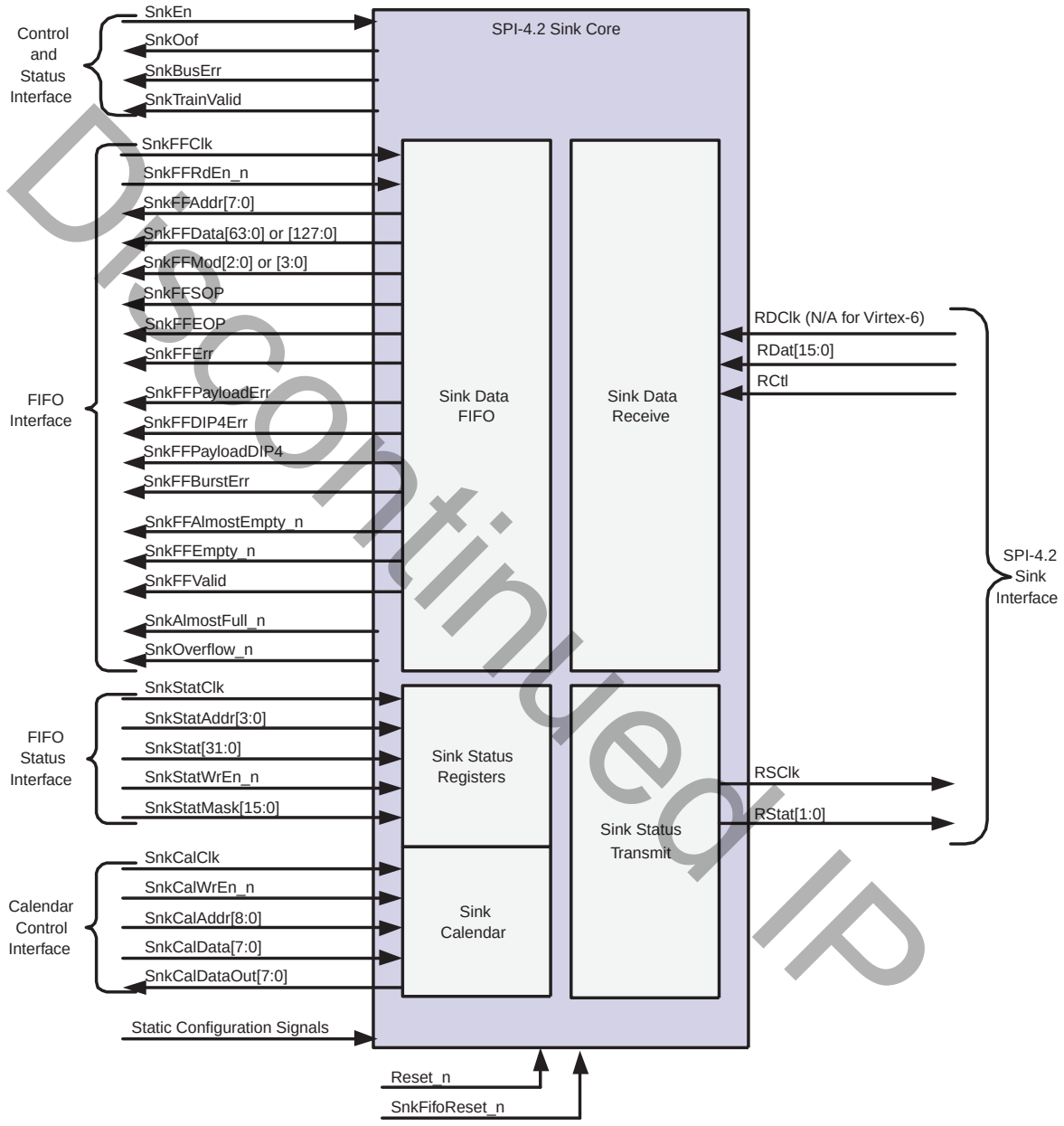

Figure 2: **Sink Core Block Diagram and I/O Interface Signals**

Source Core

The Source core transmits 16-bit source synchronous data on its SPI-4.2 interface by processing and formatting 64-bit or 128-bit data words from the user interface. The core also processes 2-bit status (for each channel) on the SPI-4.2 interface appearing on the user interface. In addition to data, other signals associated with operational state 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. Figure 3 shows input and output signals and functional blocks of the Source core. The interface signals to each of the functional modules are described in Core Interfaces, page 5.

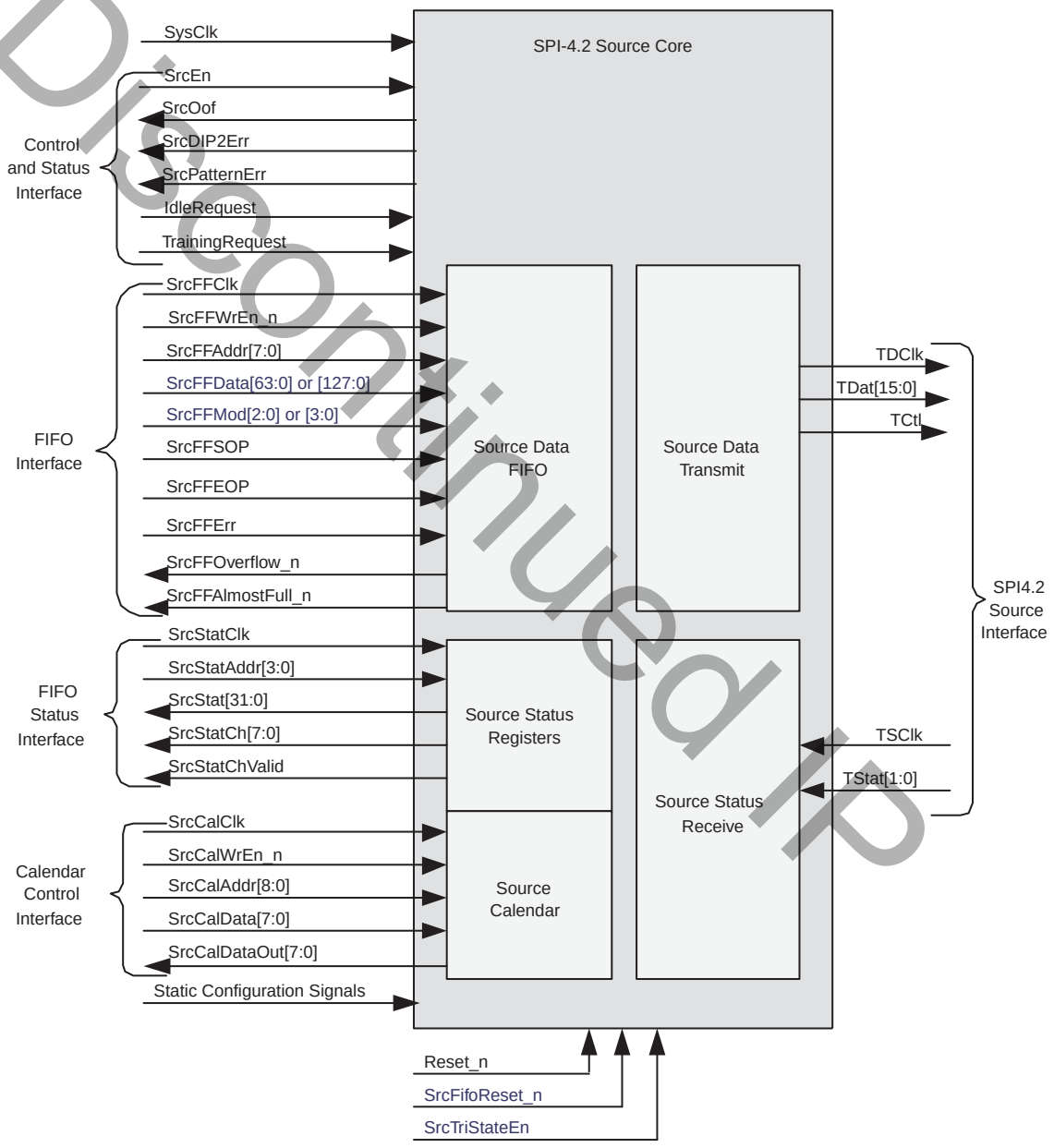

Figure 3: **Source Core Block Diagram and I/O Interface Signals**

Core Interfaces

This section provides definitions of the interface signals for 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 contains the Sink SPI-4.2 interface signals.

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 the following 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 specific channel (port) or packet. Table 2 defines these Sink control and status interface signals.

Table 2: **Sink Control and Status Interface 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.** Active low signal that 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.
SnkIdelayRefClk (optional)	Input	n/a	**Reference Clock.** User-supplied 200 MHz reference clock. This reference clock provides a time reference to the IDELAYCTRL modules to calibrate the individual delay elements (IDELAY) in the clock region. This clock must be routed on a global clock buffer. The SnkIdelayRefClk signal is present when option "Include IDELAYCTRL modules" is selected and must be connected to a user-supplied 200 MHz clock.
SnkIdelayCtlRst (optional)	Input	n/a	**Dedicated IDELAYCTRL Reset**. Active high signal that resets all the IDELAYCTRL primitives embedded in the Sink core. If not used, the Reset_n signal resets all IDELAYCTRL primitives. This signal is present when the "Include IDELAYCTRL modules" option is selected.
SnkIdelayCtlRdy	Input	RDClkDiv_GP	**IDELAYCTRL Ready.** Active high signal that indicates when the IDELAY modules are calibrated. The SnkIdelayCtlRdy signal is present when option "Include IDELAYCTRL modules" is not selected. When option "Include IDELAYCTRL modules" is not selected, the IDELAYCTRLs must be instantiated in the wrapper by the user to calibrate the IDELAYs connected to RDat[15:0], RCtl, RDClk. Additionally, all the ready signals from these IDELAYCTRLs must be ANDed together to provide the signal that connects the SnkIdelayCtlRdy signal.
SnkOof	Output	SnkFFClk	**Sink Out-of-Frame**. Active high signal indicating that the SPI-4.2 Sink block is not in frame. This signal is asserted when SnkEn is deasserted or the Sink block loses synchronization with the data received on the SPI-4.2 Interface. This signal is deasserted once the Sink block reacquires synchronization with the received SPI-4.2 data.

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

Name	Direction	Clock Domain	Description
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
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). When dynamic phase alignment configuration is used, SnkBusErrStat can be used to monitor the alignment status. See the *SPI-4.2 User Guide* for more information.
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 bus cycles or 5 RDClkDiv_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 64-bit or 128-bit data word, control and status signals (including error signals) are associated with a particular channel or packet. For example, these status signals will flag improper packet format, DIP4 error, and FIFO Status. Table 3 provides the Sink FIFO interface signals and a description of each.

Table 3: **Sink FIFO Interface 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[127:0]	Output	**Sink FIFO Data Out**. The Sink FIFO data bus. Bit 0 is the LSB. The core can be configured to have a 64-bit or 128-bit Interface. The 128-bit interface enables the user to run at half the clock rate required for a 64-bit interface.

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

Name	Direction	Description
SnkFFMod[2:0] or SnkFFMod[3:0]	Output	**Sink FIFO Modulo**. This signal indicates which bytes on the SnkFFData bus are valid when the SnkFFEOP signal is asserted. SnkFFMod[2:0] is used with a 64-bit interface. SnkFFMod[3:0] is used with a 128-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.
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 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 core in the Design Example (see the pl4_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, SnkFFPayloadErr and SnkFFPayloadDIP4 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 DIP4 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 core 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 Calendar Control Interface

The Sink core calendar control interface determines status channel order and frequency. Through this interface, the user can program the calendar buffer to determine the order and frequency with 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 describes 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 may be automatically or manually implemented. Table 5 describes 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 DIP2 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 DIP2 value that it transmits.

Table 5: **Sink Status FIFO Signals *(Cont'd)***

Name	Direction	Clock Domain	Description
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 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. 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 static configuration signals.

Table 6: **Sink Static Configuration Signals**

Name	Direction	Range	Description
NumDip4Errors[3:0]	Static Input	1-15; Value of 0 gets set to 1.	**Number of DIP-4 Errors**. The Sink interface goes out-of-frame (assert SnkOof) and stops 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 gets 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.
SnkAFThresAssert[8:0]	Static Input	1-508 Values less than 1 get set to 1. Values greater than 508 get set to 508.	**Sink Almost Full Threshold Assert**. Defines the minimum number of empty FIFO locations that exist when SnkAlmostFull_n is asserted. 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.

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

Name	Direction	Range	Description
SnkAFThresNegate[8:0]	Static Input	SnkAFThresAssert to 508 Values less than SnkAFThresAssert get set to SnkAFThresAsset. Values greater than 508 get set to 508.	**Sink Almost Full Threshold Negate**. 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 (deasserts SnkAlmostFull_n) 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**. 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**. 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
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 Interface

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

Table 7: **Source SPI-4.2 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 the following 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 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, and causes the core to 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**. 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, SrcPatternErr and SrcStatFrameErr. 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 SPI-4.2 Source block is not in frame. This signal is asserted when the Source block has lost 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 block 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.
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 FIP2 on TStat. This signal is asserted for one clock cycle each time a frame word error is detected.
SrcPatternErr	Output	SrcFFClk	**Source Data Pattern Error.** When asserted (active high), 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.

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

Name	Direction	Clock Domain	Description
IdleRequest	Input	SrcFFClk	**Idle Request**.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 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**. 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**. 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.)
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.

Source FIFO Interface

The Source core FIFO interface stores data from the user logic to be transmitted on the SPI-4.2 interface. In addition to the 64-bit or 128-bit data word, there are control and status signals (including error signals) associated with a particular channel or packet. These include signals to insert DIP4 errors, idles, training patterns, and so forth. 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[127:0]	Input	SrcFFClk	**Source FIFO Data**. The Source FIFO data bus. Bit 0 is the LSB. The core can be configured to have a 64-bit or a 128-bit interface. The 128-bit interface enables the user to run at half the clock rate required for a 64-bit interface.
SrcFFMod[2:0] or SrcFFMod[3:0]	Input	SrcFFClk	**Source FIFO Modulo.** 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[2:0] is used with a 64-bit interface. SrcFFMod[3:0] is used with a 128-bit interface.
SrcFFSOP	Input	SrcFFClk	**Source FIFO Start of Packet**. When asserted (active high), 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), 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 DIP4 value is an inversion of the correctly calculated value.
SrcFFAlmostFull_n	Output	SrcFFClk	**Source FIFO Almost Full**. When asserted (active low), 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), 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 Calendar Control Interface

The Source core calendar control interface determines 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 calendar control interface signals.

Table 10: **Source Calendar Control Interface 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 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**. Provides user access to the status of 16 channels in one clock cycle. Status is processed and stored in the Source core.
- **Transparent Status Interface**. Presents status to the user interface as it is received on TStat[1:0], with minimal latency. It provides the ideal interface for users to customize how they process the received status information. Table 11 defines 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.** Addressable Interface–all Source Status read signals are synchronous to this clock. Transparent Interface–this clock signal is not present. 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**. 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. 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**. 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 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, indicates that the value on SrcStatCh is valid. When the core is processing DIP-2 or frame words, SrcStatChValid is deasserted. 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 the user to customize 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` is a static register synchronous to `SnkFFClk`, `SrcCalendar_M` and `SrcCalendar_Len` are static registers that are synchronous to `SrcStatClk`. Table 12 defines static configuration signals.

Table 12: **Source Static Configuration Signals**

Name	Direction	Range	Description
SrcBurstMode	Static Input	0 or 1	**Source Burst Mode**. When set to zero, the Source core transmits data in the FIFO if the data is terminated by an EOP or if there is a complete credit of data. When set to1, 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, or when the channel address changes.
SrcBurstLen[9:0]	Input	1 to 1023 (SrcBurstMode = 1) 1 to 256 (SrcBurstMode = 0) Values equal to 0 gets 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 to 508 Values less than 1 get set to 1. If SrcBurstMode = 1 1 to 508 Values greater than 508 get set to 508.	**Source Almost Full Threshold Assert**. Specifies the minimum number of empty FIFO locations to exist in the Source FIFO before the Almost Full signal (SrcFFAlmostFull_n) is asserted. SrcAFThresNegate ≥ SrcAFThresAssert.
SrcAFThresNegate[8:0]	Static Input	SrcAFThresAssert to 508 Values less than SrcAFThresAssert get set to SrcAFThresAssert. Values greater than 508 get set to 508.	**Source Almost Full Threshold Negate**. Specifies the minimum number of empty FIFO locations to exist in the Source FIFO before the Almost Full signal (SrcFFAlmostFull_n) is deasserted. SrcAFThresNegate is ≥ SrcAFThresAssert

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

Name	Direction	Range	Description
SrcCalendar_M[7:0]	Input	0-255 (effective range 1–256)	**Source Calendar Period**. 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**. 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.
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 gets 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 gets 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 SPI-4.2 core uses SelectIO™ technology to implement dynamic-phase alignment (DPA) and static-phase alignment as defined by the SPI-4.2 OIF specification. Virtex-6, Virtex-5, and Virtex-4 FPGAs include ChipSync technology to enhance I/O capability when used in source-synchronous applications like SPI-4.2.

The SPI-4.2 DPA solution leverages ChipSync technology to perform data-eye detection and word alignment. Using this dedicated hardware reduces the FPGA slice resources required for DPA (~50%) and provides precision data sampling. Additionally, the ChipSync capability is supported in every I/O pin, enabling full flexibility in selecting the pin-out for the SPI-4.2 interface. The SPI-4.2 DPA solution is available for Virtex-6, Virtex-5 and Virtex-4 FPGAs.

Virtex-6, Virtex-5, and Virtex-4 FPGAs provide an abundance of clock resources for implementing multiple SPI-4.2 cores in a single device. These clock resources are available for both dynamic and static alignment, enabling the user to select the clocking scheme best suited to their design. The user interface and I/O pinout of the core remain the same, regardless of the type of alignment implemented.

IDELAYCTRL Module

The SPI-4.2 core uses ChipSync IDELAY function in Virtex-6, Virtex-5 and Virtex-4 FPGAs to perform data-eye detection. The IDELAYCTRL module continuously calibrates the IDELAY elements in its region to reduce the effects of process, voltage, and temperature variations. The IDELAYCTRL modules exist in every I/O column in every clock region.

The SPI-4.2 solution for Virtex-4 FPGAs includes the "Include IDELAYCTRL modules" option. When this option is selected, the IDELAYCTRL modules are embedded in the SPI-4.2 core. If this option is not selected, the IDELAYCTRL must be manually instantiated in the user design.

For Virtex-6 and Virtex-5 FPGAs, the "Include IDELAYCTRL modules" option is not supported. The IDELAYCTRL modules must be instantiated in the user design. See the *SPI-4.2 User Guide* for more information.

Dynamic Alignment

The Sink core can be configured to support dynamic-phase alignment (DPA) on the incoming source synchronous SPI-4.2 data stream (`RCtl` and `RDat[15:0]` with respect to the `RDClk`). DPA provides increased system timing margin on the SPI-4.2 interface by removing data skew across the ingress SPI-4.2 bus as part of an interface timing budget. The DPA solution leverages ChipSync technology to achieve an ~50% reduction (compared to Virtex-II FPGA solutions) in the required FPGA fabric resources. Because of this, it is the recommended alignment solution.

In Virtex-6, Virtex-5 and Virtex-4 devices, DPA dynamic alignment is a two-phase process. First, the implementation determines the ideal sampling point for each incoming bit, independent of the timing of any other bit, by using the IDELAY/ISERDES functions of the ChipSync. The second phase uses the SPI-4.2 training pattern to achieve word alignment (bus deskew). The bus deskew phase removes any

skew induced by the independent bit sampling or bus skew on the PCB or backplane. Table 13 defines the Dynamic Phase Alignment signals.

Table 13: **Dynamic Phase Alignment SIgnals**

Signal Name	Direction	Clock Domain	Description
PhaseAlignComplete	Output	SnkFFClk	**Phase Alignment Complete.** Active high signal that indicates phase alignment is complete.
PhaseAlignRequest	Input	SnkFFClk	**Phase Alignment Request.** Initial DPA commences by asserting and deasserting PhaseAlignRequest. DPA starts on a high-to-low transition. When PhaseAlignRequest transitions from low-to-high SnkOof will be driven high (core goes out of frame).
SnkDPAFailed	Output	SnkFFClk	**Phase Alignment Failed**. Active high signal that indicates phase alignment has failed at the end of the alignment sequence.
SnkDPARamAddr [5:0]	Output	RDClkDiv_GP	**Phase Alignment RAM Address.** Bus indicating the ISERDES tap value that corresponds to the data on SnkDPARamData.
SnkDPARamData [16:0]	Output	RDClkDiv_GP	**Phase Alignment RAM Data**. Initial data collected during alignment. Used to find the valid data window for each bit of the SPI-4.2 bus. An active high on the bus indicates that sampling on the ISERDES tap corresponding to SnkDPARamAddr will result in sampling within a valid data window. Each index corresponds to a bit on the SPI-4.2 bus; RDat(0) is index 0, RDat(1) is index 1, ..., RCtl is index 16.
SnkDPARamValid	Output	RDClkDiv_GP	**Phase Alignment RAM Valid**. Active high signal indicating the information on SnkDPARamData and SnkDPARamAddr is valid.
SnkCDPAHalt (optional)	Input	RDClkDiv_GP	**Phase Alignment DPA Halt**. Active high signal that enables the user to halt the pointer adjustment of continuous DPA operation.
SnkDPADiagWin (optional)	Input	RDClkDiv_GP	**Phase Alignment DPA Diagnostics**. Active high signal that enables the user to find the valid data window during operation for each bit of the SPI-4.2 bus. After the SnkDPADiagWin is pulsed, the valid data window information is presented on SnkDPARamAddr [5:0] and SnkDPARamData [16:0] when SnkDPARamValid is asserted.
SnkDPAAddrRst (optional)	Input	RDClkDiv_GP	**Phase Alignment DPA Address Reset.** Active high signal that clears the SnkDPARamAddr counter.
SnkDPAAddrEn (optional)	Input	RDClkDiv_GP	**Phase Alignment DPA Address Enable.** Active high signal that enables the SnkDPARamAddr counter.

Static Alignment

For the Virtex architectures, 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`. For designs using global clock distribution, this alignment can be performed by using the IDELAY function or a DCM. For designs using regional clocking distribution, the IDELAY function is used to shift the clock in relation to the data bits.

Static Alignment Using IDELAY

Static alignment can be performed using the IDELAY function of the ISERDES for either global or regional clocking distribution in Virtex-6, Virtex-5, and Virtex-4 FPGAs. The ability of the IDELAY function to delay its input by small increments enables the internal RDCLK to be shifted relative to the sample 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. In this case, the primary alignment mechanism is time, shifting the internal RDCLK relative to the data bits using the IDELAY function.

Static Alignment Using DCM or MMCM

The core also supports legacy static alignment, which uses the DCM or MMCM to phase shift the RDClk. The DCM- or MMCM-based static alignment is only supported for global clocking distribution in Virtex-6, Virtex-5, and Virtex-4 FPGAs. The ability of the DCM to shift the internal clock, enables 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 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.

Clocking Options

The SPI-4.2 solution provides several clocking options for Sink and Source cores that provide the flexibility to select the most suitable option for any system. Different clock resources are used, depending on the clocking option selected. Table 14 through Table 19 provide the clocking resource count for each clocking option (not including the user interface FIFO clocks) for Virtex-6, Virtex-5 and Virtex-4 FPGAs.

It is required that the source core reference clock (SysClk) has less than 50 ps of jitter since any jitter present on the SysClk input would appear on the TDClk output. Similarly, the duty cycle distortion on the SysClk should also be minimized. For source cores that use DCM or MMCM to generate the internal clocks, it is a requirement that the SysClk duty cycle is 45/55 or tighter. For source cores that use regional clocking scheme or PMCD to generate the internal clocks, it is a requirement that SysClk duty cycle is 48/52 or tighter. To reduce the jitter and duty cycle distortion of the output TDClk, place the clocking and output component as close as possible to each other. See the UG153, *SPI-4.2 User Guide*, for more information.

Table 14: **Sink Core Clocking Options Resources for Virtex-6**

Clocking Option	BUFR	BUFIO	BUFG	MMCM
Global Clocking	0	0	4	1
Regional Clocking	1	1	0	0

Table 15: **Source Core SysClk Clocking Options Resources for Virtex-6 Devices**

Clocking Option	BUFR	BUFIO	BUFG	MMCM
Global Clocking(1)	0	0	4	1
Regional Clocking	1	1	0	0

1. Global clocking is only supported for source cores that send data to Sink cores that are configured with Dynamic Phase Alignment.

Table 16: **Source Core TSClk Clocking Options Resources for Virtex-6 Devices**

Clocking Option	BUFR	BUFIO	BUFG	MMCM
Global Clocking	0	0	3	1
Regional Clocking	1	0	0	0

Table 17: **Sink Core Clocking Option Resources for Virtex-4 and Virtex-5 FPGAs**

Clocking Option	BUFR	BUFG (1)	DCM	PMCD
Regional clocking	1	0/1	0	0
Global clocking with DCM	0	2/3	1	0
Global clocking with DCM Standby Logic (Virtex-4 FPGAs)	0	2/3	1	0
Global clocking with PMCD	0	2/3	0	1

1. The Sink Core requires SnkIdelayRefClk 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.

Table 18: **SysClk Clocking Option Resource for Virtex-4 and Virtex-5 FPGAs**

Clocking Option	BUFR	BUFG	DCM	PMCD
Regional clocking	1	0	0	0
Global clocking with DCM	0	2	1	0
Global clocking with DCM Standby Logic (Virtex-4 FPGAs)	0	2	1	0
Global clocking with PMCD (Virtex-4 FPGAs)	0	2	0	1

Table 19: **TSClk Clocking Option Resources for Virtex-4 and Virtex-5 FPGAs**

Clocking Option	BUFR	BUFG	DCM	PMCD
Regional clocking	1	0	0	0
Global clocking with DCM	0	1	1	0
Global clocking with DCM Standby Logic (Virtex-4 FPGAs)	0	1	1	0
Global clocking with PMCD (Virtex-4 FPGAs)	0	1	0	1
Global clocking without DCM or PMCD	0	1	0	0

The Global clocking option uses dedicated global (chip) routing. The Regional clocking option uses clock region-specific resources.

For Virtex-5 and Virtex-4 devices, the SPI-4.2 solution also includes an option to generate a Source core in slave-clocking mode. This mode enables the implementation of an external clocking module to generate the Source core clocks.

For Virtex-6 FPGAs, both the Sink and Source core support user-clocking mode only. User clocking allows the user to implement clocking logic outside the core. An example design is provided to demonstrate the implementation of a clocking module for the core.

Global Clocking with DCM for Virtex-4 FPGAs

To ensure that the Virtex-4 DCMs can achieve maximum frequency specifications under all conditions, the following DCM timing parameters (specified in the *Virtex-4 FPGA Data Sheet*) must be met:

- TCONFIG
- DCM_INPUT_CLOCK_STOP
- DCM_RESET

If the `DCM_INPUT_CLOCK_STOP` and `DCM_RESET` parameters cannot be guaranteed by design, the global clocking with DCM Standby Logic option must be selected. This option is provided to support long clock stop (>100 ms) or reset (>10 sec) times. The DCM Standby Logic macro detects stopped input clocks in excess of 100 ms, and toggles the DCM until the input clocks return, at which time the DCM will re-lock. If the reset input is asserted, the assertion will be translated into a short pulse at the DCM RST input, and the DCM will relock. The DCM Standby Logic will add CLB resources to the core. See Table 24 and Table 25 for more information.

The TCONFIG parameter is the maximum time to configure devices after VCCINT is applied. If this timing parameter cannot be guaranteed by design, see Answer Record 21127 for detailed information about addressing this requirement.

Multiple Core Instantiations

It is possible to implement multiple SPI-4.2 cores in a single target device. Larger Virtex-6, Virtex-5, and Virtex-4 devices can support more than four SPI-4.2 cores. See UG153, *SPI-4.2 User Guide*, for more details.

Verification

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

Using the in-house verification environment, the SPI-4.2 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-4 FPGAs -10 devices up to 800 Mbps on the SPI-4.2 Interface and 225 MHz on the User Interface (`SrcFFClk` and `SnkFFClk` clocks).
  - Virtex-4 FPGAs -11 devices up to 900 Mbps on the SPI-4.2 Interface and 250 MHz on the User Interface (`SrcFFClk` and `SnkFFClk` clocks).
  - Virtex-5 FPGAs -1 devices up to 1 Gbps on the SPI-4.2 Interface and 250 MHz on the User Interface (`SrcFFClk` and `SnkFFClk` clocks).
  - Virtex-5 FPGAs -2 devices up to 1+ Gbps on the SPI-4.2 Interface and 275 MHz on the User Interface (`SrcFFClk` and `SnkFFClk` clocks).
  - Virtex-6 FPGAs -1, -2, and -3 devices up to 1+ Gbps on the SPI-4.2 Interface and 250+ MHz on the User Interface (`SrcFFClk` and `SnkFFClk` clocks).

In addition to extensive simulation, Xilinx has verified core operation in silicon on the Virtex-6 FPGA ML623 test platform, Virtex-5 FPGA ML550, and Virtex-4 FPGA ML450 networking interface platforms. The core was verified at +500 MHz DDR (equivalent to approximately 16 Gbps aggregate data rate on the SPI-4.2 Interface and 250 MHz on the User Interface) for the Virtex architecture.

The ML450 hardware platform that is used for internal testing connects the Source core to the Sink core on a single Virtex-4 device, with 16 inches of cabling for static alignment and skewed cabling for dynamic alignment. The skewed cable introduces seven inches of skew between every adjacent bit. Pseudo-random data is generated internally by an LFSR (x^16 + x^12 + x^3 + x^1 + x^0) and transmitted by the Source core. The data received by the Sink core is then compared against the transmitted data. The SPI-4.2 solution has been tested on this platform with various packet and burst sizes. Additionally, the continuous DPA solution has also been tested on this platform by dynamically skewing the incoming `RDClk`. This is done by inserting an `IDELAY` in the `RDClk` path and changing the tap value counter up and down to emulate a shifting clock. The ML623 and the ML550 hardware boards are also configured similarly to the ML450 when testing the SPI-4.2 core.

Note: Local field offices have access to the ML550 and ML450 hardware platform, and an on-site demonstration of SPI-4.2 operating on this platform can be arranged. Contact your local Xilinx sales representative or field applications engineer for more information.

Device Utilization

Table 20 and Table 21 contain the Block RAM, LUTs, and FFs counts for the Sink and Source cores that target Virtex-6 devices.

Table 22 and Table 23 contain the Block RAM, LUTs, and FFs counts for the Sink and Source cores that target Virtex-5 devices.

Table 24 and Table 25 contain the Block RAM, LUTs, and FFs counts for the Sink and Source cores that target Virtex-4 devices.

The use of the DCM Standby Logic macro increases the slice count by 50: 36 in FFs and 75 in LUTs. The optional Continuous DPA Logic increases the slice count by 60 slices. To ensure minimal slice count, set the compression factor for area groups to 1 in the constraints file.

Table 20: **Sink Core Utilization – Virtex-6 Devices**

Core Configuration	Block RAM	LUTs	FFs
64-bit Static Alignment	1 (36k block RAM) 5(18k block RAM)	1537	1709
64-bit Dynamic Alignment (default)	1 (36k block RAM) 5(18k block RAM)	2301	2156
128-bit Static Alignment	1 (36k block RAM) 9(18k block RAM)	2323	2032
128-bit Dynamic Alignment (default)	1 (36k block RAM) 9(18k block RAM)	3291	2520

Table 21: **Source Core Utilization – Virtex-6 Devices**

Core Configuration	Block RAM	LUTs	FFs
64-bit	6 (18k block RAM)	1905	2261
128-bit	6 (18k block RAM)	1745	2227

Table 22: **Sink Core Utilization – Virtex-5 Devices**

Core Configuration	Block RAM	LUTs	FFs
64-bit Static Alignment	5 (18k block RAM) 1 (36k block RAM)	1500	1750
64-bit Dynamic Alignment (default)	5 (18k block RAM) 1 (36k block RAM)	1900	2100
128-bit Static Alignment	9 (18k block RAM) 1 (36k block RAM)	2300	2100
128-bit Dynamic Alignment (default)	9 (18k block RAM) 1 (36k block RAM)	2700	2400

Table 23: **Source Core Utilization – Virtex-5 Devices**

Core Configuration	Block RAM	LUTs	FFs
64-bit	6 (18k block RAM)	1900	2250
128-bit	6 (18k block RAM)	1750	2200

Table 24: **Sink Core Utilization – Virtex-4 Devices**

Core Configuration	Block RAM	LUTs	FFs
64-bit Static Alignment	6	1900	1700
64-bit Dynamic Alignment (default)	6	2400	2050
128-bit Static Alignment	11	3100	2100
128-bit Dynamic Alignment (default)	11	3550	2400
64-bit Static Alignment with DCM Standby Logic	6	2000	1750
64-bit Dynamic Alignment with DCM Standby Logic (default)	6	2500	2100
128-bit Static Alignment with DCM Standby Logic	11	3200	2100
128-bit Dynamic Alignment with DCM Standby Logic (default)	11	3600	2450

Table 25: **Source Core Utilization – Virtex-4 Devices**

Core Configuration	Block RAM	LUTs	FFs
64-bit	6	2450	2250
128-bit	6	2300	2200
64-bit with DCM Standby Logic	6	2600	2300
128-bit with DCM Standby Logic	6	2450	2300

Performance in Virtex-6 FPGAs

Table 26 and Table 27 contain the performance numbers for the SPI-4.2 cores that target Virtex-6 devices.

Table 26: **Performance in Virtex-6 Devices (Not Including CXT)**

Alignment Type	Speed Grade	Performance	
		Sink Core with Global Clocking	Sink Core with Regional Clocking
Static(1)	-1L , -1, -2, -3	622-700Mpbs	622-700Mpbs
Dynamic (Source core with regional clocking)	-1L	622-900Mpbs	622-900Mpbs
Dynamic (Source core with regional clocking)	-1	622-900Mpbs	622-1.1Gpbs
Dynamic (Source core with regional clocking)	-2	622-1.1Gpbs	622-1.25Gbps
Dynamic (Source core with regional clocking)	-3	622-1.4Gbps	622-1.4Gbps
Dynamic (Source core with global clocking)	-1L, -1, -2	622Mbps-900Mbps	622Mbps-900Mbps
Dynamic (Source core with global clocking)	-3	622Mpbs-1Gbps	622Mpbs-1Gbps

1. When the Sink core is configured with static alignment, only regional clocking is supported on the source core.

Table 27: **Performance in Virtex-6 CXT Devices**

Alignment Type	Speed Grade	Performance
Static(1)	-1,-2,-3	622 Mbps-700 Mbps
Dynamic	-1	622 Mbps-700 Mbps
Dynamic	-2	622 Mbps-800 Mbps

1. When Sink core is configured with static alignment, only regional clocking is supported on the source core.

Performance in Virtex-5 FPGAs

Table 28 and Table 29 contain the performance numbers for the SPI-4.2 cores with Regional Clocking and Global clocking that target Virtex-5 devices.

Table 28: **Performance with Regional Clocking in Virtex-5 Devices**

Alignment Type	Speed Grade	Performance
Static	-1, -2, -3	622 Mbps–700 Mbps
Dynamic	-1, -2	622 Mbps–1 Gbps
Dynamic	-3	622 Mbps–1.2 Gbps

Table 29: **Performance with Global Clocking in Virtex-5 Devices**

Alignment Type	Speed Grade	Performance
Static	-1, -2, -3	622 Mbps–700 Mbps
Dynamic	-1	622 Mbps–900 Mbps
Dynamic	-2	622 Mbps–1 Gbps
Dynamic	-3	622 Mbps–1.1 Gbps

Performance in Virtex-4 FPGAs

Table 30 contains the performance numbers for the SPI-4.2 cores that target Virtex-4 devices.

Table 30: **Performance in Virtex-4 Devices**

Alignment Type	Speed Grade	Performance (Mbps)
Static	-10, -11, -12	622 Mbps–700 Mbps
Dynamic	-10	622 Mbps–800 Mbps
Dynamic	-11	622 Mbps–900 Mbps
Dynamic	-12	622 Mbps–1 Gbps

Power in Virtex-6 FPGAs

Table 31 contains the power numbers for the SPI-4.2 cores that target Virtex-6 devices.

Table 31: **Power in Virtex-6 Devices** (1)

Alignment Type	Speed Grade	Clocking Option	Data Rate	Dynamic Power
Dynamic	-1	Global clocking	800 Mbps	1.74 W
Dynamic	-1	Global clocking	1 Gbps	1.97 W
Static	-1	Global clocking	700 Mbps	1.34 W

1. These numbers represent the power of the entire design consisting of Sink core, Source core, and the demo loopback design. This table was last updated with hardware measurements made with a design implemented with the ISE v13.2.

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 core is provided under the SignOnce IP Site License and can be generated using the Xilinx CORE Generator system. 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 LogiCORE modules 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
09/30/04	1.0	Initial Xilinx release.
11/11/04	1.1	Updated document with changes to performance numbers of the dynamic phase alignment configuration of the core.
04/28/05	1.2	Updated document to indicate support for Xilinx ISE Foundation v7.1i.
08/31/05	2.0	Added SP 3 to Xilinx ISE v7.1i, section about Global Clocking with DCM edited, various copy edits.
01/18/06	3.0	Updated ISE to v8.1i, revised dynamic alignment section.
07/13/06	4.0	Added support for Virtex-5, Updated ISE to v8.2i, release date.
09/21/06	4.1	Added new DPA-related signals. Updated for IP2i minor release.
02/15/07	4.2	Updated for IP1Jade release.
8/08/07	4.3	Updated for the IP1 Jade Minor release. Added SnkIdelayCtlRdy Input signal.
03/24/08	4.4	Modified description of ScerAFThresAssert[8:0] signal in Table 12, updated supported tools and core version.
09/19/08	4.5	Updated for the ISE service pack 3 release.
04/24/09	5.0	Updated core version to 9.1 and ISE to version 11.1. Added support for Virtex-6 devices.
06/24/09	5.5	Updated core version to 9.2 and ISE to version 11.2.
09/16/09	6.0	Updated core version to 9.3 and ISE to version 11.3. Added support for Virtex-6 -1L and Virtex-6 CXT devices.
04/22/10	7.0	Updated core version to 10.1 and ISE version to 12.1; added support for Spartan-6 devices.
09/21/10	8.0	Updated core version to 10.2 and ISE version to 12.3.
12/14/10	9.0	Updated core version to 10.3 and ISE version to 12.4; removed Spartan-6 support and updated Virtex-6 performance numbers and supported clocking scheme.
3/1/11	10.0	Updated core version to 10.4 and ISE version to 13.1.
6/22/11	11.0	Updated core version to 10.5 and ISE Design Suite version to 13.2.

Notice of Disclaimer