LogiCORE IP SPI-4.2 v12.2

Introduction

The Xilinx SPI-4.2 (PL4) core implements and is 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 700 MHz DDR on SPI-4.2 interface supporting 1.4 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 MMCM, global, and regional clocking resources
- SelectIO technology supports flexible pin assignment
- Configurable 64-bit or 128-bit AXI4-Stream FIFO interface, both supporting full bandwidth capabilities
- Supports unsegmented burst sizes up to 16K
- Optional continuous DPA window monitoring
- Optional advanced DPA diagnostics
- 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 Kintex™-7, and Virtex®-7 device and package configurations
- Supports AXI4-Lite memory mapped interface to manage static configuration, status, and calendar
- 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 LVDS or LVTTL (7 series only) 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[1]	Kintex-7, Virtex-7		
Supported User Interface	AXI4-Lite and AXI4-Stream		

Resources Used [2]				
Alignment Type		Performance (Mbps)/Speed	LUTs	Block RAM
FPGAs	64-bit static	622-700/ -1, -2, -3	4050	3 (36k BRAM) 9 (18k BRAM)
	128-bit static	622-700/ -1, -2, -3	4670	3 (36k BRAM) 13 (18k BRAM)
	64-bit dynamic	622-1.1 Gbps/-1 622-1.2 Gbps/-2 622-1.25Gbps/-3	5020	3 (36k BRAM) 9 (18k BRAM)
	128-bit dynamic	622-1.1 Gbps/-1 622-1.2 Gbps/-2 622-1.25 Gbps/-3	5640	3 (36k BRAM) 13 (18k BRAM)

Provided with Core	
Documentation	User Guide Release Notes
Design Files	Encrypted RTL
Example Design	VHDL and Verilog
Test Bench	VHDL and Verilog
Constraints	Example XDC
Simulation Model	VHDL and Verilog Structural Model
Supported S/W Driver	N/A

Design Tool Requirements[3]	
Design Entry	Vivado™ Design Suite 2012.2
Simulation	Mentor Graphics ModelSim Synopsys VCS and VCS MX
Synthesis	Vivado Synthesis

Support	
Provided by Xilinx, Inc.	

1. For a complete listing of supported devices, see the release notes for this core.
2. Numbers are for default configurations in Virtex-7 devices. See Table 16 through Table 18 for more information.
3. For the supported versions of the tools, see the Xilinx Design Tools: 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.
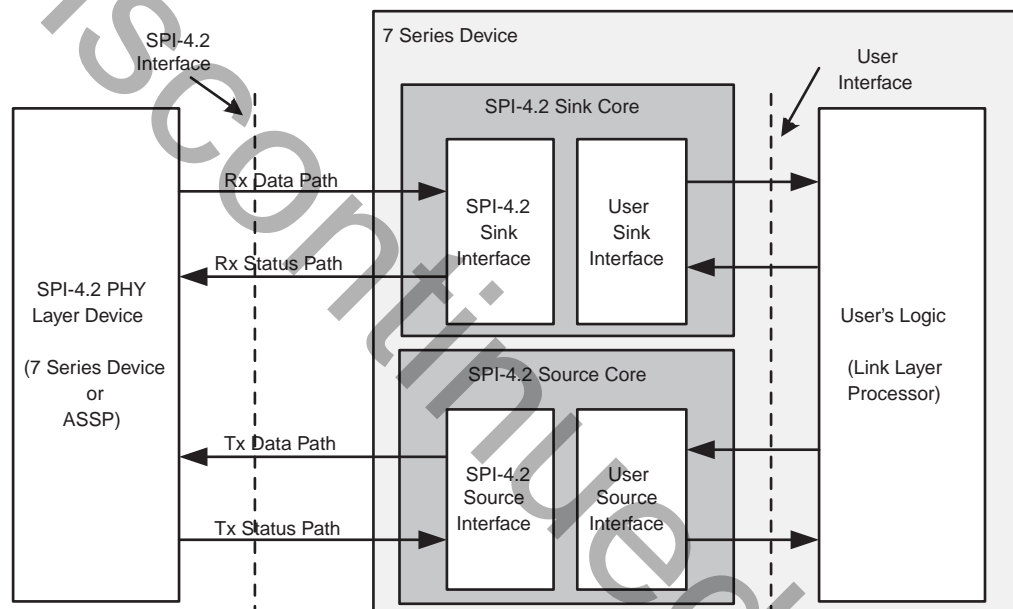

Figure 1: **SPI-4.2 Core in a Typical Link-Layer Application**

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 AXI4-Stream FIFO interface. The core also processes 2-bit status information (for each channel) from the AXI4-Lite Control 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 four primary interfaces: the SPI-4.2 interface, the AXI4-Stream FIFO interface, the AXI4-Lite Control interface, and the Control and Status 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 6.
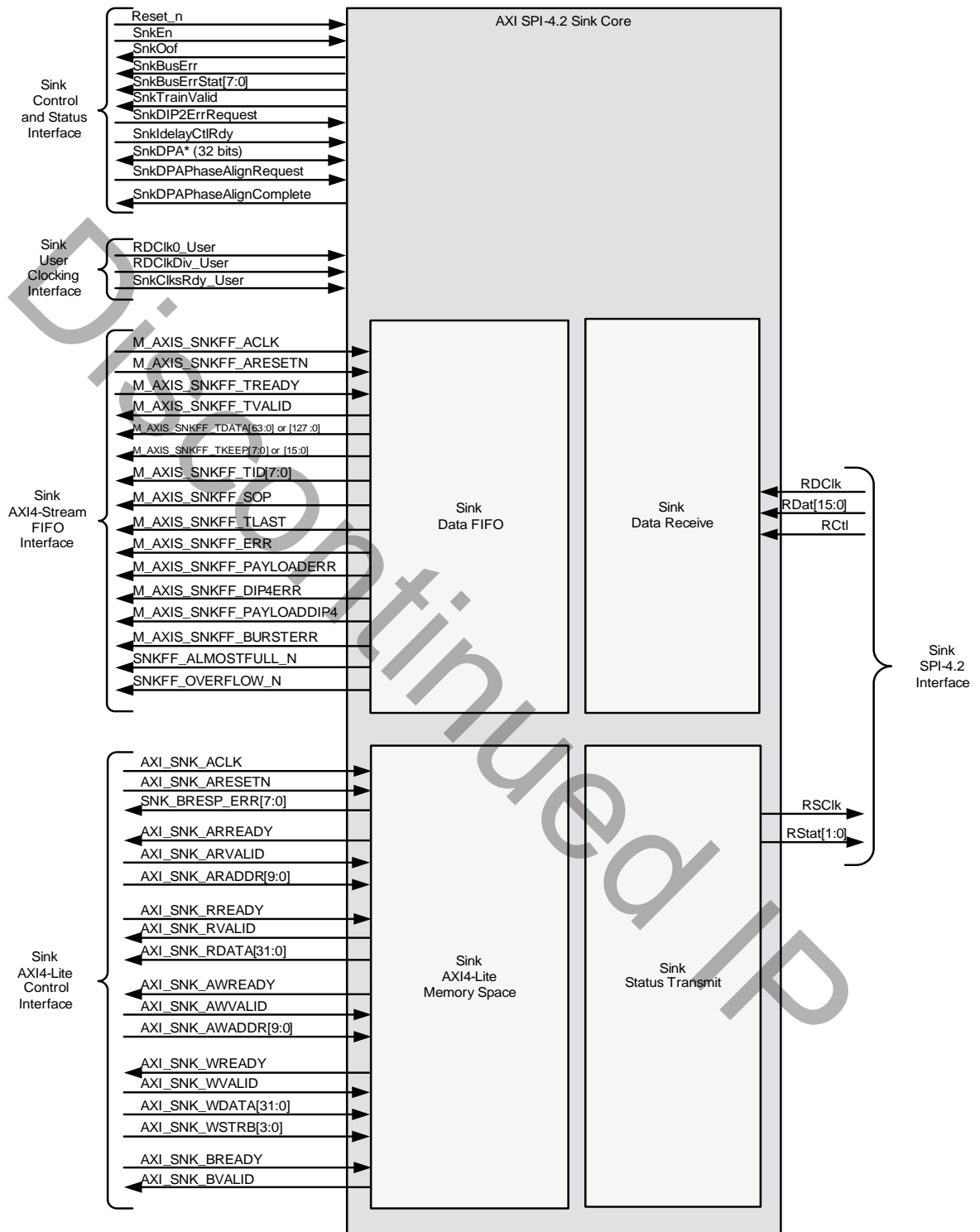

Figure 2: **Sink Core Block Diagram and I/O Interface Signals**

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 AXI4-Stream FIFO interface. The core also processes 2-bit status (for each channel) received on the SPI-4.2 interface and provides them on the AXI4-Lite Control interface (or the AXI4-Stream Status interface for Transparent mode). 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 five primary interfaces: the SPI-4.2 interface, the AXI4-Stream FIFO interface, the AXI4-Stream Status interface, the AXI4-Lite Control interface, and the Control and Status 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 6.
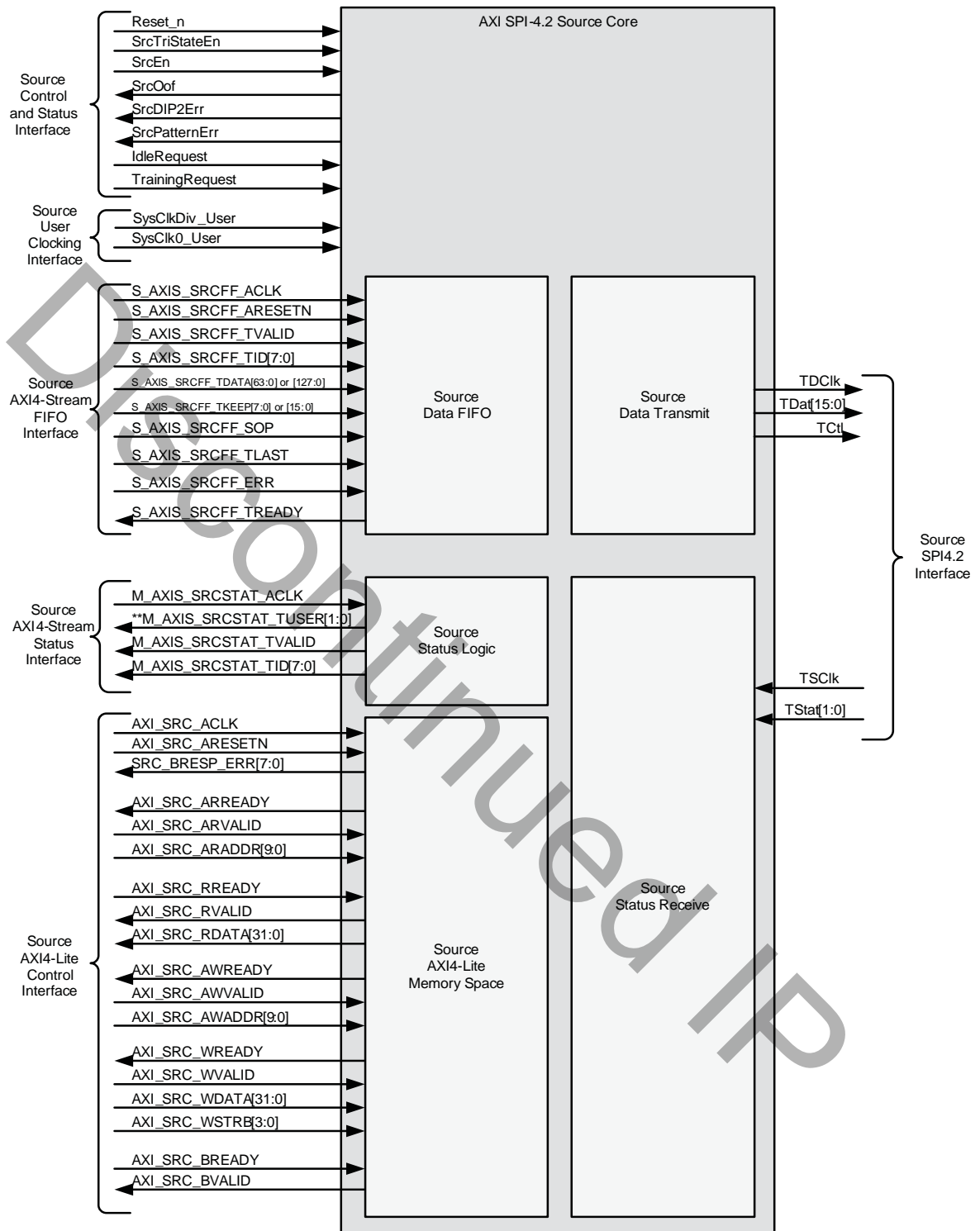
Figure 3: **Source Core Block Diagram and I/O Interface Signals**

Core Interfaces

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

Sink Interfaces

The Sink core has four primary interfaces: the SPI-4.2 interface, the AXI4-Stream FIFO interface, the AXI4-Lite Control interface, and the Control and Status 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_P RSClk_N	Output	n/a	**SPI-4.2 Receive Status Clock** (LVDS/LVTTL). 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 parameter RSClkDiv.
RStat_P[1:0] RStat_N[1:0]	Output	RSClk	**SPI-4.2 Receive FIFO Status** (LVDS/LVTTL). FIFO Status Channel flow control interface.

Sink User Interface

The Sink user interface can be divided into the following subgroups, based on function.

- Control and status interface
- AXI4-Stream FIFO interface
- AXI4-Lite Control 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. The interface also provides information regarding the status and debug of Dynamic Phase Alignment (DPA), if the DPA option is selected. These DPA status and debug signals are described in Dynamic Alignment.

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.
SnkEn	Input	RDClkDiv_User	**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.
SnkIdelayCtlRdy	Input	RDClkDiv_User	**IDELAYCTRL Ready.** Active high signal that indicates when the IDELAY modules are calibrated. 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	RDClkDiv_User	**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.
SnkBusErr	Output	RDClkDiv_User	**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 Interface Signals** *(Cont'd)*

Name	Direction	Clock Domain	Description
SnkBusErrStat[7:0]	Output	RDClkDiv_User	**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	RDClkDiv_User	**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_User clock cycles), if the training pattern received is successfully decoded.
SnkDIP2ErrRequest	Input	RDClkDiv_User	**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.

Sink AXI4-Stream FIFO Interface

The Sink AXI4-Stream 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 AXI4-Stream FIFO Interface Signals**

Name	Direction	Description
M_AXIS_SNKFF_ACLK	Input	**Sink FIFO Clock**. All Sink FIFO Interface signals are synchronous to the rising edge of this clock.
M_AXIS_SNKFF_ARESETN	Input	**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.The reset signal can be asserted asynchronously, but deassertion must be synchronous after the rising edge of ACLK. See the *AMBA® AXI4-Stream Protocol* specification for more information.
M_AXIS_SNKFF_TREADY	Input	**Sink FIFO User Ready Handshake**. Indicates the user is ready to accept streaming data. When both TREADY and TVALID are asserted on the rising edge of M_AXIS_SNKFF_ACLK, one data beat is transferred.

Table 3: **Sink AXI4-Stream FIFO Interface Signals** *(Cont'd)*

Name	Direction	Description
M_AXIS_SNKFF_TVALID	Output	**Sink FIFO Data Valid Handshake.** Indicates the core is presenting valid streaming data. When both TREADY and TVALID are asserted on the rising edge of M_AXIS_SNKFF_ACLK, one data beat is transferred. When asserted (active high), this signal indicates that the information on M_AXIS_SNKFF_TDATA, M_AXIS_SNKFF_TID, M_AXIS_SNKFF_SOP, M_AXIS_SNKFF_TLAST, M_AXIS_SNKFF_BURSTERR, M_AXIS_SNKFF_TKEEP, M_AXIS_SNKFF_ERR, M_AXIS_SNKFF_DIP4ERR, M_AXIS_SNKFF_PAYLOADERR, and M_AXIS_SNKFF_PAYLOADDIP4 are valid.
M_AXIS_SNKFF_TID[7:0]	Output	**Sink FIFO Channel Address**. Channel number associated with the data on M_AXIS_SNKFF_TDATA.
M_AXIS_SNKFF_TDATA[63:0] or M_AXIS_SNKFF_TDATA[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.
M_AXIS_SNKFF_TKEEP[7:0] or M_AXIS_SNKFF_TKEEP[15:0]	Output	**Sink FIFO Data Strobe**. This signal indicates which bytes on the M_AXIS_SNKFF_TDATA bus are valid when the M_AXIS_SNKFF_TLAST signal is asserted (i.e. byte enable). M_AXIS_SNKFF_TKEEP[7:0] is used with a 64-bit interface. M_AXIS_SNKFF_TKEEP[15:0] is used with a 128-bit interface.
M_AXIS_SNKFF_SOP	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.
M_AXIS_SNKFF_TLAST	Output	**Sink FIFO End of Packet (EOP)**. When asserted (active high), this signal indicates that the end of a packet is being read out of the Sink FIFO.
M_AXIS_SNKFF_ERR	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 M_AXIS_SNKFF_TLAST is asserted.
M_AXIS_SNKFF_DIP4ERR	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.
M_AXIS_SNKFF_PAYLOADDIP4	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.
M_AXIS_SNKFF_BURSTERR	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. M_AXIS_SNKFF_BURSTERR 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 M_AXIS_SNKFF_TLAST or M_AXIS_SNKFF_ERR.
M_AXIS_SNKFF_PAYLOADERR	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.

Table 3: **Sink AXI4-Stream FIFO Interface Signals** *(Cont'd)*

Name	Direction	Description
SNKFF_ALMOSTFULL_N	Output	**Sink FIFO 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.
SNKFF_OVERFLOW_N	Output	**Sink FIFO 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 SNKFF_OVERFLOW_N is asserted, since no data is written into the FIFO when the overflow signal is asserted.

Sink AXI4-Lite Control Interface

The Sink AXI4-Lite Control interface is used to program the calendar memory, status memory, and static configuration memory. This memory space is defined in Figure 4

The calendar determines the 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. The Status memory enables the user to send flow control data to the transmitting device. Flow control may be automatically or manually implemented. The static configuration memory enables customization of the core based on individual system requirements. These settings are statically driven inside the core by writing to registers through the Control interface. Table 4 describes the Control interface signals, and Table 5 defines the static configuration parameters.

Table 4: **Sink AXI4-Lite Control Interface Signals**

Name	Direction	Clock Domain	Description
AXI_SNK_ACLK	Input	N/A	**AXI4-Lite Clock**. All Sink AXI4-Lite signals are synchronous to the rising edge of this clock
AXI_SNK_ARESETN	Input	AXI_SNK_ACLK	**AXI4-Lite Reset.** Active-low signal that enables the user to reset the AXI4-Lite interface and all associated logic and memories to chosen CORE Generator settings. The reset signal can be asserted asynchronously, but deassertion must be synchronous after the rising edge of ACLK. See the *AMBA AXI4 Protocol* specification for more information.
Read Address Channel			
AXI_SNK_ARREADY	Output	AXI_SNK_ACLK	**AXI4-Lite Read Address Core Handshake**. Indicates the core is ready to accept a read address. When both ARREADY and ARVALID are asserted on a clock cycle, a one-word read request occurs for the address provided on ARADDR and the core fetches the contents of that address.
AXI_SNK_ARVALID	Input	AXI_SNK_ACLK	**AXI4-Lite Read Address User Handshake**. Indicates the user is presenting a valid read address. When both ARREADY and ARVALID are asserted on a clock cycle, a one-word read request occurs for the address provided on ARADDR and the core fetches the contents of that address
AXI_SNK_ARADDR[9:0]	Input	AXI_SNK_ACLK	**AXI4-Lite Read Address.** Address to be read by the user application.

Table 4: **Sink AXI4-Lite Control Interface Signals** *(Cont'd)*

Name	Direction	Clock Domain	Description
Read Data Channel			
AXI_SNK_RREADY	Input	AXI_SNK_ACLK	**AXI4-Lite Read Data User Handshake**. Indicates the user is ready to accept read data. When both RREADY and RVALID are asserted on a clock cycle, one data beat is read out of the core. This handshake completes the transaction begun by the ARREADY/ARVALID handshake.
AXI_SNK_RVALID	Output	AXI_SNK_ACLK	**AXI4-Lite Read Data Core Handshake**. Indicates the core is presenting valid read data. When both RREADY and RVALID are asserted on a clock cycle, one data beat is read out of the core. This handshake completes the transaction begun by the ARREADY/ARVALID handshake.
AXI_SNK_RDATA[31:0]	Output	AXI_SNK_ACLK	**AXI4-Lite Read Data.** Data read from address location provided on ARADDR is presented here. Bit 0 is the LSB.
Write Address Channel			
AXI_SNK_AWREADY	Output	AXI_SNK_ACLK	**AXI4-Lite Write Address Core Handshake.** Indicates the core is ready to accept a write address. When both AWREADY and AWVALID are asserted on a clock cycle, one write address beat is accepted by the core. Once both the WREADY/WVALID and AWREADY/AWVALID handshakes occur independently, one write transaction is complete.
AXI_SNK_AWVALID	Input	AXI_SNK_ACLK	**AXI4-Lite Write Address User Handshake.** Indicates the user is presenting a valid write address. When both AWREADY and AWVALID are asserted on a clock cycle, one write address beat is accepted by the core. Once both the WREADY/WVALID and AWREADY/AWVALID handshakes occur independently, one write transaction is complete
AXI_SNK_AWADDR[9:0]	Input	AXI_SNK_ACLK	**AXI4-Lite Write Address**. Address to be written by the user application.
Write Data Channel			
AXI_SNK_WREADY	Output	AXI_SNK_ACLK	**AXI4-Lite Write Data Core Handshake**. Indicates the core is ready to accept write data and strobe. When both WREADY and WVALID are asserted on a clock cycle, one write address beat is accepted by the core. Once both the WREADY/WVALID and AWREADY/AWVALID handshakes occur independently, one write transaction is complete.
AXI_SNK_WVALID	Input	AXI_SNK_ACLK	**AXI4-Lite Write Data User Handshake.** Indicates the user is presenting valid write data and strobe. When both WREADY and WVALID are asserted on a clock cycle, one write address beat is accepted by the core. Once both the WREADY/WVALID and AWREADY/AWVALID handshakes occur independently, one write transaction is complete.
AXI_SNK_WDATA[31:0]	Input	AXI_SNK_ACLK	**AXI4-Lite Write Data**. Data written into the core. Bit 0 is the LSB.
AXI_SNK_WSTRB[3:0]	Input	AXI_SNK_ACLK	**AXI4-Lite Write Strobe**. Byte enable associated with AXI_SNK_WDATA. Bit 0 enables WDATA[7:0], bit 1 enables WDATA[15:8], bit 2 enables WDATA[23:16], and bit 3 enables WDATA[32:24].

Table 4: **Sink AXI4-Lite Control Interface Signals** *(Cont'd)*

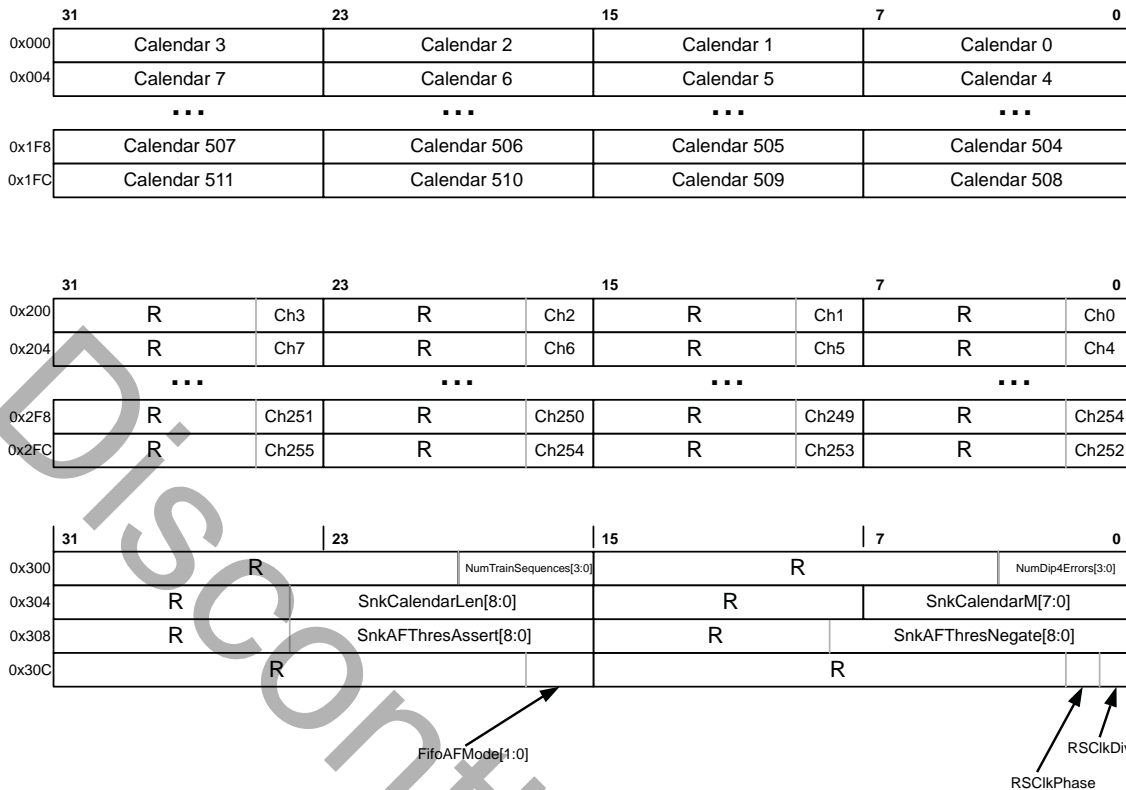| Name	Direction	Clock Domain	Description
Write Response Channel			
AXI_SNK_BREADY	Input	AXI_SNK_ACLK	**AXI4-Lite Write Response User Handshake.** Indicates the user is ready to accept a write response. When both BREADY and BVALID are asserted on a rising edge of AXI_SNK_ACLK, one write response transaction is read out of the core.
AXI_SNK_BVALID	Output	AXI_SNK_ACLK	**AXI4-Lite Write Response Core Handshake.** Indicates the core is presenting a valid write response. When both BREADY and BVALID are asserted on a rising edge of AXI_SNK_ACLK, one write response transaction is read out of the core. Each write transaction will generate a write response, regardless of whether the write succeeded. The Error Bus (below) provides further signaling in regards to errors that occur during write operations.
Sideband Error Bus			
SNK_BRESP_ERR[7:0]	Output	AXI_SNK_ACLK	**AXI4-Lite Error Bus.** Each bit of this bus corresponds to a specific AXI4-Lite Error condition. The error conditions detected are reported as follows: SNK_BRESP_ERR [0]: Write Response counter is full. No further writes or reads will be processed until at least one Write Response is read from the core using BREADY/BVALID. SNK_BRESP_ERR[1]: Write transaction to the Configuration space failed. SNK_BRESP_ERR [2]: Write transaction to the Calendar space has failed. SNK_BRESP_ERR[3]:Read/Write has missed a valid address SNK_BRESP_ERR [4:7]: Reserved bits (tied low).

	31		23		15		7		0
0x000	Calendar 3		Calendar 2		Calendar 1		Calendar 0		
0x004	Calendar 7		Calendar 6		Calendar 5		Calendar 4		
	. . .		. . .		. . .		. . .		
0x1F8	Calendar 507		Calendar 506		Calendar 505		Calendar 504		
0x1FC	Calendar 511		Calendar 510		Calendar 509		Calendar 508		

	31		23		15		7		0
0x200	R	Ch3	R	Ch2	R	Ch1	R	Ch0	
0x204	R	Ch7	R	Ch6	R	Ch5	R	Ch4	
	. . .		. . .		. . .		. . .		
0x2F8	R	Ch251	R	Ch250	R	Ch249	R	Ch254	
0x2FC	R	Ch255	R	Ch254	R	Ch253	R	Ch252	

	31		23		15		7		0
0x300	R			NumTrainSequences[3:0]	R			NumDip4Errors[3:0]	
0x304	R		SnkCalendarLen[8:0]		R		SnkCalendarM[7:0]		
0x308	R		SnkAFThresAssert[8:0]		R		SnkAFThresNegate[8:0]		
0x30C	R				R				

FifoAFMode[1:0]

RSClkDiv

RSClkPhase

Figure 4: **Sink AXI4-Lite Memory Space**

Table 5: **Sink Static Configuration Parameter Definition**

Name	Range	Description
NumDip4Errors[3:0]	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]	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]	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 programmed by the AXI4-Lite interface, 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.

Table 5: **Sink Static Configuration Parameter Definition** *(Cont'd)*

Name	Range	Description
SnkCalendar_Len[8:0]	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 programmed by the AXI4-Lite interface, 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.
SnkAFThresNegate[8:0]	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 SNKFF_ALMOSTFULL_N is deasserted. Note that the negate threshold must be greater or equal to the assert threshold (SnkAFThresAssert). When SNKFF_ALMOSTFULL_N is deasserted, the core stops sending flow control (deasserts SNKFF_ALMOSTFULL_N) and resumes transmission of valid FIFO status levels. This indicates to the transmitting device that additional data can be sent.
SnkAFThresAssert[8:0]	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 SNKFF_ALMOSTFULL_N is asserted. The assert threshold must be less than or equal to the negate threshold (SnkAFThresNegate). When SNKFF_ALMOSTFULL_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.
RSClkDiv	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

Table 5: **Sink Static Configuration Parameter Definition** *(Cont'd)*

Name	Range	Description
RSClkPhase	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]	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 SNKFF_ALMOSTFULL_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 five primary interfaces: the SPI-4.2 interface, the AXI4-Stream FIFO interface, the AXI4-Stream Status interface, the AXI4-Lite Control interface, and the Control and Status interface.

Source SPI-4.2 Interface

Table 6 defines the signals on the Source SPI-4.2 interface.

Table 6: **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_P TSClk_N	Input	n/a	**SPI-4.2 Transmit Status Clock** (LVDS/LVTTL). Source synchronous clock that is received by the Source core with TStat at 1/4 rate (or 1/8 rate) of TDClk.
TStat_P[1:0] TStat_N[1:0]	Input	TSClk	**SPI-4.2 Transmit FIFO Status** (LVDS/LVTTL). FIFO-Status-Channel flow control interface.

Source User Interface

The Source user interface can be divided into the following subgroups, based on function:

- Control and status interface
- AXI4-Stream FIFO interface
- AXI4-Lite Control interface
- AXI4-Stream Status 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 7 defines Source control and status signals.

Table 7: **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.
SrcEn	Input	SysClkDiv_User	**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	SysClkDiv_User	**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 parameter NumDip2Matches) are received and SrcEn is asserted.
SrcDIP2Err	Output	M_AXIS_SRCSTAT_ACLK	**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	M_AXIS_SRCSTAT_ACLK	**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 a frame word error is detected.
SrcPatternErr	Output	S_AXIS_SRCFF_ACLK	**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. • All S_AXIS_SRCFF_TKEEP not set to one when S_AXIS_SRCFF_TLAST is deasserted. This signal is asserted for one clock cycle each time an illegal data pattern is written into the Source FIFO.

Table 7: **Source Control and Status Signals** *(Cont'd)*

Name	Direction	Clock Domain	Description
IdleRequest	Input	SysClkDiv_User	**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	SysClkDiv_User	**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	SysClkDiv_User	**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	SysClkDiv_User	**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.

Product Specification

Source AXI4-Stream FIFO Interface

The Source AXI4-Stream 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 associated with a particular channel or packet. Table 8 defines the Source FIFO signals.

Table 8: **Sink AXI4-Stream FIFO Interface Signals**

Name	Direction	Clock Domain	Description
S_AXIS_SRCFF_ACLK	Input	n/a	**Source FIFO Clock**. All Source FIFO Interface signals are synchronous to the rising edge of this clock.
S_AXIS_SRCFF_ARESETN	Input	S_AXIS_SRCFF_ACLK	**Source FIFO Reset.** Active low signal that 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. The reset signal can be asserted asynchronously, but deassertion must be synchronous after the rising edge of ACLK. See the *AMBA AXI4-Stream Protocol* specification for more information.
S_AXIS_SRCFF_TREADY	Output	S_AXIS_SRCFF_ACLK	**Source FIFO Core Handshake**. Indicates the core is ready to accept streaming data. When both TREADY and TVALID are asserted on the rising edge of S_AXIS_SRCFF_ACLK, one data beat is transferred into the core. De-assertion of TREADY indicates the Source core has reached the AlmostFull threshold set by the parameter SrcAFThresAssert.
S_AXIS_SRCFF_TVALID	Input	S_AXIS_SRCFF_ACLK	**Source FIFO User Handshake**. Indicates the user is presenting valid streaming data. When both TREADY and TVALID are asserted on the rising edge of S_AXIS_SRCFF_ACLK, one data beat is transferred into the core. When asserted (active high), this signal indicates that the information on S_AXIS_SRCFF_TDATA, S_AXIS_SRCFF_TID, S_AXIS_SRCFF_SOP, S_AXIS_SRCFF_TLAST, S_AXIS_SRCFF_TKEEP, and S_AXIS_SRCFF_ERR are valid.
S_AXIS_SRCFF_TID[7:0]	Input	S_AXIS_SRCFF_ACLK	**Source FIFO Channel Address**. Channel number associated with the data on S_AXIS_SRCFF_TDATA.
S_AXIS_SRCFF_TDATA[63:0] or S_AXIS_SRCFF_TDATA[127:0]	Input	S_AXIS_SRCFF_ACLK	**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.

Table 8: **Sink AXI4-Stream FIFO Interface Signals** *(Cont'd)*

Name	Direction	Clock Domain	Description
S_AXIS_SRCFF_TKEEP[15:0] or S_AXIS_SRCFF_TKEEP[7:0]	Input	S_AXIS_SRCFF_ACLK	**Source FIFO Strobe.** This signal indicates which bytes on the S_AXIS_SRCFF_TDATA bus are valid when the S_AXIS_SRCFF_TLAST signal is asserted (i.e. byte enable). When S_AXIS_SRCFF_TLAST is deasserted, S_AXIS_SRCFF_TKEEP should always be set to all ones. S_AXIS_SRCFF_TKEEP[7:0] is used with a 64-bit interface. S_AXIS_SRCFF_TKEEP[15:0] is used with a 128-bit interface.
S_AXIS_SRCFF_SOP	Input	S_AXIS_SRCFF_ACLK	**Source FIFO Start of Packet**. When asserted (active high), indicates that the start of a packet is being written into the Source FIFO.
S_AXIS_SRCFF_TLAST	Input	S_AXIS_SRCFF_ACLK	**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 S_AXIS_SRCFF_SOP.
S_AXIS_SRCFF_ERR	Input	S_AXIS_SRCFF_ACLK	**Source FIFO Error**. When asserted (active high) simultaneously with the S_AXIS_SRCFF_TLAST 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. S_AXIS_SRCFF_ERR can be used in combination with S_AXIS_SRCFF_TLAST to insert erroneous DIP-4 values for testing purposes. When S_AXIS_SRCFF_ERR is asserted and S_AXIS_SRCFF_TLAST is not asserted, the core inserts an EOP (1 or 2 bytes depending on the S_AXIS_SRCFF_TKEEP value) with an erroneous DIP-4 value. The erroneous DIP4 value is an inversion of the correctly calculated value.

Source AXI4-Lite Control Interface

The Source AXI4-Lite Control interface is used to program the calendar memory, read status memory, and program static configuration memory. This memory space is defined in Figure 5.

The calendar determines the 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.

Flow control data from the SPI-4.2 Status interface can be received by the user in one of two ways:

- **Addressable Mode** (AXI4-Lite Control Interface): In this mode, the Status information is stored in the Axi4-Lite Control memory space and can be retrieved at the user's convenience.

- **Transparent Mode** (AXI4-Stream Status Interface): In this mode, the Status information is provided real-time as it is received on TStat and is not stored within the core. This means the status memory space is defined in Figure 5 is not usable ("Reserved") in this mode and reads to it will return all zeroes.

The static configuration memory enables customization of the core based on individual system requirements. These settings are statically driven inside the core by writing registers through the Control interface. Table 9 defines the Control interface signals, and Table 11 defines the static configuration parameters.

Table 9: **Source AXI4-Lite Control interface Signals**

Name	Direction	Clock Domain	Description
AXI_SRC_ACLK	Input	N/A	**AXI4-Lite Interface Clock**. All Sink AXI4-Lite signals are synchronous to the rising edge of this clock.
AXI_SRC_ARESETN	Input	AXI_SRC_ACLK	**AXI4-Lite Interface Reset.** Active-low signal that enables the user to reset the AXI4-Lite interface and all associated logic and memories to chosen CORE Generator settings. The reset signal can be asserted asynchronously, but deassertion must be synchronous after the rising edge of ACLK. See the *AMBA AXI4 Protoco*l specification for more information.
Read Address Channel			
AXI_SRC_ARREADY	Output	AXI_SRC_ACLK	**AXI4-Lite Read Address Core Handshake**. Indicates the core is ready to accept a read address. When both ARREADY and ARVALID are asserted on a clock cycle, a one-word read request occurs for the address provided on ARADDR and the core fetches the contents of that address.
AXI_SRC_ARVALID	Input	AXI_SRC_ACLK	**AXI4-Lite Read Address User Handshake**. Indicates the user is presenting a valid read address. When both ARREADY and ARVALID are asserted on a clock cycle, a one-word read request occurs for the address provided on ARADDR and the core fetches the contents of that address.
AXI_SRC_ARADDR[9:0]	Input	AXI_SRC_ACLK	**AXI4-Lite Read Address.** Address to be read by the user application.

Table 9: **Source AXI4-Lite Control interface Signals** *(Cont'd)*

Name	Direction	Clock Domain	Description
Read Data Channel			
AXI_SRC_RREADY	Input	AXI_SRC_ACLK	**AXI4-Lite Read Data User Handshake**. Indicates the user is ready to accept read data. When both RREADY and RVALID are asserted on a clock cycle, one data beat is read out of the core. This handshake completes the transaction begun by the ARREADY/ARVALID handshake.
AXI_SRC_RVALID	Output	AXI_SRC_ACLK	**AXI4-Lite Read Data Core Handshake**. Indicates the core is presenting valid read data. When both RREADY and RVALID are asserted on a clock cycle, one data beat is read out of the core. This handshake completes the transaction begun by the ARREADY/ARVALID handshake.
AXI_SRC_RDATA[31:0]	Output	AXI_SRC_ACLK	**AXI4-Lite Read Data.** Data read from address location provided on ARADDR is presented here. Bit 0 is the LSB.
Write Address Channel			
AXI_SRC_AWREADY	Output	AXI_SRC_ACLK	**AXI4-Lite Write Address Core Handshake.** Indicates the core is ready to accept a write address. When both AWREADY and AWVALID are asserted on a clock cycle, one write address beat is accepted by the core. Once both the WREADY/WVALID and AWREADY/AWVALID handshakes occur independently, one write transaction is complete.
AXI_SRC_AWVALID	Input	AXI_SRC_ACLK	**AXI4-Lite Write Address User Handshake.** Indicates the user is presenting a valid write address. When both AWREADY and AWVALID are asserted on a clock cycle, one write address beat is accepted by the core. Once both the WREADY/WVALID and AWREADY/AWVALID handshakes occur independently, one write transaction is complete.
AXI_SRC_AWADDR[9:0]	Input	AXI_SRC_ACLK	**AXI4-Lite Write Address**. Address to be written by the user application.
Write Data Channel			
AXI_SRC_WREADY	Output	AXI_SRC_ACLK	**AXI4-Lite Write Data Core Handshake**. Indicates the core is ready to accept write data and strobe. When both WREADY and WVALID are asserted on a clock cycle, one write address beat is accepted by the core. Once both the WREADY/WVALID and AWREADY/AWVALID handshakes occur independently, one write transaction is complete
AXI_SRC_WVALID	Input	AXI_SRC_ACLK	**AXI4-Lite Write Data User Handshake.** Indicates the user is presenting valid write data and strobe. When both WREADY and WVALID are asserted on a clock cycle, one write address beat is accepted by the core. Once both the WREADY/WVALID and AWREADY/AWVALID handshakes occur independently, one write transaction is complete.
AXI_SRC_WDATA[31:0]	Input	AXI_SRC_ACLK	**AXI4-Lite Write Data**. Data written into the core. Bit 0 is the LSB.
AXI_SRC_WSTRB[3:0]	Input	AXI_SRC_ACLK	**AXI4-Lite Write Strobe**. Byte enable associated with AXI_SRC_WDATA. Bit 0 enables WDATA[7:0], bit 1 enables WDATA[15:8], bit 2 enables WDATA[23:16], and bit 3 enables WDATA[32:24].

Table 9: **Source AXI4-Lite Control interface Signals** *(Cont'd)*

Name	Direction	Clock Domain	Description
Write Response Channel			
AXI_SRC_BREADY	Input	AXI_SRC_ACLK	**AXI4-Lite Write Response User Handshake.** Indicates the user is ready to accept a write response. When both BREADY and BVALID are asserted on a rising edge of AXI_SRC_ACLK, one write response transaction is read out of the core.
AXI_SRC_BVALID	Output	AXI_SRC_ACLK	**AXI4-Lite Write Response Core Handshake.** Indicates the core is presenting a valid write response. When both BREADY and BVALID are asserted on a rising edge of AXI_SRC_ACLK, one write response transaction is read out of the core. Each write transaction will generate a write response, regardless of whether the write succeeded. The Error Bus (below) provides further signaling in regards to errors that occur during write operations.
Sideband Error Bus			
SRC_BRESP_ERR[7:0]	Output	AXI_SRC_ACLK	**AXI4-Lite Error Bus.** Each bit of this bus corresponds to a specific AXI4-Lite Error condition. The error conditions detected are reported as follows: • SRC_BRESP_ERR [0]: Write Response counter is full. No further writes or reads will be processed until at least one Write Response is read from the core using BREADY/BVALID. • SRC_BRESP_ERR[1]: Write transaction to the Configuration space failed. • SRC_BRESP_ERR [2]: Write transaction to the Calendar space has failed. • SRC_BRESP_ERR[3]: Write transaction to the Status space has failed. • SRC_BRESP_ERR[4]:Read/Write transaction has missed a valid address. • SRC_BRESP_ERR [5:7]: Reserved bits (tied low).

Source AXI4-Stream Status Interface

The Source AXI4-Stream interface provides flow control information about the SPI-4.2 link.

When the core is used in Addressable Status mode, the interface provides the channel number for the most recently updated SPI-4.2 channel status on M_AXIS_SRCSTAT_TID, qualified by the M_AXIS_SRCSTAT_TVALID signal. The channel status itself must be accessed via the AXI4-Lite Control interface, with each read providing status for four channels.

When the core is used in Transparent Status mode, the M_AXIS_SRCSTAT_TUSER[1:0] bus is added. This port provides the SPI-4.2 status as received on TStat with minimal latency. In this mode, the M_AXIS_SRCSTAT_TID bus indicates the channel to which the current status shown on M_AXIS_SRCSTAT_TUSER belongs. Both M_AXIS_SRCSTAT_TID and M_AXIS_SRCSTAT_TUSER are qualified by M_AXIS_SRCSTAT_TVALID.

Table 10: **Source AXI4-Stream Status Interface Signals**

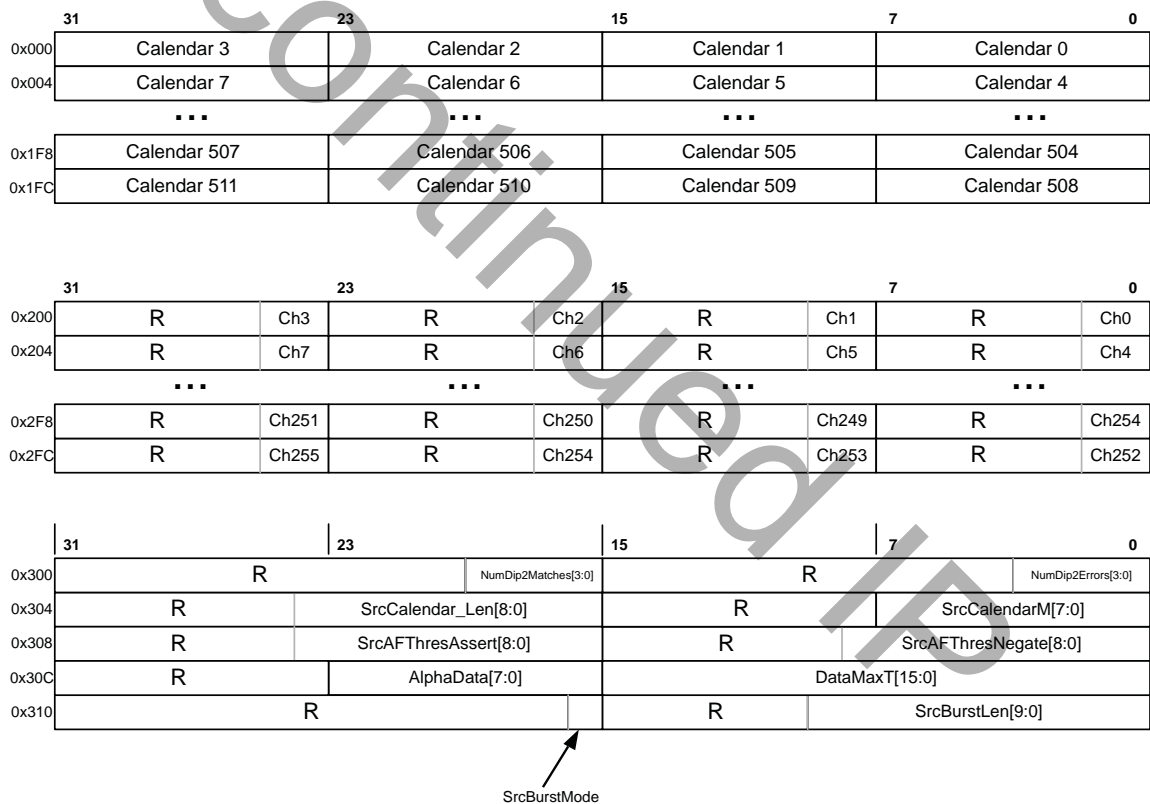| Name	Direction	Clock Domain	Description
M_AXIS_SRCSTAT_ACLK	Input	N/A	**Status Clock**. All status AXI4-Stream signals are synchronous to the rising edge of this clock.
M_AXIS_SRCSTAT_TID[7:0]	Output	M_AXIS_SRCSTAT_ACLK	**Status Channel**. The Source Status Channel is an 8-bit bus containing the channel address that is being updated on M_AXIS_SRCSTAT_TUSER on the current clock cycle.
M_AXIS_SRCSTAT_TVALID	Output	M_AXIS_SRCSTAT_ACLK	**Status Valid**. When asserted indicates that M_AXIS_SRCSTAT_TUSER and M_AXIS_SRCSTAT_TID are valid. When the core is processing DIP-2 or frame words, TVALID is deasserted. A transition of TVALID from 0 to 1 indicates that the core has started a new calendar sequence.
M_AXIS_SRCSTAT_TUSER[1:0]	Output	M_AXIS_SRCSTAT_ACLK	**Status (Only Available in Transparent Mode).** Represents the current status received on the SPI-4.2 TStat bus.

	31		23		15		7		0
0x000	Calendar 3		Calendar 2		Calendar 1		Calendar 0		
0x004	Calendar 7		Calendar 6		Calendar 5		Calendar 4		
	· · ·		· · ·		· · ·		· · ·		
0x1F8	Calendar 507		Calendar 506		Calendar 505		Calendar 504		
0x1FC	Calendar 511		Calendar 510		Calendar 509		Calendar 508		

	31		23		15		7		0
0x200	R	Ch3	R	Ch2	R	Ch1	R	Ch0	
0x204	R	Ch7	R	Ch6	R	Ch5	R	Ch4	
	· · ·		· · ·		· · ·		· · ·		
0x2F8	R	Ch251	R	Ch250	R	Ch249	R	Ch254	
0x2FC	R	Ch255	R	Ch254	R	Ch253	R	Ch252	

	31		23		15		7		0
0x300	R		NumDip2Matches[3:0]		R			NumDip2Errors[3:0]	
0x304	R		SrcCalendar_Len[8:0]		R		SrcCalendarM[7:0]		
0x308	R		SrcAFThresAssert[8:0]		R		SrcAFThresNegate[8:0]		
0x30C	R		AlphaData[7:0]		DataMaxT[15:0]				
0x310	R				R		SrcBurstLen[9:0]		

SrcBurstMode

Figure 5: **Source AXI4-Lite Memory Space**

Table 11: **Source Static Configuration Parameter Definition**

Name	Range	Description
NumDip2Errors[3:0]	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]	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.
SrcCalendarM[7:0]	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 programmed by the AXI4-Lite interface, 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.
SrcCalendarLen[8:0]	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 programmed by the AXI4-Lite interface, 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.
SrcAFThresNegate[8:0]	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 S_AXIS_SRCFF_TREADY is asserted. SrcAFThresNegate is $\geq$ SrcAFThresAssert.
SrcAFThresAssert[8:0]	If SrcBurstMode = 0: 1-508. If SrcBurstMode = 1 and SrcBurstLen < 256, range is SrcBurstLen-508. If SrcBurstMode = 1 and SrcBurstLen >=256, range is 256-508.	**Source Almost Full Threshold Assert**. Specifies the minimum number of empty FIFO locations to exist in the Source FIFO before S_AXIS_SRCFF_TREADY is de-asserted. SrcAFThresNegate $\geq$ SrcAFThresAssert.
DataMaxT[15:0]	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]	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.

Table 11: **Source Static Configuration Parameter Definition** *(Cont'd)*

Name	Range	Description
SrcBurstLen[9:0]	If SrcBurstMode = 1, 1-1023 If SrcBurstMode = 0, 1- 256 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).
SrcBurstMode	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 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, or when the channel address changes.

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-7 and Kintex-7 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-7 and Kintex-7 FPGAs.

Virtex-7 and Kintex-7 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-7 and Kintex-7 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 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. Because of this, it is the recommended alignment solution.

In Virtex-7 and Kintex-7 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 12 defines the Dynamic Phase Alignment signals.

Table 12: **Dynamic Phase Alignment SIgnals**

Signal Name	Direction	Clock Domain	Description
SnkDPAPhaseAlignComplete	Output	RDClkDiv_User	**Phase Alignment Complete.** Active high signal that indicates phase alignment is complete.
SnkDPAPhaseAlignRequest	Input	RDClkDiv_User	**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	RDClkDiv_User	**Phase Alignment Failed**. Active high signal that indicates phase alignment has failed at the end of the alignment sequence.
SnkDPARamAddr [5:0]	Output	RDClkDiv_User	**Phase Alignment RAM Address.** Bus indicating the ISERDES tap value that corresponds to the data on SnkDPARamData.
SnkDPARamData [16:0]	Output	RDClkDiv_User	**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_User	**Phase Alignment RAM Valid**. Active high signal indicating the information on SnkDPARamData and SnkDPARamAddr is valid.
SnkCDPAHalt (optional)	Input	RDClkDiv_User	**Phase Alignment DPA Halt**. Active high signal that enables the user to halt the pointer adjustment of continuous DPA operation.
SnkDPADiagWin (optional)	Input	RDClkDiv_User	**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_User	**Phase Alignment DPA Address Reset.** Active high signal that clears the SnkDPARamAddr counter.
SnkDPAAddrEn (optional)	Input	RDClkDiv_User	**Phase Alignment DPA Address Enable.** Active high signal that enables the SnkDPARamAddr counter.
SnkDPAClkDlyRst (optional)	Output	RDClkDiv_User	**Phase Alignment IDELAY Reset**. Active high signal used to reset the IDELAY inserted in the RDClk path of the external sink clocking module. This option is only available when the "DPA Clock Adjustment" feature is enabled.

Table 12: **Dynamic Phase Alignment SIgnals** *(Cont'd)*

Signal Name	Direction	Clock Domain	Description
SnkDPAClkDlyCe (optional)	Output	RDClkDiv_User	**Phase Alignment IDELAY Clock Enable**. Active high signal used to enable the increment of the IDELAY inserted in the RDClk path of the external sink clocking module.This option is only available when the "DPA Clock Adjustment" feature is enabled.
SnkDPAClkDlyInc (optional)	Output	RDClkDiv_User	**Phase Alignment IDELAY Increment**. Active high signal used to increment the delay value of the IDELAY inserted in the RDClk path of the external sink clocking module.This option is only available when the "DPA Clock Adjustment" feature is enabled.

Static Alignment

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 clocking distribution, this alignment can be performed by using the IDELAY function or an MMCM. 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. The ability of the IDELAY function to delay its input by small increments 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. 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 MMCM

The core also supports legacy static alignment, which uses the MMCM to phase shift the RDClk. The MMCM-based static alignment is only supported for global clocking distribution. The ability of the MMCM to shift the internal clock enables RDClk to be shifted relative to the sampled data. For statically aligned systems, the MMCM output clock phase offset is a critical part of the system. The static alignment solution using 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 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 13 through Table 15 provide the clocking resource count for each clocking option (not including the user interface FIFO clocks) for Virtex-7 and Kintex-7 FPGAs.

The source core reference clock (SysClk) must have 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 MMCM to generate the internal clocks, the SysClk duty cycle must be 45/55 or tighter. For SysClk cores that use regional clocking scheme to generate the internal clocks, the SysClk duty cycle must be 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. For more information, see the UG784, *SPI-4.2 User Guide*.

Table 13: **Sink Core Clocking Option for Virtex-7 and Kintex-7 FPGAs**

Clocking Option	BUFR	BUFG[1]	MMCM
Regional clocking	1	0/1	0
Global clocking	0	4/5	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 14: **Source Core SysClk Clocking Option for Virtex-7 and Kintex-7 FPGAs**

Clocking Option	BUFR	BUFG	MMCM
Regional clocking	1	0	0
Global clocking[1]	0	4	1

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

Table 15: **Source Core TSClk Clocking Option for Virtex-7 and Kintex-7 FPGAs**

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

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

The SPI-4.2 solution delivers the clocking circuitry external to the core. This allows a customized clocking solution to be created based on individual system requirements. An example design is provided to demonstrate the implementation of a clocking module for the core.

Multiple Core Instantiations

It is possible to implement multiple SPI-4.2 cores in a single target device. Larger Virtex-7 and Kintex-7 devices can support more than four SPI-4.2 cores. See *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-annotated HDL) verification
*   Gate-level with back-annotated Timing (with SDF file) verification targeting the following device/frequency combinations:

- Kintex-7 FPGAs -1, -2, and -3 devices up to 1Gbps (depending on speed grade) on the SPI-4.2 Interface and 250MHz on the User Interface (`M_AXIS_SNKFF_ACLK` and `S_AXIS_SRCFF_ACLK` clocks).
- Virtex-7 FPGAs -1, -2, and -3 devices up to 1.25 Gbps (depending on speed grade) on the SPI-4.2 Interface and 312 MHz on the User Interface (`M_AXIS_SNKFF_ACLK` and `S_AXIS_SRCFF_ACLK` clocks).

Hardware validation for 7 series devices is pending.

Device Utilization

Table 16 lists the Block RAM, LUTs, and FFs counts for Virtex-7 and Kintex-7 devices.

Table 16: **Core Utilization – Virtex-7/Kintex-7 Devices**

Core Configuration	Block RAM	LUTs	FFs
64-bit Static Alignment	3 (36k block RAM) 9 (18k block RAM)	4050	4330
64-bit Dynamic Alignment (default)	3 (36k block RAM) 9 (18k block RAM)	5020	4972
128-bit Static Alignment	3 (36k block RAM) 13 (18k block RAM)	4670	4775
128-bit Dynamic Alignment (default)	3 (36k block RAM) 13 (18k block RAM)	5640	5425

Performance in Virtex-7 FPGAs (Preliminary)

Table 17 contains the performance numbers for the SPI-4.2 cores that target Virtex-7 devices.

Table 17: **Performance in Virtex-7 Devices**

Alignment Type	Speed Grade	Performance[1]	
		Sink Core with Global Clocking	Sink Core with Regional Clocking
Static[2]	-1, -2, -3	622-700Mbps	622-700Mbps
Dynamic (source core with regional clocking)	-1	622-900Mbps	622-1.1Gbps
Dynamic (source core with regional clocking)	-2	622-1.1Gbps	622-1.2Gbps
Dynamic (source core with regional clocking)	-3	622-1.25Gbps	622-1.25Gbps
Dynamic (source core with global clocking)	-1, -2	622Mbps-900Mbps	622Mbps-900Mbps
Dynamic (source core with global clocking)	-3	622Mbps-1Gbps	622Mbps-1Gbps

1. Performance numbers shown use Virtex-7 FPGA High Performance (HP) I/O. Use of High Range (HR) I/O is supported but may limit upper bound of performance range. See DS183, *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics* for supported performance ranges using HR I/O. Between this document and DS183, the SPI-4.2 core is limited to the smaller of the two performance numbers.
2. When the sink core is configured with static alignment, only regional clocking is supported on the source core.

Performance in Kintex-7 FPGAs (Preliminary)

Table 18 contains the performance numbers for the SPI-4.2 cores that target Kintex-7 devices.

Table 18: **Performance in Kintex-7 Devices**

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

1. Performance numbers shown use Kintex-7 FPGA High Performance (HP) I/O. Use of High Range (HR) I/O is supported but may limit upper bound of performance range. See DS182, *Kintex-7 FPGAs Data Sheet: DC and Switching Characteristics* for supported performance ranges using HR I/O. Between this document and DS182, the SPI-4.2 core is limited to the smaller of the two performance numbers.

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

This Xilinx LogiCORE IP module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the Vivado Design Suite. For full access to all core functions in simulation and in hardware, you must purchase a license for the core. Contact your local Xilinx sales representative for information about pricing and availability.

For more information, visit the SPI-4.2 product page.

Information about other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
3/1/11	1.0	Initial Xilinx release.
6/22/11	2.0	Updated core to v11.2 and ISE Design Suite to v13.2.

Date	Version	Description of Revisions
10/19/11	2.5	Added footnotes for Table 17 and Table 18 clarifying use of High Performance I/O.
7/25/12	3.0	Updated core to v12.2. Removed ISE Design Suite support. Added Vivado Design Suite v2012.2 support.

Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at http://www.xilinx.com/warranty.htm; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: http://www.xilinx.com/warranty.htm#critapps.