XILINX®

XAPP553 (v1.0) March 2, 2012

Scalable Serdes Framer Interface (SFI-S) for 7 Series FPGAs

Author: Julian Kain

Summary

The Scalable Serdes Framer Interface (SFI-S) is an Optical Internetworking Forum (OIF) standard that defines the electrical connections between devices on a typical optical communications line card. An *n*-bit wide SFI-S configuration contains *n* data channels and one control channel for interface skew compensation. This application note describes a ten data channel SFI-S design targeting Xilinx 7 series FPGAs using GTX or GTH serial transceivers to implement an aggregate 111.8 Gb/s bidirectional interface. The hardware-verified Verilog HDL reference design provides significant skew compensation and fine-grained control of skew tracking. A synthesizable example design with PRBS31 generator and checker logic enables simple simulation and hardware demonstration of the reference design.

Introduction

Scalable Serdes Framer Interface (SFI-S): Implementation Agreement for Interfaces beyond 40G for Physical Layer Devices [Ref 1] specifies the point-to-point electrical connections between the optical module, forward error correction (FEC) processor, and framer devices, which comprise the typical line interface of optical communications systems with 80–160 Gb/s links. Because the maximum data rate per electrical signal is less than the optical data rate, a multi-bit bus is required. SFI-S is defined to support an *n*-bit wide data bus for *n* = 4 to 20 with each channel operating at data rates defined by the common electrical interface (CEI) and the CEI short-reach (SR) electrical specification [Ref 2]. An additional channel contains out-of-band data samples to enable the deskew algorithm, which operates continuously on the sink side of the interface to track and compensate for skew. Figure 1 shows the SFI-S System Reference Model from the OIF-SFI-S-01.0 Implementation Agreement [Ref 1].

© Copyright 2012 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. AMBA and ARM are registered trademarks of ARM in the EU and other countries. All other trademarks are the property of their respective owners.

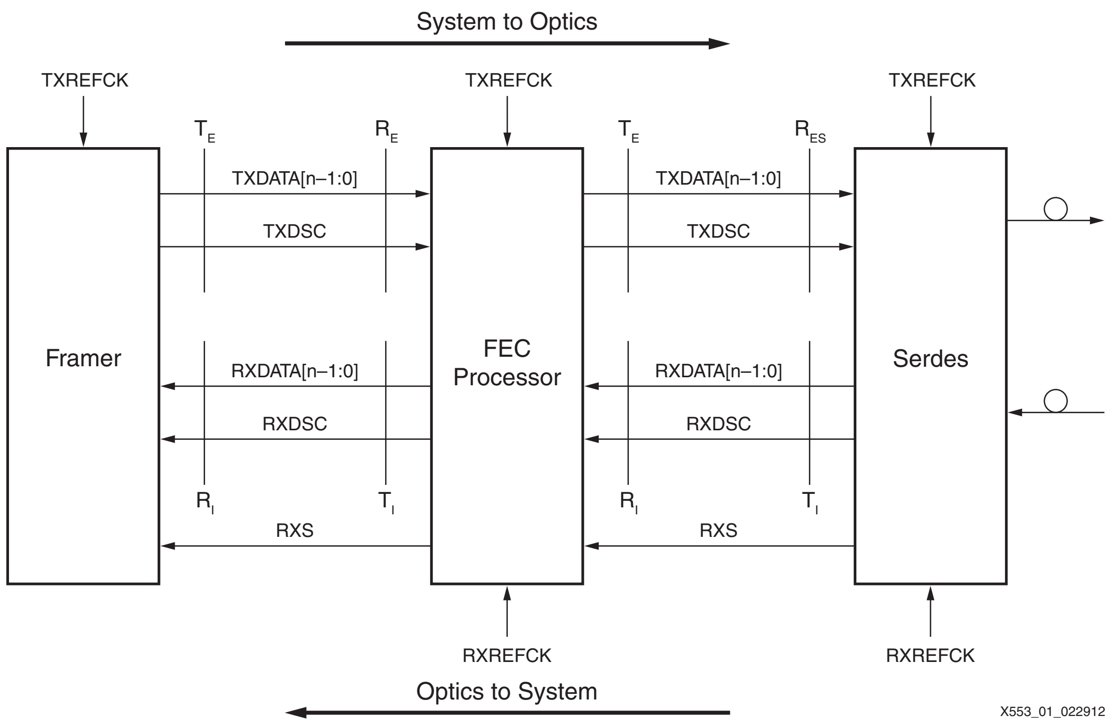

Figure 1: **SFI-S System Reference Model from OIF-SFI-S-01.0**

This application note describes the design and use of the accompanying reference design, which provides a ten data channel (plus one deskew channel) bidirectional SFI-S implementation targeting Xilinx 7 series FPGAs. The per-channel line rate as configured is 11.18 Gb/s, resulting in a link rate of 111.8 Gb/s. The reference design provides hardware-verified Verilog HDL implementations of both source and sink interfaces that can be used at any FPGA device interface in the system-to-optics (transmit) or optics-to-system (receive) direction. A synthesizable example design with PRBS31 generator and checker logic and accompanying design constraints and implementation scripts support rapid hardware demonstration. A simple test bench with SFI-S interface loopback and skew injection and accompanying simulation scripts demonstrates skew compensation and subsequent PRBS31 checker lock.

To support CEI-11G line rates, the reference design makes use of GTX serial transceivers in Kintex-7 and Virtex-7 T devices, or GTH serial transceivers in Virtex-7 XT devices. To configure the serial transceivers for CEI-11G-SR electrical characteristics and abstract much of their complexity, the reference design uses instantiation wrappers generated by the LogiCORE™ IP Xilinx 7 series FPGAs Transceivers Wizard as a starting point. Code comments in the wrapper files clearly identify any changes that were made.

SFI-S is a synchronous interface in which both the source and sink sides of a link share a reference clock for a given system direction. While both a TXREFCK (system-to-optics reference clock) and an RXREFCK (optics-to-system reference clock) are shown, the specification also allows for a single reference clock for both system directions. To use both the TX and RX paths of each serial transceiver, a total of eleven serial transceivers are required across both source and sink interface implementations for a device. The reference design therefore uses a common reference clock, and the instantiation wrappers configure the serial transceivers for bidirectional operation. If independent reference clocks are required, the design can be readily modified, in part by re-customizing the Xilinx 7 series FPGAs Transceivers Wizard using the provided core configuration (`.xco`) file.

The reference design conforms to SFI-S performance specifications. Specifically, a well-placed implementation exhibits skew of less than the 500 ps budget at the T_E and T_I system points shown in Figure 1, the sink logic is capable of compensating for skew far above the 1500 ps minimum requirement, and CEI-11G line rates are supported subject to device speed grade limitations. For GTX and GTH serial transceiver characterization reports and other Xilinx 7 series FPGAs documentation, see the Xilinx support site at http://www.xilinx.com/support.

Reference Design Overview

As shown in Figure 2, the SFI-S top-level module contains the three functional blocks of the reference design:

- The *source block* implements the parallel portion of the source interface, striping user data input onto the data channels and constructing the deskew reference frame.
- The *sink block* implements the parallel portion of the sink interface, using the deskew channel to track and compensate for skew on the interface, and de-striping the data channels onto the user data output.
- The *transceivers block* implements the eleven channels' bidirectional serial transceivers and supporting logic using the instantiation wrappers to simplify the common interface with the source and sink blocks. Clocking and reset logic are also centralized within the transceivers block.

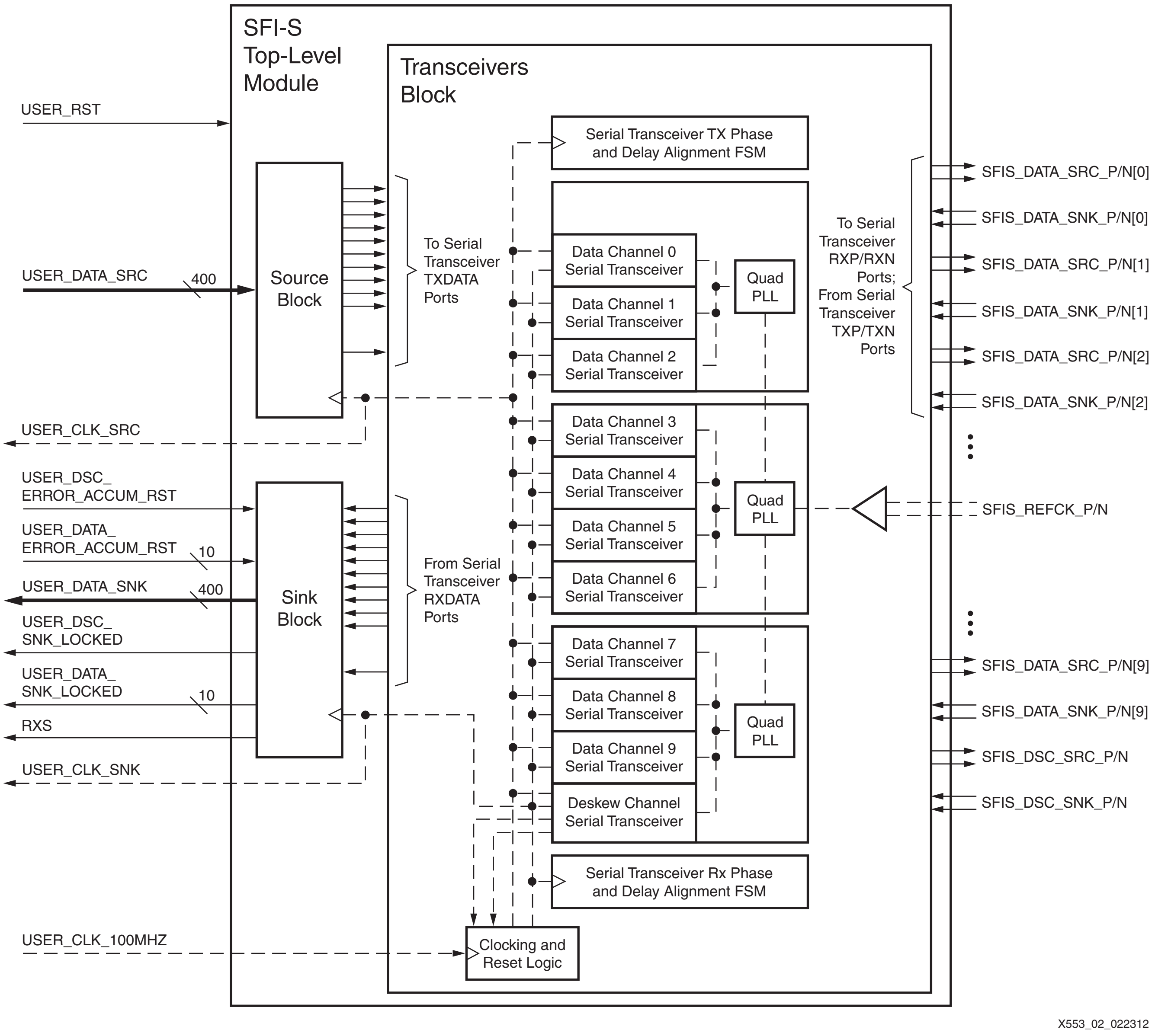
SFI-S Top-Level Module
Transceivers Block
USER_RST
USER_DATA_SRC
400
Source Block
USER_CLK_SRC
USER_DSC_ ERROR_ACCUM_RST
USER_DATA_ ERROR_ACCUM_RST
10
USER_DATA_SNK
400
Sink Block
USER_DSC_ SNK_LOCKED
USER_DATA_ SNK_LOCKED
10
RXS
USER_CLK_SNK
USER_CLK_100MHZ
Clocking and Reset Logic
To Serial Transceiver TXDATA Ports
From Serial Transceiver RXDATA Ports
Serial Transceiver TX Phase and Delay Alignment FSM
Data Channel 0 Serial Transceiver
Data Channel 1 Serial Transceiver
Data Channel 2 Serial Transceiver
Data Channel 3 Serial Transceiver
Data Channel 4 Serial Transceiver
Data Channel 5 Serial Transceiver
Data Channel 6 Serial Transceiver
Data Channel 7 Serial Transceiver
Data Channel 8 Serial Transceiver
Data Channel 9 Serial Transceiver
Deskew Channel Serial Transceiver
Serial Transceiver Rx Phase and Delay Alignment FSM
Quad PLL
Quad PLL
Quad PLL
To Serial Transceiver RXP/RXN Ports; From Serial Transceiver TXP/TXN Ports
SFIS_DATA_SRC_P/N[0]
SFIS_DATA_SNK_P/N[0]
SFIS_DATA_SRC_P/N[1]
SFIS_DATA_SNK_P/N[1]
SFIS_DATA_SRC_P/N[2]
SFIS_DATA_SNK_P/N[2]
SFIS_REFCK_P/N
SFIS_DATA_SRC_P/N[9]
SFIS_DATA_SNK_P/N[9]
SFIS_DSC_SRC_P/N
SFIS_DSC_SNK_P/N
X553_02_022312

Figure 2: **SFI-S Reference Design Top-Level Module Block Diagram**

The transceivers block organizes the serial transceivers into three quads, each containing an LC-based Quad PLL (QPLL) driven by the SFI-S differential reference clock buffer. The TX portion of the deskew channel serial transceiver provides a divided version of the reference clock, used for all source interface sequential parallel logic. The serial transceiver TX phase and delay alignment procedure enables this common clocking methodology while minimizing lane-to-lane skew at the source device pins. The RX portion of the deskew channel serial transceiver recovers the clock from the received data stream and provides a divided version which is used for all sink interface sequential parallel logic. The serial transceiver RX phase and delay alignment procedure enables this common clocking methodology while minimizing degradation of the skew compensation margin.

To initialize the design, the user provides a free-running 100 MHz clock and a master asynchronous reset pulse to the top-level module. This begins the serial transceiver bring-up sequence:

1. Each of the three QPLLs locks onto the SFI-S reference clock.

2. The TX paths of all serial transceivers are reset and initialized in sequential mode. The TX phase and delay alignment procedure is performed on each serial transceiver.
3. The RX paths of all serial transceivers are reset and initialized in sequential mode. The RX phase and delay alignment procedure is performed on each serial transceiver.

Refer to *7 Series FPGAs GTX Transceivers User Guide* [Ref 3] for a detailed description of this sequence.

The source clock provided to the user as a top-level output begins to toggle during the serial transceiver TX path bring-up sequence. Beginning as soon as is feasible following reset, the user must provide a new 400-bit data vector on the source data input port synchronous to each rising edge of the source clock throughout operation. The source block immediately begins to perform data channel striping and deskew reference frame construction for serial transceiver transmission. See Source Interface, page 7 for design and usage details.

After the serial transceiver RX path bring-up sequence, the sink block initializes the deskew algorithm that tracks and compensates for skew throughout operation. An interface alignment output qualifies the 400-bit data vector, which is provided to the user on the sink data output port synchronous to each rising edge of the sink clock. The sink clock is also provided to the user as a top-level output. A combination of design parameters and top-level inputs enable precise control of skew tracking sensitivity and tolerance to bit errors. See Sink Interface, page 10 for design and usage details.

Table 1 describes the SFI-S reference design top-level ports. All signals are active-High unless stated otherwise.

Table 1: **SFI-S Reference Design Top-Level Port Descriptions**

Port Name	Direction	Width	Clock Domain	Description
User Interface				
USER_CLK_100MHZ	Input	1		Free-running 100 MHz interface bring-up clock.
USER_RST	Input	1	Asynchronous	Master system reset. Pulse High for at least 1 USER_CLK_100MHZ cycle to initialize interface bring-up.
USER_CLK_SRC	Output	1		Source interface parallel logic clock. Frequency is 1/40 serial transceiver line rate, e.g., 279.5 MHz for 11.18 Gb/s line rate.
USER_DATA_SRC[399:0]	Input	400	USER_CLK_SRC	Source interface user data vector, striped onto and transmitted by the SFI-S source interface.
USER_CLK_SNK	Output	1		Sink interface parallel logic clock. Frequency is 1/40 serial transceiver line rate, e.g., 279.5 MHz for an 11.18 Gb/s line rate.
USER_DSC_ERROR_ACCUM_RST	Input	1	USER_CLK_SNK	Deskew channel error accumulator reset. Clears the deskew reference frame parity error accumulator.
USER_DATA_ERROR_ACCUM_RST[9:0]	Input	10	USER_CLK_SNK	Data channel error accumulator reset. Clears the data channel bit sample error accumulator. Bit *i* maps to channel *i*.

Table 1: **SFI-S Reference Design Top-Level Port Descriptions *(Cont'd)***

Port Name	Direction	Width	Clock Domain	Description
USER_DATA_SNK[399:0]	Output	400	USER_CLK_SNK	Sink interface user data vector de-striped from the SFI-S sink interface for user consumption.
USER_DSC_SNK_LOCKED	Output	1	USER_CLK_SNK	Deskew channel alignment state machine lock indicator signifying deskew reference frame alignment.
USER_DATA_SNK_LOCKED[9:0]	Output	10	USER_CLK_SNK	Data channel deskew state machine lock indicator indicating successful per-channel skew compensation. Bit *i* maps to channel *i*.
RXS	Output	1	USER_CLK_SNK	Receive status: • When 0 (idle), the sink interface is in alignment with compensated skew. • When 1 (receive alarm), the interface is out of alignment. Present on SFI-S interface in the receive system direction only.
SFI-S Interface				
SFIS_REFCK_P SFIS_REFCK_N	Input	1 (differential)		SFI-S common differential reference clock. Frequency is 1/16 serial transceiver line rate, e.g., 698.75 MHz for an 11.18 Gb/s line rate.
SFIS_DSC_SRC_P SFIS_DSC_SRC_N	Output	1 (differential)	Serial clock	Differential SFI-S source interface deskew channel.
SFIS_DATA_SRC_P[9:0] SFIS_DATA_SRC_N[9:0]	Output	10 (differential)	Serial clock	Differential SFI-S source interface data channels. Pair *i* maps to channel *i*.
SFIS_DSC_SNK_P SFIS_DSC_SNK_N	Input	1 (differential)	Serial clock	Differential SFI-S sink interface deskew channel.
SFIS_DATA_SNK_P[9:0] SFIS_DATA_SNK_N[9:0]	Input	10 (differential)	Serial clock	Differential SFI-S sink interface data channels. Pair *i* maps to channel *i*.

Table 2 describes the SFI-S reference design top-level parameters. Signals and parameters are described in further detail in the relevant sections of this application note.

Table 2: **SFI-S Reference Design Top-Level Parameter Descriptions**

Parameter Name	Default Value	Legal Range	Description
SERIAL_TRANSCEIVER_TYPE	"7SERIES_GTX"	"7SERIES_GTX", "7SERIES_GTH"	Controls which serial transceiver resources are instantiated: • Kintex-7 and Virtex-7 T devices use GTX transceivers. • Virtex-7 XT devices use GTH transceivers.
DSC_MATCH_CYC_TO_LOCK	6'd32	6'd1 … 6'd62	Sink deskew channel consecutive reference frame parity match search cycles required to lock.

Table 2: **SFI-S Reference Design Top-Level Parameter Descriptions *(Cont'd)***

Parameter Name	Default Value	Legal Range	Description
DSC_ERR_CYC_TO_UNLOCK	5'd16	5'd1 … 5'd30	Sink deskew channel reference frame parity error accumulator value required to unlock.
DATA_MATCH_CYC_TO_LOCK	6'd4	6'd1 … 6'd62	Sink data channel consecutive bit sample match cycles required to lock.
DATA_ERR_CYC_TO_UNLOCK	5'd4	5'd1 … 5'd30	Sink data channel bit sample error accumulator value required to unlock.
SIMULATION_SPEEDUP	0	0, 1	Accelerates the serial transceiver simulation model bring-up sequence.

Source Interface

A combination of FPGA logic and serial transceivers implements the SFI-S source interface, which can be used at any FPGA device interface in the system-to-optics (transmit) or optics-to-system (receive) direction. The FPGA logic continually stripes user-provided data onto a set of ten vectors corresponding to the ten data channels, each of which is serialized and transmitted by the TX path of a serial transceiver. The provided data is also continually sampled to construct a deskew reference frame vector that is serialized and transmitted by the TX path of the deskew channel serial transceiver. Figure 3 is a simplified block diagram of the source interface showing its structure, clocking, and data flow from the user interface through the TX path of the serial transceivers.

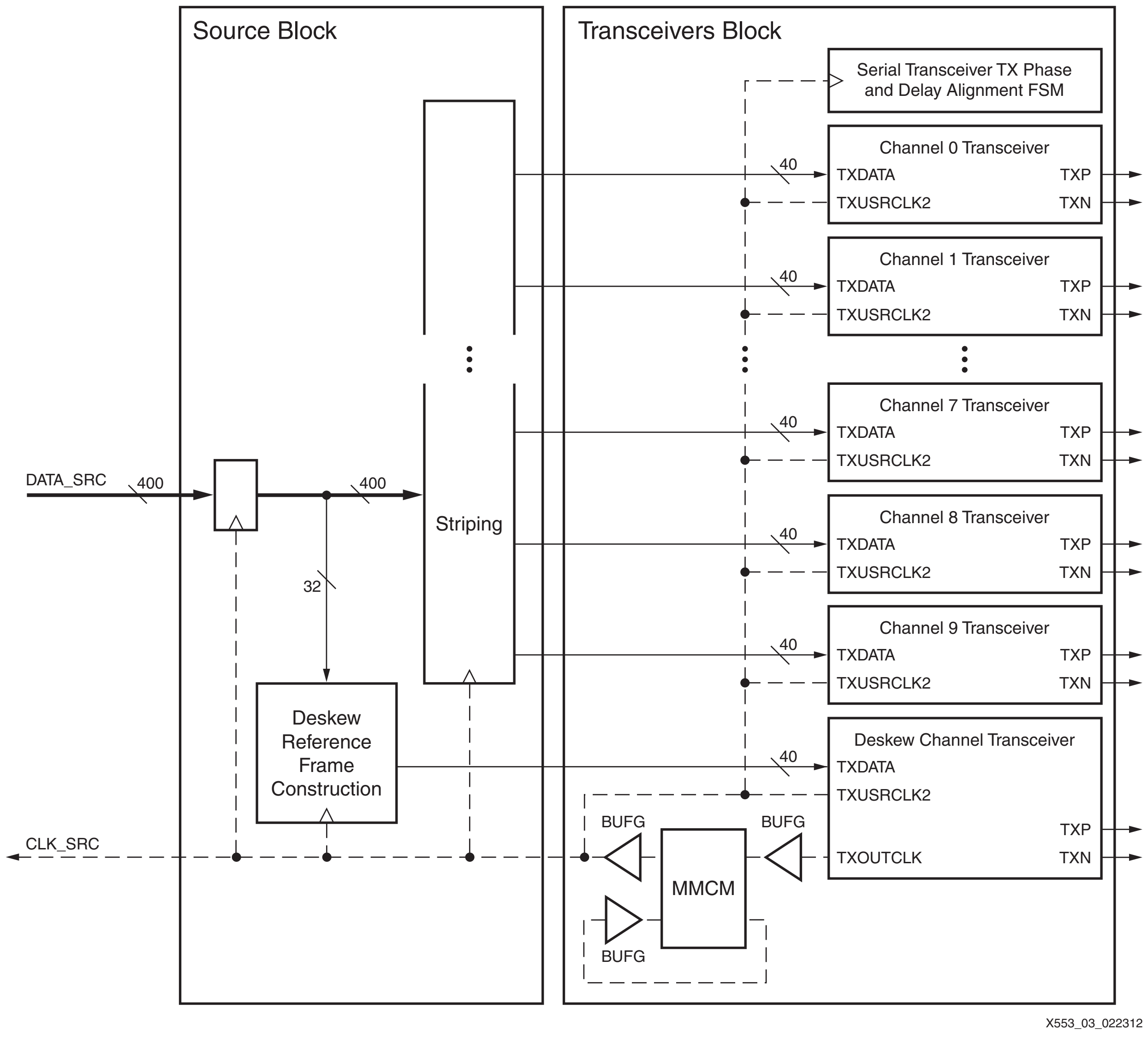

Figure 3: **Source Interface Simplified Block Diagram: Structure, Clocking, and Data Flow**

Source Block

The user is expected to provide a new 400-bit data vector synchronous to each rising edge of the source clock. This user data vector input is then registered once for timing isolation before the data channel striping and deskew reference frame construction functions.

The registered 400-bit user data vector is striped onto ten 40-bit vectors, each corresponding to one of the ten data channels. Because each serial transceiver transmits bits in the order from LSB to MSB, a Verilog generate statement implements the specified striping as an iterative mapping of the user data vector onto bit position *i* of each data channel vector in sequence, incrementing *i* from 0 towards 39 with each loop iteration. As shown in Table 3, the user data vector LSB (bit 0) maps to the first bit to be transmitted on data channel 9, while the MSB (bit 399) maps to the last bit to be transmitted on data channel 0.

Table 3: **Striped Mapping of the User Data Vector onto Data Channels**

User Data Vector Bit Number	Data Channel Number, Vector Bit Number
0	Data channel 9, bit 0
1	Data channel 8, bit 0
2	Data channel 7, bit 0
3	Data channel 6, bit 0
4	Data channel 5, bit 0
5	Data channel 4, bit 0
6	Data channel 3, bit 0
7	Data channel 2, bit 0
8	Data channel 1, bit 0
9	Data channel 0, bit 0
10	Data channel 9, bit 1
11	Data channel 8, bit 1
⋮	⋮
398	Data channel 1, bit 39
399	Data channel 0, bit 39

As illustrated in Figure 2: Example Reference Frame Generation for n = 10 of the OIF-SFI-S-01.0 Implementation Agreement [Ref 1], the specified deskew reference frame for a ten data channel interface consists of two even parity reference frame elements and one odd parity reference frame element. The source block constructs one 40-bit deskew channel vector from samples of the registered user data vector with the requisite 4-input XOR and XNOR functions on each source clock cycle. Because the deskew channel vector size (40 bits) is not an integer multiple of the deskew reference frame length (15 bits), vectors do not contain an integer number of reference frames. However, because a total of 120 bits—the least common multiple of the two quantities—are constructed in three clock cycles, reference frame orientation within the vector repeats every third cycle. A cyclical three-step state machine orients the sampling and construction of each vector resulting in correct and continuous deskew channel contents as shown in Figure 4.

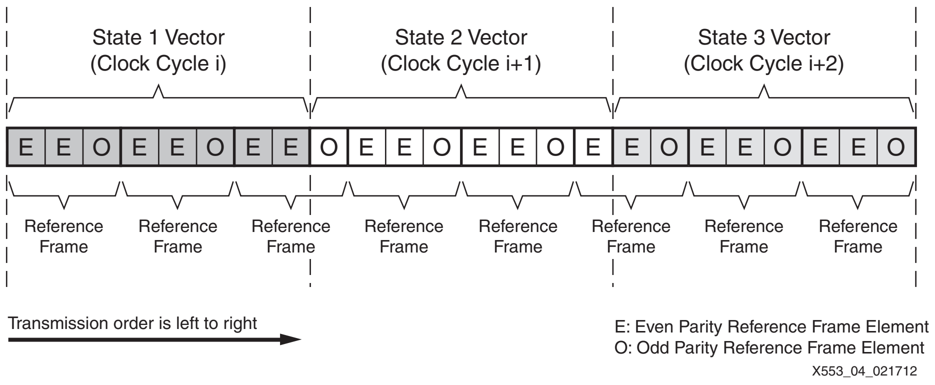

Figure 4: **Three-Step Construction of Continuous Deskew Channel Contents**

The ten data channel vectors and the deskew channel vector are presented in alignment to their respective serial transceivers, synchronous to each rising edge of the source clock.

Transceivers Block: TX Path

A total of eleven serial transceivers are used: one for the deskew channel and one for each of the ten data channels. A phase interpolator circuit within each serial transceiver effectively multiplies the reference clock to serial line frequency which then clocks bits out of a parallel-in-serial-out (PISO) structure for transmission as a serial data stream. Data vectors are written into the PISO structure with a divided version of the serial clock called the physical medium attachment (PMA) parallel clock. Because all channels share a reference clock and operate at the same line rate, the nominal frequency of each serial transceiver's PMA parallel clock is the same. Provided that phase differences are first resolved, a single clock of that frequency can therefore be used to write vectors into the user interface of all serial transceivers and to clock all source-side sequential FPGA logic. The deskew channel serial transceiver is configured and wired to provide that common source clock. The TXOUTCLK port of the deskew channel serial transceiver drives the divided QPLL reference clock, TXPLLREFCLK_DIV2, which is further divided to the appropriate frequency by a mixed-mode clock manager (MMCM).

Because the SFI-S interface does not add encoding to the data, each serial transceiver's transmit path is configured for the unencoded raw mode. To simplify deskew reference frame construction and maintain consistency with the sink interface, the 40-bit external and internal datapath width is used. The resulting common source clock frequency is:

$$f_{src} = \frac{f_{\mathrm{LineRate}}}{40} \qquad \textit{Equation 1}$$

For example, f_{src} is 279.5 MHz for the 11.18 Gb/s line rate.

Due to a combination of serial clock divider bring-up variability and FPGA clock tree skew, a phase difference initially exists between each serial transceiver's PMA parallel clock and the common source clock at its TXUSRCLK input. While the domain boundary can be crossed with the TX buffer, initial differences in buffer position between channels persist. This increases channel-to-channel skew, potentially resulting in violation of the source skew budget and diminishment or depletion of the sink-side skew compensation margin. The TX buffer is therefore bypassed in favor of phase and delay alignment by which each serial transceiver's PMA parallel clock is independently and continually aligned with its TXUSRCLK, facilitating domain crossing. The TX Phase and Delay Alignment in Manual Mode procedure is implemented as described in *7 Series FPGAs GTX Transceivers User Guide* [Ref 3].

Because each serial transceiver's PMA parallel clock writes data vectors into its PISO structure for serial transmission, a difference in PMA parallel clock phase between two serial transceivers corresponds to skew between those channels on the SFI-S source device pins, point T_E or T_I, as shown in Figure 1. Because the phase and delay alignment process aligns each serial transceiver's PMA parallel clock with its TXUSRCLK, and because the common source clock drives TXUSRCLK for all serial transceivers, skew on the common source clock net as measured between serial transceiver TXUSRCLK inputs should be minimized. Mitigation techniques such as balanced serial transceiver placement and global clock buffering are encouraged.

Sink Interface

A combination of serial transceivers and FPGA logic implements the SFI-S sink interface, which can be used at any FPGA device interface in the optics-to-system (receive) or system-to-optics (transmit) direction. For the deskew channel and each data channel, the RX path of a serial transceiver recovers and converts the received data stream to parallel data vectors with which the FPGA logic performs the deskew algorithm. Figure 5 is a simplified block diagram of the sink interface showing its structure, clocking, and data flow from the RX path of the serial transceivers to the user interface.

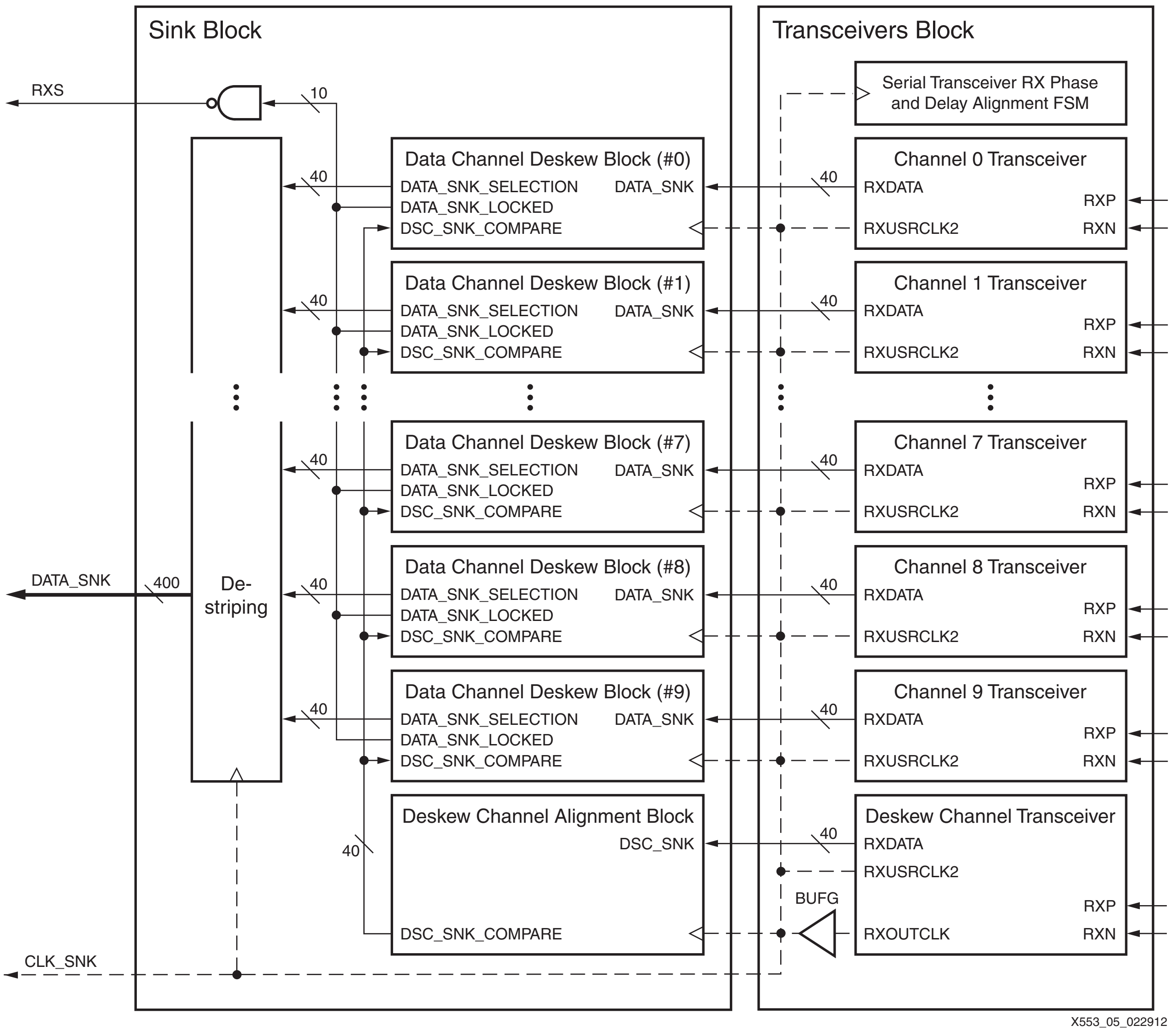

Figure 5: **Sink Interface Simplified Block Diagram: Structure, Clocking, and Data Flow**

Transceivers Block: RX Path

A total of eleven serial transceivers are used: one for the deskew channel and one for each of the ten data channels. The clock data recovery (CDR) circuit within each serial transceiver extracts the recovered clock and data from the received SFI-S signal. Using the recovered clock, the recovered data is written into a serial-in-parallel-out (SIPO) structure, from which a divided version of that clock reads out data vectors. The contents of each data channel are assumed to be sufficiently random so as to result in a reasonable toggle rate for CDR usage.

Because the source and sink sides of an SFI-S link share a reference clock, the nominal frequency of each channel's recovered clock is the same. Provided that phase differences are first resolved, a single clock of that frequency can therefore be used to read received vectors out of the user interface of all serial transceivers, and to clock all sink-side sequential FPGA logic. Because the alternating parity format of the deskew reference frame guarantees a minimum toggle rate of one in every eighteen bits, the deskew channel serial transceiver is configured and wired to provide that common sink clock. The RXOUTCLK port of the deskew channel serial transceiver drives the recovered and divided PMA parallel clock, RXOUTCLKPMA.

Due to a combination of CDR bring-up variability, SFI-S interface skew, and FPGA clock tree skew, a phase difference initially exists between each serial transceiver's recovered and divided PMA parallel clock and the common sink clock at its RXUSRCLK input. While the domain boundary can be crossed with the RX elastic buffer, initial differences in buffer position between channels persist, potentially increasing effective channel-to-channel skew and thereby diminishing the overall skew compensation margin. The RX elastic buffer is therefore bypassed in favor of phase and delay alignment by which each serial transceiver's PMA parallel clock is independently and continually aligned with its RXUSRCLK, facilitating domain crossing. The RX Phase and Delay Alignment in Manual Mode procedure is implemented as described in *7 Series FPGAs GTX Transceivers User Guide* [Ref 3].

Because the SFI-S interface is agnostic to and unaware of the data contents, inter-channel alignment techniques within the serial transceivers, such as comma alignment or channel bonding, are not available. The receive path is therefore configured for the unencoded raw mode, and skew compensation is performed in FPGA logic. To balance a reasonable sink clock frequency with the FPGA logic resource requirements of the deskew algorithm, the 40-bit external and internal datapath width is used. The resulting common sink clock frequency is:

$$f_{\text{snk}} = \frac{f_{\text{LineRate}}}{40} \qquad \textit{Equation 2}$$

For example, f_{snk} is 279.5 MHz for the 11.18 Gb/s line rate.

Sink Deskew Channel Alignment Block

The deskew algorithm depends on locating the deskew reference frame in the deskew channel. The function of the sink deskew channel alignment block is to identify and lock onto the reference frame within the vectors provided by the deskew channel serial transceiver.

By shifting three consecutive 40-bit vectors into a 120-bit triplet, a stable search space is available on every third clock cycle. Because 120 bits is a common multiple of the reference frame length (15 bits) and the serial transceiver vector size (40 bits), and because the sink clock is derived from the deskew channel, alignment nominally remains constant with respect to the triplet after interface bring-up.

The repeating reference frame can take one of fifteen possible alignments or offsets. An offset of 0 is defined as a reference frame starting at triplet bit 0 with subsequent reference frames starting at bits 15, 30, 45, 60, 75, 90, and 105. For other offsets, reference frames start at bits offset + 15*i*, for *i* = 0 to 6.

By identifying the characteristic even-even-odd parity pattern of the reference frame elements, the deskew channel alignment state machine locks onto stable deskew channel contents. Eight sets of two XOR gates and one XNOR gate originating at triplet offset 0 test each of the reference frames' elements for the expected parity pattern.

Figure 6 shows the 120-bit shift register triplet and reference frame alignment test structure. Two alignment scenarios are illustrated in detail: offset 0, in which all parity test gates drive logic 0 to indicate alignment and offset 3, in which they do not.

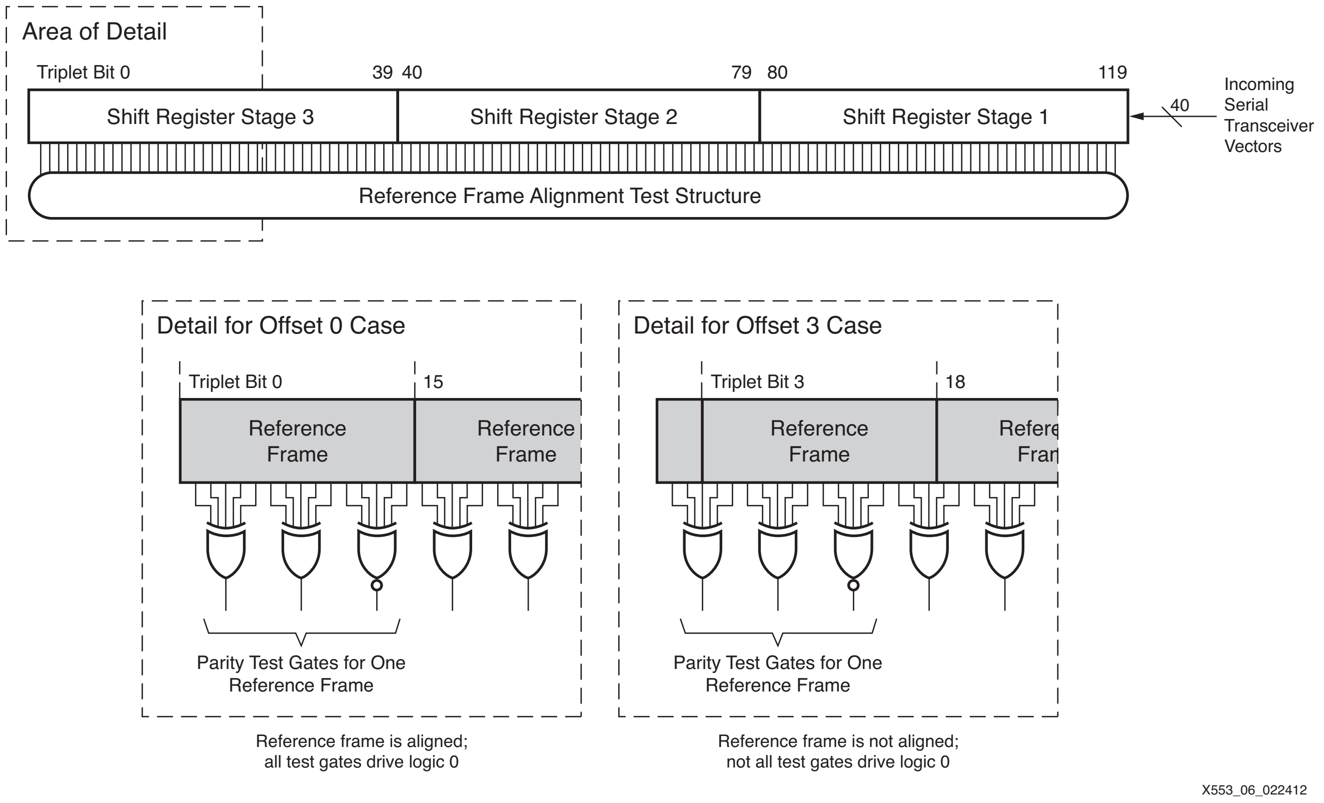
Area of Detail
Triplet Bit 0
39 40
79 80
119
Shift Register Stage 3
Shift Register Stage 2
Shift Register Stage 1
40
Incoming Serial Transceiver Vectors
Reference Frame Alignment Test Structure
Detail for Offset 0 Case
Triplet Bit 0
15
Reference Frame
Reference Frame
Parity Test Gates for One Reference Frame
Reference frame is aligned; all test gates drive logic 0
Detail for Offset 3 Case
Triplet Bit 3
18
Reference Frame
Parity Test Gates for One Reference Frame
Reference frame is not aligned; not all test gates drive logic 0
X553_06_022412

Figure 6: **Deskew Channel Shift Register and Reference Frame Alignment Test Structure**

To avoid complex multiplexing, the state machine forces reference frame alignment to offset 0 by pulsing the serial transceiver RXSLIDE input to bit-slip the provided vector as necessary. After all test gates drive logic 0 for the configurable value DSC_MATCH_CYC_TO_LOCK consecutive search cycles, the state machine transitions to the locked state.

In the locked state, the state machine has a configurable tolerance for errors on the deskew channel before losing lock. A detectable error event such as a single bit error manifests as a parity mismatch and is indicated by a test gate driving logic 1. An error accumulator counts search cycles containing one or more such errors. If the accumulator reaches the configurable value DSC_ERR_CYC_TO_UNLOCK, the state machine returns to the search state and the search procedure restarts. The user can clear the accumulator at any time by asserting the deskew channel error accumulator reset input. Figure 7 shows a simplified representation of the state machine.

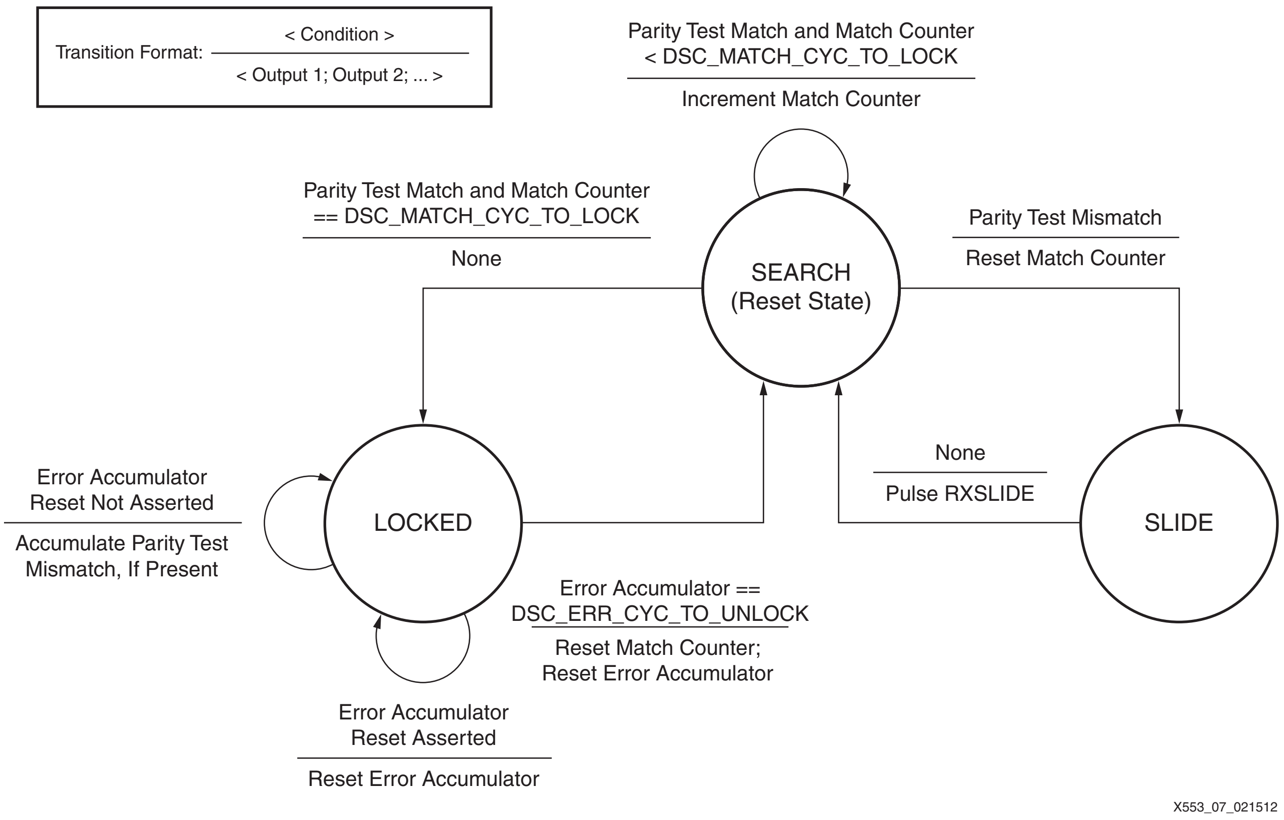
Transition Format:
< Condition >
< Output 1; Output 2; ... >
Parity Test Match and Match Counter < DSC_MATCH_CYC_TO_LOCK
Increment Match Counter
Parity Test Match and Match Counter == DSC_MATCH_CYC_TO_LOCK
None
SEARCH (Reset State)
Parity Test Mismatch
Reset Match Counter
Error Accumulator Reset Not Asserted
Accumulate Parity Test Mismatch, If Present
LOCKED
None
Pulse RXSLIDE
SLIDE
Error Accumulator == DSC_ERR_CYC_TO_UNLOCK
Reset Match Counter; Reset Error Accumulator
Error Accumulator Reset Asserted
Reset Error Accumulator
X553_07_021512

Figure 7: **Simplified Deskew Channel Alignment State Machine**

The user should consider system requirements and characteristics when determining appropriate values for the DSC_MATCH_CYC_TO_LOCK and DSC_ERR_CYC_TO_UNLOCK parameters, and the frequency of deskew channel error accumulator reset assertions. Increasing the value of DSC_MATCH_CYC_TO_LOCK logarithmically decreases the probability of locking onto erroneous or misaligned deskew channel contents at the expense of additional lock time. The value of DSC_ERR_CYC_TO_UNLOCK and the frequency of accumulator reset assertions together control error tolerance. For example, a small parameter value with no, or infrequent, accumulator resets implies a low tolerance for any errors while a larger parameter value with infrequent accumulator resets might better tolerate rare but bursty errors.

When the deskew channel alignment state machine is locked, the data channel deskew blocks are released from reset and the deskew algorithm operates.

Sink Data Channel Deskew Block

Signals within the SFI-S data bus can experience different total delays from source device to sink device, and those delays might change during operation due to variations in external conditions such as temperature and voltage. The function of the sink data channel deskew block is to implement the deskew algorithm by identifying the skew on a data channel with respect to the deskew channel and compensating for it.

One instance of the data channel deskew block is present for each data channel, and the instance operates continually and independent of other instances. Because the deskew channel is the common reference, the sink interface is fully aligned when the skew has been compensated on all data channels.

The data channel deskew block consists of:

- Bit sample comparators to detect whether the data channel is in alignment with the deskew channel.
- A barrel shifter structure to compensate for skew in positive and negative directions in 1-bit increments.
- A feedback control state machine partly used to adjust the barrel shifter based on the comparator results.

On each rising edge of the sink clock, the data channel serial transceiver provides a 40-bit vector that is shifted into a five-stage, 200-bit shift register. The barrel shifter can select any 40-bit segment of this shift register, enabling the search space used to identify and then compensate for the skew.

Although the skew is initially unknown, the deskew reference frame alignment is known and stable because the data channel deskew block operates only after the deskew channel alignment block has locked. The bit sample comparators test for the specified mapping of data channel bits onto the deskew channel by comparing the appropriate data bits from those selected by the barrel shifter to the appropriate bit samples from the stable reference frame. (A Verilog generate statement implements the comparator wiring based on the instance's channel number parameter value.) The comparators continually drive logic 1 when the selected data channel segment is in alignment with the reference frame.

In the zero skew case, the total delays of the data and deskew channels are equivalent at the FPGA logic boundary. Because there is no skew, and because both channels are fully synchronous to the sink clock, the specified mapping is observed when comparing temporally equivalent selections from their respective shift registers. Specifically, bit sample comparators indicate alignment when the barrel shifter selects the middle 40 bits of the 200-bit data channel shift register, and when the oldest 40 bits of the 120-bit deskew channel shift register are used as the reference. Figure 8 illustrates this case, where the shaded portion of the data channel shift register indicates the barrel shifter selection.

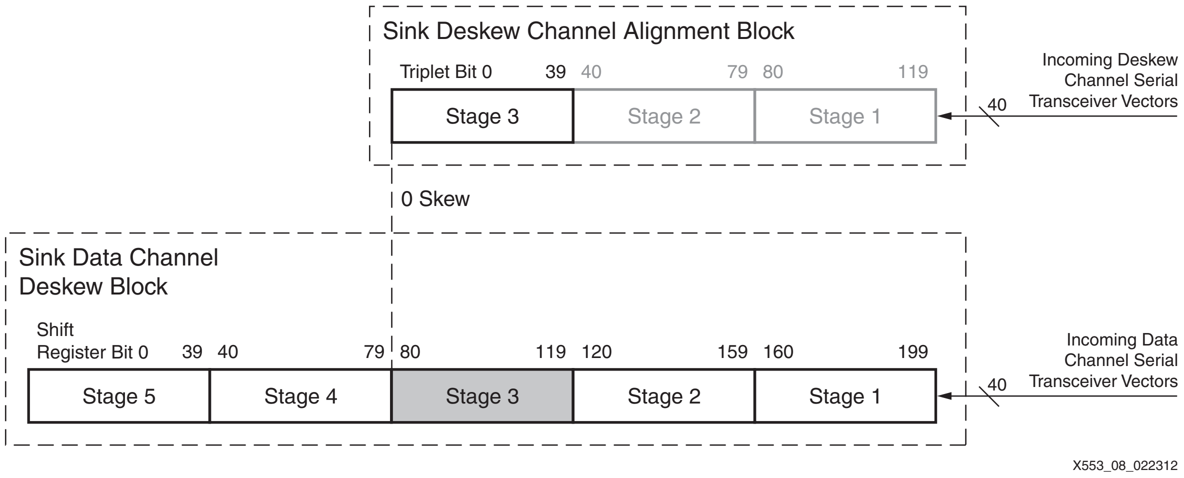
Sink Deskew Channel Alignment Block
Triplet Bit 0
39
40
79
80
119
Stage 3
Stage 2
Stage 1
Incoming Deskew Channel Serial Transceiver Vectors
40
0 Skew
Sink Data Channel Deskew Block
Shift Register Bit 0
39
40
79
80
119
120
159
160
199
Stage 5
Stage 4
Stage 3
Stage 2
Stage 1
Incoming Data Channel Serial Transceiver Vectors
40
X553_08_022312

Figure 8: **Data Channel Shift Register with Barrel Shifter Selecting for Zero Skew Compensation**

The third and final stage of the deskew channel triplet is always used as the reference for bit sample comparisons. This anchor enables the barrel shifter to pivot its selection an equidistant ±80 bits from the zero skew center point of the 200-bit data channel shift register when identifying and compensating for skew. Figure 9 illustrates barrel shifter selections for skew compensation on two arbitrary data channels:

- Data channel i, which leads the deskew channel by 31 bits
- Data channel j, which lags the deskew channel by 80 bits

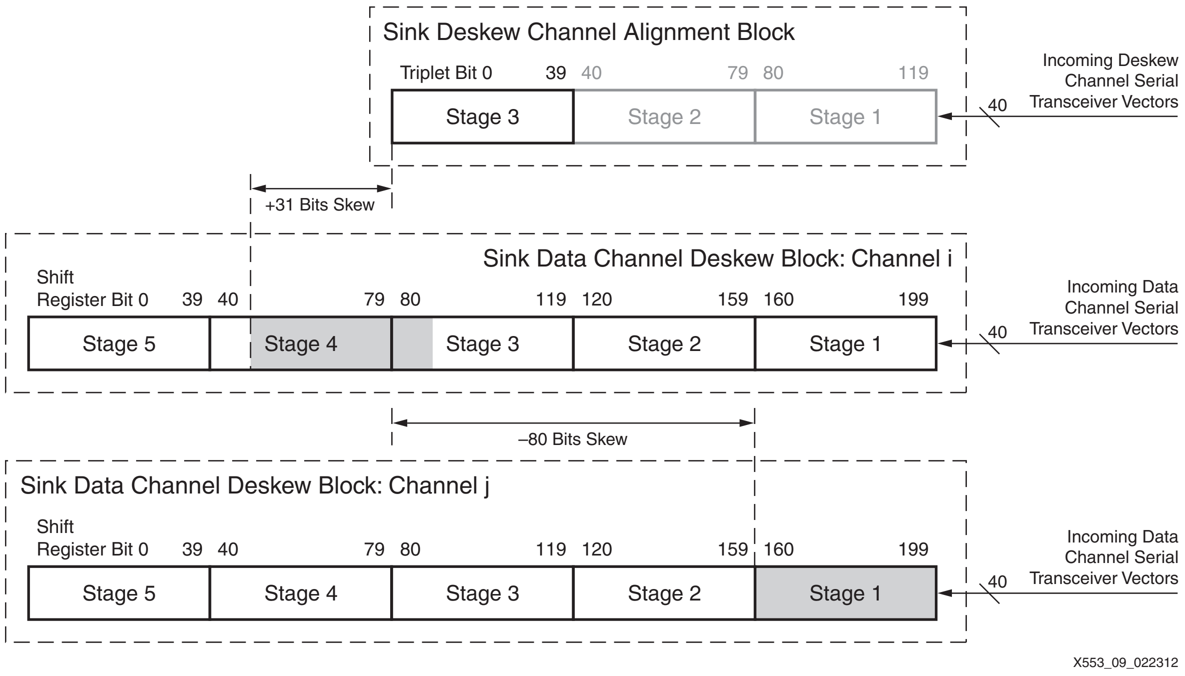
Sink Deskew Channel Alignment Block
Triplet Bit 0
39
40
79
80
119
Stage 3
Stage 2
Stage 1
40
Incoming Deskew Channel Serial Transceiver Vectors
+31 Bits Skew
Sink Data Channel Deskew Block: Channel i
Shift Register Bit 0
39
40
79
80
119
120
159
160
199
Stage 5
Stage 4
Stage 3
Stage 2
Stage 1
40
Incoming Data Channel Serial Transceiver Vectors
–80 Bits Skew
Sink Data Channel Deskew Block: Channel j
Shift Register Bit 0
39
40
79
80
119
120
159
160
199
Stage 5
Stage 4
Stage 3
Stage 2
Stage 1
40
Incoming Data Channel Serial Transceiver Vectors
X553_09_022312

Figure 9: **Two Data Channels, with Barrel Shifters Selecting for +31 Bits and –80 Bits Skew Compensation**

The data channel deskew state machine controls the barrel shifter, moving its selection throughout the shift register search space until the bit sample comparators drive logic 1 for DATA_MATCH_CYC_TO_LOCK consecutive cycles. Stability indicates that the skew has been identified, so the state machine then transitions to the locked state and holds the barrel shifter selection constant to compensate for the skew on the channel.

In the locked state, the state machine has a configurable tolerance for bit sample comparator mismatches before losing lock. An error accumulator counts cycles containing one or more mismatches. Because only one in every fifteen bits is compared with a reference frame sample, the error accumulator is not a reliable indicator of data channel integrity. Rather, the error accumulator's purpose is to detect runtime changes in skew, which manifest as frequent bit sample mismatches. If the accumulator reaches the configurable value DATA_ERR_CYC_TO_UNLOCK, the state machine returns to the search state. The user can clear the accumulator at any time by asserting the data channel error accumulator reset input.

A change in skew leading to error accumulation is likely to be small and can occur in either the positive or negative direction. Therefore, to track skew as quickly as possible after loss of lock, the state machine adjusts the barrel shifter selection outwards from its prior position by periodically incrementing a magnitude counter while toggling a sign bit, progressing the selection per Equation 3:

$$selection = selection + (sign \times magnitude) \qquad \text{Equation 3}$$

where:

sign = {-1, 1}

magnitude = 1, 2, 3, …

Meanwhile, the comparators monitor stability as previously described. Figure 10 illustrates an example of the outward progression after the state machine loses lock while having compensated for +5 bits of skew.

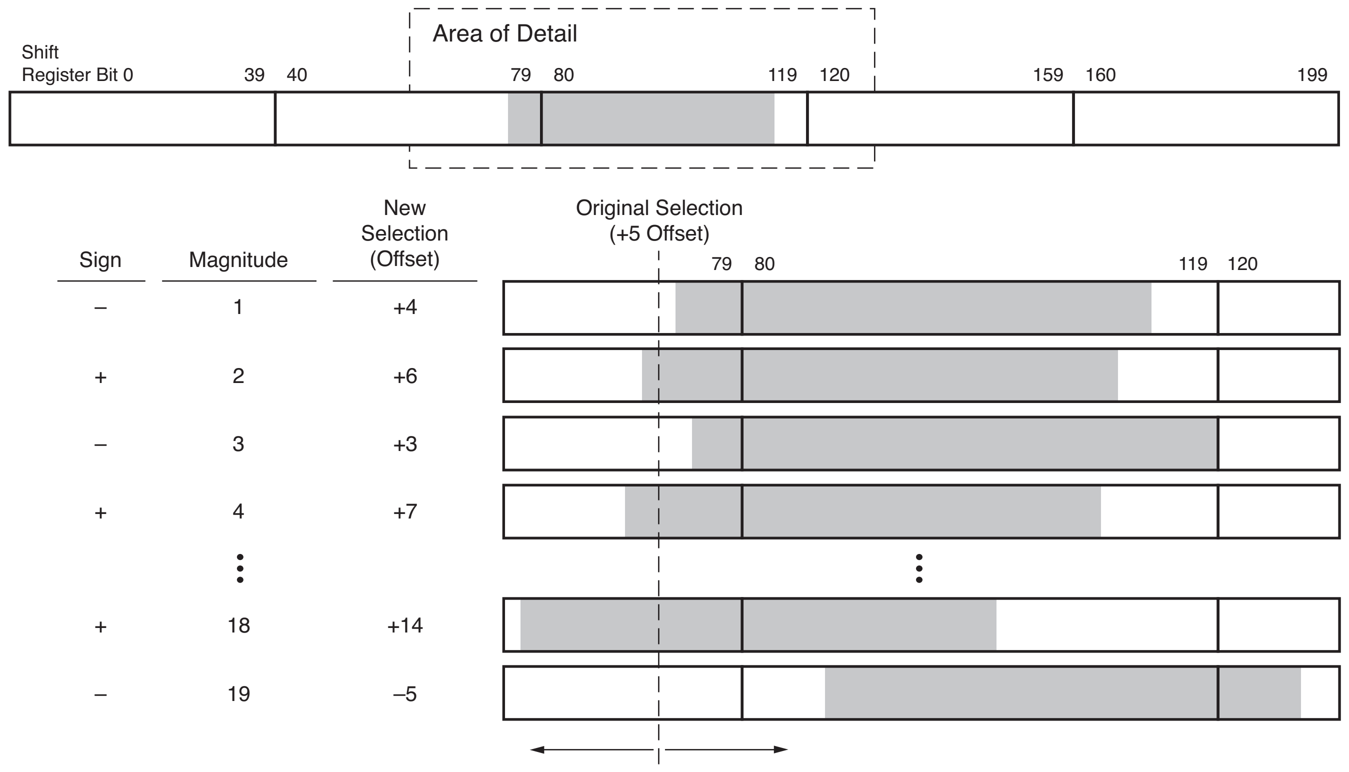

Figure 10: **Outward Progression of Barrel Shifter Selection During Skew Compensation Search**

If the state machine does not achieve lock before the magnitude exceeds 161 (thus guaranteeing that the search space is exhausted), the search restarts at the barrel shifter's reset position: the middle 40 bits of the shift register. In practice, the state machine most likely achieves lock rapidly following a change in skew. Figure 11 shows a simplified representation of the state machine.

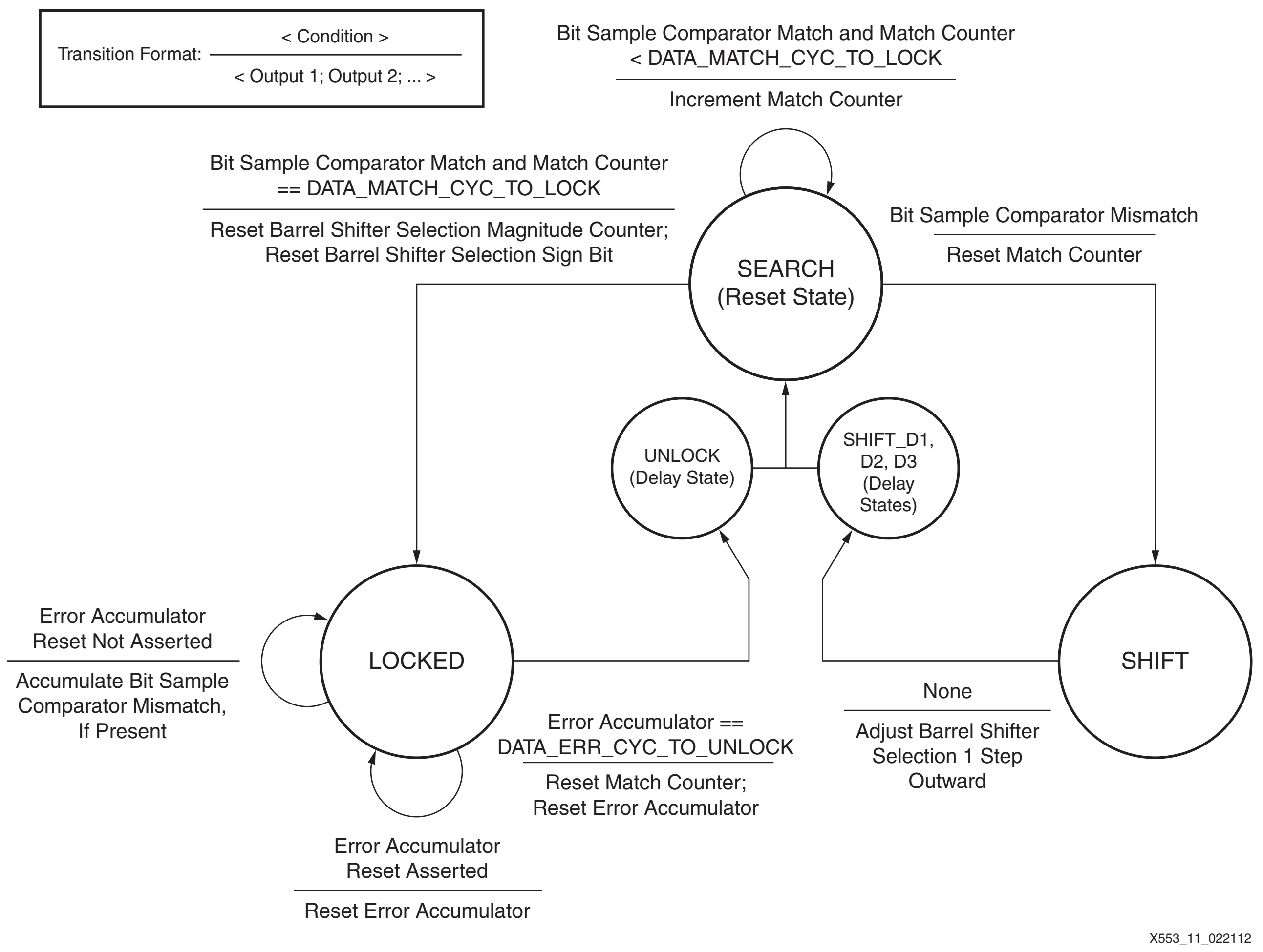
Transition Format:
< Condition >
< Output 1; Output 2; ... >
Bit Sample Comparator Match and Match Counter < DATA_MATCH_CYC_TO_LOCK
Increment Match Counter
Bit Sample Comparator Match and Match Counter == DATA_MATCH_CYC_TO_LOCK
Reset Barrel Shifter Selection Magnitude Counter; Reset Barrel Shifter Selection Sign Bit
Bit Sample Comparator Mismatch
Reset Match Counter
SEARCH (Reset State)
UNLOCK (Delay State)
SHIFT_D1, D2, D3 (Delay States)
LOCKED
SHIFT
Error Accumulator Reset Not Asserted
Accumulate Bit Sample Comparator Mismatch, If Present
Error Accumulator == DATA_ERR_CYC_TO_UNLOCK
Reset Match Counter; Reset Error Accumulator
None
Adjust Barrel Shifter Selection 1 Step Outward
Error Accumulator Reset Asserted
Reset Error Accumulator
X553_11_022112

Figure 11: **Simplified Data Channel Deskew State Machine**

System requirements and characteristics should be considered when determining appropriate values for the DATA_MATCH_CYC_TO_LOCK and DATA_ERR_CYC_TO_UNLOCK parameters and the frequency of data channel error accumulator reset assertions. Increasing the value of DATA_MATCH_CYC_TO_LOCK logarithmically decreases the probability of locking onto an erroneous selection for skew compensation at the expense of additional lock time. The value of DATA_ERR_CYC_TO_UNLOCK and the frequency of accumulator reset assertions together control error tolerance and might be useful in distinguishing bit errors from skew changes. For example, a small parameter value with infrequent accumulator resets implies a tolerance for occasional bit errors but a quick reaction to skew change. As a group, the two parameter values and the frequency of accumulator reset assertions characterize a sensitivity to skew change. The tradeoff between rapid skew tracking and tolerance for bit errors should be carefully considered.

To assist with the selection of parameter values and the frequency of accumulator reset assertions, three interface characteristics and their approximated theoretical values are shown in Table 4. This list is not complete, and more precise calculations are possible given specific system characteristics such as actual voltage, temperature, and bit error behavior.

Table 4: **Interface Characteristics and Their Approximated Theoretical Values**

Interface Characteristic	Approximated Theoretical Value
Frequency of accumulator reset assertion to mitigate nominal data channel bit error rate *BER*	Assert accumulator reset for one clock cycle in every *x* sink clock cycles, where: $x = \frac{1}{40 \times BER}$ *Equation 4*
Following discrete skew change, the average number of clock cycles to lose lock	$3 + \text{DATA\_ERR\_CYC\_TO\_UNLOCK} \times \left[1 + \sum_{i=1,2,3\ldots}\left(0.5^{\frac{8}{3}i} \times i\right)\right]$ $\approx 3 + (1.222 \times \text{DATA\_ERR\_CYC\_TO\_UNLOCK})$ sink clock cycles *Equation 5*
Following discrete skew change of ±1 UI and subsequent loss of lock, the average number of clock cycles to regain lock(1)	$8.5 + \text{DATA\_MATCH\_CYC\_TO\_LOCK} + \left\{1.5 \times \left[5 + \sum_{i=1,2,3\ldots}\left(0.5^{\frac{8}{3}i} \times i\right)\right]\right\}$ $\approx 16.333 + \text{DATA\_MATCH\_CYC\_TO\_LOCK}$ sink clock cycles *Equation 6*

Notes:
1. Assumes that skew change and its effects at the FPGA logic boundary have settled by the time lock is lost.

When the data channel deskew state machine is locked, that instance's locked status output is asserted. The value of the SFI-S RXS signal is the NAND of all channels' lock status outputs.

Skew Compensation Capability

The 200-bit shift register search space enables each data channel to be independently compensated for up to ±80 bits of skew with respect to the deskew channel. However, additional factors affect the overall skew compensation range of the SFI-S sink interface. The factors and their effects are described individually.

The ability to deskew the interface depends on the worst-case data channel skew with respect to the deskew channel as observed at the sink-side FPGA logic boundary. Let s_p be the worst-case positive skew and s_n be the worst-case negative skew, in bits. Skew compensation is possible if $s_p \leq 80$ bits and $s_n \geq -80$ bits, even if $s_p - s_n$ exceeds 80 bits. Figure 12 illustrates the barrel shifter selection in each of the ten data channel deskew blocks for an example case where data channel 5 exhibits $s_p = 31$ bits and data channel 0 exhibits $s_n = -80$ bits. Although the overall skew of the interface at the FPGA logic boundary is $s_p - s_n = 111$ bits, skew compensation is successful because both s_p and s_n are within the operable range.

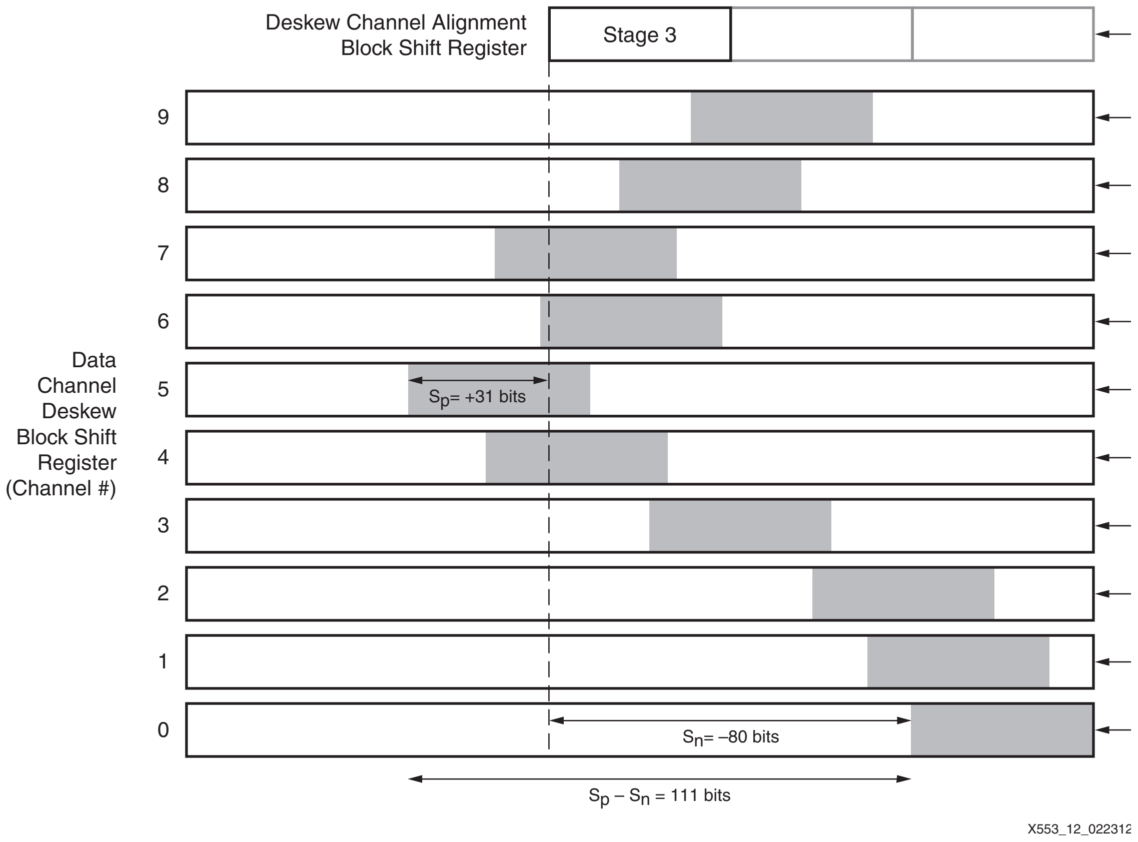
Deskew Channel Alignment Block Shift Register
Stage 3
Data Channel Deskew Block Shift Register (Channel #)
9
8
7
6
5
4
3
2
1
0
Sp= +31 bits
Sn= –80 bits
Sp – Sn = 111 bits
X553_12_022312

Figure 12: **Successful Skew Compensation Where S_p = 31 Bits and S_n = –80 Bits**

As described in Transceivers Block: RX Path, page 11, the deskew channel and each data channel use a serial transceiver. The CDR circuit within each serial transceiver extracts the recovered clock and data from the received SFI-S signal. Using the recovered clock, the recovered data is written into a SIPO structure from which completed 40-bit vectors are read. Because each CDR operates independently, the time after reset at which each SIPO is first written is not necessarily the same from channel to channel. The 40-bit vectors from any two serial transceivers therefore exhibit an unknown but constant offset between 0 and 39 bits.

The offset is effectively the skew at the FPGA logic boundary, for which the barrel shifters can be observed to compensate, even if there is no skew at the device pins. Because the offset for each channel is unknown, the net effect might be a reduction in skew at the FPGA logic boundary as compared to the device pins, a similar increase in skew, or no change in skew. The ability to deskew the interface is therefore a probabilistic function of skew as observed at the device pins. Let S be the worst-case data channel skew with respect to the deskew channel, in UI. Figure 13 shows the probability $p_d(S)$ of successful interface skew compensation as a function of S. The range from –41 UI to +41 UI is where $p_d(S)$ = 1 and is shaded to indicate reliable interface deskew capability.

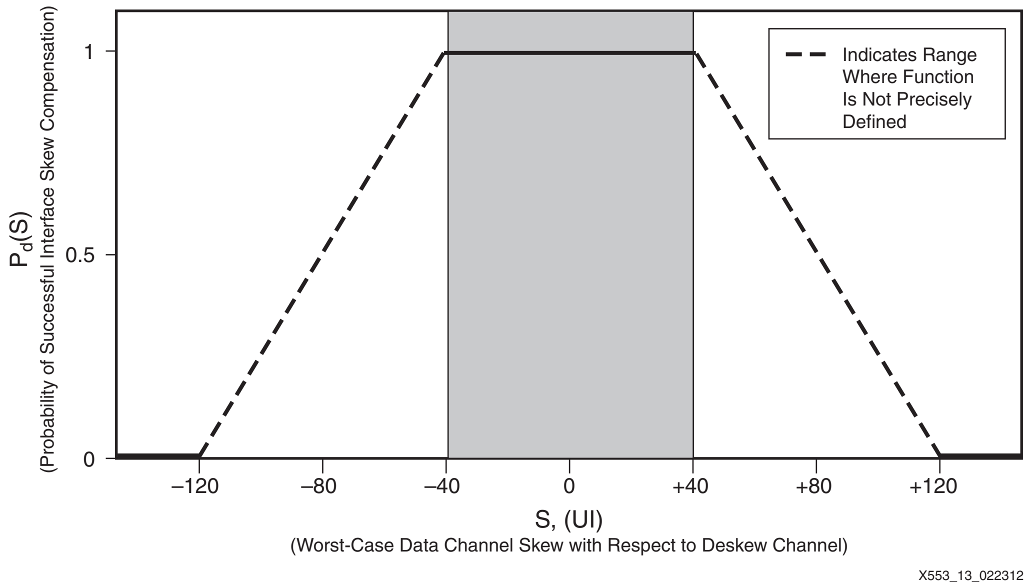

Figure 13: **Probability of Successful Interface Skew Compensation as a Function of Worst-Case Skew**

If a larger skew compensation range is required, it can be added to the design at the expense of an approximately linear growth in FPGA logic resources. To maintain a symmetrical search space around the aligned deskew reference frame, the data channel shift register should be expanded in two-stage, 80-bit increments, while the deskew channel shift register should be expanded in one-stage, 40-bit increments that always lead the triplet. The data channel barrel shifter is implemented using a multi-level reduction structure of multiplexers. The barrel shifter and its associated state machine components such as the magnitude counter must also be updated to fully traverse the expanded search space.

Sink Block

When a data channel deskew block has compensated for that channel's skew with respect to the deskew channel, its state machine is locked and its lock status output is asserted. When the lock status output is asserted for all data channels, the SFI-S RXS signal drives logic 0 to indicate that the sink interface is fully aligned. While RXS is only defined in the optics-to-system (receive) direction, it might be useful in both system directions and is always provided as an output of the sink block.

The final function of the sink interface is to de-stripe the skew-compensated 40-bit vector outputs of the ten individual data channel deskew blocks into a single, 400-bit user data vector. Since the data channel deskew block maintains the bit ordering provided by the serial transceiver—where the LSB is the first bit received—a Verilog generate statement implements the de-striping function as a round-robin mapping onto the user data vector. As shown in Table 5, the oldest skew-compensated bit received on data channel 9 maps to the user data vector LSB (bit 0) while the most recent skew-compensated bit received on data channel 0 maps to the MSB (bit 399). The user data vector is registered once for timing isolation, making available a new 400-bit output synchronous to each rising edge of the sink clock. The user data vector contents are skew-compensated when the value of the RXS output is logic 0.

Table 5: **De-Striped Mapping of Skew-Compensated Data Channels onto the User Data Vector**

Skew-Compensated Data Channel Number, Vector Bit Number	User Data Vector Bit Number
Data channel 9, bit 0	0
Data channel 8, bit 0	1
Data channel 7, bit 0	2
Data channel 6, bit 0	3
Data channel 5, bit 0	4
Data channel 4, bit 0	5
Data channel 3, bit 0	6
Data channel 2, bit 0	7
Data channel 1, bit 0	8
Data channel 0, bit 0	9
Data channel 9, bit 1	10
Data channel 8, bit 1	11
⋮	⋮
Data channel 1, bit 39	398
Data channel 0, bit 39	399

Simulation and Hardware Demonstration

Included with the SFI-S reference design is a synthesizable Verilog HDL example design intended for both simulation and hardware demonstration of the bidirectional top-level module. A simple test bench with SFI-S interface loopback and skew injection and accompanying simulation scripts demonstrate basic design operation and skew compensation. Design constraints and implementation scripts support hardware demonstration by implementing the SFI-S example design from synthesis through bitstream generation.

The SFI-S example design consists of:

- The reference design top-level module as described in Reference Design Overview, page 3
- A PRBS31 parallel data generator module that drives the source interface user data input port
- A PRBS31 parallel data checker module driven by the sink interface user data output port
- A resettable, latched PRBS31 checker error indicator.

The generator and checker modules are configurations of the macro described in *An Attribute-Programmable PRBS Generator and Checker* [Ref 4]. By both generating and checking for the specified PRBS31 sequence, the example design serves as a simulation and hardware demonstration of the reference design, and provides the test features recommended in the OIF-SFI-S-01.0 Implementation Agreement [Ref 1]. (The example design provides complete link generation and checking, and each serial transceiver can be configured for per-channel functionality).

As shown in Figure 14, the example design provides a simple mechanism to monitor received PRBS31 patterns for errors while significantly reducing the user I/O. The presence of one or more errors is indicated by a non-zero output of the PRBS31 checker. The EXAMPLE_PRBS_MATCH user output provides the current checker status synchronous to the rising edge of the sink clock, where logic 1 indicates a match. The

EXAMPLE_PRBS_ERROR_LATCHED user output is a sticky error indicator set to logic 1 by the presence of a pattern mismatch and reset to logic 0 only upon the active-High assertion of the EXAMPLE_PRBS_ERROR_LATCHED_RESET asynchronous user input. The current status output can be useful for link quality analysis, while the latched error indicator and reset are intended for basic human interaction.

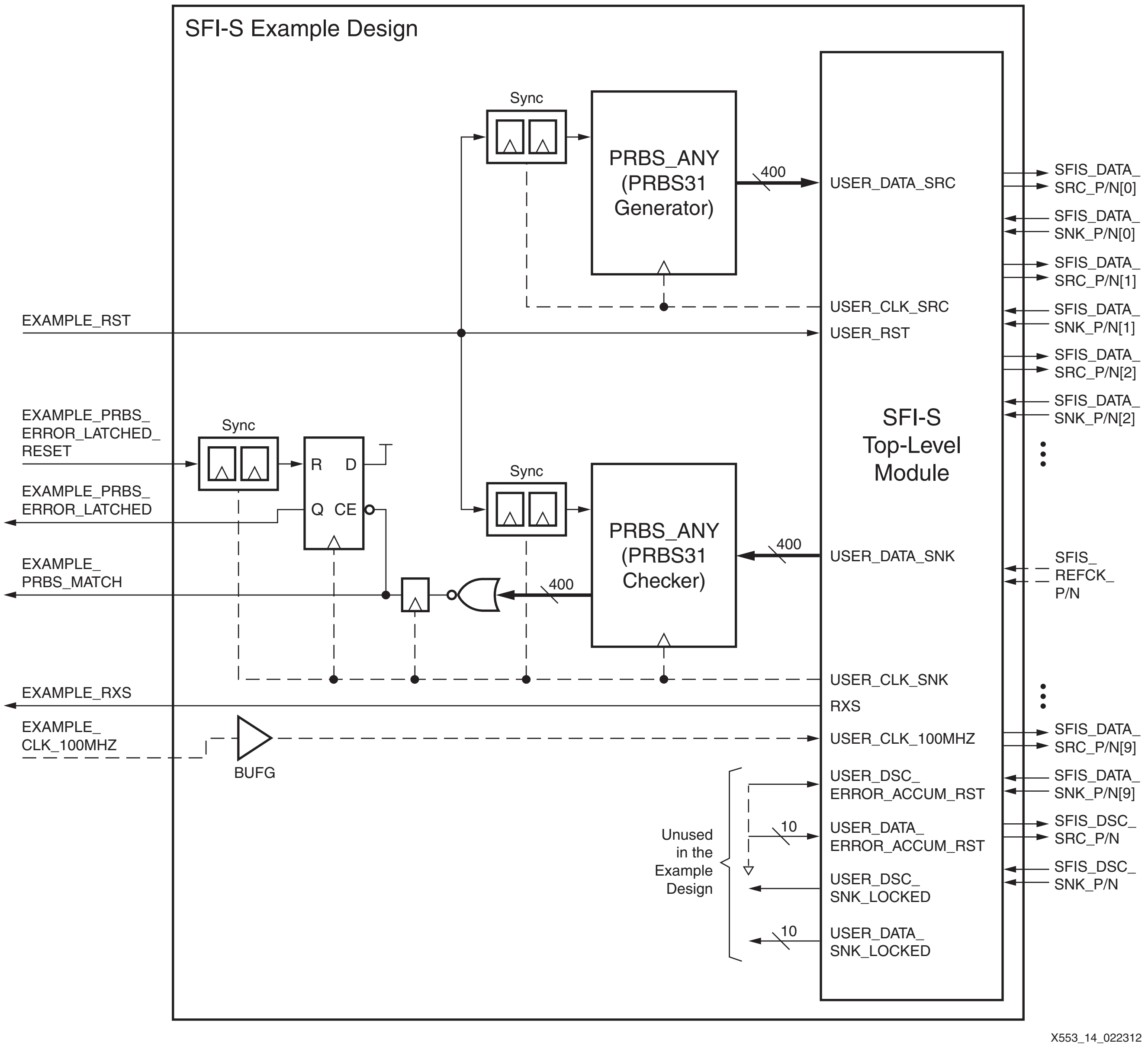

Figure 14: **SFI-S Example Design Block Diagram**

As with the reference design top-level module, a free-running 100 MHz system clock and a master asynchronous reset pulse initialize the example design. PRBS31 checker errors persist throughout bring-up and the initial skew compensation procedure, generally subsiding when interface alignment is achieved, as indicated by the RXS output transitioning to a stable logic 0. Because errors are present until interface alignment is achieved, the latched error reset input should be toggled after that event.

Simulating the Example Design

A Verilog HDL demonstration test bench and accompanying scripts facilitate simulation of the SFI-S example design. The simulation-only test bench module provides the SFI-S reference clock, free-running 100 MHz clock, and master reset stimulus to bring up the example design. The SFI-S source interface differential outputs are clocked into a shift register representing a delay line. Parameterized tap values incrementally add effective data channel skew, up to ±40 UI with respect to the deskew channel, before being looped back onto the SFI-S sink interface differential inputs.

Procedural Verilog code monitors the RXS output for a falling edge and toggles the latched error reset input. The simulation then runs for an additional 20 μs before stopping, providing the user with a waveform view of useful signal behavior throughout design bring-up, skew compensation, and finally PRBS31 checker success.

SFI-S example design simulation has been tested with Mentor Graphics ModelSim v6.6d and ISE® Design Suite 13.4 simulation models. To run the simulation, the user must have the Xilinx simulation libraries compiled for the system as described in *Synthesis and Simulation Design Guide* [Ref 5]. The user starts ModelSim from the unzipped reference design root directory and enters these commands:

```
ModelSim> cd example
ModelSim> do sfis_example_tb_sim.do
```

The simulation begins, and useful signals are displayed in a waveform window, as shown in Figure 15.

Note: Even with the SIMULATION_SPEEDUP reference design parameter set to **1**, the serial transceiver bring-up process takes a significant amount of time, and RXS might not transition to a stable logic 0 until as late as 40 μs.

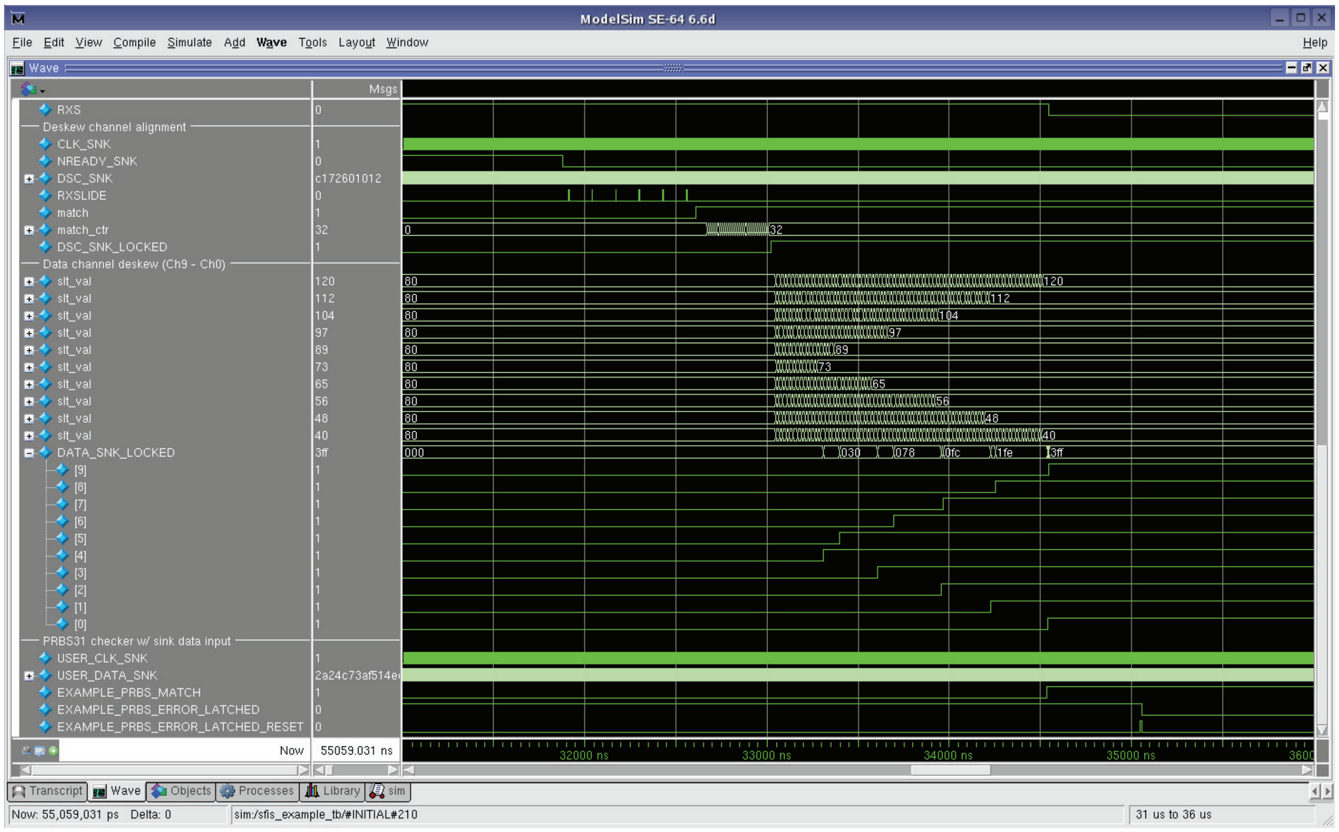

Figure 15: **SFI-S Example Design Test Bench Simulation Waveform**

Implementing the Example Design

A user constraints (`.ucf`) file and accompanying scripts facilitate implementation of the SFI-S example design for hardware operation. By targeting a platform that provides the required stimulus and connectivity, the implemented example design can be a useful demonstration and characterization tool. For example, the SFI-S interface can be physically looped back from source to sink with appropriate cabling, while the sticky PRBS31 error indicator and its reset are used to demonstrate basic link integrity. Alternatively, two devices can be configured for bidirectional, point-to-point connectivity while the PRBS31 current status indicator is continually monitored by additional logic for link characterization. As described in Hardware Verification, page 25, the implemented example design was used as the basis for reference design verification.

The user constraints file contains appropriate period constraints and by default targets the XC7VX485T device (for GTX serial transceiver use) or XC7VX690T device (for GTH serial transceiver use) in the FFG1761 package with -3 speed grade. The device choice and placement can be readily changed to accommodate the user's hardware platform, but the SERIAL_TRANSCEIVER_TYPE reference design parameter value must correspond to the serial transceiver type physically present in the chosen device.

SFI-S example design implementation has been tested with the ISE Design Suite 13.4. To implement the design, the user enters the following commands from the unzipped reference design root directory:

For Linux, when SERIAL_TRANSCEIVER_TYPE is "7SERIES_GTX":

```
% cd example
% ./sfis_example_gtx_implement.sh
```

For Windows, when SERIAL_TRANSCEIVER_TYPE is "7SERIES_GTX":

```
> cd example
> sfis_example_gtx_implement.bat
```

For Linux, when SERIAL_TRANSCEIVER_TYPE is "7SERIES_GTH":

```
% cd example
% ./sfis_example_gth_implement.sh
```

For Windows, when SERIAL_TRANSCEIVER_TYPE is "7SERIES_GTH":

```
> cd example
> sfis_example_gth_implement.bat
```

The script processes the design through XST synthesis and the Xilinx implementation tool flow, producing a downloadable bitstream file and various report files.

Hardware Verification

To verify that the SFI-S reference design operates as intended, hardware testing was performed on an implementation of the example design at the full 11.18 Gb/s per channel line rate. Test results indicated successful operation.

The hardware platform consisted primarily of an XC7VX485T Engineering Sample (ES) device on a Xilinx VC7203 board with Samtec BullsEye assemblies providing coaxial cable access to each SFI-S bus signal. Data and deskew channel cables were looped back from source to sink.

The provided example design was enhanced with various ChipScope™ tool virtual input/output (VIO) and integrated logic analyzer (ILA) cores to monitor key signals such as RXS, the current PRBS31 checker status, and the data channel deskew barrel shifter selection for each channel. Pushbuttons and LEDs on the VC7203 board were used for I/O such as the master reset, as well as to monitor and reset the latched PRBS31 checker error indicator.

In a bring-up test consisting of 250 master resets with occasional power cycling and device reprogramming, proper bring-up and skew compensation was achieved in every case. PRBS31 checker errors were observed to cease after interface alignment.

In a bit error rate test, no bit errors were observed during a period where 10^{15} data bits were transferred across the externally looped-back SFI-S interface. The example design PRBS31 generator and checker were used to generate and check the data, respectively.

In an interface characteristics test where discrete skew changes of +1 UI and –1 UI were repeatedly forced, data channel deskew state machines were observed to unlock, and then regain lock at the new alignment with durations that correlate well with the approximated theoretical values shown in Table 4.

In a skew compensation test, the settled data channel deskew barrel shifter selection was recorded for each channel after each of several master resets. When all channels were looped back using only the matched-length BullsEye coaxial cables, very little deviation was observed, either between lanes or between resets. In one variation on the test, the sequence was repeated after the addition of 24-inch coaxial cables to the differential signals of data channel 5. The data channel 5 barrel shifter selection was observed to adjust by an average of +33 taps, correlating well with the coaxial cable theoretical propagation delay.

Figure 16 shows the barrel shifter selections providing skew compensation on each of the ten data channels before and after the addition of the 24-inch cable on channel 5. The average 40-bit data vector selection is represented by the bar, with observed minimum and maximum selections extending vertically. The limited variation indicates that serial transceiver bring-up uncertainty did not substantially reduce the available skew compensation margin in this case.

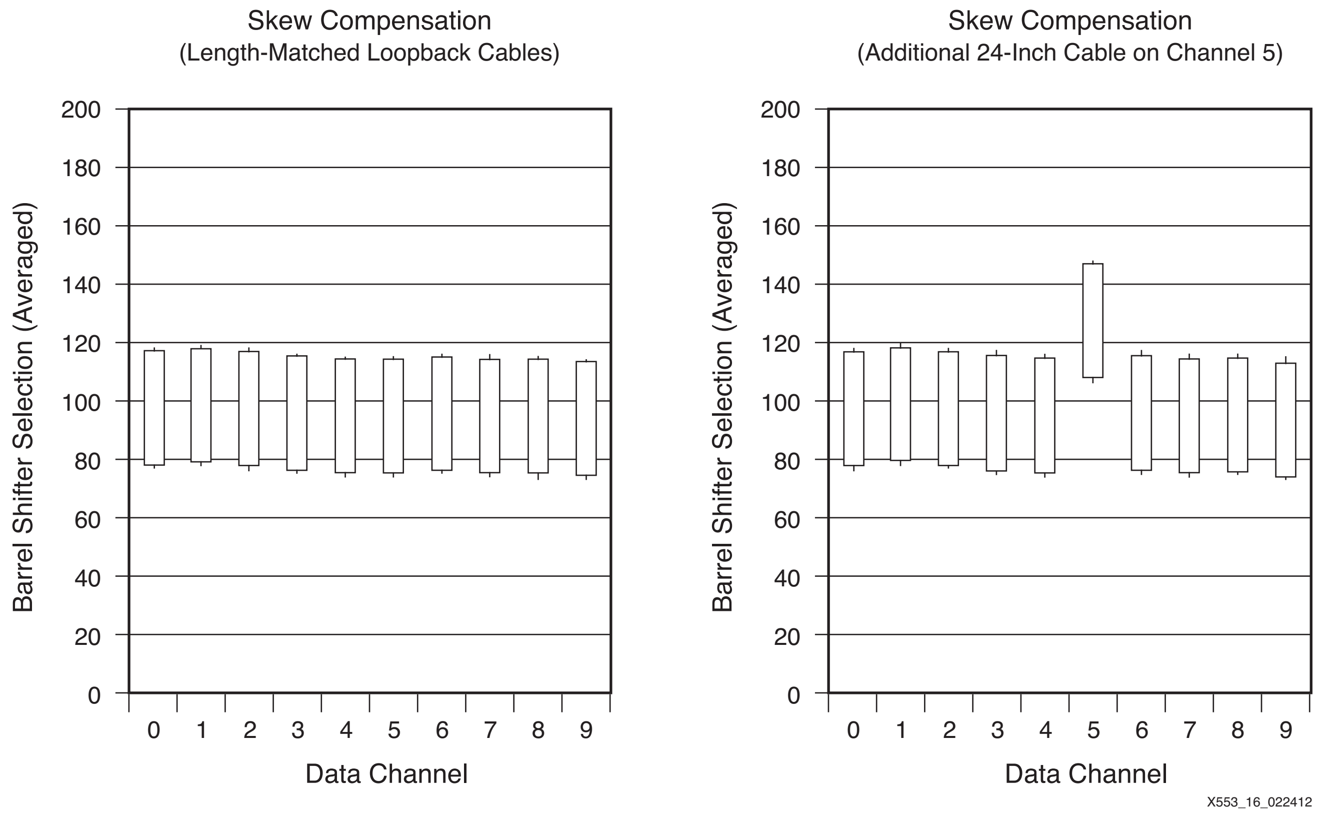

Figure 16: **Observed Skew Compensation, Before and After 24-Inch Coaxial Cable Addition to Channel 5**

Reference Design

The SFI-S reference design described in this application note can be downloaded from www.xilinx.com/member/sfis_ref_des/index.htm

As of this release, Kintex-7 and Virtex-7 T device support is preliminary, and hardware verification has been performed on ES silicon. The GTX serial transceiver instantiation wrappers were generated using a pre-production version of the Xilinx 7 series FPGAs Transceivers Wizard. Settings are subject to change, based on characterization of production devices.

As of this release, XT device support is preliminary, and hardware verification has not been performed. The GTH serial transceiver instantiation wrappers were generated using a beta version of the Xilinx 7 series FPGAs Transceivers Wizard. Settings are expected to change.

The approximate FPGA resource usage of the SFI-S reference design is shown in Table 6.

Table 6: **SFI-S Reference Design Approximate Resource Usage**

Functional Block		LUTs	Registers	BUFGs	Serial Transceivers
Transceivers block	With GTX serial transceivers (Kintex-7 and Virtex-7 T devices)	320	387	4	11 GTX
	With GTH serial transceivers (Virtex-7 XT devices)	320	387	4	11 GTH
Source block		369	540	0	0
Sink block	Deskew channel alignment block	103	186	0	0
	Data channel deskew block (for each of 10 instances)	307	334	0	0
	Sink block total, including submodules	3230	3919	0	0
TOTAL for SFI-S reference design top-level module, including submodules		3914	4850	4(1)	11

Notes:

1. Excludes buffer for the 100 MHz interface bring-up clock, which can be shared with other logic.

The contents of the reference design archive, with a short description of each item, are listed in Table 7.

Table 7: **SFI-S Reference Design File Listing and Descriptions**

Directory	File	Description
`example`	**Files for implementation and simulation of the SFI-S example design that demonstrates the reference design**	
	`prbs_any.v`	Configurable PRBS pattern generator/checker module
	`sfis_example.v`	Example design to demonstrate SFI-S reference design
	`sfis_example_gt[h|x].ucf`	Example design user constraints file for implementation
	`sfis_example_gt[h|x]_implement.bat`	Example design implementation script for Windows installations of ISE design tools
	`sfis_example_gt[h|x]_implement.sh`	Example design implementation script for Linux installations of ISE design tools
	`sfis_example_gt[h|x]_xst.prj`	Example design project file for XST synthesis tool
	`sfis_example_gt[h|x]_xst.scr`	Example design script file for XST synthesis tool
	`sfis_example_tb.v`	Loopback test bench for the example design
	`sfis_example_tb_sim.do`	Example design functional simulation script for ModelSim

Table 7: **SFI-S Reference Design File Listing and Descriptions *(Cont'd)***

Directory	File	Description
gtwizard	**Files from the LogiCORE IP Xilinx 7 series FPGAs Transceivers Wizard for use in the SFI-S reference design**	
	gtwizard_gth_cei_sfis.v	Instantiation of all GTH serial transceiver wrappers and transceiver quads
	gtwizard_gth_cei_sfis.xco	CORE Generator™ tool core configuration file for GTH transceivers wizard
	gtwizard_gth_cei_sfis_gt.v	Wrapper for each GTH serial transceiver instance
	gtwizard_gtx_cei_sfis.v	Instantiation of all GTX serial transceivers wrappers and transceiver quads
	gtwizard_gtx_cei_sfis.xco	CORE Generator tool core configuration file for GTX transceivers wizard
	gtwizard_gtx_cei_sfis_gt.v	Wrapper for each GTX serial transceiver instance
hdl	**Verilog HDL files implementing the SFI-S reference design**	
	bit_synchronizer.v	Two flip-flop synchronizer for clock domain crossing
	clocking_reset.v	Clocking and reset resources for the reference design
	gt_rx_phdly_alignment.v	Implementation of the serial transceiver RX phase and delay alignment procedure
	gt_tx_phdly_alignment.v	Implementation of the serial transceiver TX phase and delay alignment procedure
	reset_synchronizer.v	Four flip-flop asynchronous reset synchronizer
	sfis_top.v	SFI-S reference design top-level module
	snk.v	Sink block
	snk_data_channel_deskew.v	Data channel deskew block
	snk_deskew_channel_alignment.v	Deskew channel alignment block
	src.v	Source block
	transceivers.v	Transceivers block

The SFI-S reference design matrix is shown in Table 8.

Table 8: **SFI-S Reference Design Matrix**

Parameter	Description
General	
Developer name	Julian Kain
Target devices	Kintex-7, Virtex-7 T, and Virtex-7 XT devices
Source code provided	Yes
Source code format	Verilog HDL
Design uses code/IP from existing Xilinx application note/reference designs, CORE Generator software, or third party	Yes • Transceivers block uses instantiation wrappers generated by the LogiCORE IP Xilinx 7 series FPGAs Transceivers Wizard. • Provided example design uses PRBS generator and checker from *An Attribute-Programmable PRBS Generator and Checker* [Ref 4].

Table 8: **SFI-S Reference Design Matrix *(Cont'd)***

Parameter	Description
Simulation	
Functional simulation performed	Yes
Timing simulation performed	No
Test bench used for functional and timing simulations	Yes (functional simulations)
Test bench format	Verilog HDL
Simulator software/version used	ModelSim v6.6d
SPICE/IBIS simulations	No
Implementation	
Synthesis software tools/version used	XST tool v13.4
Implementation software tools/version used	ISE tool v13.4
Static timing analysis performed	Yes
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	Xilinx VC7203 board

Conclusion

This application note describes a ten data channel SFI-S design targeting Xilinx 7 series FPGAs using GTX or GTH serial transceivers to implement an aggregate 111.8 Gb/s bidirectional interface. The hardware-verified reference design provides significant skew compensation and fine-grained control of skew tracking, and the Verilog HDL source can be readily modified if necessary. A synthesizable example design with PRBS31 generator and checker logic enables simple simulation and hardware demonstration of the reference design.

References

This application note references the following documents or sites:

1. Optical Internetworking Forum, ed. November 2008. *Scalable Serdes Framer Interface (SFI-S): Implementation Agreement for Interfaces beyond 40G for Physical Layer Devices*. IA# OIF-SFI-S-01.0.
   http://www.oiforum.com/public/documents/OIF_SFI-S_01.0_IA.pdf
2. Optical Internetworking Forum, ed. February 2005. *Common Electrical I/O (CEI) - Electrical and Jitter Interoperability agreements for 6G+ bps and 11G+ bps I/O.*
   IA# OIF-CEI-02.0
   http://www.oiforum.com/public/documents/OIF_CEI_02.0.pdf
3. UG476, *7 Series FPGAs GTX Transceivers User Guide*
4. XAPP884, *An Attribute-Programmable PRBS Generator and Checker*
5. UG626, *Synthesis and Simulation Design Guide*

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
03/02/2012	1.0	Initial Xilinx release.

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.