XAPP877 (v1.0) January 5, 2010

SerDes Framer Interface Level 4 Phase 2

Author: Martin Charron and Ousama Hage

Summary

This application note describes the implementation of SerDes Framer Interface Level 4 Phase 2 (SFI4.2) in a Virtex-5 FPGA XC5VFX70T. The SFI4.2 standard is defined by the Optical Internetworking Forum (OIF) [Ref 1]. The OIF standard only specifies a 10 Gb/s interface. However, multiple interfaces can be concatenated together to create higher bandwidth interfaces such as 50 Gb/s and 100 Gb/s. This application note describes the implementation of a 10 Gb/s interface. This interface consists of four bidirectional GTX transceivers and logic to compensate for the ±32 UI skew differences between the data channels, as specified by the OIF standard. This reference design can switch to a non-standard mode that supports ±256 UI skew differences between the data channels.

Introduction

The SFI4.2 system reference model (as defined by the OIF standard) is shown in Figure 1. SFI4.2 is intended to interface between a SerDes component and a Forward Error Correction (FEC) processor, between an FEC processor and a framer, or directly between a SerDes component and FEC processor. The reference model has four data channels that can each operate up to 3.125 Gb/s. The OIF standard also specifies a Transmit Clock Source (TXCKSRC), which is frequency locked to the sink device. This clock is not part of the SFI4.2 implementation, but can be handled from the board level. Refer to Clocking an SFI4.2 Interface, page 8 for more details.

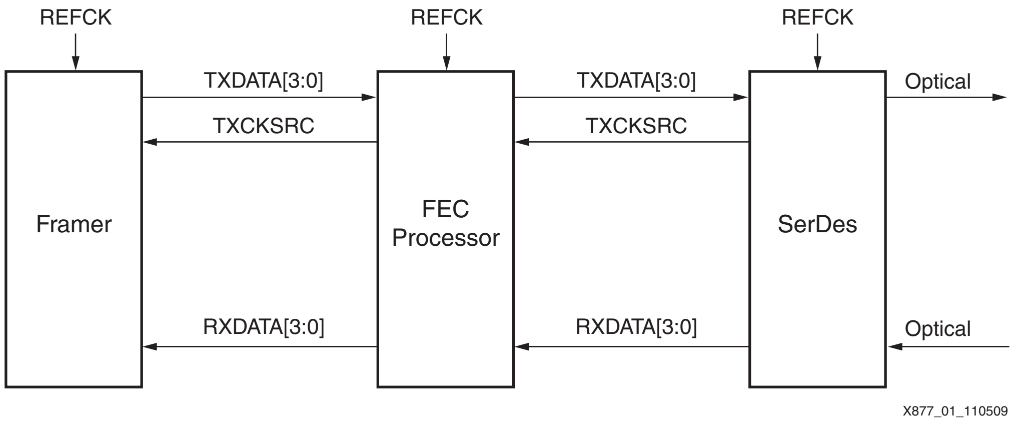


Figure 1: **SFI4.2 System Reference Model Defined by OIF**

The SFI4.2 reference model has these properties:

- The RX and TX interfaces operate at the same frequency up to 200 ppm, as defined by the RXPPMTOL specified in the *Virtex-5 FPGA Data Sheet: DC and Switching Characteristics* [Ref 2].
- TX clocking is synchronized to REFCK.
- RX is clocked off the recovered clock.

As shown in Figure 2, the incoming data is scrambled and then split into four lanes in a round robin fashion. As defined by the OIF specification, the data is sent out with 16 bits of offset between each of the lanes.

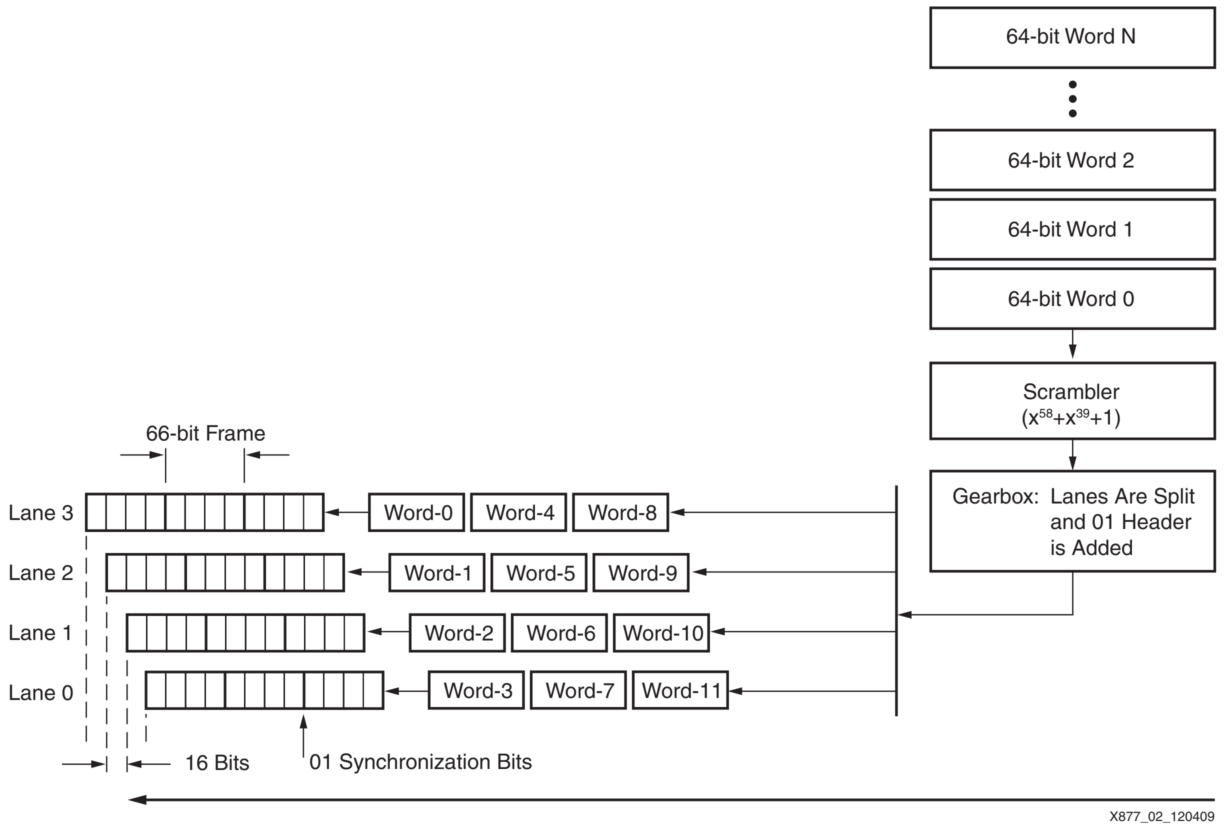


Figure 2: **SFI4.2 Transmitter Model**

As shown in Figure 3, the receive data can be skewed by ±32 UI or ±256 UI, depending on the mode of operation. The deskew algorithm realigns the data, and four independent streams are merged into one before being descrambled.

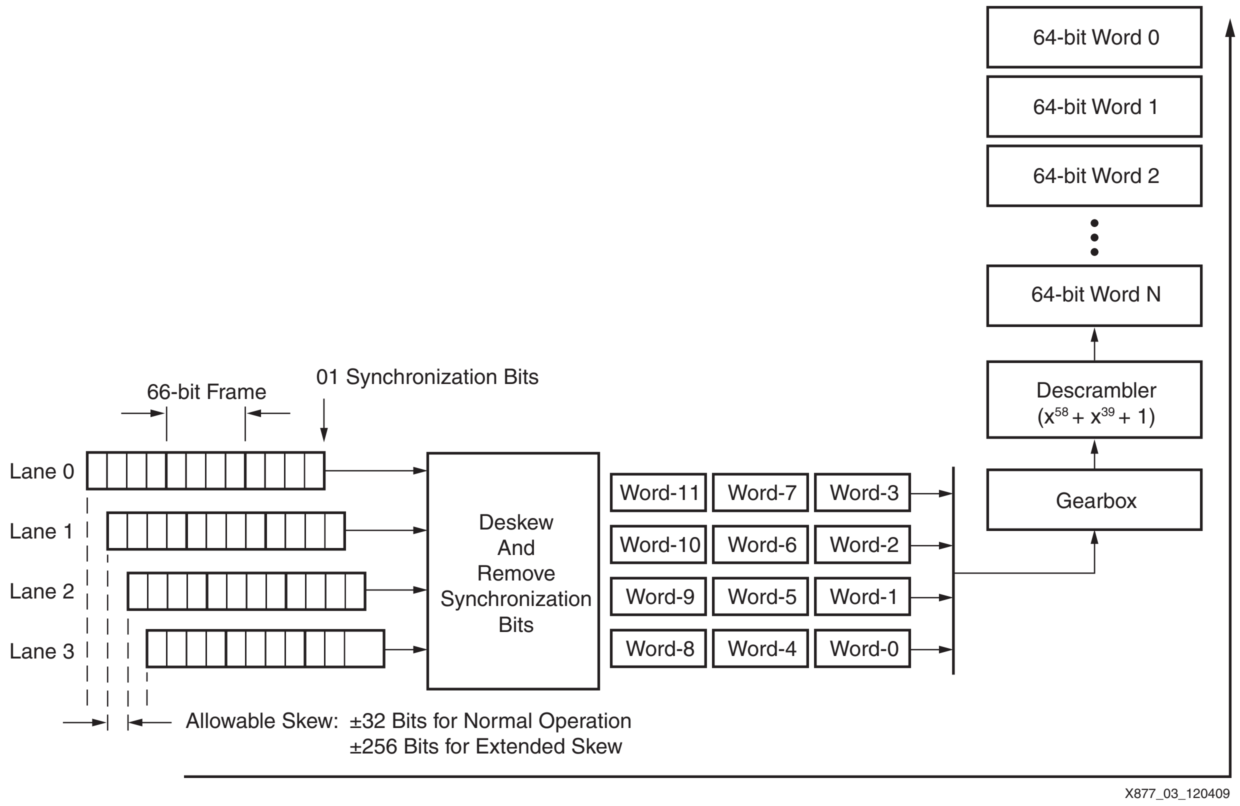


Figure 3: **SFI4.2 Receiver Model**

Port List and Descriptions

All signals in Table 1 are active-High unless stated otherwise. Optional settings must be set to either user-specific values or to the default values given in this table. The signals in Table 1 should not be left unconnected.

Table 1: **Port List and Description for SFI4.2 Interface**

Port	Type	Width	Clock Domain	Description
SFI4.2 TX Interface Signals				
ov_TXP	Output	4	Line rate	These are the SFI4.2 TX data channels (P-side).
ov_TXN	Output	4	Line rate	These are the SFI4.2 TX data channels (N-side).
SFI4.2 RX Interface Signals				
iv_RXP	Input	4	Line rate	These are the SFI4.2 RX data channels (P-side).
iv_RXN	Input	4	Line rate	These are the SFI4.2 RX data channels (N-side).
SFI4.2 Reference Clocks and Resets				
i_RST	Input	1	Async	This is the active-High Global asynchronous reset, synchronized internally.
i_MGT_REFCLK_P	Input	1	txrefclk	This is the reference clock to the GTX transceiver for the TX and RX. The frequency is 1/16 the line rate (e.g., 2.5–3.125 Gb/s → 156.25–195.31 MHz).
i_MGT_REFCLK_N	Input	1	txrefclk	
i_TRIGGER_RX_RESET	Input	1	Async	A rising-edge transition on this signal causes the entire receiver to be reset, including the GTX transceiver CDR and all logic. The RX reset does not affect the transmitter.

Table 1: **Port List and Description for SFI4.2 Interface *(Cont'd)***

Port	Type	Width	Clock Domain	Description
i_TRIGGER_TX_RESET	Input	1	Async	A rising-edge transition on this signal causes the transmitter to be reset, including the GTX transceiver. Because the entire GTX transceiver is reset, the receiver also resets.
i_CLK_SRC_RX_TX_N	Input	1	Async	This signal indicates the principle clock source. A 0 indicates that the system clock source can come from the reference clock. A 1 indicates that the system clock source comes from the recovered clock. Refer to Clocking an SFI4.2 Interface, page 8 for more details.
SFI4.2 TX Fabric Interface				
i_TX_CLK	Input	1	txcoreclk	This clock is used to clock the TX 64-bit side of the interface and is the 64/66 version of the MGT_REFCLK.
i_TX_SYNC_RST	Input	1	txcoreclk	This is the active-High synchronous reset for the txcoreclk clock domain.
iv_TX_DATAIN	Input	64	txcoreclk	This is the SFI4.2 transmit data from the FPGA logic.
SFI4.2 RX Fabric Interface				
o_RX_CLK	Output	1	rxcoreclk	This clock is used to clock the RX 64-bit side of the interface and is the 64/66 version of the RX recovered clock.
o_RX_SYNC_RST	Output	1	rxcoreclk	This is the active-High synchronous reset for the o_RX_CLK domain.
ov_RX_DATAOUT	Output	64	rxcoreclk	This is the SFI4.2 receive data.
DRP Access Port				
i_DRP_CLK	Input	1	drp_clk	This is the clock used for the DRP port. This clock must be connected for proper operation of the SFI4.2 interface. Refer to the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 3] for the valid frequency range for this clock.
i_DRP_RST	Input	1	drp_clk	This is the active-High synchronous reset for the drp_clk clock domain.
iv_DADDR_IN	Input	7	drp_clk	This is the DRP address bus.
iv_DI_IN	Input	16	drp_clk	This is the DRP write data bus.
i_DWE_EN	Input	1	drp_clk	This is the DRP write enable signal.
iv_DEN_IN	Input	2	drp_clk	This is the DRP enable signal for both GTX_DUAL tiles. Bit 0 is for GTX_DUAL 0, corresponding to lanes 0 and 1. Bit 1 is for GTX_DUAL 1, corresponding to lanes 2 and 3.
ov_DO_OUT0	Output	16	drp_clk	This is the DRP read data for GTX_DUAL_0.
ov_DO_OUT1	Output	16	drp_clk	This is the DRP read data for GTX_DUAL_1.
ov_DRDY_OUT	Output	2	drp_clk	This is the DRP ready signal. Bit 0 is for GTX_DUAL 0. Bit 1 is for GTX_DUAL 1.
MGT Control and Status Signals				
iv_MGT_LOOPBACK	Input	3	Async	This signal is connected to the LOOPBACK ports of the four SerDes. Refer to *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* for loopback details. The recommended default for this signal is `000`.

Table 1: **Port List and Description for SFI4.2 Interface *(Cont'd)***

Port	Type	Width	Clock Domain	Description
iv_RXEQMIXIN	Input	2	Async	This signal is connected to the RXEQMIX ports of the four SerDes. Refer to the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* for RXEQMIX details. The recommended default for this signal is `10`.
iv_TXDIFFCTRL	Input	3	Async	This signal is connected to the TXDIFFCTRL ports of the four SerDes. Refer to the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* for loopback details. The recommended default for this signal is `010`.
iv_TXPREEMPHASIS	Input	3	Async	This signal is connected to the TXPREEMPHASIS ports of the four SerDes. Refer to the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* for loopback details. The recommended default for this signal is `100`.
ov_MGT_PLL_LOCK	Output	2	Async	This signal is connected to the PLLLKDET signal of each GTX_DUAL tile.
ov_MGT_RESET_DONE	Output	4	Async	This signal is connected to the RESETDONE signals of each GTX_DUAL tile.
ov_MGT_CDR_STABLE	Output	4	Async	This signal indicates that the CDR of each receiver is within ±5000 ppm of the reference clock.
SFI4.2 RX Interface Control and Status				
i_EXT_SKEW_EN	Input	1	Async	A 1 on this signal puts the SFI4.2 interface into Extended-Skew mode, as opposed to Normal mode.
i_BYPASS_DESCRAMBLING	Input	1	rxcoreclk	A 1 on this signal causes the SFI4.2 descrambler to be bypassed. This is only used as a test/debug feature.
i_DESCRBL_LSB_MSB_N	Input	1	rxcoreclk	A 1 on this signal causes the descrambler to unscramble the data lsb first. Otherwise, data is unscrambled msb first, as per the OIF standard.
o_RX_FIFO_OVFL	Output	1	rxusrclk	This is the FIFO overflow indicator. The FIFO should never overflow under normal operating conditions.
o_RX_FIFO_UNDLF	Output	1	rxcoreclk	This is the FIFO underflow indicator. The FIFO should never underflow under normal operating conditions.
ov_BLOCK_LOCK	Output	4	rxusrclk	This is the SFI4.2 block lock indicator for each lane.
ov_INVALID_CNT_0	Output	4	rxusrclk	This is the current number of invalid synchronization bits found. It is used in the SFI4.2 block lock algorithm.
ov_INVALID_CNT_1				
ov_INVALID_CNT_2				
ov_INVALID_CNT_3				
ov_SYNC_COUNTER_0	Output	10	rxusrclk	This is the number of bitslips required to realign each of the SFI4.2 lanes.
ov_SYNC_COUNTER_1				
ov_SYNC_COUNTER_2				
ov_SYNC_COUNTER_3				
ov_SKEW_3_2	Output	10	rxusrclk	This is the skew between lanes 3 and 2. In Normal mode, the range is ±32. In Extended-Skew mode the range is ±512.
ov_SKEW_3_1	Output	10	rxusrclk	This is the skew between lanes 3 and 1. In Normal mode, the range is ±32. In Extended-Skew mode the range is ±512.

Table 1: **Port List and Description for SFI4.2 Interface *(Cont'd)***

Port	Type	Width	Clock Domain	Description
ov_SKEW_3_0	Output	10	rxusrclk	This is the skew between lanes 3 and 0. In Normal mode, the range is ±32. In Extended-Skew mode the range is ±512.
o_RXPLL_LOCKED	Output	1	rxcoreclk	This is the lock signal from the 32/33 RX phase-locked loop (PLL) used to generate the rxcoreclk.
SFI4.2 TX Interface Control and Status				
i_BYPASS_SCRAMBLING	Input	1	txcoreclk	A 1 on this signal causes the SFI4.2 scrambler to be bypassed (only used as a test/debug feature).
i_SCRBL_LSB_MSB_N	Input	1	txcoreclk	A 1 on this signal causes the scrambler to scramble the data lsb first, otherwise, it is scrambled msb first as per the OIF standard.
iv_CORRUPT_SYNC_BITS	Input	4	txusrclk	A 1 on any of the bits corrupts the outgoing synchronization pattern and replace it with the iv_SYNC_BITS value. Bit 0 corresponds to lane 0 and so on.
iv_SYNC_BITS	Input	2	txusrclk	This is the value to be inserted instead of the normal synchronization bit pattern when corrupting the data.
iv_TXDELAY_0	Input	5	txusrclk	This is the number of 16-bit words of delay added on each lane. This feature is used to test the deskew algorithm of the downstream receiver. The valid range for this signal is 0 to 31.
iv_TXDELAY_1				
iv_TXDELAY_2				
iv_TXDELAY_2				
o_TX_FIFO_OVFL	Output	1	txcoreclk	This is the FIFO overflow indicator. The FIFO should never overflow under normal operating conditions.
o_TX_FIFO_UNDLF	Output	1	txusrclk	This is the FIFO underflow indicator. The FIFO should never underflow under normal operating conditions.

SFI4.2 General Description

The SFI4.2 block diagram is shown in Figure 4. The standard SFI4.2 implementation can handle a maximum skew of ±32 UI between the SFI4.2 RX lanes. This is higher than the ±20 UI specified by the OIF standard. In Extended-Skew mode, the receiver can handle a maximum skew of ±256 UI.

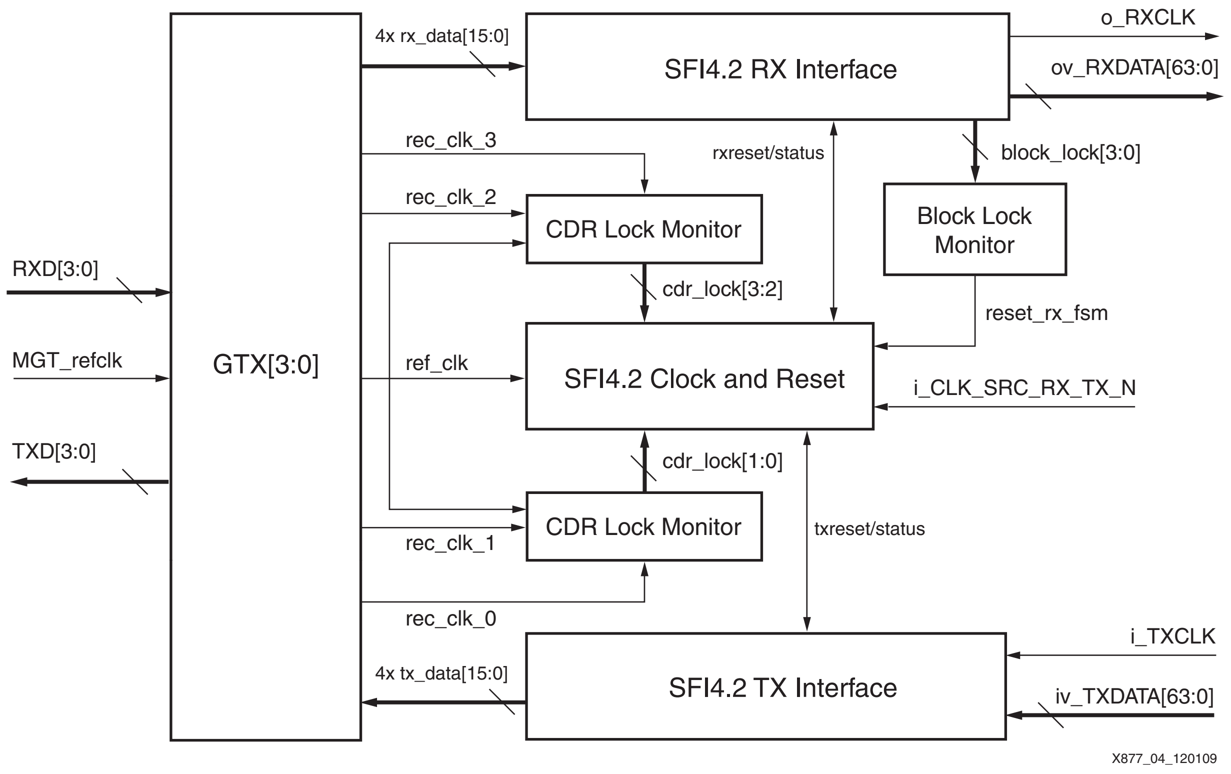


Figure 4: **SFI4.2 Top-Level Diagram**

Clocking an SFI4.2 Interface

The SFI4.2 implementation in this reference design requires that both interfaces (upstream and downstream) be completely synchronous. The SFI4.2 Clock and Reset block controls the reset sequence for both the RX and TX interfaces. Because this block runs off MGT_refclock, the manner in which MGT_refclock is generated is very important.

There are two possible clocking scenarios, as shown in Figure 5. If both devices are Xilinx devices, one device is configured as the master and the other as a slave.

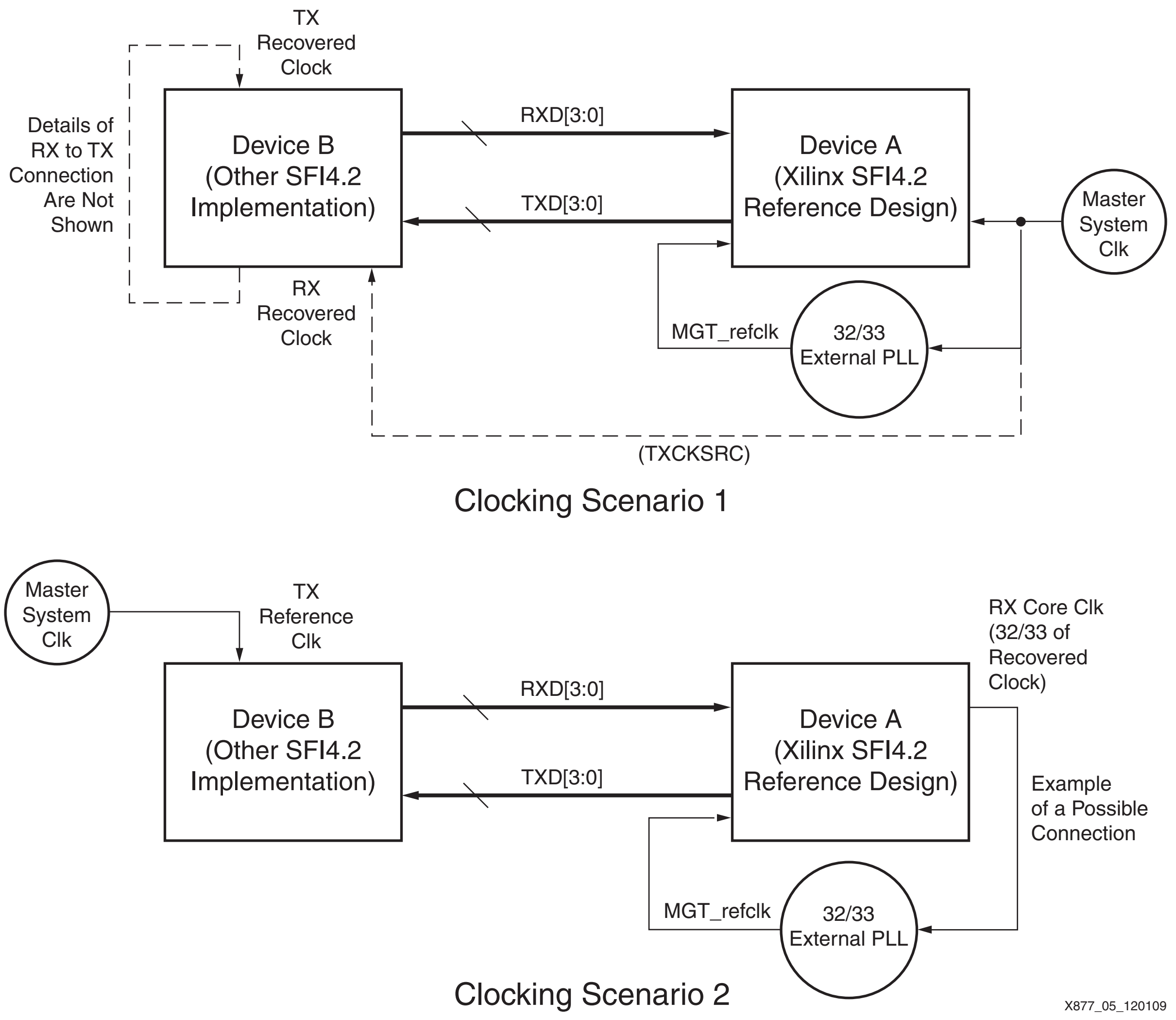

TX Recovered Clock
Details of RX to TX Connection Are Not Shown
Device B (Other SFI4.2 Implementation)
RXD[3:0]
TXD[3:0]
Device A (Xilinx SFI4.2 Reference Design)
Master System Clk
RX Recovered Clock
MGT_refclk
32/33 External PLL
(TXCKSRC)
Clocking Scenario 1
Master System Clk
TX Reference Clk
RX Core Clk (32/33 of Recovered Clock)
Device B (Other SFI4.2 Implementation)
RXD[3:0]
TXD[3:0]
Device A (Xilinx SFI4.2 Reference Design)
Example of a Possible Connection
MGT_refclk
32/33 External PLL
Clocking Scenario 2
X877_05_120109

Figure 5: **SFI4.2 Clocking Scenarios**

The simplest scenario is clocking scenario 1, where the Xilinx FPGA is the master. In this scenario, the main SFI4.2 clocking and reset block operates on the reference clock and is always present. The i_CLK_SRC_RX_TX_N input pin must be set to 0 because the master clocking source comes from the TX side relative to the SFI4.2 interface.

In clocking scenario 2, the reference clock is dependent on the recovered clock, and the recovered clock is not stable until the RX Reset state machine has completed. In this scenario, a reference clock must be generated from the 33/32 external PLL, regardless of the recovered clock. This allows the RX Finite State Machine (FSM) to operate and ensures that the recovered clock is valid. In this mode, the TX Reset state machine is held in reset until it has completed. When the interface is configured as in clocking scenario 2, the i_CLK_SRC_RX_TX_N input pin must be set to 1 because the master clocking source comes from the RX side relative to the SFI4.2 interface.

The OIF SFI4.2 specification defines a TXCKSRC clock that is used to synchronize both devices. The SFI4.2 implementation in this reference design does not use or generate the TXCKSRC clock. This clock is most likely available on the board, depending on the configuration. As shown in Figure 5, clocking scenario 1, TXCKSRC can be generated from the master clock. The use of this clock depends on the requirements of both SFI4.2 interfaces.

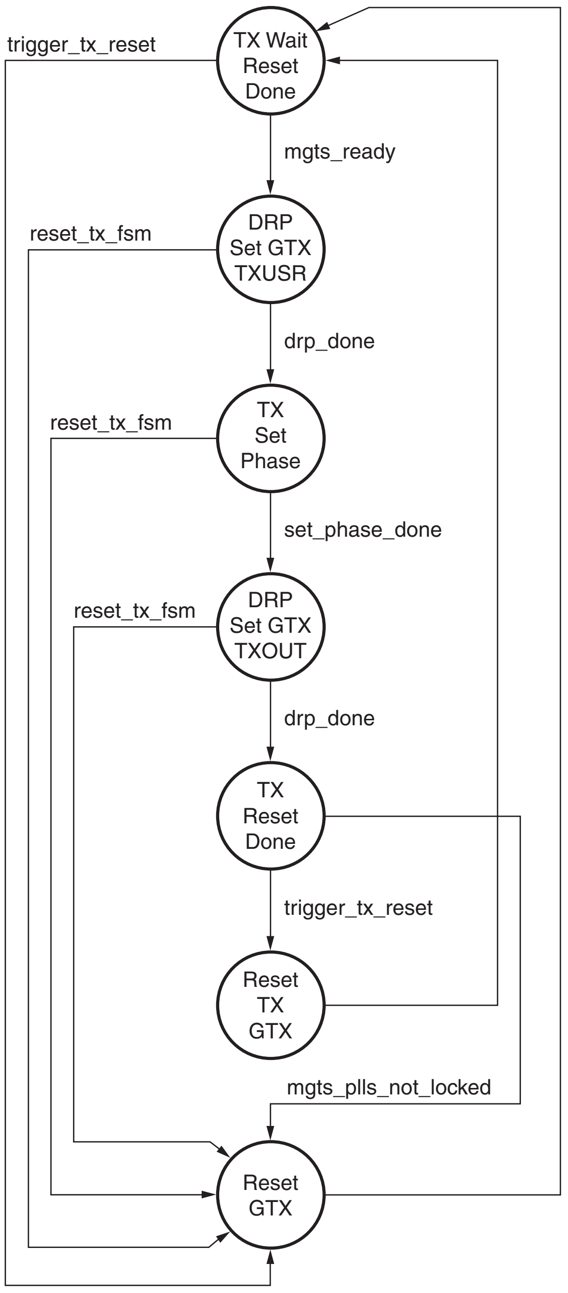
SFI4.2 Clocking and Resets

The SFI4.2 interface has a dedicated clock and reset block that handles all the clocking and reset needs of the interface. There are two main state machines: the TX Reset FSM and the RX Reset FSM. Both state machines are completely independent of each other and are run off the reference clock.

Either state machine or the entire interface can be reset. A rising-edge transition on the i_TRIGGER_RX_RESET control bit causes the RX FSM, CDR, and RX PCS to reset. A rising-edge transition on the i_TRIGGER_TX_RESET control bit causes the TX FSM and the TX PCS to reset. The entire SFI4.2, including the SerDes, is held in reset when the i_RST pin is asserted. The FSMs are described in more detail in the rest of this section.

Transmit Reset Block

The main purpose of the transmit reset block is to set the SerDes in low-skew mode. More detail on the TX low-skew mode and the implementation of the TX Reset FSM used in this design can be found in the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 3]. A diagram of the TX Reset FSM is shown in Figure 6.



X877_06_120109

Figure 6: **SFI4.2 TX Reset FSM**

Notes for Figure 6:

1. mgts_ready is mgts_pll_locked && mgt_reset_done.
2. reset_tx_fsm is not mgts_ready or trigger_tx_reset.

In the first state, the TX Reset FSM waits until the GTX transceiver PLLs are locked and the internal GTX transceiver reset sequence is complete. After completing the first state, the GTX transceiver TXUSR clock attribute is changed via the DRP port to TXUSR. Then, the TX Set Phase sequence is performed on all SerDes at the same time. After the TX Set Phase sequence is complete, the GTX transceiver TXUSR clock attribute is changed back to the TXOUT clock and the TX Reset pulse is asserted one final time before the user logic is released from reset. All logic on the TXUSRCLK domain (TX 66-bit clock domain) is held in reset whenever the TX FSM is not in the TX RESET DONE state.

Two conditions cause the state machine to restart:

- The GTX transceiver PLLs lose lock.
- The user asserts the i_TRIGGER_TX_RESET input signal.

If the GTX transceiver PLLs lose lock, a complete GTX transceiver reset is performed to reset the internal GTX transceiver PLL. This causes the RX FSM to reset as well because the PLLs and CDR lose lock. In the second case (the user triggers a TX Reset), the TX digital logic resets, but the clocks are not affected.

Transmit Clocking

The REFCLKOUT port of GTX transceiver 0 is connected to a global clock buffer (BUFG) and the output becomes the TXUSRCLK. This can be seen in Figure 10, page 17. All of the logic on the TXUSRCLK domain has an active-High synchronous reset.

The i_TX_CLK clock must be frequency locked and a 32/33 multiple of the GTX transceiver reference clock. The easiest was to accomplish this is to feed the i_TX_CLK clock to an external 33/32 PLL and feed the clock output by the PLL to the GTX transceiver reference clock port. The GTX transmitter is synchronous to the reference clock, which is slightly different than what the OIF standard calls for. The OIF standard specifies that the transmitter must be synchronous to the TXCKSRC clock, which is synchronous to the sink device. This is not an issue because the downstream receiver can operate on the recovered clock. This application note does not specify how the reference clock is generated, but it could be sourced from a PLL fed with the TXCKSRC clock. In this case, it would be fully compliant to the OIF specification.

Receive Reset Block

The FSM in the receiver reset block puts all of the RX SerDes into low-latency mode. It also controls the reset of the RX CDR. More details on low-latency mode and the FSM in the receiver reset block can be found in the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 3]. An FSM state diagram for the RX Reset State Machine is shown in Figure 7.

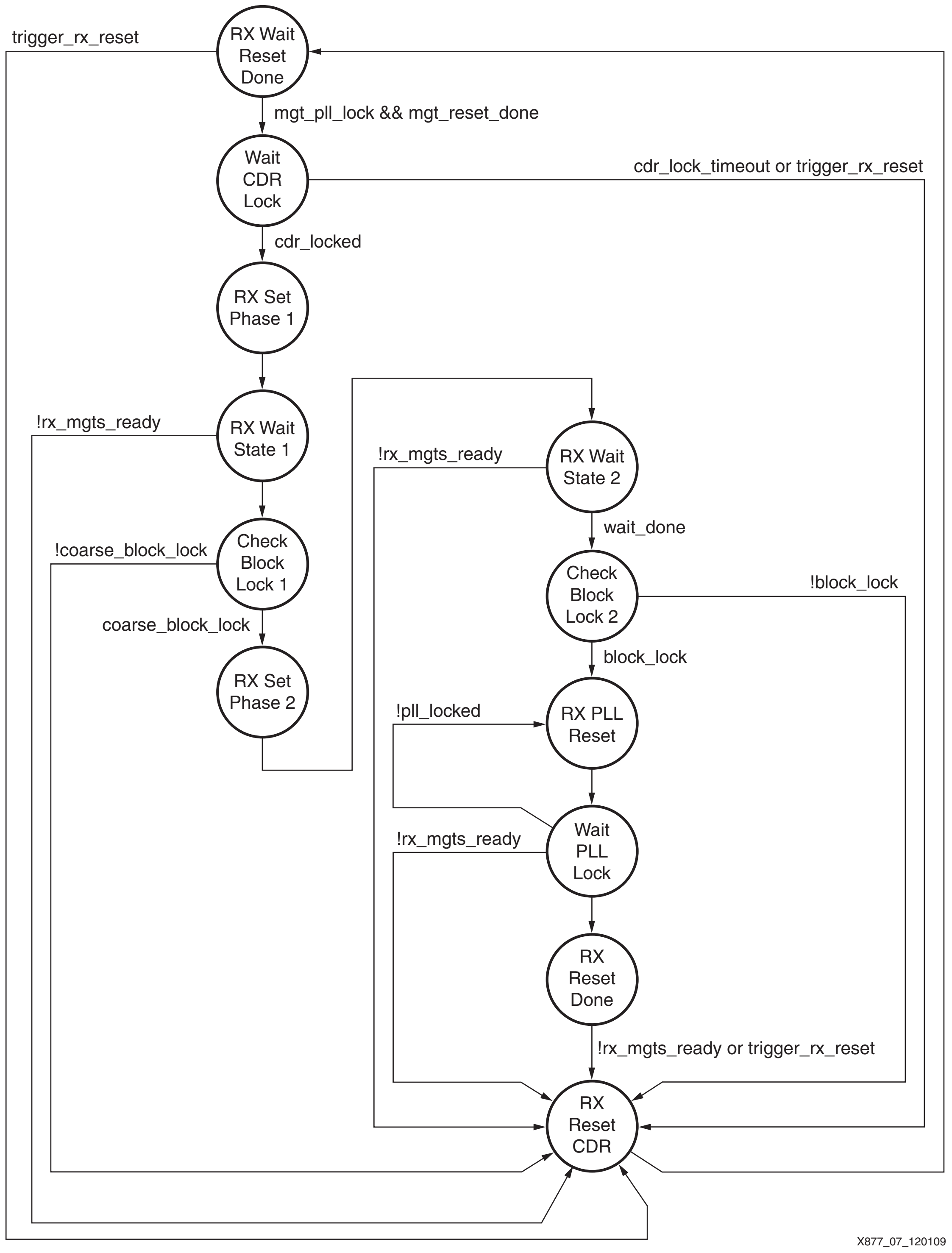


Figure 7: **SFI4.2 RX Reset FSM**

Notes for Figure 7:

1. rx_mgts_ready is mgts_pll_locked && rx_cdr_locked.
2. coarse_block_lock is defined as the receiver detecting less than 15 errors every 64 frames.
3. block_lock is defined by the OIF standard (refer to Figure 16).

The trigger_rx_reset signal referenced in Figure 7 is sourced from a logical OR of the i_TRIGGER_RX_RESET input port and the o_TRIGGER_RESET output from the

SFI42_BLOCK_LOCK_MONITOR block. This configuration ensures that the CDR and Low Latency control setting is only performed when valid data is received on the interface. Without valid data, the CDR cannot lock properly. More details on this module are provided in SFI4.2 Block Lock Monitor, page 14.

The first state in the RX FSM waits for the MGT PLL to lock and for the MGT Reset to complete. Then, the FSM waits approximately 1.5 ms for the CDR to stabilize. The CDR stabilizes in much less time than 1.5 ms. However, the FSM must wait 1.5 ms because the CDR Stable circuit is slow to determine the CDR status. If the CDR does not stabilize in 1.5 ms, it resets and the sequence restarts. For more information on the CDR Stable mechanism, refer to SFI4.2 CDR Stable Circuit, page 15.

After the CDR has stabilized, the RX SET Phase sequence is performed. This ensures that the internal clocks of all the SerDes are phase aligned with the RXUSRCLK clock, which in this case is connected to a regional clock buffer (BUFR). To ensure low latency on the receiver, the buffer is bypassed and the RX Set Phase sequence is required.

After these states are complete, the receiver should start to receive valid data. The coarse_block_lock signal is asserted when the SFI4.2 framer logic detects less than 15 errors in 64 frames. To guarantee that the setphase logic has executed, after valid data is present at the receiver, the RX Set Phase sequence is executed again.

After the second RX Set Phase, all four SFI4.2 lanes should be locked onto the 64/66B pattern. With the SFI4.2 lanes locked and the recovered clock stable, the RX PLL is reset to ensure that it is properly locked on the recovered clock. After the RX PLL is locked, the RX 66-bit and RX 64-bit clock domains are released from reset. At this point, four conditions can cause the reset sequence to restart:

1. The CDRs lose lock. This could occur if the upstream device stops transmitting data.
2. The MGT PLLs lose lock. This occurs when the GTX transceiver is reset.
3. The Block Lock Monitor triggers an RX Reset. This occurs when a valid lock is lost on the RX data.
4. The user triggers an RX Reset. This occurs when the user asserts the i_TRIGGER_RX_RESET control bit.

Receive Clocking

The SFI4.2 receive clocking can be seen in Figure 11, page 19. The recovered clock from SerDes 0 is connected to a BUFG and is fed to the RXUSRCLK of all four SerDes. The recovered clock is also fed to a PLL to get the 64-bit RX clock. The PLL performs the 32/33 multiplication on the received clock.

SFI4.2 Block Lock Monitor

The Block Lock Monitor block, shown in the SFI4.2 top-level diagram (Figure 4, page 7), is used to ensure that the SFI4.2 receiver always recovers properly in the case of data loss or any other condition. The Block Lock Monitor FSM diagram is shown in Figure 8.

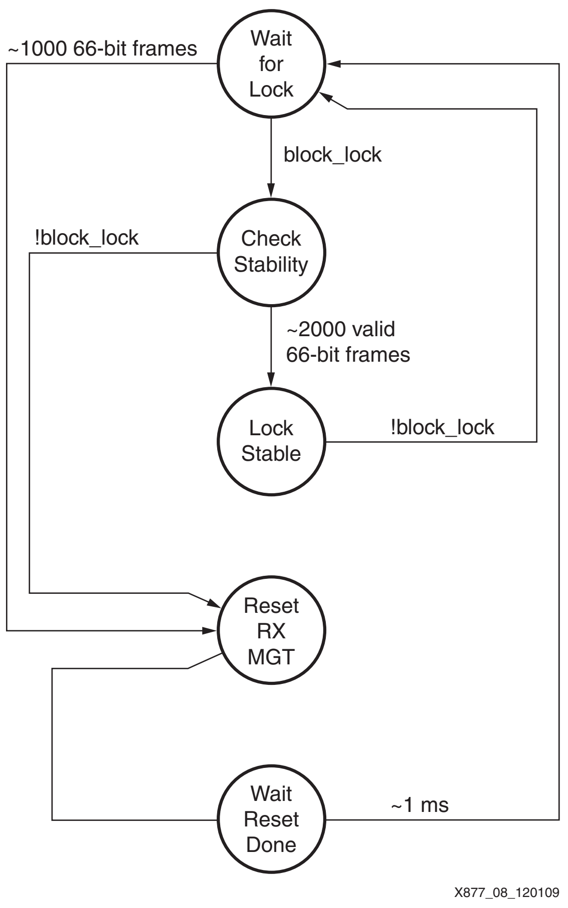


Figure 8: **SFI4.2 Block Lock Monitor FSM**

The Block Lock Monitor FSM is held in reset whenever rx_core_clk clock is held in reset, or when there is a reset on the DRP clock. The rx_core_clk clock is held in reset whenever the RX Reset FSM is not in the RX Reset Done state. The way the RX Reset FSM is coded, the logic should be in block lock at this point. If all lanes are not locked in approximately 1,000 SFI4.2 frames, the entire SFI4.2 receive interface is reset. As soon as block lock is achieved, the FSM ensures that it is stable. If approximately 2,000 valid frames are received without losing lock, the FSM goes into the Lock Stable state. Otherwise, the RX interface is reset again.

The final state for the Block Lock Monitor FSM is Wait Reset Done. This state ensures that the FSM is reset properly. If after 1 ms the FSM is not reset, a new reset trigger is asserted. This does not occur under normal operating conditions.

SFI4.2 CDR Stable Circuit

The GTX transceiver SerDes does not output a CDR Stable status indicator. Therefore, a CDR Stable Monitor Circuit (Figure 9) is provided to infer the CDR Lock status. One CDR Stable Monitor Circuit is provided for each GTX_DUAL tile. This circuit declares the CDR Lock status as stable if the difference between the SerDes Reference clock and the SerDes Recovered clock is less than 5,000 ppm.

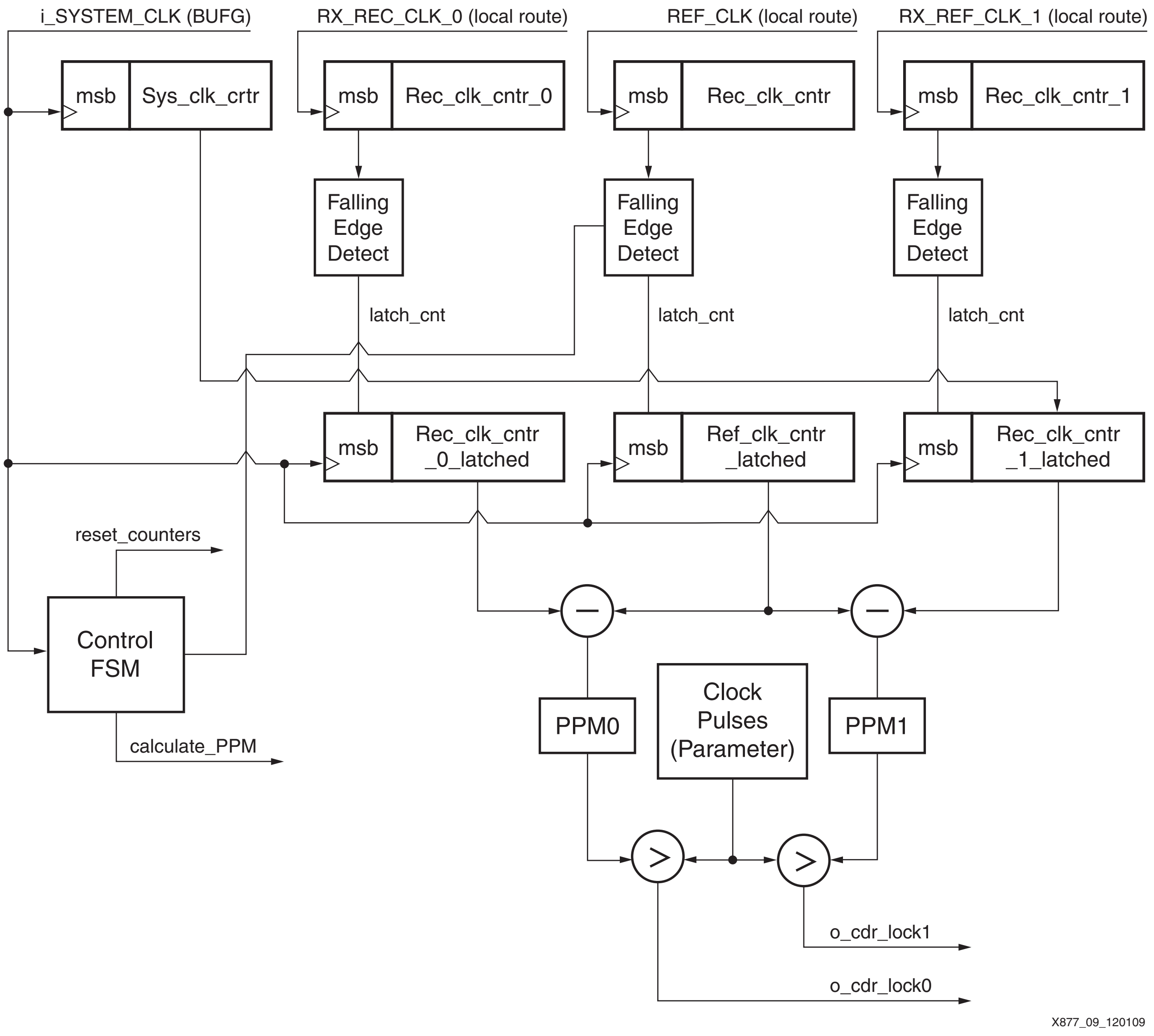


Figure 9: **SFI4.2 CDR Stable Monitor**

The CDR Stable Monitor circuit has four counters:

- **System Clock Counter**: a 20-bit counter running off the 50 MHz DRP clock.
- **Reference Clock Counter**: a 20-bit counter running off the local (non-BUFG) reference clock.
- **Recovered Clock 0 Counter**: a 20-bit counter running off the local recovered clock 0.
- **Recovered Clock 1 Counter**: a 20-bit counter running off the local recovered clock 1.

The System Clock counter is used as the base counter and to calculate the ppm differences between all of the different counters. All four counters are reset at the same time and are free running until they are all reset again after the ppm calculation is complete. When each counter rolls over, the value of the System Clock counter is latched in a register. The latched value does not change until the ppm calculation is complete and the circuit is reset. The ppm calculation is not done until the reference counter has rolled over twice. At this point, the circuit is guaranteed to have latched all three counters if the clocks are present.

The ppm difference is calculated by subtracting the reference clock latched counters from the recovered clock counters. If the difference is above a threshold, the CDR is declared unstable. The parameters in this module are set to detect a 5,000 ppm offset. All counters are 20-bit counters with a maximum count of 1,048,576. The ppm offset is calculated as shown in Equation 1.

$$\frac{5000 \times 1{,}048{,}576}{1{,}000{,}000} = 5{,}243 \qquad \textit{Equation 1}$$

The system clock counter is the counter that is latched. Therefore, this number needs to be scaled accordingly. The parameter CLOCK_PULSES is calculated as shown in Equation 2.

$$\frac{\text{PPM\_OFFSET} \times \text{sysclk\_freq}}{\text{refclk\_freq}} = \frac{5{,}243 \times 50\text{ MHz}}{181\text{ MHz}} = 1{,}448 \qquad \textit{Equation 2}$$

The GTX transceiver is also configured to perform an internal CDR reset whenever the link goes idle. This helps ensure that the data is always properly recovered. The value of 5,000 ppm is chosen to give a rough order of magnitude estimate of the clock being recovered from the receive data. This value is only used as a guide to control the RX reset state machine.

SFI4.2 Transmitter

The SFI4.2 Transmit interface receives a 64-bit data bus and a forwarding clock from the FPGA logic. The data is scrambled using the $X^{58} + X^{39} + 1$ polynomial, 64/66B encoded, and then split across the four MGT lanes. A block diagram of the TX interface is shown in Figure 10.

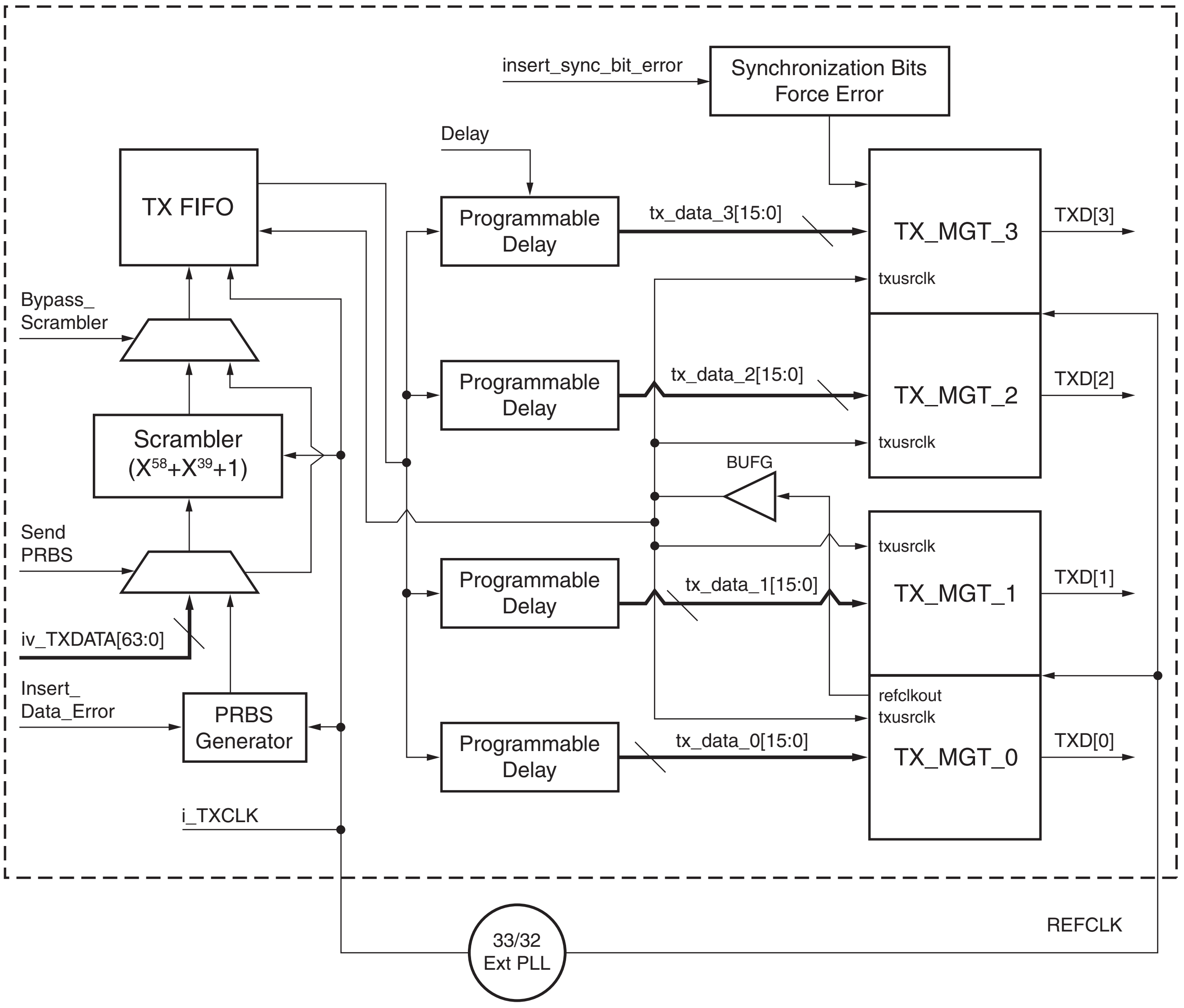


Figure 10: **SFI4.2 Transmit Interface Block Diagram**

TX Scrambler

The outgoing data is scrambled with the $X^{58} + X^{39} + 1$ polynomial as defined by the SFI4.2 standard. For debug purposes, the scrambler can be bypassed. However, unless the data is scrambled, there is no guarantee that the SFI4.2 receiver properly locks onto the synchronization bits.

Extended-Skew Mode

Some applications require a receiver skew tolerance that is larger than the skew tolerance defined in the OIF specification. Therefore, in Extended-Skew mode, the synchronization bits are inverted once every eight frames. This allows the receiver to deskew the data on the 8-frame boundary corresponding to a maximum of ±256 UI (as opposed to only ±32 bits). The user can switch between Standard and Extended-Skew mode by asserting the i_EXT_SKEW_EN input pin.

TX Gearbox

The TX SFI4.2 uses the SerDes built-in Gearbox of the GTX transceiver. To use the Gearbox, a counter that cycles from 0 to 32 is fed to the SerDes along with the SFI4.2 synchronization bits. More details on the GTX transceiver Gearbox can be found in the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 3].

The synchronization bits for each lane can also be corrupted. This is done to stress the receiver. Four input pins (iv_CORRUPT_SYNC_BITS[3:0]), when enabled, cause the normal synchronization bits to be overwritten by the iv_SYNC_BITS[1:0] input pins. Only one input signal determines the new synchronization bits.

TX Programmable Delay

To enable full testing of the RX interface in Extended-Skew mode, there are four programmable delay blocks in the TX datapath. The delay blocks are arrays of 16 SRL32s. An SRL32 is a 32-bit shift register implemented in a single LUT. For more information on the SRL, refer to the *Virtex-5 FPGA User Guide* [Ref 4].

On each lane, delay can be added in increments of 16 bits, anywhere from 0 to 31. This range allows the user to completely test the ±256 bits of skew tolerance specified by the Extended-Skew mode. Four control input signals (iv_TX_DELAY_0[4:0] to iv_TX_DELAY_3[4:0]) control this feature. By default, the delays through this block are set to 0. The programmable delay can also be used in Standard SFI4.2 mode. However, the receiver only handles ±32 bits of skew in this mode.

SFI4.2 Receiver

The SFI4.2 RX interface receives 16 bits of data from four MGT lanes. The interface must recover the clock and reassemble the data into a 64-bit data stream. The recovered data has the 64/66B encoding removed and is unscrambled using the $X^{58} + X^{39} + 1$ polynomial. The final 64-bit data stream and corresponding clock is sent to the downstream block. A block diagram of the RX interface is shown in Figure 11.

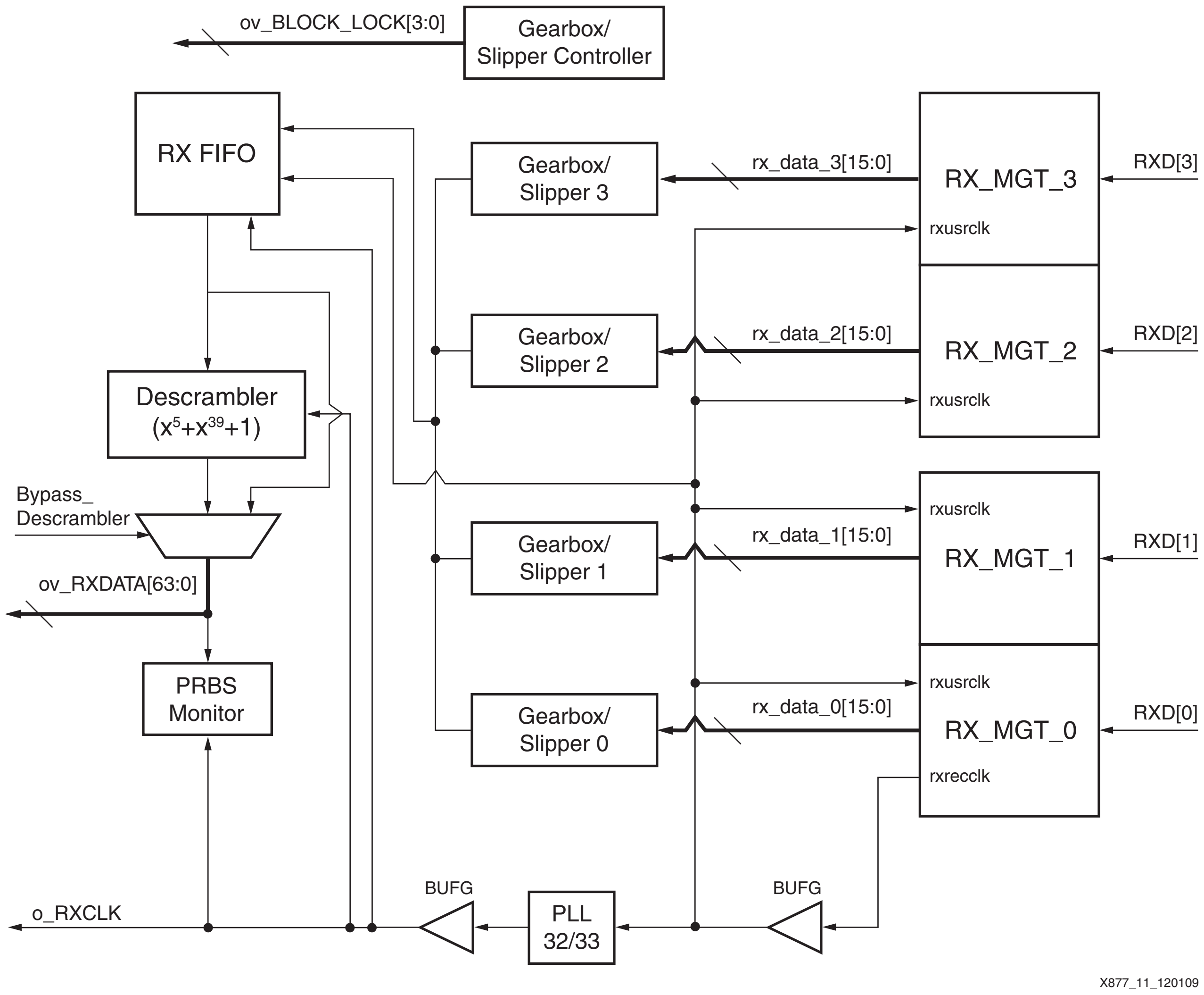


Figure 11: **SFI4.2 Receive Interface Block Diagram**

RX Gearbox and Slipper Overview

Gearboxes and slippers are used to find the SFI4.2 synchronization patterns and to deskew the lanes. In Normal mode, the receiver can deskew up to ±32 bits. In Extended-Skew mode, it can deskew up to ±256 bits.

Slippers are used to find the SFI4.2 synchronization bits and determine when there is a “block lock.” Gearboxes are used to convert the 16 bits received every clock cycle to the 66-bit SFI4.2 Frame. The gearbox also handles all physical shifting of the data.

The GTX transceiver has a built-in gearbox but shifts are not done on a bit-by-bit basis. The implementation in this reference design counts the number of slips to determine the skew between lanes. The implementation also requires the ability to switch between Normal mode and Extended-Skew mode. Therefore, a custom gearbox is implemented that heavily utilizes the SRL feature of the Virtex-5 family. This significantly reduces the amount of logic required to implement the feature.

To deskew all lanes, a four-step process is implemented:

Step 1: Initialize Deskew

The SFI4.2 protocol requires that each lane is offset from neighboring lanes by 16 bits. The first step is to remove this delay by shifting the data through an SRL16, as shown in Figure 12.

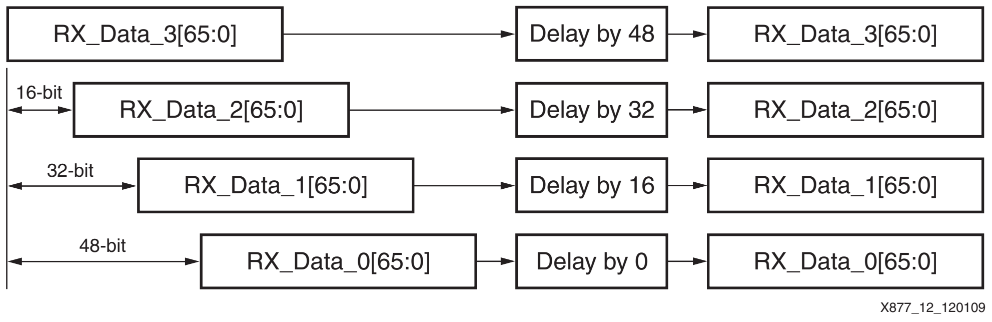


Figure 12: **SFI4.2 Receive Initial Deskew**

Step 2: Advance Lane 3

In Normal mode, the SFI4.2 can handle ±32 bits of skew relative to lane 3. To facilitate this deskew, lane 3 is advanced by 32 bits to ensure that it is ahead of the other lanes, as shown in Figure 13.

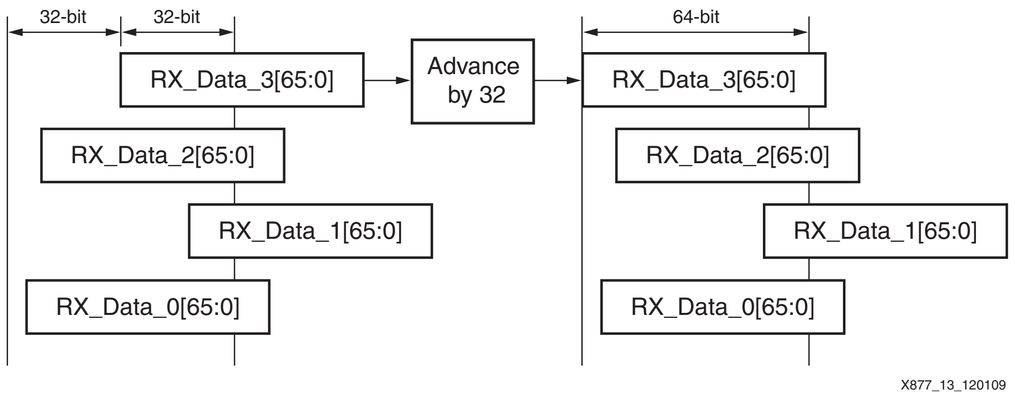


Figure 13: **SFI4.2 Receive Advance Lane 3 Normal Mode**

Similarly, in Extended-Skew mode, lane 3 is advanced by 64 bits, as shown in Figure 14.

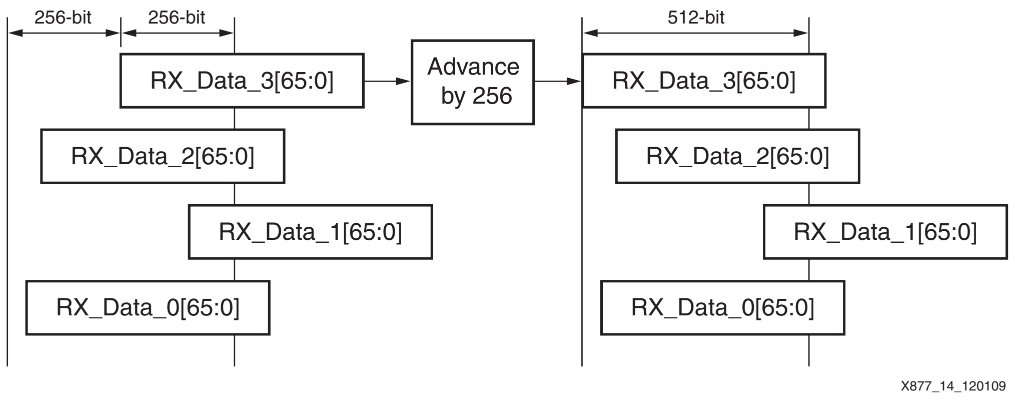


Figure 14: **SFI4.2 Receive Advance Lane 3 Extended-Skew Mode**

After completing this step, the beginning of each lane is ensured to be within the next 64 bits or 512 bits, respectively, as required by the Normal and Extended-Skew modes.

Step 3: Slip All Lanes Together Until Lane 3 is Locked

At this point, the lanes are aligned relative to each other. All lanes are slipped together until lane 3 is locked. This ensures that while the algorithm is searching for the synchronization bits of lane 3, the relative positioning of the other lanes do not change.

Step 4: Slip Lanes 2 Down to 0

At this point, the slippers for lanes 0, 1, and 2 are released, and the lanes slip until each lane is locked, as shown in Figure 15. Each slipper tracks the number of slips required to obtain block lock. The number of slips is then used to calculate the skew between lanes. Details of the skew calculation can be found in SFI4.2 Slipper Controller, page 23.

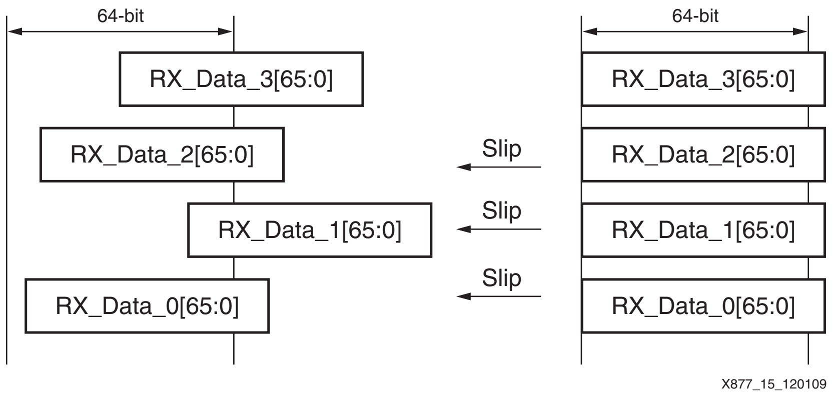


Figure 15: **SFI4.2 Receive Slip Lane 2 Down to 0**

SFI4.2 Slipper

The slipper module is used to detect SFI4.2 synchronization bits in the incoming data stream. A detailed flowchart of the SFI4.2 slipper algorithm is shown in Figure 16. This flowchart corresponds to the OIF SFI4.2 standard.

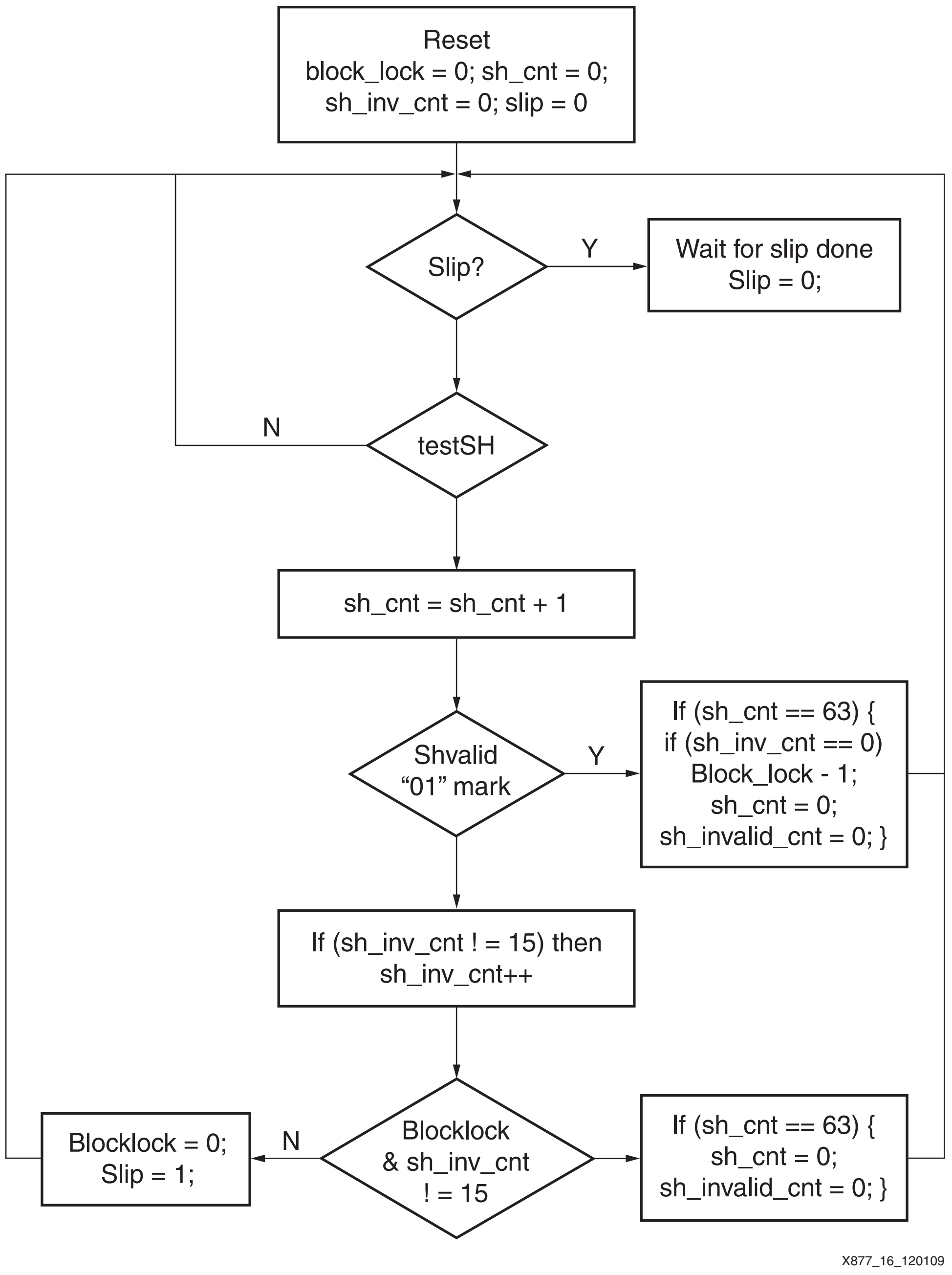


Figure 16: **SFI4.2 Slipper Algorithm**

The slipper has a counter that keeps track of the number of slips required to find the synchronization bits. In Normal mode, the counter counts from 0 to 65 and then rolls over. The synchronization bits are guaranteed to be found within the 66-bit frame.

In Extended-Skew mode, the slipper aligns itself on an 8-frame boundary. Therefore, the counter goes from 0 to 527 and then rolls over. Each lane has its own slipper, and the counter is used to determine skew between the lanes. The counter values for each lane are output on the ov_SYNC_COUNTER_3-0 signals.

When looking at the counter values, lane 3 does not appear synchronized to the other lanes. This is because all lanes are shifted when lane 3 is shifted, and they are then independently shifted to find their own block lock.

The number of invalid frames received by each lane is presented on signals ov_INVALID_CNT_3-0.

SFI4.2 Slipper Controller

The Slipper Controller is used to enable and disable the individual slippers. The slippers for lanes 0 to 2 must be disabled while lane 3 is acquiring block lock. After lane 3 is locked, the other three slippers are released.

The Slipper Controller block also calculates skew. As shown in Figure 12, page 20, the receiver removes the skew inserted on the transmitter and then adds either 32 or 256 bits of skew on lane 3, depending on the mode. To calculate the skew between lanes, the 32 or 256 bits is subtracted from the slip counters. This gives a 10-bit two's complement number, representing a maximum of ±256. If the receiver received the data without any skew (i.e., each lane was offset from each other by 16 bits), the three skew numbers would be 0. Skew numbers are output on three signals:

- ov_SKEW_3_2: Skew between lanes 3 and 2
- ov_SKEW_3_1: Skew between lanes 3 and 1
- ov_SKEW_3_0: Skew between lanes 3 and 0

SFI4.2 Gearbox Synchronization

The Gearbox Synchronization block handles converting the 16 bits of data received by the SerDes into 66-bit frames. The gearbox shifts out 16 bits of data and a 2-bit header. This cycle repeats itself every 33 clock cycles.

The header output by the gearbox is only valid once every four clock cycles. The data is valid every clock cycle except for one cycle out of every 33. Every four clock cycles, the gearbox increments its shift offset by two to keep up with the 66-bit frame. Table 2 shows the shift offset, header valid indicator, and data valid indicator for every clock cycle.

Table 2: **Gearbox Synchronization**

Cycle Count	Shift Offset	Header Valid	Data Valid
0	0	1	1
1	0	0	1
2	0	0	1
3	0	0	1
4	2	1	1
5	2	0	1
6	2	0	1
7	2	0	1
8	4	1	1
9	4	0	1
10	4	0	1
11	4	0	1
12	6	1	1
13	6	0	1
14	6	0	1
15	6	0	1

Table 2: **Gearbox Synchronization *(Cont'd)***

Cycle Count	Shift Offset	Header Valid	Data Valid
16	8	1	1
17	8	0	1
18	8	0	1
19	8	0	1
20	10	1	1
21	10	0	1
22	10	0	1
23	10	0	1
24	12	1	1
25	12	0	1
26	12	0	1
27	12	0	1
28	14	1	1
29	14	0	1
30	14	0	1
31	14	0	1
32	0	0	0

SFI4.2 Gearbox

Each lane has its own gearbox used to shift data. Figure 17 shows a block diagram of the SFI4.2 RX gearbox.

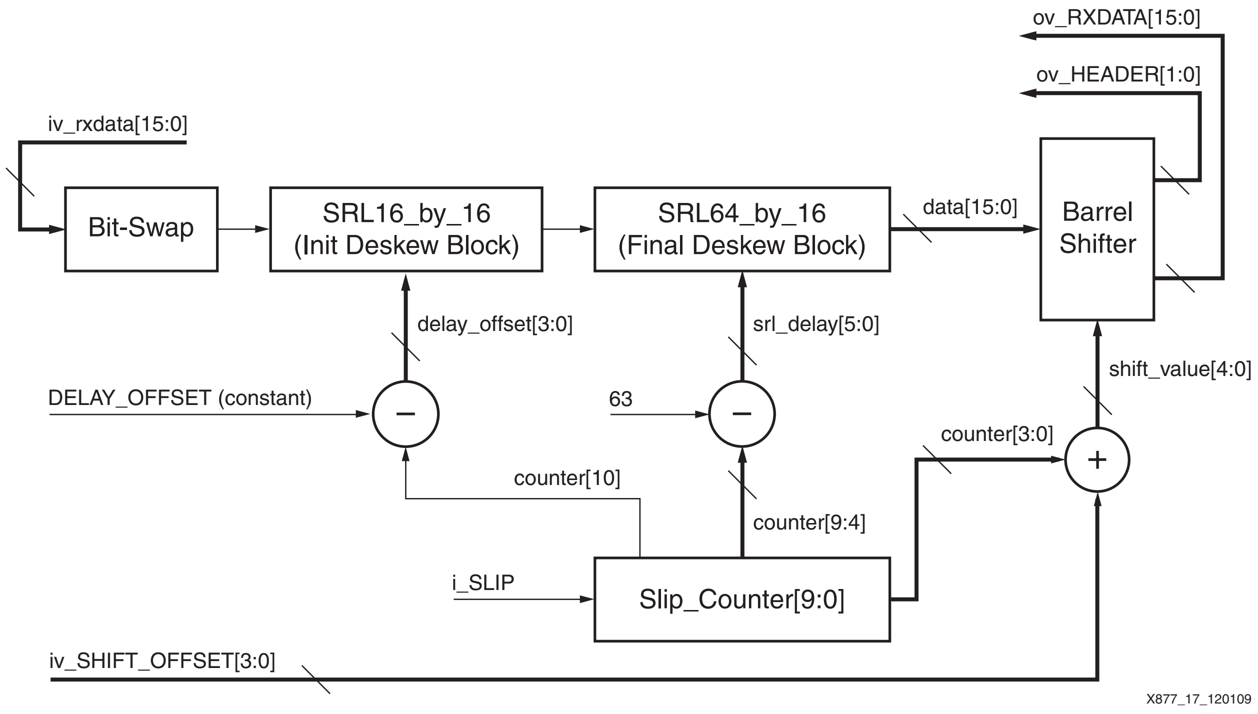


Figure 17: **SFI4.2 Gearbox**

The gearbox essentially consists of a small barrel shifter for fine shifting and SRL blocks for coarse shifting. The first portion of the gearbox is a bit swap. The bit swap is required because the GTX transceiver gearbox is used on the transmitter, but not on the receiver, and the bit ordering ends up being reversed.

As illustrated in Figure 12, page 20 and Figure 14, page 20, the SFI4.2 delay needs to be removed. This is done by the Init Deskew block. By default, lane 3 needs to be delayed by 3 slips (48 bits), lane 2 by 2 slips (32 bits) and lane 1 by 1 slip (16 bits). The next step is to advance lane 3 by either 2 slips (32 bits) or 16 slips (512 bits). In Extended-Skew mode, there cannot be a negative delay. Therefore, delay was added to all other lanes. Table 3 shows the DELAY_OFFSET for each lane.

Table 3: **Gearbox Delay Offset**

Lane	Normal Mode Delay	Extended-Skew Delay
3	3 – 2 = 1 (16 bits)	3 – 16 = –13 (+13) → 0 (0 bits)
2	2 (32 bits)	2 (+13) → 15 (240 bits)
1	1 (16 bits)	1 (+13) → 14 (224 bits)
0	0 (0 bits)	0 (+13) → 13 (208 bits)

As discussed in SFI4.2 Slipper, page 21, lane 3 can only slip up to 65 bits in Normal mode and up to 527 bits in Extended-Skew mode. All lanes slip with lane 3, and on top of the initial slip, they must slip until they are block locked. The maximum value for lanes 2 to 0 is 131 bits (65 + 66) for Normal mode and 1,039 bits (527 + 512) for Extended-Skew mode. A parameter in the RTL differentiates between lane 3 and lane 2 to 0. In the event that the synchronization bits are not found, the counter rolls over to its initial value, which corresponds to the lane 3 slip counter.

The barrel shifter is used for fine shifting. It can shift a maximum of 29 bits: 14 bits from the Gearbox Synchronization block (see Shift Offset column in Table 2, page 23) and 15 bits from the 4 least significant bits of the slip_counter. The barrel shifter is a 49-bit register with five layers of 2-to-1 multiplexers implementing the shifting. Figure 18 shows a block diagram of the barrel shifter.

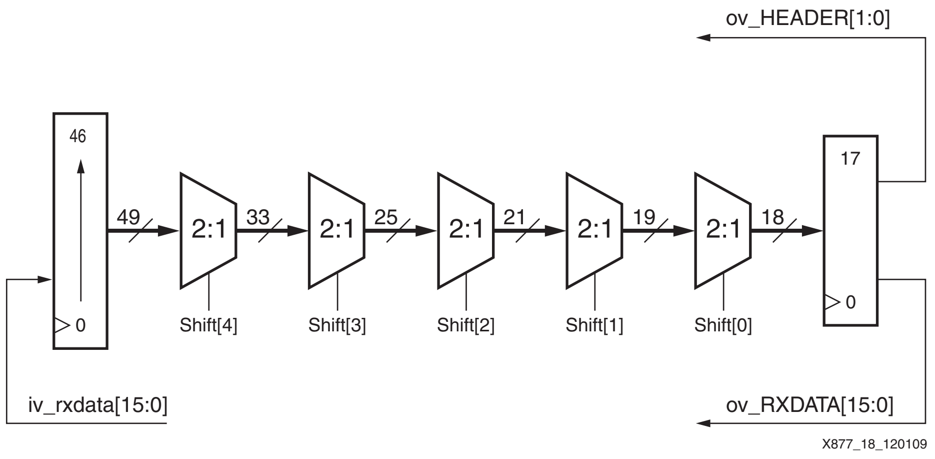


Figure 18: **SFI4.2 Barrel Shifter**

Every time the slip counter increments by 16 bits, the delay in the SRL64 is subtracted by 1. This is equivalent to a shift by 16 bits. Therefore, the maximum slip on any lane (1,039 bits) corresponds to 65 slips of 16 bits. As shown in Figure 17, page 25, the tenth bit of the slip counter is connected to the Init Deskew block because the Final Deskew block can only handle a shift of 64 bits.

RX Scrambler

The received data is unscrambled using the $X^{58} + X^{39} + 1$ polynomial, as defined by the SFI4.2 standard. For debug purposes, the descrambler can be bypassed.

SFI4.2 Resource Utilization

The FPGA resources used by the entire stand-alone SFI4.2 interface are shown in Table 4.

Table 4: **Resource Utilization Targeting a Virtex-5 FPGA XC5VFX70T**

Resource	Used	Comment
Slice Register	3,000	
Slice LUT	1,800	
FIFO36	2	RX and TX 64- to 66-bit clock domain crossing
BUFG	4	txusrclk, rxusrclk, rx_core_clk, tx_core_clk
PLL	1	RX 32/33 PLL
GTX_DUAL tile	2	4 lanes

Reference Design

The reference design files can be downloaded at https://secure.xilinx.com/webreg/clickthrough.do?cid=142154.

The reference design checklist in Table 5 includes simulation, implementation, and hardware verification details for the reference design.

Table 5: **Reference Design Checklist**

Parameter	Description
General	
Developer Name	Martin Charron
Target Devices	Virtex-5 FXT and TXT FPGAs
Source Code Provided	Yes
Source Code Format	VHDL,Verilog
Design Uses Code/IP from Existing Application	Yes, CORE Generator™ software with: • FIFO Generator, version 4.3 • ChipScope™ Pro analyzer, version 11.2
Simulation	
Functional Simulation Performed	Yes
Timing Simulation Performed	No
Testbench used for Functional Simulation	Yes
Testbench Format	VHDL, Verilog
Simulator Software Version Used	ModelSim SE, version 6.5a
SPICE/IBIS Simulation	No
Implementation	
Synthesis Software Tools/Version Used	XST, version 11.2
Implementation Software Tools/Versions Used	ISE® software, version 11.2
Static Timing Analysis Performed	Yes
Hardware Verification	
Hardware Verified	Yes
Hardware Platform Used for Verification	ML523 development board with Virtex-5 FPGA XC5VFX70T

SFI4.2 Verification Testbench

A Verilog testbench is provided with the reference design. The VHDL version can be tested if the simulator supports mixed-language simulations.

The testbench diagram is shown in Figure 19. It contains the Device Under Test (DUT), which is either the Verilog or VHDL version of the SFI42_IF_V5_FXT top level. A pattern generator and monitor are used to validate the data going through the SFI4.2 interface. A simple loopback is used to connect the TX and RX data pins.

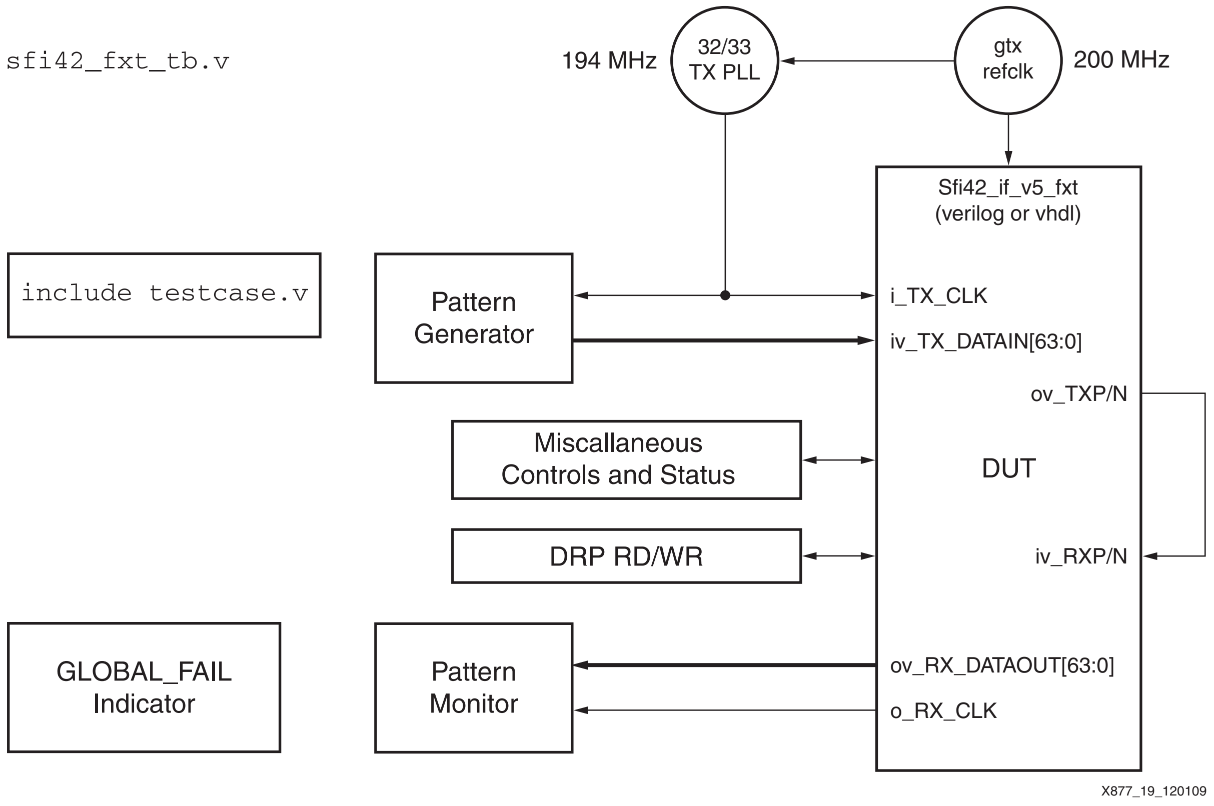


Figure 19: **SFI42 FXT Testbench**

There is a Verilog `include directive that includes the actual `testcase.v` files that contain the particular tests. To run a test case, the user must go to the test case directory, open either the Verilog or VHDL subdirectories, and run the scripts. The directory structure is shown in Figure 20.

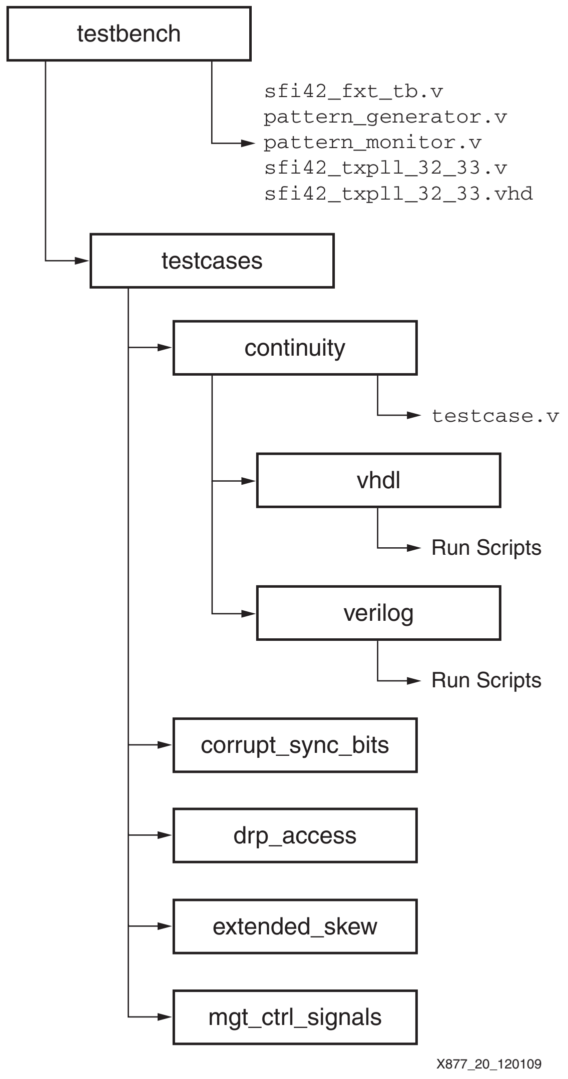


Figure 20: **SFI42 FXT Testbench Directory Structure**

Five test cases are used to test the functionality of the design:

- **Continuity Test case**: This test case is self checking. After completion, it reports if the test case has passed or failed. This test case provides a quick data continuity check through the SFI4.2 interface while set in Normal skew mode.
- **Extended-Skew Test case**: This test case is very similar to the Continuity test case. The main difference is that the DUT operates in Extended-Skew mode.
- **Corrupt Synchronization Bits**: This test case is not self checking. The waveforms must be verified. This test case ensures that the user can override the synchronization bits for each of the SFI4.2 lanes.
- **DRP Access**: This test case is self checking. It verifies that the GTX transceiver DRP port can be accessed to perform read and write cycles.
- **MGT Control Signals**: This test case is not self checking. The waveforms must be verified. This test case ensures that the Loopback, RXEQMIX, TXPREEMPHASIS, and TXDIFFCTRL GTX transceiver settings can be overridden.

SFI4.2 Hardware Testbench

The reference design is hardware tested on the ML523 development board. The SFI4.2 interface is wrapped with a pattern generator/monitor and four ChipScope analyzer Virtual I/O (VIO) cores. These are included to enable proper control and monitoring of the design. The sfi42_test_if testbench is only available in Verilog. However, the ISE software has mixed-language support, which allows the VHDL design to be tested as well. Figure 21 shows a top-level diagram of the hardware testbench.

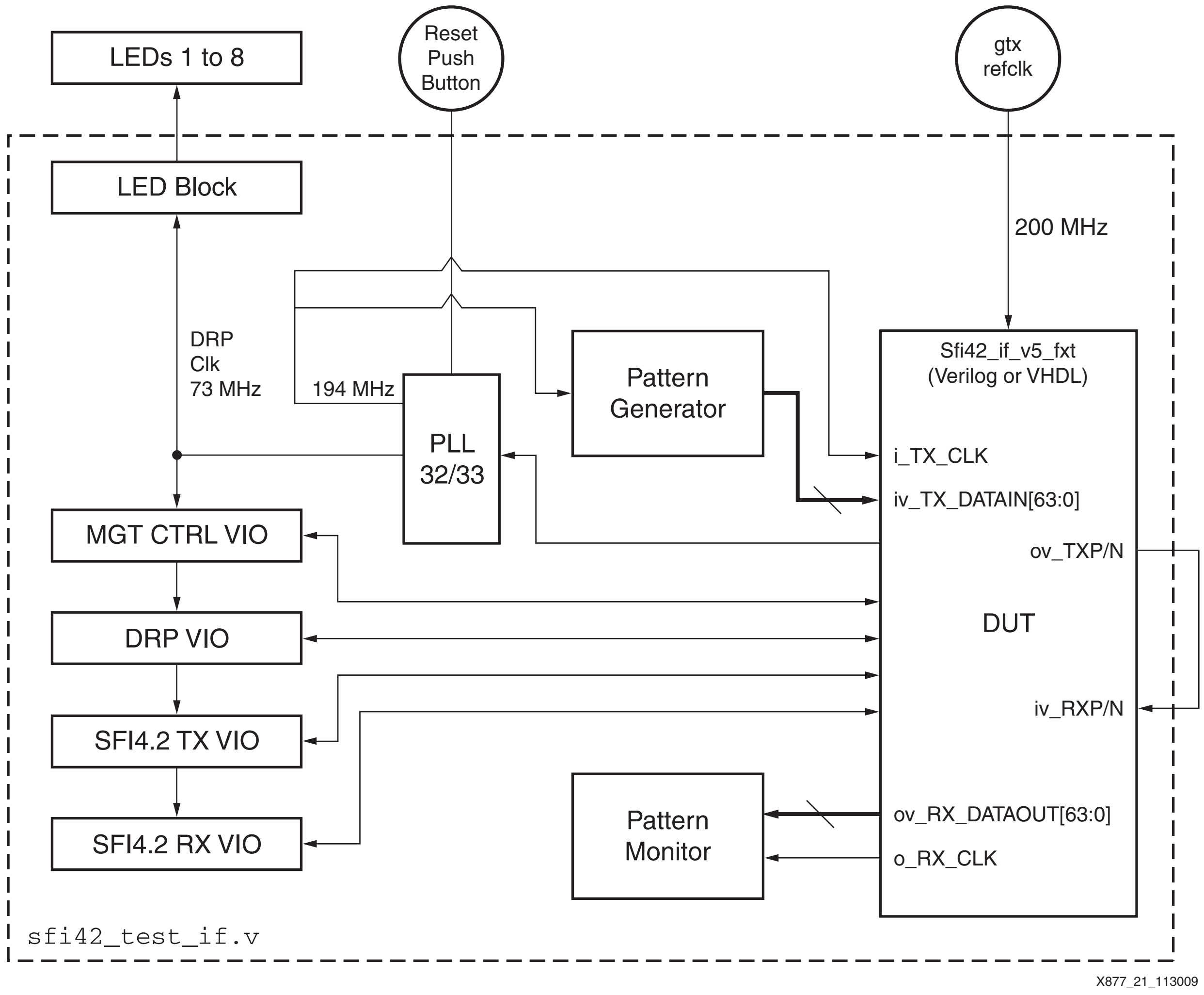
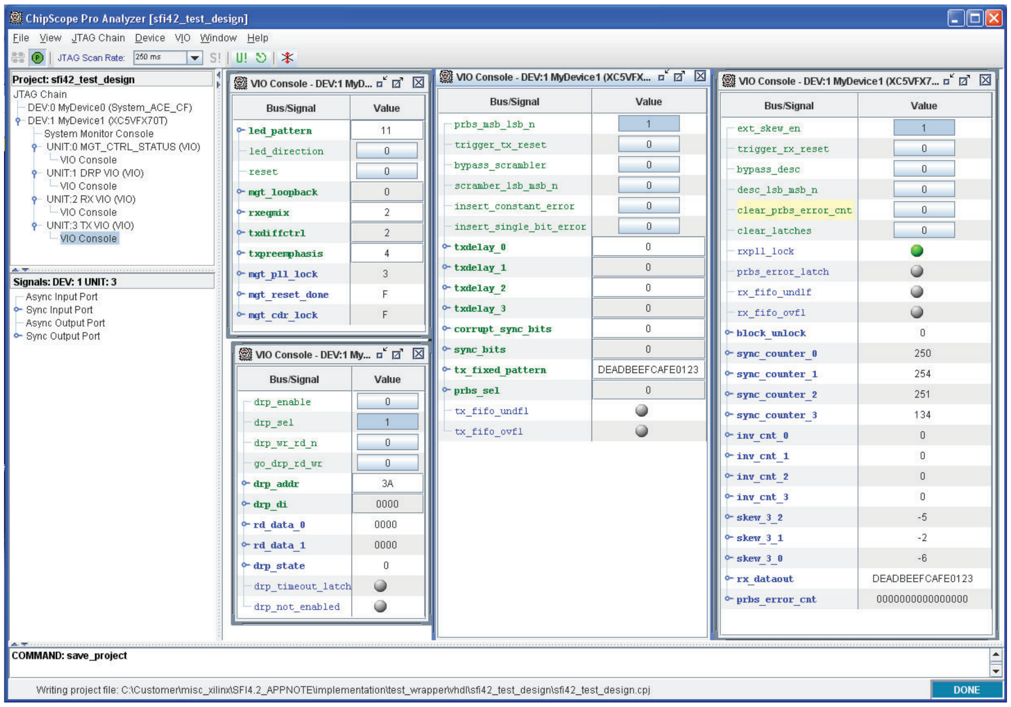


Figure 21: **SFI42 Test Interface**

ChipScope Analyzer Interface

As seen in Figure 21, four VIO cores control the design. Figure 22 contains a screen capture of the ChipScope analyzer window that shows how the ChipScope analyzer project file, `sfi42_test_design.cpj`, can be opened in the ChipScope analyzer to set up the VIO windows.


ChipScope Pro Analyzer [sfi42_test_design]
File View JTAG Chain Device VIO Window Help
JTAG Scan Rate: 250 ms
DONE

Project: sfi42_test_design

```
JTAG Chain
  DEV:0 MyDevice0 (System_ACE_CF)
  DEV:1 MyDevice1 (XC5VFX70T)
    System Monitor Console
    UNIT:0 MGT_CTRL_STATUS (VIO)
      VIO Console
    UNIT:1 DRP VIO (VIO)
      VIO Console
    UNIT:2 RX VIO (VIO)
      VIO Console
    UNIT:3 TX VIO (VIO)
      VIO Console
```

Signals: DEV: 1 UNIT: 3

```
Async Input Port
Sync Input Port
Async Output Port
Sync Output Port
```

VIO Console - DEV:1 MyD...

Bus/Signal	Value
led_pattern	11
led_direction	0
reset	0
mgt_loopback	0
rxeqmix	2
txdiffctrl	2
txpreemphasis	4
mgt_pll_lock	3
mgt_reset_done	F
mgt_cdr_lock	F

VIO Console - DEV:1 My...

Bus/Signal	Value
drp_enable	0
drp_sel	1
drp_wr_rd_n	0
go_drp_rd_wr	0
drp_addr	3A
drp_di	0000
rd_data_0	0000
rd_data_1	0000
drp_state	0
drp_timeout_latch	●
drp_not_enabled	●

VIO Console - DEV:1 MyDevice1 (XC5VFX...

Bus/Signal	Value
prbs_msb_lsb_n	1
trigger_tx_reset	0
bypass_scrambler	0
scramber_lsb_msb_n	0
insert_constant_error	0
insert_single_bit_error	0
txdelay_0	0
txdelay_1	0
txdelay_2	0
txdelay_3	0
corrupt_sync_bits	0
sync_bits	0
tx_fixed_pattern	DEADBEEFCAFE0123
prbs_sel	0
tx_fifo_undfl	●
tx_fifo_ovfl	●

VIO Console - DEV:1 MyDevice1 (XC5VFX7...

Bus/Signal	Value
ext_skew_en	1
trigger_rx_reset	0
bypass_desc	0
desc_lsb_msb_n	0
clear_prbs_error_cnt	0
clear_latches	0
rxpll_lock	🟢
prbs_error_latch	●
rx_fifo_undlf	●
rx_fifo_ovfl	●
block_unlock	0
sync_counter_0	250
sync_counter_1	254
sync_counter_2	251
sync_counter_3	134
inv_cnt_0	0
inv_cnt_1	0
inv_cnt_2	0
inv_cnt_3	0
skew_3_2	-5
skew_3_1	-2
skew_3_0	-6
rx_dataout	DEADBEEFCAFE0123
prbs_error_cnt	0000000000000000

```
COMMAND: save_project
Writing project file: C:\Customer\misc_xilinx\SFI4.2_APPNOTE\implementation\test_wrapper\vhdl\sfi42_test_design\sfi42_test_design.cpj
```

X877_22_113009

Figure 22: **ChipScope Analyzer Screen Capture**

Table 6 through Table 9 describe the reference design VIO signals.

Table 6: **MGT Control and Status VIO Signals**

VIO Signal Name	Direction	Comment
led_pattern[7:0]	Output	This LED pattern is connected to the bottom column of the user LEDs.
led_direction	Output	This signal changes the direction in which the LEDs are moving.
Reset	Output	This button resets the sfi42 interface but not the 32/33 PLL. To reset the PLL and the DUT, the SW5 pushbutton can be used.
Mgt_loopback	Output	This signal is used for loopback control for the GTX transceivers.
Rxeqmix	Output	This signal is a RXEQMIX override for the GTX transceivers.
Txdiffcntrl	Output	This signal is a TXDIFFCTRL override for the GTX transceivers.
Txpreemphasis	Output	This signal is an TXPREEMPHASIS override for the GTX transceivers.
mgt_pll_lock[1:0]	Input	This signal is a MGT PLL Lock indicator. Both bits are 1 when the MGTs are locked to the reference clock.
mgt_reset_done[3:0]	Input	This signal indicates which GTX transceiver is out of the reset state.
Mgt_cdr_lock[3:0]	Input	Each bit in this signal corresponds to the CDR for a GTX transceiver. A 1 indicates that the recovered clock is within 5,000 ppm of the reference clock.

Table 7: **DRP VIO Signals**

VIO Signal Name	Direction	Comment
drp_enable	Output	This signal must be set to 1 to perform read and write accesses. The signal must be 0 when the SFI4.2 comes out of reset.
drp_sel	Output	If set to 0, this signal selects an access to GTX_DUAL 0. If set to 1, this signal selects an access to GTX_DUAL 1.
drp_rd_wr_n	Output	If set to 1, this signal selects a read. If set to 0, this signal selects a write.
go_drp_rd_wr	Output	A 1 on this signal triggers a DRP read or write command.
drp_addr	Output	This signal outputs the DRP address for the read/write command.
drp_di	Output	This signal outputs the DRP write data.
rd_data_0	Input	This signal inputs read data from GTX_DUAL 0.
rd_data_1	Input	This signal inputs read data from GTX_DUAL 1.
drp_state	Input	This signal inputs the FSM state value. It is used for debug only.
drp_timeout_latch	Input	A red LED indicates that a DRP access from the SFI4.2 Reset FSM has timed out. A possible cause of a red LED is that the drp_enable was set while the interface came out of reset. The red LED can be cleared by pressing the clear_latches button in the RX VIO.
drp_not_enabled	Input	A red LED indicates that the user attempted a manual read or write without enabling the interface. The red LED can be cleared by pressing the clear_latches button in the RX VIO.

Table 8: **SFI4.2 TX VIO Signals**

VIO Signal Name	Direction	Comment
prbs_msb_lsb_n	Output	This signal determines if the Pseudo-Random Binary Sequence (PRBS) is generated msb or lsb first. It controls both the pattern generator and monitor.
trigger_tx_reset	Output	This signal connects to the i_TRIGGER_TX_RESET input pin.
bypass_scrambler	Output	This signal connects to the i_BYPASS_SCRAMBLING input pin.
srambler_lsb_msb_n	Output	This signal connects to the i_SCRBL_LSB_MSB_N input pin.
insert_constant_error	Output	When this signal outputs a 1, the pattern generator sends an error on every clock cycle.
insert_single_bit_error	Output	A 0 to 1 transition on this signal causes a single bit error to be generated from the pattern generator. The PRBS error count increments by 1 if the fixed pattern is selected. Otherwise, it increments by 2.
txdelay_?	Output	This signal connects to the iv_TXDELAY_? input pins.

Table 8: **SFI4.2 TX VIO Signals *(Cont'd)***

VIO Signal Name	Direction	Comment
corrupt_sync_bits	Output	This signal connects to the iv_CORRUPT_SYNC_BITS input pins.
sync_bits	Output	This signal connects to the iv_SYNC_BITS input pins.
tx_fixed_pattern	Output	This signal is the fixed pattern output by the pattern generator.
Prbs_sel	Output	This signal selects the test signal output by the pattern generator. 0: Fixed Pattern 1: PRBS15, 2: PRBS23, 3: PRBS31
tx_fifo_undfl	Input	This signal connects to the o_TX_FIFO_UNDFL output pin.
tx_fifo_ovfl	Input	This signal connects to the o_TX_FIFO_OVFL output pin.

Table 9: **SFI4.2 RX VIO Signals**

VIO Signal Name	Direction	Comment
ext_skew_en	Output	This signal connects to the i_EXT_SKEW_EN input pin.
trigger_rx_reset	Output	This signal connects to the i_TRIGGER_RX_RESET input pin.
bypass_desc	Output	This signal connects to the i_BYPASS_DESCRAMBLING input pin.
clear_prbs_error_cnt	Output	If set to 1, this signal clears the PRBS error counter.
clear_latches	Output	If set to 1, this signal clears all latched status registers.
rxpll_lock	Input	This signal connects to the o_RXPLL_LOCKED output pin.
prbs_error_latch	Input	A red LED indicates that the PRBS monitor detected an error. This LED can be cleared by pressing the clear_latches button.
rx_fifo_undfl	Input	This signal is the latched version of the o_RX_FIFO_UNDLF output. It can be cleared by pressing the clear_latches button.
rx_fifo_ovfl	Input	This signal is the latched version of the o_RX_FIFO_OVFL output. It can be cleared by pressing the clear_latches button.
block_unlock	Input	This is the latched version of the inverse of the ov_BLOCK_LOCK output. It can be cleared by pressing the clear_latches button.
sync_counter_?	Input	This signal connects to the ov_SYNC_COUNTER_? outputs.
inv_cnt_?	Input	This signal connects to the ov_INVALID_CNT_? outputs.
Skew_3_?	Input	This signal connects to the ov_SKEW_3_? outputs.

Table 9: **SFI4.2 RX VIO Signals *(Cont'd)***

VIO Signal Name	Direction	Comment
rx_dataout	Input	This signal carries data from the PRBS monitor. It is meaningful only for a fixed pattern.
prbs_error_cnt	Input	This signal carries the number of clock cycles where at least one bit error was detected on the incoming 64 bits. It can be cleared by pressing the clear_prbs_error_cnt button.

Board Setup

The ML523 development board was used to test the SFI4.2 reference design. Both the Verilog and VHDL implementations were tested. The board setup is shown in Figure 23. The SFI4.2 data lanes are all looped backed onto themselves and the receivers must all be AC coupled (DC blocks can be seen in the picture).

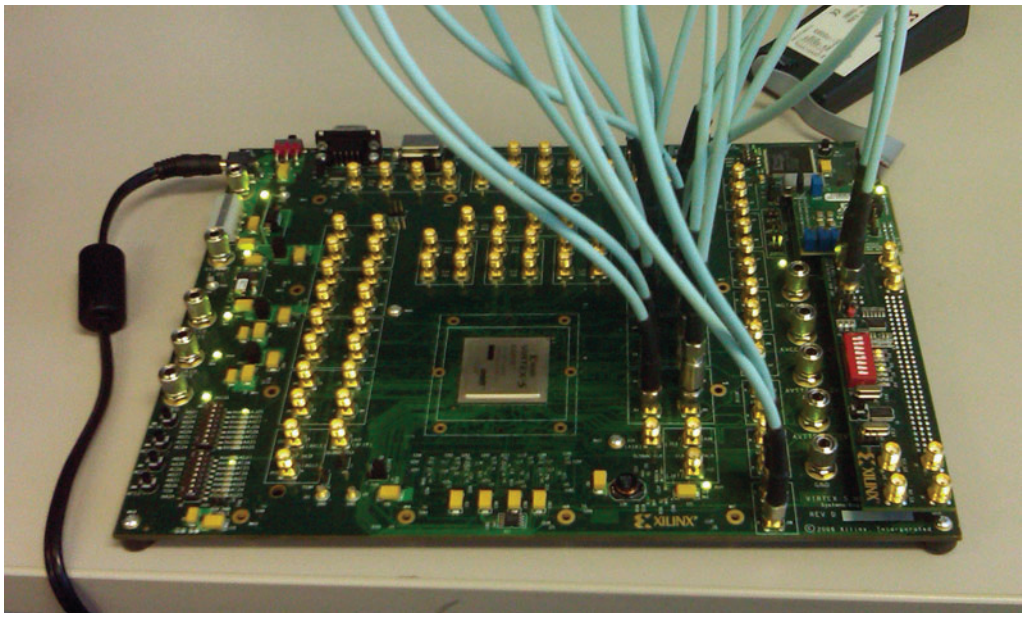


Figure 23: **ML523 Board Setup**

The lower LED column (lower left of Figure 23) is used to indicate that the reference clock is valid and that the internal PLL is locked. An external reset button located at SW5 (Pin AF14, lower left of Figure 23) is also available for use. The test design operates with a reference clock of 200 MHz. The SuperClock Module is used with the DIP switches configured as shown in Table 10.

Table 10: **DIP Switch Settings for 200 MHz Clock**

DIP Switch	Setting
N0 (DIP Switch 1)	1
N1 (DIP Switch 2)	1
N2 (DIP Switch 3)	0
M0 (DIP Switch 4)	0
M1 (DIP Switch 5)	1
M2 (DIP Switch 6)	0

Table 10: **DIP Switch Settings for 200 MHz Clock *(Cont'd)***

DIP Switch	Setting
Sel 0 (DIP Switch 7)	0
Sel 0 (DIP Switch 8)	0

External Testing

A four-channel SFI4.2 design was tested on a development board for interoperability with the Cortina CS600x OTN processor. For more information on the CS600x OTN processor, refer to the CS600x product overview [Ref 5].

References

This application note uses the following references:

1. SERDES Framer Interface Level 4 (SFI-4) Phase 2: Implementation Agreement: http://www.oiforum.com/public/documents/OIF-SFI4-02.0.pdf.
2. DS202, *Virtex-5 FPGA Data Sheet: DC and Switching Characteristics.*
3. UG198, *Virtex-5 FPGA RocketIO GTX Transceiver User Guide.*
4. UG190, *Virtex-5 FPGA User Guide.*
5. CS600x OTN Processor Product Overview http://cortina-systems.com/products/view/76.

Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
01/05/10	1.0	Initial Xilinx release.

Notice of Disclaimer