SERDES Framer Interface Level 5 (SFI-5) IP Core

User's Guide

November 2008
ipug78_01.0

Introduction

This document provides technical information about the Lattice SERDES Framer Interface Level 5 (SFI-5) IP core for the LatticeSC™ and LatticeSCM™ families of devices. The Lattice SFI-5 IP core, together with SERDES and Physical Coding Sublayer (PCS) functionality integrated in the LatticeSC/M devices, supports an inter-device interface conforming to Optical Internetworking Forum Implementation Agreement OIF-SFI5-01.02. A block diagram of the IP core is shown in Figure 1.

The SFI-5 core comes with the following documentation and files:

- User's guide
- Protected netlist/database
- Behavioral RTL simulation model
- Source files for instantiating and evaluating the core

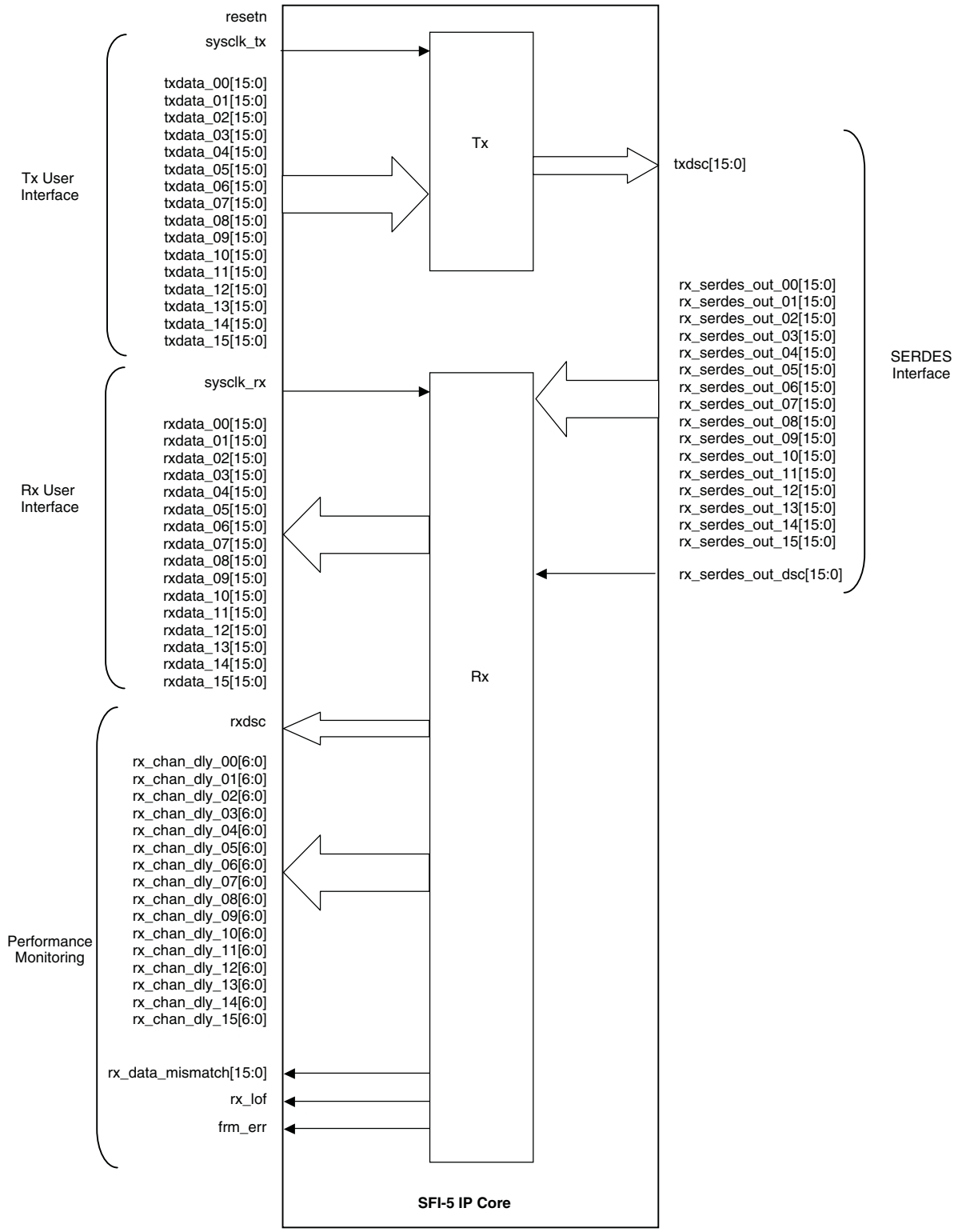
The SFI-5 IP core is available at no charge and may be directly downloaded from the Lattice website at www.latticesemi.com.

Features

- Designed to Optical Internetworking Forum Implementation Agreement OIF-SFI5-01.02
- Supports bi-directional aggregate data throughput of up to 50 Gbps
- Sixteen 16-bit wide internal receive and transmit data paths
- Data path uses 17 SERDES transceivers operating in 8-bit only mode
- Included reference design suitable for use on the Lattice Semiconductor SC-1704BGA SXI5 Evaluation Board with SERDES channels running at 2.5 Gbps
- Reference design uses the Reveal™ Logic Analyzer to observe circuit operation
- User-settable parameters to select the allowed number of framing errors for the deskew channel framer to go into or out of locked state

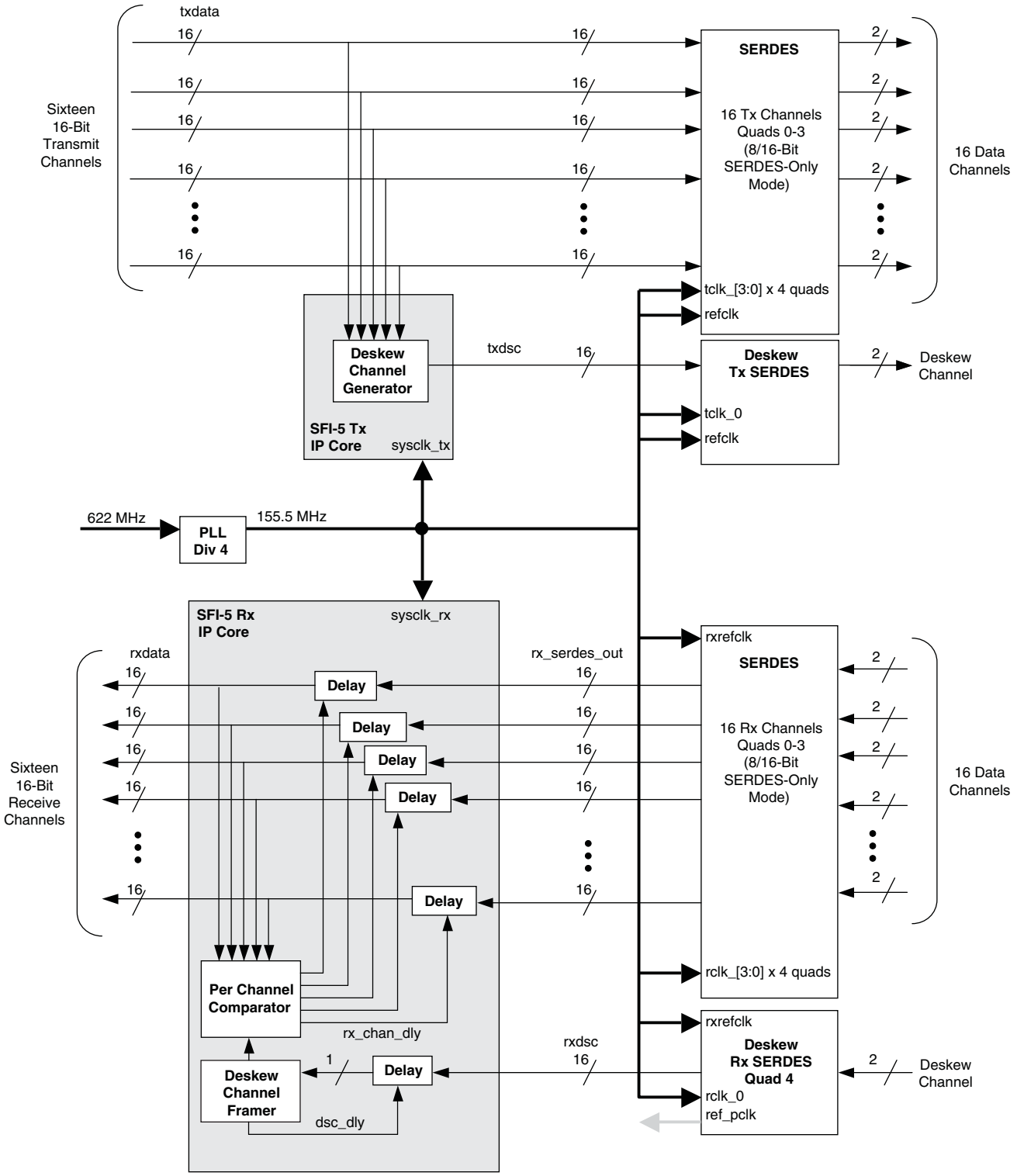
Core Block Diagram

Figure 1. SFI-5 Core Block Diagram



SFI-5 FPGA Block Diagram

Figure 2. SFI-5 FPGA Block Diagram



Signal Descriptions

Table 1. SFI-5 IP Core Input and Output Signals

Port Name	Active State	I/O Type	Description
resetn	Low	Input	Asynchronous reset signal. Resets entire core when asserted.
sysclk_tx	N/A	Input	Clock input for Tx side of the IP core. Rising edge is active. Clock frequency is 1/16th SERDES line rate.
txdata_00[15:0]	N/A	Input	Transmit data for SERDES channel 0.
txdata_01[15:0]	N/A	Input	Transmit data for SERDES channel 1.
txdata_02[15:0]	N/A	Input	Transmit data for SERDES channel 2.
txdata_03[15:0]	N/A	Input	Transmit data for SERDES channel 3.
txdata_04[15:0]	N/A	Input	Transmit data for SERDES channel 4.
txdata_05[15:0]	N/A	Input	Transmit data for SERDES channel 5.
txdata_06[15:0]	N/A	Input	Transmit data for SERDES channel 6.
txdata_07[15:0]	N/A	Input	Transmit data for SERDES channel 7.
txdata_08[15:0]	N/A	Input	Transmit data for SERDES channel 8.
txdata_09[15:0]	N/A	Input	Transmit data for SERDES channel 9.
txdata_10[15:0]	N/A	Input	Transmit data for SERDES channel 10.
txdata_11[15:0]	N/A	Input	Transmit data for SERDES channel 11.
txdata_12[15:0]	N/A	Input	Transmit data for SERDES channel 12.
txdata_13[15:0]	N/A	Input	Transmit data for SERDES channel 13.
txdata_14[15:0]	N/A	Input	Transmit data for SERDES channel 14.
txdata_15[15:0]	N/A	Input	Transmit data for SERDES channel 15.
txdsc[15:0]	N/A	Output	Data generated by the IP core to be transmitted on the SERDES deskew channel.
sysclk_rx	N/A	Input	Clock input for Rx side of the IP core. Rising edge is active. Clock frequency is 1/16th SERDES line rate.
rxdata_00[15:0]	N/A	Output	Receive data from SERDES channel 0.
rxdata_01[15:0]	N/A	Output	Receive data from SERDES channel 1.
rxdata_02[15:0]	N/A	Output	Receive data from SERDES channel 2.
rxdata_03[15:0]	N/A	Output	Receive data from SERDES channel 3.
rxdata_04[15:0]	N/A	Output	Receive data from SERDES channel 4.
rxdata_05[15:0]	N/A	Output	Receive data from SERDES channel 5.
rxdata_06[15:0]	N/A	Output	Receive data from SERDES channel 6.
rxdata_07[15:0]	N/A	Output	Receive data from SERDES channel 7.
rxdata_08[15:0]	N/A	Output	Receive data from SERDES channel 8.
rxdata_09[15:0]	N/A	Output	Receive data from SERDES channel 9.
rxdata_10[15:0]	N/A	Output	Receive data from SERDES channel 10.
rxdata_11[15:0]	N/A	Output	Receive data from SERDES channel 11.
rxdata_12[15:0]	N/A	Output	Receive data from SERDES channel 12.
rxdata_13[15:0]	N/A	Output	Receive data from SERDES channel 13.
rxdata_14[15:0]	N/A	Output	Receive data from SERDES channel 14.
rxdata_15[15:0]	N/A	Output	Receive data from SERDES channel 15.
rx_serdes_out_00[15:0]	N/A	Input	Receive data from SERDES channel 0.
rx_serdes_out_01[15:0]	N/A	Input	Receive data from SERDES channel 1.
rx_serdes_out_02[15:0]	N/A	Input	Receive data from SERDES channel 2.
rx_serdes_out_03[15:0]	N/A	Input	Receive data from SERDES channel 3.

Table 1. SFI-5 IP Core Input and Output Signals (Continued)

Port Name	Active State	I/O Type	Description
rx_serdes_out_04[15:0]	N/A	Input	Receive data from SERDES channel 4.
rx_serdes_out_05[15:0]	N/A	Input	Receive data from SERDES channel 5.
rx_serdes_out_06[15:0]	N/A	Input	Receive data from SERDES channel 6.
rx_serdes_out_07[15:0]	N/A	Input	Receive data from SERDES channel 7.
rx_serdes_out_08[15:0]	N/A	Input	Receive data from SERDES channel 8.
rx_serdes_out_09[15:0]	N/A	Input	Receive data from SERDES channel 9.
rx_serdes_out_10[15:0]	N/A	Input	Receive data from SERDES channel 10.
rx_serdes_out_11[15:0]	N/A	Input	Receive data from SERDES channel 11.
rx_serdes_out_12[15:0]	N/A	Input	Receive data from SERDES channel 12.
rx_serdes_out_13[15:0]	N/A	Input	Receive data from SERDES channel 13.
rx_serdes_out_14[15:0]	N/A	Input	Receive data from SERDES channel 14.
rx_serdes_out_15[15:0]	N/A	Input	Receive data from SERDES channel 15.
rx_serdes_out_dsc[15:0]	N/A	Input	Receive data from SERDES deskew channel.
Signals to Monitor Circuit Operation			
rxdsc[15:0]	N/A	Output	Data received on the SERDES deskew channel. This data has been delayed to provide word alignment and framing has been recovered.
rx_chan_dly_00[15:0]	N/A	Output	Delay value for receive channel 0.
rx_chan_dly_01[15:0]	N/A	Output	Delay value for receive channel 1.
rx_chan_dly_02[15:0]	N/A	Output	Delay value for receive channel 2.
rx_chan_dly_03[15:0]	N/A	Output	Delay value for receive channel 3.
rx_chan_dly_04[15:0]	N/A	Output	Delay value for receive channel 4.
rx_chan_dly_05[15:0]	N/A	Output	Delay value for receive channel 5.
rx_chan_dly_06[15:0]	N/A	Output	Delay value for receive channel 6.
rx_chan_dly_07[15:0]	N/A	Output	Delay value for receive channel 7.
rx_chan_dly_08[15:0]	N/A	Output	Delay value for receive channel 8.
rx_chan_dly_09[15:0]	N/A	Output	Delay value for receive channel 9.
rx_chan_dly_10[15:0]	N/A	Output	Delay value for receive channel 10.
rx_chan_dly_11[15:0]	N/A	Output	Delay value for receive channel 11.
rx_chan_dly_12[15:0]	N/A	Output	Delay value for receive channel 12.
rx_chan_dly_13[15:0]	N/A	Output	Delay value for receive channel 13.
rx_chan_dly_14[15:0]	N/A	Output	Delay value for receive channel 14.
rx_chan_dly_15[15:0]	N/A	Output	Delay value for receive channel 15.
dsc_dly[15:0]	N/A	Output	Delay value for receive deskew channel.
rx_data_mismatch[15:0]	High	Output	Each bit indicates that a mismatch has been detected between the receive data and the deskew channel. Each of the 16 bits corresponds to the 16 data channels. These signals remain high for one clock cycle only and are not latched.
rx_lof	High	Output	Loss of frame indicator for the deskew channel. This signal indicates that the framing pattern for the deskew channel has been lost.
frm_err	High	Output	Framing pattern error. This signal indicates that a mismatch occurred between the expected and received framing patterns. This signal goes high for one clock cycle each time a framing error occurs. This signal is not latched.

Parameter Descriptions

The SFI-5 core includes three user-settable parameters which are shown in Table 2.

Table 2. SFI-5 Configuration Parameters

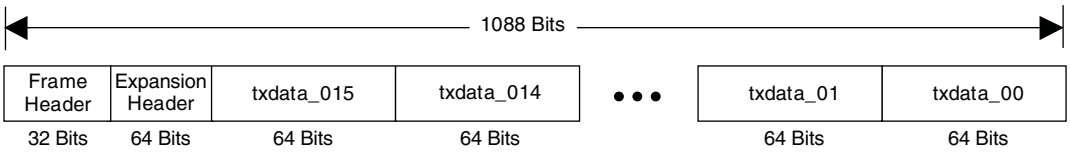
Parameter	Value Range	Default	Description
NUM_FRM_TO_LOCK	1 - 64	4	This parameter sets the number of consecutive frames which must contain the exact framing pattern (0xf6f6 followed by 0x2828) for the framer to go in-frame.
NUM_FRM_TO_UNLOCK	1 - 64	4	This parameter sets the number of consecutive frames in which the framing pattern must contain at least a 1-bit mismatch of the framing pattern for the framer to go out-of-frame and begin searching for a new framing pattern.
NUM_RX_MISMATCH_ALLOWED	1 - 64	4	This parameter sets the number of consecutive frames in which at least a 1-bit mismatch occurs between the receive data and the deskew data before a channel will begin searching for a new match.

Functional Description

The SFI-5 IP Core implements a SERDES Framer Interface Level 5, as defined by the Optical Internetworking Forum in OIF-SFI5-01.02. The SFI-5 defines a communications interface for a 40 Gbps optical link which typically consists of a Framer, FEC (Forward Error Correction) Processor, and SERDES. The purpose of the SFI-5 interface is to transmit data across multiple channels in parallel, where channels may incur different skews between the transmitter and receivers. The SFI-5 receiver delays the data received on all of the channels to match that channel which incurred the longest delay. This removes any skew variation between the channels.

As shown in Figure 2, the SFI-5 IP core has sixteen 16-bit transmit data inputs (256 signals). The Tx logic generates a 17th deskew channel (16 bits wide) which consists of four framing bytes, four bytes of expansion header, and 128 bytes of sample data. The framing pattern is a fixed pattern of 0xF6F6 followed by 0x2828. The expansion header is a fixed value of 0xAAAA followed by 0xAAAA. The data samples are sourced from the parallel transmit data where four 16-bit words are sampled sequentially from each of the 16 channels. The reference frame is shown in Figure 3.

Figure 3. SFI-5 Reference Frame



The receiver detects the framing pattern received in the deskew channel. To do this, it uses a controllable delay element, or barrel shifter. The framer increments the delay value one bit at a time until the framing pattern is found in the delayed data. The NUM_FRM_TO_LOCK parameter determines how many consecutive frames must be seen with the correct framing pattern before the framer enters the in-frame state. Once the receiver framer is locked the Rx logic can recover the original samples from each of the 16 channels, and each channel finds the necessary delay value that must be inserted into the data path such that the incoming data matches the samples from the deskew channel.

Logic External to the IP Core

Some supporting logic (not shown in Figure 2) is required for the SFI-5 IP core to work correctly. This logic is included in the IP core evaluation directory in the file <username>_top.v which may be viewed and modified by the user. This supporting logic includes a clock source, flip-flops to correctly align the parallel data, and bidirectional SERDES. Also, if desired, the supporting logic can connect the 256 input/output data signals using a different sequence than is provided in the default IP core package.

In the transmit direction, the parallel txdata inputs connect directly to the IP core, and to the transmit SERDES after being registered through a flip-flop delay. This one clock cycle delay between the IP core transmit data inputs and the transmit SERDES inputs is necessary because the deskew channel outputs are registered within the IP core, and so a one-cycle delay is needed in the txdata path external to the IP core so that the transmit data and the deskew channel are aligned at the SERDES inputs.

Before connecting the parallel transmit data and the parallel deskew channel data to the SERDES inputs, a bit reversal is performed. This is necessary so that the MSB of the parallel data is transmitted first from each SERDES output.

If desired, the top-level logic can be modified to perform data striping. In this case, instead of sending data for each of the 16 parallel txdata inputs on a different SERDES, data from each txdata input is "striped" across all SERDES channels. An example of this is shown in the top-level RTL in the evaluation directory.

Register Description

There are no registers in the SFI-5 core. All control and status information is passed between this core and the top level of the device through individual I/O ports on the core. Registers must be added by the user to the top level to control and monitor these ports. In the reference design, configuration of the SERDES is done when a bitstream is loaded into the FPGA. The values to which the SERDES registers are configured are set in the serdes.txt file located in the /impl directory. The user can enable data loopbacks internal to the SERDES by modifying the serdes.txt file and generating a new bitstream.

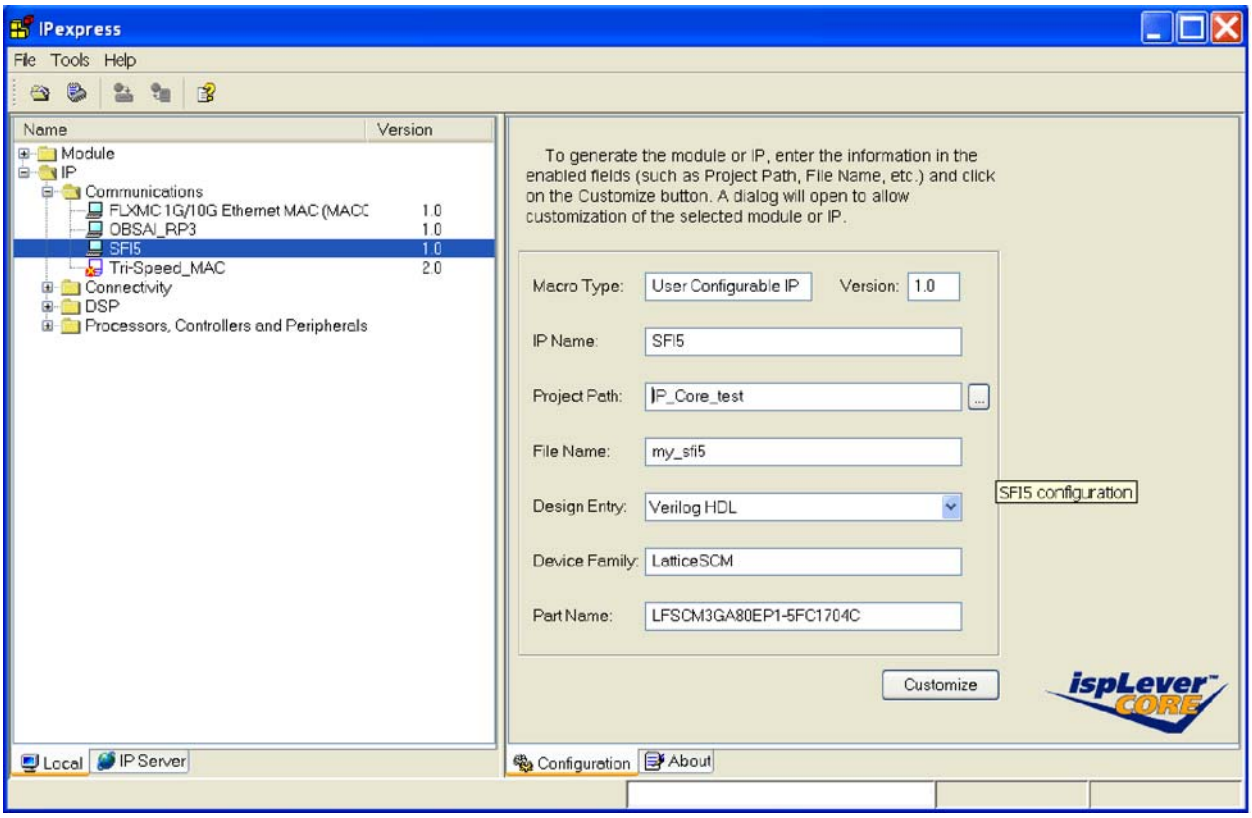
Core Generation

The SFI-5 IP core is available for download from the Lattice website at www.latticesemi.com. The IP files are automatically installed using ispUPDATE technology in any directory the user specifies.

The ispLEVER® IPexpress™ GUI window for the SFI-5 core is shown in Figure 4. To generate a specific IP core configuration the user specifies:

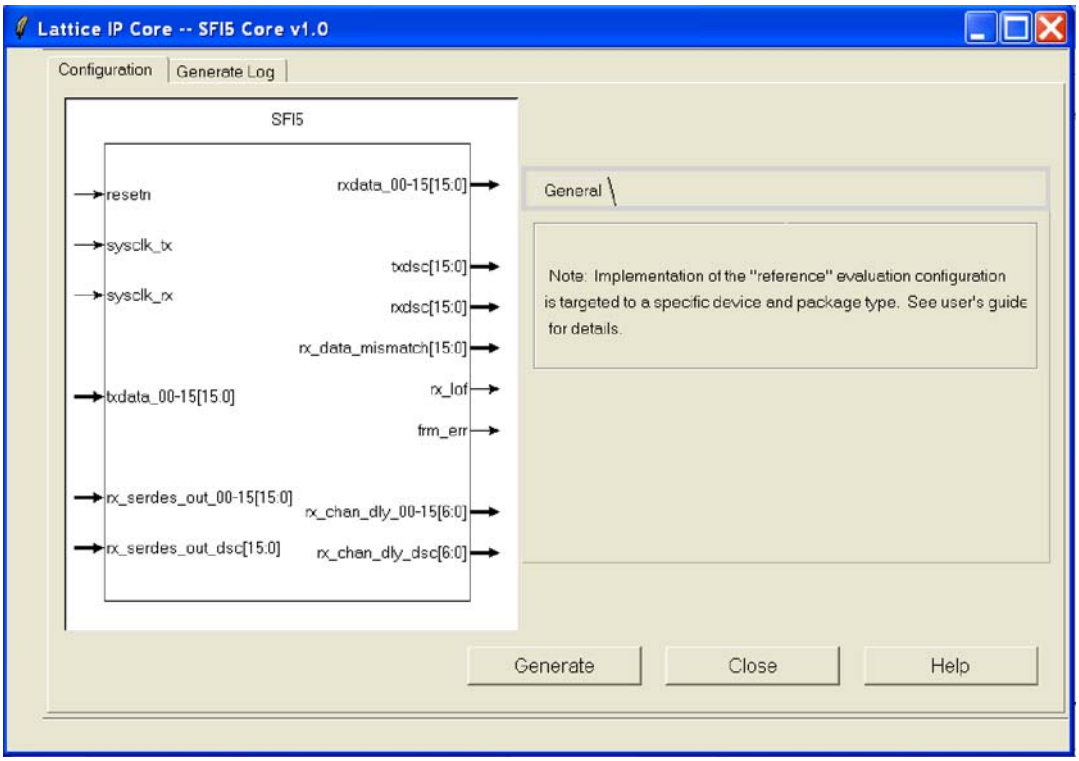
- **Project Path** – Path to the directory where the generated IP files will be loaded.
- **File Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **Design Entry Type** – Verilog HDL or VHDL.
- **Device Family** – Device family to which the IP is to be targeted. For the SFI-5 core, only the LatticeSCM™ and LatticeSC™ devices are supported.
- **Part Name** – Specific targeted part within the selected Device Family. For the SFI-5 core, only the LFSCM3GA80EP1-5fC1704C part is supported.

Figure 4. SFI-5 IPexpress GUI Window



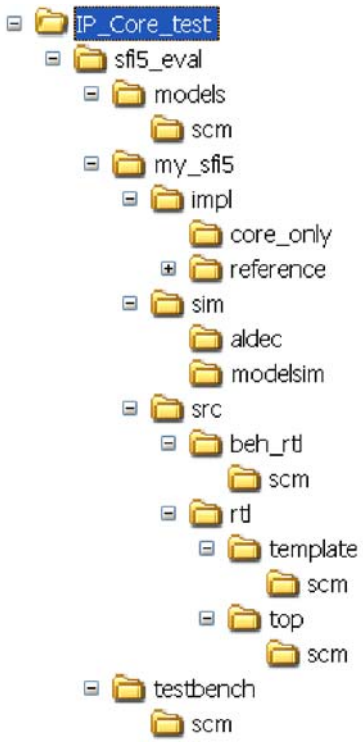
To create a custom configuration, click on the **Customize** button to display the SFI-5 IP core Configuration GUI, shown in Figure 5. There are no user selectable parameters in the IPexpress GUI for the SFI-5 IP core. The user may edit the SFI-5 parameters in the top-level RTL file.

Figure 5. SFI-5 IPexpress GUI Window



When the user clicks the **Generate** button, the IP core and supporting files are generated in the user's project directory. The directory structure of the generated files is shown in Figure 6.

Figure 6. SFI-5 IP Core Generated Directory Structure



The following files are generated in the user's project directory (\IP_Core_test in Figure 6):

- <username>.lpc – IP parameter file (may be directly modified by users).
- <username>.ngo – Synthesized and mapped IP core.
- <username>_bb.v – Black box module wrapper for synthesis.
- <username>_inst.v – Example of instantiation template to be included in the user's design.
- <username>_beh.v – Behavioral simulation model (\<username>\src\beh_rtl).

These are all of the files needed to implement and verify the SFI-5 IP core in a top-level design.

The \sfi5_eval directory is created by IPexpress the first time the core is generated and updated each time the core is regenerated. A \<username> directory is created by IPexpress each time the core is generated and regenerated each time the core with the same file name is regenerated. A separate \<username> directory is generated for cores with different names, e.g. \<sfi5_0>, \<sfi5_1>, etc.

Instantiating the Core

The generated SFI-5 IP core package includes black-box (<username>_bb.v) and instance (<username>_inst.v) templates that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file is provided in:

```
\<project_dir>\sfi5_eval\<username>\src\rtl\top\scm
```

The top-level file <usrename>_top.v is the same top-level file that is used in the simulation model described in the next section. This top-level reference may be used as the starting template for the top level of a user's complete design. Included in <username>_top.v are the IP core instance, a PLL, SERDES modules, and logic supporting the reference design configuration for use on the Lattice SXI5 evaluation board.

The file <username>_eval_top.v contains the top-level reference design example. This file instantiates the <username>_top.v, and adds a pattern generator and checker circuit.

Running Functional Simulation

A simulation environment is provided that supports both ModelSim® and Aldec® Active-HDL® simulators. The simulations use a behavioral model of the SFI-5 IP core (<username>_beh.v).

The ModelSim environment is located in **\<project_dir>\sfi5_eval\<username>\sim\modelsim**. Users may run the ModelSim simulation by doing the following:

1. Open ModelSim.
2. Under the **File** tab, select **Change Directory** and choose folder **\<project_dir>\sfi5_eval\<username>\sim\modelsim**.
3. Under the **Tools** tab, select **Tcl > Execute Macro** and execute one of the ModelSim "do" scripts shown, depending on which version of ModelSim is used (ModelSim SE or the Lattice OEM version).

The Aldec Active-HDL environment is located in **\<project_dir>\sfi5_eval\<username>\sim\aldec**. Users may run the Aldec evaluation simulation by doing the following:

1. Open Active-HDL.
2. Under the **Tools** tab, select **Execute macro**.
3. Browse to the directory **\<project_dir>\sfi5_eval\<username>\sim\aldec** and execute the Active-HDL "do" script shown.

The simulation waveform results will be displayed in the ModelSim or Aldec Active-HDL Wave window The simulation is self-checking and will report a pass or fail in the transcript window. In the Wave window, the user will see that the receive side SFI-5 framer locks and the rx_lof signal goes low. Next, each of the 16 receive channels will try to acquire a match between incoming receive data and the deskew channel. When all channels have acquired matches the 16 rx_data_mismatch signals will all be low. At this point, the SFI-5 IP core is stable. The delay values that have been inserted on each of the 17 receive channels are seen on the dsc_dly[6:0] and rx_chan_dly_[15-00][6:0] signals. The relative skew between all channels is apparent by examining the delay required to align the individual channels.

Synthesizing and Implementing the Core in a Top-Level Design

The SFI-5 IP core itself is synthesized and is provided in .ngo format when the core is generated. Users may synthesize the core in their own top-level design by instantiating the core as described previously and then synthesizing the entire design with either Synplify® or Precision® RTL Synthesis.

Two example RTL top-level configurations supporting SFI-5 core top-level synthesis and implementation are provided with the SFI-5 IP core in \<project_dir>\sfi5_eval\<username>\impl. The first is a reference design which instantiates the IP core along with the necessary top-level logic (PLL, registers, SERDES) into a test environment which contains a pattern generator and checker. The pattern generator and checker are the same modules used for the simulation environment described earlier. This configuration can be synthesized, placed and routed, and a bitstream created which can be loaded into the Lattice SXI5 evaluation board.

The second example is a core-only implementation. This example shows how to build the IP core along with the required top-level support logic (<username>_top.v), but without the pattern generator/checker included in the reference design. The core-only implementation is provided to show how to determine the FPGA resource utilization (LUT/SLICE/register count) needed for the SFI-5 core exclusive of any other logic. The core-only implementation does not represent a completely functional design.

Push-button implementation of both top-level configurations is supported via the ispLEVER project files, <username>_reference_eval.syn and <username>_core_only_eval.syn. These files are located in \<project_dir>\sfi5_eval\<username>\impl\<configuration>.

To use these project files:

1. Select **Open Project** under the **File** tab in ispLEVER.
2. Browse to the **\<project_dir\sfi5_eval\<username>\impl directory** and select either the **\core_only** or **\reference** directory in the **Open Project** dialog box.
3. Select and open either **<username>_reference_eval.syn** or **username>_core_only_eval.syn**. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Select the device top-level source file in the left-hand GUI window.
5. Implement the complete design via the standard ispLEVER GUI flow.

Hardware Evaluation Support

The SFI-5 IP core is available at no charge and may be evaluated and implemented in LatticeSC/M devices with no restrictions. The SFI-5 IP core package includes a reference design in the eval directory. This reference design is target to a Lattice SXI5 Evaluation Board. A bitstream can be generated and downloaded directly to this board for evaluation. Instructions for generating the reference design bitstream are described in the SXI5 Evaluation Board section. The reference design includes a test pattern generator/checker combination which sends a fixed non-random repeating test pattern.

SXI5 Evaluation Board

The reference design included under the \sfi5_eval directory is designed for use on the Lattice SC-1704BGA SXI5 Evaluation Board with an on-board 622 MHz oscillator. The IP core runs at 155.5 MHz, and the SERDES operate at 2.5 Gbps. The I/O constraints included in the .lpf file of the ispLEVER project will allow the generated bitstream to be downloaded directly to the evaluation board using ispVM™ System software. The pattern generator/checker circuits included in the evaluation testbench transmit a fixed repeating test pattern that has a sufficient number of 0/1 transitions to allow the SERDES to operate correctly. This reference design has been verified to work with the SERDES transmit outputs looped back to the receive inputs using a Lattice Semiconductor MSA Loopback Board.

The reference design has also been verified with a Finisar OC-768/STM-256 40Gbps transponder using a different pattern generator/checker circuit than the one included. The test pattern from the included generator is not compatible with the CDR of the 40 Gbps fiber interface. Please contact Lattice Semiconductor for a copy of the testbench that works with the 40 Gbps transponder.

The testbench operation on the evaluation board can be verified two ways. First, the pattern checker will light one of four LEDs (AV37, AW34, AP34, and AT39) if a pattern mismatch is seen between the pattern sent from the generator and the pattern received at the checker. If all four of the LEDs are off, then the received pattern which has been looped back at the SERDES interface is correct. To verify that the pattern checker is operating correctly, an error can be inserted into the transmit pattern by momentarily toggling switch AW42 from off to on. The signal from this switch is edge detected so only a single error is inserted into the transmit pattern each time the switch is toggled from off to on. All four of the LEDs listed above will illuminate momentarily when a transmit error is inserted.

The second way to verify correct circuit operation is by using the Reveal Logic Analyzer, which has been included in the ispLEVER project. Reveal allows signals within the reference design to be probed and displayed on the host PC during circuit operation. The user can use Reveal “as-is” in the provided configuration, or the user can modify the probed signals and the trigger conditions by opening the Reveal Inserter within the ispLEVER project, editing the Reveal options, generating a new Reveal core, and then using ispLEVER to synthesize, map, place, route, and generate a new bitstream. If Reveal is not desired, remove the sfi5_reference_eval.rvl file from the ispLEVER project before building the project.

Once Reveal has been included in the project and a bitstream generated, download the bitstream to the evaluation board using ispVM System software and reset the SFI-5 IP core as described in the Resetting the SERDES section of this document. At this point, Reveal can be run by following these steps:

1. Select **Reveal Logic Analyzer** under the **Tools** tab in ispLEVER or click on the **Reveal Logic Analyzer icon** on the toolbar.

2. If a **New Project** window opens (Figure 7) then enter a **Project Name** and browse to the **Project Directory**, otherwise select **New** from the **File** tab.

3. In the **Device Information** window (Figure 8), select the **Reveal Inserter File** (sfi5_reference_eval.rvl) from the project directory and click **Finish**. This will import the desired settings from the Reveal Inserter to the Reveal Analyzer, and the Reveal Analyzer will open.

4. The Reveal tool included in the package has been pre-built with two logic analyzers, LA0 for receive-side signals and LA1 for transmit-side signals. Deselect the checkbox next to LA1 on the toolbar at the top of the window (Figure 9). LA0 has been set up to trigger on one of three conditions, a pattern mismatch on one of the 16 data channels in the SFI-5 receiver, the force error signal to the pattern generator from toggle switch AW42 going active, or a loss of frame on the deskew channel in the SFI-5 receiver. If the SFI-5 is working correctly (all LEDs are off) then once the RUN button is clicked, Reveal will configure and wait for a trigger condition. The user can trigger Reveal by either toggling the force error signal to the pattern generator (switch AW42) or by clicking on either the STOP or TRIG icons on the Reveal toolbar.

5. Once triggered, Reveal will upload the captured data to the PC and display it in the Waveform View window as shown in Figure 10.

Figure 10 shows the delay which the IP core has inserted on the deskew channel needed to recover framing, and the delays on each of the 16 receive data channels needed to match samples from the deskew channel with the individual channels. The values of the rx_chan_dly_[15-00] signals represent the relative skews between channels. In the example of Figure 10, all 17 channels have a relative skew within seven clock cycles of each other (min. delay = 23, max. delay = 30).

Figure 7. Reveal "New Project" Window

Figure 8. Reveal "Device Information" Window

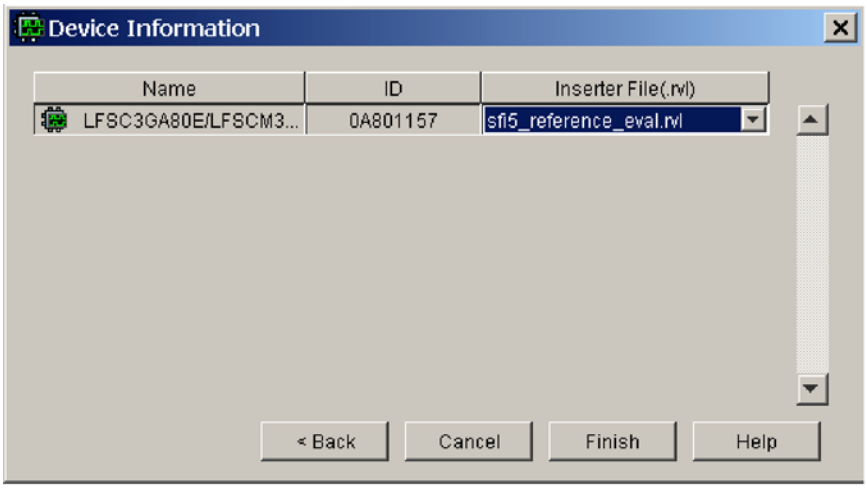


Figure 9. Reveal Trigger Signal Setup Window

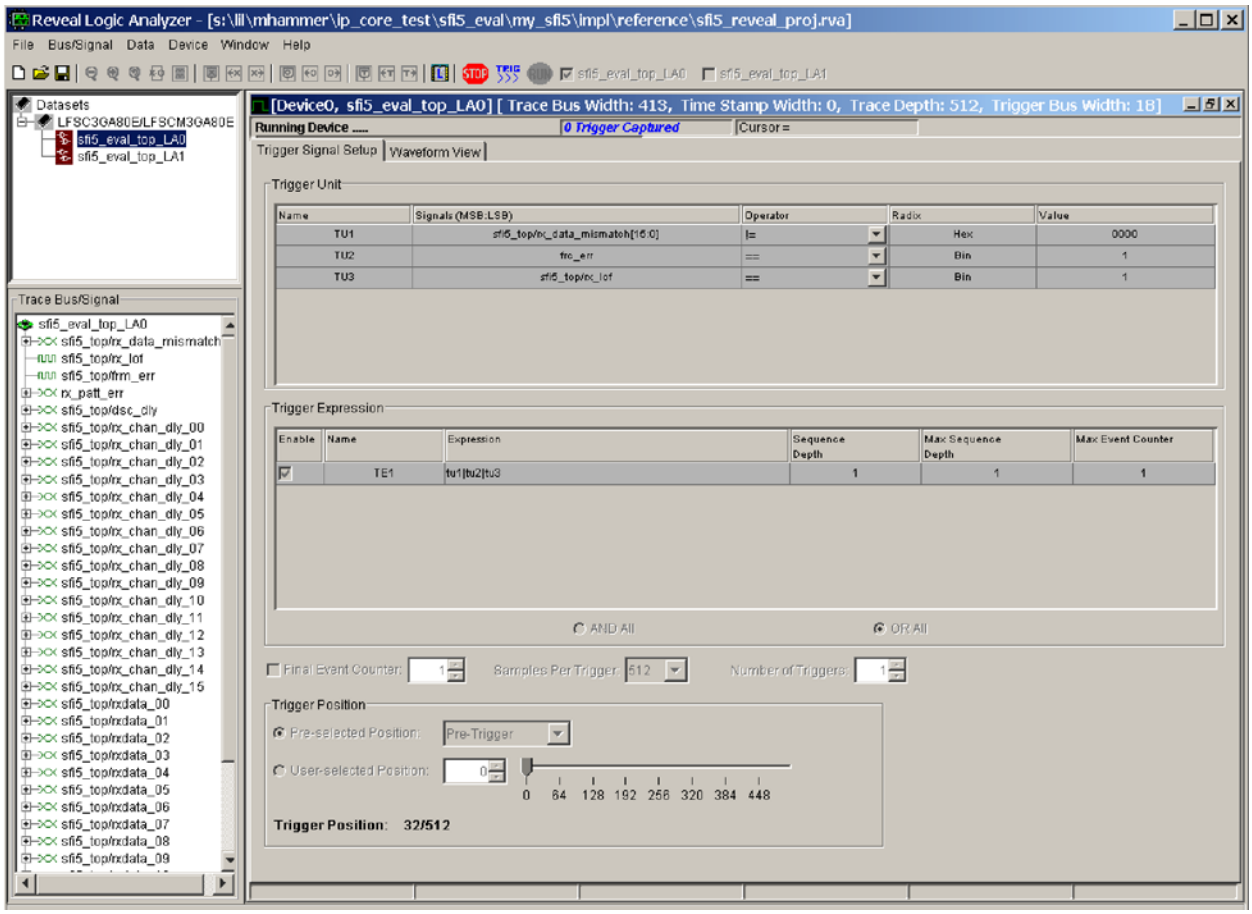
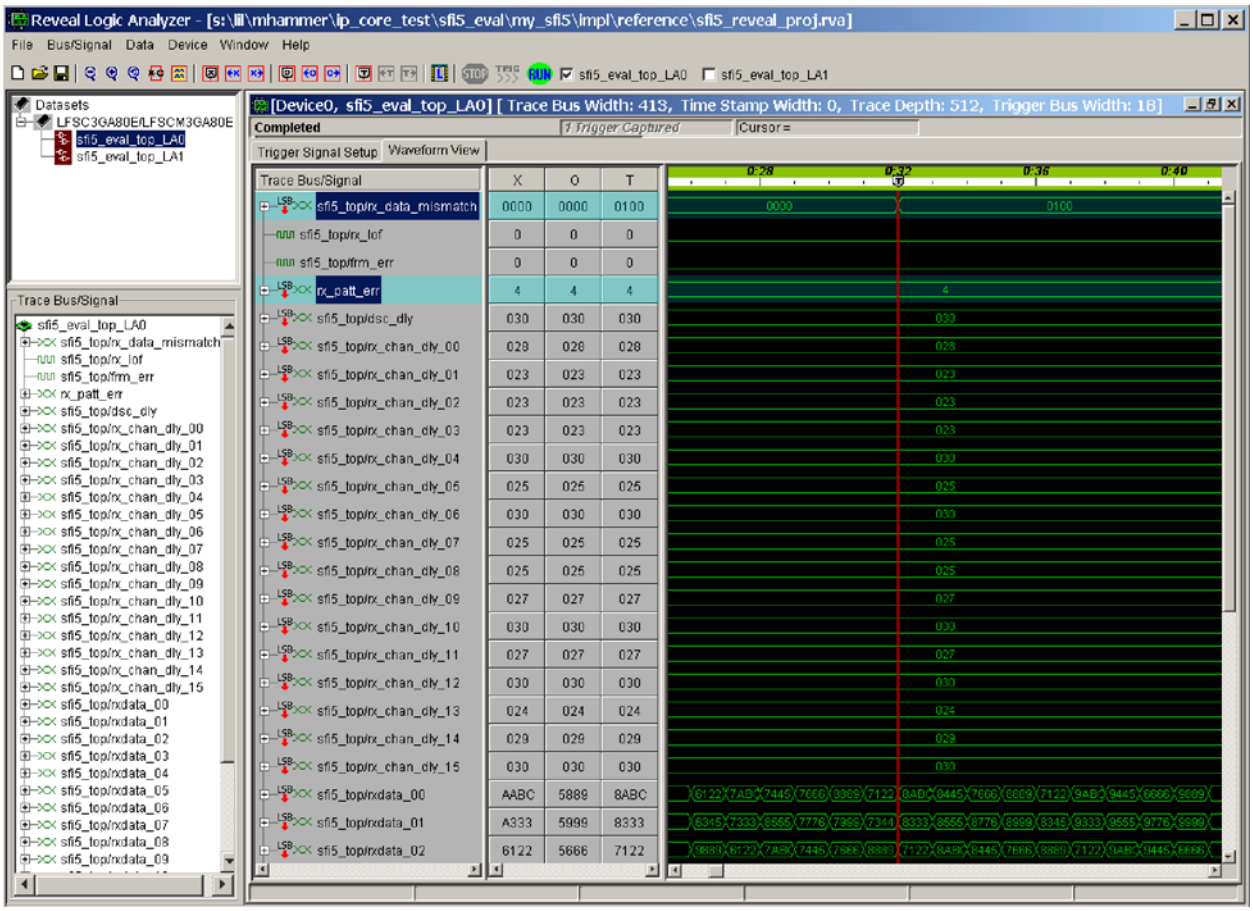


Figure 10. Reveal Waveform View Window



It is easy to use Reveal to probe or trigger on other signals in the design. For example, to change the default trigger expression from triggering on all three TU output to trigger on the frc_err signal (from switch AW42) going active, change the TE1 trigger expression from “tu1 | tu2 | tu3” to “tu2” and click the **RUN** button. To trigger on the framing pattern in the received deskew channel data, follow these steps:

1. Open the Reveal Inserter tool and open the existing Reveal project (will open by default).

2. In the **Trigger Signal Setup Window** (Figure 9) click **ADD** under the **TU** section to add TU4.

3. Drag the signals **sfi5_top/rxdsc[15:0]** from the **Design Hierarchy** frame to the **TU4 Signals** box.

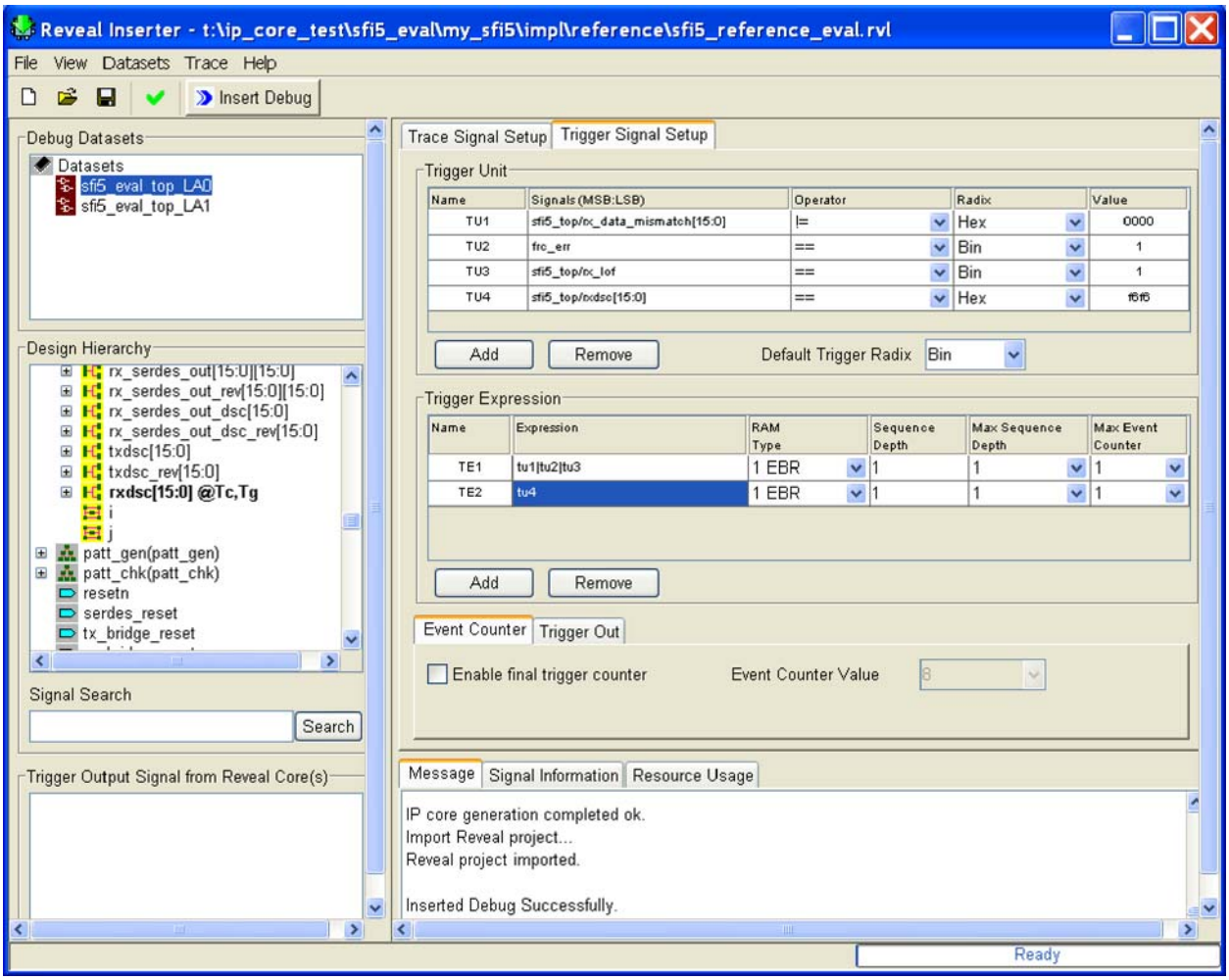
4. Set the radix for TU4 to **hex** using the pull-down menu and set the value to **f6f6**.

5. Then, either change the TE1 trigger expression to **tu4** or add a second trigger expression (TE2) and set that to **tu4**.

6. Finally, click on the **INSERT DEBUG** button and verify in the Message window that the Reveal core has been inserted successfully.

The new Reveal core will automatically be included in the existing ispLEVER project. Double-click on **Generate Bitstream Data** in the ispLEVER Project Navigator to rebuild the project and create a new bitstream. Download the bitstream to the evaluation board and run the Reveal Analyzer as described earlier. Since the Reveal signals have changed, it will be necessary to open Reveal as a new project.

Figure 11. Reveal Inserter



Resetting the SERDES

The reference design has four reset signals brought to switches on the evaluation board. The global reset, or GSRN, connects to a momentary push-button switch. The three SERDES resets (serdes_rst, tx_bridge_rst, and rx_bridge_rst) connect to toggle switches. The SERDES must be reset after the receive data stream to the SERDES receivers is stable, and the resets should be activated in the order listed. The tx_bridge_rst is needed to align all five of the transmit SERDES quads (17 channels) so that the minimum transmit skew is achieved. The rx_bridge_rst needs to be activated last for the SFI-5 IP core to function correctly.

All five transmit SERDES quads need to see the tx_bridge_rst at the same time to achieve the minimum skew between transmit SERDES. To insure that this happens, the reset signal from the switch has been registered internal to the FPGA and fans out to five individual registers, one per SERDES quad. These five registers have been hard-located using preferences in the .lpf file so that they are located as near to their respective SERDES as possible. The same is true for both the serdes_rst and the rx_bridge_rst signals. The preferences included in the reference design are:

```
LOCATE COMP "FF_RX_BRIDGE_RST_QUAD0" SITE "R14C8B" ;
LOCATE COMP "FF_RX_BRIDGE_RST_QUAD1" SITE "R14C21B" ;
LOCATE COMP "FF_RX_BRIDGE_RST_QUAD2" SITE "R14C35B" ;
LOCATE COMP "FF_RX_BRIDGE_RST_QUAD3" SITE "R14C49B" ;
LOCATE COMP "FF_RX_BRIDGE_RST_QUAD4" SITE "R14C80B" ;
```

```
LOCATE COMP “FF_TX_BRIDGE_RST_QUAD0” SITE “R14C8D” ;
LOCATE COMP “FF_TX_BRIDGE_RST_QUAD1” SITE “R14C21D” ;
LOCATE COMP “FF_TX_BRIDGE_RST_QUAD2” SITE “R14C35D” ;
LOCATE COMP “FF_TX_BRIDGE_RST_QUAD3” SITE “R14C49D” ;
LOCATE COMP “FF_TX_BRIDGE_RST_QUAD4” SITE “R14C80D” ;
LOCATE COMP “FF_SERDES_RST_QUAD0” SITE “R14C8C” ;
LOCATE COMP “FF_SERDES_RST_QUAD1” SITE “R14C21C” ;
LOCATE COMP “FF_SERDES_RST_QUAD2” SITE “R14C35C” ;
LOCATE COMP “FF_SERDES_RST_QUAD3” SITE “R14C49C” ;
LOCATE COMP “FF_SERDES_RST_QUAD4” SITE “R14C80C” ;
```

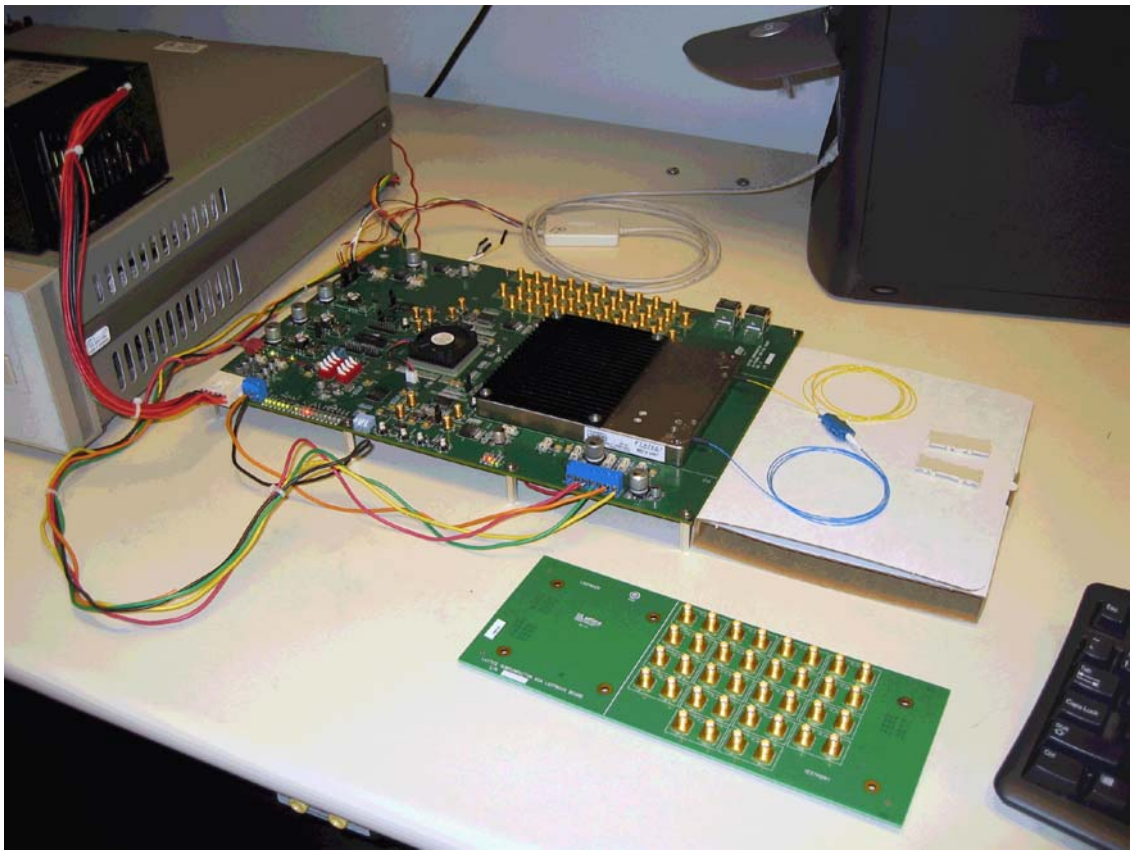
The user can change these constraints as needed as long as they keep the final reset register as near to the SERDES as possible.

On the evaluation board, the SERDES resets are driven from toggle switches. The switches should be kept in the inactive state (off), and momentarily switched to the active state one-by-one in the following order:

- serdes_rst – Switch AM34 - LED BA40
- tx_bridge_rst – Switch AV41
- rx_bridge_rst – Switch AK30 - LED AY41

The serdes_rst and rx_bridge_rst signals are brought out of the FPGA to LEDs. When each reset is activated, the LED will light. Once the circuit has been reset and is operating correctly, if the data stream to the receive side SERDES inputs is interrupted, it is typically necessary to again activate the rx_bridge_rst momentarily.

Figure 12. Hardware Evaluation Setup Using the Lattice SXI5 Evaluation Board with Finisar 40 Gbps Transponder Module (Lattice MSA Loopback Board Also Shown)



References

The following documents provide more information on implementing this core:

- FPGA Design with ispLEVER Tutorial
- ispLeverCORE IP Module Evaluation Tutorial

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
November 2008	01.0	Initial release.

Appendix for LatticeSC/M FPGAs

Table 3. Performance and Resource Utilization[1]

Mode	SLICEs	LUTs	Registers	External Pins[2]	sysMEM™ EBRs	f_{MAX} (MHz)
Core-only, without PLL and top-level logic	4858	3203	2922	N/A	0	185
Reference design configuration with PLL, top-level registers, and pattern gen/chk	4858	5995	6138	N/A	22	195

1. Performance and utilization characteristics are created from Lattice's ispLEVER 7.1 software with Synplify synthesis and targeting a LatticeSCM LFSCM3GA80EP1-5FC1704C device. When using this IP core in a different density, speed, or grade within the LatticeSCM family or in a different software version, performance may vary.
2. The SFI-5 core itself does not use any external pins. However, in an application the core is used together with IODDR, I/O buffers, and SERDES integrated into the LatticeSCM series FPGA. Thus, the application implementing the SFI-5 specification will utilize I/O pins.

Ordering Part Number

The SFI-5 IP core is available free of charge and may be downloaded directly from the Lattice web site at www.latticesemi.com. The IPexpress software tool can be used to help generate new configurations of this IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help system. For more information on the ispLEVER design tools, visit the Lattice website.

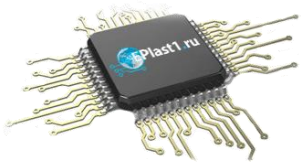
Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.



Как с нами связаться

Телефон: 8 (812) 309 58 32 (многоканальный)
Факс: 8 (812) 320-02-42
Электронная почта: org@eplast1.ru
Адрес: 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.