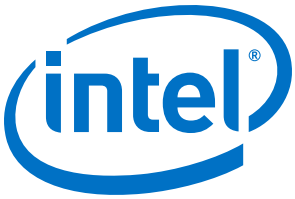


Serial Lite IV Intel® Agilex™ FPGA IP Design Example User Guide

Updated for Intel® Quartus® Prime Design Suite: **19.3**

IP Version: **1.1.0**

Subscribe

Send Feedback

UG-20242 | 2019.09.30

Latest document on the web: **PDF** | **HTML**

Contents

1. About the Serial Lite IV Intel® Agilex™ FPGA IP Design Example User Guide

This user guide provides features, usage guidelines, and functional description of the Serial Lite IV Intel® FPGA IP design examples using E-tile transceivers in Intel Agilex® devices.

Intended Audience

This user guide is intended for:

- Design architects to make IP selection during system level design planning phase.
- Hardware designers when integrating the IP into their system level design.
- Validation engineers during system level simulation and hardware validation phase.

Related Documents

The following table lists other reference documents which are related to the Serial Lite IV Intel FPGA IP.

Table 1. Related Documents

Reference	Description
Serial Lite IV Intel FPGA IP User Guide	This user guide provides IP features, architecture description, steps to generate, and guidelines to design the Serial Lite IV Intel FPGA IP using the E-tile transceivers.
Serial Lite IV Intel Stratix® 10 Design Example User Guide	This document provides features, usage guidelines, and functional description of the Serial Lite IV Intel FPGA IP design examples in Intel Stratix 10 devices.
E-tile Hard IP User Guide: E-tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IPs	This document describes the features, functionality, and guidelines of the E-Tile Hard IP for Ethernet and E-Tile CPRI PHY Intel FPGA IP cores in Intel Stratix 10 and Intel Agilex devices.
Intel Agilex Device Data Sheet	This document describes the electrical characteristics, switching characteristics, configuration specifications, and timing for Intel Agilex devices.
E-Tile Transceiver PHY User Guide	This document describes the features, functionality, and guidelines of the E-Tile Transceiver PHY in Intel Stratix 10 and Intel Agilex devices.

Acronyms and Glossary

Table 2. Acronym List

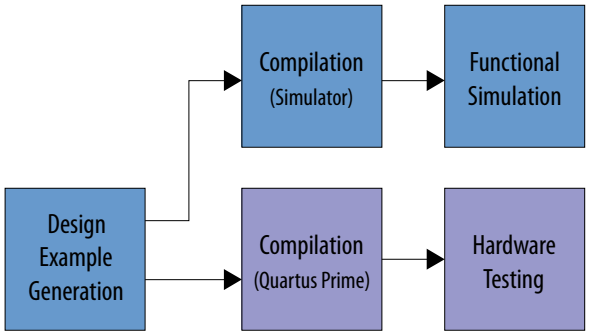
Acronym	Expansion
CW	Control Words
RS-FEC	Reed-Solomon Forward Error Correction
PMA	Physical Medium Attachment
TX	Transmitter
RX	Receiver
PAM4	Pulse-Amplitude Modulation 4-Level
NRZ	Non-return-to-zero
PCS	Physical Coding Sublayer
MII	Media Independent Interface
XGMII	10Gigabit Media Independent Interface

2. Quick Start Guide

The Serial Lite IV Intel FPGA IP provides the capability of generating design examples for selected configurations.

The Serial Lite IV Intel FPGA IP design example for Intel Agilex devices features a simulating testbench and a hardware design that supports compilation and hardware testing. The design example demonstrates loopback mode designs in basic or full mode for duplex configurations.

Figure 1. Development Stages for the Design Example



2.1. Design Example Block Diagram

Figure 2. High-level Block Diagram for Intel Agilex Design Examples

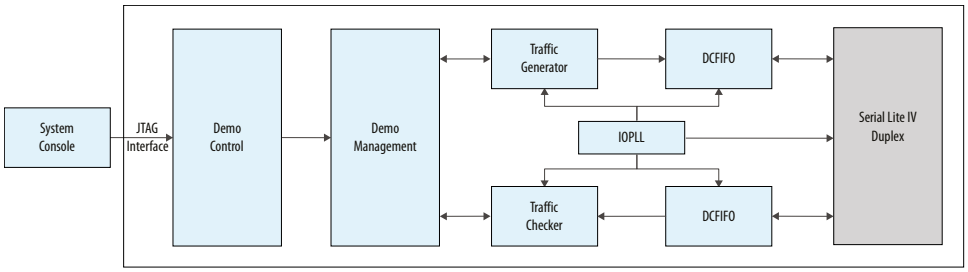


Table 3. Design Example Components

Components	Description
Serial Lite IV IP variation	The Serial Lite IV IP variation accepts data from the traffic generator and formats the data for transmission. It also receives data from the link, strips the headers, and presents it to the traffic checker for analysis.

continued...

Components	Description
	You generate the IP using the parameter editor in the Intel Quartus® Prime Pro Edition software.
System console	The system console is a Intel Quartus Prime tool that provides a user-friendly interface for you to do first-level debugging and monitor status on the IP, and the traffic generator and checker.
Demo control	The demo control module consists of Avalon-MM pipeline bridges connected to the transceiver reconfiguration and the demo management interfaces separately. The design also instantiates JTAG master, parallel input/output (PIO), and ISSP (In-system Source and Probe) modules for system console debugging purposes.
Demo management	The demo management module implements control and status registers (CSRs) to control and monitor the design operation. This includes CSRs to monitor and log errors that occur during the operation.
User clock—IOPLL	For Intel Agilex E-tile devices, the design example uses an IOPLL to generate a user clock for transmitting data into the Serial Lite IV IP. The design uses the `iopll_ref_clk` clock signal as an IOPLL reference clock to connect to the clock generator. The `iopll_ref_clk` should have the same frequency as the `pll_refclk` and come from the same clock module.
Traffic generator	The traffic generator generates traffic in a deterministic format to verify that data is transmitted correctly across the link.
Traffic checker	The traffic checker performs inspections to verify that the received data conforms to the expected format.
Dual-clock FIFO (DCFIFO)	The DCFIFO blocks handle data streaming and control signals during the crossing between the user clock domain to the core clock domain and vice-versa.

Related Information

- Traffic Generator on page 13
- Traffic Checker on page 13
- DCFIFO on page 13

2.2. Software Requirements

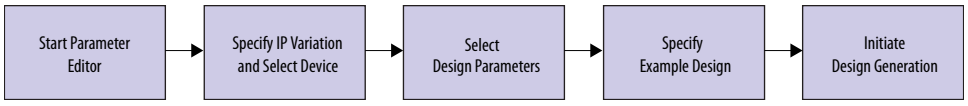
Intel uses the following software to test the design examples in a Linux system:

- Intel Quartus Prime Pro Edition software version 19.3
- ModelSim*(1), Xcelium*, NCSim (Verilog only), or VCS*/VCS MX simulator

2.3. Generating the Design

You can use the IP parameter editor in the Intel Quartus Prime Pro Edition software to generate the design example.

Figure 3. Generating the Design Flow



(1) ModelSim - Intel FPGA Edition is not supported for this IP.

To generate the design example from the IP parameter editor:

1. In the **Tools ➤ IP Catalog**, locate and select **Serial Lite IV Intel FPGA IP**. The IP parameter editor appears.
2. Specify the parameters for your design.
3. Click the **Generate Example Design** button.

The software generates all design files in the sub-directories. You need these files to run simulation, compilation, and hardware testing.

Note: Hardware design example and toolkit for Intel Agilex devices will only be available in future Intel Quartus Prime Pro Edition releases.

2.3.1. Design Example Parameters

The Serial Lite IV IP parameter editor includes an *Example Design* tab for you to specify certain parameters before generating the design example.

Figure 4. Example Design Tab

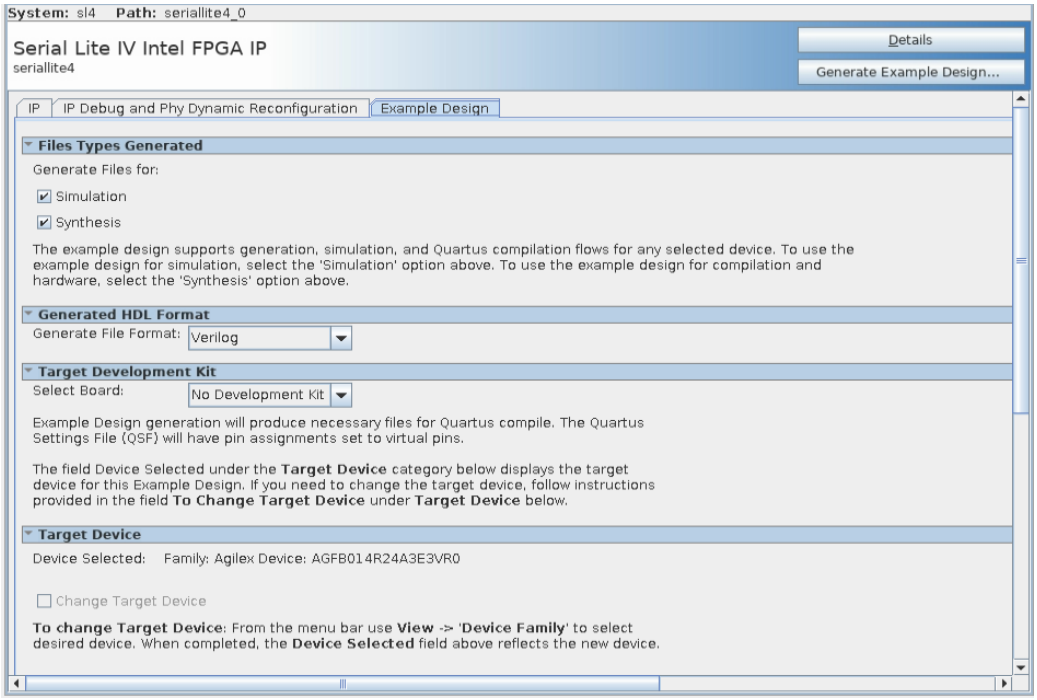


Table 4. Parameters in the Example Design Tab

Parameter	Description
Generate Files for Simulation	When selected, the IP generates the necessary files for simulating the design example.
Generate Files for Synthesis	When selected, the IP generates the synthesis files. Use these files to compile the design in the Intel Quartus Prime Pro Edition software for hardware testing.

continued...

Parameter	Description
Generate File Format	The format of the RTL files for simulation—Verilog or VHDL.
Select Board	Supported hardware for design implementation. When you select an Intel FPGA development board, the **Target Device** is the one that matches the device on the Development Kit. If this menu is grayed out, there is no supported board for the options that you select. **Custom Development Kit**: This option allows you to test the design example on a third party development kit with Intel FPGA IP device, a custom designed board with Intel FPGA IP device, or a standard Intel FPGA IP development kit not available for selection. You can also select a custom device for the custom development kit. **No Development Kit**: This option excludes the hardware aspects for the design example.
Change Target Device	Select a different device grade for Intel FPGA IP development kit. For device-specific details, refer to the device datasheet on the Intel FPGA website.

2.3.2. Directory Structure

The Intel Quartus Prime Pro Edition software generates the design example files in the following folders:

- `<user_defined_design_example_directory>/ed_sim`
- `<user_defined_design_example_directory>/ed_synth`

The following diagrams show the directories that contain the generated files for the design example.

Figure 5. Directory Structure for Intel Agilex Serial Lite IV Design Example

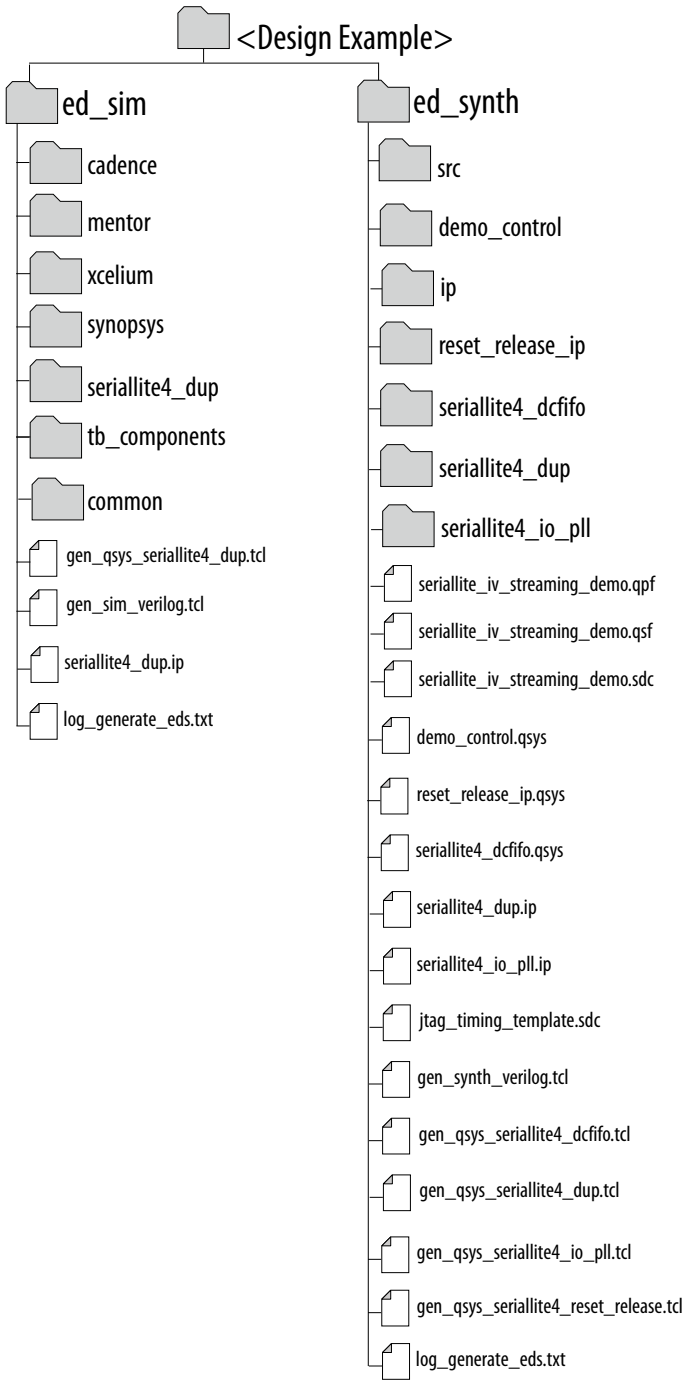
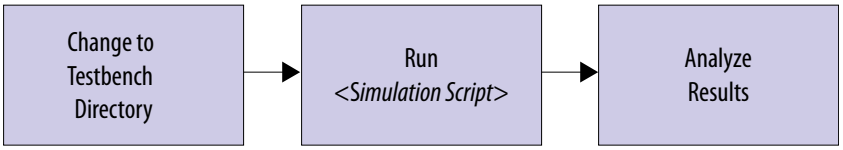


Table 5. Directory and File Description for Design Example Folder

Directory/File	Description
ed_sim/tb_components	The folder that contains the testbench files.
ed_sim/common	The folder that contains the .tcl scripts for all the simulators.
ed_sim/cadence ed_sim/mentor ed_sim/xcelium ed_sim/synopsys/vcs	The folder that contains the simulation script. It also serves as a working area for the simulator.
ed_sim/seriallite4_dup	The folder that contains the design example simulation source files.
ed_sim/seriallite4_dup.ip	IP-XACT representation of the design.
ed_synth/seriallite_iv_streaming_demo.qpf	Intel Quartus Prime Pro Edition project file.
ed_synth/seriallite_iv_streaming_demo.qsf	Intel Quartus Prime Pro Edition settings file.
ed_synth/seriallite_iv_streaming_demo.sdc	Synopsys Design Constraints (SDC) file.
ed_synth/src	The folder that contains the design example synthesizable components.
ed_synth/src/seriallite_iv_streaming_demo.v	Design example top-level HDL.
ed_synth/demo_control	The folder for each synthesizable component including Platform Designer-generated IPs, such as Demo Management and Demo Control modules.

2.4. Compiling and Simulating the Design

The design example testbench simulates your generated design.



1. Change the working directory to *<example_design_directory>*/ed_sim/*<simulator>*

   Note: In Serial Lite IV Intel FPGA IP version 1.1.0, the ModelSim simulator does not capture the top-level IP signals in the waveform. To capture the top-level IP signals, add ld_debug command in ed_sim/mentor/run_tb.tcl file.

2. Run the simulation script for the simulator of your choice.

Table 6. Testbench Simulation Scripts

Simulator	File Directory	Command
ModelSim	*<variation name>*seriallite4_0_example_design/ed_sim/mentor	do run_tb.tcl

continued...

Simulator	File Directory	Command
Note: ModelSim - Intel FPGA Edition is not supported for this IP.		
VCS	<*variation name*>seriallite4_0_example_design/ed_sim/synopsys/vcs	sh run_tb.sh
VCS MX	<*variation name*>seriallite4_0_example_design/ed_sim/synopsys/vcsmx	sh run_tb.sh
NCSim	<*variation name*>seriallite4_0_example_design/ed_sim/cadence	sh run_tb.sh
Xcelium	<*variation name*>seriallite4_0_example_design/ed_sim/xcelium	sh run_tb.sh

3. When simulation is complete, you can now analyze the results and verify the design. A successful simulation ends with the following message, "Test Passed."

```
#    ***************************** Data Forwarding Test Completed
**************************
#
#    ********************************** Test Completed
**********************************
#
#    End time                       = 534579600
#
#    Total words tranferred         = 10000
#
#    Number of bursts               = 0
#
#    Random number generator seed   = 1756255697
#
#    Link Latency                   = 434 ns
#
#    ************************************ Test Passed
************************************
```

3. Detailed Description for Serial Lite IV Design Example

This design example demonstrates the functionality of data streaming using basic and full mode.

You can specify the parameters settings of your choice and generate the design example.

The design example is available only in duplex mode.

Note: Hardware design example and toolkit for Intel Agilex devices will only be available in future Intel Quartus Prime Pro Edition releases.

3.1. Features

The design example enables you to demonstrate and test the features of the Serial Lite IV IP.

The design example includes the following features:

- Provides basic and full transmission modes
  - Basic mode—This is a pure streaming mode where data is sent without the start-of-packet, empty cycle, and end-of-packet to increase bandwidth. The IP takes the first valid data as the start of a burst.
  - Full mode—This is a packet transfer mode. This mode sends a burst and a sync cycle at the start and end of a packet as delimiters.
- Support transceiver data rate up to:
  - 56 Gbps per lane with a maximum of eight PAM4 lanes in a single link(2)
  - 28 Gbps per lane with a maximum of 16 NRZ lanes(2)
- Support data error reporting includes PCS error, loss of alignment, CRC error, and data invalid error on RX datapath
- Traffic checker for data verification and lane deskew verification
- Debugging sequence

(2) The maximum data rate that the IP can achieve depends on the device speed grade. Refer to *Intel Agilex Device Data Sheet* for more information about maximum data rate for each device speed grade.

3.2. Design Example Components

3.2.1. Traffic Generator

The traffic generator generates traffic in a deterministic format to verify that data is transmitted correctly across the link. The traffic consists of sets of sample words, one for each lane on the link, that are presented to the source user interface.

If you configure the Serial Lite IV IP in full mode, the traffic generator also asserts the `tx_is_usr_cmd` signal at random to signify the packet is from user data for testing purposes. The Serial Lite IV IP asserts the `num_valid_bytes_eob` control signal to signify the number of valid bytes of the burst packet.

Figure 6. Traffic Generator Sample Word Format

This figure shows the format of the sample words generated for each lane.

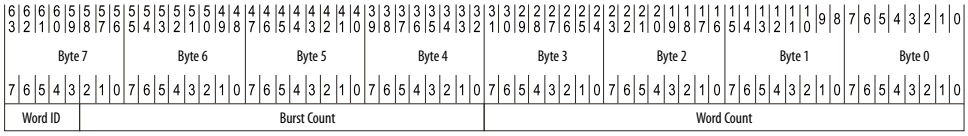


Table 7. Traffic Generator Sample Word Fields

Field	Bits	Description
Word ID	63–59	Contains a static value to distinguish which 64-bit word on the user interface that this sample was presented on. The Word ID value ranges from 0 to (lanes – 1).
Burst Count	58–32	Tracks the number of bursts used to transfer the sample data. This field value starts with one after reset and is incremented each time the `start_of_burst` signal is asserted on the source user interface.
Word Count	31–0	Tracks the number of valid sample words that have been transferred, across all bursts, to the source user interface.

3.2.2. Traffic Checker

The traffic checker performs the following inspections to verify that the received data conforms to the expected format:

- Checks each sample word to verify that the expected word ID was received.
- Checks each sample word to verify that the word count value is higher than the word count value from the last valid sample word.
- Verifies that lane de-skew has been properly performed by validating that the word count and burst count values from the sample word are the same as the values received from the adjacent lane.
- If the `start_of_burst` signal is asserted on the user interface, verifies that the burst count value in the current sample word is higher than the burst count value from the last valid sample word. Otherwise, it verifies that the burst count value has not changed.

3.2.3. DCFIFO

The design uses two DCFIFO blocks at both TX and RX paths.

The DCFIFO blocks handle data streaming and control signals during the crossing between the user clock domain to the core clock domain and vice-versa.

Table 8. TX and RX DCFIFO Configuration

Parameter	Value
lpm_width	(Number of lanes x 64)+32
lpm_numwords	64

The format of the data that passes through the FIFO is similar to format generated by the traffic generator.

Figure 7. Data Format

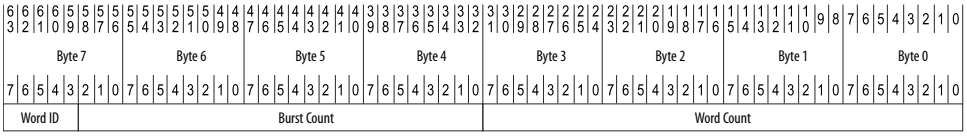


Table 9. Control Signals

The table lists how the IP concatenates the TX and RX control signals with the data bus signals and passes through the data.

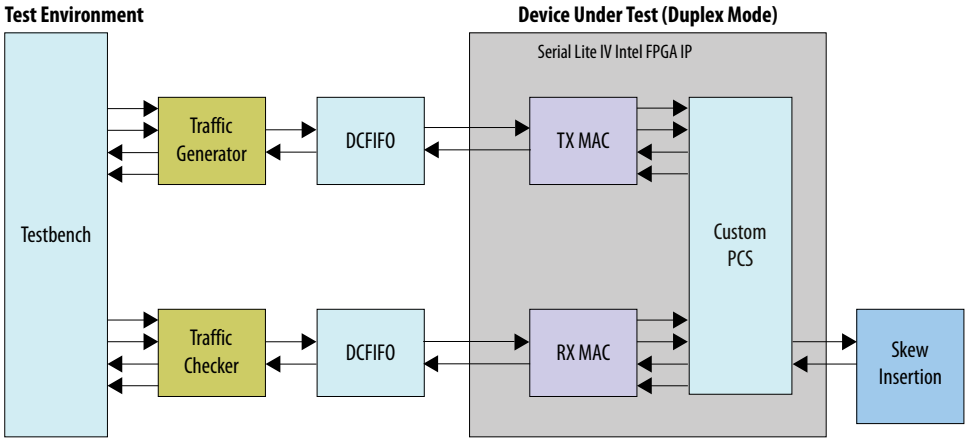
Control DCFIFO Data Out Bit	Signal	Description
[20]	tx_valid rx_valid	Indicates TX or RX data is valid for Full and Basic modes.
[19]	tx_start_of_packet rx_start_of_packet	Indicates the start of a TX or RX data packet. For Full mode only.
[18]	tx_end_of_packet rx_end_of_packet	Indicates the end of a TX or RX data packet. For Full mode only.
[17:10]	tx_channel rx_channel	The channel number for data being transmitted or received on the current cycle number. For Full mode only.
[9:5]	tx_empty rx_empty	Indicates the number of non-valid words in the final burst of the TX or RX data. For Full mode only.
[4:1]	tg_tx_num_valid_bytes_eob tc_rx_num_valid_bytes_eob	Indicates the number of valid bytes in the last word of the final burst. For Full mode only.
[0]	tg_tx_is_usr_cmd tc_rx_is_usr_cmd	Initiates a user-defined information cycle. • Full mode: Must coincide with tx_startofpacket or rx_startofpacket • Basic mode: Not supported.

Related Information

- FIFO Intel FPGA IP User Guide
- Serial Lite IV Intel FPGA IP User Guide: Serial Lite IV Intel FPGA IP Interface Signals

3.3. Simulation

Figure 8. Example Testbench (Duplex) for Intel Agilex E-tile Devices



The simulation test cases demonstrate streaming of 10,000 sample words from the traffic generator to the Serial Lite IV TX core, and externally loopback to the RX core. The words are either separated into different bursts or continuously transferred in a single burst and the transfer modes are randomized by the testbench.

The simulation test case performs the following steps:

1. Initializes and configures Serial Lite IV IP, traffic generator and traffic checker.
2. Traffic generator generates data and starts data transmission.
3. Logs and displays link up status and burst information.
4. Traffic checker verifies received data and stops transmission.
5. Testbench logs and displays test result and test information.

3.3.1. Simulation Result for Basic Mode

The generated example testbench is dynamic and has the same configuration as the IP.

In basic mode, the traffic generator generates 10,000 words and transferred to the IP once the TX and RX links are established. The bursts are then loopback to the RX MAC and verified by the traffic checker. In the simulation results, you can find the following information:

- TX and RX link status
- Configuration settings of the IP
- Number of words transferred per burst
- Test result with total number of words transferred and link latency value.

The following is a sample of the basic mode simulation result:

```
#    Waiting for TX Link Up
#
#    Phy TX Lanes Stable asserted at time 354081776
#
#    Phy EHIP ready asserted at time 475222181
```

```
#
#    TX link_up asserted at time 475228320
#       TX user_clock frequency              = 3.731343e+02 MHz
#       RX user_clock frequency              = 3.731343e+02 MHz
#
#    ************************************ Test Started
**********************************
#
#    Tests started at time 475239040
#
#    LANES                          = 4
#
#    Streaming Mode                 = BASIC
#
#    SRL4 Align Period              = 128
#
#    RSFEC Enable                   = 0
#
#    PER LANE CRC ENABLE Enable     = 0
#
#    ****************************** Data Forwarding Test Initialize
****************************
#
#    Waiting for RX Link Up
#
#    Phy block lock asserted at time 495699771
#
#    Phy RX PCS Ready asserted at time 495731094
#
#    RX link_up asserted at time 495786600
#
#    ****************************** Data Forwarding Test Started
****************************
#
#    Test Mode:            Burst
#    User Stall Insertion: Disabled
#
#
#    Traffic Generator: 98 sample burst started at time 495797320
#
#    Traffic Generator: 17 sample burst started at time 496110880
.
.
.
#    Traffic Generator: 96 sample burst started at time 533156520
#
#    Traffic Generator: 85 sample burst started at time 533515640
#
#    ****************************** Data Forwarding Test Completed
****************************
#
#    ************************************ Test Completed
**********************************
#
#    End time                       = 534579600
#
#    Total words tranferred         = 10000
#
#    Number of bursts               = 0
#
#    Random number generator seed   = 1756255697
#
#    Link Latency                   = 434 ns
#
#    ************************************ Test Passed
**********************************
```

3.3.2. Simulation Result for Full Mode

In full mode, the traffic generator generates 10,000 words of transfer in a random number of burst to the IP once the TX and RX links are established. The words are loopback to the RX MAC and verified by the traffic checker. In the simulation results, you can find the following information:

- TX and RX link status
- Configuration settings of the IP
- Number of sample words sent per burst
- Test result with total number of words transferred, number of bursts, and link latency value.

The following is a sample of the full mode simulation result:

```
#    Waiting for TX Link Up
#
#    Phy TX Lanes Stable asserted at time 354081776
#
#    Phy EHIP ready asserted at time 475222181
#
#    TX link_up asserted at time 475228320
#       TX user_clock frequency              = 3.731343e+02 MHz
#       RX user_clock frequency              = 3.731343e+02 MHz
#
#    ************************************ Test Started
**********************************
#
#    Tests started at time 475239040
#
#    LANES                          = 4
#
#    Streaming Mode                 = FULL
#
#    SRL4 Align Period              = 128
#
#    RSFEC Enable                   = 0
#
#    PER LANE CRC ENABLE Enable     = 0
#
#    ****************************** Data Forwarding Test Initialize
****************************
#
#    Waiting for RX Link Up
#
#    Phy block lock asserted at time 495699771
#
#    Phy RX PCS Ready asserted at time 495731094
#
#    RX link_up asserted at time 495786600
#
#    ****************************** Data Forwarding Test Started
****************************
#
#    Test Mode:            Burst
#    User Stall Insertion: Disabled
#
#
#    Traffic Generator: 98 sample burst started at time 495797320
#
#    Traffic Generator: 17 sample burst started at time 496110880
.
.
.
#    Traffic Checker: Burst start descriptor read at time 534212440
#                     Sync value 242
```

```
#                     Sample number 9907
#
#    Traffic Checker: Burst end descriptor read at time 534534040
#                     Inter-burst interval 7
#                     Empty value 0
#                     Sample number 10000
#
#    ***************************** Data Forwarding Test Completed
**************************
#
#    ********************************** Test Completed
**********************************
#
#    End time                       = 534810080
#
#    Total words tranferred         = 10000
#
#    Number of bursts               = 196
#
#    Random number generator seed   = 1756255697
#
#    Link Latency                   = 0 ns
#
#    ********************************** Test Passed
**********************************
```

3.4. Error Handling

The Serial Lite IV IP detects error conditions, and the behaviors in response to these error conditions.

Table 10. Error Condition Behavior

In this table, N represents the number of lanes.

Signal	Width	Location	Direction	Clock Domain	Error Indication
tx_error	5	Top-level signal	Output	tx_core_clkout	Not used.
rx_error	(*N**2*2)+3 (PAM4 mode) (*N**2)*3 (NRZ mode)	Top-level signal	Output	rx_core_clkout	When asserted, indicates that error conditions occur in the RX datapath. • [(*N**2+2):*N*+3] = Indicates PCS error for specific lane. • [*N*+2] = Indicates alignment error. Re-initialize lane alignment if this bit is asserted. • [*N*+1]= Indicates data is forwarded to the user logic when user logic is not ready. • [*N*] = Indicates loss of alignment. • [(*N*-1):0] = Indicates the data contains CRC error.

continued...

Signal	Width	Location	Direction	Clock Domain	Error Indication
tx_adaptation_fifo_full	1	Top-level TX DCFIFO signal	Output	TX user clock	This vector indicates the write domain TX buffer is full and cannot accept data.
rx_adaptation_fifo_full	1	Top-level TX DCFIFO signal	Output	TX user clock	This vector indicates the write domain RX buffer is full and cannot accept data.
readfull	1	Top-level RX DCFIFO signal	Output	RX user clock	This vector indicates the read domain buffer is full and cannot accept data.

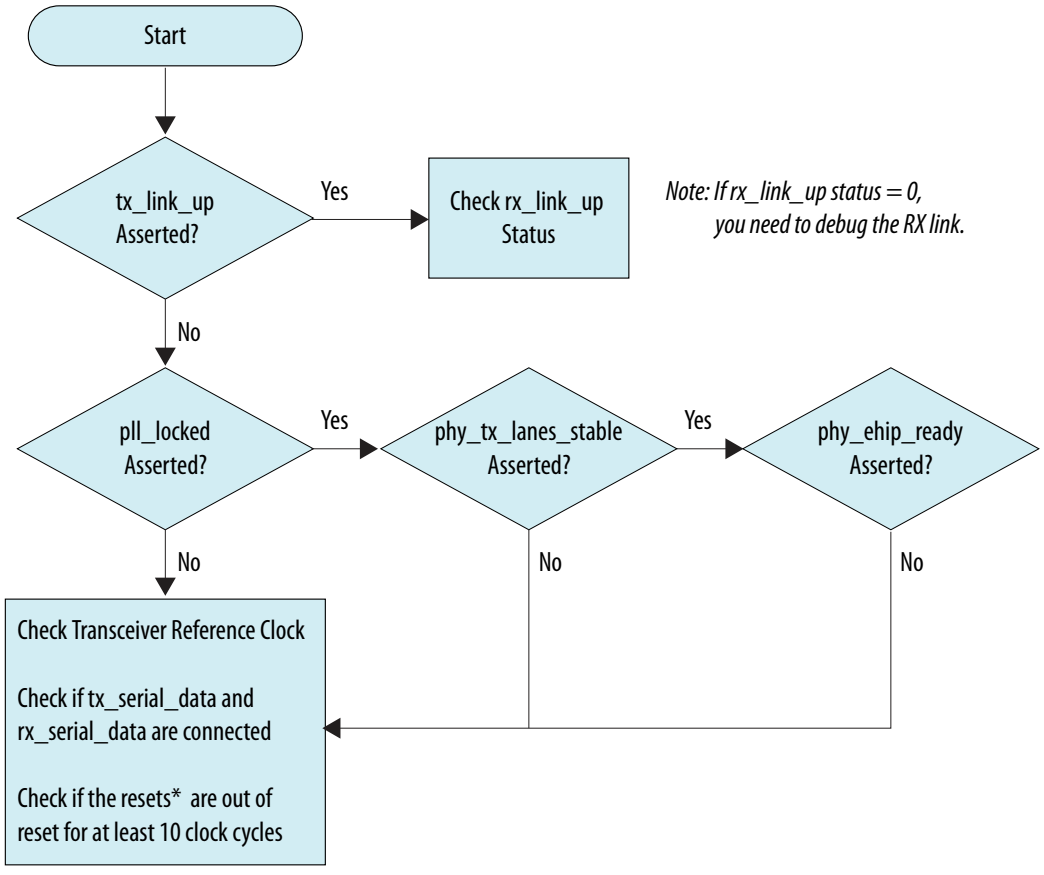
Related Information

FIFO Intel FPGA IP User Guide

3.5. Link Debugging Sequence

The Serial Lite IV IP provides a link debugging sequence for TX and RX that you can use when debugging your design.

Figure 9. TX Link Debugging Flowchart

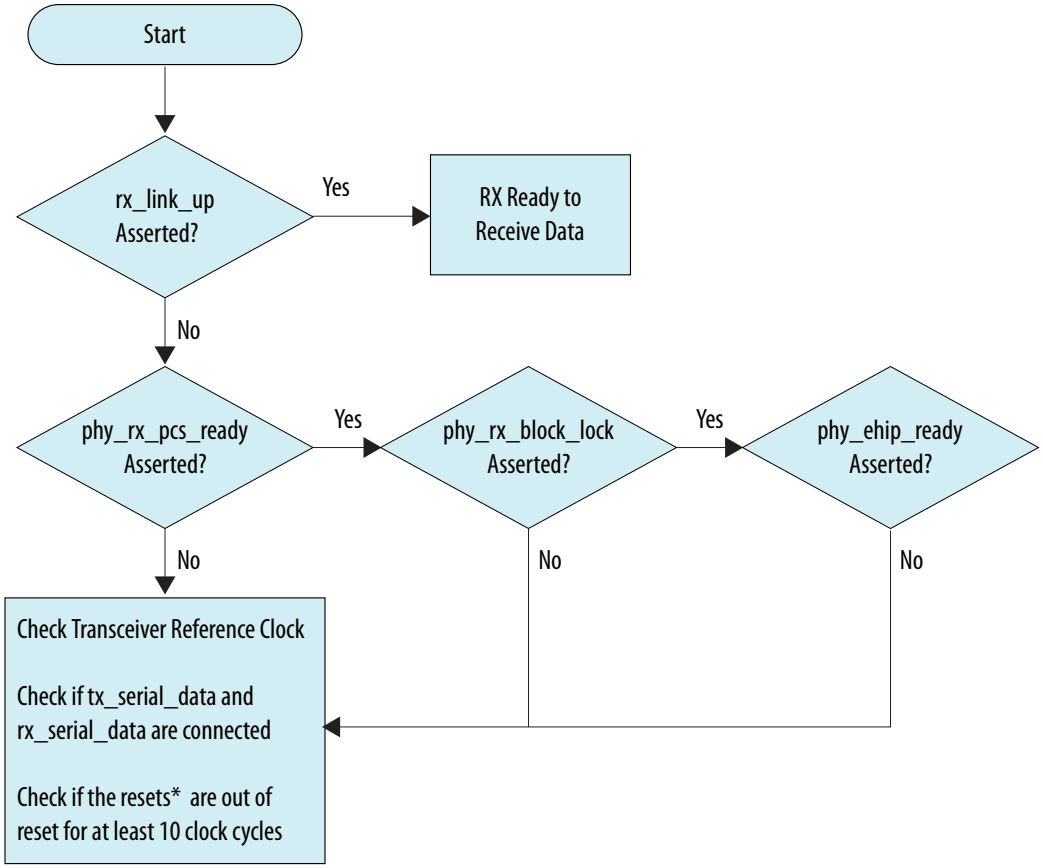


* Resets = tx_pcs_fec_phy_reset_n, rx_pcs_fec_phy_reset_n, and reconfig_reset

Table 11. TX Link Debugging Signals

Signal	Location	Description
tx_link_up	Top-level TX signal	The IP asserts this signal to indicate that the initialization sequence is complete, and the IP is ready to transmit the data.
tx_pll_locked	Top-level PHY signal	This active-high signal indicates that the transceivers are locked to the reference clock.
phy_tx_lanes_stable	Top-level PHY signal	The IP asserts this signal when TX datapath is ready to send data.
phy_ehip_ready[(n*2)-1:0]	Top-level PHY signal	The IP asserts this signal after the tx_pcs_fec_phy_reset_n and tx_pcs_fec_phy_reset_n signals deassert to indicate that the custom PCS has completed internal initialization and is ready for transmission.

Figure 10. RX Link Debugging Flowchart



* Resets = tx_pcs_fec_phy_reset_n, rx_pcs_fec_phy_reset_n, and reconfig_reset

Table 12. RX Link Debugging Signals

Signal	Location	Description
`rx_link_up`	Top-level RX signal	The IP asserts this signal to indicate that the initialization sequence is complete, and the IP is ready to receive data.
`phy_rx_pcs_ready[(n*2)-1:0]`	Top-level PHY signal	The IP asserts this signal when RX datapath is ready to receive data.
`phy_rx_block_lock[(n*2)-1:0]`	Top-level PHY signal	The IP asserts this signal to indicate the 66b block alignment has completed for the lanes.

Related Information

Serial Lite IV Intel FPGA IP User Guide: Serial Lite IV Intel FPGA IP Interface Signals

4. Document Revision History for the Serial Lite IV Intel Agilex FPGA IP Design Example User Guide

Document Version	Intel Quartus Prime	IP Version	Changes
2019.09.30	19.3	1.1.0	Initial release.