SerialLite III Streaming IP Core Design Example User Guide for Intel Arria 10 Devices

Updated for Intel® Quartus® Prime Design Suite: 17.1
1 Quick Start Guide

The SerialLite III Streaming IP core provides the capability of generating design examples for selected configurations.

Figure 1. Development Stages for the Design Example

Related Links
Intel Arria 10 SerialLite III Streaming IP Core Design Example User Guide Document Archives on page 35
Provides a list of user guides for previous versions of the SerialLite III Streaming IP Core IP Core Design Example.

1.1 Directory Structure

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

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

The following diagrams show the directories that contain the generated files for the design examples.
Figure 2. Directory Structure for Intel Arria® 10 SerialLite III Streaming Design Example

![Figure 2. Directory Structure for Intel Arria® 10 SerialLite III Streaming Design Example](image)

Table 1. Directory and File Description for Design Example Folder

<table>
<thead>
<tr>
<th>Directory/File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ed_sim/tb_components</td>
<td>The folder that contains the testbench files.</td>
</tr>
<tr>
<td>ed_sim/cadence</td>
<td>The folder that contains the simulation script. It also serves as a working area for the simulator.</td>
</tr>
<tr>
<td>ed_sim/mentor</td>
<td></td>
</tr>
<tr>
<td>ed_sim/seriallite_iii_streaming</td>
<td>The folder that contains the design example simulation source files.</td>
</tr>
<tr>
<td>ed_sim/synopsys/vcs or ed_sim/synopsys/vcsmx</td>
<td></td>
</tr>
<tr>
<td>ed_synth/seriallite_iii_streaming_demo.qpf</td>
<td>Quartus project file.</td>
</tr>
<tr>
<td>ed_synth/seriallite_iii_streaming_demo.qsf</td>
<td>Quartus settings file.</td>
</tr>
<tr>
<td>ed_synth/seriallite_iii_streaming_demo.sdc</td>
<td>Synopsys Design Constraints (SDC) file.</td>
</tr>
<tr>
<td>ed_synth/src</td>
<td>The folder that contains the design example synthesizable components.</td>
</tr>
<tr>
<td>ed_synth/src/seriallite_iii_streaming_demo.v</td>
<td>Design example top-level HDL.</td>
</tr>
<tr>
<td>ed_synth/src/seriallite_iii_streaming/synth/seriallite_iii_streaming.v</td>
<td>Design example DUT top-level files.</td>
</tr>
<tr>
<td>ed_synth/src/demo_control</td>
<td>The folder for each synthesizable component including Platform Designer generated IPs, such as demo_mgmt and demo_control.</td>
</tr>
<tr>
<td>ed_hwtest</td>
<td>The folder that contains the design example hardware setup files.</td>
</tr>
<tr>
<td>ed_hwtest/Readme.txt</td>
<td>Instruction file to download the generated design example on the development kit.</td>
</tr>
<tr>
<td>ed_hwtest/build_demo_control.sh</td>
<td>A script to generate demo control and PLL IPs with NIOS II processor.</td>
</tr>
</tbody>
</table>

continued...
1.2 Design Example Components

**Figure 3.** Design Example Block Diagram

### Directory/File

<table>
<thead>
<tr>
<th>Directory/File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ed_hwtest/master_export.v</td>
<td>User interface Verilog design file.</td>
</tr>
<tr>
<td>ed_hwtest/master_export_hw.tcl</td>
<td>Component description file for master export custom IP.</td>
</tr>
<tr>
<td>ed_hwtest/software</td>
<td>The folder that contains scripts to download the demo_control program into NIOS II processor and open an interactive terminal to run the design example.</td>
</tr>
</tbody>
</table>

**1.3 Generating the Design**

You can use the SerialLite III Streaming IP core parameter editor in the Intel Quartus Prime software to generate the design example.

**Figure 4.** Procedure
1.3.1 Procedure

This is a general procedure on how to generate the design example.

To generate the design example from the IP parameter editor:

1. In the IP Catalog (Tools > IP Catalog), locate and select **SerialLite III Streaming**. The IP parameter editor appears.
2. Specify a top-level name and the folder for your custom IP variation, and the target device. Click **OK**.
3. Select a design from the **Presets** library. When you select a design, the system automatically populates the IP parameters for the design.
   
   **Note:** If you select another design, the settings of the IP parameters change accordingly.

4. Specify the parameters for your design.
5. Click the **Generate Example Design** button.

The software generates all design files in the sub-directories. These files are required to run simulation, compilation, and hardware testing.

1.3.2 Design Example Parameters

The SerialLite III Streaming IP parameter editor includes an **Example Design** tab for you to specify certain parameters before generating the design example.

<table>
<thead>
<tr>
<th>Table 2. Parameters in the Example Design Tab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>Select Design</td>
</tr>
<tr>
<td>Generate Files for</td>
</tr>
</tbody>
</table>
### 1.3.3 Presets

Standard presets allow instant entry of pre-selected parameter values in the **IP** and **Example Design** tabs. You can select the presets at the lower right window in the parameter editor.

The parameter values chosen for the presets belong to the group of supported SerialLite III Streaming IP configurations for design example generation. You can select one of the presets available for your target device to quickly generate a design example without having to manually set each parameter in the **IP** tab and verifying that the parameter matches the supported configurations set.

There are four preset settings available in the library that support Duplex, Sink and Source modes:

- Advanced Clocking Mode 2x10G
- Advanced Clocking Mode 6x12.5G
- Standard Clocking Mode 2x10G
- Standard Clocking Mode 6x12.5G

<table>
<thead>
<tr>
<th>Presets</th>
<th>Advanced Clocking Mode 2x10G</th>
<th>Advanced Clocking Mode 6x12.5G</th>
<th>Standard Clocking Mode 2x10G</th>
<th>Standard Clocking Mode 6x12.5G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direction</td>
<td>Duplex, Sink, and Source</td>
<td>Duplex, Sink, and Source</td>
<td>Duplex, Sink, and Source</td>
<td>Duplex, Sink, and Source</td>
</tr>
<tr>
<td>Number of lanes</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

**Table 3. Parameter Settings for Intel Arria 10 Design Example Presets**

---

*continued...*
### Presets

<table>
<thead>
<tr>
<th>Presets</th>
<th>Advanced Clocking Mode 2x10G</th>
<th>Advanced Clocking Mode 6x12.5G</th>
<th>Standard Clocking Mode 2x10G</th>
<th>Standard Clocking Mode 6x12.5G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meta frame length in words</td>
<td>200</td>
<td>8191</td>
<td>200</td>
<td>8191</td>
</tr>
<tr>
<td>Transceiver reference clock frequency (MHz)</td>
<td>644.53125</td>
<td>312.5</td>
<td>644.531187</td>
<td>312.5</td>
</tr>
<tr>
<td>Enable M20K ECC support</td>
<td>ON and OFF</td>
<td>ON and OFF</td>
<td>ON and OFF</td>
<td>ON and OFF</td>
</tr>
<tr>
<td>Clocking Mode</td>
<td>Advanced clocking mode</td>
<td>Advanced clocking mode</td>
<td>Standard clocking mode</td>
<td>Standard clocking mode</td>
</tr>
<tr>
<td>Required user clock frequency (MHz)</td>
<td>150.8395522</td>
<td>186.4760558</td>
<td>146.484375</td>
<td>177.556818</td>
</tr>
<tr>
<td>Transceiver data rate (Gbps)</td>
<td>10.3125</td>
<td>12.5</td>
<td>10.312499</td>
<td>12.5</td>
</tr>
</tbody>
</table>

### 1.4 Simulating the Design

These general steps describe how to compile and run the design example simulation. For specific commands for each design example variant, refer to its respective section.

![Diagram](1.4.1 Procedure)

#### 1.4.1 Procedure

To compile and simulate the design:

1. Change the working directory to `<example_design_directory>example/ed_sim/<simulator>`.
2. Run the simulation script for the simulator of your choice.

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModelSim*</td>
<td>do run_tb.tcl</td>
</tr>
<tr>
<td>VCS®/VCS MX</td>
<td>sh run_tb.sh</td>
</tr>
<tr>
<td>Aldec™</td>
<td>do run_tb.tcl</td>
</tr>
<tr>
<td>NC-Sim</td>
<td>sh run_tb.sh</td>
</tr>
</tbody>
</table>

A successful simulation ends with the following message, "Test Passed."

After successful completion, you can analyze the results.
1.5 Compiling and Testing the Design

The SerialLite III Streaming IP Core parameter editor allows you to compile and run the design example on a target development kit.

Follow these steps to compile and test the design in hardware:

1. Launch the Intel Quartus Prime software and change the directory to `/ed_synth/` and open the `seriallite_iii_streaming_demo.qpf` file.

2. Click Processing > Start Compilation to compile the design. The timing constraints for the design example and the design components are automatically loaded during compilation.

3. Connect the development board to the host computer.

4. Configure the FPGA on the development board using the generated `.sof` file (Tools > Programmer).

   The Intel Quartus Prime version 15.1 only supports programming file generation for Arria 10 engineering sample devices. For more information on support for Arria 10 production devices, contact your local Intel representative or use the support link from Intel website.

The design examples target the Intel Arria 10 Transceiver Signal Integrity Development Kit.

The design includes an SDC script as well as a QSF with verified constraints in loopback mode. If you use the design example with another device or development board, you may need to update the device setting and constraints in the QSF file.

You must use correct pin constraints when using the core in simplex mode or when using more than one reconfiguration controller. The synthesized design typically includes a reconfiguration interface for at least three channels because three channels share an Avalon® Memory Mapped (Avalon-MM) slave interface, which connects to the Transceiver Reconfiguration Controller IP core. Conversely, you cannot connect three channels that share an Avalon-MM interface to different Transceiver Reconfiguration Controller IP cores or you will receive a Fitter error.

Related Links

SerialLite III Streaming IP Core User Guide
2 Detailed Description for Intel Arria 10 SerialLite III Streaming Standard Clocking Mode

These design examples demonstrate the functionalities of data streaming using standard clocking mode.

To generate the design examples, select the following preset(s):
- Standard Clocking Mode 2x10G
- Standard Clocking Mode 6x12.5G

By default, the design examples are generated as duplex core. To generate the design examples in simplex core, select Simplex for the Direction parameter.

Related Links
- Testbench on page 17
- SerialLite III Errata

2.1 Features

Features for Standard Clocking Mode 2x10G design example includes:
- Support 2 lanes with 10Gpbs transceiver data rate
- Support simplex and duplex transmission modes
- Traffic checker for data verification and lane de-skew verification
- Support CRC error injection using Nios II processor

Features for Standard Clocking Mode 6x12.5G design example includes:
- Support 6 lanes with 12.5Gpbs transceiver data rate
- Support simplex and duplex transmission modes
- Traffic checker for data verification and lane de-skew verification
- Support CRC error injection using Nios II processor

2.2 Hardware and Software Requirements

Intel uses the following hardware and software to test the example designs in a Linux system:
- Intel Quartus Prime software
- ModelSim - Intel FPGA Edition, Modelsim-SE, NC-Sim (Verilog only), or VCS simulator
- Intel Arria 10 GX Transceiver Signal Integrity Development Kit for hardware testing
2.3 Functional Description

The design examples consist of various components. The following block diagrams show the design components and the top level connections of the design examples.

**Figure 6. Design Example for Simplex Core in Standard Clocking Mode**
2.3.1 Design Example Components

The design example consists of following components:

- SerialLite III Streaming IP core variation
- ATX PLL
- Traffic generator
- Traffic checker
- Demo control
- Demo management
- Nios® II processor code

2.3.1.1 SerialLite III Streaming IP Core

The SerialLite III Streaming IP core in this variant can either accepts data from the traffic generator and format it for transmission or receive data from the link, strips the headers, and presents it to the traffic checker for analysis. The core is generated with the parameter settings you select using the parameter editor in the Intel Quartus Prime software.

2.3.1.2 ATX PLL

The ATX PLL generates transmit transceiver clock to the SerialLite III Streaming IP core.
2 Detailed Description for Intel Arria 10 SerialLite III Streaming Standard Clocking Mode

UG-20030 | 2017.11.06

Related Links

How to compensate the jitter of PLL cascading or non-dedicated clock path for Intel Arria 10 reference clock?

In previous Quartus Prime versions, reference clock from a cascaded PLL output, global clock, or core clock causes additional jitter in Intel Arria 10 SerialLite III Streaming IP core design example. If you are migrating your design from earlier versions to Quartus Prime 17.1, refer to the KDB link.

2.3.1.3 Traffic Generator

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

Figure 8. Traffic Generator Sample Word Format

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

Table 4. Traffic Generator Sample Word Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word ID</td>
<td>63–59</td>
<td>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).</td>
</tr>
<tr>
<td>Burst Count</td>
<td>58–32</td>
<td>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.</td>
</tr>
<tr>
<td>Word Count</td>
<td>31–0</td>
<td>Tracks the number of valid sample words that have been transferred, across all bursts, to the source user interface.</td>
</tr>
</tbody>
</table>

2.3.1.4 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.

2.3.1.5 Demo Control

The demo control module is a Nios II processor system, generated in Platform Designer (Standard), to control the demo hardware.
Demo control module also consists of a timer to track interrupt occurrence, Avalon-MM interface to access demo management and the SerialLite III Streaming IP PHY interface, a reset controller, an UART interface, and an Avalon Streaming (Avalon-ST) interface.

### 2.3.1.6 Demo Management

The demo management module controls the user modules interaction with the SerialLite III Streaming IP core such as enable and disable traffic generator and traffic checker, enable CRC error insertion, and provide user clock reset for SerialLite III Streaming IP core. The module also implements CSRs to control and monitor the design operation. This includes CSRs to monitor and log errors that occur during the operation.

### 2.3.1.7 Nios II Processor Code

The Nios II processor controls the options exercised in the design example. The code also enables CRAM bits for CRC-32 error injection support. The error injection support in 10G PCS is based on groups of three channels or triplets. Setting the corresponding bit for a given channel in the triplet enables CRC error injection for all of the lanes that use any channel in the given triplet.

The design example sets the bit for channel 0 that connects to lane 0 in the design example. Therefore, CRC error injection is exercisable for lane 0 only. Refer to the Nios II processor source code (`demo_control.c`) for information on setting bits for other channels.

### 2.3.2 Reset Scheme

The `mgmt_reset_n` reset signal controls the overall reset structure for the design example. This is an asynchronous and active-low signal. Asserting this signal resets the demo control module and the SerialLite III Streaming IP core. The traffic generator and traffic checker modules get reset through the demo management and the SerialLite III Streaming IP core.

The following diagrams show the reset scheme implemented in the design examples.
2.3.3 Clocking Scheme

The following diagrams show the clocking scheme for the design examples.
Figure 11. Clocking Scheme for Intel Arria 10 SerialLite III Streaming Simplex Core in Standard Clocking Mode
2.4 Simulation

The simulation test cases demonstrate continuous streaming of 2000 sample data for all lanes from traffic generator to the SerialLite III Streaming source core and externally loopback to the sink core in standard clocking mode.

The simulation test case performs the following steps:
1. Initialize and configures SerialLite III Streaming IP core, traffic generator and traffic checker.
2. Traffic generator generates data and starts data transmission.
3. Logs and display link up status and burst information.
4. Traffic checker verifies received data and stop transmission.
5. Testbench logs and displays test result and test information.

2.4.1 Testbench

If your design targets Intel Arria 10 devices, the generated example testbench is dynamic and has the same configuration as the IP.

When you choose the sink or duplex direction, the parameter editor generates an external transceiver ATX PLL for use in the Intel Arria 10 testbench.
2.5 Hardware Testing

Once you download the design and accompanying software into the FPGA, you can test the design operation through the interactive session. The interactive session provides helpful statistics, as well as controls for controlling various aspects of the design.
You can control the following operations through the interactive session by entering the option numbers listed below:

- **1) Enable Data Generator/Checker** - Enables the traffic generator and start sending out data.
- **2) Disable Data Generator/Checker** - Disables traffic generation.
- **3) Reset Source Core** - Resets the source core and traffic generator.
- **4) Reset Sink Core** - Resets the sink core and traffic checker.
- **5) Display Error Details** - Displays the error statistics.
- **6) Toggle Burst/Continuous Mode** - Resets the source and sink MACs and switches the traffic generator to generate a burst or continuous traffic stream. By default, the design example is set to burst mode. When in continuous mode, the burst count will always show 1. Disable the data generator/checker before switching mode to avoid transmission error.
- **7) Toggle CRC Error Insertion** - Turns CRC error injection off or on. By default, the design example has CRC error injection turned off.

### 2.5.1 Design Setup

The design example targets the Intel Arria 10 Transceiver Signal Integrity Development Kit.

The design includes an SDC script as well as a QSF with verified constraints in loopback mode. If you use the design example with another device or development board, you may need to update the device setting and constraints in the QSF file.

You must use correct pin constraints when using the core in simplex mode or when using more than one reconfiguration controller. The synthesized design typically includes a reconfiguration interface for at least three channels because three channels share an Avalon-MM slave interface, which connects to the Transceiver Reconfiguration Controller IP core. Conversely, you cannot connect three channels that share an Avalon-MM interface to different Transceiver Reconfiguration Controller IP cores or you will receive a Fitter error.

**Related Links**

*Altera Transceiver PHY IP Core User Guide*

More information about the Interlaken PHY core.

### 2.5.2 Error Details

These are the list of supported errors in the design example.

#### Table 5. List of Error Details

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source Error:</strong></td>
<td></td>
</tr>
<tr>
<td>Adaptation FIFO Overflow</td>
<td>To indicate source adaptation FIFO overflow error.</td>
</tr>
<tr>
<td><strong>Sink Errors:</strong></td>
<td></td>
</tr>
<tr>
<td>Adaptation FIFO Overflow</td>
<td>To indicate sink adaptation FIFO overflow error.</td>
</tr>
<tr>
<td>Loss of Alignment During Normal Operation</td>
<td>To indicate loss of alignment error (error_rx[1]).</td>
</tr>
</tbody>
</table>

**continued...**
<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meta Frame CRC Errors</td>
<td>To indicate CRC errors.</td>
</tr>
<tr>
<td>Lane Swap Errors</td>
<td>To indicate lane swap errors in traffic checker.</td>
</tr>
<tr>
<td>Lane Sequence Errors</td>
<td>To indicate lane sequence error in traffic checker.</td>
</tr>
<tr>
<td>Lane Alignment Errors</td>
<td>To indicate lane alignment error in traffic checker.</td>
</tr>
</tbody>
</table>

### 2.6 Signals

**Figure 15. Top-level Signals for Intel Arria 10 SerialLite III Streaming Standard Clocking Mode Design Example**

![Signal Diagram](image)

**Table 6. Design Example Interface Signals**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mgmt_clk</td>
<td>Input</td>
<td>1</td>
<td>Input clock for:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Avalon-MM PHY management interface for SerialLite III Streaming IP core</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Demo management module</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Demo control module</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Transceiver reset controller</td>
</tr>
</tbody>
</table>

*continued...*
<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pll_ref_clk</td>
<td>Input</td>
<td>1</td>
<td>This reference clock is used by the Clock Data Recovery (CDR) unit in the transceiver. It serves as a reference for the CDR to recover the clock from the serial line. The frequency of this clock must match the frequency you select in the IP parameter editor. It should also match the frequency of the tx_pll_ref_clk reference clock for the TX PLL at the Source variant.</td>
</tr>
<tr>
<td>mgmt_reset_n</td>
<td>Input</td>
<td>1</td>
<td>Design example asychrounous master reset. Assert this reset signal to reset the overall design example system. This is an active low signal.</td>
</tr>
<tr>
<td>snk_core_reset_n</td>
<td>Output</td>
<td>1</td>
<td>Demo management module asserts this signal to reset traffic checker module.</td>
</tr>
<tr>
<td>src_core_reset_n</td>
<td>Output</td>
<td>1</td>
<td>Demo management module asserts this signal to reset traffic generator module.</td>
</tr>
<tr>
<td>Data Signal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rx[n]</td>
<td>Input</td>
<td>Based on Number of Lanes value</td>
<td>This vector carries the transmitted streaming data from the core. N represents the number of lanes.</td>
</tr>
<tr>
<td>tx[n]</td>
<td>Output</td>
<td>Based on Number of Lanes value</td>
<td>This vector carries the transmitted streaming data to the core. N represents the number of lanes.</td>
</tr>
<tr>
<td>Status Signal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rx_activity_n</td>
<td>Output</td>
<td>1</td>
<td>This single bit signal indicates that the data is valid.</td>
</tr>
<tr>
<td>tx_activity_n</td>
<td>Output</td>
<td>1</td>
<td>This single bit signal indicates that the data is valid.</td>
</tr>
<tr>
<td>snk_link_up_n</td>
<td>Output</td>
<td>1</td>
<td>The core asserts this signal to indicate that the core initialization is complete and is ready to receive user data.</td>
</tr>
<tr>
<td>src_link_up_n</td>
<td>Output</td>
<td>1</td>
<td>The core asserts this signal to indicate that the core initialization is complete and is ready to transmit user data.</td>
</tr>
</tbody>
</table>
3 Detailed Description for Intel Arria 10 SerialLite III Streaming Advanced Clocking Mode

These design examples demonstrate the functionalities of data streaming using advanced clocking mode.

To generate the design examples, select the following preset(s):
- Advanced Clocking Mode 2x10G
- Advanced Clocking Mode 6x12.5G

By default, the design examples are generated as duplex core. To generate the design examples in simplex core, select **Simplex** for the **Direction** parameter.

### 3.1 Features

Features for Advanced Clocking Mode 2x10G design example includes:
- Support 2 lanes with 10Gbps transceiver data rate
- Support simplex and duplex transmission modes
- Traffic checker for data verification and lane de-skew verification
- Support CRC error injection using Nios II processor

Features for Advanced Clocking Mode 6x12.5G design example includes:
- Support 6 lanes with 12.5Gbps transceiver data rate
- Support simplex and duplex transmission modes
- Traffic checker for data verification and lane de-skew verification
- Support CRC error injection using Nios II processor

### 3.2 Hardware and Software Requirements

Intel uses the following hardware and software to test the example designs in a Linux system:
- Intel Quartus Prime software
- ModelSim - Intel FPGA Edition, Modelsim-SE, NC-Sim (Verilog only), or VCS simulator
- Intel Arria 10 GX Transceiver Signal Integrity Development Kit for hardware testing

### 3.3 Functional Description

The design examples consist of various components. The following block diagrams show the design components and the top-level connections of the design examples.
Figure 16. Design Example for Simplex Core in Advanced Clocking Mode
3.3.1 Design Example Components

The design example consists of following components:

- SerialLite III Streaming IP core variation
- ATX PLL
- Source user clock—I/O PLL
- Traffic generator
- Traffic checker
- Demo control
- Demo management
- Nios II processor code

3.3.1.1 SerialLite III Streaming IP Core

The SerialLite III Streaming IP core 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. The core is generated using the parameter editor in the Intel Quartus Prime software.
3.3.1.2 Source User Clock - I/O PLL

The I/O PLL generates a user clock for sourcing data into the SerialLite III Streaming IP core when configured in Advanced Clocking Mode.

3.3.1.3 ATX PLL

The ATX PLL generates transmit transceiver clock to the SerialLite III Streaming IP core.

Related Links

How to compensate the jitter of PLL cascading or non-dedicated clock path for Intel Arria 10 reference clock?

In previous Quartus Prime versions, reference clock from a cascaded PLL output, global clock, or core clock causes additional jitter in Intel Arria 10 SerialLite III Streaming IP core design example. If you are migrating your design from earlier versions to Quartus Prime 17.1, refer to the KDB link.

3.3.1.4 Traffic Generator

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

Figure 18. 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

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word ID</td>
<td>63–59</td>
<td>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).</td>
</tr>
<tr>
<td>Burst Count</td>
<td>58–32</td>
<td>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.</td>
</tr>
<tr>
<td>Word Count</td>
<td>31–0</td>
<td>Tracks the number of valid sample words that have been transferred, across all bursts, to the source user interface.</td>
</tr>
</tbody>
</table>
3.3.1.5 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.

3.3.1.6 Demo Control

The demo control module is a Nios II processor system, generated in Platform Designer (Standard), to control the demo hardware.

Demo control module also consists of a timer to track interrupt occurrence, Avalon-MM interface to access demo management and the SerialLite III Streaming IP PHY interface, a reset controller, an UART interface, and an Avalon Streaming (Avalon-ST) interface.

3.3.1.7 Demo Management

The demo management module controls the user modules interaction with the SerialLite III Streaming IP core such as enable and disable traffic generator and traffic checker, enable CRC error insertion, and provide user clock reset for SerialLite III Streaming IP core. The module also implements CSRs to control and monitor the design operation. This includes CSRs to monitor and log errors that occur during the operation.

3.3.1.8 Nios II Processor Code

The Nios II processor controls the options exercised in the design example. The code also enables CRAM bits for CRC-32 error injection support.

The design example sets the bit for channel 0 that connects to lane 0 in the design example. Therefore, CRC error injection is exercisable for lane 0 only. Refer to the Nios II processor source code (`demo_control.c`) for information on setting bits for other channels.

3.3.2 Reset Scheme

The `mgmt_reset_n` reset signal controls the overall reset structure for the design example. This is an asynchronous and active-low signal. Asserting this signal resets the demo control module and the SerialLite III Streaming IP core. The traffic generator and traffic checker modules get reset through the demo management and the SerialLite III Streaming IP core.

The following diagrams show the reset scheme implemented in the design examples.
Figure 19. Reset Scheme for Intel Arria 10 SerialLite III Streaming Simplex Core in Advanced Clocking Mode
3.3.3 Clocking Scheme

The following diagrams show the clocking scheme for the design examples.
Figure 21. Clocking Scheme for Intel Arria 10 SerialLite III Streaming Simplex Core in Advanced Clocking Mode
3.4 Simulation

The simulation test cases demonstrate continuous streaming of 2000 sample data for all lanes from traffic generator to the SerialLite III Streaming source core and externally loopback to the sink core in advanced clocking mode.

The simulation test case performs the following steps:
1. Initialize and configures SerialLite III Streaming IP core, traffic generator and traffic checker.
2. Traffic generator generates data and starts data transmission.
3. Logs and display link up status and burst information.
4. Traffic checker verifies received data and stop transmission.
5. Testbench logs and displays test result and test information.
3.4.1 Testbench

If your design targets Intel Arria 10 devices, the generated example testbench is dynamic and has the same configuration as the IP.

*Note:* The Intel Arria 10 example testbench includes the external transceiver PLL; the IP core does not include the transceiver PLL for these devices.

**Figure 23.** SerialLite III Streaming Example Testbench (Duplex) for Intel Arria 10 Devices

**Figure 24.** SerialLite III Streaming Example Testbench (Simplex) for Intel Arria 10 Devices
3.5 Hardware Testing

Once you download the design and accompanying software into the FPGA, you can test the design operation through the interactive session. The interactive session provides helpful statistics, as well as controls for controlling various aspects of the design.

You can control the following operations through the interactive session by entering the option numbers listed below:

1. **Enable Data Generator/Checker**—Enables the traffic generator and start sending out data. This option enable data streaming in continuous mode.
2. **Disable Data Generator/Checker**—Disables traffic generation.
3. **Reset Source Core**—Resets the source core and traffic generator.
4. **Reset Sink Core**—Resets the sink core and traffic checker.
5. **Display Error Details**—Displays the error statistics.
6. **Toggle Burst/Continuous Mode**—Resets the source and sink MACs and switches the traffic generator to generate a burst or continuous traffic stream. By default, the design example is set to burst mode. When in continuous mode, the burst count will always show 1. Disable the data generator/checker before switching mode to avoid transmission error.
7. **Toggle CRC Error Insertion**—Turns CRC error injection off or on. By default, the design example has CRC error injection turned off.

3.5.1 Design Setup

The design example targets the Intel Arria 10 Transceiver Signal Integrity Development Kit.

The design includes an SDC script as well as a QSF with verified constraints in loopback mode. If you use the design example with another device or development board, you may need to update the device setting and constraints in the QSF file.

You must use correct pin constraints when using the core in simplex mode or when using more than one reconfiguration controller. The synthesized design typically includes a reconfiguration interface for at least three channels because three channels share an Avalon-MM slave interface, which connects to the Transceiver Reconfiguration Controller IP core. Conversely, you cannot connect three channels that share an Avalon-MM interface to different Transceiver Reconfiguration Controller IP cores or you will receive a Fitter error.

**Related Links**

-Altera Transceiver PHY IP Core User Guide
  More information about the Interlaken PHY IP core.

3.5.2 Error Details

These are the list of supported errors in the design example.
Table 8. List of Error Details

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source Error:</strong></td>
<td></td>
</tr>
<tr>
<td>Adaptation FIFO Overflow</td>
<td>To indicate source adaptation FIFO overflow error.</td>
</tr>
<tr>
<td><strong>Sink Errors:</strong></td>
<td></td>
</tr>
<tr>
<td>Adaptation FIFO Overflow</td>
<td>To indicate sink adaptation FIFO overflow error.</td>
</tr>
<tr>
<td>Loss of Alignment During Normal Operation</td>
<td>To indicate loss of alignment error (error_rx[1]).</td>
</tr>
<tr>
<td>Meta Frame CRC Errors</td>
<td>To indicate CRC errors.</td>
</tr>
<tr>
<td>Lane Swap Errors</td>
<td>To indicate lane swap errors in traffic checker.</td>
</tr>
<tr>
<td>Lane Sequence Errors</td>
<td>To indicate lane sequence error in traffic checker.</td>
</tr>
<tr>
<td>Lane Alignment Errors</td>
<td>To indicate lane alignment error in traffic checker.</td>
</tr>
</tbody>
</table>

3.6 Signals

Figure 25. Top-level Signals for Intel Arria 10 SerialLite III Streaming Advanced Clocking Mode Design Example
Table 9. Design Example Interface Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock and Reset Signal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mgmt_clk</td>
<td>Input</td>
<td>1</td>
<td>Input clock for:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Avalon-MM PHY management interface for SerialLite III Streaming IP core</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Demo management module</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Demo control module</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Transceiver reset controller</td>
</tr>
<tr>
<td>pll_ref_clk</td>
<td>Input</td>
<td>1</td>
<td>This reference clock is used by the Clock Data Recovery (CDR) unit in the transceiver. It serves as a reference for the CDR to recover the clock from the serial line. The frequency of this clock must match the frequency you select in the IP parameter editor. It should also match the frequency of the tx_pll_ref_clk reference clock for the TX PLL at the Source variant.</td>
</tr>
<tr>
<td>mgmt_reset_n</td>
<td>Input</td>
<td>1</td>
<td>Design example asynchronous master reset. Assert this reset signal to reset the overall design example system. This is an active low signal.</td>
</tr>
<tr>
<td>snk_core_reset_n</td>
<td>Output</td>
<td>1</td>
<td>Demo management module asserts this signal to reset traffic checker module.</td>
</tr>
<tr>
<td>src_core_reset_n</td>
<td>Output</td>
<td>1</td>
<td>Demo management module asserts this signal to reset traffic generator module.</td>
</tr>
<tr>
<td>Data Signal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rx[n]</td>
<td>Input</td>
<td>Based on Number of Lanes value</td>
<td>This vector carries the transmitted streaming data from the core. N represents the number of lanes.</td>
</tr>
<tr>
<td>tx[n]</td>
<td>Output</td>
<td>Based on Number of Lanes value</td>
<td>This vector carries the transmitted streaming data to the core. N represents the number of lanes.</td>
</tr>
<tr>
<td>Status Signal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rx_activity_n</td>
<td>Output</td>
<td>1</td>
<td>This single bit signal indicates that the data is valid.</td>
</tr>
<tr>
<td>tx_activity_n</td>
<td>Output</td>
<td>1</td>
<td>This single bit signal indicates that the data is valid.</td>
</tr>
<tr>
<td>snk_link_up_n</td>
<td>Output</td>
<td>1</td>
<td>The core asserts this signal to indicate that the core initialization is complete and is ready to receive user data.</td>
</tr>
<tr>
<td>src_link_up_n</td>
<td>Output</td>
<td>1</td>
<td>The core asserts this signal to indicate that the core initialization is complete and is ready to transmit user data.</td>
</tr>
</tbody>
</table>
A Intel Arria 10 SerialLite III Streaming IP Core Design Example User Guide Document Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

<table>
<thead>
<tr>
<th>IP Core Version</th>
<th>User Guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.0</td>
<td>SerialLite III Streaming IP Core Design Example User Guide Document Archives</td>
</tr>
<tr>
<td>16.1</td>
<td>SerialLite III Streaming IP Core Design Example User Guide Document Archives</td>
</tr>
<tr>
<td>16.0</td>
<td>SerialLite III Streaming IP Core Design Example User Guide Document Archives</td>
</tr>
</tbody>
</table>

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Empirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel’s standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.
# B Document Revision History for SerialLite III Streaming IP Core Design Example User Guide for Intel Arria 10

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| November 2017 | 2017.11.06 | • Renamed the document as *SerialLite III Streaming IP Core Design Example User Guide for Intel Arria 10 Devices*.  
• Added Knowledge Base link *How do I compensate for the jitter of PLL cascading or non-dedicated clock path for Arria 10 PLL reference clock?* on page 12 chapter.  
• Updated for latest branding standards. |
| May 2017    | 2017.05.08 | • Rebranded as Intel.  
• Restructured document to migrate Stratix 10 device support to a new design example user guide.  
• Updated the Compiling and Testing the Design topic.  
• Updated the Directory and File Description for Design Example Folder table.  
• Updated the Testbench sub-topic for Detailed Description for Arria 10 SerialLite III Streaming Standard Clocking Mode chapter.  
• Updated the following figures:  
  — Design Example for Simplex Core in Advanced Clocking Mode  
  — Design Example for Duplex Core in Advanced Clocking Mode  
  — Reset Scheme for Arria 10 SerialLite III Streaming Simplex Core in Advanced Clocking Mode  
  — Reset Scheme for Arria 10 SerialLite III Streaming Duplex Core in Advanced Clocking Mode  
  — Clocking Scheme for Arria 10 SerialLite III Streaming Simplex Core in Advanced Clocking Mode  
  — Clocking Scheme for Arria 10 SerialLite III Streaming Duplex Core in Advanced Clocking Mode  
• Updated the Testbench sub-topic for Detailed Description for Arria 10 SerialLite III Streaming Advanced Clocking Mode chapter.  
• Minor typographical corrections. |
| December 2016 | 2016.12.09 | • Added 6x17.4Gbps presets for Intel Stratix® 10 Standard and Advanced Mode design examples.  
• Added error details list. |
| October 2016 | 2016.10.28 | • Added Intel Stratix 10 device support.  
• Restructured document. |
| May 2016    | 2016.05.02 | Initial release. |