SerialLite III Streaming MegaCore Function User Guide

Last updated for Altera Complete Design Suite: 14.0a10
UG-01126
2014.08.18

Subscribe
Send Feedback

101 Innovation Drive
San Jose, CA 95134
www.altera.com

Contents

SerialLite III Streaming MegaCore Function Quick Reference

1

2014.08.18

Subscribe Send Feedback

The Altera® SerialLite III Streaming MegaCore® IP function is a lightweight protocol suitable for high bandwidth streaming data in chip-to-chip, board-to-board, and backplane applications.

The SerialLite III Streaming IP core is part of the MegaCore IP Library, which is distributed with the Quartus® II software and is downloadable from the Altera website at **www.altera.com**.

Note: For system requirements and installation instructions, refer to the *Altera Software Installation and Licensing Manual*.

Table 1-1: SerialLite III Streaming MegaCore Function

Item		Description	
Release Information	Version	14.0	14.0a10
	Release Date	June 2014	August 2014
	IP Catalog Name	SerialLite III Streaming	Arria 10 SerialLite III Streaming
	Ordering Code	IP-SLITE3/ST	
	Product ID	010A	
	Vendor ID	6AF7	

Item		Description
IP Core Information	Core Features	• Supports 1–24 serial lanes in configurations that provide nominal bandwidths from 3.125 gigabits per second (Gbps) to over 300 Gbps. • Avalon® Streaming (Avalon-ST) user interfaces on the transmit and receive datapaths.
	Protocol Features	• Simplex and duplex operations • Support for single or multiple lanes • 64B/67B physical layer encoding • Payload and idle scrambling • Error detection • Low overhead framing • Low point-to-point transfer latency
	Typical Application	• High resolution video • Radar processing • Medical imaging • Baseband processing in wireless infrastructure
	Device Family Support	Arria® 10, Arria V GZ, and Stratix® V FPGA devices. Refer to the *What's New in Altera IP* page of the Altera website for detailed information.
	Design Tools	• Parameter editor in the Quartus II software for IP design instantiation and compilation • TimeQuest timing analyzer in the Quartus II software for timing analysis • ModelSim-Altera software, MATLAB, or third-party tool using NativeLink for design simulation or synthesis

Related Information

- **Altera Software Installation and Licensing**
- **What's New in Altera IP**

About the SerialLite III Streaming IP Core 2

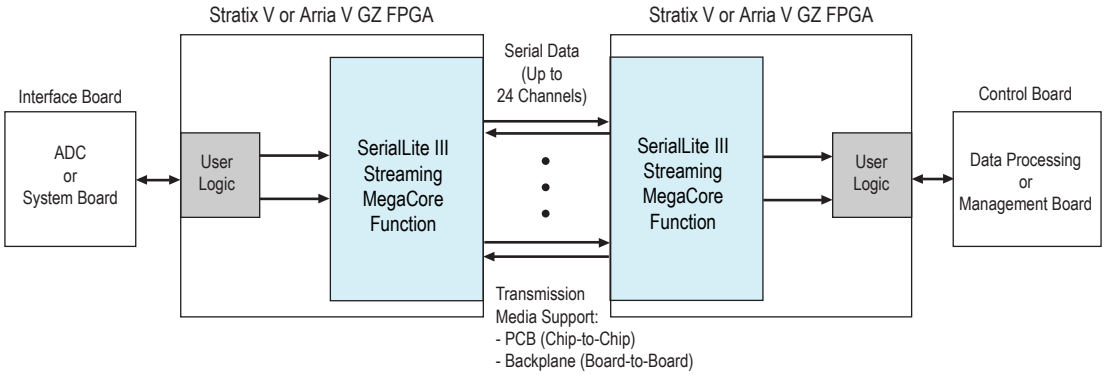
2014.08.18

UG-01126 Subscribe Send Feedback

The SerialLite III Streaming IP core is a high-speed serial communication protocol for chip-to-chip, board-to-board, and backplane application data transfers. This protocol offers high bandwidth, low overhead frames, low I/O count, and supports scalability in both number of lanes and lane speed.

The SerialLite III Streaming IP core incorporates a physical coding sublayer (PCS), a physical media attachment (PMA), and a media access control (MAC) block. The IP core transmits and receives streaming data through the Avalon-ST interface on its FPGA fabric interface.

Figure 2-1: Typical System Application



SerialLite III Streaming Protocol

The SerialLite III Streaming IP core implements a protocol which supports the transfer of high bandwidth streaming data over a unidirectional or bidirectional, high-speed serial link.

The SerialLite III Streaming IP core has the following protocol features:

- Simplex and duplex operations
- Support for single or multiple lanes
- 64B/67B physical layer encoding
- Payload and idle scrambling
- Error detection
- Low protocol overhead

- Low point-to-point transfer latency
- Uses the hardened Native PHY IP core (Arria 10 devices) or Interlaken PHY IP core (Stratix V and Arria V GX devices) to reduce soft logic resource utilization

SerialLite III Streaming Protocol Operating Modes

The protocol defines two operating modes for different applications: continuous and burst mode. This section defines these two operating modes, and describes the targeted application models and their key characteristics. The following table shows the key differences of the two operating modes.

Table 2-1: Continuous vs. Burst Mode Characteristics

Characteristics	Continuous Mode	Burst Mode
Buffering	Minimal	Burst size
Can connect directly to a data converter (ADC, DAC)	Yes	No
Asynchronous clock and data recovery support	No	Yes

Continuous Mode

A SerialLite III Streaming link operating in continuous mode accepts and transmits user data over the link, and presents it on the user interface at the receiving link at the same rate and without gaps in the stream. When operating in this mode, a link implementing the protocol looks like a data pipe that can transparently forward all data presented on the user interface to the far end of the link.

Continuous mode is appropriate for applications that require a simple interface to transmit a single, high bandwidth data stream. An example of this application is sensor data links for radar and wireless infrastructure. With this mode, data converters can connect to either end of the link with minimal interface logic. This mode requires both ends of the link to operate from a common clock.

Burst Mode

A SerialLite III Streaming link operating in burst mode accepts bursts of data across the user interface and transmits each burst across the link as a discrete data burst.

Burst mode is appropriate for applications where the data stream is divided into bursts of data. An example of this application is uncompressed digital video where the data stream is divided into lines of display raster. This mode provides more flexibility to the clocking and also supports multiplexing of multiple data streams across the link.

Performance and Resource Utilization

The following table lists the resources and expected performance for different SerialLite III Streaming IP core variations. These results are obtained using the Quartus II software targeting the Stratix V GX (5SGXMA7H2F35C2), the Arria V GZ (5AGZME7K2F40I3L), and the Arria 10 (10AX115N1F45E1LG) FPGA device.

Note: The numbers of ALMs and logic registers in the following table are rounded up to the nearest 100.

Table 2-2: SerialLite III Streaming IP Core FPGA Performance and Resource Utilization

Device	Direction	Clocking Mode	Parameters			ALMs	Logic Registers		M20K
			Number of Lanes	Per-Lane Data Rate (Mbps)	ECC		Primary	Secondary	
Arria 10	Source	Standard	12	10312.50	Disabled	1400	1900	100	24
		Standard	12	10312.50	Enabled	2400	3500	300	36
		Advanced	12	10312.50	Disabled	1500	1900	100	44
		Advanced	12	10312.50	Enabled	3100	4700	400	61
	Sink	Standard	12	10312.50	Disabled	2200	3500	200	24
		Standard	12	10312.50	Enabled	2700	4400	400	36
		Advanced	12	10312.50	Disabled	2100	3400	200	24
		Advanced	12	10312.50	Enabled	2600	4300	400	36
	Duplex	Standard	12	10312.50	Disabled	2800	5300	200	48
		Standard	12	10312.50	Enabled	5100	7800	700	72
		Advanced	12	10312.50	Disabled	3500	5200	200	68
		Advanced	12	10312.50	Enabled	5900	8800	900	97
Stratix V GX and Arria V GZ	Source	Standard	12	10312.50	Disabled	3600	2300	10	24
		Standard	12	10312.50	Enabled	4500	3700	700	36
		Advanced	12	10312.50	Disabled	3600	2300	100	44
		Advanced	12	10312.50	Enabled	5400	5100	400	61
	Sink	Standard	12	10312.50	Disabled	3100	4000	100	24
		Standard	12	10312.50	Enabled	3700	5000	400	36
		Advanced	12	10312.50	Disabled	3100	3900	200	24
		Advanced	12	10312.50	Enabled	3500	5000	400	36
	Duplex	Standard	12	10312.50	Disabled	5600	6000	200	48
		Standard	12	10312.50	Enabled	7000	8000	1000	72
		Advanced	12	10312.50	Disabled	5700	6000	200	68
		Advanced	12	10312.50	Enabled	8000	9800	800	97

Related Information

Fitter Resources Reports

More information about Quartus II resource utilization reporting.

Getting Started 3

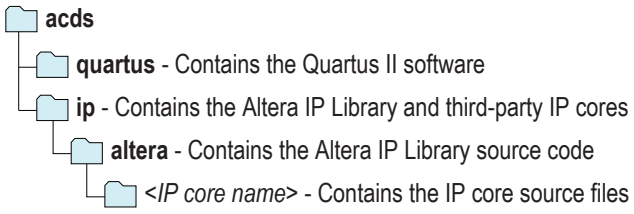
2014.08.18

UG-01126 Subscribe Send Feedback

Installing and Licensing IP Cores

The Altera IP Library provides many useful IP core functions for production use without purchasing an additional license. You can evaluate any Altera IP core in simulation and compilation in the Quartus II software using the OpenCore evaluation feature. Some Altera IP cores, such as MegaCore® functions, require that you purchase a separate license for production use. You can use the OpenCore Plus feature to evaluate IP that requires purchase of an additional license until you are satisfied with the functionality and performance. After you purchase a license, visit the Self Service Licensing Center to obtain a license number for any Altera product.

Figure 3-1: IP Core Installation Path



Note: The default IP installation directory on Windows is **<*drive*>:\altera**<*version number*>; on Linux it is <*home directory*>**/altera/** <*version number*>.

Related Information

- **Altera Licensing Site**
- **Altera Software Installation and Licensing Manual**

OpenCore Plus IP Evaluation

Altera's free OpenCore Plus feature allows you to evaluate licensed MegaCore IP cores in simulation and hardware before purchase. You need only purchase a license for MegaCore IP cores if you decide to take your design to production. OpenCore Plus supports the following evaluations:

- Simulate the behavior of a licensed IP core in your system.
- Verify the functionality, size, and speed of the IP core quickly and easily.
- Generate time-limited device programming files for designs that include IP cores.
- Program a device with your IP core and verify your design in hardware

OpenCore Plus evaluation supports the following two operation modes:

- Untethered—run the design containing the licensed IP for a limited time.
- Tethered—run the design containing the licensed IP for a longer time or indefinitely. This requires a connection between your board and the host computer.

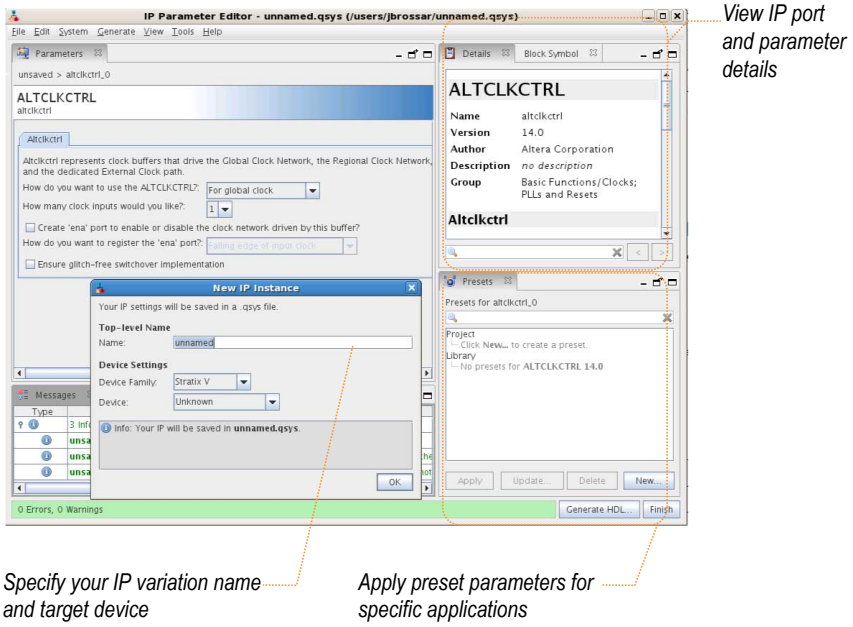
Note: All IP cores that use OpenCore Plus time out simultaneously when any IP core in the design times out.

Specifying IP Core Parameters and Options

The parameter editor GUI allows you to quickly configure your custom IP variation. Use the following steps to specify IP core options and parameters in the Quartus II software. Refer to *Specifying IP Core Parameters and Options (Legacy Parameter Editors)* for configuration of IP cores using the legacy parameter editor.

1. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name for your custom IP variation. The parameter editor saves the IP variation settings in a file named <*your_ip*>**.qsys**. Click **OK**.
3. Specify the parameters and options for your IP variation in the parameter editor, including one or more of the following. Refer to your IP core user guide for information about specific IP core parameters.
   - Optionally select preset parameter values if provided for your IP core. Presets specify initial parameter values for specific applications.
   - Specify parameters defining the IP core functionality, port configurations, and device-specific features.
   - Specify options for processing the IP core files in other EDA tools.
4. Click **Generate HDL**, the **Generation** dialog box appears.
5. Specify output file generation options, and then click **Generate**. The IP variation files generate according to your specifications.
6. To generate a simulation testbench, click **Generate** > **Generate Testbench System**.
7. To generate an HDL instantiation template that you can copy and paste into your text editor, click **Generate** > **HDL Example**.
8. Click **Finish**. The parameter editor adds the top-level **.qsys** file to the current project automatically. If you are prompted to manually add the **.qsys** file to the project, click **Project** > **Add/Remove Files in Project** to add the file.
9. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports.

Figure 3-2: IP Parameter Editor



SerialLite III Parameter Editor

Based on the values you set, the SerialLite III streaming parameter editor automatically calculates the rest of the parameters, and provides you with the following values or information:

- Input data rate per lane
- Transceiver data rate per lane
- A list of feasible transceiver reference clock frequencies, one of which you select to provide to the core
- Information related to the core overheads

Important: If your design targets Stratix V or Arria V GZ devices, you cannot migrate your design to Arria 10 devices automatically. For Arria 10 devices, the transceiver reconfiguration functionality is embedded inside the transceivers. Therefore, you must re-instantiate the IP core to target Arria 10 devices.

Arria 10 Designs

If your design targets Arria 10 devices:

- The parameter editor displays a message about the required output clock frequency of the external TX PLL IP clock. For source or duplex modes, connect the reset controller to the TX PLL to ensure the appropriate HSSI power-up sequence.
- For source only Arria 10 implementations, the parameter editor does not provide the transceiver reference clock frequency because the user is expected to provide the transmit serial clock. If you use an on-chip PLL to generate the transmit serial clock, you can use the same PLL reference clock frequency that you provide to the core in the sink direction, operating at the same user clock frequency (or equivalent transceiver lane data rate).

- The SerialLite IP core expects the user to provide transmitter's serial clock. If you compile the IP without the proper serial clock, the Quartus II Compiler issues a compilation error. Refer to **Arria 10 Simulation Testbench** for an example design.
- When generating the example testbench, the SerialLite IP core instantiates an external transceiver ATX PLL for the transmit serial clock based on the required user clock only when configured in sink or duplex mode. The Arria 10 simulation testbench uses the external transceiver ATX PLL. The transceiver ATX PLL core is configured with the transceiver reference clock specified in the parameter editor and transmit serial clock.
- To generate the SerialLite III Arria 10 example testbench using the parameter editor, select **Generate > Example Designs > seriallite_iii_a10_0 - example** (alternatively, turn on the **Example Design** option in the parameter editor). Altera recommends that you generate the Arria 10 simulation testbench for the sink or duplex direction.

Related Information

- **SerialLite III Streaming IP Core Parameters** on page 3-4
  More information about the IP core parameters.
- **Arria 10 Simulation Testbench** on page 3-11
- **Arria 10 versus Stratix V and Arria V GZ Variations** on page 4-8

SerialLite III Streaming IP Core Parameters

Table 3-1: SerialLite III Streaming IP Core Parameters

Parameter		Value	Default	Description
General Design Options	**Direction**	Source, Sink, Duplex	Duplex	Indicates the direction of the core's variant.
	Lanes	1–24	4	Specifies the number of input lanes (equal to physical transceiver links) that are used to transfer the streaming data.
	Device speed grade	1–4	2	Specifies the device speed grade (Stratix V and Arria V GZ devices only).
	MetaFrame length	200–8191	8191	Specifies the metaframe length in 8-byte words.
	ECC protection	Yes/No	No	Select to use error correcting code (ECC) protection to strengthen the FIFO buffers from single-event upset (SEU) changes.

Parameter		Value	Default	Description
Clocking and Data Rates	**Advanced clocking mode**	Yes/No	No	Select to use the advanced clocking mode for your design. The default setting is standard clocking mode.
	Required user clock frequency	Minimum: 50 MHz Maximum: Limited by the supported transceiver data rates	146.484375 MHz	Specifies the clock generator's fractional PLL (fPLL) output frequency used to drive the `user_clock` signal. This range is device-specific and is tied with the lane data rate and fPLL minimal clocking constraints. In advanced clocking mode, this signal specifies the frequency required for the `user_clock` input.
	Generated user clock frequency(1)	Minimum: 50 MHz Maximum: Limited by the supported transceiver data rates	146.484375 MHz	Specifies the actual user clock frequency as produced by the fPLL and is ideally the same as the required clock frequency. In certain very high precision situations where the desired user clock is provided up to higher decimal places, this value can vary slightly due to the fPLL constraints. Change the required clock frequency to correct the issue if the minute variation is intolerable.
	Interface clock frequency	Lane rate/40	205.078125 MHz	Specifies the clock frequency of the source, sink, or duplex user interface in advanced clocking mode.
	Core clock frequency	(Lane rate/40)–(Lane rate/67)	205.078125 MHz	The core clock is used internally between the user domain and the Native PHY IP core (Arria 10 devices) or Interlaken PHY IP core (Stratix V and Arria V GZ devices). (2)
	fPLL reference clock frequency	Lane rate/40	257.812500 MHz	Specifies the fPLL reference clock frequency in standard clocking mode.(2)
	Transceiver reference clock frequency	Range supported by the transceiver PLLs (Lane rate/*N*)	644.53125 MHz	Specifies the transceiver reference clock frequency. The default value for the Input clock frequency is lane rate/16. Sample values of *N* include 80, 64, 50, 40, 32, 25, 20, 16, 12.5, 10, and 8. Altera recommends that you select the highest frequency among the available options in the drop-down list.

(1) The parameter editor automatically calculates this parameter value based on the general design options.
(2) The clock frequency value is useful if you want to simulate designs at different data rates. You should apply the displayed value in your testbench parameters.

Parameter		Value	Default	Description
Clocking and Data Rates	**Input data rate per lane**	64 × (User clock frequency)	9.375 Gbps	Input data rate that the core can support.
	Transceiver data rate per lane	Input data rate × Overheads	10.3125 Gbps	The effective data rate at the output of the transceivers, incorporating transmission and other overheads. The parameter editor automatically calculates this value by adding the input data rate with transmission overheads to provide you with a selection of user clock frequency.(2)
	Aggregate input data rate	Lanes × Input data rate	36.6210938 Gbps	Aggregate input data rate that the core can support.

Transceiver Reconfiguration Controller for Stratix V and Arria V GZ Designs

If your design targets Stratix V or Arria V GZ devices, the transceiver reconfiguration controller is not included in the generated IP core. To create a complete system, refer to the design example block diagram on how to connect the transceiver reconfiguration controller.

Note: If your design targets Arria 10 devices, the transceiver reconfiguration functionality is embedded inside the transceivers. The interface to access the internal reconfiguration controller is provided at the top level. Refer to the Arria 10 simulation testbench for further details.

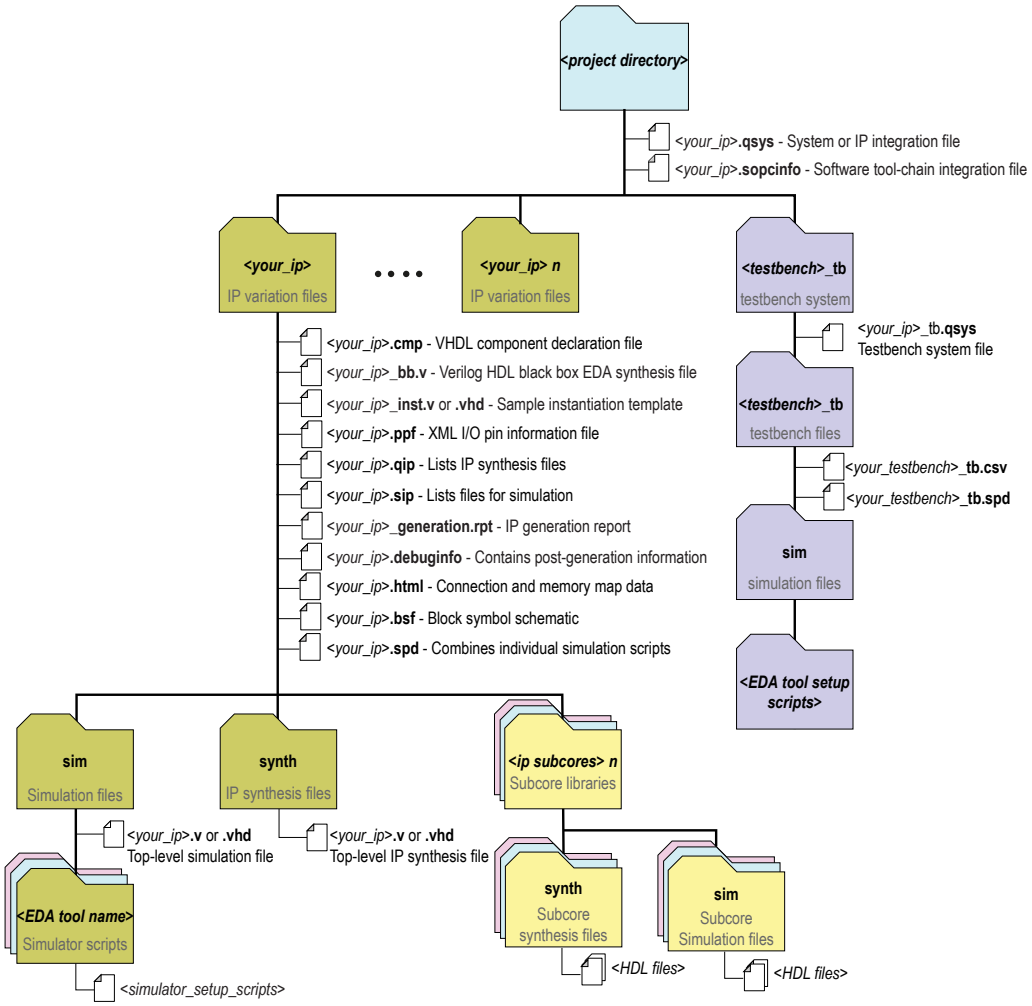
Related Information

- **Arria 10 Simulation Testbench** on page 3-11

Files Generated for Altera IP Cores

The Quartus II software generates the following files during generation of your IP core variation.

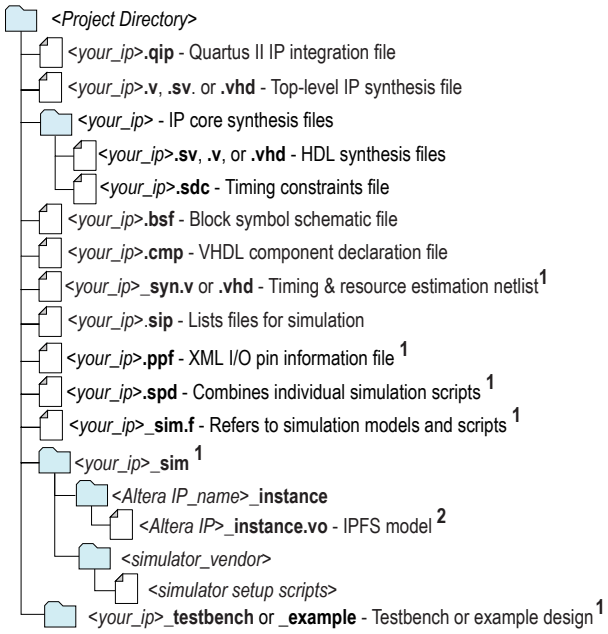
Figure 3-3: IP Core Generated Files



Files Generated for Altera IP Cores (version 14.0 and previous)

The Quartus II software version 14.0 and previous generates the following output for your IP core.

Figure 3-4: IP Core Generated Files



Notes:
1. If supported and enabled for your IP variation
2. If functional simulation models are generated

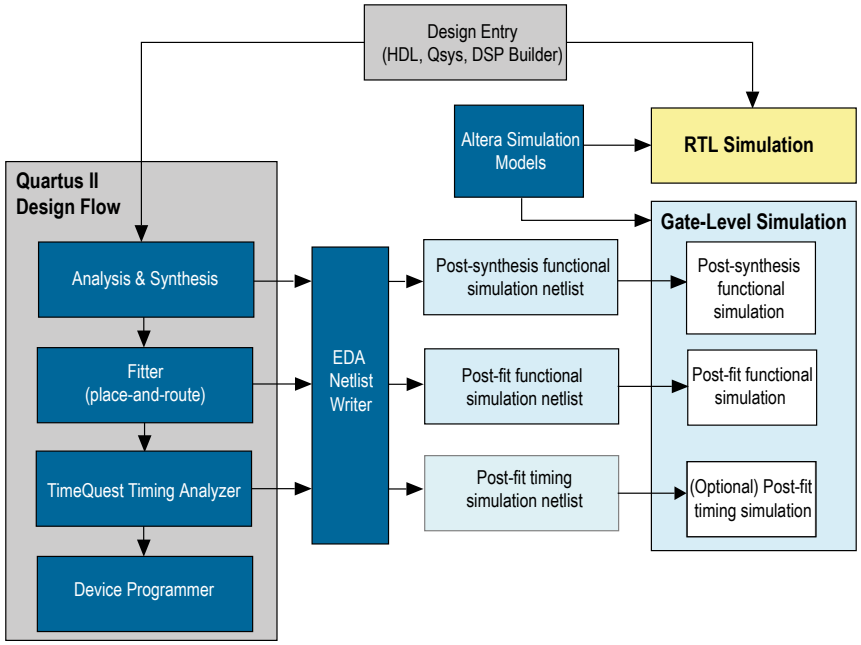
Simulating

Simulating Altera IP Cores in other EDA Tools

The Quartus II software supports RTL and gate-level design simulation of Altera IP cores in supported EDA simulators. Simulation involves setting up your simulator working environment, compiling simulation model libraries, and running your simulation.

You can use the functional simulation model and the testbench or example design generated with your IP core for simulation. The functional simulation model and testbench files are generated in a project subdirectory. This directory may also include scripts to compile and run the testbench. For a complete list of models or libraries required to simulate your IP core, refer to the scripts generated with the testbench. You can use the Quartus II NativeLink feature to automatically generate simulation files and scripts. NativeLink launches your preferred simulator from within the Quartus II software.

Figure 3-5: Simulation in Quartus II Design Flow



Note: Post-fit timing simulation is not supported for 28nm and later device archetectures. Altera IP supports a variety of simulation models, including simulation-specific IP functional simulation models and encrypted RTL models, and plain text RTL models. These are all cycle-accurate models. The models support fast functional simulation of your IP core instance using industry-standard VHDL or Verilog HDL simulators. For some cores, only the plain text RTL model is generated, and you can simulate that model. Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

Related Information

Simulating Altera Designs

Simulation Parameters

After design generation, simulation files are available for you to simulate your design. To simulate your design, ensure that the SerialLite III Streaming IP core source and sink cores are both generated with the same parameters or are duplex cores.

- Stratix V and Arria V GZ files are located in the <*variation name*>**_sim** directory
- Arria 10 files are located in the <*variation name*> directory

For Arria 10 devices, the example testbench simulates the core using the user-specified configuration (except for the metaframe length). The **test_env.v** file sets the metaframe length to 200 words to speed simulation.

For Stratix V and Arria V GZ devices, the example testbench uses the direction and clocking mode for which the core is generated, however, it uses preset values for other parameters. That is, parameter settings other than direction and clocking mode are not included in the example testbench. If you want to simulate the IP core for a different variant, change the parameter settings in the **test_env.v** file as appropriate.

Table 3-2: Stratix V and Arria V GZ Testbench Default Simulation Parameters

Parameter	Default Value	Comments
Generated user clock frequency (`user_clock_frequency`)	Standard clocking: 145.98375 MHz Advanced clocking: 146.484375	—
Lanes (`lanes`)	2	The simulation script may overwrite this parameter. Refer to the simulation scripts listed in Table 3-3 for details.
Transceiver reference clock frequency (`pll_ref_freq`)	644.53125 MHz	—
Transceiver data rate per lane (`data_rate`)	10312.5 Mbps	—
Meta frame length (`meta_frame_length`)	200	—
fPLL reference clock frequency (`reference_clock_frequency`)	257.8125 MHz	Not used in advanced clocking mode.
Core clock frequency (`coreclkin_frequency`)	205.078125 MHz	Not used in advanced clocking mode.
Simulation-specific parameters		
Total samples to transfer (`total_samples_to_transfer`)	2000	Total samples to transfer during simulation.
Mode (`mode`)	Continuous/burst	The testbench environment may automatically choose one of the modes depending on the random seed with which it is provided. Refer to the simulation scripts listed in Table 3-3 for details.
Skew insertion enable (`skew_insertion_enable`)	Yes	Skew testing is enabled. The testbench environment randomly inserts skew in the lanes within the range 0 - 107 UI.
ECC protection enabled (`ecc_enable`)	0	When set, the core is simulated with the ECC-enabled variant. Use the ECC-enabled variant in the test environment. When ECC mode is disabled, the two most significant bits of the error buses in the source or sink direction are don't care.

For more information about Altera simulation models, refer to the *Simulating Altera Designs* chapter in volume 3 of the Quartus II Handbook.

Related Information

Simulating Altera Designs

Arria 10 Simulation Testbench

If your design targets Arria 10 devices, the generated example testbench is dynamic and has the same configuration as the IP (except for the metaframe length). When you choose the sink or duplex direction, the parameter editor generates an external transceiver ATX PLL for use in the Arria 10 testbench. Therefore, Altera recommends that you generate the Arria 10 simulation testbench for designs using the sink or duplex direction.

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

Figure 3-6: SerialLite III Streaming Example Testbench (Duplex) for Arria 10 Devices

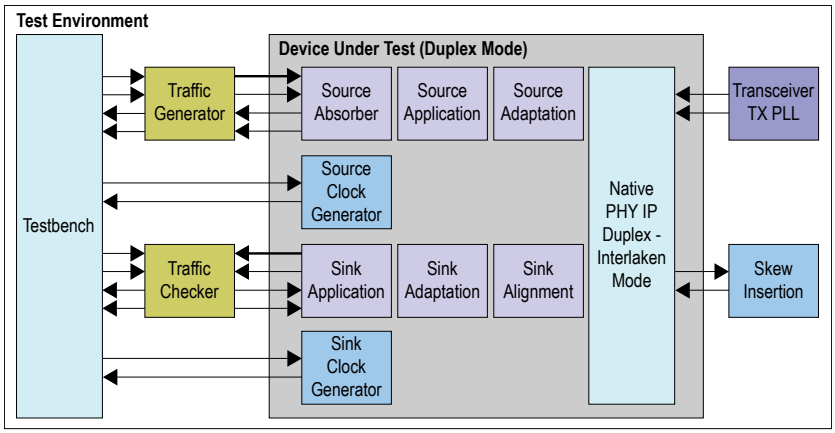
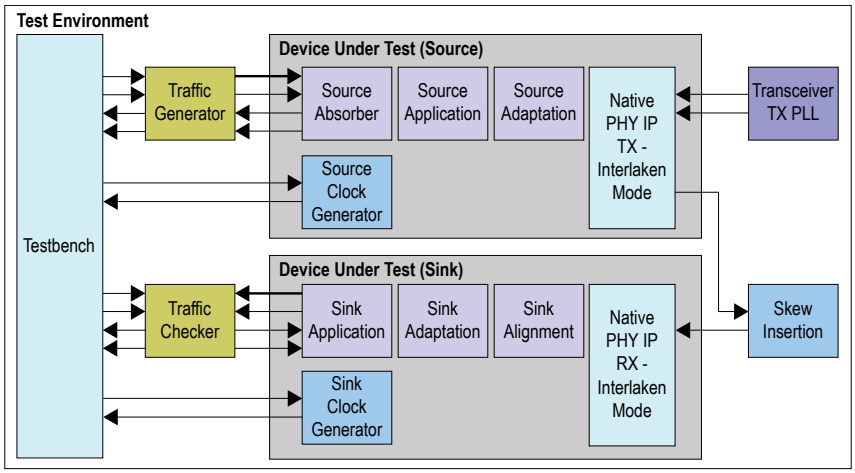


Figure 3-7: SerialLite III Streaming Example Testbench (Simplex) for Arria 10 Devices



Simulating and Verifying the Design

1. Set the environment variables:
   - QUARTUS_ROOTDIR to the location of the Quartus II software directory.
   - For ModelSim-Altera only: MODELSIM_ALTERA_LIBS to the location of the precompiled simulation libraries.
2. For simplex mode, set the following environment variables:
   - SRC_SIM_LOCATION to the location where the simulation files for the source core are generated. For instance, if the name of the source core is '**src**', then the directory name is **/src_sim** for Stratix V and Arria V GZ devices and **/src** for Arria 10 devices.
   - SNK_SIM_LOCATION to the location where the simulation files for sink core are generated. For instance, if the name of the sink core is '**snk**', then the directory name is **/snk_sim** for Stratix V and Arria V GZ devices and **/snk** for Arria 10 devices.
3. For duplex mode, set the following environment variable:
   - SIM_LOCATION to the location where the simulation files for the duplex core are generated. For instance, if the name of the duplex core is '**duplex**', then the directory name is **/duplex_sim** for Stratix V and Arria V GZ devices and **/duplex** for Arria 10 devices.
4. Set the parameters in the **test_env.v** file (optional).
5. Run the provided scripts to simulate the testbench in the ModelSim-Altera SE/AE, VCS, VCS MX, or Aldec Riviera simulators. The following table lists the provided scripts.

Table 3-3: Testbench Simulation Scripts

Simulator	File Directory	Device Family	Script
ModelSim-Altera SE/AE	<*example design name*>**/example_testbench/vsim/**	Arria 10	**vsim -c -do run_vsim.do**
	<*variation name*>**_example/seriallite_iii_sv/example_testbench/vsim/**	Stratix V and Arria V GZ	
VCS/VCS MX	<*example design name*>**/example_testbench/vcs/**	Arria 10	**run_vcs.sh**
	<*variation name*>**_example/seriallite_iii_sv/example_testbench/vcs/**	Stratix V and Arria V GZ	
NCSim	<*example design name*>**/example_testbench/ncsim/**	Arria 10	**run_ncsim.sh**
	<*variation name*>**_example/seriallite_iii_sv/example_testbench/ncsim/**	Stratix V and Arria V GZ	
Aldec Riviera	<*example design name*>**/example_testbench/aldec/**	Arria 10	**run_aldec.sh**
	<*variation name*>**_example/seriallite_iii_sv/example_testbench/aldec/**	Stratix V and Arria V GZ	

By default, the parameter editor generates simulator-specific scripts containing commands to compile, elaborate, and simulate Altera IP models and simulation model library files. You can copy the commands into your simulation testbench script, or edit these files to add commands for compiling, elaborating, and simulating your design and testbench.

Table 3-4: Altera IP Core Simulation Scripts

Simulator	File Directory	Device Family	Script
ModelSim-Altera SE/AE	<*variation name*>**_sim/mentor**	Stratix V and Arria V GZ	**msim_setup.tcl**
	<*variation name*>**/sim/mentor**	Arria 10	
VCS	<*variation name*>**_sim/synopsys/vcs**	Stratix V and Arria V GZ	**vcs_setup.sh**
	<*variation name*>**/sim/synopsys/vcs**	Arria 10	
VCS MX	<*variation name*>**_sim/synopsys/vcsmx**	Stratix V and Arria V GZ	**vcsmx_setup.sh** **synopsys_sim.setup**
	<*variation name*>**/sim/synopsys/vcsmx**	Arria 10	
NCSim	<*variation name*>**_sim/cadence**	Stratix V and Arria V GZ	**ncsim_setup.sh**
	<*variation name*>**/sim/cadence**	Arria 10	
Aldec Riviera	<*variation name*>**_sim/aldec**	Stratix V and Arria V GZ	**rivierapro_set.tcl**
	<*variation name*>**/sim/aldec**	Arria 10	

For more information about Altera simulation models, refer to the *Simulating Altera Designs* chapter in volume 3 of the Quartus II Handbook.

Related Information

Simulating Altera Designs

SerialLite III Streaming IP Core Functional Description

4

2014.08.18

UG-01126

Subscribe Send Feedback

The SerialLite III Streaming IP core implements a protocol that defines streaming data encapsulation at the link layer and data encoding at the physical layer. This protocol integrates transparently with existing hardware and provides a reliable data transfer mechanism in applications that do not need additional layers between the data link and application.

IP Core Architecture

The SerialLite III Streaming IP core has three variations:

- *Source*—Formats streaming data from the user application and transmits the data over serial links.
- *Sink*—Receives the serial stream data from serial links, removes any formatting information, and delivers the data to the user application.
- *Duplex*—Composed of both the source and sink cores. The streaming data can be transmitted and received in both directions.

All three variations include the Altera Native PHY IP core (Arria 10 devices) or Interlaken PHY IP core (Stratix V and Arria V GZ devices) that utilizes hardened PCS and PMA modules. The source and sink cores use the Native PHY or Interlaken PHY IP core in simplex mode, and the duplex core uses the Native PHY or Interlaken PHY IP core in duplex mode.

Table 4-1: MegaCore Variant and Function

Core	Function
Source	• Data encapsulation • Generates idle characters • Lane striping for multi-lane link • User synchronization and burst marker insertion
Sink	• Multi-lane alignment • Data encapsulation removal • Deletes idle characters • Lane de-striping • User synchronization and burst marker demultiplexing

Core	Function
Duplex	• Data encapsulation and decapsulation • Generates and removes idle characters • Lane striping and de-striping • User synchronization and burst marker insertion and deletion

The simplex and duplex cores support the following clocking schemes:

- *Standard clocking*—This mode is for pure streaming designs in which the core provides input/output clocks to drive the user logic. Pure streaming operation ensures an exact replica of the output data as it was presented at the input without any output gaps.
- *Advanced clocking*—This mode allows the core's input interface to be clocked with the user-preferred clock by trading-off pure streaming operation.

Figure 4-1: SerialLite III Streaming Simplex Core (Standard Clocking)

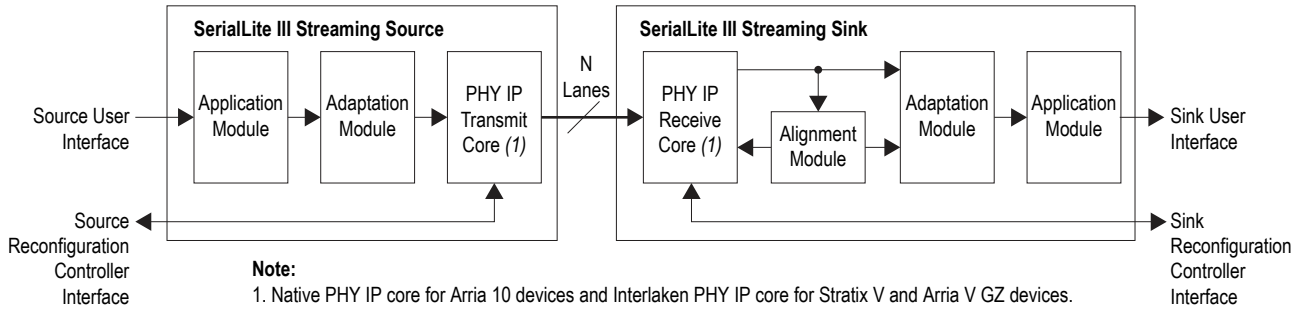
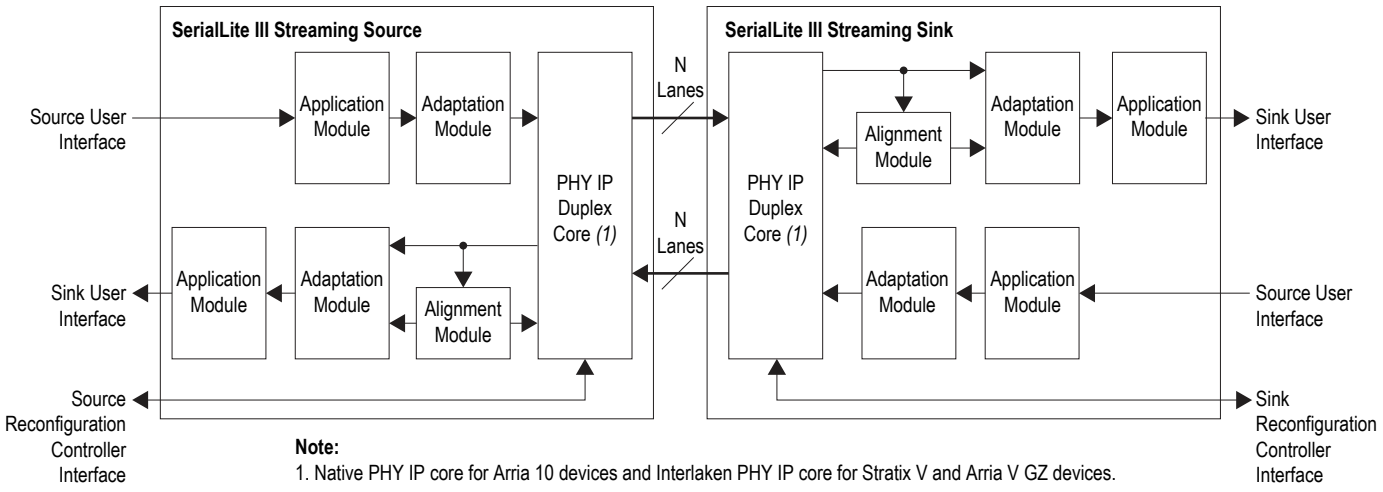


Figure 4-2: SerialLite III Streaming Duplex Core (Standard Clocking)



The block diagram for advanced clocking is similar to standard clocking, however, it also includes a PPM absorption FIFO at the source user interface.

Related Information

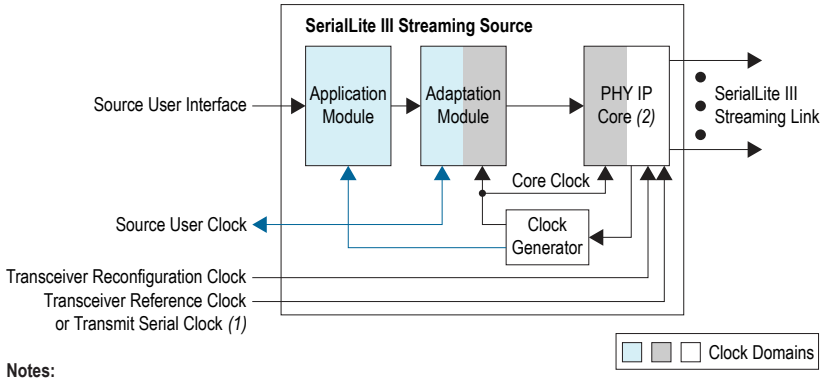
Altera Transceiver PHY IP Core User Guide
For more information about the sequence for bringing up the link.

SerialLite III Streaming Source Core

The source core consists of five major functional blocks (the implementation varies depending on the clocking mode):

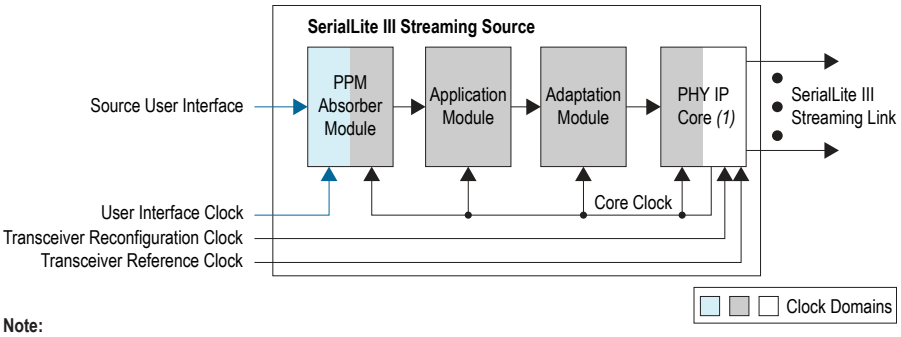
- Source application module
- Clock generator (in the standard clocking mode)
- Source adaptation module
- Native PHY IP TX core - Interlaken mode (Arria 10 devices)
- Interlaken PHY IP TX core (Stratix V and Arria V GZ devices)
- PPM-Absorption module (in the advanced clocking mode only)

Figure 4-3: SerialLite III Streaming Source Core (Standard Clocking Mode)



Notes:
1. Transceiver reference clock for Stratix V and Arria V GZ devices; transmit serial clock for Arria 10 devices.
2. Native PHY IP core for Arria 10 devices and Interlaken PHY IP core for Stratix V and Arria V GZ devices.

Figure 4-4: SerialLite III Streaming Source Core (Advanced Clocking Mode)



Note:
1. Native PHY IP core for Arria 10 devices and Interlaken PHY IP core for Stratix V and Arria V GZ devices.

Source PPM-Absorption Module on page 4-5

Source Application Module

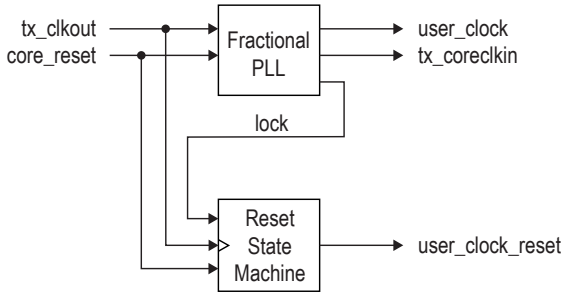
The application module performs the following functions:

- *Burst encapsulation*—Inserts burst control words into the data stream to define the beginning and the end of streaming data bursts.
- *Idle insertion*—Inserts idle control words (in the standard clocking mode) into all lanes of the data stream interface when streaming data is not available.

Clock Generator

The clock generator in the source core synthesizes the user clock (`user_clock`) and core clock signals (`tx_coreclockin`) from the Native PHY IP core (Arria 10 devices) or Interlaken PHY IP (Stratix V and Arria V GZ devices) core's output clock signal (`tx_clkout`). This clock generator also consists of a fractional PLL and a state machine responsible for clocks generation and reset sequencing. The `user_clock_reset` is not released until the fPLL is locked. The module is used in the standard clocking mode only.

Figure 4-5: Clock Generator Block Diagram



Source Adaptation Module

This module provides adaptation logic between the application module and the Native PHY IP core (Arria 10 devices) or Interlaken PHY IP (Stratix V and Arria V GZ devices) core. The adaptation module performs the following functions:

- *Rate adaptation*—Includes a dual-clock FIFO buffer to cushion the Interlaken core's bursty read requests and to provide a streaming user write interface. The FIFO also transfers streaming data between the `user_clock` and `tx_coreclkin` clock domains (in standard clocking mode).
- *Control signal translation*—The state machines maps the control signal semantics on the framing interface to the semantics of the Native PHY or Interlaken PHY IP core TX interface. It also handles the sequencing of the `phy_mgmt_clk_reset` signal that resets the Native PHY or Interlaken PHY IP core.
- *Non-user idle insertion*—Inserts non-user idle control words in the absence of user data to manage the minimum data rate requirements of the Interlaken protocol. The control words are removed at the sink adaptation.

Interlaken PHY IP TX Core or Native PHY IP TX Core - Interlaken Mode

For Arria 10 devices, this block is an instance of the Native PHY IP core configured for Interlaken - TX only operation.

For Stratix V and Arria V devices, the Interlaken PHY IP TX core is an instance of the Interlaken PHY IP core configured for TX only operation, and is generated by the Quartus II parameter editor. The core requires

a transceiver dynamic reconfiguration interface for transceiver calibration. The TX core initially requires as many reconfiguration interfaces as the number of transceivers and channels that the TX PLLs use.

Related Information

- **Arria 10 Transceiver PHY User Guide**
  For more information about the Arria 10 Native PHY IP core.
- **Altera Transceiver PHY IP Core User Guide**
  For more information about the Interlaken PHY IP core.

Source PPM-Absorption Module

This optional module is available when the SerialLite III Streaming IP core is instantiated with advanced clocking mode. This module allows you to use your own clock to interface data or to compensate the clock difference between the user clock and source interface clock.

Related Information

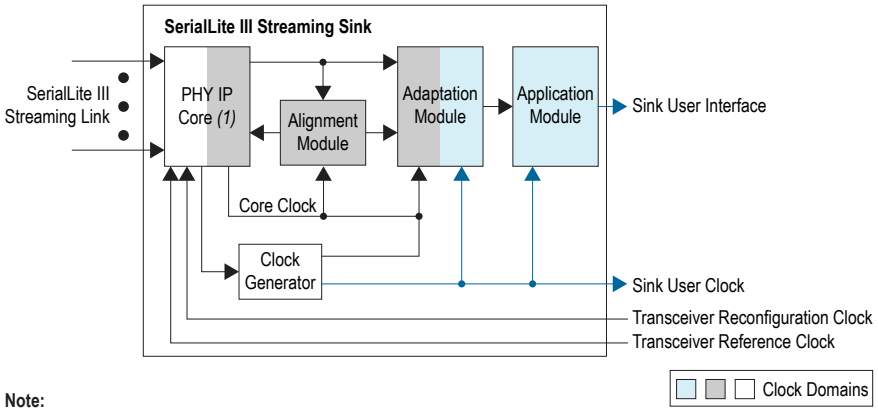
- **Advanced Clocking Mode** on page 4-11

SerialLite III Streaming Sink Core

The sink core consists of five major functional blocks:

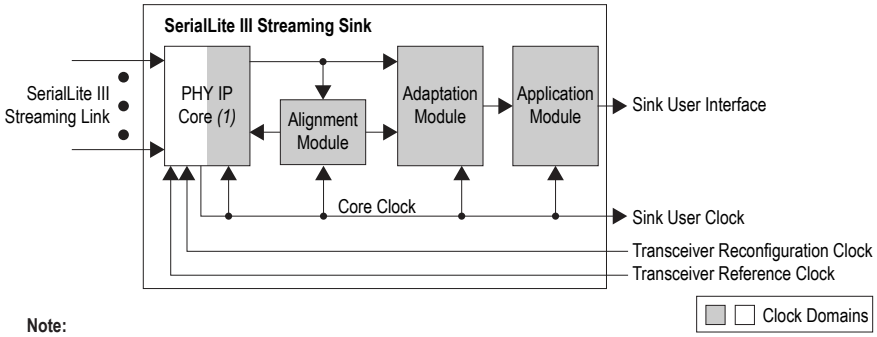
- Native PHY IP RX core - Interlaken mode (Arria 10 devices)
- Interlaken PHY IP RX core (Stratix V or Arria V GZ devices)
- Lane alignment module
- Clock generator (standard clocking mode only)
- Sink adaptation module
- Sink application module

Figure 4-6: SerialLite III Streaming Sink Core (Standard Clocking Mode)



Note:
1. Native PHY IP core for Arria 10 devices and Interlaken PHY IP core for Stratix V and Arria V GZ devices.

Figure 4-7: SerialLite III Streaming Sink Core (Advanced Clocking Mode)



Note:
1. Native PHY IP core for Arria 10 devices and Interlaken PHY IP core for Stratix V and Arria V GZ devices.

Sink Application Module

The sink application module performs the following functions:

- Removes Interlaken protocol control words and burst markers from the received serial data stream and presents the data to the user interface.
- Decodes idle control words inserted by the source application module when the data stream is not available and mirrors the data unavailability at the source by deasserting the output valid signal.

The encapsulation stripping process removes burst control words that define the beginning and the end of streaming data bursts from the data stream. This process adjusts the received data stream to repack the data words into a contiguous sequence.

In the standard clocking mode (pure streaming), the decoding process checks the received data stream to detect idle control words that the source application module inserts. When the sink application module detects the idle control words, it deasserts the valid signal on the user interface until it receives valid user streaming data.

In the advanced clocking mode, the sink application module does not insert or delete any idle words. Instead, the sink application module deasserts the output valid signal to indicate an absence of data coming from the sink adaptation module.

Clock Generator

The clock generator is similar to the clock generator in the source core, and is only instantiated in standard clocking mode. The clock generator synthesizes the user clock (`user_clock`) and core clock (`rx_coreclkin`) signals from the Native PHY IP core (Arria 10 devices) or Interlaken PHY IP (Stratix V and Arria V GZ devices) core's output clock signal. The clock generator consists of a fractional PLL and a state machine responsible for clock generation and reset sequencing.

Related Information

- **Clock Generator** on page 4-4

Sink Adaptation Module

The sink adaptation module provides rate adaptation logic between the application module and the Native PHY IP core or Interlaken PHY IP core. The adaptation module implements the following functions:

- *Rate adaptation*—Uses the lane FIFO buffers to do rate matching and absorb any data jitter between the lanes on the recovered clock. The FIFO buffers also transfer data between the lanes on the recovered clock. It also handles the Interlaken core's bursty write requests to present the user with the streaming interface. In standard clocking mode, the FIFO buffers also help transfer data between the `rx_coreclkin` and `user_clock` domains.
- *Interlaken framing layer stripping*—Strips Interlaken framing layer symbols and diagnostic control words from the data stream.
- *Non-user idle deletion*—Strips off any non-user idle control words that the source adaptation module inserts.
- *Management interface tie-off*—Removes the Avalon® Memory-Mapped (Avalon-MM) PCS management and dynamic reconfiguration interfaces in the Native PHY IP core or Interlaken PHY IP Core to an idle state. The core does not use any of these features.

Lane Alignment Module

The lane alignment module interfaces with the Native PHY or Interlaken PHY IP core to access incoming data. This module removes lane skew from the incoming serial data streams and aligns various lanes using the Interlaken's synchronization marker. After alignment is achieved, the module continuously monitors the synchronization markers in the Interlaken metaframes for any loss of alignment.

Interlaken PHY IP RX Core or Native PHY IP RX Core - Interlaken Mode

For Arria 10 devices, this block is an instance of the Native PHY IP core configured for Interlaken - RX only operation.

For Stratix V and Arria V devices, the Interlaken module is an instance of the Interlaken PHY IP core configured for RX only operation, and is generated by the Quartus II parameter editor. The core requires a transceiver dynamic reconfiguration interface for transceiver calibration. The interface size is initially equal to the number of transceiver channels that the sink core uses.

Related Information

- **Arria 10 Transceiver PHY User Guide**
  For more information about the Arria 10 Native PHY IP core.
- **Altera Transceiver PHY IP Core User Guide**
  For more information about the Interlaken PHY IP core.

SerialLite III Streaming Duplex Core

For Arria 10 devices, the duplex core is composed of source and sink cores interfaced with the Native PHY IP Duplex core in Interlaken mode.

For Stratix V and Arria V GZ devices, the duplex core is composed of source and sink cores interfaced with the Interlaken PHY IP in duplex mode.

Interlaken PHY IP Duplex Core or Native PHY IP Duplex Core - Interlaken Mode

For Arria 10 devices, this block is an instance of the Native PHY IP core configured for duplex Interlaken operation.

For Stratix V and Arria V GZ devices, the Interlaken module is an instance of the Interlaken PHY IP core configured for duplex operation, and is generated by the Quartus II parameter editor. The core requires a transceiver dynamic reconfiguration interface for transceiver calibration. The duplex core initially requires as many reconfiguration interfaces as the number of transceivers and channels that the IP core uses, equivalent to the PHY IP core in TX mode.

Related Information

- **Arria 10 Transceiver PHY User Guide**
  For more information about the Arria 10 Native PHY IP core.
- **Altera Transceiver PHY IP Core User Guide**
  For more information about the Interlaken PHY IP core.

Arria 10 versus Stratix V and Arria V GZ Variations

The Arria 10 transceiver is not the same as the Stratix V or Arria V GZ transceiver. Therefore, the SerialLite III IP core is implemented differently for these device families, and the example testbenches are different.

- When targeting Arria 10 devices, the IP core does not contain the transceiver PLL. Refer to the example design and testbench for information about how to include the transceiver PLL in your design.
- When targeting Arria 10 devices, The IP core does not include a reconfiguration controller.
- When you create an instance of the IP core, it generates an example testbench dynamically. This testbench has the same configuration as the IP core instance.
- The SerialLite III IP core does not include a hardware demonstration example design for Arria 10 devices.

For Arria 10 devices, the Native PHY IP core (Interlaken mode) uses external transmit PLLs. Instantiate the external transceiver PLLs and then connect the transmit serial clock output to the `tx_serial_clk` input (see **Signals**). The Seriallite III Streaming IP core uses a transmit serial clock bus input bus (`tx_serial_clk`) and `tx_pll_locked` input to connect the external transmit PLL to the Arria 10 Native PHY IP core. Refer to the *Arria 10 Transceiver PHY User Guide* for more information.

Related Information

- **Arria 10 Transceiver PHY User Guide**
  For more information about the Arria 10 Native PHY IP core.
- **Altera Transceiver PHY IP Core User Guide**
  For more information about the Interlaken PHY IP core.
- **Signals** on page 4-16

Clock Domains

The SerialLite III Streaming IP core contains different clock domains, depending on the clocking mode. In addition to these clock domains, there are another four clock domains in isolation within the transceivers.

Table 4-2: SerialLite III Streaming IP Core Clock Domains and Signals

Clock Domain		Description	Standard Clocking Mode	Advanced Clocking Mode
Source Core	user_clock	Source user interface clock	X	X
	phy_mgmt_clk	Source Native PHY or Interlaken PHY IP core reconfiguration interface clock	X	X
	pll_ref_clk	Source transceiver reference clock (Stratix V and Arria V GZ only)	X	X
	tx_coreclkin	Source core clock (in standard clocking mode)	X	
	tx_clkout	Source core clock (in advanced clocking mode)		X
	tx_serial_clk	Transmit transceiver clock (Arria 10 only)	X	X
Sink Core	user_clock	Sink user interface clock (in standard clocking mode)	X	
	phy_mgmt_clk	Sink Native PHY or Interlaken PHY IP core reconfiguration interface clock	X	X
	xcvr_pll_ref_clk	Sink transceiver reference clock	X	X
	rx_coreclkin	Sink core clock (in standard clocking mode)	X	
	rx_clkout	Sink core and user interface clock (in advanced clocking mode)		X
Duplex Core	user_clock_tx	Source user interface clock	X	X
	user_clock_rx	Sink user interface clock (in standard clocking mode)	X	
	phy_mgmt_clk	Native PHY or Interlaken PHY IP core reconfiguration interface clock	X	X
	xcvr_pll_ref_clk	Transceiver reference clock	X	X
	tx_coreclkin	Source core clock (in standard clocking mode)	X	
	tx_clkout	Source core clock (in advanced clocking mode)		X
	rx_coreclkin	Sink core clock (in standard clocking mode)	X	
	rx_clkout	Sink core and user interface clock (in advanced clocking mode)		X
	tx_serial_clk	Transmit transceiver clock (Arria 10 only)	X	X

Core Clocking

The SerialLite III Streaming IP core comes with standard and advanced clocking modes; select the mode in the parameter editor.

Table 4-3: Comparing Standard and Advanced Clocking Modes

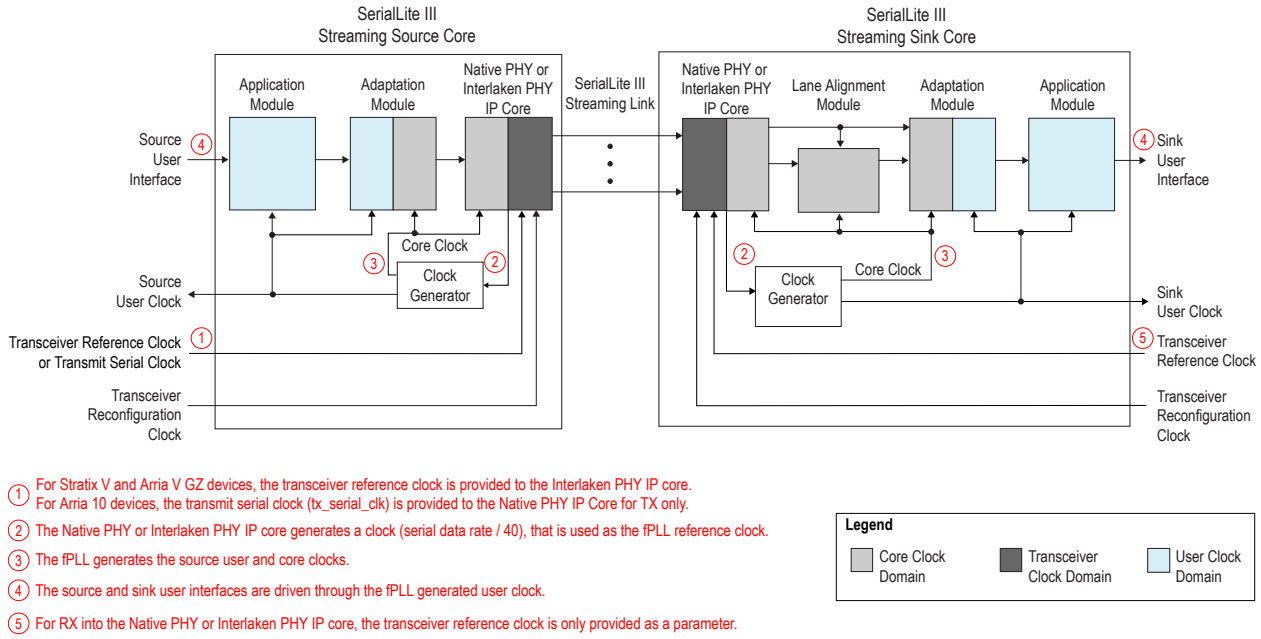
Resource	Standard Mode	Advanced Mode	Description
Source user clocking	Core generated	User provided	If the PPM difference between the generated and user clocks is not acceptable, use the advanced clocking mode.
MAC fPLL	Uses one fPLL per direction	Does not use fPLLs	If the design uses many fPLLs and clock crossing is an issue in the user environment, use the advanced clocking mode.
Transmission overhead	1.1 x <*input data rate*>	<*Interlaken Overhead*> x <*input data rate*>	The advanced clocking mode overhead is less than the standard clocking mode overhead.
Streaming variation	Pure streaming where the output data appears exactly as it was input	Output streaming data is accompanied by numerous empty clock cycles	If empty cycles (where no valid data is present) at the output are intolerable, use pure streaming (standard clocking mode). Alternatively, create your own sink interface to remove the empty cycles.
Sink interface	Fixed	You can include your own logic or FIFO to receive the output data	**Advanced Clocking Mode** on page 4-11

Standard Clocking Mode

In the standard clocking mode, the SerialLite III Streaming IP core operates in a pure streaming manner, exactly replicating the source input data at the sink end. The SerialLite III Streaming IP core generates the user clock at both the source and sink to drive the user interface.

In this mode, you initially specify the user clock frequency through the SerialLite III Streaming parameter editor. The Quartus II software then automatically calculates the reference clock coming from the Native PHY or Interlaken PHY IP core and the two clock outputs from the fPLL in the clock generator module. After the calculation, the Quartus II software provides a list of transceiver reference clock values for you to select. Depending on the clock constraints, the generated value for the user clock should be very close, if not identical, to the user clock frequency that you specify. The Quartus II software shows the generated user clock value as well as transceiver reference clock values.

Figure 4-8: SerialLite III Streaming IP Core Block Diagram in Standard Clocking Mode



Note: The SerialLite III Streaming IP core uses the transmit serial clock bus (`tx_serial_clk`) and the `tx_pll_locked` signal to connect the external transmit PLL to the Arria 10 Native PHY IP core.

Advanced Clocking Mode

The advanced clocking mode allows the user to use a user-specified clock to interface with the source core. This mode is useful when PPM differences between the user clock (generated by the fPLL) and the user's interface clock are intolerable. In the advanced clocking mode, the source core is generated with the PPM-absorption FIFO wrapper module.

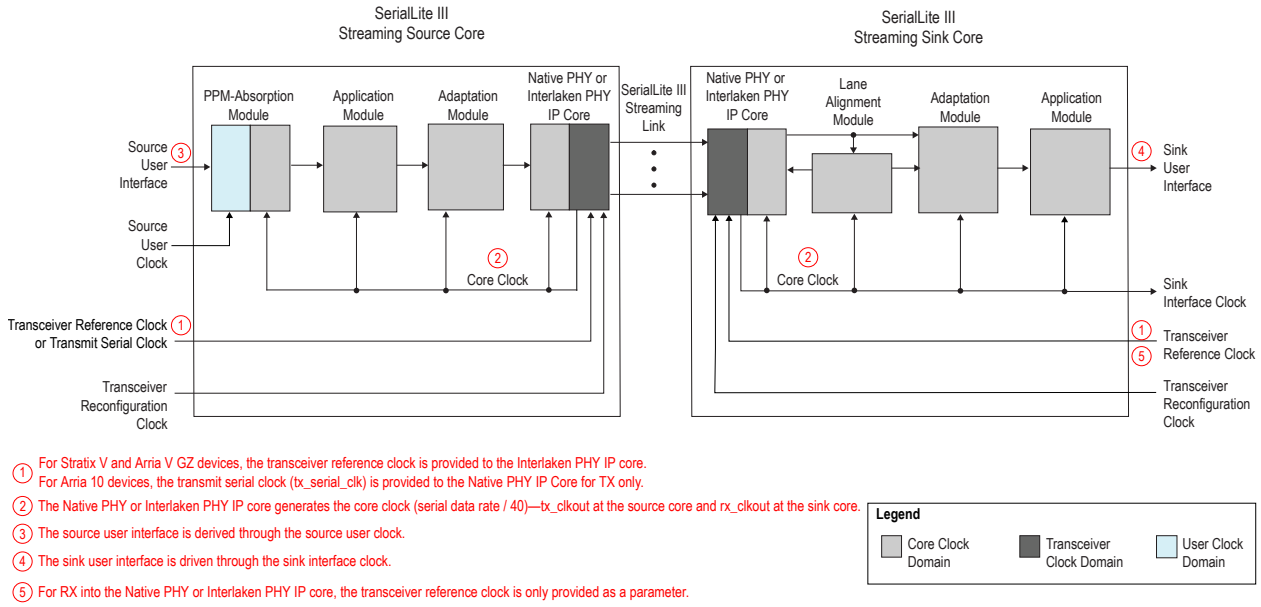
Similar to the standard clocking mode, you must specify the user clock frequency through the SerialLite III Streaming parameter editor. Based on the user clock frequency value, the Quartus II software automatically calculates the lane rate and the core clock.

The parameter editor provides guidance in selecting a source user clock frequency that meets the transceiver data rate constraints. For more information about the lane rate calculation, refer to the “Transmission Overheads and Lane Rate Calculations” section.

In advanced clocking mode, the core clock is faster than the source user clock when data is inserted in the core. Therefore, the sink user interface may run out of valid data to transmit. The valid signal at the sink user interface is deasserted to indicate an absence of data at the sink core since the core clock is greater than the user clock.

Note: The core operates at higher clock rates in Advanced Clocking Mode. Therefore, when operating in this mode, it may be difficult to close timing at higher data rates and/or number of lanes.

Figure 4-9: SerialLite III Streaming IP Core Block Diagram in Advanced Clocking Mode



Note: The SerialLite III Streaming IP core uses the transmit serial clock bus (`tx_serial_clk`) and the `tx_pll_locked` signal to connect the external transmit PLL to the Arria 10 Native PHY IP core.

Related Information

- **Transmission Overheads and Lane Rate Calculations** on page 4-12

Core Latency

The table below lists the latency measurement for the SerialLite III Streaming duplex core in standard and advanced clocking mode. An average value is taken from a set of samples during hardware testing.

Table 4-4: Latency Measurement for Duplex Core

Clocking Mode	Parameters		Latency (ns)
	Number of Lanes	Per-Lane Data Rate (Mbps)	
Standard	5	10312.50	362
Advanced	5	10312.50	281

Transmission Overheads and Lane Rate Calculations

The SerialLite III Streaming IP core lane data rate (transceiver data rate) is composed of the input data rate and transmission overheads.

Lane Rate = Input Data Rate × Transmission Overheads

The parameter editor uses the above equation to ensure that the lane rate is within the maximum supported transceiver lane rates. This puts an upper limit on the input data rate or the user clock frequency, where the user clock frequency equates to:

User Clock Frequency = Input Data Rate/64

The SerialLite III Streaming IP core uses the Interlaken protocol for transferring data and therefore incurs encoding and metaframe overheads. In the standard clocking mode, the IP core employs an fPLL for clock generation. To ensure that the fPLL generates the clock as close as possible to the user clock specified by you, the fPLL incurs additional overheads. The transmission overheads can thus be derived in the following functions:

Transmission Overheads = Maximum (Interlaken Overheads, fPLL Overheads)

where,

Interlaken Overheads = 67/64 × (MetaFrame Length) / (MetaFrame length - 4)

To ensure the Interlaken interoperability as well as user clocking requirements, the fPLL overheads in the standard clocking mode are chosen to be slightly higher than the Interlaken overheads.

Transmission Overheads in standard clocking mode = 1.1

Note: Assuming maximum metaframe overhead with a metaframe size of 200, the standard clocking mode overheads are independent of Interlaken overheads. For more details, refer to the SerialLite III data efficiency calculator.

Tip: You can obtain the SerialLite III Streaming MegaCore Function Data Efficiency Calculator for 28 nm Altera devices from your local Altera sales representative or by emailing **SLIII_support@altera.com**.

Therefore, the lane rate in the standard clocking mode equals:

Lane Rate = Input Data Rate × 1.1

In the advanced clocking mode, the transmission overheads equals the Interlaken overheads because no fPLL is present. Therefore, the lane rate in advanced clocking mode equals:

Lane Rate = Input Data Rate × Interlaken overheads

Reset

Each core has a separate active high reset signal, `core_reset` , that asynchronously resets all logic in the core.

Each core also includes the Native PHY or Interlaken PHY IP reset signal, `phy_mgmt_clk_reset`. This reset signal must be on the same clock domain as the clock used to drive the reconfiguration controllers, `phy_mgmt_clk`. The Native PHY or Interlaken PHY IP core requires the assertion of this reset signal to synchronize with the reconfiguration controller reset signal.

Note: Altera recommends using the same reset signals for both the Native PHY or Interlaken PHY IP core and the reconfiguration controller.

Link-Up Sequence

Refer to the topics on source and sink core link debugging for information about the transmit and receive core link-up sequence.

Related Information

- **Source Core Link Debugging** on page 5-7
- **Sink Core Link Debugging** on page 5-9

CRC-32 Error Injection

In the Quartus II software version 13.1 and higher, the SerialLite III IP core supports CRC error injection with the 10G PCS CRC-32 generator. This feature enables corruption of the CRC-32 value of the CRC-32 generator.

To insert CRC errors for a given lane, the IP interface includes a CRC error injection control signal. Asserting this control signal inserts CRC errors for all the lanes and transceivers that have enabled support for error injection. The error injection for a given transceiver is enabled by setting CRAM bits using DPRIO. The provided example design demonstrates how to use the Nios II processor to set the respective CRAM bits.

Related Information

- **SerialLite III Streaming IP Core Design Example for Stratix V Devices** on page 5-1

FIFO ECC Protection

In the Quartus II software version 13.1 and higher, the SerialLite III IP core can be protected from Single-Event Upset (SEU) changes using error correcting code (ECC) protection. You enable this feature using the ECC protection option in the parameter editor. The ECC protection provides additional error status bits that tell you if the ECC was able to perform a correction from the SEU change or if an uncorrectable error has occurred.

Note: Enabling ECC protection incurs additional logic and latency overhead.

User Data Interface Waveforms

The following waveforms apply to the source user interface in source-only and duplex cores.

Figure 4-10: Source Waveform for Burst Mode

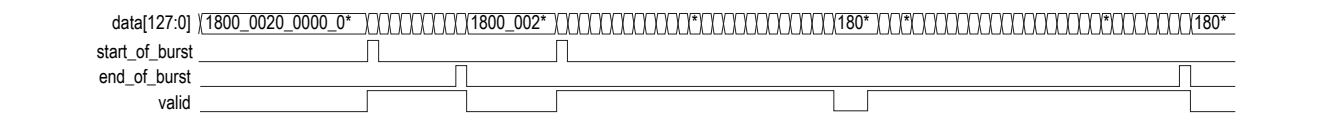


Figure 4-11: Source Waveform for Burst Mode (Sync)

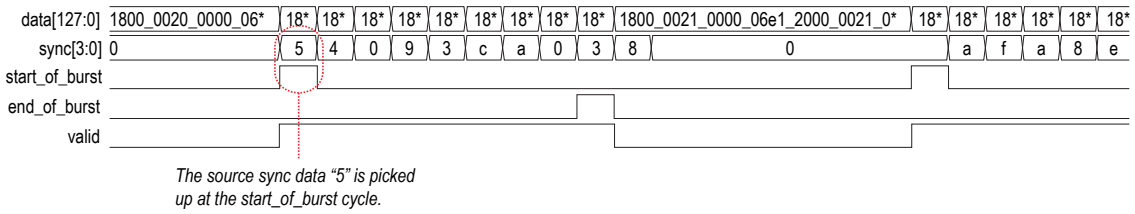
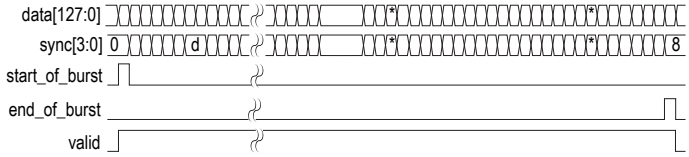


Figure 4-12: Source Waveform for Continuous Mode



- `start_of_burst` pulses for one clock cycle, indicating that the data burst starts at that clock cycle.
- `end_of_burst` pulses for one clock cycle, indicating that the data burst ends at that clock cycle.
- The `valid` signal indicates valid data. It should be turned off between two data bursts that are between the current data burst's `end_of_burst` clock cycle and next data burst's `start_of_burst` clock cycle. The `valid` signal can be pulled low in the middle of a data burst transferring between the same data burst's `start_of_burst` and `end_of_burst`, indicating non-valid data at that clock cycle.
- The sync vector is used in burst mode. It is valid only when `start_of_burst` and `valid` are high.

The following waveforms apply to the sink user interface in sink-only and duplex cores.

Figure 4-13: Sink Waveform for Burst Mode

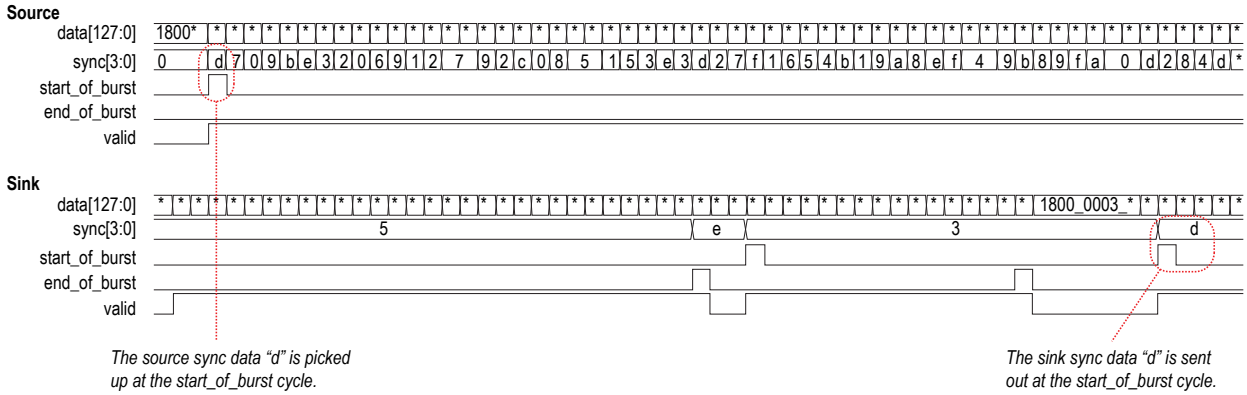
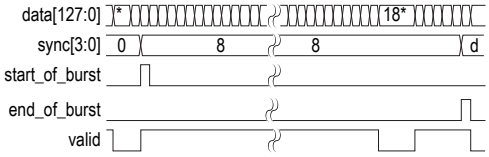


Figure 4-14: Sink Waveform for Continuous Mode



- `start_of_burst` pulses for one clock cycle, indicating that the data burst starts at that clock cycle.
- `end_of_burst` pulses for one clock cycle, indicating that the data burst ends at that clock cycle.
- The `valid` signal indicates valid data. It is turned off between two data bursts that are between the current data burst's `end_of_burst` clock cycle and the next data burst's `start_of_burst` clock cycle. The valid signal can be pulled low in the middle of a data burst after a data burst's `start_of_burst` and before the data burst's `end_of_burst`, indicating non-valid data at that clock cycle.
- The sync vector is used in burst mode. The sync data picked up at the source's `start_of_burst` high cycle is sent out at the sink as shown in the waveform.

Signals

The following tables list all the input and output signals of the SerialLite III Streaming IP core.

Table 4-5: SerialLite III Streaming IP Core Source Core Signals

Signal	Width	Clock Domain	Direction	Description
`tx_serial_clk`	*N*	N.A.	Input	This high-speed serial clock input from the external transceiver PLL. The width is the same as the number of lanes specified in the parameter editor. Each bit of the vector corresponds to serial clock of the transmit channel. (Arria 10 devices only) *N* represents the number of lanes.
`tx_pll_locked`	1	N.A.	Input	This signal indicates that all external transceiver PLLs are locked. If more than one external transceiver PLL is required for higher lanes, each instantiation outputs a bit that indicates whether the PLL providing the high-speed clock for a corresponding transceiver has achieved its lock status. The `pll_locked` output signal from the external transceiver PLLs should be ANDed together before being input to the IP core. (Arria 10 devices only)
`core_reset`	1	N.A.	Input	Asynchronous master reset for the core. Assert this signal high to reset all logic, including the internal clocking components.

Signal	Width	Clock Domain	Direction	Description
xcvr_pll_ref_clk	1	N.A.	Input	Reference clock for the transceivers. This signal is not used for Arria 10 devices in the source only direction.
user_clock	1	N.A.	Input/ Output	Clock for data transfers across the source core interface. • Input: Using advanced clocking mode • Output: Using standard clocking mode
user_clock_reset	1	user_ clock	Input/ Output	In the standard clocking mode, the core asserts this signal when the core_reset signal is high and deasserts this signal when the reset sequence is complete. In the advanced clocking mode, the core asserts this signal to reset the adaptation module FIFO buffer. • Input: Using advanced clocking mode • Output: Using standard clocking mode
reconfig_clk	1	N.A.	User application to IP core	This clock is for the transceiver reconfiguration interface. It also sequences the reset state machine in the clock generation logic.
link_up	1	user_ clock.	Output	The core asserts this signal to indicate that the core initialization is complete and is ready to transmit user data.
data	64x*N*	user_ clock	Input	This vector carries the transmitted streaming data to the core. *N* represents the number of lanes.
sync	4	user_ clock	Input	The sync vector is a 4 bit bus. The data value at the start of a burst is captured and transported across the link. **Note:** This vector is not associated with Interlaken channelization or flow control schemes.
valid	1	user_ clock	Input	This vector indicates that the transmitted streaming data is valid.

Signal	Width	Clock Domain	Direction	Description
start_of_burst	1	user_clock	Input	When you configure the core for burst mode operation, asserting this signal indicates that the information on the data vector is the beginning of a burst. Because continuous mode is one long burst, in this mode the signal is asserted only once at the start of the data.
end_of_burst	1	user_clock	Input	When you configure the core for burst mode operation, asserting this signal indicates that the information on the data vector is the end of a burst. You can optionally send an end of burst signal at the end of continuous mode.
error	3 or 4	user_clock	Output	This vector indicates an overflow in the source adaptation module's FIFO buffer. • Bit 0: Source adaptation module's FIFO buffer overflow • Bit 1: Source PPM-absorption module's FIFO buffer overflow • Bit 2: An SEU error occurred and was corrected (ECC enabled) Don't care (ECC disabled) • Bit 3: An SEU error occurred and could not be corrected (ECC enabled) Don't care (ECC disabled) The width of this signal depends on the clocking mode: • 3: Standard clocking mode • 4: Advanced clocking mode
crc_error_inject	1	user_clock	Input	This signal is used for CRC-32 error injection.

Table 4-6: SerialLite III Streaming IP Core Sink Core Signals

Signal	Width	Clock Domain	Direction	Description
core_reset	1	N.A.	Input	Asynchronous master reset for the core. Assert this signal high to reset all logic, including the internal clocking components.

Signal	Width	Clock Domain	Direction	Description
xcvr_pll_ref_clk	1	N.A.	Input	Reference clock for the transceivers.
user_clock	1	N.A.	Output	Clock for data transfers across the sink core interface in the standard clocking mode.
user_clock_reset	1	user_clock	Output	The core asserts this signal when the core_reset signal is high and deasserts this signal when the reset sequence is complete in the standard clocking mode.
interface_clock	1	core_clock	Output	Clock for data transfer across the sink core interface in the advanced clocking mode.
interface_clock_reset	1	core_clock	Output	The core asserts this signal when the core_reset signal is high and deasserts this signal when the reset sequence is complete in the advanced clocking mode.
link_up	1	Standard clocking: user_clock Advanced clocking: core_clock	Output	The core asserts this signal to indicate that the core initialization is complete and is ready to transmit user data.
data	64x*N*	Standard clocking: user_clock Advanced clocking: core_clock	Output	This vector carries the transmitted streaming data from the core. *N* represents the number of lanes.
sync	4	Standard clocking: user_clock Advanced clocking: core_clock	Output	The sync vector is a 4 bit bus. The data value at the start of a burst is captured and transported across the link. **Note:** This vector is not associated with Interlaken channelization or flow control schemes.
valid	1	Standard clocking: user_clock Advanced clocking: core_clock	Output	This vector indicates that the data is valid.

Signal	Width	Clock Domain	Direction	Description
start_of_burst	1	Standard clocking: user_clock Advanced clocking: core_clock	Output	When you configure the core for burst mode operation, assertion of this signal indicates that the information on the data vector is the beginning of a burst. Because continuous mode is one long burst, in this mode the signal is asserted only once at the start of the data.
end_of_burst	1	Standard clocking: user_clock Advanced clocking: core_clock	Output	When you configure the core for burst mode operation, assertion of this signal indicates that the information on the data vector is the end of a burst.
error	*N*+5	Standard clocking: user_clock Advanced clocking: core_clock	Output	This vector indicates the state of the sink adaptation module's FIFO buffer. *N* represents the number of lanes: • [*N*+4]: An SEU error occurred and could not be corrected (ECC enabled); Don't care (ECC disabled) • [*N*+3]: An SEU error occurred and was corrected (ECC enabled); Don't care (ECC disabled) • [*N*+2]: FIFO buffer overflow • [*N*+1]: FIFO buffer underflow • [*N*]: Loss of alignment • [*N*-1:0]: RX CRC 32 error

Table 4-7: SerialLite III Streaming IP Core Duplex Core Signals

Signal	Width	Clock Domain	Direction	Description
tx_serial_clk	*N*	N.A.	Input	This high-speed serial clock input from the external transceiver PLL. The width is the same as the number of lanes specified in the parameter editor. Each bit of the vector corresponds to serial clock of the transmit channel. (Arria 10 devices only) *N* represents the number of lanes.

Signal	Width	Clock Domain	Direction	Description
tx_pll_locked	1	N.A.	Input	This signal indicates that all external transceiver PLLs are locked. If more than one external transceiver PLL is required for higher lanes, each instantiation outputs a bit that indicates whether the PLL providing the high-speed clock for a corresponding transceiver has achieved its lock status. The pll_locked output signal from the external transceiver PLLs should be ANDed together before being input to the IP core. (Arria 10 devices only)
core_reset	1	N.A.	Input	Asynchronous master reset for the core. Assert this signal high to reset all logic, including the internal clocking components.
xcvr_pll_ref_clk	1	N.A.	Input	Reference clock for the transceivers.
user_clock_tx	1	N.A.	Input/ Output	Clock for data transfers across the transmit interface. • Input: Using advanced clocking mode • Output: Using standard clocking mode
user_clock_ reset_tx	1	user_clock_tx	Input/ Output	In the standard clocking mode, the core asserts this signal when the core_reset signal is high and deasserts this signal when the reset sequence is complete. In the advanced clocking mode, the core asserts this signal to reset the adaptation module FIFO buffer. • Input: Using advanced clocking mode • Output: Using standard clocking mode
interface_clock_ reset_tx	1	core_clock	Output	In the advanced clocking mode, the core asserts this signal when the core_reset signal is high and deasserts this signal when the reset sequence is complete.
link_up_tx	1	Standard clocking: user_ clock Advanced clocking: core_ clock	Output	The core asserts this signal to indicate that the core initialization is complete and is ready to transmit user data.

Signal	Width	Clock Domain	Direction	Description
data_tx	64x*N*	Standard clocking: user_clock Advanced clocking: core_clock	Input	This vector carries the transmitted streaming data to the core. *N* represents the number of lanes.
sync_tx	4	Standard clocking: user_clock Advanced clocking: core_clock	Input	The sync vector is a 4 bit bus. The data value at the start of a burst is captured and transported across the link. **Note:** This vector is not associated with Interlaken channelization or flow control schemes.
valid_tx	1	Standard clocking: user_clock Advanced clocking: core_clock	Input	This vector indicates that the data is valid.
start_of_burst_tx	1	Standard clocking: user_clock Advanced clocking: core_clock	Input	When you configure the core for burst mode operation, assertion of this signal indicates that the information on the data vector is the beginning of a burst. Because continuous mode is one long burst, in this mode the signal is asserted only once at the start of the data.
end_of_burst_tx	1	Standard clocking: user_clock Advanced clocking: core_clock	Input	When you configure the core for burst mode operation, assertion of this signal indicates that the information on the data vector is the end of a burst.

Signal	Width	Clock Domain	Direction	Description
error_tx	3 or 4	Standard clocking: user_clock Advanced clocking: core_clock	Output	This vector indicates an overflow in the source adaptation module's FIFO buffer. • Bit 0: Source adaptation module's FIFO buffer overflow • Bit 1: Source PPM-absorption module's FIFO buffer overflow ECC option: • MSB-1: An SEU error occurred and was corrected (ECC enabled). This bit is bit 2 in advanced clocking mode and bit 1 in standard clocking mode. Don't care (ECC disabled) • MSB: An SEU error occurred and could not be corrected (ECC enabled). This bit is bit 3 in advanced clocking mode and bit 2 in standard clocking mode. Don't care (ECC disabled) The width of this signal depends on the clocking mode: • 3: Using standard clocking mode • 4: Using advanced clocking mode
user_clock_rx	1	N.A.	Output	Clock for data transfers across the sink core interface in the standard clocking mode.
user_clock_reset_rx	1	user_clock_rx	Output	In the standard clocking mode, the core asserts this signal when the core_reset signal is high and deasserts this signal when the reset sequence is complete.
interface_clock_rx	1	core_clock	Output	Clock for data transfers across the sink core interface in the advanced clocking mode.
interface_clock_reset_rx	1	core_clock	Output	In the advanced clocking mode, the core asserts this signal when the core_reset signal is high and deasserts this signal when the reset sequence is complete.

Signal	Width	Clock Domain	Direction	Description
link_up_rx	1	Standard clocking: user_clock Advanced clocking: core_clock	Output	The core asserts this signal to indicate that the core initialization is complete and is ready to transmit user data.
data_rx	64x*N*	Standard clocking: user_clock Advanced clocking: core_clock	Output	This vector carries the transmitted streaming data from the core. *N* represents the number of lanes.
sync_rx	4	Standard clocking: user_clock Advanced clocking: core_clock	Output	The sync vector is a 4 bit bus. The data value at the start of a burst is captured and transported across the link. **Note:** This vector is not associated with Interlaken channelization or flow control schemes.
valid_rx	1	Standard clocking: user_clock Advanced clocking: core_clock	Output	This vector indicates that the data is valid.
start_of_burst_rx	1	Standard clocking: user_clock Advanced clocking: core_clock	Output	When you configure the core for burst mode operation, asserting this signal indicates that the information on the data vector is the beginning of a burst. Because continuous mode is one long burst, in this mode the signal is asserted only once at the start of the data.
end_of_burst_rx	1	Standard clocking: user_clock Advanced clocking: core_clock	Output	When you configure the core for burst mode operation, asserting this signal indicates that the information on the data vector is the end of a burst. You can optionally send an end of burst signal at the end of continuous mode.

Signal	Width	Clock Domain	Direction	Description
error_rx	*N*+5	Standard clocking: user_clock Advanced clocking: core_clock	Output	This vector indicates the state of the sink adaptation module's FIFO buffer. *N* represents the number of lanes: • [*N*+4]: An SEU error occurred and could not be corrected (ECC enabled); Don't care (ECC disabled) • [*N*+3]: An SEU error occurred and was corrected (ECC enabled); Don't care (ECC disabled) • [*N*+2]: FIFO buffer overflow • [*N*+1]: FIFO buffer underflow • [*N*]: Loss of alignment • [*N*-1:0]: RX CRC 32 error
crc_error_inject	1	Standard clocking: user_clock_tx Advanced clocking: core_clock_tx	Input	This signal is used for CRC-32 error injection.

Table 4-8: Interlaken PHY IP Core Signals and Native PHY IP Core Signals (Interlaken Mode)

Signal	Width	Clock Domain	Direction	Description
phy_mgmt_clk	1	N.A.	Input	Clock input for the Avalon-MM PHY management interface within the Interlaken PHY IP core or Native PHY IP core. This signal also clocks the transceiver reconfiguration interface and sequences the reset state machine in the clock generation logic.
phy_mgmt_clk_reset	1	phy_mgmt_clk	Input	Global reset signal that resets the entire Interlaken PHY IP core or Native PHY IP core. This signal is active high and level sensitive.
phy_mgmt_addr[*N*:0]	9 (Stratix V and Arria V GZ) 11 - 16 (Arria 10)	phy_mgmt_clk	Input	Control and status register (CSR) address. For Arria 10 devices, the width depends on the number of lanes. The parameter editor determines the required width for you.
phy_mgmt_writedata[31:0]	32	phy_mgmt_clk	Input	CSR write data.

Signal	Width	Clock Domain	Direction	Description
phy_mgmt_readdata[31:0]	32	phy_mgmt_clk	Output	CSR read data.
phy_mgmt_write	1	phy_mgmt_clk	Input	Active high CSR write signal.
phy_mgmt_read	1	phy_mgmt_clk	Input	Active high CSR read signal.
phy_mgmt_waitrequest	1	phy_mgmt_clk	Output	CSR read or write request signal. When asserted, this signal indicates that the Avalon-MM slave interface is unable to respond to a read or write request.
reconfig_busy	1	phy_mgmt_clk	Input	When asserted, this signal indicates that a reconfiguration operation is in progress and no further reconfiguration operations should be performed. You can monitor this signal to determine the status of the Transceiver Reconfiguration Controller.
reconfig_to_xcvr	• Source core: 140x*N* • Sink core: 70x*N* • Duplex core: 140x*N*	phy_mgmt_clk	Input	Dynamic reconfiguration input for the hard transceiver. (Stratix V and Arria V GX devices only) *N* represents the number of lanes.
reconfig_from_xcvr	• Source core: 92x*N* • Sink core: 46x*N* • Duplex core: 92x*N*	phy_mgmt_clk	Output	Dynamic reconfiguration output for the hard transceiver. (Stratix V and Arria V GX devices only) *N* represents the number of lanes.
tx_serial_data	*N*	—	Output	The serial output data from the core. *N* represents the number of lanes.
rx_serial_data	*N*	—	Input	The serial input data to the core. *N* represents the number of lanes.

Note: For Arria 10 devices, the phy_mgmt bus interface connects to the reconfiguration interface of the instantiated Native PHY IP core.

Related Information

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

SerialLite III Streaming IP Core Design Guidelines

5

2014.08.18

UG-01126 Subscribe Send Feedback

SerialLite III Streaming IP Core Design Example for Stratix V Devices

The SerialLite III Streaming IP core for Stratix V devices includes design examples for its four variations:

- SerialLite III Streaming simplex core in standard clocking mode
- SerialLite III Streaming duplex core in standard clocking mode
- SerialLite III Streaming simplex core in advanced clocking mode
- SerialLite III Streaming duplex core in advanced clocking mode

Note: The SerialLite III streaming IP core does not include a hardware design example for Arria 10 devices. Instead, use the provided Arria 10 example testbench design as a guide to implement your design. Refer to **Arria 10 Simulation Testbench** for details.

The IP core variations were generated using the parameter editor. These designs serve as a demonstration platform for highlighting the core's features and also show how to integrate the core in a typical system environment.

Figure 5-1: Design Example for Simplex Core in Standard Clocking Mode

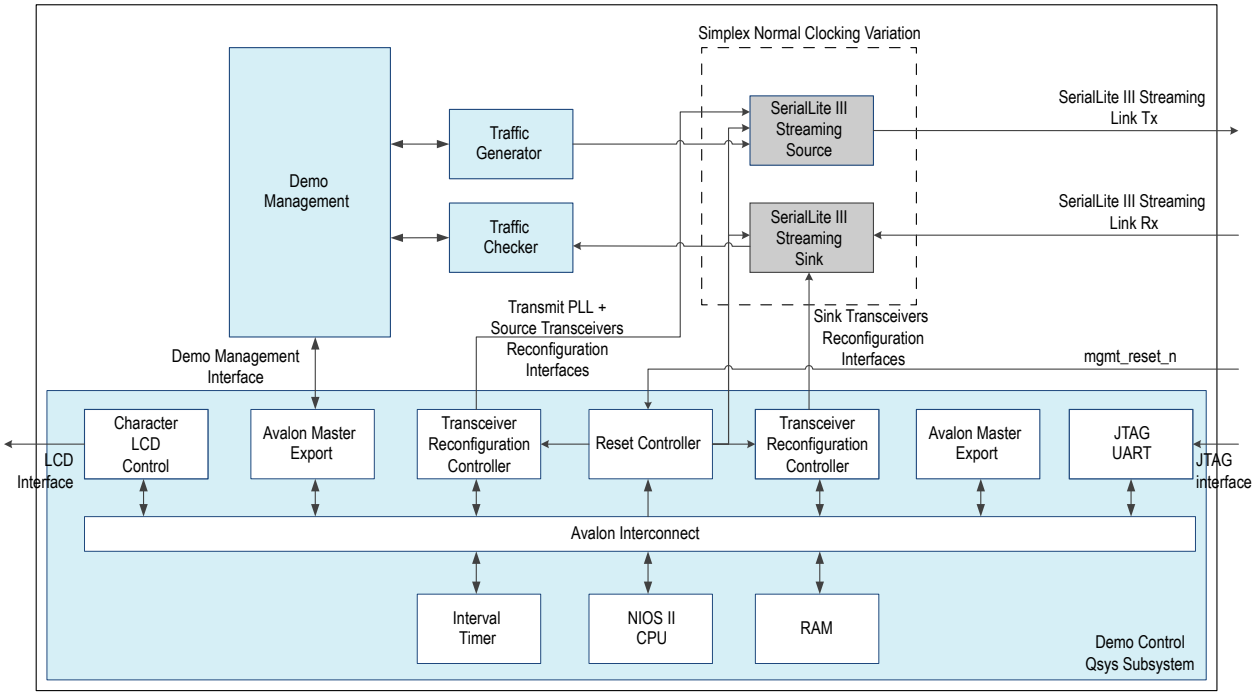
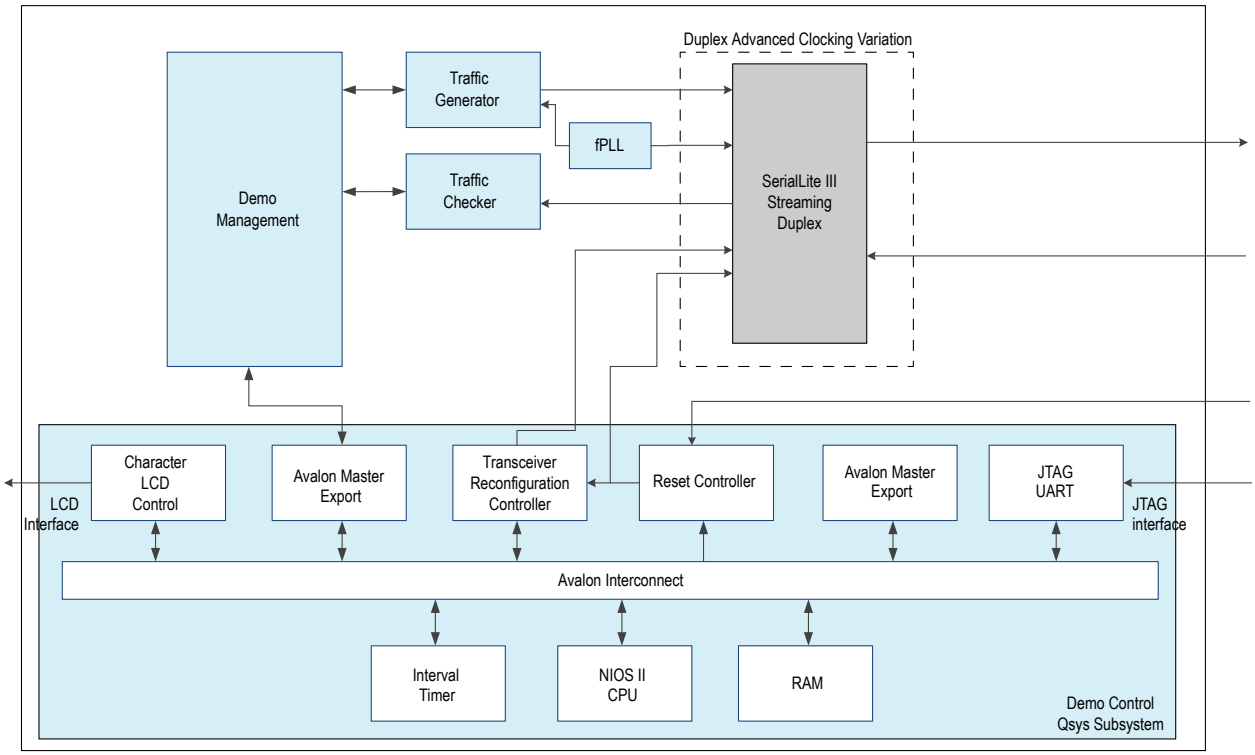


Figure 5-2: Design Example for Duplex Core in Advanced Clocking Mode



Design Example Components

The design example consists of following components:

- SerialLite III Streaming IP core variation
- Source fPLL (to generate source user clock in advanced clocking mode)
- Traffic generator
- Traffic checker
- Demo control
- Demo management

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 Quartus II software.

Source User Clock

The fPLL is available only in designs utilizing the advanced clocking mode to generate a user clock for sourcing data into the SerialLite III Streaming IP core.

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 5-3: Traffic Generator Sample Word Format

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

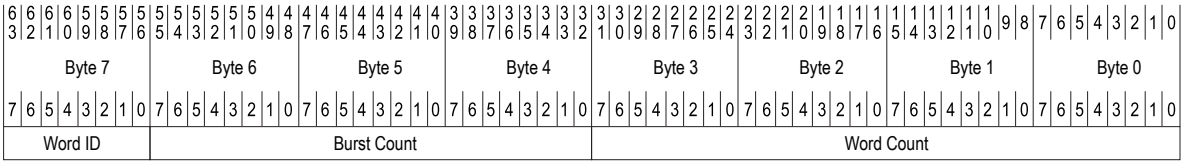


Table 5-1: Traffic Generator Sample Word Fields

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.

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.

Demo Control

The demo control module is a Nios® II processor system, generated in Qsys, to control the demo hardware. In addition to the Nios II processor system, this module also includes reconfiguration controllers for the transceivers and PLL channels in the SerialLite III Streaming IP core. The number of reconfiguration interfaces equal to the number of transceivers plus PLL channels for the source and duplex cores, and the number of transceivers for the sink cores.

Demo Management

The demo management module implements CSRs to control and monitor the design operation. This includes CSRs to monitor and log errors that occur during the operation.

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 is connected to lane 0 in the example design. 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.

Design Setup

The design example targets the Transceiver Signal Integrity Development Kit, Stratix V GT Edition. The design includes an SDC script as well as a QSF with verified constraints and settings for a 2-lane design in loopback. If you use the design example with another device or development board, you may need to update the device setting and constraints.

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 the three channels that share an Avalon-MM interface to different Transceiver Reconfiguration Controller IP cores or you will receive a Fitter error.

Related Information

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

Design Example Compilation and Download

After generating the IP core design example, you can compile the design example for a SerialLite III Streaming two lane loopback design. The design example files are located in the <*variation name*>**_example/seriallite_iii_sv** directory.

Note: The design example consists of a Qsys subsystem and Nios II processor system. You must compile both systems for the design example to operate correctly.

To compile the design example Qsys subsystem, perform the following steps:

1. Open a Nios II command window.
2. Change the project directory to **/demo_control/**.
3. Type the following command to set up the required libraries and compile the generated design example: `>source build_demo_control.sh`
4. In the Quartus II software, change the directory to **/demo/** and open the **seriallite_iii_streaming_demo.qpf** file.
5. Compile the **seriallite_iii_streaming_demo** project in the Quartus II software.
6. If you have the supported development kit, download the <*project name*>**. sof** file onto the board . Refer to the *Development Kits/Cables* page of the Altera website for more information.

To compile the design example Nios II processor system, perform the following steps:

1. In a Nios II command window, change the directory to **/demo_control/software**.
2. Type the following command to compile the Nios II processor: `> source batch_script.sh`

The script generates a **demo_control.elf** file under the **/app/** directory. This file can later be downloaded into the FPGA.

To download the design example and subsystem, and operate the design, perform the following steps:

1. Start the Quartus II software.
2. In the Tools menu, click **Qsys**.
3. In the Qsys Tools menu, click **Nios Command Shell [gcc4]** to launch the Nios II command shell.
4. Type the following command to download the **demo_control.elf** file into the FPGA on the board and to specify the USB cable number (`$CABLE_NUMBER`): `>nios2-download -g -r $CABLE_NUMBER ../demo_control/software/app/demo_control.elf`
5. Type the following command to start a terminal connection with the board (using the same cable number): `>nios2-terminal $CABLE_NUMBER`

The terminal should now display an interactive session for the SerialLite III Streaming IP core design example.

Related Information

- **Design Example Compilation and Download** on page 5-5
- **Development Kits/Cables**
- **Quartus II Incremental Compilation for Hierarchical and Team-Based Design**
  More information about the design compilation.

- **Nios II Processor**
  More information about the Nios II processor and its use.

Design Example Operation

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:

1. *Enable source*—Enables the traffic generator and start sending out data.
2. *Disable source*—Disables traffic generation.
3. *Reset source*—Resets the source core and traffic generator.
4. *Reset sink*—Resets the sink core and traffic checker.
5. *Display error statistics*—Displays the error statistics.
6. *Toggle burst/continuous mode*—Resets the source and sink MACs and toggles the traffic generator to generate a burst or continuous traffic stream.
7. *Toggle CRC error injection for lane 0*—Turns CRC error injection off or on for lane 0.

SerialLite III Streaming Link Debugging

The following section describes the link-up sequence that you can use when debugging the SerialLite III Streaming IP core.

Source Core Link Debugging

Figure 5-4: Source Core Link Debugging Flow Chart

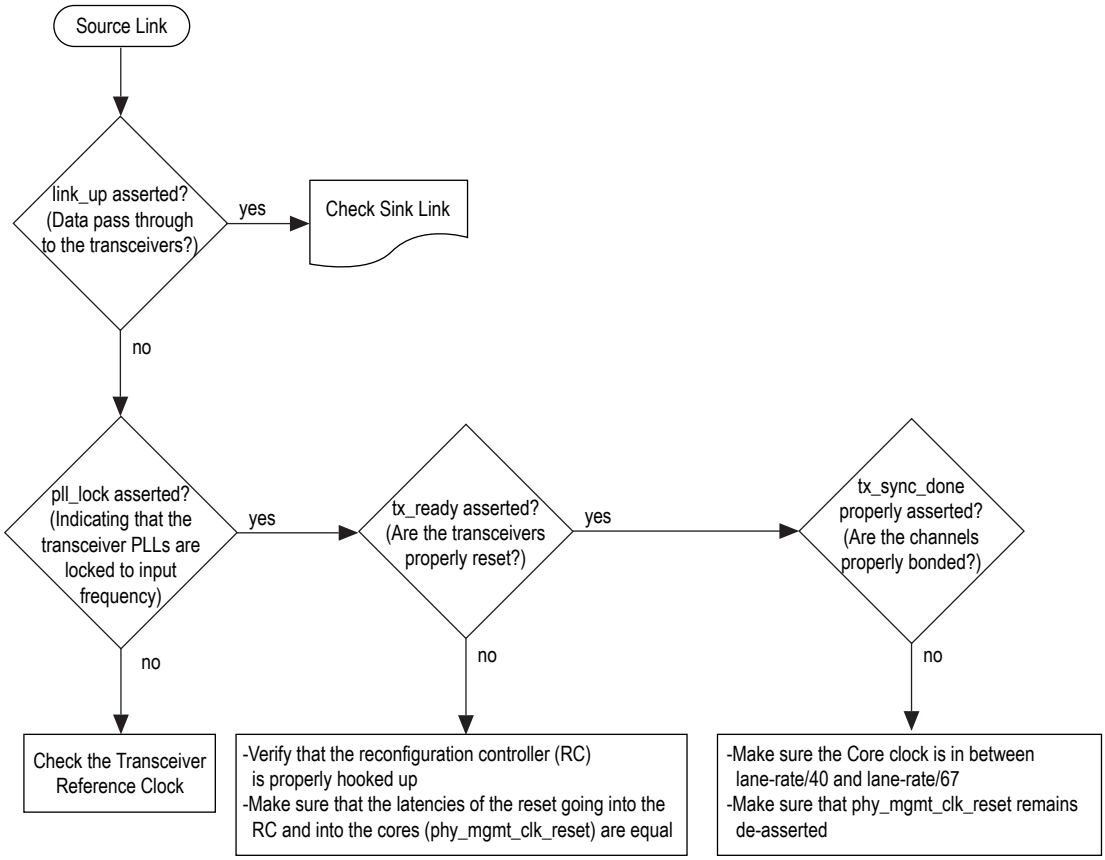


Table 5-2: Source Link Debugging Signals

Signal Name	Location	Description
`link_up`	Top level source signal	The core asserts this signal to indicate that initialization sequence is complete and the core is ready to transmit the data.
`xcvr_pll_locked`	**/source/xcvr_pll_locked**	This active high signal indicates that the transceivers are locked to the reference clock.
`tx_ready`	**/source/tx_ready**	This active high signal indicates that the reset sequence for the source PCS is complete and is ready to accept data.

Signal Name	Location	Description
tx_sync_done	**/source/tx_sync_done**	This active high signal indicates that all the lanes are bonded by the Native PHY or Interlaken PHY IP core. This signal should be properly asserted for normal operation. A rapidly toggling signal indicates that the source FIFO is having either too much or too little data, or the core reset is having issues.
tx_cal_busy	**/source/Interlaken_phy_ip_tx/ sv_ilk_inst**	Sink transceiver calibration status. This active high signal can be used for debugging if the reconfiguration controller is actively calibrating during the initialization sequence.

Sink Core Link Debugging

Figure 5-5: Sink Core Link Debugging Flow Chart

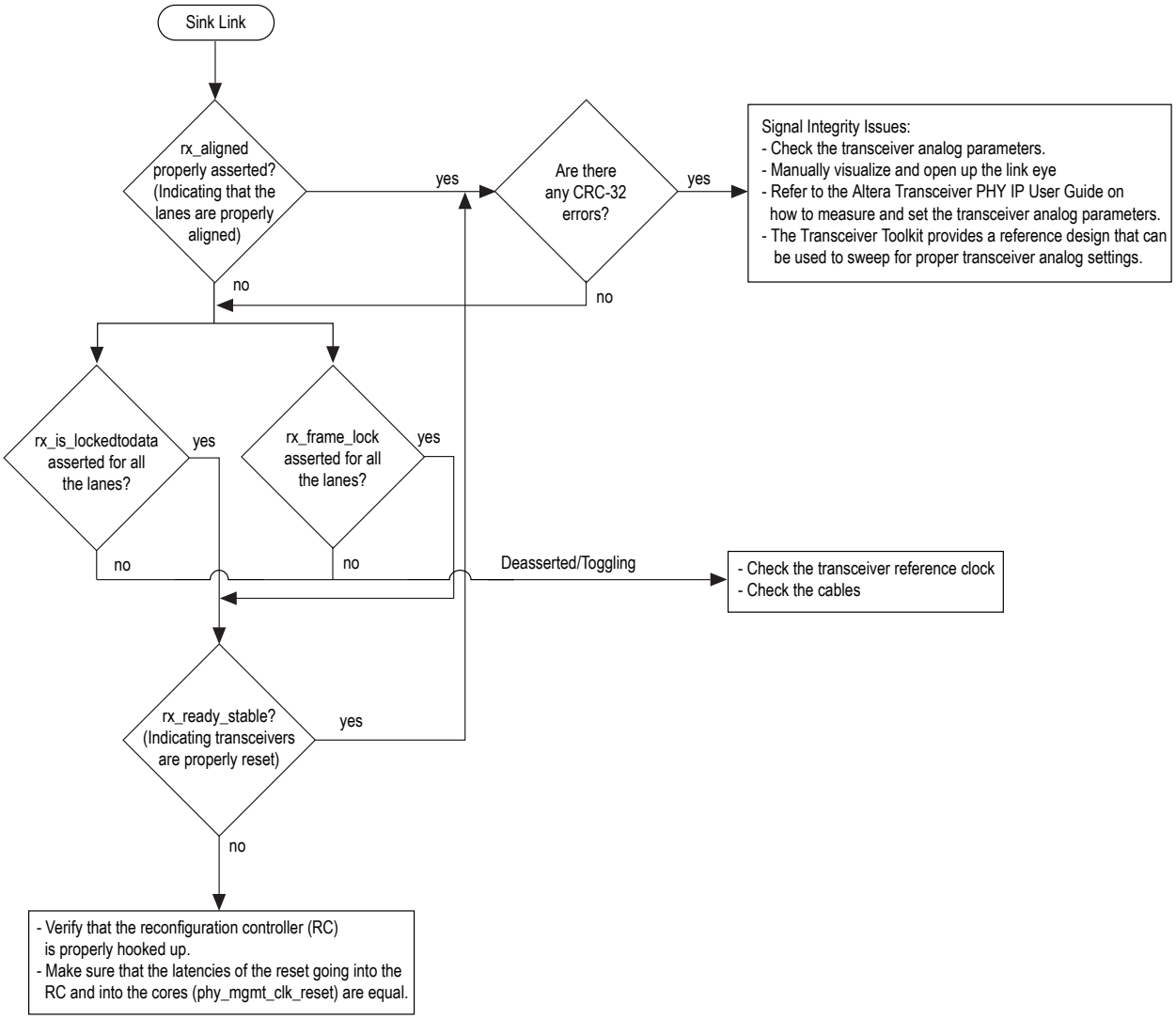


Table 5-3: Sink Link Debug Signals

Signal Name	Location	Description
rx_aligned	**/sink/rx_aligned**	This active high signal indicates that the lanes are properly aligned. This signal should remain asserted for proper operation.
rx_ready	**/sink/rx_ready**	An asserted value for this active high signal indicates that the reset sequence for the sink PCS is complete.
rx_crc32	**/sink/rx_crc32**	This active high signal indicates CRC-32 error from the CRC checker.

Signal Name	Location	Description
rx_frame_lock [lanes-1:0]	**/sink/rx_frame_lock**	This active high signal indicates that four Interlaken synchronization words are found for a given lane.
rx_is_lockedto-data [lanes-1:0]	**/sink/Interlaken_phy_ip_rx/sv_ilk_inst**	This active high signal indicates that the transceiver channel PLL has locked itself to the incoming data.
rx_cal_busy	**/sink/Interlaken_phy_ip_rx/sv_ilk_inst**	Sink transceiver calibration status. This active high signal can be used for debugging if the reconfiguration controller is actively calibrating during the initialization sequence.

Error Handling

Table 5-4: Error Conditions and Core Behavior

This table lists the error conditions that the core detect and their behavior in response to each condition.

Condition		Error Indication	Core Behavior
Source Core	Rate adaptation FIFO buffer overflow in source interface	The source core asserts the error flag for one clock cycle.	There is an overflow on the rate adaptation FIFO buffer in the source interface. The core behavior depends on the operation mode: • Continuous mode—error is flagged once an overflow is detected. • Burst mode—error is flagged only when an overflow occurs during burst data transfer across the user interface.

Condition		Error Indication	Core Behavior
Sink Core	Diagnostic code word CRC-32 error	The sink core asserts error[(lanes+3)-lane] flag for one clock cycle.	The sink interface detects a metaframe CRC-32 error on one of the lanes. These errors are reported on a per-lane basis for diagnostic purposes.
	Lane alignment failure during normal operation	The sink core asserts error[2] flag for one clock cycle.	The sink interface detects a loss of lane alignment during normal operation.
	Burst code word received during normal operation	The sink core asserts error[1] flag for one clock cycle.	The sink interface receives a burst control word after achieving normal operation. Normally, the sink interface receives only a single burst control word at the end of link initialization.
	Rate adaptation FIFO buffer underflow in sink interface	The sink core asserts error[0] flag for one clock cycle.	There is an underflow on the rate adaptation FIFO buffer in the sink interface. The core behavior depends on the operation mode: • Continuous mode—error is flagged once an underflow is detected. • Burst mode—error is flagged only when an overflow occurs during burst data transfer across the user interface.

Additional Information 6

2014.08.18

UG-01126

Additional information about the document and Altera.

Document Revision History

Date	Version	Changes
August 2014	2014.08.18	Added information about Arria 10 support.
June 2014	2014.06.30	Replaced references to MegaWizard Plug-In Manager with IP catalog or parameter editor. Minor text changes.
November 2013	2013.11.04	Added information on CRC-32 error injection. Added information on the FIFO ECC protection option.
May 2013	2013.05.13	Initial release

How to Contact Altera

Table 6-1: Altera Contact Information

Contact[3]		Contact Method	Address
Technical support		Website	www.altera.com/support
Technical training		Website	www.altera.com/training
		Email	custrain@altera.com
Product literature		Website	www.altera.com/literature
Nontechnical support	General	Email	nacomp@altera.com
	Software licensing	Email	apgcs@altera.com

[3] You can also contact your local Altera sales office or sales representative.

Related Information

- www.altera.com/support
- www.altera.com/training
- www.altera.com/literature