SDI MegaCore Function User Guide

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

Software Version: 9.0
Document Date: March 2009

UG-SDI1005-9.0

I.S. EN ISO 9001

Contents

Chapter 1. About This MegaCore Function

Release Information . . . 1–1
Device Family Support . . . 1–1
Features . . . 1–2
General Description . . . 1–2
OpenCore Plus Evaluation . . . 1–4
Resource Utilization . . . 1–5

Chapter 2. Getting Started

Design Flow . . . 2–1
SDI Walkthrough . . . 2–3
Create a New Quartus II Project . . . 2–3
Launch MegaWizard Plug-In Manager . . . 2–4
Parameterize . . . 2–5
Set Up Simulation . . . 2–5
Generate . . . 2–6
Simulate the Design . . . 2–7
Simulate with IP Functional Simulation Models . . . 2–7
Simulate with the ModelSim Simulator . . . 2–7
Simulating in Third-Party Simulation Tools Using NativeLink . . . 2–8
Compile the Design . . . 2–8
Program a Device . . . 2–9
Set Up Licensing . . . 2–9

Chapter 3. Functional Description

Block Description . . . 3–1
Transmitter . . . 3–2
HD-SDI LN Insertion . . . 3–3
HD-SDI CRC Generation and Insertion . . . 3–3
Scrambling and NRZI Coding . . . 3–4
Receiver . . . 3–4
NRZI Decoding and Descrambling . . . 3–5
Word Alignment . . . 3–5
Video Timing Flags Extraction . . . 3–6
RP168 Switching Compliance . . . 3–6
HD-SDI LN Extraction . . . 3–7
HD-SDI CRC Checking . . . 3–7
Transceiver—Soft-Logic Implementation . . . 3–7
Transmitter . . . 3–7
Transmitter Clocks . . . 3–7
Receiver . . . 3–8
Receiver Clocks . . . 3–8
Transceiver—Stratix GX Devices . . . 3–8
Transmitter Clocks . . . 3–8
Receiver Clocks . . . 3–10
Transmitter Transceiver Interface . . . 3–11
Receiver Transceiver Interface . . . 3–12
Transceiver—Arria GX, Arria II GX, Stratix II GX, and Stratix IV Devices . . . 3–13

Transmitter Clocks . . . 3–14
Receiver Clocks . . . 3–15
Transmitter Transceiver Interface . . . 3–17
Receiver Transceiver Interface . . . 3–17
Locking to the Incoming SDI Stream . . . 3–18
DPRIO for Dual Standard and Triple Standard Receivers . . . 3–19
OpenCore Plus Time-Out Behavior . . . 3–24
Signals . . . 3–24
Parameter . . . 3–33
MegaCore Verification . . . 3–34

Appendix A. Constraints
Introduction . . . A–1
Specifying TimeQuest Timing Analyzer Constraints . . . A–1
Specify Clock Characteristics . . . A–4
Set Multicycle Paths . . . A–5
Specify Clocks that are Exclusive or Asynchronous . . . A–5
Define the Setup and Hold Relationship between 135-MHz Clocks and 337.5-MHz zero-degree Clocks . . . A–6
Minimize Timing Skew . . . A–6
Constraints for the SDI Soft Transceiver . . . A–7
Non-Cyclone Devices . . . A–7
Classic Timing Analyzer . . . A–7
TimeQuest Timing Analyzer . . . A–8
Cyclone Devices Only . . . A–8
Classic Timing Analyzer . . . A–8
TimeQuest Timing Analyzer . . . A–9

Appendix B. Clocking
Transceiver Clocks . . . B–1

Additional Information
Revision History . . . Info–i
How to Contact Altera . . . Info–i
Typographic Conventions . . . Info–ii

1. About This MegaCore Function

Release Information

Table 1–1 provides information about this release of the Altera® Serial Digital Interface MegaCore® function.

Table 1–1. Release Information

Item	Description
Version	9.0
Release Date	March 2009
Ordering Code	IP-SDI
Product ID(s)	00AE
Vendor ID	6AF7

For more information about this release, refer to the *MegaCore IP Library Release Notes and Errata*.

Altera verifies that the current version of the Quartus® II software compiles the previous version of each MegaCore function. The *MegaCore IP Library Release Notes and Errata* report any exceptions to this verification. Altera does not verify compilation with MegaCore function versions older than one release.

Device Family Support

MegaCore functions provide either full or preliminary support for target Altera device families:

- *Full support* means the MegaCore function meets all functional and timing requirements for the device family and may be used in production designs.
- *Preliminary support* means the MegaCore function meets all functional requirements, but may still be undergoing timing analysis for the device family; it may be used in production designs with caution.

Table 1–2 shows the level of support offered by the SDI MegaCore function to each Altera device family.

Table 1–2. Device Family Support (Part 1 of 2)

Device Family	Support
Arria® GX	Full
Arria II GX	Preliminary
Cyclone®	Full *(1)*
Cyclone II	Full
Cyclone III	Full
Stratix®	Full

Table 1–2. Device Family Support (Part 2 of 2)

Device Family	Support
Stratix II	Full
Stratix GX	Full
Stratix II GX	Full
Stratix III	Full
Stratix IV	Preliminary
Other device families	No support

Note to Table 1–2:

(1) Cyclone support is limited to –6 speed grade devices.

Features

This section summarizes the features of the SDI MegaCore function.

- Support for multiple SDI standards and video formats (refer to Table 1–3 and Table 1–4)
- Transmitter includes:
  - Cyclical redundancy check (CRC) encoding (high definition (HD) only)
  - Line number (LN) insertion (HD only)
  - Word scrambling
- Receiver includes:
  - CRC decoding (HD only)
  - LN extraction (HD only)
  - Framing and extraction of video timing signals
  - Word alignment and descrambling
- Easy-to-use MegaWizard™ interface
- IP functional simulation models for use in Altera-supported VHDL and Verilog HDL simulators
- Support for RP168 video switch line requirement
- Support for OpenCore Plus evaluation
- Support for the Quartus II IP Advisor

General Description

The Altera SDI MegaCore function implements a receiver, transmitter, or full-duplex SDI at standard definition (SD), HD, or 3 gigabits per second (3G). The core also supports dual standard (HD-SDI and SD-SDI) and triple standard (SD-SDI, HD-SDI, and 3G-SDI). These modes provide automatic receiver rate detection.

The Society of Motion Picture and Television Engineers (SMPTE) have defined an SDI that video system designers use widely as an interconnect between equipment in video production facilities.

The SDI MegaCore function can handle the following SDI data rates:

- 270 megabits per second (Mbps) SD-SDI, as defined by *SMPTE259M-1997 10-Bit 4:2:2 Component Serial Digital Interface*
- 1.5-Gbps HD-SDI, as defined by *SMPTE292M-1998 Bit-Serial Digital Interface for High Definition Television Systems*
- 3-Gbps SDI, as defined by *SMPTE425M-AB 2006 3Gb/s Signal/Data Serial Interface – Source Image Format Mapping*
- Preliminary support for dual link SDI, as defined by *SMPTE372M-Dual Link 1.5Gb/s Digital Interface for 1920×1080 and 2048×1080 Picture Formats*
- Dual standard support for 270-Mbps and 1.5-Gbps SDI
- Triple standard support for 270-Mbps, 1.5-Gbps, and 3-Gbps SDI
- SMPTE425M Level A support (direct source image formatting)
- SMPTE425M Level B support (dual link mapping)

Table 1–3 shows the SDI standard support for various devices.

Table 1–3. SDI Standard Support *(Note 1)*

Device Family	SDI Standard					
	SD-SDI	**HD-SDI**	**3G-SDI**	**HD-SDI Dual Link**	**Dual Standard**	**Triple Standard**
Arria GX	✓	✓	✓	✓	✓	✓
Arria II GX	✓	✓	✓	✓	✓	✓
Cyclone	✓	—	—	—	—	—
Cyclone II	✓	—	—	—	—	—
Cyclone III	✓	—	—	—	—	—
Stratix	✓	—	—	—	—	—
Stratix II	✓	—	—	—	—	—
Stratix III	✓	—	—	—	—	—
Stratix IV *(2)*	✓ *(3)*	✓ *(3)*	✓	✓ *(3)*	✓ *(3)*	✓ *(3)*
Stratix II GX	✓	✓	✓	✓	✓	✓
Stratix GX	✓	✓	—	✓	✓	—

Notes to Table 1–3:

(1) All standards, other than SD-SDI, require a transceiver based or "GX" device.
(2) Only Stratix IV variants with transceivers support all SDI rates.
(3) All standards, other than 3G-SDI, support Stratix IV GT only in soft logic mode.

Table 1–4 shows the HD-SDI standard video format specification.

Table 1–4. HD-SDI Video Format Specification *(Note 1) (2)*

SMPTE292M	Video Format	Sample per Active Line	Active Line per Frame	Sample per Total Line	Total Line per Frame	Frame Rate	SDI 8.1/9.0 Support
274M	1920×1080	1920	1080	2200	1125	60	Yes
				2640	1125	50	Yes
				2200	1125	30	Yes
				2640	1125	25	Yes
				2750	1125	24	Yes
296M	1280×720	1280	720	1650	750	60	Yes
				1980	750	50	Yes
				3300	750	30	Yes
				3960	750	25	Yes
				4125	750	24	Yes
260M	1920×1035	1920	1035	2200	1125	30	Yes
295M	1920×1080	1920	1080	2376	1250	25	Yes
				2376	1250	50	Yes

Notes to Table 1–4:

(1) The video formats support 4:2:2(YC'$_B$C'$_R$)/10-bit, 4:4:4(RGB)/(YC'$_B$C'$_R$), 4:4:4:4 (RGB+A)/(YC'$_B$C'$_R$+A)/10-bit, 4:4:4(YC'$_B$C'$_R$)/12-bit, 4:4:4(RGB)/12-bit, and 4:2:2 (YC'$_B$C'$_R$)/12-bit mapping structures.

(2) 3G-SDI is similar to HD-SDI except the data bit rate is twice that of HD-SDI or approximately 3 Gbps.

OpenCore Plus Evaluation

With Altera's free OpenCore Plus evaluation feature, you can perform the following actions:

- Simulate the behavior of a megafunction (Altera MegaCore function or AMPP^{SM} megafunction) within your system
- Verify the functionality of your design, as well as evaluate its size and speed quickly and easily
- Generate time-limited device programming files for designs that include MegaCore functions
- Program a device and verify your design in hardware

You only need to obtain a license for the MegaCore function when you are completely satisfied with its functionality and performance, and want to take your design to production.

For more information about OpenCore Plus hardware evaluation using the SDI, refer to "OpenCore Plus Time-Out Behavior" on page 3–24 and *AN 320: OpenCore Plus Evaluation of Megafunctions*.

Resource Utilization

Table 1–5 shows the typical resource utilization for various parameters, using the Quartus II software, version 9.0.

Table 1–5. Resource Utilization *(Note 1)*

Device	Video Standard	LEs	Combinational ALUTs	Logic Registers
Cyclone	SD-SDI	875	—	—
Cyclone II	SD-SDI	867	—	—
Cyclone III	SD-SDI	874	—	—
Arria GX	SD-SDI	—	834	640
	HD-SDI	—	919	683
	3G HD-SDI	—	1,161	865
	Dual link HD-SDI	—	1,906	1,423
	Dual standard RX	—	1,188	831
	Dual standard TX	—	247	185
	Triple standard	—	1,794	1,215
Stratix	SD-SDI	875	—	—
Stratix II	SD-SDI	—	581	533
Stratix III	SD-SDI	—	602	565
Stratix GX	SD-SDI	1,182	—	—
	HD-SDI	1,316	—	—
	Dual link HD-SDI	2,703	—	—
	Dual standard	1,819	—	—
Stratix II GX	SD-SDI	—	834	640
	HD-SDI	—	919	683
	3G HD-SDI	—	1,161	865
	Dual link HD-SDI	—	1,906	1,423
	Dual standard RX	—	1,188	831
	Dual standard TX	—	247	185
	Triple standard	—	1,794	1,215
Stratix IV	SD-SDI	—	839	680
	HD-SDI	—	978	833
	3G HD-SDI	—	1,259	1,015
	Dual link HD-SDI	—	2,029	1,711
	Dual standard RX	—	1,257	926
	Dual standard TX	—	267	180
	Triple standard	—	1,891	1,305

Note to Table 1–5:

(1) The resource utilization of the MegaCore function is based on the bidirectional interface settings unless otherwise specified.

Design Flow

To evaluate the SDI MegaCore function using the OpenCore Plus feature, include these steps in your design flow:

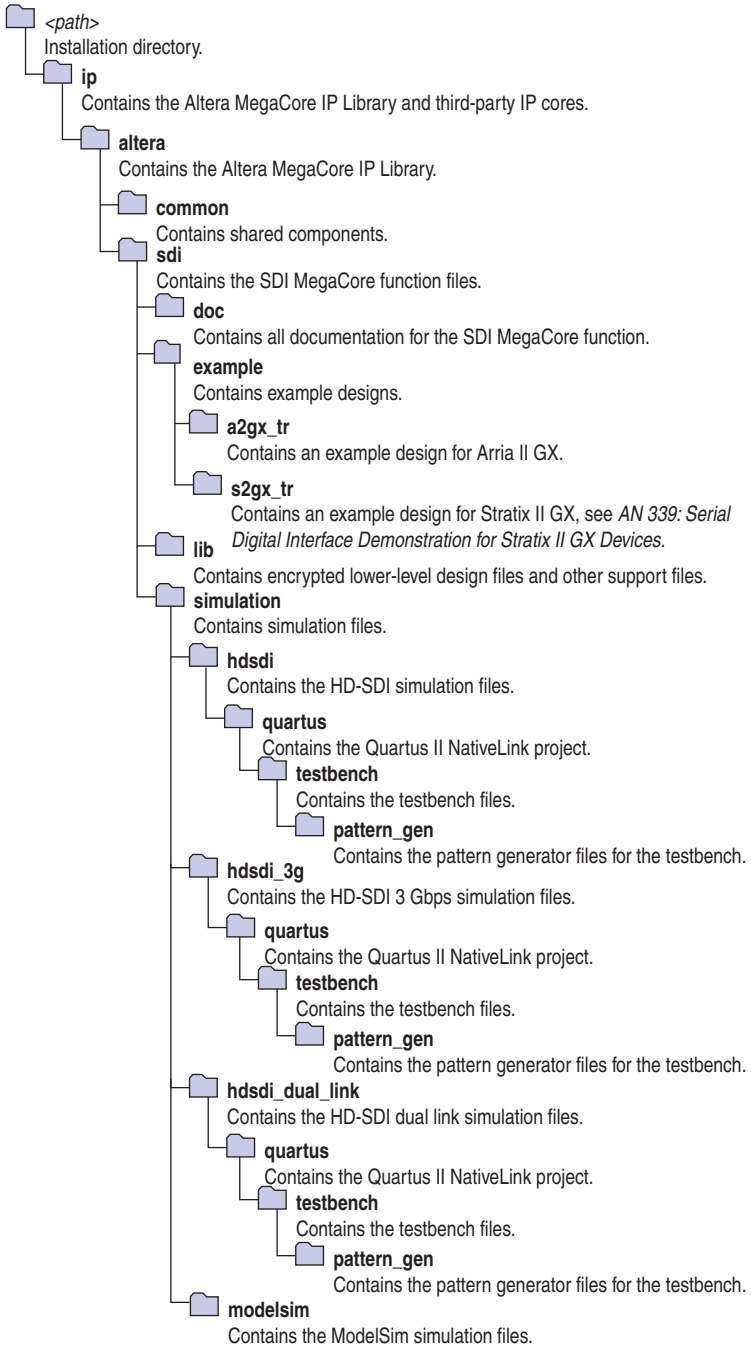
1. Obtain and install the SDI MegaCore function.

   The SDI MegaCore function is part of the MegaCore IP Library, which is distributed with the Quartus II software and downloadable from the Altera website at www.altera.com.

   For system requirements and installation instructions, refer to *Quartus II Installation & Licensing for Windows and Linux Workstations*.

   Figure 2–1 on page 2–2 shows the directory structure after you install the SDI MegaCore function, where <*path*> is the installation directory. The default installation directory on Windows is **c:\altera**\<*version*>; on Linux, it is **/opt/altera**<*version*>.

Figure 2–1. Directory Structure



2. Create a custom variation of the SDI MegaCore function.
3. Implement the rest of your design using the design entry method of your choice.
4. Use the IP functional simulation model to verify the operation of your design.

For more information on IP functional simulation models, refer to the *Simulating Altera IP in Third-Party Simulation Tools* chapter in volume 3 of the *Quartus II Handbook*.

5. Use the Quartus II software to compile your design.

You can also generate an OpenCore Plus time-limited programming file, which you can use to verify the operation of your design in hardware.

6. Purchase a license for the SDI MegaCore function.

After you have purchased a license for the SDI MegaCore function, follow these additional steps:

1. Set up licensing.
2. Generate a programming file for the Altera device or devices on your board.
3. Program the Altera device or devices with the completed design.

SDI Walkthrough

This walkthrough explains how to create an SDI using the MegaWizard Plug-In Manager and the Quartus II software. After you generate a custom variation of the SDI MegaCore function, you can incorporate it into your overall project.

You can alternatively use the IP Advisor to help start your SDI MegaCore design. On the Quartus II Tools menu, point to **Advisors**, and then click **IP Advisor**. The IP Advisor guides you through a series of recommendations for selecting, parameterizing, evaluating, and instantiating an SDI MegaCore function into your design. It then guides you through a complete Quartus II compilation of your project.

This walkthrough requires the following steps:

1. Create a New Quartus II Project
2. Launch MegaWizard Plug-In Manager
3. Parameterize
4. Set Up Simulation
5. Generate

Create a New Quartus II Project

You need to create a new Quartus II project with the **New Project Wizard**, which specifies the working directory for the project, assigns the project name, and designates the name of the top-level design entity. To create a new project, follow these steps:

1. Choose **Programs > Altera > Quartus II** *<version>* (Windows Start menu) to run the Quartus II software. Alternatively, you can use the Quartus II Web Edition software.
2. On the File menu, click **New Project Wizard**.

3. Click **Next** in the **New Project Wizard: Introduction** page (the introduction page does not display if you turned it off previously).
4. In the **New Project Wizard: Directory, Name, Top-Level Entity** page, enter the following information:
   a. Specify the working directory for your project. For example, this walkthrough uses the **c:\altera\projects\sdi_project** directory.

   ☞ The Quartus II software automatically specifies a top-level design entity that has the same name as the project. This walkthrough assumes that the names are the same.

   b. Specify the name of the project. This walkthrough uses **project** for the project name.
5. Click **Next** to close this page and display the **New Project Wizard: Add Files** page.

   ☞ When you specify a directory that does not already exist, a message prompts you to create a specified directory. Click **Yes** to create the directory.

6. If you installed the MegaCore IP Library in a different directory from where you installed the Quartus II software, you must add the user libraries:
   a. Click **User Libraries**.
   b. Type <*path*>\ip into the **Library name** field, where <*path*> is the directory in which you installed the SDI.
   c. Click **Add to** add the path to the Quartus II project.
   d. Click **OK** to save the library path in the project.
7. Click **Next** to close this page and display the **New Project Wizard: Family & Device Settings** page.
8. On the **New Project Wizard: Family & Device Settings** page, choose the target device family in the **Family** list.
9. The remaining pages in the **New Project Wizard** are optional. Click **Finish** to complete the Quartus II project.

Launch MegaWizard Plug-In Manager

To launch MegaWizard Plug-In Manager in the Quartus II software, follow these steps:

1. On the Tools menu, click **MegaWizard Plug-In Manager**.

   ☞ Refer to Quartus II Help for more information on how to use the MegaWizard Plug-In Manager.

2. Specify that you want to create a new custom megafunction variation and click **Next**.
3. Expand the **Interfaces > SDI** folder and click **SDI** <*version*>.

4. Select the output file type for your design; the wizard supports VHDL and Verilog HDL.
5. The MegaWizard Plug-In Manager shows the project path that you specified in the **New Project Wizard**. Append a variation name for the MegaCore function output files *<project path>\<variation name>*.
6. Click **Next** to display the **Parameter Settings** page for the SDI MegaCore function.

You can change the page that the MegaWizard Plug-In Manager displays by clicking **Next** or **Back** at the bottom of the dialog box. You can move directly to a named page by clicking the **Parameter Settings**, **EDA**, or **Summary** tab.
Also, you can directly display individual parameter settings by clicking on the **Protocol Options**, **Transceiver Options**, or **Receiver/Transmitter Options** tab.

Parameterize

To parameterize your MegaCore function, follow these steps:

1. Select the video standard. Some of the standards may be grayed out, because they are not supported on the currently selected device family.
2. Select **Bidirectional**, **Receiver**, or **Transmitter** interface direction.
3. Click the **Transceiver Options** tab.
4. Under **Transceiver and Protocol**, click **Generate transceiver and protocol blocks**.
5. For SD-SDI only, turn on **Use soft logic for transceiver** to implement the transceiver in logic, rather than using Stratix IV GX, Stratix II GX or Stratix GX transceivers.
6. Select the starting channel number.
7. Click the **Receiver/Transmitter Options** tab.
8. Turn on the required receiver options.
9. Turn on the required transmitter options.
10. Click **Next** (or the **EDA** tab) to display the **EDA** page.

For more information on parameters, refer to "Parameter" on page 3–33 and, for more information on the protocol options, refer to Table 3–17 on page 3–33.
For more information on the transceiver options, refer to Table 3–18 on page 3–33.
For more information on the receiver/transmitter options, refer to Table 3–19 on page 3–34.

Set Up Simulation

An IP functional simulation model is a cycle-accurate VHDL or Verilog HDL model produced by the Quartus II software. The model allows for fast functional simulation of IP using industry-standard VHDL and Verilog HDL simulators.

You may only use these models for simulation and expressly not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

To generate an IP functional simulation model for your MegaCore function, follow these steps:

1. Turn on **Generate simulation model**.
2. Some third-party synthesis tools can use a netlist that contains only the structure of the MegaCore function, but not detailed logic, to optimize performance of the design that contains the MegaCore function. If your synthesis tool supports this feature, turn on **Generate netlist**.
3. Click **Next** (or the **Summary** tab) to display the **Summary** page.

Generate

You can use the check boxes on the **Summary** page to enable or disable the generation of specified files. A gray checkmark indicates a file that is automatically generated; a red checkmark indicates an optional file.

You can click **Back** to display the previous page, or click **Parameters Settings**, **EDA**, or **Summary**, to change any of the MegaWizard options.

To generate the files, follow these steps:

1. Turn on the files you wish to generate.

   At this stage, you can still click **Back** to display any of the other pages in the MegaWizard Plug-In Manager to change any of the parameters.

2. To generate the specified files and close the MegaWizard Plug-In Manager, click **Finish**.

   The generation phase may take several minutes to complete.

   The Quartus II IP File (**.qip**) is a file generated by the MegaWizard interface, and contains information about the generated IP core. You are prompted to add this **.qip** file to the current Quartus II project at the time of file generation. In most cases, the **.qip** file contains all of the necessary assignments and information required to process the core or system in the Quartus II compiler. Generally, a single **.qip** file is generated for each MegaCore function or system in the Quartus II compiler.

3. Click **Exit** to close the **Generation** window.

Table 2–1 describes the generated files and other files that may be in your project directory. The names and types of files specified in the MegaWizard Plug-In Manager report vary based on whether you created your design with VHDL or Verilog HDL

Table 2–1. Generated Files (Part 1 of 2)

Extension	Description
<*variation name*>**.v** or **.vhd**	A MegaCore function variation file, which defines a VHDL or Verilog HDL description of the custom MegaCore function. Instantiate the entity defined by this file inside of your design. Include this file when compiling your design in the Quartus II software.
<*variation name*>**.cmp**	A VHDL component declaration file for the MegaCore function variation. Add the contents of this file to any VHDL architecture that instantiates the MegaCore function.

Table 2–1. Generated Files (Part 2 of 2)

Extension	Description
<*variation name*>**.bsf**	Quartus II symbol file for the MegaCore function variation. You can use this file in the Quartus II block diagram editor.
<*variation name*>**.html**	MegaCore function report file.
<variation name>**.ppf**	This XML file describes the MegaCore pin attributes to the Quartus II Pin Planner. MegaCore pin attributes include pin direction, location, I/O standard assignments, and drive strength. If you launch IP Toolbench outside of the Pin Planner application, you must explicitly load this file to use Pin Planner.
<*variation name*>**.vo** or **.vho**	VHDL or Verilog HDL IP functional simulation model.
<*variation name*>**_bb.v**	A Verilog HDL black-box file for the MegaCore function variation. Use this file when using a third-party EDA tool to synthesize your design.
<*variation name*>**.qip**	Contains Quartus II project information for your MegaCore function variations.

You can now integrate your custom MegaCore function variation into your design, simulate, and compile.

Simulate the Design

This section describes the following simulation techniques:

- Simulate with IP Functional Simulation Models
- Simulate with the ModelSim Simulator
- Simulating in Third-Party Simulation Tools Using NativeLink

Simulate with IP Functional Simulation Models

You can simulate your design using the MegaWizard-generated VHDL and Verilog HDL IP functional simulation models.

You can use the IP functional simulation model with any Altera-supported VHDL or Verilog HDL simulator.

To use the IP functional simulation model that you created in "Set Up Simulation" on page 2–5, create a suitable testbench.

For more information on IP functional simulation models, refer to the *Simulating Altera IP in Third-Party Simulation Tools* chapter in volume 3 of the *Quartus II Handbook*.

Simulate with the ModelSim Simulator

Altera provides two fixed testbenches as examples in the **simulation\modelsim**<*video standard*>**\modelsim** directory, where <*video standard*> is **hdsdi** or **hdsdi_dual_link**. The testbenches instantiate the design and tests the HD-SDI or dual link mode of operation. To use one of these testbenches with the ModelSim®-Altera simulator, follow these steps:

1. In a text editor, open the simulation batch file, **simulation\modelsim**<*video standard*>**\modelsim\sdi_sim.bat**. Edit it to point to your installation of the ModelSim-Altera simulator, and edit the path:

```
set PATH = %MODELSIM_DIR%\win32aloem
```

Where <*video standard*> is **hdsdi** or **hdsdi_dual_link**.

2. Start the ModelSim-Altera simulator.
3. Run **sdi_sim.bat** in the **simulation\modelsim**<*video standard*>**\modelsim** directory. This file compiles the design and starts the ModelSim-Altera simulator. A selection of signals appears on the waveform viewer. The simulation runs automatically, providing a pass/fail indication on completion.

To test the transmitter operation, the testbench generates a reference clock and parallel video data. The design encodes and serializes this parallel video data. The serial output is sampled, non-return to zero inverted (NRZI) decoded, descrambled, and then reconstructed into parallel form. The testbench detects the presence of time reference signal (TRS) tokens (end of active video (EAV) and start of active video (SAV)) in the output to check the correct operation.

To test the receiver operation, the testbench connects the serial transmitter data to the receiver input. The testbench checks that the receiver achieves word alignment and verifies that the extracted LN is correct.

Simulating in Third-Party Simulation Tools Using NativeLink

You can perform a simulation in a third-party simulation tool from within the Quartus II software, using NativeLink.

For more information on NativeLink, refer to the *Simulating Altera IP in Third-Party Simulation Tools* chapter in volume 3 of the *Quartus II Handbook*.

Altera provides the following three Quartus II projects for use with NativeLink in the **ip\altera\sdi\simulation** directory:

- HD-SDI in the **hdsdi** directory
- HD-SDI 3 Gbps in the **hdsdi_3g** directory
- HD-SDI dual link in the **hdsdi_dual_link** directory

To set up simulation in the Quartus II software using NativeLink, follow these steps:

1. On the File menu, click **Open Project**. Browse to the desired directory: **hdsdi**, **hdsdi_3g**, or **hdsdi_dual_link**.
2. Open **sdi_sim.qpf**.
3. Check that the absolute path to your third-party simulator executable is set. On the Tools menu, click **Options** and select **EDA Tools Options**.
4. On the Processing menu, point to **Start** and click **Start Analysis & Elaboration**.
5. On the Tools menu, point to **Run EDA Simulation Tool** and click **EDA RTL Simulation**.

Compile the Design

You can use the Quartus II software to compile your design. Refer to Quartus II Help for instructions on performing compilation.

You can find an example design using an SDI MegaCore in the **ip/sdi/example** directory. This design is targeted at the Stratix II GX audio video development kit.

For more information about the example design, refer to *AN 339: Serial Digital Interface Demonstration for Stratix II GX Devices*, and for information about the development kits, refer to *Audio Video Development Kit, Stratix II GX Edition*.

Program a Device

After you have compiled the example design, you can program your targeted Altera device to verify the design in hardware.

With Altera's free OpenCore Plus evaluation feature, you can evaluate the SDI MegaCore function before you obtain a license. OpenCore Plus evaluation allows you to generate an IP functional simulation model, and produce a time-limited programming file.

For more information on OpenCore Plus hardware evaluation using the SDI MegaCore function, refer to "OpenCore Plus Evaluation" on page 1–4, "OpenCore Plus Time-Out Behavior" on page 3–24, and *AN 320: OpenCore Plus Evaluation of Megafunctions*.

Set Up Licensing

You need to purchase a license for the MegaCore function only when you are completely satisfied with its functionality and performance and want to take your design to production.

After you purchase a license for SDI MegaCore function, you can request a license file from the Altera website at www.altera.com/licensing and install it on your computer. When you request a license file, Altera emails you a **license.dat** file. If you do not have Internet access, contact your local Altera representative.

3. Functional Description

The SDI MegaCore function implements a receiver, transmitter, or full-duplex interface. The SDI MegaCore function can handle SD, HD, and/or 3G SDIs.

The SDI MegaCore function consists of the following elements:

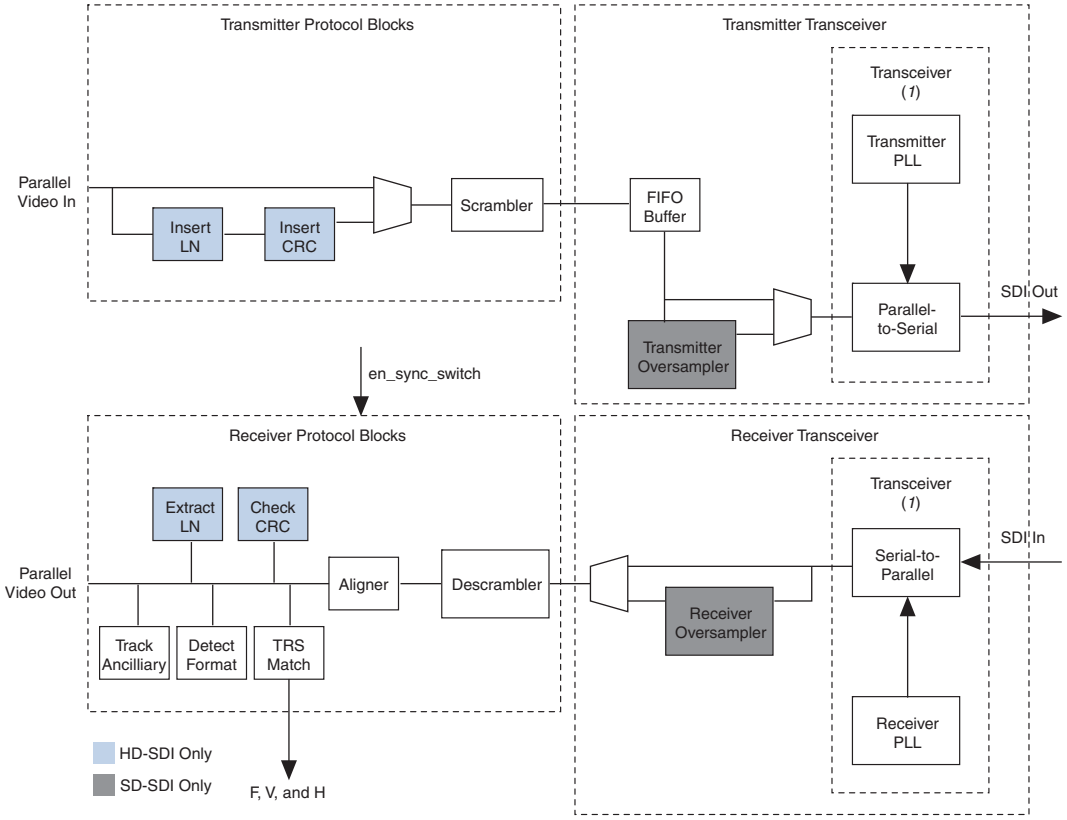
- Protocol blocks
  - SDI receiver
  - SDI transmitter
- A transceiver
- A transceiver controller

In the MegaWizard Plug-In Manager, you can specify either protocol or transceiver blocks or both for your design. For example, if you have multiple protocol blocks in a design, you can multiplex them into one transceiver. The transceiver can be either a soft-logic implementation or a GX transceiver.

Block Description

Figure 3–1 on page 3–2 shows the SDI MegaCore function block diagram.

Figure 3–1. Block Diagram



Note to Figure 3–1:

(1) For SD-SDI designs only, you can have a soft-logic implementation of the transceiver.

Transmitter

The transmitter contains the following elements:

- SD/HD-SDI transmitter scrambler
- HD-SDI transmitter data formatter, which includes a CRC and LN insertion
- Transceiver, plus control, and interface logic with multirate (dual or triple standard) SD/HD-SDI transmitter operation

The transmitter performs the following functions:

- HD-SDI LN insertion
- HD-SDI CRC generation and insertion
- Scrambling and NRZI coding

For HD-SDI, the transmitter accepts 20-bit parallel video data; for SD-SDI, 10-bit parallel data (refer to Table 3–14 on page 3–25 for `txdata` bus definition).

Table 3–1 shows the bit allocation for txdata.

Table 3–1. Bit Allocation for txdata for Supported Video Standards

txdata	SD-SDI	HD-SDI	3G-SDI Level A	3G-SDI Level B
[19:10]	Unused	Y	Y	Y, Cr, Y, Cb multiplex (channel B)
[9:0]	Y, Cr, Y, Cb multiplex	C	C	Y, Cr, Y, Cb multiplex (channel A)

For HD-SDI operation, the current video line number is inserted at the appropriate point in each line. A CRC is also calculated and inserted for the luma and chroma channels.

The parallel video data is scrambled and NRZI encoded according to the SDI specification.

The transceiver converts the encoded parallel data into the high-speed serial output (parallel-to-serial conversion).

HD-SDI LN Insertion

SMPTE292M section 5.4 defines the format of two words that are included in each HD-SDI video line to indicate the current line number. The HD-SDI LN insertion module takes the 11-bit tx_ln and formats and inserts it as two words in the output data. The HD-SDI LN insertion module accepts the current line number as an input.

If the system does not know the line number, you can implement logic to detect the output video format and then determine the current line. This function is outside the scope of this SDI MegaCore function.

The LN words (LN0 and LN1) overwrite the two words that follow the "XYZ" word of the EAV TRS sequence. The same value is included in the luma and chroma channels.

For correct LN insertion, the tx_trs signal must be asserted for the first word of both EAV and SAV TRSs (refer to Figure 3–17 and Figure 3–18 on page 3–29).

HD-SDI CRC Generation and Insertion

SMPTE292M section 5.5 defines a CRC that is included in the chroma and luma channels for each HD-SDI video line. The HD-SDI CRC module generates, formats, and inserts the required CRC in the output data.

The HD-SDI CRC module identifies the words that are to be included in the CRC calculation, and also determines where the words should be inserted in the output data. The formatted CRC data words (YCR0 and YCR1 for the luma channel, CCR0 and CCR1 for the chroma channel) overwrite the two words that follow the line number words after the EAV. A separate calculation is provided for the luma and chroma channels.

The CRC is calculated for all words in the active digital line, starting with the first active word line and finishing with the final word of the line number (LN1). The initial value of the CRC is set to zero, then the polynomial generator equation $CRC(X) = X18 + X5 + X4 + 1$ is applied.

The HD-SDI CRC module implements the CRC calculation by iteratively applying the polynomial generator equation to each bit of the output data, processing the LSB first.

For correct CRC generation and insertion, the `tx_trs` signal must be asserted for the first word of both EAV and SAV TRS (refer to Figure 3–17 and Figure 3–18 on page 3–29).

Scrambling and NRZI Coding

SMPTE292M section 5 and SMPTE292M section 7 define a common channel coding that is used for both SDI and HD-SDI. This channel coding consists of a scrambling function (G1(X) = X9 + X4 + 1) followed by NRZI encoding (G2(X) = X + 1). The scrambling module implements this channel coding. The module can be configured to process either 10-bit or 20-bit parallel data.

The scrambling module implements the channel coding by iteratively applying the scrambling and NRZI encoding algorithm to each bit of the output data, processing the LSB first. The algorithm implemented is as shown in figure C.1 of SMPTE259M.

Receiver

The receiver contains the following elements:

- Transceiver, plus control, and interface logic with multirate (dual or triple standard) SD/HD-SDI receiver operation
- SD/HD-SDI receiver descrambler and word aligner
- HD-SDI receiver CRC and LN extractor
- Receiver framing, with extraction of video timing signals
- Identification and tracking of ancillary data

The SDI receiver consists of the following functions:

- NRZI decoding and descrambling
- Word alignment
- Video timing flags extraction
- RP168 switching compliance
- HD-SDI LN extraction
- HD-SDI CRC

The received data is NRZI decoded and descrambled and then presented as a word-aligned parallel output—20 bit for HD-SDI; 10 bit for SD-SDI (refer to Table 3–14 on page 3–25 for `rxdata` bus definition).

Table 3–2 shows the bit allocation for `rxdata`.

Table 3–2. Bit Allocation for rxdata for Supported Video Standards

rxdata	SD-SDI	HD-SDI	3G-SDI Level A	3G-SDI Level B
[19:10]	Unused	Y	Y	Y, Cr, Y, Cb multiplex (channel B)
[9:0]	Y, Cr, Y, Cb multiplex	C	C	Y, Cr, Y, Cb multiplex (channel A)

The receiver interface extracts and tracks the F, V, and H timing signals in the received data. Active picture and ancillary data words are also identified for your use.

For HD-SDI, the received CRC is checked for the luma and chroma channels. The LN is also extracted and provided as an output from the design.

NRZI Decoding and Descrambling

The descrambler module provides the channel decoding function that is common to both SDI and HD-SDI. It implements the NRZI decoding followed by the required descrambling. The algorithm indicated by SMPTE259M figure C.1 is iteratively applied to the receiver data, with the LSB processed first.

Word Alignment

The aligner word aligns the descrambled receiver data such that the bit order of the output data is the same as that of the original video data.

The EAV and SAV sequences determine the correct word alignment. Table 3–3 shows the pattern for each standard.

Table 3–3. EAV and SAV Sequences

Video Standard	EAV and SAV Sequences
SDI	`3FF 000 000`
HD-SDI	`3FF 3FF 000 000 000 000`
3G-SDI Level A	`3FF 3FF 000 000 000 000`
3G-SDI Level B	`3FF 3FF 3FF 3FF 000 000 000 000 000 000 000 000`

The aligner matches the selected pattern in the descrambled receiver data. If the pattern is detected at any of the possible word alignments, then a flag is raised and the matched alignment is indicated. This process is applied continuously to the receiver data.

The second stage of the aligner determines the correct word alignment for the data. It looks for three consecutive TRSs with the same alignment, and then stores that alignment. If two consecutive TRSs are subsequently detected with a different alignment, then this new alignment is stored.

The final stage of the aligner applies a barrel shift function to the received data to generate the correctly aligned parallel word output. For this SDI MegaCore function, the barrel shifter allows the design to instantly switch from one alignment to another.

Video Timing Flags Extraction

The TRS match module extracts the F, V, and H video timing flags from the received data. You can use these flags for receiver format detection, or in the implementation of a flywheel function.

The TRS match module also identifies the line number and CRC words for HD-SDI.

RP168 Switching Compliance

To meet the RP168 requirements, the transceiver must be able to recover by the end of the switching line. Table 3–4 shows the supported video switching type.

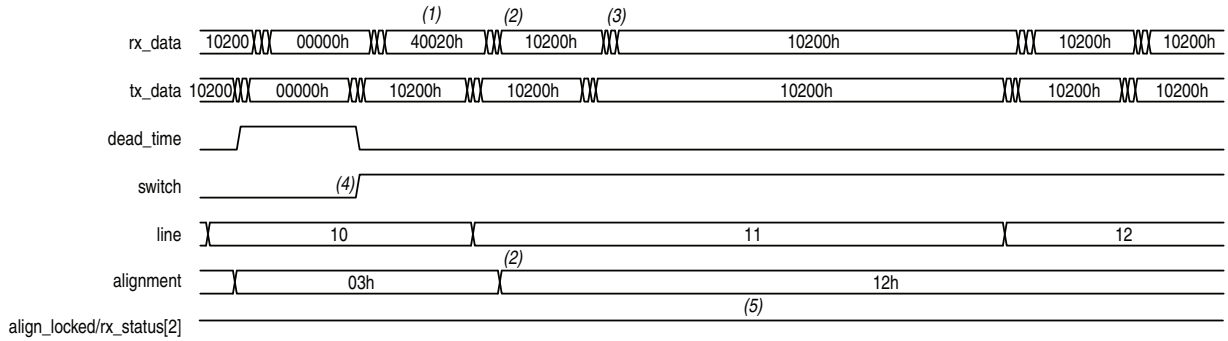
Table 3–4. Supported Video Switching Type

Standard/ Data Rate	Format	RP168 Support	Switching Source
Fixed	Switch (same format)	Yes	HD-1080i30 to HD-1080i30
Fixed	Switch	No	HD-1080 to HD-720
Switch	Fixed	No	HD-1080 to SD-525
Switch	Switch	No	HD-1080 to SD-525

Figure 3–2 and Figure 3–3 show the behaviors of the aligner and format blocks during the RP168 switching.

The aligner block immediately aligns to the next TRS timing based on the user input `en_sync_switch` signal.

Figure 3–2. Aligner Block Behavior

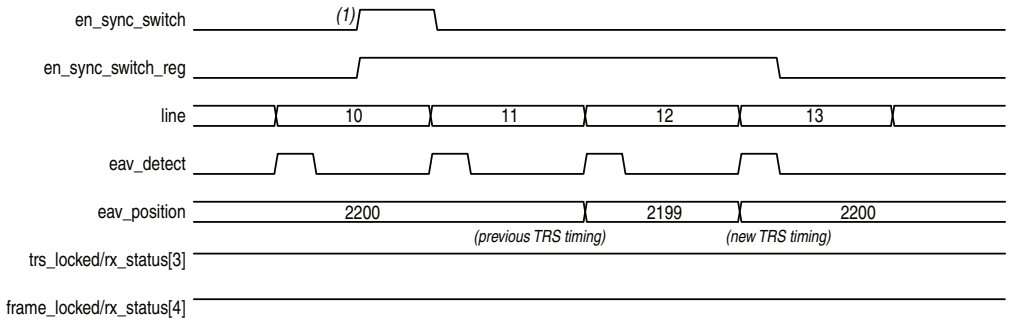


Notes to Figure 3–2:

(1) Mismatch in alignment.
(2) New alignment on the next TRS.
(3) Data aligned to new alignment.
(4) Switch from 148.5 MHz to 148.35 MHz.
(5) Zero interrupt.

The format block latches the user input `en_sync_switch` signal for three lines to realign to a new TRS alignment immediately. During switching, you should see zero interrupt at downstream. The `trs_locked` and `frame_locked` signals never get deasserted during sync switch.

Figure 3–3. Format Block Behavior



Note to Figure 3–3:

(1) Switch from 148.5 MHz to 148.35 MHz.

HD-SDI LN Extraction

The HD-SDI LN extraction module extracts and formats the LN words defined by SMPTE292M section 5.4 from the HD-SDI chroma channel. The design provides the LN as an output.

HD-SDI CRC Checking

The CRC module checks the CRC defined by SMPTE292M section 5.5 for the HD-SDI luma and chroma channels.

This module is common to the receiver and the transmitter.

The check is implemented by recalculating the CRCs for each received video line and then checking the results against the CRC data received. If the results differ, an error flag is asserted. There are separate error flags for the luma and chroma channels. The flag is held asserted until the next check is performed.

Transceiver—Soft-Logic Implementation

The soft-logic implementation differs for the transmitter and the receiver.

Transmitter

For the transmitter, in the soft-logic transceiver a 10-bit parallel word is converted into a serial data output format. A 10-bit shift register loaded at the word rate from the encoder and unloaded at the bit rate of the LVDS output buffer is implemented for that function. A phase-locked loop (PLL) that multiplies a 27-MHz reference clock by ten provides the bit-rate clock and enables jitter-controlled SDI transmit serialization.

Transmitter Clocks

The serializer requires a 270-MHz clock, which you can generate from an external source (`tx_sd_refclk_270`).

The 27-MHz parallel video clock (`tx_pclk`) samples and processes the parallel video input.

Receiver

For the receiver, in the soft-logic transceiver the serial data stream from the LVDS input buffer is sampled using four different clocks phase-shifted by 90° from each other. Two out of these four clocks are created from an on-chip PLL. The two remaining ones are created by inversion of the PLL clock outputs.

Samples are then all converted to the same clock domain and deserialized into a 10-bit parallel word. The serial clock that samples the bit stream must be 337.5 MHz, which is 5/4 of the incoming bit (270-bit rate × 5/4 × 4 sample per clock = 1,350 Mbps)

The parallel clock that extracts data from the deserializer is running at 135 MHz.

To achieve timing performance, you must correctly constrain your design, refer to "Constraints" on page A–1.

Receiver Clocks

The deserializer requires three clocks (refer to Table 3–13 on page 3–25), which you can generate from an external source.

Transceiver—Stratix GX Devices

The Stratix GX transceiver deserializes the high-speed serial input. For HD-SDI, the clock data recovery (CDR) function performs the deserialization and locks the receiver PLL to the receiver data. For SD-SDI, the transceiver provides a fixed frequency oversample of the serial data with the receiver PLL constantly locked to a reference clock, which allows the transceiver to support the 270-Mbps data rate.

The transceiver can process either SD-SDI or HD-SDI data. The data rate can be automatically detected so that the interface can handle both SD-SDI and HD-SDI without the need for device reconfiguration.

In Stratix GX devices, the transmitters in a quad share a common reference clock, which prevents them from operating independently.

Receivers in a quad share a common training clock, but have independent receiver PLLs. Because the same training clock is used for SD-SDI and HD-SDI, receivers can accommodate the different standards within a single quad.

Transmitter Clocks

The transmitter requires two clocks: a parallel video clock (`tx_pclk`) and a transmitter reference clock (`tx_serial_refclk`).

The parallel video clock samples and processes the parallel video input. For SD-SDI, it is 27 MHz; for HD-SDI, it is 74.25 or 74.175 MHz.

The transceiver uses the transmitter reference clock to generate the high-speed serial output. The transceiver is configured for 20-bit operation, so the reference clock is 1/20th of the serial data rate.

For SD-SDI, because of the oversampling implementation, the serial data rate is five times the SDI bit rate (for example, 1,350 Mbps).

For HD-SDI operation, the transmitter reference clock can be driven by `pclk`.

For SD-SDI operation, the transmitter reference clock can be derived from `pclk` by using one of the Stratix GX PLLs. The PLL can multiply the 27-MHz `pclk` signal by 5/2.

For dual standard operation, use an external multiplexer to select between the SD-SDI and HD-SDI reference clock.

The Stratix GX architecture allows each group of four transmitters (a transceiver quad) to have a separate transmitter reference clock.

Table 3–5 shows the transmitter `clock tx_serial_refclk` frequencies.

Table 3–5. Transmitter Clock Frequency—Stratix GX Devices

Video Standard *(Note 1)*	Clock Frequency (MHz)
SD-SDI	67.5
HD-SDI (including dual link)	74.175/74.25 *(2)*
HD-SDI with two times oversample	148.35/148.5 *(2)*
Dual standard	67.5/74.175/74.25 *(2)*

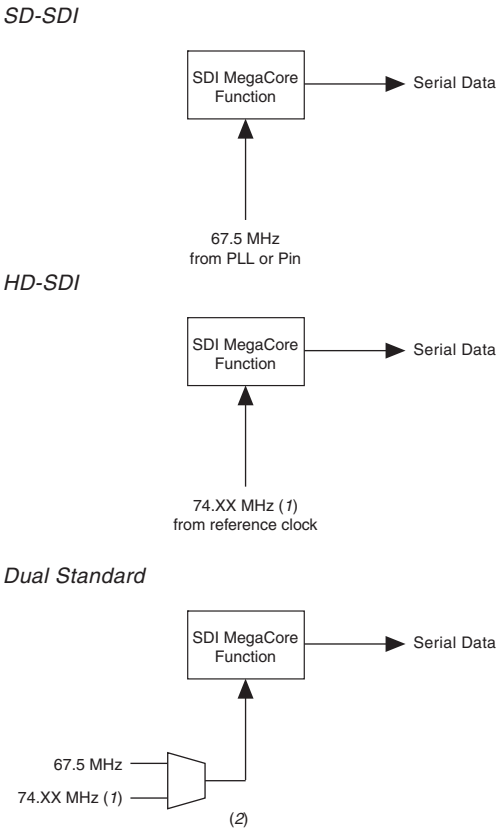
Notes to Table 3–5:

(1) Stratix GX devices do not support 3G and triple standard modes.

(2) The `tx_serial_refclk` signal must be externally multiplexed.

Figure 3–4 shows the transmitter clocks for different video standards.

Figure 3–4. Transmitter Clocks—Stratix GX Devices



Notes to Figure 3–4:

(1) This frequency can be either 74.175 or 74.25 MHz, to support 1.4835 or 1.485 Gbps HD-SDI respectively.

(2) The multiplexer must not be in the device.

Receiver Clocks

The transceiver requires a receiver reference clock, `rx_serial_refclk`. This clock trains the receiver PLL in the transceiver.

For HD-SDI operation, the clock should be nominally $1/20^{th}$ of the serial data rate. It does not need to be frequency locked to the data, because it is only used for the training of the receiver PLL.

For SD-SDI operation, the clock should be nominally $1/4^{th}$ of the serial data rate (for example, 67.5 MHz). The clock does not need to be frequency locked to the data.

For dual standard operation, the receiver reference clock should be 67.5 MHz, which allows the transceiver to sample the data for SD-SDI at the correct frequency. For HD-SDI, the receiver PLL trains with the 67.5-MHz reference and then tracks to the actual incoming data rate.

All receiver interfaces can share a common receiver reference clock.

Table 3–6 shows the receiver clock `rx_serial_refclk` frequencies.

Table 3–6. Receiver Clock Frequency

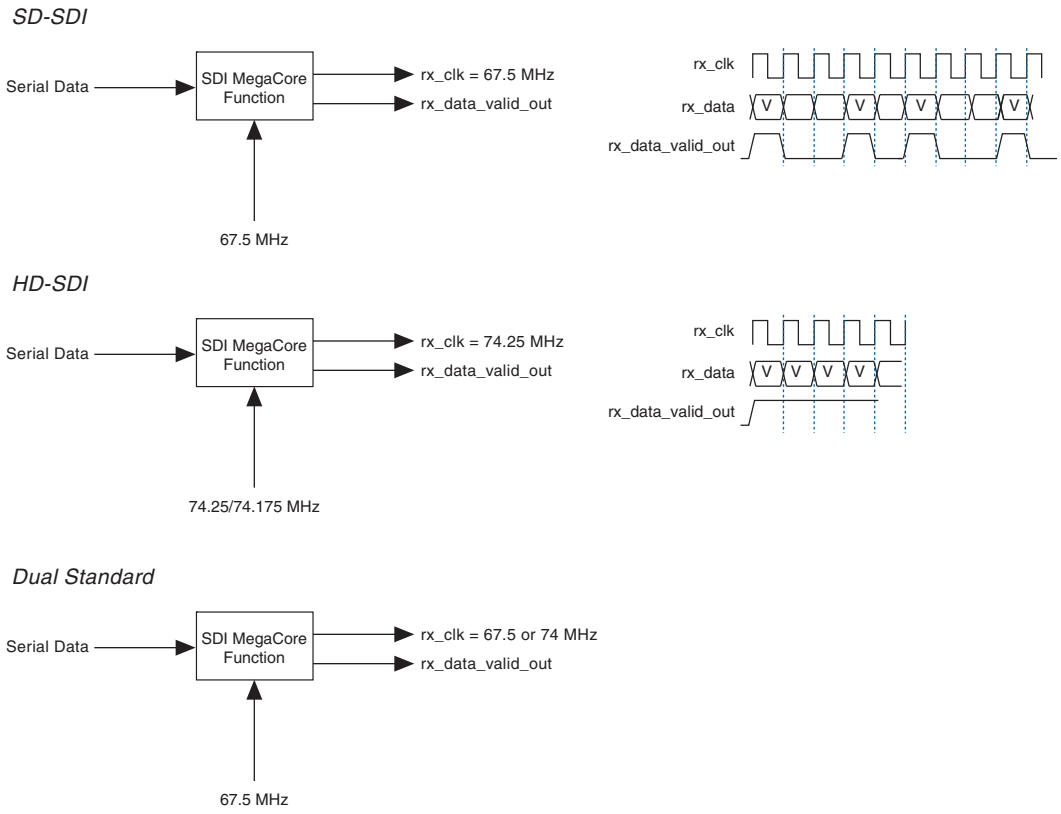
Video Standard *(Note 1)*	Clock Frequency (MHz)
SD-SDI	67.5
HD-SDI (including dual link)	74.175/74.25 *(2)*
Dual standard	67.5

Notes to Table 3–6:

(1) Stratix GX devices do not support 3G-SDI and triple standard modes.
(2) The `rx_serial_refclk` signal must be externally multiplexed.

Figure 3–5 shows the receiver clocks for different video standards.

Figure 3–5. Receiver Clocks



Transmitter Transceiver Interface

Altera provides a transceiver interface, which interfaces the transceiver to the SDI function. The transceiver interface implements the following functions:

- Transmitter Retiming
- HD-SDI Two-Times Oversampling
- SD-SDI Transmitter Oversampling

When using the two-times oversampling transmitters in Stratix GX devices, you cannot have HD-SDI receivers in the same quad. The quad requires the same frequency reference clocks for both the receivers and transmitters within a quad. HD-SDI receivers and two-times oversampling transmitters have different frequency reference clocks (refer to Table 3–5 and Table 3–6 on page 3–11).

Transmitter Retiming

The `txdata` parallel data input to the transceiver must be synchronous and phase aligned to the `tx_coreclk` transceiver clock input. SD-SDI (and optionally HD-SDI) requires a retiming function, because of the oversampling logic. The transmitter uses a small 4 × 20 FIFO buffer for the retiming.

For HD-SDI, the FIFO buffer realigns the parallel video input to the transceiver `tx_coreclk` clock. It is written on every `tx_pclk` clock, and read on every `tx_coreclk`.

For SD-SDI, the FIFO buffer also provides the rate conversion required by the transmitter oversampling logic. It is written on every other `tx_pclk`, using the SD-SDI data width conversion logic. It is read on every fifth `tx_coreclk`. This operation ensures that the transmitter oversampling logic is provided with a word of parallel video data on every fifth clock.

HD-SDI Two-Times Oversampling

This mode performs two-times oversampling and runs the transceiver at double rate, which gives better output jitter performance. This mode requires a higher rate reference clock, refer to Table 3–5 on page 3–9.

SD-SDI Transmitter Oversampling

SD-SDI requires a 270-Mbps serial data rate, which is achieved by transmitting a 1,350 Mbps signal with each bit repeated five times. This process ensures that the transceiver runs at a supported frequency.

Receiver Transceiver Interface

Altera provides a transceiver interface, which interfaces the transceiver to the SDI function. The transceiver interface implements the following functions:

- SD-SDI Receiver Oversampling
- Transceiver Controller

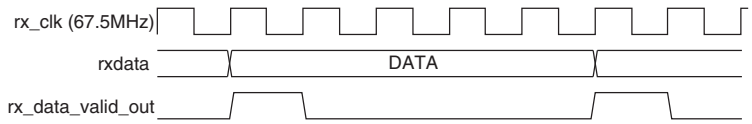
When using the two-times oversampling transmitters in Stratix GX devices, you cannot have HD-SDI receivers in the same quad. The quad requires the same frequency reference clocks for both the receivers and transmitters within a quad. HD-SDI receivers and two-times oversampling transmitters have different frequency reference clocks (refer to Table 3–5 and Table 3–6 on page 3–11).

SD-SDI Receiver Oversampling

The Stratix GX transceiver does not support CDR for data rates less than 500 Mbps. The receiver uses fixed frequency oversampling for the reception of 270-Mbps SD-SDI. The serial data is sampled by the transceiver at 1,350 Mbps and the original 270-Mbps data is extracted by the SD-SDI receiver oversampling logic.

Figure 3–6 shows an example of the receiver data timing.

Figure 3–6. Receiver Data Timing



Transceiver Controller

To achieve the desired receiver functionality for the SDI, the transceiver controller controls the transceiver.

When the interface receives SD-SDI, the transceiver receiver PLL locks to the receiver reference clock.

When the interface receives HD-SDI, the transceiver receiver PLL is first trained by locking to the receiver reference clock. When the PLL is locked, it can then track the actual receiver data rate. If a period of time passes without a valid SDI signal, the PLL is retrained with the reference clock and the process is repeated.

The transceiver controller allows the transceiver to support the reception of both SD-SDI and HD-SDI data by using an algorithm that alternately searches for one rate then the other. First it looks for an HD-SDI signal, training the PLL then letting it track the serial data rate. If a valid HD-SDI signal is not seen within 0.1 s, the receiver path is reset and the PLL is trained for SD-SDI. Conversely, if a valid SD-SDI signal is not seen within 0.1 s, the receiver path is reset and the process repeated. The transceiver controller also resets and starts searching again if the SDI receiver indicates that the signal is no longer valid.

For HD-SDI operation, if 100 consecutive bits with the same value are seen, the receiver is reset and the PLL is retrained. The maximum legal run length for HD-SDI is 59 bits.

For more information on the Stratix GX transceiver, refer to the *Stratix GX Device Handbook*.

Transceiver—Arria GX, Arria II GX, Stratix II GX, and Stratix IV Devices

The Arria GX, Arria II GX, Stratix II GX, or Stratix IV transceiver deserializes the high-speed serial input. For HD-SDI, the CDR function performs the deserialization and locks the receiver PLL to the receiver data. For SD-SDI, the transceiver provides a fixed frequency oversample of the serial data with the receiver PLL constantly locked to a reference clock, which allows the transceiver to support the 270-Mbps data rate.

The transceiver can process either SD-SDI or HD-SDI data. The data rate can be automatically detected so that the interface can handle both SD-SDI and HD-SDI without the need for device reconfiguration.

Arria GX, Arria II GX, Stratix II GX, and Stratix IV devices have two transmitter PLLs per quad, which allows two independent transmitter rates.

Receivers in a quad share a common training clock, but have independent receiver PLLs. Because the same training clock is used for SD-SDI and HD-SDI, receivers can accommodate the different standards within a single quad.

Transmitter Clocks

The transmitter requires two clocks: a parallel video clock (`tx_pclk`) and a transmitter reference clock (`tx_serial_refclk`).

The parallel video clock samples and processes the parallel video input. For SD-SDI, it is 27 MHz; for HD-SDI, it is 74.25 or 74.175 MHz; for 3G-SDI, it is 148.5 or 148.35 MHz.

The transceiver uses the transmitter reference clock to generate the high-speed serial output. The transceiver is configured for 20-bit operation, so the reference clock is 1/20th of the serial data rate.

For SD-SDI, because of the oversampling implementation, the serial data rate is five times the SDI bit rate (for example, 1,350 Mbps); for the triple-standard SDI, the oversampling rate is 11.

For SD-SDI operation, the transmitter reference clock can be derived from `pclk` by using one of the transceiver PLLs. The PLL can multiply the 27-MHz `pclk` signal by 5/2.

For all other standards, use an external multiplexer to select between the alternative reference clocks.

Table 3–7 shows the transmitter clock `tx_serial_refclk` frequencies.

Table 3–7. Transmitter Clock Frequency—Arria GX, Arria II GX, Stratix II GX, and Stratix IV Devices

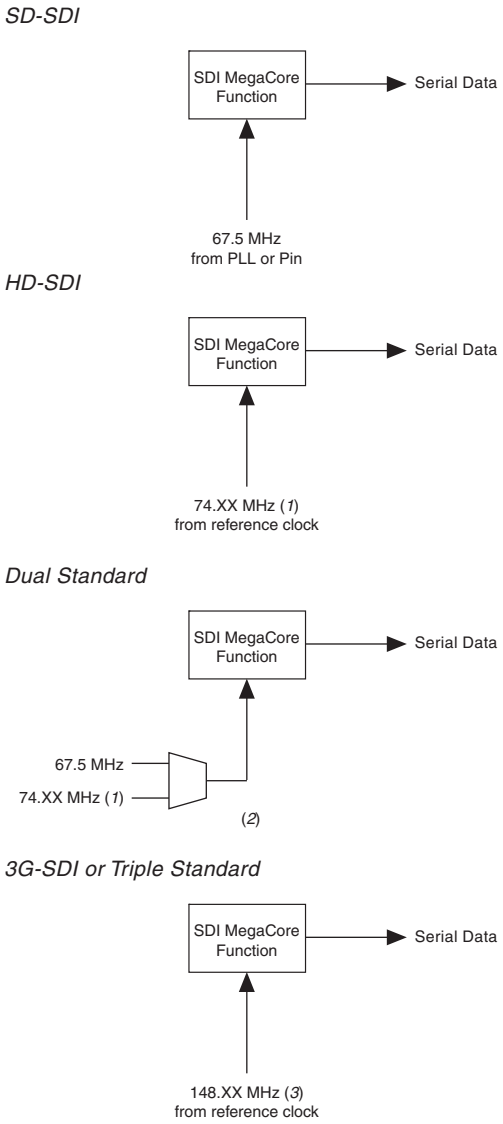
Video Standard	Clock Frequency (MHz)
SD-SDI	67.5
HD-SDI (including dual link)	74.175/74.25 *(1)*
HD-SDI with two times oversample	148.35/148.5 *(2)*
Dual standard	67.5/74.175/74.25 *(2)*
Triple standard	148.35/148.5 *(2)*
3G-SDI	148.35/148.5 *(2)*

Note to Table 3–7:

(1) The `tx_serial_refclk` signal must be externally multiplexed.

Figure 3–7 shows the transmitter clocks for different video standards.

Figure 3–7. Transmitter Clocks—Arria GX, Arria II GX, Stratix II GX, and Stratix IV Devices



Notes to Figure 3–7:

(1) This frequency can be either 74.175 or 74.25 MHz, to support 1.4835 or 1.485 Gbps HD-SDI respectively.
(2) The multiplexer must not be in the device.
(3) This frequency can be either 148.35 or 148.5 MHz, to support 2.967 or 2.970 Gbps HD-SDI respectively.

Receiver Clocks

The transceiver requires a receiver reference clock, `rx_serial_refclk`. This clock trains the receiver PLL in the transceiver.

For HD-SDI operation, the clock should be nominally 1/20th of the serial data rate. It does not need to be frequency locked to the data, because it is only used for the training of the receiver PLL.

For SD-SDI operation, the clock should be nominally 1/4th of the serial data rate (for example, 67.5 MHz). The clock does not need to be frequency locked to the data.

For dual or triple standard operation, the receiver reference clock should be 148.5 MHz. In this mode, the transceiver oversamples the SD-SDI signals by a factor of 11.

All receiver interfaces can share a common receiver reference clock.

Table 3–8 shows the receiver clock `rx_serial_refclk` frequencies.

Table 3–8. Receiver Clock Frequency—Arria GX, Arria II GX, Stratix II GX, and Stratix IV Devices

Video Standard	Clock Frequency (MHz)
SD-SDI	67.5
HD-SDI (including dual link)	74.175/74.25 *(1)*
Dual or triple standard	148.35/148.5 *(1) (2)*
3G-SDI	148.35/148.5 *(1)*

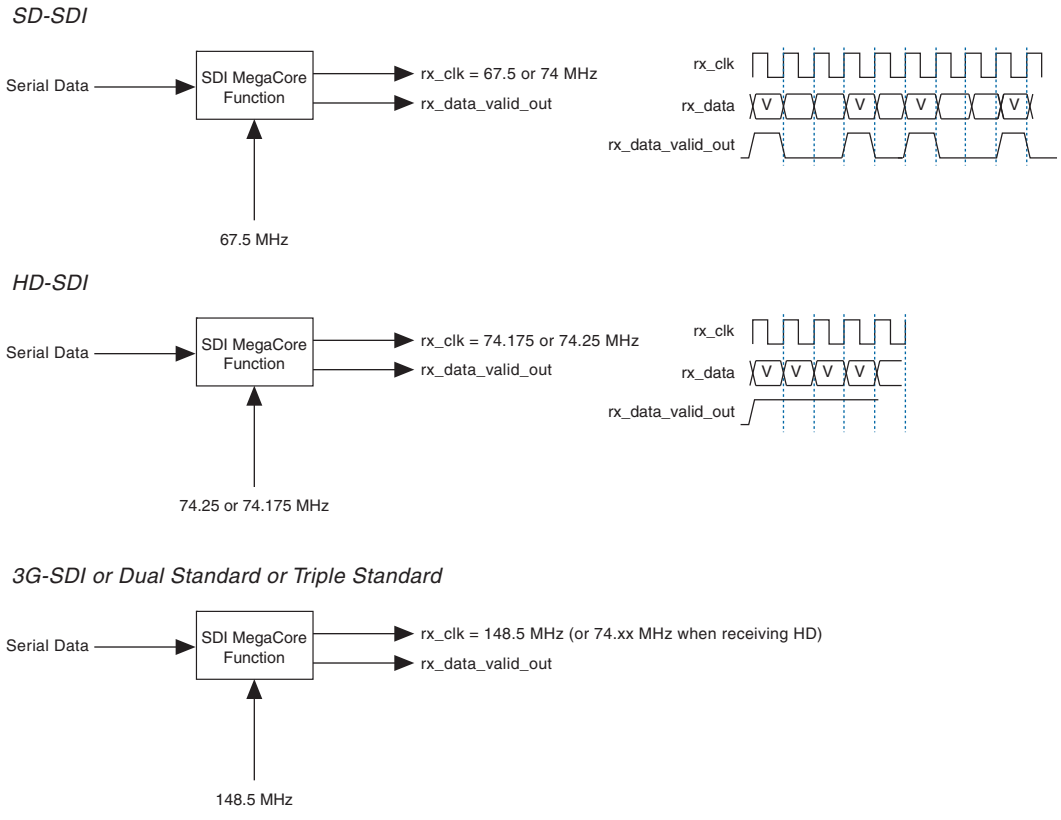
Notes to Table 3–8:

(1) You can use either reference clock for training.

(2) Must be 148.5 MHz for correct SD-SDI operation.

Figure 3–8 shows the receiver clocks for different video standards.

Figure 3–8. Receiver Clocks



Transmitter Transceiver Interface

Altera provides a transceiver interface, which interfaces the transceiver to the SDI function. The transceiver interface implements the following functions:

- Retiming from the parallel video clock domain to the transceiver transmitter clock domain
- Optional two-times oversampling for HD
- Transmitter oversampling for SD

Transmitter Retiming

The `txdata` parallel data input to the transceiver must be synchronous and phase aligned to the `tx_coreclk` transceiver clock input. SD-SDI (and optionally HD-SDI) requires a retiming function, because of the oversampling logic. The transmitter uses a small 4 × 20 FIFO buffer for the retiming.

For HD-SDI, the FIFO buffer realigns the parallel video input to the transceiver `tx_coreclk` clock. It is written on every `tx_pclk` clock, and read on every `tx_coreclk`.

For SD-SDI, the FIFO buffer also provides the rate conversion required by the transmitter oversampling logic. It is written on every other `tx_pclk`, using the SD-SDI data width conversion logic. It is read on every fifth or eleventh `tx_coreclk`. This operation ensures that the transmitter oversampling logic is provided with a word of parallel video data on every fifth or eleventh clock.

HD-SDI Two-Times Oversampling

This mode performs two-times oversampling and runs the transceiver at double rate, which gives better output jitter performance. This mode requires a higher rate reference clock, refer to Table 3–5 on page 3–9.

SD-SDI Transmitter Oversampling

SD-SDI requires a 270-Mbps serial data rate, which is achieved by transmitting a 1,350 Mbps signal with each bit repeated five times. This process ensures that the transceiver runs at a supported frequency. In triple standard mode, bit are transmitted at 2,970 Mbps with each bit repeated 11 times.

Receiver Transceiver Interface

Altera provides a transceiver interface, which interfaces the transceiver to the SDI function. The transceiver interface implements the following functions:

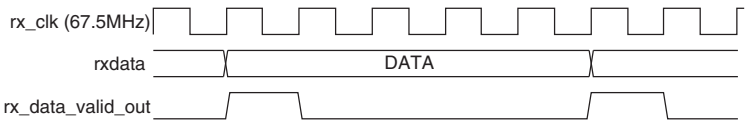
- SD-SDI Receiver Oversampling
- Transceiver Controller

SD-SDI Receiver Oversampling

Arria GX and Stratix II GX transceivers do not support CDR for data rates less than 622 Mbps. Arria II GX and Stratix IV transceivers do not support CDR for data rates less then 600 Mbps. The receiver uses fixed frequency oversampling for the reception of 270-Mbps SD-SDI. The serial data is sampled by the transceiver at 1,350 or 2,970 Mbps and the original 270-Mbps data is extracted by the SD-SDI receiver oversampling logic.

Figure 3–9 shows an example of the receiver data timing.

Figure 3–9. Receiver Data Timing



Transceiver Controller

To achieve the desired receiver functionality for the SDI, the transceiver controller controls the transceiver.

When the interface receives SD-SDI, the transceiver receiver PLL locks to the receiver reference clock.

When the interface receives HD-SDI, the transceiver receiver PLL is first trained by locking to the receiver reference clock. When the PLL is locked, it can then track the actual receiver data rate. If a period of time passes without a valid SDI signal, the PLL is retrained with the reference clock and the process is repeated.

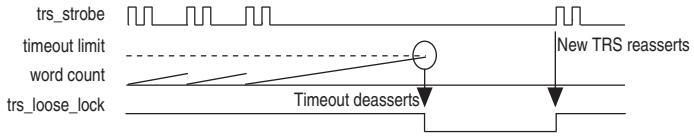
Firstly, the transceiver controller makes a coarse rate detection of the incoming data stream. Then the transceiver is reprogrammed using DPRIO (refer to "DPRIO for Dual Standard and Triple Standard Receivers" on page 3–19) to the correct rate for the standard that has been detected. Then the transceiver attempts to lock to the incoming stream. If no valid data is seen in 0.1 s, the receiver path is reset and the rate detection is performed again.

At the start of the rate detection process, the level of the three `enable_xx` signals is sampled. The level of these signals and the knowledge of the currently programmed state of the transceiver is used to judge if the transceiver requires programming. This process ensures that the transceiver is reprogrammed only when necessary.

Locking to the Incoming SDI Stream

The transceiver control state machine uses the presence (or absence) of TRSs on the stream to determine if SDI is being correctly received. A single, valid TRS indicates to the control state machine that the receiver is acquiring some valid SDI samples. This flag is only deasserted when no TRS sequences are detected within a certain number of incoming data words (16,384 clocks). At this point, the controller state machine is reset and it performs the relock algorithm, refer to Figure 3–10.

Figure 3–10. Locking Algorithm



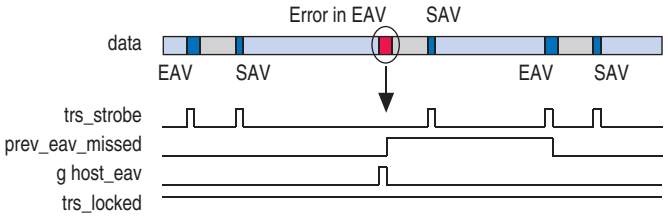
Since the aligner realigns to a new alignment if two consecutive TRSs with the same alignment are detected, this scheme allows for an SDI source switch and an alignment change without the transceiver reset state machine being affected (provided that the incoming SDI has a TRS within the time-out limit).

The SDI MegaCore also monitors the incoming EAV and SAV signals to ensure their spacing is consistent over a number of lines. This is done by incrementing a counter on each incoming SDI word and storing the count values at which an EAV or SAV is detected. If the EAV and SAV spacing is consistent over 6 video lines, the MegaCore indicates `trs_locked` on the `rx_status[3]` output.

An enhancement in the current SDI MegaCore function allows a single missing EAV or SAV to be tolerated without the `trs_locked` signal being deasserted. In version 8.1 and previous, a single missing or misplaced TRS causes the `trs_locked` to be deasserted. The operation of this missing or misplaced TRS tolerance is shown in Figure 3–11 and Figure 3–12.

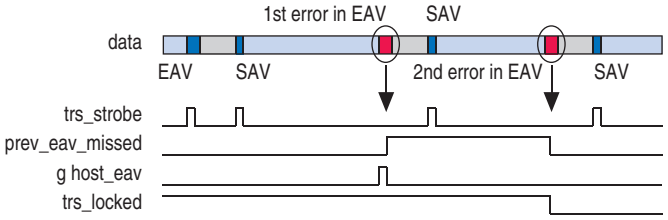
A single missing EAV sets a "missed" flag but does not cause `trs_locked` to deassert. A good EAV in the correct position resets the "missed" flag.

Figure 3–11. Single Missing EAV Signal



Two consecutive missing EAVs cause the `trs_locked` signal to be deasserted.

Figure 3–12. Two Consecutive Missing EAV Signal



DPRIO for Dual Standard and Triple Standard Receivers

Dual standard and triple standard SDI receivers (or receivers of duplex SDIs) require the dynamic partial reconfigurable I/O (DPRIO) feature of Arria GX, Arria II GX, Stratix II GX, and Stratix IV devices, to perform autodetection and locking to different SDI rates.

For more information on DPRIO, refer to the *Arria GX Device Handbook*, *Arria II GX Device Handbook*, *Stratix II GX Device Handbook*, and *Stratix IV Device Handbook*.

Table 3–9 shows the DPRIO requirements.

Table 3–9. DPRIO Requirements

SDI Standard	Receiver	Transmitter *(1)*	Duplex
SD-SDI	No	No	No
HD-SDI	No	No	No
Dual standard	Yes	No	Yes
Triple standard	Yes	No	Yes

Note to Table 3–9:

(1) SDI transmitters do not require the use of DPRIO but in order to enable the cores to merge into a transceiver quad that has DPRIO enabled, these ports must be correctly connected. For further information, refer to Table 3–16 on page 3–32.

DPRIO allows the settings of the device transceivers (ALT2GXB or ALT4GXB) to be changed at any time. DPRIO allows the transceivers to be reprogrammed to support the three SDI rates.

The triple standard SDI uses 11 times oversampling for receiving SD-SDI. Hence, only two Arria GX or Stratix II GX transceiver configurations are required as the rates for 3G-SDI and SD-SDI 11 times are the same.

Table 3–10 shows the rates for the different SDI standards.

Table 3–10. SDI Standard Rates

SDI Standard	Data Rate	Oversampling	Transceiver Rate (MHz)	Transceiver Reference Clock (MHz)	rx_clk Rate
SD-SDI	270 Mbps	11 times	2,970	148.5	148.5
HD-SDI	1.485 Gbps	None	1,485	148.5 *(1)*	74.25 *(1)*
3G-SDI	2.970 Gbps	None	2,970	148.5 *(1)*	148.5 *(1)*

Note to Table 3–10:

(1) Also supports the 1/1.001 rates.

The reprogramming of the transceivers requires the ALT2GXB_RECONFIG megafunction. However, the reprogramming of Arria II GX or Stratix IV device family requires a slightly different configuration of the ALT2GXB_RECONFIG megafunction. This parameterization can be found in the **example\source\sdi_dprio_siv** directory in the example design.

For more information on the ALT2GXB_RECONFIG megafunction, refer to the *Stratix II GX ALT2GXB_RECONFIG Megafunction User Guide*.

Transceiver Operation

The alternative setups for the transceiver settings are stored in ROMs within the device. These are 28 words by 16 bits for Arria GX and Stratix II GX devices, and 41 words by 16 bits for Arria II GX and Stratix IV devices. ALT2GXB megafunction has a serial reprogramming interface, so the ALT2GXB_RECONFIG block must serialize this parallel data before loading.

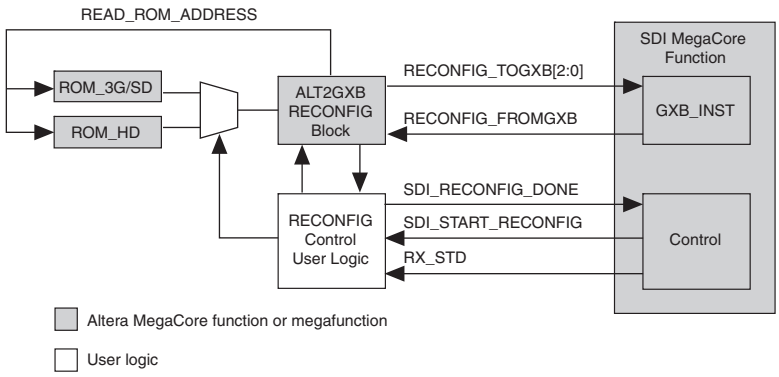
The following sequence of events occur during an SDI receiver rate change:

- SDI MegaCore function detects the incoming video rate and requests reprogramming.
- The ALT2GXB_RECONFIG block reads the appropriate ROM and serializes the data.
- The ALT2GXB_RECONFIG block applies the serial data to the correct transceiver instance.
- When this process has finished, the ALT2GXB_RECONFIG block indicates to the SDI that reprogramming is complete.
- The SDI starts the process of locking to the incoming data.

☞ Some user logic is required to handle the handshaking between the SDI MegaCore function and the ALT2GXB_RECONFIG megafunction. For examples, refer to the example designs in the **example\source\sdi_dprio** and **example\source\sdi_dprio_siv** directories.

Figure 3–13 shows a block diagram of the design elements.

Figure 3–13. DPRIO Block Diagram



The ROM_HD and ROM_3G/SD ROMs hold the transceiver setting information for each of the video standards. The setup for SD-SDI and 3G-SDI is the same, so only two ROMs are required: one for SD-SDI and 3G-SDI and one for HD-SDI.

The ALT2GXB_RECONFIG block handles the programming of the ROM contents into the transceiver megafunction. It performs data serialization and also handles protection of certain data bits in the serial stream. Only this block can be connected to the reprogramming ports of the transceiver instance.

The reconfiguration control user logic selects the correct ROM and also provides the handshaking between the SDI MegaCore function and the ALT2GXB_RECONFIG block.

The transceiver megafunction is embedded inside the SDI MegaCore function. The reprogramming ports for the transceiver megafunction are brought to the top-level interfaces of the MegaCore function. This interface can only be connected to the ALT2GXB_RECONFIG block.

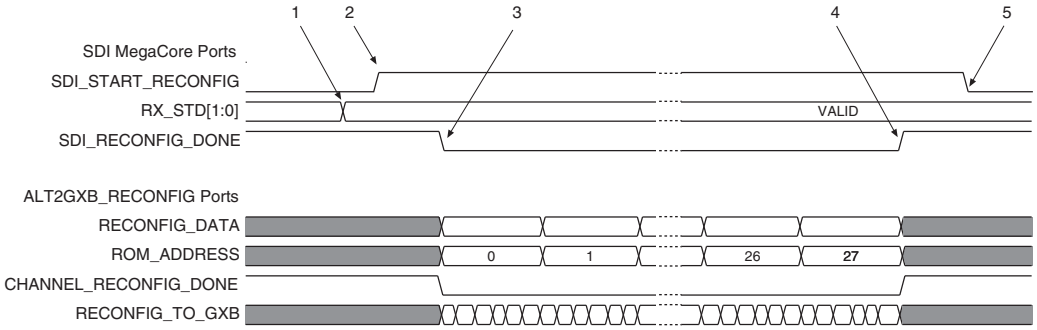
ALT2GXB_RECONFIG Connections

The SDI_START_RECONFIG and SDI_RECONFIG_DONE signals handle the handshaking between the SDI MegaCore function and the user logic. The RX_STD signal is also required to select the correct ROM instance.

Table 3–16 on page 3–32 shows the five signals that handle the DPRIO operation.

Figure 3–14 shows the handshaking between the SDI MegaCore function and the user logic, and the expected output of some of the ALT2GXB_RECONFIG signals.

Figure 3–14. Handshaking



The following sequence of events occur for handshaking to the reconfiguration logic:

1. The SDI MegaCore function sets rx_std[1:0] to the desired video standard. This action is performed as part of the video standards detection algorithm.
2. The SDI MegaCore function asserts SDI_START_RECONFIG, to make a reconfiguration request.
3. The user logic sets SDI_RECONFIG_DONE to 0, which indicates to the MegaCore function that reconfiguration is in progress.
4. When the reconfiguration has been performed, the user logic should set SDI_RECONFIG_DONE to logic 1, which indicates to the SDI MegaCore function that it can start to lock to the incoming data.
5. The SDI MegaCore function sets the SDI_START_RECONFIG line to 0 to indicate that the request has been completed and acknowledged.

Generation of ROM Contents

For Arria GX and Stratix II GX devices, the contents of the ROM are set by Memory Initialization Files (**.mif**). The Quartus II software outputs the **.mif** file for the configuration settings of the ALT2GXB instance that is set by the design.

This file generation is not performed by default. You must adjust the fitter settings. On the Assignments menu, click **Settings**. In the **Settings** dialog box, click **Fitter Settings**, and then click **More Settings**. In the **Name** list, select **Generate GXB Reconfig MIF** and in the **Setting** list, select **On**.

For the SDI MegaCore function, the Quatus II-generated **.mif** file is for 3G-SDI setup.

These **.mif** files relate to a specific ALT2GXB instance in the device. Therefore, you cannot use the same **.mif** files or ROMs for multiple ALT2GXBs in the same device.

For the SDI MegaCore function, the differences between the ALT2GXB setups are very small. Only three bits of the ROMs change between the HD-SDI and SD-SDI, or 3G-SDI setups. The three bits are in word 23 and can be seen in the following examples of the **.mif** files (Example 3–1 and Example 3–2).

Example 3–1. 3G-SDI ROM Content Example

```
....
 22 : 1010100000011111;
 23 : 0111110000010100;
 24 : 0001000101101000;
 ....
```

Example 3–2. HD-SDI ROM Content Generated from 3G-SDI Version

```
 ...
22 : 1010100000011111;
 23 : 0111110000001101;
 24 : 0001000101101000;
....
```

This particular word is static over all SDI ALT2GXB instances in the device. The **.mif** file for the HD-SDI ROM can be generated from the **.mif** file that the Quartus II software generates by modifying this memory word within the **.mif** file.

A further simplification of this scheme is included in the SDI MegaCore function example design, **example\source\sdi_dprio\sdi_mif_intercept.v**. This design uses a single ROM and **.mif** file and has logic that modifies the ROM read data when word 23 is read.

For Arria II GX and Stratix IV devices, the **.mif** file generation is not currently supported. Instead, you are advised to use the **.mif** file that is included in the **example\source\sdi_dprio_siv** directory for your design.

Starting Channel Number

To correctly address each transceiver by the ALT2GXB_RECONFIG block, you must specify a starting channel number for each transceiver instance in the MegaWizard interface. This starting channel number must meet certain criteria for the DPRIO.

For more information on the criteria, refer to the *Arria GX Device Handbook*, *Arria II GX Device Handbook*, *Stratix II GX Device Handbook*, and *Stratix IV Device Handbook*.

Quartus II Design Flow

For Arria GX and Stratix II GX devices, a two-pass compilation is required for SDI MegaCore function designs using DPRIO. The first compilation writes out the ALT2GXB setup as a **.mif** file. During this compilation, the **.mif** ROMs in the design should be set to have a dummy **.mif** file for their initialization.

Before the second compilation, the ROMs should have their initialization **.mif** files set to be those generated in the first compilation. This second compilation therefore sets up the ROMs to have the correct settings for the ALT2GXB megafunction. This process has the following steps:

1. Set the reconfiguration ROMs in the designs with a dummy **.mif** file.
2. Run the Quartus II compilation and ensure the **.mif** files are written out.
3. Copy and modify the **.mif** files for the HD-SDI ROMs and edit word 23.
4. Set the appropriate ROMs to use the **.mif** files generated in steps 2 and 3.
5. Run the Quartus II compilation.

Since the **.mif** file generation is not supported for the Arria II GX and Stratix IV devices, the design flow for Arria II GX and Stratix IV is simplified. You must set the ROMs to use the fixed **.mif** file in the **example\source\sdi_dprio_siv** directory and compile once. Ensure that you use the supporting reconfiguration code in the same directory for your design.

OpenCore Plus Time-Out Behavior

OpenCore Plus hardware evaluation can support the following two modes of operation:

- *Untethered*—the design runs for a limited time
- *Tethered*—requires a connection between your board and the host computer. If tethered mode is supported by all megafunctions in a design, the device can operate for a longer time or indefinitely

All megafunctions in a device time out simultaneously when the most restrictive evaluation time is reached. If there is more than one megafunction in a design, a specific megafunction's time-out behavior may be masked by the time-out behavior of the other megafunctions.

For MegaCore functions, the untethered time-out is 1 hour; the tethered time-out value is indefinite.

Your design stops working after the hardware evaluation time expires and the `rst` signal goes high.

For more information on OpenCore Plus hardware evaluation, refer to "OpenCore Plus Evaluation" on page 1–4 and *AN320: OpenCore Plus Evaluation of Megafunctions*.

Signals

Table 3–11 shows the receiver clock signals.

Table 3–11. Receiver Clock Signals (Part 1 of 2)

Signal	Direction	Description
`gxb2_cal_clk`	Input	Calibration clock for Arria GX and Stratix II GX transceivers only.
`gxb4_cal_clk`	Input	Calibration clock for Arria II GX and Stratix IV transceivers only.

Table 3–11. Receiver Clock Signals (Part 2 of 2)

Signal	Direction	Description
rx_27_refclk	Input	27-MHz clock reference, which is an input to the PLL, if you include PLLs.
rx_sd_oversample_clk_in	Input	67.5-MHz oversample clock input. SD-SDI only.
rx_serial_refclk	Input	Tranceiver training clock for HD-SDI, dual standard and triple standard.
gxb_tx_clkout	Output	Transmitter clock out of Stratix II GX transceiver. This clock is the output of the VCO and is the clock that you should use to clock parallel logic in the design. It connects internally to the tx_clkout signal of the ALT2GXB megafunction. For more information on tx_clkout, refer to the *Stratix II GX ALT2GXB Megafunction User Guide*.
rx_clk	Output	Transceiver CDR clock.
rx_sd_oversample_clk_out	Output	67.5-MHz oversample clock output for cascading MegaCore functions. SD-SDI only.

Table 3–12 shows the transmitter clock signals.

Table 3–12. Transmitter Clock Signals

Signal	Direction	Description
tx_27_refclk	Input	27-MHz clock reference, which is an input to the PLL if you include PLLs.
tx_pclk	Input	Transmitter parallel clock input. For SD-SDI = 27 MHz; for HD-SDI = 74 MHz and for 3G-SDI = 148 MHz.
tx_serial_refclk	Input	Transceiver reference clock input. Low jitter. Refer to Table 3–5.
tx_pclk_out	Output	Transmitter parallel clock output (if you include PLLs).
tx_sd_oversample_clk_out	Output	67.5-MHz oversample clock output for cascading MegaCore functions (if you include PLLs). SD-SDI only.

Table 3–13 shows the soft transceiver clock signals.

Table 3–13. Soft Transceiver Clock Signals

Signal	Direction	Description
rx_sd_refclk_337	Input	Soft transceiver 337.5-MHz sampling clock.
rx_sd_refclk_337_90deg	Input	Soft transceiver 337.5-MHz sampling clock with 90° phase shift.
rx_sd_refclk_135	Input	Soft transceiver 135-MHz parallel clock for receiver.
tx_sd_refclk_270	Input	Soft transceiver 270-MHz parallel clock for transmitter.

Table 3–14 shows the interface signals.

Table 3–14. Interface Signals (Part 1 of 4)

Signal	Width	Direction	Description
enable_crc	[(*N* – 1):0]	Input	Enables CRC insertion for HD-SDI and 3G-SDI.
enable_hd_search	1	Input	Enables search for HD-SDI signal in dual or triple standard mode.

Table 3–14. Interface Signals (Part 2 of 4)

Signal	Width	Direction	Description
enable_sd_search	1	Input	Enables search for SD-SDI signal in dual or triple standard mode.
enable_3g_search	1	Input	Enables search for 3G-SDI signal in triple standard mode.
enable_ln	[(*N* – 1):0]	Input	Enables line number insertion for HD-SDI and 3G-SDI modes.
en_sync_switch	1	Input	Enables aligner and format blocks to realign immediately so that the downstream is completely non-disruptive.
rst	1	Input	Reset signal, which holds the receiver and transmitter in reset. It must be synchronous to rx_serial_refclk clock domain for the receiver. The reset synchronization for the transmitter is handled within the SDI MegaCore function. The video mode (tx_std_select_hd_sdn/tx_std) and clocks must be set up and stable prior to device bring-up or core reset. Issues a reset to the SDI MegaCore after power-up to ensure reliable operation. Refer to Figure 3–15 and Figure 3–16.
rx_protocol_clk	[(*N* – 1):0]	Input	External clock for protocol data.
rx_protocol_hd_sdn	[(*N* – 1):0]	Input	Selection of HD-SDI or SD-SDI processing for dual or triple standard protocol block. This signal only appears on dual or triple standard protocol blocks and indicates 3G-SDI(1), HD-SDI(1) or SD-SDI(0) data on the rx_protocol_in signal. This signal should be connected to the rx_std_flag_hd_sdn output of the transceiver block in a split protocol/transceiver design.
rx_protocol_in	[(20*N* – 1):0]	Input	External data in for protocol only mode.
rx_protocol_locked	[(*N* – 1):0]	Input	Input to transceiver control logic. When active, this signal indicates to the transceiver control logic that the protocol blocks are locked, to stop the transceiver search algorithm at the current rate.
rx_protocol_rst	[(*N* – 1):0]	Input	Reset for the protocol block. This signal resets the protocol blocks. It can be connected to the rx_status[1] pin (sdi_reset) in a split transceiver/protocol design.
rx_protocol_valid	[(*N* – 1):0]	Input	External data valid in for protocol only mode.
rx_xcvr_trs_lock	[(*N* – 1):0]	Input	Input to transceiver control logic. It must be connected to the rx_status[3] pin (trs_locked) of the protocol only receiver block.
sdi_rx	[(*N* – 1):0]	Input	Serial input.
txdata	[(20*N* – 1):0]	Input	User-supplied transmitter parallel data. SD-SDI uses 9:0; HD-SDI uses 20*N* – 1:0. SD: bits 19:10 unused; bits 9:0 Y, Cr, Y, Cb multiplex HD: bits 19:10 Y; bits 9:0 C 3G-SDI Level A: bits 19:10 Y; bits 9:0 C 3G-SDI Level B: bits 19:10 Y, Cr, Y, Cb multiplex (channel B); bits 9:0 Y, Cr, Y, Cb multiplex (channel A)

Table 3–14. Interface Signals (Part 3 of 4)

Signal	Width	Direction	Description
tx_data_type_a_bn	1	Input	Identifies the transmitter data word on txdata as either SPMTE 425M-A or 425M-B. This signal is used on triple standard or 3G-SDI transmitters only. For SPMTE 425M-B, this should be set to logic level 0. For all other data formats, this should be set to logic level 1 (including HD-SDI).
tx_ln	[21:0]	Input	Transmitter line number. For use in HD-SDI and 3G-SDI line number insertion. HD-SDI: bits 21:11 11'd0; bits 10:0 LN Dual link: bits 21:11 LN link B; bits 10:0 LN link A 3G-SDI Level A: bits 21:11 11'd0; bits 10:0 LN 3G-SDI Level B: bits 21:11 LN link B; bits 10:0 LN link A
tx_std_select_hd_sdn	1	Input	Selects HD-SDI or SD-SDI for dual standard. 1 = HD-SDI; 0 = SD-SDI. This signal must be set up and stable prior to device bring-up or core reset.
tx_trs	[(*N* – 1):0]	Input	Transmitter TRS input. For use in HD-SDI LN and CRC insertion. Assert on first word of both EAV and SAV TRSs. Refer to Figure 3–17 and Figure 3–18.
tx_std	[1:0]	Input	Transmitter standard. 00 for SD-SDI; 01 for HD-SDI; 11 for 3G-SDI. This signal must be set up and stable prior to device bring-up or core reset.
trs_loose_lock	[(*N* – 1):0]	Output	TRS locking signal for protocol only receiver mode. It can be connected to the rx_protocol_locked pin of the transceiver only receiver block.
crc_error_y	[(*N* – 1):0]	Output	CRC error on luma channel.
crc_error_c	[(*N* – 1):0]	Output	CRC error on chroma channel.
rx_AP	[1:0]	Output	Receiver active picture synchronization output. HD-SDI/SD-SDI: bit 1 unused; bit 0 rx_ap Dual link: bit 1 link B rx_ap; bit 0 link A rx_ap 3G-SDI Level A: bit 1 unused; bit 0 rx_ap 3G-SDI Level B: bit 1 link B rx_ap; bit 0 link A rx_ap
rxdata	[(20*N* – 1):0]	Output	Receiver parallel data. SD-SDI uses 9:0; HD-SDI uses 20*N* – 1:0. SD-SDI bits 19:10 unused; bits 9:0 Y, Cr, Y, Cb multiplex HD-SDIbits 19:10 Y; bits 9:0 C 3G-SDI Level A: bits 19:10 Y; bits 9:0 C 3G-SDI Level B: bits 19:10 Y, Cr, Y, Cb multiplex (channel B); bits 9:0 Y, Cr, Y, Cb multiplex (channel A)

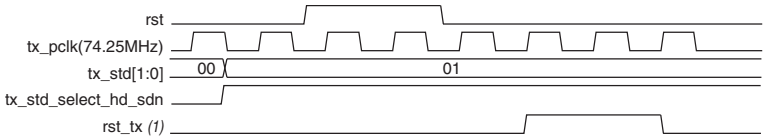
Table 3–14. Interface Signals (Part 4 of 4)

Signal	Width	Direction	Description
rx_data_valid_out	[1:0]	Output	Data valid from the oversampling logic. Asserted to indicate current data on rxdata is valid. Bit 0 of this bus indicates valid data on rxdata. When receiving SMPTE 425M-B signals In 3G-SDI or triple standard, bit 1 indicates that data on rxdata is from virtual channel B; bit 0 indicates the data is from virtual channel A. Refer to Figure 3–19 and Figure 3–20, and *SMPTE425M-B 2006 3Gb/s Signal/Data Serial Interface – Source Image Format Mapping*.
rx_F	[1:0]	Output	Receiver frame synchronization output. HD-SDI/SD: bit 1 unused; bit 0 rx_f Dual link: bit 1 link B rx_f; bit 0 link A rx_f 3G-SDI Level A: bit 1 unused; bit 0 rx_f 3G-SDI Level B: bit 1 link B rx_f; bit 0 link A rx_f
rx_H	[1:0]	Output	Receiver horizontal synchronization output. HD-SDI/SD-SDI: bit 1 unused; bit 0 rx_h Dual link: bit 1 link B rx_h; bit 0 link A rx_h 3G-SDI Level A: bit 1 unused; bit 0 rx_h 3G-SDI Level B: bit 1 link B rx_h; bit 0 link A rx_h
rx_ln	[21:0]	Output	Receiver line number output. HD-SDI: bits 21:11 unused; bits 10:0 LN Dual link: bits 21:11 LN link B; bits 10:0 LN link A 3G-SDI Level A: bits 21:11 unused; bits 10:0 LN 3G-SDI Level B: bits 21:11 LN link B; bits 10:0 LN link A
rx_std_flag_hd_sdn	1	Output	Indicates received standard for dual or triple standard only. HD-SDI = 1; SD-SDI = 0.
rx_V	[1:0]	Output	Receiver vertical synchronization output. HD-SDI/SD-SDI: bit 1 unused; bit 0 rx_v Dual link: bit 1 link B rx_v; bit 0 link A rx_v 3G-SDI Level A: bit 1 unused; bit 0 rx_v 3G-SDI Level B: bit 1 link B rx_v; bit 0 link A rx_v
rx_xyz	[(*N* – 1):0]	Output	Receiver output that indicates current word is XYZ word.
xyz_valid	[(*N* – 1):0]	Output	Receiver output that indicates current TRS format is legal (XYZ word is correct).
rx_eav	[(*N* – 1):0]	Output	Receiver output that indicates current TRS is EAV.
rx_trs	[(*N* – 1):0]	Output	Receiver output that indicates current word is TRS. This signal is asserted at the first word of 3FF 000 000 TRS.
sdi_tx	[(*N* – 1):0]	Output	Serial output.
tx_protocol_out	[(20*N* – 1):0]	Output	Data out (protocol only mode).

Figure 3–15 through Figure 3–20 show the behavior of some signals in Table 3–14.

Figure 3–15. Power-up Reset for the Receiver

Figure 3–16. Power-up Reset for the Transmitter



Note to Figure 3–16:

(1) Internally synchronized reset for the transmit circuits.

Figure 3–17. Behavior of tx_data_type_a_bn—425MA

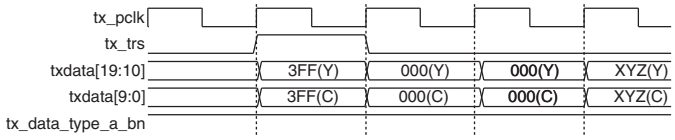


Figure 3–18. Behavior of tx_data_type_a_bn—425MB

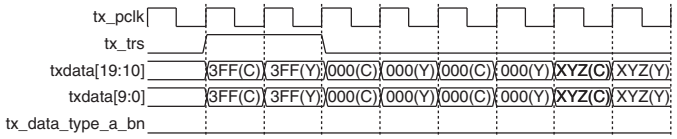


Figure 3–19. Behavior of rx_data_valid—425MA

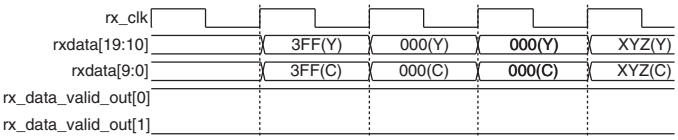


Figure 3–20. Behavior of rx_data_valid—425MB

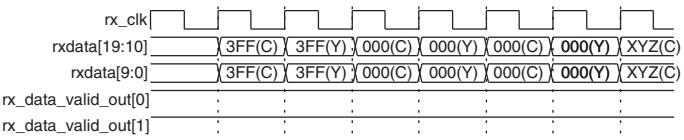


Table 3–15 shows the status signals.

Table 3–15. Status Signals (Part 1 of 2)

Signal	Width	Direction	Description
rx_anc_data	[(20*N* – 1):0]	Output	Received ancillary data. For SD-SDI, rx_anc_data [9:0] is the data, ([19:10] is unused); bit 9 is the MSB. For HD-SDI, rx_anc_data [9:0] is the data for chroma, bit 9 is the MSB; rx_anc_data [19:10] is the data for luma, bit 19 is the MSB.
rx_anc_error	[(2*N* – 1):0]	Output	Ancillary data or checksum error. For SD-SDI, rx_anc_error [0] is the error flag, (1 is unused). For HD-SDI, rx_anc_error [0] is the error flag for chroma; rx_anc_error [1] is the error flag for luma.
rx_anc_valid	[(2*N* – 1):0]	Output	Ancillary data valid. Asserted to accompany data ID (DID), secondary data ID/data block number (SDID/DBN), data count (DC), and user data words (UDW) on rx_anc_data. For SD-SDI, rx_anc_valid [0] is the valid flag, (1 is unused). For HD-SDI, rx_anc_valid [0] is the valid flag for chroma; rx_anc_valid [1] is the valid flag for luma.

Table 3–15. Status Signals (Part 2 of 2)

Signal	Width	Direction	Description
rx_status	[10:0]	Output	Receiver status: ■ rx_status[10] dual link ports aligned ■ rx_status[9] link 2 frame locked ■ rx_status[8] link 2 TRS locked (six consecutive TRSs with same timing) ■ rx_status[7] link 2 alignment locked (a TRS has been spotted and word alignment performed) ■ rx_status[6] link 2 receiver in reset ■ rx_status[5] link 2 transceiver PLL locked ■ rx_status[4] link 1 Frame locked ■ rx_status[3] link 1 TRS locked (six consecutive TRSs with same timing) ■ rx_status[2] link 1 alignment locked (a TRS has been spotted and word alignment performed) ■ rx_status[1] link 1 receiver in reset ■ rx_status[0] link 1 transceiver PLL locked For non HD-SDI dual link versions, only bits 4:0 are active. In Stratix GX designs, rx_status[0] and rx_status[5] connect to the pll_locked signal of the ALTGX megafunction; in Stratix II GX designs, they are the inverse of the rx_pll_locked signal on the ALT2GXB. This active low signal indicates lock of the PLL when the ALT2GXB is training from a refclk source. This signal may oscillate when the ALT2GXB is correctly locked to incoming data in HD-SDI or 3G-SDI modes. In SD-SDI modes, this signal should remain at logic 0 at all times. For rx_status[3] and rx_status[8], the TRS spacing is not required to meet a particular SMPTE standard, but it must be consistent over time for this signal to remain active.
tx_status	[(*N* – 1):0]	Output	Transmitter status, which indicates the transmitter PLL has locked to the tx_serial_refclk signal. This signal is active high for Stratix GX devices and active low for Stratix II GX devices.

Table 3–16 shows the signals that handle the DPRIO operation.

Table 3–16. DPRIO Signals

Signal	Direction	Description
SDI_RECONFIG_DONE	Input	Indication back to MegaCore function that reconfiguration has finished
SDI_RECONFIG_TOGXB[3:0] *(1) (3) (4) (5)*	Input	Data input for the embedded transceiver instance. *(2)*
SDI_RECONFIG_CLK *(3)*	Input	Clock input for the embedded transceiver instance. *(2)*
SDI_GXB_POWERDOWN	Input	Powers down and resets all circuits in the transceiver instance. *(6)*
SDI_START_RECONFIG	Output	Request from MegaCore function to start reconfiguration.
SDI_RECONFIG_FROMGXB[16:0] *(1) (3) (4) (5)*	Output	Data output from embedded transceiver instance. *(2)*
RX_STD[1:0]	Output	Receive video standard. 00 = SD-SDI, 01 = HD-SDI, 10 = 3G-SDI. The SDI MegaCore function can recover both *SMPTE 425M-A* and *425M-B* formatted streams. The receiver indicates which format it detects by setting the level of the rx_std bus: ■ rx_std[1:0] = 2'b11 = *425M-A* ■ rx_std[1:0] = 2'b10 = *425M-B*

Notes to Table 3–16:

(1) These signals must be connected directly to a reconfiguration megafunction.

(2) The transceivers are available for Arria GX, Arria II GX, Stratix II GX, and Stratix IV only.

(3) SDI transmitters do not require the use of DPRIO but in order to enable the cores to merge into a transceiver quad that has DPRIO enabled, you must connect these ports correctly.

(4) In the Quartus II software version 8.1 and later, the Stratix IV transceivers need RX buffer calibration through an ALTGX_RECONFIG (DPRIO) controller. The additional RECONFIG port bits are used for RX buffer calibration. You must connect these ports to the ALTGX_RECONFIG controller externally. For further information on the RX buffer calibration, refer to the *Stratix IV Dynamic Reconfiguration* chapter in volume 2 of the *Stratix IV Device Handbook*. If you are using the Quartus II software version 9.0, make sure to upgrade the SDI MegaCore function to version 9.0 as well.

(5) Bits SDI_RECONFIG_TOGXB [3] and SDI_RECONFIG_FROMGXB [16:1] are negligible for the Arria GX and Stratix II GX DPRIO operations.

(6) The SDI_GXB_POWERDOWN signal of all the instances that are to be combined in a single transceiver block must be connected to a single point (same input or same logic). Any difference in the driving logic prevents the instances from being combined in a single transceiver block.

Parameter

The parameters can only be set in the MegaWizard Plug-In Manager (refer to "Parameterize" on page 2–5).

Table 3–17 shows the protocol options.

Table 3–17. Protocol Options

Parameter	Value	Description
Video standard	SD-SDI, HD-SDI, 3G-SDI, HD-SDI dual link, dual or triple standard SDI	Selection of HD-SDI or SD-SDI. Selecting HD-SDI switches in LN insertion and extraction and CRC generation and extraction blocks; selecting SD-SDI switches out LN insertion and extraction and CRC generation and extraction blocks. Selecting SD-SDI also includes oversampling logic. Selecting dual or triple standard SDI includes the processing blocks for both SD-SDI and HD-SDI standards. In addition, logic for bypass paths and logic to automatically switch between the input standards is included.
Interface direction	Bidirectional, receiver, transmitter	Selects the ports to be receiver, transmitter, or bidirectional. It switches in or out the receiver and transmitter supporting logic appropriately. The same setting is applied to all channels in the variation. If you want some to be transmitter and some to be duplex, simply create two different MegaCore variations.

Table 3–18 shows the transceiver options.

Table 3–18. Transceiver Options

Parameter	Value	Description
Transceiver and Protocol	Generate transceiver and protocol blocks, generate transceiver block only, or generate protocol block only	Selects transceiver or protocol blocks or both. When non-GX device is chosen, only SD-SDI protocol block is permitted. If you want to generate HD-SDI or 3G-SDI protocol block, you must select a GX device.
Use soft logic for transceiver	On or off	Uses soft logic to implement the transceiver logic, rather than using Stratix II GX, Stratix IV or Stratix GX transceivers. SD-SDI only. For example, if you run out of hard transceivers in your device, you can implement the function in soft logic. If you have spare transceivers in a device, you may wish to use them.
Starting channel number	0, 4, 8, ..., 156	Dual or triple standard only. Each dual or triple standard SDI must have a unique starting channel number.

Table 3–19 shows the receiver/transmitter options.

Table 3–19. Receiver/Transmitter Options

Parameter	Value	Description
CRC error output	On or off	Turns on or off CRC monitoring (HD-SDI and 3G-SDI only).
SDI synchronization output	On or off	Provides synchronization outputs.
Two times oversample mode	On or off	HD-SDI transmitter only. When turned on, runs the transceiver at twice the rate and has improved jitter performance. Requires 148.5-MHz `tx_serial_refclk` reference clock.

MegaCore Verification

The MegaCore verification involves testing to the following standards:

- For the SD-SDI to *SMPTE259M-1997 10-Bit 4:2:2 Component Serial Digital Interface*
- For the HD-SDI to *SMPTE292M-1998 Bit-Serial Digital Interface for High Definition Television Systems*

A. Constraints

Introduction

For the SDI MegaCore function to work reliably, you must implement the following Quartus II constraints:

- Specify clock characteristics
- Set timing exceptions such as false path, maximum and minimum delays, and multicycle path
- Minimize the timing skew among the paths from I/O pins to the four sampling registers
- Set the oversampling clock that is used by the oversampling interface to 135 MHz as an independent clock domain

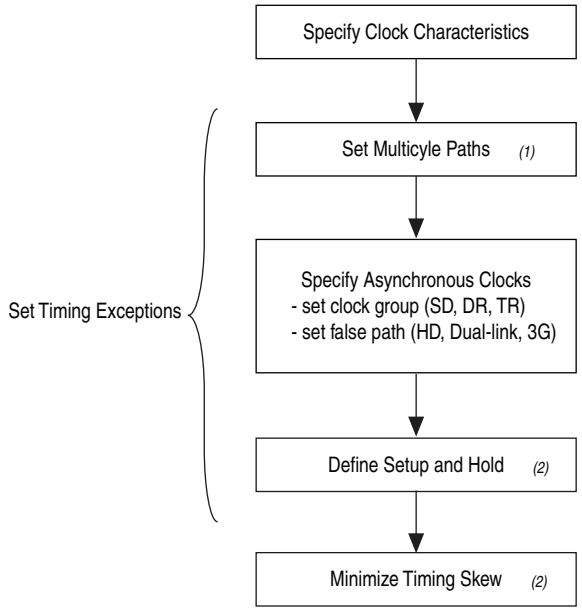
Specifying TimeQuest Timing Analyzer Constraints

To ensure your design meets timing and other requirements, you must constrain the design. This section provides the necessary steps to properly constrain your SDI design using the TimeQuest timing analyzer.

1. Make sure that TimeQuest is specified as the default timing analyzer in the **Timing Analysis Settings** page of the **Settings** dialog box.
2. Perform initial compilation to create an initial design database before you specify timing constraints for your design. On the Processing menu, click **Start Compilation**. A message indicates when compilation is complete.
3. On the Tools menu, click **TimeQuest Timing Analyzer**.
4. Create timing netlist, double-click **Create Timing Netlist** in the **Tasks** pane. The timing netlist appears in the **Report** pane.
5. Specify timing constraints and exceptions. To enter your timing requirements, you can use constraint entry dialog boxes or edit the previously created **.sdc** file.
6. To save your constraints in an **.sdc** file, on the Constraints menu, click **Write SDC File**.

Figure A–1 shows the flow of the constraint design.

Figure A–1. Constraints Design Flow



Notes to Figure A–1:

(1) Applicable for SD-SDI only.

(2) Applicable for Soft SERDES only.

Table A–1 through Table A–3 describe briefly the constraint design flow.

Table A–1. Step 1: Specify Clock Characteristics (Part 1 of 2)

Standard	Clocks	Units
SDI-SD	transceiver_data_rate	270 Mbps
	tx_pclk	27 MHz
	tx_serial_refclk	67.5 MHz
	rx_sd_oversample_clk_in	67.5 MHz
HD-SDI, HD-SDI dual link	transceiver_data_rate	1485 Mbps
	tx_pclk	74.25 MHz
	tx_serial_refclk	74.25 MHz
	rx_serial_refclk	74.25 MHz
3G-SDI	transceiver_data_rate	2970 Mbps
	tx_pclk	148.5 MHz
	tx_serial_refclk	148.5 MHz
	rx_serial_refclk	148.5 MHz
DR, TR	transceiver_data_rate	2970 Mbps
	tx_pclk	148.5 MHz
	tx_serial_refclk	148.5 MHz
	rx_serial_refclk	148.5 MHz

Table A–1. Step 1: Specify Clock Characteristics (Part 2 of 2)

Standard	Clocks	Units
Soft transceiver SDI	rx_sd_refclk_135	135 MHz
	rx_sd_refclk_337	337 MHz
	rx_sd_refclk_337_90°	337 MHz
	tx_sd_refclk_270	270 MHz
	tx_pclk	27 MHz

Table A–2. Step 2: Set Timing Exceptions

Standard	Set Multicycle Paths	set_clock-group	set_false_path	Define Setup and Hold Relationship
SD-SDI	u_format* to u_format	tx_pclk, transmit_pcs0|clkout (gxb_tx_coreclk)	switchline, get_clocks receive_pcs0|clkou (gxb_rxclk)	—
HD-SDI, HD-SDI dual link, 3G-SDI, DR, TR	—	rx_serial_refclk, receive_pcs0|clkout(gxb_rxclk)	switchline, get_clocks receive_pcs0|clkout (gxb_rxclk)	—
		tx_pclk, transmit_pcs0|clkout (gxb_tx_coreclk)	—	
Soft transceiver SDI	—	—	switchline, get_clocks receive_pcs0|clkout (gxb_rxclk)	Setup—1.5 clocks (4.43 ns) from the 337.5-MHz zero-degree clock to the 135-MHz clock
				Hold—zero clocks from the 337.5-MHz clock to the 135-MHz clock

Table A–3. Step 3: Minimize the Timing Skew

Standard	Minimize Timing Skew
SD-SDI	—
HD-SDI, HD-SDI dual link	—
3G-SDI	—
DR, TR	—
Soft transceiver SDI	I/O to sample_a|b|c|d[0] path as short as possible

The following constraints are specifically used to constrain a duplex SDI MegaCore targeting Stratix IV device:

- Specify Clock Characteristics
- Set Multicycle Paths
- Minimize Timing Skew

Specify Clock Characteristics

Use the following constraints for the TimeQuest timing analyzer:

- SD-SDI (rx_sd_oversample_clk_in = 67.5 MHz, tx_pclk = 27 MHz, tx_serial_refclk = 67.5 MHz)

```
create_clock -name {rx_sd_oversample_clk_in} -period 14.814 -waveform {
0.000 7.407 } [get_ports {rx_sd_oversample_clk_in}]

create_clock -name {tx_pclk} -period 14.814 -waveform { 0.000 7.407 }
[get_ports {tx_pclk}]

create_clock -name {tx_serial_refclk} -period 14.814 -waveform { 0.000
7.407 } [get_ports {tx_serial_refclk}]
```

- HD-SDI, HD-SDI dual link (rx_serial_refclk = 74.25 MHz, tx_pclk = 74.25 MHz, tx_serial_refclk = 74.25 MHz)

```
create_clock -name {rx_serial_refclk} -period 13.468 -waveform { 0.000
6.734 } [get_ports {rx_serial_refclk}]

create_clock -name {tx_pclk} -period 13.468 -waveform { 0.000 6.734 }
[get_ports {tx_pclk}]

create_clock -name {tx_serial_refclk} -period 13.468 -waveform { 0.000
6.734 } [get_ports {tx_serial_refclk}]
```

- 3G-SDI (rx_serial_refclk = 148.5 MHz, tx_pclk = 148.5 MHz, tx_serial_refclk = 148.5 MHz)

```
create_clock -name {rx_serial_refclk} -period 6.734 -waveform { 0.000
3.367 } [get_ports {rx_serial_refclk}]

create_clock -name {tx_pclk} -period 6.734 -waveform { 0.000 3.367 }
[get_ports {tx_pclk}]

create_clock -name {tx_serial_refclk} -period 6.734 -waveform { 0.000
3.367 } [get_ports {tx_serial_refclk}]
```

- Dual standard, triple standard SDI

```
create_clock -name {rx_serial_refclk} -period 6.734 -waveform { 0.000
3.367 } [get_ports {rx_serial_refclk}]

create_clock -name {tx_serial_refclk} -period 6.734 -waveform { 0.000
3.367 } [get_ports {tx_serial_refclk}]

create_clock -name {tx_pclk} -period 6.734 -waveform { 0.000 3.367 }
[get_ports {tx_pclk}]
```

- Soft transceiver SDI

```
create_clock -name {rx_sd_refclk_135} -period 7.407 -waveform { 0.000
3.703 } [get_ports {rx_sd_refclk_135}]

create_clock -name {rx_sd_refclk_337} -period 2.967 -waveform { 0.000
1.484 } [get_ports {rx_sd_refclk_337}]

create_clock -name {rx_sd_refclk_337_90deg} -period 2.967 -waveform {
0.000 1.484 } [get_ports {rx_sd_refclk_337_90deg}]

create_clock -name {tx_sd_refclk_270} -period 3.703 -waveform { 0.000
1.852 } [get_ports {tx_sd_refclk_270}]

create_clock -name {tx_pclk} -period 37.037 -waveform { 0.000 18.519 }
[get_ports {tx_pclk}]
```

Set Multicycle Paths

In some device families and speed grades, timing violations may occur in the format block of the SDI MegaCore function. For SD-SDI, these violations are multicycle, and can be fixed by applying the following constraints to your design, (these constraints apply only to SD-SDIs; they are single-cycle paths in all other video standards).

```
set_multicycle_path -setup -end -from [get_keepers
{sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[0].u_txrx_port|sdi_format:format_gen.u_format|*}] -to [get_keepers
{sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[0].u_txrx_port|sdi_format:format_gen.u_format|*}] 2
```

```
set_multicycle_path -hold -end -from [get_keepers
{sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[0].u_txrx_port|sdi_format:format_gen.u_format|*}] -to [get_keepers
{sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[0].u_txrx_port|sdi_format:format_gen.u_format|*}] 1
```

Specify Clocks that are Exclusive or Asynchronous

The SDI MegaCore function may show timing violations in slower speed grade devices. These paths are not required to have fast timing, so you can use the following constraints to remove these timing paths. You can use the command `set_clock_groups` or `set_false_path`.

The following SDC commands are only applicable for duplex core and Stratix IV devices, you must use the constraint entry dialog boxes to constrain the separate RX/TX core and other device families.

- SD-SDI

```
set_clock_groups -exclusive -group [get_clocks {tx_pclk}] -group
[get_clocks
{sdi_megacore_top_inst|sdi_txrx_port_gen[0].u_txrx_port|gen_duplex_alt
4gxb.u_gxb|alt4gxb_component|auto_generated|transmit_pcs0|clkout}]
```

```
set_false_path -from [get_keepers
{sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[0].u_txrx_port|switchline}] -to [get_clocks
{sdi_megacore_top_inst|sdi_txrx_port_gen[0].u_txrx_port|gen_duplex_alt
4gxb.u_gxb|alt4gxb_component|auto_generated|receive_pcs0|clkout}]
```

- HD-SDI, 3G-SDI, dual standard, triple standard SDI

```
set_clock_groups -exclusive -group [get_clocks {rx_serial_refclk}] -
group [get_clocks
{sdi_megacore_top_inst|sdi_txrx_port_gen[0].u_txrx_port|gen_duplex_alt
4gxb.u_gxb|alt4gxb_component|auto_generated|receive_pcs0|clkout}]
```

```
set_clock_groups -exclusive -group [get_clocks {tx_pclk}] -group
[get_clocks
{sdi_megacore_top_inst|sdi_txrx_port_gen[0].u_txrx_port|gen_duplex_alt
4gxb.u_gxb|alt4gxb_component|auto_generated|transmit_pcs0|clkout}]
```

```
set_false_path -from [get_keepers
{sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[0].u_txrx_port|switchline}] -to [get_clocks
{sdi_megacore_top_inst|sdi_txrx_port_gen[0].u_txrx_port|gen_duplex_alt
4gxb.u_gxb|alt4gxb_component|auto_generated|receive_pcs0|clkout}]
```

- HD-SDI dual link (for the additional channel)

```
set_clock_groups -exclusive -group [get_clocks {rx_serial_refclk}] -
group [get_clocks
{sdi_megacore_top_inst|sdi_txrx_port_gen[1].u_txrx_port|gen_duplex_alt
4gxb.u_gxb|alt4gxb_component|auto_generated|receive_pcs0|clkout}]

set_false_path -from [get_keepers
{sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[0].u_txrx_port|switchline}] -to [get_clocks
{sdi_megacore_top_inst|sdi_txrx_port_gen[0].u_txrx_port|gen_duplex_alt
4gxb.u_gxb|alt4gxb_component|auto_generated|receive_pcs0|clkout}]

set_clock_groups -exclusive -group [get_clocks {tx_pclk}] -group
[get_clocks
{sdi_megacore_top_inst|sdi_txrx_port_gen[1].u_txrx_port|gen_duplex_alt
4gxb.u_gxb|alt4gxb_component|auto_generated|transmit_pcs0|clkout}]

set_false_path -from [get_keepers
{sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[1].u_txrx_port|switchline}] -to [get_clocks
{sdi_megacore_top_inst|sdi_txrx_port_gen[1].u_txrx_port|gen_duplex_alt
4gxb.u_gxb|alt4gxb_component|auto_generated|receive_pcs0|clkout}]
```

Define the Setup and Hold Relationship between 135-MHz Clocks and 337.5-MHz zero-degree Clocks

These constraints apply only to soft transceiver SDI.

- Setup—1.5 clocks (4.43 ns) from the 337.5-MHz zero-degree clock to the 135-MHz clock
- Hold—zero clocks from the 337.5-MHz clock to the 135-MHz clock

Use the `set_min_delay` command to specify an absolute minimum delay for a given path.

```
set_min_delay -from  [get_clocks {rx_sd_refclk_337}]  -to  [get_clocks
{rx_sd_refclk_135}] 0.000
```

Use the `set_max_delay` command to specify an absolute maximum delay for a given path.

```
set_max_delay -from [get_clocks {rx_sd_refclk_337}] -to [get_clocks
{rx_sd_refclk_135}] 4.430
```

Minimize Timing Skew

You should minimize the timing skew among the paths from I/O pins to the four sampling registers (`sample_a[0]`, `sample_b[0]`, `sample_c[0]`, and `sample_d[0]`). To minimize the timing skew, manually place the sampling registers close to each other and to the serial input pin. Because these four registers are using four different clock domains, place two of the four registers in one LAB and the other two in another LAB. Furthermore, place the two chosen LABs within the same row regardless of the placement of the serial input. Finally, do not place the four sampling registers at the immediate rows or columns next to the I/O, but at the second row or column next to the I/O bank. This location is because inter-LAB interconnects between I/O banks and their immediate rows or columns are much faster than core interconnect.

The following code is an example of a constraint, which you can set using the Quartus II Assignment Editor:

```
set_location_assignment PIN_99 -to sdi_rx

set_location_assignment LC_X32_Y17_N0 -to
"sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[0].u_txrx_port|soft_serdes_rx:rx_soft_serdes_gen.soft_serdes_rx_inst
|serdes_s2p:u_s2p|sample_a[0]"

set_location_assignment LC_X33_Y17_N0 -to
"sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[0].u_txrx_port|soft_serdes_rx:rx_soft_serdes_gen.soft_serdes_rx_inst
|serdes_s2p:u_s2p|sample_b[0]"

set_location_assignment LC_X32_Y17_N1 -to
"sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[0].u_txrx_port|soft_serdes_rx:rx_soft_serdes_gen.soft_serdes_rx_inst
|serdes_s2p:u_s2p|sample_c[0]"

set_location_assignment LC_X33_Y17_N1 -to
"sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_ge
n[0].u_txrx_port|soft_serdes_rx:rx_soft_serdes_gen.soft_serdes_rx_inst
|serdes_s2p:u_s2p|sample_d[0]"
```

Constraints for the SDI Soft Transceiver

There are constraints specific only to Cyclone devices and there are other constraints that apply to the other device families (including Cyclone II and Cyclone III device families). There are also different constraints that apply to the Classic timing analyzer and the TimeQuest timing analyzer.

Non-Cyclone Devices

These constraints apply to all device families (excluding Cyclone, but including Cyclone II and Cyclone III device families) that are configured to use a soft transceiver for their receivers.

Define the following setup and hold relationship between the 135-MHz clocks and the 337.5-MHz zero-degree clocks:

- Setup—1.5 clocks (4.43 ns) from the 337.5-MHz zero-degree clock to the 135-MHz clock
- Hold—zero clocks from the 337.5-MHz clock to the 135-MHz clock

If you choose to include the PLLs inside the MegaCore function, modify the following constraints and apply them to your design. Alternatively, apply similar constraints to the clocks connected to the `rx_sd_refclk_337` and `rx_sd_refclk_135` signals on your SDI MegaCore function.

Classic Timing Analyzer

Use the following constraints for the Classic timing analyzer:

```
set_instance_assignment -name SETUP_RELATIONSHIP "4.43 ns" -from
"<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_
inst|sdi_clocks:u_sdi_clocks|stratix_c2_pll_sclk:u_rx_pll|altpll:altpl
l_component|_clk0" -to
"<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_
inst|sdi_clocks:u_sdi_clocks|stratix_c2_pll_sclk:u_rx_pll|altpll:altpl
l_component|_clk2"
```

```
set_instance_assignment -name HOLD_RELATIONSHIP "0 ns" -from
"<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_
inst|sdi_clocks:u_sdi_clocks|stratix_c2_pll_sclk:u_rx_pll|altpll:altpl
l_component|_clk0" -to
"<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_
inst|sdi_clocks:u_sdi_clocks|stratix_c2_pll_sclk:u_rx_pll|altpll:altpl
l_component|_clk2"
```

TimeQuest Timing Analyzer

Use the following constraints for the TimeQuest timing analyzer:

```
set_max_delay 4.43 -from
{<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_
inst|sdi_clocks:u_sdi_clocks|stratix_c2_pll_sclk:u_rx_pll|altpll:altpl
l_component|_clk0} -to
{<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_
inst|sdi_clocks:u_sdi_clocks|stratix_c2_pll_sclk:u_rx_pll|altpll:altpl
l_component|_clk2}

set_min_delay 0 -from {
<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_i
nst|sdi_clocks:u_sdi_clocks|stratix_c2_pll_sclk:u_rx_pll|altpll:altpll
_component|_clk0} -to
{<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_
inst|sdi_clocks:u_sdi_clocks|stratix_c2_pll_sclk:u_rx_pll|altpll:altpl
l_component|_clk2}
```

Cyclone Devices Only

These constraints apply to Cyclone devices only (not Cyclone II, Cyclone III, or other device families).

Classic Timing Analyzer

Use the following constraints for the Classic timing analyzer:

```
set_global_assignment -name FMAX_REQUIREMENT "27 MHz" -section_id
input_refclk

set_instance_assignment -name CLOCK_SETTINGS input_refclk -to
rx_27_refclk

set_instance_assignment -name CLOCK_SETTINGS rxclk -to
"<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_s
di_clocks|clkdiv_2p5:cyc_rx_pll_gen.u_clkdiv|clkdiv"

set_global_assignment -name BASED_ON_CLOCK_SETTINGS input_refclk -
section_id rxclk

set_global_assignment -name MULTIPLY_BASE_CLOCK_PERIOD_BY 5 -section_id
rxclk

set_global_assignment -name DIVIDE_BASE_CLOCK_PERIOD_BY 25 -section_id
rxclk

set_global_assignment -name ENABLE_CLOCK_LATENCY ON

set_instance_assignment -name SETUP_RELATIONSHIP "4.43 ns" -from
"<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_s
di_clocks|pll_sclk:cyc_rx_pll_gen.u_rx_pll|altpll:altpll_component|_cl
k0" -to
"<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_s
di_clocks|clkdiv_2p5:cyc_rx_pll_gen.u_clkdiv|clkdiv"
```

```
set_instance_assignment -name HOLD_RELATIONSHIP "0 ns" -from
"<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_s
di_clocks|pll_sclk:cyc_rx_pll_gen.u_rx_pll|altpll:altpll_component|_cl
k0" -to
"<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_s
di_clocks|clkdiv_2p5:cyc_rx_pll_gen.u_clkdiv|clkdiv"
```

TimeQuest Timing Analyzer

Use the following constraints for the TimeQuest timing analyzer:

```
derive_pll_clocks -use_tan_name


create_clock -name rx_27_refclk -period 37.037 -waveform { 0.000 18.518
} [get_ports {rx_27_refclk}]

create_clock -name tx_27_refclk -period 37.037 -waveform { 0.000 18.518
} [get_ports {tx_27_refclk}]

create_generated_clock -name
<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_sd
i_clocks|clkdiv_2p5:cyc_rx_pll_gen.u_clkdiv|clkdiv \
    -source
<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_sd
i_clocks|pll_sclk:cyc_rx_pll_gen.u_rx_pll|altpll:altpll_component|_clk
0 \
    -multiply_by 2 \
    -divide_by 5


set_max_delay -from [get_clocks
{<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_s
di_clocks|pll_sclk:cyc_rx_pll_gen.u_rx_pll|altpll:altpll_component|_cl
k0}] -to [get_clocks
{<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_s
di_clocks|clkdiv_2p5:cyc_rx_pll_gen.u_clkdiv|clkdiv}] 4.430

set_min_delay -from [get_clocks
{<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_s
di_clocks|pll_sclk:cyc_rx_pll_gen.u_rx_pll|altpll:altpll_component|_cl
k0}] -to [get_clocks
{<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_s
di_clocks|clkdiv_2p5:cyc_rx_pll_gen.u_clkdiv|clkdiv}] 0.000
```

Transceiver Clocks

Figure B–1 shows the transceiver clocks for the SDI MegaCore function for version 7.0 and previous.

Figure B–1. Version 7.0 and Earlier Clocks

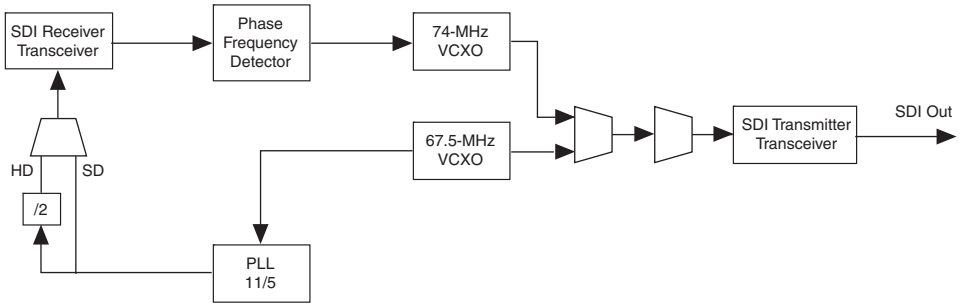


Figure B–2 shows how you should clock the transceivers for current SDI cores. You can now derive all clocks from a single 148.5-MHz voltage controlled crystal oscillator (VCXO) and the transceivers require no external multiplexing.

Figure B–2. Version 7.1 and Later Clocks

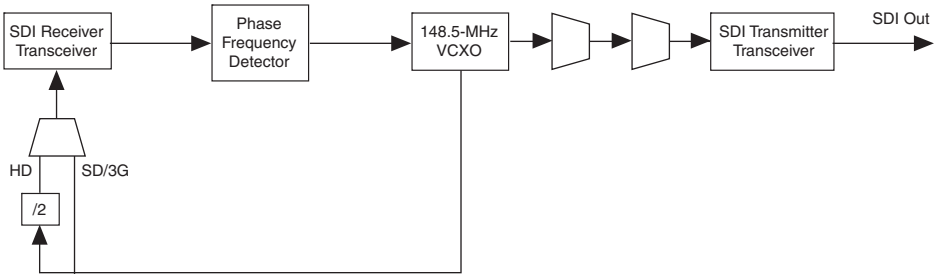
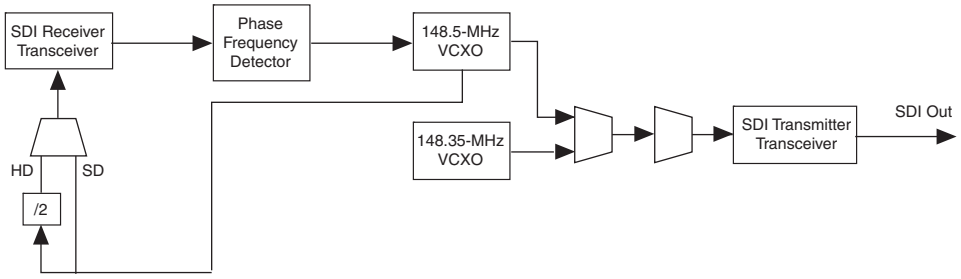


Figure B–3 shows how you should clock the transceivers for version 7.1 and later SDI cores with international clocking. Both American and European standards are catered for.

Figure B–3. Version 7.1 and Later International Clocks



Revision History

The following table shows the revision history for this user guide.

Date	Version	Changes Made
February 2009	9.0	■ Added Arria II GX support. ■ Added a section on RP168. ■ Updated information on video formats.
November 2008	8.1	■ Added a section on Locking Algorithm. ■ Added new signals and updated existing signal descriptions. ■ Updated Appendix A: Constraints.
May 2008	8.0	■ Added Stratix IV support. ■ Improved receiver lock algorithm. ■ Updated 425MB support.
October 2007	7.2	■ Updated device support. ■ Updated standards support—3G-SDI now supports *SMPTE425M-B 2006 3Gb/s Signal/Data Serial Interface – Source Image Format Mapping*. ■ Changed `rx_std` signal description. ■ Added `tx_data_valid_a_bn` signal.
May 2007	7.1	■ Updated device support. ■ Added dual and triple standard information. ■ Added DPRIO information.
December 2006	7.0	Added support for Cyclone III devices.
December 2006	6.1	Updated for new MegaWizard Plug-In Manager.

How to Contact Altera

For the most up-to-date information about Altera products, refer to the following table.

Contact *(Note 1)*	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Altera literature services	Email	literature@altera.com
Non-technical support (General)	Email	nacomp@altera.com
(Software Licensing)	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Indicates command names, dialog box titles, dialog box options, and other GUI labels. For example, **Save As** dialog box.
bold type	Indicates directory names, project names, disk drive names, file names, file name extensions, and software utility names. For example, **\qdesigns** directory, **d:** drive, and **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Indicates document titles. For example, *AN 519: Stratix IV Design Guidelines.*
Italic type	Indicates variables. For example, *n* + 1. Variable names are enclosed in angle brackets (< >). For example, *<file name>* and *<project name>***.pof** file.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and the Options menu.
"Subheading Title"	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, "Typographic Conventions."
`Courier type`	Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. Active-low signals are denoted by suffix `n`. For example, `resetn`. Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets indicate a list of items when the sequence of the items is not important.
	The hand points to information that requires special attention.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The angled arrow instructs you to press Enter.
	The feet direct you to more information about a particular topic.