



SDI MegaCore Function

User Guide



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

MegaCore Version: 7.2
Document Date: October 2007

Copyright © 2007 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



UG-SDI1005-5.0

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 | 1-3 |
| Performance and Resource Utilization | 1-4 |

Chapter 2. Getting Started

| | |
|---|------|
| Design Flow | 2-1 |
| SDI Walkthrough | 2-3 |
| Create a New Quartus II Project | 2-4 |
| Launch MegaWizard Plug-In Manager | 2-5 |
| Parameterize | 2-7 |
| Set Up Simulation | 2-10 |
| Step 3: Generate | 2-12 |
| Simulate the Design | 2-14 |
| Simulate with IP Functional Simulation Models | 2-14 |
| Simulate with the ModelSim Simulator | 2-14 |
| Simulating in Third-Party Simulation Tools Using NativeLink | 2-15 |
| Compile the Design | 2-16 |
| Program a Device | 2-16 |
| Set Up Licensing | 2-16 |

Chapter 3. Functional Description

| | |
|---|------|
| Block Description | 3-1 |
| Transmitter | 3-2 |
| Receiver | 3-4 |
| Transceiver—Soft-Logic Implementation | 3-7 |
| Transceiver—Stratix GX Devices | 3-8 |
| Transceiver—Stratix II GX Devices | 3-15 |
| OpenCore Plus Time-Out Behavior | 3-27 |
| Signals | 3-27 |
| Parameters | 3-33 |
| MegaCore Verification | 3-34 |

Appendix A. Constraints

| | |
|--|-----|
| Introduction | A-1 |
| Minimize Timing Skew | A-1 |
| Constraints for the SDI Soft Transceiver | A-2 |

Contents

| | |
|--|-----|
| Non-Cyclone Devices | A-2 |
| Cyclone Devices Only | A-3 |
| Multicycle Paths in SDI Format Block | A-5 |
| Classic Timing Analyzer | A-5 |
| TimeQuest Timing Analyzer | A-5 |
| False Paths in Stratix GX Devices | A-6 |
| Classic Timing Analyzer | A-6 |
| TimeQuest Timing Analyzer | A-6 |

Appendix B. Clocking

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® SDI MegaCore® function.

| Table 1–1. Release Information | |
|---------------------------------------|--------------------|
| Item | Description |
| Version | 7.2 |
| Release Date | October 2007 |
| Ordering Code | IP-SDI |
| Product ID(s) | 00AE |
| Vendor ID | 6AF7 |

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 |
| Cyclone® | Full (1) |
| Cyclone II | Full |
| Cyclone III | Preliminary |
| HardCopy® II | No support (2) |
| Stratix® | Full |
| Stratix II | Full |

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

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

Note to Table 1–2:

- (1) Cyclone support is limited to –6 speed grade devices.
- (2) Contact your Altera representative for the current HardCopy II support status.

Features

- Support for multiple SDI standards (see Table 1–3)
- Transmitter includes:
 - Cyclical redundancy check (CRC) encoding (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 OpenCore Plus evaluation

General Description

The Altera SDI MegaCore function implements a receiver, transmitter, or full-duplex serial digital interface (SDI) at SD, HD, or 3-gigabits per second (Gbps). The core also supports dual or triple rate standards, which provide automatic receiver rate detection.

The Society of Motion Picture and Television Engineers (SMPTE) have defined an SDI that video system designers widely use 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 1920x1080 and 2048x1080 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

Table 1–2 shows the SDI standard support for various devices.

| Table 1–3. SDI Standard Support | | | | | | |
|--|--------------|----|------------|------------------|---------------|-----------------|
| Device Family | SDI Standard | | | | | |
| | SD | HD | 3-Gbps SDI | HD-SDI Dual Link | Dual Standard | Triple Standard |
| Cyclone | ✓ | | | | | |
| Cyclone II | ✓ | | | | | |
| Cyclone III | ✓ | | | | | |
| Stratix | ✓ | | | | | |
| Stratix II | ✓ | | | | | |
| Stratix III | ✓ | | | | | |
| Stratix II GX | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Stratix GX | ✓ | ✓ | | ✓ | ✓ | |

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 AMPPSM 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 on OpenCore Plus hardware evaluation using the SDI, see [“OpenCore Plus Time-Out Behavior” on page 3–27](#) and *AN 320: OpenCore Plus Evaluation of Megafunctions*.

Performance and Resource Utilization

Table 1–4 shows the typical expected performance for different parameters, using the Quartus®II software, version 7.2.

| Table 1–4. Performance | | | | |
|-------------------------------|-----------------------|------------|----------------------------|------------------------|
| Device | Video Standard | LEs | Combinational ALUTs | Logic Registers |
| Cyclone II | SD-SDI | 840 | – | – |
| Cyclone III | SD-SDI | 839 | – | – |
| Stratix II | SD-SDI | – | 519 | 487 |
| Stratix II GX | SD-SDI | – | 671 | 499 |
| | HD-SDI | – | 770 | 610 |
| | 3-Gpbs HD-SDI | – | 776 | 618 |
| | Dual standard | – | 1,142 | 732 |
| Stratix III | SD-SDI | – | 519 | 490 |

Design Flow

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

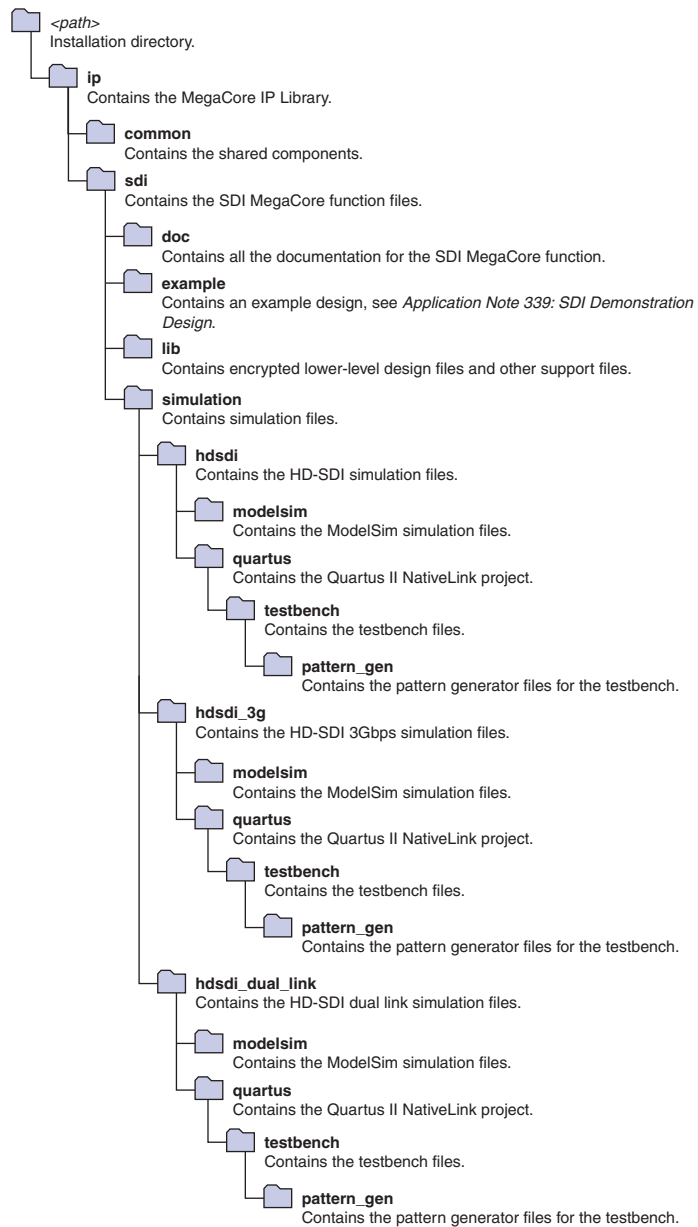
1. Obtain and install the SDI MegaCore function.

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, www.altera.com.



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

Figure 2–1 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\72`; on UNIX and Solaris it is `/opt/altera/72`.

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, see 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(s) on your board.
3. Program the Altera device(s) 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. When you are finished generating a custom variation of the SDI MegaCore function, you can incorporate it into your overall project.

This walkthrough requires the following steps:

- “Create a New Quartus II Project” on page 2–4
- “Launch MegaWizard Plug-In Manager” on page 2–5
- “Parameterize” on page 2–7
- “Set Up Simulation” on page 2–10
- “Step 3: Generate” on page 2–12

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. Choose **New Project Wizard** (File menu).
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 asks if the specified directory should be created. 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** box, 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.

You have finished creating your new Quartus II project.

Launch MegaWizard Plug-In Manager

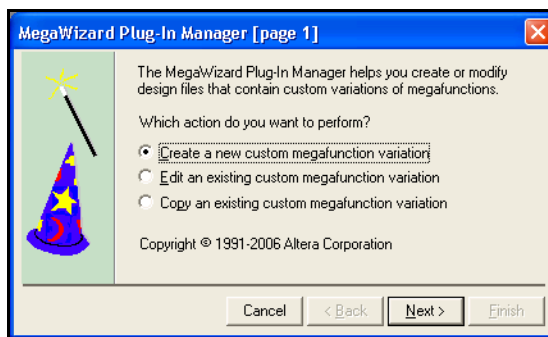
To launch MegaWizard Plug-In Manager in the Quartus II software, follow these steps:

1. Start the MegaWizard® Plug-In Manager by choosing **MegaWizard Plug-In Manager** (Tools menu). The **MegaWizard Plug-In Manager** dialog box displays (see [Figure 2–2](#)).



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

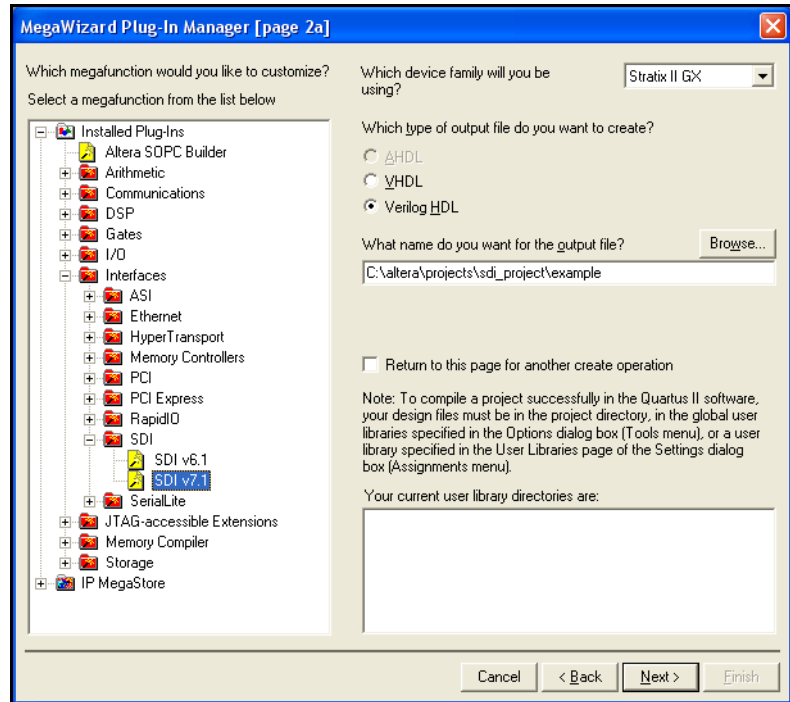
Figure 2–2. MegaWizard Plug-In Manager



2. Specify that you want to create a new custom megafunction variation and click **Next**.
3. Expand the in the **Interfaces > SDI** directory then click **SDI v7.2**.

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>`. [Figure 2–3](#) shows the wizard after you have made these settings.

Figure 2–3. Select the MegaCore Function



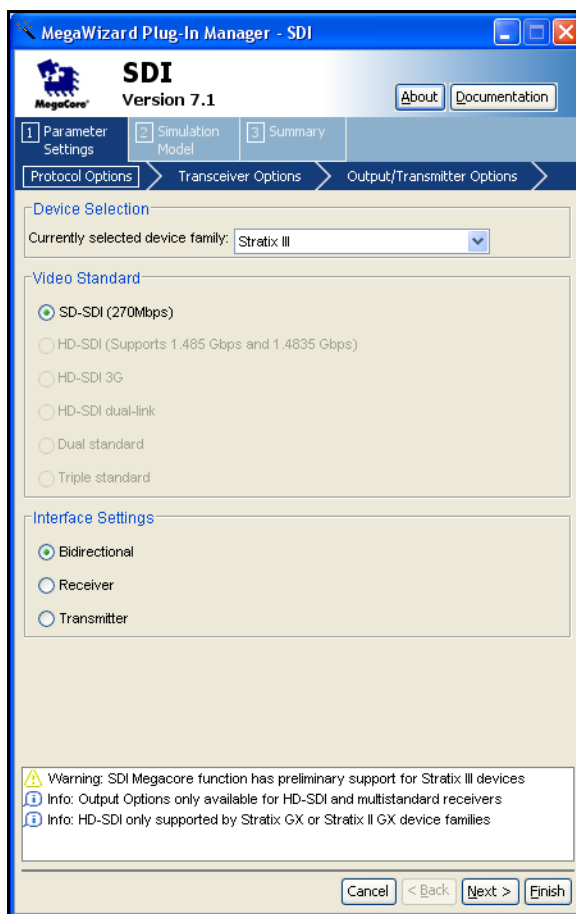
6. Click **Next** to display the **Parameter Settings** page for the SDI MegaCore function (see [Figure 2–4](#)).



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**, **Simulation Model**, or **Summary** tab.

Also, you can directly display individual parameter settings by clicking on the **Protocol Options**, **Transceiver Options**, **Clocking Options**, or **Output Options** tab.

Figure 2–4. Parameters



Parameterize

To parameterize your MegaCore function, follow these steps:



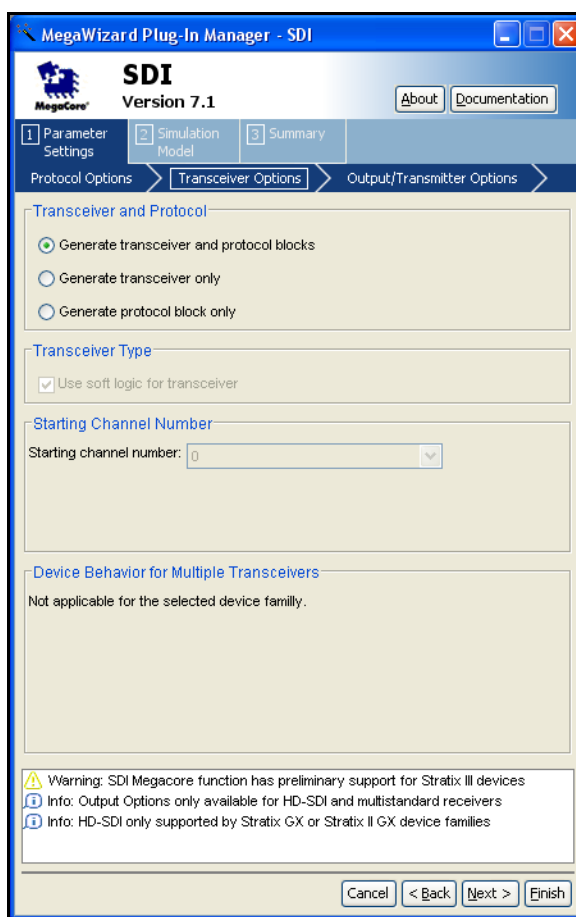
For more information on parameters, see [“Parameters” on page 3–34](#); for more information on the protocol options, refer to [Table 3–13 on page 3–35](#).

1. Select the video standard. Some of the standards may be greyed 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 (see Figure 2–5).



For more information on the transceiver options, refer to [Table 3–14](#) on [page 3–35](#).

Figure 2–5. Select the Transceiver Options



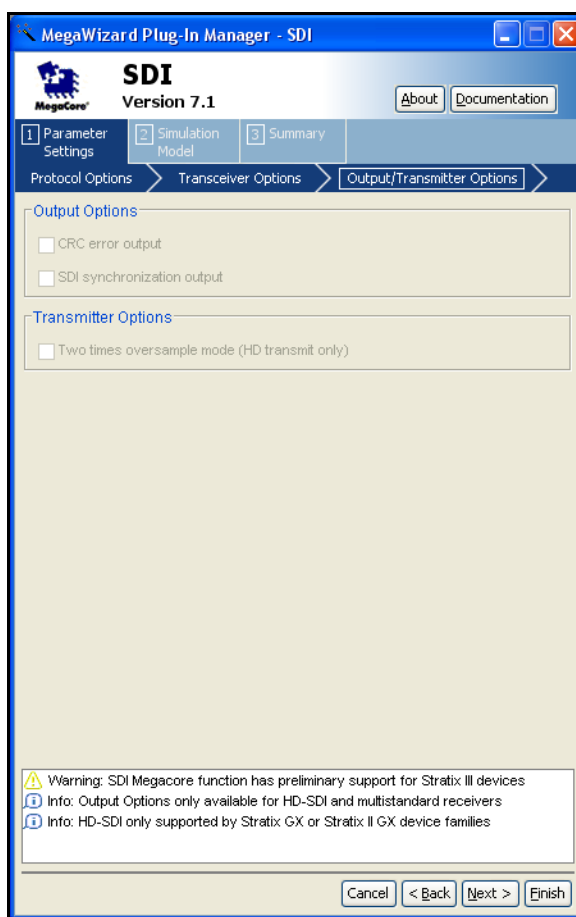
4. Select the transceiver and protocol.

5. For SD only, turn on **Use soft logic transceiver** to implement the transceiver in logic, rather than using Stratix II GX, or Stratix GX transceivers.
6. Select the starting channel number.
7. Click the **Output Options** tab (see Figure 2–6).



For more information on the output options, refer to [Table 3–15 on page 3–36](#).

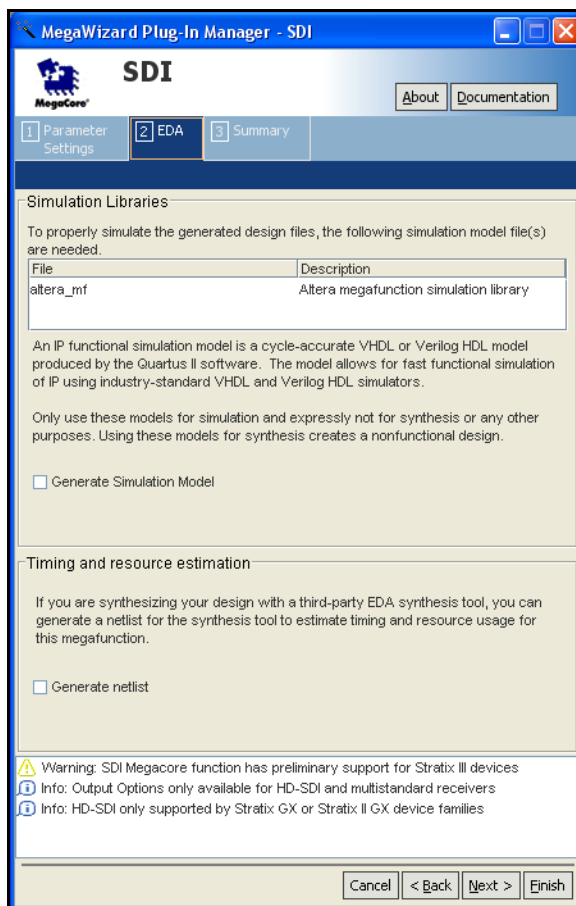
Figure 2–6. Output Options



8. Turn on the required output options.

9. Turn on the required transmitter options.
10. Click **Next** (or the **Simulation Model** tab) to display the simulation setup page (see Figure 2–7).

Figure 2–7. Simulation Model



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.

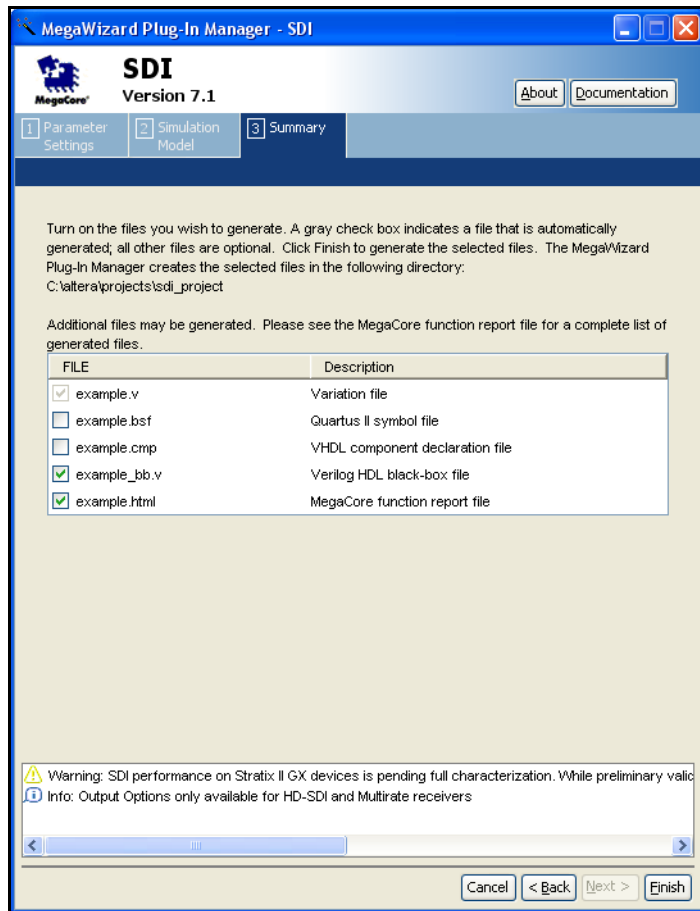


You may only use these models for simulation and expressly not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

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 (see [Figure 2-8](#)).

Figure 2–8. Summary



Step 3: 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 Setting, Simulation Library** or **Summary**, if you want 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** or the pages to display any of the other pages in the MegaWizard Plug-In Manager, if you want 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.

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 | |
|--|---|
| Extension | Description |
| <code><variation name>.vhd</code> , or <code>.v</code> | 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. |
| <code><variation name>.cmp</code> | 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. |
| <code><variation name>.bsf</code> | Quartus II symbol file for the MegaCore function variation. You can use this file in the Quartus II block diagram editor. |
| <code><variation name>.html</code> | MegaCore function report file. |
| <code><variation name>.ppf</code> | 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. |
| <code><variation name>.vo</code> or <code>.vho</code> | VHDL or Verilog HDL IP functional simulation model. |
| <code><variation name>_bb.v</code> | 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. |

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 2–10](#), 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\<video standard>\modelsim** directory, where *<video standard>* is **hdsdi** or **hdsdi_dual_link**. The testbenches instantiate the design and tests the HD 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\<video standard>\modelsim\sdi_run.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_run.bat** in the **simulation\<video standard>\modelsim** directory. This file compiles the design and starts the ModelSim-Altera simulator. A selection of signals is displayed 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, NRZI decoded, descrambled, and then reconstructed into parallel form. The testbench detects the presence of TRS tokens (EAV and 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 Using NativeLink* chapter in volume 3 of the *Quartus II Handbook*.

Altera provide the following three Quartus II projects for use with NativeLink in the `ip\sdi\simulation` directory:

- HD-SDI in the `hdsdi` directory
- HD-SDI 3Gbps 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 Assignments menu click **Settings**, expand **EDA Tool Settings** and select **Simulation**. Select a simulator under **Tool Name** and in **NativeLink Settings**, select **Compile Test Bench** and click **Test Benches**.
6. Click **New**.

7. Enter a name for the **Test bench name**.
8. Enter the name of the project testbench, `tb_sdi_megacore_top`, in **Test bench entity**.
9. Enter the name of the top-level instance in **Instance**.
10. Change **Run for** to 500 μ s.
11. Add the testbench files. In the **File name** field browse to the location of the testbench, `tb_sdi_megacore_top`, click **OK** and click **Add**.
12. Click **OK**.
13. Click **OK**.
14. On the Tools menu point to **EDA Simulation Tool** and click **Run 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.

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, see “[OpenCore Plus Evaluation](#)” on page 1–3, “[OpenCore Plus Time-Out Behavior](#)” on page 3–27, and *Application Note 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.

The SDI MegaCore® function implements a receiver, transmitter, or full-duplex interface. The SDI MegaCore function can handle standard definition (SD) and/or high definition (HD) serial digital interfaces (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 shows the SDI MegaCore function block diagram.



The transmitter contains the following elements:

- The transmitter performs the following functions:

- Altera Corporation**
October 2007

- HD CRC generation and insertion
- Scrambling and non-return to zero inverted (NRZI) coding

For HD the transmitter accepts 20-bit parallel video data; for SD 10-bit parallel data (see [Table 3–10](#) for `txdata` bus definition).

For HD 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 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 LN insertion module takes the 11-bit `tx_ln` and formats and inserts it as two words in the output data. The HD LN insertion module accepts the current line number as an input.

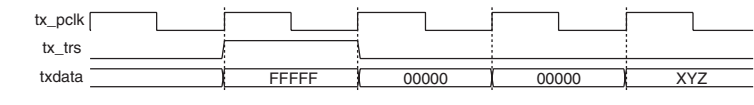


If the system does not know the line number, you can implement logic to detect the output video format and then determine the current line. This function is outside the scope of this SDI MegaCore function.

The LN words (LN0 and LN1) overwrite the two words that follow the “XYZ” word of the end of active video (EAV) timing reference signal (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 the both EAV and SAV TRS (see [Figure 3–2](#)).

Figure 3–2. The `tx_trs` Timing



HD CRC Generation & 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 CRC module generates, formats, and inserts the required CRC in to the output data.

The HD 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 $CRC(X) = X^{18} + X^5 + X^4 + 1$ is applied.

The HD 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 the both EAV and SAV TRS (see [Figure 3–2](#)).

Scrambling & 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 ($G_1(X) = X^9 + X^4 + 1$) followed by NRZI encoding ($G_2(X) = X + 1$). 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 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 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
- HD LN extraction
- HD CRC

The received data is NRZI decoded and descrambled and then presented as a word-aligned parallel output—20-bit for HD; 10-bit for SD (see [Table 3–10](#) for `rxdata` bus definition).

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, 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 & 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. For SDI, the unique `3FF 000 000` pattern is used. For HD SDI, the `3FF 3FF 000 000 000 000` pattern that is found in the combination of the chroma and luma channels is used.

The aligner matches the selected pattern in the descrambled receiver data. If the pattern is seen 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 TRS with the same alignment. When this is seen then that alignment is stored. If two consecutive TRS are subsequently seen with a different alignment then this new alignment is stored.

The aligner allows for an immediate switch from one alignment to another during certain video lines. You can use this feature to support immediate realignment at the vertical interval switching point as defined by SMPTE RP168-1993. This SDI MegaCore function does not use this feature.

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.

HD LN Extraction

The HD 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 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 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 has to be 337.5 MHz, which is $5/4$ of the incoming bit (270-bit rate $\times 5/4 \times 4$ 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, see [“Constraints” on page A-1](#).

Receiver Clocks

The deserializer requires three clocks (see [Table 3-9 on page 3-28](#)), which you can generate from an external source.

Transceiver—Stratix GX Devices

The SDI MegaCore function uses either the appropriate Stratix II GX or Stratix GX transceiver. Interfaces are implemented using the 20-bit interface mode.



For more information on the SDI MegaCore function using the Stratix II GX transceiver, “[Transceiver—Stratix II GX Devices](#)” on [page 3–15](#).



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

The Stratix GX transceiver deserializes the high-speed serial input. For HD, the clock data recovery (CDR) function performs the deserialization and locks the receiver PLL to the receiver data. For SD, 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 or HD data. The data rate can be automatically detected so that the interface can handle both SD and HD 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 and HD SDI, receivers can accommodate the different standards within a single quad.

Transmitter Clocks—Stratix GX Devices

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 it is 27 MHz; for HD 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, because of the oversampling implementation, the serial data rate is five times the SDI bit rate (i.e., 1,350 Mbps).

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

For SD operation, the transmitter reference clock can be derived from `pclk` by using one of the Stratix GX phase-locked loops (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 and HD reference clock.

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

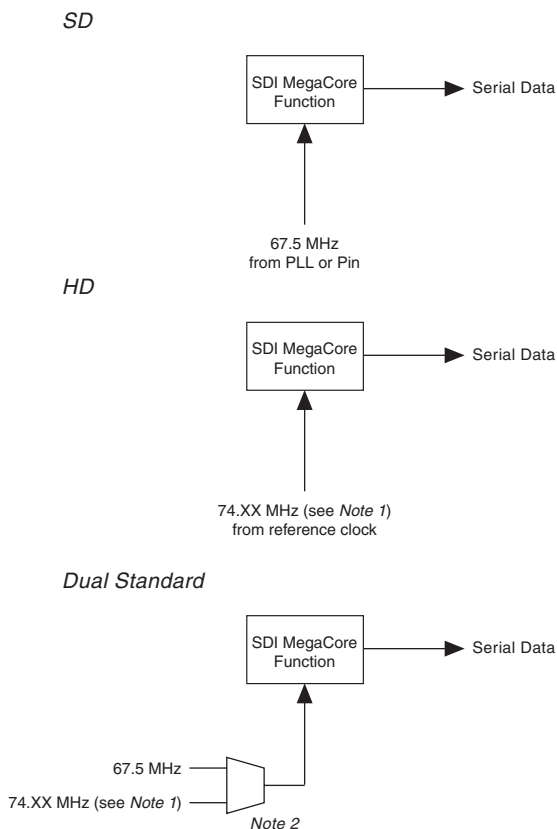
Table 3–1 shows the transmitter clock `tx_serial_refclk` frequencies.

| Table 3–1. Transmitter Clock Frequency—Stratix GX Devices | |
|--|------------------------------|
| Video Standard | Clock Frequency (MHz) |
| SD | 67.5 |
| HD (including dual link) | 74.175/74.25 (1) |
| HD with two times oversample | 148.35/148.5 (1) |
| Dual standard | 67.5/74.175/74.25 (1) |

Note to Table 3–1:

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

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

Figure 3–3. Transmitter Clocks—Stratix GX Devices**Notes to Figure 3–3:**

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

Receiver Clocks—Stratix GX Devices

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

For HD operation, the clock should be nominally $1/20^{\text{th}}$ 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 operation, the clock should be nominally $1/4^{\text{th}}$ of the serial data rate (i.e., 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 at the correct frequency. For HD, 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 3–2 shows the receiver clock `rx_serial_refclk` frequencies.

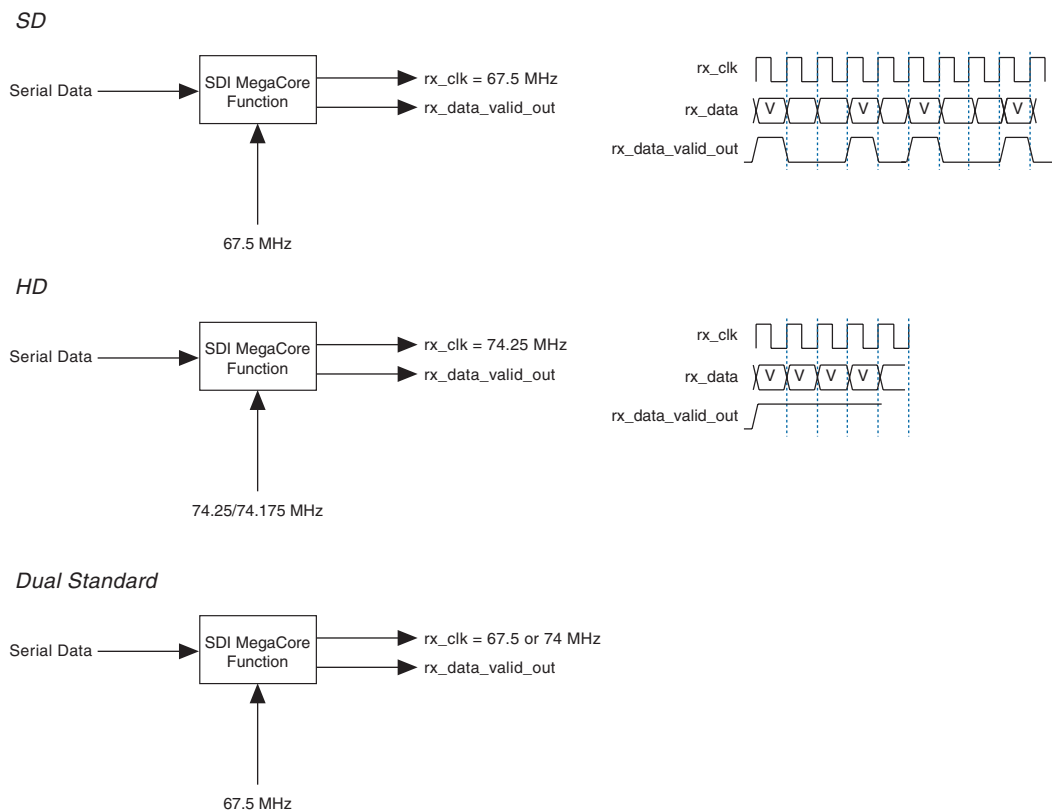
| Table 3–2. Receiver Clock Frequency | |
|--|------------------------------|
| Video Standard | Clock Frequency (MHz) |
| SD | 67.5 |
| HD (including dual link) | 74.175/74.25 (1) |
| Dual standard | 67.5 |

Note to Table 3–1:

- (1) The `rx_serial_refclk` signal must be externally multiplexed.

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

Figure 3–4. Receiver Clocks



Transmitter Transceiver Interface—Stratix GX Devices

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



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 (see [Tables 3–1 and 3–2](#)).

Transmitter Retiming

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

For HD, 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, 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 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 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, see [Table 3–1](#).

SD 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—Stratix GX Devices

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

- Receiver oversampling for SD
- Transceiver receiver mode control



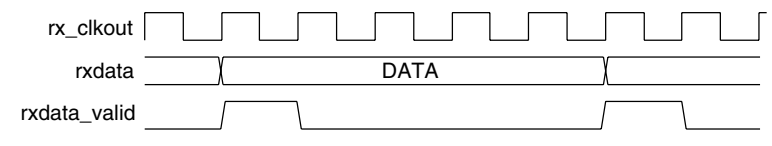
When using the two-times oversampling transmitters in Stratix GX devices, you cannot have HD-SDI receivers in the same quad. The quad requires the same frequency reference clocks for both the receivers and transmitters within a quad. HD-SDI receivers and two-times oversampling transmitters have different frequency reference clocks (see [Tables 3–1](#) and [3–2](#)).

SD 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 receiver oversampling logic.

[Figure 3–5](#) shows an example of the receiver data timing.

Figure 3–5. 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, the transceiver receiver PLL locks to the receiver reference clock.

When the interface receives HD, 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 and HD data by using an algorithm that alternately searches for one rate then the other. First it looks for an HD signal, training the PLL then letting it track the serial data rate. If a valid HD signal is not seen within 0.1 s, the receiver path is reset and the PLL is trained for SD. Conversely, if a valid SD 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 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.

Transceiver—Stratix II GX Devices

The SDI MegaCore function uses either the appropriate Stratix II GX or Stratix GX transceiver. Interfaces are implemented using the 20-bit interface mode.



For more information on the SDI MegaCore function using the Stratix GX transceiver, [“Transceiver—Stratix GX Devices” on page 3–8](#).



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

The Stratix II GX transceiver deserializes the high-speed serial input. For HD, the clock data recovery (CDR) function performs the deserialization and locks the receiver PLL to the receiver data. For SD, 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 or HD data. The data rate can be automatically detected so that the interface can handle both SD and HD without the need for device reconfiguration.

Stratix II GX 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 and HD SDI, receivers can accommodate the different standards within a single quad.

Transmitter Clocks—Stratix II GX Devices

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 it is 27 MHz; for HD it is 74.25 or 74.175 MHz; for 3-Gpbs 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, because of the oversampling implementation, the serial data rate is five times the SDI bit rate (i.e., 1,350 Mbps); for the triple-standard SDI, the oversampling rate is 11.

For SD operation, the transmitter reference clock can be derived from `pclk` by using one of the Stratix II GX phase-locked loops (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.

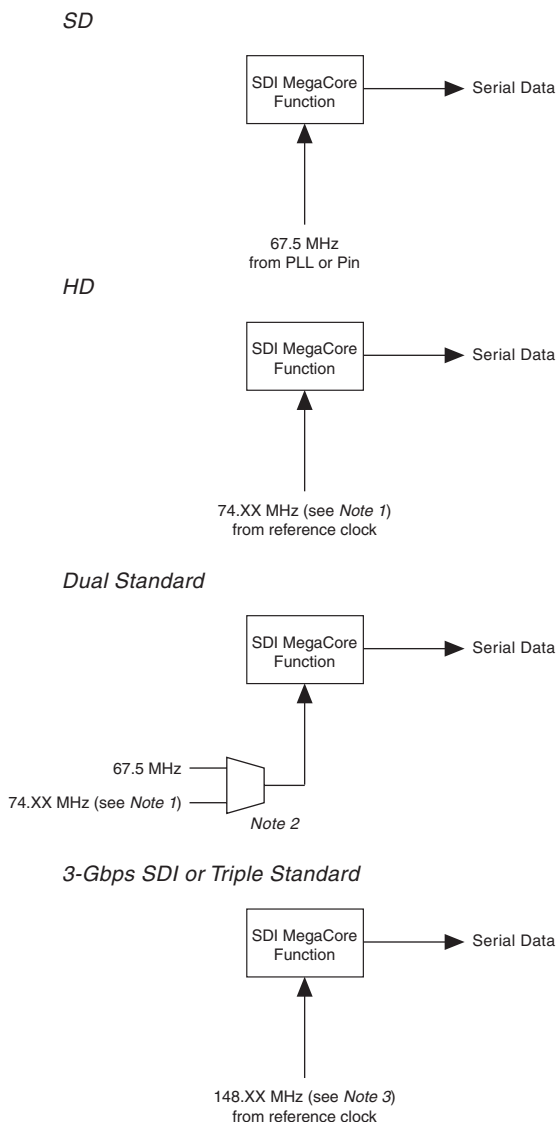
Table 3–3 shows the transmitter clock `tx_serial_refclk` frequencies.

| Table 3–3. Transmitter Clock Frequency—Stratix II GX Devices | |
|---|------------------------------|
| Video Standard | Clock Frequency (MHz) |
| SD | 67.5 |
| HD (including dual link) | 74.175/74.25 (1) |
| HD with two times oversample | 148.35/148.5 (1) |
| Dual standard | 67.5/74.175/74.25 (1) |
| Triple standard | 148.35/148.5 (1) |
| 3Gbps SDI | 148.35/148.5 (1) |

Note to Table 3–3:

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

Figure 3–6 shows the transmitter clocks for different video standards.

Figure 3–6. Transmitter Clocks—Stratix II GX Devices**Notes to Figure 3–6:**

- (1) This frequency can be either 74.175 or 74.25 MHz, to support 1.4835/1.485 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 2.967/2.970 Gbps HD-SDI respectively.

Receiver Clocks—Stratix II GX Devices

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

For HD operation, the clock should be nominally $1/20^{\text{th}}$ 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 operation, the clock should be nominally $1/4^{\text{th}}$ of the serial data rate (i.e., 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 signals by a factor of 11.

All receiver interfaces can share a common receiver reference clock.

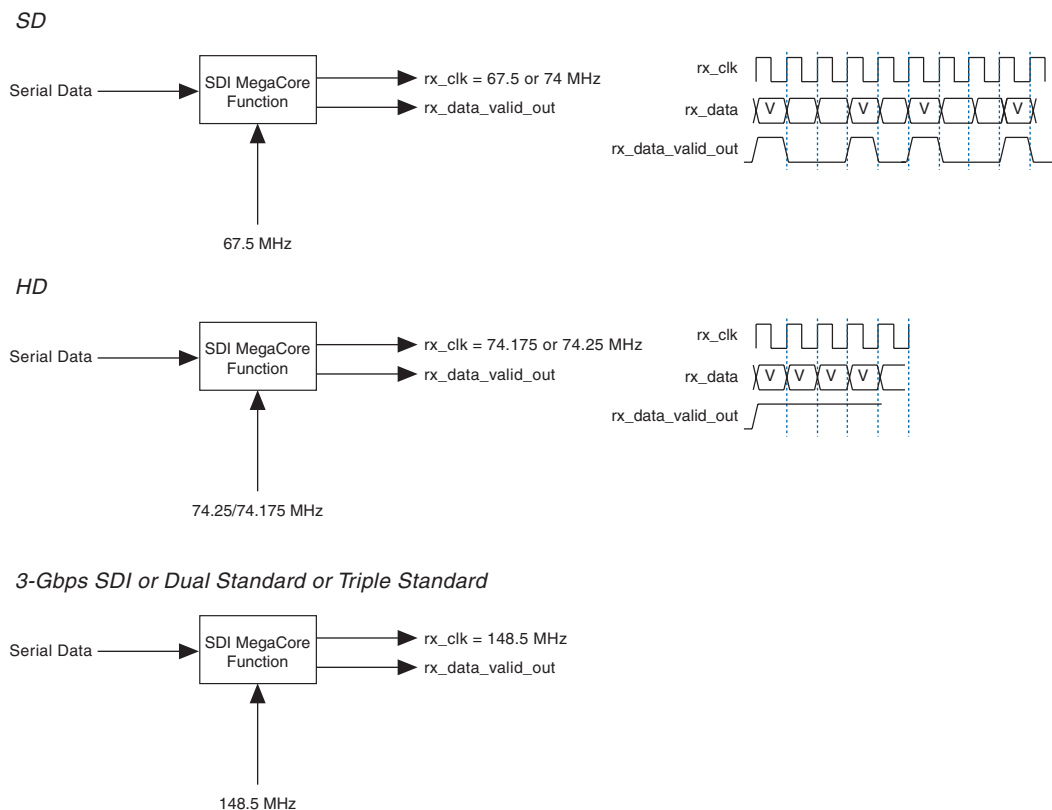
Table 3–4 shows the receiver clock `rx_serial_refclk` frequencies.

| Table 3–4. Receiver Clock Frequency—Stratix II GX Devices | |
|--|------------------------------|
| Video Standard | Clock Frequency (MHz) |
| SD | 67.5 |
| HD (including dual link) | 74.175/74.25 (1) |
| Dual or triple standard | 148.35/148.5 (1), (2) |
| 3Gbps SDI | 148.35/148.5 (1) |

Note to Table 3–4:

- (1) You can use either reference clock for training.
- (2) Must be 148.5 MHz for correct SD operation.

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

Figure 3–7. Receiver Clocks

Transmitter Transceiver Interface—Stratix II GX Devices

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

For HD, 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, 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 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 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, see [Table 3–1](#).

SD 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—Stratix II GX Devices

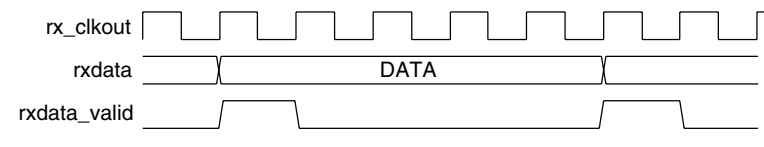
Altera provides a transceiver interface, which interfaces the transceiver to the SDI function. The transceiver interface implements the following functions:

- Receiver oversampling for SD
- Transceiver receiver mode control

SD Receiver Oversampling

The Stratix II GX transceiver does not support CDR for data rates less than 622 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 receiver oversampling logic.

[Figure 3–8](#) shows an example of the receiver data timing.

Figure 3–8. 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, the transceiver receiver PLL locks to the receiver reference clock.

When the interface receives HD, 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 (see [“DPRIO for Dual Standard & Triple Standard Receivers” on page 3–21](#)) 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.

DPRIO for Dual Standard & Triple Standard Receivers

Dual standard and triple standard SDI receivers (or receivers of duplex SDIs) require the dynamic partially reconfigurable IO (DPRIO) feature of Stratix II GX devices, to perform autodetection and locking to different SDI rates.



For more information on DPRIO, refer to the *Stratix II GX Handbook*.

[Table 3–5](#) shows the DPRIO requirements.

| Table 3–5. DPRIO Requirements | | | |
|--------------------------------------|-----------------|--------------------|---------------|
| SDI Standard | Receiver | Transmitter | Duplex |
| SD | No | No | No |
| HD | No | No | No |

Table 3–5. DPRIO Requirements

| SDI Standard | Receiver | Transmitter | Duplex |
|-----------------|----------|-------------|--------|
| Dual standard | Yes | No | Yes |
| Triple standard | Yes | No | Yes |

DPRIO allows the settings for the Stratix II GX transceiver (ALT2GXB) to be changed at any time. DPRIO allows the transceiver to be reprogrammed to support the three SDI rates.

The triple standard SDI uses 11 times oversampling for receiving SD-SDI. Hence, only two Stratix II GX transceiver configurations are required as the rates for 3-Gbps SDI and SD 11 times are the same. [Table 3–6](#) shows the rates.

Table 3–6. Rates

| SDI Standard | Data Rate | Oversampling | Transceiver Rate (MHz) | ALT2GXB Reference Clock (MHz) |
|--------------|------------|--------------|------------------------|-------------------------------|
| SD-SDI | 270 Mbps | 11 times | 2,970 | 148.5 |
| HD-SDI | 1.485 Gbps | None | 1,485 | 148.5 (1) |
| 3-Gbps SDI | 2.970 Gbps | None | 2,970 | 148.5 (1) |

Note to [Table 3–6](#):

(1) Also supports the 1/1.001 rates.

The reprogramming of the ALT2GXB requires the ALT2GXB_RECONFIG megafunction.



For more information on the ALT2GXB_RECONFIG megafunction, refer to the *ALT2GXB Megafunction User Guide*.

ALT2GXB Operation

The alternative setups for the ALT2GXB settings are stored in 28 words by 16-bit ROMs within the Stratix II GX device. The 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.

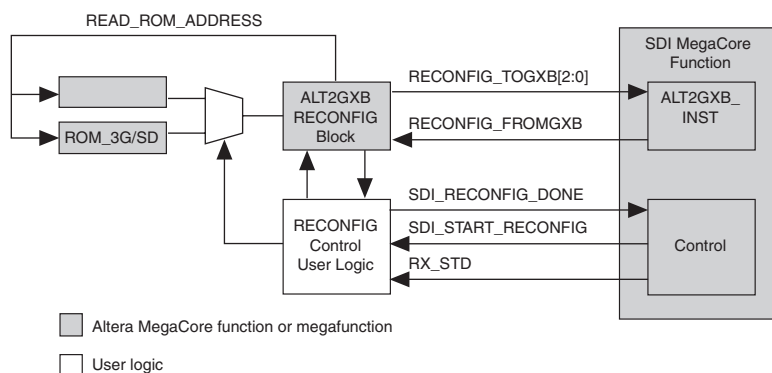
- The ALT2GXB_RECONFIG block reads the appropriate ROM and serializes the data.
- The ALT2GXB_RECONFIG block applies the serial data to the correct ALT2GXB instance.
- When this process has finished, the ALT2GXB_RECONFIG block indicates to the SDI that reprogramming is complete.
- The SDI starts the process of locking to the incoming data.



Some user logic is required to handle the handshaking between the SDI MegaCore function and the ALT2GXB megafunction. For an example see the example design in the `example\source\sdi_dprio` directory.

Figure 3–9 shows a block diagram of the design elements.

Figure 3–9. DPRIO Block Diagram



The ROM_HD and ROM_3G/SD ROMs hold the ALT2GXB setting information for each of the video standards. The setup for SD and 3-Gbps SDI is the same, so only two ROMs are required: one for SD and 3Gbps SDI and one for HD.

The ALT2GXB_RECONFIG block handles the programming of the ROM contents into the ALT2GXB 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 ALT2GXB 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 ALT2GXB megafunction is embedded inside the SDI MegaCore function. The reprogramming ports for the ALT2GXB 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

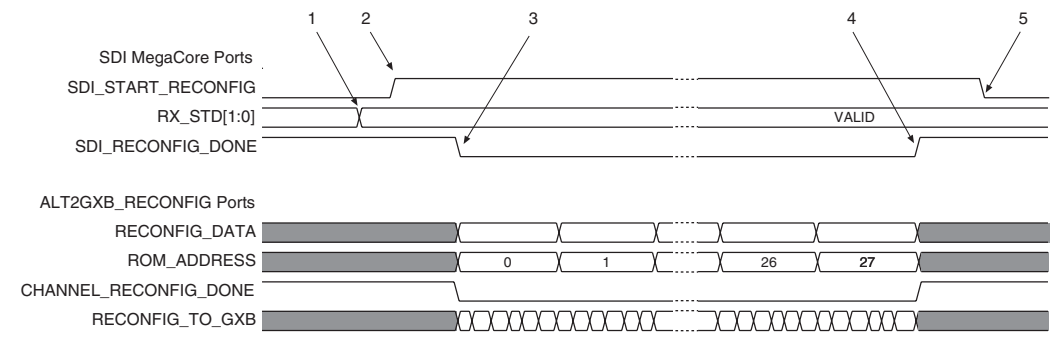
The SDI_START_RECONFIG and SDI_RECONFIG_DONE signals handle the handshaking between the SDI MegaCore function and the user logic. The RX_STD signal is also required to select the correct ROM instance.



Table 3–12 on page 3–34 shows the five signals that handle the DPRIO operation.

Figure 3–9 shows this handshaking and the expected output of some of the ALT2GXB_RECONFIG signals.

Figure 3–10. 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

The contents of the ROM are set by memory initialization files (MIFs). 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 to turn this feature on. On the Assignments menu click **Settings**. Expand Fitter Settings and click **More Settings**. Select **Generate Stratix II GX Reconfig MIF** and select **On**.

For the SDI MegaCore function, the Quartus II-generated MIF file is for 3-Gbps SDI setup.



These MIF files relate to a specific ALT2GXB instance in the device. Therefore, you cannot use the same MIF 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 and SD or 3-Gbps setups. The three bits are in word 23 and can be seen in the following examples of the MIF files (see [Example 3–1](#) and [Example 3–2](#)).

Example 3–1. 3G ROM Content Example

```
....
22 : 1010100000011111;
23 : 01111110000010100;
24 : 0001000101101000;
....
```

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

```
...
22 : 1010100000011111;
23 : 0111111000001101;
24 : 0001000101101000;
```

....

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

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 and has logic that modifies the ROM read data when word 23 is read.

Starting Channel Number

So each transceiver can be correctly addressed by the ALT2GXB_RECONFIG block, a starting channel number must be specified for transceiver instance. This starting channel number must meet certain criteria for the DPRIO.



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

You can select the starting channel number for each SDI instance in the SDI MegaCore interface.

Quartus II Design Flow

A two-pass compilation is required for SDI MegaCore 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 ROMs and edit word 23.
4. Set the appropriate ROMs to use the MIF files generated in steps 2 and 3.

- Run the Quartus II compilation.

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 timeout is 1 hour; the tethered timeout 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, see [“OpenCore Plus Evaluation” on page 1–3](#) and *AN 320: OpenCore Plus Evaluation of Megafunctions*.

Signals

[Table 3–7](#) shows the receiver clock signals.

| Table 3–7. Receiver Clock Signals | | |
|--|------------------|---|
| Signal | Direction | Description |
| <code>gxb2_cal_clk</code> | Input | Calibration clock for the Stratix II GX transceivers only. |
| <code>rx_27_refclk</code> | Input | 27-MHz clock reference, which is an input to the PLL, if you include PLLs . |
| <code>rx_sd_oversample_clk_in</code> | Input | 67.5-MHz oversample clock input. SD only. |
| <code>rx_serial_refclk</code> | Input | Transceiver training clock. HD only. |
| <code>gxb_tx_clkout</code> | 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 <code>tx_clkout</code> signal of the <code>alt2gxb</code> megafunction. For more information on <code>tx_clkout</code> , see the <i>Stratix II GX alt2gxb Megafunction User Guide</i> . |

Table 3–7. Receiver Clock Signals

| Signal | Direction | Description |
|--------------------------|-----------|---|
| rx_clk | Output | Transceiver clock data recovery (CDR) clock. |
| rx_sd_oversample_clk_out | Output | 67.5-MHz oversample clock output for cascading MegaCore functions. SD only. |

Table 3–8 shows the transmitter clock signals.

Table 3–8. 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 = 27 MHz; for HD = 74 MHz and for 3Gbps SDI = 148 MHz. |
| tx_serial_refclk | Input | Transceiver reference clock input. Low jitter. See Table 3–1. |
| tx_pclk_out | Output | Transmitter parallel clock output. |
| tx_sd_oversample_clk_out | Output | 67.5-MHz oversample clock output for cascading MegaCore functions. SD only. |

Table 3–9 shows the soft transceiver clock signals.

Table 3–9. 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 the receiver. |
| tx_sd_refclk_270 | Input | Soft transceiver 270-MHz serial clock for the transmitter. |

Table 3–10 shows the interface signals.

Table 3–10. Interface Signals (Part 1 of 4)

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

Table 3–10. Interface Signals (Part 2 of 4)

| Signal | Width | Direction | Description |
|--------------------|---------------|-----------|--|
| enable_sd_search | – | Input | Enables search for SD signal in dual or triple standard mode. |
| enable_ln | $[(N-1):0]$ | Input | Enables LN insertion. HD only. |
| enable_tx_pattern | $[(N-1):0]$ | Input | Turns on or off the pattern generator in the transmitter path; one per channel. |
| rst | – | Input | Reset signal, which holds the receiver and transmitter in reset. |
| rx_protocol_clk | $[(N-1):0]$ | Input | External clock for protocol data. |
| rx_protocol_hd_sdn | $[(N-1):0]$ | Input | Selection of HD or SD processing for dual or triple standard protocol block. This signal only appears on dual or triple standard protocol blocks and indicates HD(1) or SD(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 | $[(20N-1):0]$ | 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. |
| sdi_rx | $[(N-1):0]$ | Input | Serial input. |
| select_tx_pattern | $[1:0]$ | Input | Selects output test pattern. Bit 1 enables pathological test pattern; bit 0 when 1 enables 100% color bars, when 0 enables 75% color bars. |
| txdata | $[(20N-1):0]$ | Input | <p>User-supplied transmitter parallel data. SD uses 9:0; HD uses $20N-1:0$.</p> <p>For SD, txdata [9:0] is interleaved chroma and luma data, (19:10 are unused); bit 9 is the MSB.</p> <p>For HD, txdata [9:0] is the data for chroma, bit 9 is the MSB; txdata [19:10] is the data for luma, bit 19 is the MSB.</p> |

Table 3–10. Interface Signals (Part 3 of 4)

| Signal | Width | Direction | Description |
|----------------------|---------------|-----------|---|
| tx_data_valid_a_bn | 1 | Input | Identifies the transmitter data word on txdata as either virtual channel A data (logic high) or virtual channel B data (logic low) (triple standard or 3-Gbps SDI transmitter or duplex cores only). For the <i>SPMTE 425M-A</i> format, set to a logic high level. For <i>SMPTE 425M-B</i> format, set to logic 1 when a virtual channel A word is presented to the core. The next word (by definition) should be a virtual channel B word. At this point tx_data_valid_a_bn should be driven to logic 0. Continuing this interleaving scheme, tx_data_valid_a_bn signal is 1 for a clock cycle, then 0 for a cycle, then 1 for a cycle—a square wave. See Figures 3–13 and 3–14 and <i>SMPTE425M-B 2006 3Gb/s Signal/Data Serial Interface – Source Image Format Mapping</i> . |
| tx_ln | [(11N – 1):0] | Input | Transmitter line number. For use in HD LN insertion. |
| tx_pattern_std | [1:0] | Input | Selects video test pattern standard. 00 = 1080p, 01 = 1080i, 10 = NTSC (SD), 11 = PAL(SD). |
| tx_std_select_hd_sdn | – | Input | Selects HD or SD for dual or triple standard . 1 = HD; 0 = SD. |
| tx_trs | [(N – 1):0] | Input | Transmitter TRS input. For use in HD LN and CRC insertion. Assert on first word of both EAV and SAV TRS (see Figure 3–2). |
| 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 | [(N – 1):0] | Output | Receiver active picture synchronization output. |
| rxdata | [(20N – 1):0] | Output | Receiver parallel data. SD uses 9:0; HD uses 20N – 1:0. For SD, rxdata [9:0] is interleaved chroma and luma data, (19:10 are unused); bit 9 is the MSB. For HD, rxdata [9:0] is the data for chroma, bit 9 is the MSB; rxdata [19:10] is the data for luma, bit 19 is the MSB. |

Table 3–10. 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 <code>rxdata</code> is valid. Bit [0] of this bus indicates valid data on <code>rxdata</code> . When receiving <i>SMPTE 425M-B</i> signals in 3-Gbps SDI or triple standard, bit [1] indicates that data on <code>rxdata</code> is from virtual channel B; bit [0] indicates the data is from virtual channel A. See Figures 3–11 and 3–12 and <i>SMPTE425M-B 2006 3Gb/s Signal/Data Serial Interface – Source Image Format Mapping</i> . |
| rx_F | $[(N-1):0]$ | Output | Receiver frame synchronization output. |
| rx_H | $[(N-1):0]$ | Output | Receiver horizontal synchronization output. |
| rx_ln | $[(N-1):0]$ | Output | Receiver line number output. |
| rx_std_flag_hd_sdn | – | Output | Indicates received standard for dual or triple standard only. HD = 1; SD = 0. |
| rx_V | $[(N-1):0]$ | Output | Receiver vertical synchronization output. |
| sdi_tx | $[(N-1):0]$ | Output | Serial output. |
| tx_protocol_out | $[(20N-1):0]$ | Output | Data out (protocol only mode). |

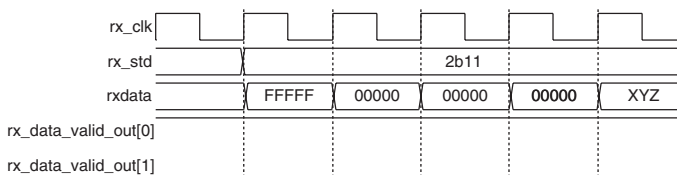
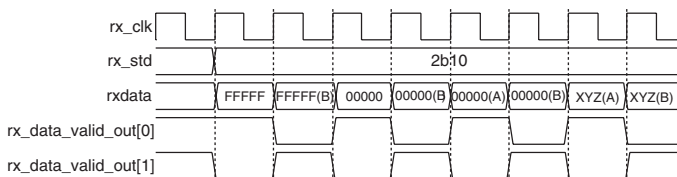
Figure 3–11. Behavior of rx_data_valid_out—425A**Figure 3–12. Behavior of rx_data_valid_out—425B**

Figure 3–13. Behavior of tx_data_valid_a_bn—425A

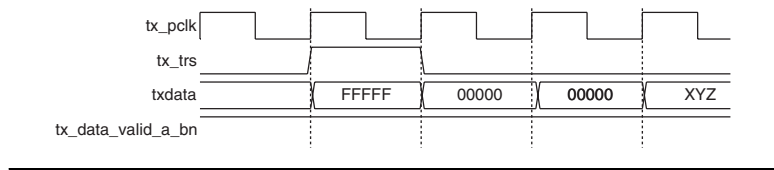


Figure 3–14. Behavior of tx_data_valid_a_bn—425B

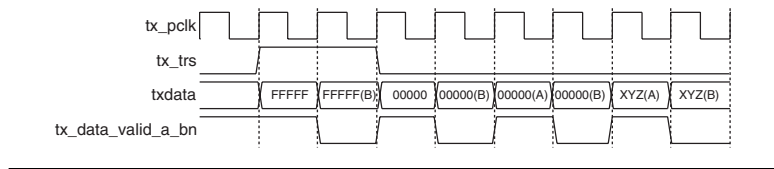


Table 3–11 shows the status signals

| Table 3–11. Status Signals (Part 1 of 3) | | | |
|---|---------------|------------------|--|
| Signal | Width | Direction | Description |
| rx_anc_data | [(20N – 1):0] | Output | Received ancillary data. For SD, rx_anc_data [9:0] is the data, (19:10 are unused); bit 9 is the MSB. For HD, 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 | [(2N – 1):0] | Output | Ancillary data or checksum error. For SD, rx_anc_error [0] is the error flag, (1 is unused). For HD, rx_anc_error [0] is the error flag for chroma; rx_anc_error [1] is the error flag for luma. |
| rx_anc_valid | [(2N – 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, rx_anc_valid [0] is the valid flag, (1 is unused). For HD, rx_anc_valid [0] is the valid flag for chroma; rx_anc_valid [1] is the valid flag for luma. |

Table 3–11. Status Signals (Part 2 of 3)

| Signal | Width | Direction | Description |
|------------------------|-------------|-----------|---|
| <code>rx_status</code> | 10:0 | Output | <p>Receiver status:</p> <ul style="list-style-type: none"> • <code>rx_status[10]</code> dual-link ports aligned • <code>rx_status[9]</code> link 2 frame locked • <code>rx_status[8]</code> link 2 TRS locked (six consecutive TRS with same timing) • <code>rx_status[7]</code> link 2 alignment locked (a TRS has been spotted and word alignment performed) • <code>rx_status[6]</code> link 2 receiver in reset • <code>rx_status[5]</code> link 2 transceiver PLL locked • <code>rx_status[4]</code> link 1 Frame locked • <code>rx_status[3]</code> link 1 TRS locked (six consecutive TRS with same timing) • <code>rx_status[2]</code> link 1 alignment locked (a TRS has been spotted and word alignment performed) • <code>rx_status[1]</code> link 1 receiver in reset • <code>rx_status[0]</code> link 1 transceiver PLL locked <p>For non HD-SDI dual-link versions, only bits 4:0 are active.</p> <p>In Stratix GX designs, <code>rx_status[0]</code> and <code>rx_status[5]</code> connect to the <code>pll_locked</code> signal of the ALT2GXB megafunction; in Stratix II GX designs, they are the inverse of the <code>rx_pll_locked</code> signal on the ALT2GXB. This active low signal indicates lock of the PLL when the ALT2GXB is training from a <code>refclk</code> source. This signal may oscillate when the ALT2GXB is correctly locked to incoming data in HD or 3-Gbps modes. In SD modes, this signal should remain at logic 0 at all times.</p> <p>For <code>rx_status[3]</code> and <code>rx_status[8]</code> 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.</p> |
| <code>tx_status</code> | $[(N-1):0]$ | Output | <p>Transmitter status, which indicates the transmitter PLL has locked to the <code>tx_serial_refclk</code> signal. This signal is active high for Stratix GX devices and active low for StratixII GX devices.</p> |

Table 3–11. Status Signals (Part 3 of 3)

| Signal | Width | Direction | Description |
|--------|-------|-----------|--|
| tx_std | [1:0] | Output | Transmitter standard. 00 for SD; 01 for HD; 11 for 3-Gbps. |

Note to Table 3–11:

(1) N is the channel count.

Table 3–12 shows the five signals that handle the DPRIO operation.

Table 3–12. DPRIO Signals

| Name | Direction | Description |
|-----------------------------|-----------|---|
| SDI_START_RECONFIG | Output | Request from MegaCore function to start reconfiguration. |
| SDI_RECONFIG_DONE | Input | Indication back to MegaCore function that reconfiguration has finished. |
| SDI_RECONFIG_TOGXB[2:0] (1) | Input | Data input for the embedded ALT2GXB instance. |
| SDI_RECONFIG_FROMGXB (1) | Output | Data output from embedded ALT2GXB instance. |
| SDI_RECONFIG_CLK | Input | Clock input for the embedded ALT2GXB instance. |
| RX_STD[1:0] | Output | <p>Receive video standard. 00 = SD, 01 = HD, 10 = 3G.</p> <p>The SDI MegaCore function can recover both <i>SMPTE 425M-A</i> and <i>425MB</i> formatted streams. The receiver indicates which format it detects by setting the level of the rx_std bus:</p> <ul style="list-style-type: none"> rx_std[1:0] = 2'b11 = <i>425M-A</i> rx_std[1:0] = 2'b10 = <i>425M-B</i> |

Note to Table 3–12:

(1) These signals must be connected directly to an ALT2GXB_RECONFIG instance.

Parameters

The parameters can only be set in the MegaWizard Plug-In Manager (see “Parameterize” on page 2-7).

Table 3–13 shows the protocol options.

| Table 3–13. Protocol Options | | |
|-------------------------------------|--|--|
| Parameter | Value | Description |
| Video standard | SD SDI, HD SDI, HD-SDI 3G, HD-SDI dual-link, dual or triple standard SDI | Selection of HD or SD SDI. Selecting HD switches in LN insertion and extraction and CRC generation and extraction blocks; selecting SD switches out LN insertion and extraction and CRC generation and extraction blocks. Selecting SD also includes oversampling logic. Selecting dual or triple standard SDI includes the processing blocks for both SD 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, just create two different MegaCore variations. |

Table 3–14 shows the transceiver options.

| Table 3–14. Transceiver Options | | |
|--|--|---|
| Parameter | Value | Description |
| Transceiver or Protocol | Generate transceiver and protocol blocks, generate transceiver block only, or generate protocol block only | Selects transceiver or protocol blocks or both. |
| Use soft logic for transceiver | On or off | Uses soft logic to implement the transceiver logic, rather than using Stratix II GX, or Stratix GX transceivers. SD 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 3–15 shows the output options.

| Table 3–15. Output Options | | |
|-----------------------------------|--------------|---|
| Parameter | Value | Description |
| CRC error output | On or off | Turns on or off CRC monitoring (HD only). |
| SDI synchronization outputs | On or off | Provides synchronization outputs (HD only). |
| Two times oversample mode | On or off | HD transmitter only. When turned on runs the transceiver at twice the rate and has improved jitter performance. Requires 148.5-MHz <code>tx_serial_refclk</code> 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*

Introduction

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

- 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

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 2 chosen LABs within the same row whatever the placement of the serial input. Finally, do not place the four sampling registers at the immediate rows or columns next to the IO, but the second one 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_gen[0]
.u_txrx_port|soft_serdes_rx:rx_soft_serdes_gen.soft_serdes_rx_inst|serdes_s2p:u_s2p|sample_a[0]"

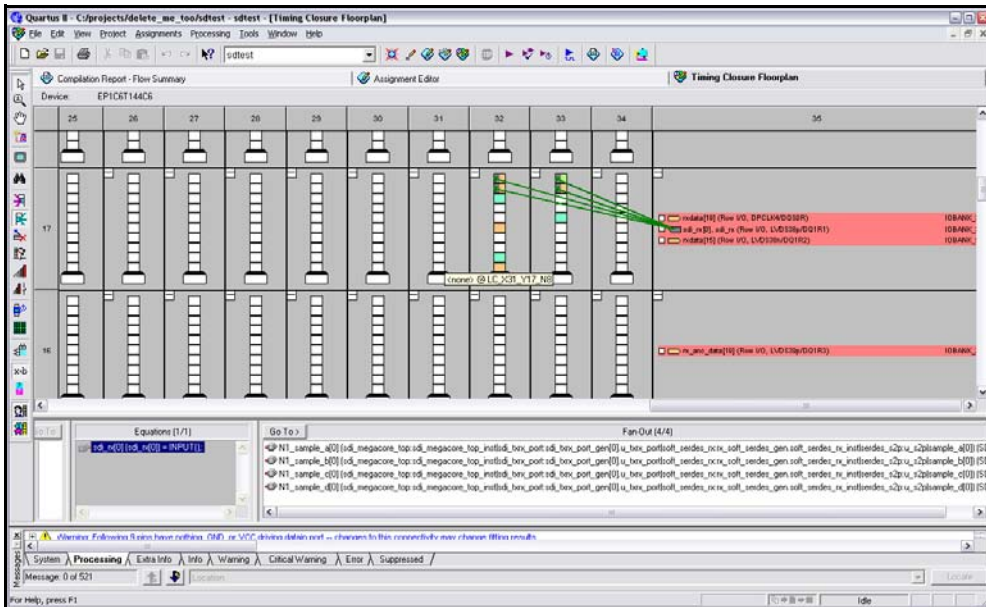
set_location_assignment LC_X33_Y17_N0 -to
"sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[0]
.u_txrx_port|soft_serdes_rx:rx_soft_serdes_gen.soft_serdes_rx_inst|serdes_s2p:u_s2p|sample_b[0]"

set_location_assignment LC_X32_Y17_N1 -to
"sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[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_gen[0]
.u_txrx_port|soft_serdes_rx:rx_soft_serdes_gen.soft_serdes_rx_inst|serdes_s2p:u_s2p|sample_d[0]"
```

Figure A–1 shows the placement of these registers in the Quartus II timing closure floorplan.

Figure A–1. Register Placement



Constraints for the SDI Soft Transceiver

There are different constraints for Cyclone® and non-Cyclone devices and for the Classic Timing Analyzer and the TimeQuest Analyzer.

Non-Cyclone Devices

These constraints apply to all device families (excluding Cyclone) 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:altpll_comp
onent|_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:altpll_comp
onent|_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:altpll_comp
onent|_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:altpll_comp
onent|_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:altpll_comp
onent|_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:altpll_comp
onent|_clk2}

set_min_delay 0 -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:altpll_compo
nent|_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:altpll_comp
onent|_clk2}
```

Cyclone Devices Only

These constraints apply to Cyclone designs only.

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_sdi_
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_sdi_
clocks|pll_sclk:cyc_rx_pll_gen.u_rx_pll|altpll:altpll_component|_clk0" -
to
"<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_sdi_
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_sdi_
clocks|pll_sclk:cyc_rx_pll_gen.u_rx_pll|altpll:altpll_component|_clk0" -
to
"<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_sdi_
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_sdi_c
locks|clkdiv_2p5:cyc_rx_pll_gen.u_clkdiv|clkdiv \

    -source
<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_sdi_c
locks|pll_sclk:cyc_rx_pll_gen.u_rx_pll|altpll:altpll_component|_clk0 \

    -multiply_by 2 \

    -divide_by 5

set_max_delay -from [get_clocks
{<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_sdi_
clocks|pll_sclk:cyc_rx_pll_gen.u_rx_pll|altpll:altpll_component|_clk0}] -
to [get_clocks
{<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_sdi_
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_sdi_
clocks|pll_sclk:cyc_rx_pll_gen.u_rx_pll|altpll:altpll_component|_clk0}] -
to [get_clocks
{<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_clocks:u_sdi_
clocks|clkdiv_2p5:cyc_rx_pll_gen.u_clkdiv|clkdiv}] 0.000

```

Multicycle Paths in SDI Format Block

In some device families and speed grades, there may be timing violations in the format block of the SDI MegaCore function. For SD-SDI, these paths 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.

Classic Timing Analyzer

Use the following constraints for the Classic Timing Analyzer:

```

set_instance_assignment -name MULTICYCLE 2 -from
"<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_ins
t|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|sdi_format:u_format|*" -
to
"<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_ins
t|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|sdi_format:u_format|*"

```

TimeQuest Timing Analyzer

Use the following constraints for the TimeQuest Timing Analyzer:

```

set_multicycle_path -setup 2 -from
{<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|sdi_format:u_format|*} -to
{<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|sdi_format:u_format|*}

set_multicycle_path -hold 1 -from
{<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|sdi_format:u_format|*} -to
{<your_megacore:your_megacore_inst>|sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|sdi_format:u_format|*}

```

False Paths in Stratix GX Devices

The SDI MegaCore function may show timing violations in slower speed grade Stratix® GX devices. These paths are not required to have fast timing, so you can use the following constraints to remove these timing paths:

Classic Timing Analyzer

Use the following constraints for the Classic Timing Analyzer:

```

set_instance_assignment -name CUT ON -from
"<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|sdi_gxb_interface:gen_u_gxb_if.u_gxb_if|sdi_gxb_ctrl:altgxb_ctrl.u_gxb_ctrl|t_rst_sdi" -to
"<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|*"

```

TimeQuest Timing Analyzer

Use the following constraints for the TimeQuest Timing Analyzer:

```

set_false_path -from [get_keepers
{<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|sdi_gxb_interface:gen_u_gxb_if.u_gxb_if|sdi_gxb_ctrl:altgxb_ctrl.u_gxb_ctrl|t_rst_sdi}] -to [get_clocks
{<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|*}]

set_false_path -from [get_keepers
{<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|sdi_format:u_format|trs_locked}] -to
[get_keepers
{<your_megacore>|sdi_megacore_top:sdi_megacore_top_inst|sdi_txrx_port:sdi_txrx_port_gen[0].u_txrx_port|sdi_gxb_interface:gen_u_gxb_if.u_gxb_if|sdi_gxb_ctrl:altgxb_ctrl.u_gxb_ctrl|sdi_locked_d1}]

```

Figure B-1 shows the transceiver clocks for the SDI MegaCore® function for version 7.0 and earlier.

Figure B-1. Version 7.0 and Earlier Clocks

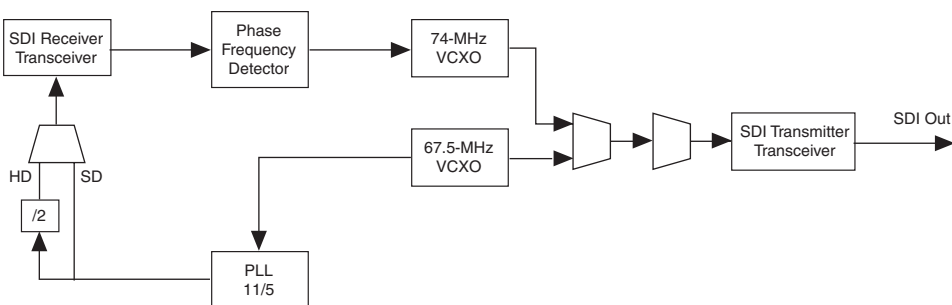


Figure B-2 shows how you should clock the transceivers for current SDI cores. You can now derive all clocks from a single 148.5-MHz voltage controlled crystal oscillator (VCXO) and the transceivers require no external multiplexing.

Figure B-2. Version 7.1 Clocks

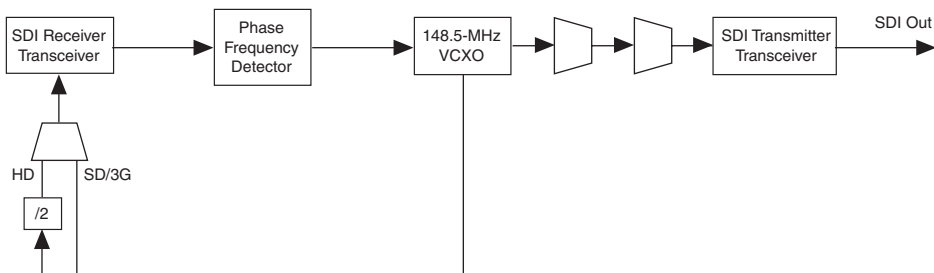
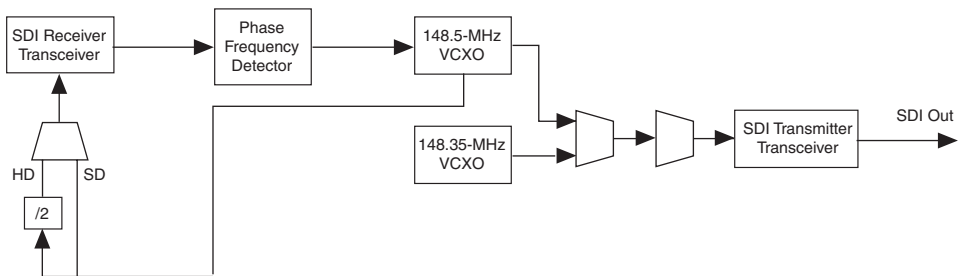


Figure B-3 shows how you should clock the transceivers for version 7.1 SDI cores with international clocking. Both American and European standards are catered for.

Figure B–3. Version 7.1 Internal Clocks





Additional Information

Revision History

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

| Date | Version | Changes Made |
|---------------|---------|--|
| October 2007 | 7.2 | <ul style="list-style-type: none">• Updated device support• Updated standards support—3Gbps SDI now supports <i>SMPTE425M-B 2006 3Gb/s Signal/Data Serial Interface – Source Image Format Mapping</i>• Changed rx_std signal description• Added tx_data_valid_a_bn signal |
| May 2007 | 7.1 | <ul style="list-style-type: none">• 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, see the following table.








| Contact (1) | Contact Method | Address |
|---------------------------------|----------------|--|
| Technical support | Website | www.altera.com/support |
| Technical training | Website | www.altera.com/training |
| | Email | custrain@altera.com |
| Product literature | Website | www.altera.com/literature |
| 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 | Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: Save As dialog box. |
| bold type | External timing parameters, directory names, project names, disk drive names, file names, file name extensions, and software utility names are shown in bold type. Examples: f_{MAX} , lqdesigns directory, d: drive, chiptrip.gdf file. |
| <i>Italic Type with Initial Capital Letters</i> | Document titles are shown in italic type with initial capital letters. Example: <i>AN 75: High-Speed Board Design</i> . |
| <i>Italic type</i> | Internal timing parameters and variables are shown in italic type. Examples: <i>t_{PIA}</i> , <i>n + 1</i> . Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <file name>, <project name>.pdf file. |
| Initial Capital Letters | Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu. |
| “Subheading Title” | References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: “Typographic Conventions.” |
| Courier type | Signal and port names are shown in lowercase Courier type. Examples: data1, tdi, input. Active-low signals are denoted by suffix n, e.g., resetn. Anything that must be typed exactly as it appears is shown in Courier type. For example: c:\qdesigns\tutorial\chiptrip.gdf. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword SUBDESIGN), as well as logic function names (e.g., TRI) are shown in Courier. |
| 1., 2., 3., and a., b., c., etc. | Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure. |
|  | Bullets are used in a list of items when the sequence of the items is not important. |
|  | The checkmark indicates a procedure that consists of one step only. |
|  | 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 the user's work. |
|  | A warning calls attention to a condition or possible situation that can cause injury to the user. |
|  | The angled arrow indicates you should press the Enter key. |
|  | The feet direct you to more information on a particular topic. |