ASI MegaCore Function

User Guide

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

MegaCore Version: 7.1
Document Version: 7.1
Document Date: May 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.

I.S. EN ISO 9001

UG-ASI0106-4.0

Contents

About This User Guide

Revision History

The table below displays the revision history for the chapters in this user guide.

Date	Version	Changes Made
May 2007	7.1	● Updated device support ● Added packet synchronization information
December 2006	7.0	Added support for Cyclone® III devices.
December 2006	6.1	● Updated for new MegaWizard® Plug-In Manager ● Added extra files to generation table
April 2006	1.0.0	First published.

How to Contact Altera

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

Information Type	Contact *Note (1)*
Technical support	www.altera.com/mysupport/
Technical training	www.altera.com/training/
Technical training services	custrain@altera.com
Product literature	www.altera.com/literature
Product literature services	literature@altera.com
FTP site	ftp.altera.com

Note to table:
(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

This document uses the typographic conventions shown below.

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, filenames, filename extensions, and software utility names are shown in bold type. Examples: $\mathbf{f_{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.*
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>*.**pof** 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.
⚠ CAUTION	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.
⚠ WARNING	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.

1. About This MegaCore Function

Release Information

Table 1–1 provides information about this release of the Altera® ASI MegaCore® function.

Table 1–1. Release Information

Item	Description
Version	7.1
Release Date	May 2007
Ordering Code	IP-ASI
Product ID	00B9
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 ASI 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	Full
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.

Features

- IP functional simulation models for use in Altera-supported VHDL and Verilog HDL simulators
- Easy-to-use MegaWizard® interface
- Support for OpenCore Plus evaluation

General Description

The ASI MegaCore function implements a receiver or transmitter digital video broadcast asynchronous serial interface (DVB-ASI) that transports MPEG-2 packets over copper-based cables or optical networks. DVB-ASI is used as a serial link between equipment in broadcast facilities.

The ASI MegaCore function demonstrates how to transmit or receive packets over an ASI. The ASI MegaCore function works with 270 megabits per second (Mbps) DVB-ASI, as defined by the DVB-ASI specification *EN 50083-9 from CENELEC / December 2002 "Cable networks for television signals, sound signals and interactive services. Part 9: Interfaces for CATV/SMATV head-ends and similar professional equipment for DVB/MPEG2 transport streams."*

For information on ASI MegaCore function demonstration on the Altera Cyclone Video Demonstration Board, refer to the *Cyclone Video Demonstration Board Data Sheet*.

OpenCore Plus Evaluation

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

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

- Program a device and verify your design in hardware

You only need to obtain a license for the megafunction 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 ASI, see "OpenCore Plus Time-Out Behavior" on page 3–6 and *AN 320: OpenCore Plus Evaluation of Megafunctions*.

Performance

Table 1–3 shows typical expected resource usage for the ASI MegaCore function, with the Quartus® II software version 7.1.

Table 1–3. Resource Usage

Device	Parameters	LEs	Combinational ALUTs	Logic Registers
Cyclone II	Receiver	538	–	–
	Transmitter	115	–	–
Cyclone III	Receiver	517	–	–
	Transmitter	115	–	–
Stratix II	Receiver	–	294	239
	Transmitter	–	64	64
Stratix III	Receiver	–	299	238
	Transmitter	–	64	64
Stratix II GX	Receiver	–	293	187
	Transmitter	–	85	76

2. Getting Started

Design Flow

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

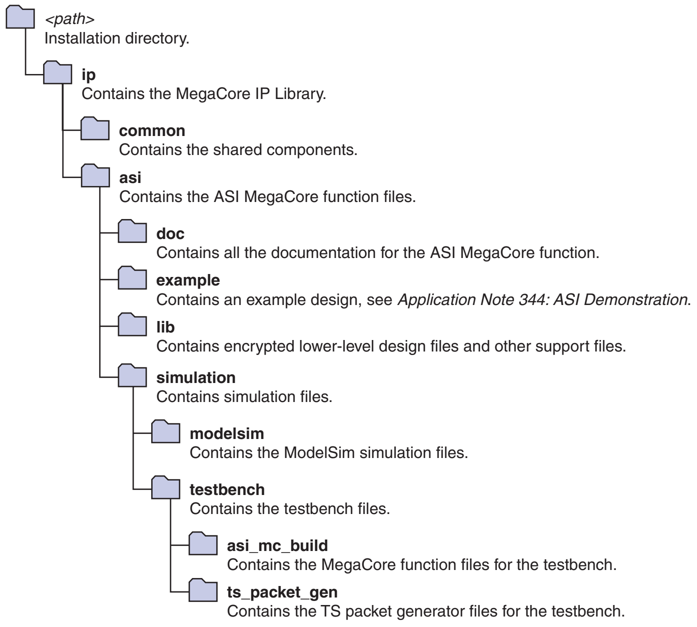
1. Obtain and install the ASI MegaCore function.

The ASI 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 ASI MegaCore function, where <*path*> is the installation directory. The default installation directory on Windows is **c:\altera\71**; on UNIX and Solaris it is **/opt/altera/71**.

Figure 2–1. Directory Structure



<path>
Installation directory.
ip
Contains the MegaCore IP Library.
common
Contains the shared components.
asi
Contains the ASI MegaCore function files.
doc
Contains all the documentation for the ASI MegaCore function.
example
Contains an example design, see *Application Note 344: ASI Demonstration*.
lib
Contains encrypted lower-level design files and other support files.
simulation
Contains simulation files.
modelsim
Contains the ModelSim simulation files.
testbench
Contains the testbench files.
asi_mc_build
Contains the MegaCore function files for the testbench.
ts_packet_gen
Contains the TS packet generator files for the testbench.

2. Create a custom variation of the ASI 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 also can 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 ASI MegaCore function.

Once you have purchased a license, 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.

ASI Walkthrough

This walkthrough explains how to create an asynchronous serial interface (ASI) using the the MegaWizard Plug-In Manager and the Quartus II software on a PC. When you are finished generating a custom variation of the ASI MegaCore function, you can incorporate it into your overall project.

This walkthrough requires the following steps:

- "Create a New Quartus II Project" on page 2–3
- "Launch the MegaWizard Plug-In Manager" on page 2–4
- "Parameterize" on page 2–7
- "Set Up Simulation" on page 2–8
- "Generate" on page 2–9

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\asi_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 ASI MegaCore function.

   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.

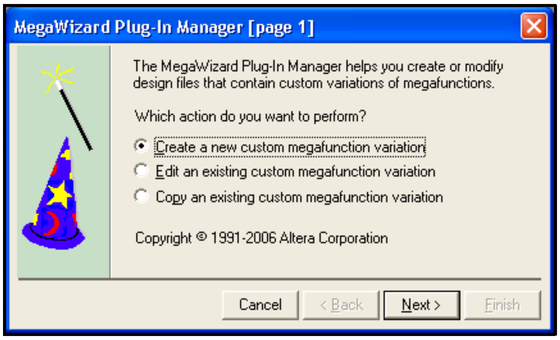
Launch the MegaWizard Plug-In Manager

To launch the 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



The MegaWizard Plug-In Manager helps you create or modify design files that contain custom variations of megafunctions.

Which action do you want to perform?

- (●) Create a new custom megafunction variation
- ( ) Edit an existing custom megafunction variation
- ( ) Copy an existing custom megafunction variation

Copyright © 1991-2006 Altera Corporation

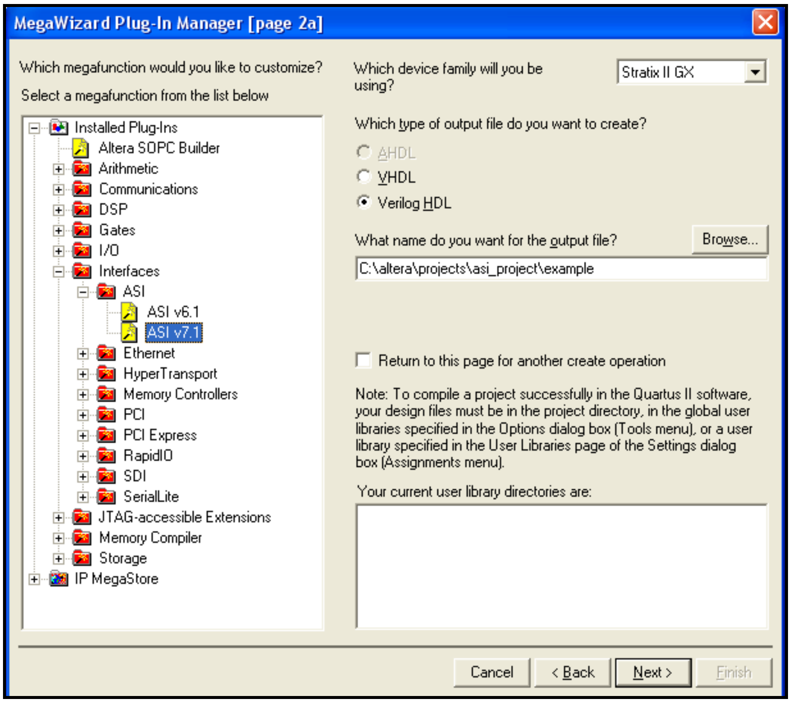
2. Specify that you want to create a new custom megafunction variation and click **Next**.

3. Expand the **Interfaces > ASI** directory then click **ASI v6.1**.

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



Which device family will you be using? Stratix II GX

Which type of output file do you want to create?

AHDL
VHDL
[✓ Verilog HDL]

What name do you want for the output file? C:\altera\projects\asi_project\example

Return to this page for another create operation

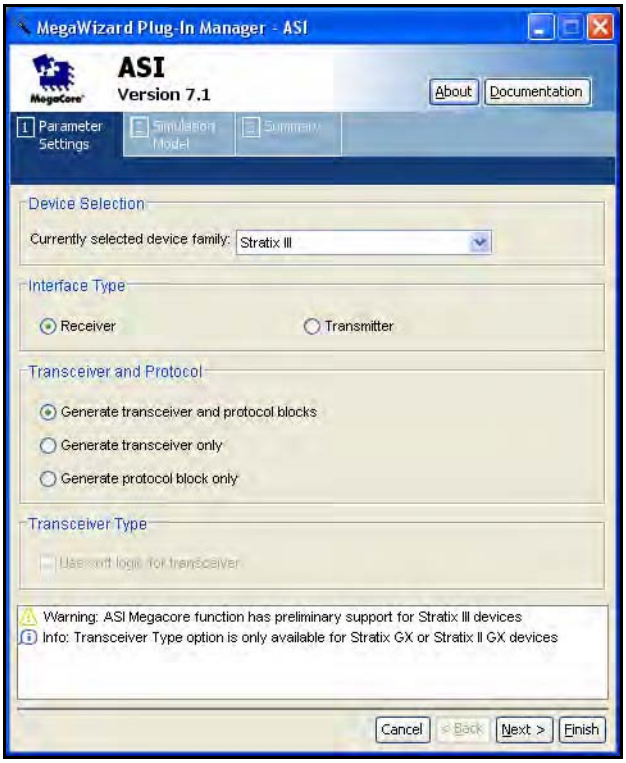
Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in the global user libraries specified in the Options dialog box (Tools menu), or a user library specified in the User Libraries page of the Settings dialog box (Assignments menu).

Your current user library directories are:

6. Click **Next** to display the **Parameter Settings** page for the ASI 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.

Figure 2–4. Parameter Settings



Device Selection

Currently selected device family: Stratix III

Interface Type

[✓ Receiver] Transmitter

Transceiver and Protocol

[✓ Generate transceiver and protocol blocks]
Generate transceiver only
Generate protocol block only

Transceiver Type

Use soft logic for transceiver

Warning: ASI Megacore function has preliminary support for Stratix III devices
Info: Transceiver Type option is only available for Stratix GX or Stratix II GX devices

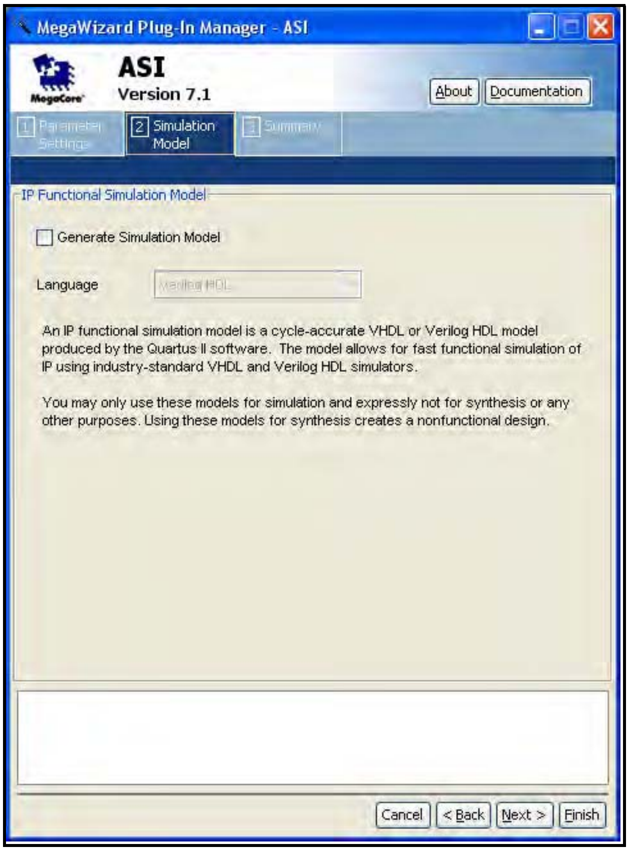
Parameterize

To parameterize your MegaCore function, follow these steps:

For more information on the parameters, see Table 3–2 on page 3–8.

1. Select interface type.
2. Select the transceiver and protocol.
3. Turn on **Use soft logic transceiver** to implement the transceiver in logic, rather than using Stratix II GX, or Stratix GX transceivers.
4. Click **Next** (or the **Simulation Model** tab) to display the simulation setup page (see Figure 2–5).

Figure 2–5. Simulation Model



IP Functional Simulation Model

☐ Generate Simulation Model

Language: Verilog HDL

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.

Set Up Simulation

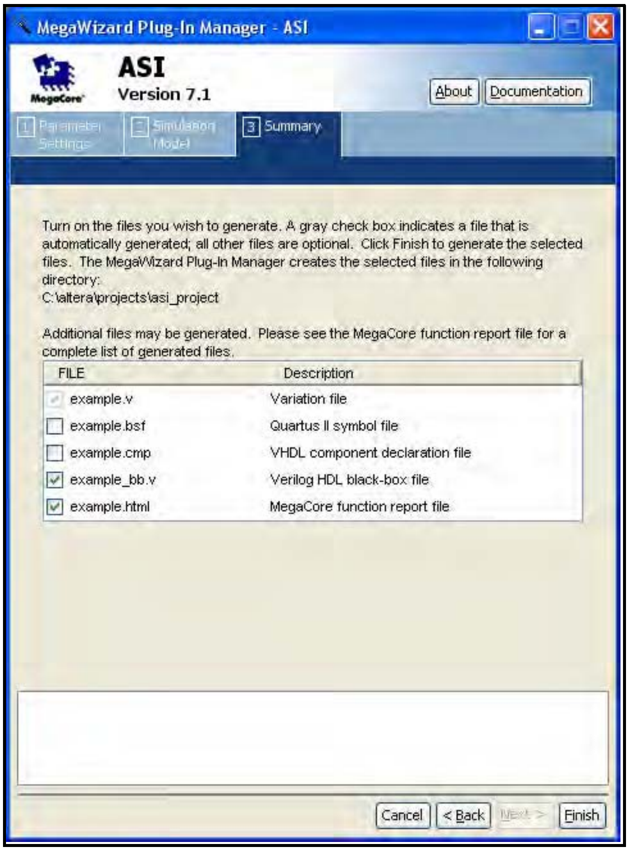
An IP functional simulation model is a cycle-accurate VHDL or Verilog HDL model produced by the Quartus II software. The model allows for fast functional simulation of IP using industry-standard VHDL and Verilog HDL simulators.

CAUTION 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. Choose the language from the **Language** list.
3. Click **Next** (or the **Summary** tab) to display the summary page (see Figure 2–6).

Figure 2–6. IP Toolbench—Generate



Turn on the files you wish to generate. A gray check box indicates a file that is automatically generated; all other files are optional. Click Finish to generate the selected files. The MegaWizard Plug-In Manager creates the selected files in the following directory:
C:\altera\projects\asi_project

Additional files may be generated. Please see the MegaCore function report file for a complete list of generated files.

FILE	Description
✓ example.v	Variation file
☐ example.bsf	Quartus II symbol file
☐ example.cmp	VHDL component declaration file
✓ example_bb.v	Verilog HDL black-box file
✓ example.html	MegaCore function report file

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 (Part 1 of 2) *Notes (1)* ***& 2***

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

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

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

Notes to Table 2–1:
(1) *<variation name>* is the variation name.
(2) *<device name>* is the device family name.

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 2–8, 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 a fixed testbench as an example in the **simulation\testbench\** directory. The testbench instantiates the design and tests the ASI operation. To use the testbench with the ModelSim® simulator, follow these steps:

1. In a text editor open the simulation batch file, **simulation\modelsim\asi_run.bat**, and edit it to point to your installation of the ModelSim-Altera simulator.

2. Start the ModelSim-Altera simulator.

3. Run **asi_run.bat** in the **simulation\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.

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 provides a Quartus II project for use with NativeLink in the **ip\asi\simulation** 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 **ip\asi\simulation** directory.

2. Open **asi_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_asi_mc`, 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_asi_mc**, 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 your design, program your targeted Altera device and verify your design in hardware.

With Altera's free OpenCore Plus evaluation feature, you can evaluate the ASI MegaCore function before you purchase 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 IP functional simulation models, refer to the *Simulating Altera IP in Third-Party Simulation Tools* chapter in volume 3 of the *Quartus II Handbook*.

You can simulate the ASI in your design and perform a time-limited evaluation of your design in hardware.

For more information on OpenCore Plus hardware evaluation using the ASI MegaCore function, see "OpenCore Plus Evaluation" on page 1–2, "OpenCore Plus Time-Out Behavior" on page 3–6, and *AN 320: OpenCore Plus Evaluation of Megafunctions*.

Set Up Licensing

You need to obtain 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 obtain a license for ASI, you can request a license file from the Altera web site at **www.altera.com/licensing** and install it on your computer. When you request a license file, Altera emails you a **license.dat** file. If you do not have Internet access, contact your local Altera representative.

3. Specifications

Functional Description

The ASI MegaCore® function consists of the following elements:

- Low voltage differential signalling (LVDS) inputs and outputs (I/Os) for the receiver and transmitter
- Aynchronous serial interface (ASI) transmitter
- ASI receiver
- Two PLLs for frequency multiplication—one for the transmitter, one for the receiver

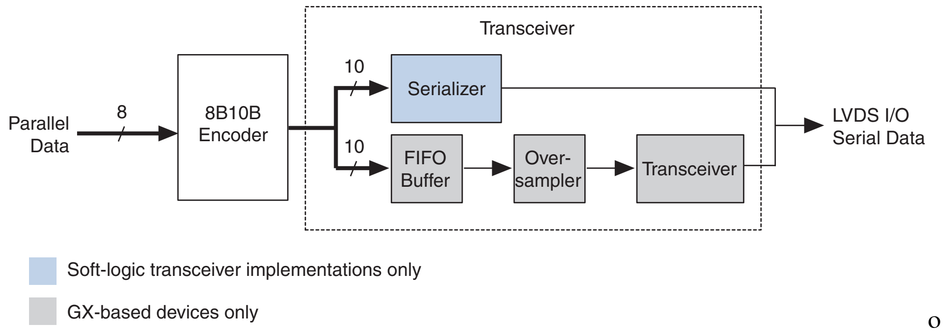
Transmitter

The transmitter comprises the following elements:

- 8B10B encoder
- Serializer

Figure 3–1 shows the ASI transmitter.

Figure 3–1. ASI Transmitter



8B10B Encoder

The 8B10B encoder converts an 8-bit wide word to a 10-bit wide word. The complete list of codes can be found in the *DVB-ASI EN50083-9* standard.

A control code input inserts comma characters (K28.5) when no data is available at the input of the encoder.

Transceiver

The transceiver can be either a serializer for soft-logic implementations or GX transceivers.

Serializer

The serializer converts a 10-bit parallel word 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. You should use a PLL that multiplies a 27-MHz reference clock by ten to provide the bit-rate clock and enables jitter-controlled ASI transmit serialization.

GX Transceivers

For GX-based devices, in the MegaWizard Plug-In Manager you can select either a soft-logic transceiver or a GX transceiver. If you are using GX transceivers, the transmitter has a FIFO buffer, oversampler, and a transceiver, which replace the soft-logic serializer.

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

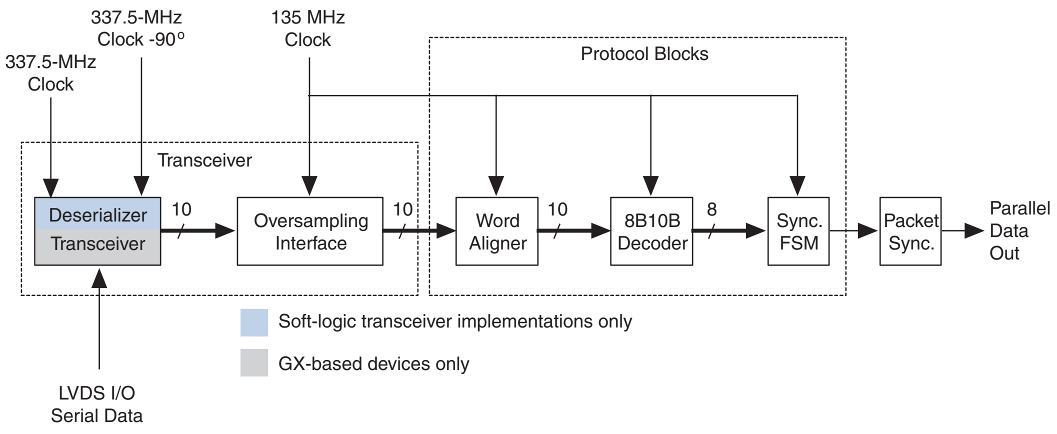
Receiver

The receiver comprises the following elements:

- Deserializer
- Oversampling Interface
- Word Aligner
- 8B10B Decoder
- Synchronization State Machine

Figure 3–2 shows the ASI receiver.

Figure 3–2. ASI Receiver



Transceiver

The transceiver can be either a deserializer for soft-logic implementations or a GX transceiver.

Deserializer

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 clocks are created by inversion of the PLL clock outputs and should be 337.5-MHz clocks.

Samples are then all converted to the same clock domain and de-serialized into a 10-bit parallel word. The serial clock that samples the bit stream has to be 5/4 of the incoming bit (i.e., 270-bit rate × 5/4 × 4 sample per clock = 1350 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.

For GX-based devices, you can optionally perform the deserialization in a transceiver.

GX Transceiver
For GX-based devices, in the MegaWizard Plug-In Manager you can select either a soft-logic transceiver or a GX transceiver. If you are using GX transceivers they replace the soft-logic deserializer.

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

Oversampling Interface

A 5× over-sampling scheme implements data recovery and bit synchronization, which corresponds to a sampling rate of 1350 Mbps.

The deserializer provides a fixed frequency sampling of the serial data. Approximately 5 samples are taken for each bit. These samples are accumulated by the deserializer and passed to the over-sampling interface in a parallel format. Logic extracts the data from the sets of samples generated by the deserializer.

Firstly, the transition points within the received word are determined. The ASI receiver uses these transition points to determine the best sample to extract for each data bit. The logic continuously realigns to the transition points in the incoming data, and can adapt to a frequency mismatch between the sampling clock and the incoming data rate. The extracted samples for each data bit are accumulated into a parallel word for processing by the rest of the ASI receiver.

Word Aligner

The word aligner is consistently looking for two consecutive comma characters (K28.5) in the parallel data stream coming out of the over-sampling interface. The word-aligner computes the matching position and shifts words accordingly.

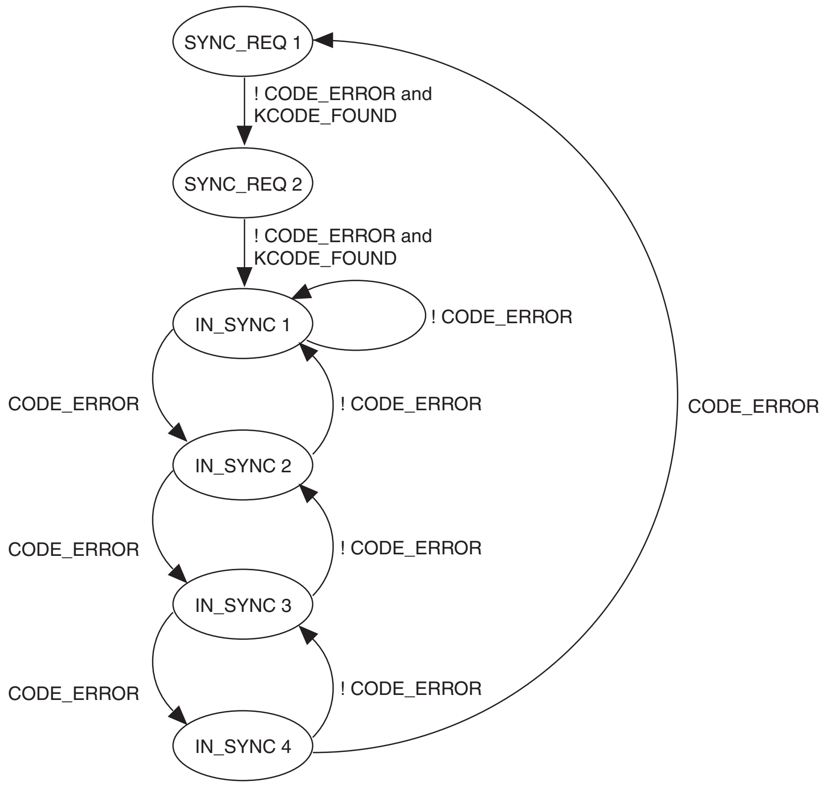
8B10B Decoder

The 8B10B decoder converts 10-bit wide parallel data from 8B10B codes into 8-bit wide raw data. The decoder detects special characters, code errors (unused codes), and disparity errors and signals their presence with various flags.

Synchronization State Machine

Two consecutive comma characters without any disparity or code error enables word synchronization. Four consecutive disparity or code errors enables loss of synchronization and so disable the word synchronization flag. The word synchronization flag gates the rate matching FIFO write request. Figure 3–3 shows the synchronization state machine.

Figure 3–3. Synchronization State Machine



Packet Synchronization

The packet synchronization block looks for the presence of valid TS packets. Valid packets have either 188 bytes or 204 bytes between synchronization bytes. The synchronization byte takes the value `0x47`.

The block first looks for the synchronization byte that indicates the start of the packet, which is indicated by `rx_ts_status[1]`. The block then counts valid bytes in the incoming stream. If a synchronization byte is seen 188 or 204 bytes after the first sync byte is seen, lock is indicated on `rx_ts_[5:4]` and end of packet is indicated on `rx_ts_status[2]`. If no synchronization byte is seen at either 188 or 204 bytes, the packet is deemed to have an error and `rx_ts_status[3]` is asserted. The block then again starts the search for synchronization bytes.

Testbench

The testbench instantiates two ASI MegaCore functions—one ASI transmitter, one ASI receiver.

To test a realistic ASI link, an ASI packet generator creates packets that are sent from the instantiation of the ASI transmitter to the instantiation of the ASI receiver. A random serial data delay generator is inserted on the way to mimic random jitter on the link. The transmitter and receiver are clocked with asynchronous clock sources—the frequencies differ by 200 ppm, which maximizes the stress that the ASI receiver sees and is similar to a real link.

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 1–2 and *AN 320: OpenCore Plus Evaluation of Megafunctions*.

Signals

Table 3–1 shows the signals.

Table 3–1. Signals

Signal	Direction	Description
`asi_rx`	Input	ASI input.
`cal_blk_clk`	Input	Calibration clock for Stratix II GX transceiver.
`rst`	Input	Reset.
`rx_clk135`	Input	135-MHz clock from external PLL.
`rx_protocol_in[9:0]`	Input	Protocol input (for split SERDES/protocol).
`rx_protocol_in_valid`	Input	Valid signal for `rx_protocol_in`.
`rx_serial_clk`	Input	337.5-MHz clock from external PLL.
`rx_serial_clk90`	Input	337.5-MHz clock from external PLL with + 90° phase shift.
`tx_clk270`	Input	270-MHz clock from external PLL.
`tx_clk135`	Input	135-MHz clock from external PLL (only for hard SERDES).
`tx_data[7:0]`	Input	TS parallel data input into encoder.
`tx_en`	Input	Transmit enable. Assert to indicate valid data on `tx_data`.
`tx_refclk`	Input	27-MHz reference clock for transmitter.
`tx_serdes_in[9:0]`	Input	Direct input to transceiver block for split protocol/tranceiver mode.
`asi_tx`	Output	ASI output.
`rx_data[7:0]`	Output	Decoded parallel TS data out of receiver.
`rx_data_clk`	Output	135-MHz parallel clock, which you can use to clock `rx_data[7:0]`.
`rx_serdes_out[9:0]`	Output	Raw data from transceiver block before decoding.
`rx_serdes_out_valid`	Output	Valid signal out of the transceiver.
`rx_ts_status[7:0]`	Output	TS status bits. 0 indicates receiver data valid 1 indicates start of packet 2 indicates end of packet 3 indicates receiver error 5:4 indicates 00 is unlocked; 01 is 204 byte packet lock, 11 is 188 byte packet lock 6 indicates TS serial polarity 7 indicates that the data on `rx_data[7:0]` is a valid word from the 8B10B decoder. Unlike `rx_ts_status[0]`, this signal is not dependent on the correct packet or synchronization structure of the stream.
`tx_protocol_out[9:0]`	Output	Output from transmitter protocol block for split protocol/transceiver mode.

Parameters

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

Table 3–2 shows the parameters.

Table 3–2. Parameters

Parameter	Range	Description
Currently selected device family	–	Shows the device family that you chose in your Quartus II project.
Interface type	Receiver or transmitter	Select a receiver or transmitter for you custom variation.
Transceiver and protocol	Generate transceiver and protocol, or generate transceiver only, or generate protocol blocks only	Select the blocks for your custom variation.
Use soft logic for transceiver	On or off	For Stratix II GX and Stratix GX devices, specify soft logic for the transceiver. When you turn on **Use soft logic for transceiver**, the transceiver is implemented in the device's logic, otherwise the design uses a transceiver.

MegaCore Verification

The ASI MegaCore verification involves the testing of the DVB-ASI specification *EN 50083-9 from CENELEC / December 2002 "Cable networks for television signals, sound signals and interactive services. Part 9: Interfaces for CATV/SMATV head-ends and similar professional equipment for DVB/MPEG2 transport streams."*

Appendix A. Constraints

Introduction

For the ASI 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

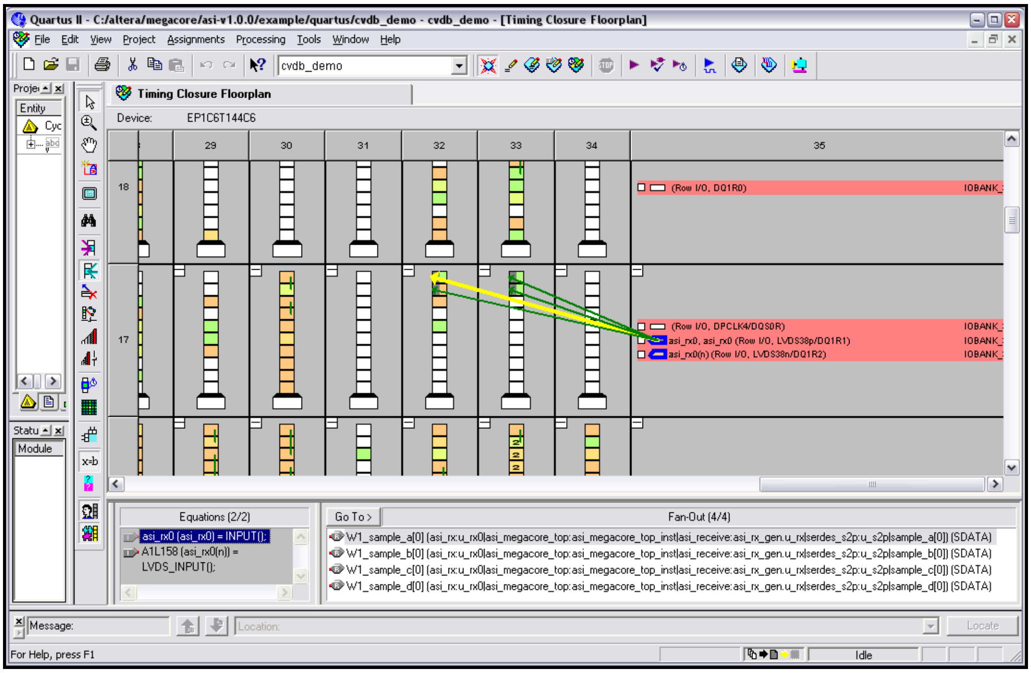
Minimize Timing Skew

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 asi_rx0

set_location_assignment LC_X32_Y17_N0 -to
"asi_rx:u_rx0|asi_megacore_top:asi_megacore_top_inst|asi_receive:asi_rx_g
en.u_rx|serdes_s2p:u_s2p|sample_a[0]"

set_location_assignment LC_X33_Y17_N0 -to
"asi_rx:u_rx0|asi_megacore_top:asi_megacore_top_inst|asi_receive:asi_rx_g
en.u_rx|serdes_s2p:u_s2p|sample_b[0]"

set_location_assignment LC_X32_Y17_N1 -to
"asi_rx:u_rx0|asi_megacore_top:asi_megacore_top_inst|asi_receive:asi_rx_g
en.u_rx|serdes_s2p:u_s2p|sample_c[0]"

set_location_assignment LC_X33_Y17_N1 -to
"asi_rx:u_rx0|asi_megacore_top:asi_megacore_top_inst|asi_receive:asi_rx_g
en.u_rx|serdes_s2p:u_s2p|sample_d[0]"
```

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


Quartus II - C:/altera/megacore/asi-v1.0.0/example/quartus/cvdb_demo - cvdb_demo - [Timing Closure Floorplan]
File Edit View Project Assignments Processing Tools Window Help
cvdb_demo
Timing Closure Floorplan
Device: EP1C6T144C6
29 30 31 32 33 34 35
18
17
(Row I/O, DQ1R0) IOBANK_
(Row I/O, DPCLK4/DQS0R) IOBANK_
asi_rx0, asi_rx0 (Row I/O, LVDS38p/DQ1R1) IOBANK_
asi_rx0(n) (Row I/O, LVDS38n/DQ1R2) IOBANK_
Message:
Location:
Locate
For Help, press F1
Idle

Equations (2/2)

asi_rx0 (asi_rx0) = INPUT();
A1L158 (asi_rx0(n)) = LVDS_INPUT();

Go To > Fan-Out (4/4)

W1_sample_a[0] (asi_rx:u_rx0|asi_megacore_top:asi_megacore_top_inst|asi_receive:asi_rx_gen.u_rx|serdes_s2p:u_s2p|sample_a[0]) (SDATA)
W1_sample_b[0] (asi_rx:u_rx0|asi_megacore_top:asi_megacore_top_inst|asi_receive:asi_rx_gen.u_rx|serdes_s2p:u_s2p|sample_b[0]) (SDATA)
W1_sample_c[0] (asi_rx:u_rx0|asi_megacore_top:asi_megacore_top_inst|asi_receive:asi_rx_gen.u_rx|serdes_s2p:u_s2p|sample_c[0]) (SDATA)
W1_sample_d[0] (asi_rx:u_rx0|asi_megacore_top:asi_megacore_top_inst|asi_receive:asi_rx_gen.u_rx|serdes_s2p:u_s2p|sample_d[0]) (SDATA)

Figure A–1. Register Placement

Constraints For ASI Receivers

You must add constraints to receiver ASI MegaCore functions. 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.

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

Modify the following constraints and apply them to your design. Alternatively, apply similar constraints to the clocks connected to the `rx_serial_clk` and `rx_clk135` signals on your ASI MegaCore function.

Classic Timing Analyzer

Use the following constraints for the Classic Timing Analyzer:

```
set_instance_assignment -name SETUP_RELATIONSHIP "4.43 ns" -from
"u_rx_pll|c0" -to "u_rx_pll|c2"
```

Where `c0` is a 337.5-MHz PLL output and `c2` is the 135-MHz PLL output.

```
set_instance_assignment -name HOLD_RELATIONSHIP "0 ns" -from "u_rx_pll|c0"
-to "u_rx_pll|c2"
```

TimeQuest Timing Analyzer

Use the following constraints for the TimeQuest Timing Analyzer:

```
set_max_delay 4.43 -from {u_rx_pll|altpll:altpll_component|_clk0} -to
{u_rx_pll|altpll:altpll_component|_clk2}

set_min_delay 0 -from {u_rx_pll|altpll:altpll_component|_clk0} -to
{u_rx_pll|altpll:altpll_component|_clk2}
```

Cyclone Devices Only

These constraints apply to Cyclone device families.

Classic Timing Analyzer

Use the following constraints for the Classic Timing Analyzer:

```
set_instance_assignment -name CLOCK_SETTINGS input_refclk -to rx_refclk

set_global_assignment -name FMAX_REQUIREMENT "27 MHz" -section_id
input_refclk

set_instance_assignment -name CLOCK_SETTINGS rxclk -to "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 HOLD_RELATIONSHIP "0 ns" -from
"u_rx_pll|altpll:altpll_component|_clk0" -to "u_clkdiv|clkdiv"

set_instance_assignment -name SETUP_RELATIONSHIP "4.43 ns" -from
"u_rx_pll|altpll:altpll_component|_clk0" -to "u_clkdiv|clkdiv"
```

Timequest Timing Analyzer

Use the following constraints for the TimeQuest Timing Analyzer:

```
derive_pll_clocks -use_tan_name

create_clock -period "27 MHz"  -name {rx_refclk} {rx_refclk}

create_generated_clock -divide_by 5 -multiply_by 2  \

                       -source u_rx_pll|altpll:altpll_component|_clk0 \

                       -name {u_clkdiv|clkdiv} \

set_max_delay 4.43 -from {u_rx_pll|altpll:altpll_component|_clk0} -to
{u_clkdiv|clkdiv}

set_min_delay 0 -from {u_rx_pll|altpll:altpll_component|_clk0} -to
{u_clkdiv|clkdiv}
```