



ASI MegaCore Function User Guide



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

Software Version: 9.1
Document Date: November 2009

Copyright © 2009 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.

Chapter 1. About This MegaCore Function

Release Information	1-1
Device Family Support	1-1
Features	1-2
General Description	1-2
MegaCore Verification	1-3
Resource Utilization	1-3
Installation and Licensing	1-3
OpenCore Plus Evaluation	1-4
OpenCore Plus Time-Out Behavior	1-5

Chapter 2. Getting Started

Design Flow	2-1
Select Flow	2-1
SOPC Builder Flow	2-2
Specify Parameters	2-2
Complete the SOPC Builder System	2-3
Simulate the System	2-3
MegaWizard Plug-In Manager Flow	2-3
Specify Parameters	2-4
Simulate the Design	2-5
Simulate with IP Functional Simulation Models	2-5
Simulate with the ModelSim Simulator	2-6
Simulating in Third-Party Simulation Tools Using NativeLink	2-6
Compile the Design and Program a Device	2-7

Chapter 3. Parameter Settings

Chapter 4. Functional Description

Transmitter	4-1
8B10B Encoder	4-1
Transceiver	4-1
Serializer	4-2
GX Transceivers	4-2
Receiver	4-2
Transceiver	4-3
Deserializer	4-3
GX Transceiver	4-3
Oversampling Interface	4-3
Word Aligner	4-3
8B10B Decoder	4-4
Synchronization State Machine	4-4
Packet Synchronization	4-4
Testbench	4-5
Signals	4-5

Appendix A. Constraints

Introduction	A-1
Constrain Design With TimeQuest Timing Analyzer	A-1
Specify Clock Characteristics	A-2
Define the Setup and Hold Relationship between the 135-MHz Clocks and the 337.5-MHz zero-degree Clocks	A-2
Specify Clocks that are Exclusive or Asynchronous	A-3
Minimize Timing Skew	A-3
Constraints For ASI Receivers	A-4
Non-Cyclone Devices	A-4
Classic Timing Analyzer	A-5
TimeQuest Timing Analyzer	A-5
Cyclone Devices Only	A-5
Classic Timing Analyzer	A-5
TimeQuest Timing Analyzer	A-6

Additional Information


Revision History	Info-1
How to Contact Altera	Info-1
Typographic Conventions	Info-2

Release Information

Table 1–1 provides information about this release of the Altera® Asynchronous Serial Interface (ASI) MegaCore® function.

Table 1–1. Release Information

Item	Description
Version	9.1
Release Date	November 2009
Ordering Code	IP-ASI
Product ID	00B9
Vendor ID	6AF7

 For more information about this release, refer to the [MegaCore IP Library Release Notes and Errata](#).

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

Device Family Support

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

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

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

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

Device Family	Support
Arria® GX	Full
Arria II GX	Preliminary
Cyclone®	Full (1)
Cyclone II	Full
Cyclone III	Full
Cyclone III LS (2)	Preliminary

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

Device Family	Support
Cyclone IV (2)	Preliminary
Stratix® (2)	Full
Stratix II (2)	Full
Stratix II GX	Full
Stratix III (2)	Full
Stratix IV (3)	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) The Cyclone series of devices, and the Stratix I, Stratix II, and Stratix III devices only support soft SERDES.
- (3) Stratix IV GT only supports soft logic mode.

Features

This section summarizes the features of the ASI MegaCore function.

- IP functional simulation models for use in Altera-supported VHDL and Verilog HDL simulators
- Easy-to-use MegaWizard™ interface
- SOPC Builder ready
- 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](#).

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"*.

Resource Utilization

Table 1–3 shows estimated resource usage for the ASI MegaCore function, with the Quartus II software version 9.1.

Table 1–3. Resource Usage

Device Family	Parameters	LEs	Combinational ALUTs	Logic Registers
Arria GX	Receiver		314	191
	Transmitter		70	64
Cyclone	Receiver	573	—	—
	Transmitter	78	—	—
Cyclone II	Receiver	576	—	—
	Transmitter	78	—	—
Cyclone III	Receiver	577	—	—
	Transmitter	78	—	—
Cyclone III LS	Receiver	587	—	—
	Transmitter	78	—	—
Cyclone IV	Receiver	564	—	—
	Transmitter	78	—	—
Stratix	Receiver	577	—	—
	Transmitter	78	—	—
Stratix GX	Receiver	506	—	—
	Transmitter	98	—	—
Stratix II	Receiver	—	323	243
	Transmitter	—	47	49
Stratix II GX	Receiver	—	314	191
	Transmitter	—	70	64
Stratix III	Receiver	—	321	241
	Transmitter	—	47	49
Stratix IV	Receiver	—	328	191
	Transmitter	—	65	62

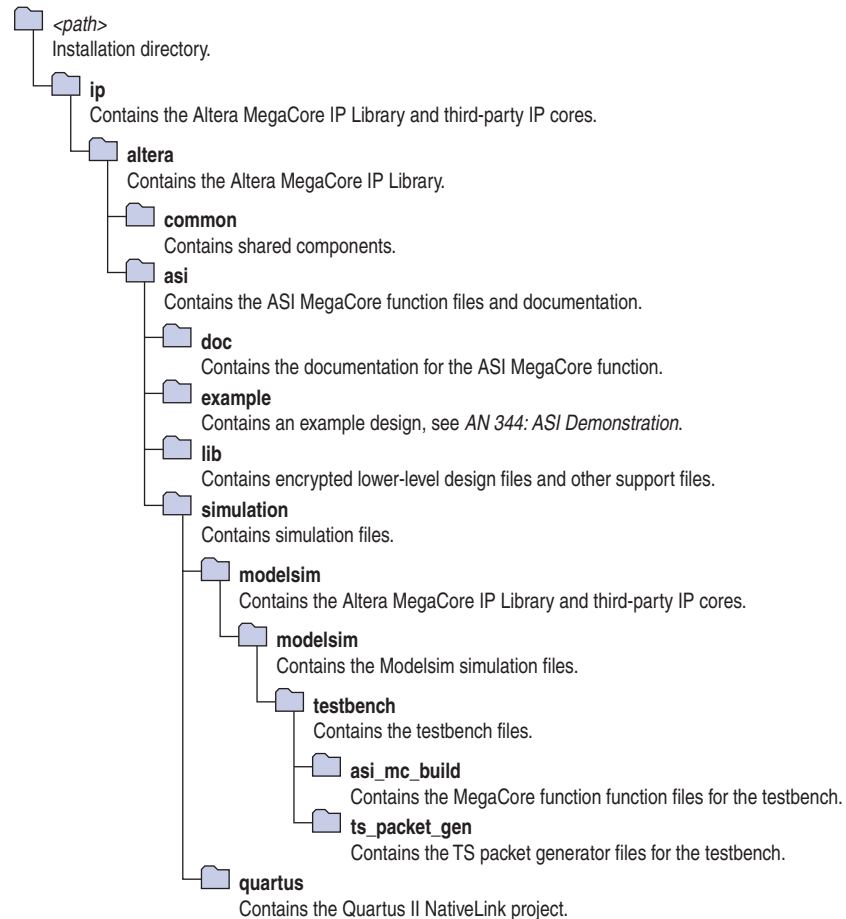
Installation and Licensing

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 *Altera Software Installation & Licensing*.

Figure 1-1 on page 1-4 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\<version>`; on Linux it is `/opt/altera<version>`.

Figure 1-1. Directory Structure



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.

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 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, refer to *AN 320: OpenCore Plus Evaluation of Megafunctions*.

OpenCore Plus Time-Out Behavior

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

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

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

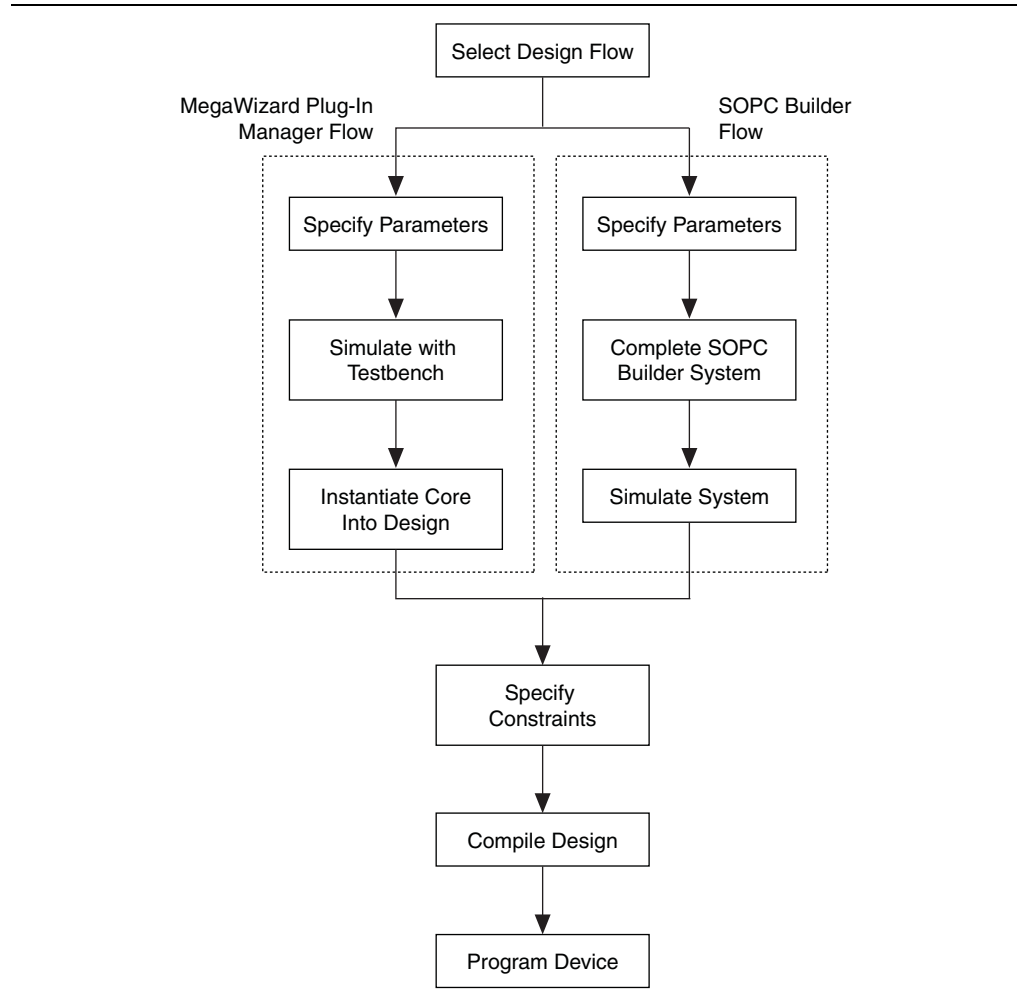
 For MegaCore functions, the untethered time-out is 1 hour; the tethered time-out value is indefinite.

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

Design Flow

Figure 2-1 shows the stages for creating a system with the ASI MegaCore function and the Quartus II software. The sections in this chapter describe each stage.

Figure 2-1. Design Flow



Select Flow

You can parameterize the ASI MegaCore function using either one of the following flows:

- SOPC Builder flow
- MegaWizard Plug-In Manager flow

Table 2-1 summarizes the advantages offered by the different parameterization flows.

Table 2-1. Advantages of the Parameterization Flows

SOPC Builder Flow	MegaWizard Plug-In Manager Flow
<ul style="list-style-type: none"> ■ Rapidly create a new SOPC Builder system design that includes an ASI ■ Use the ASI MegaCore function with other components available in SOPC Builder, for example the Nios II processor, external memory controllers and the scatter/gather DMA controller ■ Use the accompanying Nios II/Interniche TCP/IP protocol stack software driver support in your system 	<ul style="list-style-type: none"> ■ Parameterize the ASI MegaCore function to create a variant that you can instantiate manually in your design

SOPC Builder Flow

The SOPC Builder flow allows you to add the ASI MegaCore function directly to a new or existing SOPC Builder system. You can also easily add other available components to quickly create an SOPC Builder system with an ASI, such as the Nios II processor, external memory controllers and scatter/gather DMA controllers. SOPC Builder automatically creates the system interconnect logic and system simulation environment.



For more information about SOPC Builder, refer to [volume 4](#) of the *Quartus II Handbook*. For more information on the Quartus II software, refer to the Quartus II Help.

Specify Parameters

To specify ASI parameters using the SOPC Builder flow, follow these steps:

1. In the Quartus II software, create a new Quartus II project with the **New Project Wizard**.
2. On the Tools menu, click **SOPC Builder**.
3. For a new system, specify the system name and language.
4. Add **ASI** to your system from the **System Contents** tab.



The **ASI** is in the **Interface Protocols > ASI** directory.

5. Specify the required parameters on all pages in the **Parameter Settings** tab.




For detailed explanation of the parameters, refer to [“Parameter Settings” on page 3-1](#).

6. Click **Finish** to complete the ASI MegaCore function and add it to the system.

Complete the SOPC Builder System

To complete the SOPC Builder system, follow these steps:


1. Add and parameterize any additional components to the system.
A typical SOPC builder system that enables Ethernet connectivity uses a scatter/gather DMA controller on each of the transmit and receive paths, and a Nios II processor for configuration and control.
2. Connect the components using the SOPC Builder patch panel.
3. To simulate your SOPC builder system, select **Simulate** on the **System Generation** tab to generate a functional simulation model for the system.
4. Click **Generate** to generate the system.

 Among the files generated by SOPC Builder is the Quartus II IP File (.qip). This file contains information about a generated IP core or system. In most cases, the .qip file contains all of the necessary assignments and information required to process the MegaCore function or system in the Quartus II compiler. Generally, a single .qip file is generated for each SOPC Builder system. However, some more complex SOPC Builder components generate a separate .qip file. In that case, the system .qip file references the component .qip file.

Simulate the System


During system generation, SOPC Builder optionally generates a simulation model and testbench for the entire system, which you can use to easily simulate your system in any of Altera's supported simulation tools. SOPC Builder also generates a set of ModelSim Tcl scripts and macros that you can use to compile the testbench, IP functional simulation models and plain-text RTL design files that describe your system in the ModelSim simulation software.

 For more information about simulating SOPC Builder systems, refer to [volume 4](#) of the *Quartus II Handbook* and [AN 351: Simulating Nios II Systems](#).

 Before compiling your design, you must run the Tcl constraint script.

MegaWizard Plug-In Manager Flow

The MegaWizard Plug-In Manager flow allows you to customize the ASI MegaCore function, and manually integrate the function into your design.

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

 For more information about the MegaWizard Plug-In Manager and the IP Advisor, refer to the Quartus II Help.

Specify Parameters

To specify ASI parameters using the MegaWizard Plug-In Manager flow, follow these steps:

1. In the Quartus II software, create a new Quartus II project with the **New Project Wizard**.
2. On the Tools menu click **MegaWizard Plug-In Manager** and follow the steps to start the MegaWizard Plug-In Manager.



The ASI MegaCore function is in the **Interface > ASI** directory.

3. Specify the parameters on all pages in the **Parameter Settings** tab.



For detailed explanation of the parameters, refer to the “**Parameter Settings**” on page 3-1.

4. On the **EDA** tab, turn on **Generate simulation model** to generate an IP functional simulation model for the MegaCore function.

An IP functional simulation model is a cycle-accurate VHDL or Verilog HDL model produced by the Quartus II software.



Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.



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**.

5. On the **Summary** tab, select the files you want to generate. A gray checkmark indicates a file that is automatically generated. All other files are optional.



For more information about the files generated in your project directory, refer to [Table 2-2](#).

6. Click **Finish** to generate the MegaCore function and supporting files.
7. If you generate the MegaCore function instance in a Quartus II project, you are prompted to add the **.qip** files to the current Quartus II project.



The **.qip** file is generated by the MegaWizard interface, and contains information about the generated IP core. In most cases, the **.qip** file contains all of the necessary assignments and information required to process the MegaCore function or system in the Quartus II compiler. The MegaWizard interface generates a single **.qip** file for each MegaCore function.

8. After you review the generation report, click **Exit** to close the MegaWizard Plug-In Manager.

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

Table 2–2. Generated Files

File Name	Description
<variation name>.v or .vhd	A MegaCore function variation file, which defines a VHDL or Verilog HDL description of the custom MegaCore function. Instantiate the entity defined by this file inside of your design. Include this file when compiling your design in the Quartus II software.
<variation name>.cmp	A VHDL component declaration file for the MegaCore function variation. Add the contents of this file to any VHDL architecture that instantiates the MegaCore function.
<variation name>.bsf	Quartus II symbol file for the MegaCore function variation. You can use this file in the Quartus II block diagram editor.
<variation name>.html	MegaCore function report file.
<variation name>.ppf	This XML file describes the MegaCore pin attributes to the Quartus II Pin Planner. MegaCore pin attributes include pin direction, location, I/O standard assignments, and drive strength. If you launch IP Toolbench outside of the Pin Planner application, you must explicitly load this file to use Pin Planner.
<variation name>.vo or .vho	VHDL or Verilog HDL IP functional simulation model.
<variation name>_bb.v	A Verilog HDL black-box file for the MegaCore function variation. Use this file when using a third-party EDA tool to synthesize your design.
<variation name>.qip	Contains Quartus II project information for your MegaCore function variations.

You can now integrate your custom MegaCore function variation into your design and 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, 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\modelsim\asi_sim.bat`, and edit it to point to your installation of the ModelSim-Altera simulator.
2. Start the ModelSim-Altera simulator.
3. Run `asi_sim.bat` in the `simulation\modelsim\modelsim` directory. This file compiles the design and starts the ModelSim-Altera simulator. A selection of signals appears on the waveform viewer. The simulation runs automatically, providing a pass/fail indication on completion.

Simulating in Third-Party Simulation Tools Using NativeLink

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



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

Altera provides a Quartus II project for use with NativeLink in the `ip\asi\simulation\quartus` 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\quartus` directory.
2. Open `asi_sim.qpf`.
3. Set up the Quartus II NativeLink.
 - a. On the Assignments menu, click **Settings**.
 - b. In the **Category** list, expand **EDA Tool Settings** and select **Simulation**.
 - c. In **Tool name** list, select a simulation tool.



Check that the absolute path to your third-party simulator executable is set. On the Tools menu, click **Options** and select **EDA Tools Options**.

- d. Under **NativeLink settings**, select **Compile test bench** and click **Test Benches**.
- e. Click **New** in the **Test Benches** dialog box to create a testbench.
4. In the **New Test Bench Settings** dialog box, perform the following steps:
 - a. In the **Test bench name** box, type the testbench setup name.
 - b. In the **Top level module in test bench** box, type the following as the project testbench name, `tb_asi_mc`.
 - c. In the **Design instance in test bench** box, type the name of the top-level instance.
 - d. Under **Simulation period**, set **End simulation at** to 500 μ s.

- e. Add the testbench files. In the **File name** field, browse to the location of the testbench, **tb_asi_mc**, click **Open** and then click **Add**.
- f. Select the files and click **OK**.
5. On the Processing menu, point to **Start** and click **Start Analysis & Elaboration**.
6. On the Tools menu, point to **Run EDA Simulation Tool** and click **EDA RTL Simulation**.

Compile the Design and Program a Device


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

After you have compiled your design, program your targeted Altera device and verify your design in hardware.

Table 3–1 summarizes the parameters.

Table 3–1. 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 blocks, 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, Stratix IV 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 Stratix II GX, Stratix IV GX or Stratix GX transceiver.

 You can change the page that the MegaWizard Plug-In Manager displays by clicking **Next** or **Back** at the bottom of the dialog box. You can move directly to a named page by clicking the **Parameter Settings**, **EDA**, or **Summary** tab.

The ASI MegaCore function consists of the following elements:

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

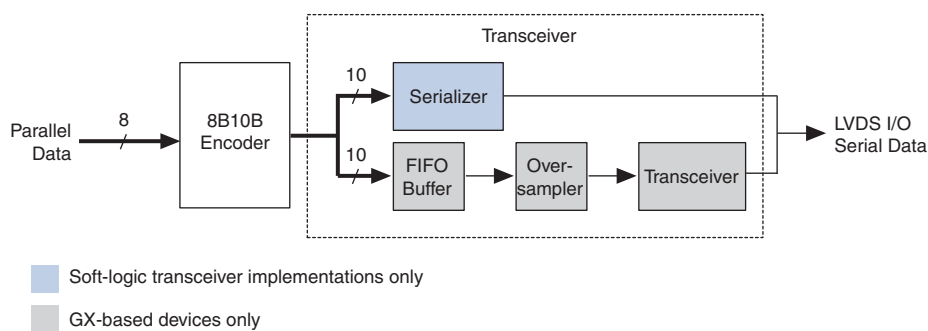
Transmitter

The transmitter comprises the following elements:

- 8B10B encoder
- Serializer

Figure 4–1 shows the ASI transmitter.

Figure 4–1. 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 IV transceiver, refer to the *Stratix IV Device Handbook*; for more information on the Stratix II GX transceiver, refer to the *Stratix II GX Device Handbook*; and for more information on the Stratix GX transceiver, refer to the *Stratix GX Device Handbook*.

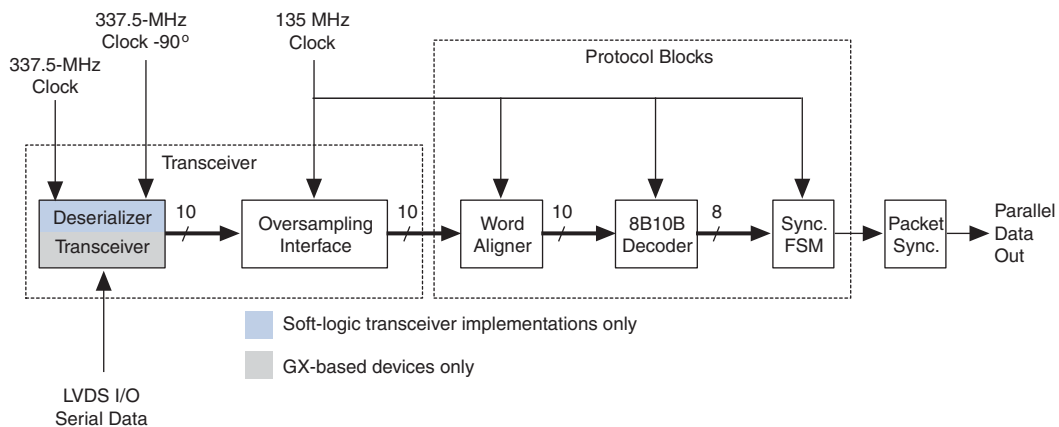
Receiver

The receiver comprises the following elements:

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

Figure 4-2 shows the ASI receiver.

Figure 4-2. 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 (for example, 270-bit rate \times 5/4 \times 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, refer to “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 IV transceiver, refer to the *Stratix IV Device Handbook*; for more information on the Stratix II GX transceiver, refer to the *Stratix II GX Device Handbook*; and for more information on the Stratix GX transceiver, refer to the *Stratix GX Device Handbook*.

Oversampling Interface

A 5 \times 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.

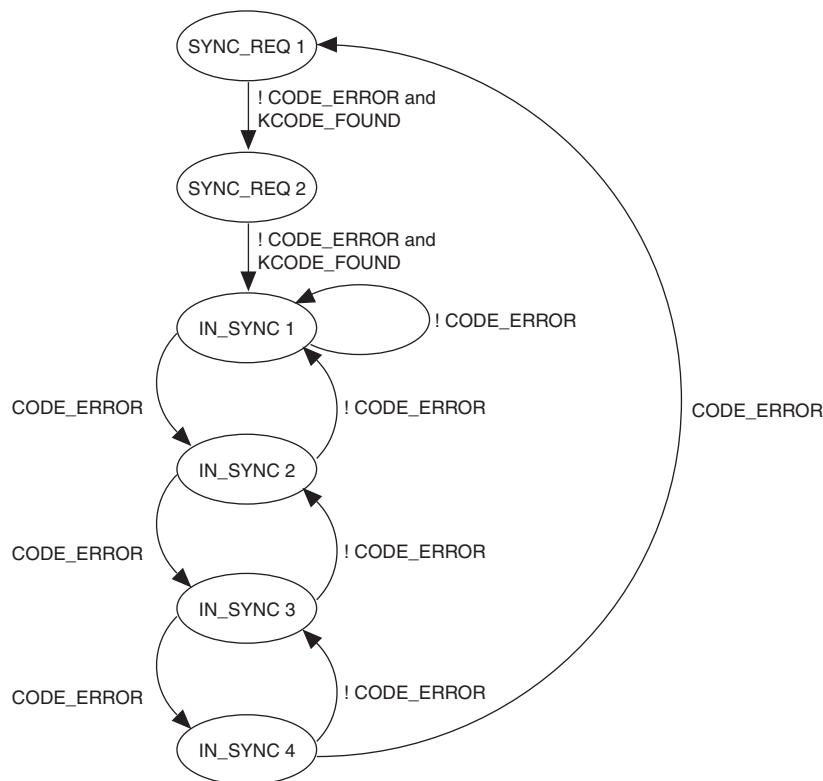
8B10B Decoder

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 4-3 shows the synchronization state machine.

Figure 4-3. 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_status[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.

Signals

Table 4-1 shows the signals.

Table 4-1. Signals (Part 1 of 2)

Signal	Direction	Description
asi_rx	Input	ASI input.
cal_blk_clk	Input	Calibration clock for Arria GX, Stratix II GX, and Stratix IV transceiver.
gxb_powerdown (1)	Input	Transceiver block reset and power down. This signal of all the instances that are to be combined into a single transceiver block must be connected to a single point; for example, the same input pin or same logic.
reconfig_clk (1) (2)	Input	Clock input for the embedded transceiver instance.
reconfig_togxb[3:0] (1) (2)	Input	Data input for the embedded transceiver instance.
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/transceiver mode.
asi_tx	Output	ASI output.
reconfig_fromgxb[16:0] (1) (2)	Output	Data output from the embedded transceiver instance.
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.

Table 4-1. Signals (Part 2 of 2)

Signal	Direction	Description
<code>rx_ts_status[7:0]</code>	Output	<p>TS status bits.</p> <p>0 indicates receiver data valid</p> <p>1 indicates start of packet</p> <p>2 indicates end of packet</p> <p>3 indicates receiver error</p> <p>5:4 indicates 00 is unlocked; 01 is 204 byte packet lock, 11 is 188 byte packet lock</p> <p>6 indicates TS serial polarity</p> <p>7 indicates that the data on <code>rx_data[7:0]</code> is a valid word from the 8B10B decoder. Unlike <code>rx_ts_status[0]</code>, this signal is not dependent on the correct packet or synchronization structure of the stream.</p>
<code>tx_protocol_out[9:0]</code>	Output	Output from transmitter protocol block for split protocol/transceiver mode.

Note for Table 4-1:

- (1) This signal is available for Stratix IV transceivers only.
- (2) In Quartus II version 8.1 and higher, the Stratix IV transceivers need RX buffer calibration through an `altgx_reconfig` (DPRIO) controller. You must connect the ports to the `altgx_reconfig` controller externally. For further information on the RX buffer calibration, refer to Stratix IV DPRIO documentation. If you are using Quartus II version 9.1, upgrade the ASI MegaCore to version 9.1 as well.

Introduction

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

- Specify clock characteristics
- Set timing exceptions such as false path, minimum delay and maximum delay
- 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

Constrain Design With TimeQuest Timing Analyzer

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

1. Set up the Quartus II TimeQuest timing analyzer.
 - a. To specify the Quartus II TimeQuest timing analyzer as the default timing analyzer, on the Assignments menu, click **Settings**.
 - b. In the **Settings** dialog box, under the **Category** list, select **Timing Analysis Settings**.
 - c. Turn on **Use TimeQuest Timing Analyzer during compilation** option, and click **OK**.
2. Perform initial compilation to create an initial design database before you specify timing constraints for your design. On the Processing menu, click **Start Compilation**.
3. Run the Quartus II TimeQuest timing analyzer. On the Tools menu, click **TimeQuest Timing Analyzer**.
4. Create timing netlist based on the fully annotated database from the post-fit results, after you perform a full compilation. Double-click **Create Timing Netlist** in the **Tasks** pane.
5. Write SDC constraint file. The Quartus II software does not automatically update **.sdc** files. You must explicitly write new or update constraints in the TimeQuest timing analyzer. On the Constraints menu, click **Write SDC File** to write your constraints to an **.sdc** file.
6. Specify timing constraints and exceptions. To enter your timing requirements, you can use constraint entry dialog boxes or edit the previously created **.sdc** file.

The following constraints demonstrates how to properly constrain the ASI MegaCore RX and TX targeting Stratix IV device.

Specify Clock Characteristics

Use the following constraints for the TimeQuest timing analyzer:

- ASI RX (Hard Transceiver) (rx_clk135 = 135 MHz)

```
create_clock -name {rx_clk135} -period 7.407 -waveform { 0.000 3.703 }
[get_ports {rx_clk135}]
```

- ASI TX (Hard Transceiver) (tx_clk135 = 135 MHz, tx_refclk = 27 MHz)

```
create_clock -name {tx_clk135} -period 7.407 -waveform { 0.000 3.703 }
[get_ports {tx_clk135}]
create_clock -name {tx_refclk} -period 37.037 -waveform { 0.000 18.518 }
[get_ports {tx_refclk}]
```

- ASI RX (Soft Transceiver) (rx_clk135 = 135 MHz, rx_serial_clk = 337.5 MHz, rx_serial_clk90 = 337.5 MHz)

```
create_clock -name {rx_clk135} -period 7.407 -waveform { 0.000 3.703 }
[get_ports {rx_clk135}]
create_clock -name {rx_serial_clk} -period 2.963 -waveform { 0.000 1.481 }
[get_ports {rx_serial_clk}]
create_clock -name {rx_serial_clk90} -period 2.963 -waveform { 0.000 1.481 }
[get_ports {rx_serial_clk90}]
```

- ASI TX (Soft Transceiver) (tx_clk270 = 270 MHz, tx_refclk = 27 MHz)

```
create_clock -name {tx_clk270} -period 3.704 -waveform { 0.000 1.852 }
[get_ports {tx_clk270}]
create_clock -name {tx_refclk} -period 37.037 -waveform { 0.000 18.518 }
[get_ports {tx_refclk}]
```

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

These constraints apply only to Soft Transceiver ASI.

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

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

- ASI RX (Soft Transceiver)

```
set_min_delay -from [get_clocks {rx_serial_clk}] -to [get_clocks
{rx_clk135}] 0.000
```

- ASI TX (Soft Transceiver)

```
set_min_delay -from [get_clocks {tx_refclk}] -to [get_clocks
{tx_clk270}] 0.000
```

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

- ASI RX (Soft Transceiver)

```
set_max_delay -from [get_clocks {rx_serial_clk}] -to [get_clocks
{rx_clk135}] 4.430
```

- ASI TX (Soft Transceiver)

```
set_max_delay -from [get_clocks {tx_refclk}] -to [get_clocks
{tx_clk270}] 33.333
```

Specify Clocks that are Exclusive or Asynchronous

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



The following SDC commands are only applicable for Stratix IV, you must use the constraint entry dialog boxes to constrain for other device families.

- ASI TX (Hard Transceiver)

```
set_clock_groups -exclusive -group [get_clocks {tx_clk135}] -group
[get_clocks
{asi_megacore_top_inst|asi_tx_gen.u_tx|u_gxb4_tx.u_gxb|alt4gxb_compone
nt|auto_generated|transmit_pcs0|clkout}]
```

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 I/O, 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_gen.u_rx|serdes_s2p:u_s2p|sample_a[0]"

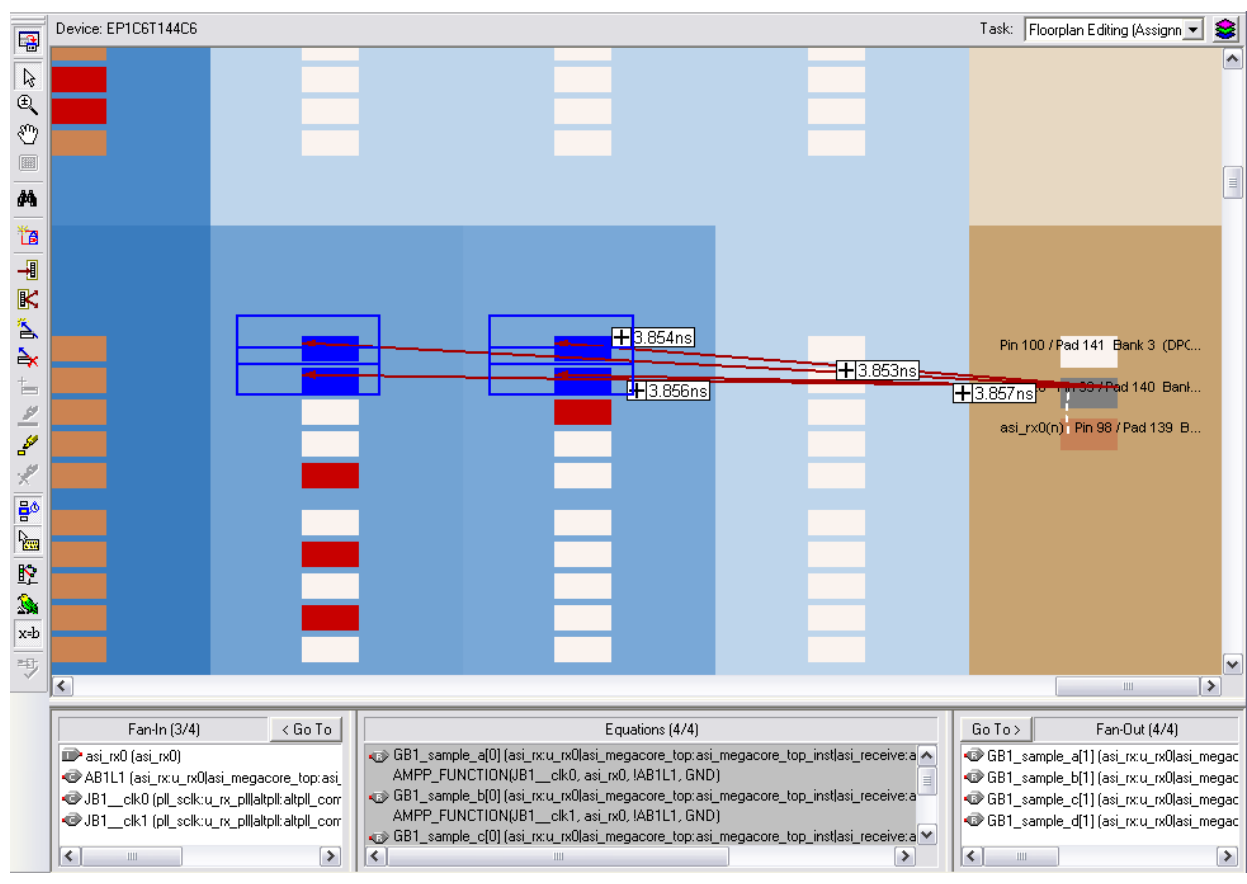
set_location_assignment LC_X33_Y17_N0 -to
"asi_rx:u_rx0|asi_megacore_top:asi_megacore_top_inst|asi_receive:asi
rx_gen.u_rx|serdes_s2p:u_s2p|sample_b[0]"

set_location_assignment LC_X32_Y17_N1 -to
"asi_rx:u_rx0|asi_megacore_top:asi_megacore_top_inst|asi_receive:asi
rx_gen.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_gen.u_rx|serdes_s2p:u_s2p|sample_d[0]"
```

Figure A-1 shows the placement of these registers in the Quartus II chip planner floorplan.

Figure A-1. Register Placement



Constraints For ASI Receivers

You must add constraints to receiver ASI MegaCore functions. There are constraints specific only to Cyclone devices and there are other constraints that apply to the other device families (including Cyclone II, Cyclone III, and Cyclone IV devices). There are different constraints that apply to the Classic timing analyzer and the TimeQuest timing analyzer.

Non-Cyclone Devices

These constraints apply to all device families (excluding Cyclone, but including Cyclone II, Cyclone III, and Cyclone IV devices) 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.



To avoid the `u_rx_pll|c0` and `u_rx_pll|c2` nodes from getting synthesized away during analysis and synthesis, make sure the reset port of the core is connected to an input pin and not to the ground; and apply the following the following additional constraints on the two nodes:

```
set_instance_assignment -name IMPLEMENT_AS_OUTPUT_OF_LOGIC_CELL ON -to
"u_rx_pll|c0"
```

```
set_instance_assignment -name IMPLEMENT_AS_OUTPUT_OF_LOGIC_CELL ON -to
"u_rx_pll|c2"
```

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 devices only (not Cyclone II, Cyclone III or other 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}

```

Revision History

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

Date	Version	Changes Made
November 2009	9.1	Added support for Cyclone III LS and Cyclone IV.
March 2009	9.0	Added support for Arria II GX.
November 2008	8.1	Updated Appendix A: Constraints.
May 2008	8.0	Added support for Stratix IV.
October 2007	7.2	Added SOPC Builder information.
May 2007	7.1	<ul style="list-style-type: none"> ■ Updated device support. ■ Added packet synchronization information.
December 2006	7.0	Added support for Cyclone III devices.
December 2006	6.0	<ul style="list-style-type: none"> ■ 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, see the following table.






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

Note:

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

Typographic Conventions

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

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