

LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard v1.11

User Guide

UG516 (v1.11) October 19, 2011



The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials, or to advise you of any corrections or update. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.

© Copyright 2009–2011 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. CPRI is a trademark of Siemens AG. PCI, PCIe and PCI Express are trademarks of PCI-SIG and used under license. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/24/09	1.2	Initial Xilinx release.
09/16/09	1.3	Tools and Wizard updates. Added support for the Virtex-6 HXT family. Added Chapter 4, Quick Start Example Design and renumbered Chapter 5, Detailed Example Design .
12/02/09	1.4	Tools and Wizard updates. Added support for Virtex-6 Lower Power family. Added Recommended Design Experience in Chapter 1 . Replaced Appendix A: References with Related Xilinx Documents in Chapter 1 . Updated Table 3-18 .
04/19/10	1.5	Tools and Wizard updates. Added CXT support. Added REFCLK0/01 Q9 in Table 3-6 , updated TXUSRCLK and RXUSRCLK in Table 3-7 , added scripts to Table 5-8 .
07/23/10	1.6	Tools and Wizard updates.
09/21/10	1.7	Wizard v1.7 release.
12/14/10	1.8	<p>Wizard v1.8 release.</p> <p>Integrated the <i>LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard Data Sheet (DS708)</i> and <i>Getting Started Guide (UG516)</i>. The integrated document is named the <i>LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard v1.11 User Guide (UG516)</i>.</p> <p>Modified the following chapters to incorporate the integration:</p> <ul style="list-style-type: none">• Chapter 1, Introduction• Chapter 2, Installing the Wizard• Chapter 3, Running the Wizard <p>Other changes:</p> <ul style="list-style-type: none">• Configuring and Generating the Wrapper in Chapter 3:<ul style="list-style-type: none">• Modified heading name to be more descriptive.• Chapter 4, Quick Start Example Design:<ul style="list-style-type: none">• Rearranged the sections• Renamed the simulation section to Functional Simulation of the Example Design
03/01/11	1.9	Wizard v1.9 release. Tools and Wizard versions updates. Removed the Conventions section from the Preface. In Chapter 2, Installing the Wizard , changed section title from Tested Design Tools to Design Tools and removed reference to service packs. In Chapter 3, Running the Wizard , in Overview , added note about GUI screenshots used in the guide. Changed title of “Wrapper Overview” section to Structure of the GTX Transceiver Wrapper, Example Design, and Testbench and replaced its text, removed component numbers from and renamed Figure 3-2 . Changed title of “Example Design Overview” section to Example Design—XAUI Configuration and renamed Figure 3-3 . In Chapter 5, Detailed Example Design , renamed Figure 5-1 .

Date	Version	Revision
06/22/11	1.10	<p>Wizard v1.10 release.</p> <p>In Chapter 1, Introduction, revised About the Wizard, Features, Supported Devices, Recommended Design Experience, and Related Xilinx Documents.</p> <p>In Chapter 2, Installing the Wizard, revised Tools and System Requirements and Installing the Wizard.</p> <p>In Chapter 3, Running the Wizard, revised title and content of Structure of the GTX Transceiver Wrapper, Example Design, and Testbench. Renamed Figure 3-2. Updated Data Path Width descriptions and added four entries for 64B/66B_with_Ext_Seq_Ctr to Encoding in Table 3-1. Updated Data Path Width descriptions and added two entries for 64B/66B to Decoding in Table 3-2. Added TXRATE, GTXTEST, and RXRATE to Table 3-8. Update Comma Mask description in Table 3-9. Added instruction to paragraph following Table 3-12.</p> <p>In Chapter 4, Quick Start Example Design, revised Source Directories in Table 4-1.</p> <p>In Chapter 5, Detailed Example Design, revised ChipScope Pro tools description and added PlanAhead software support in Table 5-5.</p>
10/19/11	1.11	<p>Wizard v1.11 release.</p> <p>Chapter 1: Revised About the Wizard. In Ordering Information, replaced 13.2 with 13.3.</p> <p>Chapter 2: In Operating Systems, replaced 13.2 with 13.3.</p> <p>Chapter 3: In Table 3-1, shaded all entries for Encoding, except for 8B/10B. In Table 3-2, shaded all entries for Decoding, except for 8B/10B. In Table 3-6, renamed REFCLK0/01 to REFCLK0/1, and renumbered Q4 through Q9 as Q3 through Q8. Renamed Table 3-14. In Table 3-17, removed shading from all rows except RXPRBSERR_LOOPBACK. In Table 3-19, removed sentence about XAUI example using 7 from description of Sequence 2 Max Skew.</p> <p>Chapter 4: Added Netlist Simulation of the Example Design.</p> <p>Chapter 5: In Table 5-7, added rows for <code>sim_reset_mgt_model.vhd</code> and <code>demo_tb_imp.v[hd]</code>. Added simulation/netlist, containing Table 5-9.</p>

Table of Contents

Revision History	3
Preface: About This Guide	
Guide Contents	7
Additional Resources	7
Chapter 1: Introduction	
About the Wizard	9
Features	10
Supported Devices	10
Provided with the Wizard	10
Recommended Design Experience	11
Related Xilinx Documents	11
Technical Support	11
Ordering Information	11
Feedback	12
Wizard	12
Document	12
Chapter 2: Installing the Wizard	
Tools and System Requirements	13
Operating Systems	13
Design Tools	13
Before You Begin	14
Installing the Wizard	14
Verifying Your Installation	14
Chapter 3: Running the Wizard	
Overview	15
Functional Overview	15
Structure of the GTX Transceiver Wrapper, Example Design, and Testbench	17
Example Design—XAUI Configuration	18
Setting Up the Project	19
Creating a Directory	19
Setting the Project Options	20
Configuring and Generating the Wrapper	21
Line Rates and Encoding	22
GTX Transceiver Placement and Clocking	27
Synchronization and Alignment	29
Preemphasis, Termination, and Equalization	32
PCI Express, SATA, OOB, PRBS, and RX Loss of Sync	34
Channel Bonding Sequence	37

Clock Correction Sequence	39
Summary	40

Chapter 4: Quick Start Example Design

Overview	41
Functional Simulation of the Example Design	41
Using ModelSim	41
Using the ISE Simulator	42
Implementing the Example Design	42
Netlist Simulation of the Example Design	43
Using ModelSim	43

Chapter 5: Detailed Example Design

Directory and File Structure	45
Directory and File Contents	46
<project directory>	46
<project directory>/<component name>	46
<component name>/doc	47
<component name>/example design	47
<component name>/implement	48
implement/results	49
<component name>/simulation	49
simulation/functional	49
simulation/netlist	50
Example Design Description	50
Example Design Hierarchy	51

About This Guide

This guide describes the Virtex®-6 FPGA GTX Transceiver Wizard (hereinafter called the Wizard).

Guide Contents

This guide contains the following chapters:

- [Chapter 1, Introduction](#), describes the Wizard and related information, including additional resources, technical support, and submitting feedback to Xilinx.
- [Chapter 2, Installing the Wizard](#), provides information about installing the Wizard.
- [Chapter 3, Running the Wizard](#), provides an overview of the Wizard and a step-by-step tutorial to generate a sample GTX transceiver wrapper with the CORE Generator™ tool.
- [Chapter 4, Quick Start Example Design](#), introduces the example design that is included with the GTX transceiver wrappers. The example design demonstrates how to use the wrappers and demonstrates some of the key features of the GTX transceiver.
- [Chapter 5, Detailed Example Design](#), provides detailed information about the example design, including a description of files and the directory structure generated by the CORE Generator tool, the purpose and contents of the provided scripts, the contents of the example HDL wrappers, and the operation of the demonstration testbench.

Additional Resources

To find additional documentation, see:

<http://www.xilinx.com/support/documentation/index.htm>

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see:

<http://www.xilinx.com/support/mysupport.htm>

Introduction

This chapter describes the Virtex®-6 FPGA GTX Transceiver Wizard and provides related information, including additional resources, technical support, and instructions for submitting feedback to Xilinx.

About the Wizard

The Virtex-6 FPGA GTX Wizard automates the task of creating HDL wrappers to configure the high-speed serial GTX transceivers in Virtex-6 FPGAs.

The menu-driven interface gives the user the option to configure one or more GTX transceivers either by using predefined templates for popular industry standards, or by creating custom templates to support a wide variety of custom protocols.

The Wizard produces a wrapper, an example design, and a testbench to enable rapid integration and verification of the serial interface with your custom function. The wrapper instantiates and configures one or more GTX transceivers ([Figure 1-1](#)).

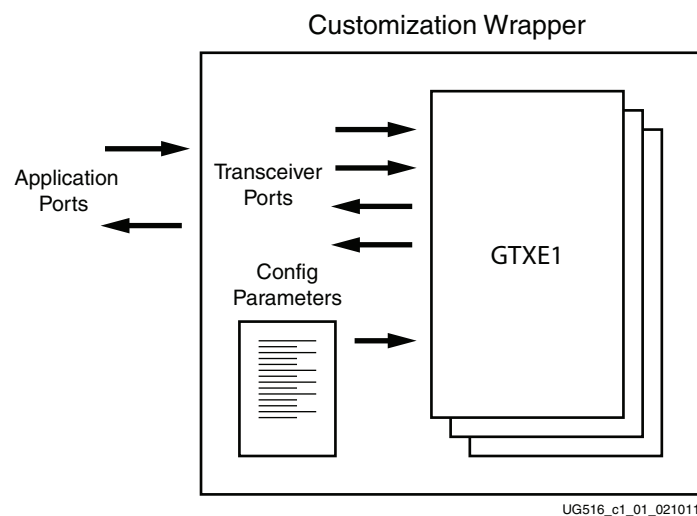


Figure 1-1: GTX Transceiver Wizard Wrapper

The Wizard is accessed from the ISE® software CORE Generator tool. For information about system requirements and installation, see [Chapter 2, Installing the Wizard](#).

For the latest information on the Wizard, refer to the Architecture Wizards product information page at:

http://www.xilinx.com/products/design_resources/conn_central/solution_kits/wizards

For documentation, see the Virtex-6 FPGA GTX Transceiver Wizard page:
http://www.xilinx.com/support/documentation/ipfpgafeaturedesign_iointerface_v6gtxwizard.htm

Features

The Wizard has these features:

- Creates customized HDL wrappers to configure Virtex-6 FPGA GTX transceivers:
 - Predefined templates automate transceiver configuration for industry standard protocols such as: Aurora (8B/10B and 64B/66B), Common Packet Radio Interface (CPRI™), DisplayPort, Fibre Channel, Gigabit Ethernet, High-Definition Serial Digital Interface (HD-SDI), 3 Gb/s Serial Digital Interface (3G-SDI), Interlaken, Open Base Station Architecture Initiative (OBSAI), OC-48, PCI EXPRESS® (PCIe®) generation I and II, Serial Advanced Technology Attachment (SATA) 1.5 Gb/s, SATA 3 Gb/s, Serial RapidIO generation I and II, 10 Gb Attachment Unit Interface (XAUI), and RXAUI-Dune Networks
 - Custom protocols can be specified using the **Start from Scratch** option in the GUI.
- Automatically configures transceiver analog settings of the Virtex-6 FPGA GTX transceivers
- Supports 8B/10B, 64B/66B and 64B/67B encoding/decoding
- Includes an example design with a companion testbench, as well as implementation and simulation scripts

Supported Devices

The Wizard supports the Virtex-6 LXT, SXT, and HXT families. For a complete listing of supported devices, see [XTP025](#), *IP Release Notes Guide* for this Wizard. For more information on the Virtex-6 FPGAs, see [DS150](#), *Virtex-6 Family Overview*.

Provided with the Wizard

The following are provided with the Wizard:

- Documentation: This user guide
- Design Files: Verilog and VHDL
- Example Design: Verilog and VHDL
- Testbench: Verilog and VHDL
- Constraints File: Synthesis constraints file
- Simulation Model: Verilog and VHDL

Recommended Design Experience

The Wizard is a fully verified solution that helps automate the task of defining parameter settings for Virtex-6 FPGA GTX transceivers. The additional challenge associated with implementing a complete design depends on the configuration and required functionality of the application. For best results, previous experience building high-performance, pipelined FPGA designs using Xilinx implementation software and user constraints files (UCF) is recommended.

For those with less experience, Xilinx offers various training classes to help with various aspects of designing with Xilinx FPGAs. These include classes on such topics as designing for performance and designing with multi-gigabit serial I/O. For more information, see <http://www.xilinx.com/training>.

Xilinx sales representatives can provide a closer review and estimation of specific design requirements.

Related Xilinx Documents

For detailed information and updates about the Wizard, see the following:

- [UG516](#), *LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard v1.11 User Guide*
- [XTP025](#), *IP Release Notes Guide* for the Wizard

Prior to generating the Wizard, users should be familiar with the following:

- [DS150](#), *Virtex-6 Family Overview*
- [UG366](#): *Virtex-6 FPGA GTX Transceivers User Guide*
- ISE software documentation at <http://www.xilinx.com/ise>

Technical Support

For technical support, go to www.xilinx.com/support. Questions are routed to a team of engineers with expertise using this Wizard.

Xilinx provides technical support for use of this product as described in this guide. Xilinx cannot guarantee timing, functionality, or support of this product for designs that do not follow these guidelines.

Ordering Information

The Wizard is provided free of charge under the terms of the [Xilinx End User License Agreement](#). The Wizard can be generated by the ISE software CORE Generator tool 13.3 or higher, which is a standard component of the ISE Design Suite. For more information, visit the [Architecture Wizards web page](#). Information about additional LogiCORE modules is available at the [IP Center](#). For pricing and availability of other LogiCORE modules and software, contact a local Xilinx [sales representative](#).

Feedback

Xilinx welcomes comments and suggestions about the Wizard and the accompanying documentation.

Wizard

For comments or suggestions about the Wizard, submit a WebCase from www.xilinx.com/support. (Registration is required to log in to WebCase.) Be sure to include the following information:

- Product name
- Wizard version number
- List of parameter settings
- Explanation of any comments, including whether the case is requesting an *enhancement* (improvement) or reporting a *defect* (something is not working correctly)

Document

For comments or suggestions about this document, submit a WebCase from www.xilinx.com/support. (Registration is required to log in to WebCase.) Be sure to include the following information:

- Document title
- Document number
- Page number(s) to direct applicable comments
- Explanation of any comments, including whether the case is requesting an *enhancement* (improvement) or reporting a *defect* (something is not working correctly)

Installing the Wizard

This chapter provides instructions to install the Virtex®-6 FPGA GTX Transceiver Wizard in the ISE® Design Suite CORE Generator™ tool.

Tools and System Requirements

Operating Systems

For a list of system requirements, see [13.3 Release Notes/ Known Issues](#).

Design Tools

Design Entry

- ISE Design Suite CORE Generator software 13.3
- PlanAhead™ software 13.3

Simulation

- ISE Simulator (ISim) software 13.3
- Mentor Graphics ModelSim 6.6d
- Cadence Incisive Enterprise Simulator (IES) 10.2
- Synopsys Verilog Compiler Simulator (VCS) and VCS MX E-2011.03

See [XTP025](#), *IP Release Notes Guide* for the Wizard for the required service pack. ISE software service packs can be downloaded from <http://www.xilinx.com/support/download.htm>

Synthesis

- XST 13.3
- Synopsys Synplify Pro E-2011.03-SP2

Before You Begin

Before installing the Wizard, you must have a MySupport account and the ISE 13.3 software installed on your system. If you already have an account and have the software installed, go to [Installing the Wizard](#), otherwise:

1. Click **Login** at the top of the Xilinx home page and follow the onscreen instructions to create a MySupport account.
2. Install the ISE 13.3 software.

For the software installation instructions, see the ISE Design Suite Release Notes and Installation Guide available in ISE software Documentation.

Installing the Wizard

The Wizard is included with the ISE 13.3 software. Follow the ISE software 13.3 installation instructions in the ISE Installation and Release Notes available at <http://www.xilinx.com/support/documentation> under the Design Tools tab.

Verifying Your Installation

Use the following procedure to verify a successful installation of the Wizard in the CORE Generator tool.

1. Start the CORE Generator tool.
2. The IP core functional categories appear at the left side of the window ([Figure 2-1](#)).

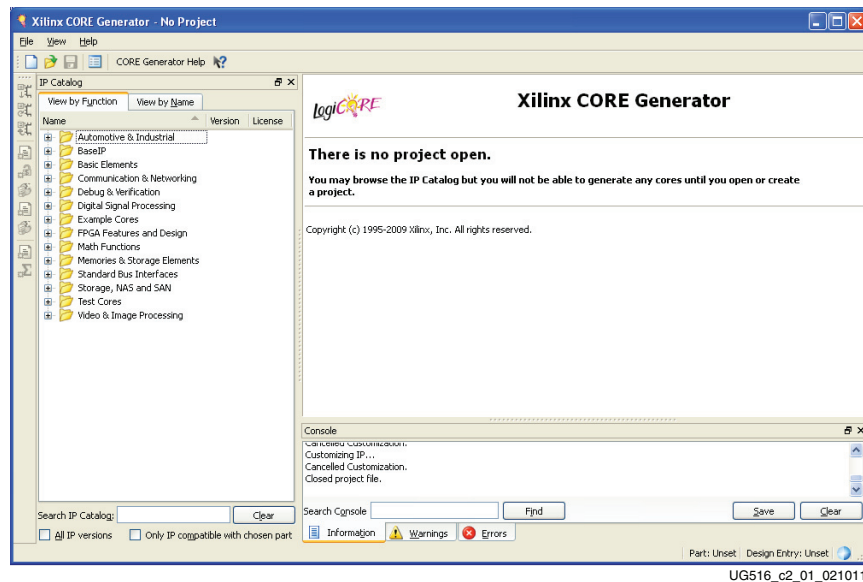


Figure 2-1: CORE Generator Window

3. Click to expand or collapse the view of individual functional categories, or click the **View by Name** tab at the top of the list to see an alphabetical list of all cores in all categories.
4. Determine if the installation was successful by verifying that Virtex-6 FPGA GTX Transceiver Wizard 1.11 appears at the following location in the Functional Categories list: /FPGA Features and Design/IO Interfaces

Running the Wizard

Overview

This chapter provides a step-by-step procedure for generating a Virtex®-6 FPGA GTX transceiver wrapper, implementing the wrapper in hardware using the accompanying example design, and simulating the wrapper with the provided example testbench.

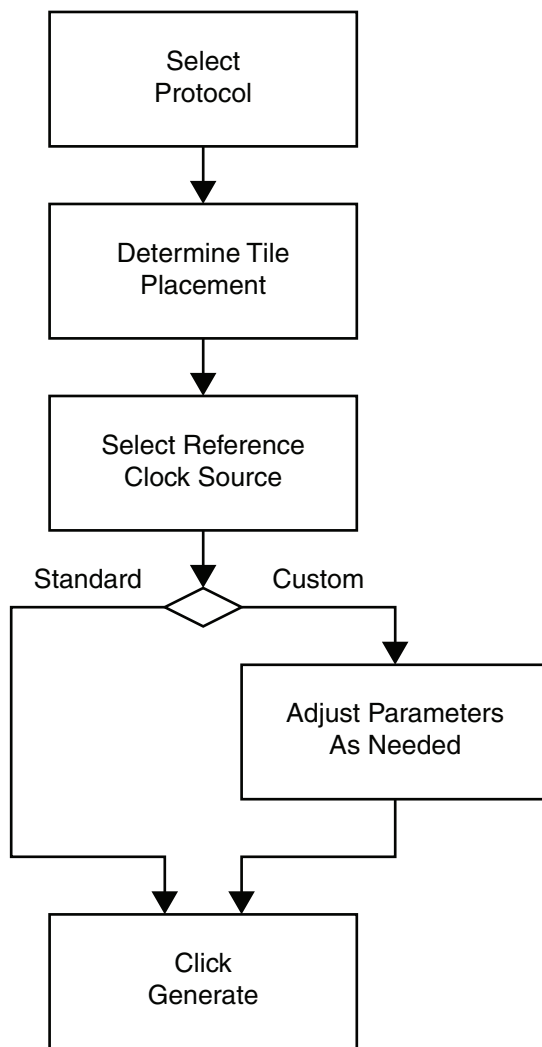
Note: The screen captures in this chapter are conceptual representatives of their subjects and provide general information only. For the latest information, see the CORE Generator™ tool.

Functional Overview

Figure 3-1, page 16 shows the steps required to configure GTX transceivers using the Wizard. Start the CORE Generator software and select the Virtex-6 FPGA GTX Transceiver Wizard, then follow the chart to configure the transceivers and generate a wrapper that includes the accompanying example design.

- To use an existing template with no changes, click **Generate**.
- To modify a standard template or start from scratch, proceed through the Wizard and adjust the settings as needed.

See [Configuring and Generating the Wrapper](#), page 21 for details on the various transceiver features and parameters available.

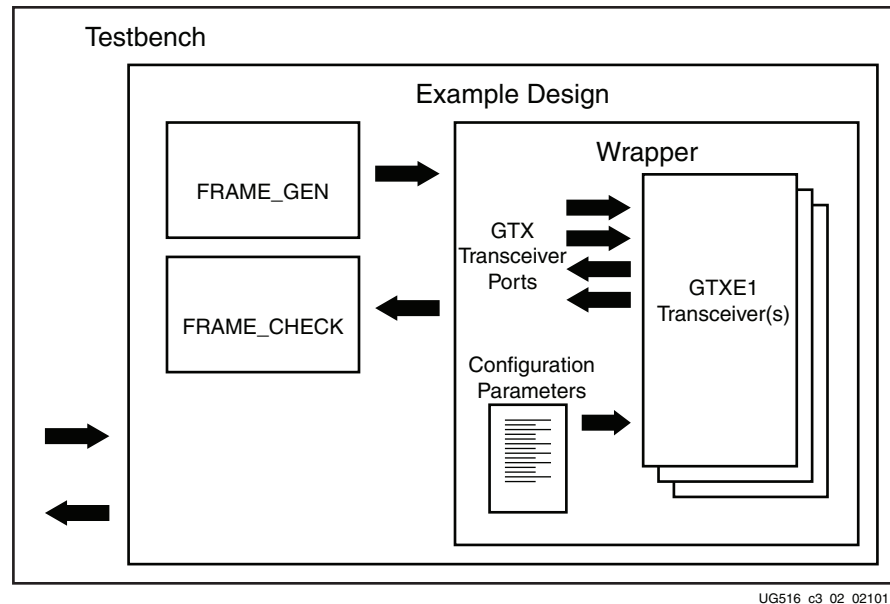


UG516_c3_01_021011

Figure 3-1: Wizard Configuration Steps

Structure of the GTX Transceiver Wrapper, Example Design, and Testbench

Figure 3-2 shows the relationship of the GTX transceiver wrapper, example design, and testbench files generated by the Wizard. For details, see [Example Design Description](#), page 50.



UG516_c3_02_021011

Figure 3-2: Structure of the GTX Transceiver Wrapper, Example Design, and Testbench

These files are generated by the Wizard to illustrate the components needed to simulate the configured transceiver:

- GTX transceiver wrapper, which includes:
 - Specific gigabit transceiver configuration parameters set using the Wizard.
 - GTXE1 transceiver primitive(s) selected using the Wizard.
- Example design demonstrating the modules required to simulate the wrapper. These include:
 - FRAME_GEN module: Generates a user-definable data stream for simulation analysis.
 - FRAME_CHECK module: Tests for correct transmission of data stream for simulation analysis.
- Testbench:
 - Top-level testbench that demonstrates how to stimulate the design.

Example Design—XAUI Configuration

The example design covered in this section is a wrapper that configures a group of GTX transceivers for use in a XAUI application. Guidelines are also given for incorporating the wrapper in a design and for the expected behavior in operation. For detailed information, see [Chapter 4, Quick Start Example Design](#).

The XAUI example consists of the following components:

- A single GTX transceiver wrapper implementing a 4-lane XAUI port using four GTX transceivers
- A demonstration testbench to drive the example design in simulation
- An example design providing clock signals and connecting an instance of the XAUI wrapper with modules to drive and monitor the wrapper in hardware, including optional ChipScope™ Pro tool support
- Scripts to synthesize and simulate the example design

The Wizard example design has been tested with Synplify Pro E-2011.03 and XST 13.3 for synthesis and ModelSim 6.6d for simulation.

[Figure 3-3](#) shows a block diagram of the default XAUI example design.

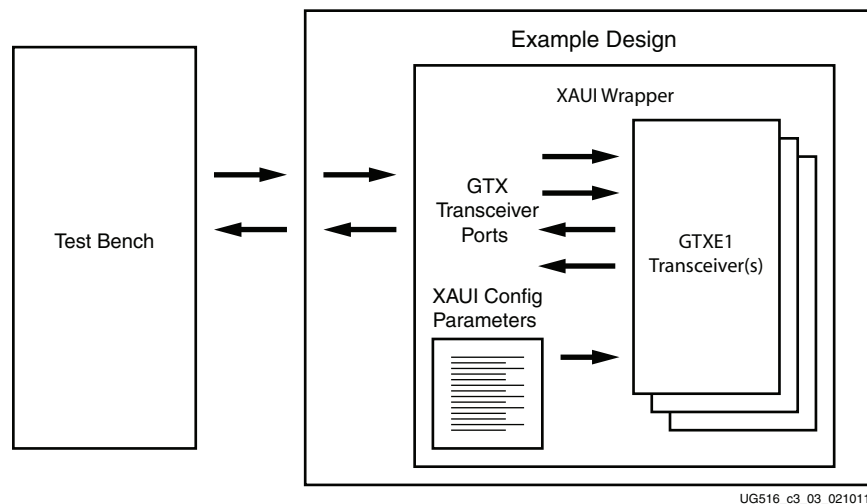


Figure 3-3: Example Design and Testbench—XAUI Configuration

Setting Up the Project

Before generating the example design, set up the project as described in [Creating a Directory](#) and [Setting the Project Options](#).

Creating a Directory

To set up the example project, first create a directory using the following steps:

1. Change directory to the desired location. This example uses the following location and directory name:

```
/Projects/xau1_example
```

2. Start the CORE Generator software.

For help starting and using the CORE Generator software, see *CORE Generator Help*, available in ISE® software documentation.

3. Choose **File > New Project** (Figure 3-4).
4. Change the name of the CGP file (optional).
5. Click **Save**.

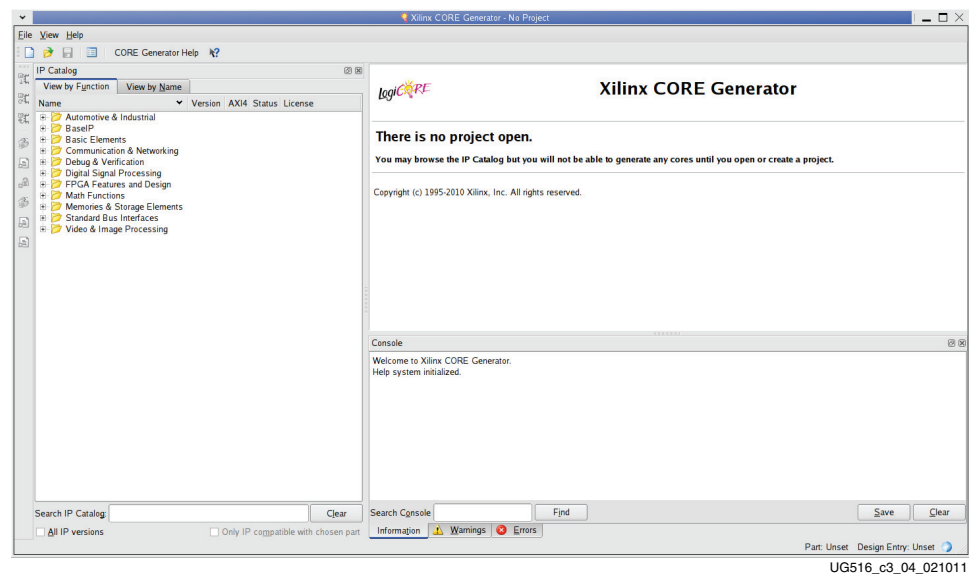


Figure 3-4: Starting a New Project

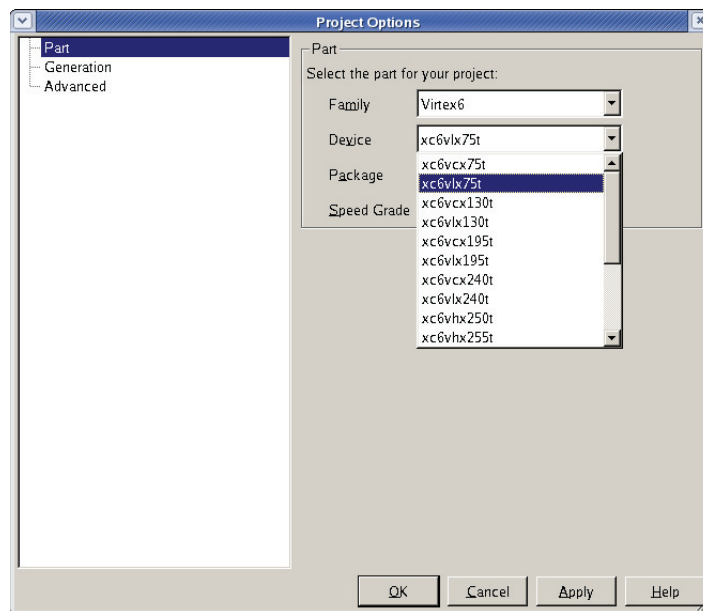
Setting the Project Options

Set the project options using the following steps:

1. Click **Part** in the option tree.
2. Select **Virtex6/Virtex6 Lower Power** from the Family list.
3. Select a device from the Device list that supports GTX transceivers.
4. Select an appropriate package from the Package list. This example uses the XC6VLX75T device (see [Figure 3-5](#)).

Note: If an unsupported silicon family is selected, the Virtex-6 FPGA GTX Transceiver Wizard remains light grey in the taxonomy tree and cannot be customized. Only devices containing Virtex-6 FPGA GTX transceivers are supported by the Wizard. See [DS150](#), *Virtex-6 Family Overview* for a list of devices containing GTX transceivers.

5. Click **Generation** in the option tree and select either Verilog or VHDL as the output language.
6. Click **OK**.



UG516_c3_05_021011

Figure 3-5: Target Architecture Setting

Configuring and Generating the Wrapper

This section provides instructions for generating an example GTX transceiver wrapper using the default values. The wrapper, associated example design, and supporting files are generated in the project directory. For additional details about the example design files and directories see [Chapter 5, Detailed Example Design](#).

1. Locate the Virtex-6 FPGA GTX Transceiver Wizard 1.11 in the taxonomy tree under: /FPGA Features & Design/IO Interfaces. (See [Figure 3-6](#))
2. Double-click **Virtex-6 FPGA GTX Transceiver Wizard 1.11** to launch the Wizard.

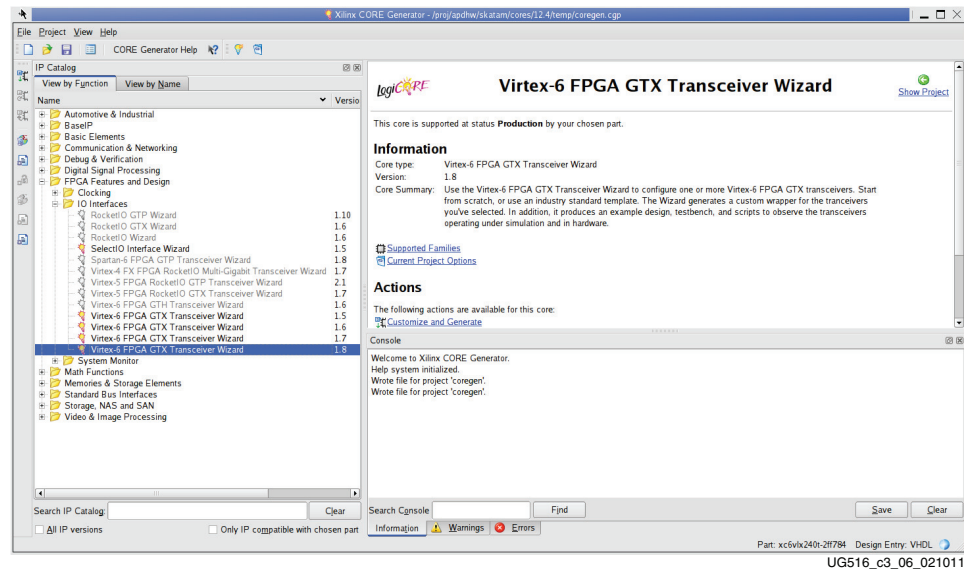


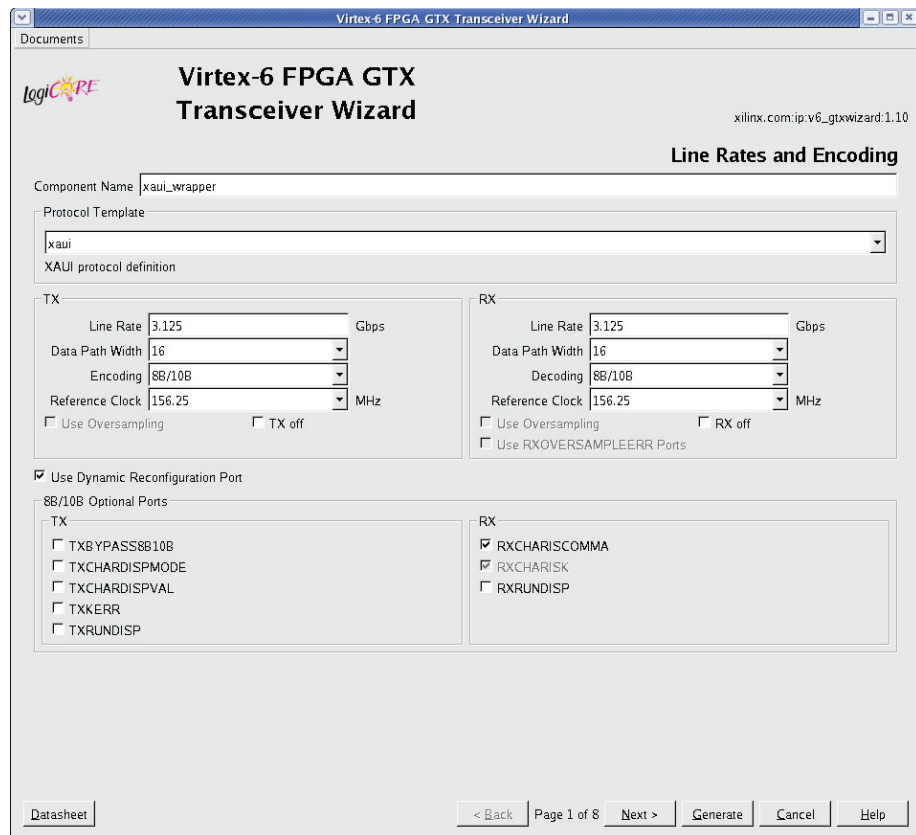
Figure 3-6: Locating the GTX Transceiver Wizard

Line Rates and Encoding

Page 1 of the Wizard (Figure 3-7) allows you to select the component name and determine the line rate, reference clock frequency, encoding/decoding method, and data width. In addition, this page specifies a protocol template.

1. In the Component Name field, enter a name for the Wizard instance. This example uses the name `xau_i_wrapper`.
2. From the Protocol Template list, select **Start from scratch** to manually set all parameters.

Select one of the available protocols from the list to begin designing with a predefined protocol template. The XAUI example uses the XAUI protocol template.



UG516_c3_07_060211

Figure 3-7: Line Rates and Encoding—Page 1

3. Use Tables 3-1 through 3-4 to determine the line rate, encoding, decoding, reference clock, and optional ports settings available on this page.

Table 3-1: TX Settings

Options		Description
Line Rate		Set to the desired target line rate in Gb/s. Can be independent of the receive line rate. The XAUI example uses 3.125 Gb/s.
Data Path Width	8	Sets the transmitter application interface datapath width to a single 8-bit byte. If 8B/10B encoding is selected, the internal transmitter datapath width will be set to two 10-bit bytes (20 bits). If 8B/10B encoding is not selected, the internal transmitter datapath width will be set to two 8-bit bytes (16 bits).
	10	Sets the transmitter application interface datapath width to a single 10-bit byte. Sets the internal transmitter datapath width to two 10-bit bytes (20 bits).
	16	If 8B/10B encoding is selected, the internal transmitter datapath width will be set to two 10-bit bytes (20 bits). If 8B/10B encoding is not selected, the internal transmitter datapath width will be set to two 8-bit bytes (16 bits).
	20	Sets the transmitter application interface datapath width to two 10-bit bytes (20 bits). Sets the internal transmitter datapath width to two 10-bit bytes (20 bits).
	32	Sets the transmitter application interface datapath width to four 8-bit bytes (32 bits). If 8B/10B encoding is selected, the internal transmitter datapath width will be set to two 10-bit bytes (20 bits). If 8B/10B encoding is not selected, the internal transmitter datapath width will be set to two 8-bit bytes (16 bits).
	40	Sets the transmitter application interface datapath width to four 10-bit bytes (40 bits). Sets the internal transmitter datapath width to two 10-bit bytes (20 bits).
Encoding	None	Data stream is passed with no conversion.
	None (MSB First)	Same as above but reorders bits for applications expecting most significant bit first.
	8B/10B	Data stream is passed to an internal 8B/10B encoder prior to transmission.
	64B/66B _with_Ext_Seq_Ctr	Data stream is passed through the 64B/66B gearbox and scrambler. Sequence counter for the gearbox is implemented in the FRAME_GEN module which is instantiated in the example design.
	64B/66B _with_Int_Seq_Ctr	Data stream is passed through the 64B/66B gearbox and scrambler. Sequence counter for the gearbox is implemented within the GTX transceiver.
	64B/67B _with_Ext_Seq_Ctr	Data stream is passed through the 64B/67B gearbox and scrambler. Sequence counter for the gearbox is implemented in the FRAME_GEN module which is instantiated in the example design.
64B/67B _with_Int_Seq_Ctr	Data stream is passed through the 64B/67B gearbox and scrambler. Sequence counter for the gearbox is implemented within the GTX transceiver.	
Reference Clock		Select from the list the optimal reference clock frequency to be provided by the application. The XAUI example uses 156.25 MHz.

Table 3-1: TX Settings (Cont'd)

Options	Description
Use Oversampling	The Wizard supports oversampling for line rates between 60 Mb/s and 1300 Mb/s. For line rates of 120–600 Mb/s, this option is automatically selected and the check box is disabled. For line rates of 600–1300 Mb/s, the checkbox is enabled allowing optional selection of this feature. This option is not available for XAUI since the line rate exceeds the permissible range.
No TX	Selecting this option disables the TX path of the GTX transceiver. The transceiver will function as a receiver only. The XAUI example design requires both TX and RX functionality.

Note: Options not used by the XAUI example are shaded.

Table 3-2: RX Settings

Options	Description	
Line Rate	Set to the desired target line rate in Gb/s. Can be independent of the transmit line rate. The XAUI example uses 3.125 Gb/s.	
Data Path Width	8	Sets the receiver application interface datapath width to a single 8-bit byte. If 8B/10B decoding is selected, the internal receiver datapath width will be set to two 10-bit bytes (20 bits). If 8B/10B decoding is not selected, the internal receiver datapath width will be set to two 8-bit bytes (16 bits).
	10	Sets the receiver application interface datapath width to a single 10-bit byte. Sets the internal receiver datapath width to two 10-bit bytes (20 bits).
	16	Sets the receiver application interface datapath width to two 8-bit bytes (16 bits). If 8B/10B decoding is selected, the internal receiver datapath width will be set to two 10-bit bytes (20 bits). If 8B/10B decoding is not selected, the internal receiver datapath width will be set to two 8-bit bytes (16 bits).
	20	Sets the receiver application interface datapath width to two 10-bit bytes (20 bits). Sets the internal receiver datapath width to two 10-bit bytes (20 bits).
	32	Sets the receiver application interface datapath width to four 8-bit bytes (32 bits). If 8B/10B decoding is selected, the internal receiver datapath width will be set to two 10-bit bytes (20 bits). If 8B/10B decoding is not selected, the internal receiver datapath width will be set to two 8-bit bytes (16 bits).
	40	Sets the receiver application interface datapath width to four 10-bit bytes (40 bits). Sets the internal receiver datapath width to two 10-bit bytes (20 bits).
Decoding	None	Data stream is passed with no conversion.
	None (MSB First)	Same as above but reorders bytes for applications expecting most significant byte first.
	8B/10B	Data stream is passed to an internal 8B/10B encoder prior to transmission.
	64B/66B	Data stream is passed through the 64B/66B gearbox and descrambler.
	64B/67B	Data stream is passed through the 64B/67B gearbox and descrambler.

Table 3-2: RX Settings (Cont'd)

Options	Description
Reference Clock	Select from the list the optimal reference clock frequency to be provided by the application. The XAUI example uses 156.25 MHz.
Use Oversampling	The Wizard supports oversampling for line rates between 600 Mb/s and 1300 Mb/s. For line rates of 120–600 Mb/s, this option is automatically selected and the check box is disabled. For line rates of 600–1300 Mb/s, the checkbox is enabled allowing optional selection of this feature. This option is not available for XAUI since the line rate exceeds the permissible range.
Use RXOVERSAMPLEERR Ports	Select this option to have the RXOVERSAMPLEERR signals from the transceiver available to the application. The XAUI example does not use this signal.
No RX	Selecting this option disables the RX path of the GTX transceiver. The transceiver will function as a transmitter only. The XAUI example design requires both TX and RX functionality.

Note: Options not used by the XAUI example are shaded.

Table 3-3: Additional Options

Option	Description
Use Dynamic Reconfiguration Port	Select this option to have the dynamic reconfiguration port signals available to the application. This feature is used by the XAUI example.

Table 3-4: 8B/10B Optional Ports

Option		Description
TX	TXBYPASS8B10B	2-bit wide port disables 8B/10B encoder on a per-byte basis. High-order bit affects high-order byte of datapath.
	TXCHARDISPMODE	2-bit wide ports control disparity of outgoing 8B/10B data. High-order bit affects high-order byte of datapath.
	TXCHARDISPVAL	
	TXKERR	2-bit wide port flags invalid K character codes as they are encountered. High-order bit corresponds to high-order byte of datapath.
TXRUNDISP	2-bit wide port indicates current running disparity of the 8B/10B encoder on a per-byte basis. High-order bit affects high-order byte of datapath.	
RX	RXCHARISCOMMA	2-bit wide port flags valid 8B/10B comma characters as they are encountered. High-order bit corresponds to high-order byte of datapath.
	RXCHARISK	2-bit wide port flags valid 8B/10B K characters as they are encountered. High-order bit corresponds to high-order byte of datapath.
	RXRUNDISP	2-bit wide port indicates current running disparity of the 8B/10B decoder on a per-byte basis. High-order bit corresponds to high-order byte of datapath.

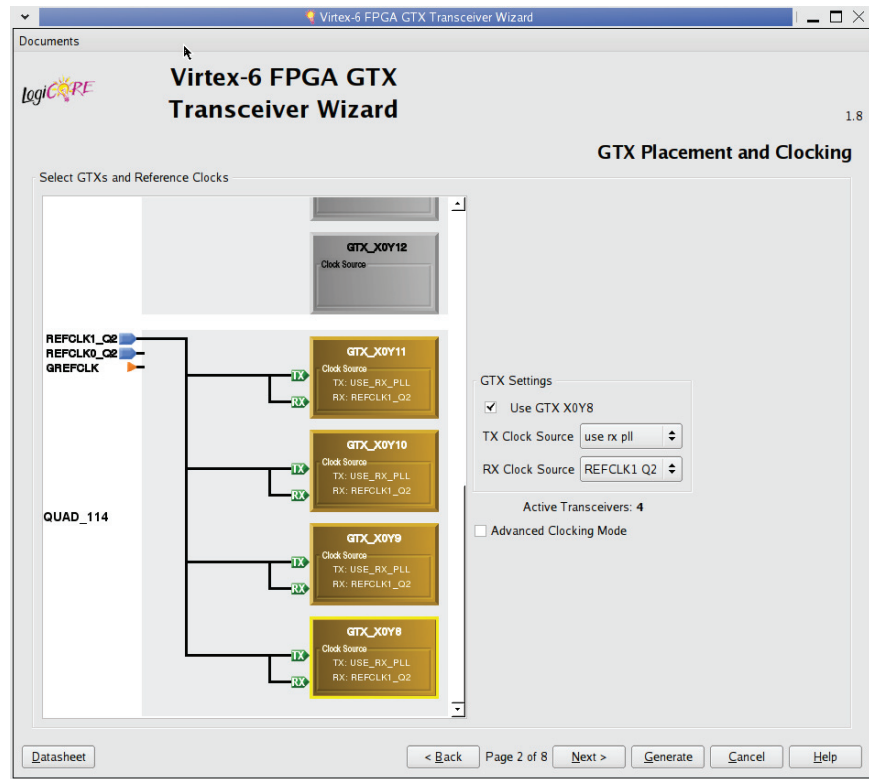
Note: Options not used by the XAUI example are shaded.

GTX Transceiver Placement and Clocking

Page 2 of the Wizard (Figure 3-8) allows you to determine the placement of the GTX transceivers and the reference clock source.

The number of available GTX transceivers appearing on this page depends on the selected target device and package. The XAUI example design uses four GTX transceivers.

Table 3-5, page 28 describes the GTX transceiver selection and reference clock options and Table 3-6, page 28 describes the reference clock source options.



UG516_c3_08_021011

Figure 3-8: GTX Placement and Clocking—Page 2

Table 3-5: Select Transceiver and Reference Clocks

Option	Description
GTX	Select the individual GTX transceivers by location to be used in the target design. The XAUI example requires four transceivers.
TXREFCLK Source	Determines the source for the reference clock signal provided to each selected GTX transceiver (see Table 3-6). Two differential clock signal input pin pairs, labeled REFCLK0 and REFCLK1 are provided for every four transceivers. The groups are labeled Q0 through Q4 starting at the bottom of the transceiver column. Each transceiver has access to the local signal group and one or two adjacent groups depending upon the transceiver position. The XAUI example uses the REFCLK1 signal from the group local to the four selected GTX transceivers (REFCLK1 Q0 option).
RXREFCLK Source	
Advanced Clocking Mode	This mode will bring the following ports to the top-level GTX transceiver wrapper: GREFCLKRX, GREFCLKTX, MGTREFCLKRX[1:0], MGTREFCLKTX[1:0], NORTHREFCLKRX[1:0], NORTHREFCLKTX[1:0], PERFCLKRX, PERFCLKTX, RXPLLREFSELDY[2:0], SOUTHREFCLKRX[1:0], SOUTHREFCLKTX[1:0], TXPLLREFSELDY[2:0].

Table 3-6: Reference Clock Source Options

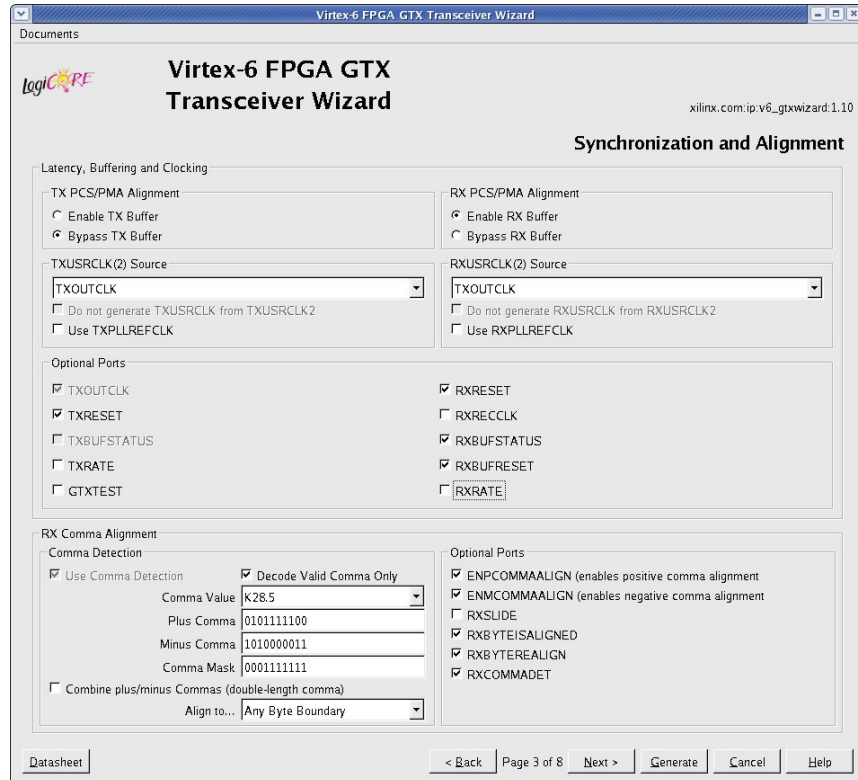
Option	Description
use rx pll	Available for TXREFCLK only. Internally connects the TXREFCLK input to the receiver PLL.
REFCLK0/1 Q0	GTX reference clock local to transceivers Y0-Y3.
REFCLK0/1 Q1	GTX reference clock local to transceivers Y4-Y7.
REFCLK0/1 Q2	GTX reference clock local to transceivers Y8-Y11.
REFCLK0/1 Q3	GTX reference clock local to transceivers Y12-Y15.
REFCLK0/1 Q4	GTX reference clock local to transceivers Y16-Y19.
REFCLK0/1 Q5	GTX reference clock local to transceivers Y20-Y23.
REFCLK0/1 Q6	GTX reference clock local to transceivers Y24-Y27.
REFCLK0/1 Q7	GTX reference clock local to transceivers Y28-Y31.
REFCLK0/1 Q8	GTX reference clock local to transceivers Y32-Y35.
GREFCLK	Reference clock driven by internal FPGA interconnect logic. Lowest performance option.

Synchronization and Alignment

Page 3 of the Wizard ([Figure 3-9](#)) allows you to control latency, buffering, and clocking of the transmitter and receiver. The RX comma alignment settings are also provided.

The TX PCS/PMA Alignment setting controls whether the TX buffer is enabled or bypassed. See [UG366](#), *Virtex-6 FPGA GTX Transceivers User Guide* for details on this setting. The XAUI example uses the TX phase alignment circuit.

The RX PCS/PMA Alignment setting controls whether the RX phase alignment circuit is enabled. The XAUI example does not use the RX phase alignment circuit.



UG516_c3_09_051011

Figure 3-9: Synchronization and Alignment—Page 3

Table 3-7 details the TXUSRCLK and RXUSRCLK source signal options.

Table 3-7: TXUSRCLK and RXUSRCLK Source

Option		Description
TX	TXOUTCLK	TXUSRCLK is driven by TXOUTCLK.
	Generate TXUSRCLK from TXUSRCLK2	Brings the TXUSRCLK input signal out to a port at the top level of the wrapper so it can be provided by the application. Optionally available when single-byte datapath width is used and TX buffer bypass is disabled. Not available for 2-byte datapath width. Mandatory with 4-byte datapath width. The XAUI example does not use this feature.
	Use TXPLLREFCLK	TXUSRCLK is driven by internally generated TXPLLREFCLK.
RX	TXOUTCLK	RXUSRCLK is driven by TXOUTCLK. This option is not available if the RX buffer is bypassed.
	RXRECCLK	RXUSRCLK is driven by RXRECCLK. This option is required if the RX buffer is bypassed.
	Generate RXUSRCLK from RXUSRCLK2	Brings the RXUSRCLK input signal out to a port at the top level of the wrapper so it can be provided by the application. Optionally available when single-byte datapath width is used without channel bonding. Not available for 2-byte datapath width. Mandatory with 4-byte datapath width. The XAUI example does not use this feature.
	Use RXPLLREFCLK	RXUSRCLK is driven by internally generated RXPLLREFCLK.

Note: Options not used by the XAUI example are shaded.

Table 3-8 shows the optional ports available for latency and clocking.

Table 3-8: Optional Ports

Option	Description
TXOUTCLK	Parallel clock signal generated by the GTX transceiver. This option is required when selected as an input to either TXUSRCLK or RXUSRCLK.
TXRESET	Active-High reset signal for the transmitter PCS logic.
TXBUFSTATUS	2-bit signal monitors the status of the TX elastic buffer. This option is not available when the TX buffer is bypassed.
TXRATE	2-bit signal controls the setting for the TX serial clock divider for low line rate support. This input port is used in combination with the TXPLL_DIVSEL_OUT attribute.
GTXTEST	Selecting this input port will OR the bit(1) of the input to this port with the double reset circuit output.
RXRESET	Active-High reset signal for the receiver PCS logic.
RXRECCLK	Recovered clock signal from the CDR logic. This option is required when selected as an input to RXUSRCLK.
RXBUFSTATUS	Indicates the condition of the RX elastic buffer. This option is not available when the RX buffer is bypassed.
RXBUFRESET	Active-High reset signal for the RX elastic buffer logic. This option is not available when the RX buffer is bypassed.
RXRATE	2-bit signal controls the setting for the RX serial clock divider for low line rate support. This input port is used in combination with the RXPLL_DIVSEL_OUT attribute.

Note: Options not used by the XAUI example are shaded.

Table 3-9 shows the settings for receive comma alignment.

Table 3-9: Comma Detection

Option		Description
Use Comma Detection		Enables receive comma detection. Used to identify comma characters and SONET framing characters in the data stream.
Decode Valid Comma Only		When receive comma detection is enabled, limits the detection to specific defined comma characters.
Comma Value		Select one of the standard comma patterns or User Defined to enter a custom pattern. The XAUI example uses K28.5.
Plus Comma		10-bit binary pattern representing the positive-disparity comma character to match. The rightmost bit of the pattern is the first bit to arrive serially. The XAUI example uses 0101111100 (K28.5).
Minus Comma		10-bit binary pattern representing the negative-disparity comma character to match. The rightmost bit of the pattern is the first bit to arrive serially. The XAUI example uses 1010000011 (K28.5).
Comma Mask		10-bit binary pattern representing the mask for the comma match patterns. A 1 bit indicates the corresponding bit in the comma patterns is to be matched. A 0 bit indicates <i>don't care</i> for the corresponding bit in the comma patterns. The XAUI example matches the lower seven bits (K28.5 OR K28.1 OR K28.7).
Combine plus/minus commas		Causes the two comma definition patterns to be combined into a single 20-bit pattern which must be contiguously matched in the data stream. The mask value remains 10 bits and is duplicated for the upper and lower 10-bit portions of the extended pattern. This option can be used to search for SONET framing character patterns. Not used by XAUI.
Align to...	Any Byte Boundary	When a comma is detected, the data stream is aligned using the comma pattern to the nearest byte boundary.
	Even Byte Boundaries	When a comma is detected, the data stream is aligned using the comma pattern to the nearest even byte boundary.
Optional Ports	ENPCOMMAALIGN	Active-High signal which enables the byte boundary alignment process when the plus comma pattern is detected.
	ENMCOMMAALIGN	Active-High signal which enables the byte boundary alignment process when the minus comma pattern is detected.
	RXSLIDE	Active-High signal that causes the byte alignment to be adjusted by one bit with each assertion. Takes precedence over normal comma alignment.
	RXBYTEISALIGNED	Active-High signal indicating that the parallel data stream is aligned to byte boundaries.
	RXBYTEREALIGN	Active-High signal indicating that byte alignment has changed with a recent comma detection. Data errors can occur with this condition.
	RXCOMMADET	Active-High signal indicating the comma alignment logic has detected a comma pattern in the data stream.

Note: Options not used by the XAUI example are shaded.

Preemphasis, Termination, and Equalization

Page 4 of the Wizard (Figure 3-10) allows you to set the preemphasis, termination, and equalization options.

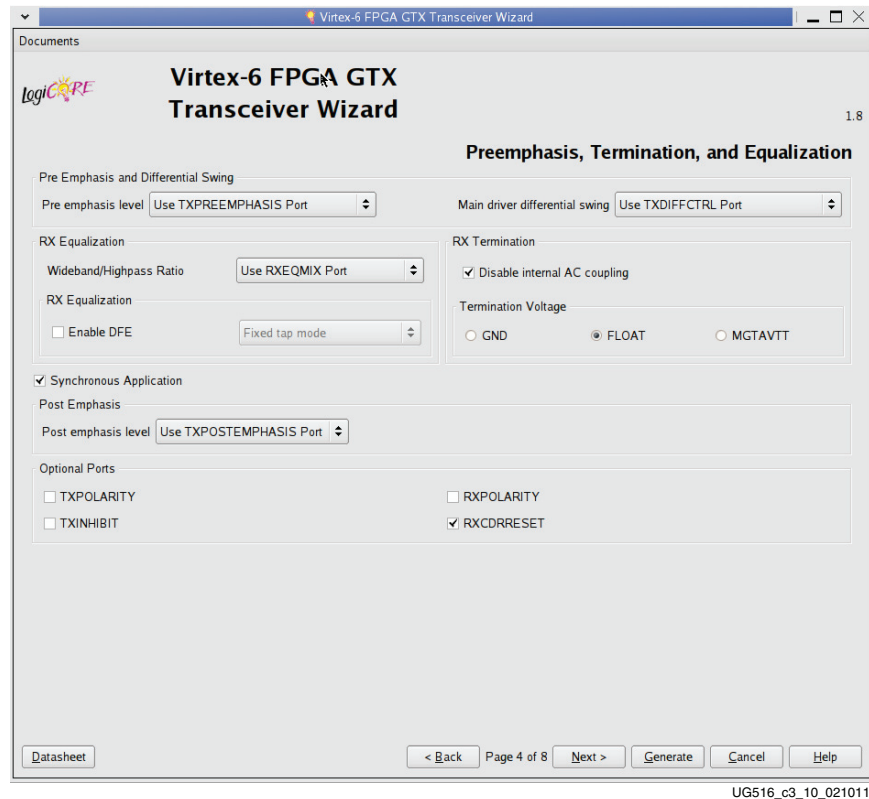


Figure 3-10: Preemphasis, Termination, and Equalization—Page 4

Table 3-10 details the preemphasis and differential swing settings.

Table 3-10: Preemphasis and Differential Swing

Option	Description
Preemphasis Level	Specifies the transmitter pre-cursor preemphasis level setting. Selecting the Use TXPREEMPHASIS port enables the optional TXPREEMPHASIS configuration port to dynamically set the preemphasis level. The XAUI example uses the TXPREEMPHASIS port to dynamically set the preemphasis level. See UG366 , <i>Virtex-6 FPGA GTX Transceivers User Guide</i> for a table mapping TXPREEMPHASIS value settings to preemphasis levels.
Main Driver Differential Swing	Specifies the differential swing level for the transmitter main driver. Can also be set to zero. Selecting Use TXDIFFCTRL port enables the optional TXDIFFCTRL configuration port to dynamically set the swing level. The XAUI example uses the TXDIFFCTRL port to dynamically set the swing level. See UG366 , <i>Virtex-6 FPGA GTX Transceivers User Guide</i> for a table mapping TXDIFFCTRL value settings to differential swing levels.

Table 3-11 describes the RX Equalization settings.

Table 3-11: RX Equalization

Option	Description
Wide Band/High Pass Ratio	Controls the proportion of signal derived from the high pass filter and from the unfiltered receiver (wide band) when RX equalization is active. Select a percentage ratio from the drop down list. The XAUI protocol uses the RXEQMIX port to dynamically set the RX Equalizer.
Enable DFE	Enables the decision feedback equalizer and brings out the required ports. See UG366 , <i>Virtex-6 FPGA GTX Transceivers User Guide</i> for details on the decision feedback equalizer. The XAUI example leaves this option disabled.
DFE Mode	Sets the operational mode of the decision feedback equalizer. this version supports fixed tap mode only.

Note: Options not used by the XAUI example are shaded.

Table 3-12 describes the RX termination settings.

Table 3-12: RX Termination

Option	Description
Disable Internal AC Coupling	Bypasses the internal AC coupling capacitor. Use this option for DC coupling applications or for external AC coupling.
Termination Voltage	Selecting GND grounds the internal termination network. Selecting FLOAT isolates the network. Selecting MGTAVTT applies an internal voltage reference source to the termination network. The XAUI example uses the FLOAT setting.

Check the Synchronous Application check box for TX/RX PPM offset values less than ± 100 ppm. For values greater than ± 100 ppm and less than ± 2000 ppm, do not check the Synchronous Application check box.

The post-emphasis level setting specifies the transmitter post-cursor preemphasis level setting. See [UG366](#), *Virtex-6 FPGA GTX Transceivers User Guide* for details.

Table 3-13 lists the optional ports available on this page.

Table 3-13: Optional Ports

Option	Description
TXPOLARITY	Active-High signal to invert the polarity of the transmitter output.
TXINHIBIT	Active-High signal forces transmitter output to steady state.
RXPOLARITY	Active-High signal inverts the polarity of the receive data signal.
RXCDRRESET	Active-High reset signal causes the CDR logic to unlock and return to the shared PLL frequency.

Note: Options not used by the XAUI example are shaded.

PCI Express, SATA, OOB, PRBS, and RX Loss of Sync

Page 5 of the Wizard (Figure 3-11) allows you to configure the receiver for PCI Express® and Serial ATA (SATA) features. In addition, configuration options for the RX out-of-band signal (OOB), PRBS detector, and the loss of sync state machine settings are provided.

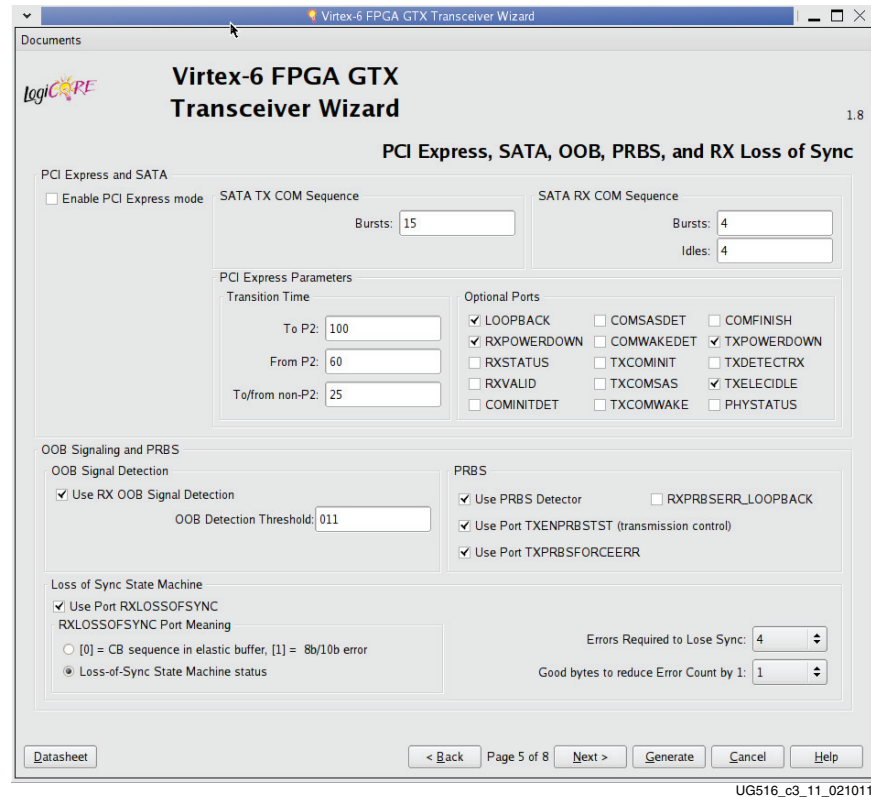


Figure 3-11: PCI Express, SATA, OOB, PRBS, and RX Loss of Sync—Page 5

Table 3-14 details the PCI Express and SATA configuration options.

Table 3-14: PCI Express and Serial ATA Options

Options		Description
Enable PCI Express mode		Selecting this option enables certain operations specific to PCI Express, including enabling options for PCI Express powerdown modes and PCIe® channel bonding. This option should be activated whenever the transceiver is used for PCI Express.
SATA TX COM Sequence	Bursts	Integer value between 0 and 15 indicating the number of busts to define a TX COM sequence.
SATA RX COM Sequence	Bursts	Integer value between 0 and 7 indicating the number of burst sequences to declare a COM match. This value defaults to 4, which is the burst count specified in the SATA specification for COMINIT, COMRESET, and COMWAKE.
	Idles	Integer value between 0 and 7 indicating the number of idle sequences to declare a COM match. Each idle is an OOB signal with a length that matches COMINIT/COMRESET or COMWAKE.

Note: Options not used by the XAUI example are shaded.

Table 3-15 details the receiver PCI Express configuration options.

Table 3-15: PCI Express Parameters

Option		Description
Transition Time	To P2	Integer value between 0 and 65,535. Sets a counter to determine the transition time to the P2 power state for PCI Express. See UG366, Virtex-6 FPGA GTX Transceivers User Guide for details on determining the time value for each count. The XAUI example does not require this feature and uses the default setting of 100.
	From P2	Integer value between 0 and 65,535. Sets a counter to determine the transition time from the P2 power state for PCI Express. See UG366, Virtex-6 FPGA GTX Transceivers User Guide for details on determining the time value for each count. The XAUI example does not require this feature and uses the default setting of 60.
	To/From non-P2	Integer value between 0 and 65,535. Sets a counter to determine the transition time to or from power states other than P2 for PCI Express. See UG366, Virtex-6 FPGA GTX Transceivers User Guide for details on determining the time value for each count. The XAUI example does not require this feature and uses the default setting of 25.
Optional Ports	LOOPBACK	3-bit signal to enable the various data loopback modes for testing.
	RXPOWERDOWN	2-bit PCI Express compliant receiver powerdown control signal.
	RXSTATUS	3-bit receiver status signal. The encoding of this signal is dependent on the setting of RXSTATUS encoding format.
	RXVALID	Active-High, PCI Express RX OOB/beacon signal. Indicates symbol lock and valid data on RXDATA and RXCHARISK[3:0].
	TXCOMSTART	Active-High signal initiates the transmission of the SATA COM sequence selected by the setting of TXCOMTYPE. This option is not available if RXSTATUS encoding format is set to PCI Express. Activate the RXSTATUS optional port when using this option.
	TXCOMTYPE	Active-High signal selects SATA COMWAKE sequence when asserted, otherwise selects COMINIT. The sequence is initiated upon assertion of TXCOMSTART. This option is not available if RXSTATUS encoding format is set to PCI Express.
	TXPOWERDOWN	2-bit PCI Express compliant transmitter powerdown control signal.
	TXDETECTRX	PIPE interface for PCI Express specification-compliant control signal. Activates the PCI Express receiver detection feature. Function depends on the state of TXPOWERDOWN, RXPOWERDOWN, TXELECIDLE, TXCHARDISPMODE, and TXCHARDISPVAL. This port is not available if RXSTATUS encoding format is set to SATA.
	TXELECIDLE	Drives the transmitter to an electrical idle state (no differential voltage). In PCI Express mode this option is used for electrical idle modes. Function depends on the state of TXPOWERDOWN, RXPOWERDOWN, TXELECIDLE, TXCHARDISPMODE, and TXCHARDISPVAL.
	PHYSTATUS	PCI Express receive detect support signal. Indicates completion of several PHY functions.

Note: Options not used by the XAUI example are shaded.

Table 3-16 shows the OOB signal detection options.

Table 3-16: OOB Signal Detection

Option	Description
Use RX OOB Signal Detection	Enables the internal out-of-band signal detector (OOB). OOB signal detection is used for PCIe and SATA.
OOB Detection Threshold	Specifies a binary value representing a differential receive signal voltage level. Valid values are 110 and 111, with 111 recommended. When the signal drops below this level, it is determined to be an OOB signal. This option is not available if the Use RX OOB Signal Detection option is not selected. See UG366 , <i>Virtex-6 FPGA GTX Transceivers User Guide</i> for more information about the OOB detection threshold levels.

Table 3-17 details the PRBS settings.

Table 3-17: PRBS

Option	Description
Use PRBS Detector	Enables the internal pseudo random bitstream sequence detector (PRBS). This feature can be used by an application to implement a built-in self-test.
Use Port TXENPRBSTST	Enables the PRBS transmission control port. This port is used to start/stop PRBS generation.
Use Port TXPRBSFORCEERR	Enables the PRBS force error control port. This port controls the insertion of errors into the bitstream.
RXPRBSERR_LOOPBACK	Select this option to loop back RXPRBSERR bit to TXPRBSFORCEERR of the same GTX transceiver.

Note: Options not used by the XAUI example are shaded.

Table 3-18 describes the loss of sync state machine settings.

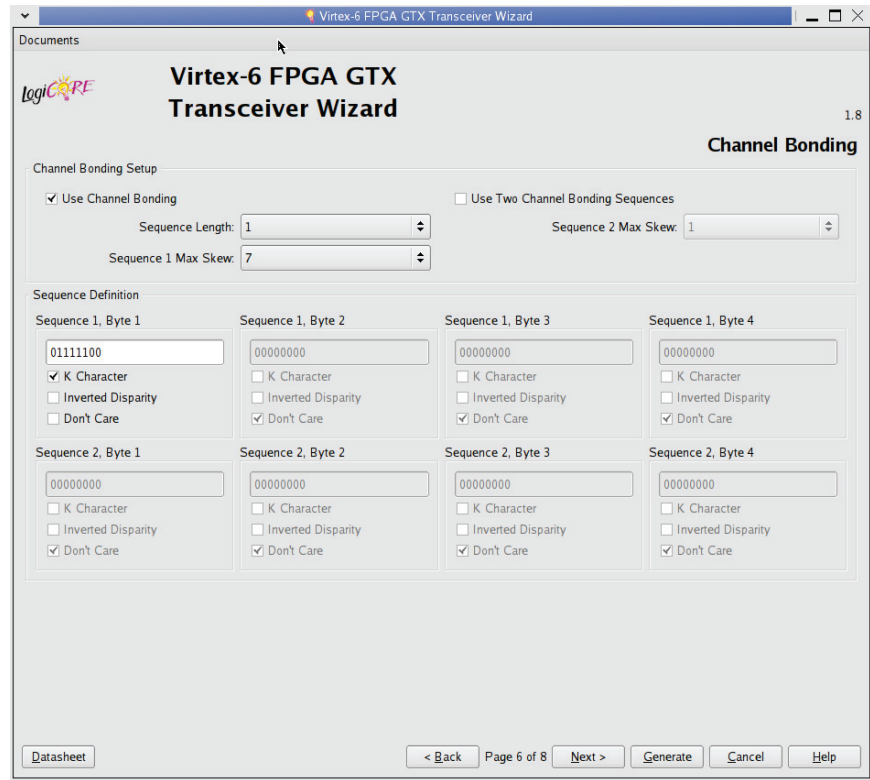
Table 3-18: Loss of Sync State Machine

Option	Description				
Use Port RXLOSSOFSYNC	2-bit multi-purpose status port. The meaning of the bits is determined by the settings below.				
RXLOSSOFSYNC Port Meaning	<table border="1"> <tbody> <tr> <td>[0] = CB Sequence in Elastic Buffer [1] = 8B/10B Error</td> <td>Bit 0 of the RXLOSSOFSYNC status port indicates a channel bonding sequence is present in the receive elastic buffer. Bit 1 indicates the detection of an 8B/10B coding error.</td> </tr> <tr> <td>Loss of Sync State Machine Status</td> <td>Bit 0 of the RXLOSSOFSYNC status port indicates sync state is active due to channel bonding or realignment. Bit 1 indicates sync lost due to invalid characters or reset.</td> </tr> </tbody> </table>	[0] = CB Sequence in Elastic Buffer [1] = 8B/10B Error	Bit 0 of the RXLOSSOFSYNC status port indicates a channel bonding sequence is present in the receive elastic buffer. Bit 1 indicates the detection of an 8B/10B coding error.	Loss of Sync State Machine Status	Bit 0 of the RXLOSSOFSYNC status port indicates sync state is active due to channel bonding or realignment. Bit 1 indicates sync lost due to invalid characters or reset.
[0] = CB Sequence in Elastic Buffer [1] = 8B/10B Error	Bit 0 of the RXLOSSOFSYNC status port indicates a channel bonding sequence is present in the receive elastic buffer. Bit 1 indicates the detection of an 8B/10B coding error.				
Loss of Sync State Machine Status	Bit 0 of the RXLOSSOFSYNC status port indicates sync state is active due to channel bonding or realignment. Bit 1 indicates sync lost due to invalid characters or reset.				
Errors Required to Lose Sync	Integer value between 4 and 512 representing the count of invalid characters received, above which sync is determined to be lost. The XAUI example uses 4.				
Good bytes to reduce Error Count by 1	Integer value between 1 and 128 representing the number of consecutive valid characters needed to cancel out the appearance of one invalid character. The XAUI example uses 1.				

Note: Options not used by the XAUI example are shaded.

Channel Bonding Sequence

Page 6 of the Wizard (Figure 3-12) allows you to define the channel bonding sequence(s). Table 3-19, page 38 describes the channel bonding setup options. Table 3-20, page 38 describes the sequence definition settings.



UG516_c3_12_021011

Figure 3-12: Channel Bonding—Page 6

Table 3-19: Channel Bonding Setup

Option	Description
Use Channel Bonding	Enables receiver channel bonding logic using unique character sequences. When recognized, these sequences allow for adding or deleting characters in the receive buffer to byte-align multiple data transceivers.
Sequence Length	Select from the drop down list the number of characters in the unique channel bonding sequence. The XAUI example uses 1.
Sequence 1 Max Skew	Select from the drop down list the maximum skew in characters that can be handled by channel bonding. Must always be less than half the minimum distance between channel bonding sequences. The XAUI example uses 7.
Use Two Channel Bonding Sequences	Activates the optional second channel bonding sequence. Detection of either sequence triggers channel bonding.
Sequence 2 Max Skew	Select from the drop down list the maximum skew in characters that can be handled by channel bonding. Must always be less than half the minimum distance between channel bonding sequences.

Note: Options not used by the XAUI example are shaded.

Table 3-20: Channel Bonding and Clock Correction Sequences

Option	Description
Byte (Symbol)	Set each symbol to match the pattern the protocol requires. The XAUI sequence length is 8 bits. 01111100 is used for channel bonding. 00011100 is used for clock correction. The other symbols are disabled because the sequence length is set to 1.
K Character	This option is available when 8B/10B decoding is selected. When checked, as is the case for XAUI, the symbol is an 8B/10B K character.
Inverted Disparity	Some protocols with 8B/10B decoding use symbols with deliberately inverted disparity. This option should be checked when such symbols are expected in the sequence.
Don't Care	Multiple-byte sequences can have wild card symbols by checking this option. Unused bytes in the sequence automatically have this option set.

Note: Options not used by the XAUI example are shaded.

Clock Correction Sequence

Page 7 of the Wizard (Figure 3-13) allows you to define the clock correction sequence. Table 3-21, page 39 describes the clock correction setup parameters. See Table 3-20, page 38 for the sequence definition settings.

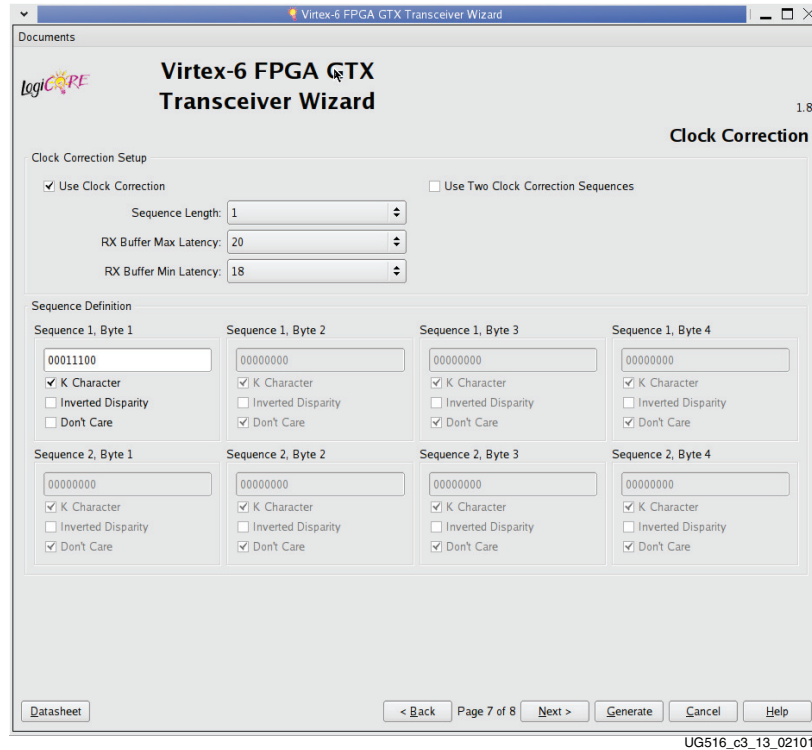


Figure 3-13: Clock Correction—Page 7

Table 3-21: Clock Correction Setup

Option	Description
Use Clock Correction	Enables receiver clock correction logic using unique character sequences. When recognized, these sequences allow for adding or deleting characters in the receive buffer to prevent buffer underflow/overflow due to small differences in the transmit/receive clock frequencies.
Sequence Length	Select from the drop down list the number of characters (subsequences) in the unique clock correction sequence. The XAUI example uses 1.
RX Buffer Max Latency	Select from the drop down list the maximum number of characters to permit in the receive buffer before clock correction attempts to delete incoming clock correction sequences. Also determines the maximum latency of the receive buffer in RXUSRCLK cycles. The XAUI example uses 20.
RX Buffer Min Latency	Select from the drop down list the minimum number of characters to permit in the receive buffer before clock correction attempts to add extra clock correction sequences to the receive buffer. Also determines the minimum latency of the receive buffer in RXUSRCLK cycles. The XAUI example uses 18.
Use Two Clock Correction Sequences	Activates the optional second clock correction sequence. Detection of either sequence triggers clock correction.

Note: Options not used by the XAUI example are shaded.

Summary

Page 8 of the Wizard (Figure 3-14) provides a summary of the selected configuration parameters. After reviewing the settings, click **Generate** to exit and generate the wrapper.



Figure 3-14: Summary—Page 8

Quick Start Example Design

Overview

This chapter introduces the example design that is included with the Virtex®-6 FPGA GTX transceiver wrappers. The example design demonstrates how to use the wrappers and demonstrates some of the key features of the GTX transceiver. For detailed information about the example design, see [Chapter 5, Detailed Example Design](#).

Functional Simulation of the Example Design

The Virtex-6 FPGA GTX Transceiver Wizard provides a quick way to simulate and observe the behavior of the wrapper using the provided example design and script files.

To simulate simplex designs, the SIMPLEX_PARTNER environment variable should be set to the path of the complementary core generated to test the simplex design. For example, if a design is generated with RX OFF, a simplex partner design with RX enabled is needed to simulate the device under test (DUT). The SIMPLEX_PARTNER environment variable should be set to the path of the RX enabled design. The name of the simplex partner should be the same as the name of the DUT with a prefix of tx or rx as applicable. In the current example, the name of the simplex partner design would be prefixed with rx.

Using ModelSim

Prior to simulating the wrapper with ModelSim, the functional (gate-level) simulation models must be generated. All source files in the following directories must be compiled to a single library as shown in [Table 4-1](#). For instructions on how to compile ISE® simulation libraries, see the latest version of [UG626: Synthesis and Simulation Design Guide](#).

Table 4-1: Required ModelSim Simulation Libraries

HDL	Library	Source Directories
Verilog	UNISIMS_VER	<Xilinx dir>/verilog/src/unisims <Xilinx dir>/secureip/mti
VHDL	UNISIM	<Xilinx dir>/vhdl/src/unisims/primitive <Xilinx dir>/secureip/mti

The Wizard provides a command line script for use within ModelSim. To run a VHDL or Verilog ModelSim simulation of the wrapper, use the following instructions:

1. Launch the Modelsim simulator and set the current directory to:


```
<project_directory>/<component_name>/simulation/functional
```

2. Set the MTI_LIBS variable:


```
modelsim> setenv MTI_LIBS <path to compiled libraries>
```
3. Launch the simulation script:


```
modelsim> do simulate_mti.do
```

The ModelSim script compiles the example design and testbench, and adds the relevant signals to the wave window.

Using the ISE Simulator

When using the ISE Simulator (ISim), the required simulation device libraries are precompiled, and are updated automatically when service packs and IP updates are installed. There is no need to run CompXlib to compile libraries, or to manually download updated libraries.

Table 4-2: Required ISim Simulation Libraries

HDL	Library	Source Directories
Verilog	UNISIMS_VER	<Xilinx dir>/verilog/hdp/<OS>/unisims_ver
VHDL	UNISIM	<Xilinx dir>/vhdl/hdp/<OS>/unisim

Note: OS refers to the following operating systems: lin, lin64, nt, nt64.

The Wizard also generates a perl script for use with ISim. To run a VHDL or Verilog simulation of the wrapper, use the following instructions:

1. Set the current directory to


```
<project_directory>/<component_name>/simulation/functional
```
2. Launch the simulation script:


```
prompt> simulate_isim.sh
```

The ISim script compiles the example design and testbench, and adds the relevant signals to the wave window.

Implementing the Example Design

When all of the parameters are set as desired, clicking **Generate** creates a directory structure under the provided Component Name. Wrapper generation proceeds and the generated output populates the appropriate subdirectories.

The directory structure for the XAUI example is provided in [Chapter 5, Detailed Example Design](#).

After wrapper generation is complete, the results can be tested in hardware. The provided example design incorporates the wrapper and additional blocks allowing the wrapper to be driven and monitored in hardware. The generated output also includes several scripts to assist in running the software.

From the command prompt, navigate to the project directory and type the following:

For Windows

```
> cd xaui_wrapper\implement
> implement.bat
```

For Linux

```
% cd xau_i_wrapper/implement
% implement.sh
```

Note: Substitute *Component Name* string for `xau_i_wrapper`.

These commands execute a script that synthesizes, builds, maps, places, and routes the example design and produces a bitmap file. The resulting files are placed in the `implement/results` directory.

Netlist Simulation of the Example Design

The Virtex-6 FPGA GTX Transceiver Wizard (hereinafter called the Wizard) provides a script to perform simulations on the routed netlist of the example design.

Note: Timing checks are not enabled.

Using ModelSim

Prior to performing netlist simulation with ModelSim, the generated design should successfully pass through implementation. All source files in the following directories must be compiled to a single library, as shown in [Table 4-3](#). See the Synthesis and Simulation Design Guide for the ISE tools, v13.3 available in the ISE software documentation, for instructions on how to compile ISE simulation libraries.

Table 4-3: Required ModelSim Netlist Simulation Libraries

HDL	Library	Source Directories
Verilog	SIMPRIMS_VER	<Xilinx dir>/verilog/src/simprims <Xilinx dir>/secureip/mti
VHDL	SIMPRIM	<Xilinx dir>/vhdl/src/simprims/primitive <Xilinx dir>/secureip/mti

The Wizard provides a command line script for use within ModelSim. To run a VHDL or Verilog ModelSim simulation of the wrapper, use the following instructions:

1. Launch the ModelSim simulator and set the current directory to:


```
<project_directory>/<component_name>/simulation/netlist
```
2. Set the MTI_LIBS variable:


```
modelsim> setenv MTI_LIBS <path to compiled libraries>
```
2. Launch the simulation script:


```
modelsim> do simulate_mti.do
```

The ModelSim script compiles and simulates the routed netlist of the example design and testbench.

Detailed Example Design

This chapter provides detailed information about the example design, including a description of files and the directory structure generated by the CORE Generator™ tool, the purpose and contents of the provided scripts, the contents of the example HDL wrappers, and the operation of the demonstration testbench.

Directory and File Structure

- 📁 **<project directory>**
Top-level project directory; name is user-defined
 - 📁 **<project directory>/<component name>**
Wizard release notes file
 - 📁 **<component name>/doc**
Product documentation
 - 📁 **<component name>/example design**
Verilog and VHDL design files
 - 📁 **<component name>/implement**
Implementation script files
 - 📁 **implement/results**
Results directory, created after implementation scripts are run, and contains implement script results
 - 📁 **<component name>/simulation**
Simulation scripts
 - 📁 **simulation/functional**
Functional simulation files
 - 📁 **simulation/netlist**
Netlist simulation files

Directory and File Contents

The Virtex®-6 FPGA GTX Transceiver Wizard directories and their associated files are defined in the following sections.

<project directory>

The <project directory> contains all the CORE Generator tool's project files.

Table 5-1: Project Directory

Name	Description
<component_name>.v[hd]	Main GTX transceiver wrapper. Instantiates individual GTX transceiver wrappers. For use in the target design.
<component_name>.[veo vho]	GTX transceiver wrapper files instantiation templates. Includes templates for the GTX transceiver wrapper module, the IBUFDS_GTXE1, and essential GTX transceiver support modules (such as TX_SYNC).
<component_name>.xco	Log file from the CORE Generator tool describing which options were used to generate the GTX transceiver wrapper. An XCO file is generated by the CORE Generator tool for each Wizard wrapper that it creates in the current project directory. An XCO file can also be used as an input to the CORE Generator tool.
<component_name>_gtx.v[hd]	Individual GTXE1 transceiver wrapper to be instantiated in the main GTX transceiver wrapper. Instantiates the selected GTXE1 transceivers with settings for the selected protocol.

[Back to Top](#)

<project directory>/<component name>

The <component name> directory contains the README file provided with the Wizard, which might include last-minute changes and updates.

Table 5-2: GTX Transceiver Wrapper Component Name

Name	Description
<project_dir>/<component_name>	
v6_gtxwizard_readme.txt	README file for the Wizard.
<component_name>.pf	Protocol description for the selected protocol from the Wizard.

[Back to Top](#)

<component name>/doc

The doc directory contains the PDF documentation provided with the Wizard.

Table 5-3: Doc Directory

Name	Description
<project_dir>/<component_name>/doc	
ug516_v6_gtxwizard.pdf	<i>LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard v1.11 User Guide</i>

[Back to Top](#)

<component name>/example design

The example design directory contains the example design files provided with the Wizard wrapper.

Table 5-4: Example Design Directory

Name	Description
<project_dir>/<component_name>/example_design	
frame_check.v[hd]	Frame-check logic to be instantiated in the example design.
frame_gen.v[hd]	Frame-generator logic to be instantiated in the example design.
gtx_attributes.ucf	Constraints file containing the GTX transceiver attributes generated by the GTX transceiver Wizard GUI settings.
tx_sync.v[hd]	TX sync logic module to be instantiated in the example design. Performs phase synchronization for all active TX datapaths. Available for use in the target design.
<component_name>_top.ucf	Constraint file for mapping the GTX transceiver wrapper example design onto a Virtex-6 device.
<component_name>_top.v[hd]	Top-level example design. Contains the GTX transceiver wrapper, reset logic, and instantiations for frame generator, frame-checker, and TX sync logic. Also contains definitions for test frame data and ChipScope™ Pro tools module instantiation. See Figure 3-3, page 18 .

[Back to Top](#)

<component name>/implement

The implement directory contains the implementation script files provided with the Wizard wrapper.

Table 5-5: Implement Directory

Name	Description
<project_dir>/<component_name>/implement	
chipscope_project.cpj	ChipScope Pro tools project file.
data_vio.ngc	Netlist of the design generated by the ChipScope Pro Virtual Input/Output (VIO) Wizard.
icon.ngc	Netlist of the design generated by the ChipScope Pro Integrated Controller (ICON) Wizard.
ila.ngc	Netlist of the design generated by the ChipScope Pro Integrated Logic Analyzer (ILA) Wizard.
implement.bat	A Windows batch file that processes the example design through the tool flow.
implement.sh	A Linux shell script that processes the example design through the tool flow.
implement_synplify.bat	A Windows batch file that processes the example design through Synplify synthesis and the tool flow.
implement_synplify.sh	A Linux shell script that processes the example design through Synplify synthesis and the tool flow.
synplify.prj	Synplify project file for the example design.
xst.prj	The XST project file for the example design. The file lists all of the source files to be synthesized.
xst.scr	The XST script file for the example design that is used to synthesize the Wizard. It is called from the implement script described above.
planAhead_ise.bat	A windows batch file that processes the example design through PlanAhead™ software-based ISE® design tools flow.
planAhead_ise.sh	A Linux shell script that processes the example design through PlanAhead software-based ISE design tools flow.
planAhead_ise.tcl	A TCL file that contains tool settings and file list.

[Back to Top](#)

implement/results

The results directory is created by the implement script, after which the implement script results are placed in the results directory.

Table 5-6: UCF Directory

Name	Description
<project_dir>/<component_name>/implement/results	
Implement script result files.	

[Back to Top](#)

<component name>/simulation

The simulation directory contains the simulation scripts provided with the Wizard wrapper.

Table 5-7: Simulation Directory

Name	Description
<project_dir>/<component_name>/simulation	
demo_tb.v[hd]	Testbench to perform functional simulation of the provided example design. See Functional Simulation of the Example Design, page 41 .
sim_reset_mgt_model.vhd	Reset module for VHDL required for emulating the GSR pulse at the beginning of functional simulation to correctly reset the VHDL MGT smart model.
demo_tb_imp.v[hd]	Testbench to perform netlist simulation of the provided example design. See Netlist Simulation of the Example Design, page 43 .

[Back to Top](#)

simulation/functional

The functional directory contains functional simulation scripts provided with the Wizard wrapper.

Table 5-8: Functional Directory

Name	Description
<project_dir>/<component_name>/simulation/functional	
simulate_isim.sh	ISE simulator (ISim) simulation script.
simulate_mti.do	ModelSim simulation script.
simulate_ncsim.sh	Linux script for running simulation using Cadence Incisive Enterprise Simulator (IES).
simulate_vcs.sh	Linux script for running simulation using Synopsys Verilog Compiler Simulator (VCS) and VCS MX.
ucli_command.key	Command file for VCS simulator.

Table 5-8: Functional Directory (Cont'd)

Name	Description
vcs_session.tcl	Script for adding GTX transceiver wrapper signals to VCS wave viewer.
wave_isim.tcl	Script for adding GTX transceiver wrapper signals to the ISim wave viewer.
wave_mti.do	Script for adding GTX transceiver wrapper signals to the ModelSim wave viewer.
wave_ncsim.sv	Script for adding GTX transceiver wrapper signals to the Cadence IES wave viewer.

[Back to Top](#)

simulation/netlist

The netlist directory contains netlist simulation scripts provided with the Wizard wrapper.

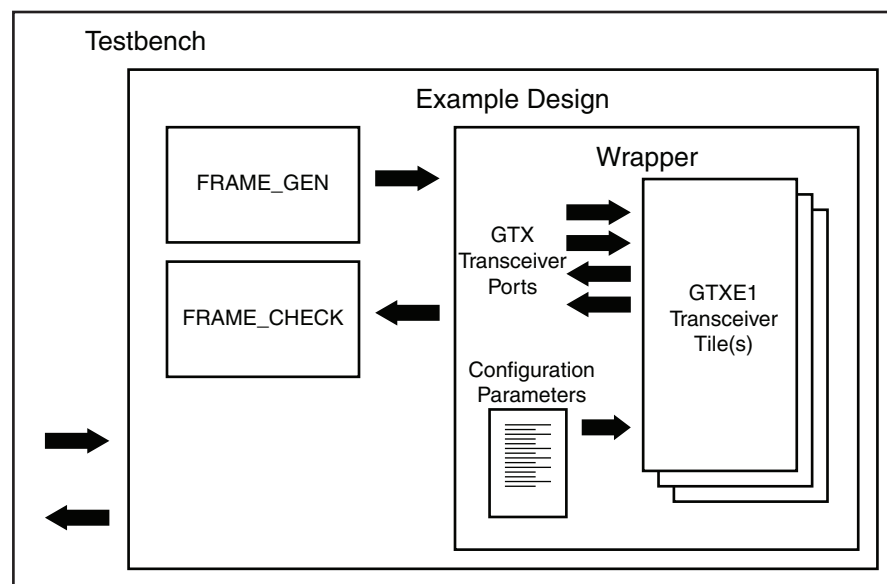
Table 5-9: Netlist Directory

Name	Description
simulate_mti.do	ModelSim netlist simulation script.

[Back to Top](#)

Example Design Description

The example design that is delivered with the wrappers helps Wizard designers understand how to use the wrappers and GTX transceivers in a design. The example design is shown in [Figure 5-1](#).



UG516_c5_01_021011

Figure 5-1: Diagram of Example Design and Testbench

The example design connects a frame generator and a frame checker to the wrapper. The frame generator transmits an incrementing counting pattern while the frame checker monitors the received data for correctness. The frame generator counting pattern is stored in the block RAM. This pattern can be easily modified by altering the parameters in the frame generator instantiation. The frame checker contains the same pattern in the block RAM and compares it with the received data. An error counter in the frame checker keeps a track of how many errors have occurred.

If comma alignment is enabled, the comma character will be placed within the counting pattern. Similarly, if channel bonding is enabled, the channel bonding sequence would be interspersed within the counting pattern. The frame check works by first scanning the received data for the START_OF_PACKET_CHAR. In 8B/10B designs, this is the comma alignment character. After the START_OF_PACKET_CHAR has been found, the received data will continuously be compared to the counting pattern stored in the block RAM at each RXUSRCLK2 cycle. After comparison has begun, if the received data ever fails to match the data in the block RAM, checking of receive data will immediately stop, an error counter will be incremented and the frame checker will return to searching for the START_OF_PACKET_CHAR.

For 64B/66B and 64B/67B example designs, the frame generator has scrambler logic while the frame checker has descrambler and block synchronization logic.

If the TX buffer is bypassed, the TX_SYNC module is instantiated in the example design and connected to the wrapper. The module performs the TX phase alignment procedure outlined in [UG366](#), *Virtex-6 FPGA GTX Transceivers User Guide*. Similarly, if the RX buffer is bypassed, the RX_SYNC module is instantiated in the example design and connected to the wrapper. The RX_SYNC module demonstrates the RX phase alignment procedure outlined in [UG366](#), *Virtex-6 FPGA GTX Transceivers User Guide*.

The example design also demonstrates how to properly connect clocks to GTX transceiver ports TXUSRCLK, TXUSRCLK2, RXUSRCLK, and RXUSRCLK2. Properly configured mixed mode clock managers (MMCM) wrappers are also provided if they are required to generate user clocks for the instantiated GTX transceivers.

The example design can be synthesized using XST or Synplify Pro, implemented with ISE software and then observed in hardware using the ChipScope Pro tools. RX output ports such as RXDATA can be observed on the ChipScope Pro ILA core while input ports can be controlled from the ChipScope Pro VIO core. A ChipScope Pro tools project file is also included with each example design.

For the example design to work properly in simulation or in hardware, both the transmit and receive side need to be configured with the same encoding, and datapath width in the GUI.

Example Design Hierarchy

The hierarchy for the design used in this example is:

```

EXAMPLE_TB
|___EXAMPLE_MGT_TOP
|   |___XAUI_WRAPPER
|   |   |___XAUI_WRAPPER_GTX (1 per transceiver)
|   |
|   |___TX_SYNC (1 per transceiver)
|   |___FRAME_GEN (1 per transceiver)
|   |___FRAME_CHECK (1 per transceiver)
|   (optional Chipscope support)
|   |___shared_vio

```

```
|__icon  
|__data_vio (TX)  
|__data_vio (RX)  
|__ila      (RX)
```