

# LogiCORE IP 7 Series FPGAs Transceivers Wizard v1.5

## *User Guide*

UG769 (v3.0) 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 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
03/01/11	1.0	Initial Xilinx release.
06/22/11	2.0	Wizard 1.4 release. In Chapter 1, Introduction, revised <a href="#">Features, Supported Devices, Provided with the Wizard, and Related Xilinx Documents</a> . In Chapter 2, Installing the Wizard, revised <a href="#">Tools and System Requirements</a> . In Chapter 3, Running the Wizard, modified title and content of <a href="#">Structure of the Transceiver Wrapper, Example Design, and Testbench</a> and <a href="#">Example Design—XAUI Configuration</a> . Added PPM Offset and Periodicity of the CC Sequence options to <a href="#">Table 3-23</a> , and deleted RX Buffer Max Latency and RX Buffer Min Latency from <a href="#">Table 3-23</a> . In Chapter 4, Quick Start Example Design, updated <a href="#">Table 4-1</a> . Added <a href="#">Using ChipScope Pro Cores with the Wizard</a> . In Chapter 5, Detailed Example Design, added <a href="#">gt_rom_init_tx.dat</a> and <a href="#">gt_rom_init_rx.dat</a> to <a href="#">Table 5-4</a> . Added <a href="#">chipscope_project.cpj</a> , <a href="#">data_vio.ngc</a> , <a href="#">icon.ngc</a> , <a href="#">ila.ngc</a> and PlanAhead software support to <a href="#">Table 5-5</a> . Added paragraph about ChipScope Pro tools to <a href="#">Example Design Description</a> .

Date	Version	Revision
10/19/11	3.0	<p>Wizard 1.5 release.</p> <p><a href="#">Chapter 1</a>: Updated <a href="#">About the Wizard</a>.</p> <p><a href="#">Chapter 2</a>: Added ISim 13.3 to <a href="#">Simulation</a>.</p> <p><a href="#">Chapter 3</a>: In <a href="#">Table 3-5</a>, renumbered Q4 through Q9 as Q3 through Q8. Updated <a href="#">Figure 3-8</a>. Updated row shading and descriptions in <a href="#">Table 3-6</a> and <a href="#">Table 3-7</a>. In <a href="#">Table 3-17</a>, updated row shading, removed RXCDRRESET, and added RXQPISENN, RXQPISENP, and RXQPIEN. Updated row shading in <a href="#">Table 3-20</a>. Removed shading from RX rows in <a href="#">Table 3-9</a>.</p> <p><a href="#">Chapter 4</a>: Added <a href="#">Timing Simulation of the Example Design</a>, including <a href="#">Table 4-2</a>.</p> <p><a href="#">Chapter 5</a>: Added <code>&lt;component_name&gt;_top.xdc</code> and <code>&lt;component_name&gt;_top_synplify.sdc</code> to <a href="#">Table 5-4</a>. Added <code>implement_synplify.bat</code>, <code>implement_synplify.sh</code>, and <code>synplify.prj</code> to <a href="#">Table 5-5</a>. Added <code>demo_tb_imp.v[hd]</code> and <code>sim_reset_mgt_model.vhd</code> to <a href="#">Table 5-7</a>. Added <code>simulate_isim.sh</code>, <code>simulate_isim.bat</code>, <code>wave_isim.tcl</code>, and <code>simulate_ncsim.bat</code> to <a href="#">Table 5-8</a>. Added <a href="#">simulation/timing</a>, including <a href="#">Table 5-9</a>.</p>



# Table of Contents

---

Revision History .....	2
<b>Preface: About This Guide</b>	
Guide Contents .....	7
Additional Resources .....	7
<b>Chapter 1: Introduction</b>	
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
<b>Chapter 2: Installing the Wizard</b>	
Tools and System Requirements .....	13
Operating Systems .....	13
Design Tools .....	13
Before You Begin .....	13
Installing the Wizard .....	14
Verifying Your Installation .....	14
<b>Chapter 3: Running the Wizard</b>	
Overview .....	15
Functional Overview .....	15
Structure of the 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 Rate, Transceiver Selection, and Clocking .....	22
Encoding and Optional Ports .....	25
Alignment, Termination, and Equalization .....	30
PCI Express, SATA, OOB, PRBS, Channel Bonding, and Clock Correction Selection .....	34
Channel Bonding and Clock Correction Sequence .....	38
Summary .....	39

## Chapter 4: Quick Start Example Design

Overview .....	41
Functional Simulation of the Example Design .....	41
Using ModelSim .....	41
Implementing the Example Design .....	42
Timing Simulation of the Example Design .....	42
Using ModelSim .....	42
Using ChipScope Pro Cores with the Wizard .....	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/timing .....	50
Example Design Description .....	50
Example Design Hierarchy .....	52

# About This Guide

---

This guide describes the 7 Series FPGAs Transceivers 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 transceiver wrapper with the CORE Generator™ tool.
- [Chapter 4, Quick Start Example Design](#) introduces the example design that is included with the transceiver wrappers. The example design demonstrates how to use the wrappers and demonstrates some of the key features of the 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 7 Series FPGAs Transceivers Wizard and provides related information, including additional resources, technical support, and instruction for submitting feedback to Xilinx.

## About the Wizard

The 7 Series FPGAs Transceiver Wizard automates the task of creating HDL wrappers to configure the high-speed serial transceivers in Kintex™-7 and Virtex®-7 FPGAs.

The menu-driven interface allows the user to configure one or more transceivers using predefined templates for popular industry standards, or by using custom templates, to support a wide variety of custom protocols. The Wizard produces a wrapper, an example design, and a testbench for rapid integration and verification of the serial interface with your custom function.

The Wizard produces a wrapper that instantiates one or more properly configured transceivers for custom applications (Figure 1-1).

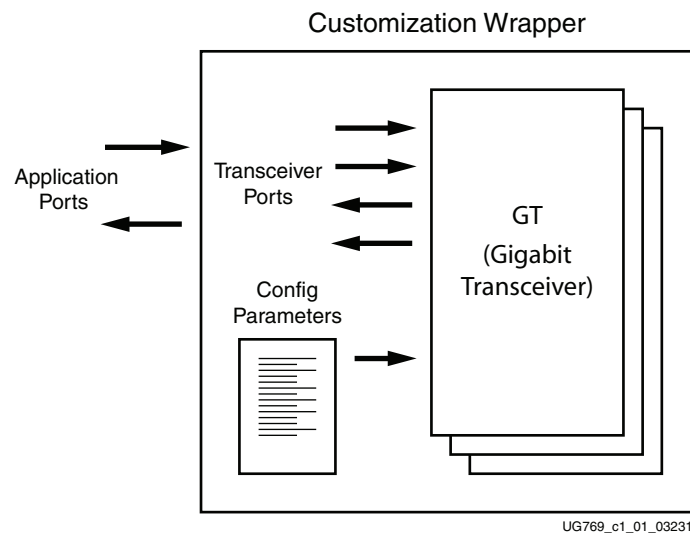


Figure 1-1: Transceiver Wizard Wrapper

The Wizard can be 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 this wizard, refer to the Architecture Wizards product information page:

[http://www.xilinx.com/products/design\\_resources/conn\\_central/solution\\_kits/wizards](http://www.xilinx.com/products/design_resources/conn_central/solution_kits/wizards)

For documentation, see the 7 Series FPGAs Transceivers Wizard page:

[http://www.xilinx.com/support/documentation/ipfpgafeaturedesign\\_iointerface\\_7series-transceivers-wizard.htm](http://www.xilinx.com/support/documentation/ipfpgafeaturedesign_iointerface_7series-transceivers-wizard.htm)

## Features

The Wizard has these features:

- Creates customized HDL wrappers to configure transceivers in the Kintex-7 and Virtex-7 FPGAs:
  - Predefined templates automate transceiver configuration for industry standard protocols such as: Common Packet Radio Interface (CPRI™), DisplayPort, Gigabit Ethernet, Open Base Station Architecture Initiative (OBSAI), Serial RapidIO, 10 Gb Attachment Unit Interface (XAUI), Aurora 8B/10B, Aurora 64B/66B, CEI-6G, Interlaken, OC192, OC48, 10GBASE-R, RXAUI and XLAUI
  - Custom protocols can be specified using the **Start from Scratch** option in the GUI.
- Automatically configures transceiver analog settings
- Supports 64B/66B, 64B/67B, and 8B/10B encoding/decoding
- Includes an example design with a companion testbench as well as implementation and simulation scripts

## Supported Devices

The Wizard supports the Kintex-7 and Virtex-7 FPGAs. For a complete listing of supported devices, see [XTP025](#), *IP Release Notes Guide* for this Wizard. For more information on the 7 series FPGAs, see [DS180](#), *7 Series FPGAs 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
- Constraints File: Synthesis constraints file
- Testbench: Verilog and VHDL
- 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 7 series FPGAs multi-gigabit serial 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:

- [UG769](#), *LogiCORE IP 7 Series FPGAs Transceivers Wizard v1.5 User Guide*
- [XTP025](#), *IP Release Notes Guide* for the Wizard

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

- [DS180](#), *7 Series FPGAs Overview*
- [UG476](#), *7 Series FPGAs GTX Transceivers User Guide*
- ISE software documentation: <http://www.xilinx.com/ise>

## Technical Support

For technical support, go to [www.xilinx.com/support](http://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, please 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](http://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](http://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 for installing the 7 Series FPGAs Transceivers Wizard in the ISE® Design Suite CORE Generator™ tool.

## Tools and System Requirements

### Operating Systems

For a list of system requirements, see the [ISE Design Suite 13: Release Notes Guide](#).

### Design Tools

#### Design Entry

- ISE Design Suite CORE Generator software 13.3

#### Simulation

- ISim 13.3
- Mentor Graphics ModelSim 6.6d
- Cadence Incisive Enterprise Simulator (IES) 10.2
- Synopsys Verilog Compiler Simulator (VCS) and VCS MX 2010.06

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 do the following:

1. Click **Login** at the top of the Xilinx home page then 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 13.3 installation instructions in the ISE Installation and Release Notes available at [www.xilinx.com/support/documentation](http://www.xilinx.com/support/documentation) under the Design Tools tab.

## Verifying Your Installation

Use the following procedure to verify a successful installation 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).

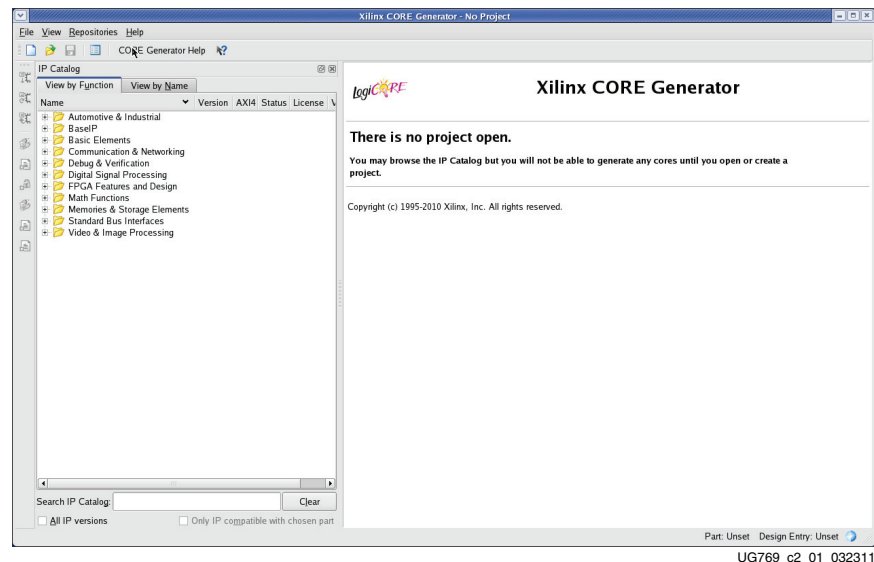


Figure 2-1: CORE Generator Window

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 7 Series FPGAs Transceiver Wizard 1.4 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 7 series FPGAs transceiver wrapper, implementing the wrapper in hardware using the accompanying example design, and simulating the wrapper with the provided example testbench.

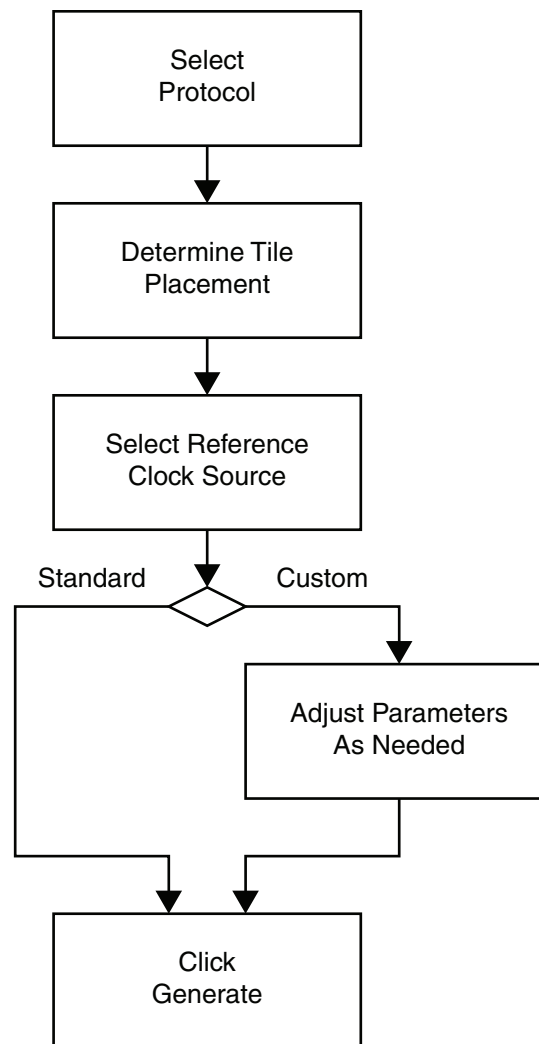
**Note:** The screen captures in this chapter are conceptual representatives of their subjects and provide general information only. For the latest information, see the CORE Generator™ tool.

## Functional Overview

Figure 3-1, page 16 shows the steps required to configure transceivers using the Wizard. Start the CORE Generator software, select the 7 Series FPGAs Transceivers 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.



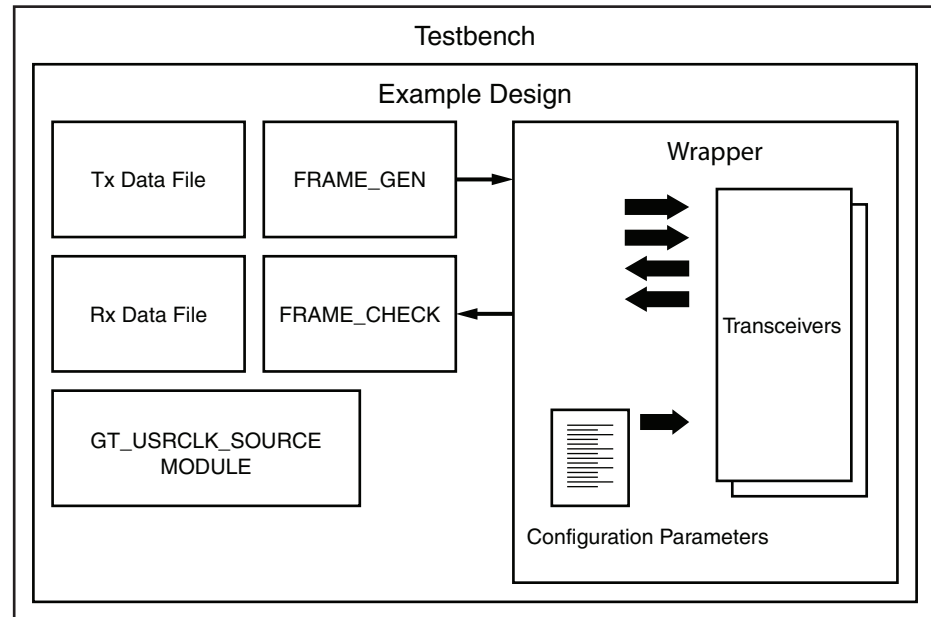
UG769\_c3\_01\_121610

Figure 3-1: Wizard Configuration Steps



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

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



UG769\_c3\_02\_010911

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

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

- Transceiver wrapper, which includes:
  - Specific gigabit transceiver configuration parameters set using the Wizard.
  - Transceiver primitive 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 demonstrating how to stimulate the design.

## Example Design—XAUI Configuration

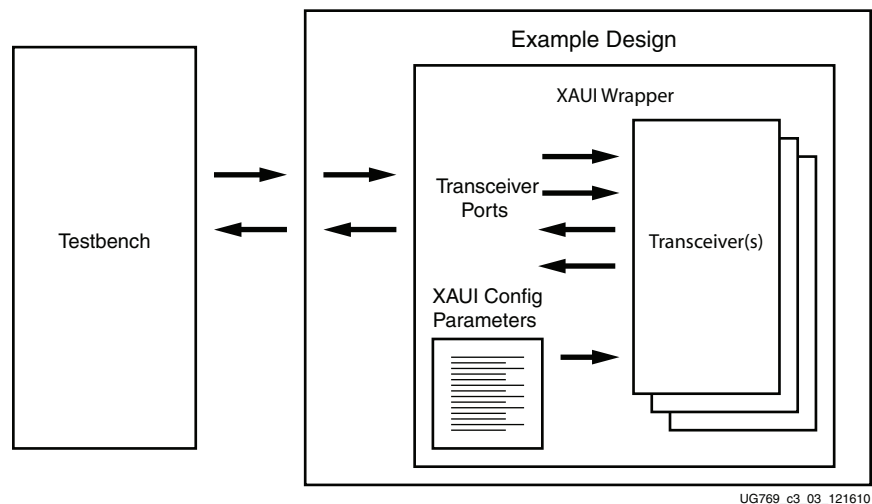
The example design covered in this section is a wrapper that configures a group of transceivers for use in a XAUI application. Guidelines are also given for incorporating the wrapper in a design and for the expected behavior in operation. For detailed information, see [Chapter 4, Quick Start Example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

The XAUI example design consists of the following components:

- A single transceiver wrapper implementing a 4-lane XAUI port using four 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 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/xau_i_example`
2. Start the CORE Generator software.  
For help starting and using the CORE Generator software, see *CORE Generator Help*, available in the ISE® software documentation.
3. Choose **File > New Project** (Figure 3-4).
4. Change the name of the CGP file (optional).
5. Click **Save**.

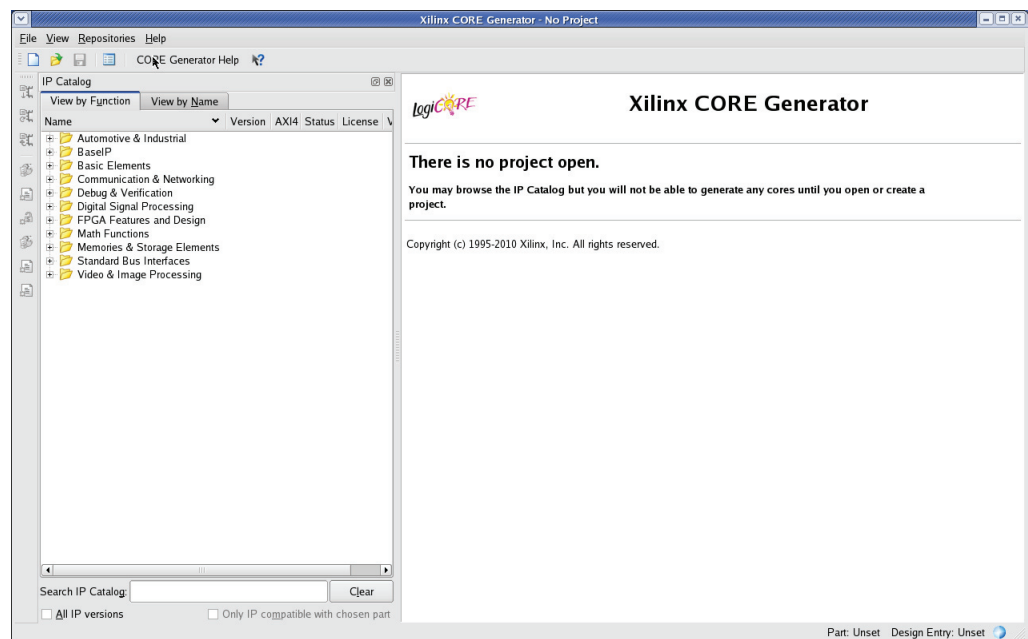


Figure 3-4: Starting a New Project

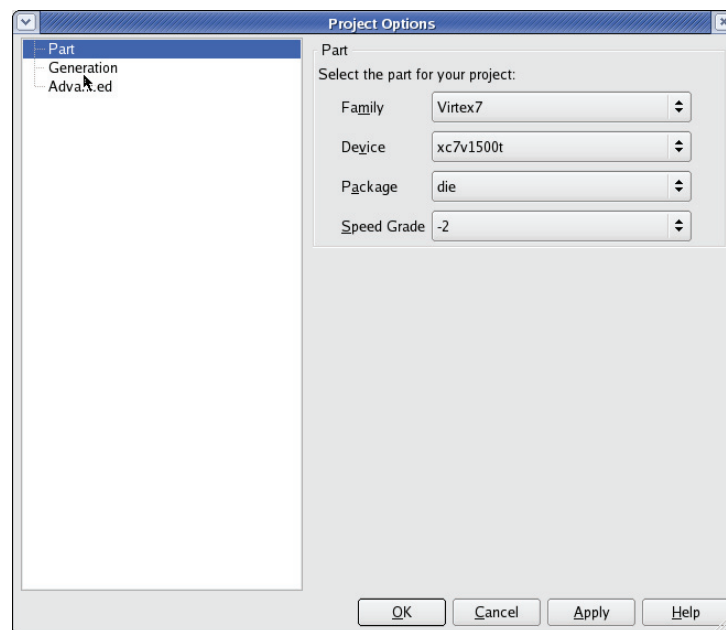
## Setting the Project Options

Set the project options using the following steps:

1. Click **Part** in the option tree.
2. Select **Virtex7** from the Family list.
3. Select a device from the Device list that supports transceivers.
4. Select an appropriate package from the Package list. This example uses the XC7V1500T device (see [Figure 3-5](#)).

**Note:** If an unsupported silicon family is selected, the Wizard remains light grey in the taxonomy tree and cannot be customized. Only devices containing 7 series FPGAs transceivers are supported by the Wizard. See [DS180, 7 Series FPGAs Overview](#) for a list of devices containing the 7 series FPGAs transceivers.

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



UG769\_c3\_05\_122010

Figure 3-5: Target Architecture Setting

## Configuring and Generating the Wrapper

This section provides instructions for generating an example 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 7 Series FPGAs Transceivers Wizard 1.5 in the taxonomy tree under: /FPGA Features & Design/IO Interfaces. (See [Figure 3-6](#))
2. Double-click **7 Series FPGAs Transceivers Wizard 1.5** to launch the Wizard.

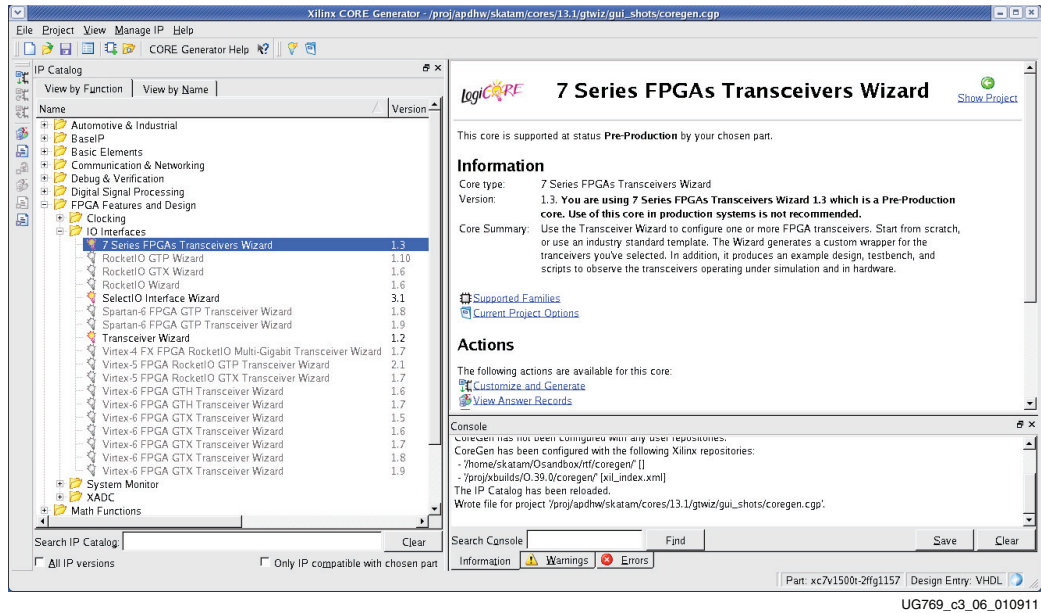


Figure 3-6: Locating the Transceiver Wizard

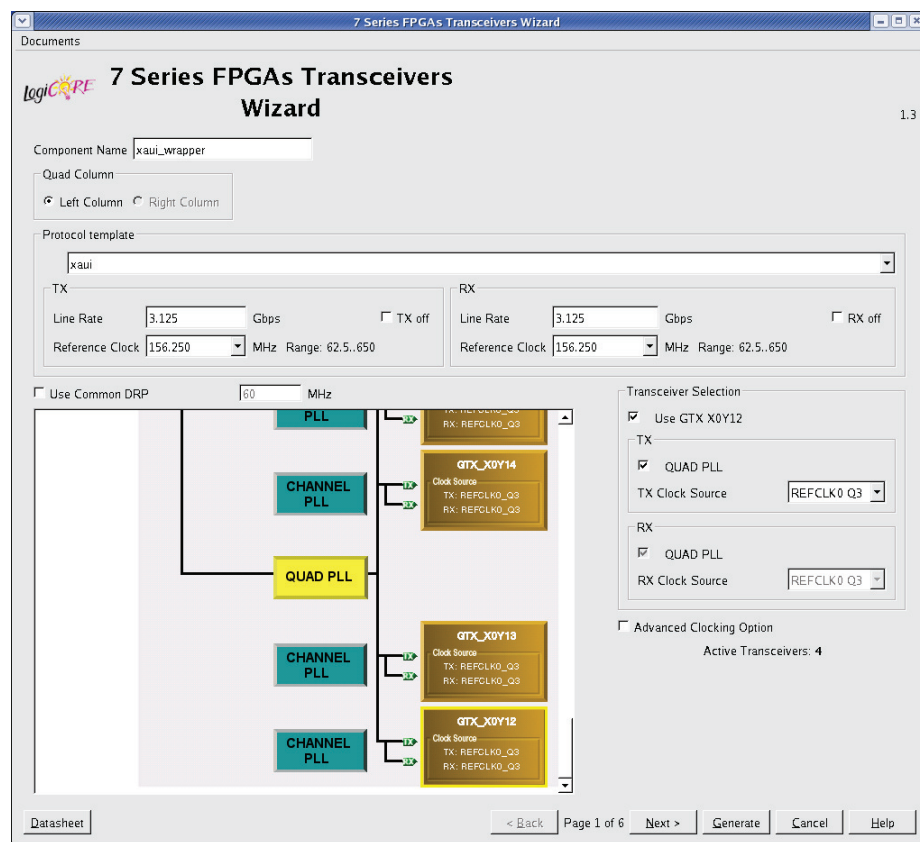
## Line Rate, Transceiver Selection, and Clocking

Page 1 of the Wizard (Figure 3-7) allows you to select the component name and determine the line rate, reference clock frequency, transceiver selection, and clocking. In addition, this page specifies a protocol template.

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

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** if you wish 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.



UG769\_c3\_07\_010911

Figure 3-7: Line Rates, Transceiver Selection, and Clocking—Page 1

3. Use Tables 3-1 through 3-5 to determine the line rate, 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.
Reference Clock	Select from the list the optimal reference clock frequency to be provided by the application. The XAUI example uses 156.25 MHz.
TX off	Selecting this option disables the TX path of the transceiver. The transceiver will act 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. The XAUI example uses 3.125 Gb/s.
Reference Clock	Select from the list the optimal reference clock frequency to be provided by the application. The XAUI example uses 156.25 MHz.
RX off	Selecting this option disables the RX path of the transceiver. The transceiver will act 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 Common DRP	Select this option to have the dynamic reconfiguration port signals of the COMMON block available to the application.
QUAD PLL	Use the Quad PLL when all four transceivers of the quad are used to save power. Quad PLL is shared across four transceivers of a quad.
Advanced Clocking Option	Use this check box to bring out all possible reference clock ports to the generated wrapper. Used for dynamic clock switching.

Table 3-4: Select the Transceiver and the Reference Clocks

Option	Description
GT	Select the individual transceivers by location to be used in the target design. The XAUI example requires four transceivers.
TX Clock Source	Determines the source for the reference clock signal provided to each selected transceiver (see Table 3-5). 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 REFCLK0 signal from the group local to the four selected transceivers (REFCLK0 Q0 option).
RX Clock Source	

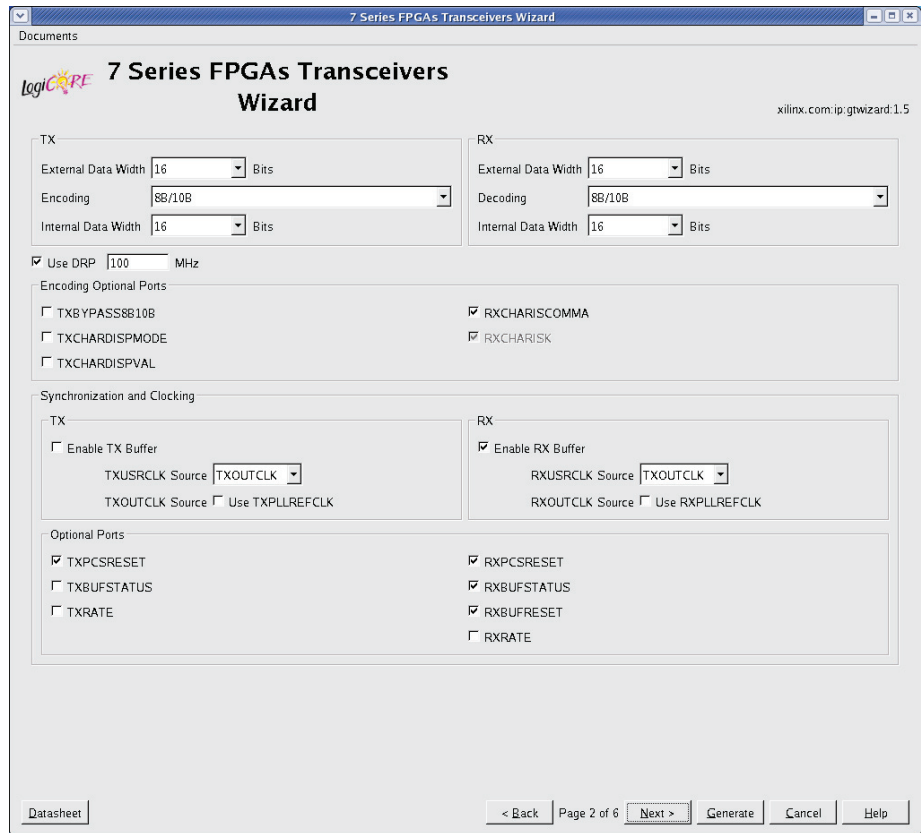
Table 3-5: Reference Clock Source Options

Option	Description
REFCLK0/1 Q0	Reference clock local to transceivers Y0-Y3.
REFCLK0/1 Q1	Reference clock local to transceivers Y4-Y7.
REFCLK0/1 Q2	Reference clock local to transceivers Y8-Y11.
REFCLK0/1 Q3	Reference clock local to transceivers Y12-Y15.
REFCLK0/1 Q4	Reference clock local to transceivers Y16-Y19.
REFCLK0/1 Q5	Reference clock local to transceivers Y20-Y23.
REFCLK0/1 Q6	Reference clock local to transceivers Y24-Y27.
REFCLK0/1 Q7	Reference clock local to transceivers Y28-Y31.
REFCLK0/1 Q8	Reference clock local to transceivers Y32-Y35.
GREFCLK	Reference clock driven by internal FPGA logic. Lowest performance option.



## Encoding and Optional Ports

Page 2 of the Wizard (Figure 3-8) allows you to select encoding and 8B/10B optional ports. Tables 3-6 through 3-12 list the available options.



UG769\_c3\_08\_090711

Figure 3-8: Encoding and Optional Ports—Page 2

Table 3-6: TX Settings

Options		Description
External Data Width	16	Sets the transmitter application interface data width to two 8-bit bytes.
	20	The transmitter application interface datapath width is set to 20 bits.
	32	Sets the transmitter application interface data width to two 8-bit bytes.
	40	Sets the transmitter application interface data width to 40 bits.
	64	Sets the transmitter application interface datapath width to eight 8-bit bytes (64 bits).
	80	Sets the transmitter application interface data width to 80 bits.
Encoding	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 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 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 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 transceiver.
Internal Data Width	16	Selects the internal data width as 16.
	20	Selects the internal data width as 20.
	32	Selects the internal data width as 32.
	40	Selects the internal data width as 40.

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

Table 3-7: RX Settings

Options		Description
External Data Width	16	Sets the receiver application interface data width to two 8-bit bytes.
	20	Sets the receiver application interface data width to 20 bits.
	32	Sets the receiver application interface datapath width to four 8-bit bytes (32 bits).
	40	Sets the receiver application interface data width to 40 bits.
	64	Sets the receiver application interface datapath width to eight 8-bit bytes (64 bits).
	80	Sets the receiver application interface data width to 80 bits.
Decoding	8B/10B	Data stream is passed to an internal 8B/10B decoder.
	64B/66B	Data stream is passed through the 64B/66B gearbox and de-scrambler.
	64B/67B	Data stream is passed through the 64B/67B gearbox and de-scrambler.
Internal Data Width	16	Selects the internal data width as 16.
	20	Selects the internal data width as 20.
	32	Selects the internal data width as 32.
	40	Selects the internal data width as 40.

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

Table 3-8: DRP

Option	Description
Use DRP	Select this option to have the dynamic reconfiguration port signals of the CHANNEL block available to the application

The TX PCS/PMA Phase Alignment setting controls whether the TX buffer is enabled or bypassed. See [UG476, 7 Series FPGAs GTX Transceivers User Guide](#) for details on this setting. The RX PCS/PMA alignment setting controls whether the RX phase alignment circuit is enabled.

[Table 3-9](#) shows the optional ports for 8B/10B.

Table 3-9: 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	
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.

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

[Table 3-10](#) details the TXUSRCLK and RXUSRCLK source signal options.

Table 3-10: TXUSRCLK and RXUSRCLK Source

Option		Description
TX	TXOUTCLK	TXUSRCLK is driven by TXOUTCLK.
RX	TXOUTCLK	RXUSRCLK is driven by TXOUTCLK. This option is not available if the RX buffer is bypassed. For RX buffer bypass mode, RXOUTCLK is used to source RXUSRCLK.

[Table 3-11](#) details the TXOUTCLK and RXOUTCLK source signal options.

Table 3-11: TXOUTCLK and RXOUTCLK Source

Option		Description
TX	Use TXPLLREFCLK	If the check box Use TXPLLREFCLK is checked, TXOUTCLK <sup>(1)</sup> is generated from the input reference clock; otherwise, the Wizard selects the appropriate source for TXOUTCLK.
RX	Use RXPLLREFCLK	If the check box Use RXPLLREFCLK is checked, RXOUTCLK <sup>(1)</sup> is generated from the input reference clock; otherwise, the Wizard selects the appropriate source for RXOUTCLK.

1. See [UG476, 7 Series FPGAs GTX Transceivers User Guide](#) for more information on TXOUTCLK and RXOUTCLK control.

Table 3-12 shows the optional ports available for latency and clocking.

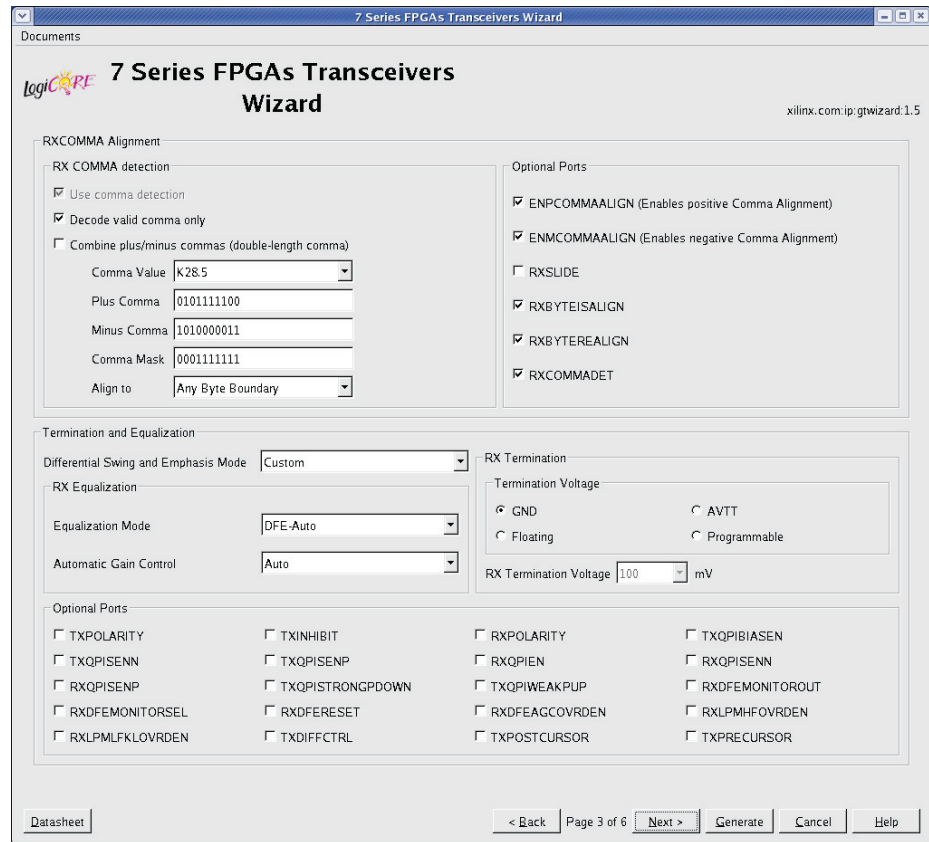
Table 3-12: Optional Ports

Option	Description
TXPCSRESET	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	Transmit rate change port.
RXPCSRESET	Active-High reset signal for the receiver PCS logic.
RXBUFSTATUS	Indicates condition of the RX elastic buffer. 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	Receive rate change port.

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

## Alignment, Termination, and Equalization

Page 3 of the Wizard ([Figure 3-9](#)) allows you to set comma characters and control receive equalization and terminal voltage.



UG769\_c3\_09\_090711

**Figure 3-9: Synchronization and Alignment—Page 3**

Table 3-13 shows the receive comma alignment settings.

Table 3-13: 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).
Align to...	Any Byte Boundary	When a comma is detected, the data stream is aligned using the comma pattern to the nearest byte boundary.
	Two Byte Boundary	When a comma is detected, the data stream is aligned using the comma pattern to the 2-byte boundary.
	Four Byte Boundary	When a comma is detected, the data stream is aligned using the comma pattern to the 4-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. Note that 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.

Table 3-14 details the preemphasis and differential swing settings.

Table 3-14: Preemphasis and Differential Swing

Option	Description
Differential Swing and Emphasis Mode	Specifies the transmitter pre-cursor preemphasis mode setting. Selecting Custom mode enables user driven settings for differential swing and preemphasis level. The XAUI example uses the Custom mode to dynamically set the preemphasis level. See <a href="#">UG476, 7 Series FPGAs GTX Transceivers User Guide</a> for details.

Table 3-15 describes the RX equalization settings.

Table 3-15: RX Equalization

Option	Description
Equalization Mode	Sets the equalization mode in the receiver. See <a href="#">UG476, 7 Series FPGAs GTX Transceivers User Guide</a> for details on the decision feedback equalizer. The XAUI example uses DFE-Auto mode.
Automatic Gain Control	Sets the automatic gain control of the receiver. The value can be set to Auto or Manual.

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

Table 3-16 describes the RX termination settings.

Table 3-16: RX Termination

Option	Description
Termination Voltage	Selecting GND grounds the internal termination network. Selecting Floating isolates the network. Selecting AVTT applies an internal voltage reference source to the termination network. Select the Programmable option for Termination Voltage to select RX termination voltage from a drop-down menu. The XAUI example uses the GND setting.



Table 3-17 lists the optional ports available on this page.

**Table 3-17: 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.
TXQPIBIASEN	Active-High signal to enable QPI bias.
TXQPIWEAKUP	Active-High signal transmit for QPI.
RXDFEAGCOVRDEN	Active-High signal for DFE AGC over-ride.
TXPOSTCURSOR	TXPOSTCURSOR port.
TXQPISENN	Transmit QPI port (negative polarity).
RXDFEMONITOROUT	Receive DFE monitor port.
RXLPMHFOVRDEN	Receive low pass override enable port.
TXQPISENP	Transmit QPI port (positive polarity).
RXDFEMONITORSEL	Receive DFE monitor select port.
RXLPMLFKLOVRDEN	Receive low pass override enable port.
TXQPISTRONGPDOWN	Transmit QPI power down port.
RXDFERESET	Resets the receive DFE block.
TXDIFFCTRL	Transmit driver swing control.
RXQPISENN	Sense output that registers a 1 or 0 on the MGTRXN pin.
RXQPISENP	Sense Output that registers a 1 or 0 on the MGTRXP pin.
RXQPIEN	Disables the RX termination for the QPI protocol.

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

## PCI Express, SATA, OOB, PRBS, Channel Bonding, and Clock Correction Selection

Page 4 of the Wizard (Figure 3-10) 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 channel bonding and clock correction settings are provided.



UG769\_c3\_10\_060211

Figure 3-10: PCI Express, SATA, OOB, PRBS, Channel Bonding, and Clock Correction Selection—Page 4

Table 3-18 details the receiver SATA configuration options.

Table 3-18: Receiver Serial ATA Options

Options		Description
Enable PCI Express		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 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 3-19 details the receiver PCI Express configuration options.

Table 3-19: 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 <a href="#">UG476, 7 Series FPGAs GTX Transceivers User Guide</a> 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 <a href="#">UG476, 7 Series FPGAs GTX Transceivers User Guide</a> 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 <a href="#">UG476, 7 Series FPGAs GTX Transceivers User Guide</a> 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.
	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].
	COMINITDET	Active-High initialization detection signal.
	COMSASDET	Active-High detection signal for SATA.
	COMWAKEDET	Active-High wake up detection signal.
	TXCOMINIT	Transmit initialization port.
	TXCOMSAS	OOB signal.
	TXCOMWAKE	OOB signal.
	COMFINISH	Completion of OOB.
	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 3-20 shows the OOB signal detection options.

Table 3-20: OOB Signal Detection

Option	Description
Use RX OOB Signal Detection	Enables the internal OOB signal detector. OOB signal detection is used for PCIe and SATA.

Table 3-21 details the PRBS settings.

Table 3-21: PRBS Detector

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 TXPRBSSEL	Selects the PRBS Transmission control port.
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 transceiver.

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

Table 3-22 shows the channel bonding options.

Table 3-22: 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 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.

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

Table 3-23 shows the clock correction options.

Table 3-23: 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.
PPM Offset	Indicates the PPM offset between the transmit and receive clocks.
Periodicity of the CC Sequence	Indicates the interval at which CC sequences are inserted in the data stream.
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.

## Channel Bonding and Clock Correction Sequence

Page 5 of the Wizard (Figure 3-11) allow you to define the channel bonding sequence(s). Table 3-24 describes the sequence definition settings and Table 3-23, page 37 describes the clock setup settings.

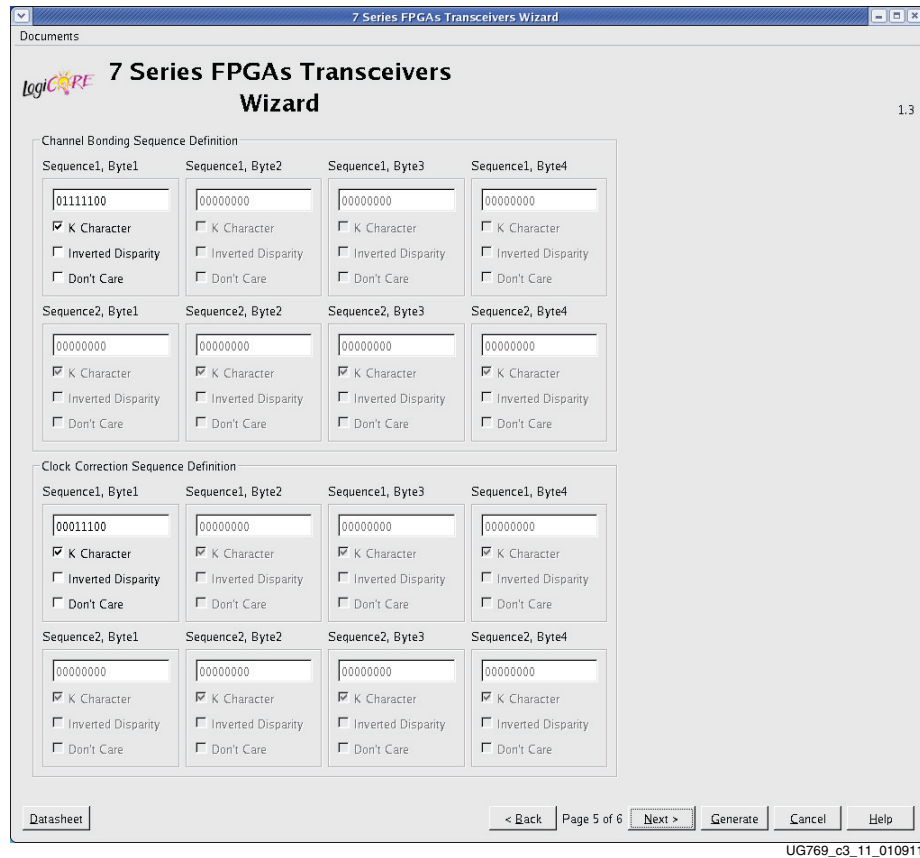


Figure 3-11: Channel Bonding and Clock Correction Sequence—Page 5

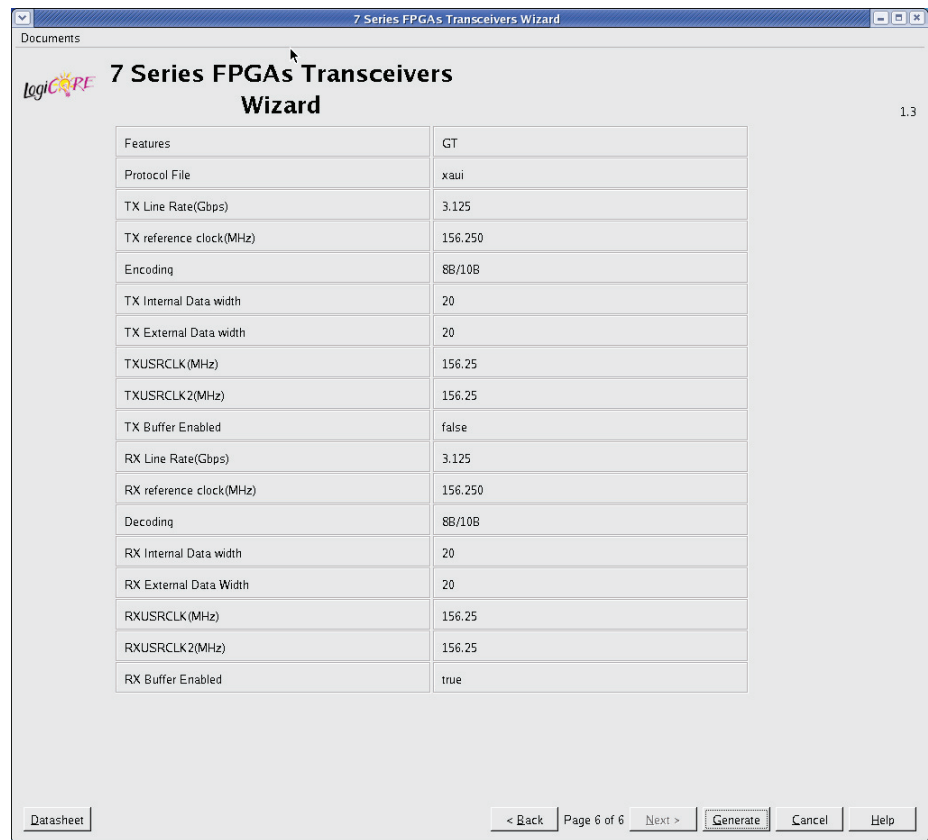
Table 3-24: Channel Bonding and Clock Correction Sequences

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.

## Summary

Page 6 of the Wizard (Figure 3-12) provides a summary of the selected configuration parameters. After reviewing the settings, click **Generate** to exit and generate the wrapper.



UG769\_c3\_12\_010911

Figure 3-12: Summary—Page 6





## Quick Start Example Design

---

### Overview

This chapter introduces the example design that is included with the 7 series FPGAs transceiver wrappers. The example design demonstrates how to use the wrappers and demonstrates some of the key features of the transceivers. For detailed information about the example design, see [Chapter 5, Detailed Example Design](#).

### Functional Simulation of the Example Design

The 7 Series FPGAs Transceivers Wizard (hereinafter called the 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](#). See the *Synthesis and Simulation Design Guide* for ISE® software 13.3 available in the ISE software documentation for instructions on how to compile ISE simulation libraries.

**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.

## 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 xau_i_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.

## Timing Simulation of the Example Design

The Wizard provides a script to observe the behavior of the example design during timing simulations.

### Using ModelSim

Prior to performing the timing simulation with ModelSim, the generated design should pass through implementation. All source files in the following directories must be compiled to a single library, as shown in [Table 4-2](#). See the *Synthesis and Simulation Design Guide* for ISE 13.3 available in the ISE software documentation for instructions on how to compile ISE simulation libraries.

Table 4-2: Required ModelSim Timing 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/timing
```
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 and simulates the routed netlist of the example design and testbench.

## Using ChipScope Pro Cores with the Wizard

The ChipScope™ Pro Integrated Controller (ICON) and Virtual Input/Output (VIO) cores aid in debugging and validating the design in board. To assist with debugging, these ChipScope Pro cores are provided with the 7 Series FPGAs Transceivers Wizard wrapper, which is enabled by setting USE\_CHIPSCOPE to 1 in the <component\_name>\_top\_example\_design file.



# 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/timing**  
Timing simulation files

## Directory and File Contents

The 7 Series FPGAs Transceivers 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 transceiver wrapper. Instantiates individual transceiver wrappers. For use in the target design.
<component_name>.[veo   vho]	Transceiver wrapper files instantiation templates. Includes templates for the transceiver wrapper module and the IBUFDS module.
<component_name>.xco	Log file from the CORE Generator tool describing which options were used to generate the 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>_gt.v[hd]	Individual transceiver wrapper to be instantiated in the main transceiver wrapper. Instantiates the selected 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: Transceiver Wrapper Component Name**

Name	Description
<project_dir>/<component_name>	
gtwizard_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	
ug769_gtwizard.pdf	<i>LogiCORE IP 7 Series FPGAs Transceivers Wizard v1.5 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	
gt_frame_check.v[hd]	Frame-check logic to be instantiated in the example design.
gt_frame_gen.v[hd]	Frame-generator logic to be instantiated in the example design.
gt_attributes.ucf	Constraints file containing the transceiver attributes generated by the transceiver Wizard GUI settings.
<component_name>_top.ucf	Constraint file for mapping the transceiver wrapper example design onto a Kintex™-7 or Virtex®-7 FPGA.
<component_name>_top.xcf	XST specific constraint file for mapping the transceiver wrapper example design onto a Kintex-7 or Virtex-7 FPGA.
<component_name>_top.v[hd]	Top-level example design. Contains transceiver, reset logic, and instantiations for frame generator and frame-checker logic. Also contains ChipScope™ Pro module instantiations.
<component_name>_gt_usrclk_source.v[hd]	Transceiver user clock module that generates clocking signals for transceiver and the user logic.
<component_name>_clock_module.v[hd]	Clock module that instantiates the MMCM.
gt_rom_init_tx.dat	Block RAM initialization pattern for gt_frame_gen module. The pattern is user modifiable.
gt_rom_init_rx.dat	Block RAM initialization pattern for gt_frame_check module. The pattern is user modifiable.
<component_name>_top.xdc	Xilinx Design Constraint file for mapping the transceiver wrapper example design onto a Kintex-7 or Virtex-7 FPGA.
<component_name>_top_synplify.sdc	Synopsys Design Constraint file for mapping the transceiver wrapper example design onto a Kintex-7 or Virtex-7 FPGA.

[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 project file.
data_vio.ngc	Netlist of the design generated by ChipScope Pro Virtual Input/Output (VIO) Wizard.
icon.ngc	Netlist of the design generated by ChipScope Pro Integrated Controller (ICON) Wizard.
ila.ngc	Netlist of the design generated by ChipScope Pro Integrated Logic Analyzer (ILA) Wizard.
implement.bat	Windows batch file that processes the example design through the tool flow.
implement.sh	Linux shell script that processes the example design through the tool flow.
xst.prj	XST project file for the example design; it lists all of the source files to be synthesized.
xst.scr	XST script file for the example design that is used to synthesize the Wizard, called from the implement script described above.
planAhead_ise.bat	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 for ISE design tools 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.

[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: Results 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 <a href="#">Functional Simulation of the Example Design, page 41</a> .
demo_tb_imp.v[hd]	Testbench to perform timing simulation of the provided example design. See <a href="#">Timing Simulation of the Example Design, page 42</a> .
sim_reset_mgt_model.vhd	Reset module for VHDL required for emulating the GSR pulse at the beginning of functional simulation. This is required to correctly reset the VHDL MGT smart model.

[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_mti.do	ModelSim simulation script.
wave_mti.do	Script for adding transceiver wrapper signals to the ModelSim wave viewer.
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 transceiver wrapper signals to VCS wave window.
wave_ncsim.sv	Script for adding transceiver wrapper signals to the Cadence IES wave viewer.
gt_rom_init_tx.dat	Block RAM initialization pattern for gt_frame_gen module. The pattern is user modifiable.
gt_rom_init_rx.dat	Block RAM initialization pattern for gt_frame_check module. The pattern is user modifiable.
simulate_isim.sh	Linux script for running simulation using ISim.
simulate_isim.bat	Windows script for running simulation using ISim.
wave_isim.tcl	Script for adding transceiver wrapper signals to the ISim wave viewer.
simulate_ncsim.bat	Windows script for running simulation using Cadence IES.

[Back to Top](#)

## simulation/timing

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

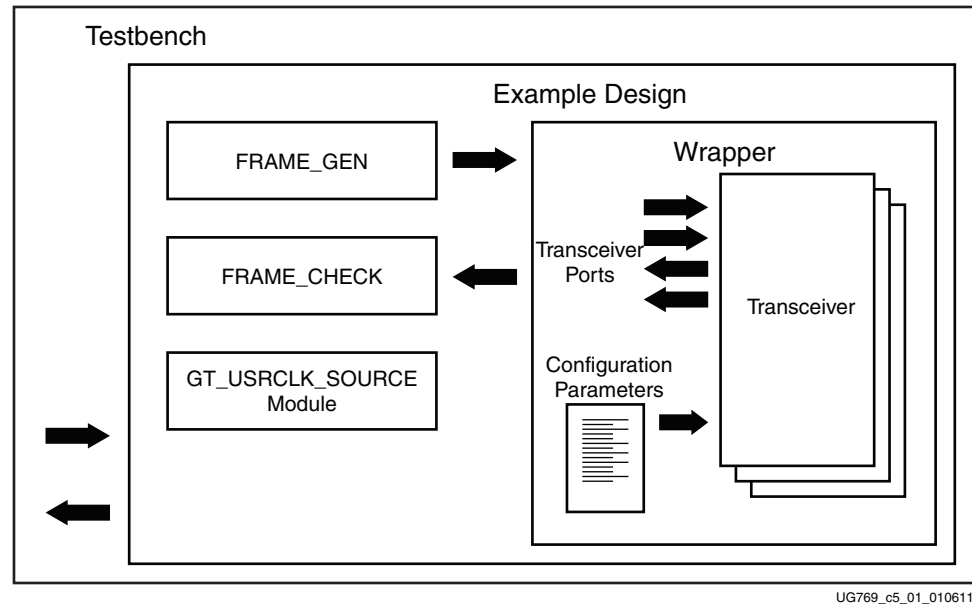
Table 5-9: Timing Directory

Name	Description
simulate_mti.do	ModelSim timing 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 transceivers in a design. The example design is shown in [Figure 5-1](#).



UG769\_c5\_01\_010611

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 `gt_rom_init_tx.dat` and `gt_rom_init_rx.dat` files. 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`.

The example design also demonstrates how to properly connect clocks to transceiver ports `TXUSRCLK`, `TXUSRCLK2`, `RXUSRCLK` and `RXUSRCLK2`. Properly configured clock module wrappers are also provided if they are required to generate user clocks for the instantiated transceivers. The logic for scrambler, descrambler and block synchronization is instantiated in the example design for 64B/66B and 64B/67B encoding.

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 project file is also included with each example design.

For the example design to work properly in simulation, 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_GT_TOP
|   |___XAUI_WRAPPER
|   |   |___XAUI_WRAPPER_GT (1 per transceiver)
|   |
|   |___GT_FRAME_GEN           (1 per transceiver)
|   |___GT_FRAME_CHECK        (1 per transceiver)
|   |___GT_USRCLK_SOURCE
|   |___CLOCK_MODULE
```