

# LogiCORE IP 7 Series FPGAs Transceivers Wizard v2.6

*User Guide*

UG769 (v4.6) June 19, 2013



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 2011–2013 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, 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.<br>In Chapter 1, Introduction, revised <a href="#">Features, Supported Devices, Provided with the Wizard, and Related Xilinx Documents</a> .<br>In Chapter 2, Installing the Wizard, revised <a href="#">Tools and System Requirements</a> .<br>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 3-32</a> , and deleted RX Buffer Max Latency and RX Buffer Min Latency from <a href="#">Table 3-32</a> .<br>In Chapter 4, Quick Start Example Design, updated <a href="#">Table 4-1</a> . Added <a href="#">Using ChipScope Pro Cores with the Wizard in the ISE Tools</a> .<br>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 5-5</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 5-6</a> . Added paragraph about ChipScope Pro tools to <a href="#">Example Design Description for GTX, GTH, and GTP Transceivers</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 3-12</a>, renumbered Q4 through Q9 as Q3 through Q8. Updated <a href="#">Figure 3-17</a>. Updated row shading and descriptions in <a href="#">Table 3-13</a> and <a href="#">Table 3-14</a>. In <a href="#">Table 3-26</a>, updated row shading, removed RXCDRRESET, and added RXQPISENN, RXQPISENP, and RXQPIEN. Updated row shading in <a href="#">Table 3-29</a>. Removed shading from RX rows in <a href="#">Table 3-16</a>.</p> <p><a href="#">Chapter 4</a>: Added <a href="#">Timing Simulation of the Example Design Using the ISE Tools</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 5-5</a>. Added <code>implement_synplify.bat</code>, <code>implement_synplify.sh</code>, and <code>synplify.prj</code> to <a href="#">Table 5-6</a>. Added <code>demo_tb_imp.v[hd]</code> and <code>sim_reset_mgt_model.vhd</code> to <a href="#">Table 5-8</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 5-9</a>. Added <a href="#">simulation/timing</a>, including <a href="#">Table 5-10</a>.</p>  |
| 01/18/12 | 4.0     | <p>Wizard 1.6 release.</p> <p><a href="#">Chapter 1</a>: Added protocols supported by GTH transceivers to <a href="#">Features</a>.</p> <p><a href="#">Chapter 3</a>: Added <a href="#">step 3</a> to <a href="#">GT Type Selection</a>. Updated <a href="#">Figure 3-15</a> and <a href="#">Figure 3-17</a>. Updated row shading in <a href="#">Table 3-13</a> and <a href="#">Table 3-14</a>. Added <a href="#">Figure 3-18</a> and <a href="#">Table 3-21</a>. Updated title and shading in <a href="#">Table 3-28</a>.</p>   |
| 05/08/12 | 4.1     | <p>Wizard 2.1 release.</p> <p><a href="#">Chapter 1</a>: Updated lists of supported protocols in <a href="#">Features</a>.</p> <p><a href="#">Chapter 3</a>: Updated <a href="#">GT Type Selection</a>, including <a href="#">Figure 3-15</a>. Added <a href="#">Line Rate, Transceiver Selection, and Clocking</a>, including <a href="#">Figure 3-16</a>. Removed QUAD PLL option from <a href="#">Table 3-9</a>. Added <a href="#">Table 3-11</a>. Updated <a href="#">Figure 3-17</a>. Updated Encoding description in <a href="#">Table 3-13</a>. Added QPLLPD, CPLLPD, PLL0PD, and PLL1PD to <a href="#">Table 3-20</a>. Updated <a href="#">Figure 3-19</a>, <a href="#">Figure 3-20</a>, <a href="#">Figure 3-21</a>, and <a href="#">Figure 3-22</a>. In <a href="#">Table 3-26</a>, removed shading from TXPOSTCURSOR row, added TXPRECURSOR, and replaced RXDFERESSET with RXDFELPMRESET. Added TXPOWERDOWN and RXPOWERDOWN to <a href="#">Table 3-28</a>.</p> <p><a href="#">Chapter 5</a>: Updated <a href="#">Table 5-5</a>, <a href="#">Table 5-6</a>, and <a href="#">Table 5-8</a>. Updated <a href="#">Example Design Description for GTX, GTH, and GTP Transceivers</a>.</p>  |
| 07/25/12 | 4.2     | <p>Wizard 2.2 release.</p> <p><a href="#">Chapter 1</a>: Added Artix-7 FPGAs. Updated lists of supported protocols in <a href="#">Features</a>.</p> <p><a href="#">Chapter 2</a>: Added Vivado Design Suite to <a href="#">Tools and System Requirements</a>.</p> <p><a href="#">Chapter 3</a>: Added GTZ transceivers throughout, including new sections in <a href="#">Configuring and Generating the Wrapper</a>. Updated <a href="#">GT Type Selection</a>, including several figures. Updated <a href="#">Line Rate, Transceiver Selection, and Clocking</a>.</p> <p><a href="#">Chapter 4</a>: Added <a href="#">Functional Simulation of the Example Design for GTZ Transceivers and Implementing the Example Design for GTZ Transceivers</a>. Updated <a href="#">Timing Simulation of the Example Design Using the ISE Tools</a> and <a href="#">Using ChipScope Pro Cores with the Wizard in the ISE Tools</a> to indicate lack of support for GTZ wizard.</p> <p><a href="#">Chapter 5</a>: Added <code>&lt;component_name&gt;_octal0.v</code> and <code>&lt;component_name&gt;_octal1.v</code> to <a href="#">Table 5-1</a>. Added <code>&lt;component name&gt;/src</code>. In <a href="#">Table 5-5</a>, added <code>frame_gen_top.v</code> and <code>frame_check_top.v</code>, and updated several rows to indicate files not generated for GTZ transceivers. In <a href="#">Table 5-6</a>, replaced <code>planAhead_ise.bat/sh/tcl</code> with <code>vivado_rdn.bat/sh/tcl</code> and updated descriptions of <code>implement_synplify.bat</code>, <code>implement_synplify.sh</code>, and <code>synplify.prj</code>.</p> |

| Date     | Version         | Revision  |
|----------|-----------------|---|
| 07/25/12 | 4.2<br>(Cont'd) | Updated to indicate that files are not generated for GTZ transceivers in <a href="#">Table 5-6</a> , <a href="#">Table 5-8</a> , <a href="#">Table 5-9</a> , and <a href="#">Table 5-10</a> . Added <a href="#">Example Design Description for GTZ Transceivers</a> .   |
| 10/23/12 | 4.3             | Wizard 2.3 release.<br><a href="#">Chapter 1</a> : Updated lists of supported protocols in <a href="#">Features</a> .<br><a href="#">Chapter 2</a> : Updated software versions in <a href="#">Tools and System Requirements</a> .<br><a href="#">Chapter 3</a> : Updated <a href="#">Figure 3-11</a> and <a href="#">Figure 3-12</a> .<br><a href="#">Chapter 4</a> : Updated <a href="#">Implementing the Example Design for GTZ Transceivers</a> .<br><a href="#">Chapter 5</a> : Added <code>v7ht.tcl</code> to <a href="#">Table 5-6</a> . Added <a href="#">CTLE3 Adaptation Modules for GTX Transceivers</a> .  |
| 12/19/12 | 4.4             | Wizard 2.4 release.<br><a href="#">Chapter 1</a> : Updated lists of supported protocols in <a href="#">Features</a> .<br><a href="#">Chapter 3</a> : Updated descriptions of TXOUTCLK source and RXOUTCLK source in <a href="#">Table 3-4</a> .<br><a href="#">Chapter 4</a> : Updated <a href="#">Using ChipScope Pro Cores with the Wizard in the ISE Tools</a> .<br><a href="#">Chapter 5</a> : Updated <a href="#">Reset Finite State Machine</a> . Updated fourth and fifth bullets after <a href="#">Figure 5-2</a> . Added <a href="#">Beachfront Module</a> , <a href="#">Dynamic Phase Deskew</a> , <a href="#">Multi-Lane Mode</a> , and <a href="#">Known Limitations of the GTZ Wizard</a> . Updated <a href="#">Example Design Hierarchy</a> .   |
| 04/05/13 | 4.5             | Wizard 2.5 release.<br><a href="#">Chapter 1</a> : Updated lists of supported protocols in <a href="#">Features</a> .<br><a href="#">Chapter 2</a> : Updated software versions in <a href="#">Tools and System Requirements</a> .<br><a href="#">Chapter 3</a> : Updated <a href="#">Figure 3-17</a> . Added note 2 to <a href="#">Table 3-14</a> . Added <a href="#">Table 3-17</a> . Updated <a href="#">Figure 3-19</a> . Added Use RX Equalizer CTLE3 Adaptation Logic option to <a href="#">Table 3-24</a> .<br><a href="#">Chapter 4</a> : Removed “Functional Simulation of the Example Design for GTZ Transceivers” and “Implementing the Example Design for GTZ Transceivers” sections. Updated all headings with ISE® tools.<br><a href="#">Chapter 5</a> : Removed <code>vivado_rdn.bat</code> , <code>vivado_rdn.sh</code> , and <code>vivado_rdn.tcl</code> rows from <a href="#">Table 5-6</a> . Removed <code>sim_reset_mgt_model.vhd</code> from <a href="#">Table 5-8</a> . Updated heading for <a href="#">CTLE3 Adaptation Modules for GTX Transceivers</a> section. Removed “AGC Loop FSM - Adaptation Lock Time” and “LPM Loop FSM - LPM Mode” sections. Updated <a href="#">CTLE AGC Comp - CTLE3 Adaptation</a> . Added <a href="#">Reset Sequence Modules for GTH and GTP Transceivers</a> . In <a href="#">Known Limitations of the GTZ Wizard</a> , removed limitation of Wizard not allowing individual channels to be configured non-identically. |
| 06/19/13 | 4.6             | Wizard 2.6 release.<br><a href="#">Chapter 1</a> : Updated lists of supported protocols in <a href="#">Features</a> .<br><a href="#">Chapter 2</a> : Updated software versions in <a href="#">Tools and System Requirements</a> .<br><a href="#">Chapter 3</a> : Updated <a href="#">Setting Up the Project</a> and <a href="#">Configuring and Generating the Wrapper</a> .<br><a href="#">Chapter 5</a> : Removed “LPM Loop FSM - LPM Mode” section. Updated filename in <a href="#">Table 5-4</a> . Updated list of assumptions for Example Reset FSM in <a href="#">Reset Finite State Machine</a> . Updated <a href="#">Reset Sequence Modules for GTH and GTP Transceivers</a> and description after <a href="#">Table 5-13</a> . Updated <a href="#">Example Design Hierarchy</a> and <a href="#">Known Limitations of the GTZ Wizard</a> .  |

# 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 .....   | 12 |
| Provided with the Wizard .....  | 12 |
| Recommended Design Experience .....                                       | 12 |
| Related Xilinx Documents .....  | 12 |
| Technical Support .....   | 13 |
| Ordering Information .....  | 13 |
| Feedback .....  | 13 |
| Wizard .....  | 13 |
| Document .....  | 13 |
| <b>Chapter 2: Installing the Wizard</b>                                   |    |
| Tools and System Requirements .....                                       | 15 |
| Operating Systems .....   | 15 |
| Design Tools .....  | 15 |
| Before You Begin .....  | 15 |
| Installing the Wizard .....   | 16 |
| Verifying Your Installation .....   | 16 |
| <b>Chapter 3: Running the Wizard</b>                                      |    |
| Overview .....  | 17 |
| Functional Overview .....   | 17 |
| Structure of the Transceiver Wrapper, Example Design, and Testbench ..... | 19 |
| Example Design—XAUI Configuration .....                                   | 20 |
| Setting Up the Project .....  | 21 |
| Creating a Directory .....  | 21 |
| Setting the Project Options .....   | 23 |
| Configuring and Generating the Wrapper .....                              | 25 |
| GTZ Transceivers .....  | 26 |
| GTX, GTH, and GTP Transceivers .....                                      | 35 |
| <b>Chapter 4: Quick Start Example Design</b>                              |    |
| Overview .....  | 55 |

|  |    |
|--|----|
| <b>Functional Simulation of the Example Design Using the ISE Tools</b> ..... | 55 |
| Using ModelSim .....   | 55 |
| <b>Implementing the Example Design Using the ISE Tools</b> .....             | 56 |
| <b>Timing Simulation of the Example Design Using the ISE Tools</b> .....     | 56 |
| Using ModelSim .....   | 56 |
| <b>Using ChipScope Pro Cores with the Wizard in the ISE Tools</b> .....      | 57 |

## Chapter 5: Detailed Example Design

|  |    |
|--|----|
| <b>Directory and File Structure</b> .....                                  | 59 |
| <b>Directory and File Contents</b> .....                                   | 60 |
| <project directory> .....  | 60 |
| <project directory>/<component name> .....                                 | 60 |
| <component name>/doc .....   | 61 |
| <component name>/src .....   | 61 |
| <component name>/example design .....                                      | 61 |
| <component name>/implement .....   | 63 |
| implement/results .....  | 64 |
| <component name>/simulation .....  | 64 |
| simulation/functional .....  | 64 |
| simulation/timing .....  | 65 |
| <b>Example Design Description for GTX, GTH, and GTP Transceivers</b> ..... | 66 |
| Reset Finite State Machine .....   | 67 |
| CTLE3 Adaptation Modules for GTX Transceivers .....                        | 69 |
| Example Design Hierarchy .....   | 69 |
| <b>Reset Sequence Modules for GTH and GTP Transceivers</b> .....           | 70 |
| <b>Example Design Description for GTZ Transceivers</b> .....               | 70 |
| CTLE Tuning .....  | 70 |
| Beachfront Module .....  | 73 |
| Dynamic Phase Deskew .....   | 76 |
| Multi-Lane Mode .....  | 77 |
| Example Design Hierarchy .....   | 77 |
| <b>Known Limitations of the GTZ Wizard</b> .....                           | 77 |

# 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>

For Vivado™ design tools user documentation, see:

<http://www.xilinx.com/cgi-bin/docs/rdoc?t=vivado+docs>

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 Artix™-7, 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).

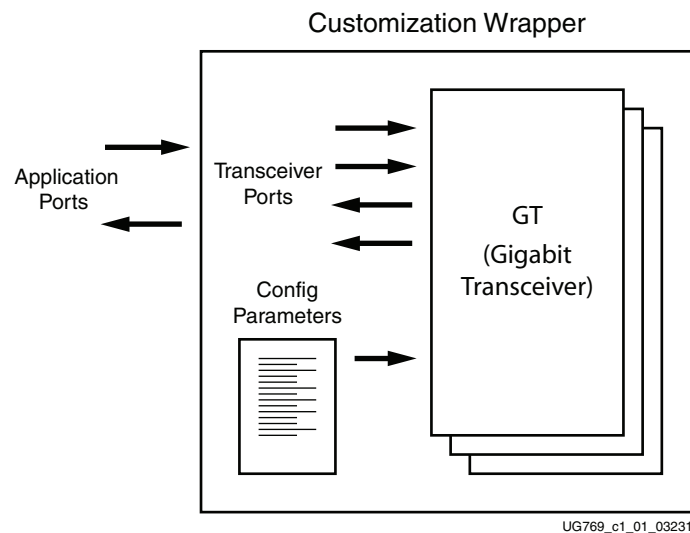


Figure 1-1: Transceiver Wizard Wrapper

The Wizard can be accessed from the ISE® software CORE Generator tool and the Vivado™ Design Suite. 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. GTX transceivers support:
    - Common Packet Radio Interface (CPRI™): 0.6, 1.2, 2.4, 3.072, 4.9, 6.144, and 9.83 Gb/s
    - OC-48: 2.488 Gb/s
    - OC-192: 9.956 Gb/s
    - Gigabit Ethernet: 1.25 Gb/s
    - Aurora 64B/66B: 12.5 Gb/s
    - Aurora 8B/10B: 6.6 Gb/s
    - PCI Express® Gen1: 2.5 Gb/s
    - PCI Express Gen2: 5 Gb/s
    - DisplayPort: 1.620, 2.7, 5.4 Gb/s
    - 10GBASE-R: 10.3125 Gb/s
    - Interlaken: 4.25, 5.0, 6.25 Gb/s
    - Open Base Station Architecture Initiative (OBSAI): 3.072 Gb/s
    - OBSAI: 6.144 Gb/s
    - 10 Gb Attachment Unit (XAUI): 3.125 Gb/s
    - 10 Gb Reduced Attachment Unit (RXAUI): 6.25 Gb/s
    - Serial RapidIO Gen1: 1.25, 2.5, 3.125 Gb/s
    - Serial RapidIO Gen2: 5.0, 6.25 Gb/s
    - JESD204: 3.0, 6.0 Gb/s
    - 100 Gb Attachment Unit Interface (CAUI): 10.3125 Gb/s
    - 10GBASE-KR: 10.3125 Gb/s
    - Common Electrical Interface (CEI) 6G-SR: 4.976–6.375 Gb/s
    - 40 Gb Attachment Unit Interface (XLAUI): 10.3125 Gb/s
    - Quad Serial Gigabit Media Independent Interface (QSGMII): 5 Gb/s
    - High-Definition Serial Digital Interface (HD-SDI)/3 Gb/s Serial Digital Interface (3G-SDI): 1.485/2.97 Gb/s
  - GTH transceivers support:
    - CEI 6G-SR: 4.976–6.375 Gb/s
    - Interlaken: 6.25 Gb/s

- 10GBASE-KR: 10.3125 Gb/s
- 10GBASE-R: 10.3125 Gb/s
- XLAUI: 10.3125 Gb/s
- CEI-11: 9.956–11.1 Gb/s
- CAUI: 10.3125 Gb/s
- OTU4: 11.18, 12.5, 13.1 Gb/s
- CPRI: 0.6, 1.2, 2.4, 3.072, 4.9, 6.144, and 9.83 Gb/s
- Gigabit Ethernet: 1.25 Gb/s
- OC-48: 2.48832 Gb/s
- OC-192: 9.956 Gb/s
- DisplayPort: 1.620, 2.7, 5.4 Gb/s
- JESD204
- Optical-channel Transport Lane (OTL) 3.4: 10.7546 Gb/s
- PCI Express Gen1: 2.5 Gb/s
- PCI Express Gen2: 5 Gb/s
- QSGMII: 5 Gb/s
- RXAUI: 6.25 Gb/s
- XAUI: 3.125 Gb/s
- Aurora 64B/66B: 12.5 Gb/s
- Aurora 8B/10B: 6.6 Gb/s
- Serial RapidIO Gen2: 5.0 Gb/s

GTP transceivers support:

- CEI 6G-SR: 4.976–6.375 Gb/s
- Aurora 64B/66B: 6.6 Gb/s
- Aurora 8B/10B: 6.6 Gb/s
- DisplayPort: 1.620, 2.7, 5.4 Gb/s
- JESD204
- CPRI: 0.6, 1.2, 2.4, 3.072, 4.9, 6.144 Gb/s
- SDI/HD-SDI/3G-SDI: 0.27/1.485/2.97 Gb/s
- V-by-One: 2.97/3.7125/1.485 Gb/s
- Gigabit Ethernet: 1.25 Gb/s
- QSGMII: 5 Gb/s
- RXAUI: 6.25 Gb/s
- XAUI: 3.125 Gb/s
- Serial RapidIO Gen1: 1.25, 2.5, 3.125 Gb/s
- Serial RapidIO Gen2: 5.0, 6.25 Gb/s

GTZ transceivers support:

- CAUI4: 27.78125 Gb/s
- Aurora 64B/66B: 25.78125 Gb/s

- 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 Artix-7, 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 v2.6 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/GTH Transceivers User Guide](#)
- ISE software documentation: <http://www.xilinx.com/ise>
- Vivado tools documentation:  
<http://www.xilinx.com/cgi-bin/docs/rdoc?t=vivado+docs>

## 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 14.6 or higher, which is a standard component of the ISE Design Suite, or the Vivado 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 and the Vivado™ Design Suite.

## Tools and System Requirements

### Operating Systems

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

### Design Tools

#### Design Entry

- ISE Design Suite CORE Generator software 14.6
- Vivado Design Suite 2013.2

#### Simulation

- ISim 14.6
- Mentor Graphics Questa 10.1a
- Cadence Incisive Enterprise Simulator (IES) 12.2
- Synopsys Verilog Compiler Simulator (VCS) G-2012.09

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 14.6
- Synopsys Synplify Pro G-2012.09-SP1

## Before You Begin

Before installing the Wizard, you must have a MySupport account and the ISE 14.6 software or Vivado tools 2013.2 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 14.6 or Vivado tools 2013.2 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 14.6 and Vivado tools 2013.2 software. Follow the ISE 14.6 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](#)).

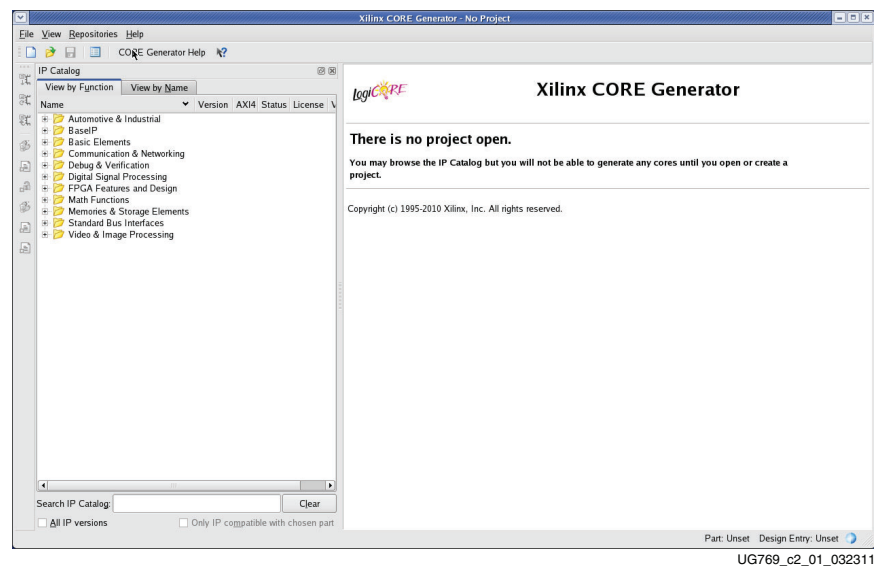


Figure 2-1: CORE Generator Window

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.

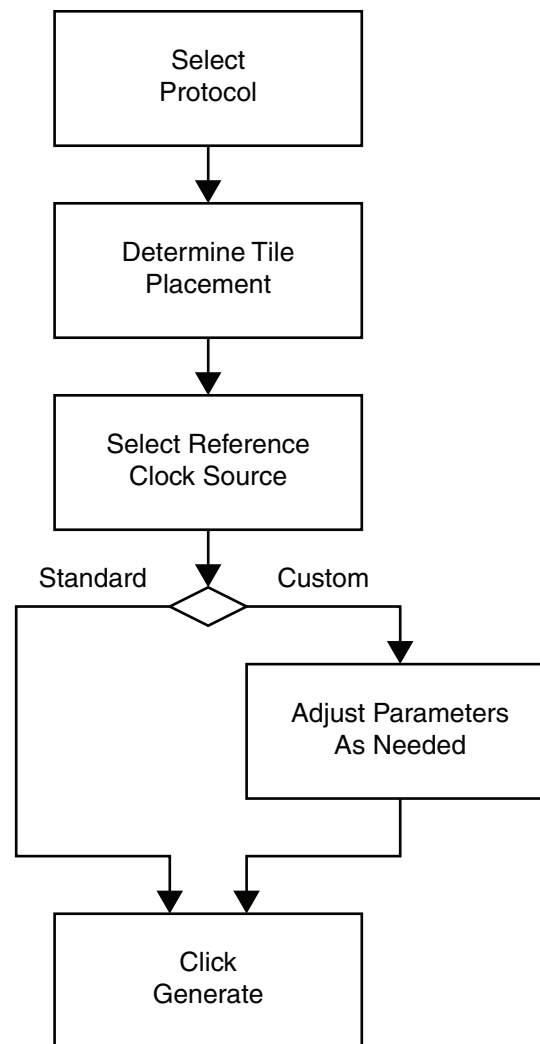
**Note:** The screen captures in this chapter are conceptual representatives of their subjects and provide general information only. For the latest information, see the CORE Generator™ tool.

## Functional Overview

Figure 3-1, page 18 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 25 for details on the various transceiver features and parameters available.

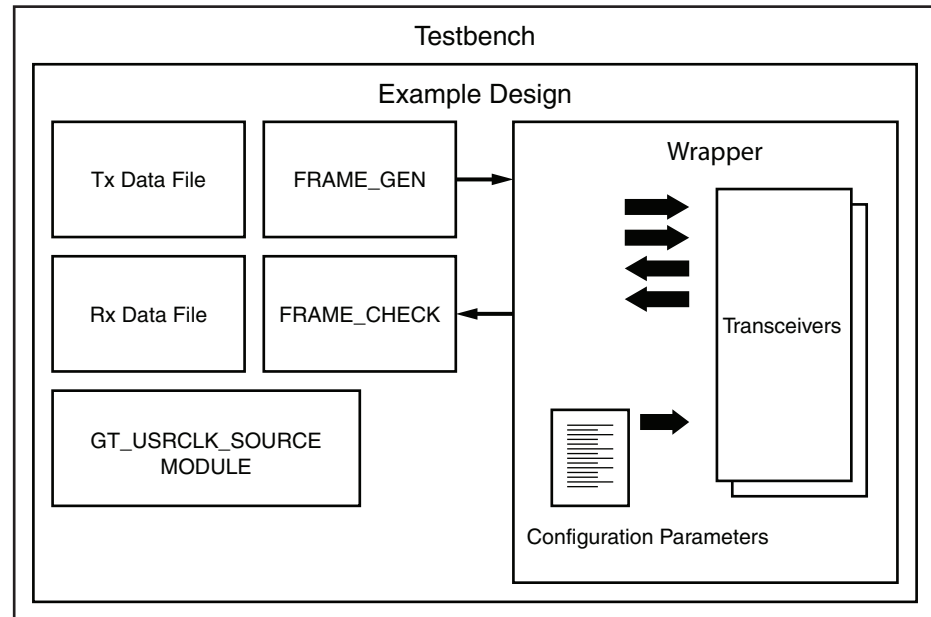


UG769\_c3\_01\_121610

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 for GTX, GTH, and GTP Transceivers](#), page 66.



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

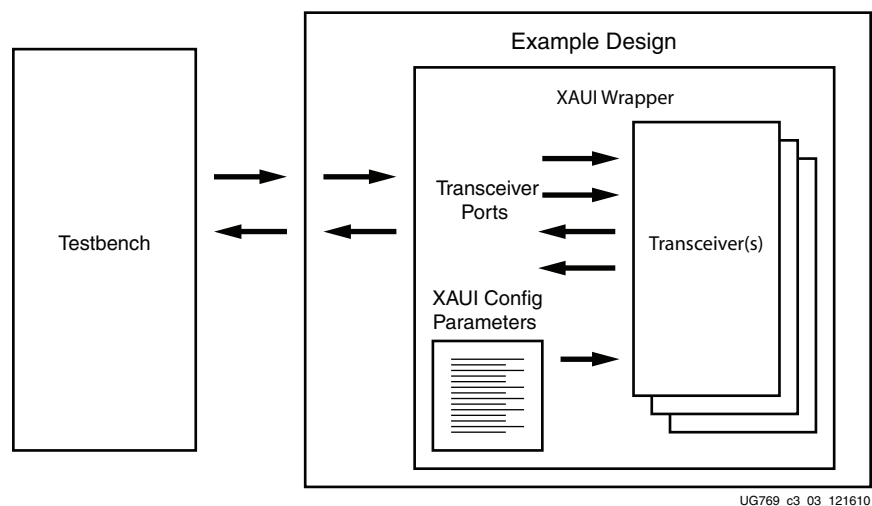
The example design covered in this section is a wrapper that configures a group of transceivers for use in a XAUI application. Guidelines are also given for incorporating the wrapper in a design and for the expected behavior in operation. For detailed information, see [Chapter 4, Quick Start Example 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 14.6 for synthesis and ModelSim 10.1a for simulation.

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



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

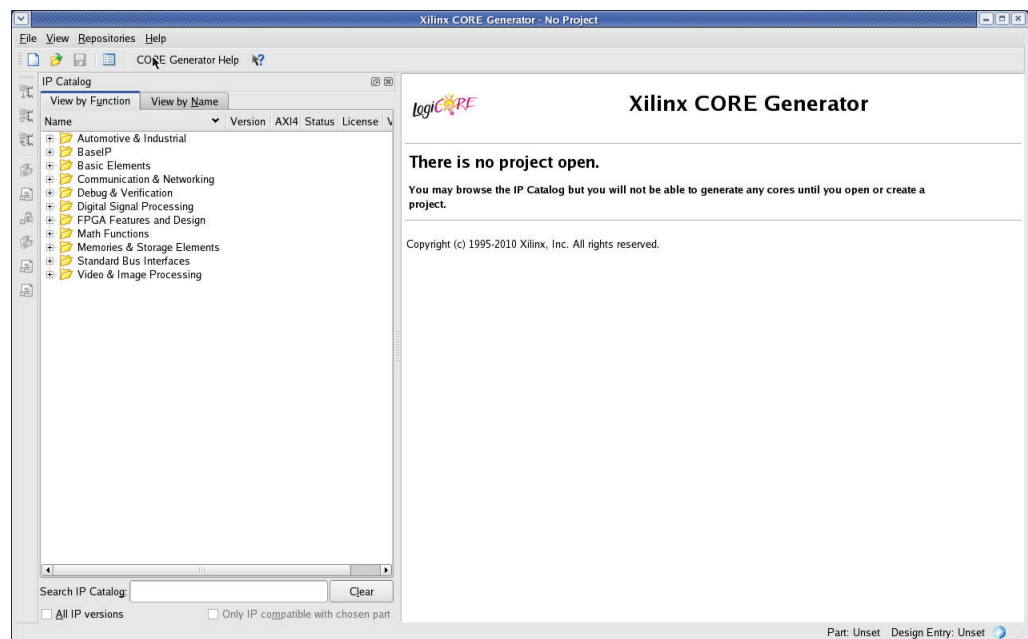
## Setting Up the Project

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 GTZ transceivers, start the Vivado® tools).  
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 or Figure 3-5).
4. Change the name of the CGP file (optional).
5. Click **Save**.



UG769\_c3\_04\_122010

Figure 3-4: Starting a New Project

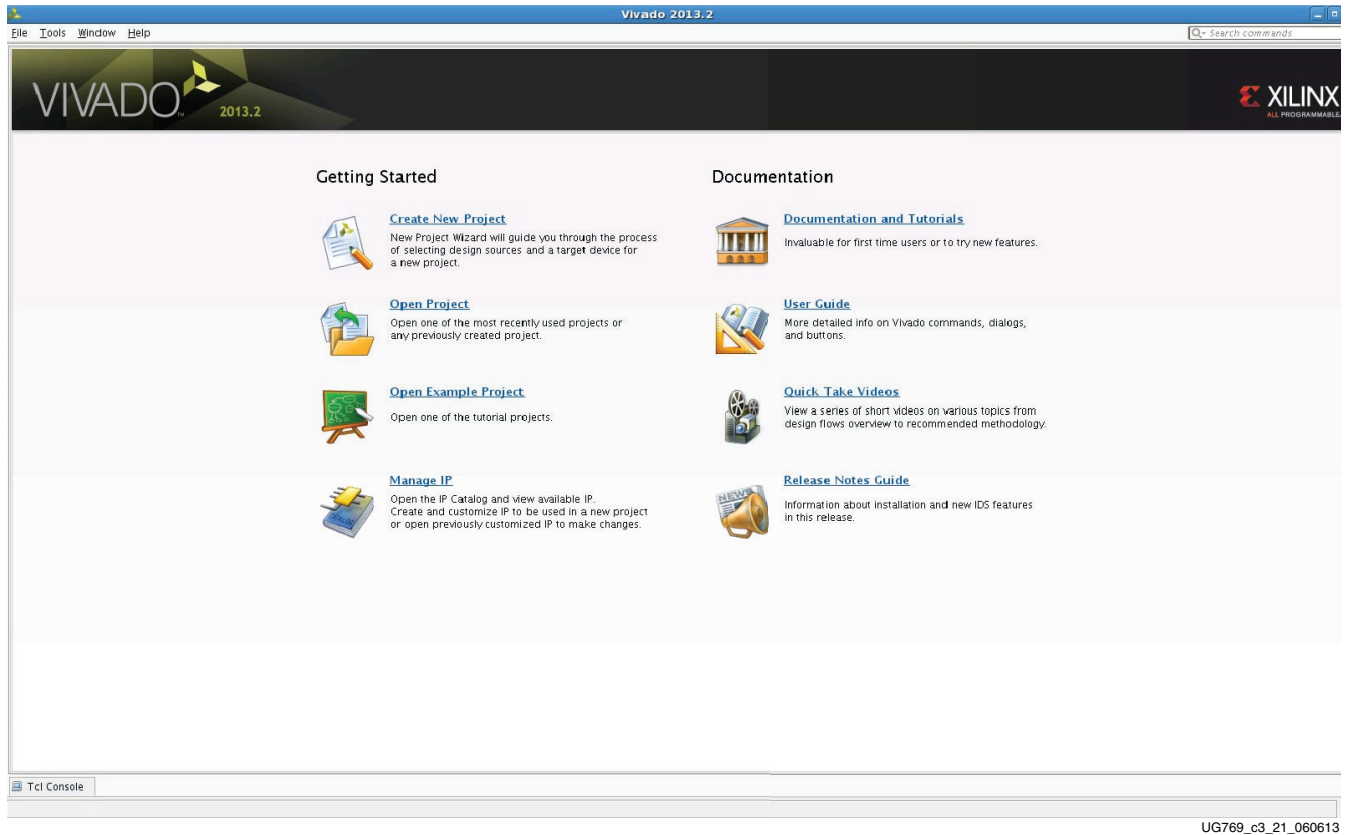


Figure 3-5: Starting a New Project (Vivado Tools)

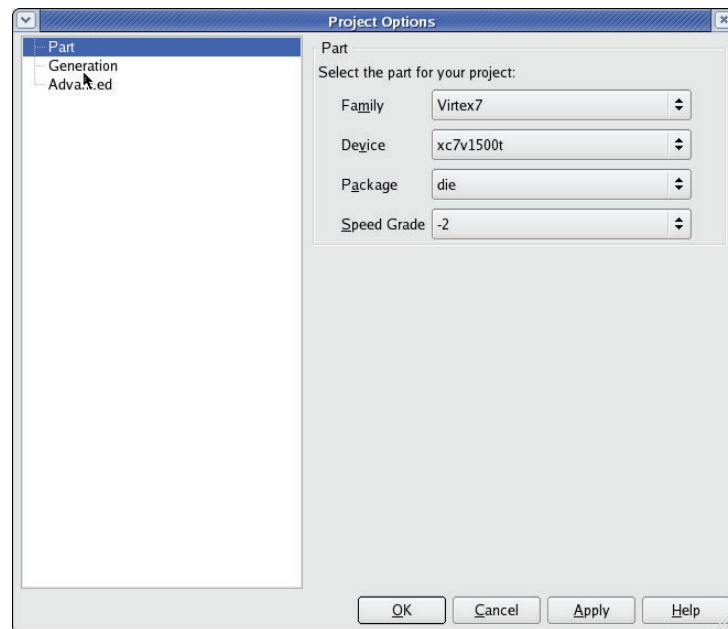
## Setting the Project Options

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 3-6](#) or [Figure 3-7](#)). For an example design using GTZ transceivers, select the XC7VH580T device.

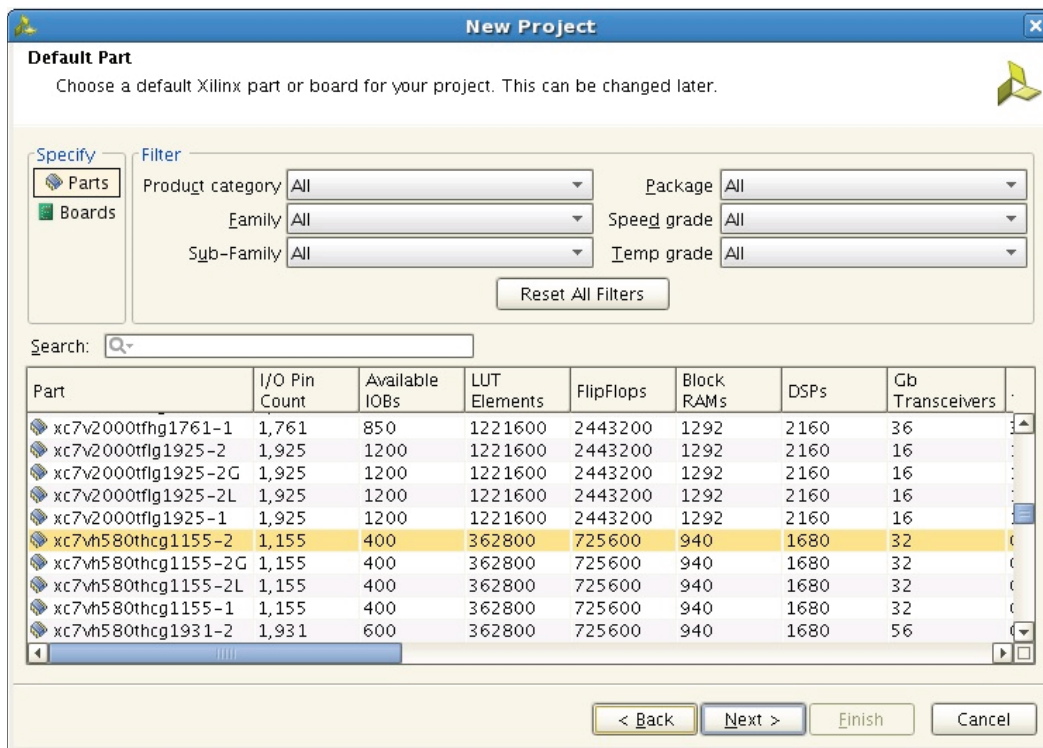
**Note:** If an unsupported silicon family is selected, the Wizard remains light grey in the taxonomy tree and cannot be customized. Only devices containing 7 series FPGAs transceivers are supported by the Wizard. See [DS180](#), *7 Series FPGAs Overview* for a list of devices containing the 7 series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

5. Click **Generation** in the option tree and select either Verilog or VHDL as the output language. For GTZ transceivers, select only Verilog. VHDL is not supported.
6. Click **OK**.



UG769\_c3\_05\_122010

Figure 3-6: Target Architecture Setting



UG769\_c3\_22\_060613

Figure 3-7: Target Architecture Setting (Vivado Tools)



## 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 2.6 in the taxonomy tree under:  
/FPGA Features & Design/IO Interfaces (See [Figure 3-8](#) or [Figure 3-9](#)).
2. Double-click **7 Series FPGAs Transceivers Wizard 2.6** to launch the Wizard.

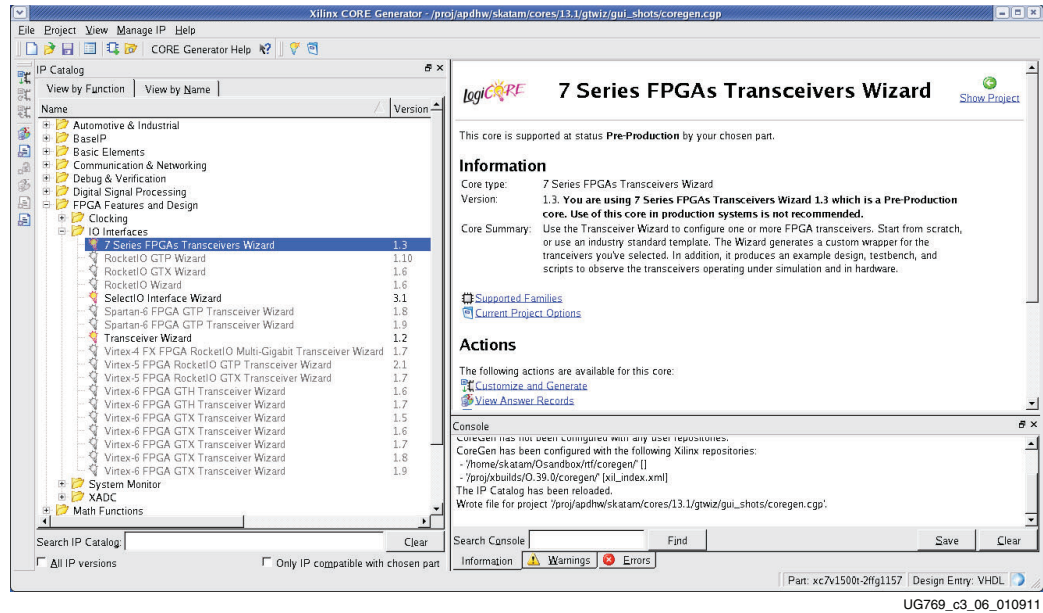
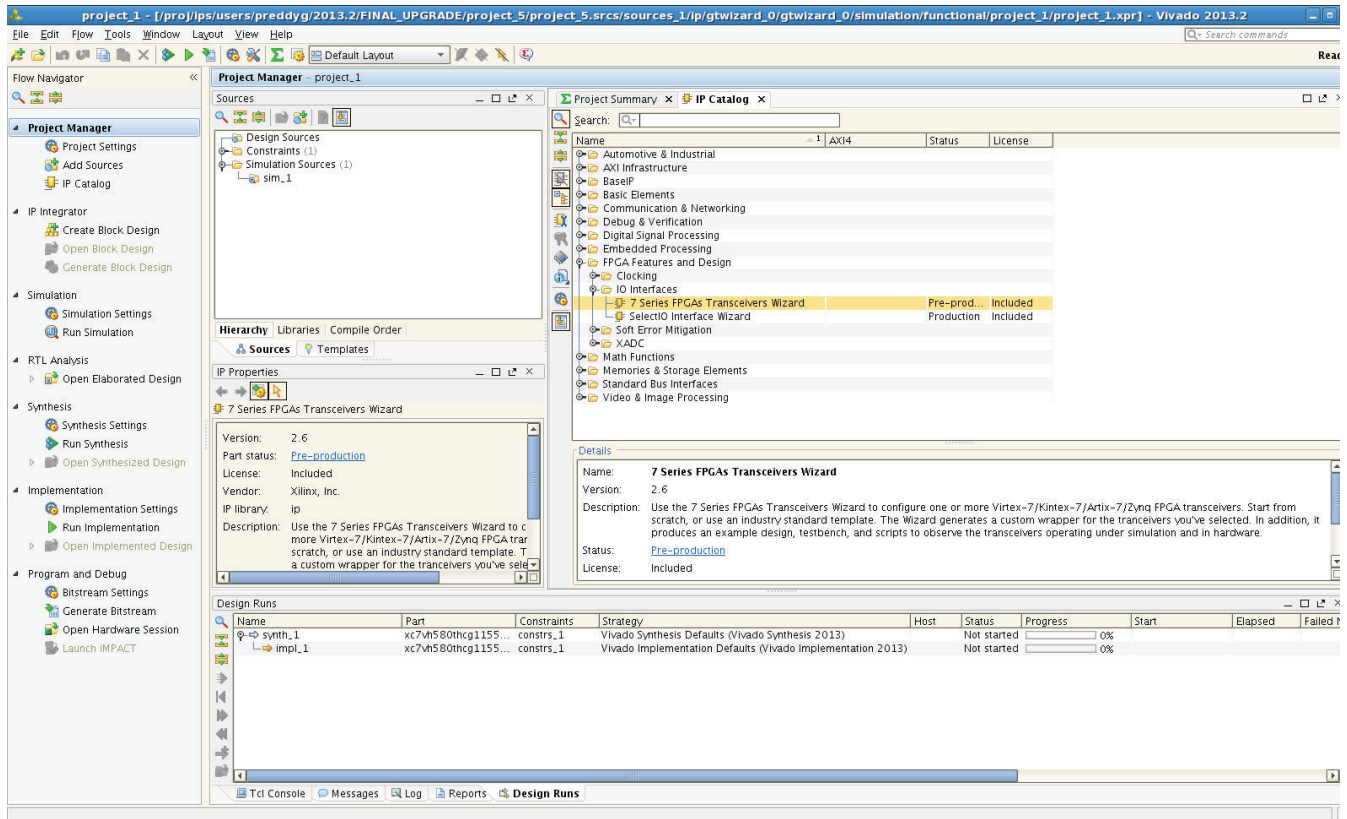


Figure 3-8: Locating the Transceiver Wizard



UG769\_c3\_23\_060613

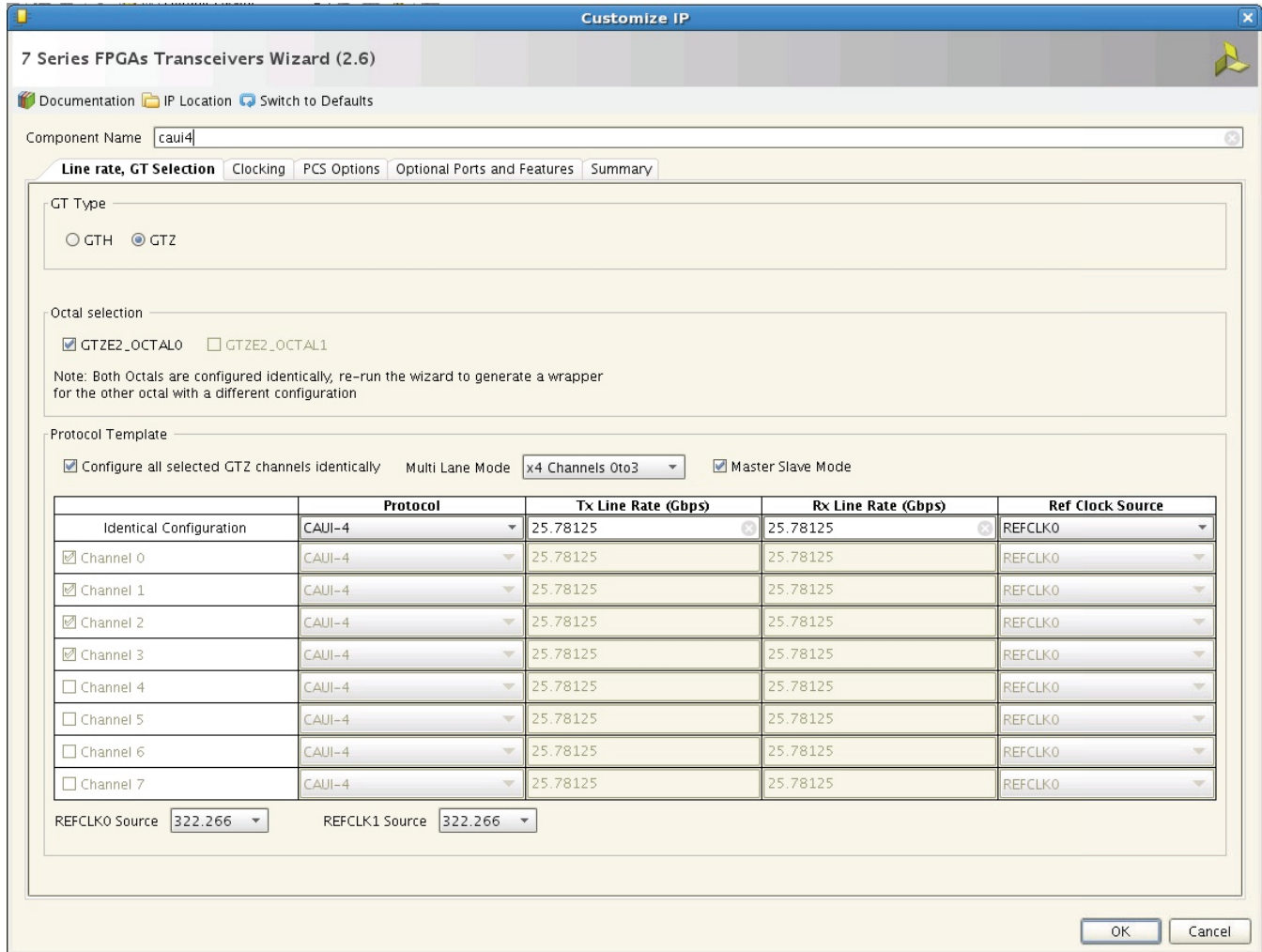
Figure 3-9: Locating the Transceiver Wizard (Vivado Tools)

## GTZ Transceivers

### Octal Selection, Channel Selection, Line Rate, and REFCLK

Page 1 of the Wizard (Figure 3-10) allows you to select the component name and determine the line rate and reference clock frequency. 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 `caui4_wrapper`.
2. In the `GT_Type` field, select **GTZ**. The type of transceiver depends on the device chosen in Project Options.
3. Select one or both of the octals `GTZE2_OCTAL0` or `GTZE2_OCTAL1`. The number of available octals depends on the target device and package.



UG769\_c3\_16\_060613

Figure 3-10: Line Rates and Transceiver Selection—GTZ Page 1

4. Select the multi-channel mode if you wish to set the octal in one of the multi-channel modes. Selecting a mode here will automatically select the channels appropriately too. If using one of the multi-channel modes, use the Master Slave mode to enable one of the channels as the master and the other as slaves.
5. The GTZ wizard supports both identical configuration and non-identical configuration of channels within an octal. To configure identically, select the **Configure all selected GTZ channels identically** checkbox.
6. 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 CAUI4 example uses the CAUI 4 protocol template. The CAUI 4 protocol template uses four channels.
7. Use [Table 3-1](#) to determine the line rate and reference clock settings. The line rate allowed is dependent on the speed grade of the device:
  - For -2G speed grade: 9.8 Gb/s – 14.025 Gb/s and 19.6Gb/s – 28.05 Gb/s
  - For other speed grades: 9.8 Gb/s – 12.890625 Gb/s and 19.6 Gb/s – 25.78125 Gb/s

Table 3-1: Line Rate and REFCLK Settings—GTZ Transceivers

| Options                | Description  |
|------------------------|--|
| Line Rate              | Set to the desired target line rate in Gb/s. Should be the same for both TX and RX.<br>The CAUI 4 example uses 25.78125 Gb/s.            |
| Reference Clock Source | Select either REFCLK0 or REFCLK1.<br>The CAUI 4 example uses REFCLK0.  |
| REFCLK0 Source         | Select from the list of the optimal reference clock frequency to be provided by the application.<br>The CAUI 4 example uses 322.266 MHz. |
| REFCLK1 Source         | Select from the list of the optimal reference clock frequency to be provided by the application.<br>The CAUI4 example uses 322.266 MHz.  |

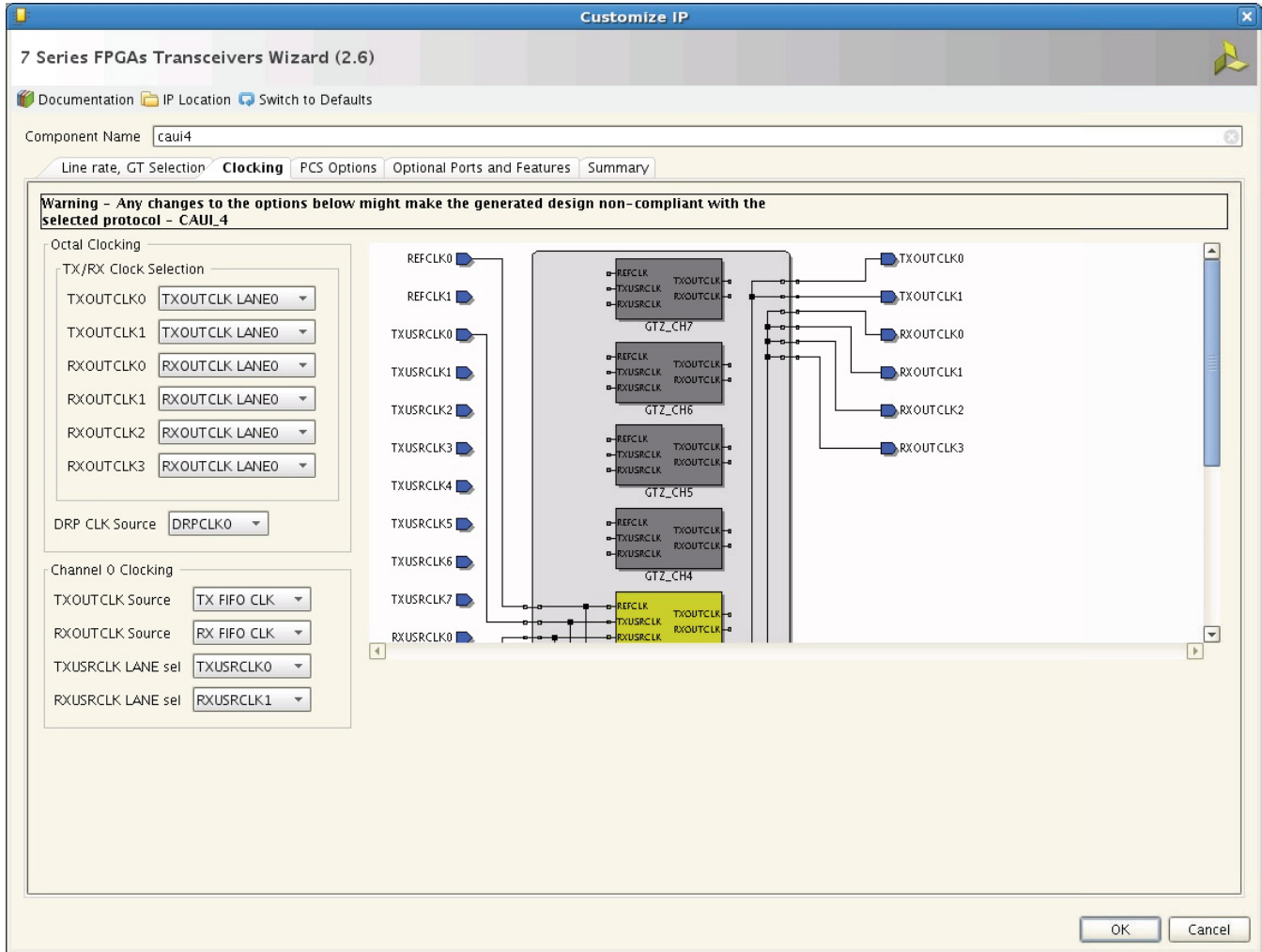
## Clocking

Page 2 of the Wizard ([Figure 3-11](#)) allows you to select the clocking for octal and the enabled channels within an octal. The user can verify all the settings and selections made on this page graphically with the image shown below the settings.

1. Select the source for TXOUTCLK0 and TXOUTCLK1 of the octal as per [Table 3-2](#).
2. Select the source for RXOUTCLK<0-3> of the octal as per [Table 3-2](#).

Table 3-2: TX/RXOUTCLKs of the Octal

| Options       | Description   |
|---------------|---|
| TXOUTCLK<0-1> | These can be sourced from the txoutclks from any of the individual channels (only channels enabled on page 1). This selection applies identically for all the octals enabled on page 1. |
| RXOUTCLK<0-3> | These can be sourced from the rxoutclks from any of the individual channels that were selected on the earlier page. This selection applies identically for all the octals on page 1.    |



UG769\_c3\_17\_060613

Figure 3-11: Octal and Channel Clocking—GTZ Page 2

3. Select the source for the user clocks TX/RXUSRCLK<0-7> (Table 3-3). The USRCLK numbering shown here in the GUI is relative to OCTAL0. These USRCLKs are mapped internally (by the wizard) to octal1 as per the following:

Table 3-3: TX/RXUSRCLKs of the Octal

| Options       | Description  |
|---------------|--|
| TXUSRCLK<0-7> | The available sources will be the TXOUTCLKs of the active octals selected in page 1. |
| RXUSRCLK<0-7> | The available sources will be the RXOUTCLKs of the active octals selected in page 1. |

- TX/RXUSRCLK0 - TX/RXUSRCLK4
- TX/RXUSRCLK1 - TX/RXUSRCLK5
- TX/RXUSRCLK2 - TX/RXUSRCLK6
- TX/RXUSRCLK3 - TX/RXUSRCLK7
- TX/RXUSRCLK4 - TX/RXUSRCLK0

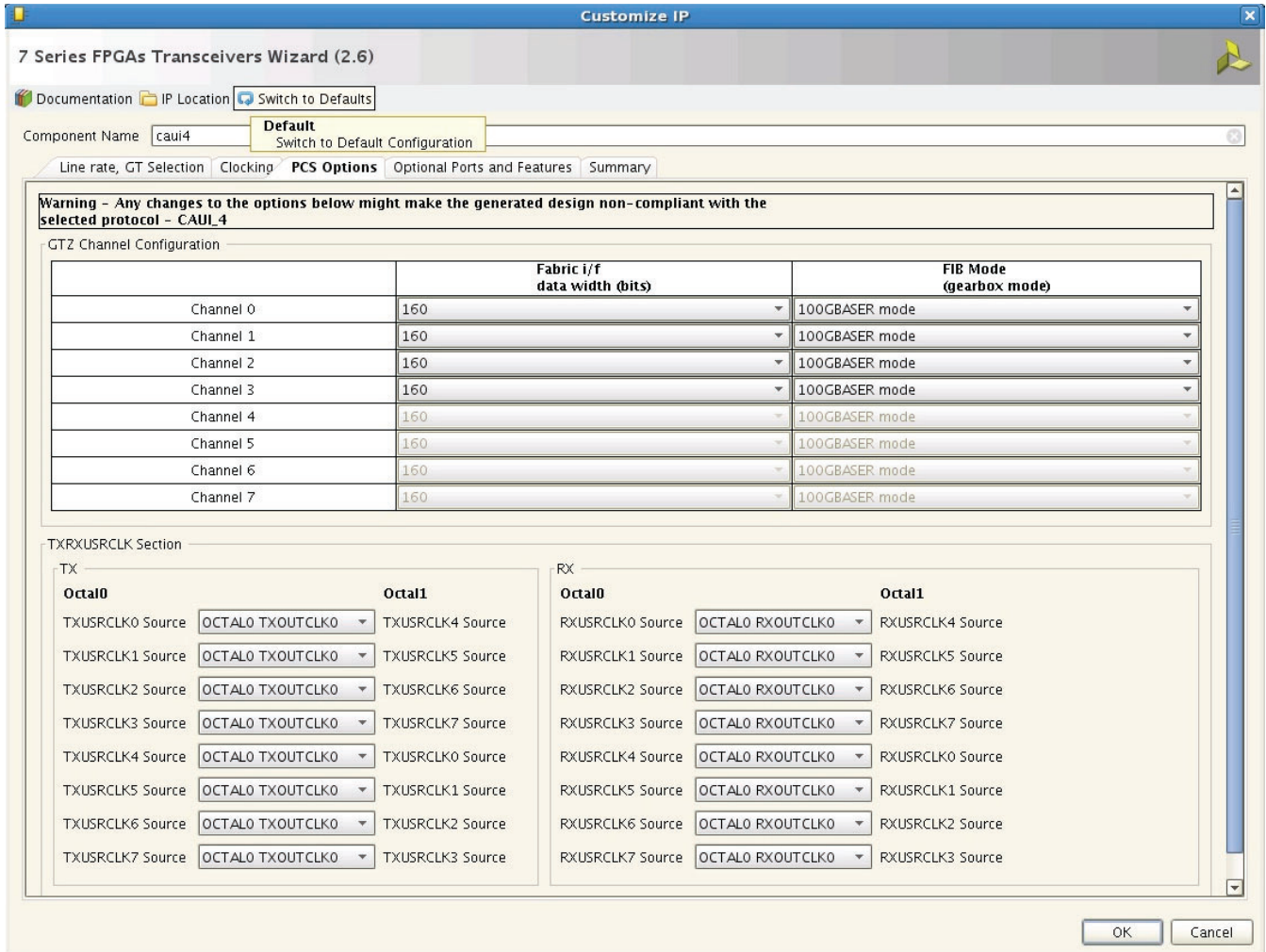
- TX/RXUSRCLK5 - TX/RXUSRCLK1
  - TX/RXUSRCLK6 - TX/RXUSRCLK2
  - TX/RXUSRCLK7 - TX/RXUSRCLK3
4. Channel clocking: First select the channel number for which you wish to configure the clocking.
  5. In the TX/RXOUTCLK source for the channel selected above, the only supported sources are TX/RX FIFO CLKs.
  6. In the TX/RXUSRCLK LANE sel, select the USRCLK that you wish to source for this channel. For channels of octal1, the USRCLK selection made here will be automatically mapped by the wizard as shown in [step 3](#).
  7. Select the source for the DRPCLK.

Table 3-4: Channel Clocking

| Options           | Description  |
|-------------------|--|
| TXOUTCLK source   | Select the TXOUTCLK source for each channel. Only the TX FIFO CLK is supported. Selecting the TX FIFO CLK selects TXOUTCLKPMA_DIV4 (refer to UG478, <i>7 Series FPGAs GTZ Transceivers User Guide</i> for more information). |
| RXOUTCLK source   | Select the RXOUTCLK source for each channel. Only the RX FIFO CLK is supported. Selecting the RX FIFO CLK selects RXOUTCLKPMA_DIV4 (refer to UG478, <i>7 Series FPGAs GTZ Transceivers User Guide</i> for more information). |
| TXUSRCLK lane sel | Select one among the available eight TX user clocks.   |
| RXUSRCLK lane sel | Select one among the available eight RX user clocks.   |

## PCS Modes

Page 3 of the Wizard (Figure 3-12) allows you to select the data width and PCS mode options.



UG769\_c3\_18\_060613

Figure 3-12: PCS Options—GTZ Page 3

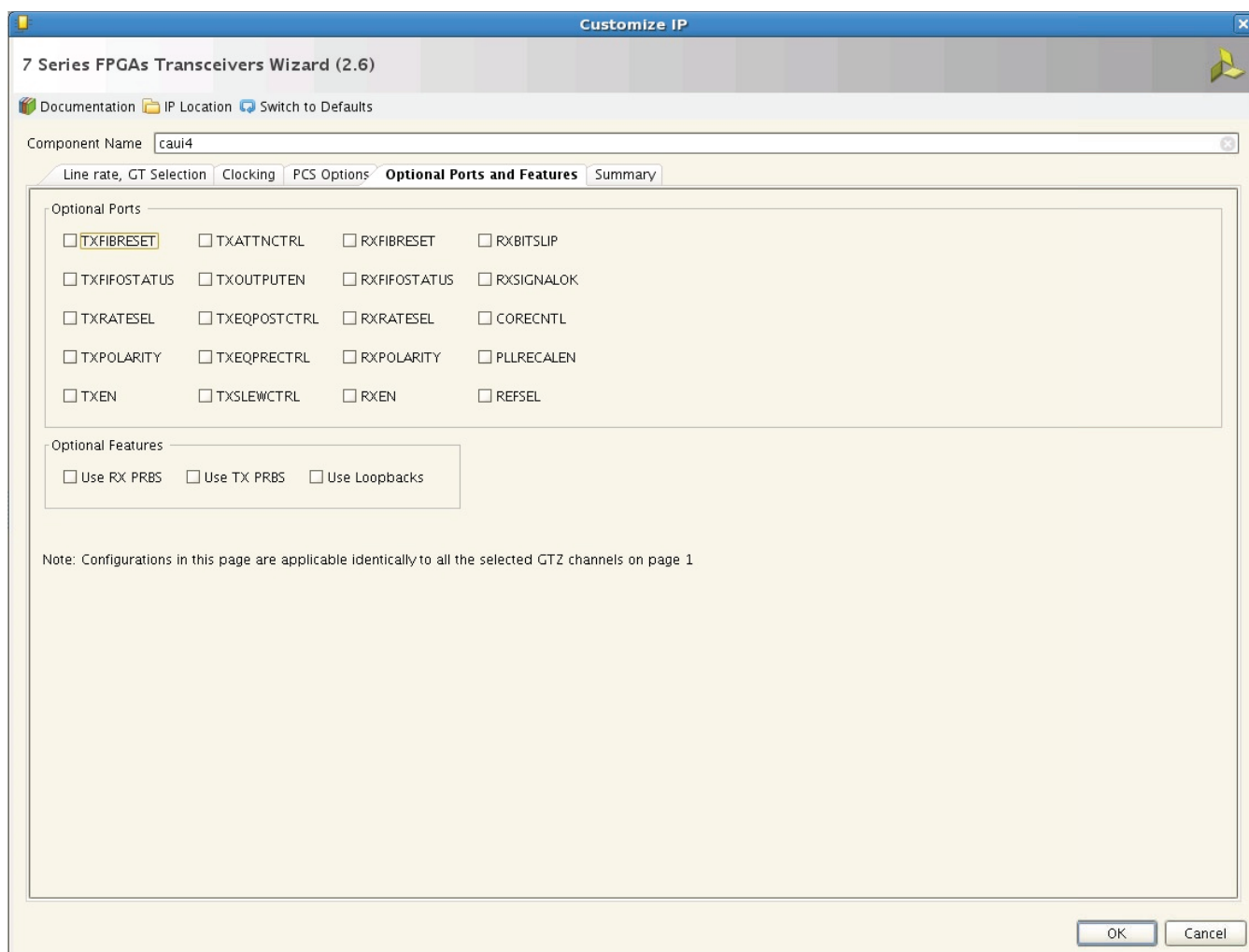
1. Select the data width for each of the channels enabled on page 1. The options shown here are dependent on the line rates entered on page 1. For 28.05, 27.95255, and 25.78125 Gb/s, only 160-bit mode is applicable. For 14.025 and 10.3125 Gb/s, both 160- and 80-bit modes are applicable. The example for the CAUI 4 template uses a 160-bit width.
2. Select the FIB mode options here. This is dependent on the data width selected above. For a data width of 160 bits, only 100GBASER mode is applicable. For 80 bits, both 100GBASER and 64B66B modes are applicable. The CAUI 4 protocol uses 100GBASER here.

Table 3-5: FIB Options

| Options                 | Description                      |
|-------------------------|----------------------------------|
| Fabric i/f data width   | Select the interface data width. |
| FIB mode (gearbox mode) | Select the gearbox mode.         |

## Optional Ports

Page 4 of the Wizard (Figure 3-13) allows you to select the optional ports that you want to bring out to the example top and multi gt wrapper.



UG769\_c3\_19\_060613

Figure 3-13: Optional Ports—GTZ Page 4

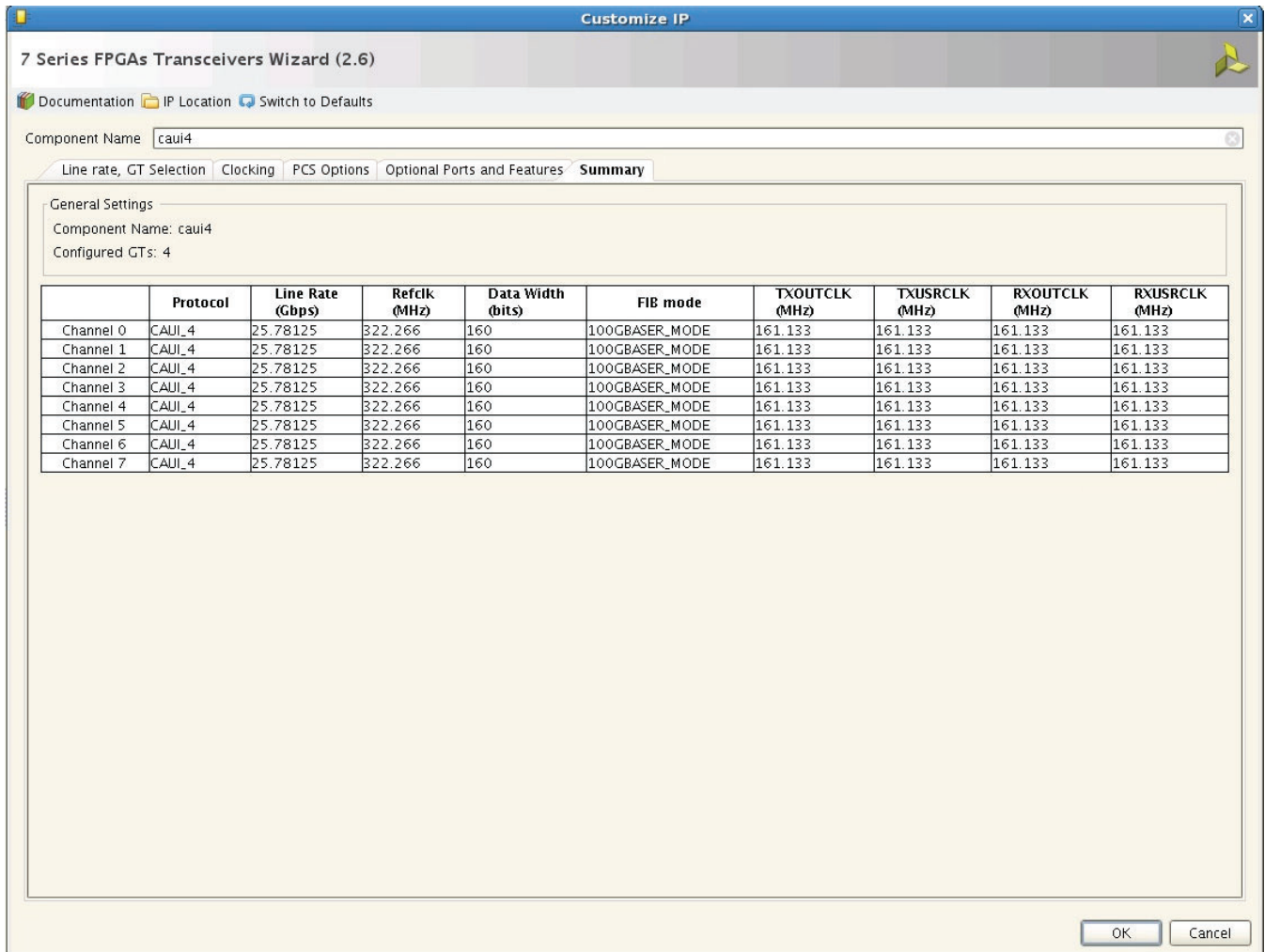


**Table 3-6: Optional Ports for GTZ Transceivers**

| Options      | Description  |
|--------------|--|
| TXFIBRESET   | Brings out the TXFIBRESET ports to the example design from which the user can control the RESET of the FIB portion of the GTZ transceiver. |
| RXFIBRESET   | Brings out the RXFIBRESET ports to the example design from which the user can control the RESET of the FIB portion of the GTZ transceiver. |
| TXFIFOSTATUS | This brings out the TXFIFOSTATUS port to the example design allowing users to learn the FIFO status.                                       |
| RXFIFOSTATUS | This brings out the RXFIFOSTATUS port to the example design allowing users to learn the FIFO status.                                       |
| TXRATESEL    | Brings the TXRATESEL ports out onto the example top level. These ports are used to control the TX PLL divider ratios.                      |
| RXRATESEL    | Brings the RXRATESEL ports out onto the example top level. These ports are used to control the RX PLL divider ratios.                      |
| TXPOLARITY   | Brings out the TXPOLARITY port to the example design.  |
| RXPOLARITY   | Brings out the RXPOLARITY port to the example design.  |
| TXEN         | Brings out the TXEN port to the example design.  |
| RXEN         | Brings out the RXEN port to the example design.  |
| TXOUTPUTEN   | Brings out the TXOUTPUTEN port to the example design.  |
| TXATTNCTRL   | Brings out the TXATTNCTRL port to the example design.  |
| TXEQPOSTCTRL | Brings out the TXEQPOSTCTRL port to the example design.  |
| TXEQPRECTRL  | Brings out the TXEQPRECTRL port to the example design.   |
| TXSLEWCTRL   | Brings out the TXSLEWCTRL port to the example design.  |
| RXBITSLIP    | Brings out the RXBITSLIP port onto the example design. This port can be used to slip data in raw mode.                                     |
| RXSIGNALOK   | Brings out the RXSIGNALOK port onto the example design.  |
| CORECNTL     | Brings out the CORECNTL ports onto the example design.   |
| REFSEL       | Brings out the REFSEL ports onto the example design.   |
| PLLRECALEN   | Brings out the PLLRECALEN ports onto the example design.   |
| RXPRBS       | Brings out all the RXPRBS related ports onto the example design.   |
| TXPRBS       | Brings out all the TXPRBS related ports onto the example design.   |
| LOOPBACK     | Brings out all the LOOPBACK control ports onto the example design.   |

## Summary

Page 5 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.



UG769\_c3\_20\_060613

Figure 3-14: Summary Page—GTZ Page 5

## GTX, GTH, and GTP Transceivers

### GT Type Selection

Page 1 of the Wizard (Figure 3-15) allows you to select the component name and determine the line rate and reference clock frequency. 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** if you wish to manually set all parameters.
3. In the GT\_Type field, select **GTX**, **GTH**, or **GTP**. The type of transceiver depends on the device chosen in Project Options.

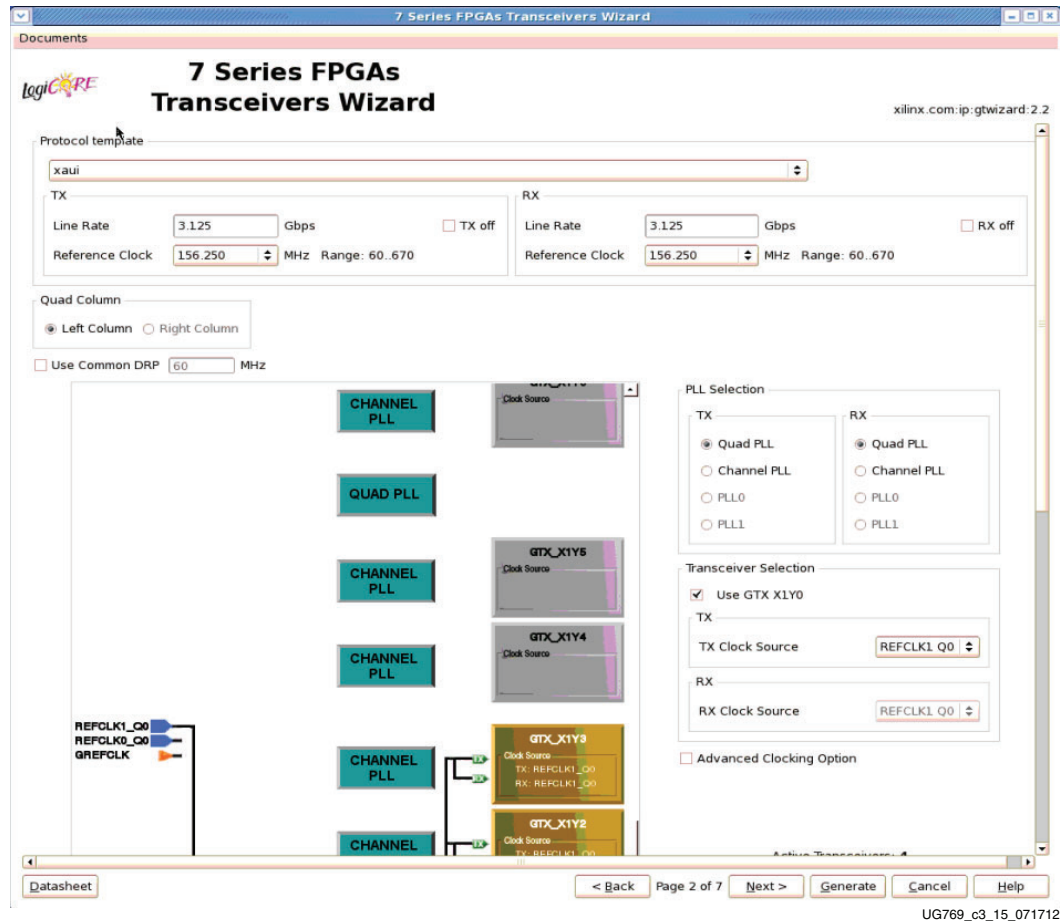
Select one of the available protocols from the list to begin designing with a predefined protocol template. The XAUI example uses the XAUI protocol template.



Figure 3-15: Line Rates and Transceiver Selection—Page 1

## Line Rate, Transceiver Selection, and Clocking

Page 2 of the Wizard (Figure 3-15) allows you to select the transceiver location and clocking. The number of available transceivers appearing on this page depends on the selected target device and package. The XAUI example design uses four transceivers.



UG769\_c3\_15\_071712

Figure 3-16: Transceiver Selection and Clocking—Page 2

Use Table 3-7 and Table 3-8 to determine the line rate and reference clock settings.

Table 3-7: TX Settings

| Options         | Description  |
|-----------------|--|
| Line Rate       | Set to the desired target line rate in Gb/s. Can be independent of the receive line rate.<br>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.<br>The XAUI example uses 156.25 MHz. |

Table 3-7: TX Settings (Cont'd)

| Options | Description   |
|---------|---|
| TX off  | Selecting this option disables the TX path of the transceiver. The transceiver will act as a receiver only.<br>The XAUI example design requires both TX and RX functionality. |

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

Table 3-8: RX Settings

| Options         | Description  |
|-----------------|--|
| Line Rate       | Set to the desired target line rate in Gb/s.<br>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.<br>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.<br>The XAUI example design requires both TX and RX functionality. |

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

Use Tables 3-9 through 3-12 to determine the optional ports settings available on this page.

Table 3-9: 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.          |
| 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 3-10: 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 3-12). 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 3-11: PLL Selection

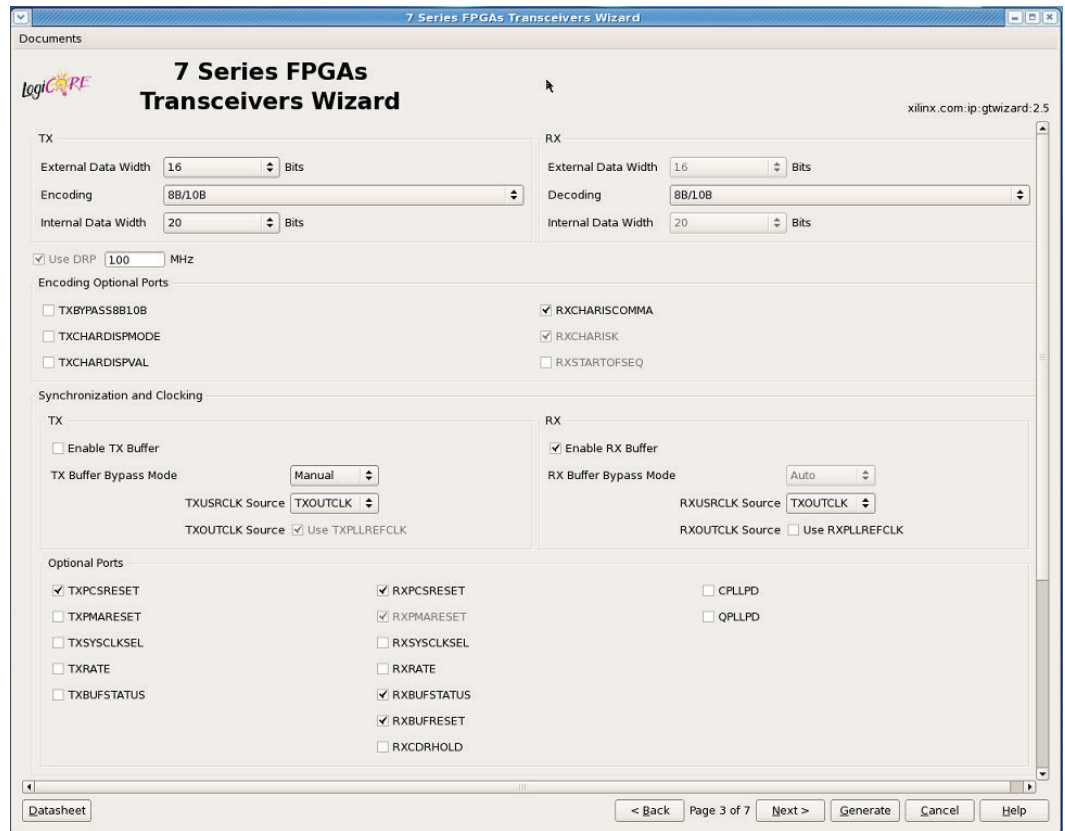
| Option | Description  |
|--------|--|
| QPLL   | GTX and GTH transceivers: Use the Quad PLL when all four transceivers of the quad are used to save power. Quad PLL is shared across four transceivers of a quad. |
| CPLL   | GTX and GTH transceivers: Use the Channel PLL based on the line rate supported by the selected transceiver.  |
| PLL0   | GTP transceivers only: PLL0 is shared across four transceivers of a quad.  |
| PLL1   | GTP transceivers only: PLL1 is shared across four transceivers of a quad.  |

Table 3-12: 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 3 of the Wizard (Figure 3-17) allows you to select encoding and 8B/10B optional ports. Tables 3-13 through 3-20 list the available options.



UG769\_c3\_08\_031813

Figure 3-17: Encoding and Optional Ports—Page 3

Table 3-13: 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 | GTX transceivers only: 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 | GTX transceivers only: 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 3-14: 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.   |

**Notes:**

- Options not used by the XAUI example are shaded.
- RX settings should be the same as TX settings.

Table 3-15: 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/GTH Transceivers User Guide* for details on this setting. The RX PCS/PMA alignment setting controls whether the RX phase alignment circuit is enabled.

[Table 3-16](#) shows the optional ports for 8B/10B.

Table 3-16: 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 3-17](#) shows the TX and RX buffer bypass options.

Table 3-17: TX and RX Buffer Bypass Options

| Option | Description           |  |
|--------|-----------------------|--|
| TX     | Enable TX Buffer      | If the Enable TX Buffer checkbox is checked, the TX buffer in the transceiver is enabled. This buffer can be bypassed for low, deterministic latency.  |
| TX     | Tx Buffer Bypass Mode | This option is only if the Tx Buffer is bypassed. It is mandatory to use the manual mode. See <a href="#">UG476</a> , <i>7 Series FPGAs GTX/GTH Transceivers User Guide</i> for details.                     |
| RX     | Enable RX Buffer      | If the Enable RX Buffer checkbox is checked, the RX elastic buffer in the transceiver is enabled. This buffer can be bypassed for low, deterministic latency.  |
| RX     | Rx Buffer Bypass Mode | This option is visible only if the RX Buffer is bypassed. Auto mode is the recommended setting. To use manual mode, refer to <a href="#">UG476</a> , <i>7 Series FPGAs GTX/GTH Transceivers User Guide</i> . |

Table 3-18 details the TXUSRCLK and RXUSRCLK source signal options.

Table 3-18: 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 3-19 details the TXOUTCLK and RXOUTCLK source signal options.

Table 3-19: 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/GTH Transceivers User Guide](#) for more information on TXOUTCLK and RXOUTCLK control.

Table 3-20 shows the optional ports available for latency and clocking.

Table 3-20: 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.   |
| QPLLPD      | Visible only when GTX or GTH transceiver is selected. Powerdown port for QPLL.  |
| CPLLPD      | Visible only when GTX or GTH transceiver is selected. Powerdown port for CPLL.  |
| PLL0PD      | Visible only when GTP transceiver is selected. Powerdown port for PLL0.   |
| PLL1PD      | Visible only when GTP transceiver is selected. Powerdown port for PLL1.   |

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

If the PCI Express® protocol is selected, page 3 of the Wizard appears as shown in Figure 3-18.

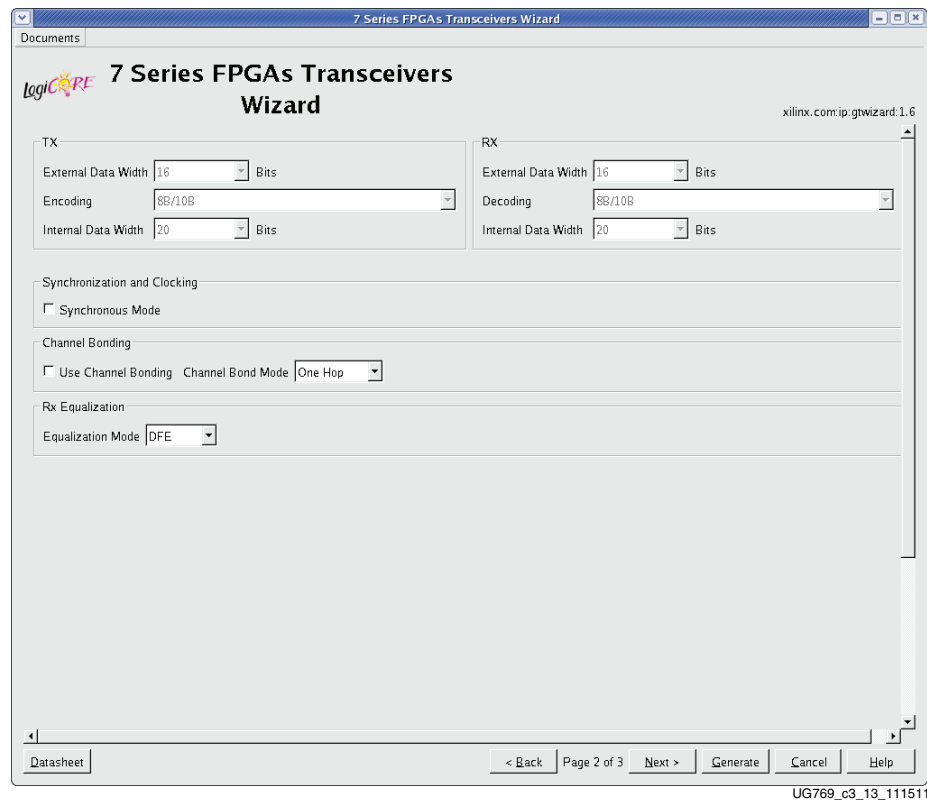


Figure 3-18: Options for PCI Express Wrapper—Page 3

Table 3-21 shows the options for customization of the PCI Express wrapper.

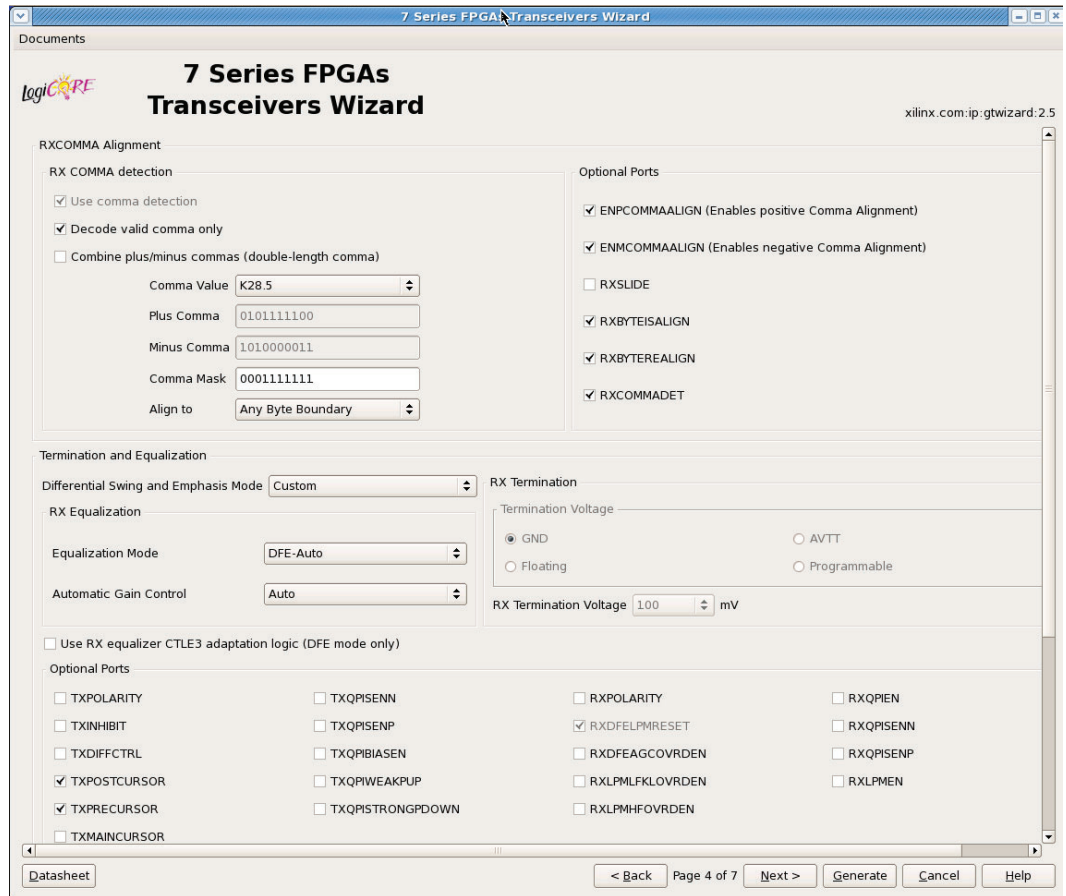
Table 3-21: Customization Options for PCI Express Wrapper

| Option              | Description  |
|---------------------|--|
| Synchronous Mode    | Select this mode if the PPM offset is 0.   |
| 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 align the data on multiple lanes. |
| Channel Bond Mode   | Selects the Channel Bond logic implementation mode. See <a href="#">UG476</a> , <i>7 Series FPGAs GTX/GTH Transceivers User Guide</i> for details on Channel Bond Mode.                                      |
| Equalization        | Sets the equalization mode in the receiver. See <a href="#">UG476</a> , <i>7 Series FPGAs GTX/GTH Transceivers User Guide</i> for details on the decision feedback equalizer.                                |

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

## Alignment, Termination, and Equalization

Page 4 of the Wizard (Figure 3-19) allows you to set comma characters and control receive equalization and terminal voltage.



UG769\_c3\_09\_031813

Figure 3-19: Synchronization and Alignment—Page 4

Table 3-22 shows the receive comma alignment settings.

Table 3-22: 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 3-23 details the preemphasis and differential swing settings.

Table 3-23: 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</a> , <i>7 Series FPGAs GTX/GTH Transceivers User Guide</i> for details. |

Table 3-24 describes the RX equalization settings.

Table 3-24: RX Equalization

| Option  | Description  |
|---|--|
| Equalization Mode                                       | Sets the equalization mode in the receiver. See <a href="#">UG476</a> , <i>7 Series FPGAs GTX/GTH Transceivers User Guide</i> 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.   |
| Use RX Equalizer CTLE3 Adaptation Logic (DFE mode only) | Applicable only to GTX transceivers. If checked, the CTLE3 adaptation logic is instantiated in the example design. For more information, refer to <a href="#">Chapter 5, Detailed Example Design</a> .             |

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

Table 3-25 describes the RX termination settings.

Table 3-25: 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 3-26 lists the optional ports available on this page.

Table 3-26: Optional Ports

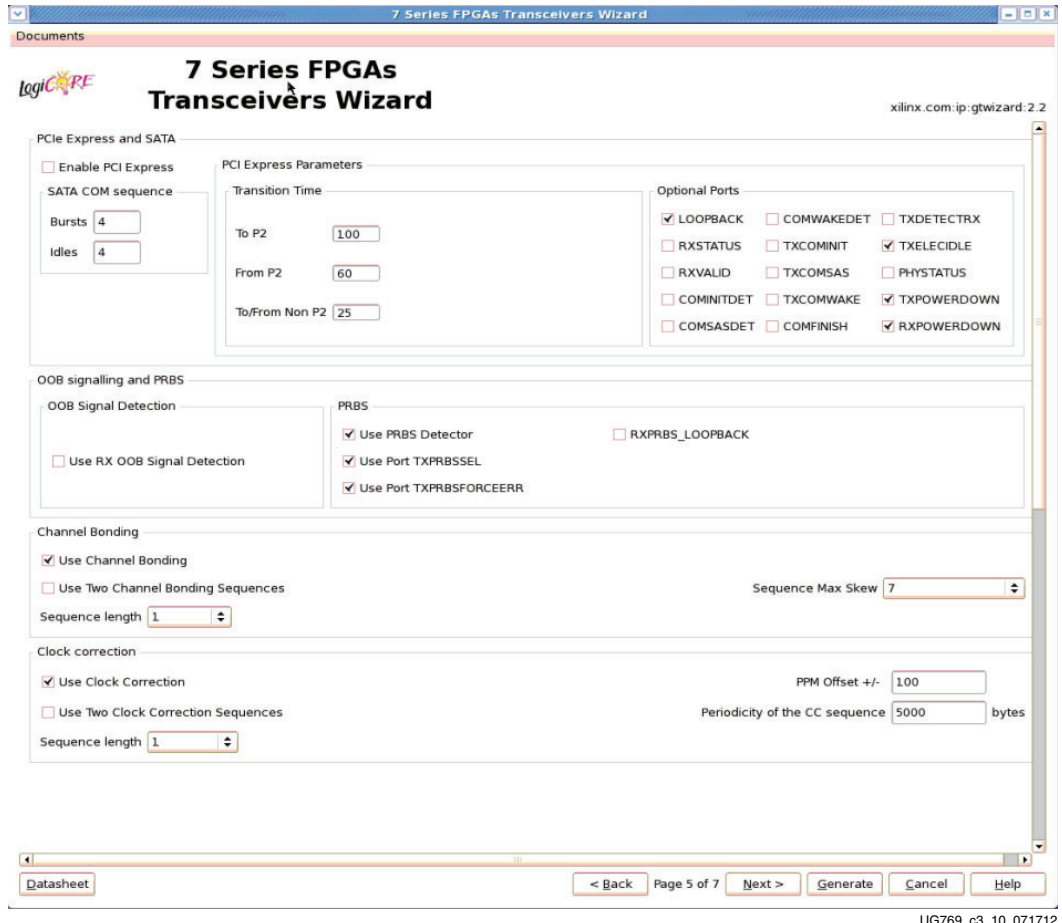
| Option           | Description  |
|------------------|--|
| TXPOLARITY       | Active-High signal to invert the polarity of the transmitter output. |
| TXINHIBIT        | Active-High signal forces transmitter output to steady state.        |
| RXPOLARITY       | Active-High signal inverts the polarity 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.   |
| TXPRECURSOR      | TXPRECURSOR 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.                                     |
| RXLPLMKLOVRDEN   | Receive low pass override enable port.                               |
| TXQPISTRONGPDOWN | Transmit QPI power down port.  |
| RXDPELPMRESET    | Resets the receive DFE/LPM 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 5 of the Wizard (Figure 3-20) 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\_071712

Figure 3-20: PCI Express, SATA, OOB, PRBS, Channel Bonding, and Clock Correction Selection—Page 5

Table 3-27 details the receiver SATA configuration options.

Table 3-27: 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 3-28 details the receiver PCI Express configuration options.

Table 3-28: PCI Express and SATA 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/GTH 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/GTH 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/GTH 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. |

Table 3-28: PCI Express and SATA Parameters (Cont'd)

| Option      | Description   |
|-------------|---|
| 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.   |
| TXPOWERDOWN | Powerdown port for the transmitter.   |
| RXPOWERDOWN | Powerdown port for the receiver.  |

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

Table 3-29 shows the OOB signal detection options.

Table 3-29: 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 3-30 details the PRBS settings.

Table 3-30: 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.  |

Table 3-30: PRBS Detector (Cont'd)

| Option                  | Description   |
|-------------------------|---|
| 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 3-31 shows the channel bonding options.

Table 3-31: 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.<br>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.<br>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 3-32 shows the clock correction options.

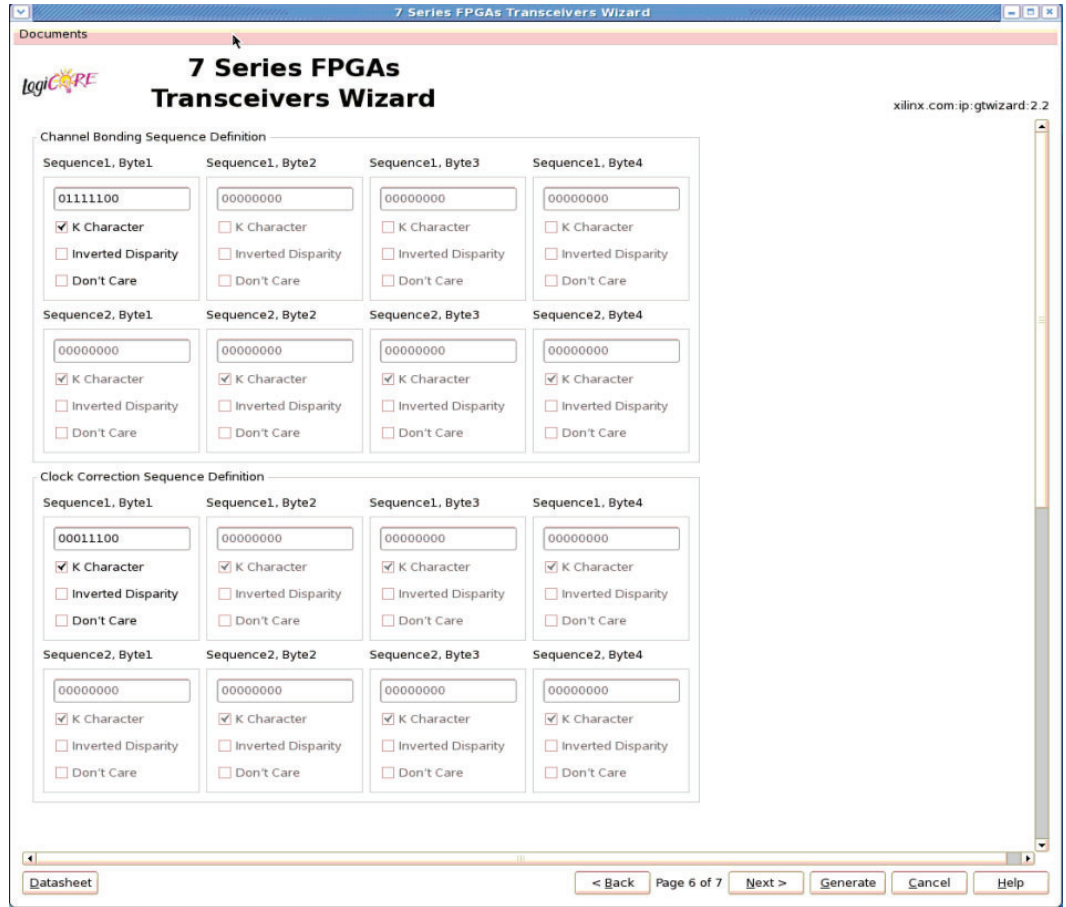
Table 3-32: Clock Correction Setup

| 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.<br>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 6 of the Wizard (Figure 3-21) allow you to define the channel bonding sequence(s). Table 3-33 describes the sequence definition settings and Table 3-32, page 52 describes the clock setup settings.



UG769\_c3\_11\_071712

Figure 3-21: Channel Bonding and Clock Correction Sequence—Page 6

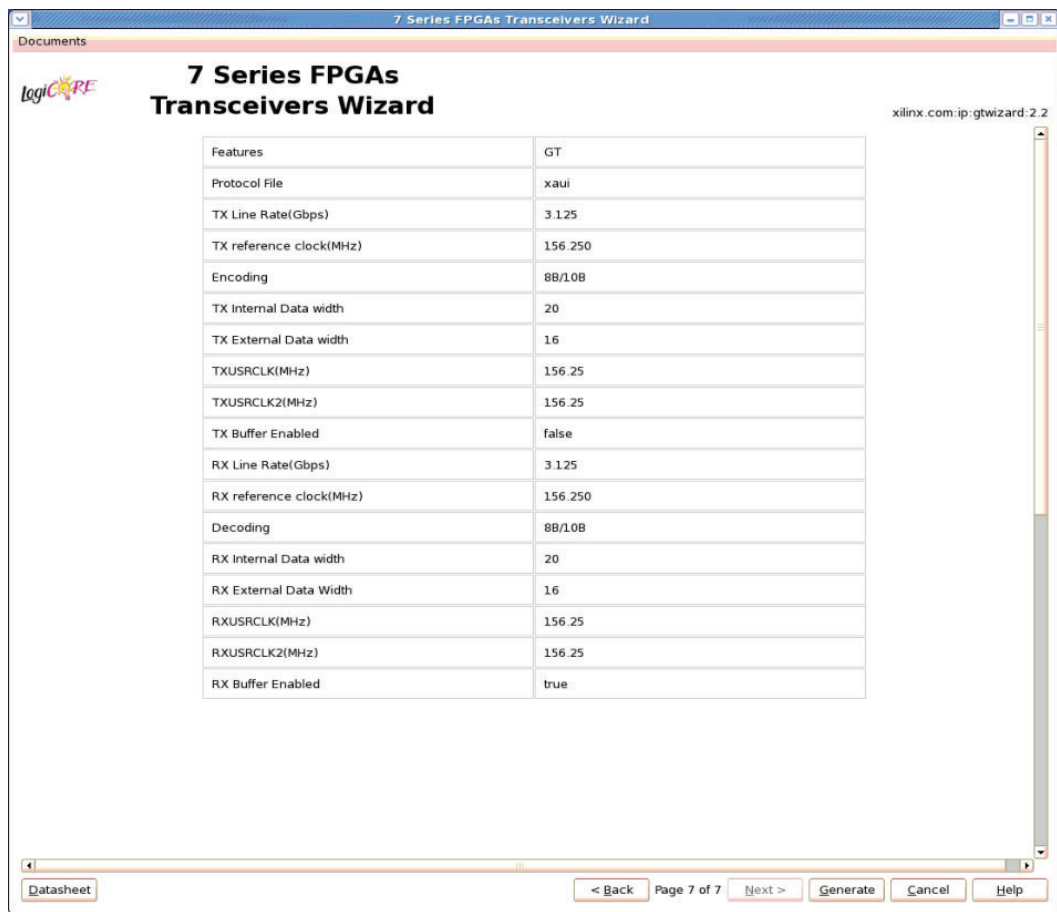
Table 3-33: Channel Bonding and Clock Correction Sequences

| 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 7 of the Wizard (Figure 3-22) provides a summary of the selected configuration parameters. After reviewing the settings, click **Generate** to exit and generate the wrapper.



UG769\_c3\_12\_071712

Figure 3-22: 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 Using the ISE Tools

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 14.6 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<br><Xilinx dir>/secureip/mti        |
| VHDL    | UNISIM      | <Xilinx dir>/vhdl/src/unisims/primitive<br><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 Using the ISE Tools

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 Using the ISE Tools

The Wizard provides a script to observe the behavior of the example design during timing simulations. Timing simulations are not supported for the GTZ wizard.

### 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 14.6 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<br><Xilinx dir>/secureip/mti        |
| VHDL    | SIMPRIM      | <Xilinx dir>/vhdl/src/simprims/primitive<br><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 in the ISE Tools

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>/src**  
Core verilog sources required
    -  **<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.   |
| <component_name>_octal0.v    | Specific to GTZ transceivers. Individual transceiver wrapper to be instantiated in the main transceiver wrapper. Instantiates the selected transceivers with settings for the selected protocol.   |
| <component_name>_octal1.v    | Specific to GTZ transceivers. 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. |

**Table 5-2: Transceiver Wrapper Component Name (Cont'd)**

| Name                | Description   |
|---------------------|---|
| <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 v2.6 User Guide</i> |

[Back to Top](#)

## <component name>/src

The src directory contains the source code files that are mandatory for certain cores.

**Table 5-4: Src Directory**

| Name                               | Description  |
|------------------------------------|--|
| <project_dir>/<component_name>/src |  |
| gtwizard_v2_6_beachfront.v         | This file should be used by the user to interface user logic with the GTZ transceiver. |

[Back to Top](#)

## <component name>/example design

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

**Table 5-5: 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.  |
| frame_gen_top.v                               | Frame-generator related files are instantiated within this top-level wrapper. Instantiations include modules such as scrambler and gt_frame_gen. |

Table 5-5: Example Design Directory (Cont'd)

| Name                                    | Description  |
|---|--|
| frame_check_top.v                       | Frame-checker related files are instantiated within this top-level wrapper. Instantiations include modules such as descrambler, gt_frame_check, and block_sync_sm.               |
| gt_attributes.ucf                       | Constraints file containing the transceiver attributes generated by the transceiver Wizard GUI settings.   |
| <component_name>_exdes.ucf              | Constraint file for mapping the transceiver wrapper example design onto a Kintex™-7 or Virtex®-7 FPGA. This file is not generated for GTZ transceivers.                          |
| <component_name>_exdes.xcf              | XST specific constraint file for mapping the transceiver wrapper example design onto a Kintex-7 or Virtex-7 FPGA. This file is not generated for GTZ transceivers.               |
| <component_name>_exdes.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. This file is not generated for GTZ transceivers.  |
| 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>_exdes.xdc              | Xilinx Design Constraint file for mapping the transceiver wrapper example design onto a Kintex-7 or Virtex-7 FPGA.   |
| <component_name>_exdes_synplify.sdc     | Synopsys Design Constraint file for mapping the transceiver wrapper example design onto a Kintex-7 or Virtex-7 FPGA. This file is not generated for GTZ transceivers.            |
| <component_name>_init.v[hd]             | Initialization module which contains the reset and buffer bypass logic.  |
| tx_startup_fsm.v[hd]                    | Reset module for transmitter. This file is not generated for GTZ transceivers.   |
| rx_startup_fsm.v[hd]                    | Reset module for receiver.   |
| recclk_monitor.v[hd]                    | Monitor to determine whether the recovered clock is stable. This file is not generated for GTZ transceivers.   |
| 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.   |

[Back to Top](#)

**<component name>/implement**

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

**Table 5-6: Implement Directory**

| Name                                     | Description   |
|--|---|
| <project_dir>/<component_name>/implement |   |
| chipscope_project.cpj                    | ChipScope Pro project file.   |
| 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.                       |
| implement_synplify.bat                   | A Windows batch file that processes the example design through Synplify synthesis and the tool flow. This file is not generated for GTZ transceivers. |
| implement_synplify.sh                    | A Linux shell script that processes the example design through Synplify synthesis and the tool flow. This file is not generated for GTZ transceivers. |
| synplify.prj                             | Synplify project file for the example design. This file is not generated for GTZ transceivers.  |
| v7ht.tcl                                 | Encrypted TCL file needed for implementation of the GTZ transceivers.   |

[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 5-7: 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 5-8: Simulation Directory**

| Name                                      | Description  |
|---|--|
| <project_dir>/<component_name>/simulation |  |
| <component_name>_tb.v[hd]                 | Testbench to perform functional simulation of the provided example design. See <a href="#">Functional Simulation of the Example Design Using the ISE Tools, page 55</a> .  |
| <component_name>_tb_imp.v[hd]             | Testbench to perform timing simulation of the provided example design. See <a href="#">Timing Simulation of the Example Design Using the ISE Tools, page 56</a> . This file is not generated for GTZ transceivers. |

[Back to Top](#)

## simulation/functional

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

**Table 5-9: Functional Directory**

| Name   | Description   |
|--|---|
| <project_dir>/<component_name>/simulation/functional |   |
| simulate_mti.do                                      | ModelSim simulation script. This file is not generated for GTZ transceivers.  |
| wave_mti.do  | Script for adding transceiver wrapper signals to the ModelSim wave viewer. This file is not generated for GTZ transceivers. |
| 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.                             |



**Table 5-9: Functional Directory (Cont'd)**

| Name               | Description  |
|--------------------|--|
| ucli_command.key   | Command file for VCS simulator.  |
| vcs_session.tcl    | Script for adding transceiver wrapper signals to VCS wave window. This file is not generated for GTZ transceivers.             |
| wave_ncsim.sv      | Script for adding transceiver wrapper signals to the Cadence IES wave viewer. This file is not generated for GTZ transceivers. |
| 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. This file is not generated for GTZ transceivers.                               |
| simulate_isim.bat  | Windows script for running simulation using ISim. This file is not generated for GTZ transceivers.                             |
| wave_isim.tcl      | Script for adding transceiver wrapper signals to the ISim wave viewer. This file is not generated for GTZ transceivers.        |
| simulate_ncsim.bat | Windows script for running simulation using Cadence IES. This file is not generated for GTZ transceivers.                      |

[Back to Top](#)

## simulation/timing

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

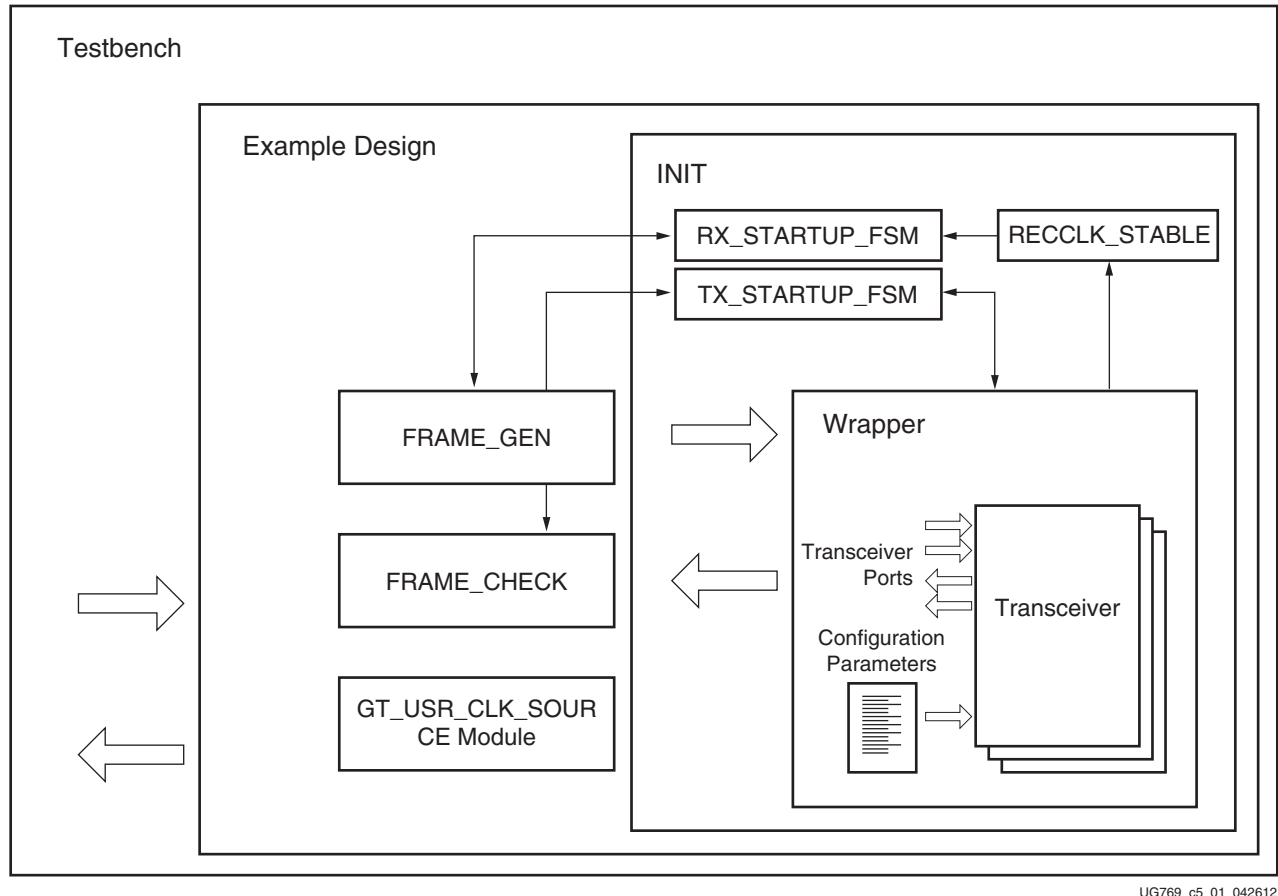
**Table 5-10: Timing Directory**

| Name            | Description   |
|-----------------|---|
| simulate_mti.do | ModelSim timing simulation script. This file is not generated for GTZ transceivers. |

[Back to Top](#)

## Example Design Description for GTX, GTH, and GTP Transceivers

The example design that is delivered with the wrappers helps designers understand how to use the wrappers and transceivers in a design. The example design is shown in Figure 5-1.



UG769\_c5\_01\_042612

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. In addition, the example design contains the initialization module, which consists of two independent finite state machines (tx\_startup\_fsm and rx\_startup\_fsm) and the recclk\_monitor block.

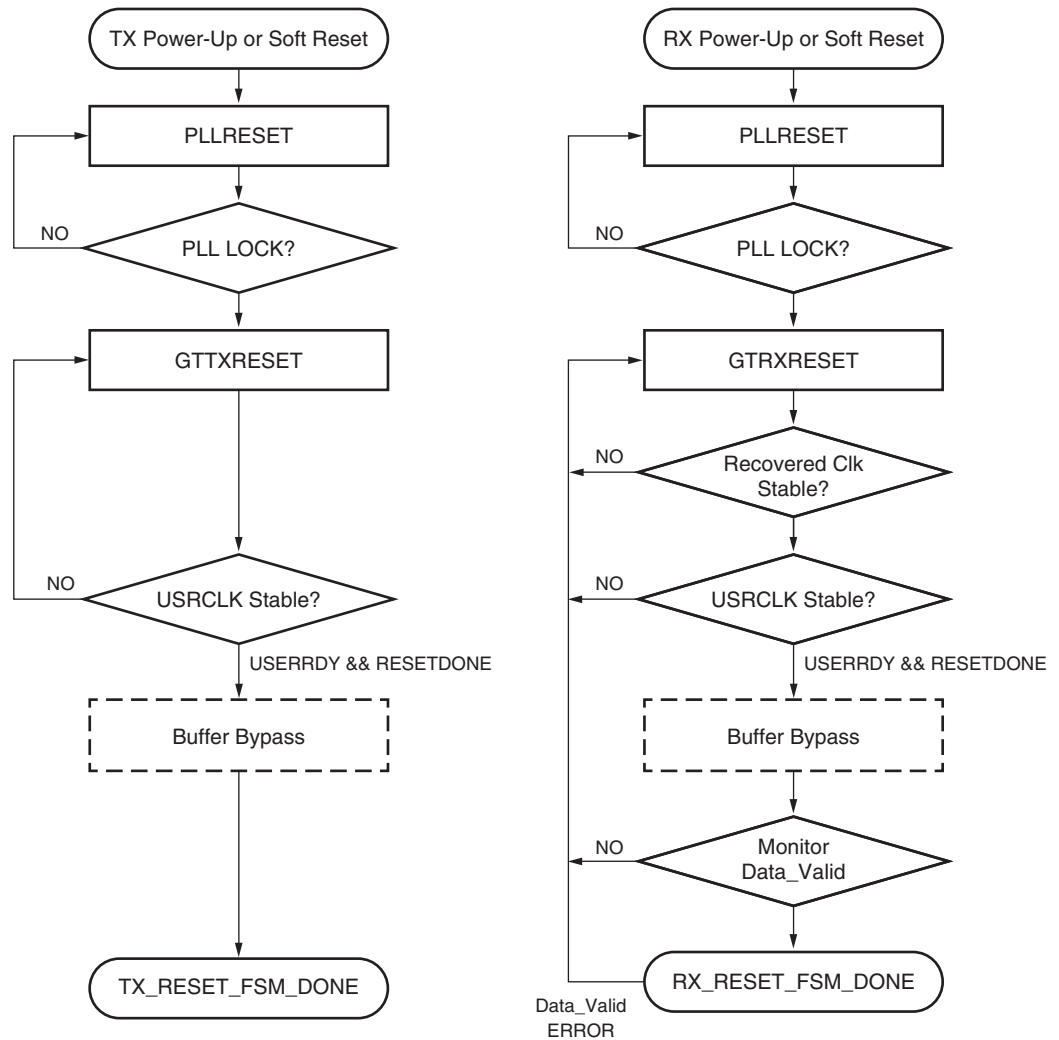
## Reset Finite State Machine

The intent for the reset finite state machine (FSM) included in the example design is to provide:

- An application example of an initialization FSM meeting the requirements described in [UG476](#), *7 Series FPGAs GTX/GTH Transceivers User Guide*.
- An example of an initialization FSM that addresses industry challenges to initialize the GT transceivers in scenarios such as post FPGA-configuration and RX data interruption or replacement (such as cable plug or unplug).

The reset FSM is constantly being improvised to provide a robust initialization and reset scheme. The FSM is to demonstrate the right methodology and should not be mistaken as a specification.

The tx\_startup\_fsm is illustrated on the left side of [Figure 5-2](#).



UG769\_c5\_02\_042312

Figure 5-2: Diagram of Simplified FSM

The C/QPLL lock is monitored along with TXUSRCLK stability prior to TXRESETDONE. Buffer bypass logic for phase alignment is implemented if the TX buffer is disabled.

The rx\_startup\_fsm is also illustrated on the right side of Figure 5-2. The C/QPLL lock, recovered clock stability, and RXUSRCLK are examined prior to RXRESETDONE followed by the buffer bypass logic for phase alignment. The FSM finally stays at the state that monitors data validity, which can be an 8B/10B error, frame sync error, or CRC from the user design until a user-defined error occurs.

These are some assumptions and notes for the Example Reset FSM:

- A stable REFCLK is assumed to be present at all times.
- All resets are assumed to be in Sequential mode. The reset FSMs run on SYSCLK, which is the same as the DRPCLK. If the SYSCLK and DRPCLK are not the same in the user design, care should be taken to add the appropriate synchronizers.
- The example design engaged additional gates available such as PLLREFCLKLOST and Wait-time. Wait-time in use should not be regarded as the specification.

- RECCLK\_STABLE is used as an indicator of RXOUTCLK (recovered clock) stability within a configured PPM offset from the reference clock (default 5000 ppm). The appropriate T<sub>DLOCK</sub> is used. (See [DS183](#), *Virtex-7 FPGAs Data Sheet: DC and Switching Characteristics*.)
- The example design defaults FRAME\_CHECKER as data\_valid. The user can identify their data valid indicator and provide the necessary hysteresis on this signal to avoid a false indication of data\_valid. Data valid is only used as an indicator to monitor the RX link. The user needs to customize the data valid indicator based on their system design.
- The user has the liberty to modify or re-invent an FSM to meet their specific system requirements while adhering to the guidelines in [UG476](#), *7 Series FPGAs GTX/GTH Transceivers User Guide*.
- There might be other restrictions in answer records or errata.

## CTLE3 Adaptation Modules for GTX Transceivers

### CTLE AGC Comp - CTLE3 Adaptation

This block compensates for the fact that the 7 series FPGA GTX transceiver does not have adaptive logic for tuning CTLE3. The purpose of the block is to adjust CTLE3 gain to keep AGC from railing. The block uses channel DRP to set CTLE3 values and the RXMONITOR ports to observe AGC. This block is activated whenever the user asserts GTRXRESET, RXPMARESET, or RXDFELPMRESET (actual operation begins at the deassertion of any of these reset signals).

**Note:** The user does not have control of the DRP interface as long as the ADAPT\_DONE signal coming out of these blocks is not High.

## Example Design Hierarchy

The hierarchy for the design used in this example is:

```

EXAMPLE_TB
|__XAU_WRAPPER_EXDES
|__XAU_WRAPPER_INIT
|__TX_STARTUP_FSM           (1 per transceiver)
|__RX_STARTUP_FSM         (1 per transceiver)
|__RECCLK_MONITOR
|__XAU_WRAPPER
|__XAU_WRAPPER_GT         (1 per transceiver)
|__XAU_WRAPPER_GT_FRAME_GEN (1 per transceiver)
|__XAU_WRAPPER_GT_FRAME_CHECK
|__XAU_WRAPPER_GT_USRCLK_SOURCE
|__XAU_WRAPPER_CLOCK_MODULE
    
```

## Reset Sequence Modules for GTH and GTP Transceivers

GTH and GTP transceivers need additional reset sequencing for production silicon. The reset sequencers for GTRXRESET, RXPMARESET, and RXRATE are instantiated in the `<component_name>_gt.v[hd]` module. These modules need exclusive access to the DRP interface of the transceiver. The `DRP_BUSY_OUT` signal that is synchronous to the `DRPCLK` indicates that the user cannot access the DRP interface. The user should implement a DRP arbitration scheme to ensure smooth operation of the various reset sequencers. In the current design, the GTRXRESET sequencer takes highest priority.

## Example Design Description for GTZ Transceivers

The example design that is delivered with the wrappers helps designers understand how to use the wrappers and transceivers in a design.

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.

The example design also demonstrates how to generate the USRCLKs out of the OUTCLKs coming out of an octal. Also note the connections of USRCLKs for octal0 and octal1. 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. The `frame_gen_top` module also shows how to drive the txsequence counter and how to pause the data and header controls based on the sequence counter values. The USRCLK source module shows how and when an MMCM is required to generate a USRCLK.

The example design can be synthesized using XST or the Vivado™ tools and implemented with the Vivado tools. It can be simulated using VCS or Cadence's Incisive Unified Simulator (IUS).

## CTLE Tuning

The example design also demonstrates how to enable the CTLE tuning for a given channel in an octal. It is essential to enable CTLE tuning to receive clean data on the serial interface of the GTZ transceiver in internal PMA loopback. The `ctle_tuning.v` module demonstrates an implementation of the procedure documented in UG478, *7 Series FPGAs GTZ Transceivers User Guide*. This module is instantiated once for each octal instantiation because it uses the DRP interface associated with an octal to make enables and monitor the CTLE tuning.

The interface for this module is shown in [Table 5-11](#).

**Table 5-11: CTLE Tuning Module Ports**

| Port                            | Direction | Description  |
|---------------------------------|-----------|--|
| nearendloopbacken_i[7:0]        | Input     | 8 bit vector - 1 bit per channel:<br>1: Loopback enabled<br>0: Loopback disabled<br>It is recommended to tie this port to the NEARLOOPBACKEN<0-7> ports of the GTZE2_OCTAL primitive.  |
| chan_id_i[3:0]                  | Input     | Channel ID. Selects the channel number for which CTLE tuning needs to be done. Values 0–7 correspond to channels 0–7. A value of 8 corresponds to all channels together. The module enables coarse and fine tuning for all eight channels when a channel ID of 8 is selected. All other values are illegal. The value of this port is disregarded when the CTLE tuning is reset using ctle_tuning_reset input. |
| redo_channel_tuning_i           | Input     | When enabled, the tuning enable FSM re-enables the CTLE tuning (thereby re-running the firmware) for the channel selected using chan_id_i.   |
| channel_ctle_tuning_done_o[7:0] | Output    | Sticky output for each channel:<br>1: Indicates tuning completed successfully without errors.<br>0: Indicates tuning not completed or completed with errors.   |
| ctle_tuning_state_o[8:0]        | Output    | The current state of the tuning enable FSM is output on this port, to be used for debug purposes.  |
| read_chan_id_o[3:0]             | Output    | Tells the current channel for which the tuning complete status is being read to be used for debug purposes.  |
| octal_ctle_tuning_reset_i       | Input     | Reset input that resets the entire state machine, forcing the tuning to be re-run for all channels by default for non-loopback mode.   |
| drpclk_i                        | Input     | DRPCLK used to run the entire state machine. It is recommended to connect this with either DRPCLK0 or DRPCLK1 of the octal based on the value of the DRPCLKSEL attribute.  |
| drpdo_i[31:0]                   | Input     | DRP interface port to be connected to the corresponding signal on the octal.   |
| drpdi_o[31:0]                   | Output    | DRP interface port to be connected to the corresponding signal on the octal.   |

Table 5-11: CTLE Tuning Module Ports (Cont'd)

| Port            | Direction | Description  |
|-----------------|-----------|--|
| drpaddr_o[15:0] | Output    | DRP interface port to be connected to the corresponding signal on the octal. |
| drprdy_i        | Input     | DRP interface port to be connected to the corresponding signal on the octal. |
| drpen_o         | Output    | DRP interface port to be connected to the corresponding signal on the octal. |
| drpwe_o         | Output    | DRP interface port to be connected to the corresponding signal on the octal. |

The CTLE tuning module is brought out of reset as soon as the RXRDY signal of all the active channels of an octal are asserted. Out of reset, tuning is enabled for both coarse and fine irrespective of whether or not the internal PMA loopback of a particular channel is enabled. After the CTLE tuning is completed, RX FIB reset should be pulsed so that the RX FIFO is reset after the CTLE tuning is completed. The `ctle_tuning_done` signal is therefore connected to the RXFIBRESET in the example design.

To tune individual channels for internal loopback only, the user should wait until the `ctle_tuning_done` for all active channels is asserted, and then follow this sequence:

1. Drive the `chan_id` value with the channel number for which tuning has to be re-run.
2. Assert the `nearendloopbacken_i` port.
3. Pulse the `redo_channel_tuning` port for at least one DRPCLK cycle.

Care must be taken not to toggle the loopback port for that channel until the CTLE tuning is completed. The FSM stops whenever it reaches the CTLE\_TUNING\_DONE state. For the FSM to re-run, the `redo_channel_tuning` port needs to be pulsed for at least one DRPCLK cycle. Thus, to re-run the CTLE tuning firmware for a particular channel, the user should wait for the FSM to reach the CTLE\_TUNING\_DONE state. This is evident by monitoring the `ctle_tuning_done` output for all active channels. To re-run the tuning, the user should follow these steps after the FSM reaches the CTLE\_TUNING\_DONE state:

1. Enable or disable near-end PMA loopback of the required channel using the `nearendloopbacken_i` port.
2. Drive the required channel number on the `chan_id` port.
3. Pulse the `redo_channel_tuning` for at least one DRPCLK cycle.
4. Wait for the `ctle_tuning_done` output of the particular channel to assert. This signifies that the FSM has again reached the TUNING\_DONE state, ready to be re-triggered for any other channels based on the value of the `redo_channel_tuning` port.



Figure 5-3 shows the CTLE tuning enable state machine.

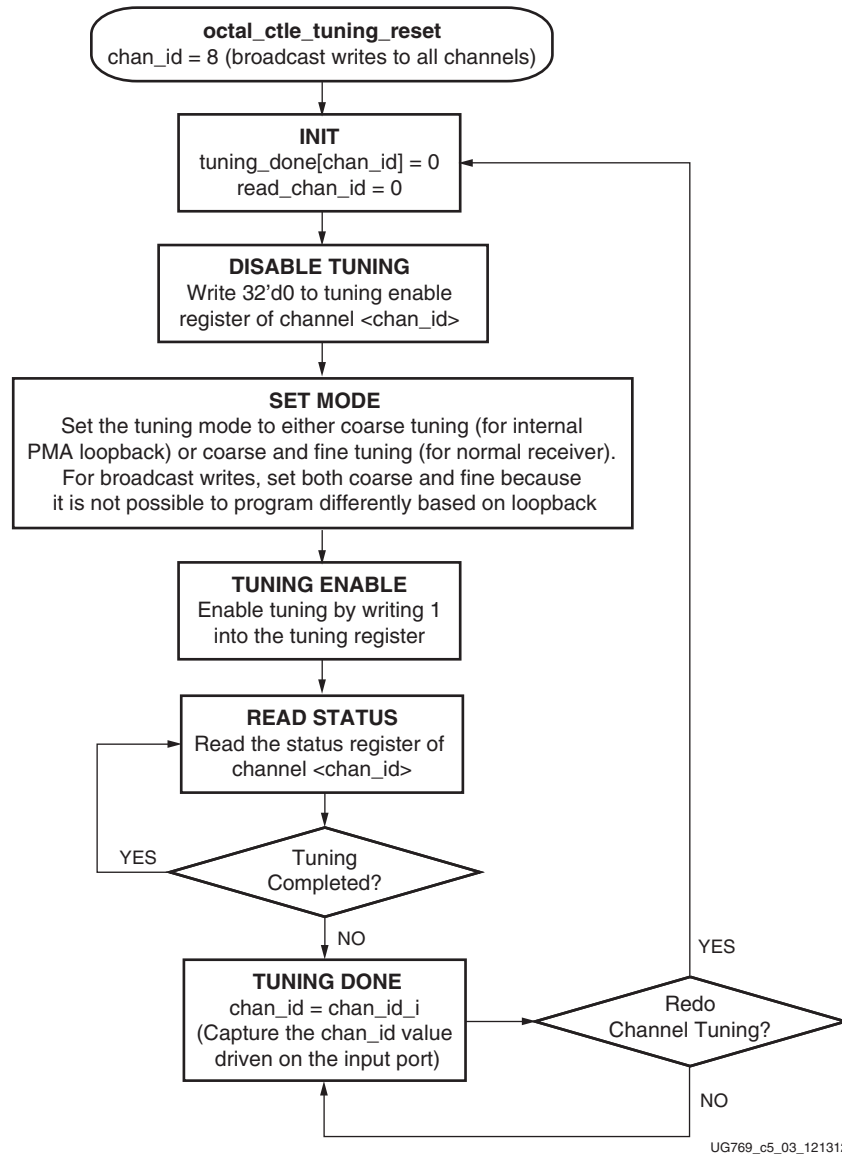


Figure 5-3: CTLE Tuning Enable State Machine

## Beachfront Module

The beachfront module is mandatory to be used along with every instantiation of the GTZE2\_OCTAL primitive. This module guarantees the timing of all the synchronous signals across the GTZ transceiver and FPGA logic interface by routing them through pre-locked flip-flops and LUTs. Each input and output signal of the GTZE2\_OCTAL passes through this module. Each of the synchronous signals are passed through flip-flops and LUTs that are pre-LOCed.

In addition, the beachfront also has the capability to instantiate the BUFG\_LB primitives to generate USERCLKs from OUTCLKs. This functionality is controlled by the parameters listed in Table 5-12. Apart from these parameters, all other parameters seen on the

beachfront are the same as the ones that are used on the GTZE2\_OCTAL primitive. For more information, refer to UG478, *7 Series FPGAs GTZ Transceivers User Guide*.

**Table 5-12: TXUSRCLK Attributes on Beachfront**

| Parameter         | Description   |
|-------------------|---|
| TXUSRCLK_SEL_MUX0 | <p>TXUSRCLK0: Selects the incoming (from FPGA logic) TXUSRCLK0 port to pass on to the GTZE2_OCTAL primitive and also use the same to clock the ports in beachfront. Users should take the TXOUTCLK0 output from the beachfront and use an MMCM or a BUFG_LB to generate the user clock.</p> <p>TXCORECLK0: Instantiates a BUFG_LB primitive whose input is TXOUTCLK0 and output is TXCORECLK0. The output is passed on to the GTZE2_OCTAL parameter as the TXUSRCLK0, and the same is also used internally to clock beachfront flip-flops. Users should use TXCORECLK0 as TXUSRCLK0 in FPGA logic in this case.</p>     |
| TXUSRCLK_SEL_MUX1 | <p>TXUSRCLK1: Selects the incoming (from FPGA logic) TXUSRCLK1 port to pass on to the GTZE2_OCTAL primitive and also use the same to clock the ports in beachfront. Users should take the TXOUTCLK1 output from the beachfront and use an MMCM or a BUFG_LB to generate the user clock.</p> <p>TXCORECLK1: Instantiates a BUFG_LB primitive whose input is TXOUTCLK1 and output is TXCORECLK1. The output is passed on to the GTZE2_OCTAL parameter as the TXUSRCLK1, and the same is also used internally to clock beachfront flip-flops. Users should use the TXCORECLK1 as TXUSRCLK1 in FPGA logic in this case.</p> |

Table 5-13 shows the RXUSRCLK attributes on the beachfront module.

**Table 5-13: RXUSRCLK Attributes on Beachfront**

| Parameter         | Description   |
|-------------------|---|
| RXUSRCLK_SEL_MUX0 | <p>RXUSRCLK0: Selects the incoming (from fabric logic) RXUSRCLK0 port to pass on to the GTZE2_OCTAL primitive and also use the same to clock the ports in beachfront. Users should take the RXOUTCLK0 output from the beachfront and use an MMCM or a BUFG_LB to generate the user clock.</p> <p>RXCORECLK0: Instantiates a BUFG_LB primitive whose input is RXOUTCLK0 and output is RXCORECLK0. The output is passed on to the GTZE2_OCTAL parameter as the RXUSRCLK0, and the same is also used internally to clock beachfront flip-flops. Users should use the RXCORECLK0 as RXUSRCLK0 in FPGA logic in this case.</p> |

Table 5-13: RXUSRCLK Attributes on Beachfront (Cont'd)

| Parameter         | Description  |
|-------------------|--|
| RXUSRCLK_SEL_MUX1 | <p>RXUSRCLK1: Selects the incoming (from fabric logic) RXUSRCLK1 port to pass on to the GTZE2_OCTAL primitive and also use the same to clock the ports in beachfront. Users should take the RXOUTCLK1 output from the beachfront and use an MMCM or a BUFG_LB to generate the user clock.</p> <p>RXCORECLK1: Instantiates a BUFG_LB primitive whose input is RXOUTCLK1 and output is RXCORECLK1. The output is passed on to the GTZE2_OCTAL parameter as the RXUSRCLK1, and the same is also used internally to clock beachfront flip-flops. Users should use the RXCORECLK1 as RXUSRCLK1 in FPGA logic in this case.</p>        |
| RXUSRCLK_SEL_MUX2 | <p>RXUSRCLK2: Selects the incoming (from fabric logic) RXUSRCLK2 port to pass on to the GTZE2_OCTAL primitive and also use the same to clock the ports in beachfront. Users should take the RXOUTCLK2 output from the beachfront and use an MMCM or a BUFG_LB to generate the user clock.</p> <p>RXCORECLK2: Instantiates a BUFG_LB primitive whose input is RXOUTCLK1 and output is RXCORECLK2. The output is passed on to the GTZE2_OCTAL parameter as the RXUSRCLK2, and the same is also used internally to clock beachfront flip-flops. Users should use the RXCORECLK2 as RXUSRCLK2 in FPGA logic in this case.</p>        |
| RXUSRCLK_SEL_MUX3 | <p>RXUSRCLK3: Selects the incoming (from FPGA logic) RXUSRCLK3 port to pass on to the GTZE2_OCTAL primitive and also use the same to clock the ports in beachfront. Users should take the RXOUTCLK3 output from the beachfront and use an MMCM or a BUFG_LB to generate the user clock.</p> <p>RXCORECLK3: Instantiates a BUFG_LB primitive whose input is RXOUTCLK3 and output is RXCORECLK3. The output is passed on to the GTZE2_OCTAL parameter as the RXUSRCLK3, and the same is also used internally to clock beachfront flip-flops. Users are supposed to use the RXCORECLK3 as RXUSRCLK3 in FPGA logic in this case.</p> |

Input ports to the beachfront from the FPGA logic are prefixed with B2M\_. Output ports from the beachfront going to the FPGA logic are prefixed with M2B\_. Thus, the user should always use either the M2B\_ or B2M\_ ports of the beachfront to connect to the GTZE2\_OCTAL. Apart from the ports listed in Table 5-14, all other ports on the beachfront interface are the same as the ones seen in the GTZE2\_OCTAL primitive. For more information, refer to UG478, *7 Series FPGAs GTZ Transceivers User Guide*.

Table 5-14: TX/RXCORECLK Ports on Beachfront

| Port       | Direction | Description  |
|------------|-----------|--|
| TXCORECLK0 | Output    | This port should be used as TXUSRCLK0 if the parameter TXUSRCLK_SEL_MUX0 is set to TXCORECLK0, else this port should be ignored. |
| TXCORECLK1 | Output    | This port should be used as TXUSRCLK1 if the parameter TXUSRCLK_SEL_MUX1 is set to TXCORECLK1, else this port should be ignored. |
| TXCORECLK0 | Output    | This port should be used as TXUSRCLK1 if the parameter TXUSRCLK_SEL_MUX1 is set to TXCORECLK1, else this port should be ignored. |
| RXCORECLK0 | Output    | This port should be used as RXUSRCLK0 if the parameter RXUSRCLK_SEL_MUX0 is set to RXCORECLK0, else this port should be ignored. |
| RXCORECLK1 | Output    | This port should be used as RXUSRCLK1 if the parameter RXUSRCLK_SEL_MUX1 is set to RXCORECLK1, else this port should be ignored. |
| RXCORECLK2 | Output    | This port should be used as RXUSRCLK2 if the parameter RXUSRCLK_SEL_MUX2 is set to RXCORECLK2, else this port should be ignored. |
| RXCORECLK3 | Output    | This port should be used as RXUSRCLK3 if the parameter RXUSRCLK_SEL_MUX3 is set to RXCORECLK3, else this port should be ignored. |

## Dynamic Phase Deskew

The example design generated by the CAUI4 protocol template of the wizard also includes a dynamic phase deskew logic within it to ensure the lane-to-lane skew is minimized. This logic is found in the `gt_usrclk_source.v` module generated in the `example_design` folder. This logic, with the help of the MMCM, adjusts the phase of the RXFIFO read clock to get the maximum margin on the right side without losing too much on the left margin.

- The user has to set the M, D, and O values of the MMCM to get the required VCO frequency and resolution. The resolution is  $1/56 F_{VCO}$ .
- After the resolution is finalized, the user should calculate SHIFT\_COUNT, which can be obtained by dividing the required phase shift with the resolution. If SHIFT\_COUNT is a fraction, it should be rounded up to the nearest decimal.
- The MMCM uses the dynamic phase variation feature. An FSM controls the phase shift of USRCLK using the phase shift interface.
- Shifting starts whenever RXRESETDONE is asserted from all channels. The Low-to-High transition of the RXRESETDONE signal triggers the adjustment again.
- The user should assert the RXFIBRESET once after the MMCM is locked. This ensures that the RX FIFO is reset after the phase adjustments are done on the read clock.
- Each channels have different left and right margins, but it is guaranteed to be above  $(3.1 + 1.5)$  when operated at 25.8 Gb/s.
- Whenever the link goes down and is later re-established using a new training sequence, it is possible to get a different write clock phase. This training sequence is always followed by an RXFIFO reset to bring the FIFO to a normal operating

condition. In that case, RXRESETDONE to the deskew logic toggles and reinitiates the phase adjustment.

## Multi-Lane Mode

The Wizard enables multi-lane mode and puts the GT transceiver in master-slave mode by default whenever the CAUI4 protocol template is selected. This is essential to reduce the lane-to-lane deskew. For more information on how to use the multi-lane mode, refer to UG478, *7 Series FPGAs GTZ Transceivers User Guide*.

## Example Design Hierarchy

The hierarchy for the design used in this example is:

```
EXAMPLE_TB
|__CAUI4_WRAPPER_EXDES
|  |__CAUI4_INIT
|  |  |__CAUI4_WRAPPER
|  |    |__CAUI4_WRAPPER_OCTAL0 (1 per octal)
|  |    |__GTWIZARD_V2_2_BEACHFRONT (1 per octal)
|  |  |__CAUI4_WRAPPER_CTLE_TUNING (1 per octal)
|  |  |__CAUI4_WRAPPER_RX_STARTUP_FSM (1 per channel)
|  |__CAUI4_WRAPPER_GT_FRAME_GEN_TOP (1 per channel)
|  |  |__CAUI4_WRAPPER_GT_FRAME_GEN
|  |  |__CAUI4_WRAPPER_SCRAMBLER (5 per channel)
|  |__CAUI4_WRAPPER_GT_FRAME_CHECK_TOP (1 per channel)
|  |  |__CAUI4_WRAPPER_GT_FRAME_CHECK (5 per channel)
|  |  |__CAUI4_WRAPPER_DESCRAMBLER (5 per channel)
|  |  |__CAUI4_WRAPPER_BLOCK_SYNC_SM (5 per channel)
|  |__CAUI4_WRAPPER_GT_USRCLK_SOURCE (contains MMCM and dynamic phase deskew)
```

## Known Limitations of the GTZ Wizard

1. Enabling CTLE tuning is mandatory for any GTZ design. Thus, the user is required to follow the steps involved to enable CTLE tuning or re-use the code in the example design `<component_name>_ctle_tuning.v` module for the same.
2. Dynamic phase deskew is necessary to reduce the lane-to-lane deskew for multi-lane protocols like CAUI4 operating at greater than 25G per line. The user is required to reuse the code as shown in the `<component_name>_gt_usrclk_source.v` module to achieve the same.
3. Even though RXFIBRESET is listed as an optional port, this is always brought out to the top level of the example design to enable users to control the RX FIFO reset for operations like CTLE tuning and dynamic phase deskew logic.
4. The GTZ Wizard shows every possible REFCLK for a selected line rate in the REFCLK drop-down menu, but only a few are supported by the GTZ transceiver. Refer to UG478, *7 Series FPGAs GTZ Transceivers User Guide* for more information on the supported REFCLK values for a given line rate.

