LogiCORE™ IP Virtex®-6 FPGA GTX Transceiver Wizard v1.7

Getting Started Guide

UG516 (v1.7) September 21, 2010

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/24/09	1.2	Initial Xilinx release.
09/16/09	1.3	Tools and Wizard updates. Added support for the Virtex-6 HXT family. Added Chapter 4, "Quick Start Example Design" and renumbered Chapter 5, "Detailed Example Design."
12/02/09	1.4	Tools and Wizard updates. Added support for Virtex-6 Lower Power family. Added "Recommended Design Experience" in Chapter 1. Replaced Appendix A: References with "Related Xilinx Documents" in Chapter 1. Updated Table 3-18.
04/19/10	1.5	Tools and Wizard updates. Added CXT support. Added REFCLK0/01 Q9 in Table 3-6, updated TXUSRCLK and RXUSRCLK in Table 3-7, added scripts to Table 5-8.
07/23/10	1.6	Tools and Wizard updates.
09/21/10	1.7	Wizard v1.7 release.

Table of Contents

Chapter 4: Quick Start Example Design

Chapter 5: Detailed Example Design

Preface

About This Guide

The *LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard v1.7 Getting Started Guide* describes the function and operation of the LogiCORE™ IP GTX Transceiver Wizard for the Virtex®-6 CXT, LXT, SXT, HXT, and lower-power families.

Guide Contents

This guide contains the following chapters:

- Preface, "About this Guide" introduces the organization and purpose of this guide, a list of additional resources, and the conventions used in this document.
- Chapter 1, "Introduction" describes the wrapper core and related information, including additional resources, technical support, and submitting feedback to Xilinx.
- Chapter 2, "Installing and Licensing" provides information about installing and licensing the Virtex-6 FPGA GTX Transceiver Wizard.
- Chapter 3, "Running the Wizard" provides an overview of the Virtex-6 FPGA GTX Transceiver Wizard, and a step-by-step tutorial to generate a sample GTX transceiver wrapper with the Xilinx® CORE Generator™ tool.
- Chapter 4, "Quick Start Example Design" introduces the example design that is included with the GTX transceiver wrappers. The example design demonstrates how to use the wrappers and demonstrates some of the key features of the GTX transceiver.
- Chapter 5, "Detailed Example Design" provides detailed information about the example design, including a description of files and the directory structure generated by the Xilinx CORE Generator tool, the purpose and contents of the provided scripts, the contents of the example HDL wrappers, and the operation of the demonstration test bench.

Additional Resources

To find additional documentation, see the Xilinx website at:

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

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

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

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

This document uses the following typographical conventions. An example illustrates each convention.

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages, prompts, and program files that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *design_name*
Helvetica bold	Commands that you select from a menu	**File** → **Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in a syntax statement for which you must supply values	**`ngdbuild`** *design_name*
	References to other manuals	See the *User Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Dark Shading	Items that are not supported or reserved	This feature is not supported
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required.	**`ngdbuild`** `[`*`option_name`*`]` *`design_name`*
Braces { }	A list of items from which you must choose one or more	**`lowpwr ={on|off}`**
Vertical bar |	Separates items in a list of choices	**`lowpwr ={on|off}`**
Angle brackets < >	User-defined variable or in code samples	<directory name>

Vertical ellipsis
.
.
.
	Repetitive material that has been omitted	`IOB #1: Name = QOUT'`
		`IOB #2: Name = CLKIN'`
		.
		.
		.

Horizontal ellipsis . . .	Repetitive material that has been omitted	**`allow block`** *block_name loc1 loc2 ... locn;*

Convention	Meaning or Use	Example
Notations	The prefix '0x' or the suffix 'h' indicate hexadecimal notation	A read of address 0x00112975 returned 45524943h.
	An '_n' means the signal is active low	`usr_teof_n` is active low.

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Additional Resources" for details. Refer to "Title Formats" in Chapter 1 for details.
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Chapter 1

Introduction

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

The Virtex-6 FPGA GTX Transceiver Wizard is a Xilinx® CORE Generator™ tool designed to support both Verilog and VHDL design environments. In addition, the example design delivered with the core is provided in Verilog or VHDL.

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

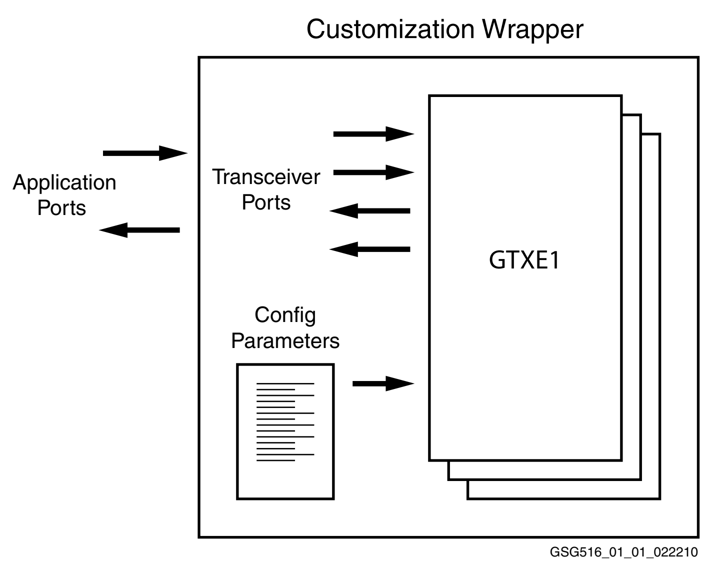

Figure 1-1: **GTX Transceiver Wizard Wrapper**

About the Wizard

The Virtex-6 FPGA GTX Transceiver Wizard is a Xilinx CORE Generator tool, available at the Xilinx IP Center. For information about system requirements, installation, and licensing options, see Chapter 2, "Installing and Licensing."

Recommended Design Experience

Although the Virtex-6 FPGA GTX Transceiver Wizard core is a fully verified solution, the challenge associated with implementing a complete design varies depending on the configuration and 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.

Contact your local Xilinx representative for a closer review and estimation for your specific requirements.

Related Xilinx Documents

Prior to generating the Virtex-6 FPGA GTX Transceiver Wizard, users should be familiar with the following:

- DS150: *Virtex-6 Family Overview*
- UG366: *Virtex-6 FPGA GTX Transceivers User Guide*
- ISE software documentation: www.xilinx.com/ise

Additional Wizard Resources

For detailed information and updates about the Virtex-6 FPGA GTX Transceiver Wizard, see the following documents located at the Architecture Wizards page:

- DS708: *LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard v1.7 Data Sheet*
- UG516: *LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard v1.7 Getting Started Guide*
- Virtex-6 FPGA GTX Transceiver Wizard Release Notes

Technical Support

For technical support, go to www.xilinx.com/support. Questions are routed to a team of engineers with expertise using the Virtex-6 GTX Wizard.

Xilinx provides technical support for use of this product as described in the *LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard v1.7 Getting Started Guide*. Xilinx cannot guarantee timing, functionality, or support of this product for designs that do not follow these guidelines.

Feedback

Xilinx welcomes comments and suggestions about the Virtex-6 FPGA GTX Transceiver Wizard and the accompanying documentation.

GTX Transceiver Wizard

For comments or suggestions about the Virtex-6 FPGA GTX Transceiver Wizard, please submit a WebCase from www.xilinx.com/support. (Registration is required to log in to WebCase.) Be sure to include the following information:

- Product name
- Wizard version number

- List of parameter settings
- Explanation of your comments, including whether the case is requesting an *enhancement* (you believe something could be improved) or reporting a *defect* (you believe something isn't working correctly).

Document

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

- Document title
- Document number
- Page number(s) to which your comments refer
- Explanation of your comments, including whether the case is requesting an *enhancement* (you believe something could be improved) or reporting a *defect* (you believe something isn't documented correctly).

Installing and Licensing

This chapter provides instructions for installing the Virtex®-6 FPGA GTX Transceiver Wizard in the Xilinx® CORE Generator™ tool.

Note: It is not necessary to obtain a license to use the Wizard.

Supported Tools and System Requirements

Operating Systems

Windows

- Windows XP Professional 32-bit/64-bit
- Windows Vista Business 32-bit/64-bit

Linux

- Red Hat Enterprise Linux WS v4.0 32-bit/64-bit
- Red Hat Enterprise Desktop v5.0 32-bit/64-bit (with Workstation Option)
- SUSE Linux Enterprise (SLE) v10.1 32-bit/64-bit

Tools

- ISE® 12.3 software
- Mentor Graphics ModelSim 6.5c
- Cadence Incisive Enterprise Simulator (IES) 9.2
- Synopsys VCS and VCS MX 2009.12

Check the release notes for the required Service Pack; ISE Service Packs can be downloaded from www.xilinx.com/support/download.htm.

Before You Begin

Before installing the Wizard, you must have a MySupport account and the ISE 12.3 software installed on your system. If you already have an account and have the software installed, go to "Installing the Wizard", otherwise do the following:

1. Click **Login** at the top of the Xilinx home page then follow the onscreen instructions to create a MySupport account.
2. Install the ISE 12.3 software.

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

Installing the Wizard

The Virtex-6 FPGA GTX Transceiver Wizard is included with the ISE 12.3 software. See *ISE CORE Generator IP Updates - Installation Instructions* for details about installing ISE 12.3.

Verifying Your Installation

Use the following procedure to verify that you have successfully installed the Virtex-6 FPGA GTX Transceiver 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, as shown in Figure 2-1.

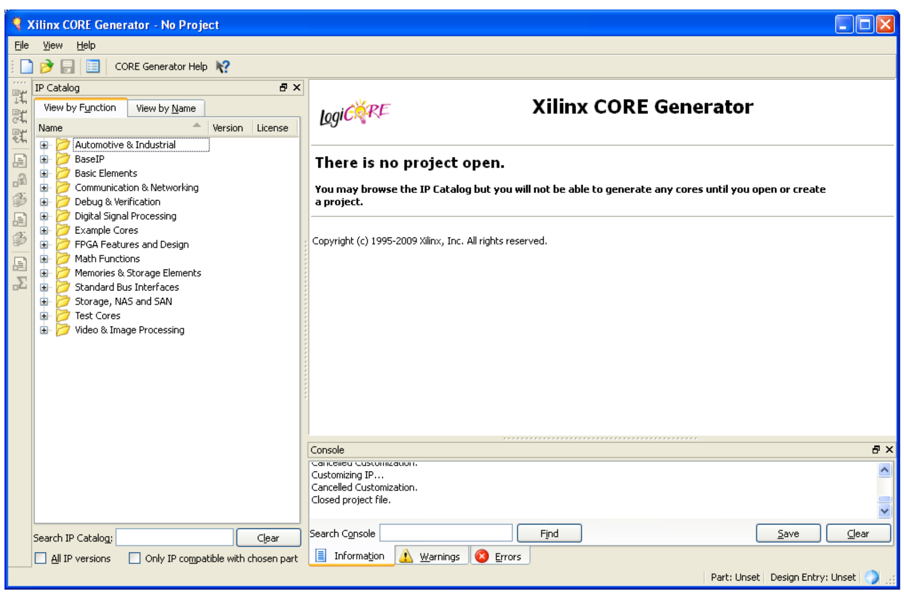

Figure 2-1: **CORE Generator Window**

3. Click to expand or collapse the view of individual functional categories, or click the **View by Name** tab at the top of the list to see an alphabetical list of all cores in all categories.
4. Determine if the installation was successful by verifying that Virtex-6 FPGA GTX Transceiver Wizard 1.7 appears at the following location in the Functional Categories list: `/FPGA Features and Design/IO Interfaces`

Running the Wizard

Overview

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

The example design covered in this section is a wrapper that configures a group of GTX transceivers for use in a XAUI application. Guidelines are also given for incorporating the wrapper in a design and for the expected behavior in operation.

The XAUI example consists of the following components:

- A single GTX transceiver wrapper implementing a four-lane XAUI port using four GTX transceivers
- A demonstration test bench 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 Virtex-6 FPGA GTX Transceiver Wizard example design has been tested with Synplify Pro D-2009.12 and XST 12.3 for synthesis and ModelSim 6.5c and above for simulation.

Figure 3-1 shows a block diagram of the default XAUI example design.

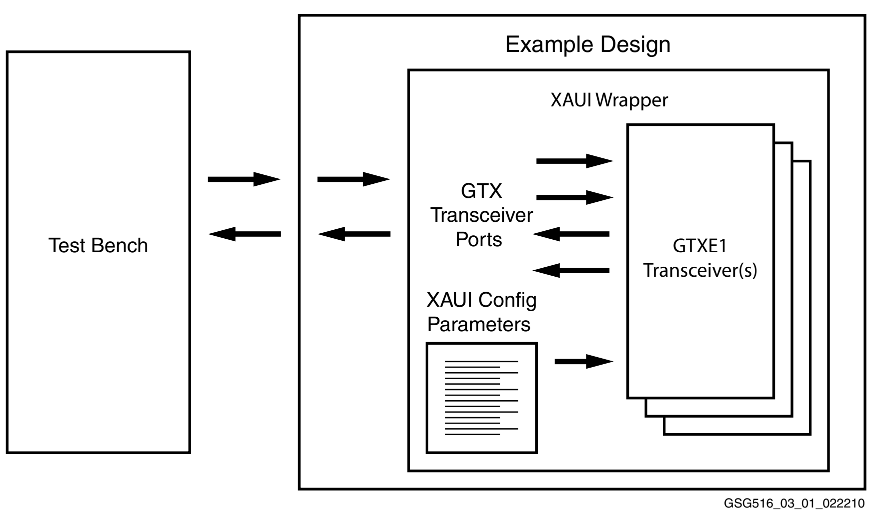

Figure 3-1: **Example Design**

Setting Up the Project

Before generating the example design, set up the project as described in "Creating a Directory" and "Setting the Project Options" of this guide.

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/xaui_example`

2. Start the Xilinx CORE Generator ™ software.

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

3. Choose **File > New Project** (Figure 3-2).
4. Change the name of the `.cgp` file (optional).
5. Click **Save**.

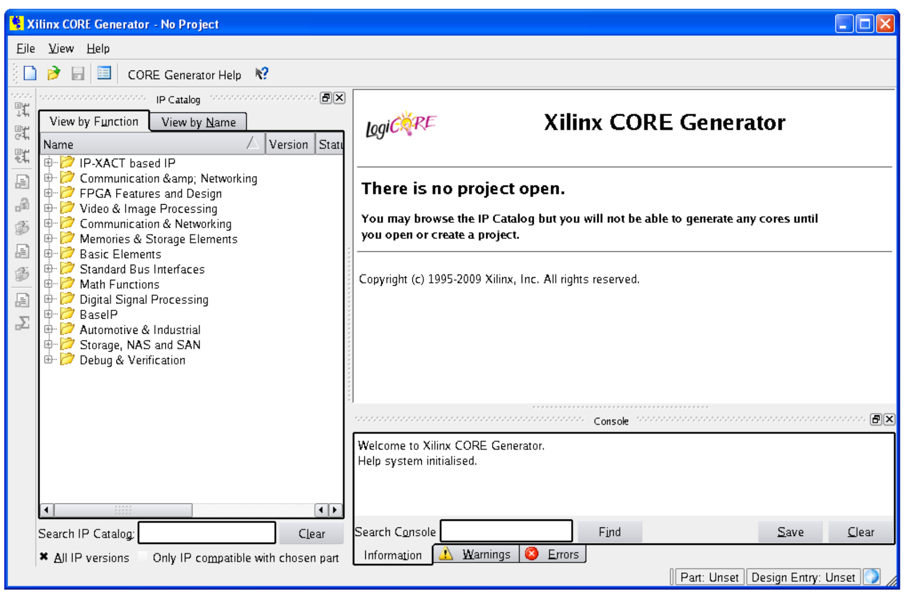

Figure 3-2: **Starting a New Project**

Setting the Project Options

Set the project options using the following steps:

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

   Note: If an unsupported silicon family is selected, the Virtex-6 FPGA GTX Transceiver Wizard remains light grey in the taxonomy tree and cannot be customized. Only devices containing Virtex-6 GTX transceivers are supported by the Wizard. See the *Virtex-6 Family Overview* for a list of devices containing GTX transceivers.
5. Click **Generation** in the option tree and select either Verilog or VHDL as the output language.
6. Click **OK**.

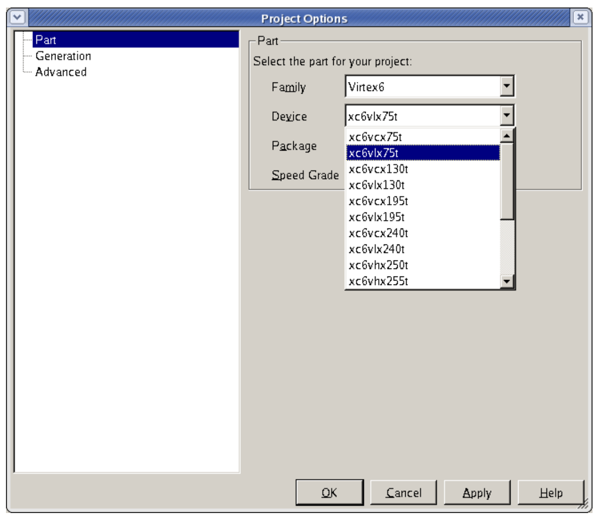

Figure 3-3: **Target Architecture Setting**

Generating the Core

This section provides instructions for generating an example GTX transceiver wrapper core using the default values. The core and its supporting files, including the example design, are generated in the project directory. For additional details about the example design files and directories see Chapter 5, "Detailed Example Design."

1. Locate Virtex-6 FPGA GTX Transceiver Wizard 1.7 in the taxonomy tree under: `/FPGA Features & Design/IO Interfaces`. (See Figure 3-4)
2. Double-click **Virtex-6 FPGA GTX Transceiver Wizard 1.7** to launch the Wizard.

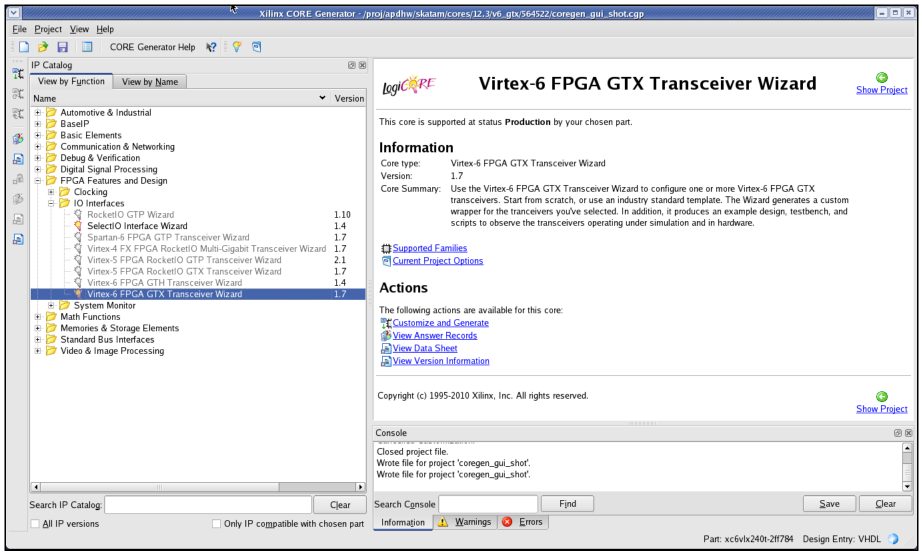

Figure 3-4: **Locating the GTX Transceiver Wizard**

Line Rates and Encoding

The Line Rates and Encoding screen (Page 1) of the Wizard (Figure 3-5) allows you to select the component name and determine the line rate, reference clock frequency, encoding/decoding method, and data width. In addition, this page specifies a protocol template.

1. In the Component Name field, enter a name for the core instance. This example uses the name **xaui_wrapper**.
2. From the Protocol Template list, select **Start from scratch** if you wish to manually set all parameters.

   Select one of the available protocols from the list to begin your design with a pre-defined protocol template. The XAUI example uses the XAUI protocol template.

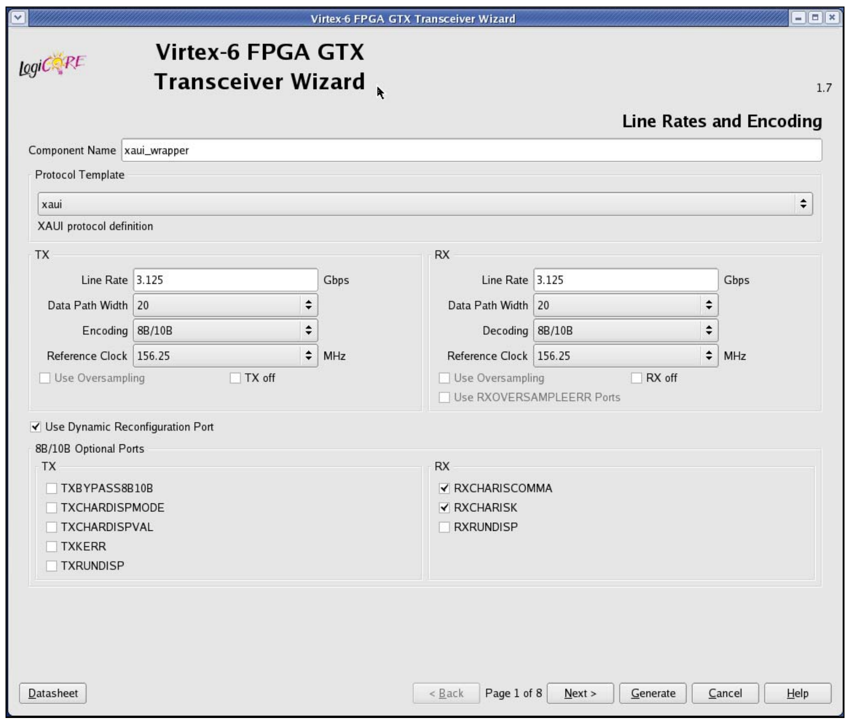

Figure 3-5: **Line Rates and Encoding - Page 1**

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

   Note: In the following tables, options not used by the XAUI example are shaded.

Table 3-1: **TX Settings**

Options		Description
Line Rate		Set to the desired target line rate in Gbps. Can be independent of the receive line rate. The XAUI example uses 3.125 Gbps.
Data Path Width	8	Sets the internal transmitter data path width to two 8-bit bytes. Sets the transmitter application interface data path width to a single 8-bit byte.
	10	Sets the internal transmitter data path width to two 10-bit bytes (20-bits). If 8B/10B encoding is selected, the transmitter application interface data path width will be set to 8-bits. If 8B/10B encoding is not selected, the transmitter application interface data path width will be set to 10-bits.
	16	Sets both the internal transmitter data path and the transmitter application interface data path width to two 8-bit bytes (16 bits).
	20	Sets the internal transmitter data path width to two 10-bit bytes (20 bits). If 8B/10B encoding is selected, the transmitter application interface data path width will be set to two 8-bit bytes (16 bits). If 8B/10B encoding is not selected, the transmitter application interface data path width will also be set to two 10-bit bytes (20 bits).
	32	Sets the internal transmitter data path width to two 8-bit bytes (16 bits). Sets the transmitter application interface data path width to four 8-bit bytes (32 bits).
	40	Sets the internal transmitter data path width to two 10-bit bytes (20 bits). If 8B/10B encoding is selected, the transmitter application interface data path width will be set to four 8-bit bytes (32 bits). If 8B/10B encoding is not selected, the transmitter application interface data path width will be set to four 10-bit bytes (40 bits).
Encoding	None	Data stream is passed with no conversion.
	None (MSB First)	Same as above but reorders bits for applications expecting most significant bit first.
	8B/10B	Data stream is passed to an internal 8B/10B encoder prior to transmission.
Reference Clock		Select from the list the optimal reference clock frequency to be provided by the application. The XAUI example uses 156.25 MHz.
Use Oversampling		The GTX Wizard supports Oversampling for line rates between 600 Mbps and 1300 Mbps. For line rates of 120 to 600 Mbps, this option is automatically selected and the check box is disabled. For line rates of 600-1300 Mbps, the checkbox is enabled allowing optional selection of this feature. This option is not available for XAUI since the line rate exceeds the permissible range.
No TX		Selecting this option disables the TX path of the GTX. The transceiver will act as a receiver only. The XAUI example design requires both TX and RX functionality.

Table 3-2: **RX Settings**

Options		Description
Line Rate		Set to the desired target line rate in Gbps. Can be independent of the transmit line rate. The XAUI example uses 3.125 Gbps.
Data Path Width	8	Sets the internal receiver data path width to two 8-bit bytes. Sets the receiver application interface data path width to a single 8-bit byte.
	10	Sets the internal receiver data path width to two 10-bit bytes. If 8B/10B encoding is selected, the receiver application interface data path width will be set to 8-bits. If 8B/10B encoding is not selected, the receiver application interface data path width will be set to 10-bits.
	16	Sets both the internal receiver data path and the receiver application interface data path width to two 8-bit bytes (16 bits).
	20	Sets the internal receiver data path width to two 10-bit bytes (20 bits). If 8B/10B encoding is selected, the receiver application interface data path width will be set to two 8-bit bytes (16 bits). If 8B/10B encoding is not selected, the receiver application interface data path width will also be set to two 10-bit bytes (20 bits).
	32	Sets the internal receiver data path width to two 8-bit bytes (16 bits). Sets the receiver application interface data path width to four 8-bit bytes (32 bits).
	40	Sets the internal receiver data path width to two 10-bit bytes (20 bits). If 8B/10B encoding is selected, the receiver application interface data path width will be set to four 8-bit bytes (32 bits). If 8B/10B encoding is not selected, the receiver application interface data path width will be set to four 10-bit bytes (40 bits).
Decoding	None	Data stream is passed with no conversion.
	None (MSB First)	Same as above but reorders bytes for applications expecting most significant byte first.
	8B/10B	Data stream is passed to an internal 8B/10B encoder prior to transmission.
Reference Clock		Select from the list the optimal reference clock frequency to be provided by the application. The XAUI example uses 156.25 MHz.
Use Oversampling		The GTX Wizard supports oversampling for line rates between 600 Mbps and 1300 Mbps. For line rates of 120 to 600 Mbps, this option is automatically selected and the check box is disabled. For line rates of 600-1300 Mbps, the checkbox is enabled allowing optional selection of this feature. This option is not available for XAUI since the line rate exceeds the permissible range.
Use RXOVERSAMPLEERR Ports		Select this option to have the RXOVERSAMPLEERR signals from the transceiver available to the application. The XAUI example does not use this signal.
No RX		Selecting this option disables the RX path of the GTX. The transceiver will act as a transmitter only. The XAUI example design requires both TX and RX functionality.

Table 3-3: **Additional Options**

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

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

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

GTX Placement and Clocking

The GTX Placement and Clocking screen (Page 2) of the Wizard (Figure 3-6) allows you to determine the placement of the GTX transceivers and the reference clock source.

The number of available GTX transceivers appearing on this page depends on the selected target device and package. The XAUI example design uses four GTX transceivers. Table 3-5 describes the GTX transceiver selection and reference clock options and Table 3-6 describes the reference clock source options.

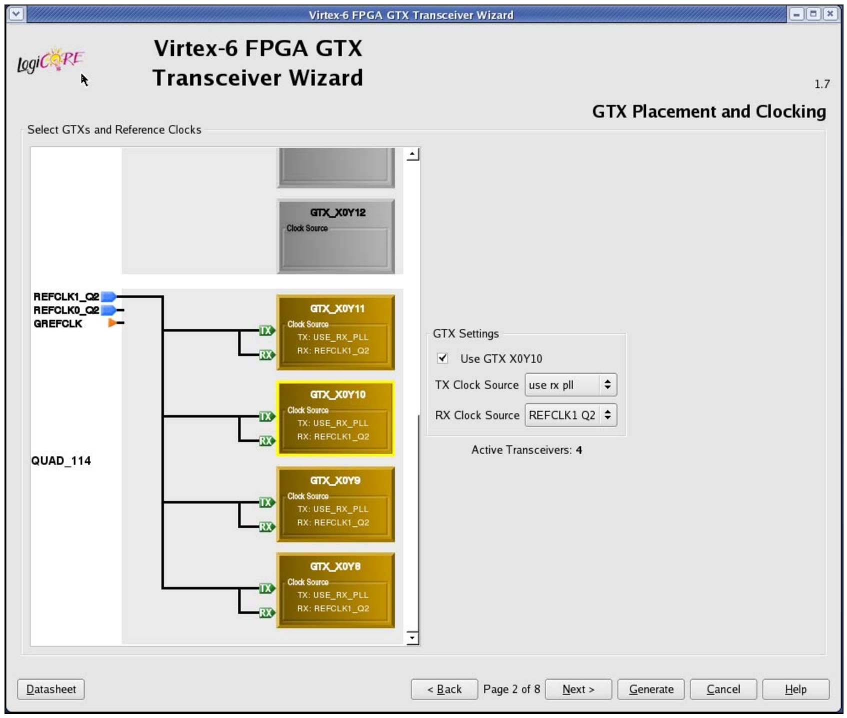

Figure 3-6: **GTX Placement and Clocking - Page 2**

Table 3-5: **Select Transceiver and Reference Clocks**

Option	Description
GTX	Select the individual GTX transceivers by location to be used in the target design. The XAUI example requires four transceivers.
TXREFCLK Source RXREFCLK Source	Determines the source for the reference clock signal provided to each selected GTX transceiver (see Table 3-6). Two differential clock signal input pin pairs, labeled REFCLK0 and REFCLK1 are provided for every four transceivers. The groups are labelled Q0 through Q4 starting at the bottom of the transceiver column. Each transceiver has access to the local signal group and one or two adjacent groups depending upon the transceiver position. The XAUI example uses the REFCLK1 signal from the group local to the four selected GTX transceivers (REFCLK1 Q0 option).

Table 3-6: **Reference Clock Source Options**

Option	Description
use rx pll	Available for TXREFCLK only. Internally connects the TXREFCLK input to the receiver PLL.
REFCLK0/01 Q0	GTX reference clock local to transceivers Y0-Y3
REFCLK0/01 Q1	GTX reference clock local to transceivers Y4-Y7
REFCLK0/01 Q2	GTX reference clock local to transceivers Y8-Y11
REFCLK0/01 Q4	GTX reference clock local to transceivers Y12-Y15
REFCLK0/01 Q5	GTX reference clock local to transceivers Y16-Y19
REFCLK0/01 Q6	GTX reference clock local to transceivers Y20-Y23
REFCLK0/01 Q7	GTX reference clock local to transceivers Y24-Y27
REFCLK0/01 Q8	GTX reference clock local to transceivers Y28-Y31
REFCLK0/01 Q9	GTX reference clock local to transceivers Y32-Y35
GREFCLK	Reference clock driven by internal fabric. Lowest performance option.

Synchronization and Alignment

The Synchronization and Alignment screen (Page 3) of the Wizard (Figure 3-7) allows you to control latency, buffering, and clocking of the transmitter and receiver. The RX comma alignment settings are also provided.

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

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

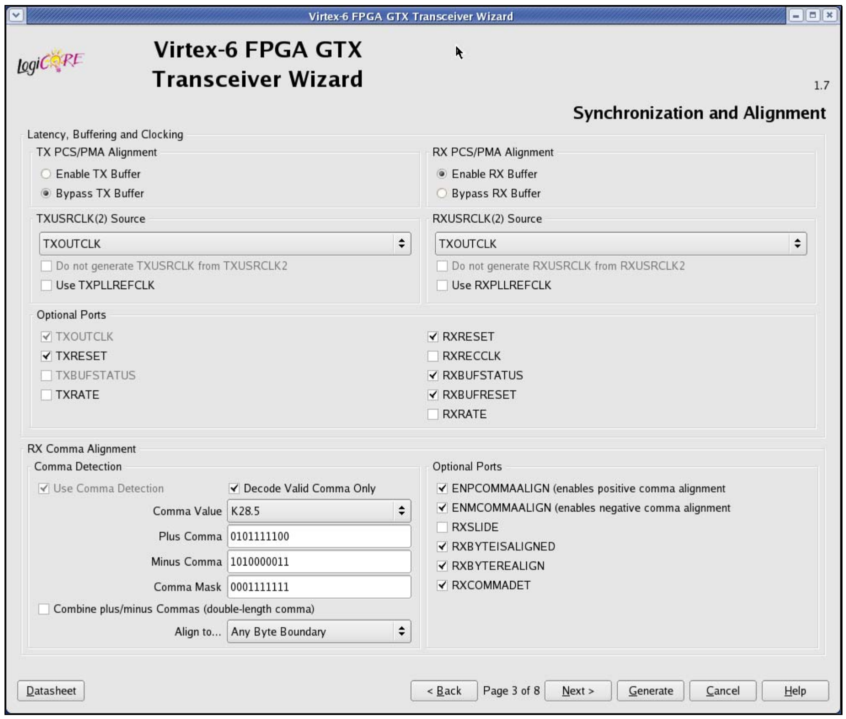

Figure 3-7: **Synchronization and Alignment - Page 3**

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

Table 3-7: **TXUSRCLK and RXUSRCLK Source**

<table>
<tr><th colspan="2">Option</th><th>Description</th></tr>
<tr><td rowspan="3">TX</td><td>TXOUTCLK</td><td>TXUSRCLK is driven by TXOUTCLK.</td></tr>
<tr><td>Generate TXUSRCLK from TXUSRCLK2</td><td>Brings the TXUSRCLK input signal out to a port at the top-level of the wrapper so it can be provided by the application. Optionally available when single-byte data path width is used and TX buffer bypass is disabled. Not available for two-byte data path width. Mandatory with 4-byte data path width.
The XAUI example does not use this feature.</td></tr>
<tr><td>Use TXPLLREFCLK</td><td>TXUSRCLK is driven by internally generated TXPLLREFCLK.</td></tr>
<tr><td rowspan="4">RX</td><td>TXOUTCLK</td><td>RXUSRCLK is driven by TXOUTCLK. This option is not available if the RX buffer is bypassed.</td></tr>
<tr><td>RXRECCLK</td><td>RXUSRCLK is driven by RXRECCLK. This option is required if the RX buffer is bypassed.</td></tr>
<tr><td>Generate RXUSRCLK from RXUSRCLK2</td><td>Brings the RXUSRCLK input signal out to a port at the top-level of the wrapper so it can be provided by the application. Optionally available when single-byte data path width is used without channel bonding. Not available for two-byte data path width. Mandatory with 4-byte data path width.
The XAUI example does not use this feature.</td></tr>
<tr><td>Use RXPLLREFCLK</td><td>RXUSRCLK is driven by internally generated RXPLLREFCLK.</td></tr>
</table>

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

Table 3-8: **Optional Ports**

Option	Description
TXOUTCLK	Parallel clock signal generated by the GTX transceiver. This option is required when selected as an input to either TXUSRCLK or RXUSRCLK.
TXRESET	Active-High reset signal for the transmitter PCS logic.
TXBUFSTATUS	Two-bit signal monitors the status of the TX elastic buffer. This option is not available when the TX buffer is bypassed.
RXRESET	Active-High reset signal for the receiver PCS logic.
RXRECCLK	Recovered clock signal from the CDR logic. This option is required when selected as an input to RXUSRCLK.
RXBUFSTATUS	Indicates the condition of the RX elastic buffer. This option is not available when the RX buffer is bypassed.
RXBUFRESET	Active-High reset signal for the RX elastic buffer logic. This option is not available when the RX buffer is bypassed.

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

Table 3-9: **Comma Detection**

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

Preemphasis, Termination, and Equalization

The Preemphasis, Termination, and Equalization screen (Page 4) of the Wizard (Figure 3-8) allows you to set the preemphasis, termination, and equalization options.

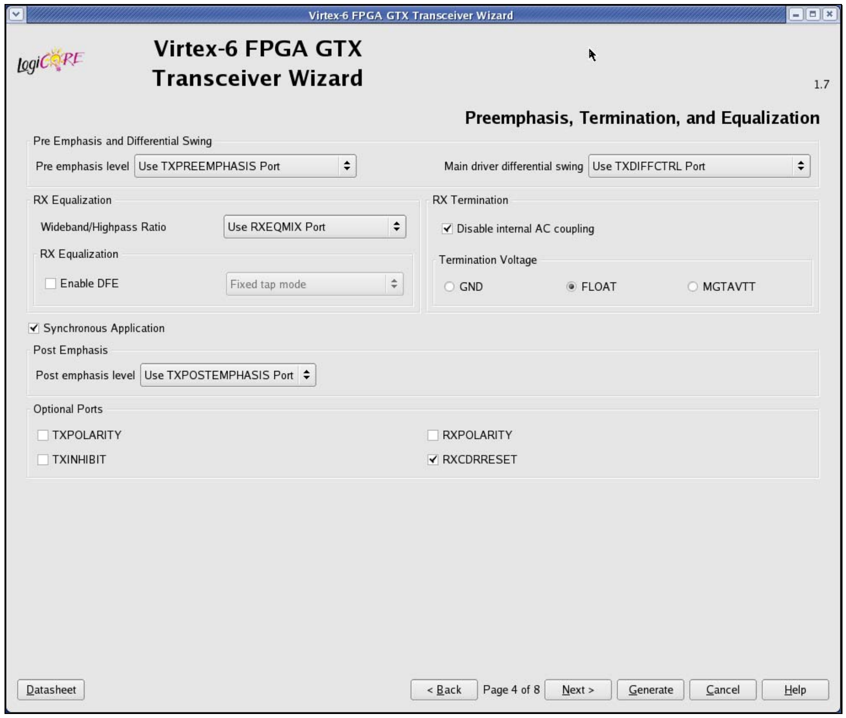

Figure 3-8: **Preemphasis, Termination, and Equalization - Page 4**

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

Table 3-10: **Preemphasis and Differential Swing**

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

Table 3-11 describes the RX Equalization settings.

Table 3-11: **RX Equalization**

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

Table 3-12 describes the RX Termination settings.

Table 3-12: **RX Termination**

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

The post emphasis level setting specifies the transmitter post-cursor pre-emphasis level setting. See the *Virtex-6 FPGA GTX Transceivers User Guide* for details.

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

Table 3-13: **Optional Ports**

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

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

The PCI Express, SATA, OOB, PRBS, and RX Loss of Sync screen (Page 5) of the Wizard (Figure 3-9) allows you to configure the receiver for PCI Express® and Serial ATA (SATA) features. In addition, configuration options for the RX out of band signal (OOB), PRBS detector, and the loss of sync state machine settings are provided.

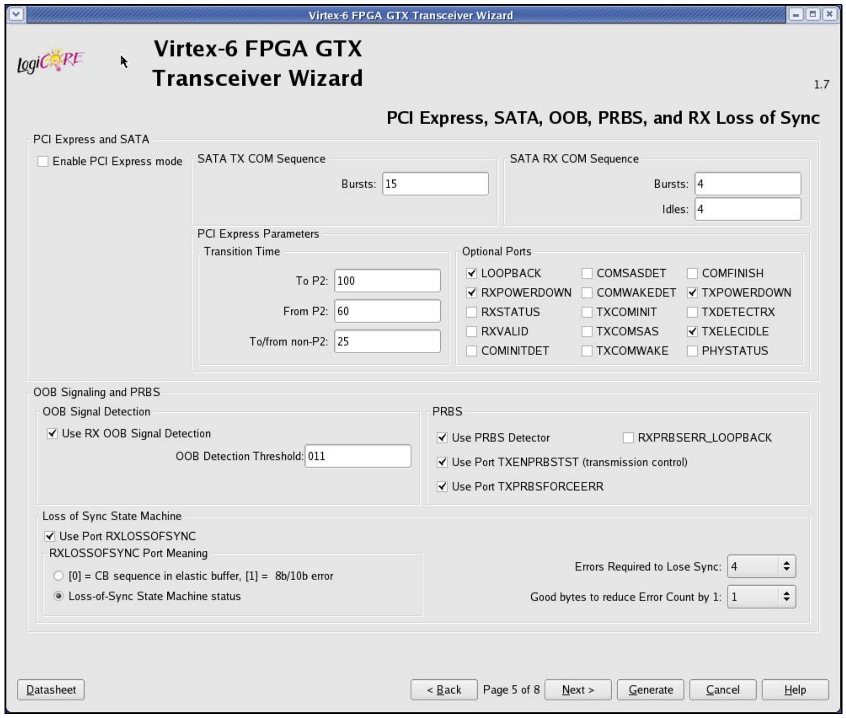

Figure 3-9: **PCI Express, SATA, OOB, PRBS, and RX Loss of Sync - Page 5**

Table 3-14 details the receiver SATA configuration options.

Table 3-14: **Receiver Serial ATA Options**

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

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

Table 3-15: **PCI Express Parameters**

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

Table 3-16 shows the OOB signal detection options.

Table 3-16: **OOB Signal Detection**

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

Table 3-17 details the PRBS settings.

Table 3-17: **PRBS**

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

Table 3-18 described the Loss of Sync State Machine settings.

Table 3-18: **Loss of Sync State Machine**

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

Channel Bonding Sequence

The Channel Bonding Sequence screen (Page 6) of the Wizard (Figure 3-10) allow you to define the channel bonding sequence(s). Table 3-19, page 33 describes the channel bonding setup options. Table 3-20, page 34 describes the sequence definition settings.

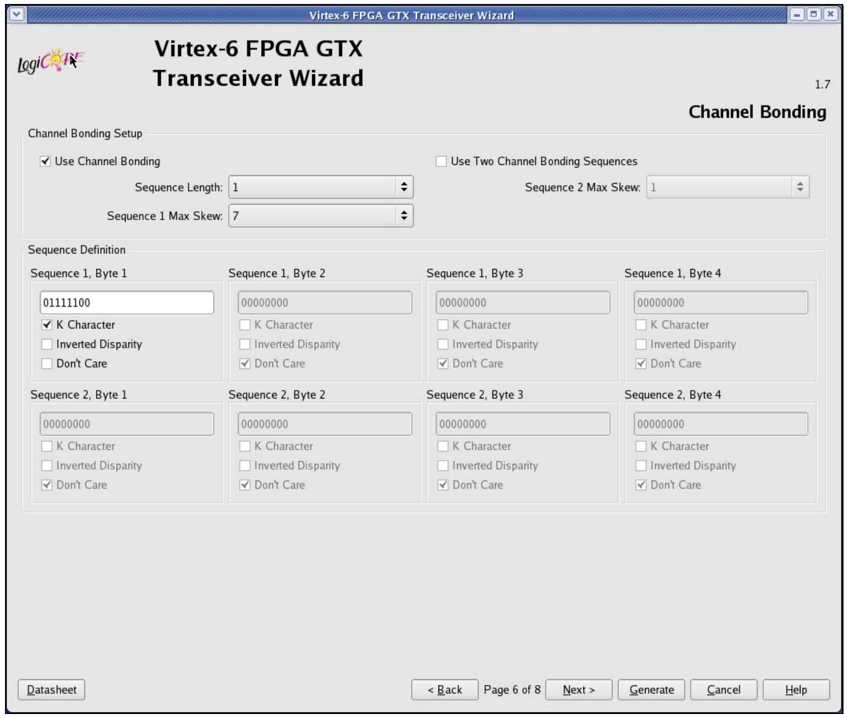

Figure 3-10: **Channel Bonding - Page 6**

Table 3-19: **Channel Bonding Setup**

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

Table 3-20: **Channel Bonding and Clock Correction Sequences**

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

Clock Correction Sequence

The Clock Correction Sequence screen (Page 7) of the Wizard (Figure 3-11) allows you to define the clock correction sequence. Table 3-21, page 35 describes the clock correction setup parameters. See Table 3-20, page 34 for a description of the sequence definition settings.

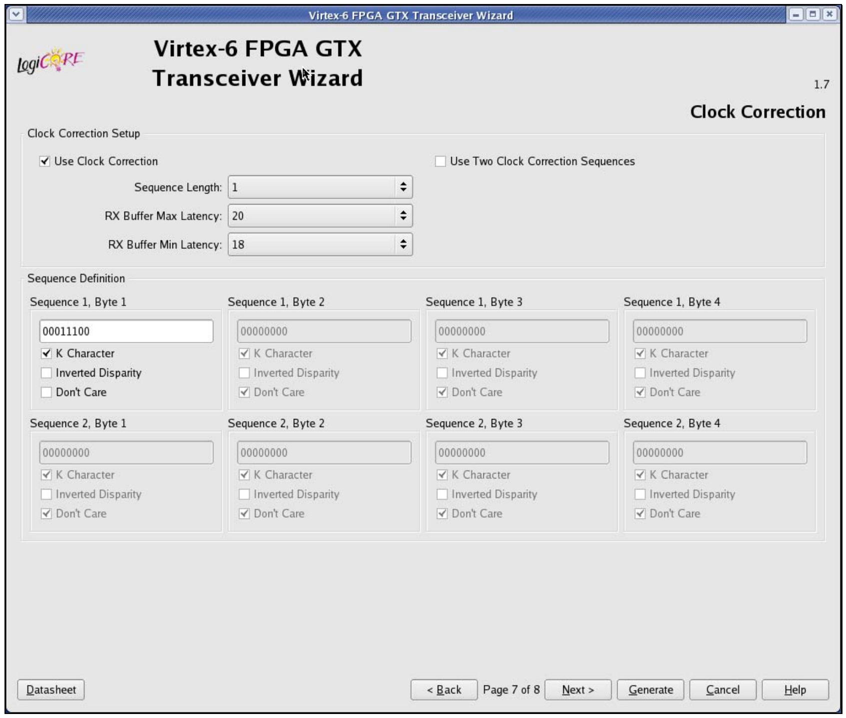

Figure 3-11: **Clock Correction - Page 7**

Table 3-21: **Clock Correction Setup**

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

Table 3-21: **Clock Correction Setup *(Cont'd)***

Option	Description
RX Buffer Min Latency	Select from the drop down list the minimum number of characters to permit in the receive buffer before clock correction attempts to add extra clock correction sequences to the receive buffer. Also determines the minimum latency of the receive buffer in RXUSRCLK cycles. The XAUI example uses 18.
Use Two Clock Correction Sequences	Activates the optional second clock correction sequence. Detection of either sequence triggers clock correction.

Summary

The Summary screen (Page 8) of the Wizard (Figure 3-12) provides a summary of the selected configuration parameters. After reviewing the settings, click Generate to exit and generate the wrapper.

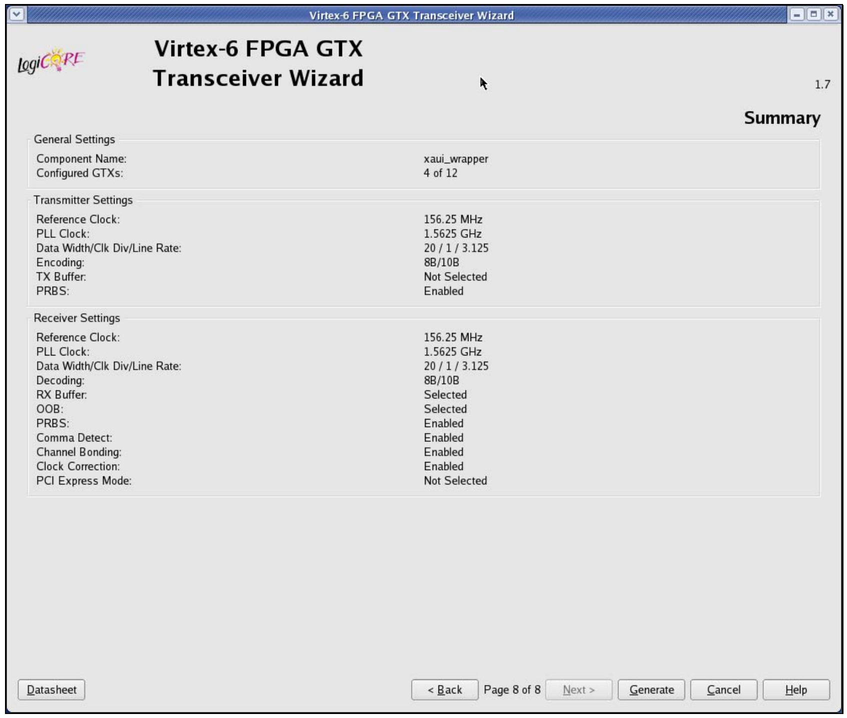

Figure 3-12: **Summary - Page 8**

Chapter 4

Quick Start Example Design

Overview

This chapter introduces the example design that is included with the GTX transceiver wrappers. The example design demonstrates how to use the wrappers and demonstrates some of the key features of the GTX transceiver. For detailed information about the example design, see Chapter 5, "Detailed Example Design."

Implementing the Example Design

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

The directory structure for the XAUI example is provided in Chapter 5, "Detailed Example Design."

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

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

For Windows

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

For Linux

```
% cd xaui_wrapper/implement
% implement.sh
```

Note: Substitute *Component Name* string for "xaui_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.

Simulating the Example Design

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

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® 12.3 available in the ISE software documentation, for instructions on how to compile ISE simulation libraries.

Table 4-1: **Required ModelSim Simulation Libraries**

HDL	Library	Source Directories
Verilog	UNISIMS_VER	`<Xilinx dir>/virtex6/verilog/src/unisims` `<Xilinx dir>/virtex6/secureip/mti`
VHDL	UNISIM	`<Xilinx dir>/virtex6/vhdl/src/unisims/primitive` `<Xilinx dir>/virtex6/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 test bench, and adds the relevant signals to the wave window.

Using the ISE Simulator

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

Table 4-2: **Required ISim Simulation Libraries**

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

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

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

1. Set the current directory to
   `<project_directory>/<component_name>/simulation/functional`
2. Launch the simulation script:
   prompt> `simulate_isim.sh`

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

Chapter 5

Detailed Example Design

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

Directory and File Structure

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

Directory and File Contents

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

<project directory>

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

Table 5-1: **Project Directory**

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

Back to Top

<project directory>/<component name>

The <component name> directory contains the release notes file provided with the core, which may include last-minute changes and updates.

Table 5-2: **GTX Wrapper Component Name**

Name	Description
<project_dir>/<component_name>	
`v6_gtxwizard_readme.txt`	Release notes for the GTX wizard.
`<component_name>.pf`	Protocol description for the selected protocol from the GTX wizard.

Back to Top

<component name>/doc

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

Table 5-3: **Doc Directory**

Name	Description
<project_dir>/<component_name>/doc	
`v6_gtxwizard_ds708.pdf`	*LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard v1.7 Data Sheet*
`v6_gtxwizard_gsg516.pdf`	*LogiCORE IP Virtex-6 FPGA GTX Transceiver Wizard v1.7 Getting Started Guide*

Back to Top

<component name>/example design

The example design directory contains the example design files provided with the core.

Table 5-4: **Example Design Directory**

Name	Description
<project_dir>/<component_name>/example_design	
`frame_check.v[hd]`	Frame-check logic to be instantiated in the example design.
`frame_gen.v[hd]`	Frame-generator logic to be instantiated in the example design.
`gtx_attributes.ucf`	Constraints file containing the GTX attributes generated by the GTX Wizard GUI settings.
`tx_sync.v[hd]`	TX sync logic module to be instantiated in the example design. Performs phase synchronization for all active TX data paths. Available for use in the target design.
`<component_name>_top.ucf`	Constraint file for mapping the GTX wrapper example design onto a Virtex-6 device.
`<component_name>_top.v[hd]`	Top-level example design. Contains GTX transceiver wrapper, reset logic, and instantiations for frame generator, frame-checker, and TX sync logic. Also contains definitions for test frame data and ChipScope™ Pro tools module instantiation. See Figure 3-1, page 15.

Back to Top

<component name>/implement

The implement directory contains the core implementation script files.

Table 5-5: **Implement Directory**

Name	Description
<project_dir>/<component_name>/implement	
chipscope_project.cpj	ChipScope Pro tools project file.
data_vio.ngc	ChipScope Pro Virtual Input/Output (VIO) core netlist.
icon.ngc	ChipScope Pro Integrated Controller (ICON) core netlist.
ila.ngc	ChipScope Pro Integrated Logic Analyzer (ILA) core netlist.
implement.bat	A Windows batch file that processes the example design through the Xilinx tool flow.
implement.sh	A Linux shell script that processes the example design through the Xilinx tool flow.
implement_synplify.bat	A Windows batch file that processes the example design through Synplify synthesis and the Xilinx tool flow.
implement_synplify.sh	A Linux shell script that processes the example design through Synplify synthesis and the Xilinx tool flow.
synplify.prj	Synplify project file for the example design.
xst.prj	The XST project file for the example design; it lists all of the source files to be synthesized.
xst.scr	The XST script file for the example design that is used to synthesize the core, called from the implement script described above.

Back to Top

/implement/results

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

Table 5-6: **UCF Directory**

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

Back to Top

<component name>/simulation

The simulation directory contains the simulation scripts provided with the core.

Table 5-7: **Simulation Directory**

Name	Description
<project_dir>/<component_name>/simulation	
`demo_tb.v[hd]`	Test bench to simulate the provided example design. See “Simulating the Example Design,” page 40.

Back to Top

/simulation/functional

The functional directory contains functional simulation scripts provided with the core.

Table 5-8: **Functional Directory**

Name	Description
<project_dir>/<component_name>/simulation/functional	
`simulate_isim.sh`	ISE® simulator (ISim) simulation script.
`simulate_mti.do`	ModelSim simulation script.
`simulate_ncsim.sh`	Linux script for running simulation using Cadence Incisive Enterprise Simulator (IES).
`simulate_vcs.sh`	Linux script for running simulation using Synopsys VCS and VCS MX.
`ucli_command.key`	Command file for VCS simulator
`vcs_session.tcl`	Script for adding GTX wrapper signals to VCS wave window.
`wave_isim.tcl`	Script for adding GTX wrapper signals to the ISim wave viewer.
`wave_mti.do`	Script for adding GTX wrapper signals to the ModelSim wave viewer.
`wave_ncsim.sv`	Script for adding GTX wrapper signals to the Cadence IUS wave viewer.

Back to Top

Example Design Description

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

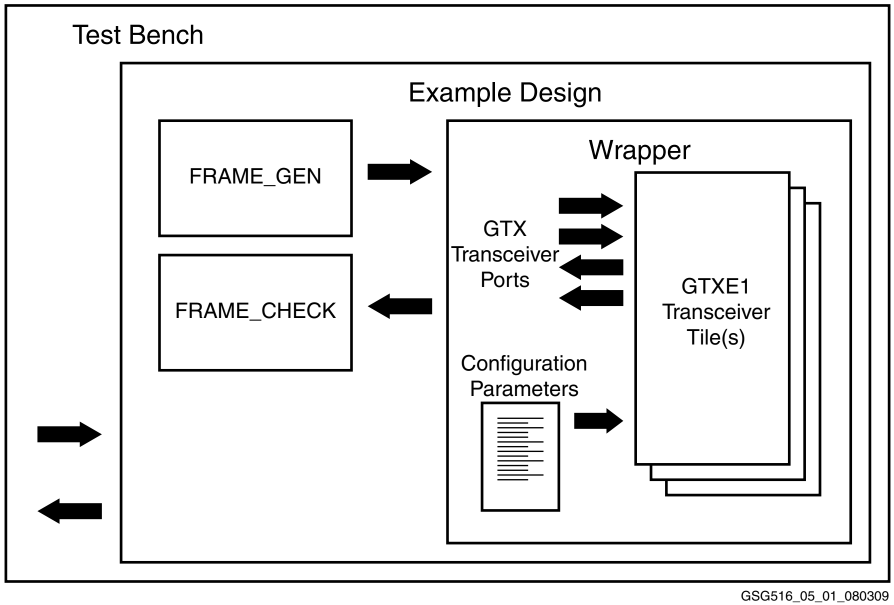

Figure 5-1: **Wrapper Block Diagram**

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 BRAM. This pattern can be easily modified by altering the parameters in the frame generator instantiation. The frame checker contains the same pattern in BRAM 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. Once the START_OF_PACKET_CHAR has been found, the received data will continuously be compared to the counting pattern stored in the BRAM at each RXUSRCLK2 cycle. Once comparison has begun, if the received data ever fails to match the data in the BRAM, checking of receive data will immediately stop, an error counter will be incremented and the frame checker will return to searching for the START_OF_PACKET_CHAR.

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

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

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

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

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

Example Design Hierarchy

The hierarchy for the design used in this example is shown below.

```
EXAMPLE_TB
   |___EXAMPLE_MGT_TOP
       |___XAUI_WRAPPER
       |   |___XAUI_WRAPPER_GTX (1 per transceiver)
       |
       |___TX_SYNC              (1 per transceiver)
       |___FRAME_GEN            (1 per transceiver)
       |___FRAME_CHECK          (1 per transceiver)
      (optional Chipscope support)
       |___shared_vio
       |___icon
       |___data_vio (TX)
       |___data_vio (RX)
       |___ila      (RX)
```