LogiCORE™ IP Virtex®-5 FPGA RocketIO™ GTX Transceiver Wizard v1.7

Getting Started Guide

UG204 (v1.7) April 19, 2010

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/24/08	1.2	Initial Xilinx release.
03/24/08	1.2.1	Minor typographical changes.
04/25/08	1.3	Updated version numbers and screen shots. Revised Installing the Wizard section. Minor edits: • Verifying Your Installation: Removed redundant support link. • Added Table 3-9. • Table 3-12: Change Wideband/Highpass ratio setting.
06/26/08	1.4	Revised version numbers and screen shots. Replaced page headings with descriptive names in Generating the Core. Edits: • Virtex-5 FPGA RocketIO GTX Transceiver Wizard: Added Wizard name in heading. • Generating the Core: Revised parameter tables and shading to match current values. • Line Rate and Protocol Template: Removed references to "Silicon Version." • Latency, Buffering, and Clocking: Revised TX PCS/PMA Phase Alignment section. • Implementing the Example Design: ♦ Added source file details. ♦ Added Table 3-23. ♦ Corrected directory name and added note in script examples.
09/19/08	1.5	Updated version numbers and screen shots. Updated "System Requirements." Edits: • Verifying Your Installation: Added reference to TXT devices. • Setting the Project Options, Item 4: Changed example device to XC5VTX150T. • Component Name and Tile Placement: ♦ Added paragraph describing column selection option. ♦ Removed "introducing graphic tile selection feature" sentence. ♦ Added description of single-column selection limitation. ♦ Added Table 3-1. • Added Detailed Example Design section.
06/24/09	1.6	Updated version numbers and screen shots. Miscellaneous edits throughout. Moved the example design and directory hierarchy to the new Chapter 4, "Detailed Example Design."
04/19/10	1.7	Tools and Wizard updates. Updated version numbers and screen shots. Updated Table 3-4, Table 3-17, Table 4-1, Table 4-4, Table 4-5, and Table 4-8.

Table of Contents

Chapter 4: Detailed Example Design

Appendix A: References

Preface

About This Guide

The *LogiCORE IP Virtex-5 FPGA RocketIO GTX Transceiver Wizard Getting Started Guide* describes the function and operation of the LogiCORE™ IP Virtex-5 FPGA RocketIO GTX Transceiver Wizard for the Virtex®-5 FXT and TXT families.

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, "Installation and Licensing" provides information about installing and licensing the Virtex-5 FPGA RocketIO GTX Transceiver Wizard.
- Chapter 3, "Running the Wizard" provides an overview of the Virtex-5 FPGA RocketIO GTX Transceiver Wizard, and a step-by-step tutorial to generate a sample RocketIO GTX transceiver wrapper with the Xilinx® CORE Generator™ tool.
- Chapter 4, "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.

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 *Development System Reference 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.
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}`**

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;*

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.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-II Platform FPGA User Guide.*
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Introduction

This chapter introduces the Virtex®-5 FPGA RocketIO™ GTX Transceiver Wizard core and provides related information, including additional resources, technical support, and submitting feedback to Xilinx.

The Virtex-5 FPGA RocketIO 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 RocketIO GTX transceivers for custom applications (Figure 1-1).

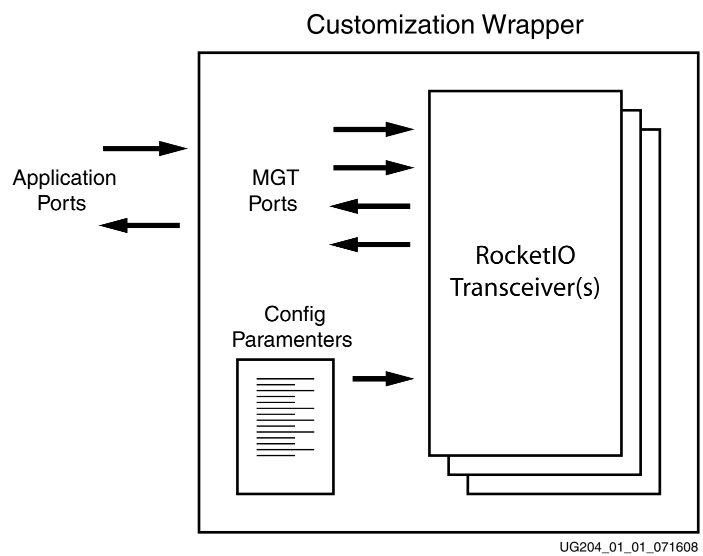

Figure 1-1: **GTX Transceiver Wizard Wrapper**

About the Wizard

The Virtex-5 FPGA RocketIO 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, "Installation and Licensing."

Additional Wizard Resources

For detailed information and updates about the Virtex-5 FPGA RocketIO GTX Transceiver Wizard, see the following documents located at the Architecture Wizards page:

- DS601: *Virtex-5 FPGA RocketIO GTX Transceiver Wizard Data Sheet*, [Ref 3]
- Virtex-5 FPGA RocketIO 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-5 FPGA RocketIO GTX Wizard.

Xilinx provides technical support for use of this product as described in the *LogiCORE IP Virtex-5 FPGA RocketIO GTX Transceiver Wizard 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-5 FPGA RocketIO GTX Transceiver Wizard and the accompanying documentation.

Virtex-5 FPGA RocketIO GTX Transceiver Wizard

For comments or suggestions about the Virtex-5 FPGA RocketIO 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).

Chapter 2

Installation and Licensing

This chapter provides instructions for installing the Virtex®-5 FGPA RocketIO™ GTX Transceiver Wizard in the Xilinx® CORE Generator™ tool. It is not necessary to obtain a license to use the Wizard.

System Requirements

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

Software

- ISE® 12.1 software
- Mentor Graphics ModelSim 6.5c

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 12.1 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 12.1 software. For the software installation instructions, see the ISE Design Suite Release Notes and Installation Guide available in ISE software Documentation [Ref 4].

Installing the Wizard

The Virtex-5 FPGA Virtex-5 FPGA RocketIO GTX Transceiver Wizard is included with the ISE 12.1 software. See *ISE CORE Generator IP Updates - Installation Instructions* for details on the installation of ISE 12.1.

Verifying Your Installation

Use the following procedure to verify that you have successfully installed the Virtex-5 FPGA RocketIO GTX Transceiver Wizard in the CORE Generator tool.

1. Start the CORE Generator tool.
2. After creating a new Virtex-5 FXT or TXT family project or opening an existing one, the IP core functional categories appear at the left side of the window, as shown in Figure 2-1.

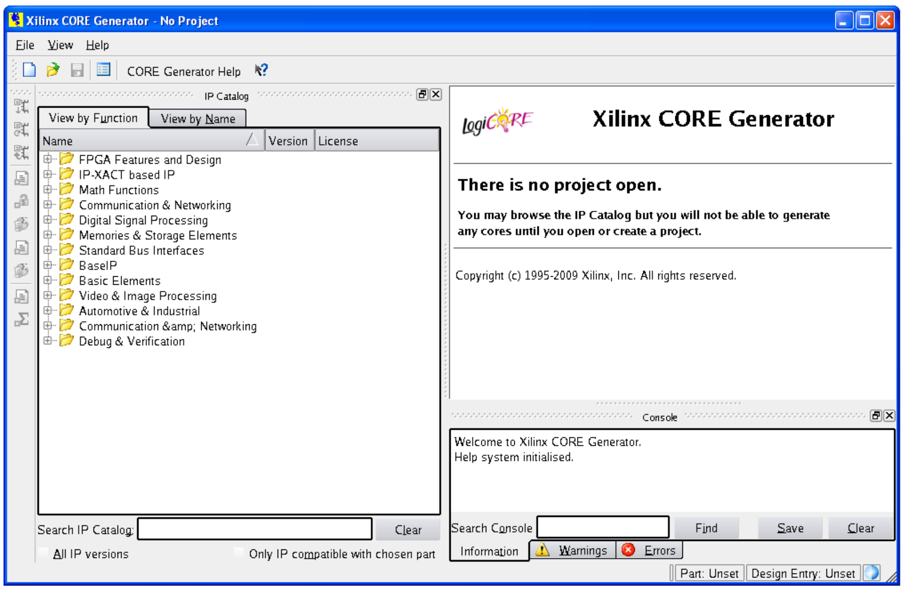

Figure 2-1: **CORE Generator Window**

3. Determine if the installation was successful by verifying that Virtex-5 FPGA RocketIO 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 RocketIO™ 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 RocketIO 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 RocketIO GTX transceiver wrapper implementing a four-lane XAUI port using four RocketIO GTX transceivers on two GTX_DUAL tiles
- 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 cores
- Scripts to synthesize and simulate the example design

The Virtex®-5 FPGA Virtex-5 FPGA RocketIO GTX Transceiver Wizard example design has been tested with Synplify Synopsys Synplify Pro D-2009.12 and XST 12.1 for synthesis and ModelSim 6.5c for simulation.

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

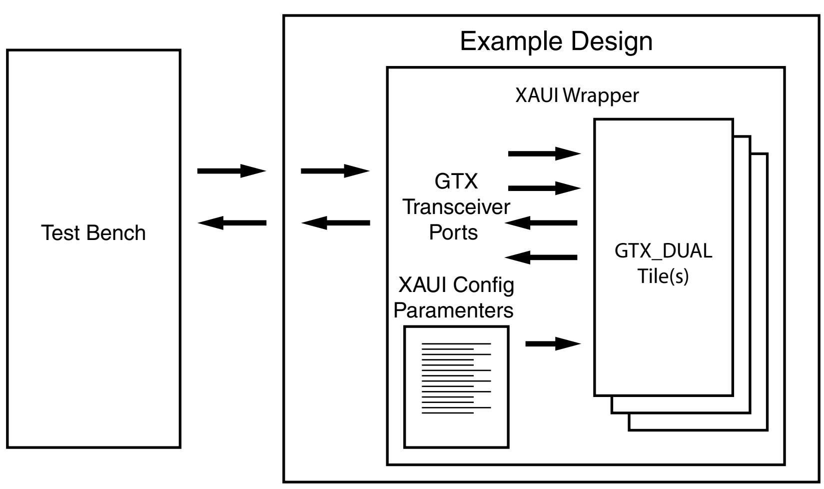

Figure 3-1: **Example Design**

Setting Up the Project

Before generating the example design, set up the project as shown 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/xaui_example`

2. Start the CORE Generator™ tool.

   For help starting and using the CORE Generator tool, see *CORE Generator Help*, available in ISE software documentation [Ref 4].

3. Choose **File** → **New Project** (Figure 3-2).
4. Optionally change the name of the `.cgp` file
5. Click **Save**.

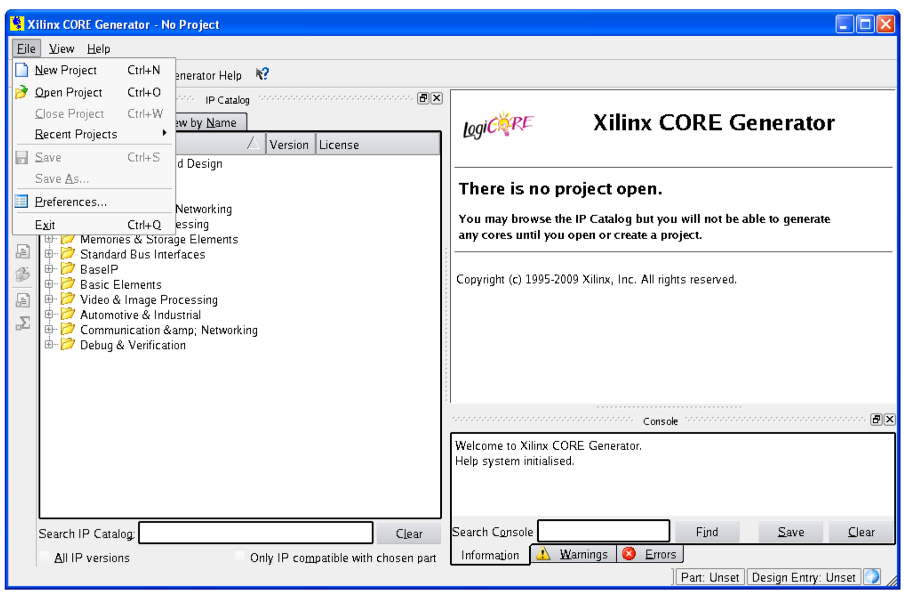

Figure 3-2: **Starting a New Project**

Setting the Project Options

Set the project options using the following steps:

1. Click **Part** in the option tree
2. Select **Virtex5** from the Family list.
3. Select a device from the Device list which supports RocketIO GTX transceivers.
4. Select an appropriate package from the Package list. This example uses the XC5VTX150T device (see Figure 3-3).

   Note: If an unsupported silicon family is selected, the Virtex-5 FPGA RocketIO GTX Transceiver Wizard appears light grey in the taxonomy tree and cannot be customized. Only devices containing RocketIO GTX transceivers are supported by the Wizard. See the *Virtex-5 Family Overview* [Ref 1] for a list of devices containing RocketIO GTX transceivers.
5. Click the **Generation** tab and select either Verilog or VHDL as the output language.
6. Click **OK**.

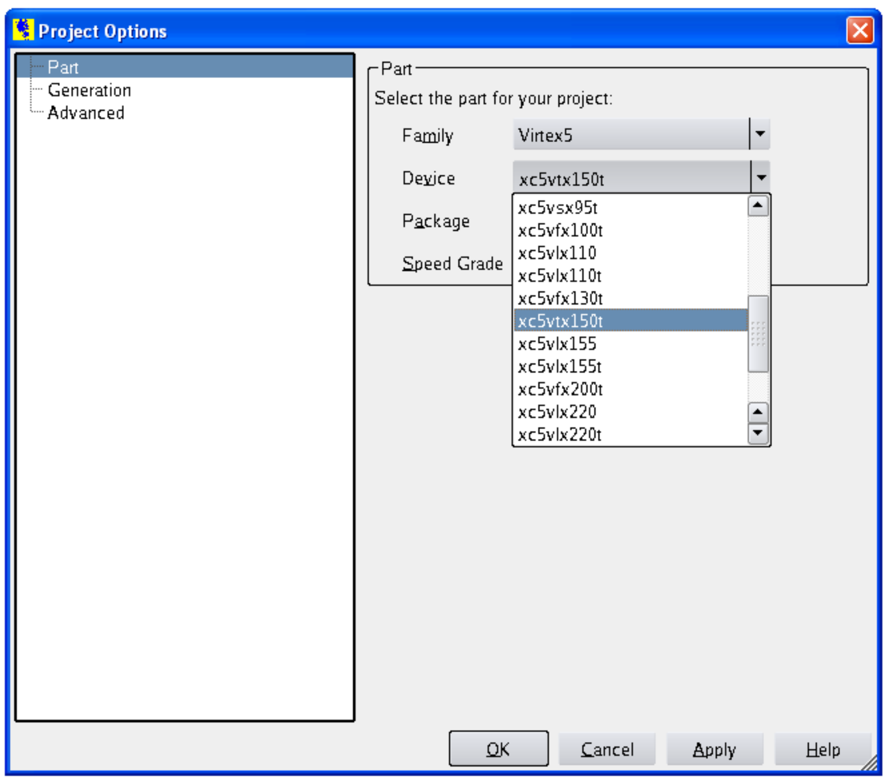

Figure 3-3: **Target Architecture Setting**

Generating the Core

This section provides instructions for generating an example RocketIO 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 4, "Detailed Example Design."

1. Locate the Virtex-5 FPGA RocketIO GTX Transceiver Wizard 1.7 in the taxonomy tree under:

   `/FPGA Features & Design/IO Interfaces`. (See Figure 3-4)

2. Double-click **Virtex-5 FPGA RocketIO GTX Transceiver Wizard 1.7** to launch the Wizard.

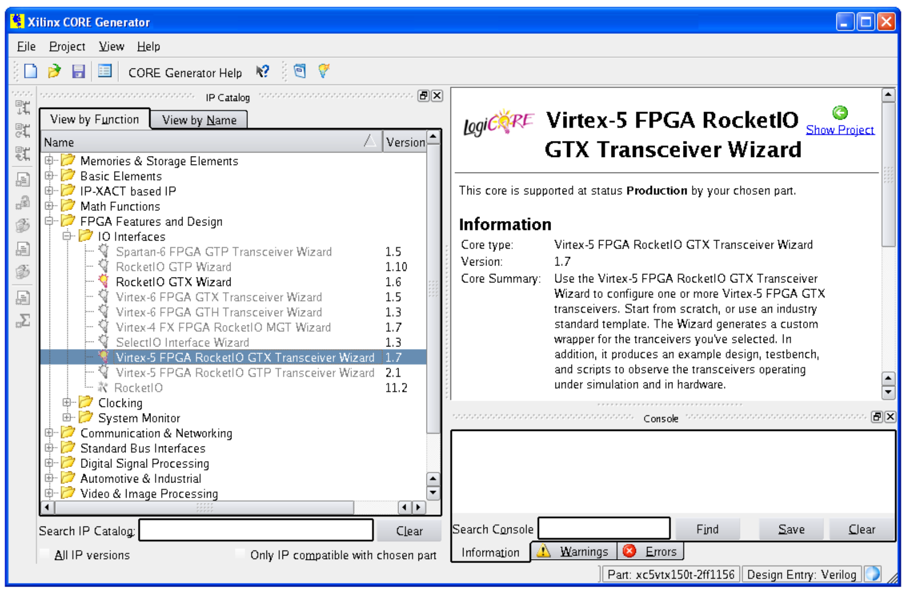

Figure 3-4: **Locating the RocketIO GTX Wizard**

Component Name and Tile Placement

Page 1 of the Wizard is for selecting a component name, determining the placement of the GTX_DUAL tiles, and selecting the reference clock source.

If the selected device features two transceiver columns, such as the XC5VTX150T used in this example, the option to select the column represented appears in the upper right of the page (Figure 3-5). If the selected device features a single transceiver column, this option is not displayed.

Each available GTX_DUAL tile is represented by a graphic block bearing the identification of the specific tile. Inactive tiles are grey with black lettering while active (selected) tiles are in color with white lettering and a drop shadow. To select a tile, click the checkbox in the upper left corner of the block. Transceiver tiles can be selected from one column only. If the target design uses tiles from both columns of a two-column device, the Wizard must be run twice.

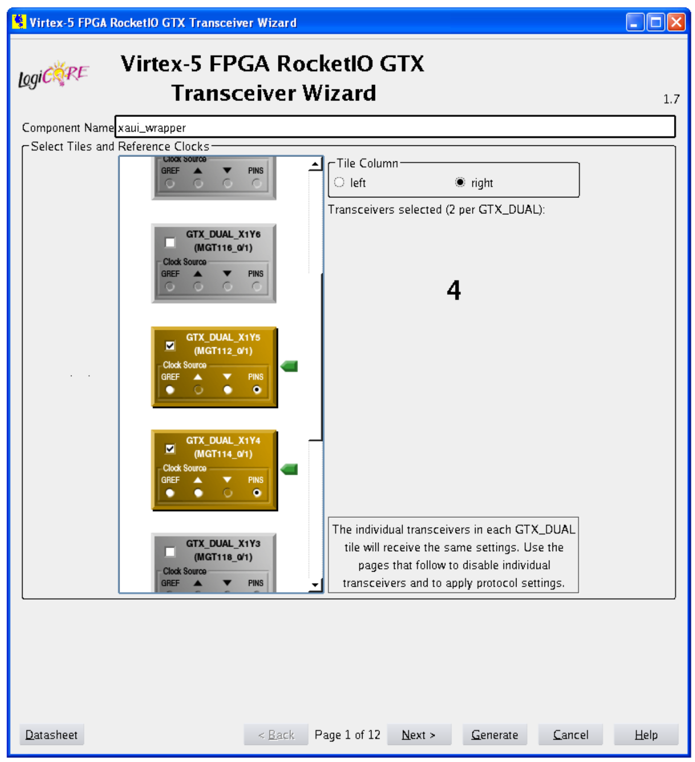

Figure 3-5: **RocketIO GTX Wizard Page 1**

The reference clock source is selected with the four radio buttons at the bottom of the block. Green arrow graphics indicate the source and routing of the clock signal. The center two radio buttons allow the use of the signal from the local pins of adjacent tiles. A single local reference clock signal can span up to three adjacent, active tiles.

1. In the Component Name field, enter a name for the core instance. This example uses the name **xaui_wrapper**.

The number of available GTX_DUAL tiles appearing on this page depends on the selected target device and package. The XAUI example design uses two tiles for a total of four GTX transceivers. Table 3-1, page 20 to Table 3-3, page 20 describe the GTX_DUAL tile selection and reference clock options. Note that the Column Selection option, Table 3-1, is visible only when a device is selected that features two transceiver columns.

Table 3-1: **Column Selection**

Option	Description
Left	Enables selection of tiles from the left-hand (X0) column of transceivers.
Right	Enables selection of tiles from the right-hand (X1) column of transceivers.

Table 3-2: **Select Tile and Reference Clocks**

Control	Description
Checkbox	Enables/disables the individual GTX_DUAL tiles. Determines the location to be used in the target design.
Clock Source Buttons	Determines the source for the reference clock signal provided to each selected GTX_DUAL (see Table 3-3). The XAUI example uses the reference clock from the differential input pins for the upper GTX_DUAL (X0Y4 option) for both tiles.

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

REFCLK Sources	Description
GREF	Reference clock driven by BUFG or BUFR. Lowest performance option.
PINS	Dedicated GTX reference clock from differential input pins for the specific tile represented.
▲ (North)	Dedicated GTX reference clock for the tile above. Source clock signal can span up to three selected tiles.
▼ (South)	Dedicated GTX reference clock for the tile below. Source clock signal can span up to three selected tiles.

Line Rate and Protocol Template

Page 2 of the Wizard (Figure 3-6) determines the line rate, reference clock frequency, encoding/decoding method, and data width. In addition, this page specifies a protocol template.

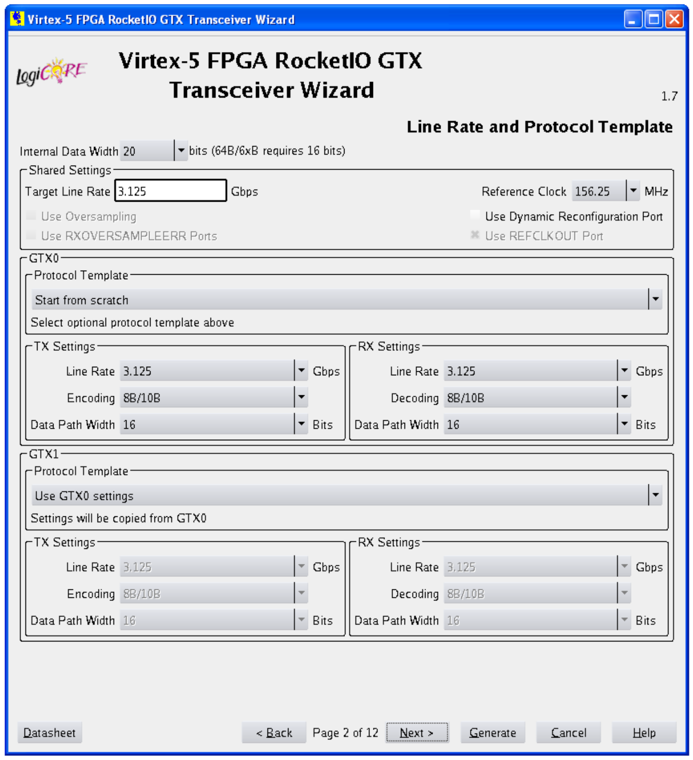

Figure 3-6: **RocketIO GTX Wizard Page 2**

1. Set the Internal Data Width to 16 or 20 as needed. If 8B/10B encoding/decoding is used, select 20-bit Internal Data Width to accommodate the encoded values. The application interface remains 8, 16, or 32 bits. With no encoding, a 20-bit Internal Data Width yields a 10-, 20-, or 40-bit application interface. If 64B/66B or 64B/67B encoding/decoding is used, select 16-bit Internal Data Width. The XAUI example requires 20 bits.

   Table 3-4, page 22 shows the options for the Shared Settings. These options establish the shared PMA PLL settings for both GTX transceivers on each tile.

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

   The remaining options are divided into GTX0 and GTX1 groups with identical parameters. These apply to the two GTX transceivers present in each GTX_DUAL tile. The remaining discussion in this chapter describes only the GTX0 portion.

Table 3-4: **Shared Settings**

Option	Description
Target Line Rate	Line rate in Gb/s desired for the target design. The XAUI example uses 3.125 Gb/s.
Reference Clock	Select from the list the optimal reference clock frequency to be provided by the application. The XAUI example uses 156.25 MHz.
Use Oversampling	The GTX Wizard supports Oversampling for line rates between 150 Mb/s and 1300 Mb/s. For line rates of 150 to 750 Mb/s, this option is automatically selected and the check box is disabled. For line rates of 750-1300 Mb/s, the checkbox is enabled allowing optional selection of this feature. This option is not available for XAUI since the line rate exceeds the permissible range.
Use RXOVERSAMPLERR Ports	Select this option to have the RXOVERSAMPLEERR signals from both transceivers available to the application. The XAUI example does not use this signal.
Use Dynamic Reconfiguration Port	Select this option to have the Dynamic Reconfiguration Port signals available to the application.
Use REFCLKOUT Port	Select this option to have the REFCLKOUT signal available to the application. Any options selected on the following Wizard pages that require this signal causes the forced selection of this option. The XAUI example requires this signal. (1)

Notes:
1. See Table 3-10, page 27.

2. From the Protocol Template list, select **Start from scratch** if you wish to manually set all parameters. Select from the list one of the available protocols to begin your design with a pre-defined protocol template. For GTX1 only, select **Use GTX0 settings** to automatically copy the settings from GTX0.

   The XAUI example uses the XAUI protocol template. Because both GTX transceivers are configured identically, the protocol template option for GTX1 is set to **Use GTX0 settings**.

Table 3-5 details the TX Settings options.

Table 3-5: **TX Settings**

Option		Description
Line Rate		Allows selection of the optimal line rate based on the shared PMA PLL settings divided by 1, 2, or 4. This option is typically set to the value of the Target Line Rate, but allows the transmit line rate to differ as needed. The XAUI example uses 3.125 Gb/s.
Encoding / Decoding	None	Data stream is passed with no conversion.
	None (MSB First)	Same as above but reorders bytes for applications expecting most significant byte first.
	8B/10B	Data stream is passed to an internal 8B/10B encoder prior to transmission.
	64B/66B with Int Seq Ctr	Enables Gear Box to insert 2-bit sync header into data stream using the internal sequence counter.
	64B/66B with Ext Seq Ctr	Enables Gear Box to insert 2-bit sync header into data stream with user-provided external sequence counter.
	64B/67B with Int Seq Ctr	Enables Gear Box to insert 3-bit sync header into data stream using the internal sequence counter.
	64B/67B with Ext Seq Ctr	Enables Gear Box to insert 3-bit sync header into data stream with user-provided external sequence counter.
Data Path Width	8	Sets the transmitter application interface data path width to a single 8-bit byte.
	16	Sets the transmitter application interface data path width to two 8-bit bytes (16 bits).
	32	Sets the transmitter application interface data path width to four 8-bit bytes (32 bits).
	10	Sets the transmitter application interface data path width to a single 10-bit byte. This option is only available if internal data width is 20 and no encoding is used.
	20	Sets the transmitter application interface data path width to two 10-bit bytes (20 bits). This option is only available if internal data width is 20 and no encoding is used.
	40	Sets the transmitter application interface data path width to four 10-bit bytes (40 bits). This option is only available if internal data width is 20 and no encoding is used.

Table 3-6 details the RX Settings options.

Table 3-6: **RX Settings**

Option		Description
Line Rate		Allows selection of the optimal line rate based on the shared PMA PLL settings divided by 1, 2, or 4. This option is typically set to the value of the Target Line Rate, but allows the receive line rate to differ as needed. The XAUI example uses 3.125 Gb/s.
Encoding / 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 decoder after receiving.
	64B/66B	Enables Gear Box to extract 2-bit sync header from data stream.
	64B/67B	Enables Gear Box to extract 3-bit sync header from data stream.
Data Path Width	8	Sets the receiver application interface data path width to a single 8-bit byte.
	16	Sets the receiver application interface data path width to two 8-bit bytes (16 bits).
	32	Sets the receiver application interface data path width to four 8-bit bytes (32 bits).
	10	Sets the receiver application interface data path width to a single 10-bit byte. This option is only available if internal data width is 20 and no encoding is used.
	20	Sets the receiver application interface data path width to two 10-bit bytes (20 bits). This option is only available if internal data width is 20 and no encoding is used.
	40	Sets the receiver application interface data path width to four 10-bit bytes (40 bits). This option is only available if internal data width is 20 and no encoding is used.

8B/10B Optional Ports

Page 3 of the Wizard (Figure 3-7) is for selecting the 8B/10B-specific optional ports. Placing a check next to one of the listed optional port names makes that port available in the wrapper for use by the application. Table 3-7, page 25 details the available TX and RX 8B/10B optional ports.

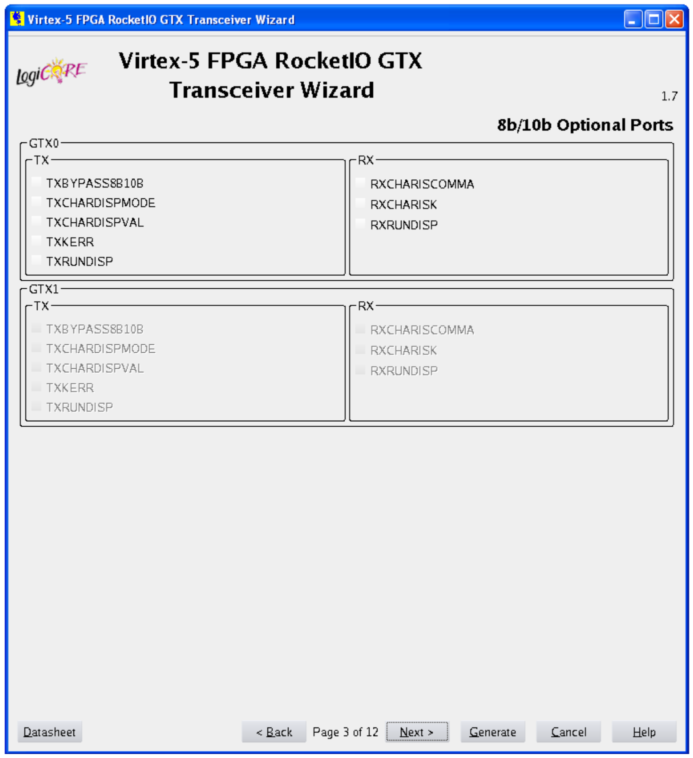

Figure 3-7: **RocketIO GTX Wizard Page 3**

Table 3-7: **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	Two-bit wide ports control disparity of outgoing 8B/10B data. High-order bit affects high-order byte of data path.
	TXCHARDISPVAL	
	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.

Table 3-7: **8B/10B Optional Ports *(Cont'd)***

Option		Description
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.

Latency, Buffering, and Clocking

Page 4 of the Wizard (Figure 3-8) is for controlling latency, buffering, and clocking of the transmitter and receiver.

The **TX PCS/PMA Phase Alignment** setting controls whether the TX buffer is enabled or bypassed. This setting is available for both GTX transceivers independently. See the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 2] for details on this setting.

The XAUI example uses the lane-to-lane deskew option.

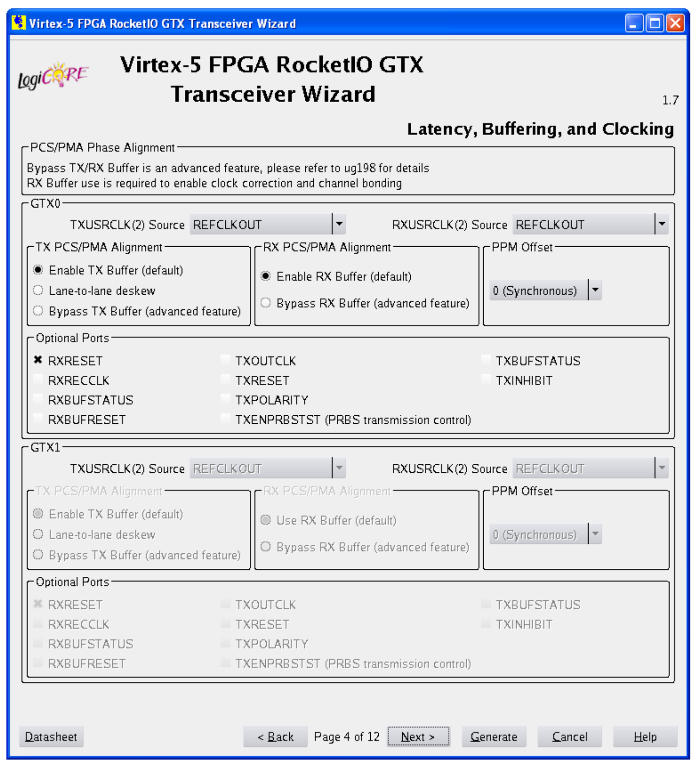

Figure 3-8: **RocketIO GTX Wizard Page 4**

Table 3-8 details the TXUSRCLK and RXUSRCLK source signal options.

Table 3-8: **TXUSRCLK and RXUSRCLK Source**

Option		Description
TX	TXOUTCLK	TXUSRCLK is driven by TXOUTCLK. This option is not available if the TX Phase Alignment Circuit is used.
	REFCLKOUT	TXUSRCLK is driven by REFCLKOUT. This option is required if the TX Phase Alignment Circuit is used.
RX	TXOUTCLK	RXUSRCLK is driven by TXOUTCLK. This option is not available if the RX Phase Alignment Circuit is used.
	RXRECCLK	RXUSRCLK is driven by RXRECCLK**.** This option is required if the RX Phase Alignment Circuit is used.
	REFCLKOUT	RXUSRCLK is driven by REFCLKOUT. This option is not available if the RX Phase Alignment Circuit is used.

The RX PCS/PMA Alignment setting controls whether the RX Phase Alignment circuit is enabled. Like the TX Phase alignment circuit, this setting is available for both GTX transceivers independently.

The XAUI example does not use the RX Phase Alignment circuit.

The PPM Offset setting optimizes the receiver CDR logic for the desired PPM tolerance range. Table 3-9 shows the available PPM offset settings.

Table 3-9: **PPM Offset**

Option	Description
0	Use with synchronous applications (zero tolerance).
≤ ±100	Use with applications where clock tolerance is less than or equal to 100 PPM
> ±100 or SSC	Use with applications where clock tolerance is greater than 100 PPM or spread-spectrum clocking is used.

Table 3-10 shows the optional ports available on this page.

Table 3-10: **Latency, Buffering, and Clocking Optional Ports**

Option	Description
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 Phase Alignment circuit is used.
RXBUFRESET	Active-High reset signal for the RX elastic buffer logic. This option is not available when the RX Phase Alignment circuit is used.
TXOUTCLK	Parallel clock signal generated by the GTX transceiver. This option is required when selected as an input to either TXUSRCLK or RXUSRCLK. This option is not available when the TX Phase Alignment circuit is used.
TXRESET	Active-High reset signal for the transmitter PCS logic.
TXPOLARITY	Active-High signal to invert the polarity of the transmitter output.

Table 3-10: **Latency, Buffering, and Clocking Optional Ports**

Option	Description
TXENPRBSTST	Two-bit signal to enable the PRBS test pattern generator.
TXBUFSTATUS	Two-bit signal monitors the status of the TX elastic buffer. This option is not available when the TX Phase Alignment circuit is used.
TXINHIBIT	Active-High signal forces transmitter output to steady state.

Preemphasis, Termination, and Equalization

Page 5 of the Wizard (Figure 3-9) is for setting the Preemphasis, Termination, and Equalization options.

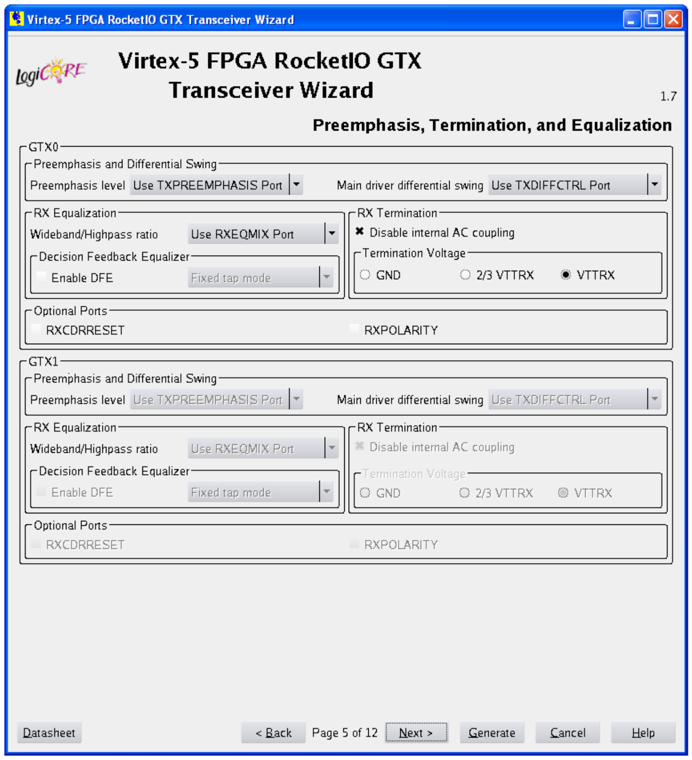

Figure 3-9: **RocketIO GTX Wizard Page 5**

Table 3-11 details the Preemphasis and Differential Swing settings.

Table 3-11: **Preemphasis and Differential Swing**

Option	Description
Preemphasis Level	Specifies the output pre-emphasis setting in 6.5% steps from 0% to approximately 45%. Selecting **Use TXPREEMPHASIS port** enables the optional TXPREEMPHASIS configuration port to dynamically set the pre-emphasis level. The XAUI example uses the default setting of `000` (0%). See the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 2] 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 in 100 mV steps from approximately 800 mV to 200 mV. 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 default setting `000` (800 mV). See the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 2] for a table mapping TXDIFFCTRL value settings to differential swing levels.

Table 3-12 describes the RX Equalization settings.

Table 3-12: **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 setting `11` (bypass with gain).
Enable DFE	Enables the Decision Feedback Equalizer and brings out the required ports. See the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 2] 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 3-13 describes the RX Termination settings.

Table 3-13: **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 **2/3 VTTRX** or **VTTRX** applies an internal voltage reference source to the internal termination network at the level specified. The XAUI example uses the **2/3 VTTRX** setting.

Table 3-14 describes the optional ports.

Table 3-14: **Optional Ports**

Option	Description
RXCDRRESET	Active-High reset signal causes the CDR logic to unlock and return to the shared PLL frequency.
RXPOLARITY	Active-High signal inverts the polarity of the receive data signal.

RX OOB, PRBS, and Loss of Sync

Page 6 of the Wizard (Figure 3-10) is for configuring the RX Out of Band signal (OOB) and PRBS Detector options. Also on this page are the Loss of Sync State Machine settings.

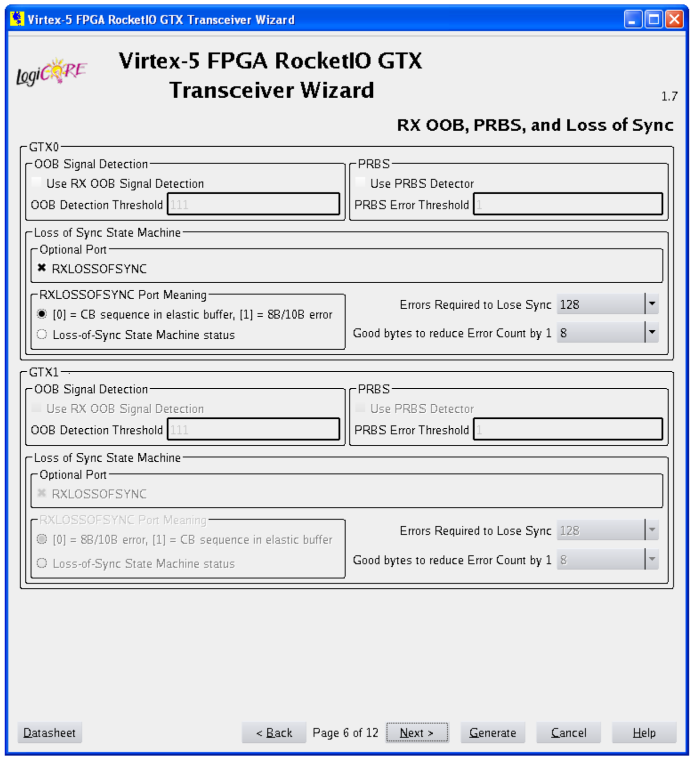

Figure 3-10: **RocketIO GTX Wizard Page 6**

Table 3-15 shows the OOB signal detection options. Table 3-16, page 32 details the PRBS settings. The Loss of Sync State Machine settings are described in Table 3-17, page 32.

Table 3-15: **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-5 Family Overview* [Ref 1] for more information about the OOB Detection Threshold levels.

Table 3-16: **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.
PRBS Error Threshold	Specifies an integer value between 0 and 4294967295 (32-bit hex value), which represents an error count threshold. When the number of errors detected exceeds the specified threshold an error signal is asserted. This option is not available if the **Use PRBS Detector option** is not selected.

Table 3-17: **Loss of Sync State Machine**

<table>
<tr><th colspan="2">Option</th><th>Description</th></tr>
<tr><td colspan="2">RXLOSSOFSYNC Optional Port</td><td>Two-bit multi-purpose status port. The meaning of the bits is determined by the settings below.</td></tr>
<tr><td rowspan="2">RXLOSSOFSYNC Port Meaning</td><td>[0] = CB Sequence in Elastic Buffer
[1] = 8B/10B Error</td><td>Bit 0 of the RXLOSSOFSYNC status port indicates a Channel Bonding sequence is present in the receive elastic buffer.
Bit 1 indicates the detection of an 8B/10B coding error.</td></tr>
<tr><td>Loss of Sync State Machine Status</td><td>Bit 0 of the RXLOSSOFSYNC status port indicates sync state is active due to channel bonding or realignment. Bit 1 indicates sync lost due to invalid characters or reset.</td></tr>
<tr><td colspan="2">Errors Required to Lose Sync</td><td>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.</td></tr>
<tr><td colspan="2">Good Bytes to Reduce Error Count by 1</td><td>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.</td></tr>
</table>

RX Comma Alignment

Page 7 of the Wizard (Figure 3-11) allows configuration of the RX comma detection and alignment logic. The settings are detailed in Table 3-18, page 34.

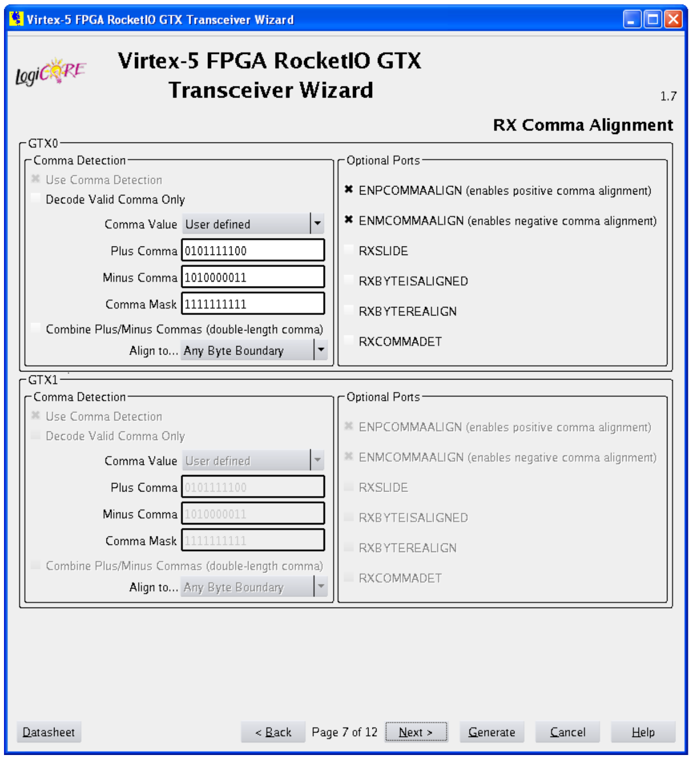

Figure 3-11: **RocketIO GTX Wizard Page 7**

Table 3-18: **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 several standard comma patterns or **User Defined** to enter a custom pattern. The XAUI example is **User Defined**.
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`.
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`.
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 (`0001111111`).
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.
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. This option is available only for 16 and 20 bit RX data interfaces.
Optional Ports	ENPCOMMAALIGN	Active-High signal which enables the byte boundary alignment process when Plus Comma pattern is detected.
	ENMCOMMAALIGN	Active-High signal which enables the byte boundary alignment process when 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.

Channel Bonding, Clock Correction

Page 8 of the Wizard (Figure 3-12) enables Channel Bonding and Clock Correction as detailed in Table 3-19.

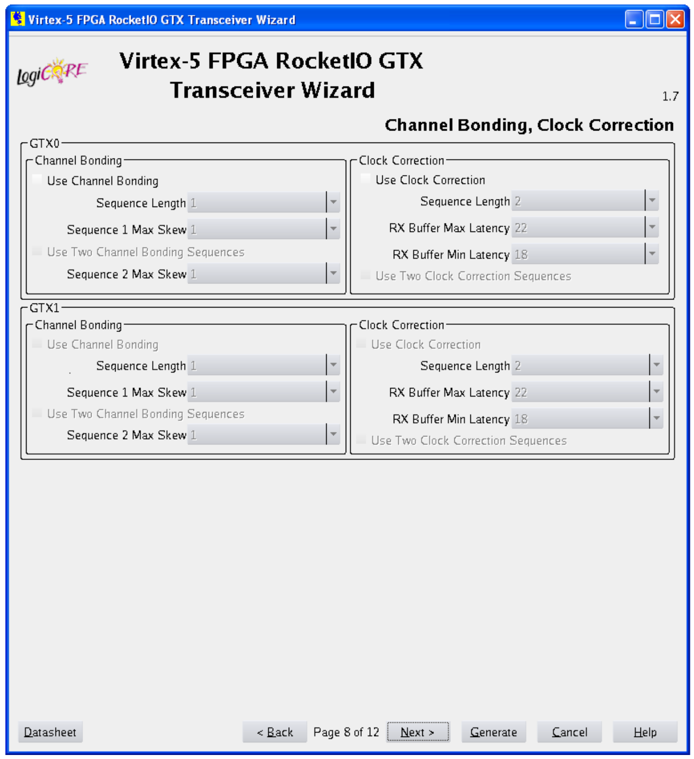

Figure 3-12: **RocketIO GTX Wizard Page 8**

Table 3-19: **Channel Bonding and Clock Correction**

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 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.

Table 3-19: **Channel Bonding and Clock Correction**

Option	Description
Sequence 2 Max Skew	Same as Sequence 1 Max Skew.
Use Clock Correction	Enables receiver clock correction logic using unique character sequences. When recognized, these sequences allow for adding or deleting characters in the receive buffer to prevent buffer underflow/overflow due to small differences in the transmit/receive clock frequencies.
Sequence Length	Select from the drop down list the number of characters (subsequences) in the unique clock correction sequence. The XAUI example uses **1**.
RX Buffer Max Latency	Select from the drop down list the maximum number of characters to permit in the receive buffer before clock correction attempts to delete incoming clock correction sequences. Also determines the maximum latency of the receive buffer in RXUSRCLK cycles. The XAUI example uses **20**.
RX Buffer Min Latency	Select from the drop down list the minimum number of characters to permit in the receive buffer before clock correction attempts to add extra clock correction sequences to the receive buffer. Also determines the minimum latency of the receive buffer in RXUSRCLK cycles. The XAUI example uses **18**.
Use Two Clock Correction Sequences	Activates the optional second Clock Correction sequence. Detection of either sequence triggers clock correction.

Channel Bonding Sequence

Page 9 (Figure 3-13) defines the Channel Bonding sequence(s). Similarly, Page 10 (Figure 3-14, page 38) defines the Clock Correction sequence(s). Table 3-20, page 37 describes their use.

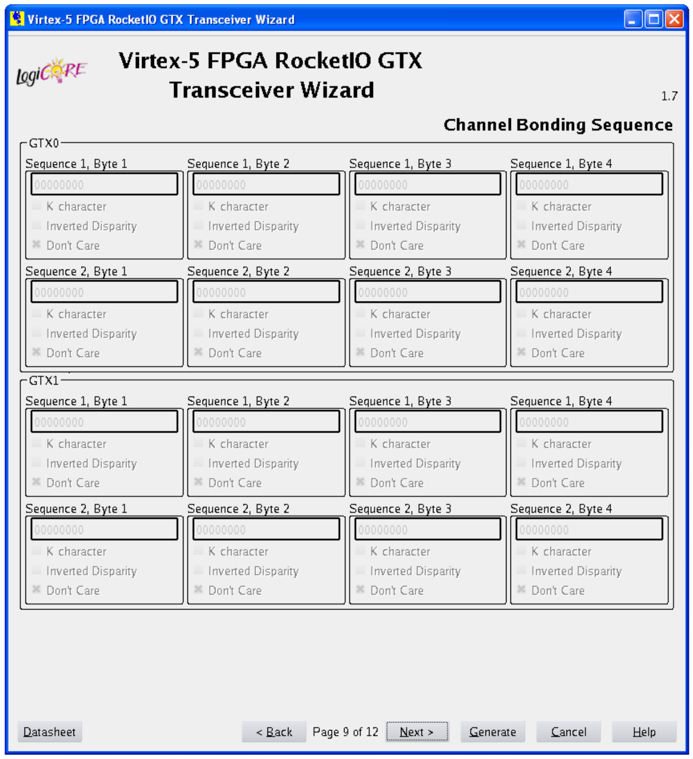

Figure 3-13: **RocketIO GTX Wizard Page 9**

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

Page 10 (Figure 3-14) defines the Clock Correction sequence. See Table 3-20, page 37 for details.

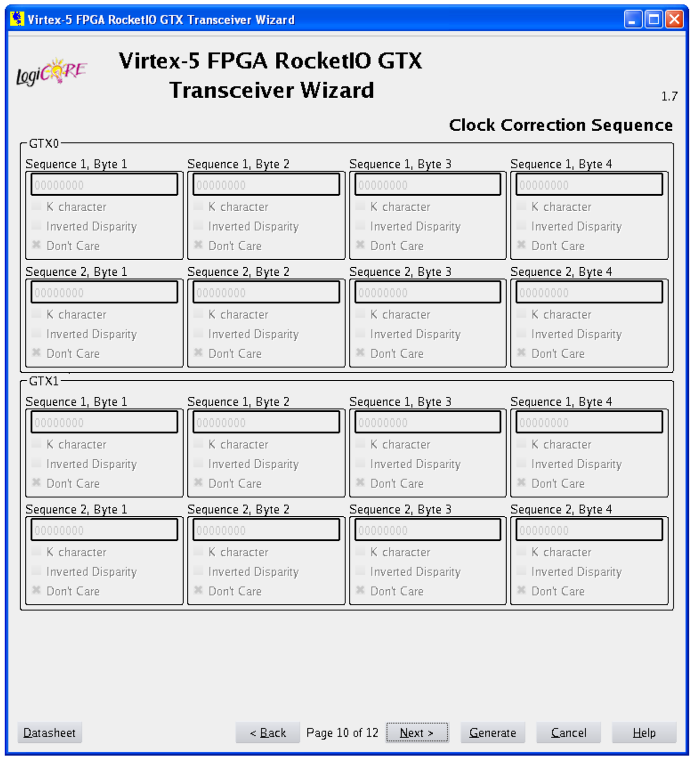

Figure 3-14: **RocketIO GTX Wizard Page 10**

RX PCI Express, SATA Features

Page 11 (Figure 3-15) configures the receiver for PCI Express® and Serial ATA (SATA) features as detailed in Table 3-21, page 40 and Table 3-22, page 41.

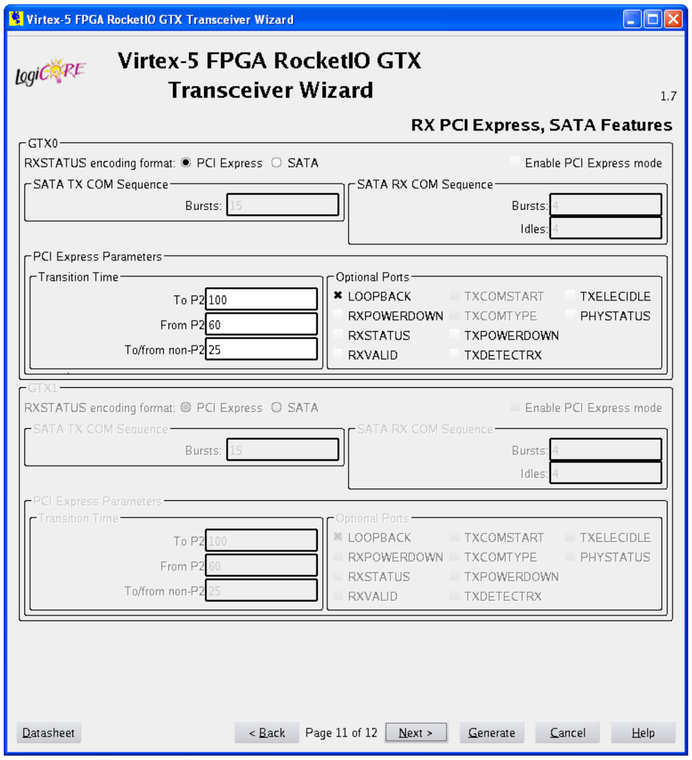

Figure 3-15: **RocketIO GTX Wizard Page 11**

Table 3-21: **Receiver Serial ATA Options**

Option		Description
RXSTATUS Encoding Format	PCI Express	Default setting. The **RXSTATUS** optional port presents status information for the PIPE interface. See the *Virtex-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 2] for more details.
	SATA	The **RXSTATUS** optional port presents codes for the SATA COM sequence status.
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. This option is not available if **RXSTATUS encoding format** is set to **SATA**
SATA TX COM Sequence	Bursts	Integer value between 0 and 15 indicating the number of busts to define a TX COM sequence. This option is not available if **RXSTATUS encoding format** is set to **PCI Express**.
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. This value defaults to 3 per the SATA specification. This option is not available if **RXSTATUS encoding format** is set to **PCI Express**.

Table 3-22: **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-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 2] 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-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 2] 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-5 FPGA RocketIO GTX Transceiver User Guide* [Ref 2] 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**. This option is not available if **RXSTATUS encoding format** is set to **SATA**.
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.

Table 3-22: **PCI Express Parameters *(Cont'd)***

Option		Description
Optional Ports (cont'd)	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	Active-High, PCI Express receive detect support signal. Indicates completion of several PHY functions.

Summary Page

Page 12 of the Wizard (Figure 3-16) is a summary of the selected configuration parameters. After reviewing the settings, click **Generate** to exit and generate the wrapper.

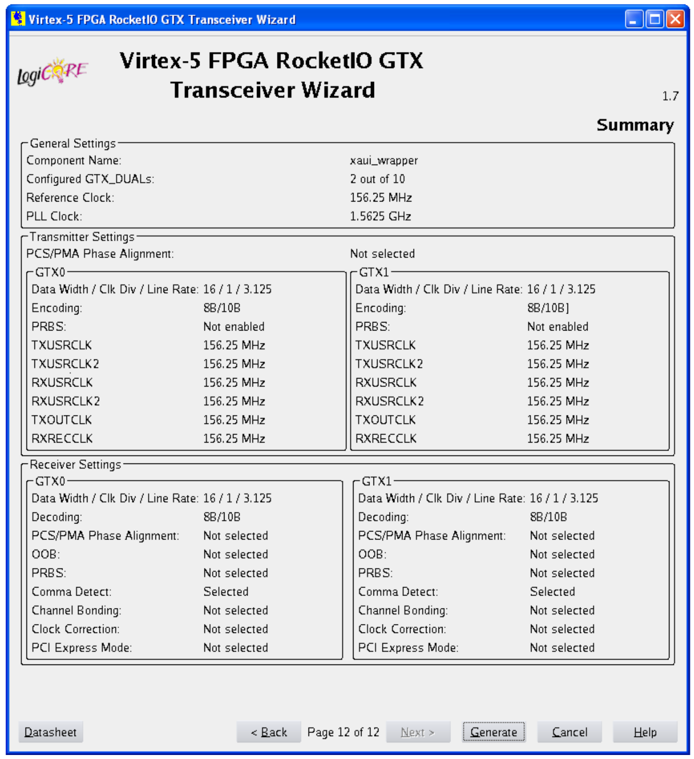

Figure 3-16: **RocketIO GTX Wizard Page 12**

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 4, "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-5 FPGA Virtex-5 FPGA RocketIO 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 3-23. See the *Synthesis and Simulation Design Guide* for ISE 12.1, available in the ISE® Software Documentation [Ref 4], for instructions on how to compile ISE simulation libraries.

Table 3-23: **Required ModelSim Simulation Libraries**

HDL	Library	Source Directories
Verilog	UNISIMS_VER	`<Xilinx dir>/verilog/src/unisims` `<Xilinx dir>/smartmodel/<OS>/wrappers/mtiverilog`
VHDL	UNISIM	`<Xilinx dir>/vhdl/src/unisims` `<Xilinx dir>/smartmodel/<OS>/wrappers/mtivhdl`

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

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>/rocketio_wrapper/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 3-24: **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>/<wrapper_name>/simulation/functional`

2. Launch the simulation script:

   prompt> `xilperl run_isim.pl`

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

Chapter 4

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®-5 FPGA RocketIO™ 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 project files.

Table 4-1: **Project Directory**

Name	Description
`<component_name>.v[hd]`	Main GTX transceiver wrapper. Instantiates individual GTX tile wrappers. For use in the target design.
`<component_name>_tile.v[hd]`	Individual GTX_DUAL transceiver wrapper to be instantiated in the main GTX transceiver wrapper. Instantiates the selected GTX_DUAL transceivers with settings for the selected protocol.
`<component_name>.[veo | vho]`	GTX Wrapper files instantiation templates. Includes templates for the GTX Wrapper module, the IBUFDS, and essential GTX support modules (such as TX_SYNC).
`<component_name>.xco`	Log file from CORE Generator describing which options were used to generate the GTX Wrapper. An XCO file is generated by CORE Generator 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.

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 4-2: **GTX Wrapper Component Name**

Name	Description
<project_dir>/<component_name>	
`v5_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 4-3: **Doc Directory**

Name	Description
<project_dir>/<component_name>/doc	
`v5_gtxwizard_ds601.pdf`	*Virtex-5 FPGA RocketIO GTX Transceiver Wizard Data Sheet*
`v5_gtxwizard_gsg204.pdf`	*LogiCORE IP Virtex-5 FPGA RocketIO GTX Transceiver Wizard Getting Started Guide*

Back to Top

<component name>/example design

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

Table 4-4: **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-5 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 module instantiation. See Figure 3-1, page 15.

Back to Top

<component name>/implement

The implement directory contains the core implementation script files.

Table 4-5: **Implement Directory**

Name	Description
<project_dir>/<component_name>/implement	
`chipscope_project.cpj`	Chipscope project file.
`data_vio.ngc`	ChipScope VIO netlist.
`icon.ngc`	ChipScope ICON netlist.
`ila.ngc`	ChipScope ILA 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.
`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 4-6: **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 4-7: **Simulation Directory**

Name	Description
<project_dir>/<component_name>/simulation	
`demo_tb.v`	Test bench to simulate the provided example design. See **"Simulating the Example Design," page 45.**

Back to Top

/simulation/functional

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

Table 4-8: **Functional Directory**

Name	Description
<project_dir>/<component_name>/simulation/functional	
`run_isim.pl`	ISim simulation script.
`simulate_mti.do`	ModelSim simulation script.
`simulate_ncsim.sh`	Linux script for running simulation using Cadence IUS.
`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 Hierarchy

The hierarchy for the design used in this example is as follows:

```
example_tb
|___example_mgt_top
    |___mgt_userclk_source_pll
    |___ibufds
    |___frame_gen
    |___frame_check
    |___tx_sync
    |___rocketio_wrapper
        |___rocketio_wrapper_tile
            |___gtx_dual
```

Appendix A

References

Documents Specific to Virtex®-5 FPGAs

1. DS100: *Virtex-5 Family Overview*

Documents Specific to RocketIO™ Transceivers

2. UG198: *Virtex-5 FPGA RocketIO GTX Transceiver User Guide*
3. DS601: *Virtex-5 FPGA RocketIO GTX Transceiver Wizard Data Sheet*

Xilinx ISE® Tools and Solutions

4. ISE Software Manuals