LogiCORE™ IP Spartan®-6 FPGA GTP Transceiver Wizard v1.8

Getting Started Guide

UG546 (v1.8) December 14, 2010

XILINX®

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	Update version of tools and Wizard.
12/02/09	1.4	Update version of tools and Wizard. Added Recommended Design Experience, page 10 and replaced Appendix A: References with Related Xilinx Documents, page 10.
04/19/10	1.5	Wizard 1.5 release. Update version of tools and Wizard. Edits to System Requirements, Table 3-17, and Table 5-8.
07/23/10	1.6	Update version of tools and Wizard.
09/21/10	1.7	Update version of tools and Wizard v1.7.
12/14/10	1.8	Update version of tools and Wizard v1.8. Added Advanced Clocking feature to Table 3-1, page 20.

Table of Contents

Revision History 2

Preface: About This Guide

Guide Contents 5
Additional Resources 5
Conventions 5
Typographical 6
Online Document 7

Chapter 1: Introduction

About the Wizard 9
Recommended Design Experience 10
Related Xilinx Documents 10
Additional Wizard Resources 10
Technical Support 10
Feedback 10
Spartan-6 FPGA GTP Transceiver Wizard 10
Document 11

Chapter 2: Installation and Licensing

System Requirements 13
Before You Begin 13
Installing the Wizard 14
Verifying Your Installation 14

Chapter 3: Running the Wizard

Overview 15
Setting Up the Project 16
Creating a Directory 16
Setting the Project Options 17
Generating the Core 18
GTP Placement and Clocking 19
Line Rate and Protocol Template 21
8B/10B Optional Ports 24
Synchronization and Clocking 25
RX Comma Alignment 27
Preemphasis, Termination, and Equalization 29
RX OOB, PRBS, and Loss of Sync 31
RX PCI Express, SATA Features 33
Channel Bonding, Clock Correction 36
Clock Correction Sequence 39
Summary 40

Chapter 4: Quick Start Example Design

Overview 41
Implementing the Example Design 41
Functional Simulation of the Example Design 42
Using ModelSim 42
Using the ISE Simulator 43
Using ChipScope Pro Cores with the Spartan-6 FPGA GTP Transceiver Wizard Core 43

Chapter 5: Detailed Example Design

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

About This Guide

This getting started guide describes the function and operation of the LogiCORE™ IP GTP Transceiver Wizard for the Spartan®-6 LXT family.

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, Installation and Licensing provides information about installing and licensing the Spartan-6 FPGA GTP Transceiver Wizard.
- Chapter 3, Running the Wizard provides an overview of the Spartan-6 FPGA GTP Transceiver Wizard, and a step-by-step tutorial to generate a sample GTP transceiver wrapper with the Xilinx® CORE Generator™ tool.
- Chapter 4, Quick Start Example Design introduces the example design that is included with the GTP Transceiver wrappers. The example design demonstrates how to use the wrappers and demonstrates some of the key features of the GTP 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

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;*
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 Spartan®-6 FPGA GTP Transceiver Wizard core and provides related information, including additional resources, technical support, and submitting feedback to Xilinx.

The Spartan-6 FPGA GTP 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 GTP transceivers for custom applications (Figure 1-1).

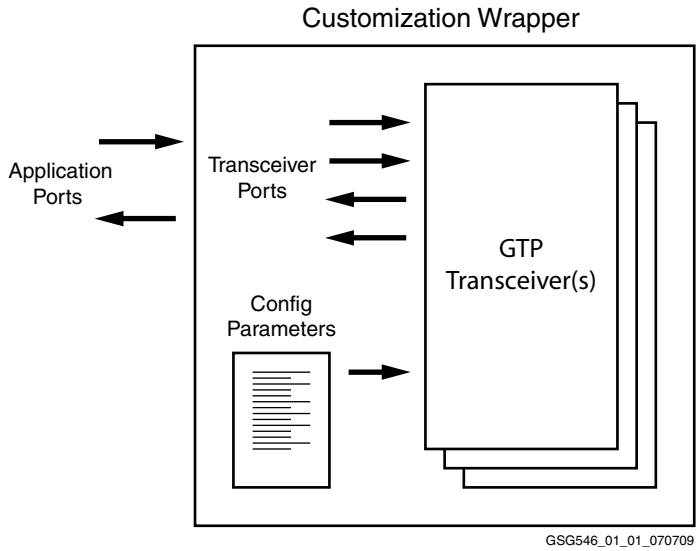

Figure 1-1: **GTP Transceiver Wizard Wrapper**

About the Wizard

The Spartan-6 FPGA GTP 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.

Recommended Design Experience

Although the Spartan-6 FPGA GTP Transceiver Wizard 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 Spartan-6 FPGA GTP Transceiver Wizard, users should be familiar with the following:

1. DS160: *Spartan-6 Family Overview*
2. UG386: *Spartan-6 FPGA GTP Transceivers User Guide*
3. ISE® software documentation: www.xilinx.com/ise

Additional Wizard Resources

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

- DS713: *Spartan-6 FPGA GTP Transceiver Wizard v1.8 Data Sheet*
- Spartan-6 FPGA GTP 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 Spartan-6 FPGA GTP Transceiver Wizard.

Xilinx provides technical support for use of this product as described in the *LogiCORE IP Spartan-6 FPGA GTP Transceiver Wizard v1.8 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 Spartan-6 FPGA GTP Transceiver Wizard and the accompanying documentation.

Spartan-6 FPGA GTP Transceiver Wizard

For comments or suggestions about the Spartan-6 FPGA GTP 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).

XILINX®

Chapter 2

Installation and Licensing

This chapter provides instructions for installing the Spartan®-6 FPGA GTP 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) desktop and server v10.1 32-bit/64-bit

Tools

- ISE® software v12.4
- Mentor Graphics ModelSim 6.5c
- Cadence Incisive Enterprise Simulator (IES) 9.2
- Synopsys VCS and VCS MX 2009.12
- Synopsys Synplify Pro D-2010.03

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 12.4 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 12.4 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 Spartan-6 FPGA GTP Transceiver Wizard is included with the ISE 12.4 software. See *ISE CORE Generator IP Updates - Installation Instructions* for details on the installation of ISE 12.4.

Verifying Your Installation

Use the following procedure to verify that you have successfully installed the Spartan-6 FPGA GTP 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.

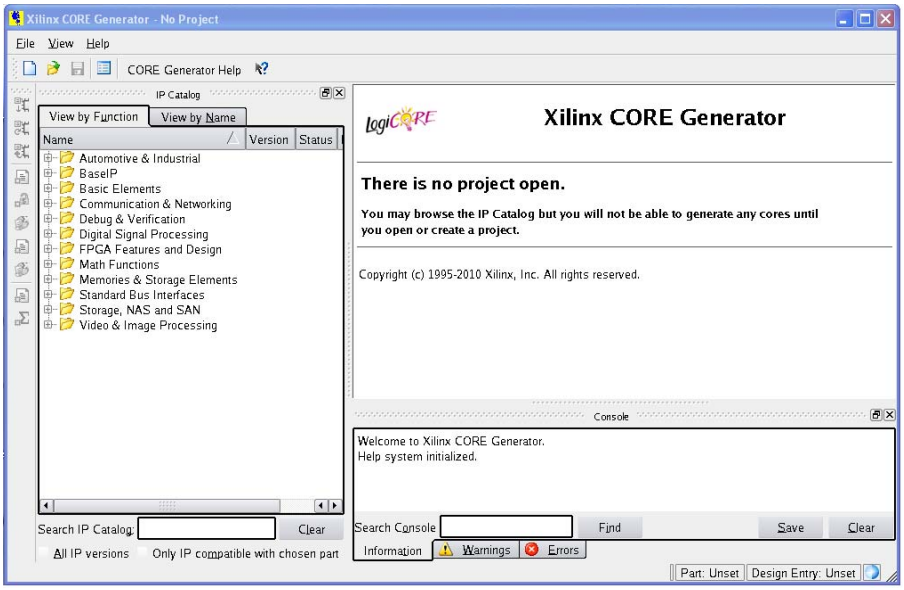

Figure 2-1: **CORE Generator Window**

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 Spartan-6 FPGA GTP Transceiver Wizard 1.8 appears at the following location in the Functional Categories list:
   `/FPGA Features and Design/IO Interfaces`

Chapter 3

Running the Wizard

Overview

This section provides a step-by-step procedure for generating a Spartan®-6 FPGA GTP 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 GTP transceivers for use in a PCI EXPRESS® application. Guidelines are also given for incorporating the wrapper in a design and for the expected behavior in operation.

The PCI EXPRESS example consists of the following components:

- A single GTP transceiver wrapper implementing a four-lane PCI EXPRESS interface using four GTP transceivers
- A demonstration test bench to drive the example design in simulation
- An example design providing clock signals and connecting an instance of the PCI EXPRESS wrapper with modules to drive and monitor the wrapper in hardware, including optional ChipScope™ Pro software support
- Scripts to synthesize and simulate the example design

The Spartan-6 FPGA GTP Transceiver Wizard example design has been tested with XST 12.4 for synthesis and ModelSim 6.5c for simulation.

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

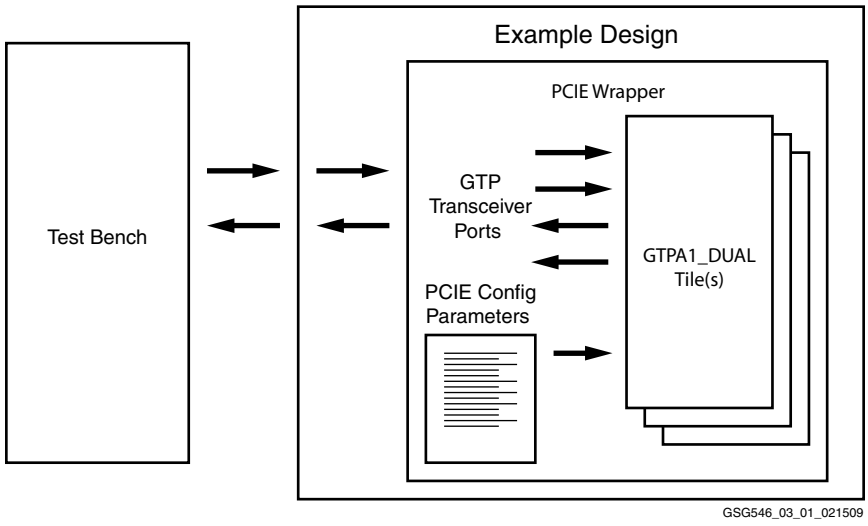

Figure 3-1: **Example Design**

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/pcie_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.

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

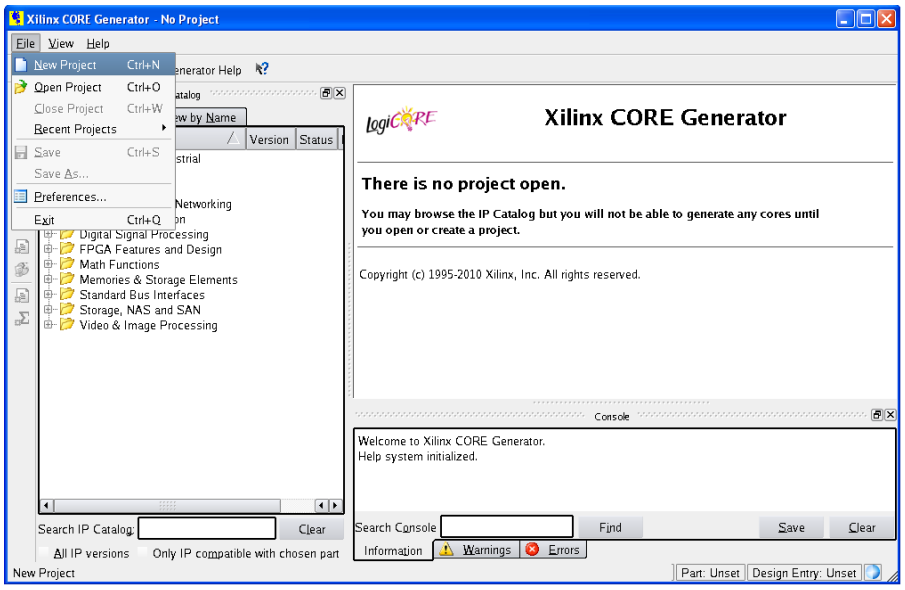

Figure 3-2: **Starting a New Project**

Setting the Project Options

Set the project options using the following steps:

1. Click **Part** in the option tree.
2. Select **Spartan6** from the Family list.
3. Select a Spartan-6 FPGA from the Device list which supports GTP transceivers.
4. Select an appropriate package from the Package list. This example uses the XC6SLX45T device (see Figure 3-3).

   Note: If an unsupported silicon family is selected, the Spartan-6 FPGA GTP Transceiver Wizard remains light grey in the taxonomy tree and cannot be customized. Only Spartan-6 FPGA devices containing GTPA1_DUAL transceiver primitives are supported by the Wizard. See the *Spartan-6 Family Overview* for a list of devices containing GTPA1_DUAL transceiver primitives.
5. Click **Generation** in the option tree and select either Verilog or VHDL as the output language.
6. Click **OK**.

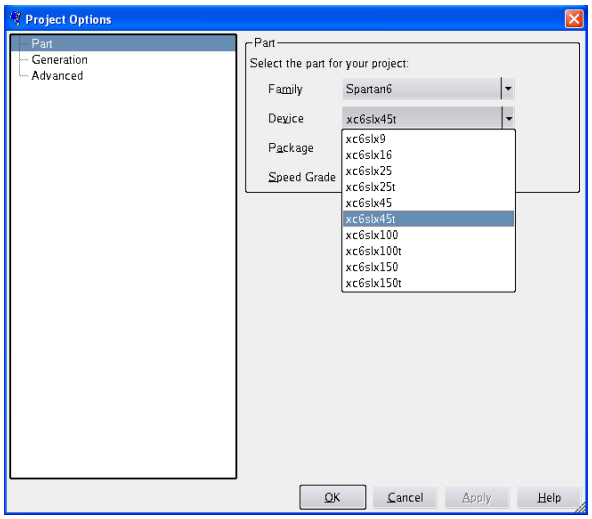

Figure 3-3: **Target Architecture Setting**

Generating the Core

This section provides instructions for generating an example GTP 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 Implementing the Example Design, page 41.

1. Locate the Spartan-6 FPGA GTP Transceiver Wizard 1.8 in the taxonomy tree under: `/FPGA Features & Design/IO Interfaces`. (See Figure 3-4)
2. Double-click **Spartan-6 FPGA GTP Transceiver Wizard 1.8** to launch the Wizard.

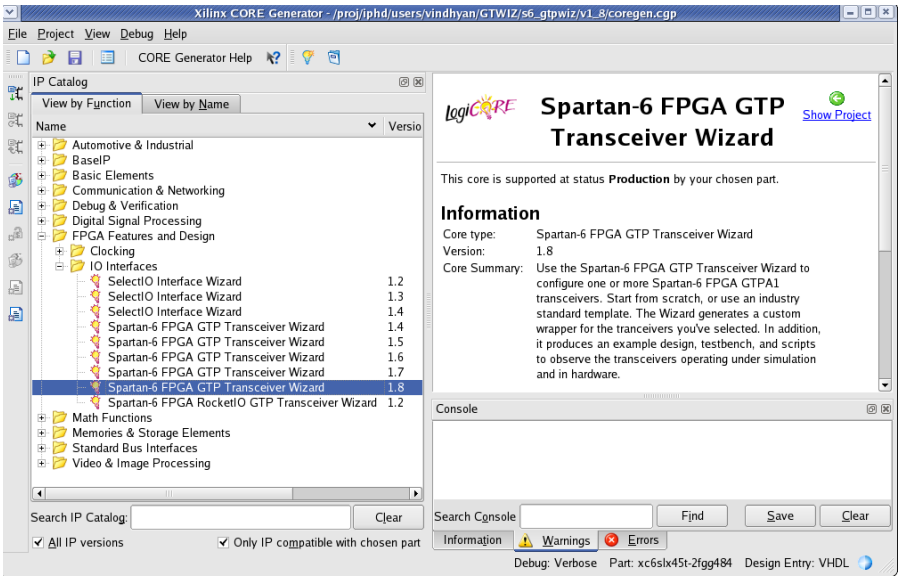

Figure 3-4: **Locating the GTP Transceiver Wizard**

GTP Placement and Clocking

The GTP Placement and Clocking screen (page 1) of the Wizard (Figure 3-5) allows you to select the component name and determine the placement of the GTP transceivers and the reference clock source.

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

The number of GTPA1_DUAL transceiver primitives appearing on this page depends on the selected target device and package. The PCI EXPRESS example design uses four GTP transceivers (two GTPA1_DUAL primitives). Table 3-1 describes the GTP transceiver selection and Reference Clock options.

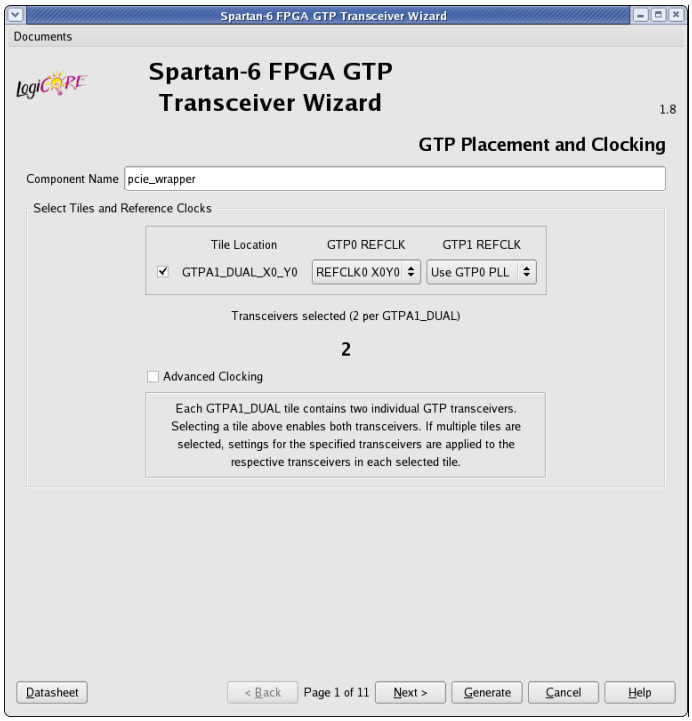

Figure 3-5: **GTP Placement and Clocking - Page 1**

Table 3-1: **Select Transceiver and Reference Clocks**

Option	Description
Tile Location	Select the individual GTPA1_DUAL transceiver pairs by location to be used in the target design. The PCI EXPRESS example requires one transceiver pair.
GTP0 REFCLK	Determines the source for the reference clock signal provided to the GTP0 transceiver in the selected GTPA1_DUAL primitive (see Table 3-2, page 20). Differential clock signal input pin pairs are provided for each GTPA1_DUAL. Individual transceivers have access to the reference clock signals for the two horizontally adjacent GTPA1_DUAL primitives allowing two primitives to share a single reference clock signal. The PCI EXPRESS example uses the **REFCLK X0Y0** signal from the upper pair of selected primitives.
GTP1 REFCLK	Same as above for the GTP1 transceiver.
Advanced Clocking	Use this check box to bring out all possible reference clock ports to the generated wrapper.

Table 3-2: **Reference Clock Source Options**

Option	Description
GREFCLK	Reference clock driven by internal fabric. Lowest performance option.
REFCLK_X0Y0	External GTPA1_DUAL reference clock signal local to first upper transceiver pair.
REFCLK_X1Y0	External GTPA1_DUAL reference clock signal local to second upper transceiver pair.
REFCLK_X0Y1	External GTPA1_DUAL reference clock signal local to first lower transceiver pair. Not available on devices with fewer than four transceiver pairs.
REFCLK_X1Y1	External GTPA1_DUAL reference clock signal local to second lower transceiver pair. Not available on devices with fewer than four transceiver pairs.

Line Rate and Protocol Template

The Line Rate and Protocol Template screen (page 2) of the Wizard (Figure 3-6) allows you to select the line rate, reference clock frequency, encoding/decoding method, and data width. In addition, this page specifies protocol templates.

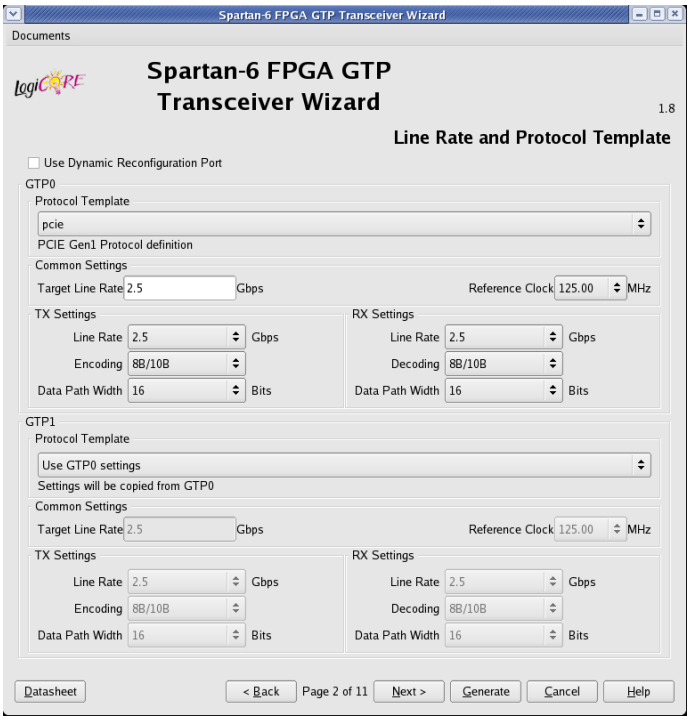

Figure 3-6: **Line Rates and Protocol Template - Page 2**

1. Place a check next to the **Use Dynamic Reconfiguration Port** option if you wish to bring this signal out for use by the application.

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

The remaining options are divided into GTP0 and GTP1 groups with identical parameters. These apply to the two GTP transceivers present in each GTPA1_DUAL primitive. The remaining discussion in this chapter describes only the GTP0 portion.

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. For GTP1 only, select **Use GTP0 settings** to automatically copy the settings from GTP1. The PCI EXPRESS example uses the **pcie** protocol template. Because both transceivers are configured identically, the protocol template option for GTP1 is set to **Use GTP0 settings**.

Table 3-3 shows the options for the Common Settings. These options establish the shared PMA PLL settings for both GTP transceivers in each pair.

Table 3-3: **Common Settings**

Option	Description
Target Line Rate	Line rate in Gb/s desired for the target design. The PCI EXPRESS example uses 2.5 Gbps.
Reference Clock	Select from the list the optimal reference clock frequency to be provided by the application. The PCI EXPRESS example uses 125.0 MHz.

3. Use the following tables to determine the line rate, encoding, decoding, and reference clock settings available on this page.

Table 3-4 details the TX Settings options.

Table 3-4: **TX Settings**

Option		Description
Line Rate		Set to the desired target line rate in Gbps. Can be independent of the receive line rate. The line rate has option of selecting No_TX. This option disables all the TX ports in the wrapper when selected. The PCI EXPRESS example uses 2.5 Gbps.
Encoding	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.
Data Path Width	8	Sets both the internal transmitter data path and the transmitter application interface data path width to a single 8-bit byte.
	10	Sets the internal transmitter data path width to a single 10-bit byte. If 8B/10B encoding is selected, the transmitter application interface data path width will be set to 8 bits. If 8B/10 encoding is not selected, the transmitter application interface data path width is set to 10 bits.
	16	Sets the internal transmitter data path width to a single 8-bit byte. Sets the transmitter application interface data path width to two 8-bit bytes (16 bits).
	20	Sets the internal transmitter data path width to a single 10-bit byte. 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 be set to two 10-bit bytes (20 bits).
	32	Sets the internal transmitter data path width to a single 8-bit byte. Sets the transmitter application interface data path width to four 8-bit bytes (32 bits).
	40	Sets the internal transmitter data path width to a single 10-bit byte. 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).

Table 3-5 details the RX Settings options.

Table 3-5: **RX Settings**

Option		Description
Line Rate		Set to the desired target line rate in Gbps. Can be independent of the transmit line rate. The line rate has option of selecting No_RX. This option disables all the RX ports in the wrapper when selected. The PCI EXPRESS example uses 2.5 Gbps.
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.
Data Path Width	8	Sets both the internal receiver data path and the receiver application interface data path width to a single 8-bit byte.
	10	Sets the internal receiver data path width to a single 10-bit byte. If 8B/10B encoding is selected, the receiver application interface data path width will be set to 8 bits.
	16	Sets the internal receiver data path width to a single 8-bit byte. Sets the receiver application interface data path width to two 8-bit bytes (16 bits).
	20	Sets the internal receiver data path width to a single 10-bit byte. 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 be set to two 10-bit bytes (20 bits).
	32	Sets the internal receiver data path width to a single 8-bit byte. Sets the receiver application interface data path width to four 8-bit bytes (32 bits).
	40	Sets the internal receiver data path width to a single 10-bit byte. 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).

8B/10B Optional Ports

The 8B/10B Optional Ports screen (page 3) of the Wizard (Figure 3-7) provides selections for 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 3-6 details the available TX and RX 8B/10B optional ports.

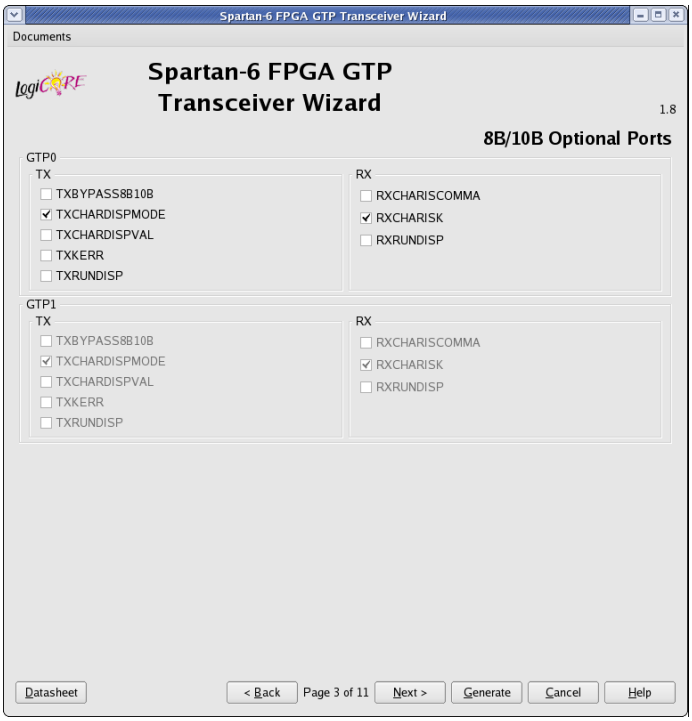

Figure 3-7: **8B/10B Optional Ports - Page 3**

Table 3-6: **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.
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.

Synchronization and Clocking

The Synchronization and Clocking screen (page 4) of the Wizard (Figure 3-8) provides settings for latency, buffering, and clocking of the transmitter and receiver. The RX comma alignment settings are also provided. Table 3-7 lists the source signal options and Table 3-9 lists the optional ports.

The **Enable TX buffer** setting controls whether the TX buffer is enabled or bypassed. See the *Spartan-6 FPGA GTP Transceivers User Guide* for details on this setting.

The PCI EXPRESS example uses the TX buffer.

The **Enable RX buffer** setting controls whether the RX buffer is enabled or bypassed. If the RX buffer is deselected, then the RX Phase Alignment circuit is enabled.

The PCI EXPRESS example does not use the RX Phase Alignment circuit.

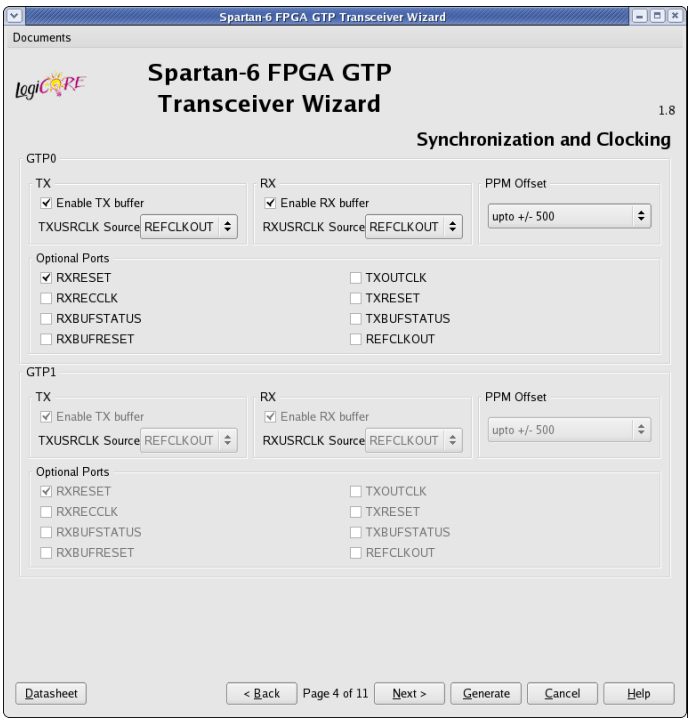

Figure 3-8: **Synchronization and Clocking - Page 4**

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

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

Option		Description
TX	TXOUTCLK	TXUSRCLK is driven by TXOUTCLK. 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.
	Enable External TXUSRCLK	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.
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.
	Enable External RXUSRCLK	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.

Table 3-8 shows the available PPM Offset settings. The PPM Offset setting optimizes the receiver CDR logic for the desired PPM tolerance range.

Table 3-8: **PPM Offset**

Option	Description
0 (Synchronous)	Use with synchronous applications (zero tolerance).
Up to ± 100	For applications where clock tolerance is below 100 PPM.
Up to ± 500	For applications where clock tolerance is below 500 PPM.

Table 3-9 shows the optional ports available for synchronization and clocking.

Table 3-9: **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.

Table 3-9: **Optional Ports *(Cont'd)***

Option	Description
TXOUTCLK	Parallel clock signal generated by the GTP 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.
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.
REFCLKOUT	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.

RX Comma Alignment

The RX Comma Alignment screen (page 5) of the Wizard (Figure 3-9) allows you to configure the RX comma detection and alignment logic. The settings are detailed in Table 3-10, page 28.

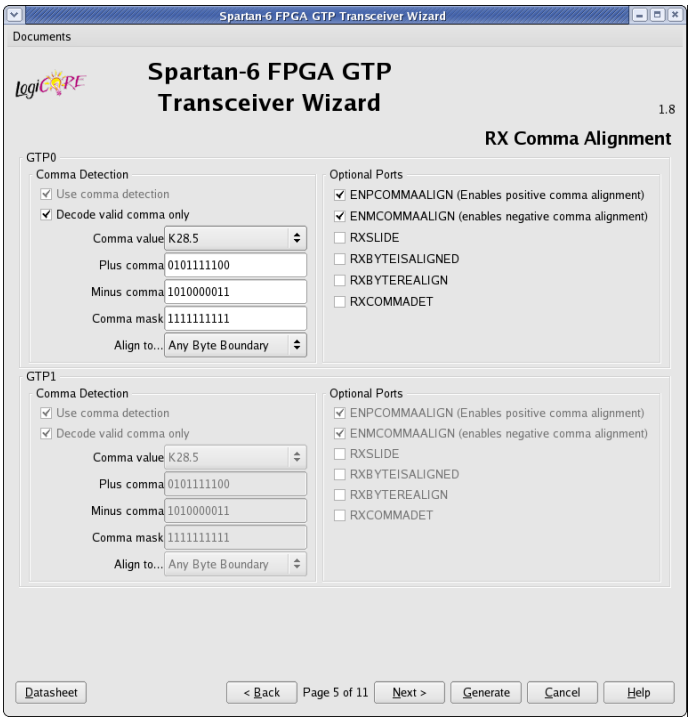

Figure 3-9: **RX Comma Alignment - Page 5**

Table 3-10: **Comma Detection**

Option		Description
Use Comma Detection		Enables receive comma detection. Used to identify special protocol comma and 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 PCI EXPRESS example is 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 PCI EXPRESS 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 PCI EXPRESS 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 PCI EXPRESS example matches all ten bits (`1111111111`).
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.

Table 3-11 shows the optional ports available on this page.

Table 3-11: **Optional Ports**

Option	Description
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.

Preemphasis, Termination, and Equalization

The Preemphasis, Termination, and Equalization screen (page 6) of the Wizard (Figure 3-10) allows you to set the preemphasis, termination, and equalization options.

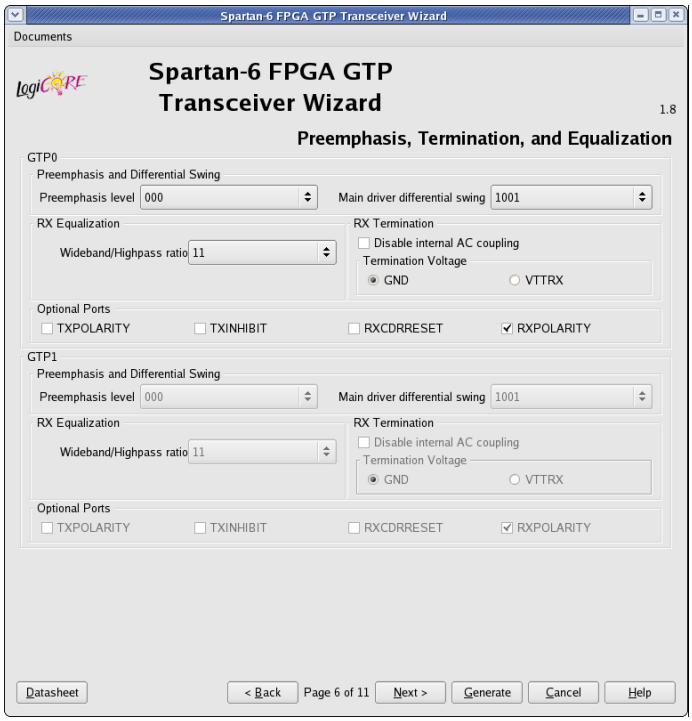

Figure 3-10: **Preemphasis, Termination and Equalization - Page 6**

Table 3-12 details the preemphasis and differential swing settings.

Table 3-12: **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 PCI EXPRESS example uses the default setting of `000` (0%). See the *Spartan-6 FPGA GTP 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 in 100 mV steps from approximately 300 mV to 1400 mV. Can also be set to zero. Selecting **Use TXDIFFCTRL port** enables the optional TXDIFFCTRL configuration port to dynamically set the swing level. The PCI EXPRESS example uses the default setting `1001` (1110 mV). See the *Spartan-6 FPGA GTP Transceivers User Guide* for a table mapping TXDIFFCTRL value settings to differential swing levels.

Table 3-13 describes the RX equalization settings.

Table 3-13: **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 PCI EXPRESS example uses setting `11` (bypass with gain).

Table 3-14 describes the RX termination settings.

Table 3-14: **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 **VTTRX** applies an internal voltage reference source to the internal termination network. The PCI EXPRESS example uses the **GND** setting.

Table 3-15 shows the optional ports available on this page.

Table 3-15: **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.
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

The RX OOB, PRBS, and Loss of Sync screen (page 7) of the Wizard (Figure 3-11) provides configuration options for the RX out of band signal (OOB), PRBS detector, and the loss of sync state machine settings.

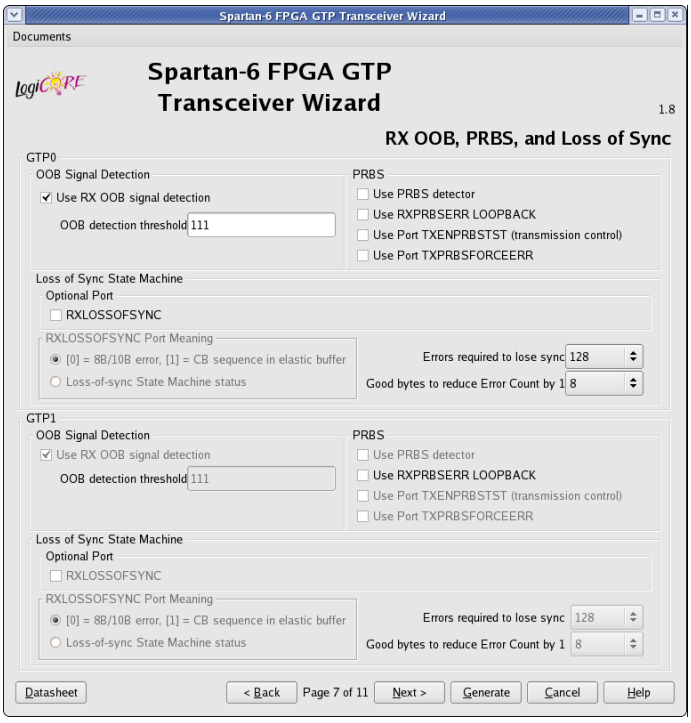

Figure 3-11: **RX OOB, PRBS, and Loss of Sync - Page 7**

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 PCI EXPRESS 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 *Spartan-6 FPGA GTP 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 RXPRBSERR LOOPBACK	Enables the PRBS loopback on the receiver side. This port allows synchronous and asynchronous jitter tolerance testing without worrying about data clock domain crossing.
Use Port TXENPRBSTST	Enables the PRBS Transmission control port. This port is used by the application to start/stop PRBS generation.
Use Port TXPRBSFORCEERR	Enables the PRBS Force Error control port. This port allows the application to insert errors into the bit stream.

The loss of sync state machine settings are described in Table 3-18.

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

Option		Description
RXLOSSOFSYNC Optional Port		Two-bit multi-purpose status port. The meaning of the bits is determined by the settings below.
RXLOSSOFSYNC Port Meaning	[0] = 8B/10B Error [1] = CB Sequence in Elastic Buffer	Bit 0 of the **RXLOSSOFSYNC** status port indicates the detection of an 8B/10B coding error. Bit 1 indicates a Channel Bonding sequence is present in the receive elastic buffer.
	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 PCI EXPRESS example uses 128.
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 PCI EXPRESS example uses 8.

RX PCI Express, SATA Features

The RX PCI Express, SATA Features screen (page 8) of the Wizard (Figure 3-12) configures the receiver for PCI EXPRESS and Serial ATA (SATA) features.

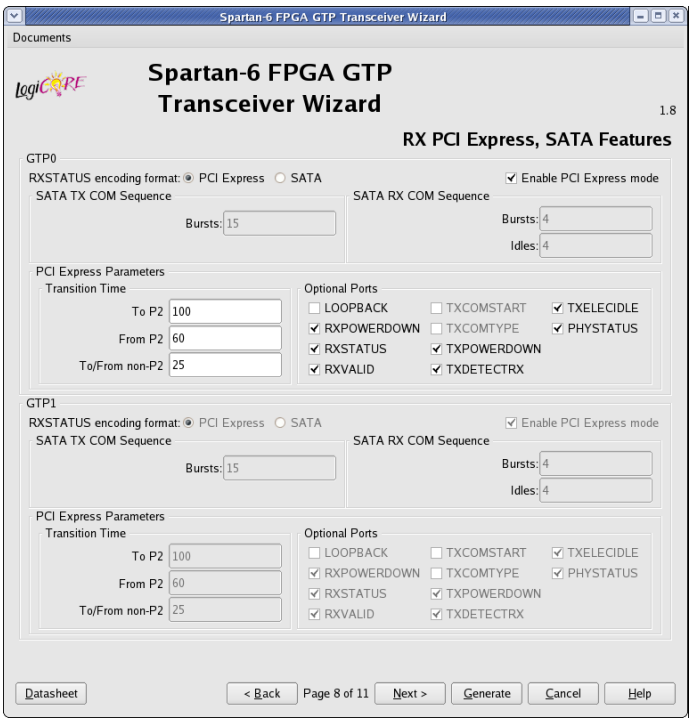

Figure 3-12: **RX PCI Express, SATA Features - Page 8**

Table 3-19, page 34 details the receiver SATA configuration options.

Table 3-19: **Receiver Serial ATA Options**

Options		Description
RXSTATUS Encoding Format	PCI Express	Default setting. The **RXSTATUS** optional port presents status information for the PIPE interface. See the *Spartan-6 FPGA GTP Transceivers User Guide* 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 3-20, page 35 details the receiver PCI Express configuration options.

Table 3-20: **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 *Spartan-6 FPGA GTP Transceivers User Guide* for details on determining the time value for each count. The PCI EXPRESS example 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 *Spartan-6 FPGA GTP Transceivers User Guide* for details on determining the time value for each count. The PCI EXPRESS example 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 *Spartan-6 FPGA GTP Transceivers User Guide* for details on determining the time value for each count. The PCI EXPRESS example 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.
	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.

Channel Bonding, Clock Correction

The Channel Bonding, Clock Correction screen (page 9) of the Wizard (Figure 3-13) defines the channel bonding and clock correction parameters. Also, the common channel bonding sequence is defined on this page.

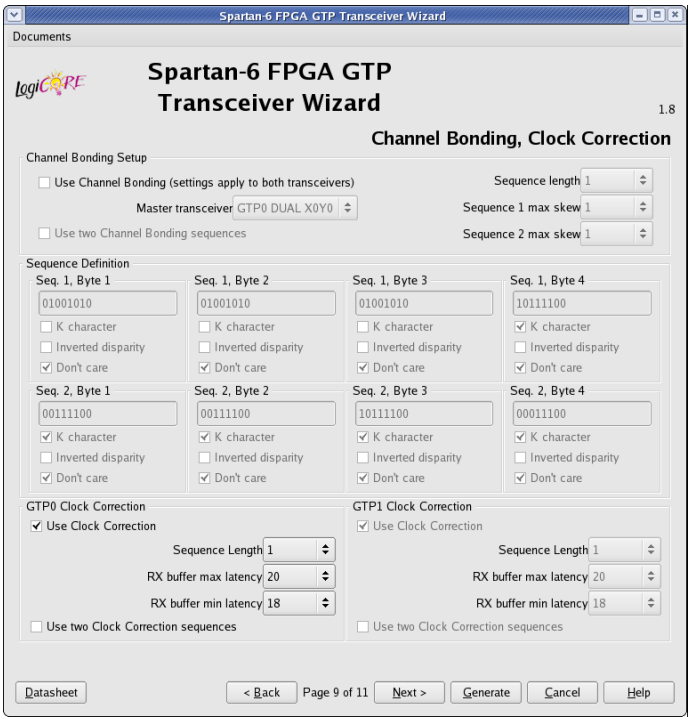

Figure 3-13: **Channel Bonding, Clock Correction - Page 9**

The Channel Bonding settings and sequence definition are common between both transceivers of each selected pair. The protocol template option for GTP1 (Wizard Page 2) must be set to **Use GTP0 settings** to enable the Channel Bonding settings.

Table 3-21, page 37 describes the common Channel Bonding settings.

Table 3-21: **Channel Bonding Setup**

Options	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.
Master Transceiver	Indicates which transceiver from the selected pair(s) will function as the Channel Bonding master.
Use Two Channel Bonding Sequences	Activates the optional second Channel Bonding sequence. Detection of either sequence triggers channel bonding.
Sequence Length	Select from the drop down list the number of characters in the unique channel bonding sequence. The PCI EXPRESS example uses **4**.
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 PCI EXPRESS example uses **7**.
Sequence 2 Max Skew	Same as Sequence 1 max skew.

Table 3-22 describes the Channel Bonding sequence definition.

Table 3-22: **Sequence Definition**

Option	Description
Byte (Symbol)	Set each symbol to match the pattern the protocol requires. The PCI EXPRESS sequence length is 8 bits. `01001010` is used for the first three symbols of sequence 1. Symbol 4, sequence 1 and symbol 3, sequence 2 are set to `10111100`. Symbols 1 and 2 of Sequence 2 are set to `00111100`. Symbol 4 of Sequence 2 is set to `00011100`.
K Character	This option is available when 8B/10B decoding is selected. When checked, 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.

Table 3-23, page 38 describes the Clock Correction settings.

Table 3-23: **Clock Correction**

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 PCI EXPRESS 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 PCI EXPRESS 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 PCI EXPRESS example uses 28.
Use Two Clock Correction Sequences	Activates the optional second Clock Correction sequence. Detection of either sequence triggers clock correction.

Clock Correction Sequence

The Clock Correction Sequence screen (page 10) of the Wizard (Figure 3-14) defines the clock correction sequence. See Table 3-24 for details.

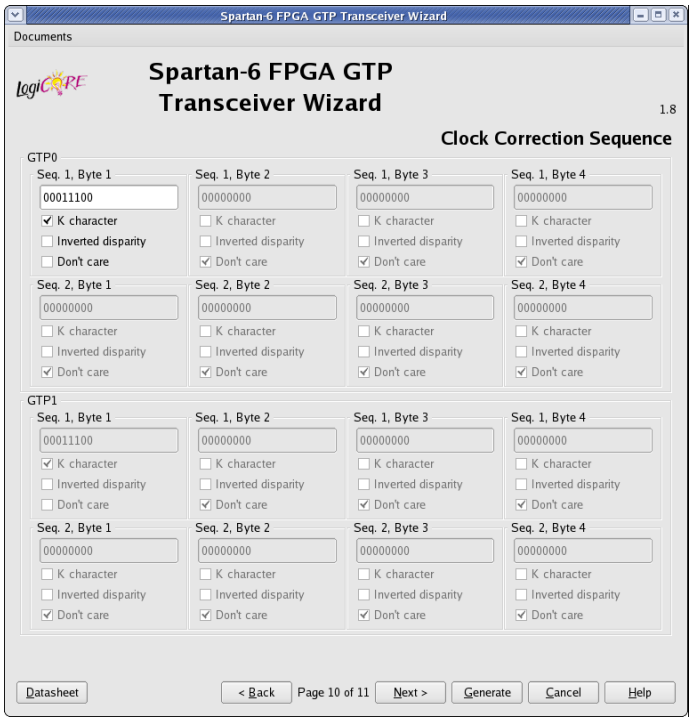

Figure 3-14: **Clock Correction Sequence - Page 10**

Table 3-24: **Clock Correction Sequence**

Option	Description
Byte (Symbol)	Set each symbol to match the pattern the protocol requires. The PCI EXPRESS sequence length is 8 bits. `00011100` is used for the first symbol of sequence 1. The remaining 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, 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.

Summary

The Summary screen (page 11) of the Wizard (Figure 3-15) provides a summary of the selected configuration parameters. After reviewing the settings, click **Generate** to exit and generate the wrapper.

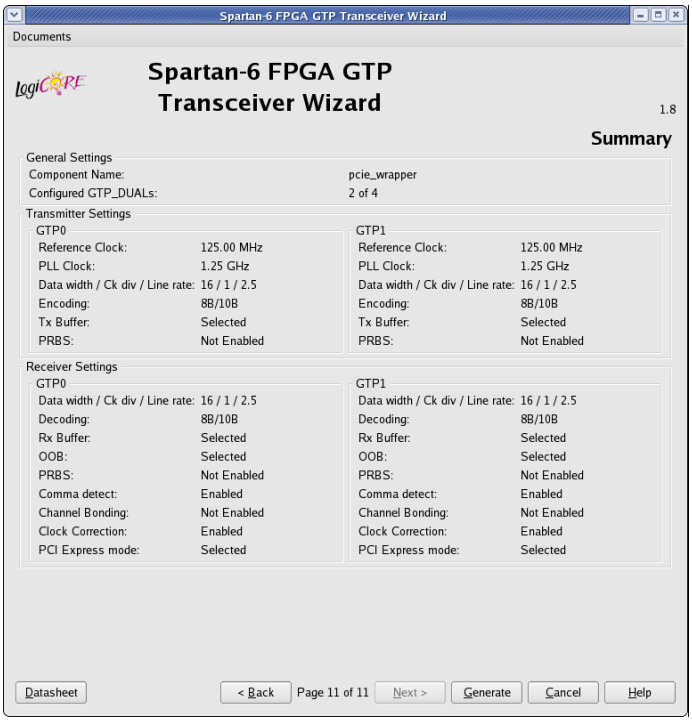

Figure 3-15: **Summary - Page 11**

XILINX®

Chapter 4

Quick Start Example Design

Overview

This chapter introduces the example design that is included with the GTP Transceiver wrappers for the Spartan®-6 LXT sub- family. The example design demonstrates how to use the wrappers and demonstrates some of the key features of the GTP 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, click **Generate** to create a directory structure under the provided Component Name. Wrapper generation proceeds and the generated output populates the appropriate subdirectories.

The directory structure for the PCI EXPRESS® 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 pcie_wrapper\implement
> implement.bat
```

For Linux

```
% cd pcie_wrapper/implement
% implement.sh
```

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.

Functional Simulation of the Example Design

Using ModelSim

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

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® 12.4, 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*>/spartan6/verilog/src/unisims <*Xilinx dir*>/spartan6/secureip/mti
VHDL	UNISIM	<*Xilinx dir*>/spartan6/vhdl/src/unisims/primitive <*Xilinx dir*>/spartan6/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:
   - For Windows
     - `simulate_isim.bat`
   - For Linux
     - `% simulate_isim.sh`

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

Using ChipScope Pro Cores with the Spartan-6 FPGA GTP Transceiver Wizard Core

The ChipScope™ Pro ICON and VIO cores aid in debugging and validating the design in board. To assist with debugging, these cores are provided with the Spartan-6 FPGA GTP Transceiver Wizard core, which is enabled by setting USE_CHIPSCOPE as 1 in the <component_name>_top_example_design file.

XILINX®

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 Spartan®-6 FPGA GTP 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 5-1: **Project Directory**

Name	Description
`<component_name>.v[hd]`	Main GTP transceiver wrapper. Instantiates individual GTP tile wrappers. For use in the target design.
`<component_name>.[veo | vho]`	GTP Wrapper files instantiation templates. Includes templates for the GTP Wrapper module, the IBUFDS, and essential GTP support modules (such as TX_SYNC).
`<component_name>.xco`	Log file from the CORE Generator tool describing which options were used to generate the GTP wrapper. An XCO file is generated by the 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>_tile.v[hd]`	Individual GTPA1_DUAL transceiver wrapper to be instantiated in the main GTP transceiver wrapper. Instantiates the selected GTPA1_DUAL 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: **GTP Wrapper Component Name**

Name	Description
<project_dir>/<component_name>	
`s6_gtpwizard_readme.txt`	Release notes for the GTP Wizard.
`<component_name>.pf`	Protocol description for the selected protocol from the GTP 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	
`s6_gtpwizard_ds713.pdf`	*Spartan-6 FPGA GTP Transceiver Wizard v1.8 Data Sheet*
`s6_gtpwizard_gsg546.pdf`	*LogiCORE IP Spartan-6 FPGA GTP Transceiver Wizard v1.8 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.
`gtp_attributes.ucf`	Constraints file containing the GTP attributes generated by the GTP Wizard GUI settings.
`<component_name>_top.ucf`	Constraint file for mapping the GTP wrapper example design onto a Spartan-6 device.
`<component_name>_top.v[hd]`	Top-level example design. Contains GTP 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 5-5: **Implement Directory**

Name	Description
<project_dir>/<component_name>/implement	
`chipscope_project.cpj`	ChipScope project file.
`data_vio.ngc`	ChipScope Virtual Input/Output (VIO) core netlist.
`icon.ngc`	ChipScope Integrated Controller (ICON) core netlist.
`ila.ngc`	ChipScope 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: **Results Directory**

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

Back to Top

<component name>/simulation

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

Table 5-7: **Simulation Directory**

Name	Description
<project_dir>/<component_name>/simulation	
demo_tb.v	Test bench to simulate the provided example design. See. Functional Simulation of the Example Design.
sim_reset_mgt_model.vhd	Reset module for VHDL required for emulating the GSR pulse at the beginning of functional simulation in order to correctly reset the VHDL MGT smart model.

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	Linux script for running simulation using ISE Simulator.
simulate_isim.bat	Windows script for running simulation using ISE Simulator.
simulate_mti.do	ModelSim simulation script.
simulate_ncsim.bat	Windows script for running simulation using Cadence IUS.
simulate_ncsim.sh	Linux script for running simulation using Cadence IUS.
simulate_vcs.sh	Linux script for running simulation using Synopsys VCS and VCS MX.
ucli_commands.key	Script for VCS commands.
vcs_session.tcl	Script for adding GTP Wrapper signals to VCS wave viewer.
wave_isim.tcl	Script for adding GTP Wrapper signals to the ISim wave viewer.
wave_mti.do	Script for adding GTP Wrapper signals to the ModelSim wave viewer.
wave_ncsim.sv	Script for adding GTP Wrapper signals to the Cadence IUS wave viewer.

Back to Top

Example Design

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

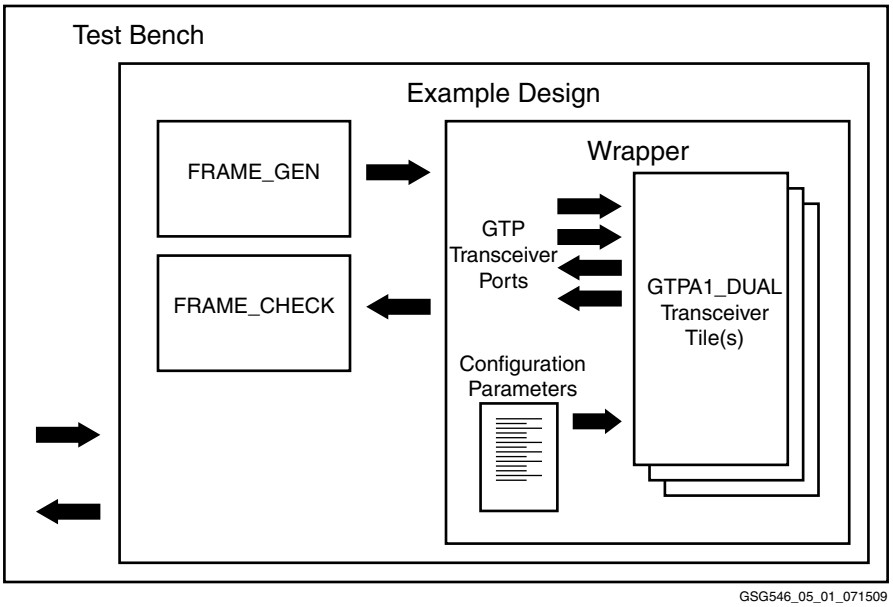

Figure 5-1: **Wrapper Block Diagram**

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 block RAM. This pattern can be easily modified by altering the parameters in the frame generator instantiation. The frame checker contains the same pattern in block RAM and compares it with the received data. An error counter in the frame checker keeps a track of how many errors have occurred.

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

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

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

the wrapper. The RX_SYNC module demonstrates the RX phase-alignment procedure outlined in the *Spartan-6 FPGA GTP Transceivers User Guide*.

The example design also demonstrates how to properly connect clocks to GTX transceiver ports TXUSRCLK, TXUSRCLK2, RXUSRCLK and RXUSRCLK2. Properly configured DCM (Digital Clock manager), PLL (Phase lock loop) wrappers are also provided if they are required to generate user clocks for the instantiated GTP 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 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 as follows:

```
example_tb
|___example_mgt_top
    |___mgt_userclk_source_pll
    |___ibufds
    |___frame_gen
    |___frame_check
    |___pcie_wrapper
        |___pcie_wrapper_tile
            |___gtpa1_dual
```