

# **LogiCORE™ Ethernet 1000BASE-X PCS/PMA or SGMII v7.0**

## **Getting Started Guide**

UG145 January 18, 2006





Xilinx is disclosing this Document and Intellectual Property (hereinafter “the Design”) to you for use in the development of designs to operate on, or interface with Xilinx FPGAs. Except as stated herein, none of the Design may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of the Design may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

Xilinx does not assume any liability arising out of the application or use of the Design; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Design. Xilinx reserves the right to make changes, at any time, to the Design as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Design.

THE DESIGN IS PROVIDED “AS IS” WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DESIGN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE DESIGN, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE DESIGN, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE DESIGN. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE DESIGN TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Design is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems (“High-Risk Applications”). Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You represent that use of the Design in such High-Risk Applications is fully at your risk.

© 2006 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

---

## Ethernet 1000BASE-X PCS/PMA or SGMII v7.0 Getting Started Guide UG145 January 18, 2006

The following table shows the revision history for this document.

Date	Version	Revision
09/30/04	1.0	Initial Xilinx release.
04/28/05	2.0	Updated to version 6.0 of the core, and Xilinx Tools v7.1i SP2.
01/18/06	3.0	Updated to version 7.0 of the core, and Xilinx Tools v8.1i, updated Licensing chapter.



# Table of Contents

---

<b>Schedule of Figures</b> .....	7
----------------------------------	---

## **Preface: About This Guide**

<b>Guide Contents</b> .....	9
<b>Additional Resources</b> .....	9
<b>Conventions</b> .....	10
Typographical.....	10
Online Document.....	11

## **Chapter 1: Introduction**

<b>About the Core</b> .....	13
<b>Recommended Design Experience</b> .....	13
<b>Additional Core Resources</b> .....	13
<b>Technical Support</b> .....	14
<b>Feedback</b> .....	14
Ethernet 1000BASE-X PCS/PMA or SGMII Core .....	14
Document .....	14

## **Chapter 2: Installing and Licensing the Core**

<b>System Requirements</b> .....	15
<b>Before You Begin</b> .....	15
<b>Installing the Core</b> .....	15
CORE Generator IP Updates Installer .....	16
Manual Installation .....	16
<b>Verifying the Installation</b> .....	17
<b>License Options</b> .....	17
Simulation Only .....	18
Full .....	18
<b>Obtaining Your License</b> .....	18
<b>Installing the License File</b> .....	18

## **Chapter 3: Quick Start Example Design**

<b>Overview</b> .....	19
<b>Generating the Core</b> .....	20
<b>Implementing the Example Design</b> .....	21
<b>Simulating the Example Design</b> .....	21
Setting up for Simulation .....	21
Functional Simulation .....	21
Timing Simulation .....	22
<b>What's Next?</b> .....	22

## Chapter 4: Detailed Example Design

<b>Directory Structure and File Descriptions</b> .....	23
VHDL Design Entry .....	23
Verilog Design Entry .....	27
<b>Implementation Scripts</b> .....	30
<b>Simulation Scripts</b> .....	30
Functional simulation .....	30
Timing simulation .....	31
<b>Core Example Design Using RocketIO</b> .....	32
Top-Level HDL .....	32
Transceiver .....	33
Transmitter Elastic Buffer .....	33
<b>Demonstration Test Bench</b> .....	34
Customizing the Test Bench .....	35
<b>Core Example Design with Ten-Bit Interface</b> .....	36
Example Design Top-Level HDL .....	36
Transmitter Elastic Buffer .....	37
Demonstration Test Bench .....	38
Customizing the Test Bench .....	39
<b>SGMII Example Design / Dynamic Switching Example Design</b> .....	40
Example Design Top-Level HDL .....	40
Transceiver .....	41
SGMII Adaptation Module .....	41
Clock Generation .....	43
Johnson Counter .....	44
Transmitter Rate Adaptation Module .....	45
Receiver Rate Adaptation Module .....	46
Demonstration Test Bench .....	48
Customizing the Test Bench .....	49

# Schedule of Figures

---

## Chapter 1: Introduction

## Chapter 2: Installing and Licensing the Core

<i>Figure 2-1: CORE Generator Window</i> .....	17
--	----

## Chapter 3: Quick Start Example Design

<i>Figure 3-1: Ethernet 1000BASE-X PCS/PMA Example Design and Test Bench</i> .....	19
--	----

<i>Figure 3-2: Core Customization Screen</i> .....	20
--	----

## Chapter 4: Detailed Example Design

<i>Figure 4-1: Core Directories and Files</i> .....	23
---	----

<i>Figure 4-2: Core Directories and Files</i> .....	27
---	----

<i>Figure 4-3: Top-Level HDL for the Ethernet 1000BASE-X PCS/PMA using RocketIO</i> ..	32
--	----

<i>Figure 4-4: Demonstration Test Bench Using RocketIO</i> .....	34
--	----

<i>Figure 4-5: Example Design Top-Level HDL for the Ethernet 1000BASE-X PCS with TBI</i> .....	36
--	----

<i>Figure 4-6: Demonstration Test Bench for the Ethernet 1000BASE-X PCS with TBI</i> ...	38
--	----

<i>Figure 4-7: Top- Level HDL for the Ethernet 1000BASE-X PCS/PMA or SGMII LogiCORE in SGMII mode</i> .....	40
---	----

<i>Figure 4-8: SGMII Adaptation Module</i> .....	42
--	----

<i>Figure 4-9: Clock Generator Output Clocks and Clock Enable</i> .....	44
---	----

<i>Figure 4-10: Transmitter Rate Adaptation Module Data Sampling</i> .....	46
--	----

<i>Figure 4-11: Receiver Rate Adaptation Module Data Sampling</i> .....	47
---	----

<i>Figure 4-12: Demonstration Test Bench for the Ethernet 1000BASE-X PCS/PMA or SGMII Core in SGMII Mode</i> .....	48
--	----





# About This Guide

---

The *LogiCORE™ Ethernet 1000Base-X PCS/PMA or SGMII v7.0 Getting Started Guide* provides information about generating an Ethernet 1000BASE-X PCS/PMA core, customizing and simulating the core using the provided example designs, and running the design files through implementation using the Xilinx tools.

## Guide Contents

The following chapters are included in this guide:

- [Preface, “About this Guide”](#) introduces the organization and purpose of the Getting Started Guide, a list of additional resources, and the conventions used in the guide.
- [Chapter 1, “Introduction”](#) describes the core and related information, including recommended design experience, additional resources, technical support, and submitting feedback to Xilinx.
- [Chapter 2, “Installing and Licensing the Core”](#) provides information about installing and licensing the core.
- [Chapter 3, “Quick Start Example Design”](#) provides instructions to quickly generate the core and run the example design through implementation and simulation using the default settings.
- [Chapter 4, “Detailed Example Design”](#) describes the demonstration test bench in detail and provides directions for how to customize the demonstration test bench for use in an application.

## Additional Resources

For additional information, go to <http://www.xilinx.com/support>. The following table lists some of the resources you can access from this website or by using the provided URLs.

Resource	Description/URL
Tutorials	Tutorials covering Xilinx design flows, from design entry to verification and debugging <a href="http://www.xilinx.com/support/techsup/tutorials/index.htm">http://www.xilinx.com/support/techsup/tutorials/index.htm</a>
Answer Browser	Database of Xilinx solution records <a href="http://www.xilinx.com/xlnx/xil_ans_browser.jsp">http://www.xilinx.com/xlnx/xil_ans_browser.jsp</a>
Application Notes	Descriptions of device-specific design techniques and approaches <a href="http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=Application+Notes">http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=Application+Notes</a>

Resource	Description/URL
Data Sheets	Device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging <a href="http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp">http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp</a>
Problem Solvers	Interactive tools that allow you to troubleshoot your design issues <a href="http://www.xilinx.com/support/troubleshoot/psolvers.htm">http://www.xilinx.com/support/troubleshoot/psolvers.htm</a>
Tech Tips	Latest news, design tips, and patch information for the Xilinx design environment <a href="http://www.xilinx.com/xlnx/xil_tt_home.jsp">http://www.xilinx.com/xlnx/xil_tt_home.jsp</a>

## 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	<code>speed grade: - 100</code>
<b>Courier bold</b>	Literal commands you enter in a syntactical statement	<b>ngdbuild</b> <i>design_name</i>
<i>Italic font</i>	Variables in a syntax statement for which you must supply values	See the <i>Development System Reference Guide</i> for more information.
	References to other manuals	See the <i>User Guide</i> for details.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are <i>not</i> 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 <b>bus [7:0]</b> , they are required.	<b>ngdbuild</b> [ <i>option_name</i> ] <i>design_name</i>
Braces { }	A list of items from which you must choose one or more	<b>lowpwr</b> = { <b>on</b>   <b>off</b> }
Vertical bar	Separates items in a list of choices	<b>lowpwr</b> = { <b>on</b>   <b>off</b> }

Convention	Meaning or Use	Example
Vertical ellipsis . . .	Repetitive material that has been omitted	IOB #1: Name = QOUT' IOB #2: Name = CLKIN' . . .
Horizontal ellipsis ...	Repetitive material that has been omitted	<b>allow block</b> <i>block_name</i> <i>loc1 loc2 ... locn</i> ;
Notations	The prefix '0x' or the suffix 'h' indicate hexadecimal notation	A read of address 0x00112975 returned 45524943h.
	A '_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 " <a href="#">Additional Resources</a> " for details. Refer to " <a href="#">Title Formats</a> " in <a href="#">Chapter 1</a> for details.
<a href="#">Blue, underlined text</a>	Hyperlink to a website (URL)	Go to <a href="http://www.xilinx.com">http://www.xilinx.com</a> for the latest speed files.



# Introduction

---

The Ethernet 1000BASE-X PCS/PMA or SGMII core is a fully-verified solution that supports Verilog-HDL and VHDL. In addition, the example design in this guide is provided in both Verilog and VHDL formats.

This chapter introduces the core and provides some related information, including recommended design experience, additional resources, technical support, and how to submit feedback to Xilinx.

## About the Core

The Ethernet 1000BASE-X PCS/PMA or SGMII core is a Xilinx CORE Generator™ IP core, included in the latest IP Update on the Xilinx IP Center. For detailed information about the core, see [http://www.xilinx.com/systemio/1gbsx\\_phy/index.htm](http://www.xilinx.com/systemio/1gbsx_phy/index.htm). For information about system requirements, installation, and licensing options, see Chapter 2, “Installing and Licensing the Core.”

## Recommended Design Experience

Although the Ethernet 1000BASE-X PCS/PMA or SGMII core is a fully-verified solution, the challenge associated with implementing a complete design varies, depending on the configuration and functionality of the application. For best results, previous experience building high-performance, pipelined FPGA designs using Xilinx implementation software and user constraint files (UCFs) is recommended.

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

## Additional Core Resources

For detailed information and updates about the Ethernet 1000BASE-X PCS/PMA or SGMII core, see the following documents, located on the Ethernet 1000BASE-X PCS/PMA or SGMII product page at [http://www.xilinx.com/systemio/1gbsx\\_phy/index.htm](http://www.xilinx.com/systemio/1gbsx_phy/index.htm).

- *Xilinx Ethernet 1000BASE-X PCS/PMA or SGMII Data Sheet*
- *Xilinx Ethernet 1000BASE-X PCS/PMA or SGMII Release Notes*
- *Xilinx Ethernet 1000BASE-X PCS/PMA or SGMII User Guide*

For updates to this document, see the *Ethernet 1000BASE-X PCS/PMA or SGMII Getting Started Guide*, also located on the Ethernet 1000BASE-X PCS/PMA or SGMII product page.

## Technical Support

For technical support, see <http://support.xilinx.com/>. Questions are routed to a team of engineers with expertise using the Ethernet 1000BASE-X PCS/PMA or SGMII core.

Xilinx will provide technical support for use of this product as described in the *Xilinx Ethernet 1000BASE-X PCS/PMA* or the *Xilinx SGMII User Guide* and the *Ethernet 1000BASE-X PCS/PMA or SGMII 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 Ethernet 1000BASE-X PCS/PMA or SGMII core and the documentation supplied with the core.

### Ethernet 1000BASE-X PCS/PMA or SGMII Core

For comments or suggestions about the Ethernet 1000BASE-X PCS/PMA or SGMII core, please submit a WebCase from [www.xilinx.com/support/clearxpress/websupport.htm/](http://www.xilinx.com/support/clearxpress/websupport.htm/)

Be sure to include the following information:

- Product name
- Core version number
- Explanation of your comments

### Document

For comments or suggestions about this document, please submit a WebCase from [www.xilinx.com/support/clearxpress/websupport.htm/](http://www.xilinx.com/support/clearxpress/websupport.htm/)

Be sure to include the following information:

- Document title
- Document number
- Page number(s) to which your comments refer
- Explanation of your comments

# Installing and Licensing the Core

---

This chapter provides instructions for installing the Ethernet 1000BASE-X PCS/PMA or SGMII core and obtaining a license for the core, which you must do before using the core in your designs. The Ethernet 1000BASE-X PCS/PMA or SGMII core is provided under the terms of the [Xilinx LogiCORE Site License Agreement](#), which conforms to the terms of the [SignOnce](#) IP License standard defined by the Common License Consortium.

## System Requirements

### Windows

- Windows® 2000 Professional with Service Pack 2-4
- Windows XP Professional with Service Pack 1

### Solaris/Linux

- Sun Solaris® 8/9
- Red Hat® Enterprise Linux 3.0 (32-bit and 64-bit)

### Software

- Xilinx ISE™ 8.1i

## Before You Begin

Before installing the core, you must have a Xilinx.com account and the ISE 8.1i software installed on your system. If you have already completed these steps, go to [“Installing the Core.”](#)

1. Click Login at the top of the [Xilinx home page](#); then follow the onscreen instructions to create a support account.
2. Install the ISE 8.1i software and the applicable Service Pack software. ISE Service Packs can be downloaded from [www.xilinx.com/support/download.htm](http://www.xilinx.com/support/download.htm).

## Installing the Core

You can install the core in two ways—using the CORE Generator IP Updates Installer, which lets you select from a list of updates—or by performing a manual installation after downloading the core from the web.

## CORE Generator IP Updates Installer

**Note:** To use this installation method behind a firewall, you must know your proxy settings. Contact your administrator to determine the proxy host address and port number before you begin, if necessary.

1. Start the CORE Generator; then open an existing project or create a new one.
2. From the main CORE Generator window, choose Tools > Updates Installer to start the Updates Installer. If you are behind a firewall, you will be prompted to enter your proxy host and port settings.
3. If necessary, enter your proxy settings; then click Set. The IP Updates installer appears.
4. Click the checkbox next to 8.1i\_IP\_Update1 to select it; then click Install Selected. Informational messages may appear indicating that additional installations are required.
5. Click OK to accept any messages and continue. The User Login dialog box appears.
6. Enter your login name and password; then click OK. The Updates Installer Generator downloads and installs the selected products, and then exits.
7. To confirm the installation, check the following file:  
C:\Xilinx\coregen\install\install\_history.

Note that this step assumes your Xilinx software is installed in C:\Xilinx.

## Manual Installation

1. Close the CORE Generator application if it is running.
2. Download the IP Update ZIP file from the following location and save it to a temporary directory:  
[http://www.xilinx.com/xlnx/xil\\_sw\\_updates\\_home.jsp?update=ip&software=8.1i](http://www.xilinx.com/xlnx/xil_sw_updates_home.jsp?update=ip&software=8.1i)
3. Unpack the ZIP files using either WinZip (Windows) or Unzip (UNIX).
4. Extract the ZIP file (ise\_81i\_ip\_update1.zip) archive to the root directory of your Xilinx software installation. (Allow the extractor utility you use to overwrite all existing files and maintain the directory structure defined in the archive.)
5. If you do not have a zip utility, do one of the following:
  - **Windows.** From a command window, type the following:  
%XILINX%/bin/nt/unzip -d %XILINX% ise\_81i\_ip\_update1.zip
  - **Linux.** From a UNIX shell, type the following:  
\$XILINX/bin/lin/unzip -d \$XILINX ise\_81i\_ip\_update1.zip
  - **Solaris.** From a UNIX shell, type the following:  
\$XILINX/bin/sol/unzip -d \$XILINX ise\_81i\_ip\_update1.zip
6. To verify the root directory of your Xilinx installation, do one of the following:
  - **Windows.** Type `echo %XILINX%` from a DOS prompt.
  - **UNIX.** If you have already installed the Xilinx ISE software, the Xilinx variable defined by your set-up script identifies the location of the Xilinx installation directory. After sourcing the Xilinx set-up script, type `echo $XILINX` to determine the location of the Xilinx installation.



## Verifying the Installation

1. Start the CORE Generator.
2. After creating a new project or opening an existing one, the IP core functional categories appear at the left side of the window.

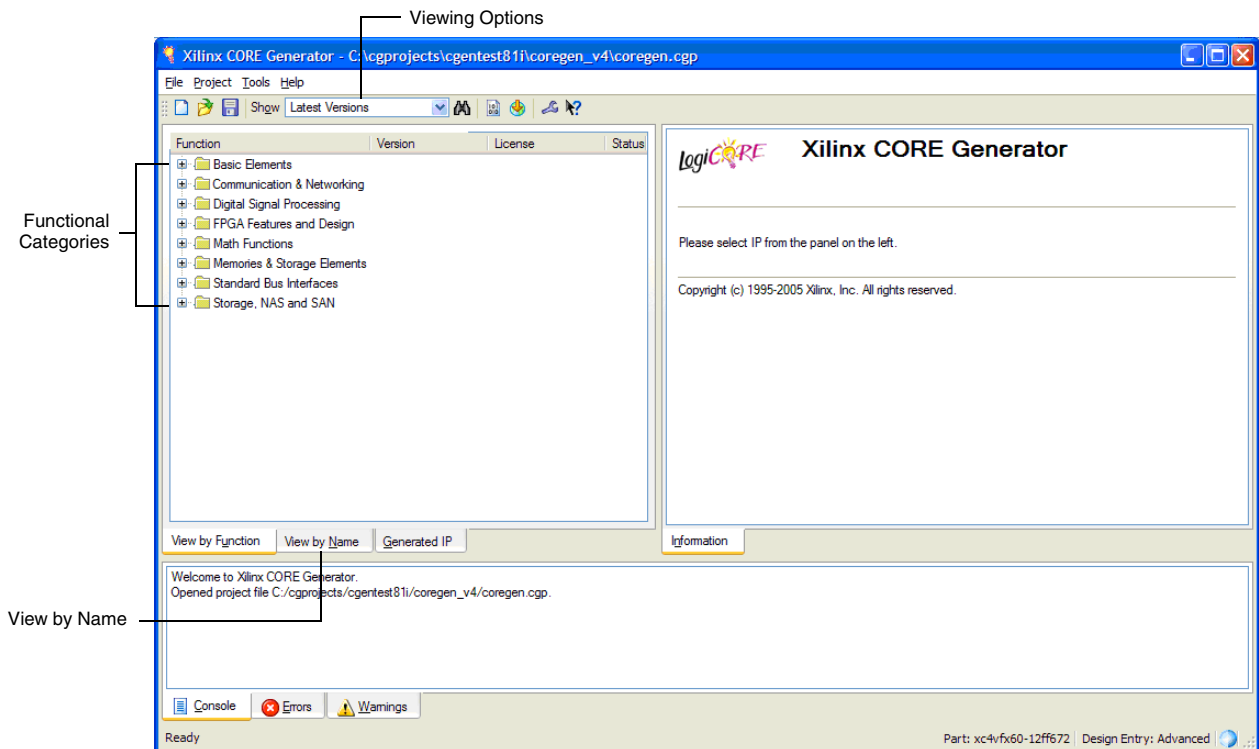


Figure 2-1: CORE Generator Window

3. Click to expand or collapse the view of individual functional categories, or click the View by Name tab at the bottom of the list to see an alphabetical list of all cores in all categories.
4. To view specific versions of the cores, choose an option from the Show drop-down list at the top of the window:
  - **Latest Versions.** Display the latest versions of all cores.
  - **All Versions.** Display all versions of cores, including new cores and new versions of cores.
  - **All Versions including Obsolete.** Display all cores, including those scheduled to become obsolete.
5. Determine if the installation was successful by verifying that the new core or cores appear in the CORE Generator GUI.  
For additional assistance installing the IP Update, contact [www.xilinx.com/support](http://www.xilinx.com/support).

## License Options

The Ethernet 1000BASE-X PCS/PMA or SGMII core has the following two licensing options.

## Simulation Only

The Simulation Only Evaluation license is provided by default with the Xilinx CORE Generator and requires no electronic license key. This license lets you assess the core functionality with either the provided example design or alongside your own design and demonstrates the various interfaces to the core in simulation. (Functional simulation is supported by a structural model generated by the CORE Generator.) When you open the core from the CORE Generator, a message appears regarding the limitations of the Simulation Only Evaluation license.

## Full

The Full license provides full access to all core functionality both in simulation and in hardware, including:

- Functional simulation support
- Back annotated gate-level simulation support
- Full implementation support including place and route and bitstream generation
- Full functionality in the programmed device with no time outs

## Obtaining Your License

**Note:** Because the Simulation Only Evaluation license is provided with the CORE Generator, no action is required.

### Obtaining a Full License

To obtain a Full license, you must register for access to the *lounge*, a secured area of the Ethernet 1000BASE-X PCS/PMA or SGMII product page.

- From the product page, click Register to register and request access to the lounge. Access to the lounge is automatic and granted immediately.
- After you receive confirmation of lounge access, click Access Lounge on the Ethernet 1000BASE-X PCS/PMA or SGMII product page and log in.
- Click Access Lounge on the product lounge page and fill out the license request form linked from this location; then click Submit to automatically generate the license. An e-mail containing the license and installation instructions will be sent immediately to the email address you specified.

## Installing the License File

The Simulation Only Evaluation license does not require a license file; it is provided with the CORE Generator. If you select the Full license option, an email will be sent to you containing instructions for installing your license file. In addition, the email provides information about advanced licensing options and technical support.

## Quick Start Example Design

The quick start steps provided in this chapter let you quickly generate an Ethernet 1000BASE-X PCS/PMA or SGMII core, run the design through implementation with the Xilinx tools, and simulate the design using the provided demonstration test bench. For detailed information about the example design, see [Chapter 4, “Detailed Example Design.”](#)

### Overview

The Ethernet 1000BASE-X PCS/PMA example design consists of:

- Ethernet 1000BASE-X PCS/PMA core netlist
- Example design HDL top-level and associated HDL files
- Demonstration test bench to exercise the example design

The Ethernet 1000BASE-X PCS/PMA example design has been tested with Xilinx ISE 8.1i, Cadence IUS 5.5 and Modelsim® PE/SE 6.1a.

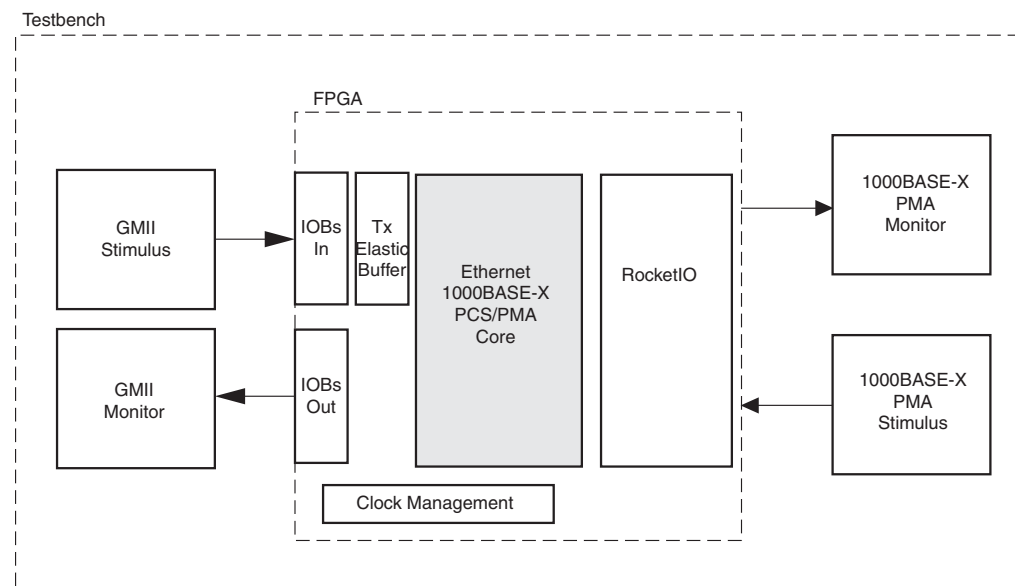


Figure 3-1: Ethernet 1000BASE-X PCS/PMA Example Design and Test Bench

## Generating the Core

This section provides detailed instructions for generating the Ethernet 1000BASE-X PCS/PMA example design core.

### To generate the example design core

1. Start the CORE Generator tool.
 

For general help with starting and using CORE Generator on your system, see the documentation supplied with ISE, including the *Core Generator Guide*. These documents can be downloaded from:  
[http://www.xilinx.com/support/software\\_manuals.htm](http://www.xilinx.com/support/software_manuals.htm).
2. Create a new project.
3. For project options, select the following:
  - A Virtex™-II Pro part to generate the default Ethernet 1000BASE-X PCS/PMA core.
  - In the Design Entry section, select VHDL or Verilog; then select Other for Vendor.
4. Locate the Ethernet 1000BASE-X PCS/PMA or SGMII core in the taxonomy tree, listed under one of the following:
  - Communications & Networking/Ethernet
  - Communications & Networking/Networking
  - Communications & Networking/Telecommunications
5. Double-click the core.
 

A warning may appear to indicate the limitations of the Simulation Only Evaluation license.
6. Click OK to display the core customization screen.

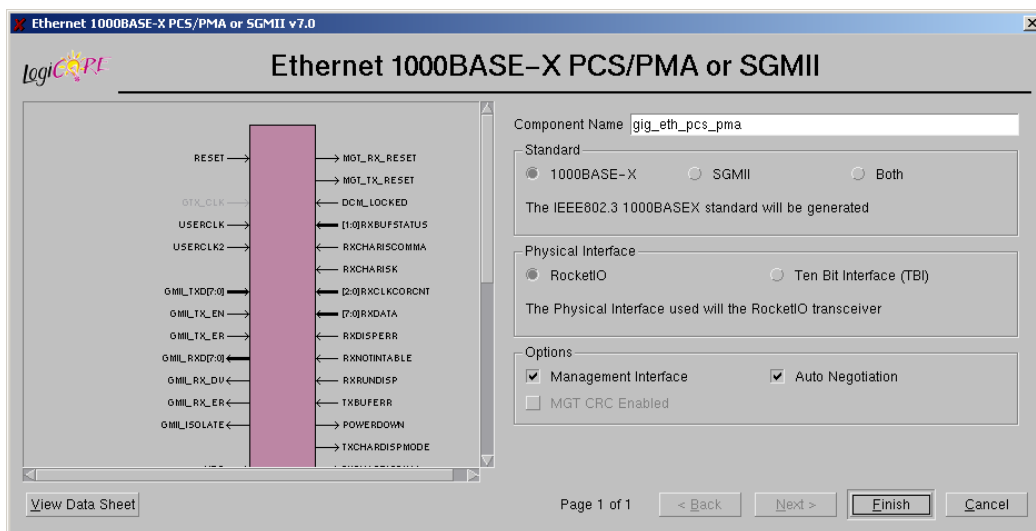


Figure 3-2: Core Customization Screen

7. Enter a name for the instance of the core in the Component Name text field
8. Click Finish to generate the core (with the default options).

The default core and its supporting files, including the example design, are generated in your project directly. For a detailed description of the design example files and directories, refer to “Directory Structure and File Descriptions” in Chapter 4.

## Implementing the Example Design

**Note:** Available only with a Full license.

After the core is generated, the netlists and example design can be processed by the Xilinx implementation tools. The generated output files include several scripts to assist you in running the Xilinx software.

**To implement the Ethernet 1000BASE-X PCS/PMA or SGMII sample design core**

From the CORE Generator project directory window, type in the following:

**For UNIX:**

```
unix-shell> cd <project_dir>/<component_name>/implement
unix-shell> ./implement.sh
```

**For Windows:**

```
ms-dos> cd <project_dir>\<component_name>\implement
ms-dos> implement.bat
```

These commands execute a script that synthesizes, builds, maps, and place-and-routes the example design. The script then creates gate-level netlist HDL files in either VHDL or Verilog, along with associated timing information (SDF) files.

## Simulating the Example Design

### Setting up for Simulation

To run the gate-level simulation you must have the Xilinx Simulation Libraries compiled for your system. See the Compiling Xilinx Simulation Libraries (COMPXLIB) in the *Xilinx ISE Synthesis and Verification Design Guide*, and the *Xilinx ISE Software Manuals and Help*. You can download these documents from:  
[http://www.xilinx.com/support/software\\_manuals.htm](http://www.xilinx.com/support/software_manuals.htm).

In addition, the simulator you use must provide SWIFT model support to simulate the Virtex-II Pro or Virtex-4 RocketIO™ transceivers.

In the simulation examples that follow, <project\_dir> is the CORE Generator project directory; <component\_name> is the component name as entered in the core customization window.

### Functional Simulation

**Note:** Available for both license types.

This section provides instructions for running a functional simulation of the Ethernet 1000BASE-X PCS/PMA or SGMII core using either VHDL or Verilog. The functional simulation model is provided when the core is generated; implementing the core before simulation is not required.

**To run a VHDL or Verilog functional simulation of the example design:**

1. Open a command prompt or shell, then set the current directory to:  
`<project_dir>/<component_name>/simulation/functional/`
2. Launch the simulation script:

```
ModelSim: vsim -do simulate_mti.do  
IUS: ./simulate_ncsim.sh
```

The simulation script compiles the functional simulation model, the example design files, the demonstration test bench, and adds relevant signals to a wave window. It then runs the simulation to completion. After completion, you can inspect the simulation transcript and waveform to observe the operation of the core.

## Timing Simulation

**Note:** Available only with a Full license.

This section contains instructions for running a timing simulation or the Ethernet 1000BASE-X PCS/PMA or SGMII core using either VHDL or Verilog. A timing simulation model is generated when run through the Xilinx tools using the implementation script. You must implement the core before attempting to run timing simulation.

**To run a VHDL or Verilog timing simulation of the example design:**

1. Run the implementation script (see [“Implementing the Example Design”](#)).
2. Open a command prompt or shell, then set the current directory to:  
`<project_dir>/<component_name>/simulation/timing/`
3. Launch the simulation script:

```
ModelSim: vsim -do simulate_mti.do  
IUS: ./simulate_ncsim.sh
```

The simulator script compiles the gate-level model and the demonstration test bench, adds relevant signals to a wave window, and then runs the simulation to completion. You can then inspect the simulation transcript and waveform to observe the operation of the core.

## What's Next?

For detailed information about the example design, including guidelines for modifying the design and extending the test bench, see [Chapter 4, “Detailed Example Design.”](#)

To begin using the Ethernet 1000BASE-X PCS/PMA or SGMII core in your own designs, see the *Xilinx Ethernet 1000BASE-X PCS/PMA or SGMII User Guide*.

## Detailed Example Design

This chapter provides detailed information about the Ethernet 1000BASE-X PCS/PMA or SGMII example design, including a description of files and the directory structure generated by the Xilinx CORE Generator, the purpose and contents of the provided scripts, the contents of the example HDL wrappers, and the demonstration test bench.

### Directory Structure and File Descriptions

#### VHDL Design Entry

Figure 4-1 shows the files and directories created by CORE Generator system in a VHDL Design Entry Project. In this example, *<project\_dir>* is the CORE Generator project directory; *<component\_name>* is the component name as entered in the core customization window.

**Note:** The implement and simulation/timing directories are only present with a Full license.

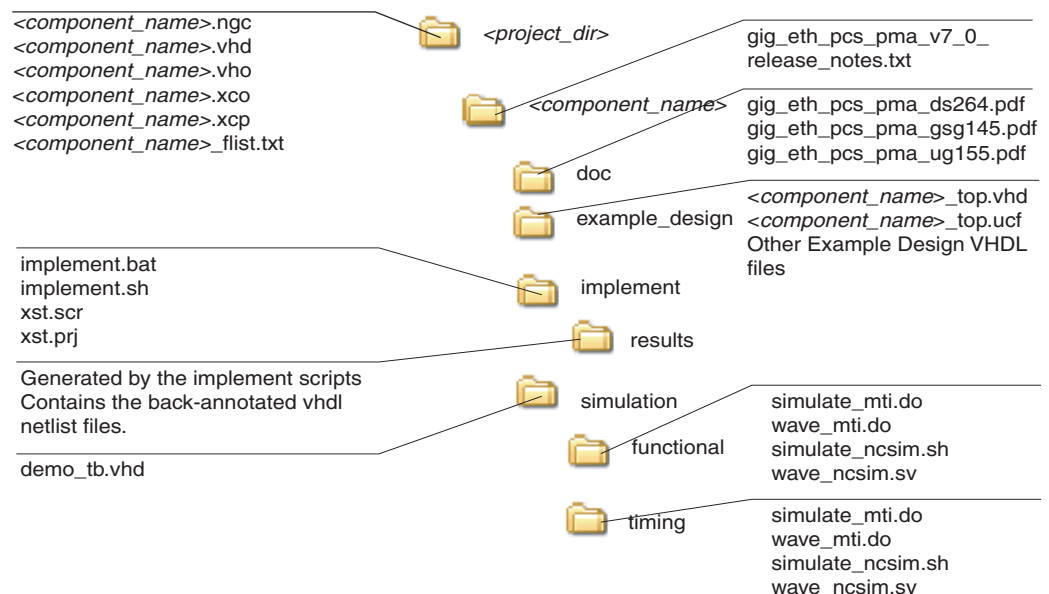


Figure 4-1: Core Directories and Files

## Project Directory (<project\_dir>)

### **<component\_name>.ngc**

The Xilinx netlist for the core. This is instantiated by the VHDL example design.

### **<component\_name>.vhd**

The VHDL simulation model used to support the VHDL functional simulation of the core. This is UNISIM based.

### **<component\_name>.vho**

This contains a VHDL instantiation template for the core.

### **<component\_name>.xco**

A log file that records the settings used to generate a core. An XCO file is generated by the 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.

### **<component\_name>.xcp**

Similar to the XCO file except that it does not specify project-specific settings, such as target architecture and output products.

### **<component\_name>\_flist.txt**

A test file listing all of the output files produced when a customized core is generated by CORE Generator.

## <project\_dir>/<component\_name>

### **gig\_eth\_pcs\_pma\_v7\_0\_release\_notes.txt**

The Ethernet 1000BASE-X PCS/PMA or SGMII core release notes text document.

## <project\_dir>/<component\_name>/doc

This directory contains the following documentation for this core.

### **gig\_eth\_pcs\_pma\_ds264.pdf**

This is the *Ethernet 1000BASE-X PCS/PMA or SGMII v7.0 Data Sheet*.

### **gig\_eth\_pcs\_pma\_gsg145.pdf**

This is the *Ethernet 1000BASE-X PCS/PMA or SGMII v7.0 Getting Started Guide*.

### **gig\_eth\_pcs\_pma\_ug155.pdf**

This is the *Ethernet 1000BASE-X PCS/PMA or SGMII v7.0 User Guide*.

## <project\_dir>/<component\_name>/example\_design

This directory contains the support files you need for a VHDL implementation of the example design.

### **<component\_name>\_top.vhd**

This file is the top-level VHDL file for the example design. Other VHDL design files may also be present. Please see the following sections for more information.

- [“Core Example Design Using RocketIO,” page 32.](#)



- “Core Example Design with Ten-Bit Interface,” page 36.
- “SGMII Example Design / Dynamic Switching Example Design,” page 40.

### <project\_dir>/implement

**Note:** This directory is only present with the Full license.

#### **implement.sh**

A UNIX shell script that processes the example design through the Xilinx tool flow. See “Implementation Scripts,” page 30 for more information.

#### **implement.bat**

A Windows batch file that process the example design through the Xilinx tool flow. See “Implementation Scripts,” page 30 for more information.

#### **xst.prj**

The XST project file for the example design; it enumerates all of the VHDL files that need to be synthesized.

#### **xst.scr**

The XST script file for the example design.

### <project\_dir>/implement/results

This directory is produced by the implementation scripts and is used to run through the example design and the core through the Xilinx implementation tools. Once the implement script has completed it contains the following files for timing simulation.

#### **routed.vhd**

The back-annotated simprim based VHDL model used for timing simulation. This is SIMPRIM-based.

#### **routed.sdf**

The timing information for simulation is contained in this file.

### <project\_dir>/<component\_name>/simulation

The simulation directory and subdirectories that provide the files necessary to test a VHDL implementation of the example design.

#### **demo\_tb.vhd**

The VHDL demonstration test bench file for the example design. See the following for more information.

- “Core Example Design Using RocketIO,” page 32.
- “Core Example Design with Ten-Bit Interface,” page 36.
- “SGMII Example Design / Dynamic Switching Example Design,” page 40.

### <project\_dir>/<component\_name>/simulation/functional

#### **simulate\_mti.do**

A ModelSim® macro file that compiles the VHDL sources and then runs the functional simulation to completion.

**wave\_mti.do**

A ModelSim macro file that opens a wave window and adds signals of interest to it. It is called by the `simulate_mti.do` macro file.

**simulate\_ncsim.sh**

An IUS script file that compiles the VHDL sources and then runs the functional simulation to completion.

**wave\_ncsim.sv**

An IUS macro file that opens a wave window and adds signals of interest to it. It is called by the `simulate_ncsim.sh` script file.

**<project\_dir>/<component\_name>/simulation/timing**

**Note:** This directory is only present with the Full license.

**simulate\_mti.do**

A ModelSim macro file that compiles the VHDL sources and then runs the timing simulation to completion.

**wave\_mti.do**

A ModelSim macro file that opens a wave window and adds signals of interest to it. It is called by the `simulate_mti.do` macro file.

**simulate\_ncsim.sh**

An IUS script file that compiles the VHDL sources and then runs the timing simulation to completion.

**wave\_ncsim.sv**

An IUS macro file that opens a wave window and adds signals of interest to it. It is called by the `simulate_ncsim.sh` script file.

## Verilog Design Entry

Figure 4-2 shows the files and directories created by CORE Generator system in a Verilog Design Entry Project. `<project_dir>` is the CORE Generator project directory; `<component_name>` is the component name as entered in the Core Customization window.

**Note:** The implement and simulation/timing directories are only present with the Full License.

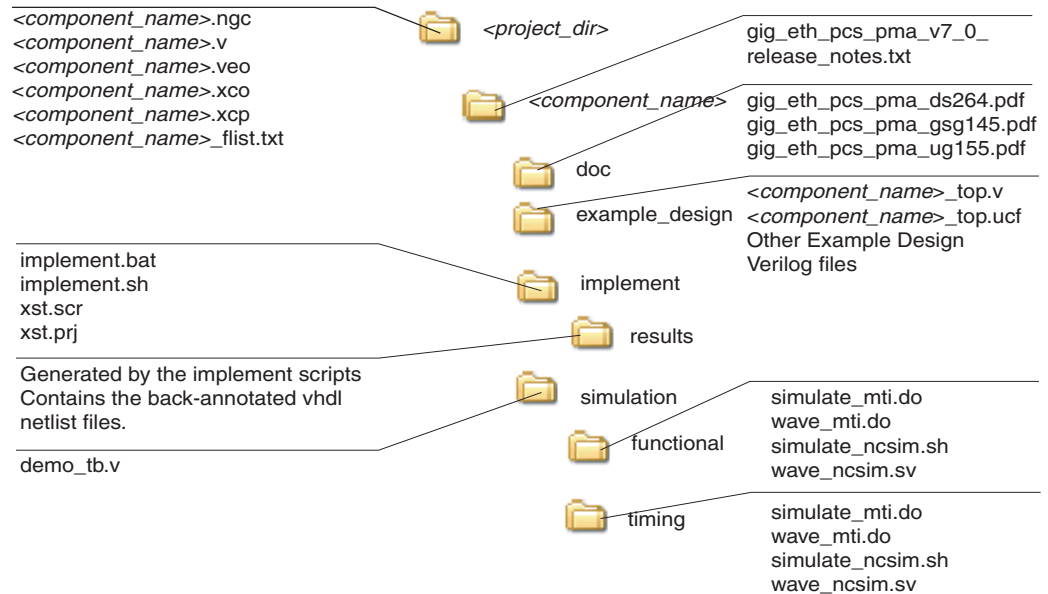


Figure 4-2: Core Directories and Files

### Project Directory (`<project_dir>`)

#### `<component_name>.ngc`

The Xilinx netlist for the core. This is instantiated by the VHDL example design.

#### `<component_name>.v`

The Verilog simulation model used to support the Verilog functional simulation of the core. This is UNISIM-based.

#### `<component_name>.veo`

This contains a Verilog instantiation template for the core.

#### `<component_name>.xco`

A log file that records the settings used to generate a core. An XCO file is generated by the 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.

#### `<component_name>.xcp`

Similar to the XCO file except that it does not specify project specific settings such as target architecture and output products.

#### `<component_name>_flist.txt`

A test file listing all of the output files produced when a customized core is generated by CORE Generator.

#### <project\_dir>/<component\_name>

##### **gig\_eth\_pcs\_pma\_v7\_0\_release\_notes.txt**

The Ethernet 1000BASE-X PCS/PMA or SGMII core release notes text document.

#### <project\_dir>/<component\_name>/doc

This directory contains documentation for the core.

##### **gig\_eth\_pcs\_pma\_ds264.pdf**

This is the *Xilinx Ethernet 1000BASE-X PCS/PMA or SGMII v7.0 Data Sheet*.

##### **gig\_eth\_pcs\_pma\_gsg145.pdf**

This is the *Xilinx Ethernet 1000BASE-X PCS/PMA or SGMII v7.0 Getting Started Guide*.

##### **gig\_eth\_pcs\_pma\_ug155.pdf**

This is the *Xilinx Ethernet 1000BASE-X PCS/PMA or SGMII v7.0 User Guide*.

#### <project\_dir>/<component\_name>/example\_design

This directory contains the support files necessary for a Verilog implementation of the example design.

##### **<component\_name>.top.v**

This file is the top-level Verilog file for the example design. Other Verilog design files may also be present. Please see the following for more information.

- [“Core Example Design Using RocketIO,” page 32](#)
- [“Core Example Design with Ten-Bit Interface,” page 36](#)
- [“SGMII Example Design / Dynamic Switching Example Design,” page 40](#)

#### <project\_dir>/implement

**Note:** This directory is only present with the Full license.

##### **implement.sh**

A UNIX shell script that processes the example design through the Xilinx tool flow. See [“Implementation Scripts,” page 30](#) for more information.

##### **implement.bat**

A Windows batch file that process the example design through the Xilinx tool flow. See [“Implementation Scripts,” page 30](#) for more information.

##### **xst.prj**

The XST project file for the example design. It enumerates all of the Verilog files that need to be synthesized.

##### **xst.scr**

The XST script file for the example design.

## <project\_dir>/implement/results

This directory is produced by the implementation scripts and is used to run through the example design and the core through the Xilinx implementation tools. Once the implement script has completed it contains the following files for timing simulation.

### **routed.v**

The back-annotated simprim based Verilog model used for timing simulation. This is SimPrim-based.

### **routed.sdf**

The timing information for simulation is contained in this file.

## <project\_dir>/<component\_name>/simulation

The simulation directory and subdirectories provide the files you need to test a Verilog implementation of the example design.

### **demo\_tb.v**

The Verilog demonstration test bench file for the example design. See the following sections in this document for more information.

- [“Core Example Design Using RocketIO,” page 32](#)
- [“Core Example Design with Ten-Bit Interface,” page 36](#)
- [“SGMII Example Design / Dynamic Switching Example Design,” page 40](#)

## <project\_dir>/<component\_name>/simulation/functional

### **simulate\_mti.do**

A ModelSim macro file that compiles the Verilog sources and then runs the functional simulation to completion.

### **wave\_mti.do**

A ModelSim macro file that opens a wave window and adds signals of interest to it. It is called by the `simulate_mti.do` macro file.

### **simulate\_ncsim.sh**

An IUS script file that compiles the Verilog sources and runs the functional simulation to completion.

### **wave\_ncsim.sv**

An IUS macro file that opens a wave window and adds signals of interest to it. It is called by the `simulate_ncsim.sh` script file.

## <project\_dir>/<component\_name>/simulation/timing

**Note:** This directory is only present with the Full license.

### **simulate\_mti.do**

A ModelSim macro file that compiles the Verilog sources and then runs the timing simulation to completion.

### **wave\_mti.do**

A ModelSim macro file that opens a wave window and adds signals of interest to it. It is called by the `simulate_mti.do` macro file.

**simulate\_ncsim.sh**

An IUS script file that compiles the Verilog sources and then runs the timing simulation to completion.

**wave\_ncsim.sv**

An IUS macro file that opens a wave window and adds signals of interest to it. It is called by the `simulate_ncsim.sh` script file.

## Implementation Scripts

**Note:** These scripts are only present with the Full license.

The implementation script is either a shell script or batch file that processes the example design through the Xilinx tool flow. It is located at:

**UNIX:**

```
<project_dir>/<component_name>/implement/implement.sh
```

**Windows:**

```
<project_dir>/<component_name>/implement/implement.bat
```

The `implement` script performs the following steps:

1. The HDL example design files are synthesized using XST.
2. Ngdbuild is run to consolidate the core netlist and the example design netlist into the NGD file containing the entire design.
3. The design is mapped to the target technology.
4. The design is placed-and-routed on the target device.
5. Static timing analysis is performed on the routed design using trce.
6. A bitstream is generated.
7. Netgen runs on the routed design to generate a VHDL or Verilog netlist (as appropriate for the Design Entry project setting) and timing information in the form of SDF files.

The Xilinx tool flow generates several output and report files. These are saved in the following directory which is created by the `implement` script:

```
<project_dir>/<component_name>/implement/results
```

## Simulation Scripts

### Functional simulation

The test script is a ModelSim or an IUS macro that automates the simulation of the test bench. It is located at:

```
<project_dir>/<component_name>/simulation/functional/
```

The test script performs the following tasks:

- Compiles the structural UNISIM simulation model

- Compiles HDL example design source code
- Compiles the demonstration test bench
- Starts a simulation of the test bench
- Opens a Wave window and adds signals of interest  
(*wave\_mti.do/wave\_ncsim.sv*)
- Runs the simulation to completion

## Timing simulation

**Note:** This script is only present with the Full license.

The test script is a ModelSim or an IUS macro that automates the simulation of the test bench. It is located at:

```
<project_dir>/<component_name>/simulation/timing/
```

The test script performs the following tasks.

- Compiles the SimPrim based gate level netlist simulation model
- Compiles the demonstration test bench
- Starts a simulation of the test bench
- Opens a Wave window and adds signals of interest (*wave\_mti.do/wave\_ncsim.sv*)
- Runs the simulation to completion

## Core Example Design Using RocketIO

### Top-Level HDL

Figure 4-3 illustrates the example design for a top-level HDL for the Ethernet 1000BASE-X PCS/PMA using RocketIO.

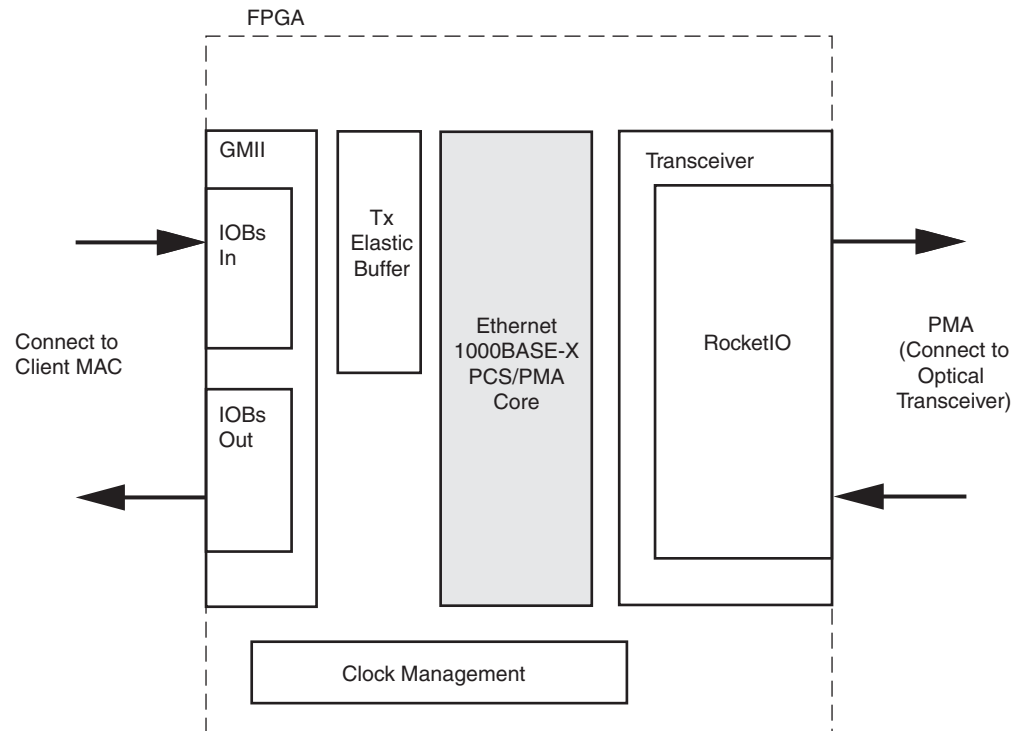


Figure 4-3: Top-Level HDL for the Ethernet 1000BASE-X PCS/PMA using RocketIO

The following files describe the top-level example design for the Ethernet 1000BASE-X PCS/PMA core using a RocketIO core.

#### VHDL

```
project_dir>/<component_name>/example_design/<component_name>_top.vhd
```

#### Verilog

```
project_dir>/<component_name>/example_design/<component_name>_top.v
```

The example design HDL top level contains the following:

- An instance of the Ethernet 1000BASE-X PCS/PMA core
- An instance of a Virtex-II Pro or Virtex-4 RocketIO transceiver
- Clock management logic for the core and the RocketIO transceiver, including DCM and Global Clock Buffer instances
- A transmitter elastic buffer
- GMIi interface logic, including IOB and DDR registers instances, where required
- Input and output buffers for other port signals of the top level



The example design HDL top-level connects the GMII of the core to external IOBs and the PHY side interface of the core directly to a RocketIO instance. This configuration allows the functionality of the core to be demonstrated using a simulation package as discussed in this guide. The example design can also be synthesized and, if required, placed on a suitable board and demonstrated in hardware.

## Transceiver

A wrapper file for the Virtex-II Pro RocketIO or Virtex-4 Multi-Gigabit Transceiver is described in the following files:

### VHDL

```
project_dir>/<component_name>/example_design/transceiver.vhd
```

### Verilog

```
project_dir>/<component_name>/example_design/transceiver.v
```

This file instances a Virtex-II Pro RocketIO or Virtex-4 RocketIO and applies Gigabit Ethernet 1000BASE-X attributes to it. This transceiver wrapper is instantiated from the top-level HDL file for the example design.

For Virtex-4 FX devices only, a Calibration Block is required. See the [Calibration Block Users Guide](#) for more information.

This is described in the following files:

### VHDL

```
project_dir/<component_name>/example_design/cal_block_v1_2_1.vhd
```

### Verilog

```
project_dir/<component_name>/example_design/cal_block_v1_2_1.v
```

## Transmitter Elastic Buffer

The Transmitter Elastic Buffer is described in the following files:

### VHDL

```
project_dir>/<component_name>/example_design/tx_elastic_buffer.vhd
```

### Verilog

```
project_dir>/<component_name>/example_design/tx_elastic_buffer.v
```

When the GMII is used externally (as in this example design), the GMII transmit signals (inputs to the core from a remote MAC at the other end of the interface) are synchronous to a clock that is likely to be derived from a different clock source to the core. For this reason, GMII transmit signals must be transferred into the core main clock domain before they can be used by the core and RocketIO. This is achieved with the Transmitter Elastic Buffer, an asynchronous FIFO implemented in distributed RAM. The operation of the elastic buffer is to attempt to maintain a constant occupancy by inserting or removing any idle sequences. This causes no corruption to the frames of data.

When the GMII is used as an internal interface, it is expected that the entire interface will be synchronous to a single clock domain and the Transmitter Elastic Buffer should be discarded. See the *LogiCORE Ethernet 1000BASE-X PCS/PMA or SGMII User Guide* for information about connecting the core to an internal GMII or an Ethernet MAC.

## Demonstration Test Bench

Figure 4-4 illustrates the demonstration test bench for the Ethernet 1000BASE-X PCS/PMA using RocketIO.

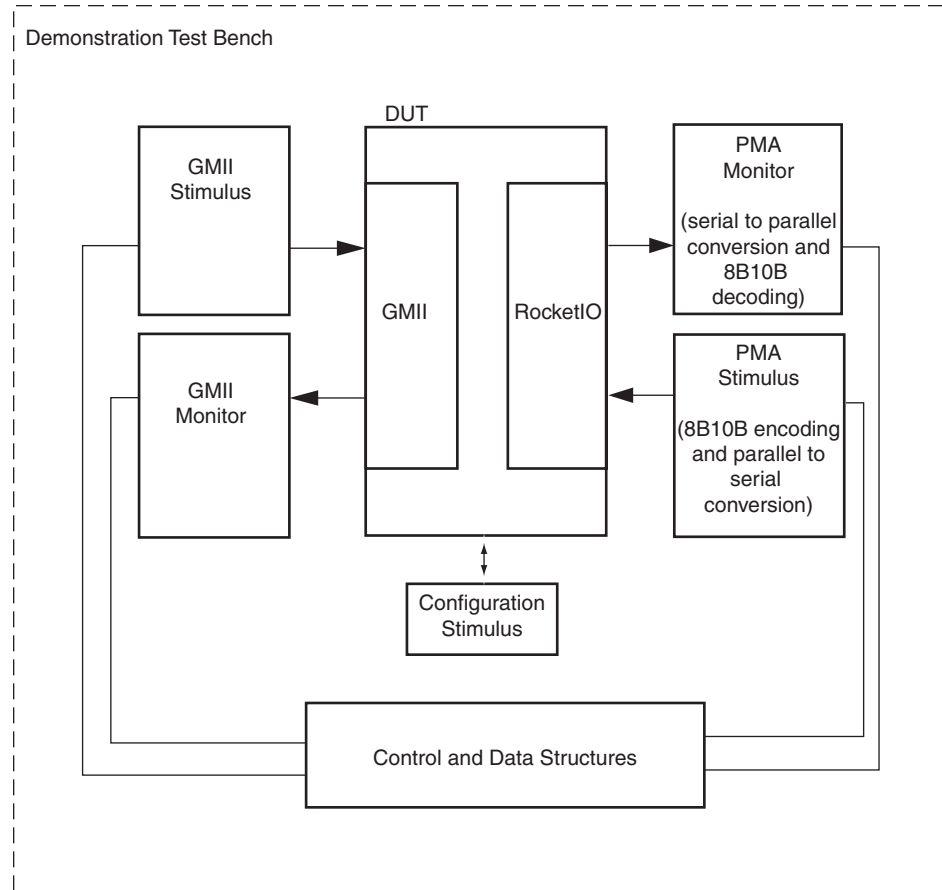


Figure 4-4: **Demonstration Test Bench Using RocketIO**

The following files describe the demonstration test bench.

### VHDL

```
project_dir>/<component_name>/simulation/demo_tb.vhd
```

### Verilog

```
project_dir>/<component_name>/simulation/demo_tb.v
```

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core itself.

### Core with MDIO Interface

The demonstration test bench performs the following tasks:

- Input clock signals are generated.
- A reset is applied to the example design.

- The Ethernet 1000BASE-X PCS/PMA core is configured through the MDIO interface by injecting an MDIO frame into the example design. This disables Auto-Negotiation (if present) and takes the core out of the Isolate state.
- Four frames are injected into the GMII transmitter by the GMII stimulus block.
  - the first frame is a minimum length frame
  - the second frame is a type frame
  - the third frame is an errored frame
  - the fourth frame is a padded frame
- The serial data received at the RocketIO transmitter interface is converted to 10-bit parallel data, then 8B10B decoded. The resultant frames are checked by the PMA Monitor against the stimulus frames injected into the GMII transmitter to ensure data integrity.
- The same four frames are generated by the PMA Stimulus block. These are 8B10B encoded, converted to serial data and injected into the RocketIO receiver interface.
- Data frames received at the GMII receiver are checked by the GMII Monitor against the stimulus frames injected into the RocketIO receiver to ensure data integrity.

## Core without MDIO Interface

The demonstration test bench performs the following tasks.

- Input clock signals are generated
- A reset is applied to the example design
- The Ethernet 1000BASE-X PCS/PMA core is configured using the Configuration Vector to take the core out of the Isolate state
- Four frames are injected into the GMII transmitter by the GMII stimulus block.
  - the first frame is a minimum length frame
  - the second frame is a type frame
  - the third frame is an errored frame
  - the fourth frame is a padded frame
- The serial data received at the RocketIO transmitter interface is converted to 10-bit parallel data, then 8B10B decoded. The resultant frames are checked by the PMA Monitor against the stimulus frames injected into the GMII transmitter to ensure data integrity.
- The same four frames are generated by the PMA Stimulus block. These are 8B10B encoded, converted to serial data and injected into the RocketIO receiver interface.
- Data frames received at the GMII receiver are checked by the GMII Monitor against the stimulus frames injected into the RocketIO receiver to ensure data is the same.

## Customizing the Test Bench

### Changing Frame Data

You can change the contents of the four frames used by the demonstration test bench by changing the *data* and *valid* fields for each frame defined. New frames can be added by defining a new frame of data. Modified frames are automatically updated in both stimulus and monitor functions.

## Changing Frame Error Status

Errors can be inserted into any of the predefined frames in any position by setting the “error” field to ‘1’ in any column of that frame. Injected errors are automatically updated in both stimulus and monitor functions.

## Changing the Core Configuration

The configuration of the Ethernet 1000BASE-X PCS/PMA core used in the demonstration test bench can be altered.

**Caution:** Certain configurations of the core will cause the test bench to fail or to cause processes to run indefinitely. For example, the demonstration test bench will not Auto-Negotiate with the example design. You determine the configurations that can safely be used with the test bench.

When the MDIO interface option is selected, the core can be reconfigured by editing the injected MDIO frame in the demonstration test bench. See the *Xilinx LogiCORE Ethernet 1000BASE-X PCS/PMA or SGMII User Guide* for more information on using the MDIO interface.

When the MDIO interface option is not selected, the core can be reconfigured by modifying the configuration vector directly. See the *Xilinx LogiCORE Ethernet 1000BASE-X PCS/PMA or SGMII User Guide* for information on using the configuration vector.

# Core Example Design with Ten-Bit Interface

## Example Design Top-Level HDL

Figure 4-5 illustrates the example design for a top-level HDL with a 10-bit interface (TBI).

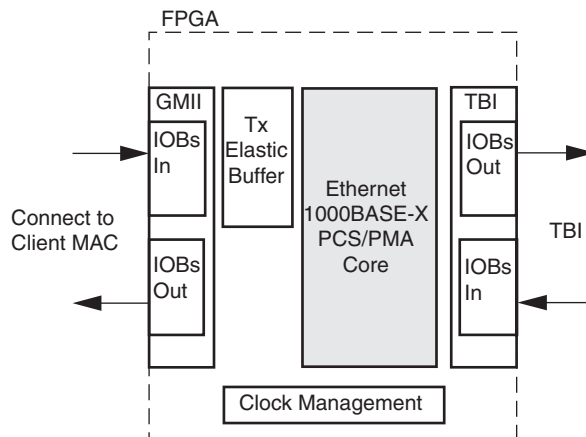


Figure 4-5: Example Design Top-Level HDL for the Ethernet 1000BASE-X PCS with TBI

The example design for the Ethernet 1000BASE-X PCS/PMA core with TBI is described in the following files:

### VHDL

```
project_dir>/<component_name>/example_design/<component_name>_top.vhd
```

## Verilog

```
project_dir>/<component_name>/example_design/<component_name>_top.v
```

The HDL example design contains the following:

- An instance of the Ethernet 1000BASE-X PCS/PMA core
- Clock management logic, including DCM and Global Clock Buffer instances, where required
- GMII interface logic, including IOB and DDR registers instances, where required
- A transmitter elastic buffer
- TBI interface logic, including IOB and DDR registers instances, where required
- Input and output buffers for other port signals of the top level

The example design HDL top level connects the GMII and the TBI of the core to external IOBs. This allows the functionality of the core to be demonstrated using a simulation package as described in this guide.

The example design can also be synthesized and placed on a suitable board and demonstrated in hardware, if required.

## Transmitter Elastic Buffer

The Transmitter Elastic Buffer is described in the following files:

### VHDL

```
project_dir>/<component_name>/example_design/tx_elastic_buffer.vhd
```

### Verilog

```
project_dir>/<component_name>/example_design/tx_elastic_buffer.v
```

When the GMII is used externally (as in this example design), the GMII transmit signals, (inputs to the core from a remote MAC at the other end of the interface) are synchronous to a clock which is likely to be derived from a different clock source to the core. For this reason, GMII transmit signals must be transferred into the core main clock domain before they can be used by the core. This is achieved with the Transmitter Elastic Buffer, an asynchronous FIFO implemented in distributed RAM. The operation of the elastic buffer is to attempt to maintain a constant occupancy by inserting or removing Idle sequences as necessary. This causes no corruption to the frames of data.

When the GMII is used as an internal interface, it is expected that the entire interface will be synchronous to a single clock domain and the Transmitter Elastic Buffer should be discarded. See the *Xilinx LogiCORE Ethernet 1000BASE-X PCS/PMA or SGMII User Guide* for information about connecting the core to an internal GMII (for example, an Ethernet MAC).

## Demonstration Test Bench

Figure 4-6 illustrates the demonstration test bench for the Ethernet 1000BASE-X PCS with TBI.

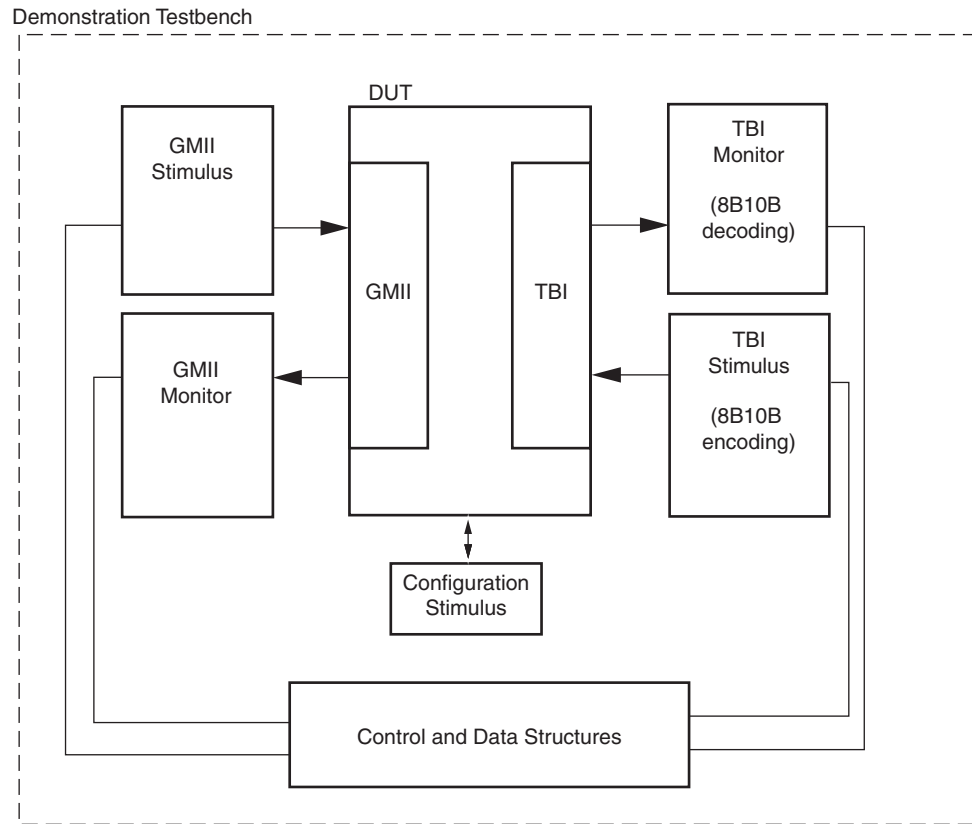


Figure 4-6: **Demonstration Test Bench for the Ethernet 1000BASE-X PCS with TBI**

The demonstration test bench is described in the following files:

### VHDL

```
project_dir>/<component_name>/simulation/demo_tb.vhd
```

### Verilog

```
project_dir>/<component_name>/simulation/demo_tb.v
```

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core itself.

## Core with MDIO Interface

The demonstration test bench performs the following tasks.

- Input clock signals are generated.
- A reset is applied to the example design.
- The Ethernet 1000BASE-X PCS/PMA core is configured through the MDIO interface by injecting an MDIO frame into the example design. This disables Auto-Negotiation (if present) and takes the core out of the Isolate state.

- The following frames are injected into the GMII transmitter by the GMII stimulus block.
  - the first is a minimum-length frame
  - the second is a type frame
  - the third is an errored frame
  - the fourth is a padded frame
- The data received at the TBI transmitter interface is 8B10B decoded. The resultant frames are checked by the TBI Monitor against the stimulus frames injected into the GMII transmitter to ensure data integrity.
- The same four frames are generated by the TBI Stimulus block. These are 8B10B encoded and injected into the TBI receiver interface.
- Data frames received at the GMII receiver are checked by the GMII Monitor against the stimulus frames injected into the TBI receiver to ensure data integrity.

## Core without MDIO Interface

The demonstration test bench performs the following tasks.

- Input clock signals are generated.
- A reset is applied to the example design.
- The Ethernet 1000BASE-X PCS/PMA core is configured via the Configuration Vector to take the core out of the Isolate state.
- The following frames are injected into the GMII transmitter by the GMII stimulus block.
  - the first is a minimum length frame
  - the second is a type frame
  - the third is an errored frame
  - the fourth is a padded frame
- The data received at the TBI transmitter interface is 8B10B decoded. The resultant frames are checked by the TBI Monitor against the stimulus frames injected into the GMII transmitter to ensure data is the same.
- The same four frames are generated by the TBI Stimulus block. These are 8B10B encoded and injected into the TBI receiver interface.
- Data frames received at the GMII receiver are checked by the GMII Monitor against the stimulus frames injected into the TBI receiver to ensure data is the same.

## Customizing the Test Bench

This section provides information about making modifications to the demonstration test bench files.

### Changing Frame Data

You can change the contents of the four frames used by the demonstration test bench by changing the *data* and *valid* fields for each frame defined in the test bench. Frames can be added by defining a new frame of data. Any modified frames are automatically updated in both stimulus and monitor functions.

## Changing Frame Error Status

Errors can be inserted into any of the predefined frames in any position by setting the “error” field to ‘1’ in any column of that frame. Injected errors are automatically updated in both stimulus and monitor functions.

## Changing the Core Configuration

The configuration of the Ethernet 1000BASE-X PCS/PMA core used in the demonstration test bench can be altered.

**Caution:** Certain configurations of the core can cause the test bench to fail, or to cause processes to run indefinitely. For example, the demonstration test bench will not auto-negotiate with the design example. Determine the configurations that can safely be used with the test bench.

If the MDIO interface option has been selected, the core can be reconfigured by editing the injected MDIO frame in the demonstration test bench. See the *Xilinx LogiCORE Ethernet 1000BASE-X PCS/PMA or SGMII User Guide* for more information about using the MDIO interface.

If the MDIO interface option has not been selected, the core can be reconfigured by modifying the configuration vector directly. See the *Xilinx LogiCORE Ethernet 1000BASE-X PCS/PMA or SGMII User Guide* for information about using the configuration vector.

# SGMII Example Design / Dynamic Switching Example Design

**Note:** This is the example design provided when the core is generated for the SGMII standard; it is also provided when the core is generated with the 1000BASE-X/SGMII dynamic switching capability.

## Example Design Top-Level HDL

Figure 4-7 illustrates an example design for top-level HDL for the Ethernet 1000BASE-X PCS/PMA or SGMII LogiCORE in SGMII mode.

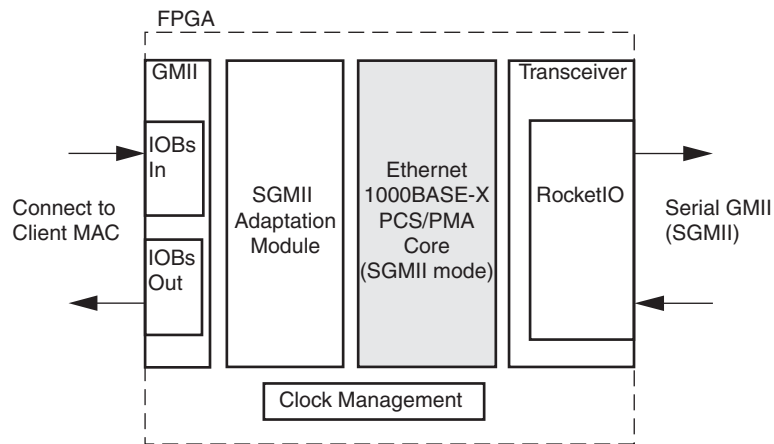


Figure 4-7: Top-Level HDL for the Ethernet 1000BASE-X PCS/PMA or SGMII LogiCORE in SGMII mode

The top-level example design for the Ethernet 1000BASE-X PCS/PMA core in SGMII mode is described in the following files:



## VHDL

```
project_dir>/<component_name>/example_design/<component_name>_top.vhd
```

## Verilog

```
project_dir>/<component_name>/example_design/<component_name>_top.v
```

The example design HDL top level contains the following:

- An instance of the Ethernet 1000BASE-X PCS/PMA core in SGMII mode.
- An instance of a Virtex-II Pro or Virtex-4 RocketIO transceiver.
- Clock management logic for the core and the RocketIO transceiver, including DCM and Global Clock Buffer instances.
- An SGMII adaptation module containing:
  - The clock management logic required to enable the SGMII example design to operate at 10 Mbps, 100 Mbps, and 1 Gbps
  - GMII logic for both transmitter and receiver paths; the GMII style 8-bit interface is run at 125 MHz for 1 Gbps operation; 12.5 MHz for 100 Mbps operation; 1.25 MHz for 10 Mbps operation.
- External GMII logic, including IOB and DDR register instances, where required.
- Input and output buffers for other port signals of the example design top level.

The example design HDL top-level connects the GMII of the core to the SGMII adaptation module and the PHY side interface of the core directly to a RocketIO instance. This allows the functionality of the core to be demonstrated using a simulation package, as described in this guide.

## Transceiver

A wrapper file for the Virtex-II Pro RocketIO or Virtex-4 Multi-Gigabit Transceiver is described in the following files:

### VHDL

```
project_dir>/<component_name>/example_design/transceiver.vhd
```

### Verilog

```
project_dir>/<component_name>/example_design/transceiver.v
```

This file instances a Virtex-II Pro RocketIO or Virtex-4 RocketIO and applies Gigabit Ethernet 1000BASE-X attributes to it. This transceiver wrapper is instantiated from the top-level HDL file for the example design.

For Virtex-4 FX devices only, a Calibration Block is required. See the [Calibration Block Users Guide](#) for more information.

This is described in the following files:

### VHDL

```
project_dir/<component_name>/example_design/cal_block_v1_2_1.vhd
```

### Verilog

```
project_dir/<component_name>/example_design/cal_block_v1_2_1.v
```

## SGMII Adaptation Module

The GMII of the core always operates at 125 MHz. The core makes no differentiation between the three speeds of operation; it always effectively operates at 1 Gbps. However, at 100 Mbps, every data byte run through the core should be repeated 10 times to achieve the required bit rate; at 10 Mbps, each data byte run through the core should be repeated 100 times to achieve the required bit rate. Dealing with this repetition of bytes is the function of the SGMII adaptation module.

The provided SGMII adaptation module (Figure 4-8) creates a GMII-style interface that clocks at a frequency of 125 MHz when operating at a speed of 1 Gbps (with no repetition of data bytes); 12.5 MHz at a speed of 100 Mbps (each data byte is repeated and run through the core 10 times); 1.25 MHz at a speed of 10 Mbps (each data byte is repeated and run through the core 100 times).

This GMII-style interface is not a standard interface (true GMII only operates at a clock frequency of 125 MHz), but it does allow a straightforward internal connection to an Ethernet MAC core. For example, the SGMII adaptation module can be used to interface the Ethernet 1000BASE-X PCS/PMA or SGMII LogiCORE, operating in SGMII mode, to the Xilinx Tri-Mode Ethernet MAC LogiCORE (see the *Xilinx LogiCORE Ethernet 1000BASE-X PCS/PMA or SGMII User Guide* for more information).

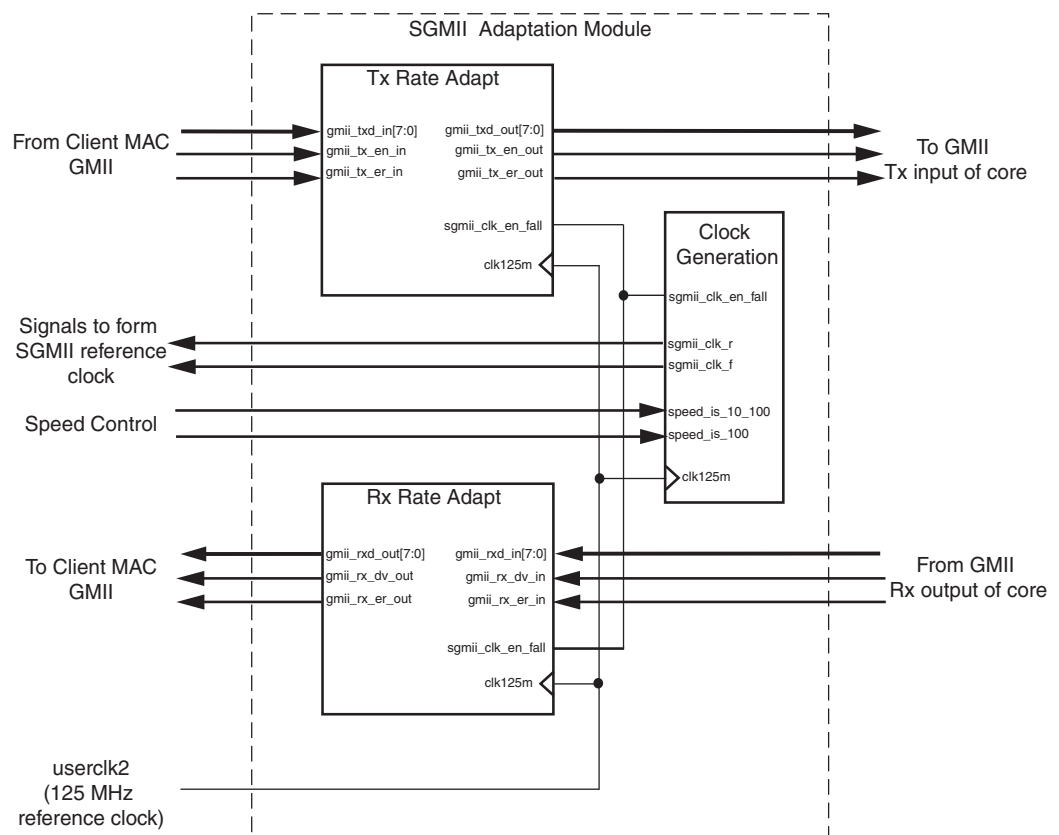


Figure 4-8: SGMII Adaptation Module

The top-level HDL for the SGMII adaptation module is described in the following files:

VHDL

```
project_dir>/<component_name>/example_design/sgmii_adapt/  
sgmii_adapt.vhd
```

### Verilog

```
project_dir>/<component_name>/example_design/sgmii_adapt/  
sgmii_adapt.v
```

The SGMII adaptation module is described in several hierarchical sub-modules as illustrated in [Figure 4-8](#). These sub-modules are described in separate HDL files as shown below.

Note that the 125 MHz reference clock (clk125m) used by the SGMII Adaptation logic is userclk2. This is the 125 MHz clock used to clock the Ethernet 1000BASE-X PCS/PMA or SGMII core and the clock routed to TXUSRCLK2 and RXUSRCLK2 of the RocketIO.

## Clock Generation

The clock generation module is described in the following files:

### VHDL

```
project_dir>/<component_name>/example_design/sgmii_adapt/clk_gen.vhd
```

### Verilog

```
project_dir>/<component_name>/example_design/sgmii_adapt/clk_gen.v
```

This file creates the necessary clocks and clock enables for use throughout the SGMII adaptation module. Clock frequencies are:

- 125 MHz at an operating speed of 1 Gbps
- 12.5 MHz at an operating speed of 100 Mbps
- 1.25 MHz at an operating speed of 10 Mbps

[Figure 4-9](#) illustrates the output clock and clock enable signals for the Clock Generation module at 1 Gbps and 100 Mbps speeds.

At 1 Gbps, sgmii\_clk\_r is fixed at logic 0; sgmii\_clk\_f is fixed at logic 1. sgmii\_clk\_r is connected to the rising edge triggered flip-flop of an IOB output DDR, clocked with clk125m. sgmii\_clk\_f is connected to the falling edge triggered flip-flop of the same IOB output DDR. The result is the production of an inverted clock, sgmii\_clk, that is forwarded off-chip. This IOB DDR output register is included in the top level HDL for the example design.

At 100 Mbps, the sgmii\_clk\_r and sgmii\_clk\_f signals toggle at the required clock frequency (every five clock periods of clk125m), also illustrated in [Figure 4-9](#). sgmii\_clk\_r is synchronous to the rising edge of the 125 MHz reference clock (clk125m); sgmii\_clk\_f is synchronous to the falling edge of the clk125m. These are routed to the rising and falling edges of the IOB DDR output register to forward the SGMII reference clock (sgmii\_clk) off chip.

At 10 Mbps, the situation is identical to that of 100 Mbps, with the exception that sgmii\_clk\_r and sgmii\_clk\_f toggle every 50 clock periods of clk125m.

sgmii\_clk\_en\_fall is used as a clock enable throughout the SGMII adaptation logic. At 1 Gbps it is fixed at logic 1 indicating that every clk125m period is significant. At 100 Mbps, this signal is valid for a single period of clk125m every ten clocks and marks the falling edge of the SGMII reference clock, sgmii\_clk. At 10 Mbps, this signal is valid for a single period of clk125m every one hundred clocks and again marks the falling edge

`sgmii_clk`. This clock enable signal is used as the control for the data byte repetition in the Transmitter and Receiver Rate Adaptation modules.

Figure 4-9 shows the clock generator output clocks and clock enable configurations.

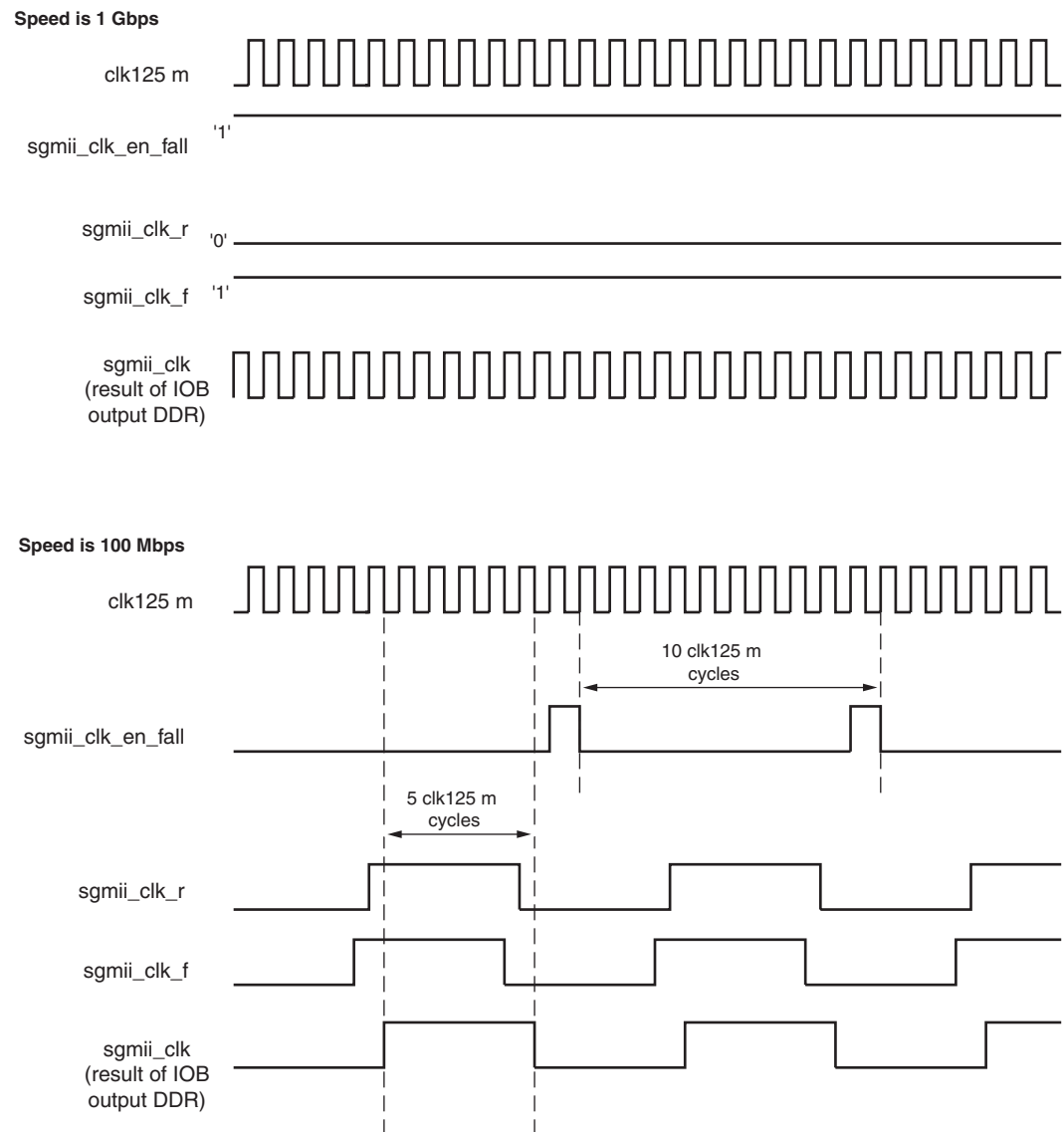


Figure 4-9: Clock Generator Output Clocks and Clock Enable

## Johnson Counter

The Johnson Counter is described in the following files:

### VHDL

```
project_dir>/<component_name>/example_design/sgmii_adapt/  
johnson_cntr.vhd
```

### Verilog

```
project_dir>/<component_name>/example_design/sgmii_adapt/  
johnson_cntr.v
```

The Johnson Counter is instantiated twice by the clock generation circuitry. The Johnson Counter is a shift register based clock divider that provides a divide-by-ten clock output. The divide-by-ten clock is output directly from a flip-flop triggered on the rising edge of the 125 MHz reference clock, `clk125m`.

Johnson Counter capabilities are extended by using the clock enables; it is only the clock-enabled cycles that trigger the shift register, and are therefore divided down.

## Transmitter Rate Adaptation Module

The Transmitter Rate Adaptation module is described in the following files:

### VHDL

```
project_dir>/<component_name>/example_design/sgmii_adapt/  
tx_rate_adapt.vhd
```

### Verilog

```
project_dir>/<component_name>/example_design/sgmii_adapt/  
tx_rate_adapt.v
```

This module accepts transmitter data from the GMII-style interface from the attached client MAC, and samples the input data on the 125 MHz reference clock, `clk125m`. This sampled data can then be connected directly to the input GMII of the Ethernet 1000BASE-X PCS/PMA or SGMII core. The 1 Gbps and 100 Mbps cases are illustrated in [Figure 4-10](#).

At 1 Gbps the client MAC should drive the GMII transmitter data synchronously to the rising edge of `clk125m`. `sgmii_clk_en_fall` (derived from the Clock Generation module) is fixed at logic 1 and the input data is sampled on every clock cycle.

At 100 Mbps and 10 Mbps, the client MAC should drive the GMII transmitter data synchronously to the rising edge of `sgmii_clk`. The data is sampled (see [Figure 4-10](#)) on the `sgmii_clk_en_fall` pulse. Since this pulse marks the falling edge of `sgmii_clk`, it guarantees that the data is stable when sampled. The frequency of the `sgmii_clk_en_fall` pulse ensures that this data is repeated exactly 10 times when operating at a speed of 100 Mbps and 100 times when operating at a speed of 10 Mbps.

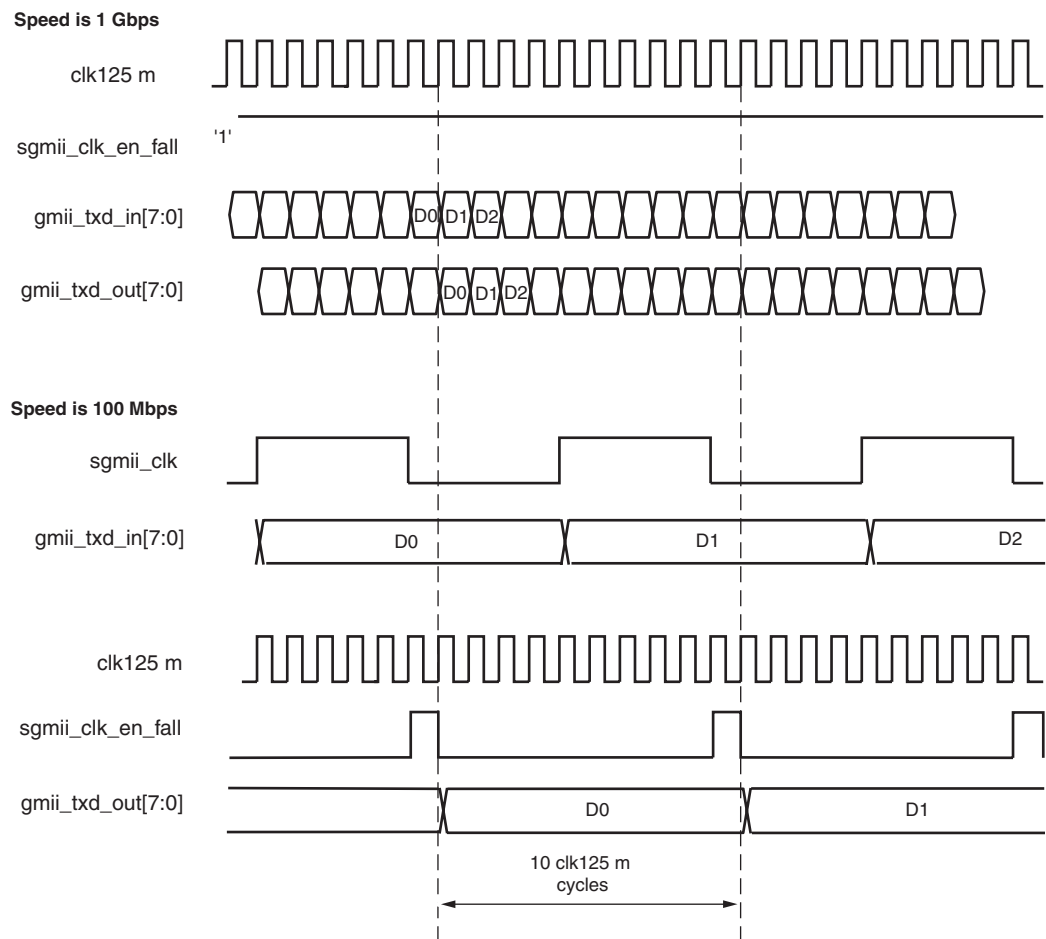


Figure 4-10: Transmitter Rate Adaptation Module Data Sampling

## Receiver Rate Adaptation Module

The Receiver Rate Adaptation module is described in the following files:

### VHDL

```
project_dir>/<component_name>/example_design/sgmii_adapt/  
rx_rate_adapt.vhd
```

### Verilog

```
project_dir>/<component_name>/example_design/sgmii_adapt/  
rx_rate_adapt.v
```

This module accepts received data from the Ethernet 1000BASE-X PCS/PMA or SGMII core. This data is sampled and sent out of the GMII receiver interface for the attached client MAC. The 1 Gbps and 100 Mbps cases are illustrated in [Figure 4-11](#).

At 1 Gbps, the data is valid on every clock cycle of the 125 MHz reference clock (clk125m). Data received from the core is clocked straight through the Receiver Rate Adaptation module.

At 100 Mbps, the data is repeated for a 10 clock period duration of `clk125m`; at 10 Mbps, the data is repeated for a 100 clock period duration of `clk125m`. The Receiver Rate Adaptation Module samples this data on the `sgmii_clk_en_fall` signal produced from the Clock Generation circuitry. Since this pulse marks the falling edge of `sgmii_clk`, it guarantees that setup and hold time is provided for the attached client MAC.

The Receiver Rate Adaptation module also performs a second function that accounts for the latency inferred in Figure 4-11. The 8-bit Start of Frame Delimiter (SFD) code is detected, and if required, it is realigned across the 8-bit data path of `gmii_rxd_out[7:0]` before being presented to the attached client MAC. It is possible that this SFD could have been skewed across two separate bytes by MACs operating on a 4-bit data path.

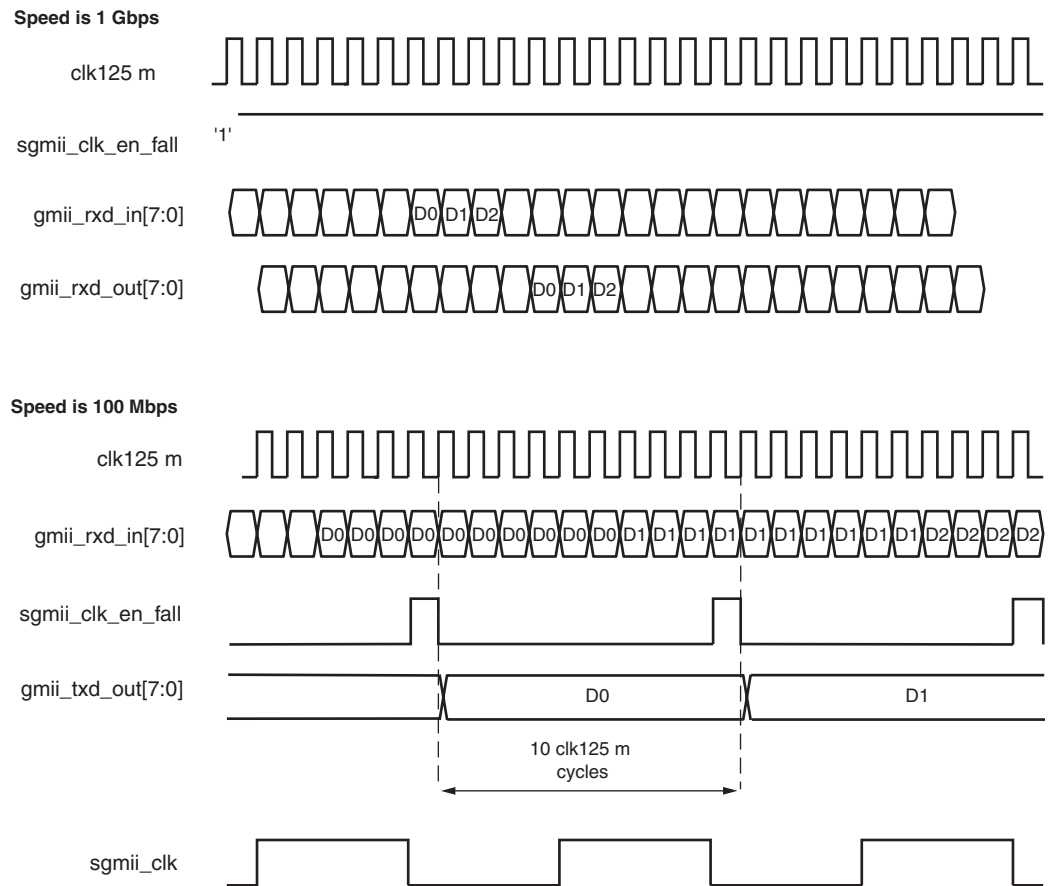


Figure 4-11: Receiver Rate Adaptation Module Data Sampling



## Demonstration Test Bench

Figure 4-12 illustrates the demonstration test bench for the Ethernet 1000BASE-X PCS/PMA or SGMII Core in SGMII mode.

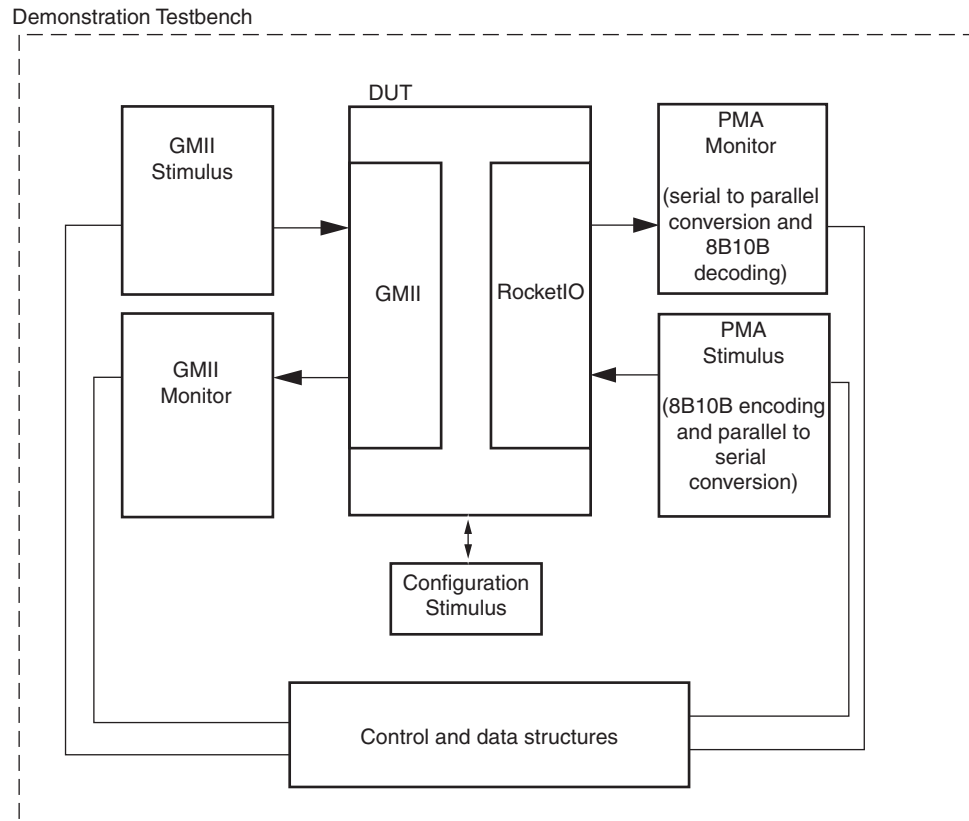


Figure 4-12: **Demonstration Test Bench for the Ethernet 1000BASE-X PCS/PMA or SGMII Core in SGMII Mode**

The demonstration test bench is described in the following files:

### VHDL

```
project_dir>/<component_name>/simulation/demo_tb.vhd
```

### Verilog

```
project_dir>/<component_name>/simulation/demo_tb.v
```

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core itself. The demonstration test bench performs the following tasks.

- Input clock signals are generated.
- A reset is applied to the example design.
- The Ethernet 1000BASE-X PCS/PMA core is configured through the MDIO interface by injecting an MDIO frame into the example design. This disables Auto-Negotiation and takes the core out of Isolate state.
- The following frames are injected into the GMI transmitter by the GMI stimulus block at 1 Gbps.

- the first is a minimum length frame
- the second is a type frame
- the third is an errored frame
- the fourth is a padded frame
- The serial data received at the RocketIO transmitter interface is converted to 10-bit parallel data, then 8B10B decoded. The resultant frames are checked by the PMA Monitor against the stimulus frames injected into the GMII transmitter to ensure data integrity.
- The same four frames are generated by the PMA Stimulus block. These are 8B10B encoded, converted to serial data and injected into the RocketIO receiver interface at 1 Gbps.
- Data frames received at the GMII receiver are checked by the GMII Monitor against the stimulus frames injected into the RocketIO receiver to ensure data integrity.

## Customizing the Test Bench

### Changing Frame Data

You can change the contents of the four frames used by the demonstration test bench by changing the *data* and *valid* fields for each frame defined in the test bench. New frames can be added by defining a new frame of data. Modified frames are automatically updated in both stimulus and monitor functions.

### Changing Frame Error Status

Errors can be inserted into any of the predefined frames in any position by setting the “error” field to ‘1’ in any column of that frame. Injected errors are automatically updated in both stimulus and monitor functions.

### Changing the Core Configuration

The configuration of the Ethernet 1000BASE-X PCS/PMA core used in the demonstration test bench can be altered.

**Caution:** Certain configurations of the core cause the test bench to fail, or to cause processes to run indefinitely. For example, the demonstration test bench will not Auto-Negotiate with the design example. Determine the configurations that can safely be used with the test bench.

The core can be reconfigured by editing the injected MDIO frame in the demonstration test bench. See the *Xilinx LogiCORE Ethernet 1000BASE-X PCS/PMA or SGMII User Guide* for information about using the MDIO interface.

### Changing the Operational Speed

SGMII can be used to carry ethernet traffic at 10 Mbps, 100 Mbps or 1 Gbps. By default, the demonstration test bench is configured to operate at 1 Gbps. The speed of both the example design and test bench can be set to the desired operational speed by editing the following settings, recompiling the test bench, then running the simulation again.

#### 1 Gbps operation

```
set speed_is_10_100 to logic 0
```

### 100 Mbps operation

```
set speed_is_10_100 to logic 1  
set speed_is_100 to logic 1
```

### 10 Mbps operation

```
set speed_is_10_100 to logic 1  
set speed_is_100 to logic 0
```

