LogiCORE™ 10-Gigabit Ethernet MAC

Getting Started Guide

UG146 July 13, 2006

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/30/04	1.0	Initial Xilinx release.
4/28/05	1.5	Updated to core version 6.0 and Xilinx tools 7.1i.
01/18/06	1.6	Updated tools version, release version and date.
7/13/06	2.0	Updated core version to 8.0; Xilinx tools to 8.2i.

Table of Contents

Chapter 4: Detailed Example Design

Schedule of Figures

About This Guide

The *10-Gigabit Ethernet Mac v8.0 Getting Started Guide* provides information about generating a Xilinx® LogiCORE™ 10-Gigabit Ethernet MAC core, customizing and simulating the core utilizing the provided example design, and running the design files through implementation using the Xilinx tools.

Guide Contents

This guide contains the following chapters:

- Preface, "About This Guide" introduces the user to the organization and purpose of the design guide, a list of additional resources and the conventions used in this document.
- Chapter 1, "Introduction" introduces the core and provides references for related material, contact information, technical support, and providing feedback to Xilinx.
- Chapter 2, "Installing and Licensing the Core" describes how to install the core and how to obtain and install a license file for 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 http://www.xilinx.com/support/techsup/tutorials/index.htm
Answer Browser	Database of Xilinx solution records http://www.xilinx.com/xlnx/xil_ans_browser.jsp
Application Notes	Descriptions of device-specific design techniques and approaches http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=Application+Notes

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

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 you enter in a syntactical statement	**`ngdbuild`** *`design_name`*
Italic font	Variables in a syntax statement for which you must supply values	See the *Development System Reference Guide* for more information.
	References to other manuals	See the *User Guide* for details.
	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}`**

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	**`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`.
	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 "Additional Resources" for details. See "Title Formats" in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-II Handbook.*
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Chapter 1

Introduction

The 10-Gigabit Ethernet MAC core is a fully verified solution design that supports Verilog-HDL and VHDL. In addition, the example design in this guide is provided in both Verilog and VHDL.

This chapter introduces the 10-Gigabit Ethernet MAC core and provides related information, including recommended design experience, additional resources, technical support, and submitting feedback to Xilinx.

About the Core

The 10-Gigabit Ethernet MAC 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/10gmac/index.htm. For information about system requirements, installation, and licensing options, see Chapter 2, "Installing and Licensing the Core."

Recommended Design Experience

Although the 10-Gigabit Ethernet MAC core is a fully verified solution, the challenge associated with implementing a complete design varies depending on the configuration and functionality of the application. For best results, previous experience building high performance, pipelined FPGA designs using Xilinx implementation software and user constraints files (UCF) is recommended.

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

Additional Core Resources

For detailed information and updates about the 10-Gigabit Ethernet MAC core, see the following documents, located on the 10-Gigabit Ethernet MAC product page at http://www.xilinx.com/systemio/10gmac/index.htm

- *10-Gigabit Ethernet MAC Core Data Sheet*
- *10-Gigabit Ethernet MAC Core Release Notes*
- *10-Gigabit Ethernet MAC Core User Guide*

For updates to this document, see the *10-Gigabit Ethernet MAC Core Getting Started Guide*, also located on the 10-Gigabit Ethernet MAC product page.

Technical Support

To obtain technical support specific to the 10-Gigabit Ethernet MAC core, visit http://support.xilinx.com/. Questions are routed to a team of engineers with expertise using the 10-Gigabit Ethernet MAC core.

Xilinx will provide technical support for use of this product as described in the *10-Gigabit Ethernet MAC User Guide* and the *10-Gigabit Ethernet MAC 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 10-Gigabit Ethernet MAC core and the documentation supplied with the core.

Core

For comments or suggestions about the 10-Gigabit Ethernet MAC core, please submit a WebCase from www.xilinx.com/support/clearexpress/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 http://www.xilinx.com/support/clearexpress/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 10-Gigabit Ethernet MAC core and obtaining a license for the core, which you must do before using the core in your designs. The 10-Gigabit Ethernet MAC 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. Purchase of the core entitles you to technical support and access to updates for a period of one year.

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

- ISE™ 8.2i with applicable Service Pack

Check the release notes for the required Service Pack; ISE Service Packs can be downloaded from www.xilinx.com/xlnx/xil_sw_updates_home.jsp?update=sp.

Before you Begin

Before installing the core, you must have a Xilinx.com account and the ISE 8.2i 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 8.2i software and the applicable Service Pack software.

Installing the Core

You can install the core in two ways—using the CORE Generator IP Software Update option to select from a list of updates, or by performing a manual installation after downloading the core from the web.

Using the CORE Generator Software Update 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 > Software Update. The WebUpdate screen appears.
3. If you are behind a firewall, click Set Proxy to either verify or set your proxy host and port settings.
4. Click Check for Updates. The Software Update installer appears.
5. Select the ISE 8.2IP Update 1 option; then click Install Selected. Informational messages may appear indicating that additional installations are required.
6. Click OK to accept any messages and continue. The User Login dialog box appears.
7. Enter your login name and password; then click OK. The selected update products are downloaded and installed.
8. To confirm the installation, check the following file:
   `C:\Xilinx\coregen\install\install_history`.

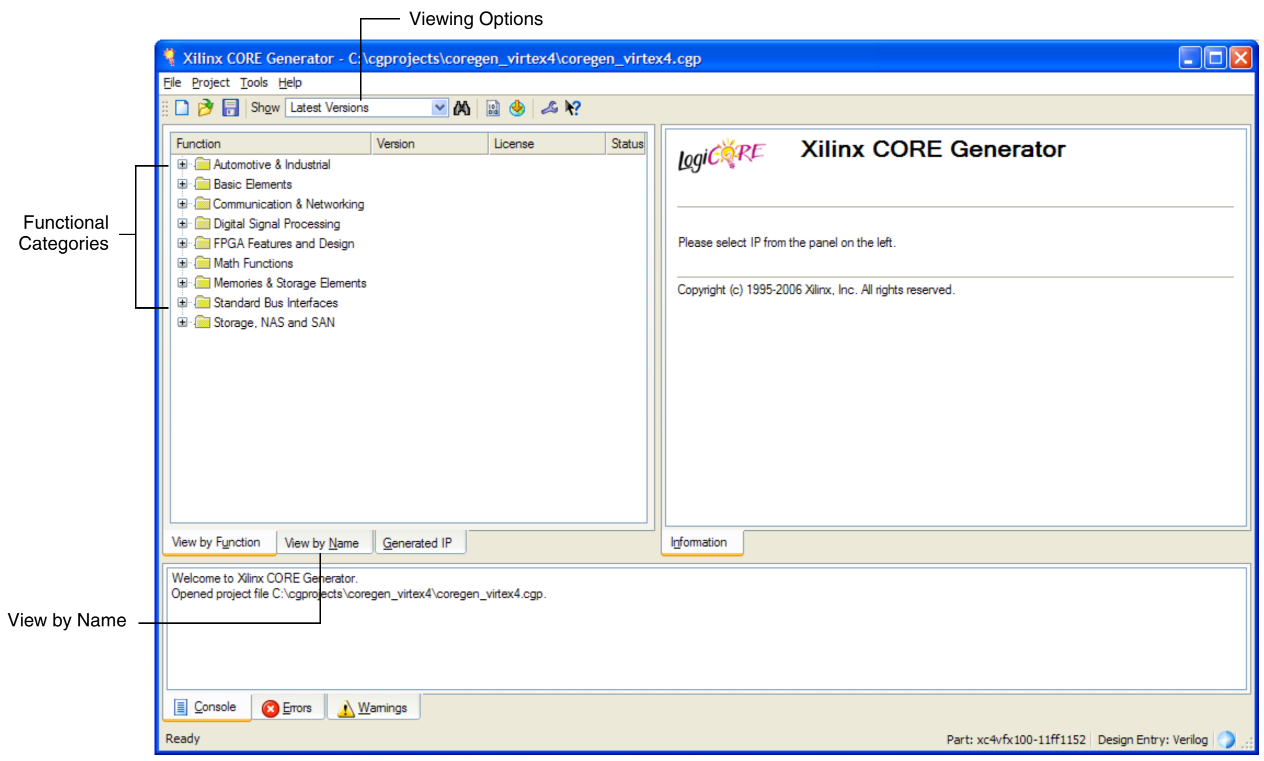
   Note that this step assumes your Xilinx software is installed in C:\Xilinx.

Manually

1. Close the CORE Generator if it is running.
2. Download the IP Update ZIP file from the following location and save it to a temporary directory: www.xilinx.com/support/download.htm.
3. Unpack the ZIP files using either WinZip (Windows) or Unzip (UNIX).
4. Extract the **ise_82i_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_82i_ip_update1.zip
     ```
   - **Linux**. From a UNIX shell, type the following:
     ```
     $XILINX/bin/lin/unzip -d $XILINX ise_82i_ip_update1.zip
     ```
   - **Solaris**. From a UNIX shell, type the following:
     ```
     $XILINX/bin/sol/unzip -d $XILINX ise_82i_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 your 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.



Please select IP from the panel on the left.

Copyright (c) 1995-2006 Xilinx, Inc. All rights reserved.

Welcome to Xilinx CORE Generator.
Opened project file C:\cgprojects\coregen_virtex4\coregen_virtex4.cgp.

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. To determine that the installation is successful, be sure that the new core or cores appear in the CORE Generator GUI.

   For additional assistance installing the IP Update, contact www.xilinx.com/support.

License Options

The 10-Gigabit Ethernet MAC core provides three licensing options. After installing the core, choose a license option, as applicable.

Simulation Only

The Simulation Only Evaluation license is provided by default with the Xilinx CORE Generator and does not require an electronic license file. This license lets you assess the core functionality with either the provided example design or alongside your own design and demonstrates the various interfaces on the core in simulation. (Functional simulation is supported by a structural model generated by the CORE Generator.)

Full System Hardware Evaluation

The Full System Hardware Evaluation license is available at no cost and lets you fully integrate the core into an FPGA design, place and route the design, evaluate timing, and perform back-annotated gate-level simulation of the core using the demonstration test bench provided.

In addition, the license lets you generate a bitstream from the placed and routed design, which can then be downloaded to a supported device and tested in hardware. The core can be tested in the target device for a limited time before *timing out* (ceasing to function) at which time it can be reactivated by reconfiguring the device.

You can obtain the Full System Evaluation license in one of the following ways, depending on the core:

- By registering on the Xilinx IP Evaluation page and filling out a form to request an automatically generated evaluation license
- By contacting your local Xilinx FAE to request a Full System Hardware Evaluation license key

Click Evaluate on the core's product page for information about how to obtain a Full System Hardware Evaluation license.

Full

The Full license is provided when you purchase the core and 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

Obtaining Full System Hardware Evaluation License

To obtain a Full System Hardware Evaluation license, do the following:

- Navigate to the 10-Gigabit Ethernet MAC core product page: http://www.xilinx.com/systemio/10gmac/index.htm
- Click Evaluate; then click Full System Hardware Evaluation.
- Follow the onscreen instructions to both download the CORE Generator files (delivered as an IP Update) and satisfy any additional requirements associated with the license.

Obtaining a Full License

To obtain a Full license, you must purchase the core. After purchase, you will receive a letter containing a serial number, which is used to register for access to the *lounge*, a secured area of the 10-Gigabit Ethernet MAC core product page.

- From the product page, click Register to register and request access to the lounge.
- Xilinx will review your access request and typically grants access to the lounge in 48 hours. (Contact Xilinx Customer Service if you need faster turnaround.)
- After you receive confirmation of lounge access, click Access Lounge on the 10-Gigabit Ethernet MAC core product page and log in.
- Follow the instructions in the lounge to fill out the license request form; then click Submit to automatically generate the license. An e-mail containing the license and installation instructions will be sent to you immediately.

Installing Your License File

After selecting either the Full System Hardware Evaluation or Full license option, you will receive an email containing instructions for installing your license. In addition, the email provides information about advanced licensing options and technical support.

Chapter 3

Quick Start Example Design

The quick start instructions let you quickly generate a 10-Gigabit Ethernet MAC core, run the design through implementation with the Xilinx tools, and simulate the example design utilizing the provided demonstration test bench. For detailed information about the example design, see Chapter 4, "Detailed Example Design."

Introduction

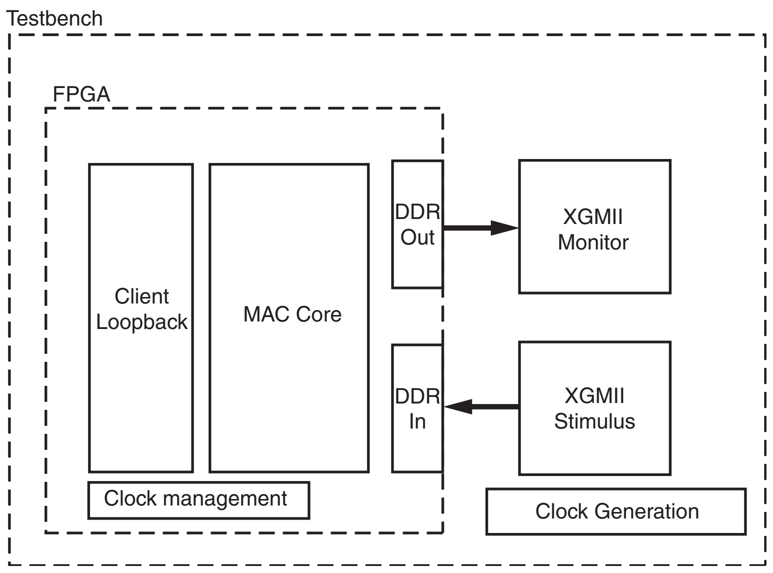


Figure 3-1: **Example Design and Test Bench**

The 10-Gigabit Ethernet MAC example design consists of the following

- The 10-Gigabit Ethernet MAC core netlist
- An example HDL wrapper/top level
- A *ping* client-side loopback circuit including asynchronous FIFO that returns received frames on the transmit port
- A demonstration test bench to exercise the example design

The 10-Gigabit Ethernet MAC example design has been tested with Xilinx ISE 8.2i, Mentor ModelSim® v6.1e, and Cadence NC-Sim IUS 5.5

Generating the Core

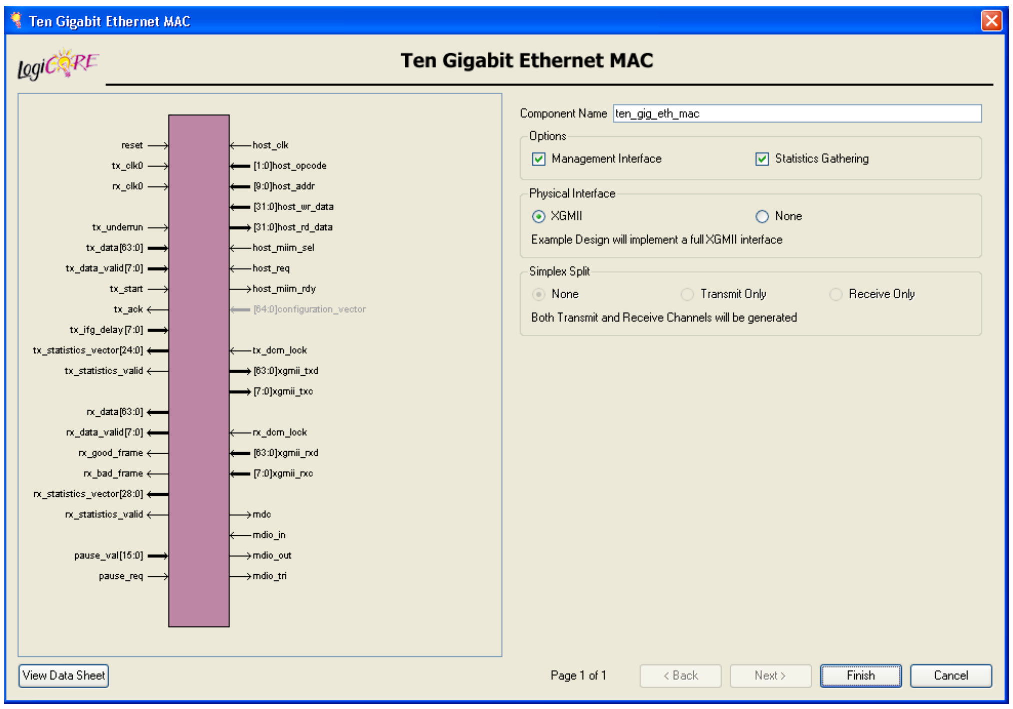
To begin working with the example design, start by generating the core with the default settings.

To generate the core:

1. Start the CORE Generator.

   For general help with starting and using the CORE Generator on your system, please see the documentation supplied with ISE.

2. Create a new project.
3. In the Project Options dialog box, select a silicon family that supports the 10-Gigabit Ethernet MAC core for example, Virtex™-II Pro.
4. In the Generate section of the Project Options, select either VHDL or Verilog, and then select Other for the Vendor.
5. Locate the 10-Gigabit Ethernet MAC core in the taxonomy tree, displayed under Communications & Networking/Ethernet.
6. Double-click the core.
7. A dialog box warning of the limitations of the Simulation Only Evaluation license may appear; click OK to continue. The 10-Gigabit Ethernet MAC customization screen appears.



Component Name: ten_gig_eth_mac

Options: [✓ Management Interface] [✓ Statistics Gathering]

Physical Interface: [✓ XGMII] None

Example Design will implement a full XGMII interface

Simplex Split: [✓ None] Transmit Only Receive Only

Both Transmit and Receive Channels will be generated

Figure 3-2: **10-Gigabit Ethernet MAC Customization Screen**

8. In the Component Name field, enter a name for the core instance. For this chapter, the name *quickstart* is used.
9. Accept the default settings; then click Finish to generate the core.

The core and its supporting files, including the example design, are generated in your project directory. For a detailed description of the design example files and directories, see Chapter 4, "Detailed Example Design."

Implementation

After the core is successfully generated, the netlist and example design HDL wrapper can be processed through the Xilinx implementation toolset. Included in the generated outputs are several scripts to assist in this processing.

To implement the example design:

Open a command prompt or shell in your project directory, then enter the following commands for either UNIX or Windows platforms:

For UNIX:

```
unix-shell> cd quickstart/implement
unix-shell> ./implement.sh
```

For Windows:

```
ms-dos> cd quickstart\implement
ms-dos> implement.bat
```

This starts a script that synthesizes the example design HDL wrapper, builds, maps, and places-and-routes the example design, and then creates gate-level netlist HDL files in both VHDL and Verilog with associated timing information (SDF) files. Finally these files are copied into the test area directories. Note that this process occurs only when using the Full System Hardware Evaluation or Full licenses.

Simulation

The example design provided with the Ten Gigabit Ethernet MAC core provides a complete environment which allows the user to simulate the core and view the outputs. Scripts are provided for pre and post layout simulation. The simulation model will either be in VHDL or Verilog depending on the CORE Generator Design Entry project option.

Setting up for Simulation

The Xilinx UniSim and SimPrim libraries must be mapped into the simulator. If the UniSim and SimPrim libraries are not set up for your environment, go to Answer Record 15338 on www.xilinx.com/support for assistance compiling Xilinx simulation models and setting up the simulator environment.

Pre-implementation Simulation

To run a functional simulation of the example design:

1. Open a command prompt or shell in your project directory, then set the current directory to:

   ```
   quickstart/simulation/functional
   ```

2. Launch the simulation script:

   ```
   modelSim: vsim -do simulate_mti.do
   nc-sim: ./simulate_ncsim.sh
   ```

The simulation script compiles the functional model and the demonstration test bench, adds some 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.

Post-implementation Simulation

To run a timing simulation of the example design:

1. Open a command prompt or shell in your project directory, then set the current directory to:

   ```
   quickstart/simulation/timing
   ```

2. Launch the simulation script:

   ```
   modelSim: vsim -do simulate_mti.do
   nc-sim: ./simulate_ncsim.sh
   ```

The simulation script compiles the gate-level model and the demonstration test bench, adds some 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?

To begin using the 10-Gigabit Ethernet MAC core in your own design, see the *10-Gigabit Ethernet MAC User Guide*.

Detailed Example Design

This chapter describes the design example in detail, including the files and directory structure generated by the CORE Generator, the purpose and contents of the implementation scripts, the contents of the example HDL wrappers, and the operation of the demonstration test bench.

Directory Structure and File Descriptions

VHDL Design Flow

Figure 4-1 displays the files and directories created by CORE Generator for a VHDL based project. <*project_dir*> is the CORE Generator project directory; <*component_name*> is the component name as entered in the 10-Gigabit Ethernet MAC core customization screen.

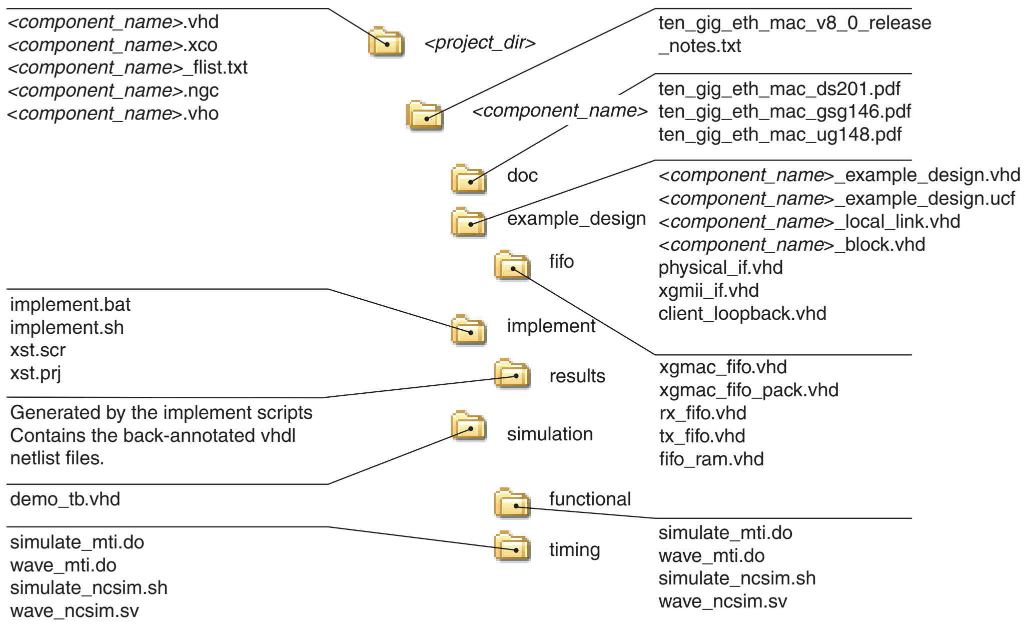


Figure 4-1: **VHDL Directory Structure**

Files and directories created by the CORE Generator are defined below. <*project_dir*> represents the CORE Generator project directory; <*component_name*> represents the component name entered on the 10-Gigabit Ethernet MAC core customization screen.

Note: The implement and timing simulation directories are only present when the core is generated with a full or a hardware evaluation license.

<project_dir>

- `<component_name>.ngc`

  A binary Xilinx implementation netlist. Describes how the core is to be implemented. Used as input to the Xilinx Implementation Tools.
- `<component_name>.vhd`

  VHDL structural simulation model. File used to support VHDL functional simulation of a core. The VHDL model passes customized parameters to the generic core simulation model.
- `<component_name>.xco`

  As an output file, the XCO file is a log file which records the settings used to generate a particular 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`

  As an output file, 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 text file listing all of the output files produced when customized core was generated in the CORE Generator.
- `<component_name>.vho`

  This contains a VHDL template for the core. This can be copied into the user design.

<project_dir>/<component_name>

Contains the 10-Gigabit Ethernet MAC v8.0 release notes.

- ten_gig_eth_mac_v8_0_release_notes.txt

<project_dir>/<component_name>/doc

This directory contains the 10-Gigabit Ethernet MAC documentation.

- `ten_gig_eth_mac_ds201.pdf`

  Contains the *10-Gigabit Ethernet MAC Data Sheet*.
- `ten_gig_eth_mac_gsg146.pdf`

  Contains the *10-Gigabit Ethernet MAC Getting Started Guide*.
- `ten_gig_eth_mac_ug148.pdf`

  Contains the *10-Gigabit Ethernet MAC User Guide*.

<project_dir>/<component_name>/example_design

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

- `<component_name>_example_design.vhd`

  The example design level VHDL file for the example design. The local link block is instanced along with the Address Swap block, if required.
- `<component_name>_example_design.ucf`

The UCF for the core and the example design.

- `<component_name>_local_link.vhd`

  The local link level VHDL file. For cores without a simplex split this will instance the local link fifo in addition to the block level.

- `<component_name>_block.vhd`

  The block level VHDL file. This instances the appropriate interface block and the MAC core.

- `address_swap.vhd`

  The address swap VHDL file. This connects to the local link interface and swaps the destination and source addresses of frame.

- `xgmii_if.vhd`

  The Xgmii interface VHDL file. This contains the DDR registers to implement the enternal XGMII interface. See "10-Gigabit Ethernet MAC with External XGMII Interface," page 33.

- `physical_if.vhd`

  The Phy interface VHDL file. This contains the SDR registers to implement the 64-bit interface. See "10-Gigabit Ethernet MAC with 64-bit SDR Interface," page 35.

- `client_loopback.vhd`

  The client loopback design example instanced in the local link level.

<project_dir>/<component_name>/example_design/fifo

This directory contains the files for the FIFO that is instanced in the client_loopback example design.

- `xgmac_fifo.vhd`

  The top level of the FIFO.

- `xgmac_fifo_pack.vhd`

  Component declarations for the FIFO.

- `rx_fifo.vhd`

  The local link receive fifo wrapper. This implements the interface to the MAC receiver including frame dropping logic and has a local link interface for the receive channel.

- `tx_fifo.vhd`

  The local link transmit fifo. This implements the interface to the MAC transmitter including logic to initiate transmission when a full frame is stored in the FIFO andhas local link interface.

- `fifo_ram.vhd`

  This is a BRAM wrapper used by both FIFOs.

<project_dir>/<component_name>/implement

- `implement.sh`

  A UNIX shell script that processes the example design through the Xilinx tool flow.

- `implement.bat`

  A Windows batch file that processes the example design through the Xilinx tool flow.

- `xst.prj`

  The XST project file for the example design; it enumerates all the HDL files that need to be synthesized.
- `xst.scr`

  The XST script file for the example design.

<project_dir>/<component_name>/implement/results

This directory is produced by the implement scripts and is used to run the example design files and the `<component_name>.ngc` file through the Xilinx implementation tools. Once these are run it contains the following files for timing simulation.

- `routed.vhd`

  The back-annotated SimPrim based VHDL design. Used for timing simulation.
- `routed.sdf`

  The timing information for simulation is contained in this file.

<project_dir>/<component_name>/simulation

- `demo_tb.vhd`

  This file is the VHDL demonstration test bench for the 10-Gigabit Ethernet MAC core. More information can either be found in "10-Gigabit Ethernet MAC with External XGMII Interface," page 33 or "10-Gigabit Ethernet MAC with 64-bit SDR Interface," page 35.

<project_dir>/<component_name>/simulation/functional

- `simulate_mti.do`

  A ModelSim macro file that compiles the example design sources and the structural simulation model then runs the functional simulation to completion.
- `wave_mti.do`

  A ModelSim macro file that opens a wave windows and adds interesting signals to it. It is called used by the simulate_mti.do macro file.
- `simulate_ncsim.sh`

  A UNIX shell script that compiles the example design sources and the structural simulation model then runs the functional simulation to completion using NC-Sim.
- `wave_ncsim.sv`

  A NC-Sim macro file that opens a wave windows and adds interesting signals to it. It is called used by the simulate_ncsim.sh script.

<project_dir>/<component_name>/simulation/timing

- `simulate_mti.do, simulate_ncsim.sh`

  A ModelSim macro file that compiles the VHDL timing model and demo test bench then runs the timing simulation to completion.
- `wave_mti.do`

  A ModelSim macro file that opens a wave windows and adds interesting signals to it. It is called used by the simulate_mti.do macro file.

- `simulate_ncsim.sh`

  A UNIX shell script that compiles the example design sources and the timing model then runs the functional simulation to completion using NC-Sim.

- `wave_ncsim.sv`

  A NC-Sim macro file that opens a wave windows and adds interesting signals to it. It is called used by the simulate_ncsim.sh script.

Verilog Design Flow

Figure 4-2 shows the files and directories created by CORE Generator for a VHDL based project. <*project_dir*> is the CORE Generator project directory; <*component_name*> is the component name as entered in the 10-Gigabit Ethernet MAC core customization dialog box.

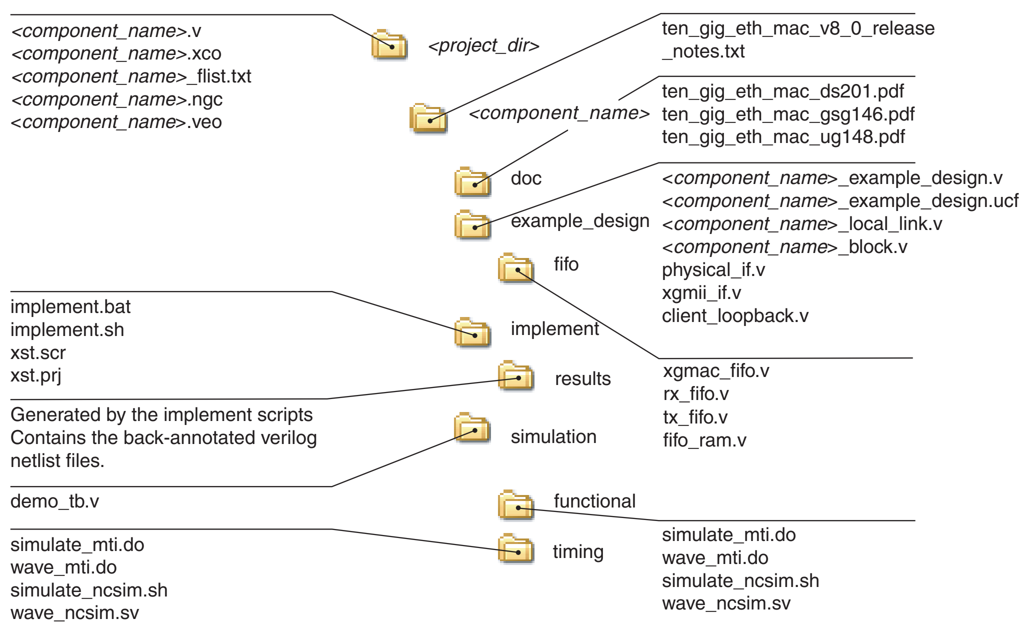


Figure 4-2: **Verilog Directory Structure**

Files and directories created by the CORE Generator are described below. <*project_dir*> represents the CORE Generator project directory; <*component_name*> represents the component name entered on the 10-Gigabit Ethernet MAC core customization screen.

The implement and timing simulation directories are only present when the core is generated with a full or a hardware evaluation license.

<project_dir>

- `<component_name>.ngc`

  A binary Xilinx implementation netlist. Describes how the core is to be implemented. Used as input to the Xilinx Implementation Tools.

- `<component_name>.v`

  Verilog structural simulation model. File used to support Verilog functional simulation of a core. The Verilog model passes customized parameters to the generic core simulation model.

- `<component_name>.xco`

  As an output file, the XCO file is a log file which records the settings used to generate a particular 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`

As an output file, 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 text file listing all of the output files produced when customized core was generated in the CORE Generator.

- `<component_name>.veo`

  This contains a Verilog template for the core. This can be copied into the user design.

<project_dir>/<component_name>

Contains the 10-Gigabit Ethernet MAC v8.0 release notes.

- ten_gig_eth_mac_v8_0_release_notes.txt

<project_dir>/<component_name>/doc

This directory contains the 10-Gigabit Ethernet MAC documentation.

- `ten_gig_eth_mac_ds201.pdf`

  Contains the *10-Gigabit Ethernet MAC Data Sheet*.

- `ten_gig_eth_mac_gsg146.pdf`

  Contains the *10-Gigabit Ethernet MAC Getting Started Guide*.

- `ten_gig_eth_mac_ug148.pdf`

  Contains the *10-Gigabit Ethernet MAC User Guide*.

<project_dir>/<component_name>/example_design

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

- `<component_name>_example_design.v`

  The example design level Verilog file for the example design. Depending on the core customization, more information can either be found in "10-Gigabit Ethernet MAC with External XGMII Interface," page 33 or in "10-Gigabit Ethernet MAC with 64-bit SDR Interface," page 35

- `<component_name>_example_design.ucf`

  The UCF for the core and the example design.

- `address_swap.v`

  The address swap Verilog file. This connects to the local link interface and swaps the destination and source addresses of frame.

- `xgmii_if.v`

  The Xgmii interface Verilog file. This contains the DDR registers to implement the enternal XGMII interface. See "10-Gigabit Ethernet MAC with External XGMII Interface," page 33.

- `physical_if.v`

  The Phy interface Verilog file. This contains the SDR registers to implement the 64-bit interface. See "10-Gigabit Ethernet MAC with 64-bit SDR Interface," page 35.

- `client_loopback.v`

The client loopback design example instanced in the local link level.

<project_dir>/<component_name>/example_design/fifo

This directory contains the files for the FIFO that is instanced in the client_loopback example design.

- `xgmac_fifo.v`

  The top level of the FIFO.
- `rx_fifo.v`

  The local link receive fifo. This implements the interface to the MAC receiver including frame dropping logic and has a local link interface.
- `tx_fifo.v`

  The local link transmit fifo. This implements the interface to the MAC transmitter including logic to initiate transmission when a full frame is stored in the FIFO and has a local link interface.
- `fifo_ram.v`

  This is a BRAM wrapper used by both FIFOs.

<project_dir>/<component_name>/implement

- `implement.sh`

  A UNIX shell script that processes the example design through the Xilinx tool flow.
- `implement.bat`

  A Windows batch file that processes the example design through the Xilinx tool flow.
- `xst.scr`

  The XST script file for the example design.
- `xst.prj`

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

<project_dir>/<component_name>/implement/results

This directory is produced by the implement scripts and is used to run the example design files and the `<component_name>.ngc` file through the Xilinx implementation tools. Once these are run it contains the following files for timing simulation.

- `routed.v`

  The back-annotated SimPrim based Verilog design. Used for timing simulation.
- `routed.sdf`

  The timing information for simulation is contained in this file.

<project_dir>/<component_name>/simulation

- `demo_tb.v`

  This file is the Verilog demonstration test bench for the 10-Gigabit Ethernet MAC core. More information can either be found in "10-Gigabit Ethernet MAC with External

XGMII Interface," page 33 or "10-Gigabit Ethernet MAC with 64-bit SDR Interface," page 35.

<project_dir>/<component_name>/simulation/functional

- `simulate_mti.do`

  A ModelSim macro file that compiles the example design sources and the structural simulation model then runs the functional simulation to completion.
- `wave_mti.do`

  A ModelSim macro file that opens a wave windows and adds interesting signals to it. It is called used by the simulate_mti.do macro file.
- `simulate_ncsim.sh`

  A UNIX shell script that compiles the example design sources and the structural simulation model then runs the functional simulation to completion using NC-Sim.
- `wave_ncsim.sv`

  A NC-Sim macro file that opens a wave windows and adds interesting signals to it. It is called used by the simulate_ncsim.sh script.

<project_dir>/<component_name>/simulation/timing

- `simulate_mti.do`

  A ModelSim macro file that compiles the Verilog timing model and demo test bench then runs the timing simulation to completion.
- `wave_mti.do`

  A ModelSim macro file that opens a wave windows and adds interesting signals to it. It is called used by the simulate_mti.do macro file.
- `simulate_ncsim.sh`

  A UNIX shell script that compiles the example design sources and the timing model then runs the functional simulation to completion using NC-Sim.
- `wave_ncsim.sv`

  A NC-Sim macro file that opens a wave windows and adds interesting signals to it. It is called used by the simulate_ncsim.sh script.

Implementation and Test Scripts

Implementation Script

The implementation script is either a shell script or batch file that processes the example design through the Xilinx tool flow. It is located in one of the following directories, depending on the platform:

For UNIX

```
project_dir/component_name/implement/implement.sh
```

For Windows

```
project_dir/component_name/implement/implement.bat
```

Full System Hardware Evaluation or Full License Implement Script

If the core is generated with the Full System Hardware Evaluation license or Full license, the implement script performs the following steps:

- The example HDL wrapper and client loopback logic is synthesized using XST
- ngdbuild is run to consolidate the core netlist and the wrapper netlist into the NGD file containing the entire design
- The design is mapped to the target technology
- The design is place-and-routed on the target device
- Static timing analysis is performed on the routed design using trce
- A bitstream is generated
- netgen runs on the routed design to generate VHDL and Verilog netlists and timing information in the form of SDF files
- These files are copied into the <core_instance>/implement/results directory

Simulation Only Evaluation License Implement Script

If the core is generated with the Simulation Only Evaluation license no implement directory or script is generated.

Simulation Scripts

Simulation macro files are provided for ModelSim and shell scripts are provided for NC-Sim. The scripts automate the simulation of the test bench and can be found in the following location:

Functional

`project_dir/component_name/simulation/functional/simulate_mti.do`

`project_dir/component_name/simulation/functional/simulate_ncsim.sh`

Timing

`project_dir/component_name/simulation/timing/simulate_mti.do`

`project_dir/component_name/simulation/timing/simulate_ncsim.sh`

The scripts perform the following tasks:

- Compiles the gate level netlist
- Compiles the demonstration test bench
- Starts a simulation of the test bench (with timing information if a Full-system Evaluation License or Full License is in use)
- Opens a Wave window and adds some interesting signals (wave.do)
- Runs the simulation to completion

10-Gigabit Ethernet MAC with External XGMII Interface

Example Design and HDL Wrapper

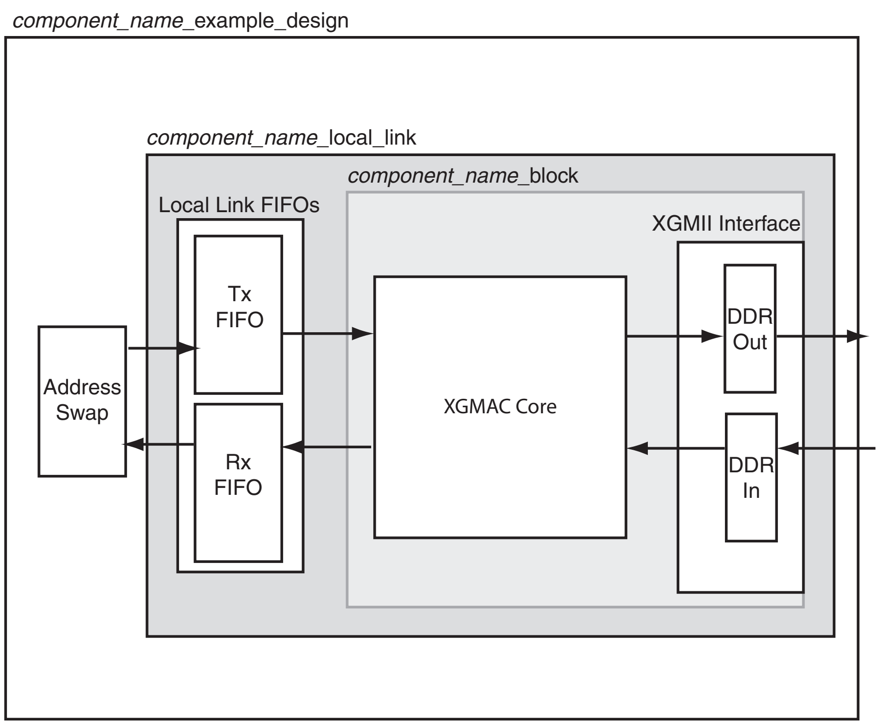


Figure 4-3: **Example Design and HDL Wrapper for 10-Gigabit Ethernet MAC with XGMII Interface**

The example design and HDL wrapper contain the following:

- Global clock buffers and Digital Clock Managers (DCMs)
- HDL sources for client loopback design

The client loopback design performs the following functions:

- Drops frame marked as bad by the core
- Crosses clock domain from received clock to transmit clock safely using an asynchronous FIFO

The address swap module performs the following function:

- Swaps the destination and source address field in the received Ethernet frame

The XGMII block performs the following functions:

- Receiver DCM and clock buffer
- DDR logic for the XGMII Interface

Demonstration Test Bench

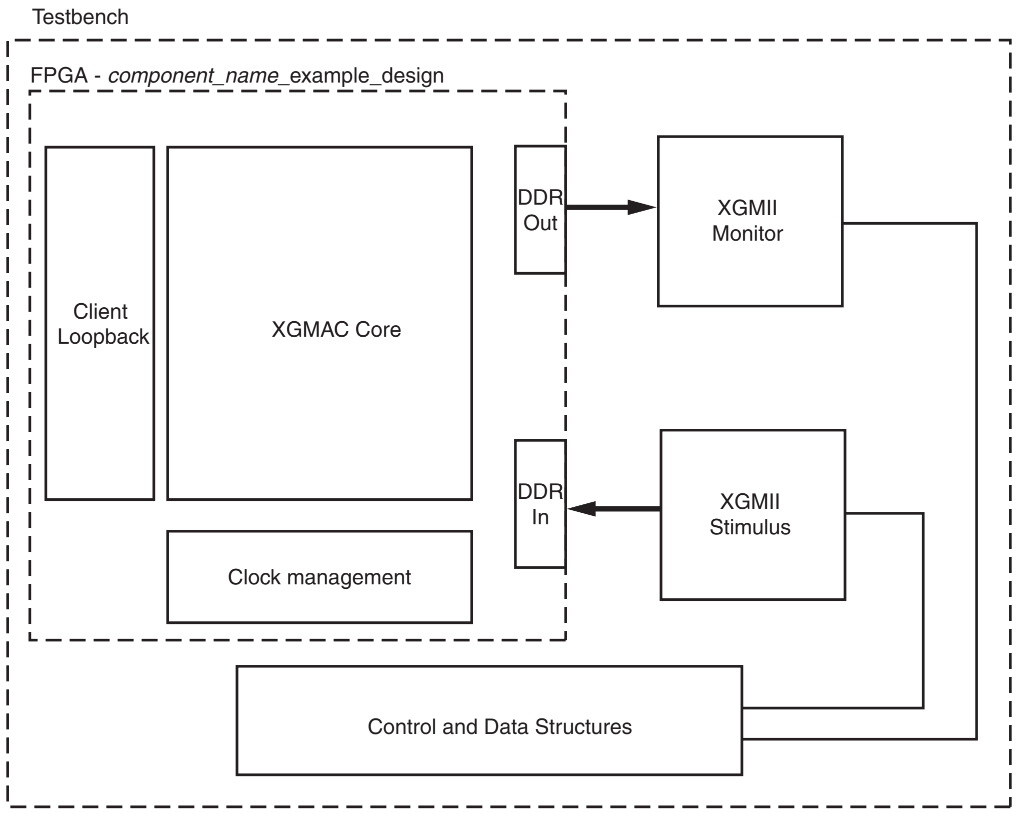


Figure 4-4: **Demonstration Test Bench for 10-Gigabit Ethernet MAC with XGMII Interface**

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core. It consists of transactor procedures or tasks that connect to the PHY-side ports of the example design, and a control program that pushes frames of varying length and content through the design and checks the values as they exit the core.

Because the address swap design swaps the destination and source field, the CRC is different on the outbound frame compared to that injected into the receiver. This is taken into account by the test bench when checking the transmitted data.

10-Gigabit Ethernet MAC with 64-bit SDR Interface

Example Design and HDL Wrapper

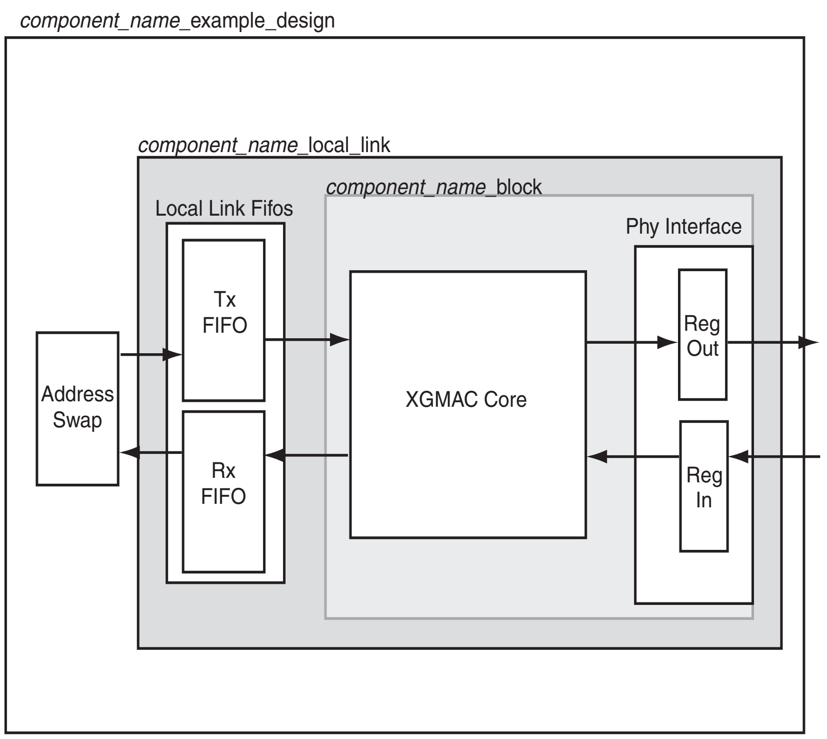


Figure 4-5: **Example Design and HDL Wrapper for 10-Gigabit Ethernet MAC with 64-bit Interface**

The example design and HDL wrappers contain the following:

- Global clock buffers and Digital Clock Managers (DCMs)
- HDL sources for client loopback design

The client loopback design performs the following functions:

- Drops frame marked as bad by the 10-Gigabit Ethernet MAC core
- Crosses clock domain from received clock to transmit clock safely using an asynchronous FIFO

The address swap module performs the following function:

- Swaps the destination and source address field in the received Ethernet frame

The physical interface block performs the following functions:

- Registers for the 64-bit SDR interface
- Receiver DCM and clock buffer

Demonstration Test Bench

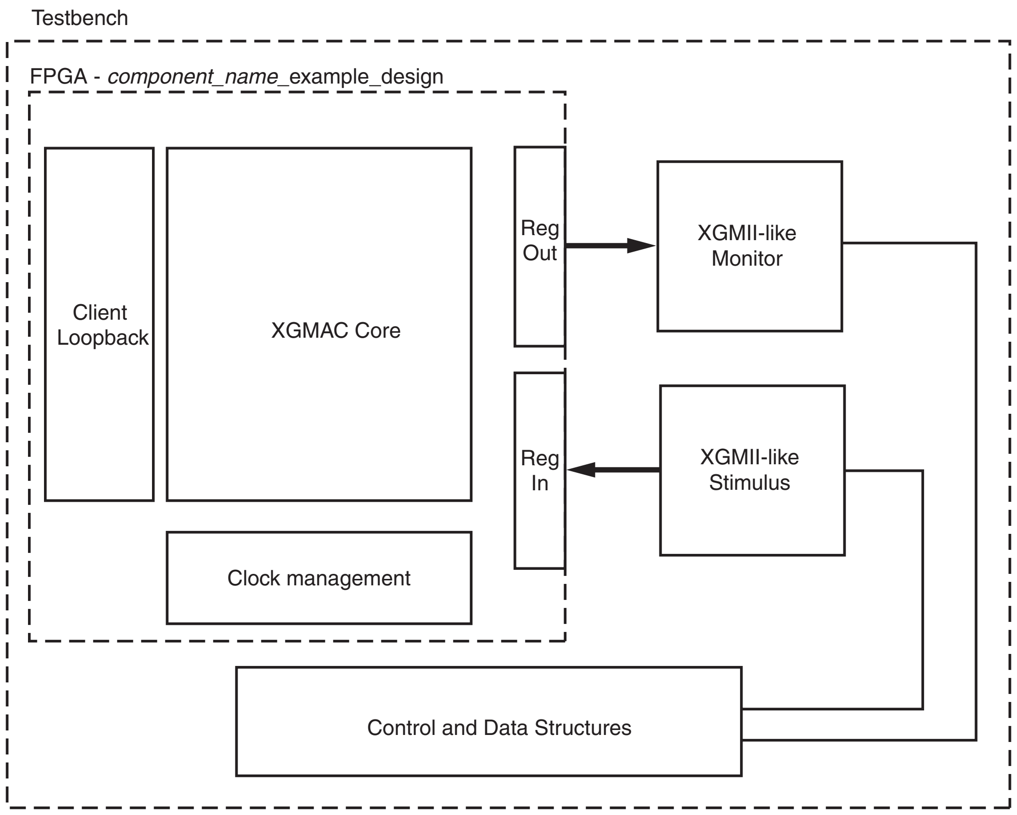


Figure 4-6: **Demonstration Test Bench for 10-Gigabit Ethernet MAC with 64-bit Interface**

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core itself. It consists of transactor procedures or tasks which connect to the PHY-side ports of the example design, and a control program that pushes frames of varying length and content through the design and checks the values as they exit the core.

Because the address swap design swaps the destination and source field, the CRC is different on the outbound frame compared to that injected into the receiver. This is taken into account by the test bench when checking the transmitted data.

Transmit-only Cores

Example Design and HDL Wrapper

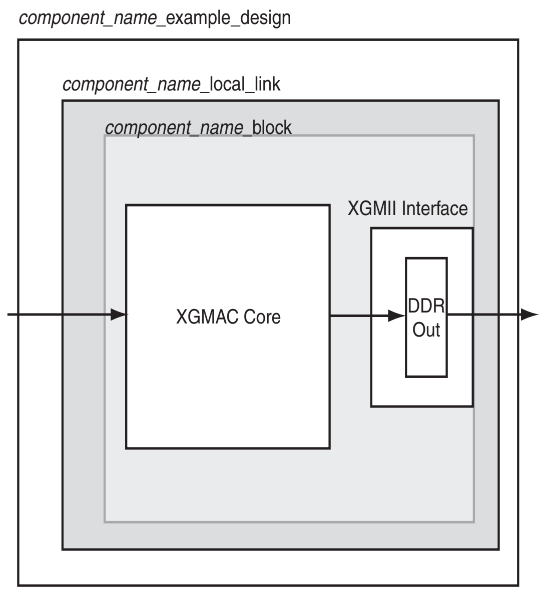


Figure 4-7: **HDL Wrapper for 10-Gigabit Ethernet MAC Transmit-only Configurations**

The example design and HDL wrapper for transmit -only cores contain the following:

- Registers for either the 64-bit SDR interface or 32-bit DDR interface
- Global clock buffers and Digital Clock Managers (DCMs)

Because this is a unidirectional design, there is no client loopback logic in this example design.

Note: This customization of the core will bring certain signals on the client-side interface such as tx_start, tx_ack off-chip through IOBs. Because these signals are designed to be connected internally in the FPGA, this can result in difficulties achieving either static timing closure or dynamic simulation integrity in the placed-and-routed example design. These difficulties do not affect the core in the final application.

Demonstration Test Bench

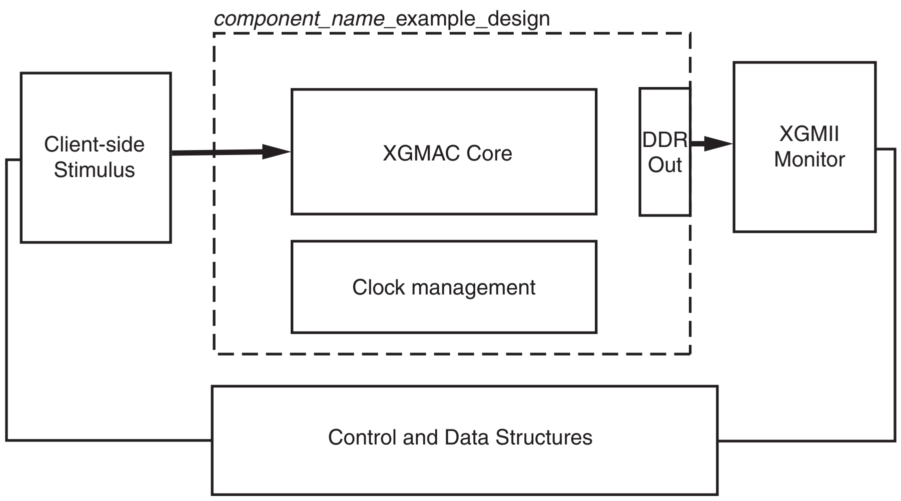


Figure 4-8: **Demonstration Test Bench for 10-Gigabit Ethernet MAC: Transmit only**

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core itself. It consists of transactor procedures or tasks which connect to the major ports of the example design, and a control program that pushes frames of varying length and content through the design and checks the values as they exit the core.

Receive-only Cores

Example Design and HDL Wrapper

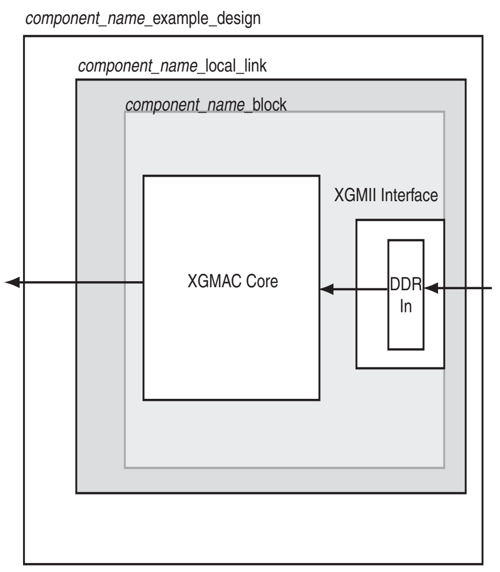


Figure 4-9: **HDL Wrapper for 10-Gigabit Ethernet MAC: Receive only**

The example design and HDL wrapper for transmit -only cores contain the following:

- Registers for either the 64-bit SDR interface or 32-bit DDR interface
- Global clock buffers and Digital Clock Managers (DCMs)

Because this is a unidirectional design, there is no client loopback logic in this example design.

Demonstration Test Bench

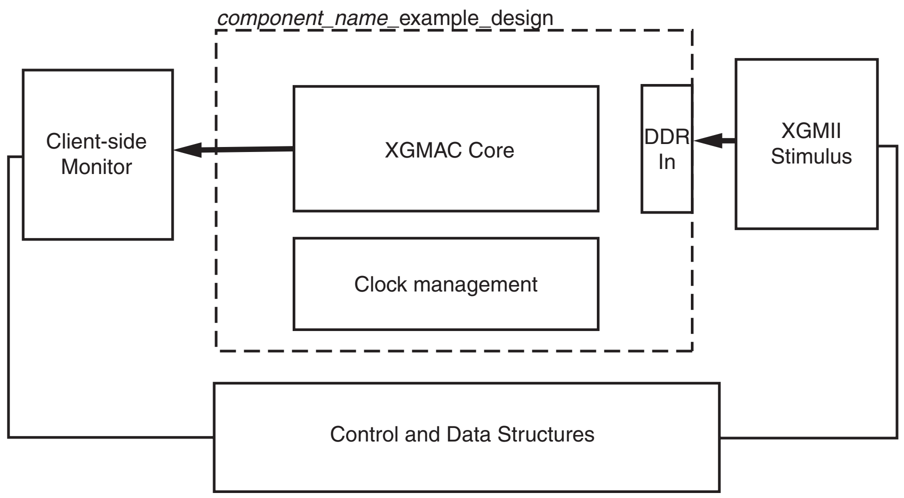


Figure 4-10: **Demonstration Test Bench for 10-Gigabit Ethernet MAC: Receive only**

The demonstration test bench is a simple VHDL or Verilog program to exercise the example design and the core itself. It consists of transactor procedures or tasks which connect to the major ports of the example design, and a control program that pushes frames of varying length and content through the design and checks the values as they exit the core.

Overview of Local Link Interface

Data is transferred on the local-link interface from source to destination, with the flow governed by the four active low control signals `sof_n, eof_n, src_rdy_n` and `dst_rdy_n`. The flow of data is controlled by the `src_rdy_n` and `dst_rdy_n` signals. Only when these signals are asserted simultaneously is data transferred from source to destination. The individual packet boundaries are marked by the `sof_n` and `eof_n` signals. The rem[2:0] signal is used to indicate the position of the sof, always denoted by "001" coincident with sof_n being asserted. When eof_n is asserted , rem[2:0] is encoded to show which of the 8 bytes contain valid frame data, see Table 4-1. For more information on the local-link interface please see:

http://www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?iLanguageID=1&key=LocalLink_UserInterface

Figure 4-11 shows the transfer of a frame without flow control.

Table 4-1: **Remainder Binary Encoding**

rem[2:0]	Bytes Valid[7:0]	Data Valid
000	00000001	[7:0]
001	00000011	[15:0]
010	00000111	[23:0]
011	00001111	[31:0]

Table 4-1: **Remainder Binary Encoding**

rem[2:0]	Bytes Valid[7:0]	Data Valid
100	00011111	[39:0]
101	00111111	[47:0]
110	01111111	[55:0]
111	11111111	[63:0]

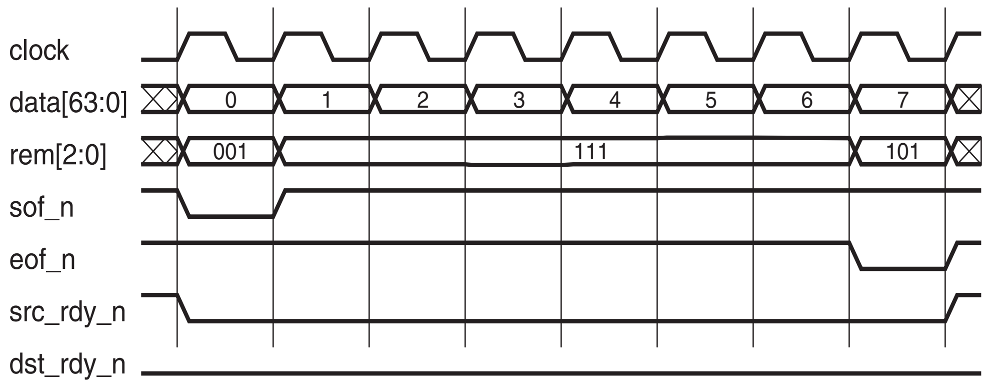


Figure 4-11: **Frame Transfer across Local-link Interface**

Figure 4-12 illustrates frame transfer of a frame, where both the `src_rdy_n` and `dst_rdy_n` signals are used to control the flow of data across the interface.

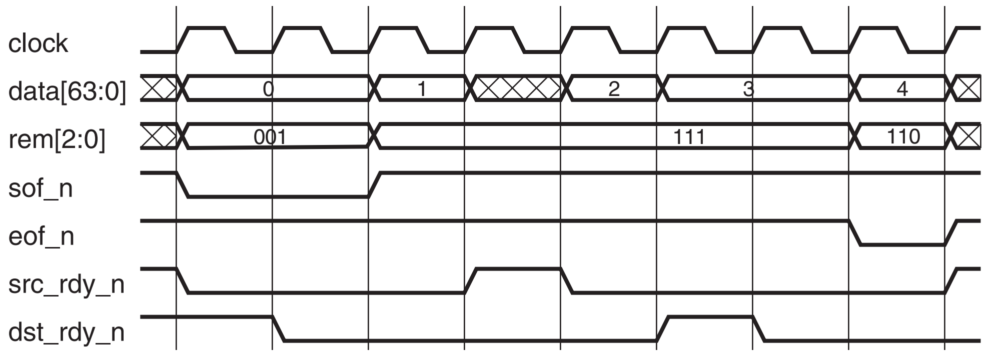


Figure 4-12: **Frame Transfer with Flow Control**