

# **LogiCORE™ Tri-Mode Ethernet MAC v3.4**

## **Getting Started Guide**

UG139 August 8, 2007





Xilinx is disclosing this Specification to you solely for use in the development of designs to operate on Xilinx FPGAs. Except as stated herein, none of the Specification 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 this Specification 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 Specification; 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 Specification. Xilinx reserves the right to make changes, at any time, to the Specification 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 Specification.

THE SPECIFICATION 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 SPECIFICATION, 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 SPECIFICATION, 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 SPECIFICATION, 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 SPECIFICATION. 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 SPECIFICATION TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Specification 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 Specification in such High-Risk Applications is fully at your risk.

© 2004-2007 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.

## Tri-Mode Ethernet MAC v3.4 Revision History

The following table shows the revision history for this document.

Date	Version	Revision
5/06/04	.5	Initial draft.
9/30/04	1.0	Initial Xilinx release.
4/28/05	2.0	Update to core v2.1, Xilinx tools v7.1i, support for Spartan-3E.
1/18/06	2.1	Update to core v2.2, release date, and Xilinx tools v8.1i.
7/13/06	3.1	Update core to v3.1, Xilinx tools 8.2i.
9/21/06	3.2	Update core to v3.2.
2/15/07	3.3	Update core to v3.3, Xilinx tools 9.1i.
8/8/07	3.4	Update core to v3.4, Xilinx tools 9.2i, Cadence IUS v5.8. (ds)

# Table of Contents

---

## Preface: About This Guide

Contents .....	7
Additional Resources .....	7
Conventions .....	8
Typographical .....	8
Online Document .....	9

## Chapter 1: Introduction

System Requirements .....	11
About the Core .....	11
Recommended Design Experience .....	11
Additional Core Resources .....	12
Technical Support .....	12
Feedback .....	12
Tri-Mode Ethernet MAC Core .....	12
Document .....	12

## Chapter 2: Licensing the Core

Before you Begin .....	15
License Options .....	15
Simulation Only .....	15
Full System Hardware Evaluation .....	15
Full .....	16
Obtaining your License .....	16
Installing your License File .....	16

## Chapter 3: Quick Start Example Design

Overview .....	17
Generating the Core .....	18
Implementing the Example Design .....	19
Running the Simulation .....	19
Functional Simulation .....	19
Timing Simulation .....	20
What's Next? .....	21

## Chapter 4: Detailed Example Design

Directory and File Contents .....	24
<project directory> .....	24
<project directory>/<component name> .....	24
<component name>/doc .....	25
<component name>/example design .....	25

example design/fifo .....	26
example_design/physical.....	26
<component name>/implement .....	27
implement/results .....	27
<component name>/simulation .....	28
simulation/functional .....	28
simulation/timing .....	28
<b>Implementation and Test Scripts .....</b>	<b>29</b>
Implementation Scripts for Timing Simulation.....	29
Test Scripts For Functional Simulation.....	30
Test Scripts For Timing Simulation.....	30
<b>Example Design.....</b>	<b>31</b>
HDL Example Design .....	31
10 Mbps /100 Mbps/1 Gbps Ethernet FIFO .....	32
Address Swap Module .....	33
<b>Demonstration Test Bench .....</b>	<b>34</b>
Test Bench Functionality.....	34
Changing the Test Bench .....	36

# *Schedule of Figures*

---

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

*Figure 3-1: Default Example Design and Test Bench* ..... 17

*Figure 3-2: Tri-Mode Ethernet MAC Main Screen.* ..... 18

## **Chapter 4: Detailed Example Design**

*Figure 4-1: HDL Example Design* ..... 31

*Figure 4-2: Frame Transfer across LocalLink Interface* ..... 32

*Figure 4-3: Modification of Frame Data by Address Swap Module* ..... 33

*Figure 4-4: Demonstration Test Bench* ..... 34



# About This Guide

---

The *Tri-Mode Ethernet MAC Getting Started Guide v3.4* guide provides information about generating a LogiCORE™ Tri-Mode Ethernet MAC (TEMAC) core, customizing and simulating the core utilizing the provided example design, and running the design files through implementation using the Xilinx tools.

## Contents

This guide contains the following chapters:

- [Preface, “About this Guide”](#) introduces the organization and purpose of this guide, including the conventions used in the guide and a list of additional resources.
- [Chapter 1, “Introduction”](#) describes the core and related information, including recommended design experience, additional resources, technical support, and submitting feedback to Xilinx.
- [Chapter 2, “Licensing the Core”](#) provides information about installing and licensing the core.
- [Chapter 3, “Quick Start Example Design”](#) describes how to quickly generate the example design using the default parameters.
- [Chapter 4, “Detailed Example Design”](#) provides detailed information about the example design and demonstration test bench.

## Additional Resources

For additional information, visit [www.xilinx.com/support](http://www.xilinx.com/support). The following table lists some of the resources you can select from this website. You can also directly access these resources 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">www.xilinx.com/support/techsup/tutorials/index.htm</a>
Answer Browser	Database of Xilinx solution records <a href="http://www.xilinx.com/support/xlnx/xil_ans_browser.jsp">www.xilinx.com/support/xlnx/xil_ans_browser.jsp</a>
Application Notes	Descriptions of device-specific design techniques and approaches <a href="http://www.xilinx.com/support/apps/appsweb.htm">www.xilinx.com/support/apps/appsweb.htm</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/support/xlnx/xweb/xil_publications_index.jsp">www.xilinx.com/support/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">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/support/xlnx/xil_tt_home.jsp">www.xilinx.com/support/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	speed grade: - 100
<b>Courier bold</b>	Literal commands that you enter in a syntactical statement	<b>ngdbuild</b> design_name
<i>Italic font</i>	Variables in a syntax statement for which you must supply values	<b>ngdbuild</b> design_name
	References to other manuals	See the <i>Development System Reference Guide</i> for more information.
	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.
<text in brackets>	User-defined variable for directory names.	<component_name>
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as <b>bus[ 7 : 0 ]</b> , they are required.	<b>ngdbuild</b> [option_name] design_name
Braces { }	A list of items from which you must choose one or more	<b>lowpwr</b> = {on   off}
Vertical bar	Separates items in a list of choices	<b>lowpwr</b> = {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	<b>allow block</b> <i>block_name</i> <i>loc1 loc2 ... locn;</i>

## 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. See " <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">www.xilinx.com</a> for the latest speed files.



## Introduction

---

The Tri-Mode Ethernet MAC (TEMAC) core is a fully verified solution that supports Verilog-HDL and VHDL. The example design in this guide is provided in both Verilog and VHDL.

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

## System Requirements

### Windows

- Windows® 2000 Professional (Service Pack 2-4)
- Windows XP Professional (Service Pack 1)

### Solaris/Linux

- Sun Solaris™ 9/10
- Red Hat™ Enterprise Linux 4.0 (32-bit and 64-bit)

### Software

#### Software

- ISE™ 9.2i 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](http://www.xilinx.com/xlnx/xil_sw_updates_home.jsp?update=sp).

## About the Core

The TEMAC 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 [www.xilinx.com/systemio/temac/index.htm](http://www.xilinx.com/systemio/temac/index.htm).

For information about system requirements, installation, and licensing options, see [Chapter 2, "Licensing the Core."](#)

## Recommended Design Experience

Although the TEMAC 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 (UCF) is recommended.

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

## Additional Core Resources

For additional details and updates, see the following documents, located on the Xilinx TEMAC product page, accessible from

[www.xilinx.com/systemio/temac/index.htm](http://www.xilinx.com/systemio/temac/index.htm)

- *Tri-Mode Ethernet MAC Data Sheet*
- *Tri-Mode Ethernet MAC Release Notes*
- *Tri-Mode Ethernet MAC User Guide*

For updates to this document, see the *Tri-Mode Ethernet MAC Getting Started Guide*, also located on the TEMAC product page.

## Technical Support

The fastest method for obtaining specific technical support for the TEMAC core is through the [support.xilinx.com/](http://support.xilinx.com/) website. Questions are routed to a team of engineers with specific expertise in using the TEMAC core.

Xilinx provides technical support for use of this product as described in the *Tri-Mode Ethernet MAC User Guide* and the *Tri-Mode 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 TEMAC core and the documentation supplied with the core.

### Tri-Mode Ethernet MAC Core

For comments or suggestions about the 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 the 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:

- Document title

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



# Licensing the Core

---

This chapter provides instructions for obtaining a license for the TEMAC core, which must be completed before using the core in your designs. The TEMAC 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.

## Before you Begin

This chapter assumes you have installed the core using either the CORE Generator IP Software Update installer or by performing a manual installation after downloading the core from the web. For information about installing the core, see the TEMAC product page at [www.xilinx.com/systemio/temac/index.htm](http://www.xilinx.com/systemio/temac/index.htm).

## License Options

The TEMAC core provides three licensing options. After installing the core, choose a license option, if applicable.

### Simulation Only

The Simulation Only Evaluation license is provided with the Xilinx CORE Generator and no license key is required. This license lets you assess the core functionality with either the provided example design or alongside your own design and demonstrate 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 a Full System Hardware Evaluation License

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

- Navigate to the TEMAC product page: [www.xilinx.com/systemio/temac/index.htm](http://www.xilinx.com/systemio/temac/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 TEMAC 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 TEMAC 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 email containing the license and installation instructions will be sent to you immediately.

## Installing your License File

The Simulation Only Evaluation license is provided with the CORE Generator and no license file is required. If you select either the Full System Hardware Evaluation license or the Full license, an email will be sent to you containing instructions for installing your license file. In addition, information about advanced licensing options and technical support is provided in the email.



## Quick Start Example Design

This chapter provides instructions for generating the TEMAC example design using the default parameters.

### Overview

The TEMAC example design consists of the following:

- A TEMAC core netlist
- An HDL example design
- A demonstration test bench to exercise the example design

Figure 3-1 shows the block diagram for the example design and the test bench provided with the TEMAC core. The example design has been tested with Xilinx ISE 9.2i, Cadence® IUS v5.8 and Mentor Graphics® ModelSim® PE/SE 6.1e.

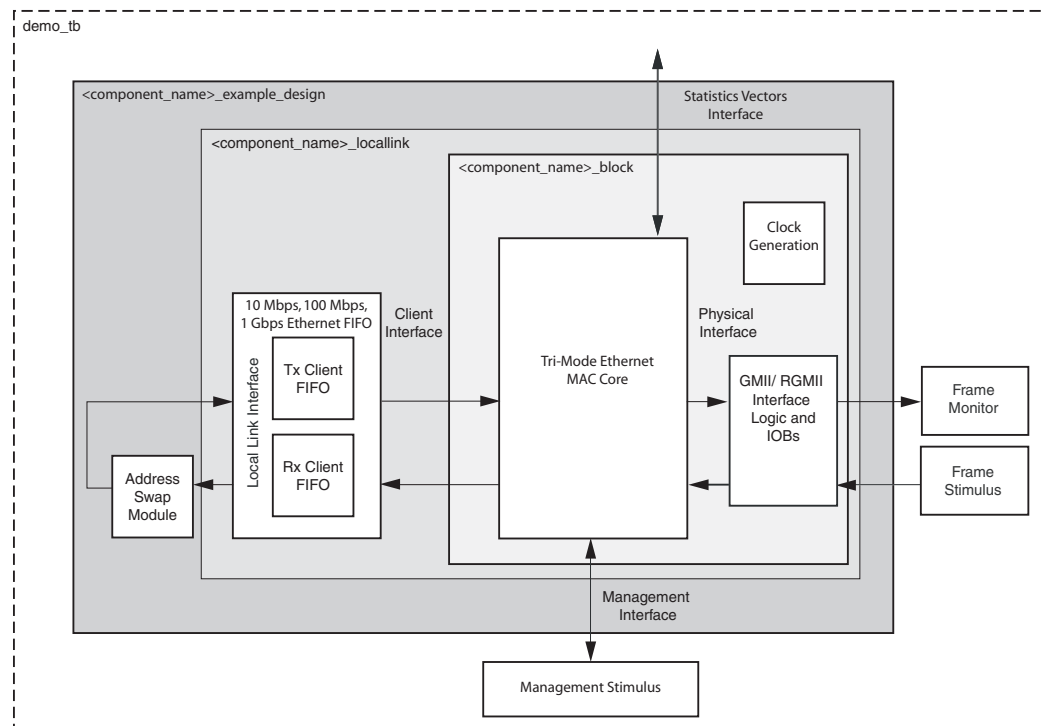


Figure 3-1: Default Example Design and Test Bench

## Generating the Core

To begin using the TEMAC example design, start by generating the core.

### To generate the core:

1. Start the CORE Generator.

For help starting and using the CORE Generator, see the documentation supplied with ISE, including the *CORE Generator Guide* at [www.xilinx.com/support/software\\_manuals.htm](http://www.xilinx.com/support/software_manuals.htm).

2. Choose File > New Project.

3. Do the following to set project options:

- From Target Architecture, select an FPGA family that supports the core, for example Virtex™-II Pro.

**Note:** If an unsupported silicon family is selected, the core will not appear in the taxonomy tree. For a list of supported architectures, see the *Tri-Mode Ethernet MAC Data Sheet*.

- For Design Entry, select either VHDL or Verilog; for Vendor, select Other.

4. After creating the project, locate the directory containing the core in the taxonomy tree. The project appears under one of the following:

- Communications & Networking /Ethernet
- Communications & Networking /Networking
- Communications & Networking /Telecommunications

5. Double-click the core. A warning may appear regarding the limitations of the simulation-only evaluation license.

6. Click OK to exit the dialog box. The core customization screen appears.

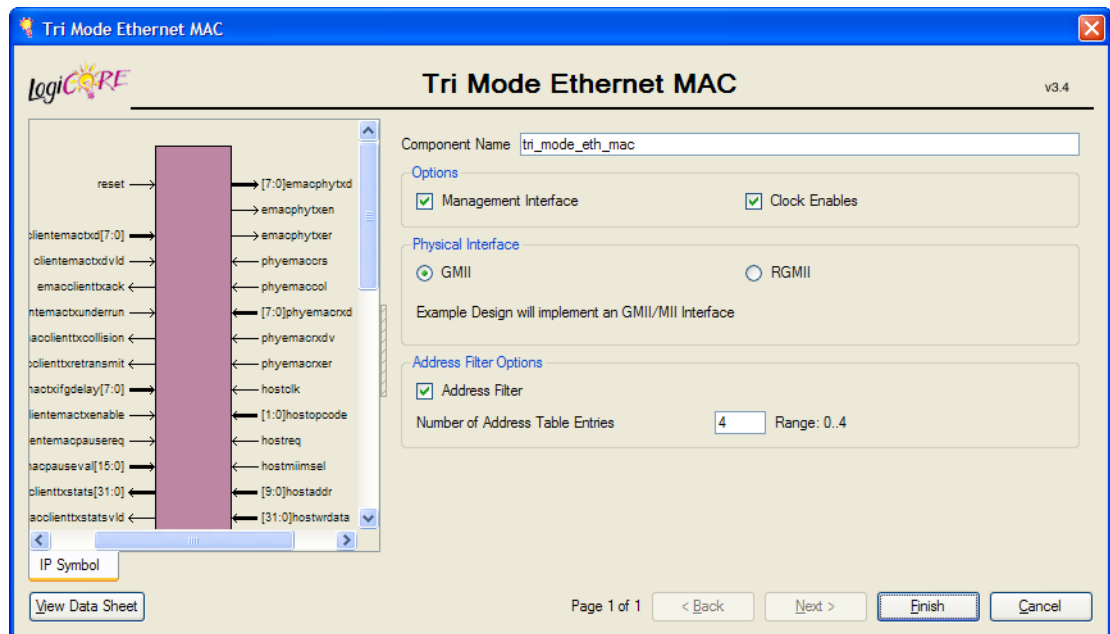


Figure 3-2: Tri-Mode Ethernet MAC Main Screen

7. In the Component Name field, enter a name for the core instance.

8. Accept the remaining defaults and click Generate to generate the core.
9. 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.”](#)

After the core is generated, a functional simulation directory is created, which contains scripts to simulate the core using the structural HDL models. For more information, see [“Functional Simulation,”](#) page 19.

## Implementing the Example Design

If the core is generated with a Hardware Evaluation or a Full license, the netlist and HDL example design can be processed through the Xilinx implementation toolset. The generated output files include several scripts to assist the user in running the Xilinx software.

**Note:** In the following examples, *<project\_dir>* is the CORE Generator project directory and *<component\_name>* is the name entered in the customization dialog box.

Open a command prompt or shell in your project directory, then enter the following commands:

### UNIX

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

### Windows

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

These commands execute a script that synthesizes, builds, maps, and place-and-routes the example design together with the core. The script also generates a gate-level model of the example design and a netlist for use in timing simulation. The resulting files are placed in the results directory, which is created by the implement script at runtime.

## Running the Simulation

### Functional Simulation

To run the functional simulation, you must have the Xilinx Simulation Libraries compiled for your system. For more information, see *Compiling Xilinx Simulation Libraries (COMPXLIB)* in the *Xilinx ISE Synthesis and Verification Design Guide*, which can be obtained from [www.xilinx.com/support/software\\_manuals.htm](http://www.xilinx.com/support/software_manuals.htm).

**Note:** In the simulation examples that follow, *<project\_dir>* is the CORE Generator project directory, and *<component\_name>* is the component name as entered in the core customization dialog box.

### VHDL Simulation

To run a VHDL functional simulation:

- Launch the simulator and set the current directory to *<project\_dir>/<component\_name>/simulation/functional*

- For ModelSim map the UniSim library:

```
ModelSim> vmap unisim <path to compiled libraries>/unisim
```

- Launch the simulation script:

```
ModelSim> do simulate_mti.do
```

```
IUS> ./implement_ncsim.sh
```

The scripts compile the structural VHDL model, the example design files and the demonstration test bench, add some relevant signals to a wave window, then run the simulation to completion. At this point, you can review the simulation transcript and waveform to observe the operation of the core.

## Verilog Simulation

To run a Verilog functional simulation:

- Launch the simulator and set the current directory to

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

- For ModelSim map the UniSim library:

```
ModelSim> vmap unisims_ver <path to compiled libraries>/unisims_ver
```

- Launch the simulation script:

```
ModelSim> do simulate_mti.do
```

```
IUS> ./implement_ncsim.sh
```

The scripts compile the structural Verilog model, the example design files and the demonstration test bench, add some relevant signals to a wave window, then run the simulation to completion. At this point, you can review the simulation transcript and waveform to observe the operation of the core.

## Timing Simulation

To run the gate-level simulation, you must have the Xilinx Simulation Libraries compiled for your system. For more information, see *Compiling Xilinx Simulation Libraries (COMPXLIB)* in the *Xilinx ISE Synthesis and Verification Design Guide*, which can be obtained from [www.xilinx.com/support/software\\_manuals.htm](http://www.xilinx.com/support/software_manuals.htm).

**Note:** 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 dialog box.

## VHDL Simulation

To run a VHDL timing simulation:

- Launch the simulator and set the current directory to

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

- For ModelSim map the SimPrim library:

```
ModelSim> vmap simprim <path to compiled libraries>/simprim
```

- Launch the simulation script:

```
ModelSim> do simulate_mti.do
```

```
IUS> ./implement_ncsim.sh
```

The scripts compile the gate-level model and the demonstration test bench, add some relevant signals to a wave window, then run the simulation to completion. At this point, you can review the simulation transcript and waveform to observe the operation of the core.

## Verilog Simulation

To run a Verilog timing simulation:

- Launch the ModelSim simulator and set the current directory to  
`<project_dir>/<component_name>/test/verilog`
- For ModelSim map the SimPrim library:

```
ModelSim> vmap simprims_ver <path to compiled_libraries>/simprims_ver
```

- Launch the simulation script:

```
ModelSim> do simulate_mti.do
```

```
IUS> ./implement_ncsim.sh
```

The scripts compile the gate-level model and the demonstration test bench, add some relevant signals to a wave window, then run the simulation to completion. At this point, you can review 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 TEMAC core in your own design, see the *Tri-Mode Ethernet MAC User Guide*.



## Detailed Example Design

---

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

-  **<project directory>**  
Top-level project directory; name is user-defined.
  -  **<project directory>/<component name>**  
Core release notes file
    -  **<component name>/doc**  
Product documentation
    -  **<component name>/example design**  
Verilog and VHDL design files
      -  **example design/fifo**  
Files for the FIFO that is instantiated in the LocalLink example
      -  **example\_design/physical**  
Files for the physical interface of the MAC
    -  **<component name>/implement**  
Implementation script files
      -  **implement/results**  
Results directory, created after implementation scripts are run, and contains implement script results
    -  **<component name>/simulation**  
Simulation scripts
      -  **simulation/functional**  
Functional simulation files
      -  **simulation/timing**  
Timing simulation files

## Directory and File Contents

The core directories and their associated files are defined in the following sections.

**Note:** The implement and timing simulation directories are only present when the core is generated with a Full System Hardware Evaluation license or a Full license.

### <project directory>

The project directory contains all the CORE Generator project files.

Table 4-1: Project Directory

Name	Description
<project_dir>	
<component_name>.ngc	Binary Xilinx implementation netlist. Describes how the core is to be implemented. Used as input to the Xilinx Implementation Tools.
<component_name>.v[hd]	VHDL or Verilog structural simulation model. File used to support functional simulation of a core. The 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>_flist.txt	Text file that defines all the output files produced when a customized core is generated using the CORE Generator.
<component_name>.{veo vho}	Verilog or VHDL template for the core. This can be copied into the user design.

[Back to Top](#)

### <project directory>/<component name>

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

Table 4-2: Component Name Directory

Name	Description
<project_dir>/<component_name>	
tri_mode_eth_mac_release_notes.txt	Core release notes file

[Back to Top](#)



## <component name>/doc

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

**Table 4-3: Doc Directory**

Name	Description
<project_dir>/<component_name>/doc	
tri_mode_eth_mac_ds297.pdf	<i>Tri-Mode Ethernet MAC Data Sheet</i>
tri_mode_eth_mac_gsg139.pdf	<i>Tri-Mode Ethernet MAC Getting Started Guide</i>
tri_mode_eth_mac_ug138.pdf	<i>Tri-Mode Ethernet MAC User Guide</i>

[Back to Top](#)

## <component name>/example design

This directory contains the support files necessary for a VHDL or Verilog implementation of the example design. See “[Example Design](#),” [page 31](#) for more information.

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

Name	Description
<project_dir>/<component_name>/example_design	
<component_name>_example_design.v[hd]	Top-level VHDL or Verilog file for the example design. This instantiates the LocalLink block along with the address swap block, providing a simple loopback function.
<component_name>_example_design.ucf	User constraints file (UCF) for the core and the example design
<component_name>_locallink.v[hd]	Example design with a LocalLink client interface. This instantiates the block level TEMAC wrapper together with a receive and a transmit FIFO.
<component_name>_block.v[hd]	Block level TEMAC wrapper containing the core and all clocking and physical interface circuitry
<component_name>_mod.v	Verilog module declaration for the core instance in the example design
address_swap.v[hd]	Top-level example design instances this to swap the source and destination addresses of the incoming frames
clk_half.v[hd]	Simple clock divider circuit used by the clock generation circuitry (clock_enables = false)
johnson_cntr.v[hd]	Clock divider circuit used to produce the RGMII clocks for operation at 10/100 Mbps (physical_interface = rgmii)

Table 4-4: Example Design Directory (Continued)

Name	Description
clk_gen.v[hd]	Top-level of the clock generation circuit
tx_clk_gen.v[hd]	Transmitter clock generator
rx_clk_gen.v[hd]	Receiver clock generator
clk_pack.vhd	VHDL package for clocking components

[Back to Top](#)

## example design/fifo

This directory contains the files for the FIFO that is instantiated in the LocalLink example design.

Table 4-5: FIFO Directory

Name	Description
<project_dir>/<component_name>/example_design/fifo	
tx_client_fifo.v[hd]	Transmit client FIFO. This takes data from the client in LocalLink format, stores it and sends it to the MAC.
rx_client_fifo.v[hd]	Receive client FIFO. This reads in and stores data from the MAC before outputting it to the client in LocalLink format.
ten_100_1G_eth_fifo.v[hd]	FIFO top level. This instantiates the transmit and receive client FIFOs.

[Back to Top](#)

## example\_design/physical

This directory contains a file for the physical interface of the MAC. A GMII or RGMII version will be delivered by CORE Generator depending on the selected option.

Table 4-6: Physical Directory

Name	Description
<project_dir>/<component_name>/example_design/physical	
gmii_if.v[hd]	All clocking and logic required to provide a GMII physical interface
rgmii_v2_0_if.v[hd]	All clocking and logic required to provide a RGMII v2.0 physical interface

[Back to Top](#)

## <component\_name>/implement

This directory contains the support files necessary for implementation of the example design with the XILINX tools. For more information, see [“Example Design,” page 31](#). Execution of an implement script results in creation of the results directory and an xst project directory.

**Table 4-7: Implement Directory**

Name	Description
<project_dir>/<component_name>/implement	
implement.sh	UNIX shell script that processes the example design through the Xilinx tool flow
implement.bat	Windows batch file that processes the example design through the Xilinx tool flow
xst.prj	XST project file for the example design; it enumerates all the HDL files that need to be synthesised.
xst.scr	XST script file for the example design

[Back to Top](#)

## implement/results

This directory is created by the implement scripts and is used to run the example design files and the <component\_name>.ngc file through the Xilinx implementation tools. On completion of an implement script, this directory contains the following files for timing simulation. Output files from the Xilinx implementation tools are also located in this directory.

**Table 4-8: Results Directory**

Name	Description
<project_dir>/<component_name>/implement/results	
routed.v[hd]	Back-annotated SimPrim based gate-level VHDL or Verilog design. Used for timing simulation.
routed.sdf	Timing information for simulation

[Back to Top](#)

## <component name>/simulation

The simulation directory and the sub-directories below it provide the files necessary to test a VHDL or Verilog implementation of the example design.

**Table 4-9: Simulation Directory**

Name	Description
<project_dir>/<component_name>/simulation	
demo_tb.v[hd]	VHDL or Verilog demonstration test bench for the TEMAC core

[Back to Top](#)

## simulation/functional

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

**Table 4-10: Functional Directory**

Name	Description
<project_dir>/<component_name>/simulation/functional	
simulate_mti.do	ModelSim macro file that compiles the example design sources and the structural simulation model then runs the functional simulation to completion.
wave_mti.do	ModelSim macro file that opens a wave window and adds interesting signals to it. It is called by the simulate_mti.do macro file.
simulate_ncsim.sh	IUS script file that compiles the example design sources and the structural simulation model and then runs the functional simulation to completion.
wave_ncsim.sv	IUS macro file that opens a wave window and adds interesting signals to it.

[Back to Top](#)

## simulation/timing

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

**Table 4-11: Timing Directory**

Name	Description
<project_dir>/<component_name>/simulation/timing	
simulate_mti.do	ModelSim macro file that compiles the VHDL gate-level model and demo test bench then runs the timing simulation to completion.

Table 4-11: Timing Directory (Continued)

Name	Description
wave_mti.do	ModelSim macro file that opens a wave window and adds interesting signals to it. It is called used by the simulate_mti.do macro file.
simulate_ncsim.sh	UNIX shell script that compiles the VHDL gate-level model and demo test bench and then runs the functional simulation to completion using Cadence IUS.
wave_ncsim.sv	IUS macro file that opens a wave window and adds interesting signals to it. It is used by the simulate_ncsim.sh script.

[Back to Top](#)

## Implementation and Test Scripts

### Implementation Scripts for Timing Simulation

When the CORE Generator is run with a Full System Hardware license or a Full license, an implement script is generated in the <project\_dir>/<component\_name>/implement directory. The implementation script is either a shell script or batch file that processes the example design through the Xilinx tool flow.

#### UNIX

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

#### Windows

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

The implement script performs the following steps:

- The HDL example design is synthesized using XST
- Ngdbuild is run to consolidate the core netlist and the HDL example 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

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
```

## Test Scripts For Functional Simulation

The functional simulation flow is available with any license type, and the test script that automates the simulation of the test bench is located in one of the following locations:

### Mentor ModelSim

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

### Cadence IUS

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

The test script performs the following tasks:

- Compiles the structural simulation model of the core
- Compiles the example design files
- Compiles the demonstration test bench
- Starts a simulation of the test bench with no timing information
- Opens a Wave window and adds some signals of interest
- Runs the simulation to completion

## Test Scripts For Timing Simulation

When CORE Generator has been run with a Full System Hardware Evaluation license or Full license, a test script for running timing simulation is generated. The test script that automates the simulation of the test bench is located at the following locations:

### Mentor ModelSim

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

### Cadence IUS

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

The test script performs the following tasks:

- Compiles the gate-level model of the example design
- Compiles the demonstration test bench
- Starts a simulation of the test bench using timing information
- Opens a Wave window and adds some signals of interest
- Runs the simulation to completion

## Example Design

### HDL Example Design

Figure 4-1 illustrates the top-level design for the TEMAC core example design.

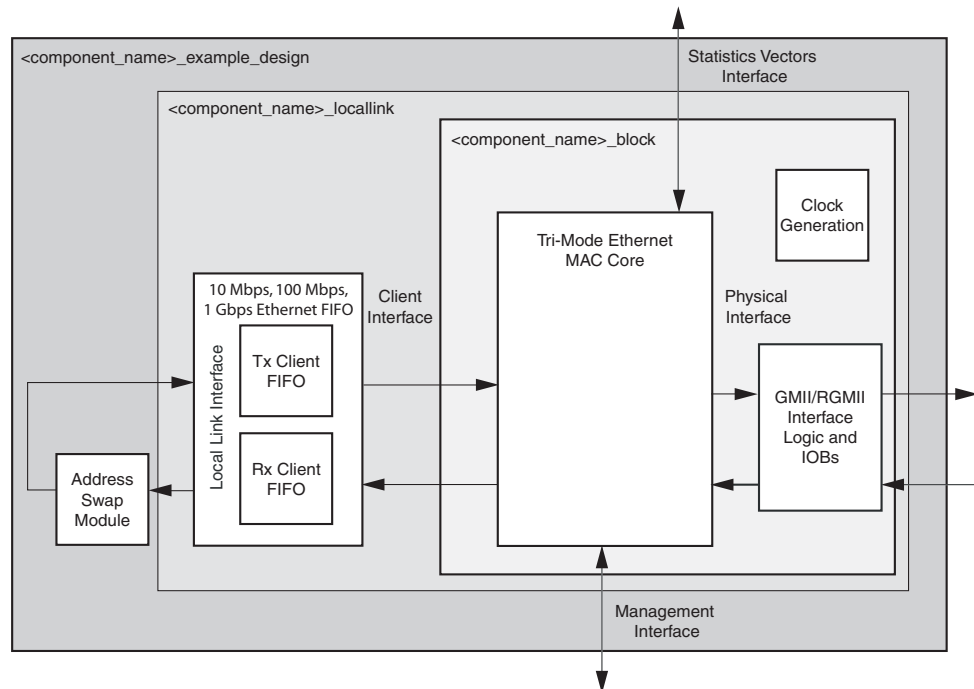


Figure 4-1: HDL Example Design

The top-level example design for the TEMAC is defined in the following files:

#### VHDL

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

#### Verilog

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

The HDL example design contains the following:

- An instance of the TEMAC core
- Clock management logic, including DCM and Global Clock Buffer instances, where required
- GMII or RGMII interface logic, including IOB and DDR registers instances, where required
- Client Transmit and Receive FIFOs with a LocalLink interface
- Client LocalLink loopback module that performs address swapping

The HDL example design provides client loopback functionality on the client side of the TEMAC core and connects the GMII/RGMII interface to external IOBs. This allows the

functionality of the core to be demonstrated either using a simulation package, as discussed in this guide, or in hardware, if placed on a suitable board.

## 10 Mbps /100 Mbps/1 Gbps Ethernet FIFO

The 10 Mbps/100 Mbps/1 Gbps Ethernet FIFO is described in the following files:

### VHDL

```
<project_dir>/<component_name>/example_design/fifo/ten_100_1g_eth_fifo.vhd
<project_dir>/<component_name>/example_design/fifo/tx_client_fifo.vhd
<project_dir>/<component_name>/example_design/fifo/rx_client_fifo.vhd
```

### Verilog

```
<project_dir>/<component_name>/example_design/fifo/ten_100_1g_eth_fifo.v
<project_dir>/<component_name>/example_design/fifo/tx_client_fifo.v
<project_dir>/<component_name>/example_design/fifo/rx_client_fifo.v
```

For a full description of the 10 Mbps/100 Mbps/1 Gbps Ethernet FIFO, see “Appendix A” of the *Tri-Mode Ethernet MAC User Guide*. See also [XAPP691](#), “Parameterizable LocalLink FIFO” at “[direct.xilinx.com/bvdocs/appnotes/xapp691.pdf](http://direct.xilinx.com/bvdocs/appnotes/xapp691.pdf)” for a more detailed description of the LocalLink interface.

The 10 Mbps/100 Mbps/1 Gbps Ethernet FIFO contains an instance of tx\_client\_fifo to connect to the TEMAC client side transmitter interface, and an instance of the rx\_client\_fifo to connect to the TEMAC client receiver interface. Both transmit and receive FIFO components implement a LocalLink user interface, through which the frame data can be read/written. [Figure 4-2](#) illustrates a straightforward frame transfer across the LocalLink interface.

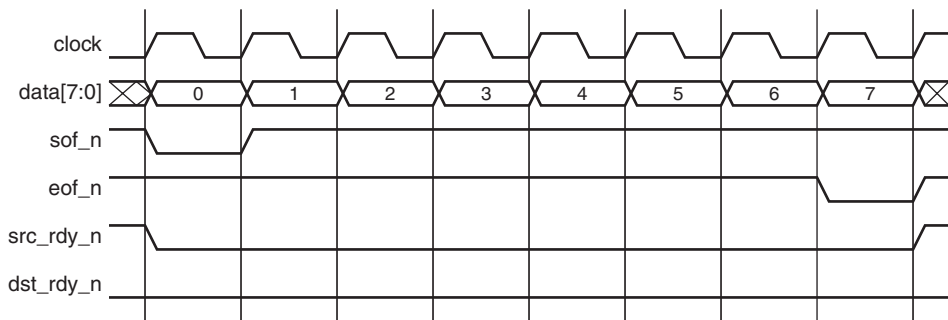


Figure 4-2: Frame Transfer across LocalLink Interface

### rx\_client\_fifo

The rx\_client\_fifo is built around two Dual Port Block RAMs, giving a total memory capacity of 4096 bytes. The receive FIFO will write in data received through the TEMAC core. If the frame is marked as good, that frame will be presented on the LocalLink interface for reading by the user, (in this case the tx\_client\_fifo module). If the frame is marked as bad, that frame is dropped by the receive FIFO.

If the receive FIFO memory overflows, the frame currently being received will be dropped, regardless of whether it is a good or bad frame, and the signal rx\_overflow will be asserted. Situations in which the memory may overflow are:



- The FIFO may overflow if the receiver clock is running at a faster rate than the transmitter clock or if the interpacket gap between the received frames is smaller than the interpacket gap between the transmitted frames. If this is the case the tx FIFO will not be able to read data from the rx FIFO as fast as it is being received.
- The FIFO size of 4096 bytes limits the size of the frames that it can store without error. If a frame is larger than 4000 bytes then the FIFO may overflow and data will be lost. It is therefore recommended that the example design is not used with the TEMAC core in jumbo frame mode for frames of larger than 4000 bytes.

### tx\_client\_fifo

The `tx_client_fifo` is built around two Dual Port block RAMs, giving a total memory capacity of 4096 bytes.

When a full frame has been written into the transmit FIFO, the FIFO will present data to the MAC transmitter. On receiving the `tx_ack` signal from the TEMAC the rest of the frame shall be transmitted.

If the FIFO memory fills up, the `dst_rdy_out_n` signal will be used to halt the LocalLink interface writing in data, until space becomes available in the FIFO. If the FIFO memory fills up but no full frames are available for transmission. For example if a frame larger than 4000 bytes is written into the FIFO, the FIFO will assert the `tx_overflow` signal and continue to accept the rest of the frame from the user. The overflow frame will be dropped by the FIFO. This ensures that the LocalLink interface does not lock up.

## Address Swap Module

### VHDL

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

### Verilog

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

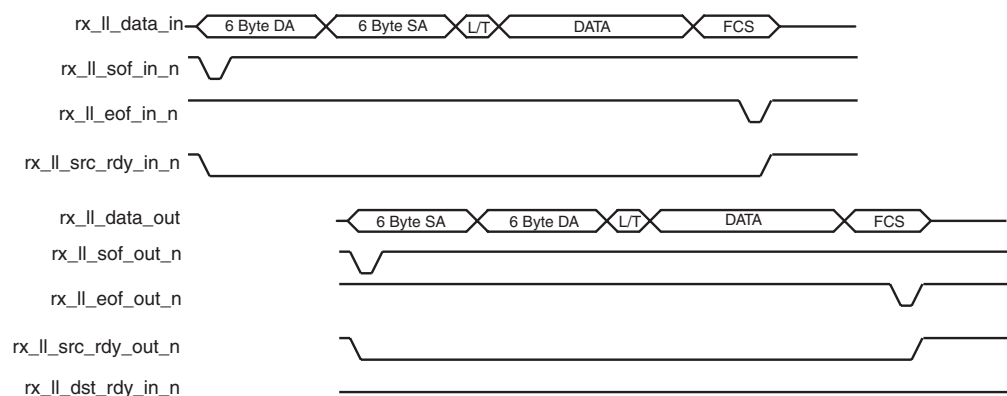


Figure 4-3: Modification of Frame Data by Address Swap Module

The address swap module takes frame data from the TEMAC receiver client LocalLink interface. The module swaps the destination and source addresses of each frame as shown in Figure 4-3 to ensure that the outgoing frame destination address matches the source

address of the link partner. The module transmits the frame control signals with an equal latency to the frame data.

## Demonstration Test Bench

### Test Bench Functionality

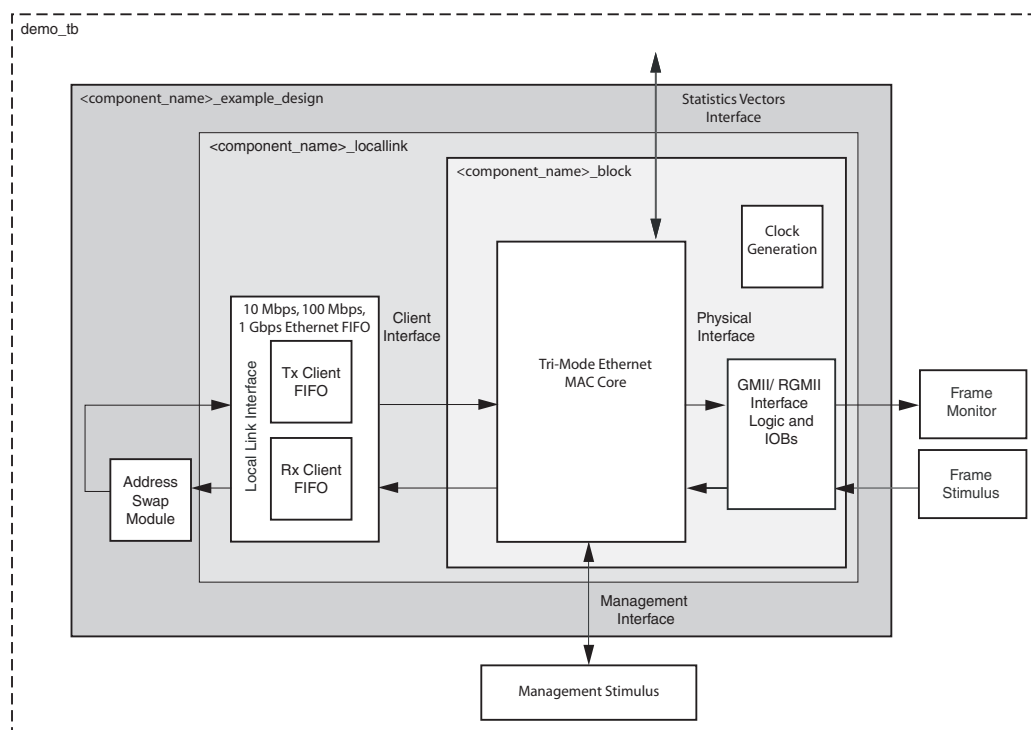


Figure 4-4: Demonstration Test Bench

The demonstration test bench is defined 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 test bench consists of the following:

- Clock generators
- A stimulus block that connects to the GMII/MII or RGMII receiver interface of the example design
- A monitor block to check data returned through the GMII/MII or RGMII transmitter interface

- A management block to exercise the management interface, if selected
- A control mechanism to manage the interaction of management, stimulus and monitor blocks

## Core with Management Interface

The demonstration test bench performs the following tasks:

- Input clock signals are generated.
- A reset is applied to the example design.
- The TEMAC core is configured through the management interface, setting up the MDC clock frequency and disabling flow control. If the Address Filter is selected, the Unicast Address registers are configured and the Address Filter is enabled.
- Four frames are pushed into the GMII/MII or RGMII receiver interface at 1 Gbps:
  - + 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 frames received at the GMII/MII or RGMII transmitter interface are checked against the stimulus frames to ensure data is the same. The monitor process takes into account the source/destination address field and FCS modifications resulting from the address swap module.
- The TEMAC core is configured through the management interface to run at 100 Mbps. The same four frames are then sent to the MII or RGMII interface and checked against the stimulus frames.
- The TEMAC core is configured through the management interface to run at 10 Mbps. The same four frames are then sent to the MII or RGMII interface and checked against the stimulus frames. If the core is generated with the RGMII option, or if the core has been generated for a Spartan™-3 or Virtex-4 device, warnings about the input clock period of the `gmi_i_rxc_dcm` component *may* appear in timing simulations. These can be safely ignored, because the `gmi_i_rxc_dcm` component is bypassed at 10 and 100 Mbps.

## Core with No Management Interface

The demonstration test bench performs the following tasks:

- Input clock signals are generated
- A reset is applied to the example design
- The TEMAC core is configured through the configuration vector, disabling flow control
- The stimulus block pushes four frames into the GMII/MII or RGMII receiver interface at 1 Gbps:
  - + 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 frames received at the GMII/MII or RGMII transmitter interface are checked against the stimulus frames to ensure data is the same. The monitor process takes into

account the source/destination address field and FCS modifications resulting from the address swap module.

- The TEMAC core is configured through the configuration vector to run at 100 Mbps. The same four frames are then sent to the MII or RGMII interface and checked against the stimulus frames.
- The TEMAC core is configured through the configuration vector to run at 10 Mbps. The same four frames are then sent to the MII or RGMII interface and checked against the stimulus frames. If the core is generated with the RGMII option, or if the core has been generated for a Spartan-3 or Virtex-4 device, warnings about the input clock period of the gmii\_rxc\_dcm component *may* appear in timing simulations. These can be safely ignored, because the gmii\_rxc\_dcm component is bypassed at 10 and 100 Mbps.

## Changing the Test Bench

### Changing Frame Data

The contents of the frame data passed into the TEMAC receiver can be changed by editing the DATA fields for each frame defined in the test bench. The test bench will automatically calculate the new FCS field to pass into the GEMAC, as well as calculating the new expected FCS value. Further frames can be added by defining a new frame of data.

### Changing Frame Error Status

Errors can be inserted into any of the pre-defined frames by changing the error field to '1' in any column of that frame.

When an error is introduced into a frame, the bad\_frame field for that frame must be set in order to disable the monitor checking for that frame.

The error currently written into the third frame can be removed by setting all error fields for the frame to '0' and unsetting the bad\_frame field.

### Changing the Tri-Mode Ethernet MAC Configuration

The configuration of the TEMAC used in the demonstration test bench can be altered.

**Caution!** Certain configurations of the TEMAC cause the test bench either to result in failure or cause processes to run indefinitely. The user must determine which configurations can safely be used with the test bench.

If the Management Interface option has been selected, the TEMAC can be reconfigured by adding further steps in the test bench management process, to write new configurations to the TEMAC. See the *Tri-Mode Ethernet MAC User Guide* for more information on using the management interface.

If the Management Interface option has not been selected, the TEMAC can be reconfigured by modifying the configuration vector directly. See the *Tri-Mode Ethernet MAC User Guide* for information about using the configuration vector.