LogiCORE™ IP Tri-Mode Ethernet MAC v4.4

Getting Started Guide

UG139 April 19, 2010

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 device.
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.
3/24/08	3.5	Update core to v3.5; Xilinx tools 10.1, Cadence IUS v6.1.
4/24/09	3.6	Update core to v4.1; Xilinx tools 11.1; Cadence IUS v8.1 -s009. Added Spartan-6 and Virtex®-6 device support.
6/24/09	3.7	Update core to v4.2; Xilinx tools 11.2; Added support for Virtex-6 CXT
09/16/09	3.8	Update core to v4.3; Xilinx tools 11.3; Added licensing support for 10/100 Mbps only core. Added support for Virtex-6 HXT and Virtex-6 -1L.
12/02/09	3.8.1	Updated licensing information.
04/19/10	3.9	Update core to v4.4; Xilinx tools 12.1.

Table of Contents

Chapter 3: Quick Start Example Design

Chapter 4: Detailed Example Design

Schedule of Figures

About This Guide

The *Tri-Mode Ethernet MAC Getting Started Guide* guide provides information about generating a LogiCORE™ IP 10/100/1000 Mbps Ethernet MAC (TEMAC) core, 1Gbps Ethernet MAC core and the 10/100 Mbps 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 organization and purpose of this guide and the conventions used in this document.
- 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.

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. Signal names also.	`speed grade: - 100`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *design_name*
Helvetica bold	Commands that you select from a menu	**File →Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in a syntax statement for which you must supply values	**`ngdbuild`** *design_name*
	References to other manuals	See the *User Guide* for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Dark Shading	Items that are not supported or reserved	This feature is not supported
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required.	**`ngdbuild`** `[`*`option_name`*`]` *`design_name`*
Braces { }	A list of items from which you must choose one or more	**`lowpwr`** **`=`**`{`**`on`**`|`**`off`**`}`
Vertical bar |	Separates items in a list of choices	**`lowpwr`** **`=`**`{`**`on`**`|`**`off`**`}`
Angle brackets < >	User-defined variable or in code samples	<directory name>

Vertical ellipsis
.
.
.
	Repetitive material that has been omitted	`IOB #1: Name = QOUT'`
		`IOB #2: Name = CLKIN'`
		.
		.
		.

Horizontal ellipsis . . .	Repetitive material that has been omitted	**`allow block`** *block_name loc1 loc2 ... locn;*

Convention	Meaning or Use	Example
Notations	The prefix '0x' or the suffix 'h' indicate hexadecimal notation	A read of address 0x00112975 returned 45524943h.
	An '_n' means the signal is active low	`usr_teof_n` is active low.

Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current document	See the section "Guide Contents" for details. See "Title Formats" in Chapter 1 for details.
Blue, underlined text	Hyperlink to a website (URL)	Go to www.xilinx.com for the latest speed files.

List of Acronyms

The following table describes acronyms used in this manual.

Acronym	Spelled Out
DCM	Digital Clock Manager
DDR	Double Data Rate
FAE	Field Application Engineer
FCS	Frame Check Sequence
FIFO	First In First Out
FPGA	Field Programmable Gate Array
Gbps	Gigabits per second
GMII	Gigabit Media Independent Interface
HDL	Hardware Description Language
IES	Incisive Enterprise Simulator (Cadence)
IOB	Input/Output Block
IP	Intellectual Property
ISE®	Integrated Software Environment
MAC	Media Access Controller
Mbps	Megabits per second
MDC	Management Data Clock
MII	Media Independent Interface
NGD	Native Generic Database
RGMII	Reduced Gigabit Media Independent Interface
TEMAC	Tri-Mode Ethernet MAC
UCF	User Constraints File
VCS	Verilog Compiled Simulator (Synopsys)
VHDL	VHSIC Hardware Description Language (VHSIC an acronym for Very High-Speed Integrated Circuits)
XCO	Xilinx CORE Generator™ software core source file
XST	Xilinx Synthesis Technology

Chapter 1

Introduction

The complete LogiCORE™ IP Tri-Mode Ethernet MAC solution consists of the 10/100/1000 Mbps Ethernet MAC, 1Gbps Ethernet MAC and 10/100 Mbps Ethernet MAC and an example design provided in both Verilog and VHDL.

The 10/100 Mbps IP core is sold standalone or you may purchase the Tri-Mode Ethernet MAC solution which includes all three IP cores. Please contact your local sales representative for ordering information.

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

System Requirements

Windows

- Windows XP Professional 32-bit/64-bit
- Windows Vista Business 32-bit/64-bit

Linux

- Red Hat Enterprise Linux WS v4.0 32-bit/64-bit
- Red Hat Enterprise Desktop v5.0 32-bit/64-bit (with Workstation Option)
- SUSE Linux Enterprise (SLE) desktop and server v10.1 32-bit/64-bit

Software

- ISE® software v12.1

About the Core

The TEMAC solution is a Xilinx® CORE Generator™ software product included in the latest IP Update on the Xilinx IP Center. For detailed information about the core, see the product page.

For information about system requirements, installation, and licensing options, see Chapter 2, "Licensing the Core."

Recommended Design Experience

Although the TEMAC solution is fully-verified, 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, updates, the data sheet, and updates to this guide, see the product page. After generating the core, in the document directory, the following documents are available:

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

Technical Support

The fastest method for obtaining specific technical support for the TEMAC solution is through the support.xilinx.com website. Questions are routed to a team of engineers with specific expertise using the TEMAC solution.

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 solution and the documentation supplied with the core.

Tri-Mode Ethernet MAC Solution

For comments or suggestions about the 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
- Configuration mode (10/100/1000 Mbps, 1Gbps, 10/100 Mbps)
- Explanation of your comments

Document

For comments or suggestions about the core, please submit a WebCase from 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

Licensing the Core

The Tri-Mode Ethernet MAC (TEMAC) solution consists of three Xilinx LogiCORE™ IP cores. This chapter provides licensing instructions for the 10/100/1000 Mbps Tri-Mode Ethernet MAC, 1Gbps Ethernet MAC and the 10/100 Mbps Ethernet MAC.

You must obtain the appropriate licenses before using the cores in your designs. These three IP cores are provided under the terms of the Xilinx LogiCORE™ IP Site License Agreement or Xilinx LogiCORE IP Project License Agreement, which conforms to the terms of the SignOnce IP License standard defined by the Common License Consortium. Purchase of the Tri-Mode Ethernet MAC core license includes licensing the 10/100/1000 Mbps Tri-Mode, 1Gbps and the 10/100 Mbps Ethernet MAC. Purchase of the 10/100 MAC core license only entitles full access to the 10/100 Mbps IP. Purchase of a core entitles you to technical support and access to updates for a period of one year.

Before you Begin

This chapter assumes that you have installed all required software specified on the product page for this core.

License Options

The TEMAC solution provides three licensing options. After installing the required Xilinx® ISE® software and IP Service Packs, choose a license option.

Simulation Only

The Simulation Only Evaluation license key is provided with the Xilinx CORE Generator™ tool. This key lets you assess core functionality with either the example design provided with the TEMAC solution, or alongside your own design and demonstrates the various interfaces to the core in simulation. (Functional simulation is supported by a dynamically generated HDL structural model.)

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 TEMAC solution using the example design and the demonstration test bench provided with the core.

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.

Full

The Full license key is available 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
- Functional 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 Key

This section contains information about obtaining a simulation, full system hardware, and full license keys.

Simulation License

No action is required to obtain the Simulation Only Evaluation license key; it is provided by default with the Xilinx CORE Generator software.

Full System Hardware Evaluation License

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

1. Navigate to the product page for this core.
2. Click Evaluate.
3. Follow the instructions to install the required Xilinx ISE software and IP Service Packs.

Obtaining a Full License

To obtain a Full license key, you must purchase a license for the core. After doing so, click the “Access Core” link on the Xilinx.com IP core product page for further instructions.

Installing your License File

The Simulation Only Evaluation license key is provided with the ISE software CORE Generator system and does not require installation of an additional license file. For the Full System Hardware Evaluation license and the Full license, an email will be sent to you containing instructions for installing your license file. Additional details about IP license key installation can be found in the ISE Design Suite Installation, Licensing and Release Notes document.

Chapter 3

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 solution 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 solution. The example design has been tested with Xilinx® ISE® software v12.1, Cadence Incisive Enterprise Simulator (IES) v9.2, Mentor Graphics ModelSim v6.5c, and Synopsys VCS and VCS MX 2009.12.

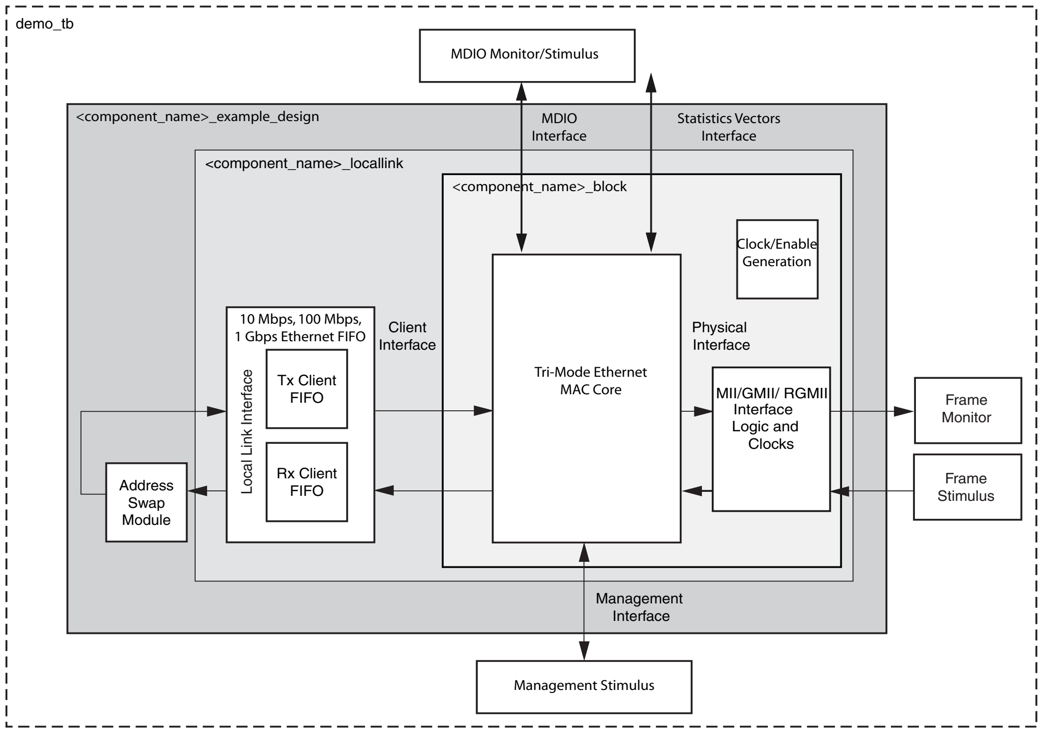


Figure 3-1: **Default Example Design and Test Bench**

Generating the Core

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

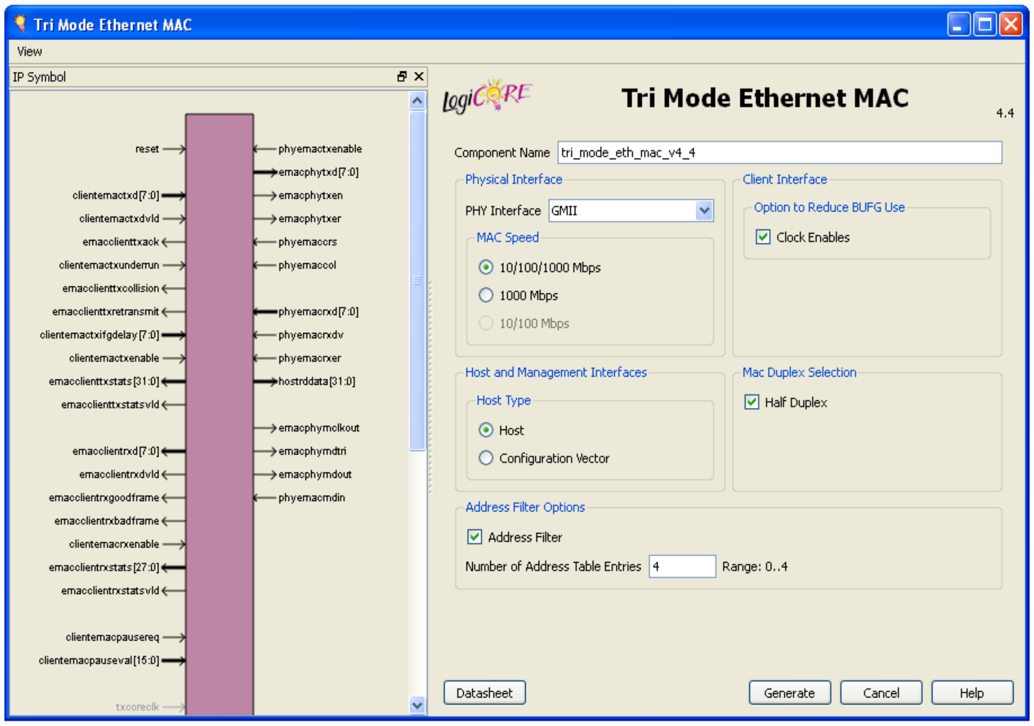
To generate the core:

1. Start the CORE Generator™ software.

   For help starting and using the CORE Generator software, see the documentation supplied with the ISE software, including the *CORE Generator Guide* at 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®-5 device.

   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.



Component Name: tri_mode_eth_mac_v4_4

Physical Interface
PHY Interface: GMII
MAC Speed: [✓ 10/100/1000 Mbps] 1000 Mbps 10/100 Mbps

Client Interface
Option to Reduce BUFG Use
✓ Clock Enables

Host and Management Interfaces
Host Type: [✓ Host] Configuration Vector

Mac Duplex Selection
✓ Half Duplex

Address Filter Options
✓ Address Filter
Number of Address Table Entries: 4 Range: 0..4

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. See "Functional Simulation," page 22.

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 you in running the Xilinx software.

Note: In the following examples, <*project_dir*> is the CORE Generator software 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:

Linux

```
% cd <component_name>/implement
% ./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. 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.

Note: *In the simulation examples that follow, <project_dir>* is the CORE Generator software 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:

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

```
ModelSim: vsim -do simulate_mti.do
IES:./simulate_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:

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

```
ModelSim: vsim -do simulate_mti.do
IES: ./simulate_ncsim.sh
VCS: ./simulate_vcs.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

If the core is generated with a Hardware Evaluation or a Full license, then Timing Simulations are supported. Please ensure that the example design has been implemented before attempting a Timing Simulation.

To run the gate-level simulation, you must have the Xilinx Simulation Libraries compiled for your system. 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.

Note: In the simulation examples that follow, *<project_dir>* is the CORE Generator software project directory; *<component_name>* is the component name entered in the core customization dialog box.

VHDL Simulation

To run a VHDL timing simulation:

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

```
ModelSim: vsim -do simulate_mti.do
IES: ./simulate_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:

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

```
ModelSim: vsim -do simulate_mti.do
IES: ./implement_ncsim.sh
VCS: ./implement_vcs.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 solution in your own design, see the *Tri-Mode Ethernet MAC User Guide*.

Chapter 4

Detailed Example Design

This chapter provides detailed information about the example design, including a description of files and the directory structure generated by the Xilinx® CORE Generator™ software, 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 instanced 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 software 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 software 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 software.
`<component_name>_flist.txt`	Text file that defines all the output files produced when a customized core is generated using the CORE Generator software.
`<component_name>.{veo|vho}`	Verilog or VHDL template for the core. This can be copied into your 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_readme.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`	Tri-Mode Ethernet MAC Data Sheet
`tri_mode_eth_mac_gsg139.pdf`	Tri-Mode Ethernet MAC Getting Started Guide
`tri_mode_eth_mac_ug138.pdf`	Tri-Mode Ethernet MAC User Guide

Back to Top

<component name>/example design

This directory (and subdirectories) contain all of the support files necessary for a VHDL or Verilog implementation of the example design. Table 4-4 defines the HDL files that are always present in this directory. Example designs for certain implementations may contain extra files for clock/clock enable generation logic.

See "Example Design," page 35 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.

Table 4-4: **Example Design Directory *(Continued)***

Name	Description
`<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_module.v[hd]`	Top-level example design instances this to swap the source and destination addresses of the incoming frames
`sync_block.v[hd]`	Synchronizer module, used for passing signals across a clock domain.
`reset_sync.v[hd]`	Local reset synchronizer, used to create a synchronous reset output signal from an asynchronous input.

Back to Top

example design/fifo

This directory contains the files for the FIFO that is instanced 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, MII or RGMII version is delivered by the CORE Generator software depending on the selected option.

Table 4-6: **Physical Directory**

Name	Description
`<project_dir>/<component_name>/example_design/physical`	
`mii_if.v[hd]`	For MII only: all clocking and logic required to provide an MII physical interface
`gmii_if.v[hd]`	For GMII only: all clocking and logic required to provide a GMII physical interface
`rgmii_v2_0_if.v[hd]`	For RGMII only: all clocking and logic required to provide a RGMII v2.0 physical interface
`dcm_reset.v[hd]`	Only present when a DCM is used (Virtex®-4 devices and all Spartan®-3 families). This is a self-contained module for resetting a DCM following Power-on-Reset or loss of lock.

Back to Top

<component name>/implement

This directory is only available when the core is generated with a Hardware Evaluation or a Full license.

This directory contains the support files necessary for implementation of the example design with the XILINX tools. See "Example Design," page 35. 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`	Linux 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
`example_design_xst.xcf`	Constraints file automatically used by XST

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 solution

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`	Cadence IES script file that compiles the example design sources and the structural simulation model and then runs the functional simulation to completion.
`wave_ncsim.sv`	Cadence IES macro file that opens a wave window and adds interesting signals to it.
`simulate_vcs.sh`	VCS script file that compiles the Verilog sources and runs the functional simulation to completion.
`ucli_commands.key`	This file is sourced by VCS at the start of simulation: it configures the simulator.
`vcs_session.tcl`	VCS macro file that opens a wave window and adds signals of interest to it. It is called by the `simulate_vcs.sh` script file.

Back to Top

simulation/timing

This directory is only available when the core is generated with a Hardware Evaluation or a Full license.

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.
`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`	Linux shell script that compiles the VHDL gate-level model and demo test bench and then runs the timing simulation to completion using Cadence IES.
`wave_ncsim.sv`	IES macro file that opens a wave window and adds interesting signals to it. It is used by the simulate_ncsim.sh script.
`simulate_vcs.sh`	VCS script file that compiles the Verilog sources and runs the timing simulation to completion.
`ucli_commands.key`	This file is sourced by VCS at the start of simulation: it configures the simulator.
`vcs_session.tcl`	VCS macro file that opens a wave window and adds signals of interest to it. It is called by the `simulate_vcs.sh` script file.

Back to Top

Implementation and Test Scripts

Implementation Scripts for Timing Simulation

When the CORE Generator software 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.

Linux

```
<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 IES

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

Synopsys VCS

```
<project_dir>/<component_name>/simulation/functional/simulate_vcs.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 software 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 IES

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

Synopsys VCS

```
<project_dir>/<component_name>/simulation/timing/simulate_vcs.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 solution example design.

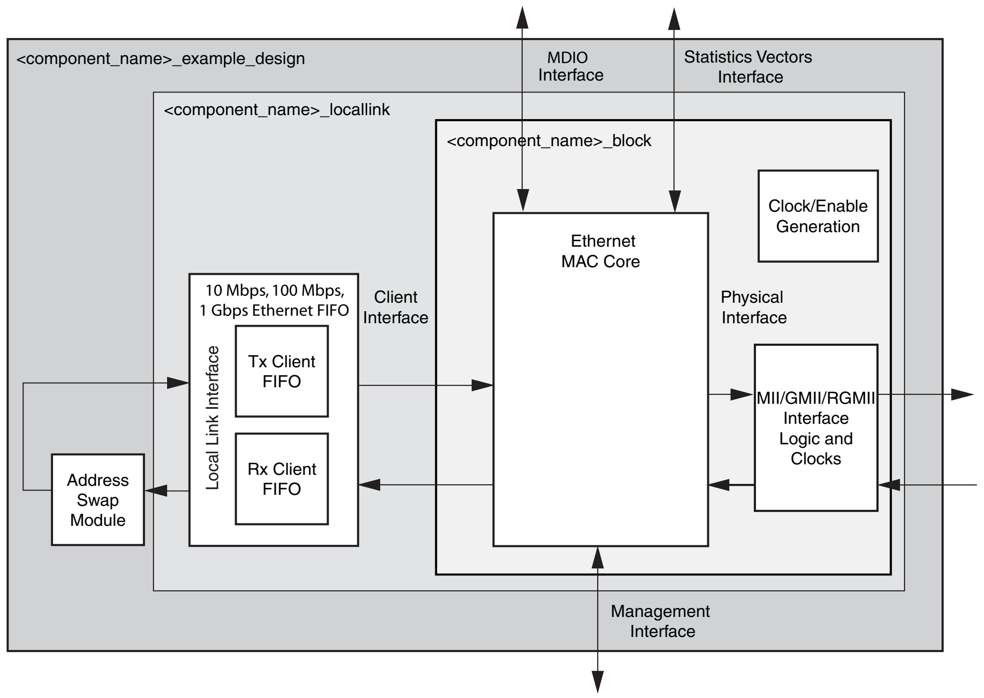


Figure 4-1: **HDL Example Design**

The top-level example design for the TEMAC solution 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 solution
- Clock management logic, including DCM and Global Clock Buffer instances, where required
- MII, 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 solution 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*. In addition, see the "*Parameterizable LocalLink FIFO Application Note*" (XAPP691) for a 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 MAC client side transmitter interface, and an instance of the rx_client_fifo to connect to the MAC 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.

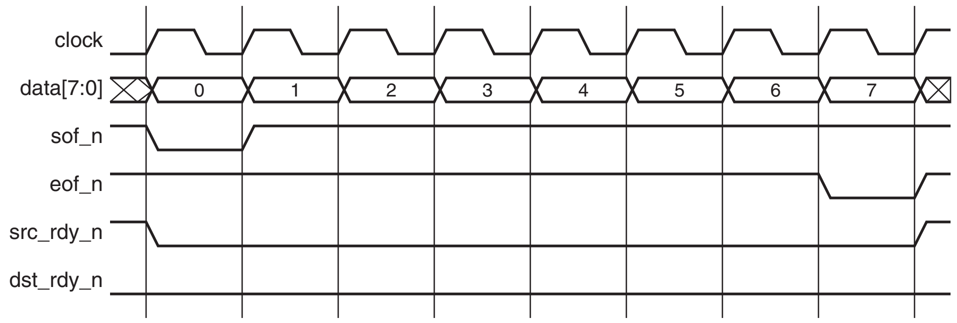


Figure 4-2: **Frame Transfer across LocalLink Interface**

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 solution. If the frame is marked as good, that frame will be presented on the LocalLink interface for reading by you, (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, the FIFO may overflow and data will be lost. It is therefore recommended that the example design is not used with the TEMAC solution 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 MAC, 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 you. 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`

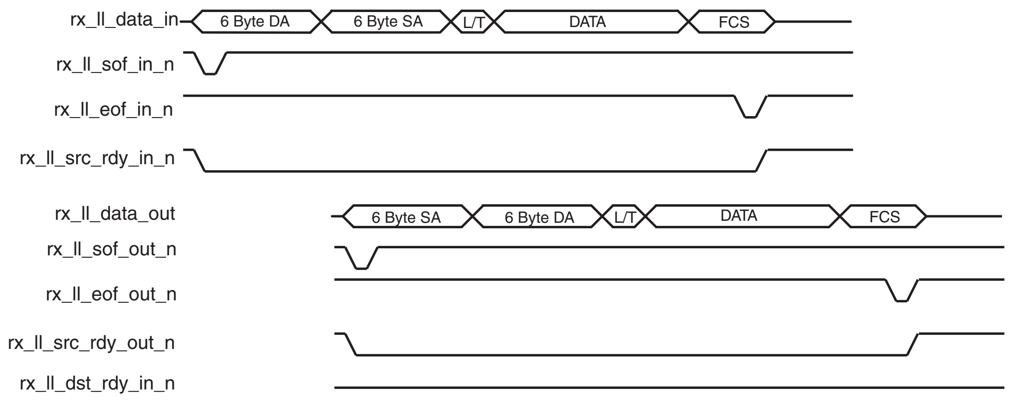


Figure 4-3: **Modification of Frame Data by Address Swap Module**

The address swap module takes frame data from the MAC 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

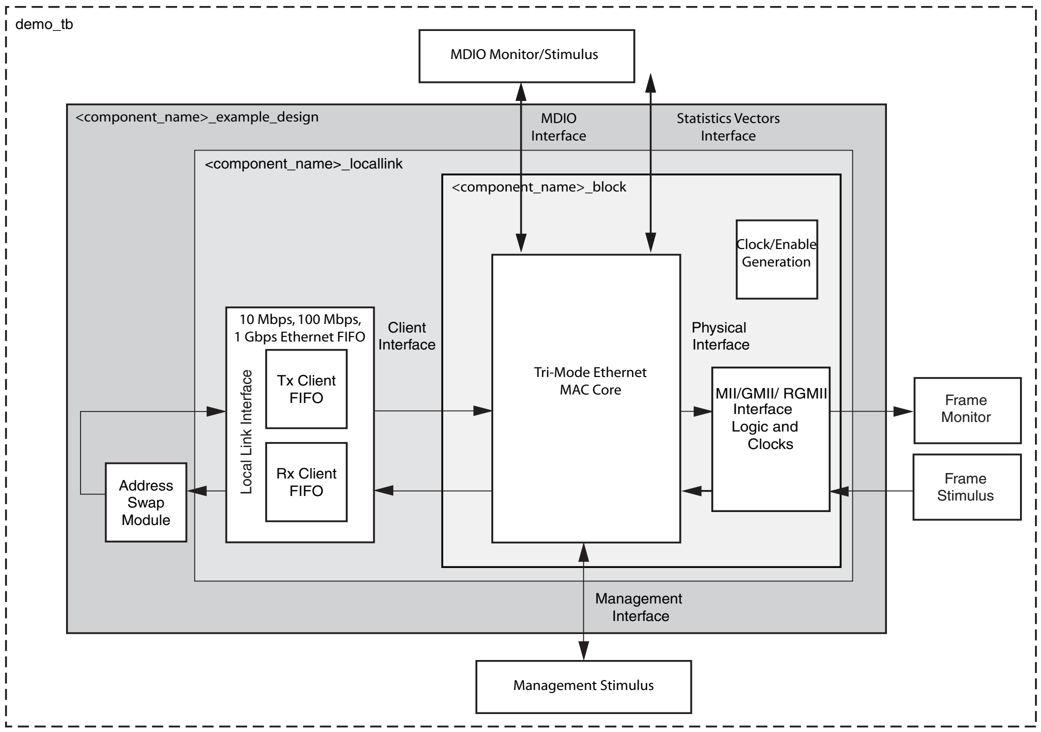


Figure 4-4: **Demonstration Test Bench**

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
- An MDIO monitor/stimulus to check and respond to MDIO accesses, if a management interface is 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 solution 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. The MAC speed is configured to the fastest supported in the configuration selected.
- Four frames are pushed into the GMII/MII or RGMII receiver interface at the fastest MAC speed supported:
  - 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.
- If either the Tri-speed or MII configurations have been selected, the TEMAC solution is configured through the management interface to run at the next fastest available speed. This is 100 Mbps or 10 Mbps respectively. The same four frames are then sent to the MII or RGMII interface and checked against the stimulus frames.
- If the Tri-speed configuration has been selected the TEMAC solution 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 device 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.
- For the Tri-speed configuration, the speed is then changed back to 1 Gbps and the same four frames are sent and checked for a final time. This tests the speed switching between 1 Gbps and 10/100 Mbps in both directions
- If the TEMAC solution is configured to support 10 Mbps then an MDIO read and write access are performed immediately prior to the four frames being sent. The MDIO Monitor will respond to the read with all '1's. The write is expected to have bit 10 set to '0' and all other bits '1'.

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 solution is configured through the configuration vector, disabling flow control and selecting the fastest supported MAC speed
- The stimulus block pushes four frames into the GMII/MII or RGMII receiver interface at the fastest speed supported by the selected configuration:
    - 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.
- If either the Tri-speed or MII configurations have been selected, the TEMAC solution is configured through the configuration vector to run at the next available speed. This is 100 Mbps or 10 Mbps respectively. The same four frames are then sent to the MII or RGMII interface and checked against the stimulus frames.
- If the Tri-speed configuration has been selected the TEMAC solution 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 device 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.
- For the Tri-speed configuration, the speed is then changed back to 1 Gbps and the same four frames are sent and checked for a final time. This tests the speed switching between 1 Gbps and 10/100 Mbps in both directions.

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 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. You 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.

Performing MDIO accesses if 10 Mbps is not supported

If the management interface is selected then the test bench is provided with the required MDIO monitor/stimulus logic. To perform an MDIO access at any other speed simply identify the p_management process and add the following commands after the relative call to the configure_mac task/procedure:

VHDL

```
p_mdio_read(<phy_addr>, <reg_addr>, rd_data_value);
p_mdio_write(<phy_addr>, <reg_addr>, wr_data_value);
```

Verilog

```
mdio_read_task(<phy_addr>, <reg_addr>, rd_data_value);
mdio_write_task(<phy_addr>, <reg_addr>, wr_data_value);
```

The MDIO monitor will only respond to a phy_addr value of 7 with the reg_addr set to 0. Upon an MDIO read the returned data will be all ‘1’s and the monitor will only provide a pass if bit 10 of wr_data_value is set to ‘0’ (all other bits have to be ‘1’).