LogiCORE™ IP 10-Gigabit Ethernet MAC v10.1

Getting Started Guide

UG146 April 19, 2010

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/30/04	1.0	Initial Xilinx release.
04/28/05	1.5	Updated to core version 6.0 and Xilinx® tools 7.1i.
01/18/06	1.6	Updated tools version, release version and date.
07/13/06	2.0	Updated core version to 8.0; Xilinx tools to 8.2i.
09/21/06	2.1	Updated core version to 8.1, release date
02/15/07	2.2	Updated core version to 8.2, Xilinx tools to 9.1i.
08/08/07	2.3	Updated core version to 8.3, Xilinx tools to 9.2i.
3/24/08	2.4	Updated supported tools for ISE® v10.1 release, removed support for Virtex®-II, added support for Synopsys vcs_mxY-2006.06-SP1.
6/27/08	8.5	Advanced the Revision History doc version to 8.5 to match core version.
9/19/08	8.6	Added support for Virtex-5 TXT device; updated core version to 8.6.
04/24/09	9.1	Updated core version to 9.1; Xilinx tools to 11.1; added Virtex-6 device.
06/24/09	9.2	Updated core version to 9.2; Xilinx tools to 11.2.
09/16/09	9.3	Updated core version to 9.3; Xilinx tools to 11.3. Added Virtex-6 HXT, Virtex-6 -1L and Virtex-6 CXT support.
04/19/10	10.1	Updated core version to 10.1; Xilinx tools to 12.1. Added Spartan®-6 support.

Table of Contents

Chapter 4: Detailed Example Design

Schedule of Figures

Chapter 1: Introduction

Chapter 2: Licensing the Core

Chapter 3: Quick Start Example Design

Chapter 4: Detailed Example Design

About This Guide

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

Guide Contents

This guide contains the following chapters:

- Chapter 1, "Introduction" introduces the core and provides references for related material, contact information, technical support, and providing feedback to Xilinx.
- Chapter 2, "Licensing the Core" describes how to install the core and how to obtain and install a license file for the core.
- Chapter 3, "Quick Start Example Design" provides instructions to quickly generate the core and run the example design through implementation and simulation using the default settings.
- Chapter 4, "Detailed Example Design" describes the demonstration test bench in detail and provides directions for how to customize the demonstration test bench for use in an application.

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.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-4 FPGA User Guide*
Blue, underlined text	Hyperlink to a website (URL)	Go to www.xilinx.com for the latest speed files.

XILINX®

Chapter 1

Introduction

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

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

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 10-Gigabit Ethernet MAC core is a Xilinx® CORE Generator™ IP, included in the latest IP Update on the Xilinx IP Center. For detailed information about the core, see the 10-Gigabit Ethernet MAC product page.

For information about licensing options, see Chapter 2, "Licensing the Core."

Recommended Design Experience

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

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

Additional Core Resources

For detailed information and updates about the 10-Gigabit Ethernet MAC core, see the following documents, located on the 10-Gigabit Ethernet MAC product page.

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

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

Technical Support

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

Xilinx will provide technical support for use of this product as described in the *10-Gigabit Ethernet MAC User Guide* and the *10-Gigabit Ethernet MAC Getting Started Guide*. Xilinx cannot guarantee timing, functionality, or support of this product for designs that do not follow these guidelines.

Feedback

Xilinx welcomes comments and suggestions about the 10-Gigabit Ethernet MAC core and the documentation supplied with the core.

For comments or suggestions about the 10-Gigabit Ethernet MAC core, please submit a WebCase from www.xilinx.com/support. Be sure to include the following information:

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

For comments or suggestions about this document, please submit a WebCase from www.xilinx.com/support. Be sure to include the following information:

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

XILINX®

Chapter 2

Licensing the Core

This chapter provides instructions for obtaining a license for the 10-Gigabit Ethernet MAC core, which you must do before using the core in your designs. The 10-Gigabit Ethernet MAC core is provided under the terms of the Xilinx LogiCORE™ Site License Agreement, which conforms to the terms of the SignOnce IP License standard defined by the Common License Consortium. Purchase of the core entitles you to technical support and access to updates for a period of one year.

Before you Begin

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

License Options

The 10-Gigabit Ethernet MAC core 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 10-Gigabit Ethernet MAC core, 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 core using the demonstration test bench provided with the core.

In addition, the license key 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
- Full implementation support including place and route, and bitstream generation
- Full functionality in the programmed device with no timeouts

Obtaining Your License Key

This section contains information about obtaining a simulation, full system hardware evaluation, 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 10-Gigabit Ethernet MAC 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 Key

To obtain a Full license key, you must purchase a license for the core. After you purchase a license, a product entitlement is added to your Product Licensing Account on the Xilinx Product Download and Licensing site. The Product Licensing Account Administrator for your site will receive an email from Xilinx with instructions on how to access a Full license and a link to access the licensing site. You can obtain a full key through your account administrator, or your administrator can give you access so that you can generate your own keys.

Further details can be found at www.xilinx.com/products/ipcenter/ipaccess_fee.htm.

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

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

Introduction

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

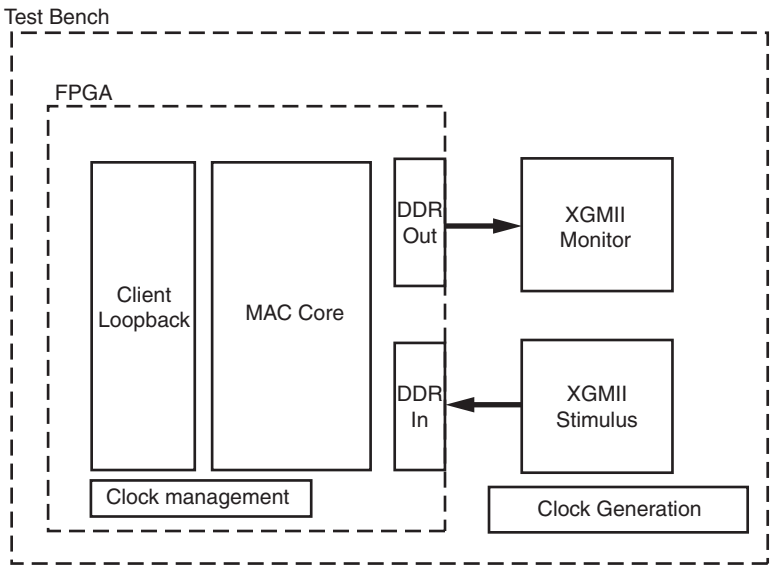

Figure 3-1: **Example Design and Test Bench**

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

The 10-Gigabit Ethernet MAC example design has been tested with Xilinx ISE® software v12.1, Mentor Graphics ModelSim v6.5c, Cadence Incisive Enterprise Simulator (IES) v9.2, and Synopsys VCS and VCS MX 2009.12.

Generating the Core

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

To generate the core:

1. Start the CORE Generator™ software.

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

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

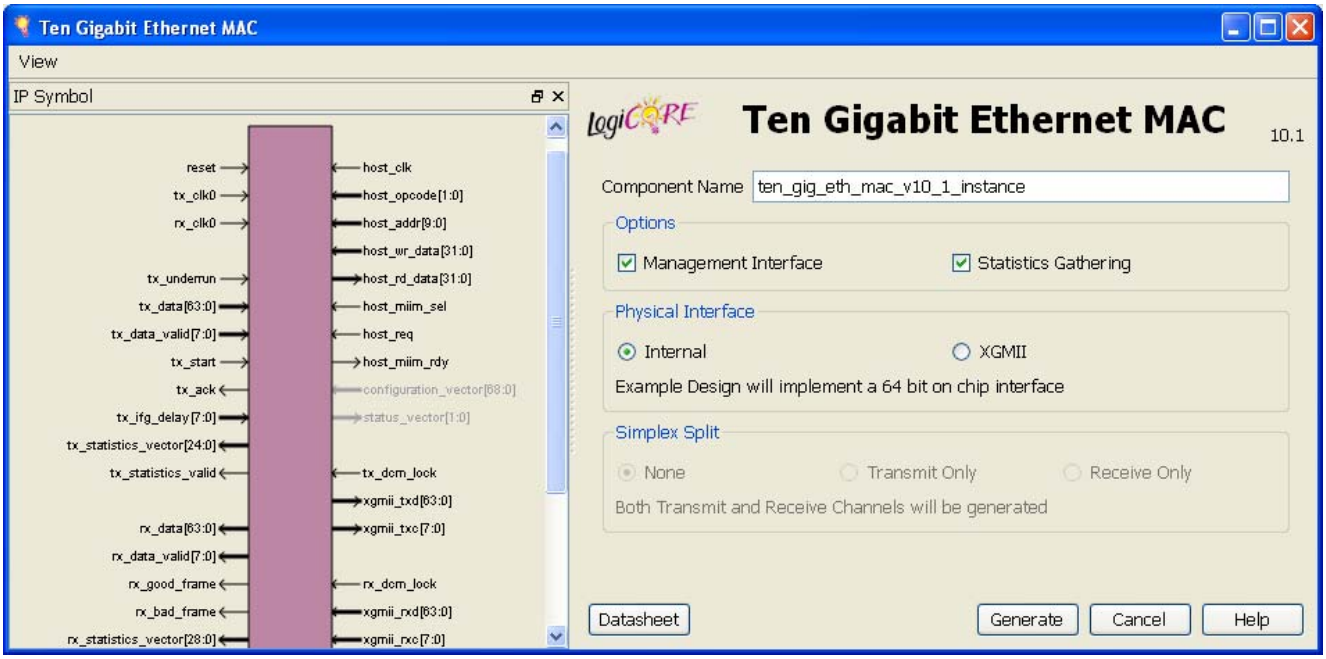

Figure 3-2: **10-Gigabit Ethernet MAC Customization Screen**

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

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

Implementation

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

To implement the example design:

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

Linux

```
% cd quickstart/implement
% ./implement.sh
```

Windows

```
> cd quickstart\implement
> implement.bat
```

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

Simulation

The example design provided with the 10-Gigabit Ethernet MAC core provides a complete environment which allows you to simulate the core and view the outputs. Scripts are provided for pre- and post-layout simulation. The simulation model will either be in VHDL or Verilog depending on the CORE Generator Design Entry project option.

Setting up for Simulation

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

Pre-implementation Simulation

To run a functional simulation of the example design:

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

   ```
   quickstart/simulation/functional
   ```

2. Launch the simulation script:

   ```
   ModelSim: vsim -do simulate_mti.do
   nc-sim: ./simulate_ncsim.sh
   vcs: ./simulate_vcs.sh
   ```

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

Post-implementation Simulation

To run a timing simulation of the example design:

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

   ```
   quickstart/simulation/timing
   ```

2. Launch the simulation script:

   ```
   ModelSim: vsim -do simulate_mti.do
   nc-sim: ./simulate_ncsim.sh
   vcs: ./simulate_vcs.sh
   ```

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

XILINX®

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 instanced in the client_loopback example design
    - <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 10-Gigabit Ethernet MAC 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 or Hardware Evaluation 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 structural simulation model. File used to support VHDL functional simulation of a core. The VHDL model passes customized parameters to the generic core simulation model.
<component_name>.xco	As an output file, the XCO file is a log file which records the settings used to generate a particular core. An XCO file is generated by the CORE Generator 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>.xcp	As an output file, similar to the XCO file, except that it does not specify project-specific settings such as target architecture and output products.
<component_name>_flist.txt	Text file listing all of the output files produced when customized core was generated in the CORE Generator software
<component_name>.{veo|vho}	VHDL or Verilog instantiation template. 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>	
`ten_gig_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	
`ten_gig_eth_mac_ds201.pdf`	10-Gigabit Ethernet MAC Data Sheet
`ten_gig_eth_mac_gsg146.pdf`	10-Gigabit Ethernet MAC Getting Started Guide
`ten_gig_eth_mac_ug148.pdf`	10-Gigabit Ethernet MAC User Guide

Back to Top

<component name>/example design

The example design directory contains the example design files provided with the core.

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

Name	Description
<project_dir>/<component_name>/example_design	
`<component_name>_example_design.v[hd]`	Example design level VHDL or Verilog file for the example design. The LocalLink block is instanced along with the address swap block, if required.
`<component_name>_example_design.ucf`	UCF for the core and the example design
`<component_name>_local_link.vhd`	LocalLink level VHDL file. For cores without a simplex split, this instances the LocalLink FIFO in addition to the block level.
`<component_name>_block.vhd`	Block level VHDL file. This instances the appropriate interface block and the MAC core.

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

Name	Description
address_swap.v[hd]	Address swap VHDL or Verilog file. This connects to the LocalLink interface and swaps the destination and source addresses of frame.
xgmii_if.v[hd]	XGMII interface VHDL or Verilog file. This contains the DDR registers to implement the external XGMII interface. See "10-Gigabit Ethernet MAC with External XGMII Interface," page 30.
physical_if.v[hd]	PHY interface VHDL or Verilog file. This contains the SDR registers to implement the 64-bit interface. See "10-Gigabit Ethernet MAC with 64-bit SDR Interface," page 32.
client_loopback.v[hd]	Client loopback design example instanced in the LocalLink level

Back to Top

example_design/fifo

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

Table 4-5: **FIFO Directory**

Name	Description
<project_dir>/<component_name>/example_design/fifo	
xgmac_fifo.v[hd]	Top-level of the FIFO
xgmac_fifo_pack.vhd	Component declarations for the FIFO
rx_fifo.v[hd]	LocalLink receive FIFO wrapper. This implements the interface to the MAC receiver including frame dropping logic and has a LocalLink interface for the receive channel.
tx_fifo.v[hd]	LocalLink transmit FIFO. This implements the interface to the MAC transmitter including logic to initiate transmission when a full frame is stored in the FIFO and has LocalLink interface.
fifo_ram.v[hd]	Block RAM wrapper used by both FIFOs.

Back to Top

<component name>/implement

The implement directory contains the core implementation script files.

Table 4-6: **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 synthesized.
xst.scr	XST script file for the example design

Back to Top

implement/results

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

Table 4-7: **Results Directory**

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

Back to Top

<component name>/simulation

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

Table 4-8: **Simulation Directory**

Name	Description
<project_dir>/<component_name>/simulation	
demo_tb.v[hd]	VHDL or Verilog demonstration test bench for the 10-Gigabit Ethernet MAC core. More information can either be found in "10-Gigabit Ethernet MAC with External XGMII Interface," page 30 or "10-Gigabit Ethernet MAC with 64-bit SDR Interface," page 32.

Back to Top

simulation/functional

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

Table 4-9: **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 windows 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 example design sources and the structural simulation model then runs the functional simulation to completion using Cadence IES.
wave_ncsim.sv	Cadence IES macro file that opens a wave windows and adds interesting signals to it. It is called used by the simulate_ncsim.sh script.
simulate_vcs.sh (verilog only)	Shell script that compiles the example design sources and the structural simulation model then runs the functional simulation to completion using VCS.
vcs_session.tcl (verilog only)	VCS DVE tcl script that opens a wave window and adds interesting signals to it. This macro is used by the simulate_vcs.sh script.
vcs_commands.key (verilog only)	VCS commands file. This file is called by the simulate_vcs.sh script.

Back to Top

simulation/timing

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

Table 4-10: **Timing Directory**

Name	Description
<project_dir>/<component_name>/simulation/timing	
simulate_mti.do	ModelSim macro file that compiles the VHDL or Verilog timing model and demo test bench then runs the timing simulation to completion.
wave_mti.do	ModelSim macro file that opens a wave windows and adds interesting signals to it. It is called used by the simulate_mti.do macro file.
simulate_ncsim.sh	Linux shell script that compiles the example design sources and the timing model then runs the functional simulation to completion using Cadence IES.
wave_ncsim.sv	Cadence IES macro file that opens a wave windows and adds interesting signals to it. It is called used by the simulate_ncsim.sh script.
simulate_vcs.sh (verilog only)	Shell script that compiles the example design sources and the structural simulation model then runs the timing simulation to completion using VCS.
vcs_session.tcl (verilog only)	VCS DVE tcl script that opens a wave window and adds interesting signals to it. This macro is used by the simulate_vcs.sh script.
vcs_commands.key (verilog only)	VCS commands file. This file is called by the simulate_vcs.sh script.

Back to Top

Implementation and Test Scripts

Implementation Script

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

Linux

project_dir/*component_name*/implement/implement.sh

Windows

project_dir/*component_name*/implement/implement.bat

Full System Hardware Evaluation or Full License Implement Script

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

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

Simulation Only Evaluation License Implement Script

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

Simulation Scripts

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

Functional

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

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

<project_dir>/<component_name>/simulation/functional/simulate_vcs.sh

Timing

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

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

<project_dir>/<component_name>/simulation/timing/simulate_vcs.sh

The scripts perform the following tasks:

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

10-Gigabit Ethernet MAC with External XGMII Interface

Note: External XGMII interface is not supported on Spartan®-6 designs.

Example Design and HDL Wrapper

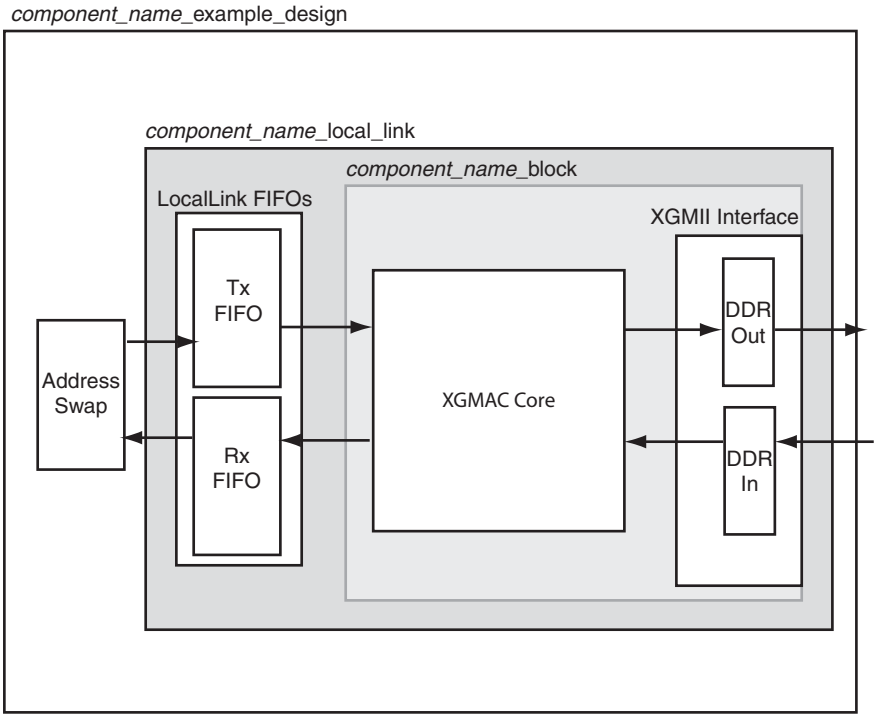

Figure 4-1: **Example Design and HDL Wrapper for 10-Gigabit Ethernet MAC with XGMII Interface**

The example design and HDL wrapper contain the following:

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

The client loopback design performs the following functions:

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

The address swap module performs the following function:

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

The XGMII block performs the following functions:

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

Demonstration Test Bench

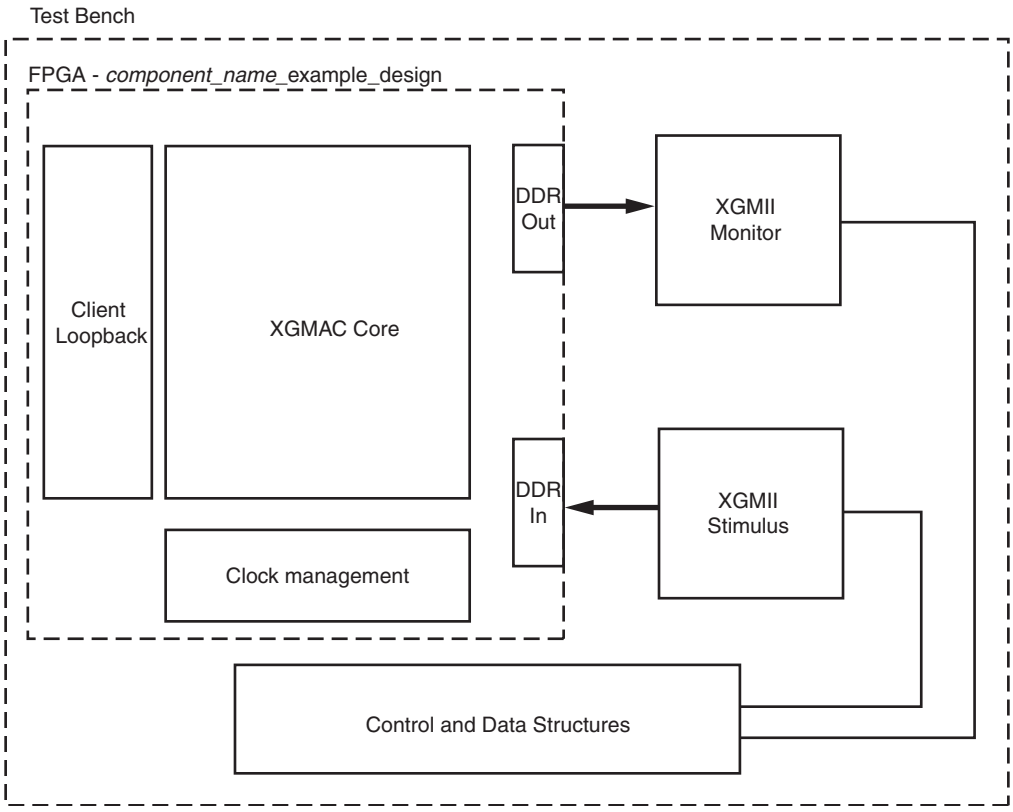

Figure 4-2: **Demonstration Test Bench for 10-Gigabit Ethernet MAC with XGMII Interface**

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

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

10-Gigabit Ethernet MAC with 64-bit SDR Interface

Example Design and HDL Wrapper

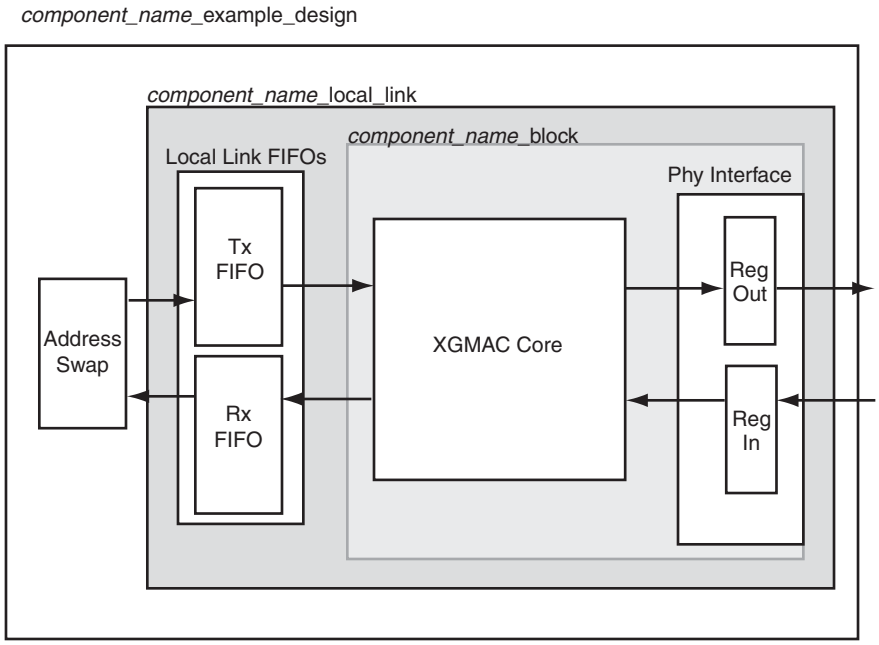

Figure 4-3: **Example Design and HDL Wrapper for 10-Gigabit Ethernet MAC with 64-bit Interface**

The example design and HDL wrappers contain the following:

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

The client loopback design performs the following functions:

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

The address swap module performs the following function:

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

The physical interface block performs the following functions:

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

Demonstration Test Bench

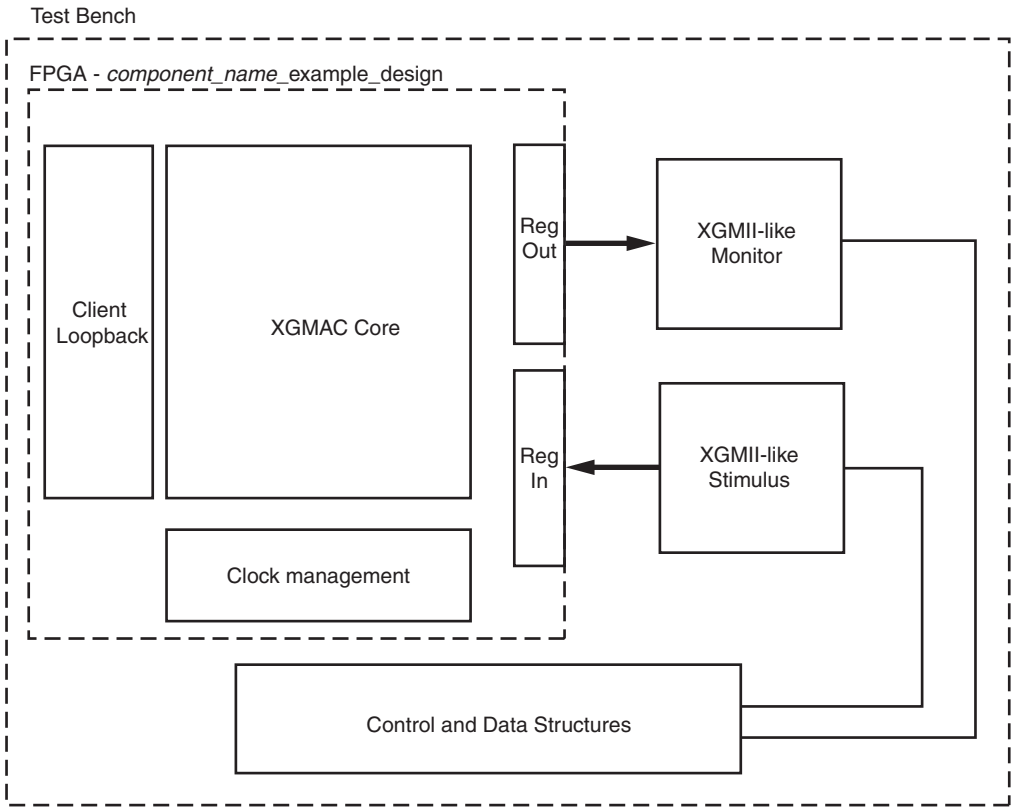

Figure 4-4: **Demonstration Test Bench for 10-Gigabit Ethernet MAC with 64-bit Interface**

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

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

Transmit-Only Cores

Example Design and HDL Wrapper

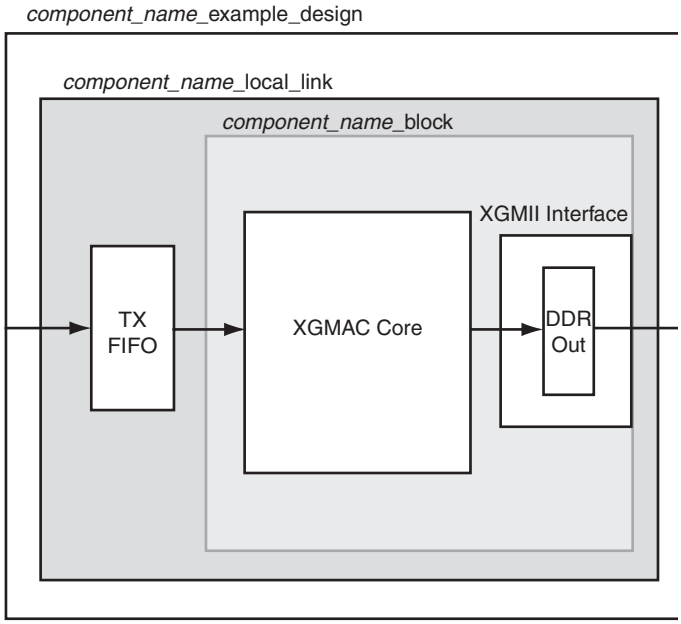

Figure 4-5: **HDL Wrapper for 10-Gigabit Ethernet MAC Transmit-Only Configurations**

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

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

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

Demonstration Test Bench

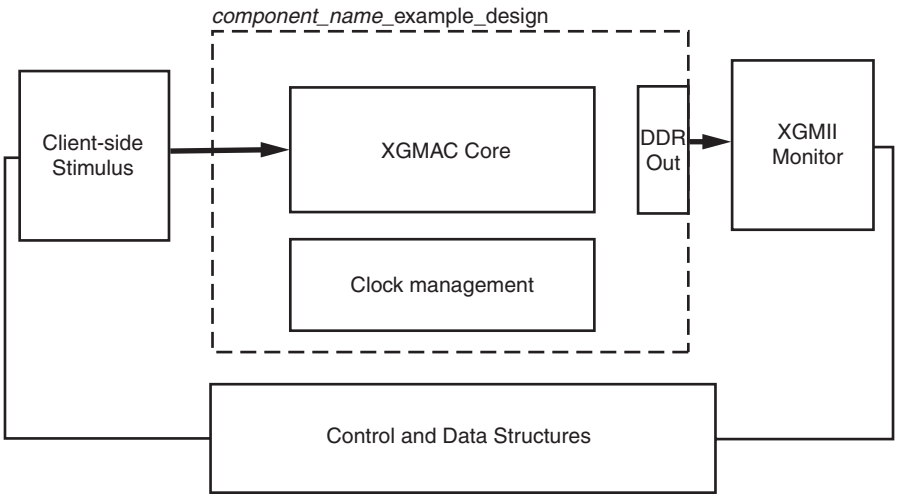

Figure 4-6: **Demonstration Test Bench for 10-Gigabit Ethernet MAC: Transmit Only**

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

Receive-Only Cores

Example Design and HDL Wrapper

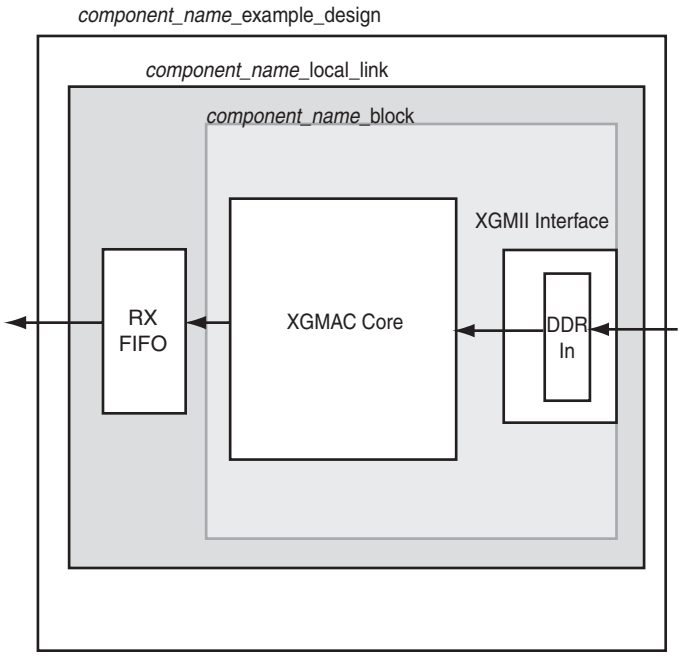

Figure 4-7: **HDL Wrapper for 10-Gigabit Ethernet MAC: Receive Only**

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

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

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

Demonstration Test Bench

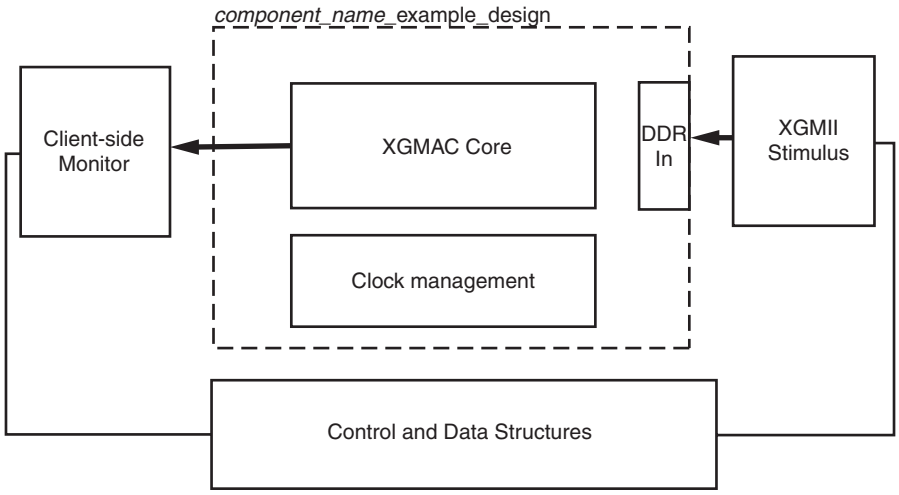

Figure 4-8: **Demonstration Test Bench for 10-Gigabit Ethernet MAC: Receive Only**

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

Overview of LocalLink Interface

Data is transferred on the LocalLink interface from source to destination, with the flow governed by the four active low control signals `sof_n`, `eof_n`, `src_rdy_n` and `dst_rdy_n`. The flow of data is controlled by the `src_rdy_n` and `dst_rdy_n` signals. Only when these signals are asserted simultaneously is data transferred from source to destination. The individual packet boundaries are marked by the `sof_n` and `eof_n` signals. The `rem[2:0]` signal is used to indicate the position of the sof, always denoted by '001' coincident with `sof_n` being asserted. When `eof_n` is asserted, `rem[2:0]` is encoded to show which of the 8 bytes contain valid frame data (see Table 4-11). For more information on the LocalLink interface see:

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

Figure 4-9 shows the transfer of a frame without flow control.

Table 4-11: **Remainder Binary Encoding**

rem[2:0]	Bytes Valid[7:0]	Data Valid
000	00000001	[7:0]
001	00000011	[15:0]
010	00000111	[23:0]
011	00001111	[31:0]
100	00011111	[39:0]
101	00111111	[47:0]
110	01111111	[55:0]
111	11111111	[63:0]

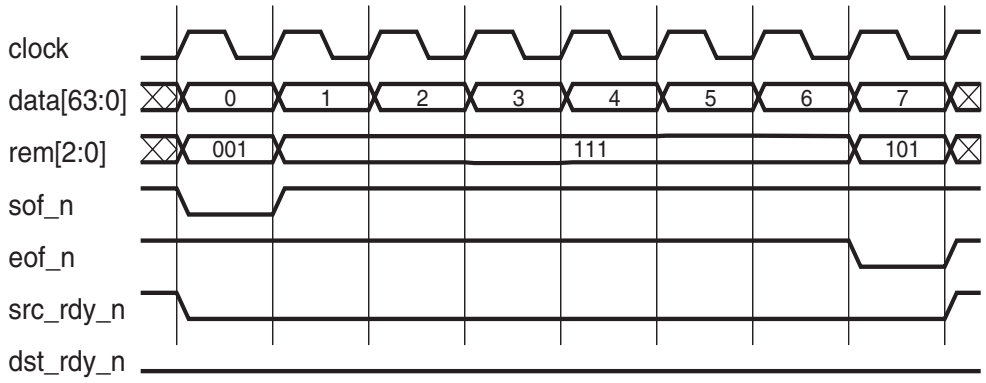

Figure 4-9: **Frame Transfer Across LocalLink Interface**

Figure 4-10 illustrates frame transfer of a frame, where both the `src_rdy_n` and `dst_rdy_n` signals are used to control the flow of data across the interface.

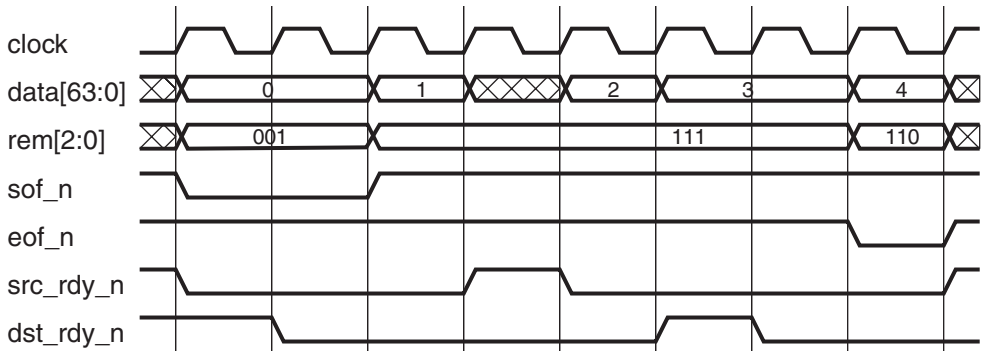

Figure 4-10: **Frame Transfer With Flow Control**