LogiCORE™ IP Ethernet Statistics v3.4

Getting Started Guide

UG169 April 19, 2010

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/28/05	1.0	Initial Xilinx® release.
01/18/06	1.1	Updated to Ethernet Statistics version 1.2, Xilinx tools v8.1i.
07/13/06	1.2	Updated to Ethernet Statistics version 2.1, Xilinx tools v8.2i.
10/23/06	1.3	Updated to core version 2.2, added Spartan® 3-A and Virtex®-5 LX FPGA families support.
02/15/07	1.4	Updated to core version 2.3, Xilinx tools 9.1i.
08/08/07	1.5	Updated to core version 2.4, Xilinx tools 9.2i.
03/24/08	1.6	Updated to core version 2.5, Xilinx tools 10.1.
04/24/09	1.7	Updated to core version 3.1, Xilinx tools 11.1. Added support for Spartan-6 and Virtex-6 FPGA families
06/24/09	1.8	Updated to core version 3.2, Xilinx tools 11.2.
09/16/09	1.9	Updated to core version 3.3, Xilinx tools 11.3.
04/19/10	2.0	Updated to core version 3.4, Xilinx tools 12.1.

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 *Ethernet Statistics v3.4 Getting Started Guide* provides the following information about generating an Ethernet Statistics core, customizing and simulating the core utilizing the provided example design, and running the design files through implementation using the Xilinx tools.

Guide Contents

The following chapters are included in this guide:

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

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 linking 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
FPGA	Field Programmable Gate Array.
HDL	Hardware Description Language
IES	Incisive Enterprise Simulator
IP	Intellectual Property
ISE®	Integrated Software Environment
MAC	Media Access Controller
SDF	Standard Delay Format
UCF	User Constraint File
VHDL	VHSIC Hardware Description Language (VHSIC an acronym for Very High-Speed Integrated Circuits).
VCS	Verilog Compiled Simulator
XCO	Xilinx CORE Generator™ software core source file
XST	Xilinx Synthesis Technology

XILINX®

Chapter 1

Introduction

The Xilinx® LogiCORE™ IP Ethernet Statistics core is a fully-verified solution that supports Verilog HDL and VHDL. In addition, the example design described in this guide is provided in both Verilog and VHDL.

This chapter introduces the Ethernet Statistics core and provides related information, including recommended design experience, additional resources, technical support, and ways to submit 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® v12.1

About the Core

The Ethernet Statistics core is a Xilinx CORE Generator™ IP software core, included in the latest IP Update on the Xilinx IP Center.

For detailed information about the core, see www.xilinx.com/products/ipcenter/ETHERNET_STATS.htm.

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

Recommended Design Experience

Although the Ethernet Statistics 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 the Xilinx implementation software and user constraints file (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 Ethernet Statistics core, see the related documents, located on the Ethernet Statistics product page at: www.xilinx.com/products/ipcenter/ETHERNET_STATS.htm.

- Ethernet Statistics *Data Sheet*
- Ethernet Statistics *Release Notes*
- Ethernet Statistics *User Guide*

For updates to this guide, see the *Ethernet Statistics Getting Started Guide*, also located on the Ethernet Statistics product page.

Technical Support

For technical support, go to www.xilinx.com/support. Questions are routed to a team of engineers with expertise using the Ethernet Statistics core.

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

Feedback

Xilinx welcomes comments and suggestions about the Ethernet Statistics core and the documentation provided with the core.

Ethernet Statistics Core

For comments or suggestions about the Ethernet Statistics 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

Document

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

Chapter 2

Licensing the Core

This chapter provides instructions for obtaining a license for the Ethernet Statistics core, which you must do before using the core in your designs. The Ethernet Statistics core is provided under the terms of the Xilinx End User License Agreement, which conforms to the terms of the SignOnce IP License standard defined by the Common License Consortium.

Before you Begin

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

License Options

The Ethernet Statistics core provides two 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 Ethernet Statistics 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

The Full license key 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 Key

This section contains information about obtaining a license key for both licensing options.

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.

Obtaining a Full License Key

To obtain a full license key:

1. Navigate to the product page for this core: www.xilinx.com/products/ipcenter/ETHERNET_STATS.htm
2. Click **Get License**.
3. Follow the instructions to install the required Xilinx ISE software and IP updates and generate a Full license key.

Installing the License File

The Simulation Only Evaluation license key is provided with the ISE CORE Generator system and does not require installation of an additional license file. For 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.

XILINX®

Chapter 3

Quick Start Example Design

The instructions provided in this chapter help you to quickly generate an Ethernet Statistics core, run the core and example design provided through implementation with the Xilinx® tools, and simulate the example design utilizing the provided demonstration test bench. For detailed information about the example design, see Chapter 4, "Detailed Example Design,"

Overview

The Ethernet Statistics example design consists of the following:

- Ethernet Statistics core netlist
- HDL block wrapper and HDL vector decoder files, the part of the example design that should be instantiated in a customer design
- Example design HDL top-level for implementation in a device as a standalone design
- Demonstration test bench to exercise the example design

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

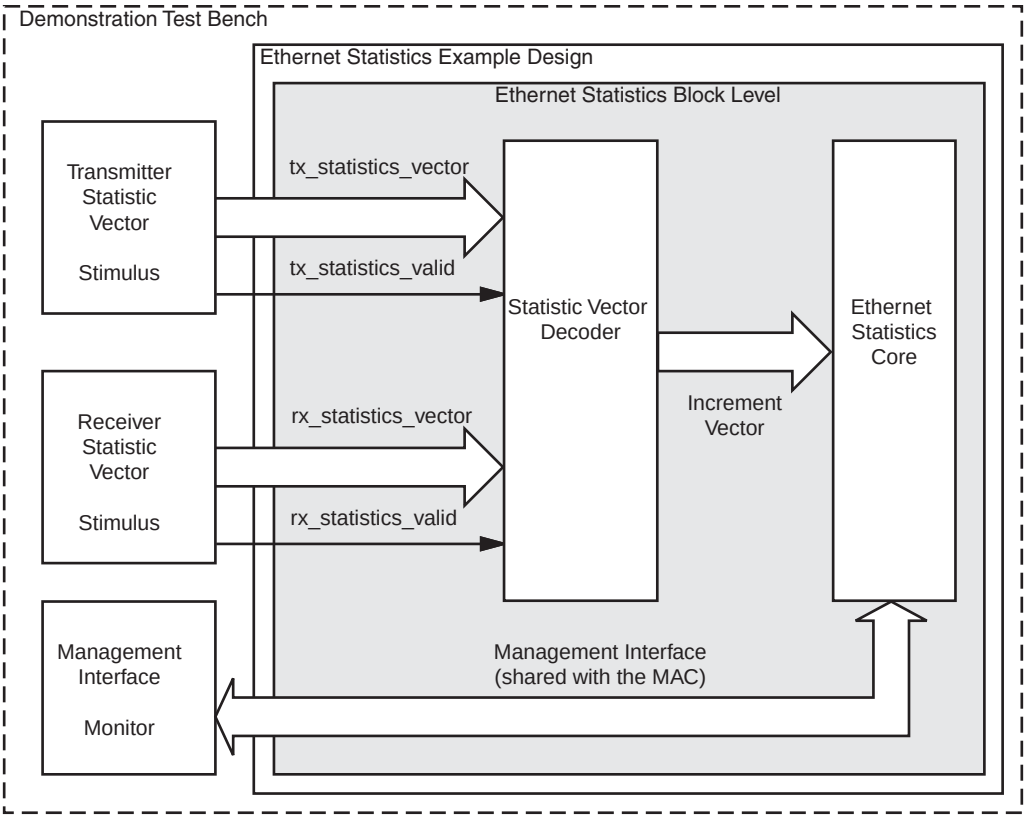

Figure 3-1: **Ethernet Statistics Example Design and Test Bench**

Generating the Core

Use the following steps to generate the Ethernet Statistics example design.

1. Start the CORE Generator™ software.

   For help starting and using the CORE Generator software, see the documentation supplied with ISE software at www.xilinx.com/support/software_manuals.htm.

2. Create a new project, either by choosing File > New Project or by clicking Create a New Project at the right side of the CORE Generator software window. After naming the project, the Project Options dialog box appears.

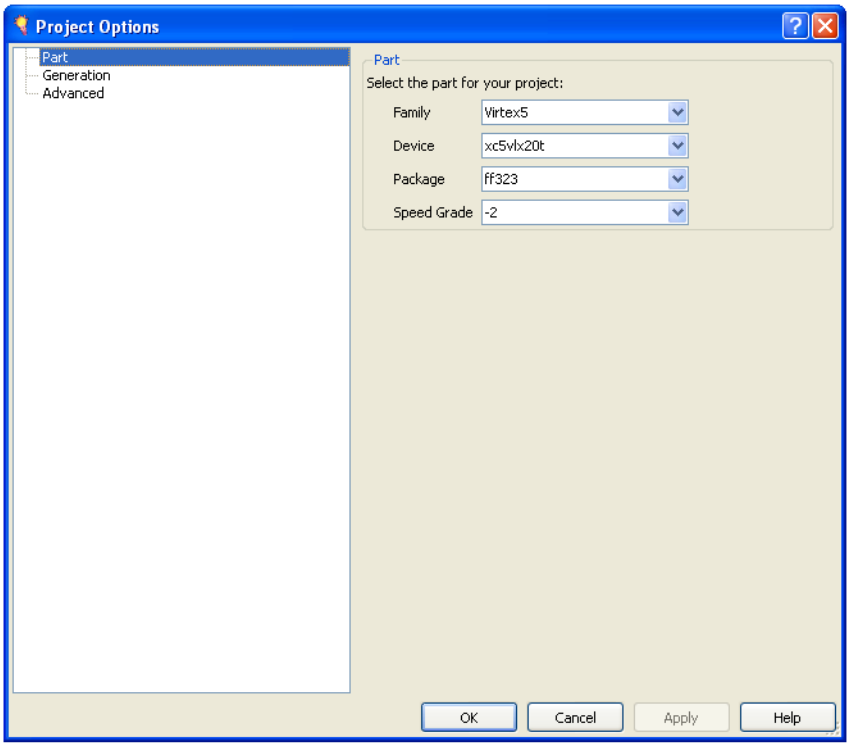

Figure 3-2: **Project Options**

3. From the Family drop-down list, choose a Virtex®-5 FPGA part, which will be used to generate an Ethernet Statistics core using the default parameters.

4. Click the Generation tab; then select either VHDL or Verilog for Design Entry. For Vendor, select Other, and then click OK.

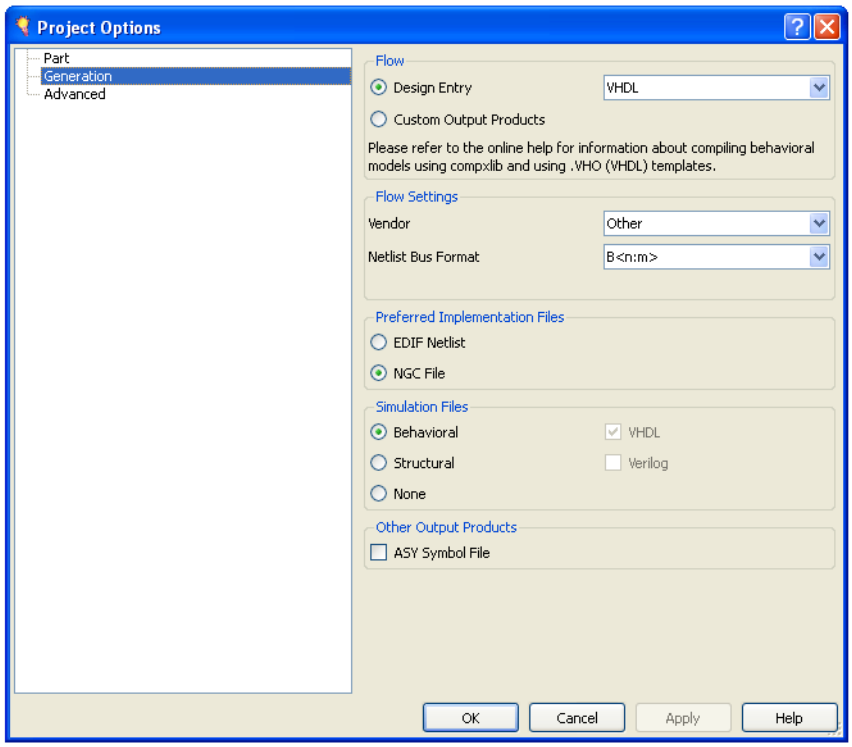

Figure 3-3: **Project Generation Options**

5. Locate the Ethernet Statistics core in the taxonomy tree, listed under one of the following:
   - Communications & Networking/Ethernet
   - Communications & Networking/Networking
   - Communications & Networking/Telecommunications
6. Double-click the core name. A warning may appear related to the limitations of the Simulation Only Evaluation license.

7. Click OK to display the Ethernet Statistics customization screen.

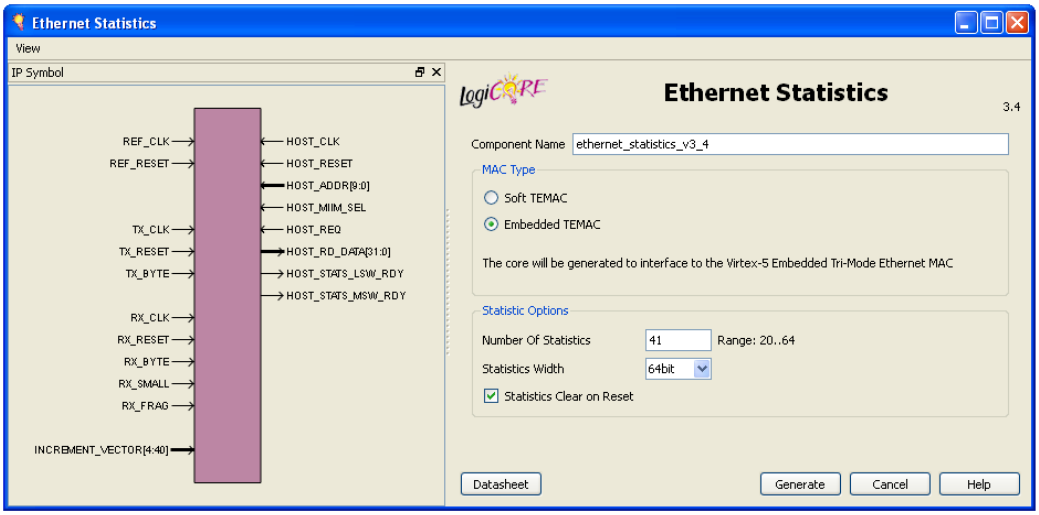

Figure 3-4: **Customization Screen**

8. In the Component Name field, enter a name for the core instance; then click Finish to generate the core using the default parameters. The core and 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,".

Implementing the Example Design

Note: This is available only with a Full license.

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

From the CORE Generator software project directory window, type the following to implement the Ethernet Statistics example design:

For LINUX

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

For Windows

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

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

Simulating the Example Design

Setting up for Simulation

To run the gate-level simulation you must have the Xilinx Simulation Libraries compiled for your system. Please see "Compiling Xilinx Simulation Libraries (COMPXLIB)," located in the *Xilinx ISE Synthesis and Verification Design Guide* (an ISE software manual) available from www.xilinx.com/support/software_manuals.htm.

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 window.

Functional Simulation

Note: Available for both the Simulation Only Evaluation and Full license.

Instructions for running a functional simulation of the Ethernet Statistics core using either VHDL or Verilog are provided in this section. The functional simulation model is provided when the core is generated; therefore, implementing the core before simulating the functional model is not required.

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

- Open a command prompt or shell and set the current directory to

  `<project_dir>/<component_name>/simulation/functional`
- Launch the simulation script:

  `ModelSim:` **`vsim -do simulate_mti.do`**

  `IES:` **`./simulate_ncsim.sh`**

  `VCS:` **`./simulate_vcs.sh`** `(Verilog only)`

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

Timing Simulation

Note: This is available only with a Full license.

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

- Run the implementation script (see "Implementing the Example Design").
- Open a command prompt or shell, then set the current directory to

  `<project_dir>/<component_name>/simulation/timing`
- Launch the simulation script:

  `ModelSim:` **`vsim -do simulate_mti.do`**

  `IES:` **`./simulate_ncsim.sh`**

  `VCS:` **`./simulate_vcs.sh`** `(Verilog only)`

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

What's Next?

For detailed information about the example design, including guidelines for modifying the design and extending the test bench, see Chapter 4, "Detailed Example Design,"

To start using the Ethernet Statistics core in your own designs, see the *Ethernet Statistics User Guide*.

XILINX®

Chapter 4

Detailed Example Design

This chapter provides detailed information about the Ethernet Statistics 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.

Directory Structure and File Descriptions

Click one of the following directory names to go to the desired directory and its associated files.

- **<project directory>**
  Top-level project directory
  - <project directory>/<component name>
    Core release notes file
    - <component name>/doc
      Product documentation
    - <component name>/example_design
      Verilog and VHDL design files
    - <component name>/implement
      Implementation script files
      - implement/results
        Created after implementation scripts are run and contain implement script results
    - <component name>/simulation
      Simulation scripts
      - simulation/functional
        Functional simulation files
      - simulation/timing
        Timing simulation files

Directory and File Contents

The Ethernet Statistics core directories and their associated files are defined in the following subsections.

<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	The Xilinx netlist for the core, instantiated by the example design.
<component_name>.v[hd]	The UniSim-based simulation model used to support the functional simulation of the core.
<component_name>.v{ho|eo}	Instantiation template for the core.
<component_name>.xco	A log file that records the settings used to generate a specific core. An XCO file is generated by the CORE Generator software for each core it creates in the current project directory. An XCO file can also be used as an input to the CORE Generator tool.
<component_name>_flist.txt	A test file listing all the output files produced when a customized core is generated by the CORE Generator software.

Back to Top

<project directory>/<component name>

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

Table 4-2: **Component Name Directory**

Name	Description
<project_dir>/<component_name>	
ethernet_statistics_readme.txt	The core release notes text document.

Back to Top

<component name>/doc

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

Table 4-3: **Doc Directory**

Name	Description
<project_dir>/<component_name>/doc	
ethernet_statistics_ds323.pdf	The *Ethernet Statistics Data Sheet*.
ethernet_statistics_gsg169.pdf	The *Ethernet Statistics Getting Started Guide.*
ethernet_statistics_ug170.pdf	The *Ethernet Statistics User Guide.*

Back to Top

<component name>/example_design

The example design directory contains all the supporting Verilog or VHDL example design files included with the core.

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

Name	Description
<project_dir>/<component_name>/example_design	
<component_name>_example_design.v[hd]	Top-level file for the example design, which allows the example design to be implemented in a device as a standalone design.
<component_name>_block.v[hd]	The block-level file for the core. This is the useful part of the example design that should be instantiated in customer designs.
vector_decode.v[hd]	The Ethernet MAC statistics vector decoder module instantiated from the block level. This entity defines the exact rules that increment each counter.
<component_name>_example_design.ucf	An example UCF for the example design included with the core.

Back to Top

<component name>/implement

The implement directory contains all the implementation script files and is present only when the Full license is in use.

Table 4-5: **Implement Directory**

Name	Description
<project_dir>/<component_name>/implement	
implement.bat	Windows batch file that process the example design through the Xilinx tool flow.
implement.sh	LINUX shell script that processes the example design through the Xilinx tool flow.
xst.prj	The XST project file for the example design, which enumerates all the VHDL files that need to be synthesized.
xst.scr	XST example design script file.

Back to Top

implement/results

The results directory is created by the implement script; implement script results are placed in the `results` directory.

Table 4-6: **Results Directory**

Name	Description
<project_dir>/<component_name>/implement/results	
These files are created by the implement scripts and contain the back-annotated Verilog or VHDL files.	

Back to Top

<component name>/simulation

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

Table 4-7: **Simulation Directory**

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

Back to Top

simulation/functional

The functional directory contains lists of files for various types of simulation.

Table 4-8: **Functional Directory**

Name	Description
<project_dir>/<component_name>/simulation/functional	
simulate_mti.do	A ModelSim macro file that compiles the Verilog or VHDL sources and then runs the functional simulation to completion.
wave_mti.do	A ModelSim macro (called by the simulate_mti.do macro) that opens a wave window and adds signals of interest.
simulate_ncsim.sh	An IES script that compiles the Verilog or VHDL sources and then runs the functional simulation to completion.
wave_ncsim.sv	An IES macro (called by the simulate_ncsim.sh script) that opens a wave window and adds signals of interest.
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

The timing directory contains additional timing files and is present only when the Full license is in use.

Table 4-9: **Functional Directory**

Name	Description
<project_dir>/<component_name>/simulation/timing	
simulate_mti.do	A ModelSim macro that compiles the Verilog or VHDL sources, and then runs the timing simulation to completion.
wave_mti.do	A ModelSim macro (called by the simulate_mti.do macro) that opens a wave window and adds signals of interest.
simulate_ncsim.sh	An IES script that compiles the Verilog or VHDL sources and then runs the timing simulation to completion.

Table 4-9: **Functional Directory *(Continued)***

Name	Description
wave_ncsim.sv	An IES macro (called by the simulate_ncsim.sh script) that opens a wave window and adds signals of interest.
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 Scripts

Note: Present only with a Full license.

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

LINUX

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

Windows

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

The implement script performs the following steps:

- Synthesizes the HDL example design files using XST
- Runs NGDbuild to consolidate the core netlist and the example design netlist into the NGD file containing the entire design
- Maps the design to the target technology
- Place-and-routes the design on the target device
- Performs static timing analysis on the routed design using Timing Analyzer (TRCE)
- Generates a bitstream
- Enables Netgen to run on the routed design to generate a VHDL or Verilog netlist (as appropriate for the Design Entry project setting) and timing information in the form of SDF files

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

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

Simulation Scripts

Functional Simulation

The test script, either a ModelSim, IES or VCS macro, automates the simulation of the test bench. The scripts are available from the following location:

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

The test script performs the following tasks:

- Compiles the structural UniSim simulation model
- Compiles HDL Example Design source code
- Compiles the demonstration test bench
- Starts a simulation of the test bench
- Opens a Wave window and adds signals of interest (wave_mti.do/wave_ncsim.sv)
- Runs the simulation to completion

Timing Simulation

Note: Present only with a Full license.

The test script, either a ModelSim, IES or VCS macro, automates the simulation of the test bench. The scripts are available from the following location:

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

The test script performs the following tasks:

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

Example Design

Top Level Example Design

The following files describe the top-level example design for the Ethernet Statistics core.

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 top-level example design adds input and output flip-flops to the block level allowing the entire design to be synthesized and implemented in a target device to provide post place-and-route gate-level *simulation.*

Block Level

The following files describe the block-level design for the Ethernet Statistics core.

VHDL

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

Verilog

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

The HDL block-level design contains the following

- An instance of the Ethernet Statistics core
- A Vector Decoder module which interprets the transmitter and receiver statistic vectors from the chosen Ethernet MAC

The block-level design should be instantiated in customer designs to connect to the chosen Ethernet MAC, as illustrated in *Figure 4-1*.

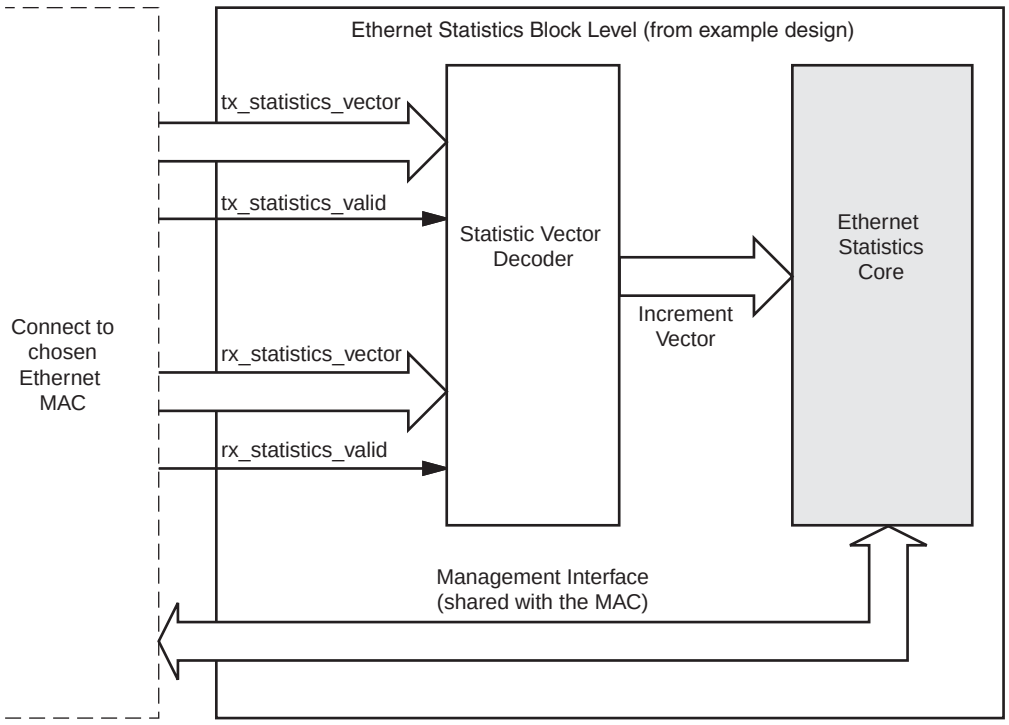

Figure 4-1: **Example Design Top-Level HDL for Ethernet Statistics Core**

Vector Decoder

The Vector Decoder is described in the following files:

VHDL

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

Verilog

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

Transmitter and Receiver Statistic Vectors from the connected Ethernet MAC are routed into the Statistic Vector Decoder module. This module decodes the vectors and implements the logic to derive each of the statistics counters. As such, this can be easily modified to create statistics counters for specific applications.

The Statistic Vector Decoder module passes generic increment signals into the statistics core where the counter values increment and are stored. See the *Ethernet Statistics User Guide* for additional information about this module.

Demonstration Test Bench

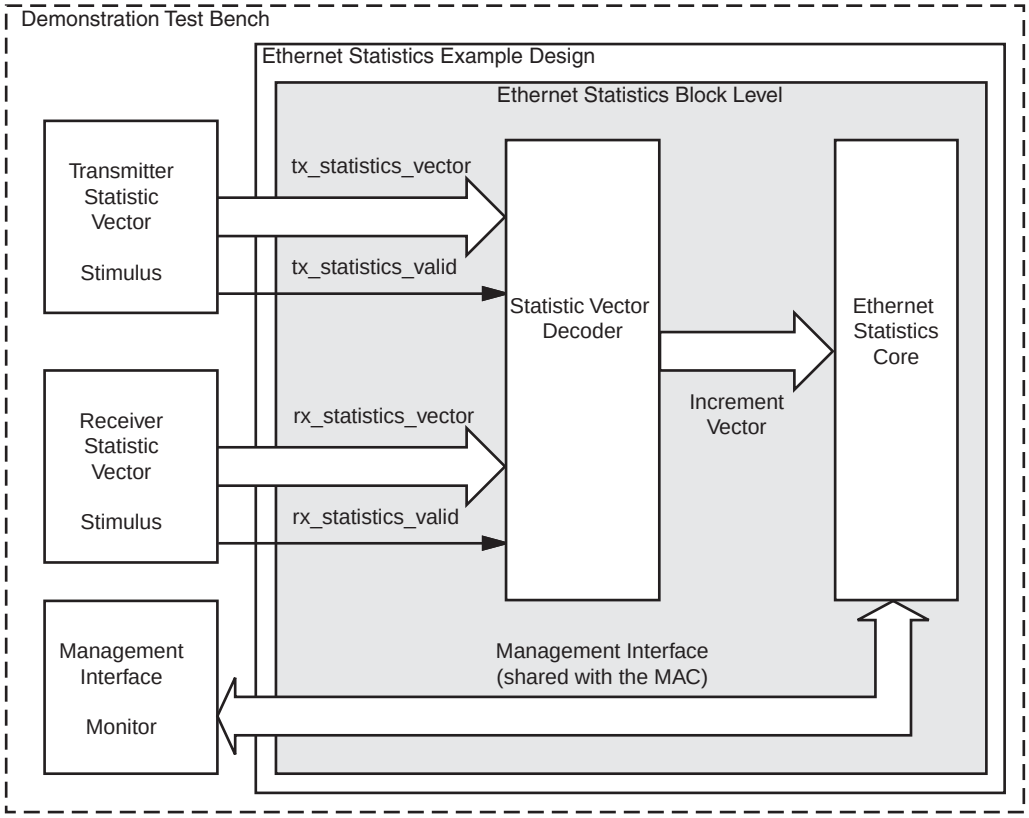

Figure 4-2: **Demonstration Test Bench for Ethernet Statistics Core and Example Design**

The following files describe the demonstration test bench:

VHDL

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

Verilog

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

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

The demonstration test bench performs the following tasks:

- Generates input clock signals
- Applies a reset to the example design
- Injects four Transmitter Statistic Vectors into the example design by the transmitter stimulus block. These simulate the following frames:
  - The first frame is a minimum-length error free frame
  - The second frame is a 65-byte frame, which is error free
  - The third frame is a minimum-length frame, which has been errored by a TX_UNDERRUN assertion by the connected MAC
  - The fourth frame is a minimum-length error free frame with a Broadcast Address
- Injects four Receiver Statistic Vectors are injected into the example design by the receiver stimulus block. These simulate the following frames:
  - The first frame is a minimum-length error free frame
  - The second frame is a minimum-length frame which is contains an Frame Check Sequence error
  - The third frame is a 65-byte frame which is error free
  - The fourth frame is a minimum-length error free frame with a Multicast Address
- After the Vectors are injected, the Management Interface performs a read from all statistic addresses. The following statistics are checked to ensure that the values read from the core are correct:
  - Received bytes
  - Frames Received OK
  - Frame Check Sequence Errors
  - Transmitted bytes
  - Frames Transmitted OK
  - Underrun Errors

Customizing the Test Bench

The core should be verified in a *complete* system when connected to the desired Ethernet MAC. However, when starting, follow these guidelines to modify the demonstration test bench provided with the core:

- It is possible to change the contents of the four transmitter and receiver statistic vectors which are injected into the core. The statistic checking performed during the Management read process should automatically update for the six statistics which are checked.
- New vectors can be added by defining new vectors of data. Make sure that the contents of these new vectors are passed into `statistic_check` function.
- Extra statistical checking can be added to the `statistic_check` function. Follow the six statistic examples already present.
- Clock frequencies applied to the core can be quickly modified for further testing.