DISCONTINUED PRODUCT

LogiCORE™ IP VLYNQ™ v1.4

Getting Started Guide

UG171 April 24, 2009

XILINX®

Xilinx is providing this product documentation, hereinafter “Information,” to you “AS IS” with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.

© 2006- 2009 Xilinx, Inc. XILINX, the Xilinx logo, Virtex, Spartan, ISE and other designated brands included herein are trademarks of Xilinx in the United States and other countries. VLYNQ is a trademark of Texas Instruments and used under license. All other trademarks are the property of their respective owners.

Revision History

The following table shows the revision history for the VLYNQ Getting Started Guide.

	Version	Revision
4/28/05	1.0	Early access release.
7/13/06	1.1	Initial Xilinx release.
9/21/06	1.2	Updated to version 1.2.
3/24/08	1.3	Updated to version 1.3.
04/24/09	1.4	Updated to version 1.4.
The product was discontinued as of October 31, 2009.		

Table of Contents

Chapter 4: Detailed Example Design

XILINX®

Preface

About This Guide

The LogiCORE™ IP *VLYNQ™ Getting Started Guide* provides information about generating a VLYNQ interface 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 this guide and the conventions used in this guide.
- Chapter 1, "Introduction" describes the core and related information, including recommended design experience, system requirements, technical support, and submitting feedback to Xilinx.
- Chapter 2, "Licensing the Core" provides information about 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.
- Chapter 4, "Detailed Example Design" provides detailed information about the example design, including a description of files and the directory structure generated by the CORE Generator™ software, as well as a definition of the purpose and content of the provided scripts and example HDL wrappers. In addition, information about the operation of the demonstration test bench is also included.

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
`Courier font`	Messages, prompts, and program files that the system displays	`speed grade: - 100`
`Courier bold`	Literal commands you enter in a syntactical statement	**`ngdbuild`** `design_name`
Italic font	References to other manuals	See the *User Guide* for details.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.
Dark Shading	Items that are not supported or reserved	This feature is not supported
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required.	**`ngdbuild`** `[option_name] design_name`
Braces { }	A list of items from which you must choose one or more	**`lowpwr ={on|off}`**
Vertical bar |	Separates items in a list of choices	**`lowpwr ={on|off}`**

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

Horizontal ellipsis ...	Omitted repetitive material	**`allow block`** `block_name loc1 loc2 ... locn;`
Notations	The prefix '`0x`' or the suffix '`h`' indicate hexadecimal notation	A read of address `0x00112975` returned `45524943`h.
	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 "Additional Resources" for details. See "Title Formats" in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See **Figure 2-5** in the *Virtex-4 FPGA Handbook.*
Blue, underlined text	Hyperlink to a website (URL)	Go to www.xilinx.com for the latest speed files.

Chapter 1

Introduction

The LogiCORE™ IP VLYNQ™ core is a fully verified, serial, low-pin-count communications interface for cost-sensitive applications. Developed by Texas Instruments, VLYNQ enables the extension of an internal bus segment to one or more external devices in a scalable manner.

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

About the Core

The VLYNQ core is a Xilinx® CORE Generator™ software IP core, included in the latest IP Update on the Xilinx IP Center. For detailed information about the core, see the product page. For information installation and licensing options, see Chapter 2, "Licensing the Core."

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 v11.1

Recommended Design Experience

Although the VLYNQ core is a fully verified solution, the challenge associated with implementing a complete VLYNQ design varies depending on the application requirements. 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 assistance in evaluating your specific requirements.

Additional Core Resources

For detailed information and updates related to the VLYNQ core, see the following documents, located on the product page:

- VLYNQ Data Sheet
- VLYNQ Release Notes

Technical Support

For technical support, visit support.xilinx.com. Questions are routed to a team of engineers with expertise using the VLYNQ interface.

Xilinx will provide technical support for use of this product as described in the *VLYNQ 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 VLYNQ core and its related documentation.

VLYNQ Core

For comments or suggestions about the core, please submit a WebCase from support.xilinx.com. 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 support.xilinx.com. 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 installing and obtaining a license for the VLYNQ™ core, which you must do before using it in your designs. The VLYNQ core is provided under the terms of the Xilinx LogiCORE™ Site License Agreement, which conforms to the terms of the SignOnce IP License standard defined by the Common License Consortium. Purchase of the core entitles you to technical support and access to updates for a period of one year.

Before you Begin

This chapter assumes you have installed the core using either the CORE Generator™ IP Software Update installer or by performing a manual installation after downloading the core from the web. For information about installing the core, see the VLYNQ product page.

License Options

The VLYNQ 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 VLYNQ 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 functional simulation of the VLYNQ core using the example design and 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 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: VLYNQ product page
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 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.

XILINX®

Chapter 3

Quick Start Example Design

The instructions in this chapter show you how to quickly generate a VLYNQ™ core, run the design through implementation using 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 VLYNQ example design consists of the following:

- A demonstration test bench, to provide stimulus to the local VLYNQ core
- Two instances of a fully compliant example VLYNQ design, implementing a local and remote core
- An example remote OPB slave device

The example design performs the following functions. It starts by polling the linked status register, 0x08. Once a link has been established between the local and remote VLYNQ nodes, the design performs a series of register writes to the local and remote cores in order to set up the address map. It then performs writes and read backs on four different regions of the memory map of the remote core. Each read data is compared to make sure that it is correct, and a message is written to the console. The test bench then sets up the 'Interrupt to configuration register' bit in the Control register (0x04) of the local and remote nodes, enabling assertion of the interrupt line on reception of packets from the remote device. It then checks if the interrupt has been asserted correctly and if correct, a "TESTS PASS" message is written to the console and the test bench exits. If any of the above checks result in an error, the test bench exits with a "TESTS FAIL" message on the console.

The VLYNQ example design has been tested with Xilinx ISE® software 11.1, ModelSim 6.4b, and Cadence IUS v8.1 -s009.

Figure 3-1 illustrates the VLYNQ example design.

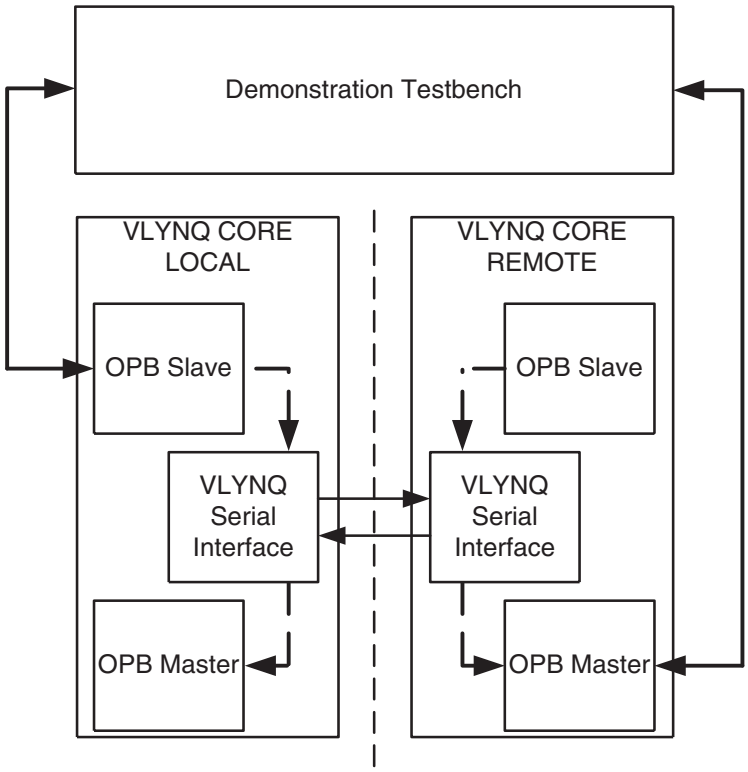

Figure 3-1: **Example Design**

Generating the Core

To generate a VLYNQ core using the CORE Generator™ software:

1. Start the CORE Generator software.

   For help starting and using the CORE Generator tool, see the *Xilinx CORE Generator Guide*, available from the ISE software documentation web page.
2. Choose File > New Project.
3. Type a directory name. In this example, the directory name *design* is used.
4. Set the following project options:
   - Part Options
     - From Target Architecture, select the desired family. For a list of supported families, see the *VLYNQ Data Sheet*.

     Note: If an unsupported silicon family is selected, the VLYNQ core does not appear in the taxonomy tree.
   - Generation Options
     - For Design Entry, select either VHDL or Verilog.
     - For Vendor, select Synplicity or Other (for XST).
5. After creating the project, locate the VLYNQ core in the taxonomy tree under Communications and Networking > Telecommunications > VLYNQ.

6. Double-click the core to display the main VLYNQ screen.

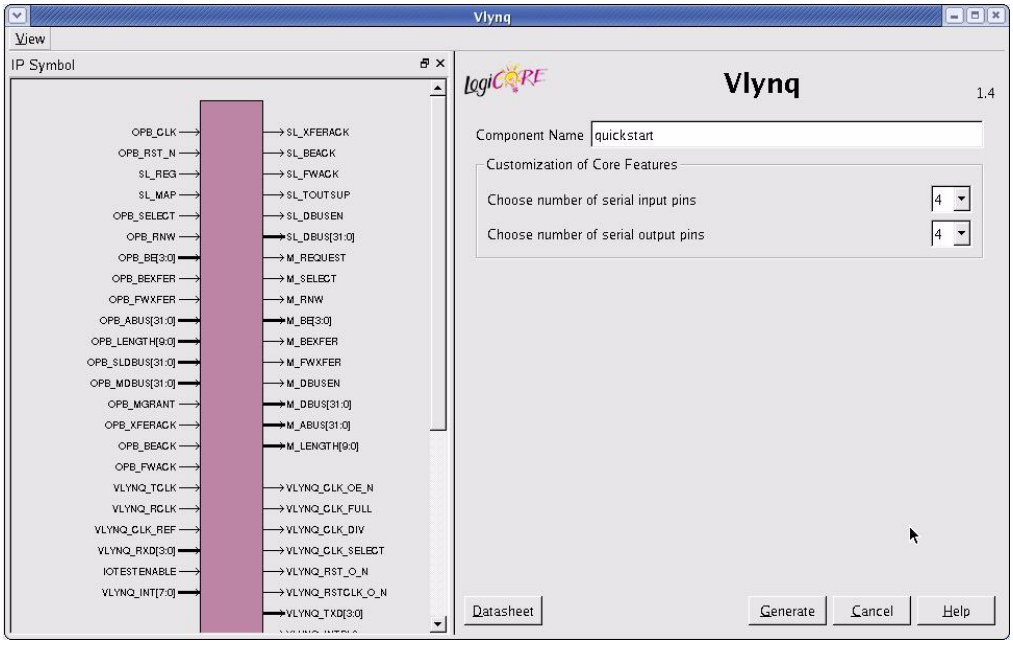

Figure 3-2: **VLYNQ Main Screen**

7. In the Component Name field, enter a name for the core instance. For the example design, the name *quickstart* is used.
8. After selecting the desired parameters from the VLYNQ screen, click Generate.

   The test bench and example design for this core have been designed to support configurability. For this reason, based on the generics selected by you, the design is automatically modified to support a different number of TX - RX lines. You need not modify the test bench files manually to obtain a configurable test bench. When the number of serial inputs and outputs are not equal, unused upper bits of VLYNQ_RXD are tied off to ground, while the unused upper bits of VLYNQ_TXD are left unconnected.

   The core and its supporting files, including the example design, are generated in the project directory. For detailed information about the example design files and directories see "Directory Structure," on page 19.

Implementing the Example Design

After generating a core with a Full System Hardware Evaluation or Full license, the netlists and the example design can be processed with the Xilinx implementation tools. The generated output files include scripts to assist you in running the Xilinx software.

To implement the VLYNQ example design, open a command prompt or terminal window and type the following commands:

Windows

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

Linux

```
% cd <project name>/<quickstart>/implement
% ./implement.sh
```

These commands execute a script that synthesizes, builds, maps, and place-and-routes the example design. The script then generates a post-par simulation model for use in timing simulation, and the resulting files are placed in the results directory.

Simulating the Example Design

The VLYNQ core provides a quick way to simulate and observe the behavior of the core utilizing the provided example design. There are two types of simulation: functional and timing. The provided simulation models will be in either VHDL or Verilog depending on the CORE Generator Design Entry project option selected. The Custom Output Products option is not supported for this core.

Setting up for Simulation

The Xilinx UNISIM and SIMPRIM libraries must be mapped to the simulator. If the UNISIM or SIMPRIM libraries are not set for your environment, see Answer Record 15338 on www.xilinx.com/support for assistance compiling Xilinx simulation models. Simulation scripts are provided for ModelSim and IUS.

Functional Simulation

Instructions for running a functional simulation of the VLYNQ core using either VHDL or Verilog are provided in the following procedure. Functional simulation models are provided when the core is generated. Implementing the core before simulating the functional models is not required.

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

1. Set the current directory to:

   ***<quickstart>*/simulation/functional/**

2. Launch the simulation script:

   ```
   ModelSim: vsim -do simulate_mti.do
   ncsim (>): simulate_ncsim.bat
   ncsim (%): ./simulate_ncsim.sh
   ```

The simulation script compiles the functional simulation models and the demonstration test bench, adds relevant signals to the wave window, and runs the simulation. To observe the operation of the core, inspect the simulation transcript and the waveform.

Timing Simulation

Timing simulation is only supported for Full System Hardware Evaluation and Full license types. Instructions for running a timing simulation of the VLYNQ core using either VHDL or Verilog are provided. A timing simulation model is generated when the core is run through the Xilinx tools using the implement script. The core must be implemented before attempting to run timing simulation.

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

1. Set the current directory to:

   ***<quickstart>*/simulation/timing/**

2. Launch the simulation script:

   ```
   ModelSim: vsim -do simulate_mti.do
   ncsim (>): simulate_ncsim.bat
   ncsim (%): ./simulate_ncsim.sh
   ```

The simulation script compiles the timing simulation model and the demonstration test bench, adds relevant signals to the wave window, and runs the simulation. To observe the operation of the core, inspect the simulation transcript and the waveform.

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 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

- **<project directory>**
  Top-level project directory; user-defined name
  - <project directory>/<component name>
    Core readme release notes file
    - <component name>/doc
      Product documentation
    - <component name>/example design
      Verilog and VHDL (or whichever, if it is only one) design files
    - <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
        Simulation files

Directory and File Contents

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

<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	Top-level netlist
<component_name>.v[hd]	Verilog or VHDL simulation model
<component_name>.{veo|vho}	VHDL or Verilog instantiation template
<component_name>.xco	CORE Generator software project-specific option file; can be used as an input to the CORE Generator software.
<component_name>.xcp	List of files delivered with the core
<component_name>_flist.txt	List of files delivered with the core

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>	
vlynq_readme.txt	Core name release notes file

Back to Top

<component name>/example design

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

Table 4-3: **Example Design Directory**

Name	Description
<project_dir>/<component_name>/example_design	
<component_name>_top.ucf	The UCF for the core provides example constraints necessary for processing the VLYNQ core using the Xilinx® implementation tools. You can modify the UCF to meet individual system requirements. The sample UCF contains timing and placement constraints for the core. The sample UCF is only generated for Full System Hardware Evaluation and Full license types. See "License Options," page 11 for more information.
<component_name>_top.v[hd] <component_name>.v	Verilog and VHDL example design files. The <component_name>.v file is only generated when the Verilog design flow is chosen.

Back to Top

<component name>/doc

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

Table 4-4: **Doc Directory**

Name	Description
<project_dir>/<component_name>/doc	
vlynq_ds324.pdf	*VLYNQ Data Sheet*
vlynq_gsug171.pdf	*VLYNQ Getting Started Guide*

Back to Top

<component name>/implement

The implement directory contains the core implementation script files, and is only generated for the Full System Hardware Evaluation and Full license types. See "License Options," page 11 for information about license options.

Table 4-5: **Implement Directory**

Name	Description
<project_dir>/<component_name>/implement	
implement.{sh | bat}	A Windows (.bat) or Linux (.sh) script that processes the example design through the Xilinx tool flow. For more information, see "Implementation and Simulation Scripts."
xst.prj	The XST script file for the example design used to synthesize the core and is called from the implement script described previously. Only available when the CORE Generator software Vendor project option is set to "Other."
xst.scr	The XST script file for the example design used to synthesize the core and is called from the implement script described previously. Only available when the CORE Generator software Vendor project option is set to "Other."
synplify.prj	The Synplicity project file for the example design. Lists all of the source files to be synthesized, and is available only when CORE Generator software Vendor project option is set to Synplicity.

Back to Top

implement/results

The results directory is created by the implement script, after which the implement script results are placed in the results directory.

Table 4-6: **Results Directory**

Name	Description
<project_dir>/<component_name>/implement/results	
Implement script result files	

Back to Top

<component name>/simulation

The simulation directory contains the necessary files to test a VHDL or Verilog example design with the demonstration test bench.

Table 4-7: **Simulation Directory**

Name	Description
<project_dir>/<component_name>/simulation	
glbl.v	Contains the necessary files to test a VHDL or Verilog example design with the demonstration test bench.
vlynq_clk_gen.v[hd]	
vlynq_driver.v[hd]	
vlynq_monitor.v[hd]	
vlynq_procedures.v[hd]	
vlynq_slave.v[hd]	
vlynq_tb.v[hd]	
vlynq_testcase_pkg.v[hd]	

Back to Top

simulation/functional

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

Table 4-8: **Functional Directory**

Name	Description
<project_dir>/<component_name>/simulation/functional	
simulate_mti.do	A macro file (Modelsim and IUS, respectively) that compiles the functional netlist and demo HDL source. The script also loads and runs the simulation for 400 µs.
simulate_ncsim.sh	
simulate_ncsim.bat	
wave.do	A macro file (Modelsim and IUS, respectively) that opens a wave window and adds key signals to the wave viewer. The respective wave file is called by the corresponding simulate file, and is displayed after the simulation is loaded.
wave.sv	

Back to Top

simulation/timing

The timing directory contains timing simulation scripts provided with the core and is only provided with the Full System Hardware or Full licenses.

Table 4-9: **Timing Directory**

Name	Description
<project_dir>/<component_name>/simulation/timing	
simulate_mti.do	A macro file (Modelsim and IUS, respectively) that compiles the post-par timing netlist and demo HDL source. The script also loads and runs the simulation for 400 ms. The implement script must be run to generate the post-par timing simulation model. Simulation can only be run after the timing simulation model is generated.
simulate_ncsim.sh	
simulate_ncsim.bat	
wave.do	A macro file (Modelsim and IUS, respectively) that opens a wave window and adds key signals to the wave viewer. The respective wave file is called by the corresponding simulate file and is displayed after the simulation is loaded.
wave.sv	

Back to Top

Implementation and Simulation Scripts

Implementation Script

The implementation script is either a shell script or batch file that runs the example design through the Xilinx tool flow.

***<project_dir>/<component_name>*/implement/**

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

- Synthesizes the example design using the selected synthesis tool (XST or Synplify)
- Runs **ngdbuild** to consolidate the core netlists, wrapper netlist, and constraints file into the common database
- Runs **map** to perform technology specific mapping of the design
- Runs **par** to perform place and route of the design
- Runs **trce** to perform static timing analysis of the routed design
- Runs **bitgen** to generate a bitstream for download to the target FPGA
- Runs **netgen** to generate a post-par simulation model for use in timing simulation

Simulation Script

The demonstration test bench generates and receives packet data across the VLYNQ interface. Simulation scripts are provided for ModelSim and IUS. The provided simulation scripts compile the simulation model and demonstration test bench, and run the simulation. The simulation files are located in one of the following directories:

`<project_dir>/<component_name>/simulation/{functional | timing}/`

For functional simulation, the simulation script performs the following tasks:

- Compiles the wrapper file, which instantiates the cores
- Compiles the demonstration test bench
- Starts a simulation of the demonstration test bench
- Opens the waveform viewer and adds key signals(wave.do or wave.sv)
- Runs the simulation

For timing simulation, the simulation script performs the following tasks:

- Compiles the post-par design example, which includes the cores
- Compiles the demonstration test bench
- Starts a simulation of the demonstration test bench
- Opens the waveform viewer and adds key signals (wave.do or wave.sv)
- Runs the simulation

Example Design Configuration

Figure 4-1 illustrates the example design configuration.

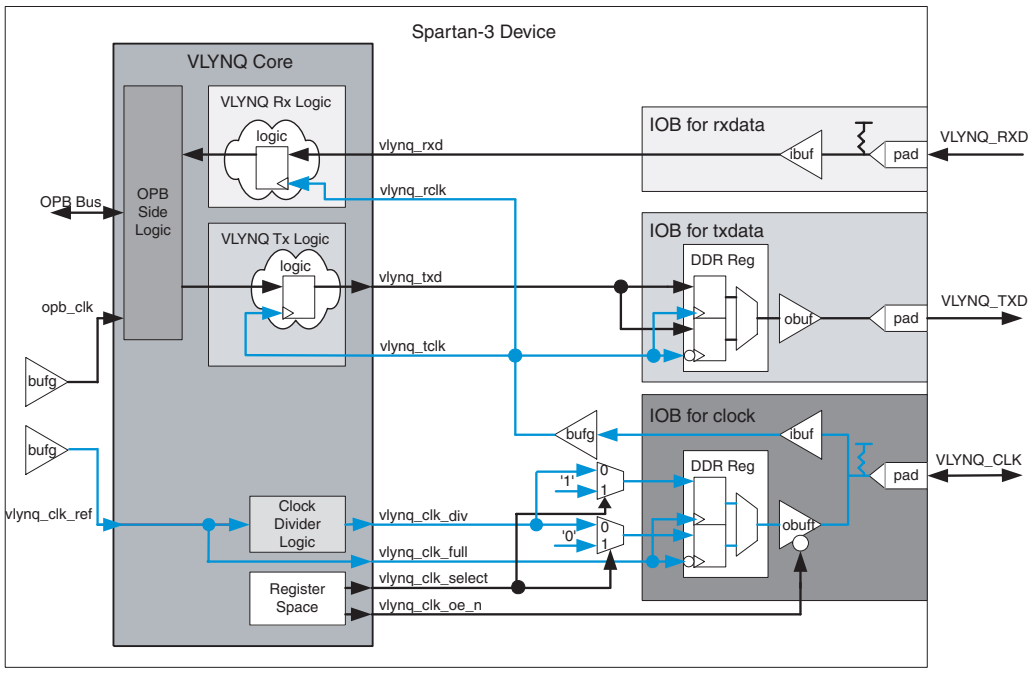

Figure 4-1: **Example Design Configuration**

Example design configuration:

- With the default generic options, the example design instantiates a VLYNQ core, five double-data rate (DDR) registers and appropriate logic to implement a fully compliant VLYNQ design.
- In general, one DDR register is instantiated to align the clock and the remaining DDR registers, which align data, are instantiated based upon the number of serial output lines.
- The VLYNQ core is instantiated as a black-box module, replaced with CORE Generator software netlists during implementation, and then replaced with the gate-level simulation model during simulation.

Demonstration Test Bench

The VLYNQ demonstration test bench is comprised of six modules, including a test case package. Figure 4-2 illustrates how these test bench modules are interfaced with the local and remote VLYNQ modules.

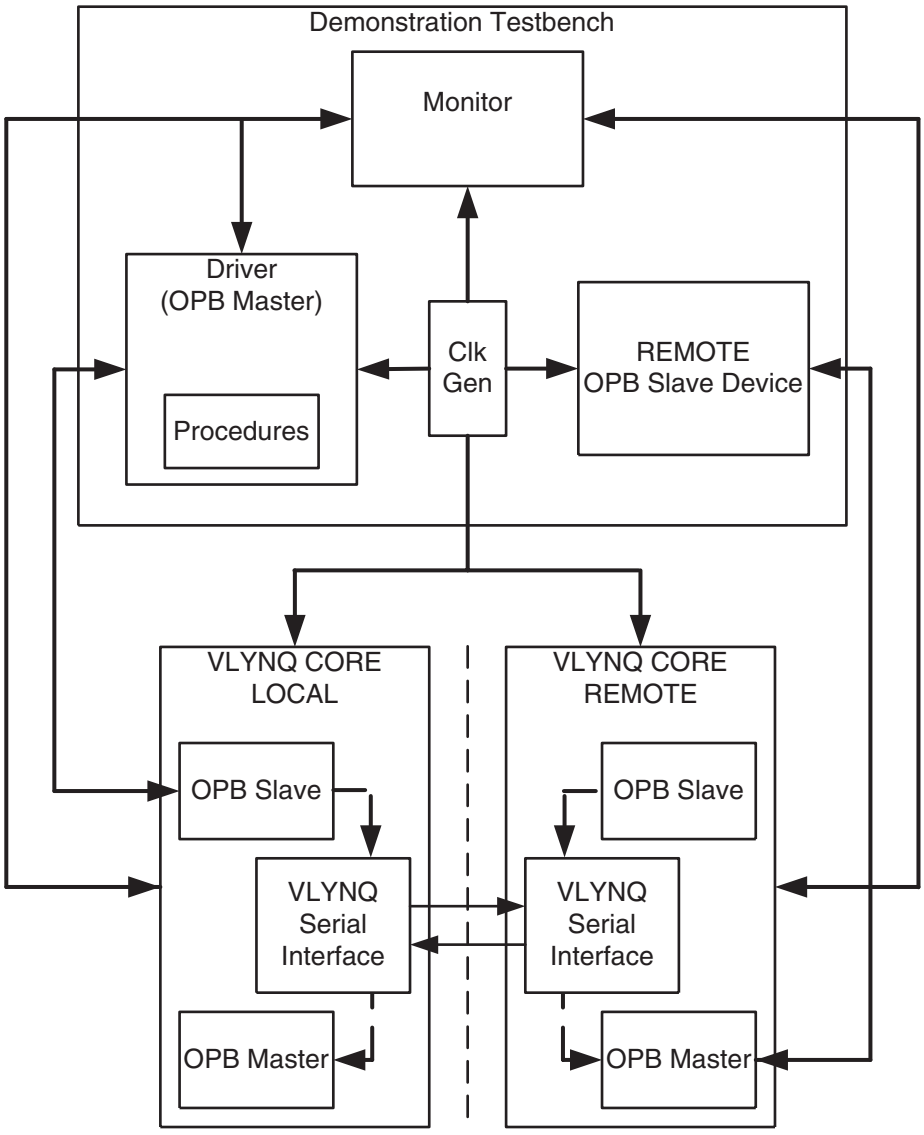

Figure 4-2: **Demonstration Test Bench**

Clock Generator Module

The clock generator module (vlynq_clk_gen.v[hd]) creates all clocks required for the demonstration test bench. The frequency of each core clock is defined in the testcase package that follows. By default, the example design connects the `vlynq_ref_clk` and `opb_clk` to the clock generator module.

Procedure Module

The procedure module (vlynq_procedure.v[hd]) contains all functions and procedures used to create the bus functional model of the OPB interface.

Testcase Package

The testcase package (vlynq_testcase_pkg.v[hd]) contains all global parameters (constants) used in the example design.

Driver Module

The driver module (vlynq_driver.v[hd]) is the customer interface for passing data to/from the VLYNQ core by transmitting data using an example OPB Master interface. Data is transmitted to the VLYNQ cores OPB Slave interface. Data is transferred from the driver to the VLYNQ core using a variety of procedure calls, which let you customize the traffic patterns being sent and observe the cores responses.

Monitor Module

The monitor module (vlynq_monitor.v[hd]) inspects the transactions on the OPB bus for validity and reports the transactions to you. Basic checking is done to ensure that the VLYNQ core is responding correctly.

Slave Module

The slave module (vlynq_slave.v[hd]) is responsible for supplying data to the remote VLYNQ core when requested. The slave module implements an example OPB slave interface and implements a basic memory structure to be queried by the remote VLYNQ core.