

# HDL Verifier™

## Getting Started Guide

**R2012a**

**MATLAB®**  
& **SIMULINK®**

## How to Contact MathWorks



[www.mathworks.com](http://www.mathworks.com) Web  
[comp.soft-sys.matlab](mailto:comp.soft-sys.matlab) Newsgroup  
[www.mathworks.com/contact\\_TS.html](http://www.mathworks.com/contact_TS.html) Technical Support



[suggest@mathworks.com](mailto:suggest@mathworks.com) Product enhancement suggestions  
[bugs@mathworks.com](mailto:bugs@mathworks.com) Bug reports  
[doc@mathworks.com](mailto:doc@mathworks.com) Documentation error reports  
[service@mathworks.com](mailto:service@mathworks.com) Order status, license renewals, passcodes  
[info@mathworks.com](mailto:info@mathworks.com) Sales, pricing, and general information



508-647-7000 (Phone)



508-647-7001 (Fax)



The MathWorks, Inc.  
3 Apple Hill Drive  
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

*HDL Verifier™ Getting Started Guide*

© COPYRIGHT 2003–2012 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

### Trademarks

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See [www.mathworks.com/trademarks](http://www.mathworks.com/trademarks) for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

### Patents

MathWorks products are protected by one or more U.S. patents. Please see [www.mathworks.com/patents](http://www.mathworks.com/patents) for more information.

## Revision History

August 2003	Online only	New for Version 1 (Release 13SP1)
February 2004	Online only	Updated for Version 1.1 (Release 13SP1)
June 2004	Online only	Updated for Version 1.1.1 (Release 14)
October 2004	Online only	Updated for Version 1.2 (Release 14SP1)
December 2004	Online only	Updated for Version 1.3 (Release 14SP1+)
March 2005	Online only	Updated for Version 1.3.1 (Release 14SP2)
September 2005	Online only	Updated for Version 1.4 (Release 14SP3)
March 2006	Online only	Updated for Version 2.0 (Release 2006a)
September 2006	Online only	Updated for Version 2.1 (Release 2006b)
March 2007	Online only	Updated for Version 2.2 (Release 2007a)
September 2007	Online only	Updated for Version 2.3 (Release 2007b)
March 2008	Online only	Updated for Version 2.4 (Release 2008a)
October 2008	Online only	Updated for Version 2.5 (Release 2008b)
March 2009	Online only	Updated for Version 2.6 (Release 2009a)
September 2009	Online only	Updated for Version 3.0 (Release 2009b)
March 2010	Online only	Updated for Version 3.1 (Release 2010a)
September 2010	Online only	Updated for Version 3.2 (Release 2010b)
April 2011	Online only	Updated for Version 3.3 (Release 2011a)
September 2011	Online only	Updated for Version 3.4 (Release 2011b)
March 2012	Online only	Updated for Version 4.0 (Release 2012a)



## Introduction

### 1

<b>Product Description</b> .....	1-2
Key Features .....	1-2
<b>HDL Cosimulation</b> .....	1-3
HDL Cosimulation with MATLAB or Simulink and the HDL Simulator .....	1-3
Communications for HDL Cosimulation .....	1-8
Hardware Description Language (HDL) Support .....	1-8
HDL Cosimulation Workflows Described in the User Guide .....	1-9
<b>FPGA Development</b> .....	1-10
FPGA Development with HDL Verifier .....	1-10
FPGA-in-the-Loop Simulation .....	1-10
FPGA Automation with Filter Design HDL Coder .....	1-11
<b>TLM Component Generation</b> .....	1-12
Generating TLM Components for Use with Virtual Platform Development .....	1-12
Typical Users and Applications .....	1-13

## Installation

### 2

<b>Installing the HDL Verifier Software</b> .....	2-2
<b>Installing Related Application Software</b> .....	2-3

## Product Requirements

---

### 3

<b>What You Need to Know</b> .....	3-2
For Cosimulating with HDL Simulators .....	3-2
For FPGA-in-the-Loop Simulation .....	3-2
For FPGA Automation .....	3-3
For Generating OSCI-Compatible TLM Components .....	3-3
Additional Useful Experience .....	3-3
Product Limitations .....	3-3
<b>Required Products</b> .....	3-4
Supported EDA Tools .....	3-4
System Requirements .....	3-7
Product Feature and Platform Support .....	3-7
Optional Application Software .....	3-9

## Getting Help

---

### 4

<b>Information Overview</b> .....	4-2
<b>Online Help</b> .....	4-3
<b>Using “What’s This?” Context-Sensitive Help</b> .....	4-5
<b>Demos and Tutorials</b> .....	4-7
Demos .....	4-7
Tutorials .....	4-7

<b>Generate HDL Cosimulation Interfaces from Existing</b>	
<b>HDL Code</b> .....	<b>5-2</b>
Create a MATLAB Function From Existing HDL Code ...	<b>5-2</b>
Create a MATLAB System Object From Existing HDL	
Code .....	<b>5-9</b>
Create an HDL Cosimulation Block From Existing HDL	
Code .....	<b>5-10</b>
Perform Cosimulation .....	<b>5-21</b>
<b>Import HDL Code For FPGA-in-the-Loop</b>	
<b>Verification</b> .....	<b>5-24</b>
Preparing to Use the FPGA-in-the-Loop (FIL) Wizard ....	<b>5-24</b>
Running the FIL Wizard .....	<b>5-24</b>
Performing FIL Simulation .....	<b>5-29</b>





# Introduction

---

- “Product Description” on page 1-2
- “HDL Cosimulation” on page 1-3
- “FPGA Development” on page 1-10
- “TLM Component Generation” on page 1-12

## Product Description

### **Verify VHDL® and Verilog® using HDL simulators and FPGA-in-the-loop test benches**

HDL Verifier™ automates Verilog and VHDL design verification using HDL simulators and FPGA hardware-in-the-loop. It provides interfaces that link MATLAB® and Simulink® with Cadence Incisive®, Mentor Graphics® ModelSim®, and Mentor Graphics Questa® HDL simulators. It also supports FPGA-in-the-loop verification with Xilinx® and Altera® FPGA boards.

HDL Verifier automates verification by using MATLAB or Simulink to stimulate your HDL code and analyze its response. This approach eliminates the need to author standalone Verilog or VHDL test benches.

### **Key Features**

- Cosimulation support for Cadence Incisive and for Mentor Graphics ModelSim and Questa
- FPGA-in-the-loop verification using Xilinx and Altera FPGA boards
- MATLAB functions and Simulink blocks
- Generation of IEEE 1666 SystemC TLM 2.0 compatible transaction-level models
- Interactive or batch-mode cosimulation and debugging
- Single-machine, multiple-machine, and cross-network cosimulation

## HDL Cosimulation

**In this section...**

“HDL Cosimulation with MATLAB or Simulink and the HDL Simulator” on page 1-3

“Communications for HDL Cosimulation” on page 1-8

“Hardware Description Language (HDL) Support” on page 1-8

“HDL Cosimulation Workflows Described in the User Guide” on page 1-9

### HDL Cosimulation with MATLAB or Simulink and the HDL Simulator

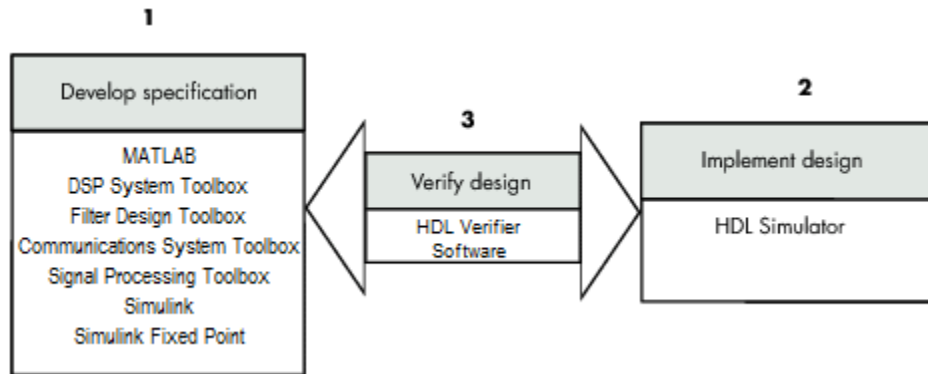
HDL Verifier functions as a cosimulation interface that provides a bidirectional link between MATLAB and Simulink and HDL simulators from Mentor Graphics and Cadence®, enabling verification of VHDL, Verilog, and mixed-language implementations. This software enables interactive and batch-mode cosimulation on a single computer, across heterogeneous platforms, or across a network.

The HDL Verifier software consists of MATLAB functions, a MATLAB System object, and a library of Simulink blocks, all of which establish communication links between the HDL simulator and MATLAB or Simulink.

HDL Verifier software streamlines FPGA and ASIC development by integrating tools available for these processes:

- 1** Developing specifications for hardware design reference models
- 2** Implementing a hardware design in HDL based on a reference model
- 3** Verifying the design against the reference design

The following figure shows how the HDL simulator and MathWorks® products fit into this hardware design scenario.



As the figure shows, HDL Verifier software connects tools that traditionally have been used discretely to perform specific steps in the design process. By connecting these tools, the link simplifies verification by allowing you to cosimulate the implementation and original specification directly. This cosimulation results in significant time savings and the elimination of errors inherent to manual comparison and inspection.

In addition to the preceding design scenario, HDL Verifier software enables you to work with tools in the following ways:

- Use MATLAB or Simulink to create test signals and software test benches for HDL code
- Use MATLAB or Simulink to provide a behavioral model for an HDL simulation
- Use MATLAB analysis and visualization capabilities for real-time insight into an HDL implementation
- Use Simulink to translate legacy HDL descriptions into system-level views

---

**Note** You can cosimulate a module using SystemVerilog, SystemC or both with MATLAB or Simulink using the HDL Verifier software. Write simple wrappers around the SystemC and make sure that the SystemVerilog cosimulation connections are to ports or signals of data types supported by the link cosimulation interface.

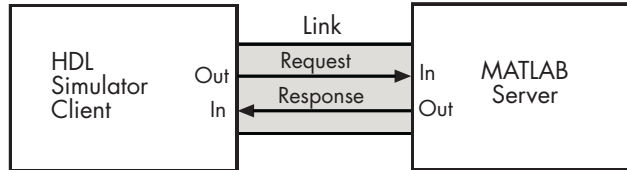
---

More discussion on how cosimulation works can be found in the following sections:

- “Linking with MATLAB and the HDL Simulator” on page 1-5
- “Linking with Simulink and the HDL Simulator” on page 1-6
- “The HDL Cosimulation Wizard” on page 1-8

### Linking with MATLAB and the HDL Simulator

When linked with MATLAB, the HDL simulator functions as the client, as the following figure shows.

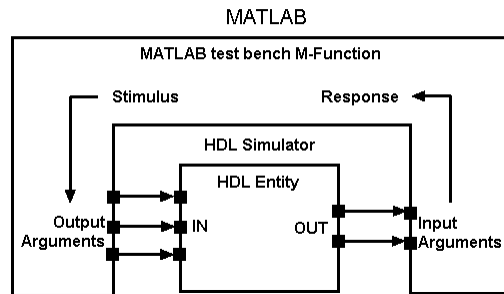


In this scenario, a MATLAB server function waits for service requests that it receives from an HDL simulator session. After receiving a request, the server establishes a communication link and invokes a specified MATLAB function that computes data for, verifies, or visualizes the HDL module (coded in VHDL or Verilog) that is under simulation in the HDL simulator.

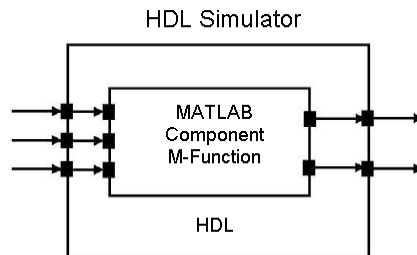
After the server is running, you can start and configure the HDL simulator or use with MATLAB with the supplied HDL Verifier function:

- `nclaunch` (Incisive)
- `vsim` (ModelSim)

The following figure shows how a MATLAB test bench function wraps around and communicates with the HDL simulator during a test bench simulation session.



The following figure shows how a MATLAB component function is wrapped around by and communicates with the HDL simulator during a component simulation session.

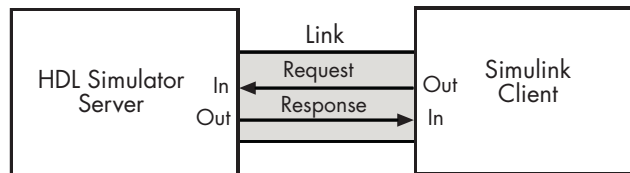


When you begin a specific test bench or component session, you specify parameters that identify the following information:

- The mode and, if applicable, TCP/IP data for connecting to a MATLAB server
- The MATLAB function that is associated with and executes on behalf of the HDL instance
- Timing specifications and other control data that specifies when the module's MATLAB function is to be called

### Linking with Simulink and the HDL Simulator

When linked with Simulink, the HDL simulator functions as the server, as shown in the following figure.



In this case, the HDL simulator responds to simulation requests it receives from cosimulation blocks in a Simulink model. You begin a cosimulation session from Simulink. After a session is started, you can use Simulink and the HDL simulator to monitor simulation progress and results. For example, you might add signals to an HDL simulator Wave window to monitor simulation timing diagrams.

Using the Block Parameters dialog box for an HDL Cosimulation block, you can configure the following:

- Block input and output ports that correspond to signals (including internal signals) of an HDL module. You can specify sample times and fixed-point data types for individual block output ports if desired.
- Type of communication and communication settings used for exchanging data between the simulation tools.
- Rising-edge or falling-edge clocks to apply to your module. You can individually specify the period of each clock.
- Tcl commands to run before and after the simulation.

HDL Verifier software equips the HDL simulator with a set of customized functions. For ModelSim, when you use the function `vsimulink`, you execute the HDL simulator with an instance of an HDL module for cosimulation with Simulink. After the module is loaded, you can start the cosimulation session from Simulink. Incisive users can perform the same operations with the function `hdlsimulink`.

HDL Verifier software also includes a block for generating value change dump (VCD) files. You can use VCD files generated with this block to perform the following tasks:

- View Simulink simulation waveforms in your HDL simulation environment

- Compare results of multiple simulation runs, using the same or different simulation environments
- Use as input to post-simulation analysis tools

## **The HDL Cosimulation Wizard**

HDL Verifier contains the Cosimulation Wizard feature, which uses existing HDL code to create a customized MATLAB function (test bench or component), MATLAB System object, or Simulink HDL Cosimulation block. For more information, see “HDL Cosimulation Wizard”.

## **Communications for HDL Cosimulation**

The mode of communication that you use for a link between the HDL simulator and MATLAB or Simulink depends on whether your application runs in a local, single-system configuration or in a network configuration. If these products and MathWorks products can run locally on the same system and your application requires only one communication channel, you have the option of choosing between shared memory and TCP/IP socket communication. Shared memory communication provides optimal performance and is the default mode of communication.

TCP/IP socket mode is more versatile. You can use it for single-system and network configurations. This option offers the greatest scalability. For more on TCP/IP socket communication, see “Choosing TCP/IP Socket Ports”.

## **Hardware Description Language (HDL) Support**

All HDL Verifier MATLAB functions and the HDL Cosimulation block offer the same language-transparent feature set for both Verilog and VHDL models.

HDL Verifier software also supports mixed-language HDL models (models with both Verilog and VHDL components), allowing you to cosimulate VHDL and Verilog signals simultaneously. Both MATLAB and Simulink software can access components in different languages at any level.



## **HDL Cosimulation Workflows Described in the User Guide**

The HDL Verifier User Guide provides instruction for using the verification software with supported HDL simulators for the following workflows:

- Simulating an HDL Component in a MATLAB Test Bench Environment
- Replacing an HDL Component with a MATLAB Component Function
- Simulating an HDL Component in a Simulink Test Bench Environment
- Replacing an HDL Component with a Simulink Algorithm
- Recording Simulink Signal State Transitions for Post-Processing

## FPGA Development

In this section...
“FPGA Development with HDL Verifier” on page 1-10
“FPGA-in-the-Loop Simulation” on page 1-10
“FPGA Automation with Filter Design HDL Coder” on page 1-11

### FPGA Development with HDL Verifier

HDL Verifier works with Simulink and HDL Coder™ or MATLAB and Filter Design HDL Coder™ and the supported FPGA development environment to prepare your automatically generated HDL Code for implementation in an FPGA. HDL Verifier creates and manages your Xilinx ISE project and integrates a clock module with your design in an automatically generated top level module.

### The FIL Wizard

HDL Verifier contains the FIL Wizard feature, which uses existing HDL code to create a customized FPGA implementation. For more information, see “Generating a FIL Block Using the FIL Wizard”.

### FPGA-in-the-Loop Simulation

FPGA-in-the-Loop simulation allows you to run a Simulink simulation with an FPGA board strictly synchronized with Simulink. This lets you get real world data into your design while accelerating your simulation with the speed of an FPGA.

You can generate a FIL programming file in one of the following ways:

- Use the HDL Verifier FIL Wizard.
- Use the HDL Coder Workflow Advisor (see HDL Coder for instruction).

The FIL Wizard uses any synthesizable HDL code including code automatically generated from Simulink models by HDL Coder software. When you use FIL in the Workflow Advisor, HDL Coder uses the loaded design to create the HDL code. Either way, this HDL code is then augmented

by customized code for FIL communication with your design and assembled into an FPGA project. The applicable downstream tools are used to process that project to create a programming file that is automatically downloaded to the FPGA device on a development board for verification. See HDL Verifier product page for a list of currently supported devices and boards.

HDL Verifier supports the use of a FIL block in a model reference block.

## **FPGA Automation with Filter Design HDL Coder**

Create, update, and manage FPGA projects with Xilinx ISE. HDL Verifier provides a Project Generator for generating HDL code from filter code using Filter Design HDL Coder.

HDL Verifier packages these files as a complete FPGA development environment project for use with Xilinx ISE.

With the interface, you can do the following:

- Take code generation process one step further and package up the generated code so you can use it with Xilinx tools (most of project info provided by HDL Verifier for project creation).
- Make changes to project info: automatically update generated code, add Simulink files to existing project, automatically manage generated files in associated project.
- Get settings from existing project and save these settings with the model.

## **FPGA Automation Workflows Described in the User Guide**

The HDL Verifier User Guide provides instruction for using the verification software with supported FPGA development environments for the following workflows:

- Creating a new FPGA project
- Adding generated files to an existing FPGA project
- Generating Tcl scripts for project generation

See “FPGA Automation with Filter Design HDL Coder”.

## TLM Component Generation

In this section...
“Generating TLM Components for Use with Virtual Platform Development” on page 1-12
“Typical Users and Applications” on page 1-13

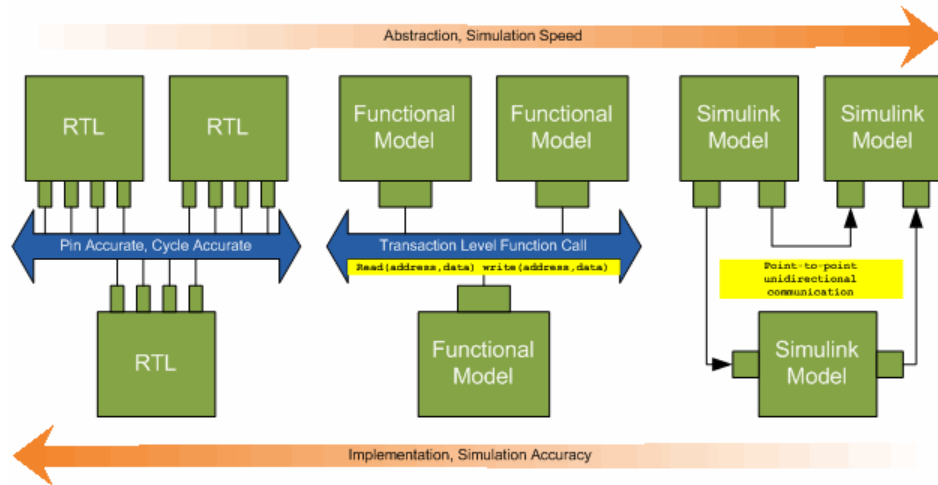
### **Generating TLM Components for Use with Virtual Platform Development**

HDL Verifier lets you create a SystemC Transaction Level Model (TLM) that can be executed in any OSCI-compatible TLM 2.0 environment, including a commercial virtual platform.

When used with virtual platforms, HDL Verifier joins two different modeling environments: Simulink for high-level algorithm development and virtual platforms for system architectural modeling. The Simulink modeling typically dispenses with implementation details of the hardware system such as processor and operating system, system initialization, memory subsystems, device configuration and control, and the particular hardware protocols for transferring data both internally and externally.

The virtual platform is a simulation environment that is concerned about the hardware details: it has components that map to hardware devices such as processors, memories, and peripherals, and a means to model the hardware interconnect between them.

Although many goals could be met with a virtual platform model, the ideal scenario for virtual platforms is to allow for software development—both high level application software and low-level device driver software—by having fairly abstract models for the hardware interconnect that allow the virtual platform to run at near real-time speeds, as demonstrated in the following diagram.



The functional model provides a sort of halfway point between the speed you can achieve with abstraction and the accuracy you get with implementation.

## Typical Users and Applications

Using HDL Verifier and Simulink, you can create a TLM-2.0-compliant SystemC Transaction Level Model (TLM) that can be executed in any OSCI-compatible TLM 2.0 environment, including a commercial virtual platform.

Typical users and applications include:

- System-level engineers designing electronic system models that include architectural characteristics
- Software developers who want to incorporate an algorithm into a virtual platform without using an instruction set simulator (ISS).
- Hardware functional verification engineers. In this case, the algorithm represents a piece of hardware going into a chip.



# Installation

---

- “Installing the HDL Verifier Software” on page 2-2
- “Installing Related Application Software” on page 2-3

## **Installing the HDL Verifier Software**

For details on how to install the HDL Verifier software, see the MATLAB installation instructions.



## Installing Related Application Software

Based on your configuration decisions and the software required for your HDL Verifier application, identify software you need to install and where you need to install it. For example, if you need to run multiple instances of the link MATLAB server on different machines, you need to install MATLAB and any applicable toolbox software on multiple systems. Each instance of MATLAB can run only one instance of the server.

For details on how to install the HDL simulator, see the installation instructions for that product. For information on installing and activating MathWorks products, see the MATLAB installation and activation instructions.



# Product Requirements

---

- “What You Need to Know” on page 3-2
- “Required Products” on page 3-4

## What You Need to Know

The documentation provided with the HDL Verifier software assumes users have a moderate level of prerequisite knowledge in the following subject areas.

In this section...
“For Cosimulating with HDL Simulators” on page 3-2
“For FPGA-in-the-Loop Simulation” on page 3-2
“For FPGA Automation” on page 3-3
“For Generating OSCI-Compatible TLM Components” on page 3-3
“Additional Useful Experience” on page 3-3
“Product Limitations” on page 3-3

### For Cosimulating with HDL Simulators

- Hardware design and system integration
- VHDL and/or Verilog
- Cadence Incisive or Mentor Graphics ModelSim simulators
- MATLAB
- Experience with Simulink and Simulink Fixed Point™ software is required for applying the Simulink component of the product
- Experience with Fixed-Point Toolbox™ software required for working with MATLAB System objects

### For FPGA-in-the-Loop Simulation

- FPGA design and implementation
- VHDL and/or Verilog
- Simulink and Simulink Fixed Point software

Some familiarity with Xilinx ISE or Altera Quartus II may be helpful (see supported versions in “Required Products” on page 3-4).

## For FPGA Automation

- FPGA design and implementation
- VHDL and/or Verilog
- Filter Design HDL Coder software

Some familiarity with Xilinx ISE may be helpful.

## For Generating OSCI-Compatible TLM Components

- Simulink
- Embedded Coder™ (some knowledge helpful)
- TLM 2.0
- System C 2.2 (compiling, linking, and executing)

## Additional Useful Experience

Depending on your application, experience with the following MATLAB toolboxes and Simulink blocksets might also be useful:

- Signal Processing Toolbox™
- Communications System Toolbox™
- DSP System Toolbox™
- Computer Vision System Toolbox™
- Simulink Fixed Point
- Embedded Coder

## Product Limitations

Compatibility with Simulink Code Generation:

- HDL Coder: The HDL Verifier HDL Cosimulation block does participate in code generation with HDL Coder.
- Simulink Coder™: The HDL Verifier HDL Cosimulation block does not participate in code generation with Simulink Coder for C code generation.

## Required Products

In this section...
“Supported EDA Tools” on page 3-4
“System Requirements” on page 3-7
“Product Feature and Platform Support” on page 3-7
“Optional Application Software” on page 3-9

### Supported EDA Tools

- “Cosimulation Requirements” on page 3-4
- “FPGA Verification Requirements” on page 3-5

### Cosimulation Requirements

- “Cadence Incisive Usage Requirements” on page 3-4
- “Mentor Graphics Questa and ModelSim Usage Requirements” on page 3-5

**Cadence Incisive Usage Requirements.** MATLAB and Simulink support Cadence verification tools using HDL Verifier.

Use one of these recommended versions, which have been fully tested against the current release:

- IES 10.2-s040
- IES 9.2-s014
- IUS 8.2-s009

The HDL Verifier shared libraries (`liblfihdls*.so`, `liblfihd1c*.so`) are built using the `gcc` included in the Cadence Incisive simulator platform distribution. Before you link your own applications into the HDL simulator, first try building against this `gcc`. See the HDL simulator documentation for more details about how to build and link your own applications.

**Mentor Graphics Questa and ModelSim Usage Requirements.**

MATLAB and Simulink support Mentor Graphics verification tools using HDL Verifier.

Use one of the following recommended versions. Each version has been fully tested against the current release:

- ModelSim SE 10.0c, 6.6d, 6.5f
- ModelSim PE 10.0c, 6.6d, 6.5f
- ModelSim DE 10.0c
- Questa 10.0a

The Linux® platform requires that HDL Verifier software run gcc c++ libraries (4.1 or later). You should install a recent version of the gcc c++ library on your computer. To determine which libraries are installed on your computer, type the command:

```
gcc -v
```

**FPGA Verification Requirements**

- “Xilinx ISE Usage Requirements” on page 3-5
- “Altera Quartus II Usage Requirements” on page 3-6
- “Supported FPGA Devices for FIL Simulation” on page 3-6
- “Supported FPGA Device Families for Clock Module Generation” on page 3-6

**Xilinx ISE Usage Requirements.** MATLAB and Simulink support Xilinx design tools using HDL Verifier.

- FPGA-in-the-Loop and FPGA Automation are tested with Xilinx ISE 13.1.
- ISE 11.1 or newer is recommended
- Additional requirements for clock module generation using FPGA Automation:

- 12.1 or later: Windows® only
- 11.4: Windows 32-bit only
- Consult Xilinx user documentation for compatibility of ISE tools with various Linux distributions.

**Altera Quartus II Usage Requirements.** MATLAB and Simulink support Altera design tools using HDL Verifier.

FPGA-in-the-Loop is tested with Altera Quartus II 11.0.

**Supported FPGA Devices for FIL Simulation.** HDL Verifier supports FIL simulation on the devices shown in the following table.

Device Family	Board
Spartan-6	Spartan-6 SP605 Spartan-6 SP601 XUP Atlys Spartan-6
Virtex-6	Virtex-6 ML605
Virtex-5	Virtex-5 ML505 Virtex-5 ML506 Virtex-5 ML507 Virtex-5 XUPV5–LX110T
Virtex-4	Virtex-4 ML401 Virtex-4 ML402 Virtex-4 ML403
Altera	Arria II GX FPGA development kit Cyclone III FPGA development kit Cyclone IV GX FPGA development kit DE2-115 development and education board

**Supported FPGA Device Families for Clock Module Generation.** For project generation with Filter Design HDL Coder, see Xilinx documentation for a full list of supported FPGA families in ISE.



With the current release, clock module generation is supported for the following device families:

- Spartan-3
- Spartan-3A and Spartan-3AN
- Spartan-3A DSP
- Spartan-3E
- Spartan-6
- Virtex-4
- Virtex-5

## System Requirements

Visit the HDL Verifier Requirements page for general system requirements and product version availability.

## Product Feature and Platform Support

Product Feature	Required Products	Recommended Products	Supported Platforms
MATLAB and HDL simulator cosimulation (function)	MATLAB	Fixed-Point Toolbox, Signal Processing Toolbox	Windows 32- and 64-bit; Linux 32- and 64-bit
MATLAB System object and HDL cosimulation	MATLAB and Fixed-Point Toolbox	Communications System Toolbox, DSP System Toolbox	Windows 32- and 64-bit; Linux 32- and 64-bit
Simulink and HDL simulator cosimulation	Simulink, Simulink Fixed Point, and Fixed-Point Toolbox	Signal Processing Toolbox, DSP System Toolbox	Windows 32- and 64-bit; Linux 32- and 64-bit

<b>Product Feature</b>	<b>Required Products</b>	<b>Recommended Products</b>	<b>Supported Platforms</b>
FPGA-in-the-Loop	Simulink, Simulink Fixed Point, Fixed-Point Toolbox, and HDL Coder	Filter Design HDL Coder	Windows 32- and 64-bit; Linux 32- and 64-bit
FPGA Automation for Filter Design	MATLAB and Filter Design HDL Coder		Windows 32- and 64-bit; Linux 32- and 64-bit
TLM Generator	Simulink Coder and Embedded Coder		Windows 32-bit and 64-bit; Linux 32- and 64-bit

## **Optional Application Software**

To create the most robust development environment for your application consider adding the following MathWorks products to your HDL Verifier setup:

### **For HDL Cosimulation**

- Communications System Toolbox
- DSP System Toolbox
- Signal Processing Toolbox
- Computer Vision System Toolbox

### **For Generating OSCI-Compatible TLM Components**

- Simulink Fixed Point
- Embedded Coder



# Getting Help

---

- “Information Overview” on page 4-2
- “Online Help” on page 4-3
- “Using “What’s This?” Context-Sensitive Help” on page 4-5
- “Demos and Tutorials” on page 4-7

## Information Overview

The following information is available with this product.

Getting Started	Explains what HDL Verifier is, the steps for installing and setting up the product, how you might apply the product to the hardware design process, and how to gain access to product documentation and online help. Directs you to product demos and tutorials.
“HDL Verification with Cosimulation”	Explains what you need to know to cosimulate with MATLAB or Simulink, using either as a component or a test bench, and your HDL simulator.
“SystemC TLM 2.0 Generation”	Provides instructions for creating a SystemC Transaction Level Model (TLM), which you can execute in any OSCI-compatible TLM 2.0 environment, including a commercial virtual platform.
“FPGA-in-the-Loop and FPGA Automation”	Provides instruction for creating, updating, and managing ISE projects using HDL generated from HDL Coder or Filter Design HDL Coder software.
“Block Reference”	Provides descriptions and examples of the blocks available for use in Simulink.
“Function Reference”	Provides descriptions and examples of the functions available for use with HDL Verifier software.
“Demos and Tutorials” on page 4-7	Provides examples of how you would use HDL Verifier software.

## Online Help

Online Help in the MATLAB Help Browser

You can access online help by either of the following methods:

- Click the HDL Verifier product link in the Help browser's Contents pane.
- Use the MATLAB `doc` command at the MATLAB command prompt:

```
>>doc edalink
```


Help for HDL Verifier MATLAB Functions

You can access function help by either of the following methods:

- By issuing the MATLAB `help` command. For example, enter the following command:

```
>>help hdldaemon
```

to get the MATLAB help for the `hdldaemon` function.

- By clicking the  icon in the MATLAB command window.

Context-sensitive “What’s This?” help

For options that appear in the TLM Generation and FPGA Automation GUIs. Click a GUI Help button or right-click on a GUI option to display help on that dialog or item. For more information on using context-sensitive help (CSH), see “Using “What’s This?” Context-Sensitive Help” on page 4-5.

Block Reference Pages

You can access block reference pages through the Simulink interface. You can also access these block reference

pages by clicking **Help** on any block dialog box.

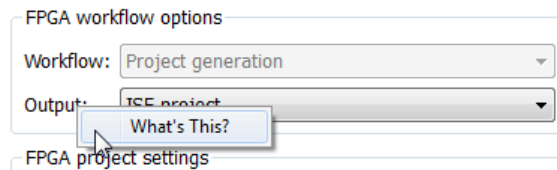


## Using “What’s This?” Context-Sensitive Help

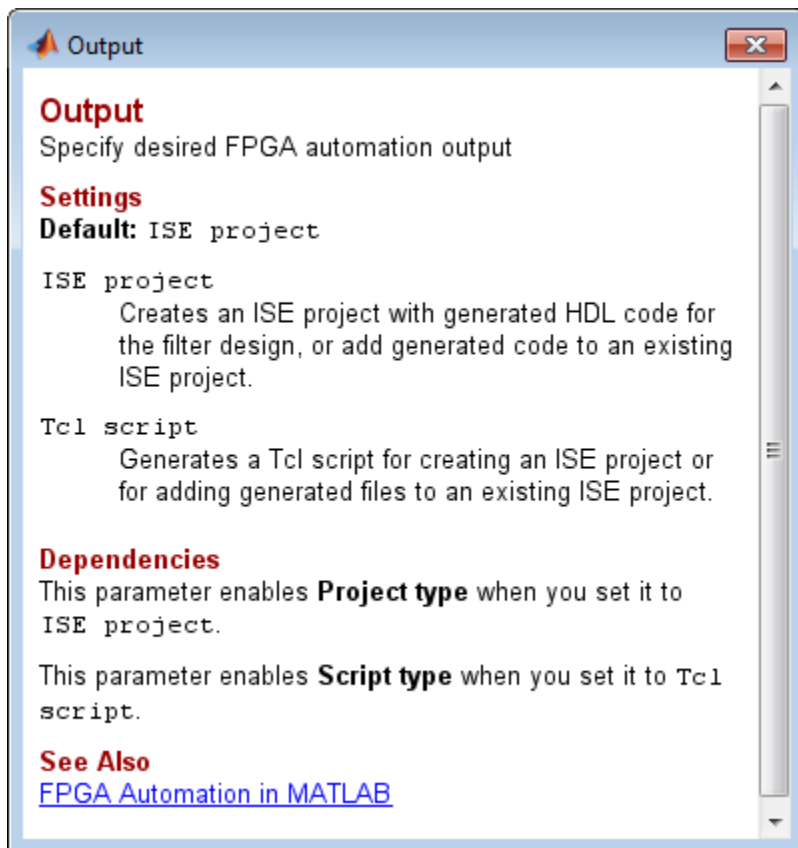
“What’s This?” context-sensitive help (CSH) topics are available for each option in the HDL Verifier GUIs. Use CSH to find more information when using the GUIs to configure options for TLM generation or FPGA project management.

To use CSH, follow these steps:

- 1 Place your cursor over the label or control for an option.
- 2 Right-click. A “What’s This?” button appears. The following display shows the “What’s This?” button appearing after a right-click on the **Workflow** option in the EDA Link pane.



- 3 Left-click on “What’s This?” to view the CSH that describes the option.



## Demos and Tutorials

In this section...
“Demos” on page 4-7
“Tutorials” on page 4-7

### Demos

The demos give you a quick view of the product’s capabilities and application examples that you can run with limited product exposure. You can find the HDL Verifier demos with the online documentation. To access demos, type at the MATLAB command prompt:

```
>> demos
```

Select **HDL Verifier > Demos** from the navigational pane.

### Tutorials

Tutorials provide procedural instruction on how to apply the product. Some focus on features while others focus on application scenarios. The tutorials listed here have a feature focus and address the use of the HDL Verifier software with HDL simulators and Simulink or MATLAB.

- “Verify Raised Cosine Filter Design With Generated MATLAB Component”
- “Verify Raised Cosine Filter Design With Generated Simulink Test Bench”
- “Verify HDL Model with MATLAB Testbench (Tutorial)”
- “Verify HDL Model with Simulink Test Bench (Tutorial)”
- “Visually Comparing Simulink Signals with HDL Signals (Tutorial)”



# HDL Code Importing

---

- “Generate HDL Cosimulation Interfaces from Existing HDL Code” on page 5-2
- “Import HDL Code For FPGA-in-the-Loop Verification” on page 5-24

## Generate HDL Cosimulation Interfaces from Existing HDL Code

In this section...
“Create a MATLAB Function From Existing HDL Code” on page 5-2
“Create a MATLAB System Object From Existing HDL Code” on page 5-9
“Create an HDL Cosimulation Block From Existing HDL Code” on page 5-10
“Perform Cosimulation” on page 5-21

### Create a MATLAB Function From Existing HDL Code

- “Invoke Cosimulation Wizard” on page 5-2
- “Select Cosimulation Type” on page 5-3
- “Select HDL Files to Import” on page 5-4
- “Specify HDL Compilation Commands” on page 5-5
- “Select HDL Modules for Cosimulation” on page 5-6
- “Specify Callback Schedule Parameters” on page 5-8
- “Generate Scripts” on page 5-9
- “Complete the Component or Test Bench Function” on page 5-9

### Invoke Cosimulation Wizard

**1** Start MATLAB.

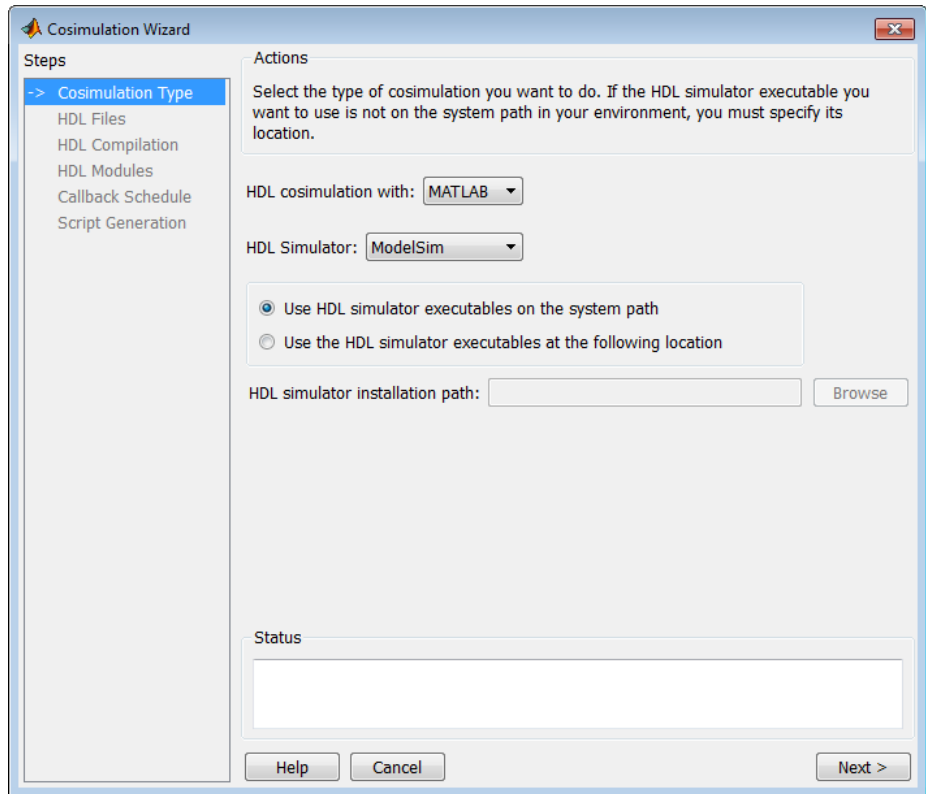
**2** Enter the following command at the command prompt:

```
>> cosimWizard
```

The Cosimulation Wizard opens.

Continue with the next task: “Select Cosimulation Type” on page 5-3.

## Select Cosimulation Type



**1** Select **HDL cosimulation with MATLAB** to create a MATLAB function template.

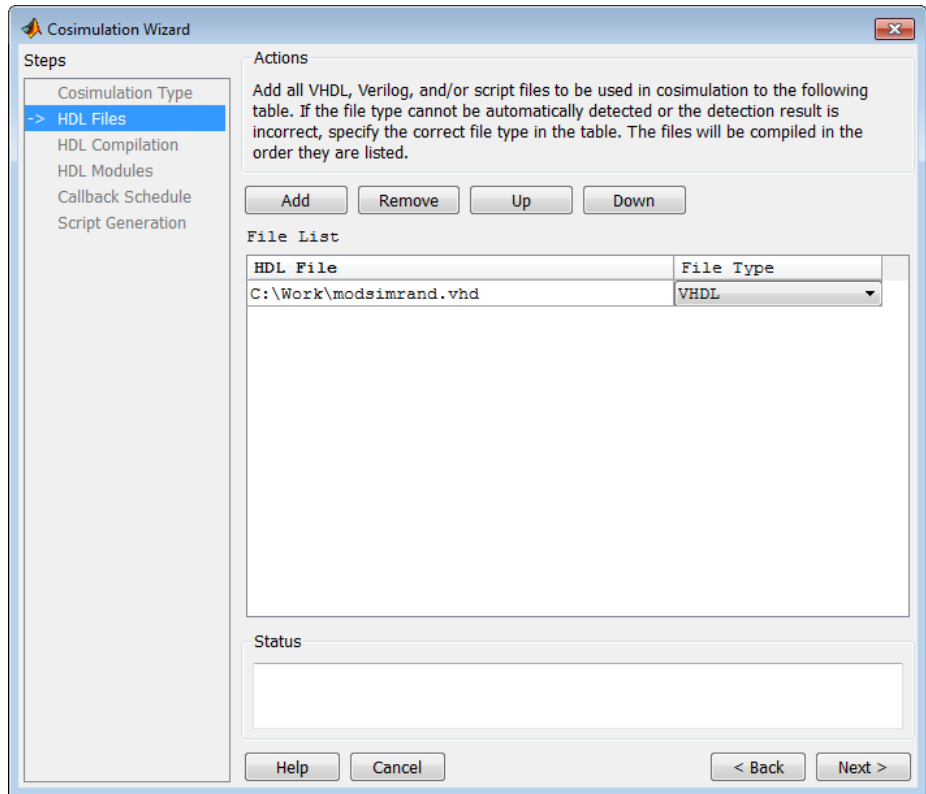
### **2 HDL Simulator**

Select your HDL simulator. You may choose either Cadence Incisive or Mentor Graphics ModelSim.

**3** Specify where the Cosimulation Wizard can find your HDL simulator executables. You must enter a valid path to the HDL simulator executables before you can continue.

4 Click **Next**.

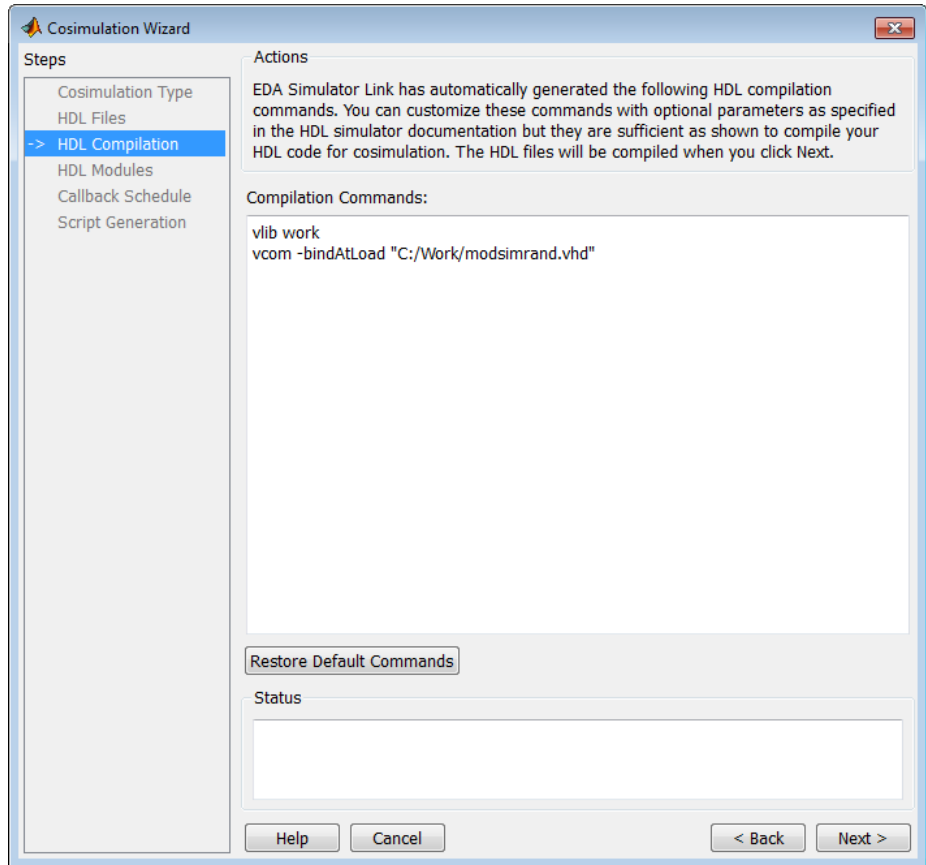
## Select HDL Files to Import



- 1 Add files by clicking **Add**.
- 2 Remove files by first highlighting the file name in the **File List**, and then clicking **Remove**.
- 3 Move file positions in the list by selecting the file to move and clicking **Up** or **Down**.
- 4 Click **Next**.



## Specify HDL Compilation Commands

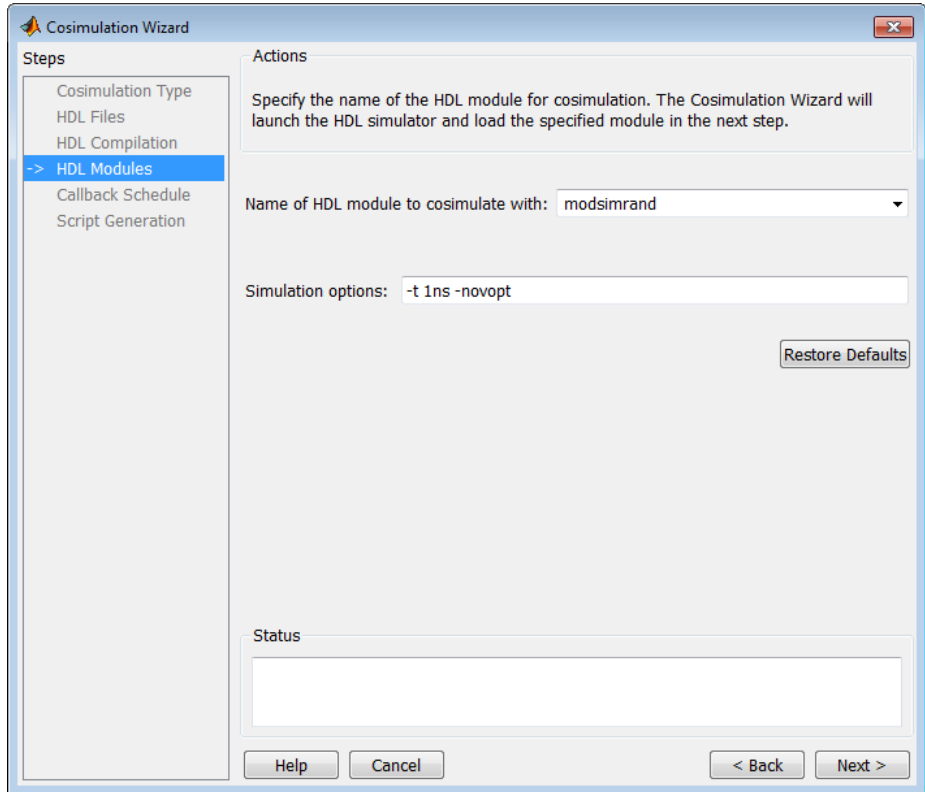


- 1 Review the automatically generated HDL compilation commands. Enter any changes to the commands in the **Compilation Commands** box.
- 2 (Optional) Click **Restore Default Commands** to go back to the automatically generated HDL compilation commands at any time.
- 3 Click **Next**.

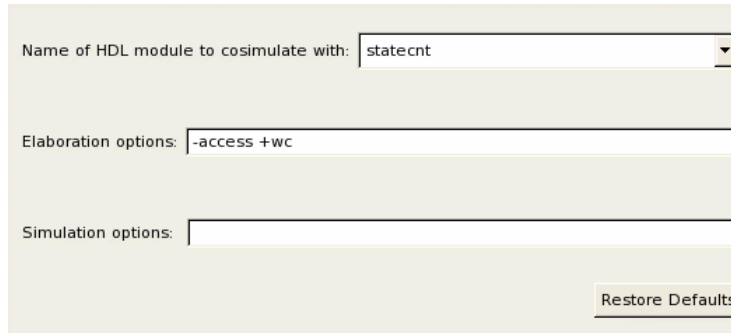
## Select HDL Modules for Cosimulation

The dialog box for **HDL Module Selection** varies depending on whether you have set **Cosimulation type** to ModelSim or Incisive®.

### HDL Module Selection for ModelSim



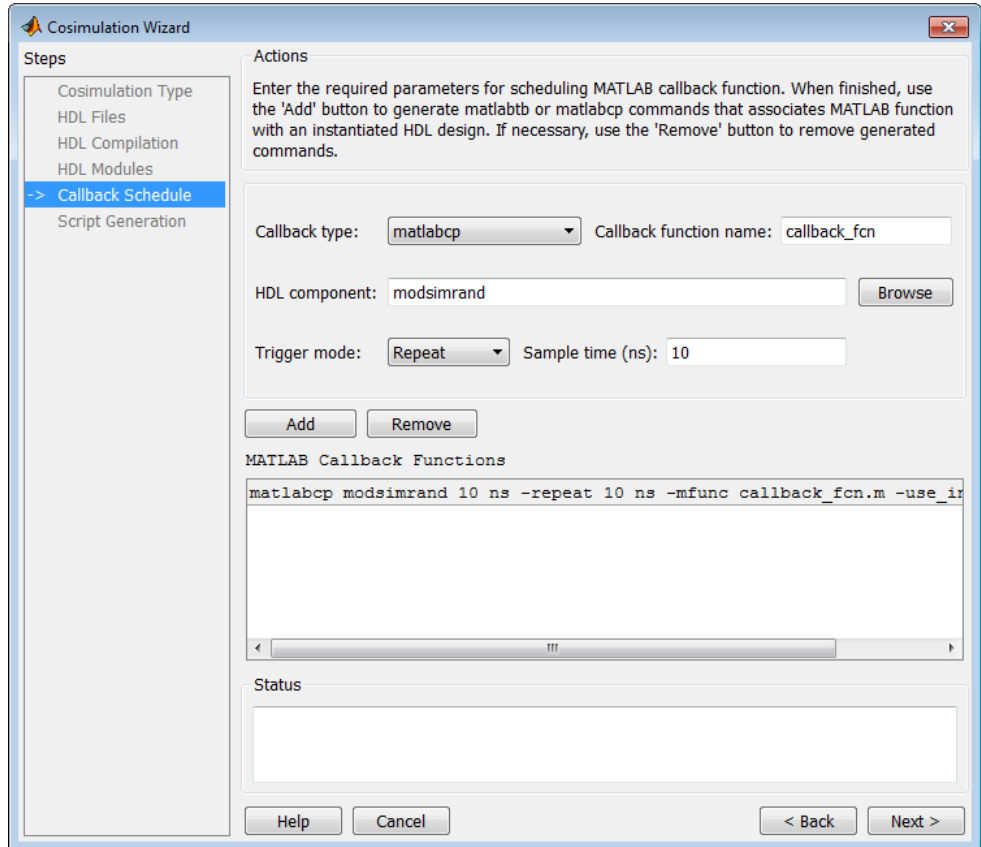
## HDL Module Selection for Incisive



The screenshot shows a dialog box titled "HDL Module Selection for Incisive". It contains three input fields and a button. The first field, labeled "Name of HDL module to cosimulate with:", has a dropdown menu with "statecnt" selected. The second field, labeled "Elaboration options:", contains the text "-access +wc". The third field, labeled "Simulation options:", is empty. A button labeled "Restore Defaults" is located at the bottom right of the dialog box.

- 1** Enter the name of the module where you see **Name of HDL module to cosimulate with**.
- 2** Provide any additional elaboration options in **Elaboration options**.
- 3** Specify additional simulation options where you see **Simulation options**.  
If you change your mind about the options you've added or changed, click **Restore Defaults**.
- 4** Click **Next** .

## Specify Callback Schedule Parameters

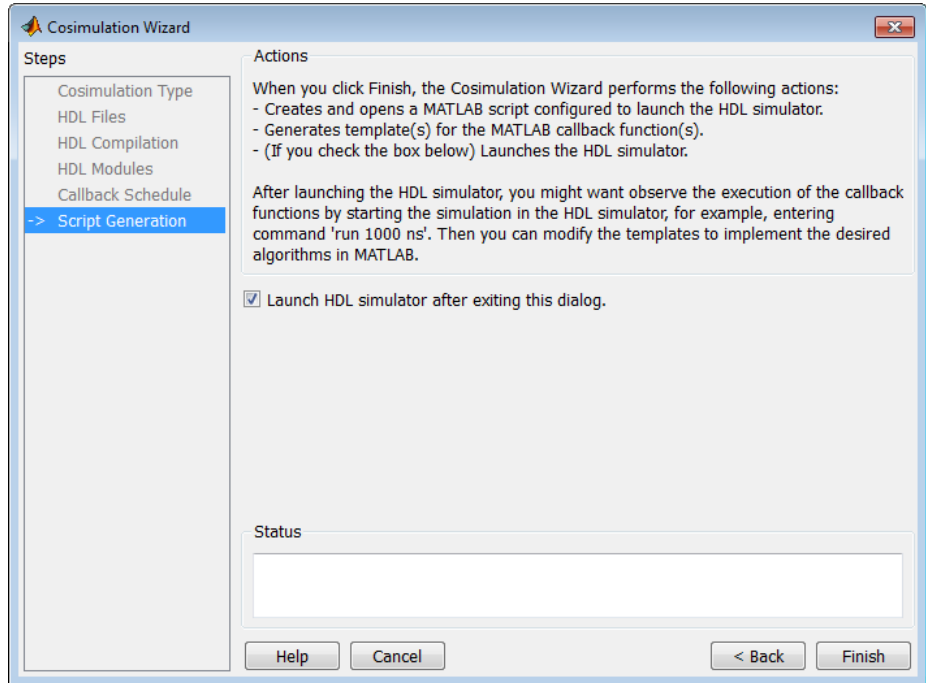


- 1 Enter one or multiple component or test bench function callbacks from the HDL simulator.
- 2 When you finish specifying the callback function parameters, click **Add** to add the command to the MATLAB Callback Functions list.

If you have more callback functions you want to schedule, repeat the preceding steps. If you want to remove any callback functions, highlight the line for that function and click **Remove**.

- 3 Click **Next**.

## Generate Scripts



- Click **Back** to review or change your settings.
- Click **Finish** to generate the scripts.

### Complete the Component or Test Bench Function

Complete the template using the MATLAB Editor. The generated template contains some simple port I/O instructions and empty routines where you add your own code.

### Create a MATLAB System Object From Existing HDL Code

For a guided tutorial on how to use the Cosim Wizard to create a MATLAB System object for HDL Cosimulation, see Cosimulation Wizard for MATLAB System Object (IN) or Cosimulation Wizard for MATLAB System Object (MQ).

You can also find out more in the user guide section “HDL Cosimulation Using MATLAB System Object”.

### **Create an HDL Cosimulation Block From Existing HDL Code**

- “Invoke Cosimulation Wizard” on page 5-10
- “Select Cosimulation Type” on page 5-11
- “Select HDL Files to Import” on page 5-12
- “Specify HDL Compilation Commands” on page 5-13
- “Select HDL Modules for Cosimulation” on page 5-14
- “Configure Simulink Ports” on page 5-16
- “Specify Output Port Details” on page 5-17
- “Specify Clock and Reset Details” on page 5-18
- “Confirm or Change Start Time Alignment” on page 5-19
- “Generate Block” on page 5-20
- “Complete the Simulink Model” on page 5-20

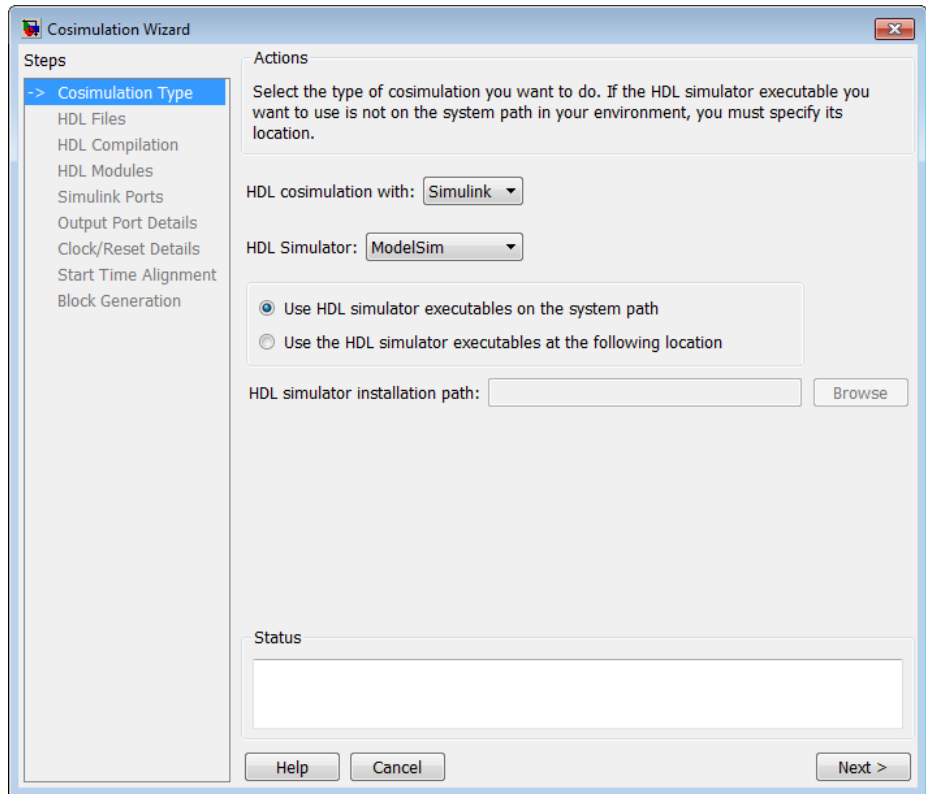
### **Invoke Cosimulation Wizard**

- 1 Start MATLAB.
- 2 Enter the following command at the command prompt:

```
>> cosimWizard
```

The Cosimulation Wizard opens.

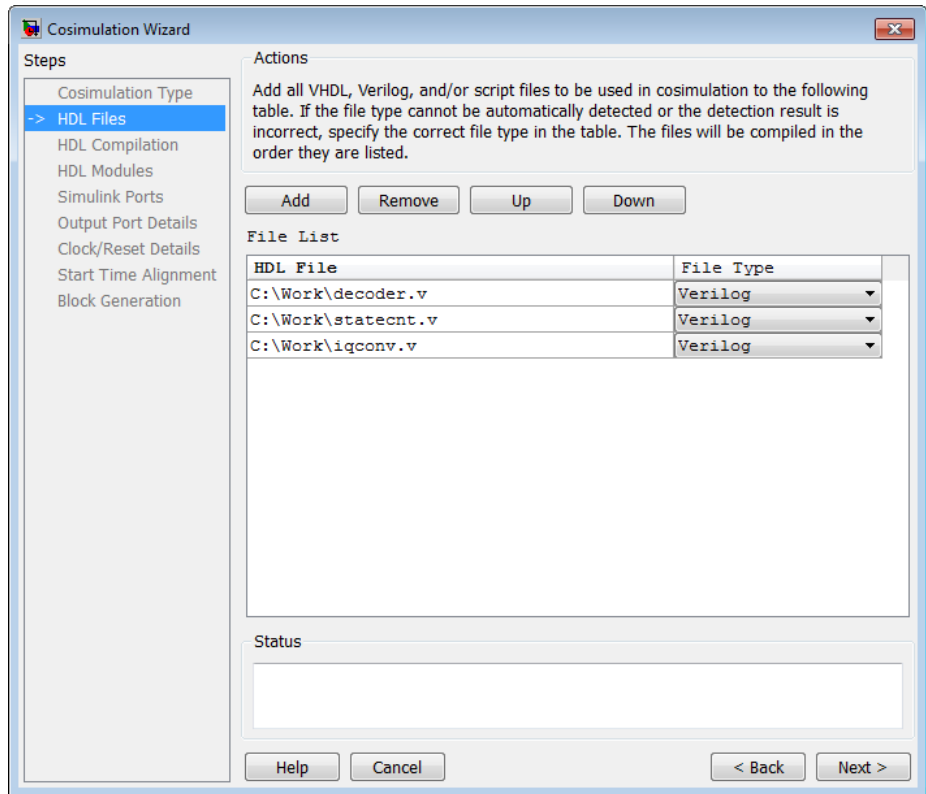
## Select Cosimulation Type



- 1** Select **HDL cosimulation with** Simulink to create an HDL Cosimulation block using the HDL code you provide to define the ports, clocks, and resets.
- 2 HDL Simulator**  
Select your HDL simulator. You may choose either Cadence Incisive or Mentor Graphics ModelSim.
- 3** Specify where the Cosimulation Wizard can find your HDL simulator executables. You must enter a valid path to the HDL simulator executables before you can continue.

4 Click **Next**.

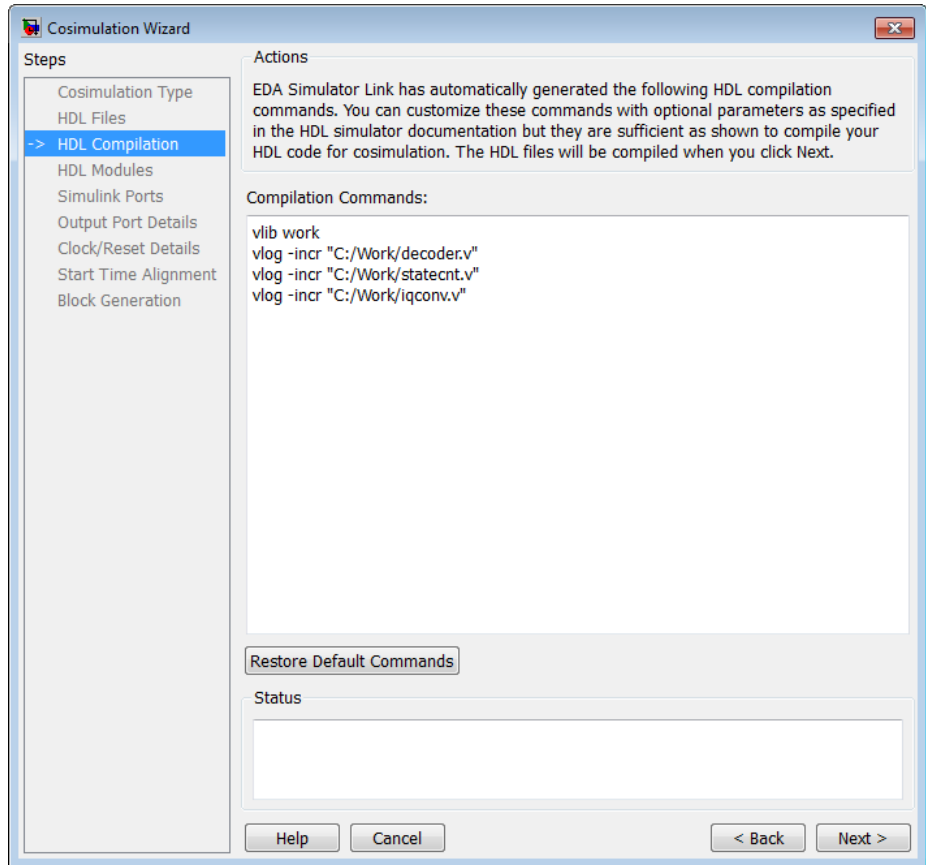
### Select HDL Files to Import



- 1 Add files by clicking **Add**.
- 2 Remove files by first highlighting the file name in the **File List**, and then clicking **Remove**.
- 3 Move file positions in the list by selecting the file to move and clicking **Up** or **Down**.
- 4 Click **Next**.



## Specify HDL Compilation Commands

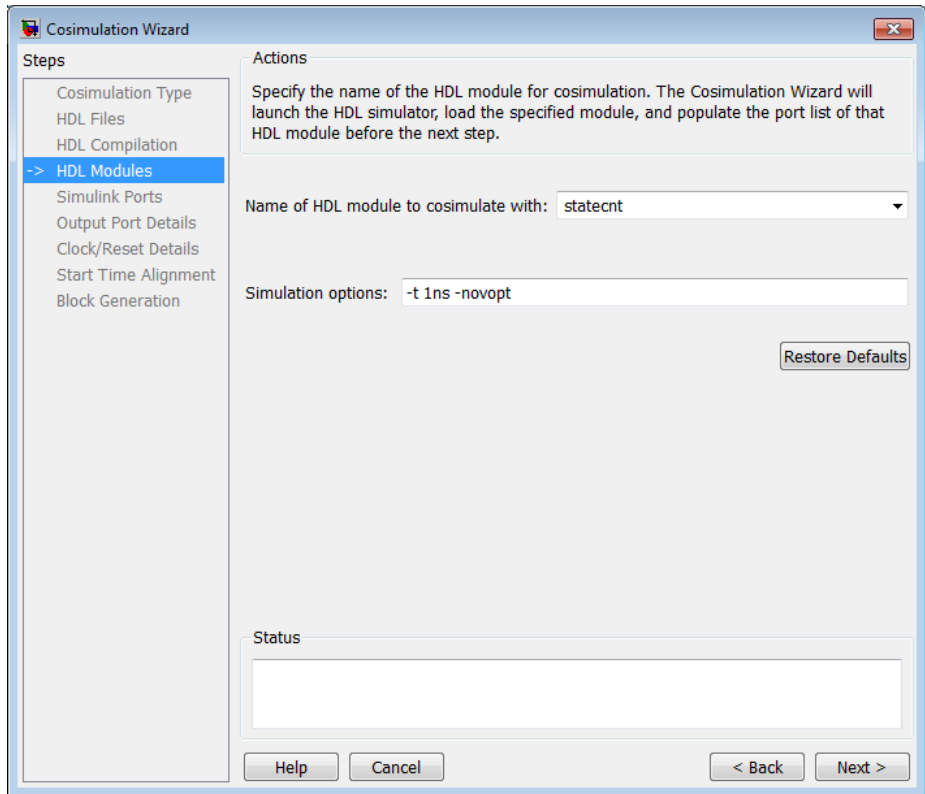


- 1 Review the automatically generated HDL compilation commands. Enter any changes to the commands in the **Compilation Commands** box.
- 2 (Optional) Click **Restore Default Commands** to go back to the automatically generated HDL compilation commands at any time.
- 3 Click **Next**.

## Select HDL Modules for Cosimulation

The dialog box for **HDL Module Selection** varies depending on whether you have set **Cosimulation type** to ModelSim or Incisive.

### HDL Module Selection for ModelSim

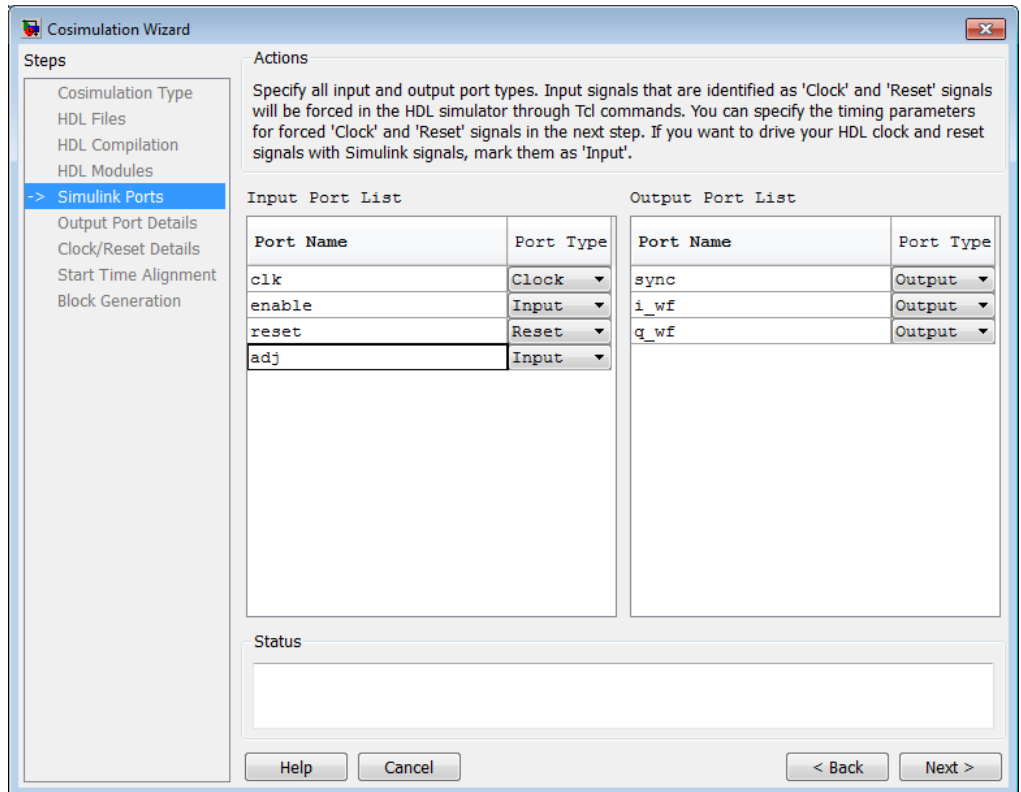


## HDL Module Selection for Incisive

The screenshot shows a dialog box titled "HDL Module Selection for Incisive". It contains three input fields and a button. The first field, labeled "Name of HDL module to cosimulate with:", has a dropdown menu with "statecnt" selected. The second field, labeled "Elaboration options:", contains the text "-access +wc". The third field, labeled "Simulation options:", is empty. A button labeled "Restore Defaults" is located at the bottom right of the dialog box.

- 1** Enter the name of the module where you see **Name of HDL module to cosimulate with**.
- 2** Provide any additional elaboration options in **Elaboration options**.
- 3** Specify additional simulation options where you see **Simulation options**.  
If you change your mind about the options you've added or changed, click **Restore Defaults**.
- 4** Click **Next** .

## Configure Simulink Ports

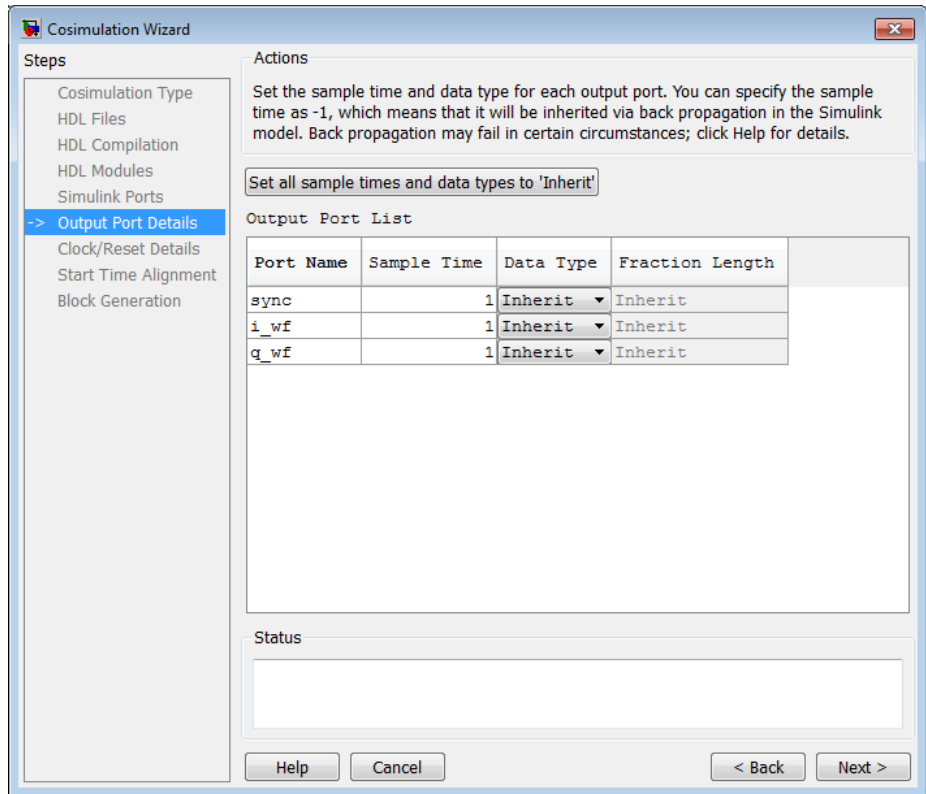


**1** For **Port Type**, confirm the auto-selected types or specify one of the following:

- Input
- Clock
- Reset
- Unused

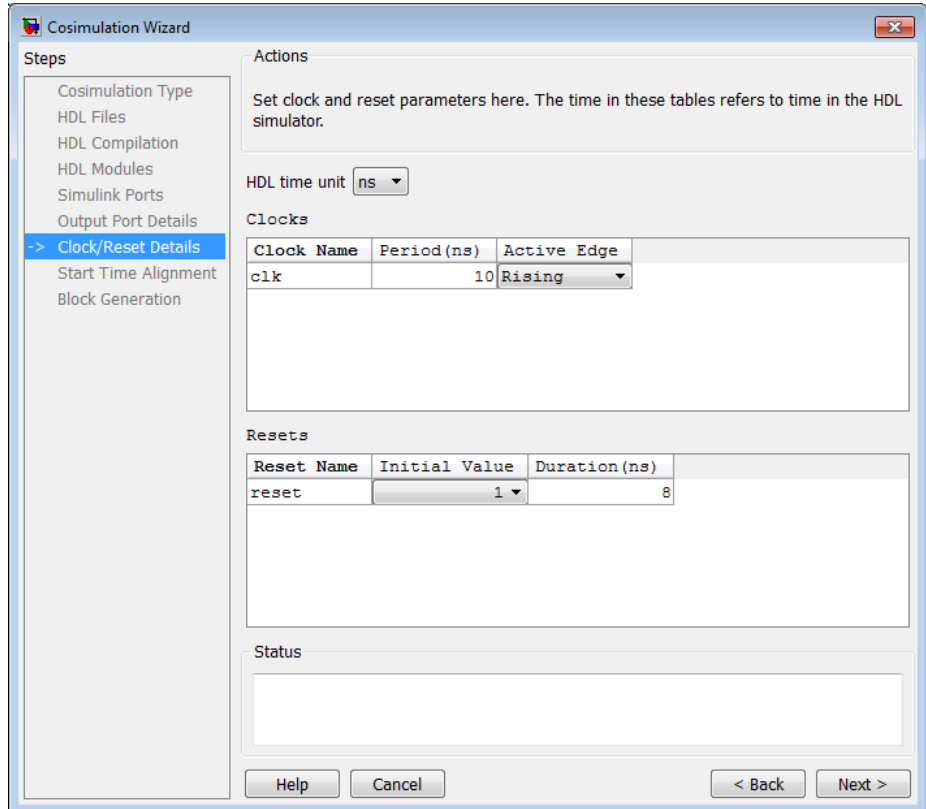
**2** Click **Next**.

## Specify Output Port Details



- 1 Verify that the default sample time and the default data type what you expect them to be. These settings are consistent with the way the HDL Cosimulation block mask (**Ports** tab) sets default settings for output ports. Do not change these values unless you are certain you do not want the default values.
- 2 Click **Next**.

## Specify Clock and Reset Details



- 1 Verify that the default clock settings and the default reset settings are as you expected. Do not change these settings unless you are certain you do not want the default values.

The next screen provides a visual display of the simulation start time where you can review how the clocks and resets line up.

- 2 Click **Next**.

## Confirm or Change Start Time Alignment

The screenshot shows the 'Cosimulation Wizard' dialog box, specifically the 'Start Time Alignment' step. The 'Steps' list on the left includes: Cosimulation Type, HDL Files, HDL Compilation, HDL Modules, Simulink Ports, Output Port Details, Clock/Reset Details, Start Time Alignment (selected), and Block Generation. The 'Actions' section contains the following text: 'The diagram below shows the current settings for forced 'Clock' and 'Reset' signals. The red line represents the time in the HDL simulation at which Simulink will start (i.e. cosimulation will start). To change the Simulink start time relative to the HDL simulation time, enter the new start time below. To avoid a race condition, make sure the start time does not coincide with the active edge of any clock signal. You can do so by moving the start time or by changing the clock active edge in the previous step (click Back).' Below this text is a text input field labeled 'HDL time to start cosimulation (ns):' with the value '10' and an 'Update Diagram' button. Two timing diagrams are shown: the top one for 'clk' and the bottom one for 'reset'. Both diagrams have a time axis from 0 to 25 ns. The 'clk' signal is a square wave with rising edges at 5, 15, and 25 ns. The 'reset' signal is high from 0 to 8 ns and then drops to 0. A vertical red line is positioned at 10 ns in both diagrams, indicating the start time of the cosimulation. At the bottom of the dialog are 'Help', 'Cancel', '< Back', and 'Next >' buttons.

Cosimulation Wizard

Steps

- Cosimulation Type
- HDL Files
- HDL Compilation
- HDL Modules
- Simulink Ports
- Output Port Details
- Clock/Reset Details
- > Start Time Alignment
- Block Generation

Actions

The diagram below shows the current settings for forced 'Clock' and 'Reset' signals. The red line represents the time in the HDL simulation at which Simulink will start (i.e. cosimulation will start).

To change the Simulink start time relative to the HDL simulation time, enter the new start time below. To avoid a race condition, make sure the start time does not coincide with the active edge of any clock signal. You can do so by moving the start time or by changing the clock active edge in the previous step (click Back).

HDL time to start cosimulation (ns): 10

clk

reset

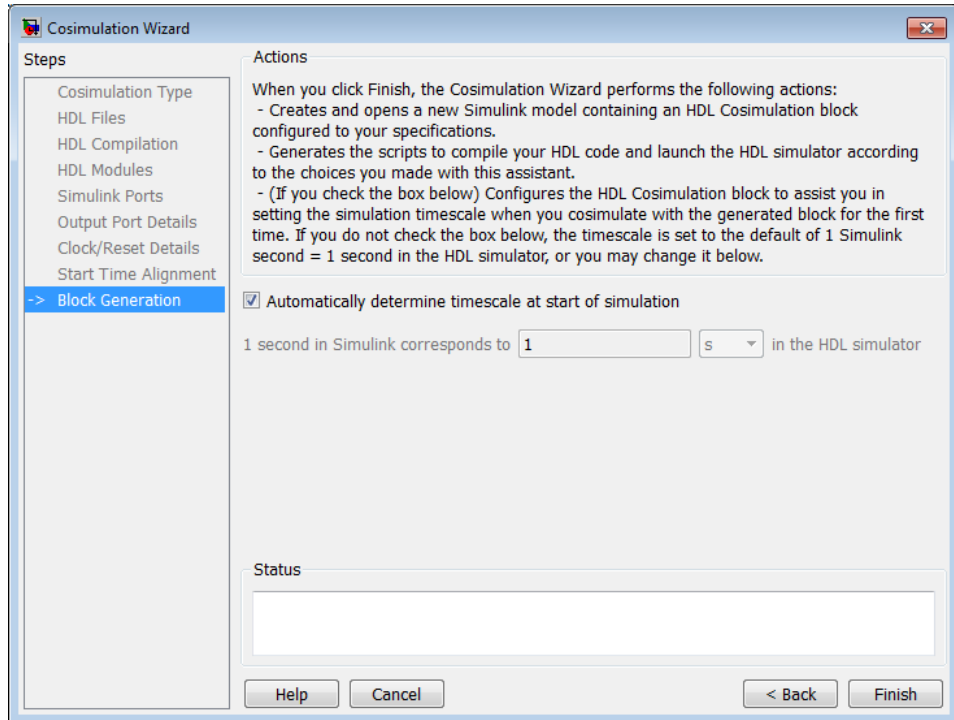
Status

Help Cancel < Back Next >

- 1 Verify that the rising or falling edge is set where you want it (from the previous step) by looking at the start time and the reset signal.

- 2 Make sure that the start time is where you want it.
- 3 Click **Next**.

### Generate Block



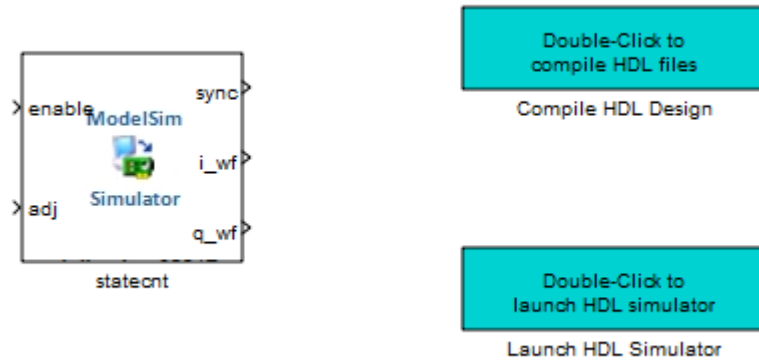
Indicate whether you want HDL Verifier software to automatically determine the timescale when you start the simulation or if you'd prefer to determine the timescale yourself. The default is to automatically determine timescale.

- Click **Back** to review or change your settings.
- Click **Finish** to generate the HDL Cosimulation block.

### Complete the Simulink Model

Insert the generated HDL Cosimulation block into your existing model.





Make any desired changes or updates to your Simulink model before you begin cosimulation.

## Perform Cosimulation

After you finish creating a function or block, select the topic that describes how you are planning to cosimulate your HDL code.

If you generated a component (`matlabcp`) or test bench (`matlabtb`) function for cosimulation with MATLAB, select one of the following topics:

### MATLAB Cosimulation

Task to Perform...	Read...
Simulate an HDL component with a MATLAB test bench	<ul style="list-style-type: none"> <li>Because you already have a completed test bench function written, you can pick up at this step: “Place Test Bench Function on MATLAB Search Path”. If you need help finishing the function, see “Create an HDL Verifier MATLAB Component Function”.</li> <li>For the full process and description, see “HDL</li> </ul>

**MATLAB Cosimulation (Continued)**

Task to Perform...	Read...
	Cosimulation Using MATLAB Test Bench Function”.
Replace HDL component with a MATLAB function	<ul style="list-style-type: none"> <li>• Because you already have a completed component function written, you can pick up at this step: “Place Component Function on MATLAB Search Path”. If you need help finishing the function, see .</li> <li>• For the full process and description, see “HDL Cosimulation Using MATLAB Component Function”.</li> </ul>

If you generate an HDL Cosimulation block for cosimulation with Simulink, select one of the following topics:

**Simulink Cosimulation**

Task to Perform...	Read...
Simulate HDL component with Simulink test bench	<ul style="list-style-type: none"> <li>• Because you already have the HDL design and a Simulink test bench model with the customized HDL Cosimulation block, and the HDL simulator is running, you can pick up at this step: “Run a Test Bench Cosimulation Session”. First, however, review the entire “Simulating an HDL Component in a Simulink Test Bench”.</li> <li>• For the full process and description, see</li> </ul>

**Simulink Cosimulation (Continued)**

Task to Perform...	Read...
	<p>“Simulating an HDL Component in a Simulink Test Bench Environment”.</p>
<p>Replace HDL component with Simulink algorithm</p>	<ul style="list-style-type: none"> <li>• Because you already have the HDL design and a Simulink component model with the customized HDL Cosimulation block, and the HDL simulator is running, you can pick up at this step: “Run a Component Cosimulation Session”. First, however, review the entire “Using Simulink to Replace an HDL Component”.</li> <li>• For the full process and description, see “Replacing an HDL Component with a Simulink Algorithm”.</li> </ul>

For additional help on HDL Verifier topics, see the “Information Overview” on page 4-2.

## Import HDL Code For FPGA-in-the-Loop Verification

In this section...
“Preparing to Use the FPGA-in-the-Loop (FIL) Wizard” on page 5-24
“Running the FIL Wizard” on page 5-24
“Performing FIL Simulation” on page 5-29

### Preparing to Use the FPGA-in-the-Loop (FIL) Wizard

Before beginning:

- 1 Have your HDL code ready and the original model opened.
- 2 Set up your project tools by specifying the path to the executables. See “Generate FIL Block” in the User Guide.

For more detailed information, see “FPGA-in-the-Loop (FIL)”. For a demonstration of FIL, see the FPGA-in-the-Loop demos under HDL Verifier.

#### Altera Board with Linux

If you are using the Altera board and a Linux distribution supported by Altera, you should first read the “USB-Blaster Download Cable User Guide” provided on the Altera web site: [http://www.altera.com/literature/ug/ug\\_usb\\_blstr.pdf](http://www.altera.com/literature/ug/ug_usb_blstr.pdf). Specifically, to program the bit file you must be a superuser. The user guide provides instructions for making a one-time modification to a rules file to give you that permission.

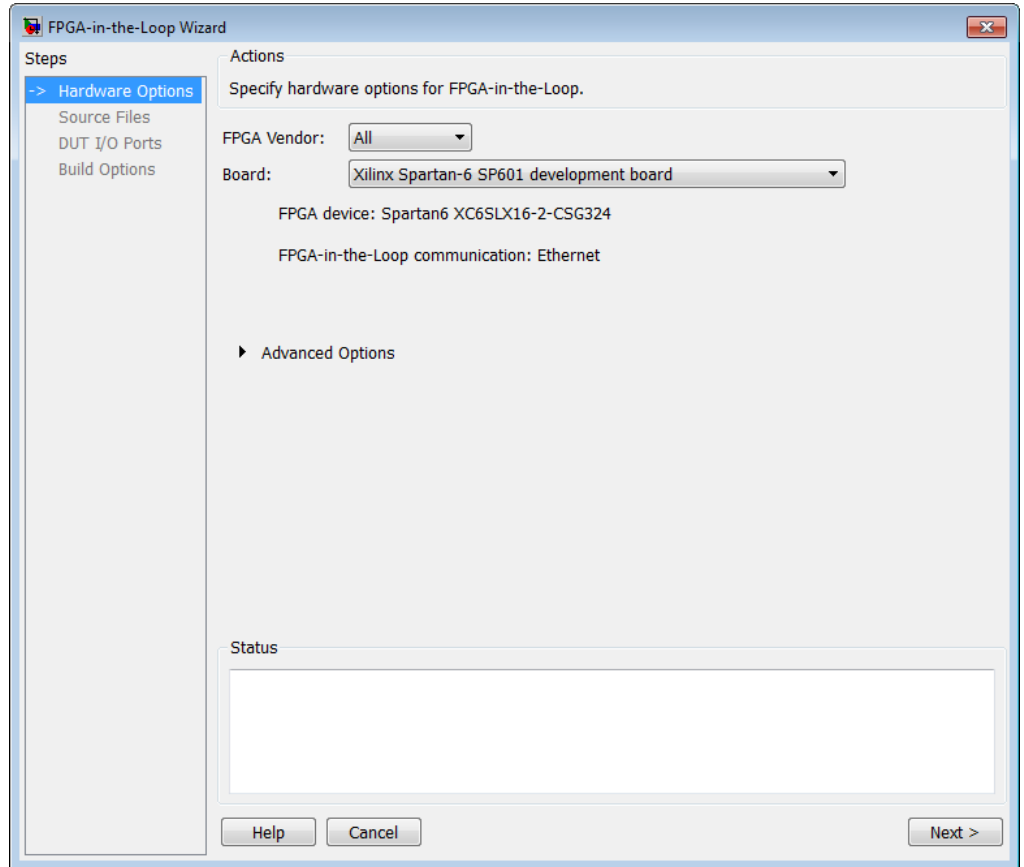
### Running the FIL Wizard

- 1 At the MATLAB prompt, enter the following:

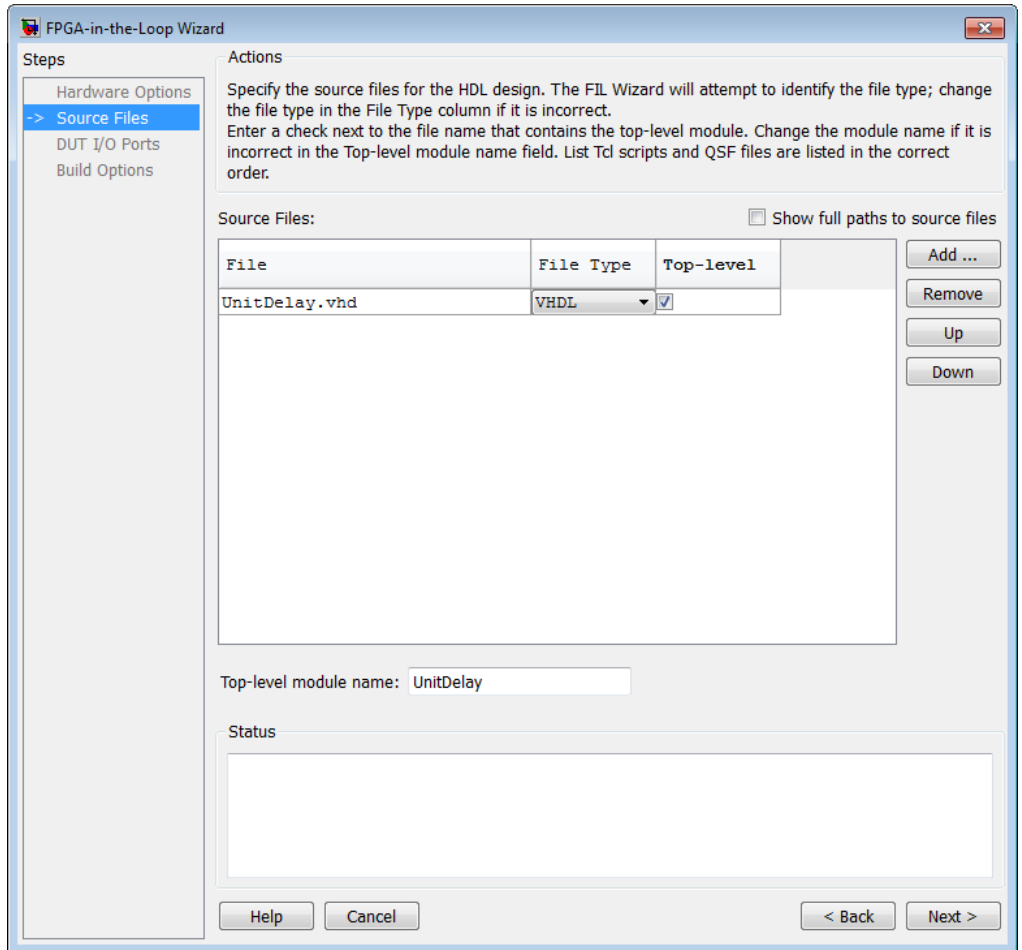
```
>> filWizard
```

- 2 Select the FPGA vendor you are using (FPGA design software) to display boards supported for that vendor. Choose either Altera or Xilinx. If you leave the selection at All, all supported boards will be displayed in the pull-down menu.

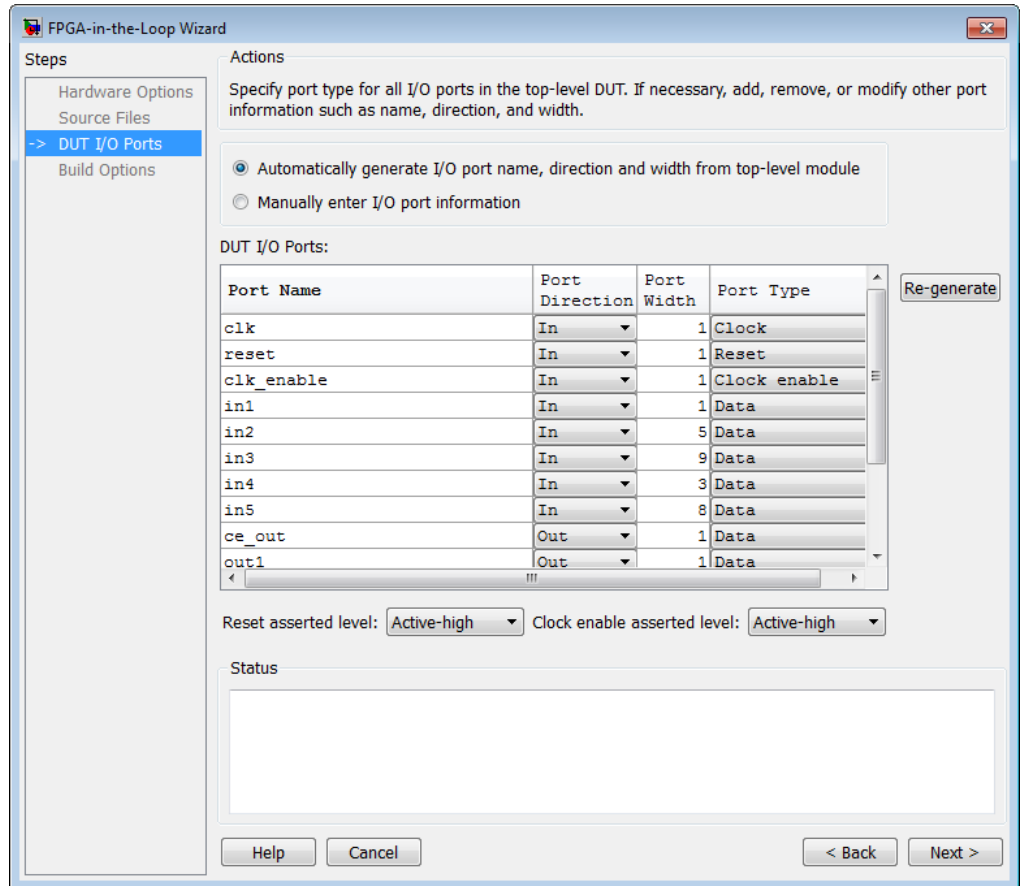
- 3 Select the board you are using. Adjust the Board IP address if applicable. Click **Next** to continue.



- 4 Select your HDL source files. Indicate the top-level module. Click **Next** to continue.

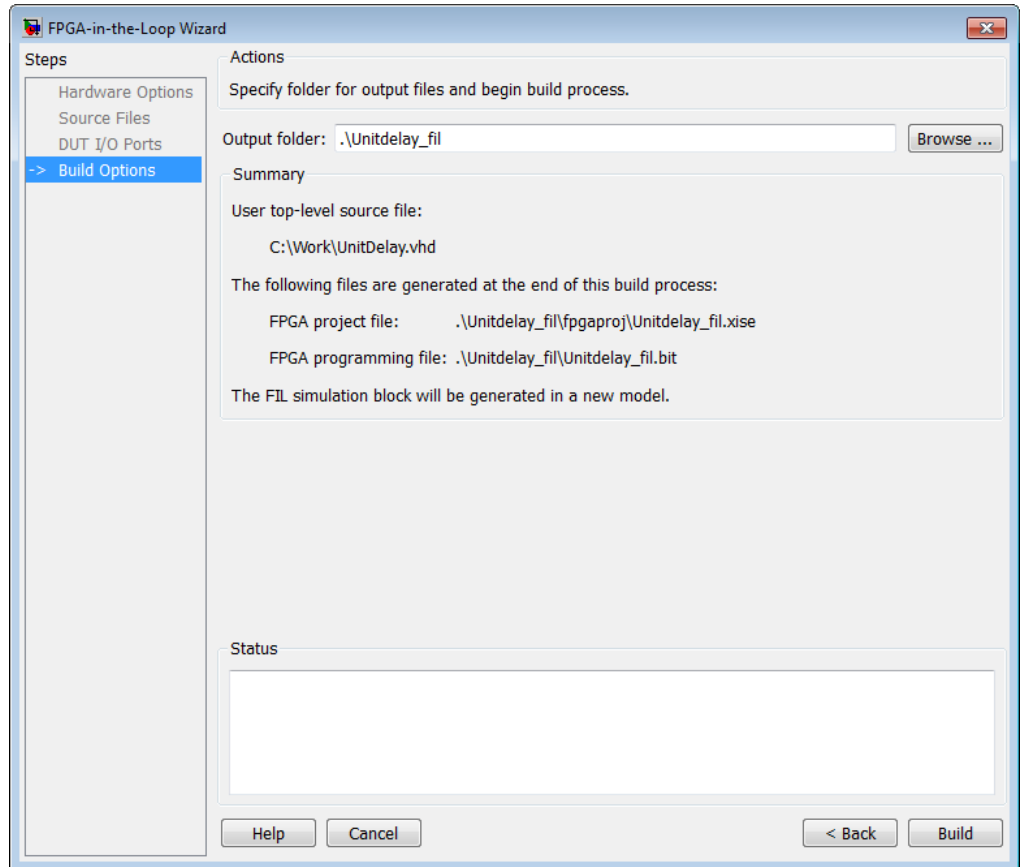


5 Review the DUT I/O ports. Change any settings if desired. Click **Next** to continue.



There must be at least one input and one output data port.

- 6 Select the output folder for the programming files. Click **Next** to continue.

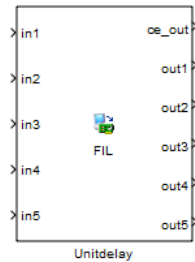


### 7 Click **Build**.

During the build process, the following actions occur:

- The FIL Wizard generates a FIL block named after the top-level module and places it in a new model.





Follow these steps to perform FPGA-in-the-Loop simulation:

1. Drag this block into a Simulink model.
2. Connect all inputs and outputs.
3. Open the block mask to optionally change:
  - output sample times, data types, and frame size
  - HDL overclocking factor
 Usually the mask defaults will work in your model without modification.
4. Attach the FPGA development board to your host computer (FPGA programming cable and Gigabit Ethernet cable).
5. Set the host IP address (see product documentation).
6. Open the block mask and click Load to download the FPGA programming file.

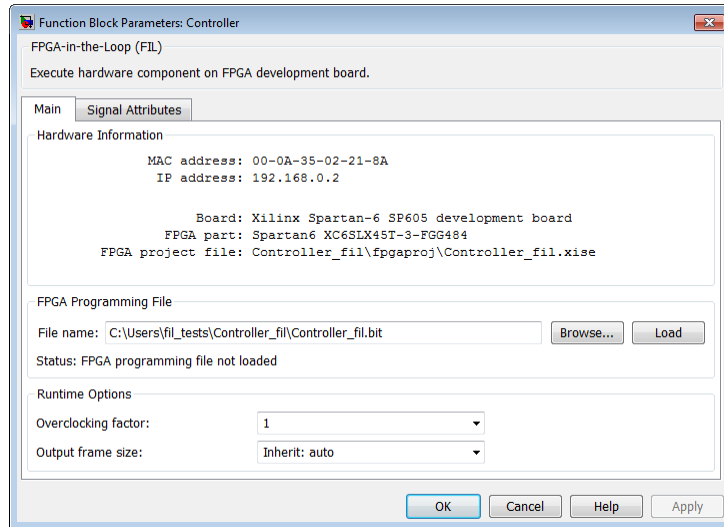
See the product documentation for more detailed instructions and troubleshooting tips.

- After new model generation, the FIL Wizard opens a command window.
  - In this window, the FPGA design software performs synthesis, fit, place-and-route, timing analysis, and FPGA programming file generation.
  - When the process is finished, a message in the command window lets you know you can close the window.

For more detailed information about the FIL Wizard, see “Generating a FIL Block Using the FIL Wizard”. For more information about the FIL process, see “FPGA-in-the-Loop (FIL)”. For a demonstration of FIL, see the FPGA-in-the-Loop demos under HDL Verifier.

## Performing FIL Simulation

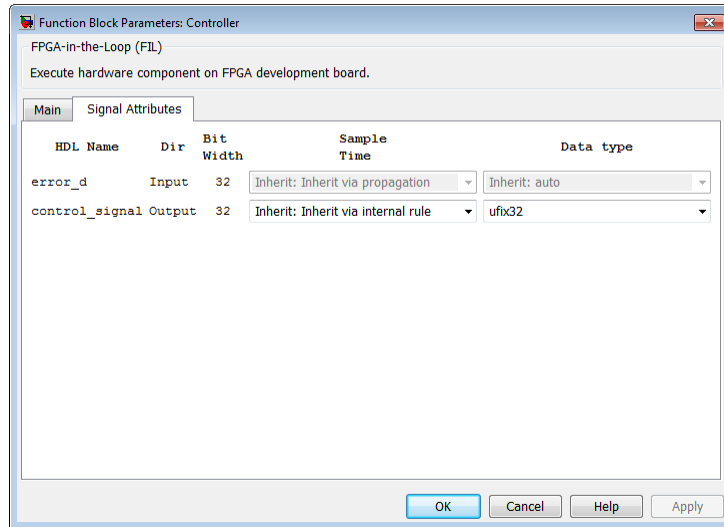
- 1** Insert the generated FIL block into the existing model.
- 2** Open the block mask and load the programming files.



### Altera Board with Linux

If you are using the Altera board and a Linux distribution supported by Altera, you should first read the “USB-Blaster Download Cable User Guide” provided on the Altera web site: [http://www.altera.com/literature/ug/ug\\_usb\\_blstr.pdf](http://www.altera.com/literature/ug/ug_usb_blstr.pdf). Specifically, to program the bit file you must be a superuser. The user guide provides instructions for making a one-time modification to a rules file to give you that permission.

- 3 Make any other adjustments on the block mask, if desired.



#### 4 Run the FIL simulation.

For more detailed information, see “FPGA-in-the-Loop (FIL)”. For a demonstration of FIL, see the FPGA-in-the-Loop demos under HDL Verifier.



## A

- application software 3-4
- application specific integrated circuits (ASICs) 1-1
- ASICs (application specific integrated circuits) 1-1

## B

- blocksets
  - installing 2-3

## C

- client
  - for MATLAB and HDL simulator links 1-3
  - for Simulink and HDL simulator links 1-3
- client/server environment
  - MATLAB and HDL simulator 1-3
  - Simulink and HDL simulator 1-3
- communication
  - modes of 1-8
- Communications System Toolbox™
  - as optional software 3-4
- cosimulation environment
  - MATLAB and HDL simulator 1-3
  - Simulink and HDL simulator 1-3

## D

- demos 4-7
  - for HDL Verifier™ 4-1
  - for use with FPGA implementations 4-1
- documentation
  - overview 4-1
  - for use with FPGA implementations 4-1
- DSP System Toolbox™
  - as optional software 3-4

## E

- EDA (Electronic Design Automation) 1-1
- Electronic Design Automation (EDA) 1-1
- environment
  - cosimulation with MATLAB and HDL simulator 1-3
  - cosimulation with Simulink and HDL simulator 1-3

## F

- field programmable gate arrays (FPGAs) 1-1
- FPGAs (field programmable gate arrays) 1-1

## H

- hardware description language (HDL). *See* HDL
- HDL (hardware description language) 1-1
- HDL Cosimulation block
  - in HDL Verifier™ environment 1-3
- HDL simulators
  - in HDL Verifier™ environment 1-3
  - installing 2-3
  - working with Simulink links to 1-3
- HDL Verifier™ software
  - definition of 1-1
  - installing 2-2
- help
  - for HDL Verifier™ software 4-1
  - for use with FPGA implementations 4-1

## I

- Incisive
  - in HDL Verifier™ cosimulation environment 1-3
  - working with MATLAB links to 1-3
- Incisive or NC simulators
  - as required software 3-4
- installation

- of HDL Verifier™ software 2-2
- of related software 2-3

## L

links

- MATLAB and HDL simulator 1-3
- Simulink and HDL simulator 1-3

## M

MATLAB

- as required software 3-4
- in HDL Verifier™ cosimulation environment 1-3
- installing 2-3
- working with HDL simulator links to 1-3

MATLAB functions

- test bench 1-3

MATLAB server

- function for invoking 1-3

ModelSim

- in HDL Verifier™ cosimulation environment 1-3
- working with MATLAB links to 1-3

ModelSim simulators

- as required software 3-4

## O

online help

- where to find it 4-1
- for use with FPGA implementations 4-1

OS platform. *See* HDL Verifier™ product

- requirements page on the MathWorks Web site

## P

platform support

- required 3-4

prerequisites

- for using HDL Verifier™ software 3-1

## R

requirements

- application software 3-4
- checking product 3-4
- platform 3-4

## S

server, MATLAB

- for MATLAB and HDL simulator links 1-3
- for Simulink and HDL simulator links 1-3

shared memory communication 1-8

Simulink

- as optional software 3-4
- in HDL Verifier™ environment 1-3
- installing 2-3
- working with HDL simulator links to 1-3

Simulink Fixed Point

- as optional software 3-4

sockets 1-8

- See also* TCP/IP socket communication

software

- installing HDL Verifier™ 2-2
- installing related application software 2-3
- optional 3-4
- required 3-4

## T

TCP/IP networking protocol 1-8

- See also* TCP/IP socket communication

TCP/IP socket communication

- mode 1-8

To VCD File block

- uses of 1-3

tutorials 4-7

- for HDL Verifier™ 4-1

for use with FPGA implementations 4-1

for HDL Verifier™ software 3-1

## **U**

users