

EDA Simulator Link™

Getting Started Guide

R2011b

MATLAB®
& **SIMULINK®**

How to Contact MathWorks



www.mathworks.com Web
comp.soft-sys.matlab Newsgroup
www.mathworks.com/contact_TS.html Technical Support



suggest@mathworks.com Product enhancement suggestions
bugs@mathworks.com Bug reports
doc@mathworks.com Documentation error reports
service@mathworks.com Order status, license renewals, passcodes
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.

EDA Simulator Link™ Getting Started Guide

© COPYRIGHT 2003–2011 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 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 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)

Introduction

1

Product Overview	1-2
Product Description	1-2
Product Limitations	1-3
Using EDA Simulator Link with HDL Simulators	1-4
HDL Cosimulation with MATLAB or Simulink and the HDL Simulator	1-4
Communications for HDL Cosimulation	1-8
Hardware Description Language (HDL) Support	1-9
HDL Cosimulation Workflows Described in the User Guide	1-9
Using EDA Simulator Link with Virtual Platforms	1-10
Generating TLM Components for Use with Virtual Platform Development	1-10
Typical Users and Applications	1-11
Using EDA Simulator Link with FPGA Development	
Environment	1-12
FPGA-in-the-Loop Simulation	1-12
FPGA Automation	1-13

Installation

2

Installing the Link Software	2-2
Installing Related Application Software	2-3

Product Requirements

3

What You Need to Know	3-2
Required Products	3-4
Supported EDA Tools	3-4
System Requirements	3-6
Product Feature and Platform Support	3-7
Optional Application Software	3-9

Getting Help

4

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

Quick Start: The EDA Simulator Link Wizards

5

Quick Start: Using The Cosimulation Wizard	5-2
Overview to the Cosimulation Wizard	5-2
Creating a Function from Existing HDL Code	5-4
Creating a Block from Existing HDL Code	5-4
Using the Cosimulation Wizard	5-5
Performing Cosimulation	5-26

Quick Start: Using the FPGA-in-the-Loop Wizard 5-28
 Preparing to Use the FPGA-in-the-Loop (FIL) Wizard **5-28**
 Running the FIL Wizard **5-28**
 Performing FIL Simulation **5-33**

Index

Introduction

- “Product Overview” on page 1-2
- “Using EDA Simulator Link with HDL Simulators” on page 1-4
- “Using EDA Simulator Link with Virtual Platforms” on page 1-10
- “Using EDA Simulator Link with FPGA Development Environment” on page 1-12

Product Overview

In this section...
“Product Description” on page 1-2
“Product Limitations” on page 1-3

Product Description

EDA Simulator Link™ provides a verification interface between MATLAB® or Simulink® and your HDL simulator or FPGA board. Using EDA Simulator Link you can verify a VHDL® or Verilog® design against your Simulink model or MATLAB algorithm using cosimulation with a Verilog or VHDL simulator, such as Mentor Graphics® ModelSim® or Cadence Incisive®. EDA Simulator Link also lets you perform hardware verification on your FPGA board using FPGA-in-the-loop simulation.

With EDA Simulator Link you can use MATLAB code and Simulink models as a test bench that generates stimulus for an HDL design and analyzes the simulation’s response. You can replace HDL design components with MATLAB code and Simulink models, enabling simulation of the complete system before all the HDL design elements are available.

EDA Simulator Link lets you create transaction-level model (TLM) components for use in virtual prototyping environments.

With EDA Simulator Link, you can also integrate MATLAB and Simulink with hardware design flows. The link software supports verification of FPGA and ASIC hardware designs by providing a cosimulation interface to HDL simulators, virtual platform development by generating a SystemC® Transaction Level Model (TLM 2.0) component and standalone test bench and FPGA deployment of automatically generated HDL by creating and managing Xilinx® ISE projects.

HDL Cosimulation with EDA Simulator Link

EDA Simulator Link 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.

EDA Simulator Link enables interactive and batch-mode cosimulation on a single computer, across heterogeneous platforms, or across a network.

The HDL Cosimulation Wizard. EDA Simulator Link contains the Cosimulation Wizard feature, which uses existing HDL code to create a customized MATLAB function (test bench or component) or a Simulink HDL Cosimulation block.

For more information, see “HDL Cosimulation Wizard”.

TLM Generation with EDA Simulator Link

EDA Simulator Link 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.

FPGA Development with EDA Simulator Link

EDA Simulator Link works with Simulink and Simulink® 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. EDA Simulator Link creates and manages your Xilinx ISE project and integrates a clock module with your design in an automatically generated top level module.

Product Limitations

Compatibility with Simulink Code Generation:

- Simulink HDL Coder: The EDA Simulator Link HDL Cosimulation block does participate in code generation with Simulink® HDL Coder™."
- Simulink® Coder™: The EDA Simulator Link HDL Cosimulation block does not participate in code generation with Simulink Coder for C code generation.

Using EDA Simulator Link with HDL Simulators

In this section...
“HDL Cosimulation with MATLAB or Simulink and the HDL Simulator” on page 1-4
“Communications for HDL Cosimulation” on page 1-8
“Hardware Description Language (HDL) Support” on page 1-9
“HDL Cosimulation Workflows Described in the User Guide” on page 1-9

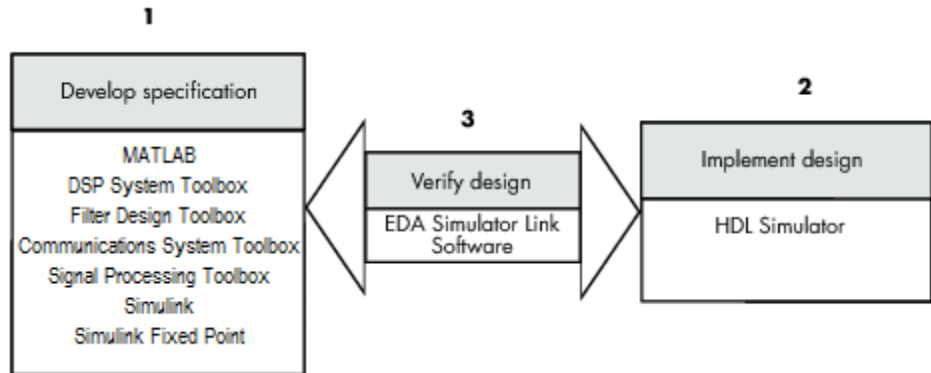
HDL Cosimulation with MATLAB or Simulink and the HDL Simulator

The EDA Simulator Link software consists of MATLAB functions that establish communication links between the HDL simulator and MATLAB and a library of Simulink blocks that you may use to include HDL simulator designs in Simulink models for cosimulation.

EDA Simulator Link 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, EDA Simulator Link 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.

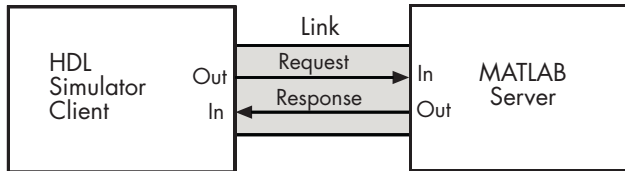
In addition to the preceding design scenario, EDA Simulator Link software enables you to work with tools in the following ways:

- 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 EDA Simulator Link 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.

Linking with MATLAB and the HDL Simulator

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

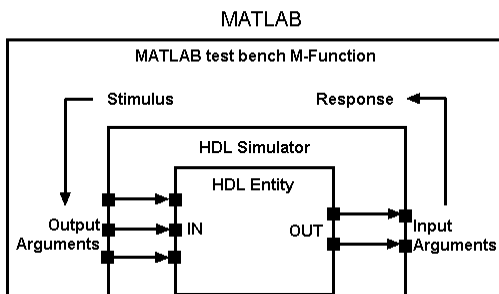


In this scenario, a MATLAB server function waits for service requests that it receives from an HDL simulator session. After receiving a request, the server establishes a communication link and invokes a specified MATLAB function that computes data for, verifies, or visualizes the HDL module (coded in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

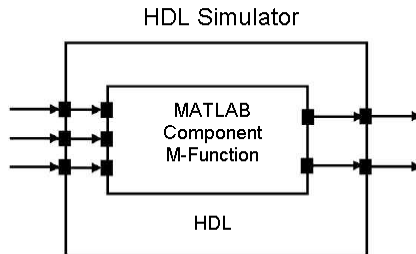
After the server is running, you can start and configure the HDL simulator or use with MATLAB with the supplied EDA Simulator Link function:

- `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.

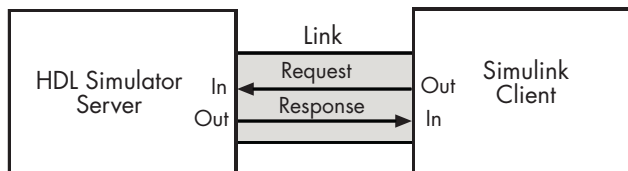


When you begin a specific test bench or component session, you specify parameters that identify the following information:

- The mode and, if appropriate, TCP/IP data necessary 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.

EDA Simulator Link 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`.

EDA Simulator Link 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

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

EDA Simulator Link 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 EDA Simulator Link User Guide provides instruction for using the link 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

See “Learning More About the EDA Simulator Link Software” for more information about the EDA Simulator Link documentation.

Using EDA Simulator Link with Virtual Platforms

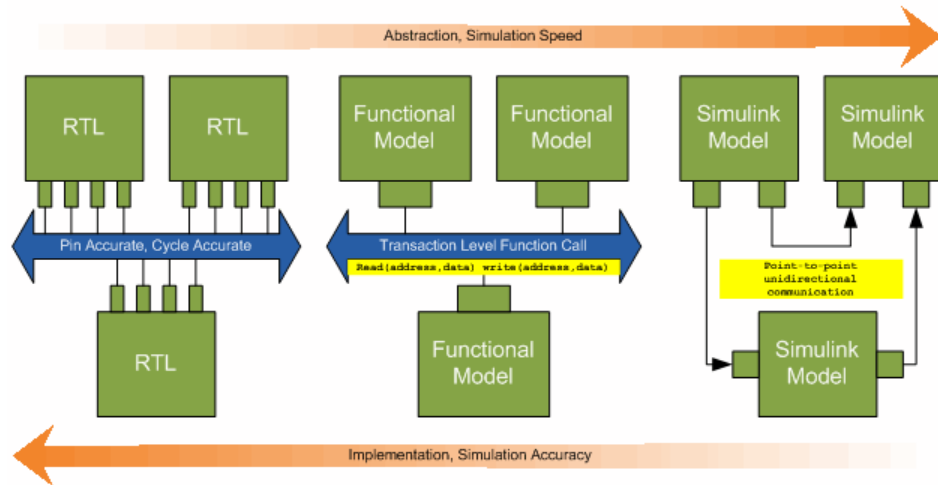
In this section...
“Generating TLM Components for Use with Virtual Platform Development” on page 1-10
“Typical Users and Applications” on page 1-11

Generating TLM Components for Use with Virtual Platform Development

When used with virtual platforms, EDA Simulator Link 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 EDA Simulator Link 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.

Using EDA Simulator Link with FPGA Development Environment

In this section...
“FPGA-in-the-Loop Simulation” on page 1-12
“FPGA Automation” on page 1-13

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 EDA Simulator Link FIL Wizard.
- Use the Simulink HDL Coder Workflow Advisor (see Simulink HDL Coder for instruction).

The FIL Wizard uses any synthesizable HDL code including code automatically generated from Simulink models by Simulink HDL Coder software. When you use FIL in the Workflow Advisor, Simulink 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 appropriate 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 EDA Simulator Link product page for a list of currently supported devices and boards.

EDA Simulator Link supports the use of a FIL block in a model reference block.

FPGA Automation

Note FPGA Automation will be removed as an option with Simulink Configuration Parameters in a future release. Use either Simulink HDL Coder or Filter Design HDL Coder as described in this section.

Create, update, and manage FPGA projects with MATLAB and Simulink for use with Xilinx ISE.

EDA Simulator Link provides a Project Generator for generating HDL code from either:

- Simulink models using Simulink HDL Coder
- MATLAB code using Filter Design HDL Coder

EDA Simulator Link 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 EDA Simulator Link 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.
- Request automatic generation of a Xilinx digital clock manager (DCM) for HDL code generated by Simulink HDL Coder for implementation in FPGA devices.

FPGA Automation Workflows Described in the User Guide

The EDA Simulator Link User Guide provides instruction for using the link software with supported FPGA development environments for the following workflows:

- Creating a new FPGA project
- Adding generated files to an existing FPGA project
- Updating generated files in associated FPGA project (Simulink only)
- Removing FPGA project associations (Simulink only)
- Generating Tcl scripts for project generation

See “FPGA Automation with Filter Design HDL Coder” and “FPGA Automation with Simulink HDL Coder and Simulink Configuration Parameters”.

Installation

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

Installing the Link Software

For details on how to install the EDA Simulator Link software, see the MATLAB installation instructions.

Installing Related Application Software

Based on your configuration decisions and the software required for your EDA Simulator Link 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 EDA Simulator Link software assumes users have a moderate level of prerequisite knowledge in the following subject areas.

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 needed for applying the Simulink component of the product

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 may be helpful.

For FPGA Automation

- FPGA design and implementation
- VHDL and/or Verilog
- Simulink and Simulink 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)

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

Additional Useful Experience

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

Required Products

In this section...
“Supported EDA Tools” on page 3-4
“System Requirements” on page 3-6
“Product Feature and Platform Support” on page 3-7
“Optional Application Software” on page 3-9

Supported EDA Tools

Cadence Incisive Usage Requirements

MATLAB and Simulink support Cadence verification tools using EDA Simulator Link.

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

Platform-Specific Software Requirements for HDL Cosimulation. The EDA Simulator Link shared libraries (`liblfihdls*.so`, `liblfihdlc*.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 EDA Simulator Link.

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

- ModelSim® SE 10.0a, 6.6d, 6.5f
- ModelSim PE 10.0a, 6.6d, 6.5f
- ModelSim DE 10.0a (Windows® 32 only)
- Questa® 10.0a

Platform-Specific Software Requirements for HDL Cosimulation. The Linux® platform requires that EDA Simulator Link software `rungcc 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
```

Xilinx ISE Usage Requirements

MATLAB and Simulink support Xilinx design tools using EDA Simulator Link.

- 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 Linux distribution compatibility of ISE tools.

Supported FPGA Devices for FIL Simulation. EDA Simulator Link 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

Supported FPGA Device Families for Clock Module Generation. For project generation, 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 [EDA Simulator Link 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	MATLAB	Fixed-Point Toolbox™, Signal Processing 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
FPGA-in-the-Loop	Simulink, Simulink Fixed Point, and Fixed-Point Toolbox	Simulink 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

Product Feature	Required Products	Recommended Products	Supported Platforms
FPGA Automation for Simulink	Simulink HDL Coder		Windows 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 ensure the most robust development environment for your application consider adding the following MathWorks products to your EDA Simulator Link 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 EDA Simulator Link 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 Simulink 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 EDA Simulator Link software.
“Demos and Tutorials” on page 4-7	Provides examples of how you would use EDA Simulator Link software.

Online Help

Online Help in the MATLAB Help Browser

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

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

```
>>doc edalink
```


Help for EDA Simulator Link 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

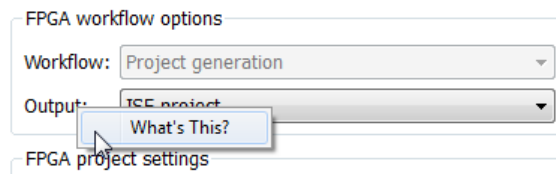
pages by clicking **Help** on any block dialog box.

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

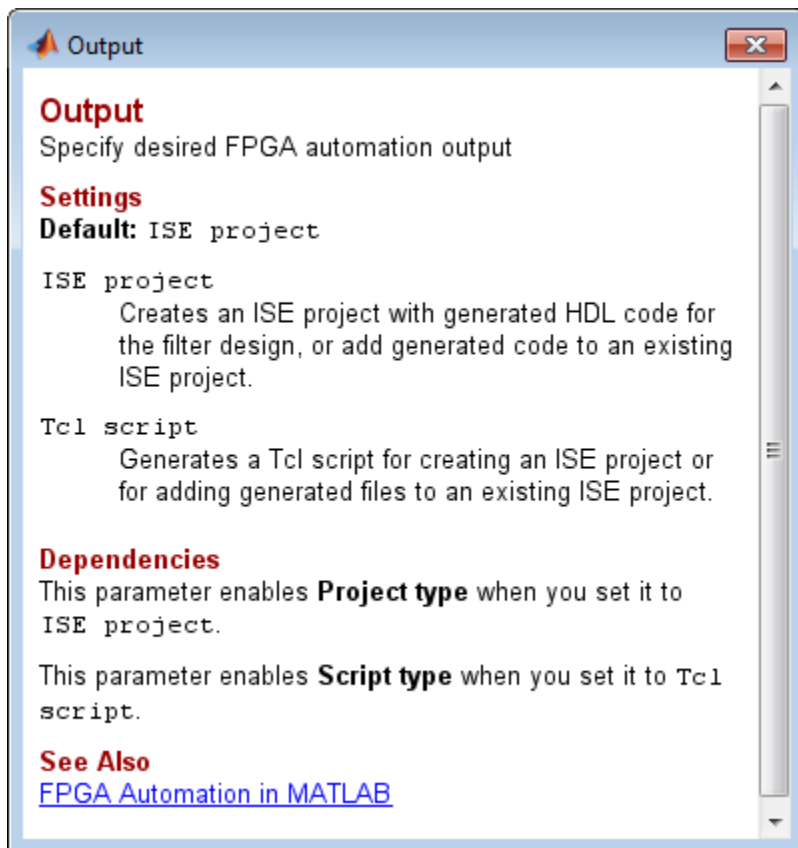
“What’s This?” context-sensitive help (CSH) topics are available for each option in the EDA Simulator Link GUIs. Use CSH as needed while using the GUIs to configure options for TLM generation or FPGA project management.

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 EDA Simulator Link demos with the online documentation. To access demos, type at the MATLAB command prompt:

```
>> demos
```

Select **EDA Simulator Link > 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 EDA Simulator Link software with HDL simulators and Simulink or MATLAB.

- “Create MATLAB Test Bench to Verify HDL Design with the Cosimulation Wizard (Tutorial)”
- “Create Simulink Test Bench to Verify HDL Design with the Cosimulation Wizard (Tutorial)”
- “Verify HDL Model with MATLAB Testbench (Tutorial)”
- “Verify HDL Model with Simulink Test Bench (Tutorial)”
- “Visually Comparing Simulink Signals with HDL Signals (Tutorial)”

Quick Start: The EDA Simulator Link Wizards

- “Quick Start: Using The Cosimulation Wizard” on page 5-2
- “Quick Start: Using the FPGA-in-the-Loop Wizard” on page 5-28

Quick Start: Using The Cosimulation Wizard

In this section...
“Overview to the Cosimulation Wizard” on page 5-2
“Creating a Function from Existing HDL Code” on page 5-4
“Creating a Block from Existing HDL Code” on page 5-4
“Using the Cosimulation Wizard” on page 5-5
“Performing Cosimulation” on page 5-26

Overview to the Cosimulation Wizard

- “Description” on page 5-2
- “Limitations” on page 5-3

Description

The EDA Simulator Link Cosimulation Wizard lets you take existing HDL code—from any source—and use that code to create a function or block. You can create a MATLAB component or test bench function or a Simulink HDL Cosimulation block for use in cosimulation.

Before you use the Cosimulation Wizard, you must understand the following about your HDL code:

- The name of the HDL files or compilation scripts to use in creating the block or function
- For Simulink cosimulation:
 - The name of the top module to be used for cosimulation
 - Output port types and sample times
 - Whether there are clocks and resets and which of them you want to use; what timing parameters are required, if any
 - Timescale
- For MATLAB cosimulation:

- Whether you want to create a component function, test bench function, or both
- How you want to trigger the callback (rising or falling edge, repeat, sensitivity)

The Cosimulation Wizard guides you in specifying the type of cosimulation, the HDL files, port and clock information required. The wizard also prompts you to enter other parameters necessary to build the block or function.

Every selection page includes a status window and navigational options:

- The status window displays the current status of the options you have selected. Warnings also display in this window.
- Clicking **Help** opens this documentation page.
- Clicking **Cancel** exits you from the Cosimulation Wizard without creating a block or function.
- Clicking **Back** and **Next** lets you navigate forward or backward through the application.

Note You can only move forward in the Cosimulation Wizard after you provide all necessary information for that step.

The last step of the Cosimulation Wizard generates the scripts (for functions) or blocks and launches the HDL simulator.

Limitations

- When creating an HDL Cosimulation block for use with Simulink, you can only access the I/O ports on the top level of the HDL design. If you want to cosimulate at multiple levels of your design, you cannot use this application to set up your HDL Cosimulation block.
- Because you can only access the top level of the HDL design, you cannot:
 - Create multiple HDL Cosimulation blocks
 - Use multiple generated HDL Cosimulation blocks in the same model

There is no need for additional blocks.

Creating a Function from Existing HDL Code

The workflow for the Cosimulation Wizard when creating a function is as follows:

- 1 Start the Cosimulation Wizard.
- 2 Select **MATLAB** for the HDL Cosimulation type, and specify where the HDL simulator executables reside
- 3 Select the HDL files to use in creating the block or function.
- 4 Specify any commands you want for HDL compilation.
- 5 Select the HDL module for cosimulation.
- 6 Specify any callback schedule parameters.
- 7 Generate scripts
- 8 Complete the component or test bench function (from template).
- 9 Run cosimulation.

Creating a Block from Existing HDL Code

The workflow for the Cosimulation Wizard when creating a block is as follows:

- 1 Start the Cosimulation Wizard.
- 2 Select **SIMULINK** for the HDL Cosimulation type, and specify where the HDL simulator executables reside.
- 3 Select HDL files to use in creating block or function.
- 4 Specify commands for HDL compilation.
- 5 Select HDL module for cosimulation.
- 6 Configure Simulink ports.

- 7 Provide output port details.
- 8 Provide clock and reset details.
- 9 Confirm or change start-time alignment.
- 10 Generate block.
- 11 Complete Simulink model.
- 12 Run cosimulation.

Using the Cosimulation Wizard

- “Common Workflows” on page 5-5
- “MATLAB Workflow” on page 5-14
- “Simulink Workflow” on page 5-18

Common Workflows

Invoke Cosimulation Wizard.

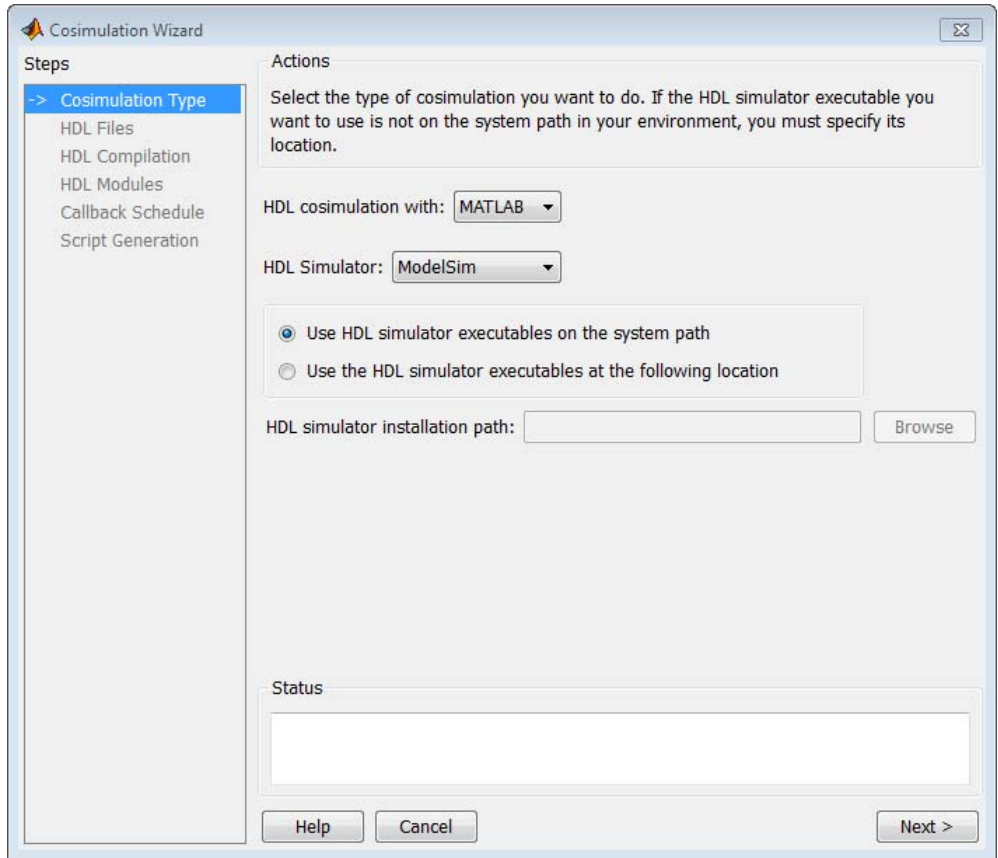
- 1 Start MATLAB.
- 2 Enter the following command at the command prompt:

```
>> cosimWizard
```

The Cosimulation Wizard opens.

Continue with the next task, “Select Cosimulation Type” on page 5-6.

Select Cosimulation Type.



On the **Cosimulation Type** page, you select options that determine whether a function or block is to be generated and for which HDL simulator.

1 HDL cosimulation with

Select either MATLAB or Simulink.

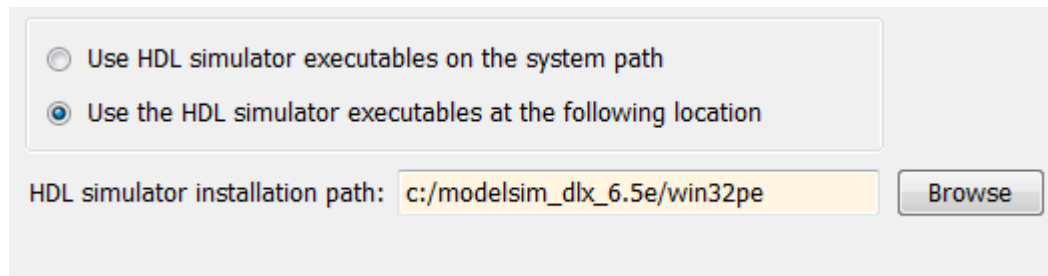
- If you choose MATLAB, the Cosimulation Wizard creates a MATLAB function template.

- If you select Simulink, the Cosimulation Wizard creates an HDL Cosimulation block, and uses 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. By default, the Cosimulation Wizard assumes that the executables appear on the system path. If they do not, select **Use the following HDL simulator executables at the following location**, and then specify the location in the text box below, as shown in the following image.

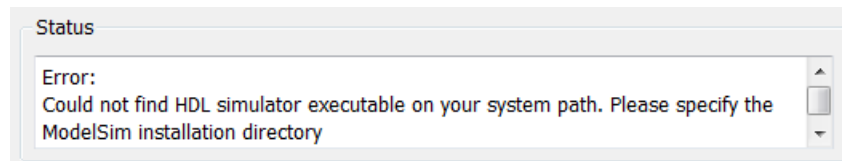


Use HDL simulator executables on the system path
 Use the HDL simulator executables at the following location

HDL simulator installation path:

If you click **Next** and the EDA Simulator Link Cosimulation Wizard does not find the executables, the following sequence of actions occurs:

- You return to the same dialog box, and the Cosimulation Wizard displays an error in the **Status** pane. The error you receive may look similar to the following.



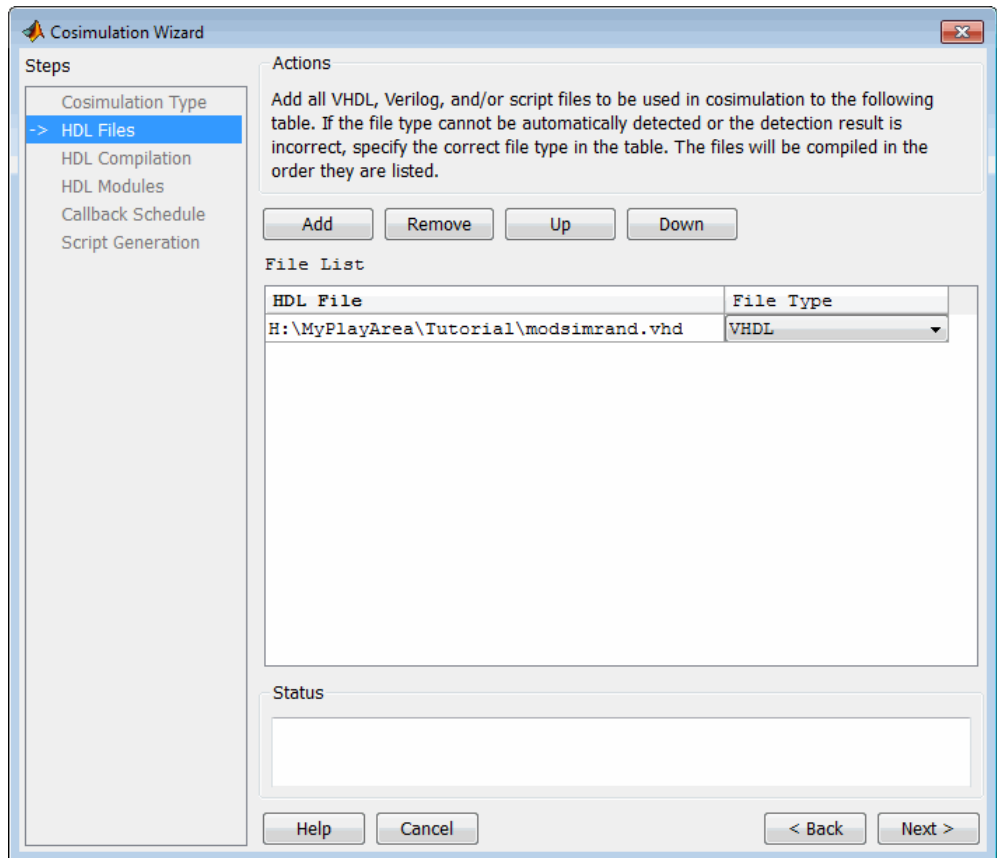
- The Cosimulation Wizard switches the option to **Use the HDL simulator executables at the following location**.

- The Cosimulation Wizard makes the **HDL simulation path** field editable.

You must enter a valid path to the HDL simulator executables before you can continue.

4 Click **Next** to proceed to “Select HDL Files to Import” on page 5-8.

Select HDL Files to Import.



On the **HDL File Selection** page, you tell the Cosimulation Wizard what files are to be used in creating the function or block.

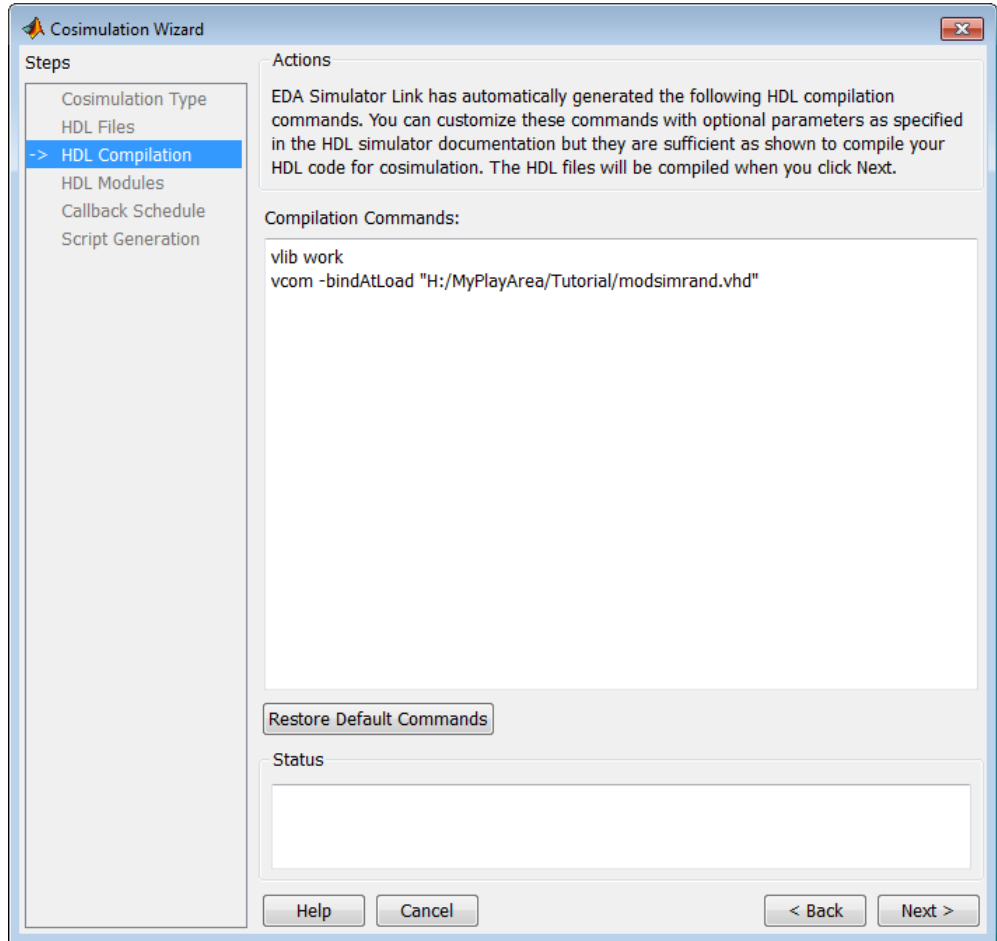
- 1** Add files by clicking **Add**. The Cosimulation Wizard prompts you to enter one or more file names.

The Cosimulation Wizard then attempts to determine the file type of each file and displays the type in the **File List** next to the file name. If the Cosimulation Wizard cannot determine the type or displays the wrong type, you can enter the correct type directly in the **File Type** column.

If you are using ModelSim software, you see compilation scripts listed as .do files (ModelSim macro file). If you are using Incisive® software, you see compilation scripts listed as shell scripts.

- 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** to proceed to “Specify HDL Compilation Commands” on page 5-10.

Specify HDL Compilation Commands.



On the **HDL Compilation** page, you can review the automatically generated HDL compilation commands. If necessary, override those commands, customize them, or do both. If you included compilation scripts instead of HDL files, this page displays the command to run the scripts.

- 1 Enter any changes to the commands in the **Compilation Commands** box.

Each command executes individually in the operating system. Some system commands may not affect subsequent commands. For example:

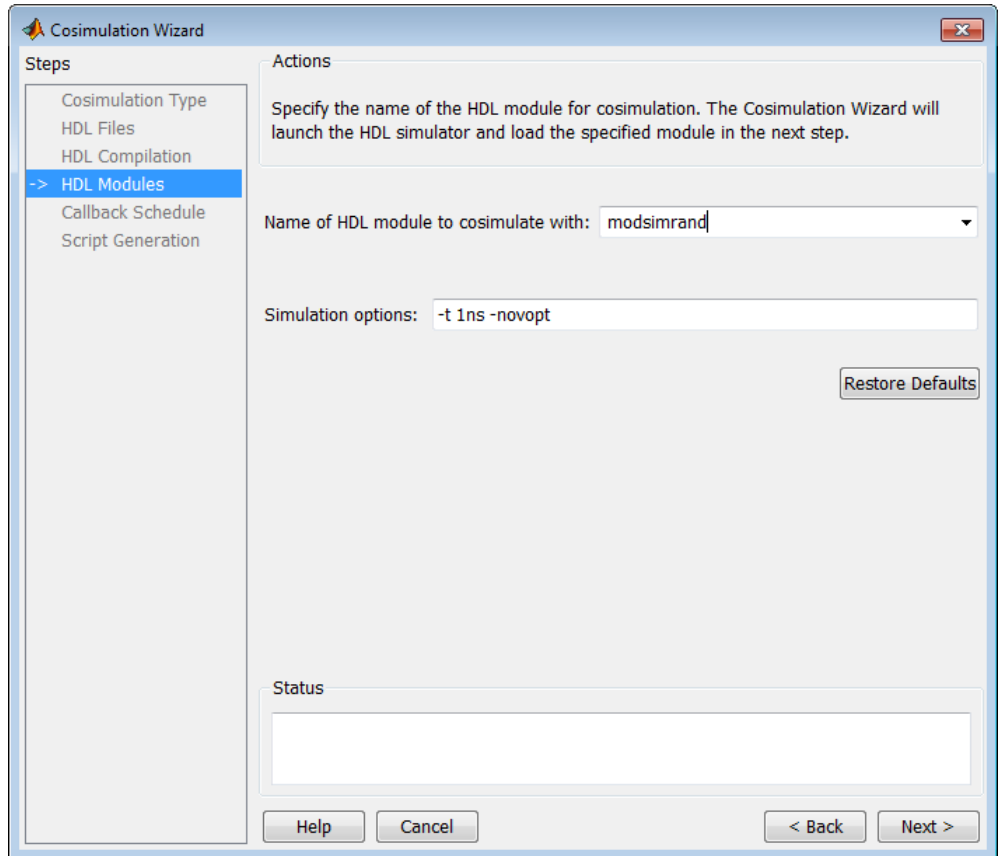
```
set file = a.vhd
vcom $file
```

These commands do not execute because the variable no longer holds the value that was set after you return to the Cosimulation Wizard from executing the first command. If you do try to include such a command, an error displays in the **Status** pane.

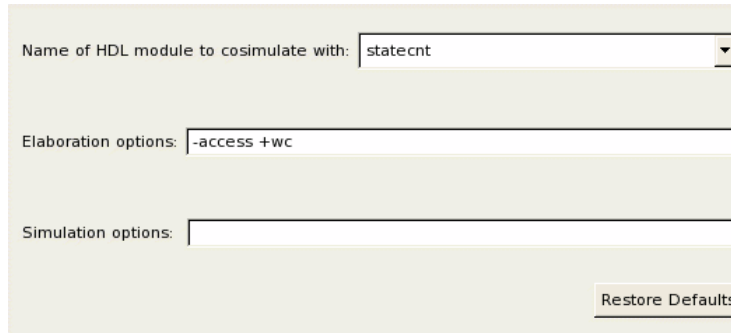
- 2** (Optional) You can click **Restore Default Commands** to go back to the automatically generated HDL compilation commands at any time. The Cosimulation Wizard prompts you to confirm that you do want to discard any changes.
- 3** Click **Next** to proceed to “Select HDL Modules for Cosimulation” on page 5-11.

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



Name of HDL module to cosimulate with: statecnt

Elaboration options: -access +wc

Simulation options:

Restore Defaults

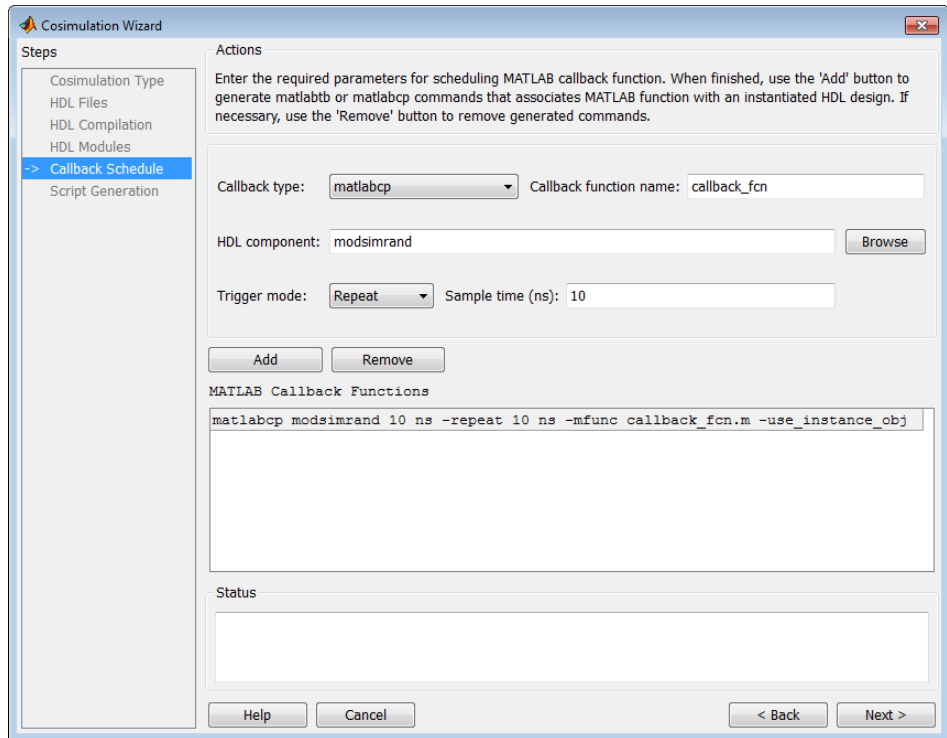
On the **HDL Module Selection** page, you provide the name of the HDL module to be used in cosimulation.

- 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** You can specify additional simulation options where you see **Simulation options**.
 - For example, the options provided by the Cosimulation Wizard specify the HDL simulator resolution and to turn off optimizations that remove signals from the simulation view.
 - If you change your mind about the options you've added or changed, you can click **Restore Defaults**.
- 4** Click **Next** to proceed to the next step:
 - If you select MATLAB as the cosimulation type, the next step in the MATLAB workflow is to “Specify Callback Schedule Parameters” on page 5-14.
 - If you selected Simulink as the cosimulation type, the next step in the Simulink workflow is to “Configure Simulink Ports” on page 5-19.

MATLAB Workflow

- “Specify Callback Schedule Parameters” on page 5-14
- “Generate Scripts” on page 5-16
- “Complete the Component or Test Bench Function” on page 5-17

Specify Callback Schedule Parameters.



- 1 On the **Callback Schedule** page, you can enter multiple component or test bench function callbacks from the HDL simulator.
 - a **Callback type:** Select `matlabcp` (create a component function) or `matlabtb` (create a test bench function).
 - b **Callback name:** The default name is `callback_fcn`. You may override this name and provide your own.

- c HDL component:** You can enter the component name manually or browse for it by clicking **Browse**.
 - d Trigger mode:** Specify one of the following to trigger the callback function:
 - Repeat
 - Rising Edge
 - Falling Edge
 - Sensitivity
 - e Sample time (ns) or Trigger Signal:** Enter a sample time or trigger signal:
 - If you select Repeat as a trigger, enter the sample time in nanoseconds.
 - If you select Rising Edge, Falling Edge, or Sensitivity, then **Sample time (ns)** changes to **Trigger Signal**. When **Trigger Signal** appears, you can enter the signal name to be used to trigger the callback.
 - You can browse the existing signals in the HDL component you specified by clicking **Browse**.
- 2** When you finish specifying the callback function parameters, click **Add** to add the command to the MATLAB Callback Functions list.

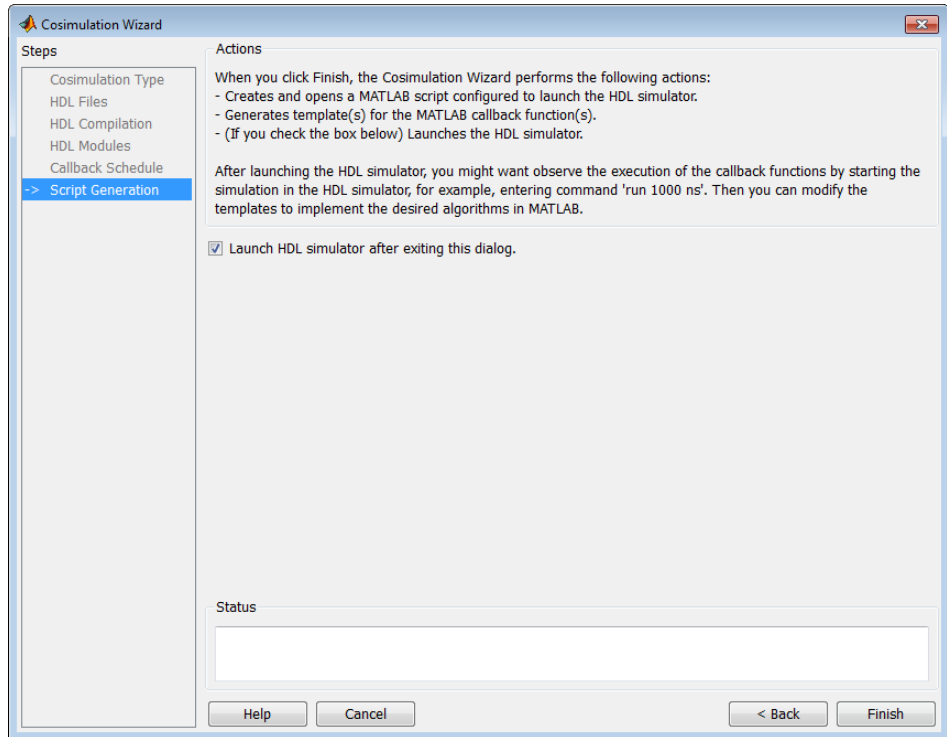
If you have more callback functions you want to schedule, repeat the preceding steps. If you want to remove any callback functions, highlight the line for that function and click **Remove**.

Note Attempting to add a callback function for the same HDL module as an existing callback function in the MATLAB Callback Functions list, results in an overwrite. The new callback function overwrites the existing one (even if you change the callback type)). A warning appears in the **Status** window:

Warning: This HDL component already has a scheduled callback function, which is replaced by this new one.

3 Click **Next** to proceed to “Generate Scripts” on page 5-16.

Generate Scripts.



The Cosimulation Wizard now has all the information it needs to generate an EDA Simulator Link component or test bench function.

- You may click **Back** to review or change your settings.
- When you are satisfied with all selections, click **Finish** to generate the scripts.

When you click **Finish**, EDA Simulator Link Cosimulation Wizard creates the following files and opens all of them in separate MATLAB editor windows:

- `launch_HDL_simulator_modulename`: Script for launching the HDL simulator for cosimulation with MATLAB
- `compile_HDL_Design_modulename`: Compilation script you can reuse for subsequent compilation of this particular component
- Function files (*.m): Component and test bench customized function templates, one for each component specified in the Cosimulation Wizard.

Complete the Component or Test Bench Function. The template, which is automatically created, contains some simple port I/O instructions and empty routines where you add your own code.

```
function callback_fcn(obj)
%
% MATLAB callback function template associated with HDL component(s):
% /inverter;
%
% File Name: callback_fcn.m
% Created: 30-Nov-2010 15:20:26
%
% Generated by Cosimulation Wizard

% --- Initialize internal state(s) of callback function ---
if (strcmp(obj.simstatus,'Init'))
    disp('Initializing states ...');
    obj.userdata.State = 0;
end

% Display obj.tnow, which is the current HDL simulation time specified in
% seconds.
disp(['Callback function is executed at time ' num2str(obj.tnow)]);

% --- Read signal from HDL component ---
% Variable obj.portvalues.PortName contains the input value of port with
% name 'PortName' on the associated HDL component. The list of readable
% ports can be determined from the fields in struct obj.portinfo.out and
% obj.portinfo.inout.
%
% If obj.portvalues.PortName is multi-valued logic vector, you can convert
```

```
% it to decimal using function mvl2dec, e.g.,
%   decValue = mvl2dec(obj.portvalues.PortName, true);
%
% Optionally, you can also translate the port value into fixed-point
% object, e.g.
%   myfiobj = fi(decValue,1, 16, 4);

% --- Write signal to HDL component ---
% The MATLAB data are passed to HDL component by assigning variable
% obj.portValues.PortName, where 'PortName' is a port on the associated HDL
% component. The list of writable ports can be determined from the fields
% in struct obj.portinfo.in and obj.portinfo.inout.
%
% If the datatype of obj.portvalues.PortName is multi-valued logic vector,
% you can convert decimal data in MATLAB to multi-valued logic vector
% through function dec2mvl, e.g.,
%   obj.portvalues.PortName = dec2mvl(obj.userdata.State, BitWidth);

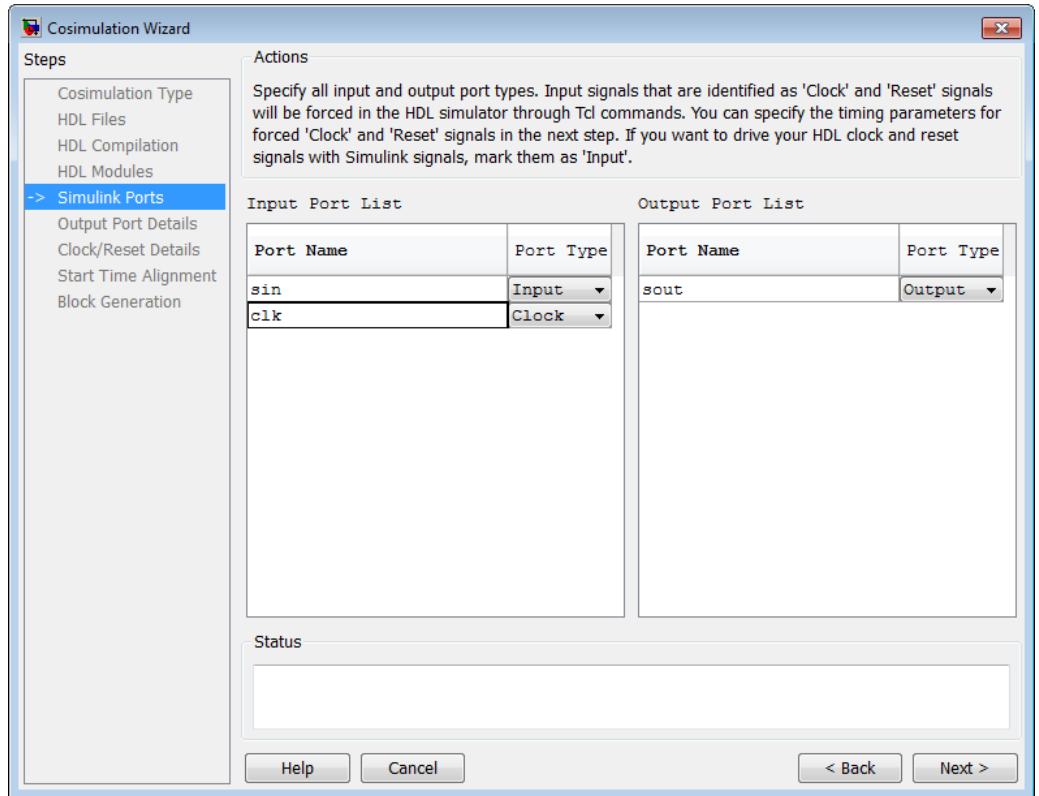
% --- Update internal state(s) ---
% Update your state(s). In the following example, we use this internal
% state to implement a one-bit counter.
disp(['Updated internal state: ' num2str(obj.userdata.State)]);
obj.userdata.State = ~obj.userdata.State;
```

If you need additional assistance, see “Performing Cosimulation” on page 5-26 or “Create MATLAB Test Bench to Verify HDL Design with the Cosimulation Wizard (Tutorial)”.

Simulink Workflow

- “Configure Simulink Ports” on page 5-19
- “Specify Output Port Details” on page 5-20
- “Specify Clock and Reset Details” on page 5-21
- “Confirm or Change Start Time Alignment” on page 5-23
- “Generate Block” on page 5-25
- “Complete the Simulink Model” on page 5-25

Configure Simulink Ports.



On the **Simulink Ports** page, you specify the type of each input and output port. The Cosimulation Wizard attempts to determine the port types for you, but you may override any setting.

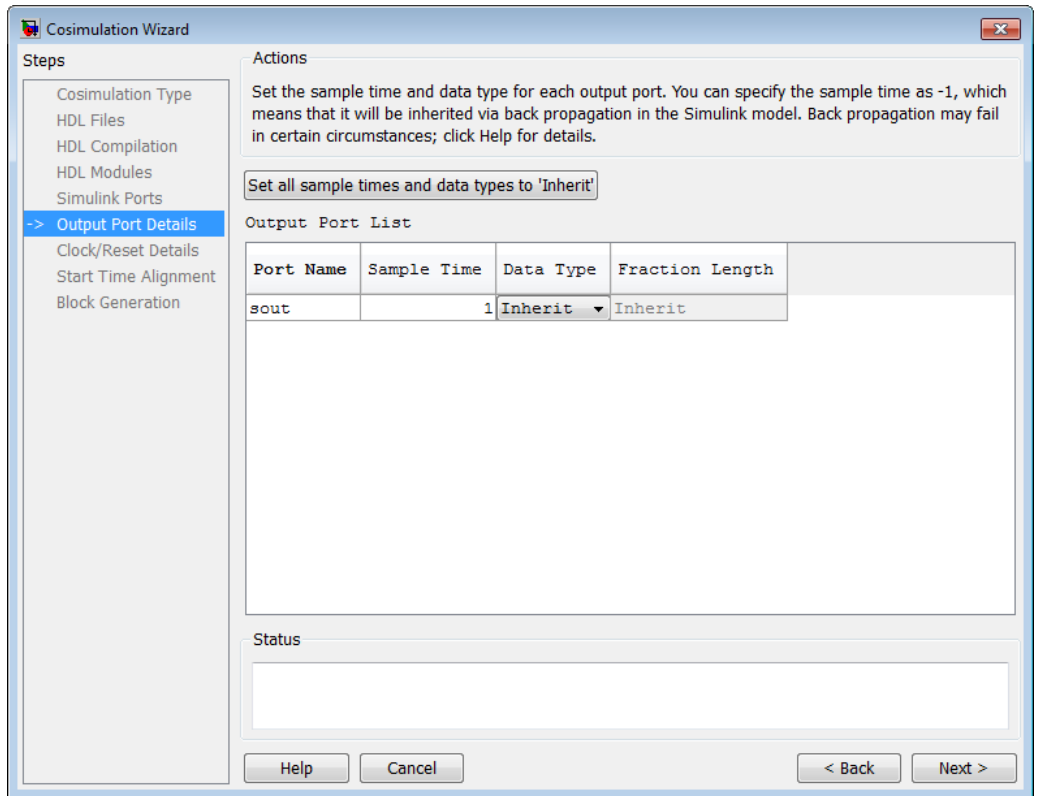
1 For **Port Type**, specify one of the following:

- Input
- Clock
- Reset
- Unused

Tcl commands force the clock and reset signals into the HDL simulator. You can specify clock and reset signal timing in a later step (see “Specify Clock and Reset Details” on page 5-21). To drive your HDL clock and reset signals with Simulink signals, mark them as Input.

2 Click **Next** to proceed to “Specify Output Port Details” on page 5-20.

Specify Output Port Details.



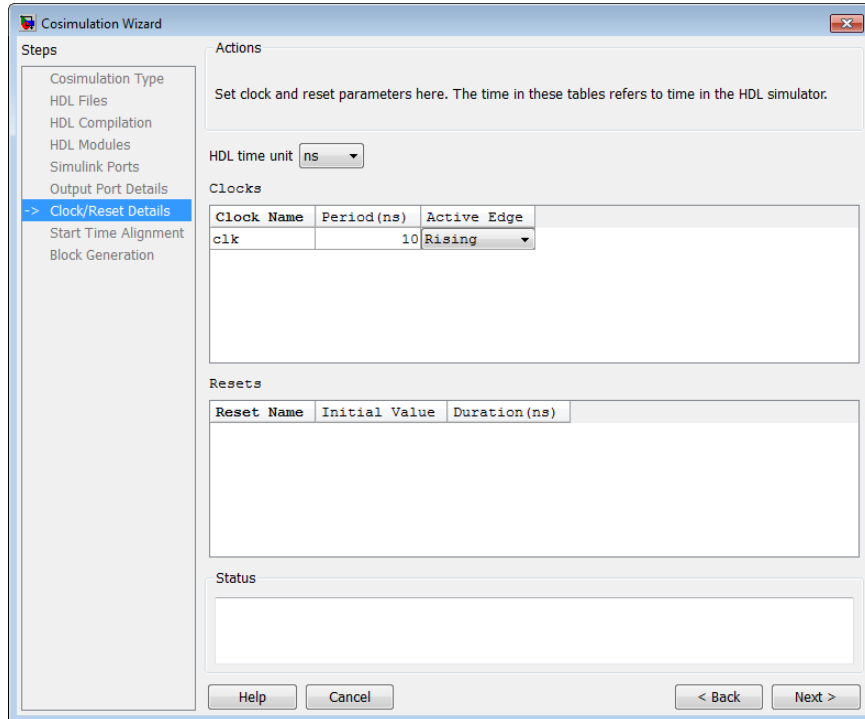
On the **Output Port Details** page, you set the sample time and data type for all output ports.

- 1 The default sample time is 1, and the default data type is `Inherit`. These settings are consistent with the way the HDL Cosimulation block mask (**Ports** tab) sets default settings for output ports.

If you click **Set all sample times and data types to "Inherit"**, the ports inherit the times via back propagation (sample times are set to `-1`), but back propagation may fail in some circumstances; see “Monitoring Backpropagation in Sample Times” in the *Simulink User’s Guide*.

- 2 Click **Next** to proceed to “Specify Clock and Reset Details” on page 5-21.

Specify Clock and Reset Details.



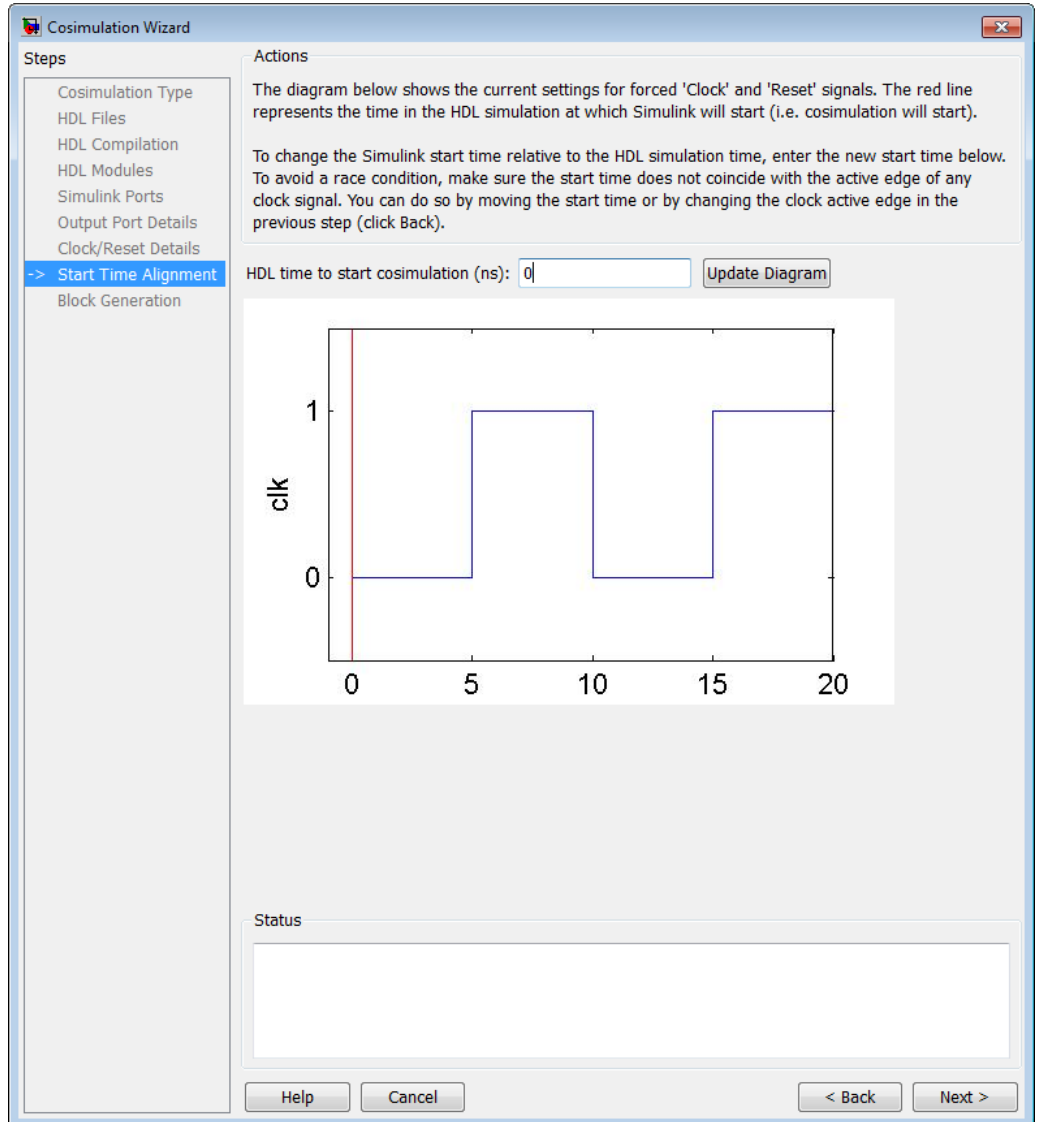
On the **Clock/Reset Details** page, you set the clock and reset parameters. The time period specified here refers to time in the HDL simulator.

- Default clock settings comprise a rising active edge and a period of 10 ns.
- Default reset settings comprise an initial value of 0 and a duration of 8 ns.

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

Click **Next** to proceed to “Confirm or Change Start Time Alignment” on page 5-23.

Confirm or Change Start Time Alignment.



On the **Start Time Alignment** page, you can review the current settings for clocks and resets. This dialog box has two purposes:

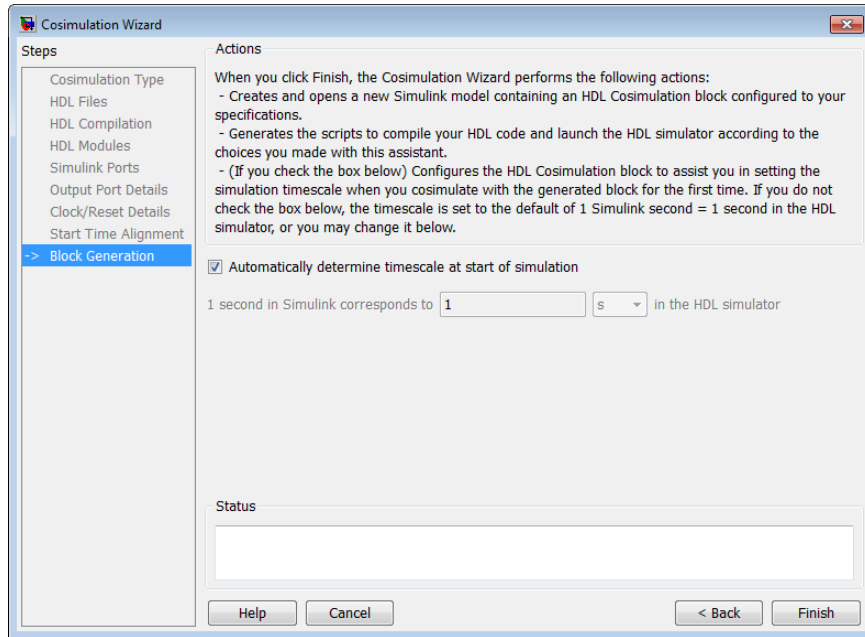
- To verify that the rising or falling edge is set properly (from the previous step). To do so:
 - Look at the start time. If the time coincides with the active edge of the clock, you need to adjust the HDL simulator start time or change the clock active edge.
 - Look at the reset signal. If the signal is synchronous and coincides with the clock active edge, you may have a possible race condition.

To avoid a race condition, make sure the start time does not coincide with the active edge of any clocks. You can do so by moving the start time or by changing clock active edges in the previous step.
- To make sure the start time is correct.

The Cosimulation Wizard calculates the HDL simulator start time from the clock and reset values on the previous page. You can change the HDL simulator start time by entering a new value where you see **HDL time to start cosimulation (ns)**. Click **Update Diagram** to see your change applied.

Click **Next** to proceed to “Generate Block” on page 5-25.

Generate Block.



Specify whether you want EDA Simulator Link software to automatically determine the timescale when you start the simulation or if you'd prefer to determine the timescale yourself. For more about timescales, see the “Timescales Pane” section in the HDL Cosimulation block reference.

The Cosimulation Wizard now has all the information required to generate an EDA Simulator Link HDL Cosimulation block.

- You may click **Back** to review or change your settings.
- After you are satisfied with all selections, click **Finish** to generate the block.

Complete the Simulink Model. Complete your Simulink model before you begin cosimulation. If you need additional assistance, see “Performing Cosimulation” on page 5-26 or “Create Simulink Test Bench to Verify HDL Design with the Cosimulation Wizard (Tutorial)”.

Performing Cosimulation

After you finish creating a function or block, select the appropriate 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"> • 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”. First, however, review the entire . • For the full process and description, see “Simulating an HDL Component in a MATLAB Test Bench Environment”.
Replace HDL component with a MATLAB function	<ul style="list-style-type: none"> • Because you already have a completed component function written, you can pick up at this step: “Place Component Function on MATLAB Search Path”. First, however, review the entire “Creating a MATLAB Component Function for Use with the HDL Simulator”. • For the full process and description, see “MATLAB Component to Represent HDL Design ”.

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

Simulink Cosimulation

Task to Perform...	Read...
<p>Simulate HDL component with Simulink test bench</p>	<ul style="list-style-type: none"> • 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”. • For the full process and description, see “Simulating an HDL Component in a Simulink Test Bench Environment”.
<p>Replace HDL component with Simulink algorithm</p>	<ul style="list-style-type: none"> • 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”. • For the full process and description, see “Replacing an HDL Component with a Simulink Algorithm”.

For additional help on EDA Simulator Link topics, see the “Information Overview” on page 4-2.

Quick Start: Using the FPGA-in-the-Loop Wizard

In this section...
“Preparing to Use the FPGA-in-the-Loop (FIL) Wizard” on page 5-28
“Running the FIL Wizard” on page 5-28
“Performing FIL Simulation” on page 5-33

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 the FPGA board. See .
- 3 Set up the gigabit Ethernet network adapter. See .
- 4 Set up your Xilinx tools. See

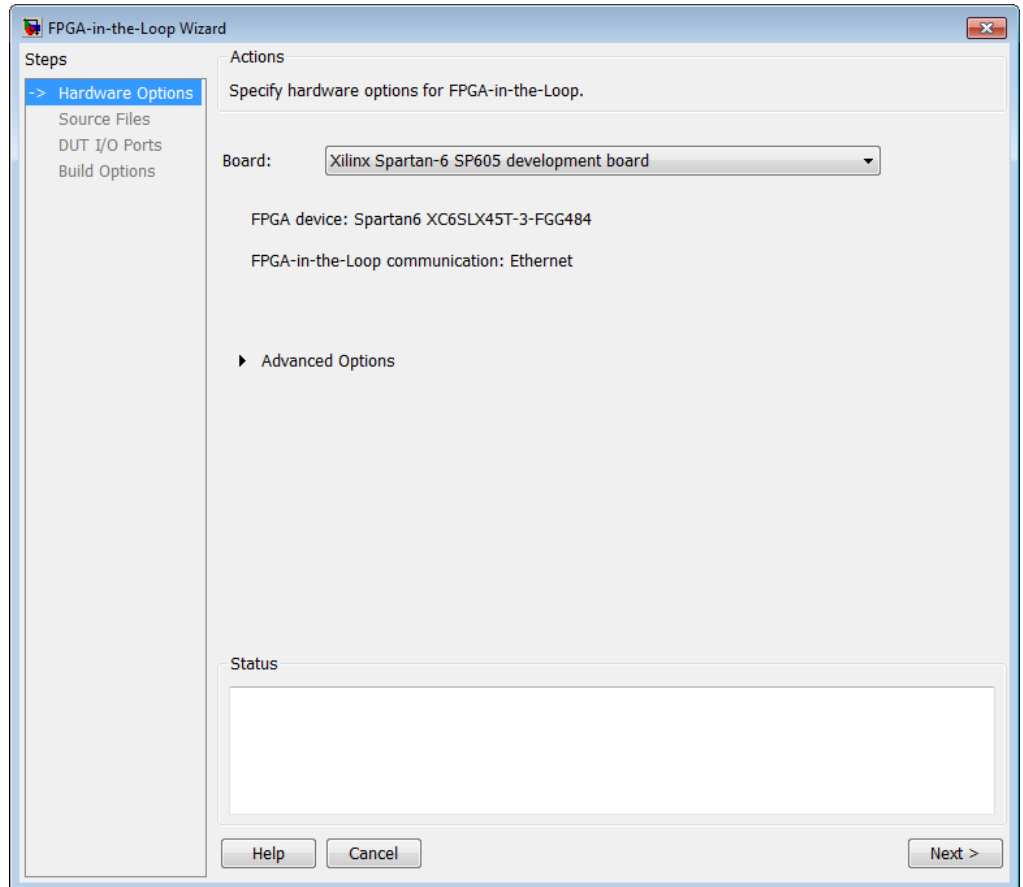
For more detailed information, see “FPGA-in-the-Loop (FIL)”. For a demonstration of FIL, see the FPGA-in-the-Loop demos under EDA Simulator Link.

Running the FIL Wizard

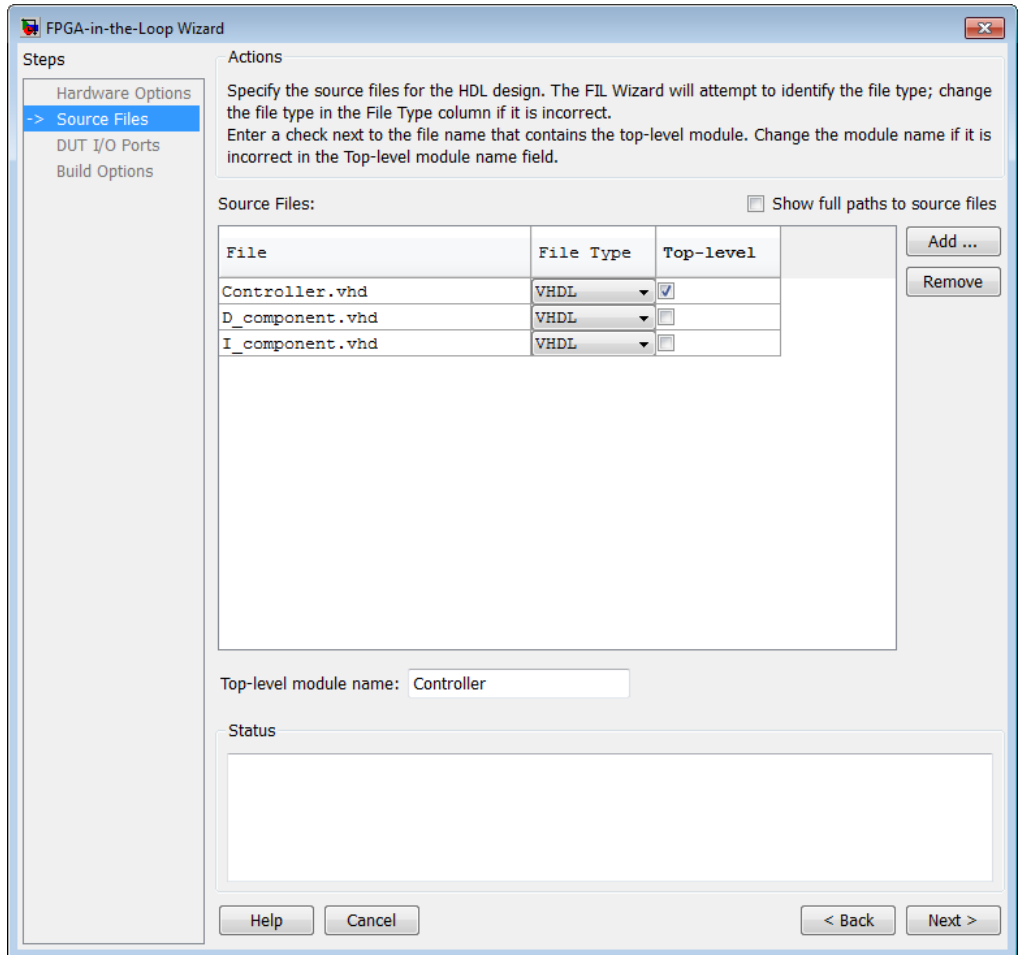
- 1 At the MATLAB prompt, enter the following:

```
>> filWizard
```

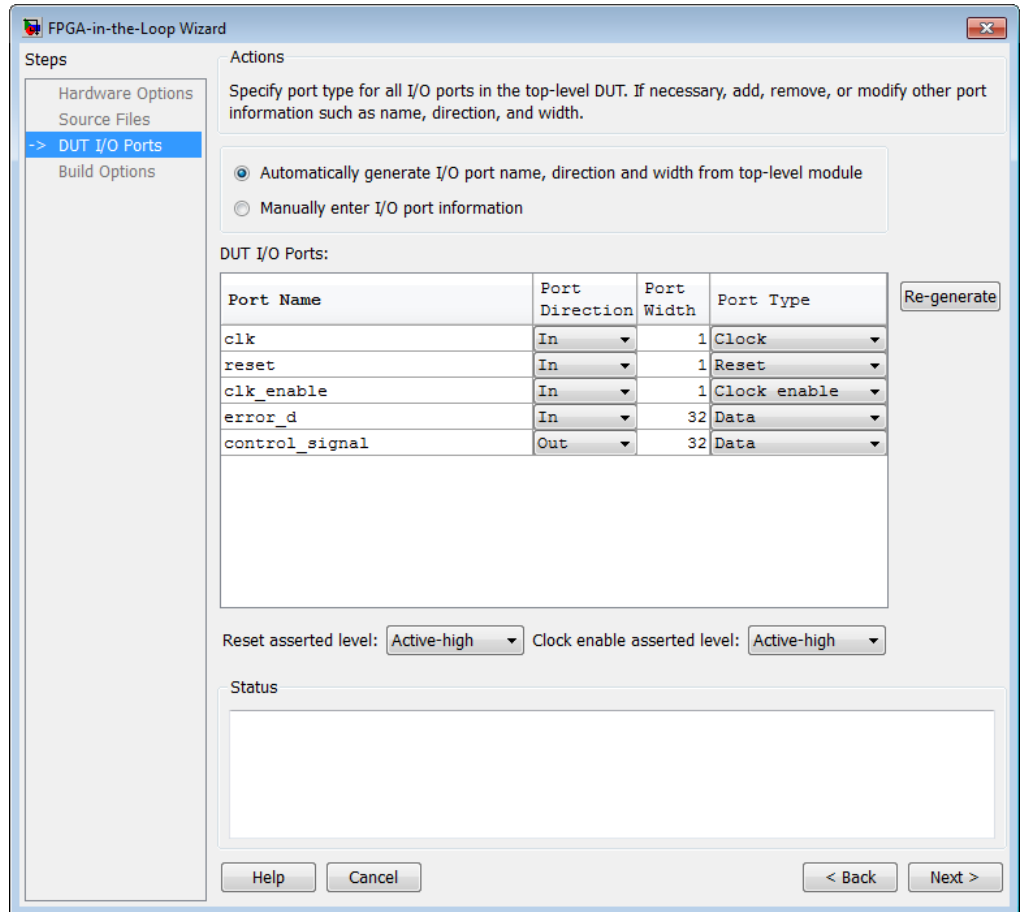
- 2 Select the board you are using. Adjust the Board IP address if necessary. Click **Next** to continue.



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

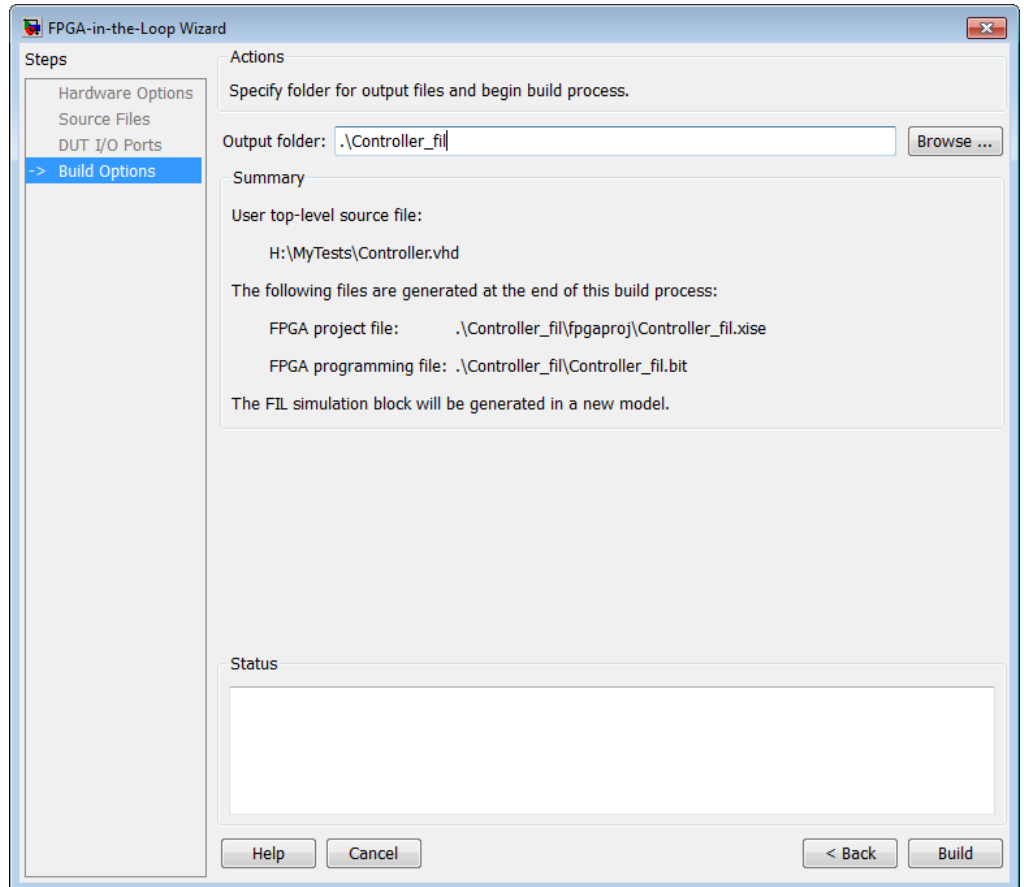


- 4 Review the DUT I/O ports. Correct any settings if necessary. Click **Next** to continue.



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

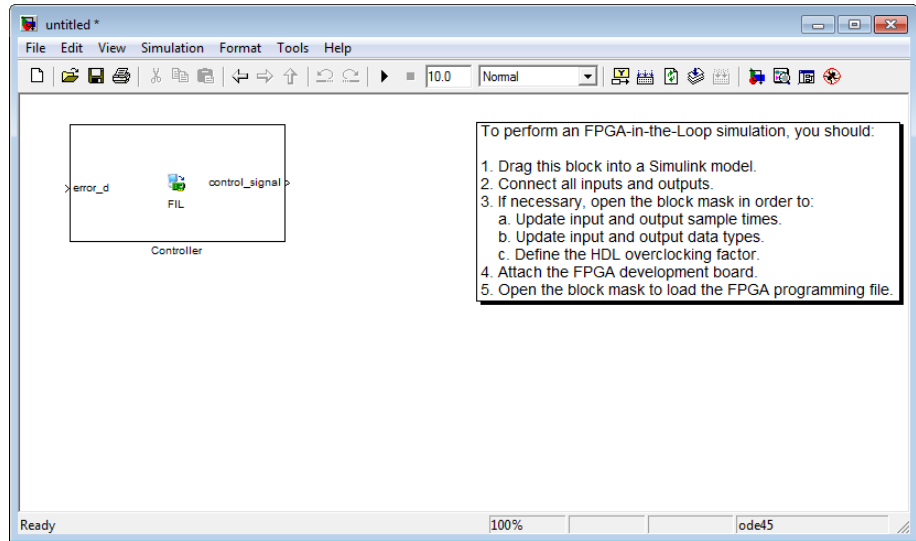
- 5 Select the output folder for the programming files. Click **Next** to continue.



6 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.

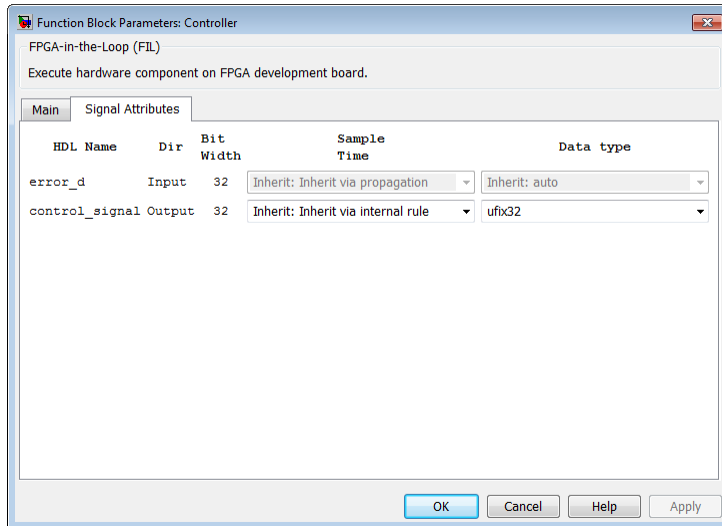
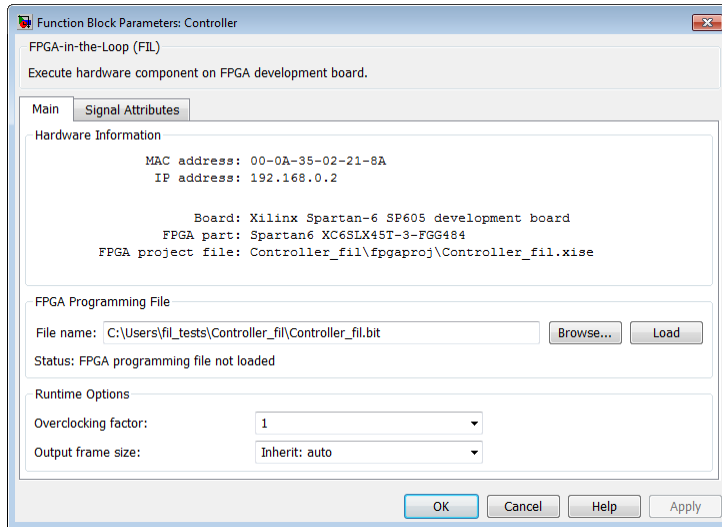


- After new model generation, the FIL Wizard opens a command window.
 - In this window, the Xilinx ISE design suite performs synthesis, fit, place-and-route, timing analysis, and FPGA programming file generation.
 - When the Xilinx ISE process is finished, a message in the command window lets you know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

For more detailed information about the FIL Wizard, see . 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 EDA Simulator Link.

Performing FIL Simulation

- 1** Insert the generated FIL block into the existing model.
- 2** Open the block mask and load the programming files.
- 3** Make any other adjustments on the block mask, if necessary.



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 EDA Simulator Link.

A

- application software 3-4
- application specific integrated circuits (ASICs) 1-1
- applications 1-2
- ASICs (application specific integrated circuits) 1-1

B

- behavioral model 1-2
- blocksets
 - installing 2-3

C

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

D

- demos 4-7
 - for EDA Simulator Link™ 4-1
 - for use with FPGA implementations 4-1
- design process, hardware 1-2
- 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
- EDA Simulator Link™ software
 - definition of 1-1
 - installing 2-2
- Electronic Design Automation (EDA) 1-1
- environment
 - cosimulation with MATLAB and HDL simulator 1-4
 - cosimulation with Simulink and HDL simulator 1-4

F

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

H

- hardware description language (HDL). *See* HDL
- hardware design process 1-2
- HDL (hardware description language) 1-1
- HDL Cosimulation block
 - in EDA Simulator Link™ environment 1-4
- HDL models 1-2
 - cosimulation 1-2
 - verifying 1-2
 - See also* VHDL models
- HDL simulators
 - in EDA Simulator Link™ environment 1-4
 - installing 2-3
 - working with Simulink links to 1-4
- help
 - for EDA Simulator Link™ software 4-1
 - for use with FPGA implementations 4-1

I

- Incisive
 - in EDA Simulator Link™ cosimulation environment 1-4
 - working with MATLAB links to 1-4
- Incisive or NC simulators
 - as required software 3-4
- installation
 - of EDA Simulator Link™ software 2-2
 - of related software 2-3

L

- links
 - MATLAB and HDL simulator 1-4
 - Simulink and HDL simulator 1-4

M

- MATLAB
 - as required software 3-4
 - in EDA Simulator Link™ cosimulation environment 1-4
 - installing 2-3
 - working with HDL simulator links to 1-4
- MATLAB functions
 - test bench 1-4
- MATLAB server
 - function for invoking 1-4
- ModelSim
 - in EDA Simulator Link™ cosimulation environment 1-4
 - working with MATLAB links to 1-4
- 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* EDA Simulator Link™ product requirements page on the MathWorks Web site

P

- platform support
 - required 3-4
- prerequisites
 - for using EDA Simulator Link™ 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 1-4
 - for Simulink and HDL simulator links 1-4
- shared memory communication 1-8
- Simulink
 - as optional software 3-4
 - in EDA Simulator Link™ environment 1-4
 - installing 2-3
 - working with HDL simulator links to 1-4
- Simulink Fixed Point
 - as optional software 3-4
- sockets 1-8
 - See also* TCP/IP socket communication
- software
 - installing EDA Simulator Link™ 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 1-4
- tutorials 4-7
 - for EDA Simulator Link™ 4-1
 - for use with FPGA implementations 4-1

U

- users
 - for EDA Simulator Link™ software 3-1

V

- VHDL models 1-2
 - See also* HDL models