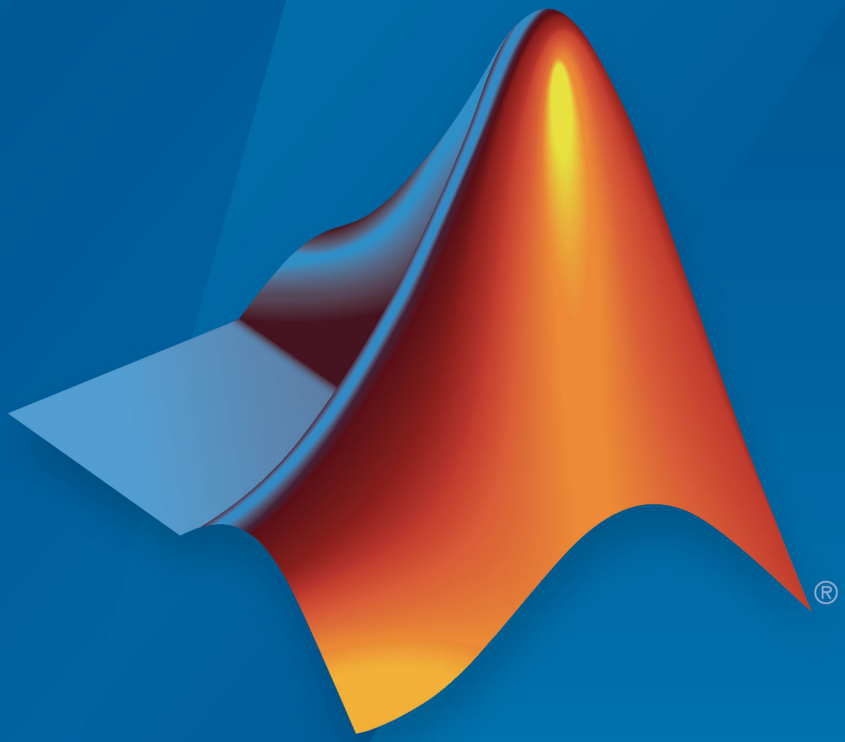
HDL Verifier™

Getting Started Guide



MATLAB® & SIMULINK®

R2017b

MathWorks®

How to Contact MathWorks

Latest news: www.mathworks.com

Sales and services: www.mathworks.com/sales_and_services

User community: www.mathworks.com/matlabcentral

Technical support: www.mathworks.com/support/contact_us

Phone: 508-647-7000

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

HDL Verifier™ Getting Started Guide

Revision History

August 2003	Online only	New for Version 1 (Release 13SP1)
February 2004	Online only	Revised for Version 1.1 (Release 13SP1)
June 2004	Online only	Revised for Version 1.1.1 (Release 14)
October 2004	Online only	Revised for Version 1.2 (Release 14SP1)
December 2004	Online only	Revised for Version 1.3 (Release 14SP1+)
March 2005	Online only	Revised for Version 1.3.1 (Release 14SP2)
September 2005	Online only	Revised for Version 1.4 (Release 14SP3)
March 2006	Online only	Revised for Version 2.0 (Release 2006a)
September 2006	Online only	Revised for Version 2.1 (Release 2006b)
March 2007	Online only	Revised for Version 2.2 (Release 2007a)
September 2007	Online only	Revised for Version 2.3 (Release 2007b)
March 2008	Online only	Revised for Version 2.4 (Release 2008a)
October 2008	Online only	Revised for Version 2.5 (Release 2008b)
March 2009	Online only	Revised for Version 2.6 (Release 2009a)
September 2009	Online only	Revised for Version 3.0 (Release 2009b)
March 2010	Online only	Revised for Version 3.1 (Release 2010a)
September 2010	Online only	Revised for Version 3.2 (Release 2010b)
April 2011	Online only	Revised for Version 3.3 (Release 2011a)
September 2011	Online only	Revised for Version 3.4 (Release 2011b)
March 2012	Online only	Revised for Version 4.0 (Release 2012a)
September 2012	Online only	Revised for Version 4.1 (Release 2012b)
March 2013	Online only	Revised for Version 4.2 (Release 2013a)
September 2013	Online only	Revised for Version 4.3 (Release 2013b)
March 2014	Online only	Revised for Version 4.4 (Release 2014a)
October 2014	Online only	Revised for Version 4.5 (Release 2014b)
March 2015	Online only	Revised for Version 4.6 (Release 2015a)
September 2015	Online only	Revised for Version 4.7 (Release 2015b)
March 2016	Online only	Revised for Version 5.0 (Release 2016a)
September 2016	Online only	Revised for Version 5.1 (Release 2016b)
March 2017	Online only	Revised for Version 5.2 (Release 2017a)
September 2017	Online only	Revised for Version 5.3 (Release 2017b)

Contents

1 Introduction

2 About HDL Verifier

3 Third-Party Product Requirements

1

Introduction

HDL Verifier Product Description

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

HDL Verifier automatically generates test benches for Verilog® and VHDL® design verification. You can use MATLAB® or Simulink® to directly stimulate your design and then analyze its response using HDL cosimulation or FPGA-in-the-loop with Xilinx® and Altera® FPGA boards. This approach eliminates the need to author standalone Verilog or VHDL test benches.

HDL Verifier also generates components that reuse MATLAB and Simulink models natively in simulators from Cadence®, Mentor Graphics®, and Synopsys®. These components can be used as verification checker models or as stimuli in more complex test-bench environments such as those that use the Universal Verification Methodology (UVM).

Key Features

- Cosimulation with Cadence Incisive®, Mentor Graphics ModelSim®, or Questa®
- FPGA-in-the-loop verification using Xilinx and Altera FPGA boards
- SystemVerilog DPI component generation from MATLAB functions and Simulink blocks
- Generation of IEEE® 1666 SystemC TLM 2.0 compatible transaction-level models
- Automated verification workflow with HDL Coder™

About HDL Verifier

HDL Cosimulation

In this section...

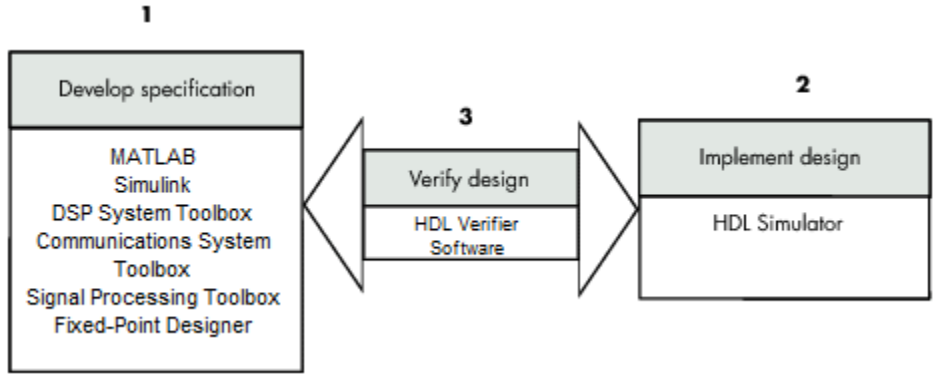
HDL Cosimulation with MATLAB or Simulink

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 the following 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

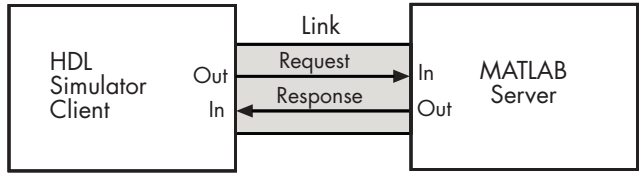
Note You can cosimulate a module using SystemVerilog, SystemC or both with MATLAB or Simulink using the HDL Verifier software. Write simple wrappers around the SystemC and make sure that the SystemVerilog cosimulation connections are to ports or signals of data types supported by the link cosimulation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

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

- "Linking with MATLAB and the HDL Simulator" on page 2-3
- "Linking with Simulink and the HDL Simulator" on page 2-5
- "The HDL Cosimulation Wizard" on page 2-6

Linking with MATLAB and the HDL Simulator

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

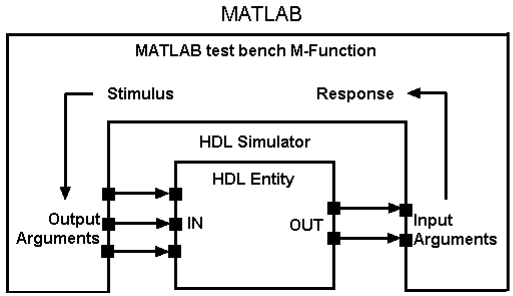


In this scenario, a MATLAB server function waits for service requests that it receives from an HDL simulator session. After receiving a request, the server establishes a communication link and invokes a specified MATLAB function that computes data for, verifies, or visualizes the HDL module (coded in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

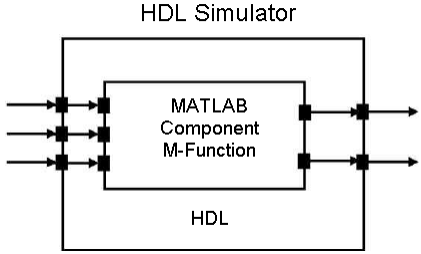
After the server is running, you can start and configure the HDL simulator or use with MATLAB with the supplied HDL Verifier function:

- `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.



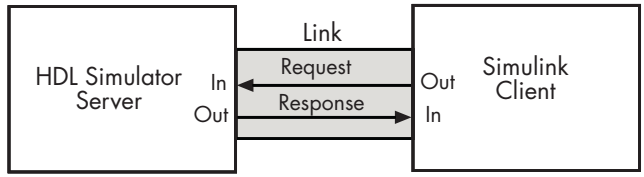
When you begin a specific test bench or component session, you specify parameters that identify the following information:

- 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 "Prepare to Import HDL Code for Cosimulation".

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 "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

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

Product Features and Platform Support

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

FPGA Verification

In this section...

FPGA Verification with HDL Verifier and HDL Coder

HDL Verifier works with Simulink or MATLAB and HDL Coder and the supported FPGA development environment to prepare your automatically generated HDL code for implementation in an FPGA. FPGA-in-the-Loop (FIL) simulation allows you to run a Simulink or MATLAB simulation with an FPGA board strictly synchronized with this software. This process 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:

- With the HDL Verifier FIL Wizard.
- With the HDL Coder Workflow Advisor.

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.

HDL Verifier supports the use of a FIL block in a model reference block and a System object in conjunction with a MATLAB program.

Product Features and Platform Support

Product Feature	Required Products	Recommended Products	Supported Platforms
FPGA-in-the-Loop	For FIL simulation with MATLAB: MATLAB, Fixed-Point Designer For FIL simulation with Simulink: Simulink, Fixed-Point Designer	HDL Coder	Windows 64-bit; Linux 64-bit

Preregistered FPGA Devices for FIL Simulation

HDL Verifier supports FIL simulation on the devices as described in “Supported FPGA Devices for FIL Simulation” on page 3-5. The FPGA board support packages contain the definition files for all supported boards. You may download one or more vendor-specific packages, but you must download one of the packages before you can use FIL or customize your own board definition file using the New FPGA Board Wizard (see “Create Custom FPGA Board Definition”).

To see the list of HDL Verifier support packages, visit “HDL Verifier Supported Hardware” on page 2-15. To download an FPGA board support package:

- On the MATLAB **Home** tab, in the **Environment** section, click **Add-Ons > Get Hardware Support Packages**.

TLM Component Generation

In this section...

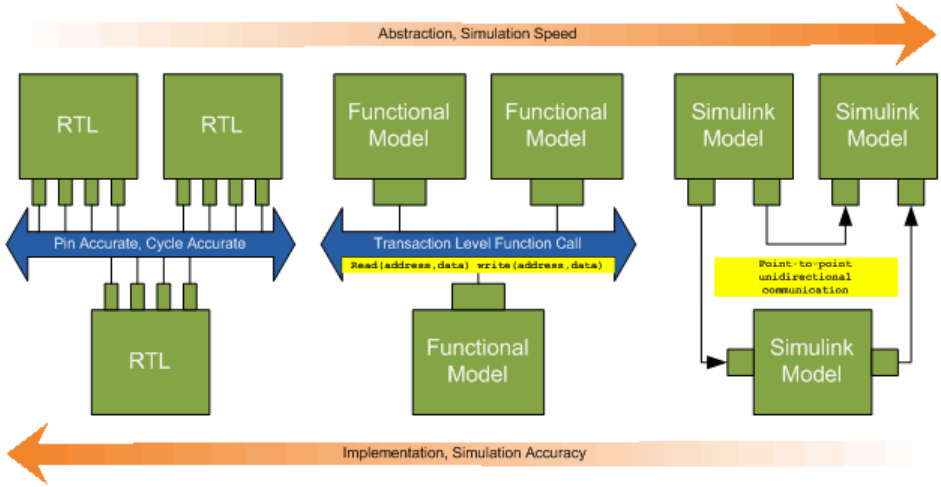
Generating TLM Components for Virtual Platform Development

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-compliant SystemC Transaction Level Model (TLM) that can be executed in any OSCI-compatible TLM 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.

Product Feature and Platform Support

Product Feature	Required Products	Recommended Products	Supported Platforms
TLM Generator	Simulink Coder™	Embedded Coder® (Simulink Coder is also required)	Windows 32-bit and 64-bit; Linux 64-bit

SystemVerilog DPI Component Generation

In this section...

Export Simulink Subsystem or MATLAB Function Using DPI Interface

You can export a Simulink subsystem or MATLAB function with a DPI interface for Verilog or SystemVerilog simulation. The coder wraps generated C code with a DPI wrapper accessed through a SystemVerilog thin interface function.

- Simulink subsystem — Access this feature from the Model Configuration Parameters dialog box, under **Code Generation**. See “Generate SystemVerilog DPI-C Component”.
- MATLAB function — Generate the component using the `dpigen` function. See “Generate DPI Component Using MATLAB”.

HDL Verifier supports SystemVerilog DPI component generation with these products and platforms.

Design Format	Required Products	Recommended Products	Supported Platforms
Simulink subsystem	Simulink and Simulink Coder	Embedded Coder	• Windows 32-bit and 64-bit • Linux 64-bit
MATLAB function	MATLAB and MATLAB Coder		• Windows 64-bit • Linux 64-bit

Generate SystemVerilog DPI Test Bench in HDL Coder

If you have an HDL Coder license, you can generate a SystemVerilog DPI-C test bench. Use the test bench to verify your generated HDL code using C code generated from your entire Simulink model, including the DUT and data sources. To use this feature, your entire model must support C code generation with Simulink Coder. You can access this feature in HDL Workflow Advisor under **HDL Code Generation > Set Testbench Options**, or in the Model Configuration Parameters dialog box, under **HDL Code**

Generation>Test Bench. Or, for command-line access, set the `GenerateSVDPITestBench` property of `makehdltb`. See "Verify HDL Design With Large Data Set Using SystemVerilog DPI Test Bench" (HDL Coder).

HDL Verifier supports SystemVerilog DPI test bench generation in HDL Coder with these products and platforms.

Design Format	Required Products	Recommended Products	Supported Platforms
Simulink subsystem	Simulink and Simulink Coder	Embedded Coder	• Windows 32-bit and 64-bit • Linux 64-bit

See Also

More About

- "DPI-C Component Generation with Simulink"
- "DPI-C Component Generation with MATLAB"

HDL Verifier Supported Hardware

As of this release, HDL Verifier supports the following hardware.

Support Package	Vendor	Earliest Release Available	Last Release Available
Intel FPGA Boards	Intel®	R2013a	Current
Xilinx FPGA Boards	Xilinx	R2013a	Current

For a complete list of supported hardware, see Hardware Support.

Third-Party Product Requirements

Supported EDA Tools and Hardware

In this section...

Cosimulation Requirements

- “Cadence Incisive Requirements” on page 3-2
- “Mentor Graphics Questa and ModelSim Usage Requirements” on page 3-3

To get started, see “Set Up MATLAB-HDL Simulator Connection” or “Start HDL Simulator for Cosimulation in Simulink”.

Cadence Incisive Requirements

MATLAB and Simulink support Cadence verification tools using HDL Verifier. Only the 64-bit version of Incisive is supported for cosimulation. Use one of these recommended versions, which have been fully tested against the current release:

- Incisive 15.2

  Note Not supported for `nclaunch` with `runmode` set to `Batch`. Set `runmode` to `CLI` instead.

- Incisive 14.1
- Incisive 13.2

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

- Questa Core/Prime 10.5b
- QuestaSim 10.4c, 10.3
- ModelSim/QuestaSim PE 10.4c, 10.3e

FPGA Verification Requirements

Xilinx Usage Requirements

MATLAB and Simulink support Xilinx design tools using HDL Verifier. Use the FPGA-in-the-loop tools with these recommended versions:

- Xilinx Vivado® 2016.4
- Xilinx ISE 14.7

Note Xilinx ISE is required for FPGA boards in the Spartan®-6, Virtex®-4, Virtex-5, and Virtex-6 families.

For tool setup instructions, see "Set Up FPGA Design Software Tools".

Intel Quartus Prime Usage Requirements

MATLAB and Simulink support Intel design tools using HDL Verifier. Use the FPGA-in-the-loop tools with these recommended versions:

- Altera Quartus® II 15.0
- Intel Quartus Prime 16.1

For tool setup instructions, see “Set Up FPGA Design Software Tools”.

Supported FPGA Board Connections for FIL Simulation

For board support, see “Supported FPGA Devices for FIL Simulation” on page 3-5.

Additional boards can be custom added with the “FPGA Board Manager”. See “Supported FPGA Device Families for Board Customization” on page 3-9.

JTAG Connection

Vendor	Required Hardware	Required Software
Altera	• USB Blaster I or USB Blaster II download cable	• USB Blaster I or II driver • For Windows operating systems: Quartus Prime executable directory must be on system path. • For Linux operating systems: versions below Quartus II 13.1 are not supported. Quartus II 14.1 is not supported. Only 64-bit Quartus is supported. Quartus library directory must be on `LD_LIBRARY_PATH` *before* starting MATLAB. Prepend the Linux distribution library path before the Quartus library on `LD_LIBRARY_PATH`. For example, `/lib/x86_64-linux-gnu:$QUARTUS_PATH`.
Xilinx	• Digilent® download cable. If your board has a standard Xilinx 14 pin JTAG connector, you can obtain the HS2 cable from Digilent.	• For Windows operating systems: Xilinx Vivado executable directory must be on system path. • For Linux operating systems: Digilent Adept2

Note When simulating your FPGA design through Digilent JTAG cable with Simulink or MATLAB, you cannot use any debugging software that requires access to the JTAG; for example, Vivado Logic Analyzer.

Ethernet Connection

Required Hardware	Supported Interfaces	Required Software
• Gigabit Ethernet card • Cross-over Ethernet cable • FPGA board with supported Ethernet connection	• Gigabit Ethernet — GMII • Gigabit Ethernet — RGMII • Gigabit Ethernet — SGMII • Ethernet — MII	There are no software requirements for an Ethernet connection, but ensure that the firewall on the host computer does not prevent UDP communication. **Note** Ethernet connection to Virtex-7 VC707 not supported for Vivado versions older than 2013.4.

PCI Express

Note FIL over PCI Express® connection is supported only for 64-bit Windows operating systems.

Device Family	Board	Required Software
Xilinx	• Kintex®-7 KC705 Evaluation Kit • Virtex -7 VC707 Evaluation Kit	Vivado 2015.2
Altera	• Cyclone® V GT FPGA Development Kit • DSP Development Kit, Stratix® V Edition	Altera Quartus II 15.0

Supported FPGA Devices for FIL Simulation

HDL Verifier supports FIL simulation on the devices shown in the following table. The board definition files for these boards are in the “Download FPGA Board Support Package”. You can add other FPGA boards for use with FIL with FPGA board customization (“FPGA Board Customization”).

Device Family	Board	Ethernet	JTAG	PCI Express	Comments
Xilinx Artix®-7	Digilent Nexys™4 Artix-7	x	x		
	Digilent Arty Board		x		
Xilinx Kintex-7	Kintex-7 KC705	x	x	x	
Xilinx Kintex UltraScale™	Kintex UltraScale FPGA KCU105 Evaluation Kit	x	x		
Xilinx Spartan-6	Spartan-6 SP605	x			
	Spartan-6 SP601	x			
	XUP Atlys Spartan-6	x			
Xilinx Virtex UltraScale	Virtex UltraScale FPGA VCU108 Evaluation Kit	x	x		
Xilinx Virtex-7	Virtex-7 VC707	x	x	x	
	Virtex-7 VC709		x	x	
Xilinx Virtex-6	Virtex-6 ML605	x			
Xilinx Virtex-5	Virtex ML505	x			
	Virtex ML506	x			
	Virtex ML507	x			
	Virtex XUPV5–LX110T	x			
Xilinx Virtex	Virtex ML401	x			
	Virtex ML402	x			
	Virtex ML403	x			

Device Family	Board	Ethernet	JTAG	PCI Express	Comments
Xilinx Zynq®	Zynq-7000 ZC702		x		
	Zynq-7000 ZC706		x		
	ZedBoard™		x		
	ZYBO™ Zynq-7000 Development Board		x		
Xilinx Zynq UltraScale+™	Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit		x		
Altera Arria® II	Arria II GX FPGA Development Kit	x	x		
Altera Arria V	Arria V SoC Development Kit		x		
	Arria V Starter Kit	x	x		
Altera Arria 10	Arria 10 SoC Development Kit		x		
Altera Cyclone IV	Cyclone IV GX FPGA Development Kit	x	x		
	DE2-115 Development and Education Board	x	x		The Altera DE2-115 FPGA development board has two Ethernet ports. FPGA-in-the-loop uses only Ethernet 0 port. Make sure that you connect your host computer with the Ethernet 0 port on the board via an Ethernet cable.
	BeMicro SDK	x	x		
Altera Cyclone III	Cyclone III FPGA Starter Kit		x		

Device Family	Board	Ethernet	JTAG	PCI Express	Comments
	Cyclone III FPGA Development Kit	x	x		
	Altera Nios II Embedded Evaluation Kit, Cyclone III Edition	x	x		
Altera Cyclone V	Cyclone V GX FPGA Development Kit	x	x		
	Cyclone V SoC Development Kit		x		
	Cyclone V GT Development Kit	x	x	x	
	Terasic Atlas-SoC Kit / DE0-Nano SoC Kit		x		
	Arrow® SoCKit Development Kit		x		
Altera MAX® 10	Arrow MAX 10 DECA	x	x		
Altera Stratix IV	Stratix IV GX FPGA Development Kit	x	x		
Altera Stratix V	DSP Development Kit, Stratix V Edition	x	x	x	

Limitations

- For FPGA development boards that have more than one FPGA device, only one such device can be used with FIL.

FPGA Board Support Packages

The FPGA board support packages contain the definition files for all supported boards. You can download one or more vendor-specific packages. To use FIL, download at least one of these packages, or customize your own board definition file. See “Create Custom FPGA Board Definition”.

To see the list of HDL Verifier support packages, visit “HDL Verifier Supported Hardware” on page 2-15. To download an FPGA board support package:

- On the MATLAB **Home** tab, in the **Environment** section, click **Add-Ons** > **Get Hardware Support Packages**.

Supported FPGA Device Families for Board Customization

HDL Verifier supports the following FPGA device families for board customization; that is, when you create your own board definition file. See “FPGA Board Customization”. PCI Express is not a supported connection for board customization.

Device Family		Restrictions
Xilinx	Artix 7	
	Kintex 7	
	Kintex UltraScale	
	Spartan 6	Ethernet PHY RGMII is not supported.
	Virtex 4	
	Virtex 5	
	Virtex 6	
	Virtex 7	Supports Ethernet PHY SGMII only.
	Virtex UltraScale	
	Zynq 7000	
	Zynq UltraScale+	
Altera	Arria II	
	Arria V	
	Arria 10	
	Cyclone III	
	Cyclone IV	
	Cyclone V	
	MAX 10	
	Stratix IV	

Device Family		Restrictions
	Stratix V	

DPI Component Generation Requirements

DPI component generation supports the same versions of Cadence Incisive and Mentor Graphics Questa and ModelSim as for cosimulation. You can generate a DPI component for use with either 64-bit or 32-bit Incisive.

Note When you run a DPI component in ModelSim 10.5b on Debian® 8.3, you may encounter a library incompatibility error:

```
** Warning: ** Warning: (vsim-7032) The 64-bit glibc RPM
does not appear to be installed on this machine.  Calls to gcc may fail.
** Fatal: ** Error: (vsim-3827) Could not compile 'STUB_SYMS_OF_fooour.so':
```

To avoid this issue, on the **Code Generation** pane in Configuration Parameters, try these options:

- Set the **Build configuration** to `Faster Runs`.
- Or, set the **Build configuration** to `Specify` and specify the compiler flag `-O3`.

TLM Generation Requirements

With the current release, TLMG includes support for:

- Compilers:
  - Visual Studio®: VS2008, VS2010, VS2012, and VS2013
  - Windows 7.1 SDK
  - gcc 4.4.6
- SystemC:
  - SystemC 2.3.1 (TLM included)

    You can download SystemC and TLM libraries at http://accellera.org. Consult the Accellera Systems Initiative website for information about how to build these libraries after downloading.

- System C Modeling Library (SCML):
  - SCML 2.2

    You can download SCML from https://www.synopsys.com.