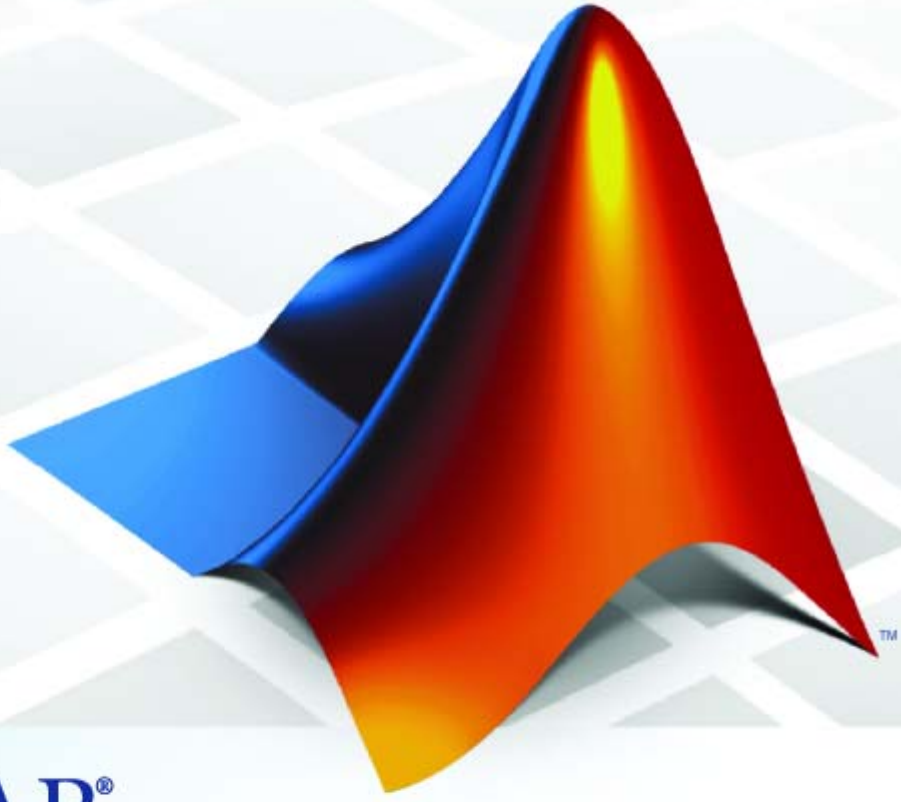
EDA Simulator Link™ 3
Getting Started Guide



MATLAB®
&SIMULINK®

The MathWorks™
Accelerating the pace of engineering and science

How to Contact The 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

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

The 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)

Contents

1 Introduction

2 Installing the EDA Simulator Link Software

3 Learning More About the EDA Simulator Link Software

1

Introduction

Product Overview

Product Description

EDA Simulator Link™ integrates MATLAB® and Simulink® with hardware design flows. It 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

EDA Simulator Link is a cosimulation interface that provides a bidirectional link between MATLAB and Simulink and HDL simulators from Mentor Graphics®, Cadence®, and Synopsys®, enabling verification of VHDL®, Verilog®, and mixed-language implementations.

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

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

TLM Generation

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

EDA Simulator Link works with Simulink, Simulink® 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.

Using EDA Simulator Link with HDL Simulators

In this section...

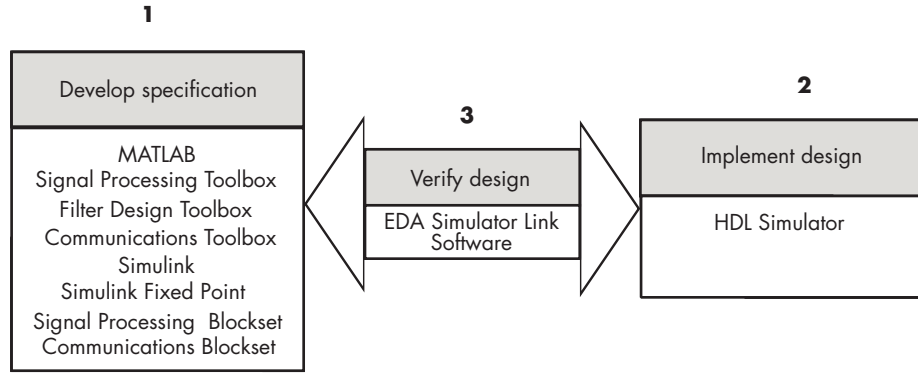
Overview to Cosimulation with MATLAB or Simulink and the HDL Simulator

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:

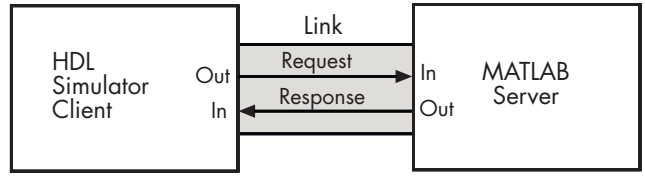
- Use MATLAB or Simulink to create test signals and software test benches for HDL code
- Use MATLAB or Simulink to provide a behavioral model for an HDL simulation
- Use MATLAB analysis and visualization capabilities for real-time insight into an HDL implementation
- Use Simulink to translate legacy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



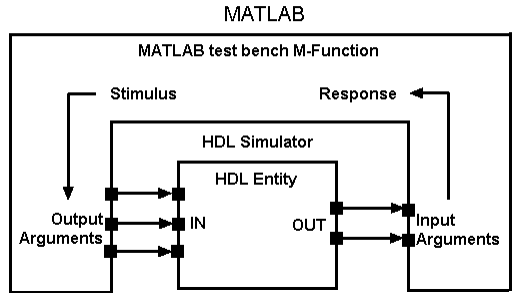
In this scenario, a MATLAB server function waits for service requests that it receives from an HDL simulator session. After receiving a request, the server establishes a communication link and invokes a specified MATLAB function that computes data for, verifies, or visualizes the HDL module (coded in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

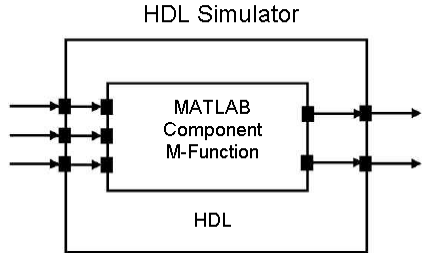
- `nclaunch` (Incisive)
- `vsim` (ModelSim)
- `launchDiscovery` (Discovery)

For more information, see “Starting the HDL simulator etc.”

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.

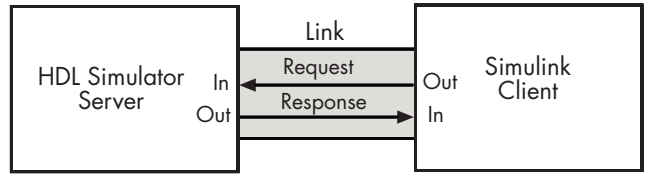


When you begin a specific test bench or component session, you specify parameters that identify the following information:

- 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. (Incisive and ModelSim only. See the reference page for `launchDiscovery` for instructions on creating clocks for use with Discovery using Tcl commands).
- Tcl commands to run before and after the simulation. (Incisive and ModelSim only. See the reference page for `launchDiscovery` for instructions on how to issue Tcl commands for use with Discovery)

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`. Discovery users can perform similar operations with the `launchDiscovery` command.

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 The 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 “Specifying TCP/IP Socket Communication”.

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.

Starting the HDL Simulator from MATLAB

For each supported HDL simulator, EDA Simulator Link has a unique command to launch the HDL simulator from within MATLAB. Each command contains a set of customized property value pairs for specifying the EDA Simulator Link library to use, the design to load, the type of communication connection, and so on.

The HDL simulator launch commands are as follows:,

HDL Simulator	EDA Simulator Link Launch Command
Cadence Incisive	`nclaunch`
Mentor Graphics ModelSim	`vsim`
Synopsys Discovery	`launchDiscovery`

You issue the launch command directly from MATLAB and provide the EDA Simulator Link library information and other required parameters (see "Using the EDA Simulator Link Libraries for HDL Cosimulation" on page 1-16). No special setup is required. This function starts and configures the HDL simulator for use with the EDA Simulator Link software. By default, the function starts the first version of the simulator executable that it finds on the system path (defined by the `path` variable), using a temporary file that is overwritten each time the HDL simulator starts.

You can customize the startup file and communication mode to be used between MATLAB or Simulink and the HDL simulator by specifying the call to the appropriate launch command with property name/property value pairs. Refer to the `nclaunch`, `vsim`, or `launchDiscovery` reference documentation for specific information regarding the property name/property value pairs.

If you want to start a different version of the simulator executable than the first one found on the system path, use the setenv and getenv MATLAB functions to set and get the environment of any sub-shells spawned by UNIX(), DOS(), or system().

When you specify a communication mode using any of the EDA Simulator Link HDL simulator launch commands, the function applies the specified communication mode to all MATLAB or Simulink/HDL simulator sessions.

See "Starting the ModelSim Simulator from MATLAB" on page 1-10, "Starting the Cadence Incisive Simulator from MATLAB" on page 1-12, and "Starting the Discovery Simulator from MATLAB" on page 1-13 for examples of using each of the EDA Simulator Link HDL simulator launch commands with various property/name value pairs and other parameters.

Diagnostic and Customization Setup Script for use with Incisive and ModelSim If you would like some assistance in setting up your environment for use with EDA Simulator Link, you can diagnose your setup (correct omissions and errors) and also customize your setup for future invocations of `nclaunch` or `vsim` by following the process in "Diagnosing and Customizing Your Setup for Use with the HDL Simulator and EDA Simulator Link Software".

Starting the ModelSim Simulator from MATLAB

To start the HDL simulator from MATLAB, enter `vsim` at the MATLAB command prompt:

```
>> vsim('PropertyName', 'PropertyValue'...)
```

The following example changes the folder location to `VHDLproj` and then calls the function `vsim`. Because the command line omits the `'vsimdir'` and `'startupfile'` properties, `vsim` creates a temporary DO file. The

`'tclstart'` property specifies Tcl commands that load and initialize the HDL simulator for test bench instance `modsimrand`.

```
cd VHDLproj
vsim('tclstart',...
 'vsimmatlab modsimrand; matlabtb modsimrand 10 ns -socket 4449')
```

The following example changes the folder location to `VHDLproj` and then calls the function `vsim`. Because the function call omits the `'vsimdir'` and `'startupfile'` properties, `vsim` creates a temporary DO file. The `'tclstart'` property specifies a Tcl command that loads the VHDL entity `parse` in library `work` for cosimulation between `vsim` and Simulink. The `'socketsimulink'` property specifies TCP/IP socket communication on the same computer, using socket port 4449.

```
cd VHDLproj
vsim('tclstart', 'vsimulink work.parse', 'socketsimulink', '4449')
```

The following example has the HDL compilation and simulation commands run automatically when you start the ModelSim software from MATLAB.

```
vsim('tclstart',
       {'vlib work', 'vlog +acc clocked_inverter.v hdl_top.v', 'vsim +acc hdl_top' });
```

This next example loads the HDL simulation just as in the previous example but it also loads in the Link to Simulink library, uses socket number 5678 to communicate with cosimulation blocks in Simulink models, and uses an HDL time precision of 10 ps.

```
vsim('tclstart',
             {'vlib work', 'vlog -novopt clocked_inverter.v hdl_top.v',
              'vsimulink hdl_top -socket 5678 -t 10ps'});
```

Or

```
vsim('tclstart',
             {'vlib work', 'vlog -novopt clocked_inverter.v hdl_top.v',
               'vsimulink hdl_top -t 10ps'},
               'socketsimulink', 5678);
```

Starting the Cadence Incisive Simulator from MATLAB

To start the HDL simulator from MATLAB, enter `nclaunch` at the MATLAB command prompt:

```
>> nclaunch('PropertyName', 'PropertyValue'...)
```

The following example changes the folder location to `VHDLproj` and then calls the function `nclaunch`. Because the command line omits the `'hdlsimdir'` and `'startupfile'` properties, `nclaunch` creates a temporary file. The `'tclstart'` property specifies Tcl commands that load and initialize the HDL simulator for test bench instance `modsimrand`.

```
cd VHDLproj
nclaunch('tclstart',...
 'hdlsimmatlab modsimrand; matlabtb modsimrand 10 ns -socket 4449')
```

The following example changes the folder location to `VHDLproj` and then calls the function `nclaunch`. Because the function call omits the `'hdlsimdir'` and `'startupfile'` properties, `nclaunch` creates a temporary file. The `'tclstart'` property specifies a Tcl command that loads the VHDL entity `parse` in library `work` for cosimulation between `nclaunch` and Simulink. The `'socketsimulink'` property specifies TCP/IP socket communication on the same computer, using socket port 4449.

```
cd VHDLproj
nclaunch('tclstart', 'hdlsimulink work.parse', 'socketsimulink', '4449')
```

Another option is to bring `ncsim` up in the terminal instead of launching the Simvision GUI, thereby allowing you to interact with the simulation. This next example lists the steps necessary for you to do this:

1. Start `hdldaemon` in MATLAB.
2. Start an `xterm` from MATLAB in the background (key point).
3. Run `ncsim` in the `xterm` shell having it call back to the hdlserver to run your `matlabcp` function as usual.
4. Have the `matlabcp` function touch a file to signal completion while an MATLAB script polls for completion.

The MATLAB script can then change test parameters and run more tests.

Note The `nclaunch` command requires the use of property name/property value pairs. You get an error if you try to use the function without them.

Starting the Discovery Simulator from MATLAB

To start the HDL simulator from MATLAB, enter `launchDiscovery` at the MATLAB command prompt:

```
>> launchDiscovery('PropertyName', 'PropertyValue'...)
```

This example compiles and launches a single-file HDL design for cosimulation with Simulink. The code allows the use of Verilog-2000 syntax in the HDL source. This code launches the Synopsys DVE software.

```
>> launchDiscovery( ...
        'LinkType',     'Simulink', ...
        'VerilogFiles', 'myinverter.v', ...
        'VlogAnFlags',  '+v2k', ...
        'TopLevel',     'myinverter', ...
        'AccFile',      'myinverter.acc' ...
    );
```

Note The `launchDiscovery` command requires the use of property name/property value pairs. You get an error if you try to use the function without them.

Starting the HDL Simulator from a Shell

Starting the ModelSim Software from a Shell

To start the HDL simulator from a shell and include the EDA Simulator Link libraries, you need to first run the configuration script. See "Diagnosing and Customizing Your Setup for Use with the HDL Simulator and EDA Simulator Link Software".

After you have the configuration files, you can start the ModelSim software from the shell by typing:

```
% vsim design_name -f matlabconfigfile
```

`matlabconfigfile` should be the name of the MATLAB configuration file you created with `syscheckmq` (Linux/UNIX) or that you created yourself using our template (Windows). If you are connecting to Simulink, this should be the name of the Simulink configuration file. You must also specify the path to the configuration file even if it resides in the same folder as `vsim.exe`. Use `design_name` if you want to also start the simulation.

The configuration file mainly defines the -foreign option to `vsim` which in turn loads the EDA Simulator Link shared library and specifies its entry point.

You can also specify any other existing configuration files you may also be using with this call.

If you are performing this step manually, the following use of -foreign with `vsim` loads the EDA Simulator Link client shared library and specifies its entry point:

```
% vsim design_name -foreign matlabclient /path/library
```

where path is the path to this particular EDA Simulator Link library. See "Using the EDA Simulator Link Libraries for HDL Cosimulation" on page 1-16 to find the correct library name for your machine. Use `design_name` if you want to also start the simulation.

Note You can also issue this exact same command from inside the HDL simulator.

Starting the Cadence Incisive HDL Simulator from a Shell

To start the HDL simulator from a shell and include the EDA Simulator Link libraries, you need to first run the configuration script. See "Using the Configuration and Diagnostic Script for UNIX/Linux".

After you have the configuration files, you can start the HDL simulator from the shell by typing:

```
% ncsim -f matlabconfigfile modelname
```

`matlabconfigfile` should be the name of the MATLAB configuration file you created with `syscheckin`. If you are connecting to Simulink, this should be the name of the Simulink configuration file. For example:

```
% ncsim -gui -f simulinkconfigfile modelname
```

Either way, you must also specify the path to the configuration file if it does not reside in the same folder as `ncsim.exe`.

You can also specify any other existing configuration files you may also be using with this call.

Starting ncsim in an xterm Terminal. If you would like to bring up ncsim in an xterm terminal, instead of launching the Simvision GUI, perform the following steps:

1. Start `hdldaemon` in MATLAB.
2. Start an xterm from MATLAB in the background.
3. Run ncsim in the xterm shell, having it call back to the hdlserver to run your `matlabtb` function as usual.
4. Specify that the `matlabtb` function use the touch command on a file to signal completion while a MATLAB script polls for completion.

   The MATLAB script can then change test parameters and run more tests.

Starting the Discovery Software from a Shell

You can run the scripts generated with a call to `launchDiscovery` (or scripts you've created yourself) to start the Incisive software and load the link libraries outside of MATLAB. See `launchDiscovery` for more information and an example.

Using the EDA Simulator Link Libraries for HDL Cosimulation

In general, you want to use the same compiler for all libraries linked into the same executable. The link software provides many versions of the same library compilers that are available with the HDL simulators (usually some version of GCC). Using the same libraries ensures compatibility with other C++ libraries that may get linked into the HDL simulator, including SystemC libraries.

If you have any of these conditions, choose the version of the EDA Simulator Link library that matches the compiler used for that code:

- Link other third-party applications into your HDL simulator.
- Compile and link in SystemC code as part of your design or testbench.
- Write custom C/C++ applications and link them into your HDL simulator.

If you do not link any other code into your HDL simulator, you can use any version of the supplied libraries (Discovery users should use any supplied library matching the installed version of GCC —gcc422— from the 2008.12 or 2009.6 VG_GNU_PACKAGE). The EDA Simulator Link launch command (`nclaunch`, `vsim`, or `launchDiscovery`) chooses a default version of this library.

For examples on specifying EDA Simulator Link libraries when cosimulating across a network, see "Performing Cross-Network Cosimulation".

Library Names

The EDA Simulator Link HDL libraries use the following naming format:

```
edalink/extensions/version/arch/lib{version_short_name}{client_server_tag}
    _{design_language}_{compiler_tag).{libext}
```

where

Argument	Incisive Users	ModelSim Users	Discovery Users
`version`	incisive	modelsim	discovery
`arch`	linux32 or linux64	linux32,linux64, or windows32	linux32 or linux64
`version_short_name`	lfihdl	lfmhdl	lfdhdl
`client_server_tag`	c (MATLAB) or s (Simulink)	c (MATLAB) or s (Simulink)	c (MATLAB) or s (Simulink)
`design_language` (Discovery only)	N/A	N/A	mixed = pure vhdl or mixed vhdl/vlog vlog = pure vlog
`compiler_tag`	gcc41, tmwgcc	gcc412, tmwgcc	gcc422
`libext`	so	dll or so	so

Not all combinations are supported. See “Default Libraries” on page 1-17 for valid combinations.

For more on MATLAB build compilers, see MATLAB Build Compilers.

Default Libraries

EDA Simulator Link scripts fully support the use of default libraries.

The following table lists all the libraries shipped with the link software for each supported HDL simulator. The default libraries for each platform are in bold text.

Default Libraries for use with ModelSim

Platform	MATLAB Library	Simulink Library
Linux32, Linux64	**liblfmhdlc_tmwgcc.so** liblfmhdlc_gcc412.so	**liblfmhdls_tmwgcc.so** liblfmhdls_gcc412.so
Windows32	**liblfmhdlc_tmwgcc.dll**	**liblfmhdls_tmwgcc.dll**

Note ModelSim uses gcc412 by default; EDA Simulator Link uses tmwgcc by default. Therefore, if you are compiling HDL code in ModelSim make sure you are compiling with the same library that EDA Simulator Link is using; either tmwgcc by default or gcc412 if you so specified with the `vsim` command.

Default Libraries for use with Incisive

Platform	MATLAB Library	Simulink Library
Linux32, Linux64	**liblfihdlc_gcc41.so** liblfihdlc_tmwgcc.so	**liblfihdls_gcc41.so** liblfihdsl_tmwgcc.so

Default Libraries for use with Discovery

Platform	MATLAB Library	Simulink Library
Linux32, Linux64	**liblfdhdlc_vlog_gcc422.so** **liblfdhdlc_mixed_gcc422.so**	**liblfdhdls__vlog_gcc422.so** **liblfdhdls__mixed_gcc422.so**

Using an Alternative Library

The EDA Simulator Link launch commands contain parameters for specifying the HDL-side library.

- "Incisive Users: Using an Alternative Library" on page 1-19
- "ModelSim Users: Using an Alternative Library" on page 1-21
- "Discovery Users: Using an Alternate Library" on page 1-23

Incisive Users: Using an Alternative Library. You can use a different HDL-side library by specifying it explicitly using the libfile parameter to the `nclaunch` MATLAB command. You should choose the version of the library that matches the compiler and system libraries you are using for any other C/C++ libraries linked into the HDL simulator. Depending on the version of your HDL simulator, you may need to explicitly set additional paths in the LD_LIBRARY_PATH environment variable.

For example, if you want to use a nondefault library:

1. Copy the system libraries from the MATLAB installation (found in *`matlabroot`*`/sys/os/`*`platform`*) to the machine with the HDL simulator (where *matlabroot* is your MATLAB installation and *platform* is one of the above architecture, for example, linux32).

2. Modify the LD_LIBRARY_PATH environment variable to add the path to the system libraries that were copied in step 1.

Example: EDA Simulator Link Alternate Library Using nclaunch

In this example, you are using the 32-bit Linux version of IUS 08.20-p001 on the same 64-bit Linux machine that is running MATLAB. Because you have your own C++ application, and you are linking into ncsim that you used twmgcc to compile, you are using the EDA Simulator Link version compiled with tmwgcc, instead of using the default library version compiled with GCC 4.1.

In MATLAB:

```
>> currPath = getenv('PATH');
>> currLdPath = getenv('LD_LIBRARY_PATH');
>> setenv('PATH', ['/tools/IUS-82/bin:' currPath]);
>> nclaunch('tclstart', { 'exec ncvhdl inverter.vhd', ...
                          'exec ncelab -access +rwc inverter', ...
                          'hdlsimulink -gui inverter' }, ...
            'libfile', liblfihdls_tmwgcc');
```

The PATH is changed to ensure we get the correct version of the HDL simulator tools. Note that the `nclaunch` MATLAB command will automatically detect the use of the 32-bit version of the HDL simulator and

use the linux32 library folder in the EDA Simulator Link installation; there is no need to specify the libdir parameter in this case.

The library resolution can be verified using `ldd` from within the ncsim console GUI.

```
ncsim> exec ldd /path/to/liblfihdls_tmwgcc.so
 linux-gate.so.1 =>  (0xf7f4f000)
 libpthread.so.0 => /lib32/libpthread.so.0 (0xf7ed9000)
 libstdc++.so.6 => /usr/lib32/libstdc++.so.6 (0xf7deb000)
 libm.so.6 => /lib32/libm.so.6 (0xf7dc7000)
 libgcc_s.so.1 => /usr/lib32/libgcc_s.so.1 (0xf7dba000)
 libc.so.6 => /lib32/libc.so.6 (0xf7c67000)
 /lib/ld-linux.so.2 (0xf7f50000)
```

Example: EDA Simulator Link Alternate Library Using System Shell

This example shows how to load a Cadence Incisive simulator session by explicitly specifying the EDA Simulator Link library (default or not). By explicitly using a system shell, you can execute this example on the same machine as MATLAB, on a different machine, and even on a machine with a different operating system.

In this example, you are running the 64-bit Linux version of Cadence Incisive 5.83p2; it does not matter what machine MATLAB is running on. Instead of using the default library version compiled with GCC 3.2.3 in the Cadence Incisive distribution, you are using the version compiled with GCC 3.4.6 in the Cadence Incisive distribution.

In a csh-compatible system shell:

```
csh> setenv PATH /tools/ius-583p2/lnx/tools/bin/64bit:${PATH}
csh> setenv LD_LIBRARY_PATH /tools/ius-583p2/lnx/tools/systemc/gcc/3.4.6-x86_64
    /install/lib64:${LD_LIBRARY_PATH}
csh> ncvhdl inverter.vhd
csh> ncelab -access +rwc inverter
csh> ncsim -tcl -loadvpi /tools/matlab-7b/toolbox/edalink/extensions/incisive/linux64
    /liblfihdlc_gcc346:matlabclient inverter.vhd
```

The PATH is changed to ensure we get the correct version of the Cadence Incisive tools. Although `ncsim` will automatically find any GCC libs in its installations, the LD_LIBRARY_PATH is changed to show how you might do this with a custom installation of GCC.

You can check the proper library resolution using `ldd` as in the previous example.

ModelSim Users: Using an Alternative Library. You can use a different HDL-side library by specifying it explicitly using the libfile parameter to the `vsim` MATLAB command. You should choose the version of the library that matches the compiler and system libraries you are using for any other C/C++ libraries linked into the HDL simulator. Depending on the version of your HDL simulator, you may need to explicitly set additional paths in the LD_LIBRARY_PATH environment variable.

For example, if you want to use a nondefault library:

1. Copy the system libraries from the MATLAB installation (found in *`matlabroot`*`/sys/os/`*`platform`*) to the machine with the HDL simulator (where *matlabroot* is your MATLAB installation and *platform* is one of the above architecture, for example, linux32).
2. Modify the LD_LIBRARY_PATH environment variable to add the path to the system libraries that were copied in step 1.

Example: EDA Simulator Link Alternative Library Using vsim

In this example, you run the 32-bit Linux version of ModelSim 6 software on the same 64-bit Linux machine which is running MATLAB. Because you want to incorporate some SystemC designs, you are using theEDA Simulator Link version compiled with GCC 4.1.2. You can download the appropriate version of GCC with its associated system libraries from Mentor Graphics, instead of using the default library version compiled with tmwgcc.

In MATLAB:

```
>> currPath = getenv('PATH');
>> currLdPath = getenv('LD_LIBRARY_PATH');
>> setenv('PATH', ['/tools/modelsim-6.5c/bin:' currPath]);
```

```
>> setenv('LD_LIBRARY_PATH', ['/tools/modelsim-6.5c/gcc-4.1.2-linux/lib:' currLdPath]);
>> setenv('MTI_VCO_MODE', '32');
>> vsim('tclstart', { 'vlib work', 'vcom inverter.vhd', 'vsimulink inverter' }, ...
       'libfile', ' liblfmhdls_gcc412);
```

You change the *PATH* to ensure that you get the correct version of the ModelSim software. You change the *LD_LIBRARY_PATH* because the HDL simulator does not automatically add the necessary path to the system libraries. The EDA Simulator Link function `vsim` automatically detects the use of the 32-bit version of the HDL simulator and uses the linux32 library folder in the link software installation; there is no need to specify the libdir parameter in this case.

The library resolution can be verified using `ldd` from within the ModelSim GUI:

```
exec ldd /path/to/liblfmhdls_gcc412.so
#  linux-gate.so.1 =>  (0xf7efc000)
#  libpthread.so.0 => /lib32/libpthread.so.0 (0xf7e8a000)
#  libstdc++.so.6 => /mathworks/hub/share/apps/HDLTools/ModelSim/modelsim-6.5c-tmw-000/
         modeltech/gcc-4.1.2-linux/lib/libstdc++.so.6 (0xf7d9c000)
#  libm.so.6 => /lib32/libm.so.6 (0xf7d78000)
#  libgcc_s.so.1 => /mathworks/hub/share/apps/HDLTools/ModelSim/modelsim-6.5c-tmw-000/
         modeltech/gcc-4.1.2-linux/lib/libgcc_s.so.1 (0xf7d6d000)
#  libc.so.6 => /lib32/libc.so.6 (0xf7c1b000)
#  /lib/ld-linux.so.2 (0xf7efd000)
```

Example: EDA Simulator Link Alternate Library Using System Shell

This example shows how to load a ModelSim session by explicitly specifying the EDA Simulator Link library (either the default or one of the alternatives). By explicitly using a system shell, you can execute this example on the same machine as MATLAB, on a different machine, and even on a machine with a different operating system.

In this example, you are running the 64-bit Linux version of QuestaSim 6.2c. It does not matter which machine is running MATLAB. Instead of using the EDA Simulator Link default library version compiled with tmwgcc, you are using the version compiled with GCC 4.1.2. You can download the appropriate version of GCC with its associated system libraries from Mentor Graphics.

In this example, you are running the 64-bit Linux version of QuestaSim 6.5c. MATLAB can be running on Windows or on any other supported platform. Instead of using the EDA Simulator Link default library version compiled with tmwgcc, you are using the version compiled with GCC 4.1.2. You can download the appropriate version of GCC with its associated system libraries from Mentor Graphics.

In a csh-compatible system shell:

```
csh> setenv PATH /tools/questasim/bin:${PATH}
csh> setenv LD_LIBRARY_PATH /tools/mtigcc/gcc-4.1.2-linux_x86_64/lib64:${LD_LIBRARY_PATH}
csh> setenv MTI_VCO_MODE 64
csh> vlib work
csh> vcom +acc+inverter inverter.vhd
csh> vsim +acc+inverter -foreign "matlabclient /tools/matlab/toolbox/edalink
    /extensions/modelsim/linux64/liblfmhdlc_gcc412.so" work.inverter
```

You change the *PATH* to ensure that you get the correct version of the ModelSim software. You change the *LD_LIBRARY_PATH* because the HDL simulator does not automatically add the necessary path to the system libraries unless you are working with 6.2+ and have placed GCC at the root of the ModelSim installation.

You can check the proper library resolution using `ldd` as in the previous example.

Discovery Users: Using an Alternate Library. The only library supported for Discovery by EDA Simulator Link is gcc422 from the 2008.12 or 2009.6 VG_GNU_PACKAGE.

Using EDA Simulator Link with Virtual Platforms

In this section...

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.

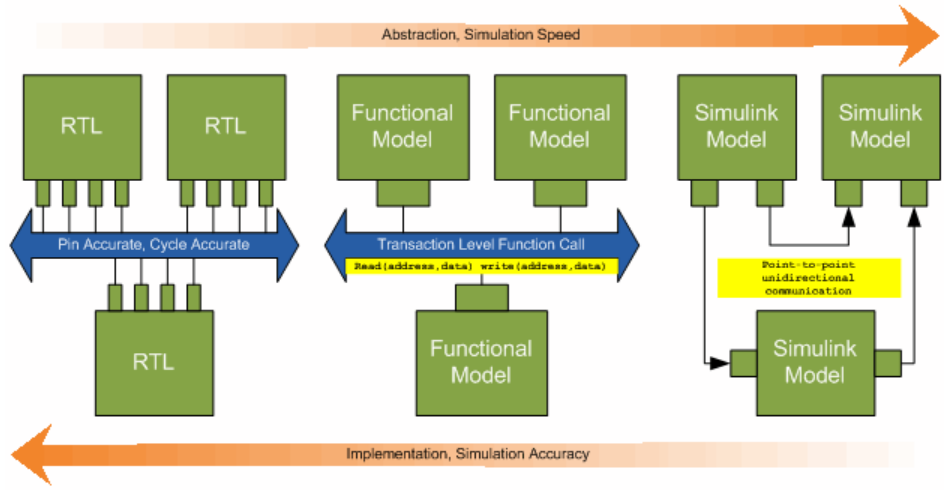
Generating TLM Components for Use with Virtual Platform Development

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.

Using EDA Simulator Link with FPGA Development Environment

In this section...

Simulation with Simulink and the FPGA Development Environment

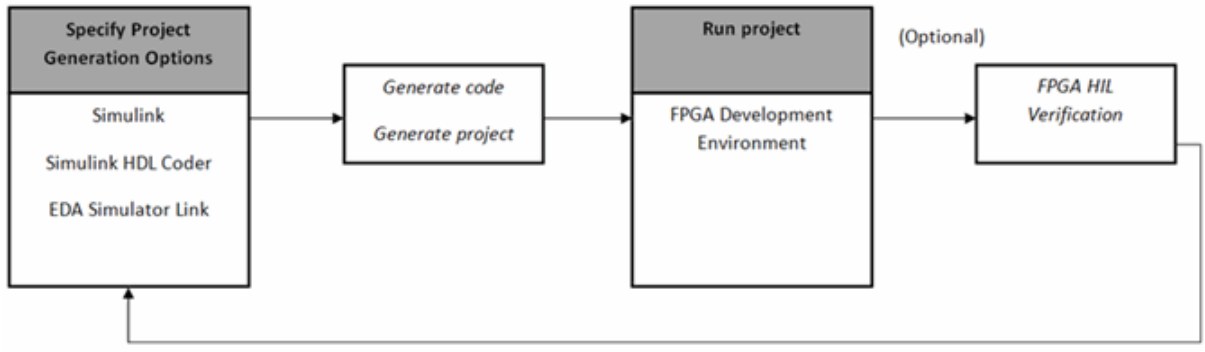
EDA Simulator Link provides a Project Generator for generating HDL code from Simulink models for FPGA implementations and packaging these files as a complete FPGA development environment project for use with Xilinx ISE.

The Project Generator component provides streamlined workflow with advanced options for project settings in the Xilinx ISE downstream workflow for FPGA implementations. The project generator uses Simulink HDL Coder to generate code to create generated FPGA implementations. 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.

In FPGA project generation, EDA Simulator Link uses the Simulink model and Simulink HDL Coder to generate HDL code and generate a Xilinx FPGA project, which can then be loaded in the Xilinx FPGA development

environment for downstream processing. This process is shown in the following diagram.



The additional FPGA hardware-in-the-loop (HIL) testing environment uses code automatically generated from Simulink models by Simulink HDL Coder software. This generated code and customized code for FPGA HIL communication are used to generated an FPGA project, run through downstream processing. The resulting programming file can be downloaded to the FPGA device for verification. See the EDA Simulator Link product page for a list of currently supported devices.

Generated FPGA Project Cosimulation 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
- Removing FPGA project associations
- Generating Tcl scripts for project generation
- Performing FPGA hardware-in-the-loop (HIL) simulation

2

Installing the EDA Simulator Link Software

Product Requirements

In this section...

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 HDL Cosimulation:
  - Hardware design and system integration
  - VHDL and/or Verilog
  - Cadence Incisive®, Mentor Graphics® ModelSim®, or Synopsys® Discovery™ simulators
  - MATLAB
  - Experience with Simulink and Simulink® Fixed Point™ software is required for applying the Simulink component of the product.
- For generated FPGA implementation
  - FPGA design and implementation
  - VHDL and/or Verilog
  - Xilinx ISE
  - Simulink and Simulink HDL Coder software
- For use with virtual platforms:
  - Simulink
  - Real-Time Workshop® Embedded Coder™ (some knowledge helpful)
  - TLM 2.0
  - System C (compiling, linking, and executing)

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

- Signal Processing Toolbox™
- Filter Design Toolbox™
- Communications Toolbox™
- Signal Processing Blockset™
- Communications Blockset™
- Video and Image Processing Blockset™
- Simulink Fixed Point
- Embedded MATLAB®

Required Products

- "Platform and Application Software Requirements" on page 2-3
- "Optional Application Software" on page 2-4

Platform and Application Software Requirements

Visit the EDA Simulator Link requirements page on The MathWorks Web site for specific platforms supported and detailed information about the software and hardware required to use EDA Simulator Link software with the current release.

Platform-Specific Software Requirements for HDL Cosimulation.

- For Use with Cadence Incisive

  The EDA Simulator Link shared libraries (`liblfihdls*.so`, `liblfihdlc*.so`) are built using the `gcc` included in the Cadence Incisive simulator platform distribution. If you are linking your own applications into the HDL simulator, the recommendation is that you also build against this `gcc`. See the HDL simulator documentation for more details about how to build and link your own applications.

- For Use with Mentor Graphics ModelSim

On the Linux platform, the `gcc` c++ libraries (4.1 or later) are required by the EDA Simulator Link software. 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
```

- For Use with Synopsys Discovery

  The EDA Simulator Link shared libraries (`liblfdhdls*.so`, `liblfdhdlc*.so`) are built using the gcc422 GCC included in the Synopsys 2008.12 and 2009.6 VG_GNU_PACKAGE distribution. To ensure compatibility with our product you must use this GCC to compile your HDL.

Platform-Specific Software Requirements for TLM Component Generation. Content TBD

Platform-Specific Software Requirements for FPGA Implementations. Content TBD

Optional Application Software

You might want to consider adding the following MathWorks products to your EDA Simulator Link setup for the most robust development environment for your application.

For HDL Cosimulation.

- Communications Blockset
- Signal Processing Blockset
- Filter Design Toolbox
- Signal Processing Toolbox
- Video and Image Processing Blockset

For Generating OSCI-Compatible TLM Components.

- Simulink Fixed Point

- Embedded MATLAB

Installation

In this section...

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.

GCCs for Synopsys Discovery

In addition to making sure Discovery is installed and on the path, you must also download and install the one supported GCC in the Synopsys 2008.12 or 2009.6 VG_GNU_PACKAGE release: gcc422. See "Using the EDA Simulator Link Libraries for HDL Cosimulation" on page 1-16.

3

Learning More About the EDA Simulator Link Software

Documentation Overview

In this section...

Documentation for HDL Cosimulation

Getting Started	Explains what the product is, the steps for installing and setting it up, how you might apply it to the hardware design process, and how to gain access to product documentation and online help. Directs you to product demos and tutorials.
"Simulating an HDL Component in a MATLAB Test Bench Environment"	Explains how to code HDL models and MATLAB test bench functions for EDA Simulator Link MATLAB applications. Provides details on how the link interface maps HDL data types to MATLAB data types and vice versa. Explains how to start and control HDL simulator and MATLAB test bench sessions.
"Replacing an HDL Component with a MATLAB Component Function"	Discusses the same topics as the chapter for test bench cosimulation using MATLAB software but instead using MATLAB to visualize an HDL module component.
"Simulating an HDL Component in a Simulink Test Bench Environment"	Explains how to use the HDL simulator and Simulink for cosimulation modeling where Simulink acts as the test bench
"Replacing an HDL Component with a Simulink Algorithm"	Explains how to use the HDL simulator and Simulink for cosimulation modeling where Simulink replaces an HDL component

"Recording Simulink Signal State Transitions for Post-Processing"	Provides instruction for adding a Value Change Dump (VCD) file block to your Simulink model for signal state change capture.
"Additional Deployment Options"	Contains several procedures for additional cosimulation arrangements: for example, performing cross-network cosimulation.
"Advanced Operational Topics"	Contains a variety of topics that provide a deeper understanding of how cosimulation works.
"Blocks — Alphabetical List"	Describes EDA Simulator Link blocks for use with Simulink.
"Functions — Alphabetical List"	Describes EDA Simulator Link functions for use with MATLAB.

Documentation for Generating TLM Components

Getting Started	Explains what the product is, the steps for installing and setting it up, how you might apply it to the hardware design process, and how to gain access to product documentation and online help. Directs you to product demos and tutorials.
"Overview to TLM Component Generation"	Provides workflow and instructions for generating a SystemC TLM 2.0 component.
"Selecting Features for the Generated TLM Component"	Describes the options available for the generated TLM component and how to select them using the Configuration Parameters dialog box.
"Creating and Applying a Test Bench for the Generated TLM Component"	Provides workflow and instructions for creating a testbench for the TLM component generated with EDA Simulator Link

"Using TLM Components in a SystemC Environment"	Describes the process necessary to provide compiler options for the generated TLM component and then execute component (and testbench) in SystemC environment.
"Configuration Parameters for TLM Generator Target"	Reference section for TLM generation Configuration Parameters.

Documentation for Generated FPGA Implementations

Getting Started	Explains what the product is, the steps for installing and setting it up, how you might apply it to the hardware design process, and how to gain access to product documentation and online help. Directs you to product demos and tutorials.
"FPGA Project Generation Overview"	Provides an overview on the benefits and features of creating an FPGA Project using EDA Simulator Link.
"FPGA Project Development"	Provides instructions for the following: • Creating generated code from a Simulink model and creating a new FPGA project to associate the generated code with • Associating generated code from Simulink model with an existing FPGA project • Updating generated code associated with an FPGA project from existing Simulink model • Removing the association between generated code from a Simulink model and an FPGA project • Generating a full or partial Tcl script for later project generation

“FPGA Hardware-in-the-Loop (HIL)”	Describes implementing the FPGA Harware-in-the-Loop feature when using EDA Simulator Link for FPGA project generation.

Documentation for Use with All EDA Simulator Link Adaptors

Getting Started	Explains what the product is, the steps for installing and setting it up, how you might apply it to the hardware design process, and how to gain access to product documentation and online help. Directs you to product demos and tutorials.
“Functions — Alphabetical List”	Describes EDA Simulator Link functions for use with MATLAB.

Online Help

In this section...

Online Help in the MATLAB Help Browser

Click the EDA Simulator Link product link in the browser's Contents or access using the MATLAB `doc` command at the MATLAB command prompt:

```
doc hdldaemon
```

at the MATLAB command prompt.

Help for EDA Simulator Link MATLAB Functions

Function help is available 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.

Block Reference Pages

Block reference pages are accessible through the Simulink interface. You can also access these block reference pages by clicking **Help** on any block dialog.

Demos and Tutorials

In this section...

Demos

The demos give you a quick view of the product’s capabilities and examples of how you might apply the product. You can run them 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 **Links and Targets > EDA Simulator Link** 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 addresses the use of the EDA Simulator Link software with the ModelSim and Simulink products.

- “Tutorial – Running a Sample ModelSim and MATLAB Test Bench Session”
- “Tutorial — Verifying an HDL Model Using Simulink, the HDL Simulator, and the EDA Simulator Link Software”
- “To VCD File Block Tutorial”

Index

I

L

M

N

O

P

R

S

T

U

V