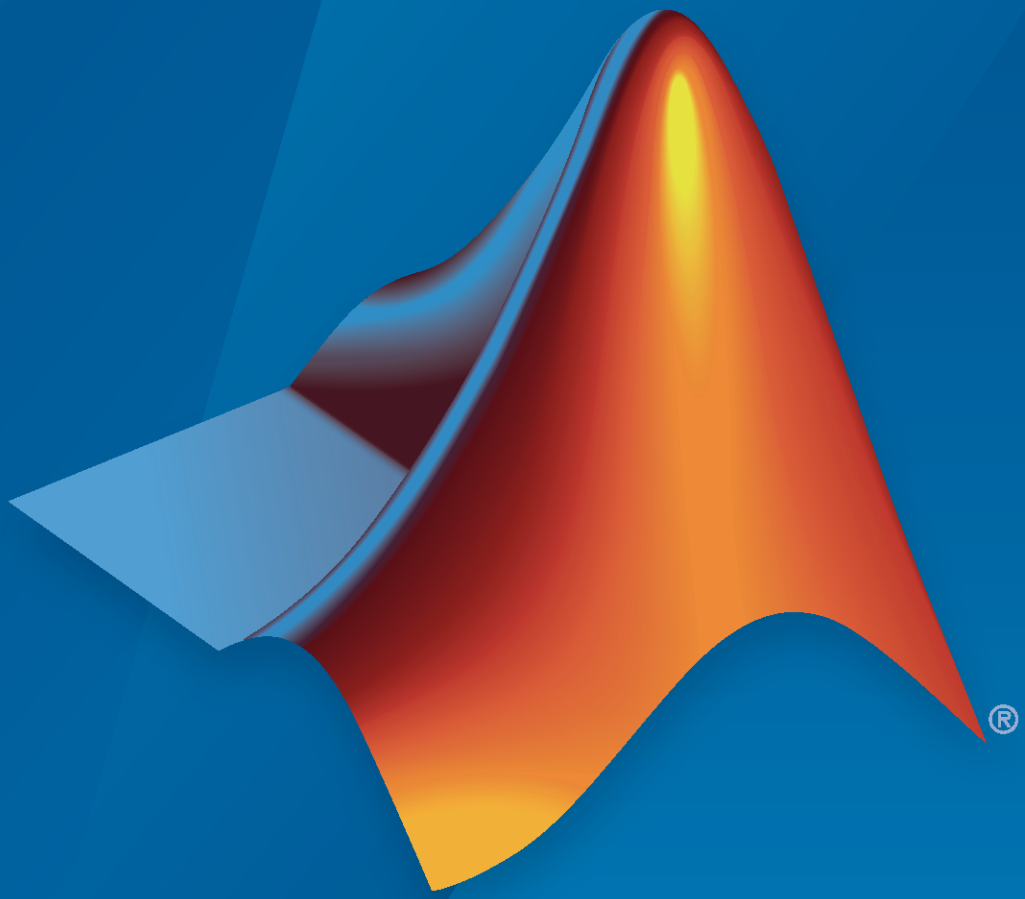
HDL Verifier™ Release Notes



MATLAB® & SIMULINK®

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.
1 Apple Hill Drive
Natick, MA 01760-2098

HDL Verifier™ Release Notes

© COPYRIGHT 2009–2023 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.

Contents

R2023a

R2022b

R2022a

R2021b

R2021a

R2020b

R2020a

R2019b

R2019a

R2018b

R2018a

R2017b

R2017a

R2016b

R2016a

R2015b

R2015a

R2014b

R2014a

R2013b

R2013a

R2012b

R2012a

R2011b

R2011a

R2010b

R2010a

R2009b

R2023a

Generate customizable SystemVerilog modules and UVM components from MATLAB using templates

Generate a unified verification methodology (UVM) component or a SystemVerilog DPI component from a MATLAB® function. Use the new `svdpiConfiguration` object to specify the kind of component and a template to use for code generation. Select a built-in template for common component kinds, or create your own custom template.

Available built-in templates are:

- Combinational module
- Sequential module
- Sequential module with variable size outputs
- Predictor component
- Sequence component
- Scoreboard component

To generate the component, use the `dpigen` function and set the `-config` argument to your `svdpiConfiguration` object.

For example, to use a function named `PulseDetectorRef` as a predictor in a UVM test bench, create a configuration object specifying the UVM-predictor kind:

```
svcfg = svdpiConfiguration('uvm-predictor');
```

Next, call the `dpigen` function specifying the `PulseDetectorRef` function, its input arguments, and the configuration object:

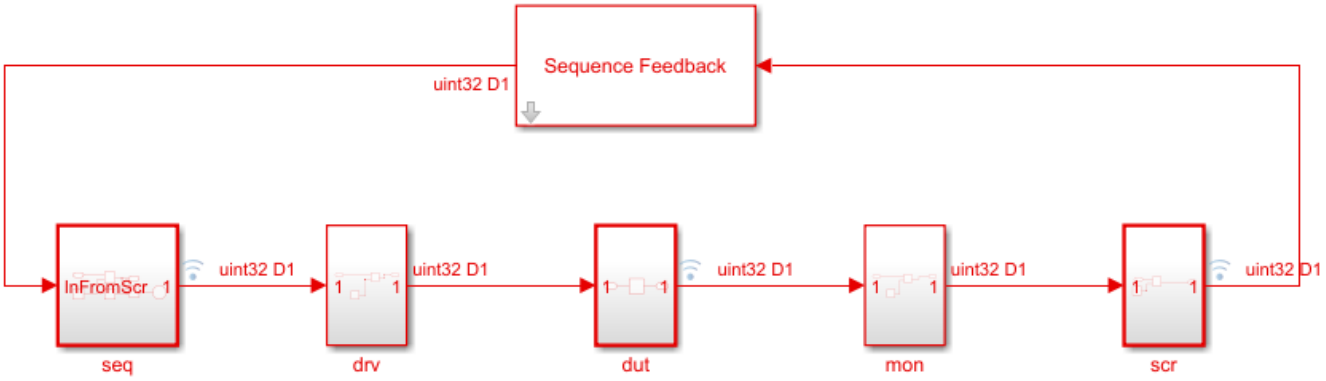
```
inargs = '{repmat(complex(1,1),[64,1]),repmat(complex(1,1),[500,1]),0}';
dpigen('PulseDetectorRef','-args',inargs,);
```

This action generates a SystemVerilog predictor based on the `PulseDetectorRef` function, an input transaction, and an output transaction.

For more information about templates, see “SystemVerilog and UVM Template Engine”. For information about template variables and directives, see “Template Engine Language Syntax”.

UVM generation enables feedback from scoreboard to sequence

When generating a unified verification methodology (UVM) test bench, you can now use the Sequence Feedback block to implement a UVM connection from the scoreboard to the sequence.



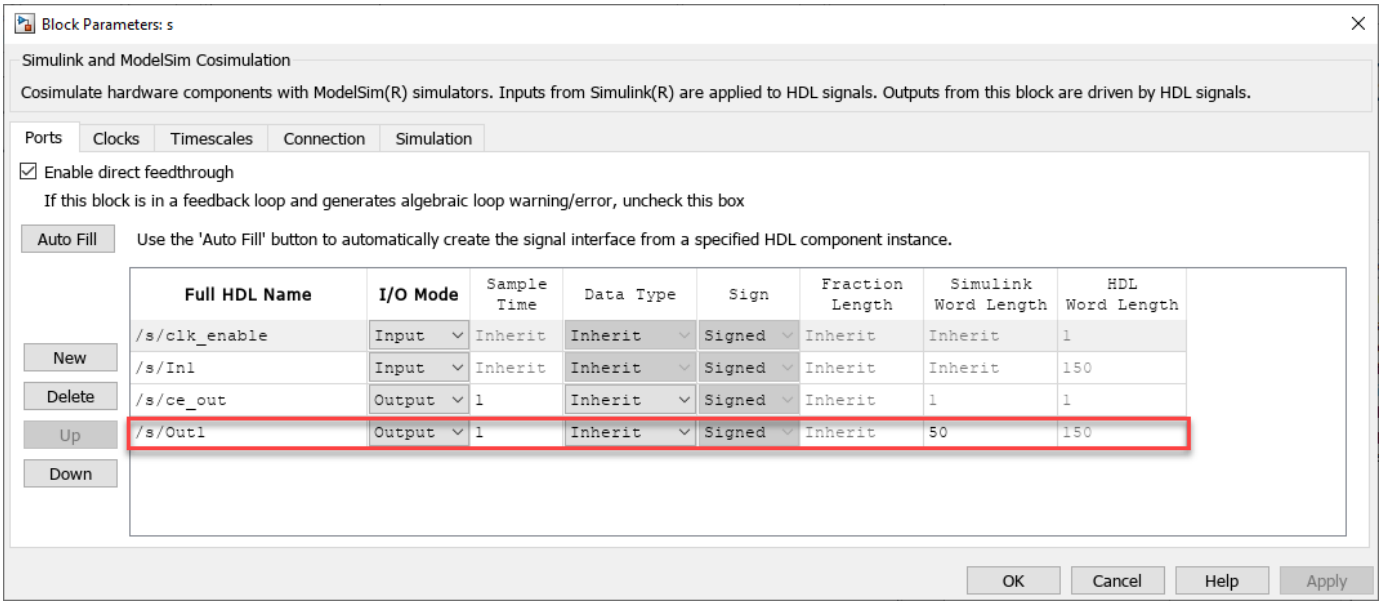
This feature allows you to tune sequence generation and control simulation according to run-time criteria calculated in the scoreboard.

For additional information about UVM, see “UVM Component Generation Overview”.

Simulink cosimulation supports port sizes greater than 128 bits

In previous releases, HDL Cosimulation blocks could support HDL designs with port sizes up to 128 bits. In this release, you can generate an HDL Cosimulation block for HDL designs with port sizes greater than 128 bits. HDL ports with data types wider than 128 bit map to a vector of Simulink® words.

You can now set the Simulink word length and observe the HDL word length (read-only) in the **Cosimulation Wizard** or in the HDL Cosimulation block mask.

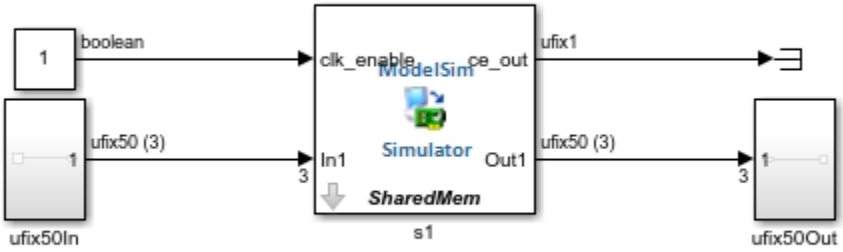


For example, consider this Verilog interface:

```
input   clk;
input   reset;
```

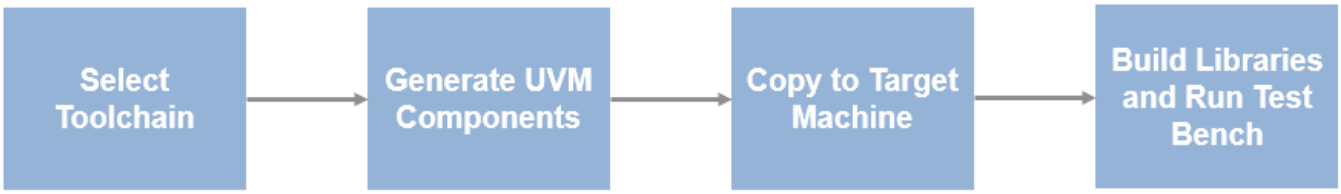
```
input   [149:0] In1;
output  [149:0] Out1;
```

If you set the Simulink word length to 50 bits, then `In1` and `Out1` are represented by Simulink ports with data type `ufix50(3)`.



Cross-platform support for UVM

You can now generate the UVM components from a Windows® operating system for building and running the component libraries on a Linux® operating system. When your target operating system is different from the host, you must select a target simulator and operating system.



Open your model, and on the **Apps** tab, click **HDL Verifier**. Select **DPI Component Generation** on the left pane, and on the **HDL Verifier** tab, click **C Code Settings**. The **Configuration Parameters** dialog opens on the **Code Generation** pane. Then, under **Build process**, select a target **Toolchain**. This option specifies the target simulator and operating system where you run simulations. You can then build the library on your target machine.

This feature only supports the `Cadence Xcelium (64-bit Linux)` cross-product toolchain. For more information, see “Generate Cross-Platform UVM Components”.

Cross-platform DPI generation workflows support Synopsys VCS and Vivado simulators

Port the SystemVerilog DPI component generated from a Windows operating system for building and running the component libraries on a Linux operating system.

Select cross-product toolchain to specify the target simulator and operating system where you run simulations. The workflows now support these cross-product toolchains.

- `Synopsys VCS (64-bit Linux)`
- `Xilinx Vivado Simulator (64-bit Linux)`

For more information about cross-platform DPI generation, for a Simulink workflow, see “Generate Cross-Platform DPI Components” and for MATLAB workflows, see “Port Generated Component and Test Bench to Linux”.

FPGA-in-the-Loop supports Versal VCK190 via Ethernet and JTAG

FPGA-in-the-Loop (FIL) now supports the Xilinx® Versal® AI Core Series VCK190 Evaluation Kit over both Ethernet and JTAG interfaces.

To execute an FIL simulation on the VCK190 board, execute one of the following workflows:

- Open the **FPGA-in-the-Loop Wizard**, and set **Board Name** to `Versal AI Core Series VCK190 Evaluation Kit`.
- Open **HDL Workflow Advisor** (requires HDL Coder™).
  1. Set **Target workflow** to `FPGA-in-the-Loop`
  2. Set **Target Platform** to `Versal AI Core Series VCK190 Evaluation Kit`

To set up the VCK190 board for Ethernet FIL:

1. Configure the network interface card in the host computer, and connect to the board.
2. Run `copyImageToHostSDCardPath` to copy the operating system image files to the SD card.
3. Insert the SD card to the SD card slot on the board.
4. Turn on the board for the system to boot.

FPGA data capture supports Versal ACAP devices via JTAG

FPGA data capture now supports Xilinx Versal ACAP devices over a JTAG interface. To run FPGA data capture on a Versal device, generate and integrate the data capture IP core for Versal into your FPGA design.

To generate a data capture IP core for a Versal device, follow these steps in the **FPGA Data Capture Component Generator** tool.

1. Set the **FPGA vendor** parameter to `Xilinx`.
2. Set the **Connection type** parameter to `JTAG`.

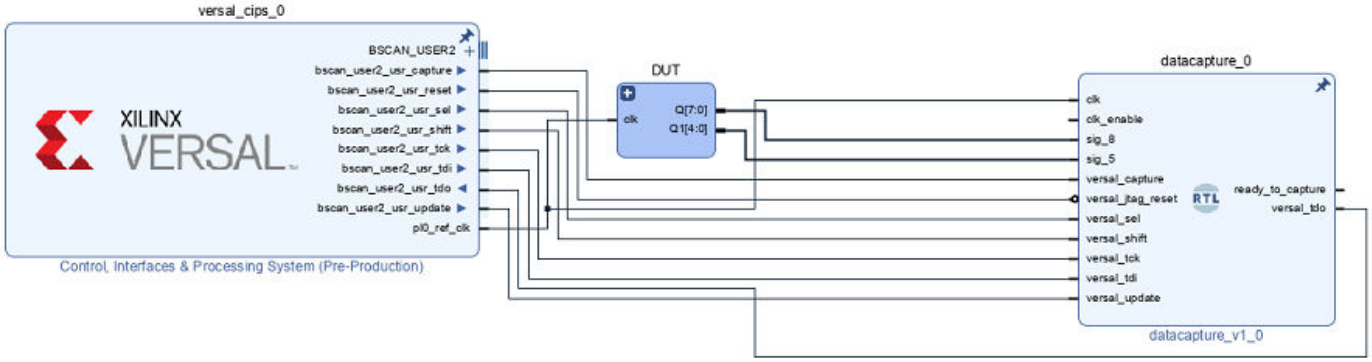
To integrate the generated data capture IP core for Versal into your FPGA design, follow these steps in the generation report.

1. Open your block design in Vivado®.
2. Navigate to the `hdlsrc` folder.
3. Insert the `datacapture` IP into your block design and connect the IP to the BSCAN_USER2 interface of the Xilinx Versal platform CIPS IP by executing this command in the Vivado Tcl console.

   ```
   source ./insertVersalFPGADataCaptureIP.tcl
   ```

To enable the BSCAN_USER2 interface, enable the PL BSCAN1 interface in the CIPS IP.

4 Complete the block design by connecting the `clk`, `clk_enable`, and input data ports of the data capture IP.



For more information, see “Data Capture Workflow”.

To use this feature, you must install the HDL Verifier Support Package for Xilinx FPGA Boards.

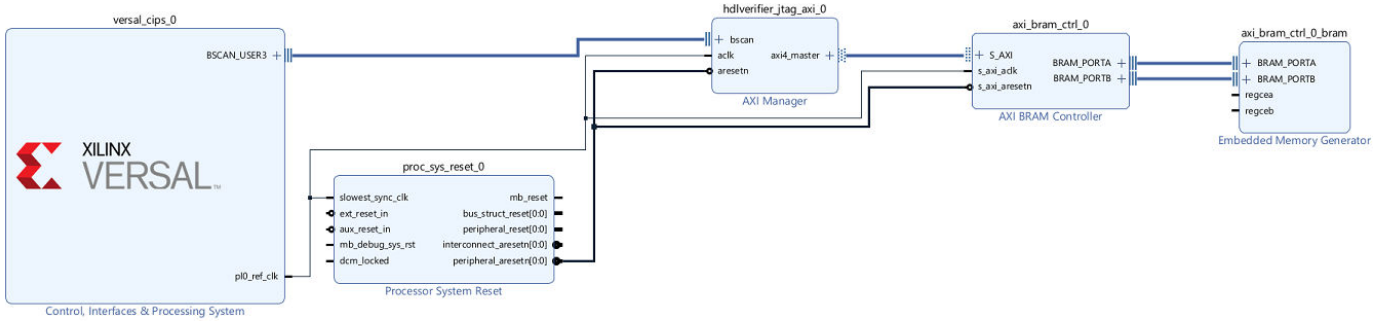
AXI manager supports Versal ACAP devices via JTAG

AXI manager now supports Xilinx Versal ACAP devices over a JTAG interface. To set up the AXI manager IP for the Versal devices, follow these steps:

1 Add the path for the AXI manager IP files to your Vivado project for the Versal device using the `setupAXIManagerForVivado` function.

2 Open Vivado and, from the IP Catalog, select the AXI Manager IP in your FPGA design.

3 Connect the AXI Manager IP to the BSCAN_USER3 interface of the Xilinx Versal platform CIPS IP. To enable the BSCAN_USER3 interface, enable the PL BSCAN2 interface in the CIPS IP.



For more information, see “JTAG AXI Manager”.

To use this feature, you must install the HDL Verifier Support Package for Xilinx FPGA Boards.

Half data type support for FPGA data capture and AXI manager

FPGA data capture and AXI manager add support for `half` data type. You can now observe signals with `half` data type from your FPGA design using FPGA data capture. You can also write and read `half` data to and from on-board memory locations using AXI manager.

The address for the write or read operation must refer to an AXI subordinate memory location controlled by the AXI manager IP on your FPGA board.

- If the AXI manager IP data width is 32 bits, the memory is 4 bytes aligned, and addresses have 4-byte increments (`0x0`, `0x4`, `0x8`).
- If the AXI manager IP data width is 64 bits, the memory is 8 bytes aligned, and addresses have 8-byte increments (`0x0`, `0x8`, `0x10`).

For more information, see AXI Manager Write and AXI Manager Read.

To use this feature, you must install the HDL Verifier Support Package for Intel® FPGA Boards or HDL Verifier Support Package for Xilinx FPGA Boards.

FPGA Data Capture Component Generator tool supports save and reload capability

Previously, the **FPGA Data Capture Component Generator** tool could save parameters of the most recent design only. Now, the tool saves design parameters into a MAT file for each successful run. You can choose and provide any of the saved MAT files as an input argument to the `generateFPGADataCaptureIP` function to reload your previously generated designs.

```
generateFPGADataCaptureIP('datacapture_gensettings.mat');
```

Where *`datacapture`* is the name of the generated HDL IP core.

To use this feature, you must install the HDL Verifier Support Package for Intel FPGA Boards or HDL Verifier Support Package for Xilinx FPGA Boards.

Applications: Vivado cosimulation, SystemVerilog Template Engine, CTLE verification

HDL Verifier adds these examples in the R2023a release:

- The “Cosimulate Vivado FFT IP Core with Simulink” example shows how to use Vivado cosimulation for simulating a Vivado IP core.
- The “Use Templates to Create SystemVerilog DPI and UVM Components” example shows how to generate a SystemVerilog DPI component using `dpigen` templates. The example then takes you though the steps of creating a UVM component from a MATLAB function using templates, and verifying the SystemVerilog component.
- The “Verify Standalone CTLE in Architectural, Behavioral, and Circuit Domains” shows how to simulate a continuous-time linear equalizer (CTLE) in several design domains. The example uses HDL Verifier to export the CTLE to SystemVerilog behavioral model, and feed the results back to the architectural model.
- The “Verify OFDM Transmit and Receive using FPGA Data Capture” example shows how to debug, visualize, and analyze the internal signals of a wireless HDL IP while you run the complete design on hardware. This example also shows how to probe the signals, deploy the design on hardware, and visualize the probed signals from the deployed design using **FPGA Data Capture**.

Additional board support for Hardware Setup tool

The **Hardware Setup** tool now supports these boards.

- Xilinx Versal AI Core Series VCK190 Evaluation Kit
- Avnet® MiniZed™
- Xilinx Kintex® UltraScale+™ FPGA KCU116 Evaluation Kit

For the full list of supported boards and interfaces, see “Supported FPGA Devices for FPGA Verification”.

Updates to supported software

The HDL Verifier product now supports these software versions. For a complete list of supported software, see “Supported EDA Tools and Hardware”.

- Xilinx Vivado 2022.1
- Intel Quartus® Prime Standard 21.1
- Mentor Graphics® Questa® Core/Prime 2022.2
- Mentor Graphics ModelSim® PE 2022.2
- Microchip Libero® SoC v12.6

Support package documentation moved into HDL Verifier documentation

HDL Verifier Support Package for Intel FPGA Boards updates

Starting in R2023a, the HDL Verifier Support Package for Intel FPGA Boards documentation is included in the HDL Verifier product documentation and all support package updates are announced in the HDL Verifier release notes. In previous releases, the support package documentation installs with the support package software. To access the release notes archive of previous support package releases, see Intel FPGA Boards R2022b Release Notes.

HDL Verifier Support Package for Microchip FPGA Boards updates

Starting in R2023a, the HDL Verifier Support Package for Microchip FPGA Boards documentation is included in the HDL Verifier product documentation and all support package updates are announced in the HDL Verifier release notes. In previous releases, the support package documentation installs with the support package software. To access the release notes archive of previous support package releases, see Microchip FPGA Boards R2022b Release Notes.

HDL Verifier Support Package for Xilinx FPGA Boards updates

Starting in R2023a, the HDL Verifier Support Package for Xilinx FPGA Boards documentation is included in the HDL Verifier product documentation and all support package updates are announced in the HDL Verifier release notes. In previous releases, the support package documentation installs with the support package software. To access the release notes archive of previous support package releases, see Xilinx FPGA Boards R2022b Release Notes.

Functionality being removed or changed

Changes to the config argument of the dpigen function
Still runs

Using the `config` argument with a `coder.config` object is not recommended. Use the new `svdpiConfiguration` object instead. This object adds support for using the “SystemVerilog and UVM Template Engine”, and allows generation of UVM components from MATLAB function in addition to SystemVerilog DPI components.

R2022b

Cosimulation supports HDL Coder workflows for Vivado simulator

When you perform HDL cosimulation using HDL Coder workflows, you now have the option to select Vivado simulator to cosimulate the generated HDL.

You can use this feature with Simulink or MATLAB in these scenarios:

- When using the `makehdltb` (HDL Coder) function to generate a test bench, set the `GenerateCosimModel` property to `Vivado Simulator`. For example:

  ```
  makehdltb('hdl_cosim_demo1/MAC',targetlang="vhdl",GenerateCosimModel="Vivado Simulator")
  ```

- When using the Simulink HDL Workflow Advisor, open the configuration parameters for your model, and on the left pane, expand **HDL Code Generation** and select **Test Bench**. Then set **Simulation tool** to `Xilinx Vivado Simulator`.
- When using MATLAB **HDL Coder** app, in the **Verify with Cosimulation** step, set the **Cosimulate for use with** parameter to `Xilinx Vivado Simulator`.

This feature requires an HDL Coder license.

Generate cosimulation artifacts using command-line interface

You can now generate a cosimulation block or System object™ from the MATLAB command line. In previous releases, you had to use the **Cosimulation Wizard** to generate cosimulation artifacts. Now, use the `cosimulationConfiguration` configuration object to specify:

- HDL simulator – ModelSim, Xcelium™, or Vivado simulator
- Workflow – MATLAB System object or Simulink
- Top level HDL module

You can optionally set additional properties for port interfaces, compilation and simulation flags, simulation timing considerations, and more.

To generate the cosimulation block or system object, use the `runWorkflow` object function from the command line. For example, create a configuration object and use the automatically determined values for port lists, port attributes, and HDL timing for generation. Set the location of the HDL files to `./source`, and create the HDL Cosimulation block. This command-line input is equivalent to clicking through the steps in the **Cosimulation Wizard** without altering the default values.

```
c = cosimulationConfiguration('ModelSim','Simulink','mytop');
c.HDLFiles = './source'
runWorkflow(c);
```

To see an example of running a Simulink cosimulation with the command-line workflow see Command-Line Workflow for Verifying Raised Cosine Filter in Simulink

Generate cosimulation artifacts from saved configuration

When you use the **Cosimulation Wizard** or `runWorkflow` function to configure and generate an HDL Cosimulation block or System object, the workflow now saves a configuration file as a `.mat` file. Use the `cosimulationConfiguration` object to generate cosimulation artifacts based on a previous execution of the **Cosimulation Wizard** or `runWorkflow` function.

```
c = cosimulationConfiguration('./mygoodrun/cosimWizard_mytop.mat');
runWorkflow(c);
```

Cosimulation supports additional SystemVerilog types

When using a SystemVerilog design for cosimulation, you can now map the following SystemVerilog types to Simulink or MATLAB types:

SystemVerilog Data Type	MATLAB or Simulink Data Type
`wire`	`ufix1`
`reg/logic`	`ufix1`
`bit`	`boolean/ufix1`
`byte`	`int8/uint8`
`shortint`	`int16/uint16`
`int`	`int32`
`longint`	`int64/uint64`
`real`	`double`
`packed array`	• Input to HDL Cosimulation block: `ufix/fix`, or matrix of `ufix/fix` • Output from HDL Cosimulation block: `ufix`[a]

a For currently supported packed array input or output to the HDL Cosimulation block, the total number of bits in the packed array has a 128-bit upper limit.

For supported data types for cosimulation, see Supported Data Types.

FPGA-in-the-loop supports Ethernet connection for Zynq boards

In previous releases, you could use only JTAG for FPGA-in-the-loop (FIL) on Zynq® SoC boards . This release you can perform FIL using an Ethernet connection. This feature enables better performance for FIL simulation. It allows you to run FIL over a network without connecting the board directly to the host. You can also run FIL while using the JTAG connection for other features.

For a list of supported boards, see Supported FPGA Devices for FPGA Verification. For information about hardware setup, see Guided Hardware Setup.

Hardware Setup tool supports Ethernet connection for Zynq boards

The **Hardware Setup** (HDL Verifier Support Package for Xilinx FPGA Boards) tool, which provides an interface for setting up hardware boards for use with the HDL Verifier product, now supports an Ethernet connection for the supported Xilinx Zynq SoC boards. After you select your Zynq SoC board name and the Ethernet interface, the tool guides you to:

1. Configure the network card on the host computer
2. Set up the SD card
3. Set the jumper switches
4. Connect to the hardware

The **Hardware Setup** tool provides an automated setup process. Use this process to configure the target hardware board for use with FPGA-in-the-loop (FIL), FPGA data capture, and AXI manager over the JTAG, Ethernet, and PCI Express® interfaces. For more information, see Guided Hardware Setup (HDL Verifier Support Package for Xilinx FPGA Boards).

UVM generation enables stopping simulation from UVM sequence

When generating a Unified Verification Methodology (UVM) test bench, you can now use the Stop Simulation (Simulink) block in the Sequence subsystem. Set a Boolean condition in the block to stop the simulation. This behavior is reflected in the generated UVM test bench.

For additional information about UVM, see UVM Component Generation Overview.

Cross-platform support for DPI generation for MATLAB code

You can now port the SystemVerilog DPI component generated for a MATLAB function and an optional test bench from a Windows operating system to a Linux operating system with easier steps. After you generate the component, use the `packNGo` (MATLAB Coder) function to package the generated files and required dependencies before copying them to the target machine.

To use your generated component, you must copy the generated package file to the Linux target machine and unzip it. You no longer need to create a generic makefile script and build the shared library on the Linux target machine to use the generated component. If you generate a test bench to exercise the generated component, you no longer need to copy the test bench folder from the Windows host machine to the Linux target machine separately.

This feature supports the Mentor Graphics ModelSim, Mentor Graphics Questa, and Cadence® Xcelium HDL simulators. For more information, see Port Generated Component and Test Bench to Linux.

Updates to supported software

The HDL Verifier product now supports these software versions. For a complete list of supported software, see Supported EDA Tools and Hardware.

- Cadence Xcelium 2021.09 with libraries `gcc48`, `gcc63`.
- Mentor Graphics Questa Core/Prime 2021.4 with libraries `gcc474`, `gcc530`, `gcc740`
- Mentor Graphics ModelSim PE 2021.4 with libraries `gcc474`, `gcc530`, `gcc740`
- Synopsys® VCS® MX vS-2021.09-1
- Intel Quartus Prime Pro 21.3 (supported for Intel Cyclone® 10 GX only)

For more information about supported library versions, see Cosimulation Libraries.

Functionality being removed or changed

Microsemi renamed to Microchip

Behavior change

Microsemi™ has been renamed to Microchip. In the software and documentation, the term Microchip replaces Microsemi.

The support package name HDL Verifier Support Package for Microsemi FPGA Boards has been changed to HDL Verifier Support Package for Microchip FPGA Boards. For more information on this support package, see HDL Verifier Support Package for Microchip FPGA Boards .

R2022a

FPGA-in-the-loop enables hardware buffering

Use hardware buffering to improve FPGA-in-the-Loop simulation performance. This feature utilizes BRAMs on the FPGA to buffer Ethernet packets in frame-based processing mode.

To enable this feature when using the **FPGA-in-the-Loop Wizard**, select **Enable data buffering on FPGA** under **Advanced Options**.

To enable this feature when using **HDL Workflow Advisor** (requires HDL Coder), select **Enable data buffering on FPGA** in step 4.1.

HDL Cosimulation adds support for Vivado simulator

You can now use the Vivado simulator when cosimulating an HDL design with a Simulink or MATLAB test bench.

To create an HDL Cosimulation block for the Vivado simulator, open the **Cosimulation Wizard**. In the **Cosimulation Type** step:

- Set **HDL Simulator** to `Vivado Simulator`.
- Set **HDL Cosimulation with** to `Simulink`.

Use the generated block in a Simulink test bench.

To create an `hdlverifier.HDLCosimulation` System object for the Vivado simulator, open the **Cosimulation Wizard**. In the **Cosimulation Type** step:

- Set **HDL Simulator** to `Vivado Simulator`.
- Set **HDL Cosimulation with** to `MATLAB System Object`.

Use the generated System object in a MATLAB test bench.

Cosimulation supports Verilog parameters and VHDL generics

You can now configure Verilog or VHDL parameters in a cosimulation. When you use the **Cosimulation Wizard** to generate an HDL Cosimulation block or an `hdlverifier.HDLCosimulation` System object, the **Simulation Options** step creates a configuration file named `parameter_`*`DUT`*`.cfg`, where *`DUT`* is the name of your HDL DUT. The configuration file includes a line for each HDL parameter, with a default value assigned. Uncomment the line for the parameter you want to configure and assign a value to override the default value.

For more information, see Use HDL Parameters in Cosimulation.

Cosimulation saves path in Tcl variable

When you use the **Cosimulation Wizard** to generate an HDL Cosimulation block or an `hdlverifier.HDLCosimulation` System object, the **HDL Compilation** step creates a tool command language (Tcl) script with compilation commands. In this script, it saves the path to the location of the HDL files in a Tcl variable (or multiple variables for multiple HDL folders). With this feature, if the location of the HDL files change, you only need to change the definition of the Tcl

variable. Previously, if the location of the HDL files changed, you had to change the full path for each HDL file in the script.

DPI component generation supports combinational designs

You can now generate a DPI component for a combinational Simulink subsystem or MATLAB function so that outputs immediately reflect changes in the inputs.

To generate a combinational SystemVerilog DPI component for a Simulink model, open the configuration parameters, select the **SystemVerilog DPI** tab on the left, and set **Component template type** to `Combinational`.

To generate a combinational SystemVerilog DPI component for a MATLAB function, use the `dpigen` function with the new `-ComponentTemplateType` argument set to `Combinational`. An example syntax follows.

```
dpigen myCombFunction -args {0,0} -ComponentTemplateType Combinational
```

UVM generation creates new sequencer.sv file

In R2022a, the `uvmbuild` function generates separate SystemVerilog files for the sequencer and sequence classes. The file names are `mw_`*`DUT`*`_sequence.sv` and `mw_`*`DUT`*`_sequence.sv`, respectively. Previously, the function generated a single file that included both sequencer and sequence classes.

For more information about the generated UVM files and folder structure, see Generated Files and Folder Structure.

Simultaneous use of FPGA data capture and AXI manager

The new nonblocking mode enables you to simultaneously use FPGA data capture and AXI manager, which share a common JTAG interface. Previously, you had to close or release the JTAG resource to switch between FPGA data capture and AXI manager.

FPGA data capture now supports these two capture modes.

- Blocking mode — FPGA data capture blocks MATLAB while retrieving captured data. Use this mode for backward compatibility.
- Nonblocking mode — FPGA data capture does not block MATLAB while retrieving captured data.

By default, FPGA data capture uses blocking mode and the JTAG resource is allocated to either FPGA data capture or AXI manager at a time. In nonblocking mode, you can use FPGA data capture and AXI manager simultaneously. Create a data capture object and then change the capture mode to nonblocking mode by entering these commands at the MATLAB command prompt.

1. Create a data capture object, `dco`, that captures data from a design running on an FPGA. `datacapture` is the generated IP name you specified in the **FPGA Data Capture Component Generator** tool.

   ```
   dco = datacapture;
   ```

2. Change the capture mode to nonblocking mode.

   ```
   dco.CaptureMode = 'nonblocking';
   ```

For an example, see Debug IP Core Using FPGA Data Capture (HDL Coder).

For more information on FPGA data capture and AXI manager, see FPGA Data Capture and AXI Manager, respectively.

FPGA data capture supports capture condition logic

You can now filter the data to capture using the capture condition logic. Include the capture condition logic in your HDL IP core when you want to:

- Capture only the valid data to debug custom designs with FPGA data capture.
- Filter the data to capture based on trigger conditions.
- Optimize the use of FPGA data capture buffer.
- Efficiently analyze the captured data when you have only a few captured samples of interest.

To include the capture condition logic in your HDL IP core, select the **Include capture condition logic** parameter while generating the IP core using the **FPGA Data Capture Component Generator** tool. Then set up a capture condition to control when to capture the data that satisfies the trigger conditions in the **FPGA Data Capture** tool, the `hdlverifier.FPGADataReader` System object, or the FPGA Data Reader block.

Automated workflow to access memory-mapped locations on FPGA using HDL Workflow Advisor

Use the **HDL Workflow Advisor** tool to generate a host interface model. The host interface model enables you to write and read from the memory-mapped locations on the target hardware over a JTAG cable by using the AXI Manager Write and AXI Manager Read blocks. To create a host interface model, follow these steps in the **HDL Workflow Advisor** tool.

- In step **1.1. Set Target Device and Synthesis Tool**, set **Target workflow** to `IP Core Generation`.
- In step **1.2. Set Target Reference Design**, set **Insert JTAG AXI Manager (HDL Verifier required)** to `on`.
- In step **1.3. Set Target Interface**, map each DUT signal that you want to capture to the `AXI4` or `AXI4-Lite` interfaces.
- In step **4.2. Generate Software Interface**, set **Host target interface** to `JTAG AXI Manager (HDL Verifier)`.
- In step **4.2. Generate Software Interface**, generate the host interface model by selecting **Generate host interface model**.

For an example, see Use JTAG AXI Manager to Control HDL Coder Generated IP Core (HDL Coder).

This feature requires the HDL Coder product.

Updates to supported software

The HDL Verifier product now supports these software versions. For a full list of supported software, see Supported EDA Tools and Hardware.

- Intel Quartus Prime Standard 20.1.1
- Mentor Graphics Questa Core/Prime 2021.2
- Xilinx Vivado 2020.2

Functionality being removed or changed

MATLAB AXI master renamed to AXI manager

Warns

MATLAB AXI master has been renamed to AXI manager. In the software and documentation, the terms "manager" and "subordinate" replace "master" and "slave," respectively.

These changes require updates to your code.

Type	Old Name	New Name	Recommendation
Functions	`setupAXIMasterForQuartus`	`setupAXIManagerForQuartus`	`setupAXIMasterForQuartus` will be removed in a future release. Use `setupAXIManagerForQuartus` instead. Replace all instances of `setupAXIMasterForQuartus` with `setupAXIManagerForQuartus`.
	`setupAXIMasterForVivado`	`setupAXIManagerForVivado`	`setupAXIMasterForVivado` will be removed in a future release. Use `setupAXIManagerForVivado` instead. Replace all instances of `setupAXIMasterForVivado` with `setupAXIManagerForVivado`.
Objects	`aximaster`	`aximanager`	`aximaster` will be removed in a future release. Use `aximanager` instead. Replace all instances of `aximaster` with `aximanager`.
Blocks	AXI Master Read	AXI Manager Read	The AXI Master Read and AXI Master Write blocks have been removed. Use the AXI Manager Read and AXI Manager Write blocks instead. In R2022a, you cannot use a Simulink model that contains the AXI Master Read or AXI Master Write blocks. Recreate your model in R2022a by using the AXI Manager Read and AXI Manager Write blocks.
	AXI Master Write	AXI Manager Write	
AXI manager IPs	MATLAB as AXI Master	AXI Manager	The AXI Manager IP is upgraded from version 1.1 to 2.0. The UDP AXI Manager IP and PCIe AXI Manager IP are upgraded from version 1.0 to 2.0.
	UDP MATLAB as AXI Master	UDP AXI Manager	

Type	Old Name	New Name	Recommendation
	PCIe MATLAB as AXI Master	PCIe AXI Manager	In R2022a, you cannot use an AXI manager hardware design created in a previous release. Recreate your hardware design in R2022a.

For more information on AXI manager, see AXI Manager.

R2021b

Version: 6.4

New Features

Bug Fixes

DPI generation supports variable-sized vectors

When you generate a SystemVerilog DPI (SVDPI) component from a MATLAB function, the function can now include an input of a variable-sized vector. To specify a supported toolchain, use a `coder.config` (MATLAB Coder) object.

For example, when generating an SVDPI component for the following function:

```
function[y,vsum] = dynamic_allocate(vector_n,amp,freq)
#codegen
y = int16(vector_n);
for i = 1:length(vector_n)
    y(i) = amp*sin(i*2*pi/(freq));
end
vsum = sum(vector_n);
```

If you specify `vector_n` as a variable-sized vector during code generation, the generated SystemVerilog includes the variable-sized input (`vector_n`) and output (`y`). The data type for output `y` is derived from the input data type. These variable-sized ports are declared as a SystemVerilog open array (`[ ]`). This code shows the generated interface for the above function, when `y` and `vector_n` are specified as variable-sized.

```
module dynamic_allocate_dpi(
    input bit clk,
    input bit clk_en,
    input bit reset,
    input real vector_n [],
    input real amp,
    input real freq,

    output shortint y [],
    output real vsum
);
```

Provide an input vector when you use the generated SystemVerilog module in a test-bench or top-level module.

For an example showing the use of variable-sized vectors, see Use Variable-Sized Vector in SystemVerilog DPI Component.

DPI generation supports complex data types for test points

When you generate SystemVerilog DPI components, you can now select signals of data type complex as test points for logging. For more information about test-point access for DPI components, see SystemVerilog DPI Component Test Point Access.

Extended support for FTDI USB-JTAG cable

Previously, FTDI USB-JTAG support was available for only MATLAB AXI master and FPGA data capture for Windows operating systems. Now, the HDL Verifier product supports an FTDI USB-JTAG connection for FPGA-in-the-loop, MATLAB AXI master, and FPGA data capture for Windows and Linux operating systems.

For more information, see Supported EDA Tools and Hardware.

MATLAB AXI master support for MII and SGMII interfaces for Xilinx boards

Ethernet MATLAB AXI master now supports MII and SGMII interfaces for Xilinx boards. With the addition of these two interfaces, you can use MATLAB AXI master over GMII, MII, and SGMII interfaces (for Xilinx boards only). For additional information, see Ethernet MATLAB AXI Master (HDL Verifier Support Package for Xilinx FPGA Boards).

To use this feature, you must install the HDL Verifier Support Package for Xilinx FPGA Boards. To access supported hardware for the HDL Verifier product, see HDL Verifier Supported Hardware.

FPGA data capture support for Ethernet connection for Xilinx boards

You can now capture signal data from a live FPGA over an Ethernet connection for Xilinx boards. This feature provides faster performance compared to FPGA data capture over a JTAG connection. To run FPGA data capture over an Ethernet interface, generate a data capture IP core for the Ethernet connection. Then, integrate the generated IP core into your FPGA design.

- To generate a data capture IP core for an Ethernet connection, follow these steps in the **FPGA Data Capture Component Generator** (HDL Verifier Support Package for Xilinx FPGA Boards) tool.

  1. In the **Target** section, set the **Connection type** parameter to `Ethernet`.
  2. In the **Ethernet settings** section, specify the **IP address** and **Port address** parameters for your target device and set the **Interface type** parameter to `GMII`, `MII`, or `SGMII` based on the interface for your target device.
- To integrate the generated HDL IP core into your FPGA design, follow these steps in the generation report.

  1. Create a Vivado project.
  2. Navigate to the `hdlsrc` folder.
  3. Run the `insertEthernet.tcl` script in the Vivado Tcl console by entering the `source ./insertEthernet.tcl` command.

For more information, see Data Capture Workflow (HDL Verifier Support Package for Xilinx FPGA Boards).

To use this feature, you must install the HDL Verifier Support Package for Xilinx FPGA Boards. To access supported hardware for the HDL Verifier product, see HDL Verifier Supported Hardware.

Additional board support

FPGA board additions for this release include:

- This board supports FPGA-in-the-loop, MATLAB AXI master, and FPGA data capture.
  - Xilinx Zynq UltraScale+ RFSoC ZCU216 Evaluation Kit
- Previously, these boards supported MATLAB AXI master and FPGA data capture. Now, these boards also support FPGA-in-the-loop.
  - Xilinx Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit

- Xilinx Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit
- Xilinx Zynq UltraScale+ RFSoC ZCU111 Evaluation Kit

For a full list of supported boards, see Supported FPGA Devices for FPGA Verification.

Updates to supported software

The HDL Verifier product now supports these software versions. For a full list of supported software, see Supported EDA Tools and Hardware.

- Intel Quartus Prime Pro 20.2 (for use with Intel Cyclone 10 GX only)
- Mentor Graphics Questa Core/Prime 2020.4

R2021a

Generate UVM predictor from Simulink subsystem

Generate a Universal Verification Methodology (UVM) predictor module from a Simulink model. To generate a predictor module, use the `uvmbuild` function, specifying the `'Predictor'` name-value argument. The predictor includes a behavioral DUT and serves as a golden model for the scoreboard. See UVM Component Generation Overview.

Collect functional coverage data from assertion blocks

SystemVerilog DPI and UVM generation workflows now support function coverage for assertion blocks from the `Simulink/Model Verification` library. You can gather functional coverage for untested, failed, and passed assertions. After you simulate your model, the HDL simulator prints coverage information from these blocks to the console, and the information is available for interactive analysis in the simulator coverage tool. For more information, see Generate SystemVerilog Assertions and Functional Coverage.

UVM Generation supports new directory structure

When you generate UVM components using the `uvmbuild` function, you can now specify a directory for the generated files. To specify the output directory of generated files, set the `'buildDirectory'` property of the `uvmcodegen.uvmconfig` configuration object. Then, specify the object as an argument for the `uvmbuild` function. For example, to place all generated files in the directory `\temp\DPI_model`, enter these commands at the MATLAB command prompt.

```
cfgObj = uvmcodegen.uvmconfig('buildDirectory','\temp\DPI_model');
uvmbuild(design,sequence,scoreboard,'config',cfgObj)
```

When you do not specify a build directory, the top directory is named `uvm_build`. The `uvm_build` directory consists of these two directories:

- *`model_name`*`_dpi_components` – *`model_name`* is the name of the top Simulink model. This directory includes all generated DPI components.
- *`model_name`*`_uvm_testbench` – *`model_name`* is the name of the top Simulink model. This directory includes the UVM testbench, the generated DUT, and shared libraries.

Preserve arrays or set to scalars on SystemVerilog DPI interface

When you generate Universal Verification Methodology (UVM) or SystemVerilog DPI components, you can now create scalar ports in the generated SystemVerilog interface when the Simulink data type is an array or matrix. In the Configuration Parameters dialog box, select **SystemVerilog DPI** in the left pane. Then, under **SystemVerilog Ports**, click **Scalarize matrix and vector ports**.

To match the interface with the interface generated by HDL Coder, set the configuration parameters to use scalarized, flattened, logic data types.

Use composite data types for tunable parameters

When you generate a SystemVerilog DPI (SV-DPI) component with a tunable parameter, you can now include `Enum`, `Complex`, and `Nonvirtual Bus` data types. For information about creating a tunable parameter with SystemVerilog DPI, see Tune Gain Parameter During Simulation.

Simulink Toolstrip support for cosimulation workflow

You can now use the Simulink Toolstrip to follow the workflow for HDL cosimulation with Simulink. Access the **HDL Verifier** tab by opening the **HDL Verifier** app from the **Apps** tab on the Simulink Toolstrip. Then, on the **HDL Verifier Mode** pane, select **HDL Cosimulation**.

FPGA Data Capture in HDL Workflow Advisor supports sequential trigger

When you use FPGA Data Capture in HDL Workflow Advisor, you can now provide a sequence of trigger conditions at multiple stages to read data from an FPGA. To enable this feature, specify the maximum number of trigger stages as a value greater than 1 for the **FPGA Data Capture maximum sequence depth** parameter in step 3.2 **Generate RTL Code and IP Core** of HDL Workflow Advisor. For more information on capturing data, see Data Capture Workflow.

To use this feature, you must install the HDL Verifier Support Package for Xilinx or Intel FPGA boards. To access supported hardware for the HDL Verifier product, see HDL Verifier Supported Hardware. This feature requires the HDL Coder product.

FPGA Data Capture integration with IP Core Generation workflow for generic Xilinx and generic Intel targets

The generic Xilinx platform and the generic Intel platform now support FPGA Data Capture in the IP Core Generation workflow of HDL Workflow Advisor. For more information on the IP Core Generation workflow, see IP Core Generation (HDL Coder).

To use this feature, you must install the HDL Verifier Support Package for Xilinx or Intel FPGA boards. To access supported hardware for the HDL Verifier product, see HDL Verifier Supported Hardware. This feature requires the HDL Coder product.

Updates to supported software

The HDL Verifier product now supports this software version. For a full list of supported software, see Supported EDA Tools and Hardware.

- Xilinx Vivado 2020.1

Functionality being removed or changed

UVM generation requires valid subsystem names

Behavior change

When you generate a UVM test bench, the names of the subsystems in your Simulink model must be valid SystemVerilog and C identifiers. Subsystem names must start with a letter and must use a combination of alphanumeric characters and underscores.

If a subsystem name does not meet these requirements, the `uvmbuild` function errors.

R2020b

Version: 6.2

New Features

Compatibility Considerations

Generate UVM driver or monitor from Simulink subsystems

You can now generate a Universal Verification Methodology (UVM) driver or monitor from your Simulink model. For the `uvmbuild` function, specify the `'Driver'` or `'Monitor'` name-value pair argument.

For more information, see UVM Component Generation Overview.

Customize generated UVM files

Configure generated UVM files to include a custom `` `timescale `` directive by using the `uvmcodegen.uvmconfig` configuration object.

Customize generated UVM files to include a custom header by modifying the description section of your Simulink top module or model subsystems.

For more information about customizing UVM files, see Customize Generated UVM Code.

UVM and SystemVerilog support for Simulink built-in port types to logic and bit types

When generating UVM or SystemVerilog DPI components, control the data type of the SystemVerilog ports by selecting `compatible C type`, `logic vector`, or `bit vector` for **Ports data type** in the Configuration Parameters.

For UVM supported data types see Supported Simulink Data Types.

For SystemVerilog DPI supported data types in Simulink see Supported Simulink Data Types.

For SystemVerilog DPI supported data types in MATLAB see Supported MATLAB Data Types.

Compatibility Considerations

In the Configuration Parameters dialog box, in the **SystemVerilog DPI** section, the **Fixed-point data type** parameter has been renamed **Ports data type** to reflect that the data type selection affects all ports.

In the `dpigen` function, the `FixedpointDataType` name-value pair argument has been renamed `PortsDataType`.

Export run time error into SystemVerilog environment

You can now export run-time errors from a Simulink execution into the DPI and UVM simulation environment. From Simulink, open the Configuration Parameters dialog box, select **SystemVerilog DPI** on the left pane, and under **Run-time Error Reporting**, select **Report run-time error**. You can also set the simulation behavior by setting **Severity** parameter to `Info`, `Warning`, `Error`, or `Fatal`.

HDL Verifier contextual tab on Simulink Toolstrip

The Simulink Toolstrip now includes a contextual tab for HDL Verifier features and workflows. To access the **HDL Verifier** tab, open the **HDL Verifier** app from the **Apps** tab on the Simulink Toolstrip. The tab supports access to DPI component generation workflows.

For more information, see “Simulink Toolstrip: Access and discover Simulink capabilities when you need them” (Simulink).

Generate functional coverage report for verify statements

SystemVerilog DPI and UVM generation workflows now support functional coverage. When utilizing `verify` statements in your Simulink test bench model, you can now gather functional coverage for a passing result during SystemVerilog simulation. After the simulation completes, the coverage information is printed to the console and is available for interactive analysis in the simulator coverage tool. The default coverage goal is at least one *pass* for each `verify` statement, but you can override the coverage goal to be higher than one or filter coverage altogether using a command line argument (`plusarg`).

This capability exists for any subsystem that utilizes `verify` statements in Test Assessment blocks, Test Sequence blocks, and Stateflow® charts. This feature generates coverage for DPI component generation and UVM test bench generation workflows.

MATLAB as AXI Master adds simulation model for Xilinx

Simulate memory read and write operations for the MathWorks® AXI Master IP core using the Vivado Simulator. Use the provided `readmemory` and `writememory` SystemVerilog tasks to simulate memory accesses to IP core registers or external memory before deploying your design to the FPGA. Once you deploy your design, use the `aximaster` object functions `readmemory` and `writememory` to access memory from MATLAB to hardware. For more information, see MATLAB as AXI Master (HDL Verifier Support Package for Xilinx FPGA Boards).

For an example that uses this feature, see Access FPGA External Memory Using MATLAB as AXI Master (HDL Verifier Support Package for Xilinx FPGA Boards).

To use this feature, you must install the HDL Verifier Support Package for Xilinx FPGA Boards.

FPGA Data Capture supports sequential trigger

Read data from an FPGA by providing a sequence of trigger conditions at various stages. Previously, FPGA Data Capture enabled you to provide only one trigger condition. Now, you can define a set of one or more trigger conditions. To enable this feature, specify the maximum number of trigger stages to a value greater than 1 in the **FPGA Data Capture Component Generator** tool and specify the required number of trigger stages in the **FPGA Data Capture** tool.

To use this feature, you must install the HDL Verifier Support Package for Xilinx or Intel FPGA boards. To access supported hardware for HDL Verifier, see HDL Verifier Supported Hardware.

Updates to supported software

HDL Verifier now supports these software versions. For a full list of supported software, see Supported EDA Tools and Hardware.

- Xilinx Vivado 2019.2
- Intel Quartus Prime Pro 19.4
- Mentor Graphics ModelSim PE 2019.4
- Mentor Graphics Questa Core/Prime 2019.4

Functionality being removed or changed

Configuration parameter renamed

In the Configuration Parameters dialog box, in the **SystemVerilog DPI** section, the **Fixed-point data type** parameter has been renamed **Ports data type** to reflect that the data type selection affects all ports.

When using the `set_param` function to set a configuration parameter, use the `FixedpointDataType` name-value pair argument. For example:

```
set_param(sys, 'DPIFixedPointDataType','CompatibleCType');
```

dpigen function input renamed

In the `dpigen` function, the `FixedpointDataType` name-value pair argument has been renamed `PortsDataType`.

R2020a

Version: 6.1

New Features

Bug Fixes

UVM support for tunable parameters

You can now use tunable parameters when generating UVM components from Simulink. This feature enables you to create generic Sequence or Scoreboard units and use them with different parameter values to either obtain different stimulus or perform various checks.

Writing generic, parametrized Simulink subsystems enables you to reuse the generated UVM test bench across different scenarios. If the Sequence subsystem contains tunable parameters, then the generated UVM sequence contains random constraints that you can override in the UVM test bench. If the Scoreboard subsystem contains tunable parameters, then the generated UVM scoreboard has an associated configuration object that you can use to tune parameters at compile time or run time. See Tunable Parameters in Sequence Subsystem, Tunable Parameters in Scoreboard Subsystem.

UVM support for Simulink nonvirtual bus, complex, and enum data types

Generate a SystemVerilog `struct` data type from a `nonvirtual bus` or `complex` data type. Choose between a `struct` or flattened data type when generating a SystemVerilog DPI component. To create a SystemVerilog `struct` data type, open the Configuration Parameters dialog box, select **SystemVerilog DPI** on the left pane, and set **Composite data type** to `Structure`.

Generate a SystemVerilog `enum` data type from a Simulink `enumerated` data type. If a subsystem contains enumerated data types, the generated UVM component has a SystemVerilog `enum` data type in its interface. For supported data types for UVM, see Supported Simulink Data Types.

MATLAB as AXI Master supports Zynq devices via Ethernet

MATLAB as AXI Master now supports AXI4 read and write operations over an Ethernet connection for these Zynq devices.

- Zynq-7000 ZC706
- ZedBoard™

Improved Data Capture performance with Logic Analyzer

Data capture performance using a data capture System object is now up to 500 times faster when you set the **Number of capture windows** property to 4 or greater. The performance varies depending on the number of signals, their data types, the capture depth, and the number of capture windows used.

The **Logic Analyzer** app display is enhanced to replace cursors with signals to show the data-capture window number and trigger position for each window. This change allows for any number of capture windows (up to the capture depth), enables a faster load time, and provides a more compact view when multiple capture windows are present.

FPGA Data Capture supports bit masking

FPGA Data Capture now supports bit masking when specifying the trigger condition. In the **FPGA Data Capture** app or the FPGA Data Reader block, specify `X` or `x` (don't-care notation) when configuring a trigger condition to mask desired bits in the trigger signal. This feature allows you to set a trigger condition based on a portion of the signal. For example, to set a trigger condition when the eight most significant bits of a 32-bit address are `'0xFF'`, use this expression.

```
address == '0xFFXX_XXXX'
```

Updates to supported software

HDL Verifier now supports these software versions. For a full list of supported software, see Supported EDA Tools and Hardware.

- Xilinx Vivado 2019.1
- Intel Quartus Prime Pro 19.2
- Microsemi Libero SoC v12.0

R2019b

Version: 6.0

New Features

Bug Fixes

Generate UVM components from Simulink model

You can now generate a Universal Verification Methodology (UVM) test bench from your Simulink model. Use the `uvmbuild` function to map your Simulink model to a UVM test bench that includes a sequence, a sequencer, a scoreboard, monitors, drivers, and a behavioral device under test (DUT). For additional information, see UVM Component Generation Overview.

FPGA Data-Capture supports additional logical operators as triggers

When specifying a trigger in the **Data Capture** tool or in the FPGA Data Reader block, you can now use these comparison operators: ==, !=, <, >, <= and >=.

This enhancement enables greater flexibility when defining a trigger. Use these operators to create triggers within value ranges. To specify a trigger, first enter `launchDataCaptureApp` at the MATLAB command prompt to open the **Data Capture** tool. Then, on the **Trigger** tab, select the desired operator from the **Operator** list.

Cosim Wizard and FIL Wizard apps are accessible from Simulink toolstrip

You can now access the **Cosim Wizard** and **FIL Wizard** apps from the Simulink toolstrip. To open either of these apps, on the **Apps** tab, under **Verification, Validation and Test**, click the **Cosim Wizard** or **FIL Wizard** icon.

Simulink support for AXI Master

You can now read and write memory-mapped locations on your FPGA by using the AXI Master Write and AXI Master Read Simulink blocks.

Extended support for parameters in IP-XACT File

When importing an IP-XACT file, you can specify port mapping of a Transaction Level Model in a `spirit:vendorExtensions` tag. For additional information, see Mapping to a Signal Port.

Extended support for memory mapping of address blocks in IP-XACT file

When generating a TLM component, you now have finer granularity for specifying what portion of the parameters are mapped to memory. You can now exclude an address block or a register within a block by setting the `MWMap` parameter to `false` in the IP-XACT file.

This example shows how to exclude a register named `DUMMY_REG_0` from the generated TLM component.

```
<spirit:register>
      <spirit:name>DUMMY_REG_0</spirit:name>
      <spirit:addressOffset>0x10</spirit:addressOffset>
      <spirit:size>64</spirit:size>
      <spirit:access>read-only</spirit:access>
      <spirit:reset>
```

```
      <spirit:value>0x00</spirit:value>
    </spirit:reset>
  <spirit:parameters>
    <spirit:parameter>
        <spirit:name>MWMap</spirit:name>
        <spirit:value>false</spirit:value>
    </spirit:parameter>
    </spirit:parameters>
  </spirit:register>
```

For additional information, see Simulink Mapping Within a Memory Map.

Additional Board Support

FPGA board additions:

- Xilinx Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit – This board supports FPGA-in-the-Loop using Digilent® HS3 JTAG and AXI-Master and Data Capture using Digilent HS3 and FTDI JTAG connections.
- Xilinx Zynq UltraScale+ MPSoC ZCU106 Evaluation Kit – This board supports FPGA-in-the-Loop using Digilent HS3 JTAG and AXI-Master and Data Capture using Digilent HS3 and FTDI JTAG connections.
- Xilinx Zynq UltraScale+ RFSoC ZCU111 Evaluation Kit – This board supports FPGA-in-the-Loop using Digilent HS3 JTAG and AXI-Master and Data Capture using Digilent HS3 and FTDI JTAG connections.
- Avnet MiniZed – This board supports AXI-Master and Data Capture using FTDI JTAG.

For a full list of supported boards, see Supported FPGA Devices for FPGA Verification.

Updates to supported software

HDL Verifier now supports these software versions. For a full list of supported software, see Supported EDA Tools and Hardware.

HDL Cosimulation

- Cadence Xcelium 19.03

FPGA Verification

- Xilinx Vivado 2018.3
- Intel Quartus Prime 18.1
- Intel Quartus Prime Pro 18.1 (for use with Intel Cyclone 10 GX only)

DPI Component Generation

- Synopsys VCS MX O-2018.09 SP2

TLM Generation

- Compilers: Visual Studio® VS2017
- System C Modeling Library (SCML): SCML 2.4.3

R2019a

SystemVerilog DPI: Generate SystemVerilog struct from Simulink non virtual buses or complex data types

DPI generation now supports SystemVerilog `struct` data types. Choose between a `struct` or a flattened data type when generating a SystemVerilog DPI component. To create a SystemVerilog `struct`, open the Configuration Parameters, select **SystemVerilog DPI** on the left pane, and set **Composite data type** to `Structure`.

SystemVerilog DPI: Generate SystemVerilog enum from Simulink or from MATLAB enumerated data type

DPI generation now supports SystemVerilog `enum` data types. When a subsystem contains enumerated data types, the generated SystemVerilog DPI component now has a SystemVerilog `enum` data type in its interface. For example, consider this enumerated class definition:

```
classdef BasicColors < Simulink.IntEnumType
  enumeration
    Red(0)
    Yellow(1)
    Blue(2)
  end
end
```

When a subsystem has an input In1 of type `BasicColors`, the generated SystemVerilog interface is:

```
module Subsystem_dpi(
    input bit clk,
    input bit clk_enable,
    input bit reset,
    /* Simulink signal name: 'In1' */
    input BasicColors In1,
    /* Simulink signal name: 'Out1' */
    input BasicColors Out1,
);
```

MATLAB as AXI Master via PCI Express adds support for Xilinx

MATLAB as AXI Master now supports AXI4 read and write operations over a PCI Express connection for Xilinx boards. This feature provides faster performance than the JTAG-based AXI Master. For additional information, see PCI Express MATLAB as AXI Master.

Updates to supported software

HDL Verifier now supports these software versions. For a full list of supported software, see Supported EDA Tools and Hardware.

HDL Cosimulation

- Cadence Xcelium 18.03

FPGA Verification

- Xilinx Vivado 2018.2

- Intel Quartus Prime 18.0
- Intel Quartus Prime Pro 18.0 (for use with Intel Cyclone 10 GX only)
- Microsemi Libero SoC 11.8 SP3

DPI Component Generation

- Synopsys VCS MX N-2017.12 SP2

Functionality being removed or changed

SystemVerilog-DPI tunable parameters retain last-assigned values when reset, and multiple-instance DPI components now share a global memory location for tunable parameter.

Behavior change

Tunable parameters retain their last-assigned values upon reset. This change differs from the previous behavior, where tunable parameters reset to values previously set in Simulink. Multiple-instance DPI components now share a global memory location for the tunable parameter. Now, if one of the instances modifies the value of the parameter, then the other DPI component instances sees this change as well.

To set tunable parameters per instance and retain the previous default reset behavior, first open **Model Explorer**. Then in the **Model Hierarchy** pane, select `Model Workspace`, and for each DPI parameter check the **argument** box and set **Storage Class** to `Model Default`.

R2018b

FPGA Data Capture Integration with HDL Coder: Specify signals to capture during FPGA testing using test points in Simulink

You can now use FPGA Data Capture in an HDL Coder workflow. Configure HDL Workflow Advisor to enable port generation for test points, and then analyze the test point output in MATLAB or Simulink. See Data Capture Workflow for additional information.

This feature requires HDL Coder.

MATLAB as AXI Master via Ethernet: Perform read and write operations on FPGA boards using MATLAB via Ethernet

MATLAB as AXI Master now supports AXI4 read and write operations over an Ethernet connection. This feature provides faster performance than the currently available JTAG-based AXI Master.

MATLAB as AXI Master via PCI Express: Perform high-speed read and write operations on FPGA boards using MATLAB via PCI Express

MATLAB as AXI Master now supports AXI4 read and write operations over a PCI Express connection. This feature provides faster performance than the currently available JTAG-based AXI Master. To use this feature, select **Reference design > PCIe MATLAB as AXI Master and DDR4 Memory Controller** in the HDL Workflow Advisor, then, from **Target Platform Interfaces**, select either **AXI4 Master Read** or **AXI4 Master Write**.

Supported for Intel Arria® 10 GX Development Kit.

SystemVerilog Assertion Generation from Simulink Test: Map Test Assessment blocks to assertions in generated DPI components

Test Assessment and Test Sequence blocks now support DPI component generation. Configure the `verify` statement in the block with steps to verify a subsystem against given specifications. You can then generate a SystemVerilog DPI component to use in an HDL simulation.

For additional information, see Generate SystemVerilog DPI Component from `verify` Statement.

This feature requires Simulink Test™.

Generate a SystemVerilog interface for DPI components

Choose between a port list or a SystemVerilog interface declaration when generating a SystemVerilog DPI component. In the Configuration Parameters, on the **SystemVerilog DPI** tab, select **Connection > Interface** to create a SystemVerilog interface.

Support added for FTDI USB-JTAG cable

HDL Verifier supports FTDI USB-JTAG connection for MATLAB as AXI Master and FPGA Data Capture. This is supported for Windows platforms.

Additional FPGA-in-the-Loop Board Support

FPGA family additions:

- Intel Cyclone 10 GX
- Xilinx Virtex® UltraScale+
- Xilinx Spartan® 7
- Xilinx Kintex UltraScale+ (supported for board customization only, via JTAG connection)
- Microsemi RTG4®

FPGA board additions:

- Intel Cyclone 10 GX FPGA Evaluation Kit
- Xilinx Virtex UltraScale+ FPGA VCU118 Evaluation Kit. This board is supported for PCI Express and JTAG connections.
- Digilent Arty S7-25
- RTG4-DEV-KIT

For a full list of supported boards and FPGA families, see Supported FPGA Devices for FPGA Verification.

Updates to supported software

HDL Verifier now supports the following software versions. For a full list of supported software, see Supported EDA Tools and Hardware.

FPGA Verification

- Xilinx Vivado 2017.4
- Intel Quartus Prime 17.1
- Intel Quartus Prime Pro 17.1 (for use with Intel Cyclone 10 GX only)

Functionality being removed or changed

FIL-Simulation will no longer support the Intel Cyclone III device family
Still runs

FPGA-in-the-Loop support for the Intel Cyclone III device family will be removed in a future release.

These boards will no longer be supported:

- Cyclone III FPGA Starter Kit
- Cyclone III FPGA Development Kit
- Nios II Embedded Evaluation Kit, Cyclone III edition

For a full list of supported boards, see Supported FPGA Devices for FPGA Verification.

In addition, the Cyclone III family will no longer be supported for board customization. For a list of supported boards, see Supported FPGA Device Families for Board Customization.

FIL-Simulation will no longer support the Xilinx Virtex-4 device family
Still runs

FPGA-in-the-Loop support for the Xilinx Virtex-4 device family will be removed in a future release.

These boards will no longer be supported:

- Virtex ML401
- Virtex ML402
- Virtex ML403

For a full list of supported boards, see Supported FPGA Devices for FPGA Verification.

In addition, the Virtex-4 family will no longer be supported for board customization. For a list of supported boards, see Supported FPGA Device Families for Board Customization.

R2018a

Version: 5.4

New Features

SystemVerilog Assertions: Generate SystemVerilog assertions from the DPI-C Assertion block

The DPI-C Assertion block generates SystemVerilog assertions. You can carry these assertions over from your simulation environment to your SystemVerilog testing environment. Previously, you could use an Assertion block in Simulink, and then you had to recreate it manually in SystemVerilog. This feature automates it.

FPGA-in-the-Loop Support for Microsemi: Perform FPGA-in-the-loop simulation using Microsemi SmartFusion2 and Polarfire development kits

This release adds the HDL Verifier Support Package for Microchip FPGA Boards.

FPGA board additions:

- Microsemi SmartFusion®2 SoC FPGA Advanced Development Kit
- Microsemi Polarfire® Evaluation Kit

To use this feature, download HDL Verifier Support Package for Microchip FPGA Boards using the Support Package Installer.

You can access the installer through the FIL Wizard `Get more boards` option. The installer guides you through selecting and installing the board support package. After the installer has completed the download, you can access any supported board through the FPGA Board Manager or the FIL Wizard.

FPGA Data Capture Improvements: Capture multiple windows

This feature enables you to capture FPGA signals into MATLAB or Simulink for multiple occurrences of a trigger. When configuring the Data Capture trigger, specify number of capture windows.

FPGA Data Capture Improvements: Ready-to-capture signal

Generated data capture HDL IPs now have an additional output signal indicating whether the data capture IP is ready to receive data. By qualifying this signal, you can capture data only when the IP is ready to receive it.

Command-line interface for FPGA-in-the-loop workflow

You can now generate a script from HDL Workflow Advisor for the FPGA-in-the-loop workflow. To generate this script, in the HDL Workflow Advisor, select **File > Export to Script**. To execute your workflow, call the generated script from the command line.

For additional information, see Run HDL Workflow with a Script, and Getting Started with the HDL Workflow Command-Line Interface.

DPI-C row major layout

You can now generate DPI-C code in a row major array. To specify this layout, at the MATLAB command prompt, enter:

```
dpigen -rowmajor
```

Alternatively, insert `coder.rowMajor` in a function body. For additional information, check Generate Code That Uses Row-Major Array Layout.

DPI-C generates SystemVerilog package file

DPI-C now generates a *`myfunction`*_dpi_pkg.sv file from MATLAB and Simulink. This file generation simplifies customization of the HDL component. Before this change, when using these function declarations, you had to copy and paste them into your SystemVerilog modules. Now, you can import the package file so that the functionality is available.

Port names more succinct for generated DPI-C modules

When generating DPI-C code, HDL Verifier no longer adds a prefix to the generated ports. The result is shorter port names in the generated code.

For example, the following is a generated interface from previous versions:

```
module Subsystem_dpi(
    input clk,
    input clk_enable,
    input reset,
    input real Subsystem_U_In1,
    output real Subsystem_Y_Out1
);
```

Now, this generated interface is:

```
module Subsystem_dpi(
    input clk,
    input clk_enable,
    input reset,
    input real In1,
    output real Out1
);
```

Additional FPGA-in-the-Loop Board Support: Simulate with Intel Cyclone 10 LP Evaluation Kit

FPGA family additions:

- Intel Cyclone 10 LP

FPGA board additions:

- Intel Cyclone 10 LP Development Kit

This board is supported for FPGA-in-the-loop using a JTAG connection.

Additional FPGA-in-the-Loop Board Support: Simulate with Intel Arria 10 GX FPGA, and connect to Intel Arria 10 SoC Development Kit over Ethernet

The Intel Arria 10 GX FPGA now supports FPGA-in-the-loop using JTAG, Ethernet and PCI Express connections.

In addition, the Intel Arria 10 SoC Development Kit now supports Ethernet connections.

Additional FPGA-in-the-Loop Board Support: Simulate with Avnet PicoZed SDR Development Kit

FPGA board additions:

- Avnet PicoZed™ SDR Development Kit

This board is supported for FPGA-in-the-loop using a JTAG connection.

Updates to supported software

HDL Verifier now supports the following software versions. For a full list of supported software, see Supported EDA Tools and Hardware.

HDL Cosimulation

- Cadence Xcelium™ 17.0
- Questa Core/Prime 10.6b
- ModelSim PE 10.6b

FPGA Verification

- Xilinx Vivado 2017.2
- Intel Quartus Prime 17.0

TLM Generation

- gcc 6.3
- Visual Studio VS2015

DPI Component Generation

- Synopsys VCS MX 2016.06

R2017b

Version: 5.3

New Features

Bug Fixes

SystemVerilog DPI Custom Port Widths: Generate SystemVerilog ports with bit widths that match non-byte-aligned fixed-point widths

You can now generate fixed-point ports as bit or logic vector types rather than C-compatible types. Use these types to generate non-byte-aligned port sizes and port sizes greater than 64 bits.

On the **SystemVerilog DPI** parameters pane, set **Fixed-point data type** to `Bit Vector` or `Logic Vector`. For example, for a 20-bit signed fixed-point port, the generated SystemVerilog port is `bit signed [19:0]` or `logic signed [19:0]`.

You can generate SystemVerilog DPI components with these types from MATLAB or Simulink. These types are also supported for tunable parameters and test-point signals. See Supported Simulink Data Types.

Additional FPGA-in-the-Loop Board Support: Simulate with Xilinx Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit

FPGA family additions:

- Xilinx Zynq UltraScale+ FPGA

FPGA board additions:

- Xilinx Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit

This board is supported for FPGA-in-the-loop using a JTAG connection.

Additional FPGA-in-the-Loop Board Support: Simulate with Terasic Atlas-SoC Kit / DE0-Nano-SoC Kit, and with KCU105 or VCU108 over Ethernet

This release adds FPGA-in-the-loop support for the Terasic Atlas-SoC Kit / DE0-Nano-SoC Kit using a JTAG connection.

It also adds Ethernet connection support for:

- Xilinx Virtex UltraScale™ FPGA VCU108 Evaluation Kit
- Xilinx Kintex UltraScale FPGA KCU105 Evaluation Kit

SystemVerilog DPI asynchronous reset

The asynchronous reset on the SystemVerilog DPI component now updates the output ports to reflect the reset values of internal states. This update applies to SystemVerilog DPI components generated from MATLAB or Simulink.

SystemVerilog DPI component generation maintains data type inheritance in model

In previous releases, if **Hardware Implementation** > **Device vendor** was not set to `Custom Processor`, generating a SystemVerilog DPI component could change the data type inheritance in

your model. In R2017b, SystemVerilog component generation is independent of your **Hardware Implementation** settings, and simulation data type inheritance is not affected.

TLM component generation periodic execution and signal port support

You can now set the generated TLM component to run as a time-periodic thread. Previously, the TLM component executed only when the inputs changed. On the **TLM Processing** parameters pane, set **Algorithm Step Function Execution** to `Periodic SystemC Thread`.

In addition, you can now map a port of the generated TLM component to the `sc_signal` type, by specifying `<spirit:wire>` in the IP-XACT file. See Prepare IP-XACT File for Import.

Updates to supported software

Supported versions of FPGA tools added this release:

- Intel Quartus Prime 16.1
- Xilinx Vivado 2016.4
- Xilinx ISE 14.7

For a full list of supported software, see Supported EDA Tools and Hardware.

Altera FPGA Boards support package renamed

The HDL Verifier Support Package for Altera® FPGA Boards has been renamed to HDL Verifier Support Package for Intel FPGA Boards.

R2017a

Version: 5.2

New Features

Bug Fixes

FPGA Data Capture: Probe internal FPGA signals to analyze in MATLAB or Simulink

Specify signals from your HDL files to capture and return to MATLAB or Simulink. To capture the signals, HDL Verifier generates an IP core that you must integrate into your HDL project and deploy to the FPGA along with the rest of your design.

For Simulink, HDL Verifier generates a block that has output ports corresponding to the signals you captured. For MATLAB, HDL Verifier generates an app to guide you through capturing data, or a System object for programmatic data capture.

To use this feature, you must install the HDL Verifier Support Package for Xilinx or Altera FPGA boards. See HDL Verifier Supported Hardware.

HDL Code Coverage: Activate HDL simulator code coverage in generated test benches

When you generate a test bench, you can now add code coverage of the generated HDL code, including the test bench. The coder generates a build-and-run script that includes the flags for code coverage.

- Simulink Configuration Parameters — On the **HDL Code Generation > Test Bench** pane, select the **HDL code coverage** check box and specify your HDL simulator.
- MATLAB Command Line — When you call `makehdl` or `makehdltb`, set the `HDLCodeCoverage` property to `'on'`. Set the `SimulationTool` property to `'Mentor Graphics Modelsim'` or `'Cadence Incisive'`. Code coverage is not supported for other simulators.
- Workflow Advisor — In step 3.1.4, select the **HDL code coverage** check box and specify your HDL simulator. Then in step 3.2, select **Generate testbench**.

To use this feature, you must have an HDL Coder license. See Generate Test Bench and Enable Code Coverage Using the HDL Workflow Advisor.

MATLAB Based AXI Master: Interactively read and write AXI4 and AXI4-Lite signals on your FPGA

Develop an FPGA design that uses AXI4 and AXI4-Lite registers without the need for an embedded processor. Use MATLAB to read and write the generated AXI4 and AXI4-Lite registers while you run the FPGA design on the board. To use this feature:

1. Include the JTAG AXI Master IP in your FPGA project or HDL Coder reference design. This IP can communicate with MATLAB.
2. Deploy the design to your board.
3. Use MATLAB to read and write the generated AXI4 and AXI4-Lite registers in the FPGA design.

See IP Core Generation Workflow without an Embedded ARM Processor: Xilinx Kintex-7 KC705, or IP Core Generation Workflow without an Embedded ARM Processor: Arrow DECA MAX 10 FPGA Evaluation Kit (HDL Coder).

To use this feature, you must install the HDL Verifier Support Package for Xilinx or Altera FPGA boards. See HDL Verifier Supported Hardware.

SystemVerilog DPI Test Bench Generation: Speed up test bench generation from Simulink models with large data sets

Reduce test bench generation and simulation time, especially when using large data sets.

- Simulink Configuration Parameters — On the **HDL Code Generation > Test Bench** pane, select **SystemVerilog DPI test bench**, and specify your HDL simulator.
- Workflow Advisor — In step 3.1.4, select **SystemVerilog DPI test bench**, and specify your HDL simulator. Then in step 3.2, select **Generate testbench**.

The coder generates a DPI component for your entire Simulink model, including your DUT and data sources. Your entire model must support C code generation with Simulink Coder™. The coder generates a SystemVerilog test bench that compares the output of the DPI component with the output of the HDL implementation of your DUT. The tool also generates a build script for your simulator. The supported simulators are:

- `Mentor Graphics Modelsim`
- `Cadence Incisive`
- `Vivado`
- `VCS`

To use this feature, you must have an HDL Coder license and a Simulink Coder license.

See Verify HDL Design With Large Dataset Using SystemVerilog DPI Test Bench (HDL Coder).

Native Floating-Point Test Bench: Generate SystemVerilog DPI, cosimulation, and FPGA-in-the-loop test benches that have single-precision data types (requires HDL Coder)

When you generate a SystemVerilog DPI component, a cosimulation model or System object, or an FPGA-in-the-loop block or System object from a DUT that has `single` data type ports, the coder generates the test bench using the HDL Coder native floating-point library.

To use this feature, you must have an HDL Coder license.

See Verify the Generated Code from Native Floating-Point (HDL Coder).

Additional FPGA Board Support: Perform FPGA-in-the-loop simulation with Xilinx Virtex UltraScale family boards

FPGA family additions

- Xilinx Virtex UltraScale FPGA

FPGA board additions:

- Xilinx Virtex UltraScale FPGA VCU108 Evaluation Kit (JTAG connection only)
- Xilinx Virtex-7 VC709 Evaluation Board (JTAG and PCI Express connection)
- Digilent Arty Board, or Xilinx Artix®-7 FPGA Development Board (JTAG connection only)

Generate SystemVerilog DPI components for models with complex ports in MATLAB

In the generated SystemVerilog component, the coder flattens complex signals into real and imaginary parts . See Supported MATLAB Data Types.

Updates to supported software

This section lists the supported versions added in the current release. For a full list of supported software, see Supported EDA Tools and Hardware.

HDL Cosimulation

- Cadence Incisive® 15.2
- Mentor Graphics ModelSim 10.5b

FPGA Verification

- Altera Quartus Prime 16.0
- Xilinx Vivado 2016.2

R2016b

Version: 5.1

New Features

Bug Fixes

FPGA-in-the-Loop for Control Applications: Return a larger output data size when using an overclocking factor

The maximum output data size of your FIL block is no longer reduced by using an overclocking factor. The output data size of your FIL block, rounded up to the nearest byte, can be up to 1467 bytes, independent of the overclocking factor.

FPGA-in-the-Loop Custom Clock Speed: Specify the FPGA system clock frequency in the FIL Wizard

In the **FIL Options** section of the FPGA-in-the-Loop Wizard, enter your **FPGA system clock frequency** in MHz.

The default clock frequency remains 25 MHz.

Multirate SystemVerilog DPI Components: Generate multirate test benches to verify that your generated component matches Simulink behavior

You can now generate a test bench for a SystemVerilog DPI component that has multirate interfaces. The generated test bench includes a timing controller that assists the stimulus and checker modules to align data at different rates.

Logic Analyzer: Visualize, measure, and analyze transitions and states over time for Simulink signals (requires DSP System Toolbox)

The Logic Analyzer visualization tool enables you to view the transitions of signals. You can use the **Logic Analyzer** to:

- Debug and analyze models
- Trace and correlate many signals simultaneously
- Detect and analyze timing violations
- Trace system execution

See Inspect and Measure Transitions Using the Logic Analyzer to explore some of its key functionality.

You must have a DSP System Toolbox™ license and a Simulink license to use this feature.

Generate SystemVerilog DPI test bench for Xilinx Vivado

When you call the `makehdltb` function, set the `GenerateSVDPITestBench` property to `'Vivado Simulator'` to create a test bench and script for the Xilinx Vivado simulator.

You must have an HDL Coder license and a Simulink Coder license to use this feature.

Cross-platform support for TLM component generation

You can generate your TLM component on any host machine for simulation on a Windows or Linux target machine. In the **TLM Compilation** parameter group, specify the **Operating System** of the

target machine. By default, the tool generates code for the same operating system as the host machine.

Generate unmapped IP-XACT registers for TLM component

When you generate a TLM component using an IP-XACT memory map, you can now include all registers in the memory map in the generated component. Previously, the component included only registers mapped to Simulink ports and parameters, indicated with the `MWMap` tag in the IP-XACT file.

To include the unmapped registers, in the **TLM Mapping** parameter group, select the **Generate code for unmapped IP-XACT registers/bitfields** check box.

Additional FPGA Board Support: Perform FPGA-in-the-loop simulation with Altera Arria 10 family boards

- FPGA family additions:
  - Altera Arria 10 FPGA
- FPGA board additions:
  - Altera Arria 10 SoC development kit

Updates to supported software

This section lists the supported versions added in the current release. For a full list of supported software, see Supported EDA Tools and Hardware.

FPGA Verification

- Altera Quartus Prime 15.1
- Xilinx Vivado 2015.4

DPI Component Generation

- 32-bit Cadence Incisive

TLM Component Generation

- Visual Studio 2013

R2016a

PCI Express FPGA-in-the-Loop: Perform FIL simulation on selected Xilinx and Altera development boards

Perform FPGA-in-the-loop simulation on selected Xilinx and Altera FPGAs using a PCI Express connection (Windows 64 only). You must have the HDL Verifier Support Package for Xilinx or Altera FPGA boards. When you install the FPGA board support package, it also installs the PCIe driver for your board.

Board Support

HDL Verifier supports FIL over a PCIe connection on these boards:

- Xilinx Virtex-7 VC707 development board
- Xilinx Kintex-7 KC705 development board
- Altera Cyclone V GT FPGA development kit
- Altera DSP development kit, Stratix® V edition

Software Requirements

- Xilinx Vivado 2015.2
- Altera Quartus II 15.0

Faster Test Bench Generation and HDL Simulation: Generate SystemVerilog DPI test benches for large data sets from HDL Coder

Reduce test bench generation and simulation time, especially when using large data sets. To use this feature, when you call the `makehdltb` function in HDL Coder, set the `GenerateSVDPITestBench` property. The coder generates a DPI component for your entire Simulink model, including your DUT and data sources. Your entire model must support C code generation with Simulink Coder. The coder generates a SystemVerilog test bench that compares the output of the DPI component with the output of the HDL implementation of your DUT. The tool also generates a build script for your simulator. You can specify '`ModelSim`', '`VCS`', or '`Incisive`'.

```
makehdltb(gcb,'GenerateSVDPITestBench','ModelSim','GenerateHDLTestbench','off')
```

You must have an HDL Coder license and a Simulink Coder license to use this feature.

Expanded Data Type Support in SystemVerilog DPI: Generate SystemVerilog DPI components for models that have buses, structures, or complex signals as I/O

For a list of supported data types for SystemVerilog DPI component generation, see:

- Supported Simulink Data Types
- Supported MATLAB Data Types

Additional FPGA Board Support: Perform FPGA-in-the-loop simulation with Xilinx Kintex UltraScale and Altera MAX 10 family boards

- FPGA family additions

- Xilinx Kintex UltraScale
- Altera MAX® 10

- FPGA board additions
  - Xilinx Kintex UltraScale FPGA KCU105 Evaluation Kit
  - Arrow® MAX 10 DECA

For a full list of supported boards, see Supported EDA Tools and Hardware.

Guided Hardware Setup for New FPGA Boards

When you download an FPGA board support package for use with HDL Verifier, the installer guides you through setting up your hardware with an Ethernet, JTAG, or PCIe connection. You can test and verify the connection before you perform FPGA-in-the-loop simulation. Doing so can eliminate problems relating to equipment and connectivity issues. Then you can focus on your algorithm and FIL performance. See Guided Hardware Setup. This feature is supported for Windows and Linux operating systems.

Updates to supported software

The following topics list the supported versions added in the current release. For a full list of supported software, see Supported EDA Tools and Hardware.

FPGA Verification

- Altera Quartus II 15.0
- Xilinx Vivado 2015.2

TLM Component Generation

- Visual Studio 2013
- Windows 7.1 SDK

Functions and Function Elements Being Removed

These TLM component generation options will be removed in a future release.

Function or Function Element Name	Use This Instead	Compatibility Considerations
Enable temporal decoupling for loosely-timed simulation	Use default timing.	Will warn in a future release.
Enable payload buffering	If you need a buffer, manually add it in your SystemC test environment.	Will warn in a future release.

R2015b

SystemVerilog DPI Component Test Points: Access the internal signals of a component from the test bench

In previous releases, only the input and output signals of a DPI component were visible. You can now access internal signals of a Simulink model in your HDL simulator.

See SystemVerilog DPI Component Test Point Access and Getting Started with SystemVerilog DPI Component Generation.

SystemC Modeling Library (SCML) Wrapper: Generate SCML as part of TLM component

SystemC Modeling Library is an add-on library for modeling TLM interfaces. SCML-instrumented components allow integration with Synopsys simulation and verification tools. In the **Code Generation** dialog, on the **TLM Mapping** tab, select **Defined by imported IP-XACT file**. Then select the **Use SCML to generate the memory map** checkbox and specify an IP-XACT file for the component.

You can specify your SCML library paths on the **TLM Compilation** tab.

See Implement Memory Map with SCML.

TLM Generator: IP-XACT field support

In your IP-XACT file, you can specify more than one `field` per `register`. The generator uses the `bitWidth` and `bitOffset` tags to construct the register fields in the TLM component.

Updates to supported software

The following topics list the supported versions added in the current release. For a full list of supported software, see Supported EDA Tools and Hardware.

FPGA Verification

- Quartus II 14.0
- Xilinx Vivado 2014.4

TLM Component Generation

- SCML 2.2

Removed support for BEEcube miniBEE hardware

HDL Verifier Support Package for BEEcube® miniBEE® Platform has been removed and is no longer available.

Compatibility Considerations

In MATLAB R2015b or later, if you run a Simulink model or MATLAB code that contains BEEcube miniBEE hardware support, you will get an error

R2015a

Version: 4.6

New Features

Bug Fixes

FPGA-in-the-loop through JTAG for Xilinx boards

Perform FPGA-in-the-loop simulation on Xilinx FPGAs using a Digilent JTAG connection.

Board Support

FIL can support any Xilinx FPGA board through a Digilent JTAG connection, as long as the board uses:

- An FPGA device in the supported Xilinx FPGA family: Virtex 7, Kintex 7, Artix 7, or Zynq 7000.
- A Digilent download cable. If your board has a standard Xilinx 14-pin JTAG connector, you can obtain the HS2 cable from Digilent.

As of this release, all Xilinx 7-series FPGA boards that HDL Verifier supports directly can perform simulation through an on-board Digilent JTAG cable. To add more boards that support FPGA devices with JTAG connections, use the FPGA Board Manager.

For more information on JTAG with FIL, see JTAG Connection.

Software Requirements

- For Windows operating systems: Xilinx Vivado 2014.2. The Vivado executable directory must be on the system path.
- For Linux operating systems: Xilinx Vivado 2014.2 and Digilent Adept2.

FPGA-in-the-Loop support for rapid accelerator mode in Simulink

The FIL Simulation block now supports simulation using Rapid Accelerator mode. Before beginning cosimulation, in the model window, select **Simulation > Rapid Accelerator Mode**. See the FIL Simulation block reference for more information about this feature and block settings.

DPI-C enhancements, including multiple-instance support and integration with build toolchain

Multiple Instance Support

With the current release, multiple instances of a generated SystemVerilog DPI-C (for both Simulink and MATLAB) are supported in the HDL simulator.

Build Toolchain Integration

With build toolchain integration, you no longer have to manually build the shared library through the command-line interface. In addition, you can now target HDL simulators running on Linux from Simulink software that is running on Windows.

To specify the toolchain you want to use for code generation and (optionally) compilation:

1. In Simulink, open **Configuration Parameters**. Specify one of the supported SystemVerilog DPI-C targets.
2. From the **Build Process** list, select your toolchain.
3. Optionally, under **Build Configuration**, select any flags you want for compilation.

IP-XACT support for TLM

- Import IP-XACT files for memory map and nonmemory map TLM generation.

  To specify the files you want to import:

  1. In Simulink, select **Simulation > Model Configuration Parameters**.
  2. In the **TLM Mapping** pane, select **Defined by imported IP-XACT file**.
  3. When prompted, provide the full path to the IP-XACT file.
- Export IP-XACT file with generated TLM component.

  The TLM generator generates the IP-XACT file automatically and export it to the same folder as the project makefile.

Additional FPGA-in-the-loop board support

- FPGA family additions
  - Xilinx Zynq-7000
  - Xilinx Artix-7
  - Altera Arria V
- FPGA board additions
  - Digilent Nexys™ 4 Artix-7 FPGA board
  - Xilinx Zynq-7000 ZC702 development board
  - Xilinx Zynq-7000 ZC706 development board
  - Zedboard
  - Digilent ZYBO™ Zynq-7000 development board
  - Altera Arria V starter kit
  - Altera Arria V SoC development kit

Process improvement for SystemVerilog DPI-C generation

When selecting options for SystemVerilog DPI-C generation, you no longer have to specify the location of `svdpi.h` and `libvsim.lib` files.

Delay propagation and extra control signals eliminated from generated SystemVerilog code

DPI component generation now provides a SystemVerilog template for AMS workflows that eliminate delay propagation and extra control signals.

TLM generation updates

With the current release, TLM Generation from HDL Verifier provides:

- Bus and structure support

- SystemC 2.3.1 support
- Multiple instance support

R2014b

Version: 4.5

New Features

Bug Fixes

SystemVerilog DPI-C component generation based on MATLAB Coder

With the current release, you can export a MATLAB design with direct programming interface (DPI) for Verilog® or SystemVerilog simulation. With this feature, you can wrap generated C code with a DPI wrapper that communicates with a SystemVerilog thin interface function in a SystemVerilog simulation.

For more on this feature, see DPI Component Generation for MATLAB Function.

Note You must have a MATLAB Coder license to use this feature.

SystemVerilog DPI-C component generation based on Simulink Coder

To generate a SystemVerilog component using Simulink, you no longer require Embedded Coder®. In Code Generation, select `systemverilog_dpi_grt.tlc` as the **System target file**.

Xilinx Vivado support for FPGA-in-the-Loop

With this release, you can use Xilinx Vivado for FIL simulation. Xilinx Vivado supports 7-series and newer FPGA families. HDL Verifier supports Xilinx Vivado version 2013.4.

In the FIL simulation workflow, specify Vivado when you Set Up Hardware and Hardware Tools:

```
hdlsetuptoolpath('ToolName','Vivado','ToolPath','c:\HDLTools\Vivado\2013.4-mw-0\Win')
```

SGMII interface support for FPGA-in-the-Loop in Xilinx Virtex-7 FPGAs

With the current release, you can use the Virtex-7 VC707 Development Board with SGMII interface for FIL simulation. This board requires Xilinx Vivado version 2013.4.

You can download support for Virtex-7 VC707 in the HDL Verifier Support Package for Xilinx FPGA Boards.

Additional FPGA-in-the-Loop board support: Xilinx Virtex-7 FPGA VC707 Evaluation Kit, Arrow SoC Kit Evaluation Board, Altera Cyclone V GT FPGA Development Kit

HDL Verifier FPGA-in-the-Loop verification has added support for the following FPGA boards:

- Xilinx Virtex-7 FPGA VC707 Evaluation Kit
- Arrow SoC Kit Evaluation Board
- Altera Cyclone V GT FPGA Development Kit

For a full list of supported boards, see Supported EDA Tools and Hardware.

Updates to supported software

The following topics list the supported versions added in the current release. For a full list of supported software, see Supported EDA Tools and Hardware.

HDL Cosimulation

- Incisive® 13.2 p002
- QuestaSim 10.3

FPGA Verification

- Xilinx 14.7
- Quartus II 13.1
- Xilinx Vivado 2013.4

TLM Component Generation

- Visual Studio: VS2012
- SystemC 2.3.1 (TLM included)

Documentation installation with hardware support package

Starting in R2014b, each hardware support package installs with its own documentation. See HDL Verifier Supported Hardware for a list of support packages available for HDL Verifier, with links to documentation.

R2014a

Version: 4.4

New Features

Bug Fixes

FPGA-in-the-Loop over JTAG for Altera FPGAs

Perform FPGA-in-the-Loop simulation on Altera FPGAs using a JTAG connection.

- Supported FPGA devices: Any Altera FPGA board within the supported FPGA family, for example:
  - Cyclone III, IV, V, and V SoC
  - Arria II
  - Stratix IV and V
  - Additional boards within the Supported FPGA Devices for FIL Simulation families can be custom added with the FPGA Board Manager.
- Hardware:
  - Altera FPGA boards
  - USB Blaster I or USB Blaster II download cable
- Software:
  - Windows: Quartus II 12.1 or higher version; Quartus II executable folder must be on system path
  - Linux: Quartus II 13.0sp1 with a patch, or Quartus II 13.1 (Quartus II library folder must be on LD_LIBRARY_PATH *before* starting MATLAB); only 64-bit Quartus are supported
  - Installation of USB Blaster I or II driver

Parameter Tuning for Generated TLM Component

The tunable parameters register allows you to make adjustments to the TLM component before or during simulation. You set this parameter in the Configuration Parameters dialog, in the **TLM Mapping** tab. See Select TLM Mapping Options.

Multiple Socket Control for Generated TLM Component

You can choose to have a single, combined TLM socket for input data, output data, and control, or you can choose three separate TLM sockets for input data, output data, and control, so that you can connect the sockets to different buses. Set this parameter in the Configuration Parameters dialog, in the **TLM Mapping** tab. See Select TLM Mapping Options.

FPGA-in-the-Loop support for Altera Cyclone V SoC FPGA boards

You can use Altera Cyclone V SoC FPGA boards for FIL simulation with a JTAG connection.

Updates to supported software and hardware

Software updates

HDL Cosimulation

- ModelSim PE 10.2c
- QuestaSim 10.2c

- Incisive 13.10-s006

FPGA Verification

- Xilinx ISE 14.6
- Altera Quartus II 13.0sp1

Hardware support updates

- Altera Cyclone V SoC development board

For a list of supported boards and device families, see Supported FPGA Devices. To add a custom board for use with FIL, see Create Custom FPGA Board Definition. For instructions on downloading the FPGA board support packages with the Support Package Installer, see Support Packages and Support Package Installer.

R2013b

SystemVerilog DPI component generation from Simulink

In R2013b, you can export a Simulink subsystem with a direct programming interface (DPI) for Verilog or SystemVerilog Simulation. With this feature, you can wrap generated C code with a DPI wrapper that communicates with a SystemVerilog thin interface function in a SystemVerilog simulation.

This feature is available in the Model Configuration Parameters dialog box:

1. In Code Generation, select `systemverilog_dpi_ert.tlc` as the **System target file**.
2. Expand Code Generation, and select DPI Generator.
3. Specify the DPI include path, and indicate whether or not you want to generate a test bench.
4. Click **OK** to accept your choices and exit the dialog box or **Apply** to continue making changes.

Note You must have an Embedded Coder license to use this feature.

BEEcube miniBEE FPGA-in-the-Loop (FIL) support package

The BEEcube miniBEE hardware platform is available for FIL simulation as an HDL Verifier support package. The miniBEE support package requires you to install a PCI Express connection for host-board communications.

Note The BEEcube miniBEE hardware platform support package works only on the CentOS that ships with miniBEE.

For instructions on downloading HDL Verifier support packages with the Support Package Installer, see FPGA Board Support Packages for FIL.

After the Support Package Installer has started, select `BEEcube miniBEE Platform` at **Select a support package to install** pane in the installer GUI. For more information about using this support package with FIL, see Support Package for BEEcube miniBEE Hardware Platform.

Additional FPGA board support for FIL, including Xilinx KC705 and Altera DSP Development Kit, Stratix V edition

Several FPGA boards have been added to the HDL Verifier FPGA board support packages, including Xilinx KC705 and Altera DSP Development Kit, Stratix V edition.

For a full list of boards added, see “Updates to supported software and hardware” on page 20-3. For instructions on downloading the FPGA board support packages with the Support Package Installer, see Support Packages and Support Package Installer.

Floating-point data type for cosimulation and FIL blocks

Double and single data types on the DUT interface are supported for HDL cosimulation and FIL test bench generation. This feature is also available for System objects.

HDL file compilation ordering in Cosimulation Wizard

VHDL® files are automatically sorted into the right compilation order in the HDL Cosimulation Wizard, saving you time and, in some cases, errors. You can add files in any order you choose. You can also still manually arrange the HDL files by using the **Up** and **Down** buttons.

Shared memory connection in Cosimulation Wizard

Shared memory is an available connection method in the HDL Cosimulation Wizard. Use shared memory communication if your firewall policy does not allow TCP/IP socket communication. In the Simulation Options pane, select `Shared Memory` for the **Connection method**.

SGMII board support for FPGA-in-the-Loop simulation

SGMII support has been added to FPGA-in-the-Loop simulation. You can now perform FIL simulation with Altera Stratix IV and Stratix V FPGA boards that require an SGMII I/O interface.

Floating point for FIL and HDL cosimulation test bench generation

With the R2013b release, HDL Verifier supports double and single data types on the DUT interface for test bench generation using HDL Coder HDL Workflow Advisor for Simulink.

Updates to supported software and hardware

Software updates

HDL Cosimulation

- ModelSim SE 10.1c

FPGA Verification

- Xilinx ISE 14.4
- Altera Quartus II 12.1sp1

Hardware support updates

Device Family Support Additions

- Cyclone V

Board Support Additions

- Altera Stratix IV GX FPGA development kit
- Altera Cyclone V GX FPGA development kit
- Altera DSP Development Kit, Stratix V edition
- BeMicro SDK
- Xilinx Kintex-7 KC705 board

For a list of supported boards and device families, see Supported FPGA Devices. To add a custom board for use with FIL, see Create Custom FPGA Board Definition. For instructions on downloading

the FPGA board support packages with the Support Package Installer, see Support Packages and Support Package Installer.

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When You Use The Function or Element?	Use This Instead	Compatibility Considerations
`FrameBasedProcessing` property	Warns	Sample mode or frame mode is automatically detected based on the size of the inputs during the step method execution.	In a future release, the use of any scripts containing this property will cause an error.
FPGA Automation pane in the Generate HDL dialog box	Errors		The FPGA Automation pane has been removed.

R2013a

FPGA-in-the-loop test bench generation through HDL Workflow Advisor for MATLAB

With the MATLAB Coder Workflow Advisor, the HDL Verification step includes automation for the following workflow:

- Verify with FPGA-in-the-Loop: Create the FPGA programming file and test bench, and, optionally, download it to your selected development board.

  Note You *do* require an HDL Verifier license to use this feature.

HDL cosimulation test bench generation through HDL Workflow Advisor for MATLAB

With the MATLAB Coder Workflow Advisor, the HDL Verification step includes automation for the following workflows:

- Verify with HDL Test Bench: Create a standalone test bench. You can choose to simulate using ModelSim or Incisive with a vector file created by the Workflow Advisor.

  Note You do not require an HDL Verifier license to use this feature.
- Verify with Cosimulation: Cosimulate the device under test (DUT) in ModelSim or Incisive with the test bench in MATLAB.

  Note You *do* require an HDL Verifier license to use this feature.

Transaction Level Model generation using Simulink Coder

You can generate Transaction Level Models with either a Simulink Coder license *or* an Embedded Coder license. You are not required to have both product licenses.

In Model Configuration Parameters, under **Code Generation**, follow these guidelines for selecting the correct system target file:

- With Simulink Coder, select: `tlmgenerator_grt.tlc`
- With Embedded Coder, select: `tlmgenerator_ert.tlc`

Compatibility Considerations

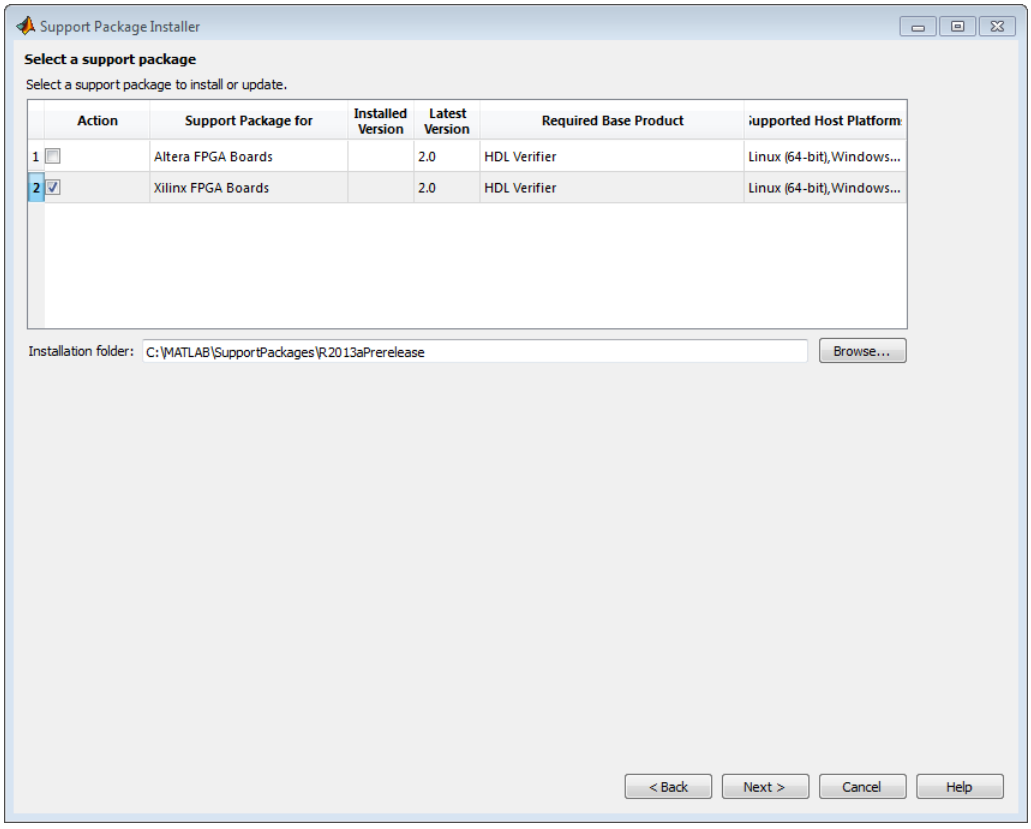
You no longer require Embedded Coder to generate Transaction Level Models with HDL Verifier.

Support Package for FPGA-in-the-Loop

All FPGA boards supported by FPGA-in-the-Loop are now available via download with the Support Package Installer. You can choose to download all supported Altera boards or all supported Xilinx boards, or both.

When you are using the FPGA Board Manager, select **Get More Boards** to start the Support Package Installer with the FPGA board download packages already displayed. You can also access the installer

through the FIL Wizard `Get more boards` option. The installer guides you through selecting and installing the board support package. After the installer has completed the download, you can access any supported board through the FPGA Board Manager or the FIL Wizard.



Code Generation for FIL Simulation Block

With Release R2013a, you can generate code for the FIL Simulation block.

Updates to supported software and boards

Software updates

FPGA Verification

- Xilinx ISE 14.2
- Altera Quartus™ II 12.0

TLM Generation

- Compilers:
  - Visual Studio: VS2005, VS2008, VS2010

  - gcc 4.4.6
- SystemC:
  - SystemC 2.3.0 (TLM included)

Board additions

- Altera DSP Stratix V

For a list of supported boards and device families, see Supported FPGA Devices. To add a custom board for use with FIL, see Create Custom FPGA Board Definition.

Compatibility Considerations

Xilinx does not ship the Digilent plugin with ISE 14.2. To get the plugin, see the Digilent plugin and related software download page on the Digilent web site.

HDL Verifier No Longer Supports Legacy FIL Programming Files

FPGA-in-the-Loop FPGA programming files generated using HDL Verifier versions older than 4.2 (current release) are not compatible with the current version of software.

Compatibility Considerations

FPGA programming files that were generated with version 4.1 or earlier of HDL Verifier are unusable with the current version of software (R2013a). You must regenerate these programming files using the FIL Wizard or HDL Workflow Advisor with the current release.

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When You Use The Function or Element?	Use This Instead	Compatibility Considerations
`FrameBasedProcessing` property	Warns	Sample mode or frame mode is automatically detected based on the size of the inputs during the step method execution.	In a future release, the use of any scripts containing this property will cause an error.
FPGA Automation pane in the Generate HDL dialog box	Warns		In a future release, the FPGA Automation pane will be removed.

R2012b

Version: 4.1

New Features

Bug Fixes

Custom board APIs for FPGA-in-the-loop

New FPGA Board Manager allows you to add custom board information so that you can use FIL simulation with an FPGA board that is not one of the pre-registered boards. See Create Custom FPGA Board Definition.

System object for FPGA-in-the-Loop

FIL System object for FIL simulation between FPGA and MATLAB. The `hdlverifier.FILSimulation` System object can be generated only with the FIL Wizard. See FIL Wizard: Generate FIL System Object.

100 Base-T Ethernet support for FPGA-in-the-loop block

HDL Verifier supports FPGA boards that have 100 Mbit/sec Ethernet PHY. The software will automatically detect the configuration of the PHY chip and employ the appropriate interface in the FPGA as necessary for FIL simulation.

Automatic verification with cosimulation using HDL Coder

With the HDL Coder HDL Workflow Advisor, you can automatically verify using your Simulink test bench with the new verification step **Run Cosimulation TestBench**. During verification, the HDL Workflow Advisor and HDL Verifier verify the generated HDL using cosimulation between the HDL Simulator and the Simulink test bench. See Automatic Verification.

Updates to supported software and boards

Software updates

- ModelSim 10.1a, 10.0c, and 6.6d
- Cadence Incisive 11.10-s005
- Xilinx ISE 13.4

Board additions

- Altera Nios II Embedded Evaluation Kit, Cyclone III Edition

For a full list of preregistered boards, see Supported FPGA Devices for FIL Simulation. To add a custom board for use with FIL, see Create Custom FPGA Board Definition.

R2012a

EDA Simulator Link Is Now HDL Verifier

Effective R2012a, EDA Simulator Link is now HDL Verifier.

FPGA-in-the-Loop for Altera Boards

FPGA-in-the-Loop now supports Altera FPGA design software and the following Altera development kits and boards:

- Altera Arria II GX FPGA development kit
- Altera Cyclone III FPGA development kit
- Altera Cyclone IV GX FPGA development kit
- Altera DE2-115 development and education board

See Required Products, Performing FPGA-in-the-Loop.

System Object for HDL Cosimulation with MATLAB, with Automatic System Object Generation

The HDL cosimulation System object provides integrated HDL cosimulation with MATLAB. When you use this workflow to cosimulate MATLAB and HDL code, you gain the following benefits of using the System object:

- Control all aspect of the cosimulation from MATLAB.
- Easily configure all test bench parameters.
- Remove need for multiple function calls.
- Create System object automatically from existing HDL code (see “Automatic System Object Generation with CosimWizard” on page 23-2).

The HDL cosimulation System object supports HDL cosimulation with both Mentor Graphics ModelSim and Cadence Incisive. You can read more about the HDL cosimulation System object and its methods and properties in the HDL Cosimulation System Objects reference page.

Automatic System Object Generation with CosimWizard

Although you can hand code an HDL cosimulation System object, you can more easily create the System object automatically using existing HDL code and the HDL Cosimulation Wizard. This workflow also creates an HDL launch script for easier startup.

See Creating a Function, System Object, or Block.

Use of FPGA Board as Source Block with FPGA-in-the-Loop

Effective R2012a, you can use the FPGA board as the source of stimuli in Simulink. Only one output is required. This feature enables high speed generation and processing of test stimulus with results brought back to Simulink for analysis.

See the example “Algorithm Verification with FIL Source Block”.

HDL Regression Testing with Simulink Design Verifier

You can now perform successive simulation runs without restarting the HDL simulator. This enhancement allows uses of automatically generated test cases, from the original behavioral model, generated from Simulink Design Verifier™. Such regression testing can achieve complete model and HDL code coverage.

Previously, commands issued in the Tcl startup file were executed only once, when the HDL simulator was started. Now, Tcl commands in the Simulation pane of the HDL Cosimulation block are stored with the block and are issued with each new simulation run. You do not have to reissue the commands in the HDL simulator.

See the example "Generating HDL Code Coverage Using Simulink and ModelSim".

New Examples for R2012a

- HDL Cosimulation with MATLAB System object:
  - Cosimulation Wizard for MATLAB System object
  - Verifying Viterbi Decoder Using MATLAB System object and Cadence Incisive
  - Verifying Viterbi Decoder Using MATLAB System object and Mentor Graphics ModelSim
- Accelerate Algorithm Verification with a FIL Source Block
- Generating HDL Code Coverage Using Simulink and Mentor Graphics ModelSim
- FIL Demos Updated for Altera Workflow

HDL Verifier Supported Software and System Updates

The HDL Verifier supported software updates for this release are listed in the following sections.

See Required Products for a complete listing of supported products.

HDL Cosimulation

- Support for ModelSim 10.0c added
- No change to supported Cadence Incisive versions

FPGA-in-the-Loop

- Altera Quartus II 11.0
- No change to supported Xilinx Design Suite versions

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When You Use The Function or Element?	Use This Instead	Compatibility Considerations
FPGA Automation in Configuration Parameters or Model Explorer	You cannot access this feature	Use the HDL Workflow Advisor.	Use replacement workflow.
`fpgamodelsetup`	Errors	Use the HDL Workflow Advisor	Use replacement workflow.
`makefpgaproject`	Errors	Use the HDL Workflow Advisor	Use replacement workflow.
`configuremodelsim`	Errors	`vsim`	Use replacement function.
`wrapverilog`	Errors	Wrapping Verilog code is no longer required.	

R2011b

FPGA-in-the-Loop Workflow in HDL Coder HDL Workflow Advisor

FPGA-in-the-Loop (FIL) is available using the HDL Coder HDL Workflow Advisor. You can verify FPGA designs with FIL as part of the HDL Workflow Advisor workflow, which will create the FPGA programming file and download it to your selected development board. See HDL Coder documentation for details.

FPGA-in-the-Loop Updates

This release removes previous limitations and now supports:

- Arbitrary vectors. Limitations on input and output signal size in FIL have been removed.
- Variable step solvers
- ode45 solvers and model referencing

Conversion of Error and Warning Message Identifiers

For R2011b, EDA Simulator Link error and warning message identifiers have changed.

Compatibility Considerations

If you have scripts or functions that use message identifiers that changed, you must update the code to use the new identifiers. Typically, message identifiers are used to turn off specific warning messages.

For example, the `edalink:filWizard:NotString` identifier has changed to `edalink:filWizard:NotString`. If your code checks for `edalink:filWizard:NotString`, you must update it to check for `edalink:filWizard:NotString` instead.

To determine the identifier for a warning that appears at the MATLAB prompt, run the following command after you see the warning:

```
[MSG,MSGID] = lastwarn;
```

This command saves the message identifier to the variable `MSGID`.

Note Warning messages indicate a potential issue with your model or code. While you can turn off a warning, a suggested alternative is to change your model or code so it does not generate warnings.

EDA Simulator Link Supported Software and System Updates

EDA Simulator Link supported software updates for this release include:

- 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
- Cadence IES 10.2-s040

- Cadence IES 9.2-s014
- Cadence IUS 8.2-s009
- Xilinx ISE 13.1

See Required Products.

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When you use the Function or Element?	Use This Instead	Compatibility Considerations
FPGA hardware-in-the-loop via the EDA Link pane in Configuration Parameters or Model Explorer	Errors	`filWizard` or FIL in the HDL Workflow Advisor (HDL Coder).	Use replacement function
FPGA Automation in Configuration Parameters or Model Explorer	Warns	Use the HDL Workflow Advisor.	Use replacement workflow
`configuremodelsim`	Errors	`vsim`	Use replacement function
`launchDiscovery`	Errors	Support for Synopsys Discovery™ has been removed.	You must remove the `launchDiscovery` command from any scripts or programming files that contain it.
HDL Cosimulation block—Discovery	Errors	Support for Synopsys Discovery has been removed.	Remove or replace the Discovery HDL Cosimulation block in your Simulink models.

R2011a

FPGA-in-the-Loop Simulation

This release provides the capability for verification of FPGA designs with FPGA-in-the-Loop (FIL) simulation. The FIL Wizard, using HDL files that you provide, creates all the FPGA programming files and downloads them to a development board. It also provides you with a FIL block to insert into your existing model so that you can then run and test your FPGA implementation on the development board using Simulink. See Generating a FIL Simulation Block for details.

EDA Simulator Link tested FIL simulation with: Xilinx ISE 12.1; Supports Windows 32, Windows 64, Linux 32, Linux 64.

Multiple Cosimulation Sessions Support with Parallel Computing

You can use the EDA Simulator Link and Parallel Computing Toolbox™ products together for up to eight cosimulation sessions on local machine. Use MATLAB Distributed Computing Server™ to farm out sessions to any number of other computers or to run more than eight sessions on a local machine.

Refer to the Parallel Computing Toolbox and the MATLAB Distributed Computing Server documentation for details and examples of using parallel computing with MATLAB and Simulink.

New User Guide Section for Using HDL Instance Object with Test Bench and Component Functions

Expanded documentation helps you learn how to use the `use_instance_obj` argument for MATLAB functions `matlabcp` and `matlabtb`. You can use this feature to pass an HDL instance object to the function as an argument. In previously releases, the `iport`, `oport`, `tnext`, `tnow`, and `portinfo` arguments of the MATLAB function definition served this purpose. With this feature, `matlabcp` and `matlabtb` function callbacks get the HDL instance object passed in: to hold state, provide read/write access protection for signals, and allow you to add state as you wish.

See Writing Functions Using the HDL Instance Object for details.

EDA Cosimulation Assistant Name Change to Cosimulation Wizard

The feature name "EDA Cosimulation Assistant" has been changed to "Cosimulation Wizard". The function used to launch the wizard has changed also. The function `edaCosimAssist` will be removed in a future release. Although `edaCosimAssist` is supported for backward compatibility, you should use function `cosimWizard` instead. See Creating a Function, System Object, or Block.

Compatibility Considerations

Replace all existing instances of `edaCosimAssist` with `cosimWizard`.

EDA Simulator Link Supported Software and System Updates

- For FPGA Automation (with Simulink or Filter Design HDL Coder™):
  - Tested with Xilinx ISE 12.1
  - Added Windows 64 support

- For HDL Cosimulation:
  - ModelSim SE 6.6c, 6.5f, 6.4g
  - ModelSim PE 6.6c, 6.5f, 6.4g
  - ModelSim DE 6.6c (Windows 32 only)
  - Questa 6.6c
  - Support for Synopsys Discovery will be removed in a future release.
- Windows 64 Support Added for TLM Component Generation

Functions and Function Elements Being Removed

Function or Function Element Name	What Happens When you use the Function or Element?	Use This Instead	Compatibility Considerations
`edaCosimAssist`	Still runs	`cosimWizard`	Replace all existing instances of `edaCosimAssist` with `cosimWizard`.
`launchDiscovery`	Warns		Support for Synopsys Discovery will be removed in a future release.
HDL Cosimulation block—Discovery	Warns		Support for Synopsys Discovery will be removed in a future release.
FPGA hardware-in-the-loop	Warns	FPGA-in-the-Loop	Support for FPGA HIL will be removed in a future release.

R2010b

Version: 3.2

New Features

Compatibility Considerations

HDL Cosimulation Updates

EDA Cosimulation Assistant Creates Blocks and Functions from Existing HDL Code

Get started quickly using existing HDL code and the EDA Cosimulation Assistant. This tool will guide you through the steps to create a test bench or component function for cosimulation with MATLAB or an HDL Cosimulation block for cosimulation with Simulink. See Generate HDL Cosimulation Interfaces from Existing HDL Code.

Updated Timescales Pane Offers New Options for Simulation Timescale Factoring

The updated timescale features allows you to choose when EDA Simulator Link software should calculate a timescale for you. In addition, you can make changes to the calculated timescale with an interactive GUI. See the HDL Cosimulation block reference for more information.

To VCD File Block Supports Simulation Using Rapid Accelerator Mode

To VCD File block now supports simulation using Rapid Accelerator mode. Select **Simulation** > **Rapid Accelerator Mode** in the model window before beginning cosimulation. See the To VCD File block reference for more information about this feature and block settings.

EDA Simulator Link Supports ModelSim DE

EDA Simulator Link software now supports ModelSim DE 6.6a (for Windows only). See Product Requirements for more information on supported products.

EDA Simulator Link Supports Cosimulation on 64-Bit Windows

This release adds support for Windows 64-bit machines. See Product Requirements for more information on supported products.

HDL Cosimulation Support for Synopsys Updated

With the release of R2010b, EDA Simulator Link software now supports VCS MXi. The software no longer supports VCS MX.

Compatibility Considerations

Attempts to cosimulate with VCS MX will be result in errors.

Zero Value of First Output for All Signals Corrected

Previously, the first output value of all signals in the HDL code was set to zero in Simulink (unless you used direct feedthrough, which works as expected). Effective this release, the output value of all signals now pass from the HDL simulator to Simulink as expected.

Compatibility Considerations

You will no longer see zero as the first output value of all signals in Simulink. You may need to modify your code or procedures to accommodate the corrected behavior.

FPGA Automation Updates

FPGA Project Generation with MATLAB and Filter Design HDL Coder

You can create Xilinx ISE projects from Filter Design HDL Coder and MATLAB. From the Filter Design & Analysis Tool GUI, select **Targets > Generate HDL**. Select the **FPGA Automation** tab. Use context-sensitive help to assist you in setting project generation options.

The software now supports FPGA project generation with MATLAB and Filter Design HDL Coder on 32-bit Windows systems and on 32- and 64-bit Linux systems.

See FPGA Project Generation with Xilinx to get started.

Clock Module Generation Now Supports Verilog

The limitation for DCM design containing only VHDL code has been removed. You may select Verilog on the HDL Coder pane or the Generate HDL pane (in Filter Design HDL Coder) when you specify options for FPGA project generation.

TLM Generation Updates

Single Source and Sink Blocks Now Supported

This release removes the previous limitation that designs under generation must contain both inputs and outputs. Your design now may have only source blocks or only sink blocks.

New Algorithm Processing Options

Choose the type of function execution trigger you want to use in the generated TLM component—SystemC thread or callback. Use a SystemC thread for a more realistic simulation (at the expense of time) and use a callback for faster execution (at the expense of accuracy).

Temporal Decoupling Added to Generated TLM and Test Bench

This release includes complete temporal decoupling implementation with quantum for faster processing for generated TLM and test bench.

R2010a

Version: 3.1

New Features

Support for Latest Synopsys Discovery Release

EDA Simulator Link now supports the latest Synopsys release. See the 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.

Enable Direct Feedthrough for HDL Designs with Pure Combinational Datapaths

The HDL Cosimulation block now supports direct feedthrough, which means that the output is controlled directly by the value of an input port. The input value change propagates to the output ports in zero time, thus eliminating one output-sample delay for HDL designs with pure combinational logic datapaths. This feature eliminates the need to modify the test bench portion of Simulink to compensate for cosimulation block delay.

New Functions for HDL Simulator Client Communication

A new function, `notifyMatlabServer`, allows you to send HDL simulator event and process IDs to MATLAB server. Another new function, `waitForHdlClient`, waits to begin processing until the specified event ID is obtained or a user-specified time-out occurs.

Batch, CLI, and GUI Mode Support Added for Cosimulation with HDL Simulators

You can execute cosimulation in batch mode for background processing or CLI mode for ease in debugging.

Use Same MATLAB Function for Multiple HDL Instances

This release adds a new argument, `use_instance_obj`, to the MATLAB functions `matlabcp` and `matlabtb`. This feature replaces the `iport`, `oport`, `tnext`, `tnow`, and `portinfo` arguments of the MATLAB function definition with an HDL instance object passed to the function as an argument. With this feature, `matlabcp` and `matlabtb` function callbacks get the HDL instance object after it has passed into hold state. They also provide read/write access protection for signals and allow you to add state as you wish.

With this feature you gain the following advantages:

- You can use the same MATLAB function to represent behavior for different instances of the same module in HDL without need to create one-off wrapper functions.
- You no longer need special `portinfo` argument on first invocation.
- You no longer need to use persistent or global variables.
- You receive better feedback and protections on reading/writing of signals.
- You can use object fields to identify the instance path and whether the call comes from a component or test bench function.
- You can use the field argument to pass user-defined arguments from the `matlabcp` or `matlabtb` instantiation on the HDL side to the function callbacks.

The new argument, `-use_instance_obj`, is identical for both `matlabcp` and `matlabtb`. See the Function Reference for `matlabcp` and `matlabtb` for instructions in using this new function argument.

Generating Transaction Level Models for Use with Virtual Platforms

- Export of Simulink algorithm models as OSCI TLM 2.0 components
- Generation of standalone SystemC test bench for generated TLM 2.0 component

Limitations The design under generation must contain both inputs and outputs. Designs that only have inputs or only have outputs (sink or source blocks) are not supported in this release. If you do not include both, EDA Simulator Link displays an error message and discontinues code generation.

Specializing FPGA Implementations

Generation of Xilinx ISE projects for FPGA designs

R2009b

Version: 3.0

New Features

EDA Simulator Link DS, EDA Simulator Link IN, and EDA Simulator Link MQ Merge

As of R2009b, EDA Simulator Link DS, EDA Simulator Link IN, and EDA Simulator Link MQ functionality are merged into a new product, EDA Simulator Link. These individual products are no longer available.