Synopsys®, Inc.
700 East Middlefield Road
Mountain View, CA 94043 USA
Phone: (U.S.) +1 650.584.5000
Website: www.synopsys.com
Support: www.solvnet.com

Synphony Model Compiler Release Notes

Version F-2012.03M Actel Edition, March 2012

Publication Version 01

Contents

About This Release

The Synphony Model Compiler software provides a high level algorithm modeling environment and a powerful DSP synthesis engine that creates optimized implementations of the algorithm in RTL. The modeling environment is based on the MATLAB® Simulink® environment from The MathWorks® with which it is integrated. The RTL (Verilog and VHDL) generated by the software requires Synplify Pro® or Synplify® Premier for logic synthesis into FPGAs.

What's New in This Release

The following table lists the features introduced in this release. For details about these features, refer to the documentation.

Feature	**Details**
New blocks	FIR2 Matrix Mult Reshape
Expanded FIR functionality	The FIR2 block lets you create fixed and reloadable coefficient filters. For this release, this is a Beta feature.
Matrix support	Many blocks now support the matrix data type and matrix input. Refer to the Block Summary appendix in the manual for a list.
Synthesis pragma specification	You can now specify Synplify Pro or Synplify Premier pragmas using tags in Simulink blocks.
Expanded list of demos	Additional demos have been added to the demo directory. See *Library of Application Examples* on page 2.

Library of Application Examples

The library of application examples is now included with the release in the demo directory. These examples are built as custom blocks. The most up-to-date version of this library is available through the Solvnet article called *Synphony Model Compiler Custom Library Examples* (`https://solvnet.synopsys.com/retrieve/030247.html`). See *Using Solvnet to Find Additional Information* on page 11 for details about accessing and using Solvnet.

Platform Support

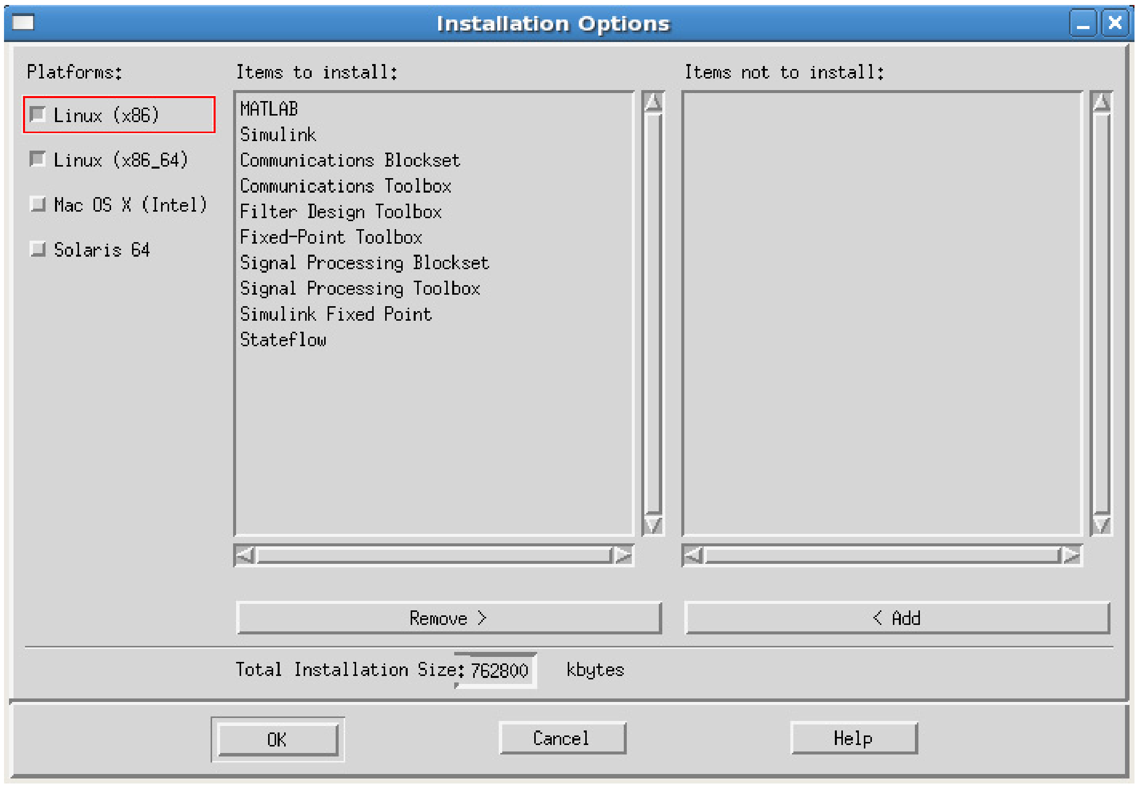
This release of the software supports the following platforms and operating systems:

Windows	Linux
Recommended	
Microsoft Windows 7: 64-bit	Red Hat Enterprise Linux 5: 64-bit
Microsoft Windows XP: 32-bit	SuSE Linux Enterprise Server 11: 64-bit
Supported	
Microsoft Windows 7: 32-bit on 64-bit machine	Red Hat Enterprise Linux 5: 32-bit on 64-bit machine. See *Running MATLAB in 32-Bit Mode* on page 3.
Microsoft Windows XP SP2: 64-bit	SuSE Linux Enterprise Server 11: 32-bit on 64-bit machine. See *Running MATLAB in 32-Bit Mode* on page 3.

Running MATLAB in 32-Bit Mode

To run MATLAB in 32-bit mode, you must set the MATLAB_ARCH environment variable to glnx86 (i.e., setenv MATLAB_ARCH glnx86 for C shell, or export MATLAB_ARCH = glnx86 for BASH shell).

For x86_64, you must select the Linux (x86) platform when you install MATLAB, as shown below:



Items to install:
MATLAB
Simulink
Communications Blockset
Communications Toolbox
Filter Design Toolbox
Fixed-Point Toolbox
Signal Processing Blockset
Signal Processing Toolbox
Simulink Fixed Point
Stateflow

Total Installation Size: 762800 kbytes

Supported Tools

The following table lists the recommended versions of other tools that you can use with the Synphony Model Compiler software. The terms are explained below:

- Recommended
  Tested with this version combination; feature development synchronized with features in these versions.
- Supported
  Some tests run with this version combination; some features may not be supported.
- Compatible
  Minimal or no testing with this version, but should be generally compatible; may require minor workarounds. Some features might not be supported.

	Recommended	**Supported**	**Compatible**
The Mathworks Tools	7.12 (2011A) 7.11 (2010B)	7.13 (2011B)	7.10 (2010A) 7.9 (2009B)
FPGA Logic Synthesis (Synplify Pro/Premier)	F-2012.03	F-2011.09	Older and newer versions, with these exceptions: • Differences in supported devices • QoR improves with newer versions
C Output Compile Environment	Linux: GCC 4.2.2 Windows: Microsoft Visual Studio 2005	Windows: Microsoft Visual Studio 2008 and 2010	Other GCC versions Newer versions of Microsoft Visual Studio
RTL Encapsulation Compile Environment	Linux: GCC 4.2.2 Windows: Microsoft Visual Studio 2005	Windows: Microsoft Visual Studio 2008 and 2010	Linux: Newer GCC versions Windows: Newer versions of Microsoft Visual Studio and express editions of Microsoft Visual Studio
RTL Simulation	Linux: VCS MX D-2010.06 Windows: Riviera 2010.02	ModelSim SE v6.5	Other versions of VCS, Riviera, ModelSim, and ActiveHDL
RTL Code Checking	LEDA D-2010.03		Older versions of LEDA

For the C output and RTL encapsulation environments, also check the MathWorks website for compilers supported by MATLAB:

MATLAB 2010b	http://www.mathworks.com/support/compilers/R2010b/index.html
MATLAB 2011a	http://www.mathworks.com/support/compilers/R2011a/index.html
MATLAB 2011b	http://www.mathworks.com/support/compilers/R2011b/index.html

Installing the Synphony Model Compiler Software

The Synphony product works off a foundation made up of the MATLAB and Simulink design tools. These applications must be installed before the Synphony product. For a list of The MathWorks tool versions supported for this release, see *Supported Tools* on page 4.

Synphony Model Compiler installation is a three-step process: you must first install MATLAB and Simulink, then install Synphony using the platform-specific installation instructions, and then ensure that it is installed in the MATLAB environment. The following procedures explain the details:

- Installing the Software on a Windows Platform, on page 5
- Installing the Software on a Linux Platform, on page 6
- MathWorks Required Features, on page 9

The installation subdirectory name consists of the product name and an associated version number. This permits multiple versions to be installed in the same product directory.

Installing the Software on a Windows Platform

The following procedure shows you how to install the Synphony tool on a Windows platform.

1. Make sure you have one of the supported versions of the MATLAB and Simulink software installed:
   - For supported versions, check *Supported Tools* on page 4.
   - For a list of required MATLAB features that must be installed, see *MathWorks Required Features* on page 9.

2. Install the software.
   - On the Solvnet website, click the Downloads tab and select the Synphony Model Compiler product from the list, to get to the product download page. Select the product release and click Download Here to download the software.
   - Double click the downloaded .exe file and follow the installation instructions.
   - If you want to set up your Windows system as a license server, refer to the instructions in the *Synopsys Common Licensing Administration Guide*, which is available on Solvnet.

3. Set up your license according to your configuration.

   The tool uses the Synopsys Common Licensing (SCL) scheme. See *Synopsys Common Licensing* on page 10 and the INSTALL_README file in your download package for more information. This file is available with the software download when you click the Download Here button to download the tool.

4. Start MATLAB. In MATLAB, change the working directory to C:\Synopsys\Synphony_MC_*version*\mathworks where *version* is the release version string such as E201012.

5. At the MATLAB command line prompt, type setup to execute the setup script (`setup.m` file) in the working directory.

   The script configures the compiler to be used with MATLAB, and which is required for certain tool features like RTL encapsulation and C-output. If you already have a supported compiler installed, select it when prompted. If not, install one of the supported compilers, and then rerun the setup script. See the table in *Supported Tools* on page 4 for a list of supported compilers.

The installation confirms the setup with a popup window. Click OK to close the window.

6. Double check the installation by typing the following at the MATLAB command line:
   - Type shlsroot. The software echoes the path where the Synphony Model Compiler tool is installed.
   - Type path. The window shows the path where the MATLAB software is installed.
   - Check the version number by typing shlsver. You see information about the installed version of the Synphony Model Compiler tool.

7. Check that you have all the necessary MATLAB features installed by typing the following at the MATLAB prompt:

```
setup('check')
```

   This command checks the platform and licenses for the MATLAB features required for the Synphony tool. It lists all the licenses it finds, and generates warnings for any missing licenses.

   For a list of the required features, see *MathWorks Required Features* on page 9.

Installing the Software on a Linux Platform

You can use one of two procedures to install the software on a Linux platform, depending on whether you have write permission to the MATLAB installation. See the following:

- Installing with Write Permissions to the MATLAB Installation, on page 6
- Installing Without Write Permissions to the MATLAB Installation, on page 7

Installing with Write Permissions to the MATLAB Installation

Use this procedure when you have write permission to the MATLAB installation and you want to install Synphony permanently into the MATLAB environment. If you do not have write permission, use the procedure described in *Installing Without Write Permissions to the MATLAB Installation* on page 7.

1. Make sure you have a supported version of the MATLAB and Simulink software from The MathWorks installed.
   - For supported versions, check *Supported Tools* on page 4.
   - For a list of required MATLAB features that must be installed, see *MathWorks Required Features* on page 9.

2. Install the Synphony Model Compiler software.
   - For a Linux installation, you must first download the Synopsys Installer, if you have not already done so for other Synopsys tools. You must have version 2.7 or later of the Synopsys Installer to install the Synphony Model Compiler software. For more information about downloading the Synopsys Installer, see *Licensing* on page 10. Use the Synopsys Installer to install the Synphony Model Compiler tool.
   - For a Windows installation, download the software and click the .exe file to install it.

   - If you are setting up a new Synopsys license server, download the SCL licensing software and complete the installation as described in *Licensing* on page 10. Note that you require two separate tar files, common.tar and linux.tar.

3. Set up your licenses, as described in *Synopsys Common Licensing* on page 10 and *Retrieving License Keys* on page 10.

4. Start MATLAB. In MATLAB, change the working directory to *installDirectory*/*synphony_mc_version*/mathworks where *synphony_mc_version* is the Synphony Model Compiler release version string.

   If you had a previous version of the Synphony Model Compiler tool installed, you must start a new MATLAB session.

5. Type setup to execute the setup script (`setup.m` file) in the working directory.

   Note that the setup script adds some paths to certain scripts in the MATLAB installation directory. If you do not have write permission to the MALAB installation, refer to *Installing Without Write Permissions to the MATLAB Installation* on page 7 for information about manually adding the addpath commands.

   The script tries to configure the compiler to be used with MATLAB. The compiler is required for some features, like RTL encapsulation and C-output. If you have a supported compiler select it when prompted; otherwise install one of the supported compiler and then rerun the setup script. See *Supported Tools* on page 4 for supported compilers.

   The installation confirms the software location. Click OK to close the popup window.

6. Double check the installation by typing the following at the MATLAB command line:
   - Type shlsroot. The software echoes the path where the Synphony Model Compiler tool is installed.
   - Type path. The window shows the path where the MATLAB software is installed. Check the version number by typing shlsver. You see information about the installed version of the Synphony Model Compiler tool.

7. Check that you have all the necessary MATLAB features installed by typing the following at the MATLAB prompt:

   ```
   setup('check')
   ```

   This command checks the platform and licenses for the MATLAB features required for the Synphony tool. It lists all the licenses it finds, and generates warnings for any missing licenses. For a list of the MATLAB features, see *MathWorks Required Features* on page 9.

Installing Without Write Permissions to the MATLAB Installation

Use this procedure to install Synphony when you do not have write permission to the MATLAB installation. If you do have write permission, use the procedure described in *Installing with Write Permissions to the MATLAB Installation* on page 6.

1. Follow steps 1 to 3 from *Installing with Write Permissions to the MATLAB Installation* on page 6, but install the Synphony Model Compiler tool in a directory where you have write permission. This directory is *installDirectory*/synphony_mc_*version* where *version* is the Synphony release version string; for example, F201109. This directory is referred to as *SMC_home* in the remainder of this procedure.

2. Start MATLAB.

3. To add Synphony to the MATLAB environment, type the following commands at the command line or add them to a startup script file such as `startup.m`. If you use a startup script, save it in the working directory.

   ```
   addpath('<SMC_home>/mathworks/toolbox/Synopsys/SynphonyHLS')
   addpath('<SMC_home>/mathworks/toolbox/Synopsys/SynphonyHLS/
      MATLAB<number>/<platform>')
   addpath('<SMC_home>/mathworks/toolbox/Synopsys/SynphonyHLS/demos')
   dspstartup
   ```

   <SMC_home> is the absolute path of the directory where you installed the Synphony Model Compiler software.

   MATLAB*<number>* is the MATLAB directory that matches the version of MATLAB you are running.

   <platform> is `linux` or `linux_a_64` for Linux or `win32` or `win64` for Windows to match the platform and operating system on which you are running the software.

   Note the following:

   – Directory order is important, so specify the commands in the order shown. If you do not specify them in this order, some Synphony Model Compiler blocks are not initialized and will not run through simulation. If you list the paths in MATLAB, the paths will be listed in reverse order from the way you specified them, but you have set your paths correctly.

   – The dspstartup command is optional but often gives better simulation performance. It is a Simulink function that sets the simulation engine to an optimal configuration for discrete-time systems.

4. For RTL encapsulation and C-output, you must set up and configure a C compiler for MATLAB. Run the `mex -setup` command and select the appropriate compiler from the populated list. If you need to, first install a supported compiler. For a list of supported compilers, see *Supported Tools* on page 4.

5. Double check the installation by typing the following at the MATLAB command line:

   – Type shlsroot. The software echoes the path where the Synphony Model Compiler tool is installed.

   – Type path. The window shows the path where the MATLAB software is installed.Check the version number by typing shlsver. You see information about the installed version of the Model Compiler software.

6. Check that you have all the necessary MATLAB features installed by typing the following at the MATLAB prompt:

   ```
   setup('check')
   ```

   This command checks the platform and licenses for the MATLAB features required for the Synphony tool. It lists all the licenses it finds, and generates warnings for any missing licenses. For a list of the MATLAB features, see *MathWorks Required Features* on page 9.

MathWorks Required Features

Your MathWorks installation must include the following features for the Synphony Model Compiler tool to run correctly. Make sure you install all the toolboxes and features described below.

Feature	Description
MATLAB	This is the main MathWorks desktop, and has scripting capabilities for algorithm development and data analysis.
Simulink	Provides an interactive graphical environment and a customizable set of block libraries that together let you accurately design, simulate, implement, and test systems. It has a built-in notion of time (continuous and multi-rate discrete).
Fixed Point Toolbox	Provides fixed-point data types and arithmetic in MATLAB. Use it to develop algorithms for testing, modeling, and verifying your fixed-point implementations. It also includes an interface from Simulink Fixed Point to MATLAB Fixed Point workspace.
Simulink Fixed Point	Enables the fixed-point data type in Simulink. Use it to execute fixed-point simulation in Simulink, Stateflow, the Signal Processing Blockset, and the Video and Image Processing Blockset. It includes tools to identify overflow and saturation errors.
Filter Design Toolbox	Required for R2010B and prior releases. Starting with R2011A, these features have been merged into the DSP System Toolbox. Lets you design, simulate, and analyze fixed-point, adaptive, and multirate digital filters. It extends the capabilities of the Signal Processing Toolbox with filter architectures and design methods, like adaptive and multirate filtering. When used with the Fixed-Point Toolbox, the Filter Design Toolbox provides functions that simplify the design of fixed-point filters and the analysis of quantization effects.
Signal Processing Blockset	Required for R2010B and prior releases. These features are now merged into the DSP System Toolbox. Provides a foundation blockset for DSP simulation in Simulink. It extends Simulink functionality with efficient frame-based processing and blocks for signal processing systems.
Signal Processing Toolbox	Required for R2010B and prior releases. These features have now been merged into the DSP System Toolbox. Lets you perform signal processing, analysis, and algorithm development. The Signal Processing Toolbox is a collection of industry-standard algorithms for analog and digital signal processing. It provides graphical user interfaces for interactive design and analysis and command-line functions for advanced algorithm development. It includes basic filter functionality (with FDATool).
DSP System Toolbox	Required for releases from R2011A onwards. Provides algorithms and tools for the design and simulation of signal processing systems.

Licensing

The Synphony Model Compiler tool uses the Synopsys licensing daemon, snpslmd. (See `http://www.synopsys.com/Support/Licensing/Licensing/Pages/default.aspx`).

Details about licensing are explained here:

- Synopsys Common Licensing, on page 10
- Retrieving License Keys, on page 10
- License Variable Settings, on page 11

Synopsys Common Licensing

The following procedure summarizes what you need to do. The INSTALL_README file in the download package (available when you click the Download Here button) goes into more detail and is more comprehensive, if you need additional information.

1. For Linux-based software, make sure you have version 2.7 or later of the Synopsys Installer.

   For details about downloading the installer, see *Installing with Write Permissions to the MATLAB Installation* on page 6. For Windows-based tools, you do not require the Synopsys Installer. Just use the InstallShield directly from the downloaded .exe file, as described in the next step.

2. Download the Synopsys Common Licensing (SCL) software by doing the following;
   - Go to the download center (`https://solvnet.synopsys.com/DownloadCenter/dc/product.jsp`)
   - Select Synopsys Common Licensing from the list of product releases, and select version 11.1 or later and follow the download instructions.
   - For Windows, download the .exe file. Click on it and follow the installation instructions to install the SCL software.
   - For Linux, download the common.tar and linux.tar files. The common.tar file contains platform-independent information and the linux.tar file contains platform-specific information. Install the SCL software using the Synopsys Installer.

3. A new license file is required for this release of the Synphony tool. License files are available from the Smart Keys website (see *Retrieving License Keys* on page 10).

Retrieving License Keys

A new license key file is required with this release of the Synphony product. To retrieve your new, site-specific license key file:

1. Log in to `http://www.synopsys.com/smartkeys`.
2. When prompted, enter your Synopsys SolvNet user name and password.
3. On the SmartKeys main screen, click Key Retrieval.

4. On the Key Retrieval page, enter your Synopsys site ID.If you do not know your site ID, you can get this information in one of these places:
   – View the site_info file in your tool tree ($SYNOPSYS/admin/license/site_info). The SiteID field identifies your site ID.
   – Open an existing Synopsys key file. Near the top, you will find your site ID.

```
     # SYNOPSYS INC. KEY CERTIFICATE
     # Site Id: xxx [where xxx is the site ID number]
```

5. Click Retrieve Licenses.

   SmartKeys processes your request and sends you the license key file by email.

License Variable Settings

After upgrading to the Synopsys Common Licensing software, set one of the following variables to point to the license key file:

LM_LICENSE_FILE=*port@licenseServerName*

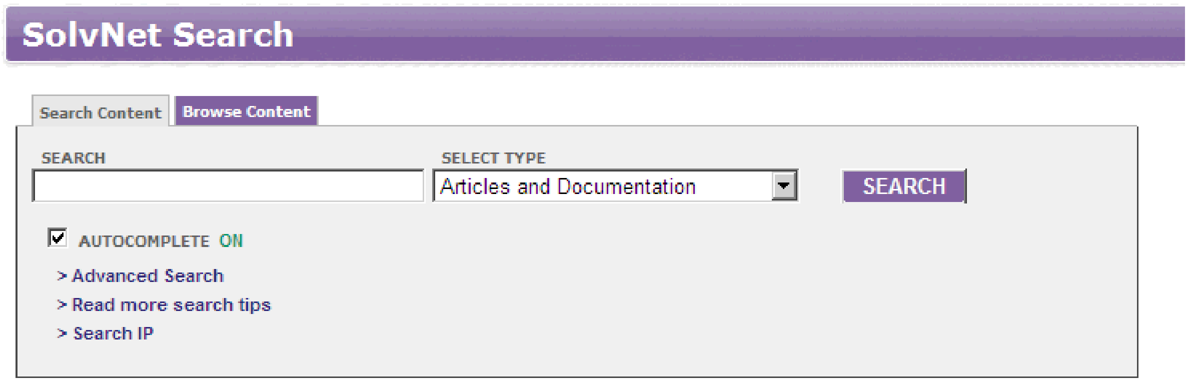
SNPSLMD_LICENSE_FILE=*port@licenseServerName*

Note that the SYNPLCTYD_LICENSE_FILE variable is no longer supported.

Using Solvnet to Find Additional Information

The Synopsys Solvnet database includes the manual for the product, as well as tutorials, release notes, application notes, articles, white papers, and product-specific FAQs and issues. The following procedure shows you how to access documentation related to the Synphony tool:

1. Go to Solvnet (www.solvnet.com) and log in.
2. On the Home page, click Browse Content. Set Select Type to Articles and Documentation to locate tutorials and application notes.



3. Scroll down to the product you want, and click on it.

   This opens the Articles by Product page.

4. Click on the type of article you want to view. For example, if you click Application Note, you see a list of available application notes.

Known Problems and Solutions

The issues have been categorized as follows:

- MathWorks Interface, on page 12
- Synphony Model Compiler Tool, on page 15
- Platform-Specific Issues, on page 19

MathWorks Interface

This section describes tool compatibility issues between the Synphony tool and the MathWorks tools, MATLAB and Simulink.

Smart Black Box HDL Cosimulation Error

When you run HDL cosimulation using the Synphony Smart Black Box block with MATLAB R2009a (or older versions) and ModelSim 6.5, you might get an error message like this one:

```
Error reported by S-function 'shdlcosim' in 'my_cosim/HDL Cosimulation/S-
function':
   The following signal names do not exist in the loaded HDL model.
   /top/sig1
   /top/sig2
   /top/sig3
   Check HDL simulator waveform window for full hierarchical names.
```

Solution: This is a known limitation of the EDA Simulator Link interface and is documented on the Mathworks web site (bug number 531802). MATLAB versions R2009a and older are not compatible with ModelSim version 6.5. These versions officially support ModelSim 6.4, 6.3, and 6.2.

There is a patch on the Mathworks web site that lets you use ModelSim 6.5 with R2009A. ModelSim version 6.5 is also officially supported with the EDA Simulator Link interface in R2009B. The following summarizes your HDL cosimulation options with the different MATLAB versions when you use the Synphony Smart Black Box:

MATLAB 2008B and older versions	Use ModelSim versions 6.4, 6.3 or 6.2 for HDL cosimulation. There is no available patch to work with ModelSim 6.5. To use ModelSim 6.5, you must upgrade to the latest MATLAB version.
MATLAB 2009A	ModelSim versions 6.4, 6.3 or 6.2 work successfully for HDL cosimulation. If you want to use ModelSim 6.5, use the patch available from the Mathworks web site.
MATLAB 2009B	You can use ModelSim versions 6.2, 6.3, 6.4 and 6.5 for HDL cosimulation.

MathWorks Installation Fails

A MathWorks installation can fail if during the installation of the Synphony Model Compiler tool or any other tool, the LM_LICENSE_FILE variable points to an infrastructure that does not include a MathWorks license. In such a case, the MathWorks software will not be able to do a full installation.

Solution: Unset LM_LICENSE_FILE during the installation of MathWorks. Refer to the *Installation and License Configuration* document for information about using the LM_LICENSE_FILE and SNPSLMD_LICENSE_FILE variables.

FDATool Compatibility Between MathWorks Versions

When you migrate a design that contains the FDATool block from one MathWorks release to another, you could get a Simulink error message about the need to redesign the filter. You get this message because of embedded data compatibility in MathWorks; the older Simulink library is not compatible with the newer library.

Solution: Select the FDATool instance that was captured with an older version of Simulink. Double-click the instance, and check that your parameter settings are still all intact. Make a small change, undo the change and then click Design Filter, and close the window. The filter coefficient database is updated to the new format required by the new version of Simulink.

Slow Initialization and Startup in Mathworks

Mathworks initialization and startup is slow.

Solution: In addition to setting your search path as described in *Error Message: @E: DSP Optimization/RTL Generation Failed* on page 17, you can speed up the initial startup time for the MathWorks tool by following these guidelines:

- Keep the LM_LICENSE_FILE environment variable to just 1 or 2 entries. Make sure there are no extra spaces. Refer to the *Installation and License Configuration* document for information about using the LM_LICENSE_FILE and SYNPLCTYD_LICENSE_FILE variables.
- Try to migrate vendor licenses to their respective <*VENDOR*>_LICENSE_FILE variables.

Simulink Data Type Errors in Designs with Loops

In a design with loops, if the data type is not explicitly stated along the loop, you can get unexpected errors during the data type propagation stage of Simulink. This is because there can be a conflict between the implicit propagated data type over the loop and the propagated data type driven back to the loop.

Solution: It is a good design practice to insert a Convert block into the loop, and explicitly cast its output data format to the desired data type.

Simulink Accelerator Mode

The Synphony Model Compiler software does not currently support Simulink Accelerator mode.

Solution: This will be fixed in a future release.

Simulink Fails to Open .mdl File

When MathWorks .mdl files with Synphony designs are exchanged between teams in different regions, Simulink sometimes does not recognize the files when you try to open them, and displays the following message: "*designName*.mdl is not a valid design name or it does not exist."

Solution: This is related to the different character encodings possible on different machines in different regions. To solve this problem, do the following:

1. Open the MDL file as text, and search for "slCharacterEncoding." Check the embedded character encoding.

2. At the MathWorks command prompt, type slCharacterEncoding. This shows the currently active character encoding, and they should match for the file to open.

3. If the two do not match, close all open Simulink models, go to the command prompt and type slCharacterEncoding('<*desiredCharacterEncodingFromMDLFile*>')

   The MDL file should open without problems now. For more information, check the information using help slCharacterEncoding from the command line.

Simulink Blocks Move When an Invalid Value is Entered in the Mask Dialog Box

There is a Simulink bug that occurs in masked subsystems initialized by M scripts. If you add a block with a position declaration in the M script and enter an illegal mask value which is used by add_block inside the script, the add_block operation fails. However, the position declaration moves the parent block to the place where the added block was supposed to be placed.

Solution: You can use any of the following methods:

- Remove mask callbacks which do set_params operations.
- In the M script, separate the add_block from the parameter settings for the added block. First, use add_block with just the block name to define the block. Then, set the position and other parameters separately with set_params.
- Contact support@mathworks.com.

Large Models Crash Simulink Environment

If you specify unrealistic sizes of RAM, ROM or FIFO, the Simulink models crash the environment.

Solution: Use realistic sizes that are a couple of MB.

Verilog-C Interface Wrapper Scripts and VHDL-Only Implementations

The generated VPI script does not function when used in a VHDL-only implementation. This limitation occurs because the script files exclusively uses test benches from the Verilog folder. In a VHDL-only implementation, the test bench is not generated. This means that the script references a non-existent test bench, which results in a failure during simulation.

Solution: Enable a Verilog implementation to generate the Verilog testbench for Verilog-C interface wrapper scripts.

Synphony Model Compiler Tool

This section describes known issues in the Synphony Model Compiler tool.

RTL Encapsulation Configuration File Not Reflected In RTL

If the configuration file of RTL Encapsulation block is modified and the model is synthesized either without updating the model or running simulation, the generated simulation model may not reflect the recent changes.

Solution: Whenever the RTL file or configuration file included in the RTL Encapsulation block is modified, do a model update or simulation run before synthesizing the model.

Crash After Error During Model Update

If there are errors in the model during model updates or simulation runs, intermittent crashes might occur.

Solution: The tool displays an error message is before the crash. Correct the error and run the model again.

RAM Block Messages

You might see the following RAM block check messages when you specify RAM blocks:

```
Prioritized write access control is not allowed for multi-rate RAM
```

```
Address ports of the RAM block should have the same integer length
```

Solution: You see these messages if you did not follow the rules for specifying RAM blocks. The following describe the situations for each message:

`Prioritized write access control is not allowed for multi-rate RAM`	You see this message if you enable the Write Prioritization option when the read and write ports of the RAM have different sample rates. When the read and write ports have different sample rates, the tool writes out a multi-rate RAM in the RTL, which cannot use write access control logic. To avoid this message, only enable the Write Prioritization option when both ports have the same rate.
`Address ports of the RAM block should have the same integer length`	You see this message when the signals that feed the address ports of the RAM do not have the same word length. To avoid it, make sure that all address ports of the RAM block use signals with the same data type.

Component Naming

The generated code may not successfully compile in Simulink designs that have signals, blocks, ports, or subsystems that share the same name as another signal, block, port or sub-system.

Solution: Use unique names for models, ports, signals, blocks, and subsystems.

Illegal Character in Subsystem or Signal Names

If a model designed with an earlier Synplify DSP version of the software has subsystem or signal names that contain an illegal character, the tool will not successfully generate RTL code. This is because the subsystem consolidation and signal tracing features introduced in the 3.8 version of the software require legal names.

Solution: Replace the illegal characters in the subsystem or signal names with legal ones.

Infeasible Path Error

You might get an infeasible path error if you have a zero-offset Downsample block immediately followed by an Upsample block, and if you choose either the Hold input sample option of the Upsample block or if the sample offset of the Upsample block is less than the upsample rate divided by the downsample rate. This is because the zero-latency implementation of the zero offset Downsample block means that its output is not valid for the first fast clock cycle. So the tool does not allow any Upsample blocks to sample the Downsample block output during this period.

You might also get this infeasible path error if you have a zero-offset Downsample block immediately followed by a Commutator or a multi-rate Mux block, as the implementation of these blocks require autogenerated Upsample blocks with zero offsets.

Solution: Insert a delay element on the infeasible path between the Downsample block and the subsequent block (Upsample/Commutator/Mux), or set the offset of the Downsample block to a non-zero value.

If the infeasible path is between a Downsample and an Upsample block you can also set the offset of the Upsample block to a non-zero value.

If possible, enable Retiming. When this optimization is enabled, the tool automatically tries to fix the infeasible path by inserting delay elements in to the path.

Using Custom User Libraries

When you use the new version of the software, you might not be able to update older designs which use your own custom libraries, especially if these libraries include Convert, Add, Mult, Gain, FFT, FIR, FIR Engine, FIR Rate Converter, or RFIR blocks.

Solution: First update you user library manually. Then start the tool and use the current version of the Synphony Model Compiler library. You must convert the initialization scripts for these blocks properly before updating the custom libraries, because the block parameters have new quantization capabilities.

License Reporting

While the software supports both LM_LICENSE_FILE and SYNPLCTYD_LICENSE_FILE environment variables, the FLEXnet lmutil lmstat command only reports the licenses in LM_LICENSE_FILE.

Solution: If you want to use the lmutil lmstat command for diagnostics, use LM_LICENSE_FILE to point to the license. However, this could result in slow initialization, as described in *Slow Initialization and Startup in Mathworks* on page 13.

Retiming Engine Might Crash with Some Design Topologies

The retiming engine might crash because of computation overflows if your design meets one or more of the following criteria: it has very long depth delays scattered around, and/or it is a multi-rate design with several high clock rate changes.

Solution: This will be fixed in an upcoming release. The current workaround is to simplify the design as much as possible by trying different algorithmic approaches.

RTL Generation Fails with Error Message

RTL generation fails with the following error message:

@E: Error: Could not find block *blockName* in the side information file. Side info file may be out of date. Please run the simulation before attempting to generate RTL.

Solution: Typically this occurs because the block name uses characters that are not supported for RTL generation (for example, ( ), and [ ]). Remove the offending characters from the name and rerun RTL generation.

Block Output Mismatch

On some blocks like Add and Gain, the size of the input fraction length can cause internal operators to require more than 128 bits, and you can get a mismatch on the output.

Solution: Simulink fixed point data type supports fixed point numbers up to 128 bits. Be aware of internal word growth required for different operands and set the inputs so that the final output does not exceed 128 bits.

syn_get_coefs Script Does Not Support Second Order Sections

The MathWorks FDA tool decomposes the IIR filter into second order sections (SOS) instead of one single section with feedback and forward coefficients, and the syn_get_coefs script does not currently support this.

Solution: Use Edit->Convert to Single Section to change it to a single section. The syn_get_coefs script can handle a single section. This will be fixed in a future release.

Error Message: @E: DSP Optimization/RTL Generation Failed

Sometimes, the synthesis run fails with this error message "@E: DSP Optimization/RTL Generation failed," because of a license access problem. If the Mathworks desktop is taking a long time to initialize at startup, it is looking for a license (which can take a couple of minutes). This can also interfere with the invocation of the synthesis algorithms.

Solution: Set the LM_LICENSE_FILE variable so that the MathWorks license is first in the path. The typical location of the MathWorks license file is *MATLABInstallationDirectory*/bin/win32/license.dat. Note that the MathWorks specific MLM_LICENSE_FILE is checked after the LM_LICENSE_FILE, so that can make matters even worse. See *Slow Initialization and Startup in Mathworks* on page 13 for additional ways to speed up initialization.

Underscore Characters in the Generated VHDL

The Synphony Model Compiler tool replaces the following illegal characters in a port or instance name with the underscore character.

'.' Dot	'/' Slash
'\n' New line	' ' Space
'(' Parenthesis, open	'[' Square bracket, open
')' Parenthesis, close	']' Square bracket, close

If the illegal character is the first character in an HDL level name, the generated VHDL code will begin with an underscore, which is illegal in VHDL. For example, if the port name is ' Output' in the Simulink MDL file, the Synphony Model Compiler tool generates VHDL with the port name _Output.

Solution: Remove the leading or trailing space in the Simulink model instance or port name. You can check for leading and trailing spaces by clicking on the name in Simulink and moving the cursor to the beginning of the name.

RTL-Aldec Simulation Mismatch During Initialization of the FIR Filter

You can get RTL-Aldec simulator mismatches when folded FIR filters are initialized, because of initialization to zero.

Solution: You can work around this by adding the -dbg switch when specifying compilation. For example:

```
acom -dbg "test.vhd" "test_Test.vhd"
```

Verilog and VHDL Keywords

If you use Verilog or VHDL keywords as names for ports or instances, the generated Verilog or VHDL code does not compile.

Solution: Make sure that the ports and instances in your designs do not have names that are the same as Verilog or VHDL keywords.

Currently Unsupported Features

The following features are not currently supported. Their support will be addressed in future releases:

- Signal clock offsets are not fully supported. The software does not support an offset for sample-rate definition. Simulink does not allow a phase offset for any multi-rate situations (using Upsample or Downsample blocks), so Synphony Model Compiler does not support it either.
- Multiport RAM is not supported for Actel targets. If you generate RTL for such blocks, they can fail synthesis.
- Asynchronous read/write RAM blocks are not supported for Actel targets. If you generate RTL for such blocks, they can fail synthesis.

Platform-Specific Issues

This section describes platform-specific issues in the Synphony Model Compiler tool.

Inconsistent License Mode Error (Linux)

When you launch synthesis with SHLSTool, you might see an Inconsistent License Mode error. This known problem occurs when the default shell is set to /bin/tcsh.

Solution: Set an environment variable to point to the sh shell. For example: `setenv MATLAB_SHELL /bin/sh`.

VPI Simulation on 64-bit SUSE Might Not Work

The tool might not work when you run VPI simulation using `vcs.ksh` on the SUSE 64-bit platform with Linux 4.2.2 GCC.

Solution: This only occurs when you use the Linux 4.2.2. GCC distribution for VPI simulation on the SUSE 64-bit platform. Work around this by using the native GCC available in SUSE, instead of 4.2.2 GCC.

Synopsys, Inc.
700 East Middlefield Road, Mountain View, CA 94043 USA
Phone: +1 650 584-5000 or +1 800 541-7737
www.solvnet.com