Synopsys®, Inc.
690 East Middlefield Road
Mountain View, CA 94043 USA
Website: www.synopsys.com

Synphony Model Compiler Release Notes

Version L-2016.03M-SP1, June 2016

Publication Version 01

Contents

About This Release

The Synopsys® Synphony Model™ Compiler (SMC) 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 the Synplify Pro® or Synplify® Premier tools for FPGA logic synthesis.

What's New in This Release

Release L-2016.03M-SP1 contains software improvements for the Synphony Model Compiler tool, which include the following:

- SMC blockset library updates
- RTL generation improvements
- Support for MATLAB R2016A

Library of Application Examples

A library of application examples is included with the release in the mathworks/toolbox/Synopsys/SynphonyHLS/demos/APPEX directory.

Discontinued Support for the SMC Blocks

For the L-2016.03M-SP1 release, the beta versions for the SMC blocks in the table below are available in the library blockset, but will no longer be supported:

ArcSinCos	Barker Code Generator	Barrel Shifter
Burst FFT	CIC2	CORDIC2
Counter2	DIT R2SDF FFT	Divider2
Dynamic Access Shift Register	Dynamic Farrow Resampler	FIR Resampler
FP Burst FFT	FP IIR Section	Frame FIR
Frame Mean	Interleaver2	Leading Zero Counter
Loadable Shift Register	Min Max Filter	MPSK Baseband Modulator
MQAM Baseband Demodulator	MQAM Baseband Modulator	Parallel CIC2
Parallel DDS2	Period Pulse Generator	Pulse Extender
RAM Based Delay	Single Clock Downsample	Single Clock Upsample

Note that these blocks will be removed from the SMC blockset in a future release.

SMC FFT2 Block Documentation Update

The order of the SMC FFT2 block output data is documented with details below:

For a serial input FFT2 block, the output is in bit-reversed order. For a parallel input FFT2, the tool uses the following calculation: For an FFT2 block with transform length N and K complex parallel inputs, the output is a 2K vector. The first K outputs (columns) are the real components and the next K outputs (columns) will be the imaginary components of the FFT.

Within each component (column), an output value of (bitreverse (i*k) + p) is available at time instance I, and this is the pth element of this output vector: 0 <= I < N/K, 0 <= p<K. Also, the columns corresponding to real and imaginary components are in bit-reversed order.

This code example shows a frame for the FFT2 block output data captured as a workspace variable and converted to natural order serial output.

```
%% Bit reverse one frame of FFT2 data. This does not work on streaming input.
One whole frame data must feed the vector.
% N - Frame size
% P - Number of parallel inputs
function naturalOrderFrame = bitrevorder_parallel_frame(data, frameSize)
% The first K columns are the real components and the next K columns will be
% the imaginary components of the FFT2. K is the number of parallel inputs.
   noParallelInputs = size(data, 2)/2;

   % Natural order will have actual frame size. One column is for real and
   % one for imaginary data
   naturalOrderFrame = zeros(frameSize, 2);

   %Get bit-reversed indices
   parallelFrameSize = frameSize/noParallelInputs;
   bitrevorderindices = bitrevorder(1:parallelFrameSize);

   %The columns are also in bit-reverse order
      for p=bitrevorder(1:noParallelInputs)
         for i=1:(frameSize/noParallelInputs)
            % For real data
            naturalOrderFrame(((p-1)* parallelFrameSize)+i, 1) =
               data(bitrevorderindices(i), p);
            % For imaginary data
            naturalOrderFrame(((p-1)* parallelFrameSize)+i, 2) =
               data(bitrevorderindices(i), p+noParallelInputs);
         end
      end
   end
```

Platform Support

This release of the software supports the following platforms and operating systems. The Recommended column lists the recommended versions, and the versions in the Supported column work, but might not support some features. For some older versions, certain features might not be supported, or workarounds might be required.

Recommended	Supported
Microsoft Windows	
Windows 7: 64-bit OS	Windows 8.1: 64-bit OS
Linux	
Red Hat Enterprise Linux 6: 64-bit OS RHEL 5: 64-bit OS	
SuSE Linux Enterprise Server 11: 64-bit OS	

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	2016a	2015b, 2015a, 2014b	Versions prior to 2014b are not compatible.
FPGA Logic Synthesis (Synplify Pro/ Synplify Premier)	L-2016.03-SP1	L-2016.03	Older and newer versions, with these exceptions: • Differences in supported devices • QoR improves with newer versions

	Recommended	Supported	Compatible
C Output Compile Environment **RTL Encapsulation Compile Environment**	Linux: GCC 4.7.2 Windows: Microsoft Visual Studio 2013. See *C Compiler Tool for C Output and RTL Encapsulation* on page 5 for details.	Windows: Microsoft Visual Studio 2010	Linux: Other GCC versions Windows: Microsoft Windows SDK 7.1
RTL Simulation	Linux: VCS MX K-2015.09	ModelSim SE 6.5	Other versions of VCS MX, ActiveHDL, Riviera, and ModelSim
RTL Code Checking	LEDA D-2010.03		Older versions of LEDA

C Compiler Tool for C Output and RTL Encapsulation

MATLAB does not include a C compiler with the 64-bit platform, so you must install this tool separately. The SMC tool requires the compiler to generate C output and for RTL encapsulation. These are the recommended tools to use:

- 32-bit Operating Systems

  Download and install Visual C++ from http://www.microsoft.com/visualstudio/en-us/products/2013-editions/visual-cpp-express.

- 64-bit Operating Systems

  Download and install the following in the order listed:

  – Visual C++ 2013 with 64-bit libraries from http://www.microsoft.com/visualstudio/en-us/products/2013-editions/visual-cpp-express.

  – Microsoft Windows 7 SDK from http://www.microsoft.com/en-us/download/details.aspx?id=8279.

See the MathWorks website for the most current information about compilers supported by MATLAB:

MATLAB R2016A	http://www.mathworks.com/support/compilers/R2016a/index.html
MATLAB R2015B	http://www.mathworks.com/support/compilers/R2015b/index.html
MATLAB R2015A	http://www.mathworks.com/support/compilers/R2015a/index.html

Installing the Synphony Model Compiler Software

The Synphony product is based on a foundation of the MATLAB and Simulink design tools. These applications must be installed before the Synphony product.

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 the SMC tool is installed in the MATLAB environment. The following procedures explain the details:

- Setting up the SMC Software for Microsemi FPGA Design, on page 6

- Setting up the SMC Software Under MATLAB, on page 7
- 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.

Setting up the SMC Software for Microsemi FPGA Design

The following procedure describes how to install and set up the tool in your environment. You must install the other required tools, set up the Synphony Model Compiler software, and integrate it into the MATLAB environment.

Setting up Other Required Tools

1. Make sure you are working with a recommended operating system.

   The recommended operating system is 64-bit Windows 7, but check *Platform Support* on page 4 for other operating systems.

2. Install Libero.
3. Check that you have access to the Synplify Pro or Synplify Premier logic synthesis tool.
4. Install MATLAB and Simulink.
   - The recommended version is the 64-bit 2016A version of the software. See *Supported Tools* on page 4 for information about other versions.
   - Make sure your setup includes all the required modules described in *MathWorks Required Features* on page 9.
5. Download and install Visual C++ with 64-bit libraries, as described in *C Compiler Tool for C Output and RTL Encapsulation* on page 5.

Setting up the Synphony Model Compiler Software

After you have set up the other tools, do the following to install the Synphony Model Compiler software and set up your license:

1. Download the Synphony Model Compiler software from the Microsemi download page: `http://www.actel.com/download/software/dsp/default.aspx`.
2. Set up your license:
   - Obtain the SMC license from `https://www.actel.com/portal/default.aspx?r=1`. You get a license.dat file with the SMC license features.
   - Download the license daemons from `http://www.actel.com/products/software/libero/licensing.aspx#daemons`.
   - Follow the Synphony Model Compiler setup instructions in the Libero Install Guide. The license.dat file also includes instructions on setting up the license server and the environment variables.
3. Run the SMC executable you downloaded.

   You can now set up the SMC software to run under MATLAB, as described in *Setting up the SMC Software Under MATLAB* on page 7.

Setting up the SMC Software Under MATLAB

Once you have installed the Synphony Model Compiler tool and set up the licensing, use the following procedure to integrate the tool into the MATLAB environment.

1. 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 a previous version of the Synphony Model Compiler tool was installed, you must start a new MATLAB session.
   - Type setup to execute the setup script (`setup.m` file) in the working directory.

     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, manually add the addpath commands to the setup script:

```
addpath('<SMC_home>/mathworks/toolbox/Synopsys/SynphonyHLS/demos/APPEX/APPEXFIR')
addpath('<SMC_home>/mathworks/toolbox/Synopsys/SynphonyHLS/demos/APPEX/Appex_scripts)
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.

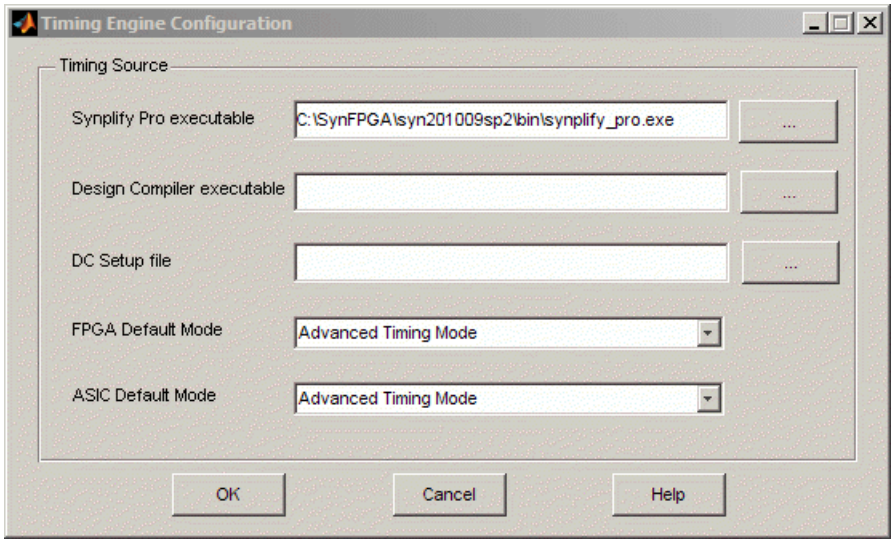
     MATLAB*<number>* is the MATLAB directory that matches the version of MATLAB you are running.

     <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.

     The tool then opens the Timing Engine Configuration dialog box. where you specify the timing mode you want to use and the timing engine to be used for advanced timing mode.

2. Specify the following in the Timing Engine Configuration dialog box:
   - If you want to use Advanced Timing Mode, where the timing engines from the logic synthesis tools are used for timing estimates, set the Default Mode options to Advanced Timing Mode. Specify the option that corresponds to the target technology for your design.

     You can also leave this option set to Estimation Mode for now and reset it later by specifying the syn_set_atm function at the command line.
   - Specify a path to the timing engine to be used for Advanced Timing Mode. For FPGA targets, specify the path to the Synplify Pro or Synplify Premier executable.



3. When prompted for a MEX compiler, specify a supported compiler and confirm the selection.

   The compiler is required for certain tool features like RTL encapsulation and C-output. You can also run the `mex -setup` command later in the process to change the compiler.

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

4. 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.
   - 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	Required for R2010B and earlier releases. Starting with R2013A, these features have been merged into the Fixed-point Designer 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.
Fixed Point Designer Toolbox	Required for releases from R2013A onwards. Provides data types and tools for developing fixed-point algorithms in MATLAB code, Simulink models and stateflow charts. It automatically proposes fixed-point data types and attributes, such as word length and rounding mode.
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.
Signal Processing 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	Provides algorithms and tools for the design and simulation of signal processing systems.

Known Problems and Solutions

The issues have been categorized as follows:

MathWorks Interface Issues

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

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 14, 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 SNPSLMD_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 good design practice to insert a Convert block into the loop, and explicitly cast its output data format to the desired data type.

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 test bench for Verilog-C interface wrapper scripts.

HLS Subsystem Block in R2013A

If you use the HLS Subsystem block in MATLAB 2013a, the model fails to update.

Solution: Use MATLAB version 2013b or 2012b if your design contains a HLS Subsystem block.

Synphony Model Compiler Tool Issues

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

ASIC RAM Extraction from Generated RTL for VHDL

You can use one of the following methods to extract RAM:

- Extracting RAM from generated RTL
- Writing RAM modules to separate files

  If you use the Extracting RAMs from Generated RTL option, the RTL can have incorrect library references in the generated VHDL code after RAM extraction. There is no problem when Verilog code is used.

Solution: Use the Writing RAM Modules to Separate Files option, which is the recommended option to extract RAMs. If you use the Extracting RAMs from Generated RTL option, you must use Verilog instead of VHDL code. For details about the RAM extraction options, see the *SMC User Guide*.

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 before the crash. Correct the error and run the model again.

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 subsystem.

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

M Functions that Evaluate to a Constant not Handled in 32-Bit Linux

On Linux 32-bit operating systems, the M Control interface for M functions that evaluate to a constant does not work, and you get an error message. This limitation extends to 32-bit Linux operating systems and 32-bit MATLAB running under 64-bit Linux operating systems.

Solution: Migrate to 64-bit Linux. From the R2012B release on, MathWorks releases are no longer available on Linux 32-bit platforms.

M Control and MATLAB Startup Script Waits for User Input

If your MATLAB setup uses a custom startup script (startup.m) which waits for user input, this process gets stuck and the model update does not complete.

Solution: If you have such a setup, rename your startup.m script and call it explicitly after MATLAB opens.

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 these blocks require automatically generated 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 your 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.

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. 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 10 for additional ways to speed up initialization.

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"
```

Simulation Mismatch with 32-bit MATLAB on a 64-bit Machine

Simulation results might not match when certain features are used with a a 32-bit version of MATLAB running on a 64-bit machine. The features include the SMC RTL Encapsulation block and C output MATLAB or Simulink wrappers.

Solution: Use a 64-bit version of MATLAB.

Currently Unsupported Features

The following features are not currently supported:

- 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 the Synphony Model Compiler software does not support it either.
- Multiport RAM is not supported for Microsemi targets. If you generate RTL for such blocks, they can fail synthesis.
- Asynchronous read/write RAM blocks are not supported for Microsemi 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.
690 East Middlefield Road
Mountain View, CA 94043 USA
solvnet.synopsys.com