

NOTICE: This document contains references to Agilent Technologies. Agilent's former Test and Measurement business has become Keysight Technologies. For more information, go to **www.keysight.com**.





September 2011
HSPICE Compatibility

© **Agilent Technologies, Inc. 2000-2011**

5301 Stevens Creek Blvd., Santa Clara, CA 95052 USA

No part of this documentation may be reproduced in any form or by any means (including electronic storage and retrieval or translation into a foreign language) without prior agreement and written consent from Agilent Technologies, Inc. as governed by United States and international copyright laws.

Acknowledgments

Mentor Graphics is a trademark of Mentor Graphics Corporation in the U.S. and other countries. Mentor products and processes are registered trademarks of Mentor Graphics Corporation. * Calibre is a trademark of Mentor Graphics Corporation in the US and other countries. "Microsoft®, Windows®, MS Windows®, Windows NT®, Windows 2000® and Windows Internet Explorer® are U.S. registered trademarks of Microsoft Corporation. Pentium® is a U.S. registered trademark of Intel Corporation. PostScript® and Acrobat® are trademarks of Adobe Systems Incorporated. UNIX® is a registered trademark of the Open Group. Oracle and Java and registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. SystemC® is a registered trademark of Open SystemC Initiative, Inc. in the United States and other countries and is used with permission. MATLAB® is a U.S. registered trademark of The Math Works, Inc.. HiSIM2 source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code in its entirety, is owned by Hiroshima University and STARC. FLEXIm is a trademark of Globetrotter Software, Incorporated. Layout Boolean Engine by Klaas Holwerda, v1.7 <http://www.xs4all.nl/~kholwerd/bool.html> . FreeType Project, Copyright (c) 1996-1999 by David Turner, Robert Wilhelm, and Werner Lemberg. QuestAgent search engine (c) 2000-2002, JObjects. Motif is a trademark of the Open Software Foundation. Netscape is a trademark of Netscape Communications Corporation. Netscape Portable Runtime (NSPR), Copyright (c) 1998-2003 The Mozilla Organization. A copy of the Mozilla Public License is at <http://www.mozilla.org/MPL/> . FFTW, The Fastest Fourier Transform in the West, Copyright (c) 1997-1999 Massachusetts Institute of Technology. All rights reserved.

The following third-party libraries are used by the NlogN Momentum solver:

"This program includes Metis 4.0, Copyright © 1998, Regents of the University of Minnesota", <http://www.cs.umn.edu/~metis> , METIS was written by George Karypis (karypis@cs.umn.edu).

Intel® Math Kernel Library, <http://www.intel.com/software/products/mkl>

SuperLU_MT version 2.0 - Copyright © 2003, The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from U.S. Dept. of Energy). All rights reserved. SuperLU Disclaimer: THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

7-zip - 7-Zip Copyright: Copyright (C) 1999-2009 Igor Pavlov. Licenses for files are: 7z.dll: GNU LGPL + unRAR restriction, All other files: GNU LGPL. 7-zip License: This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. unRAR copyright: The decompression engine for RAR archives was developed using source code of unRAR program. All copyrights to original unRAR code are owned by Alexander Roshal. unRAR License: The unRAR sources cannot be used to re-create the RAR compression algorithm, which is proprietary. Distribution of modified unRAR sources in separate form or as a part of other software is permitted, provided that it is clearly stated in the documentation and source comments that the code may not be used to develop a RAR (WinRAR) compatible archiver. 7-zip Availability: <http://www.7-zip.org/>

AMD Version 2.2 - AMD Notice: The AMD code was modified. Used by permission. AMD copyright: AMD Version 2.2, Copyright © 2007 by Timothy A. Davis, Patrick R. Amestoy, and Iain S. Duff. All Rights Reserved. AMD License: Your use or distribution of AMD or any modified version of AMD implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. AMD Availability: <http://www.cise.ufl.edu/research/sparse/amd>

UMFPACK 5.0.2 - UMFPACK Notice: The UMFPACK code was modified. Used by permission. UMFPACK Copyright: UMFPACK Copyright © 1995-2006 by Timothy A. Davis. All Rights Reserved. UMFPACK License: Your use or distribution of UMFPACK or any modified version of UMFPACK implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License

as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. UMFPACK Availability: <http://www.cise.ufl.edu/research/sparse/umfpack> UMFPACK (including versions 2.2.1 and earlier, in FORTRAN) is available at <http://www.cise.ufl.edu/research/sparse> . MA38 is available in the Harwell Subroutine Library. This version of UMFPACK includes a modified form of COLAMD Version 2.0, originally released on Jan. 31, 2000, also available at <http://www.cise.ufl.edu/research/sparse> . COLAMD V2.0 is also incorporated as a built-in function in MATLAB version 6.1, by The MathWorks, Inc. <http://www.mathworks.com> . COLAMD V1.0 appears as a column-preordering in SuperLU (SuperLU is available at <http://www.netlib.org>). UMFPACK v4.0 is a built-in routine in MATLAB 6.5. UMFPACK v4.3 is a built-in routine in MATLAB 7.1.

Qt Version 4.6.3 - Qt Notice: The Qt code was modified. Used by permission. Qt copyright: Qt Version 4.6.3, Copyright (c) 2010 by Nokia Corporation. All Rights Reserved. Qt License: Your use or distribution of Qt or any modified version of Qt implies that you agree to this License. This library is free software; you can redistribute it and/or modify it under the

terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA Permission is hereby granted to use or copy this program under the terms of the GNU LGPL, provided that the Copyright, this License, and the Availability of the original version is retained on all copies. User documentation of any code that uses this code or any modified version of this code must cite the Copyright, this License, the Availability note, and "Used by permission." Permission to modify the code and to distribute modified code is granted, provided the Copyright, this License, and the Availability note are retained, and a notice that the code was modified is included. Qt Availability: <http://www.qtsoftware.com/downloads> Patches Applied to Qt can be found in the installation at: \$HPEESOF_DIR/prod/licenses/thirdparty/qt/patches. You may also contact Brian Buchanan at Agilent Inc. at brian_buchanan@agilent.com for more information.

The HiSIM_HV source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code, is owned by Hiroshima University and/or STARC.

Errata The ADS product may contain references to "HP" or "HPEESOF" such as in file names and directory names. The business entity formerly known as "HP EEsof" is now part of Agilent Technologies and is known as "Agilent EEsof". To avoid broken functionality and to maintain backward compatibility for our customers, we did not change all the names and labels that contain "HP" or "HPEESOF" references.

Warranty The material contained in this document is provided "as is", and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this documentation and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license. Portions of this product include the SystemC software licensed under Open Source terms, which are available for download at <http://systemc.org/>. This software is redistributed by Agilent. The Contributors of the SystemC software provide this software "as is" and offer no warranty of any kind, express or implied, including without limitation warranties or conditions or title and non-infringement, and implied warranties or conditions merchantability and fitness for a particular purpose. Contributors shall not be liable for any damages of any kind including without limitation direct, indirect, special, incidental and consequential damages, such as lost profits. Any provisions that differ from this disclaimer are offered by Agilent only.

Restricted Rights Legend U.S. Government Restricted Rights. Software and technical data rights granted to the federal government include only those rights customarily provided to end user customers. Agilent provides this customary commercial license in Software and technical data pursuant to FAR 12.211 (Technical Data) and 12.212 (Computer Software) and, for the Department of Defense, DFARS 252.227-7015 (Technical Data - Commercial Items) and DFARS 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

HSPICE Compatibility

About HSPICE Compatibility	7
Creating HSPICE Compatible Designs	11
Compatible Features and Limitations	18
Administrative Tasks for HSPICE Compatibility	26
Encrypting HSPICE Compatibility Components	28

About HSPICE Compatibility

HSPICE Compatibility provides direct reading and simulation of HSPICE formatted netlists together with arbitrary ADS components within the ADS environment. Using the HSPICE Compatibility Component Wizard, you can create an ADS component using a specified HSPICE netlist and include this component in your analog/RF designs.

Using HSPICE Compatibility, you can simulate HSPICE Rx/Tx blocks from 3rd parties or internal IC design teams and take full advantage of the powerful transient/convolution, versatile passive models, Momentum, Ptolemy co-simulation, and eye diagram processing offered by ADS.

HSPICE Compatibility supports:

- most elements and models (C, D, E, F, G, H, I, K, L, M, P, Q, R, T, V, W, X)
- common syntax structures (.subckt, .include, .lib, .param)
- selected .options

HSPICE Compatibility does *not* support:

- B, J, S, and U elements
- analysis and measurements
- .protect, .data, .vec statement types

Understanding HSPICE Netlist Design Generation Wizard

To facilitate the usage of HSPICE netlists within ADS, there is a Wizard dialog that steps you through the process of creating a specialized component that represents your HSPICE file. This wizard is access from schematic menus, by choosing Tools>HSPICE Compatibility Component> Wizard... For more information, see the section *Creating HSPICE Compatible Designs* (hspice).

Intended Audience

HSPICE compatibility was developed to provide solutions for SI engineers designing transmission channels to conform to eye diagram and BER specifications.

This design flow typically involves:

1. Obtaining models of drivers and receivers in HSPICE format
2. Package model extraction in the form of S-parameters or equivalent HSPICE representations
3. Board layout (Allegro, Mentor, ADS) and simulation (Momentum, ADS, SiWave)
4. Transient/convolution simulation (ADS)

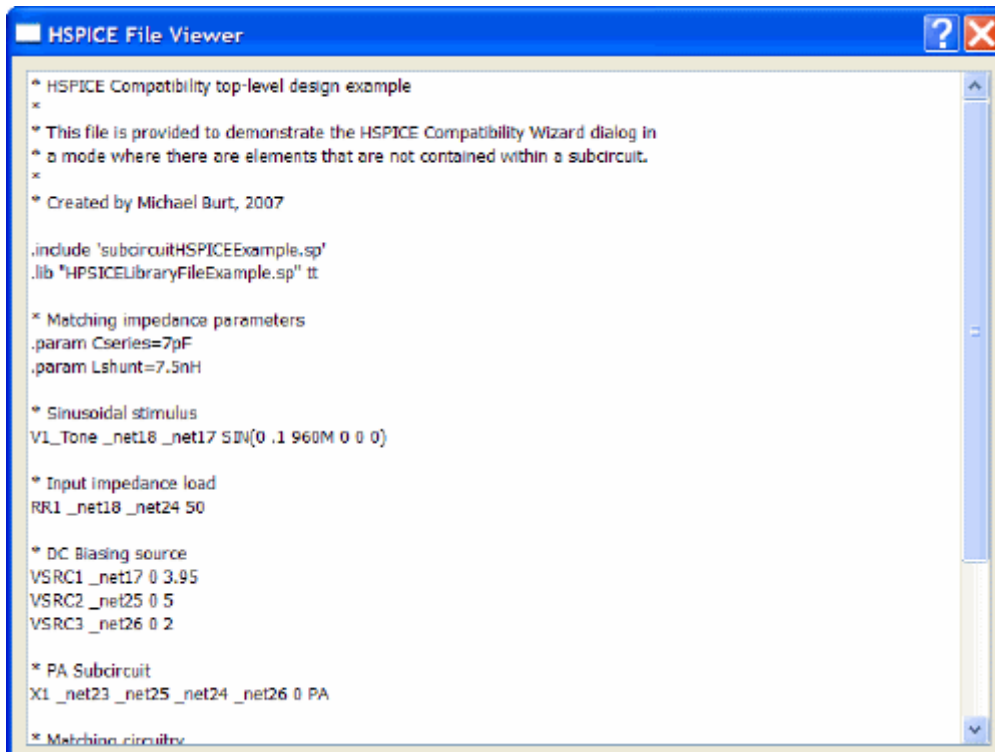
Supported Design Flows

HSPICE Compatibility provides support for two main design flows:

- Flow 1: Self contained HSPICE simulation file simulated in ADS
- Flow 2: HSPICE core design file that requires source stimulation in ADS

Flow 1: Self contained HSPICE simulation file simulated in ADS

In this flow, you will have a file that contains all of the circuit elements, source stimulation, options, and simulation directives required to run a simulation in HSPICE. There may be a core circuit that is accessed via a .include statement, and models that are accessed via .lib statements. This flow is targeted towards being able to simultaneously run simulations in ADS and HSPICE.



With HSPICE compatibility this file is used directly, with no import conversions or file changes necessary. When the HSPICE Compatibility Wizard is used, it will detect the file type and will generate an ADS component that can be placed within an ADS schematic. The wizard gives the option of promoting one or more of the nodes used by top level circuit elements to be input/output terminals that can then be connected to other ADS elements. Or the file can be used without any input/output terminals; it is simply placed as a black box. After the component is placed, you must still place ADS simulation components in the circuit; all HSPICE simulation directives are automatically ignored.

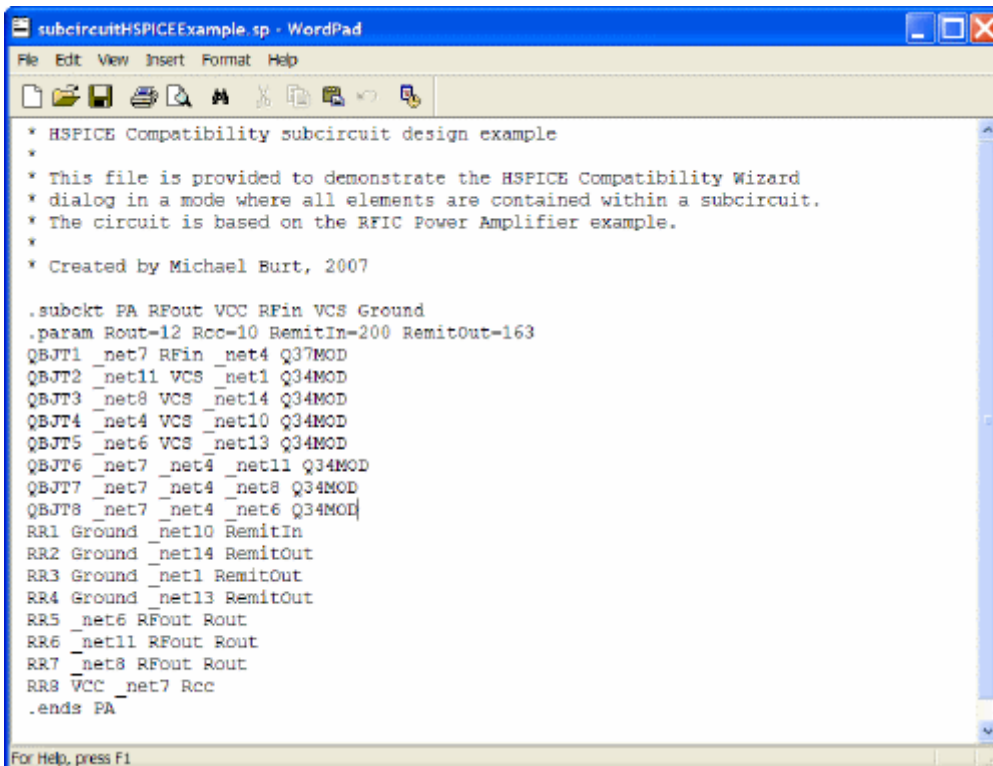
Also available in the wizard is the ability to specify that parameters that are defined in .param statements can be passed in to the circuit. This means you can modify them in ADS, without having to modify the HSPICE file. It also means you can select those parameters and modify them as a part of ADS tuning.

To modify the simulation, the push-into button is overridden so it will open the ADS default text editor. This allows you to modify the HSPICE file, and then resimulate. You do not need to use the wizard to recreate a new component with each modification; you only

recreate the component through the wizard if you want to change the I/O terminals, or the parameters that you want to have access to in the ADS environment. Because the file is not copied, this means you will see the same results with ADS that are seen with HSPICE, provided the simulation components are setup identically.

Flow 2: HSPICE core design file that requires source stimulation in ADS

In this flow, you will have an HSPICE file that contains one or more subcircuits. It will not have top level elements, parameters, or simulation directives. One example of this would be a chip level device downloaded from a vendor. In this flow, you must add new elements for matching, sources for stimulation, and simulation components in the ADS environment. It may also be necessary to add an include component for a library file. The resulting ADS simulation setup is not exportable to HSPICE.



```

* HSPICE Compatibility subcircuit design example
*
* This file is provided to demonstrate the HSPICE Compatibility Wizard
* dialog in a mode where all elements are contained within a subcircuit.
* The circuit is based on the RFIC Power Amplifier example.
*
* Created by Michael Burt, 2007

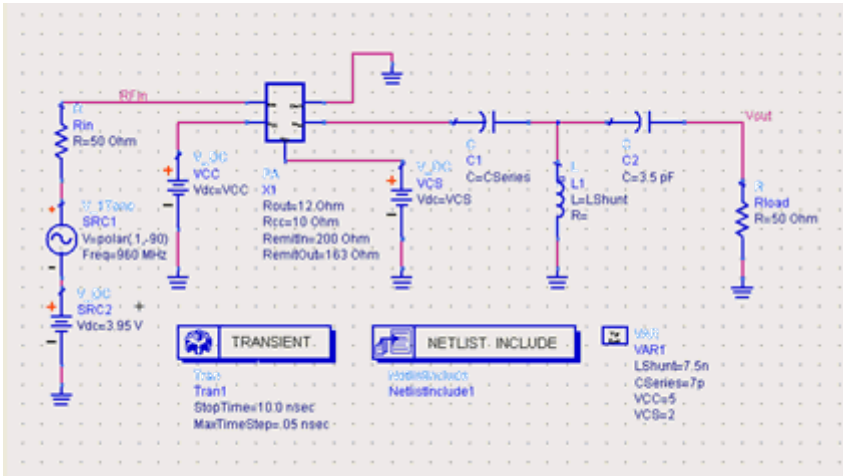
.subckt PA RfOut VCC RFin VCS Ground
.param Rout=12 Rcc=10 RemitIn=200 RemitOut=163
QBJT1 _net7 RFin _net4 Q37MOD
QBJT2 _net11 VCS _net1 Q34MOD
QBJT3 _net8 VCS _net14 Q34MOD
QBJT4 _net4 VCS _net10 Q34MOD
QBJT5 _net6 VCS _net13 Q34MOD
QBJT6 _net7 _net4 _net11 Q34MOD
QBJT7 _net7 _net4 _net8 Q34MOD
QBJT8 _net7 _net4 _net6 Q34MOD
RR1 Ground _net10 RemitIn
RR2 Ground _net14 RemitOut
RR3 Ground _net1 RemitOut
RR4 Ground _net13 RemitOut
RR5 _net6 RfOut Rout
RR6 _net11 RfOut Rout
RR7 _net8 RfOut Rout
RR8 VCC _net7 Rcc
.ends PA
  
```

With HSPICE compatibility, the file is used directly with no import conversions or file changes necessary. To use the file within ADS, you will use the HSPICE Compatibility Wizard to generate a component that represents the HSPICE subcircuit. The wizard will allow you to choose one of the subcircuits within the file that will have an ADS element created for it. If there are multiple subcircuits that you want to place in ADS, you can reuse the same HSPICE file. The file is not imported into ADS, or copied from its original location.

To modify the device after creating it, the push into feature is overridden so it will open the default text editor so that you can edit the HSPICE file. You do not need to use the wizard to recreate a component, unless you will be changing the number of I/O terminals for the subcircuit, or you want to change the ADS parameters for the ADS element. Because the modifications are being done on the original HSPICE file, changes made to the affect the ADS simulation will simultaneously affect HSPICE simulations.

The wizard will detect .param statements, and allows you to promote them to being ADS passed in parameters. All ADS parameters can be accessed by double clicking on the component, just like any other ADS element. The parameters can also be setup to be editable on the screen. By utilizing parameters, you can use the ADS tuning feature with your HSPICE netlist.

Because it is an ADS device, you can attach any ADS element to your design. This allows you to use a core design with ADS transmission line elements, ADS behavioral elements, or momentum extracted components. The device can also be in any level of ADS hierarchy, it is not limited to placement at the ADS top level, which allows for system co-simulation with your device.



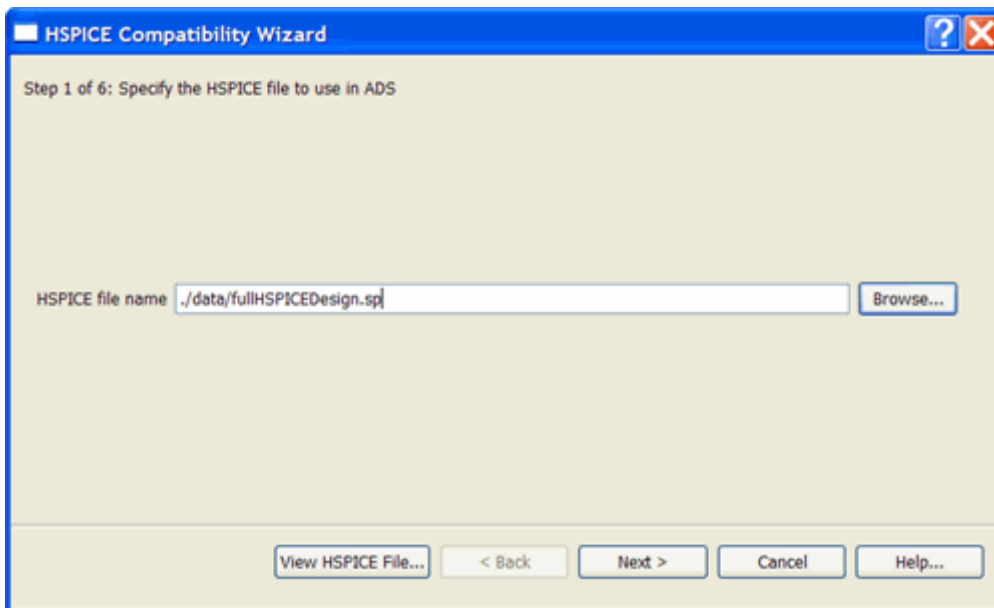
Creating HSPICE Compatible Designs

This page describes how to create an HSPICE compatible design.

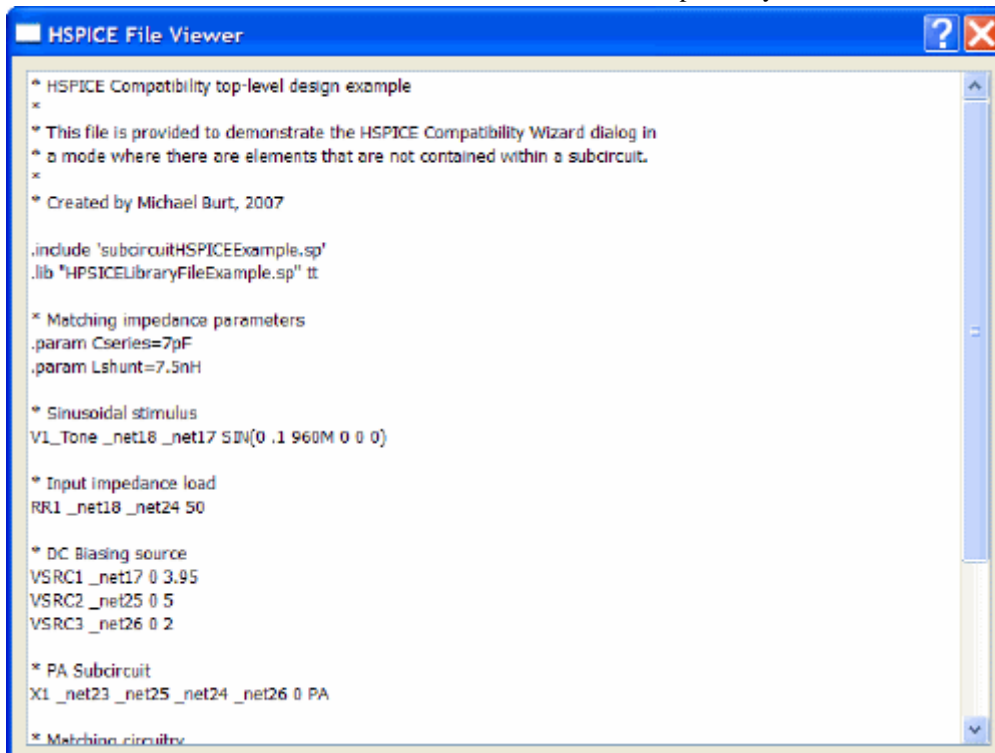
Creating a Component using the HSPICE Compatibility Wizard

1. Open the workspace in which you want to create your HSPICE compatible component.
If a schematic page does not open, open a schematic window.
2. From the schematic page, choose **Tools > HSPICE Compatibility Component > Wizard**.

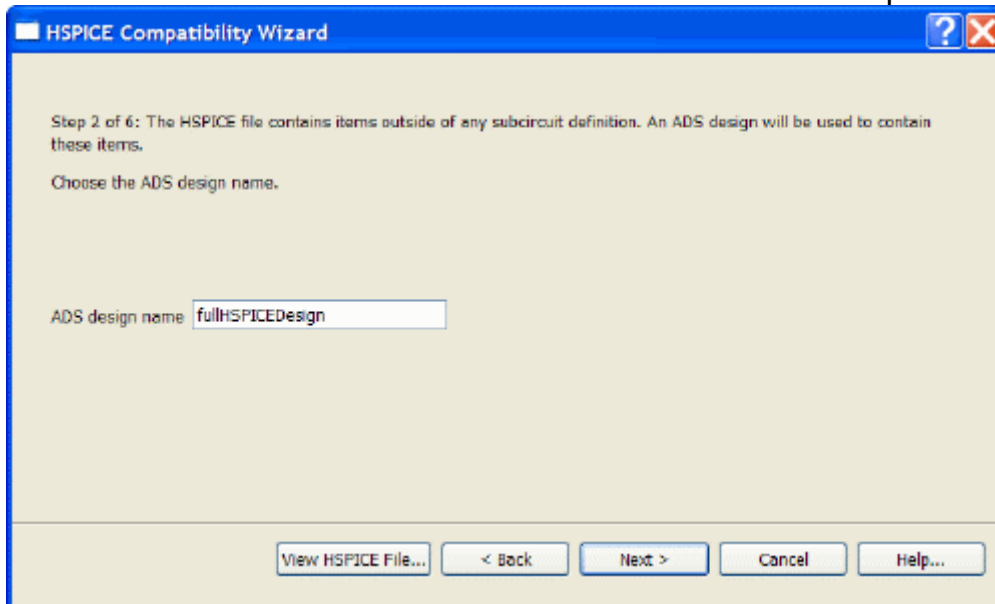
HSPICE Compatibility Component Wizard



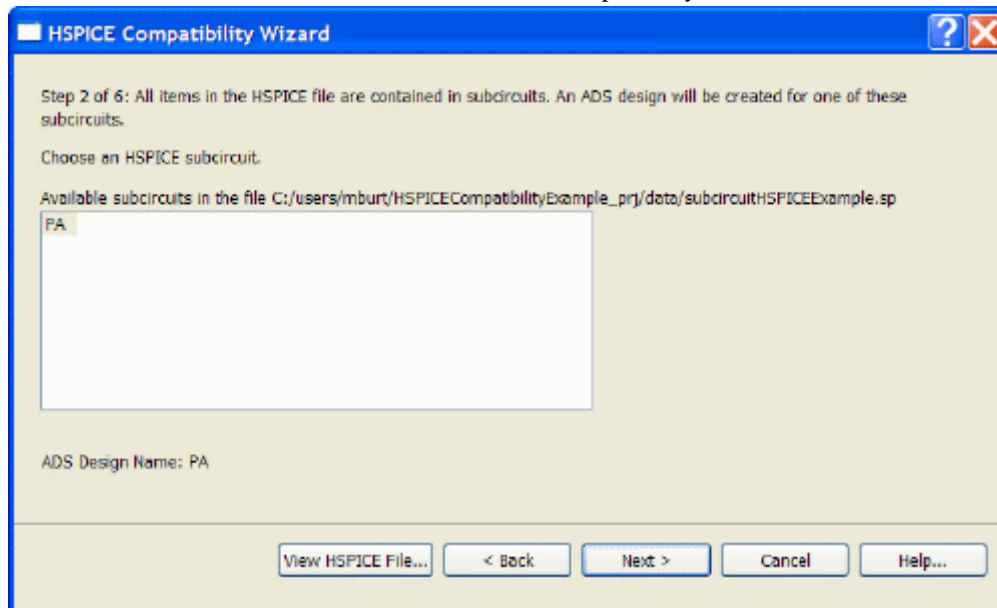
3. Select the HSPICE File that you will create a design for.
You can manually enter a file, or use the browse button to browse for a file. By default, the dialog will browse the data directory of the current workspace. If the specified file does not exist, the **View HSPICE File** and **Next** buttons will remain disabled.
4. If the file exists, click **View HSPICE File...** to display the file in a non-editable text viewer.



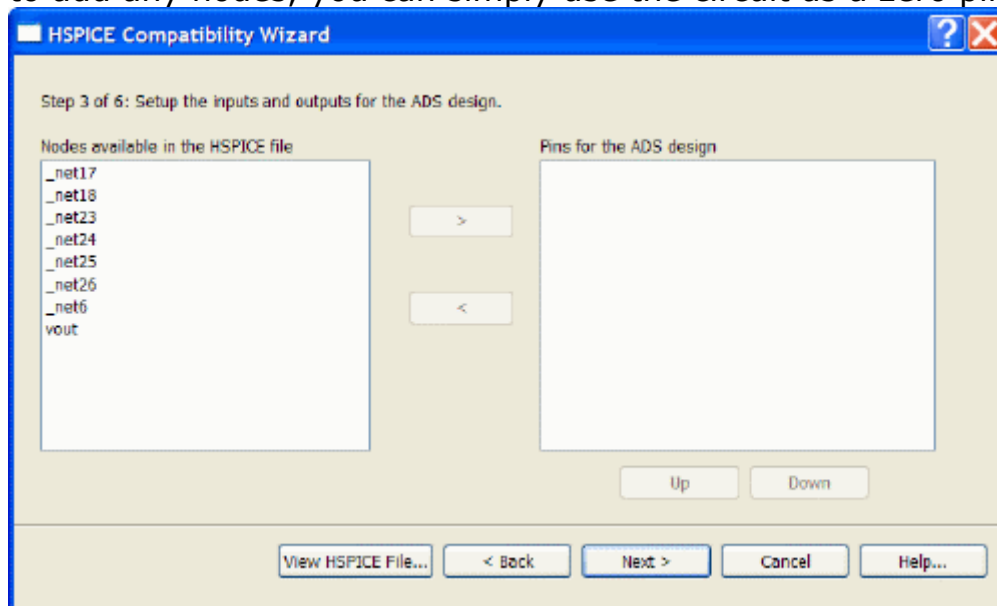
5. Click **Next** to analyze the file. Progress dialogs are shown. When finished, the dialog will either alert you to the fact that it was unable to use the specified file because it is not a valid type for the HSPICE Compatibility, or it will forward you to the appropriate next page of the wizard.
 - If the file analyzer sees elements, parameters, or models that are not contained within a subcircuit, it is considered a top level file. In ADS, the netlist is bracketed with a new subcircuit header, so that the HSPICE netlist can be used with other ADS elements. Choose the name for the top-level design in ADS.



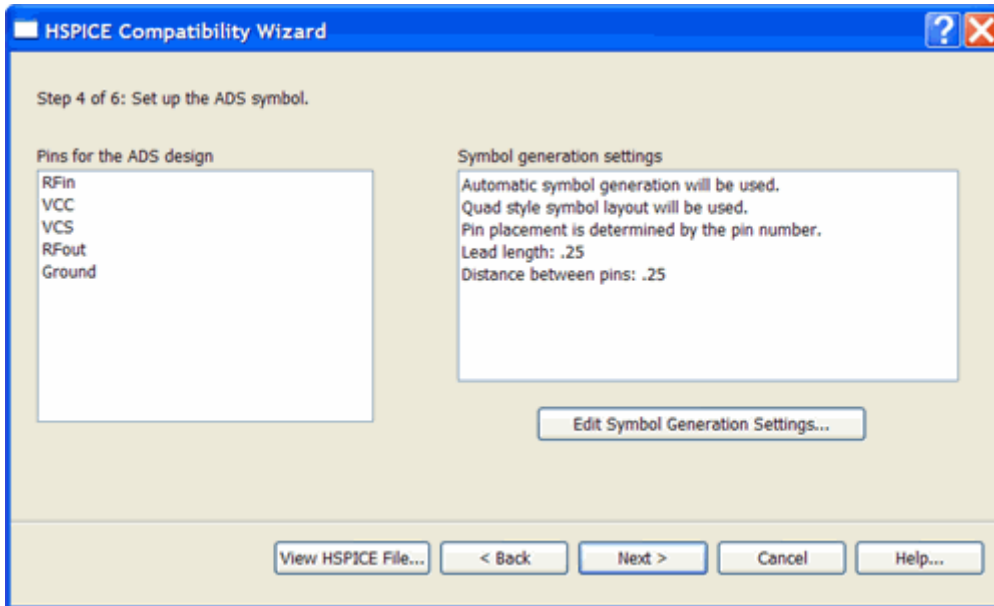
- If the file analyzer finds that all elements, models, and parameters are contained within subcircuits, it is considered a subcircuit file. The subcircuits in the file can be used directly, so an additional subcircuit header is not necessary. A list of the subcircuits available in the file is shown. Choose one of those subcircuits.



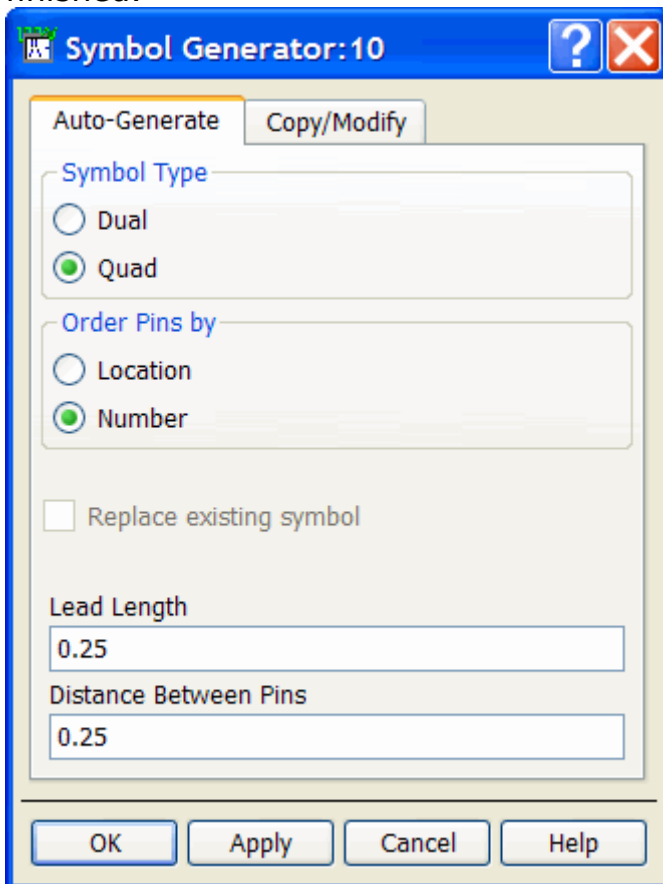
- If the analyzer determines that you have a top level design type, you will be shown a page where you can choose nodes in the top level that will be promoted to being I/O terminals. This allows you to connect inputs and outputs into your design so that the design can interact with other ADS elements (e.g. a system co-simulation). A list of all of the nodes available in the HSPICE top level is presented. You can add or delete them from the I/O terminal list. The up and down buttons allow you to set the order of the I/O terminals. You do not need to add any nodes, you can simply use the circuit as a zero pin device.



6. The next page allows you to choose the symbol generation parameters. Based on the terminal setup page (or on the .subckt terminal definition for the circuit that was chosen), the pins for the ADS design is populated. This is a non-editable field. To change the order or pins you must go back to the appropriate prior page using the back button.

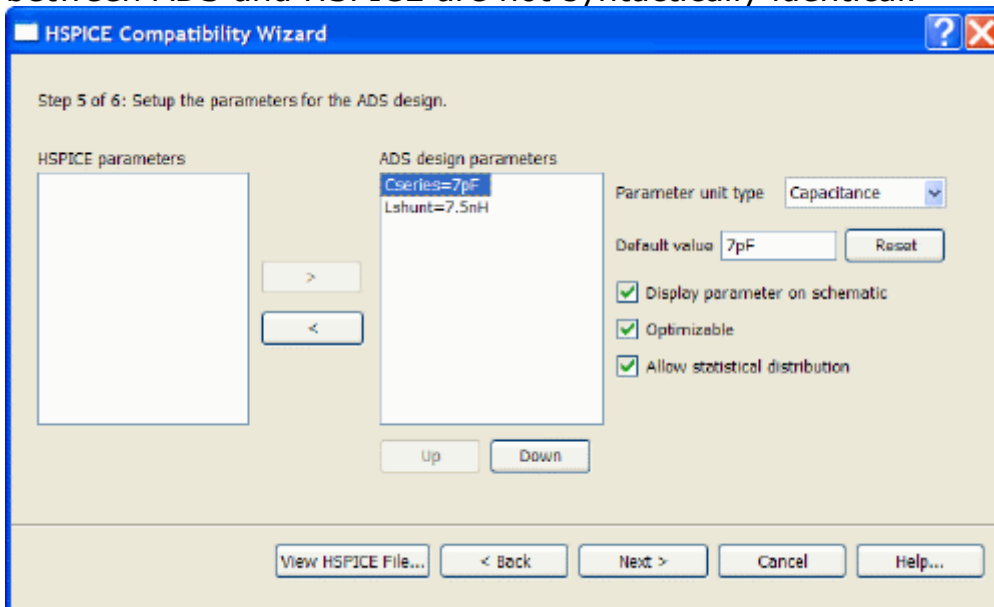


To change the settings, click **Edit Symbol Generation Settings**. This displays the standard ADS auto-generation dialog for symbols. When finished, click **OK** to update the summary. These settings will be used to generate the symbol when the wizard is finished.

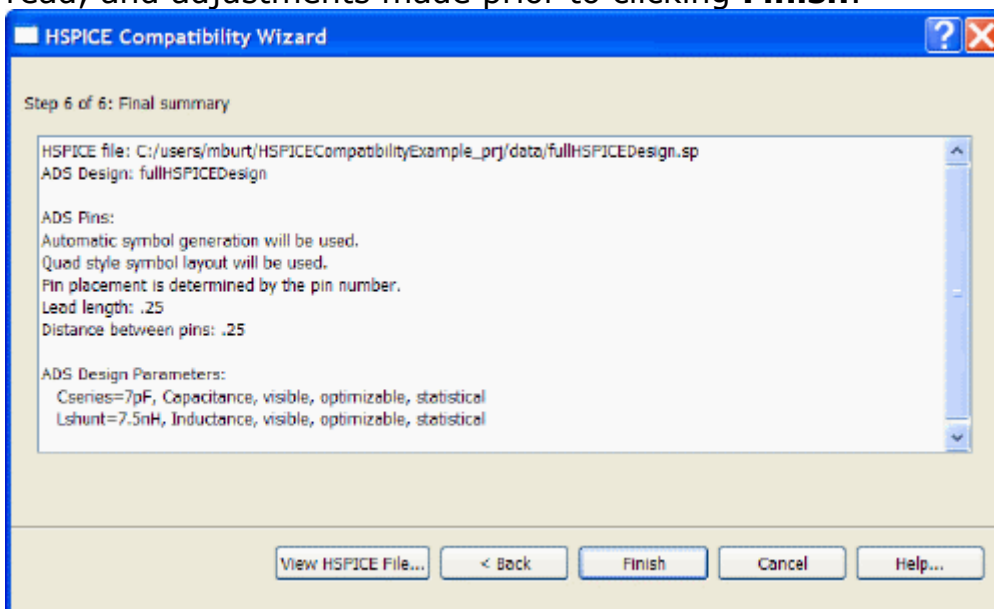


7. The last setup page allows you to specify settings for parameters. Any parameter that is specified as a value will automatically be listed in the ADS design parameters list. A parameter that is listed in the ADS design parameters list will be an instance parameter, and can be changed by using the instance parameter editing dialog. By selecting one or more of the parameters, the settings for the parameter can be changed. Clicking the < button will move it back to the HSPICE parameters list,

which shows all of the parameters in the HSPICE file that will not be instance parameters. These are typically equation-based parameters. Equation based parameters should not be promoted to instance parameters, because the equations between ADS and HSPICE are not syntactically identical.



8. Prior to creating the component, a summary page is shown. The summary should be read, and adjustments made prior to clicking **Finish**.



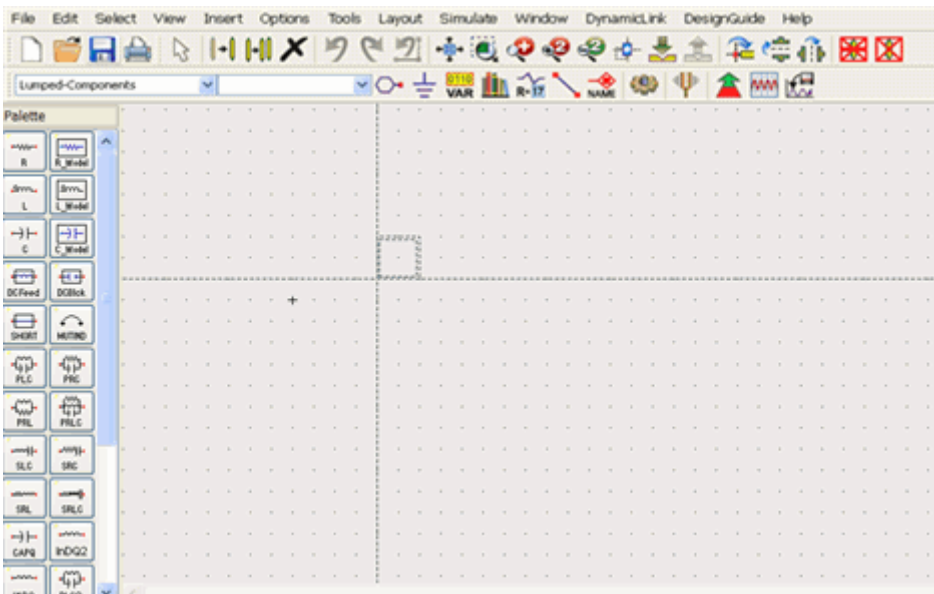
9. To complete the component, click **Finish**.
 Designs are created in the current workspace. You can cancel out if a design in the current workspace will be overwritten. Note that if the dialog is brought up again within the same ADS session, the last settings are reused. Changing the file will delete the settings.
 The generated component includes a schematic view and a symbol view based on the symbol generation settings that were specified in the wizard.

Utilizing HSPICE PDK Components in a PDE Design

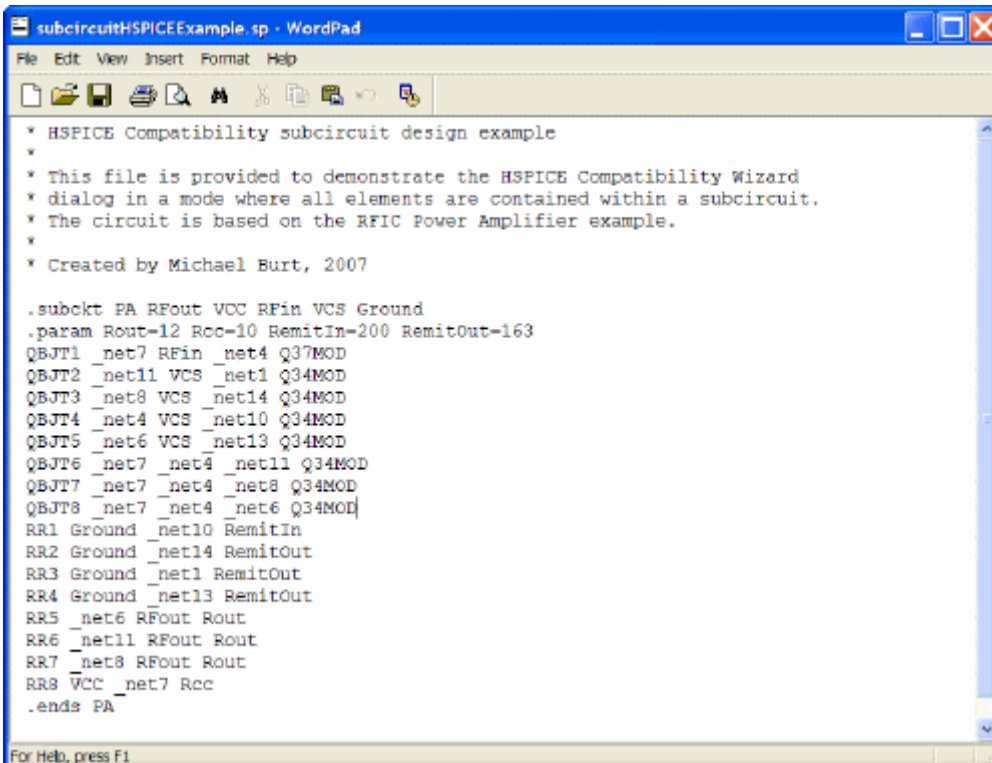
This section discusses options for utilizing HSPICE PDK components in ADS.

HSPICE Compatibility

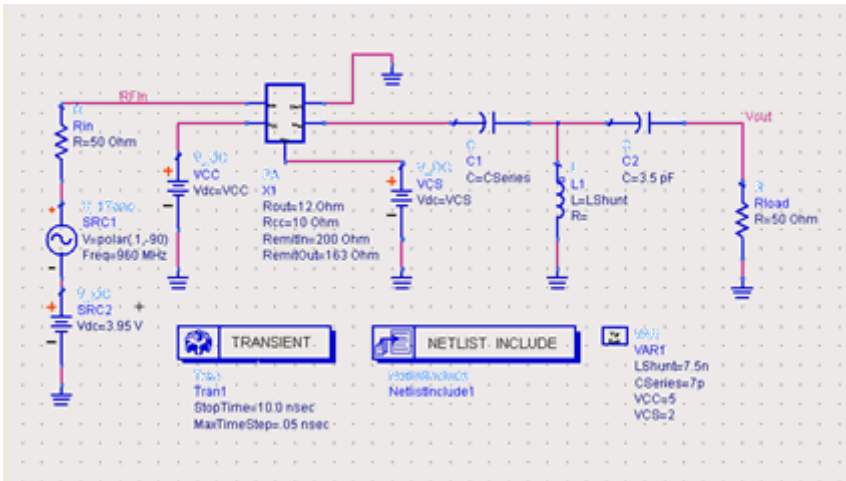
After the component has been generated, focus will return to the initial schematic window, and ADS will automatically enter into instance placement mode, where you can place the component that was just created.



If you choose to push into the HSPICE design, instead of going into the schematic, a text editor window will be opened that allows you to edit the HSPICE file.

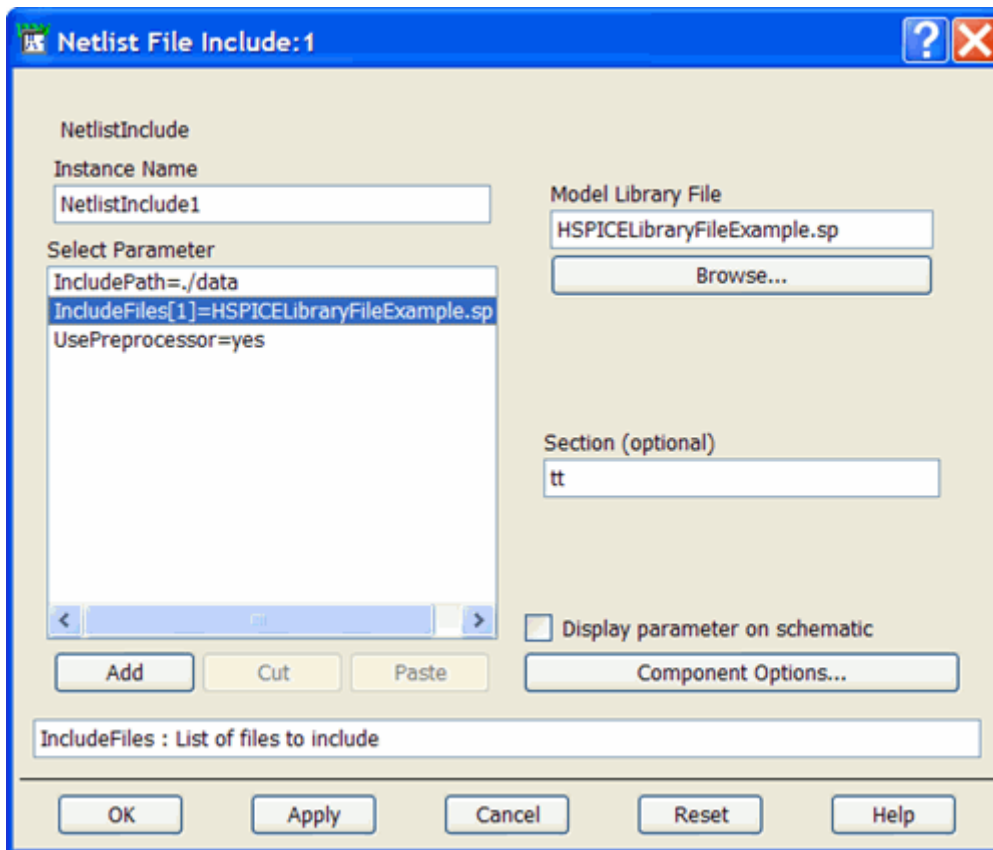


The HSPICE design can be used like any ADS component. If you have pins, you can connect it with ADS hierarchy. If a top level design was designated, it will have pins. If a subcircuit was chosen, it will have appropriate inputs.



All simulation directives in the HSPICE file are ignored. You must place ADS simulation components in your design to perform simulations.

If a top level design was chosen, the .lib and .include components will be resolved. If a subcircuit design was chosen, it may be necessary to place a NetlistInclude component. This can be done by choosing the NetlistInclude component from the Data Items palette. This component allows you to include RF circuits and models in ADS, Spectre, and HSPICE format. The component will determine the appropriate file type. For a library file, you must specify a section. Also, a model include path can be set up.



Compatible Features and Limitations

This page describes the supported elements, models, statements, and simulation options for HSPICE Compatibility.

Supported Elements

This section lists supported elements and their limitations.

C Element

Frequency-dependent and DC block capacitors are not supported.
Unsupported parameters: CONVOLUTION, FBASE, FMAX, INFINITY

D Element

Metal and poly capacitors of levels 1 and 3 are not supported.
Unsupported parameters: WP, LP, WM, LM, OFF, IC

E, F, G, H Element

Unsupported HSPICE Keywords

FOSTER

FREQ

NOISE

Unsupported Parameters

DELTA

gatetype: AND, NAND, OR,
NOR

IC

NPDELAY

SMOOTH

I Element

The independent current source element is supported for DC, AC and Transient simulation compatibility with the exception of transient functions: LFSR and tabular-form of PWL. For use model of the I element, refer to HSPICE Simulation and Analysis User Guide, Sources and Stimuli.

Unsupported HSPICE Keywords

LFSR

PWL (tabular)

J Element

Optional bulk terminal node and gate length/width are not supported.
Unsupported parameters: NB, L, W, OFF, IC

K Element

Saturable core model is not supported.
Ideal transformer is not supported.

Unsupported HSPICE Keywords

IDEAL

MAG

L Element

Frequency-dependent inductor is not supported.

Unsupported Parameters

FMAX

IGNORE_COUPLING

LTYPE

NT

SHORTALL

Unsupported HSPICE Keywords

CONVOLUTION

FBASE

M Element

Unsupported parameters:

Level 49 -- OFF, IC, MULUA, MULUB, SD, STIMOD=1 (UCB's STI model)

Level 53 -- OFF, IC, MULUA, MULUB

Level 54 -- DELTOX, OFF, IC, RDC, RSC, WNFLAG

P Element

The independent port element is supported only in voltage mode for DC, AC and Transient simulation compatibility with the exception of transient functions: PRBS and tabular-form of PWL. For use model of the P element, refer to HSPICE Simulation and Analysis User Guide, *Sources and Stimuli*. HSPICE RF parameters relating to harmonic balance are not supported. The list of unsupported keywords is shown below.

Unsupported HSPICE Keywords
HBAC
HB
POWER (!= 0)
PRBS
PWL (tabular)
RHBAC
RHB
TRANFORHB

Q Element

Unsupported parameters: OFF, IC, AREAB, AREAC

R Element

Frequency-dependent resistor is not supported. The resistor model specified in a behavioral resistor is ignored.

Unsupported parameters: AC, CONVOLUTION, FBASE, FMAX, Rs

T Element

The use of U-model is not supported.

V Element

The independent voltage source element is supported for DC, AC and Transient simulation compatibility with the exception of transient functions: LFSR and tabular-form of PWL. For use model of the V element, refer to HSPICE Simulation and Analysis User Guide, *Sources and Stimuli*.

Unsupported HSPICE Keywords
LFSR
PWL (tabular)

W Element

Both static and tabular models are supported. Data can be either in-netlist or in data files.

Unsupported HSPICE Parameters

UMODEL

FSMODEL

SMODEL

INCLUDESIMAG - supported but no option to turn it off

INCLUDEGDIMAG

DELAYOPT

NOISE

DTEMP

PRINTZO

NODEMAP

X Element

Subcircuit instances are fully supported. The number of node connections must match the number of nodes specified in the subcircuit definition. Default parameter values specified in the definition can be overridden by specifying new values as instance parameters.

Supported Models

This section lists supported models and their limitations.

C Model

All parameters are supported.

D Model

Levels 1, 3, 4, and 6 are supported.

For levels 1 and 3, tunneling current and metal and poly capacitors are not supported.

Unsupported parameters of levels 1 and 3: EXPLIR, JTUN, JTUNSW, NTUN, XTITUN, LM, WM, XM, XOM, LP, WP, XP, XOI

For Level 4, the diffusion geometric parameters are supported in components but not in models.

Unsupported parameters of level 4: AS, LS, LG

For level 6, only the latest version of JUNCAP2 from NXP is supported.

Unsupported parameters of level 6: VERSION

J Model

The basic Level 1 model is supported.

Only support gate capacitance model CAPOP=0, noise model NLEV=2, and temperature model TLEV=0 and TLEVC=0.

Unsupported parameters of level 1: ACM, ALIGN, AREA, HDIF, L, LDEL, LDIF, RG, RSH, RSHG, RSHL, W, WDEL, CAPOP, CALPHA, CAPDS, CGAMDAS, CRAT, GCAP, CVTO, TT, ND, NG, VDEL, NLEV, GDSNOI, BETATCE, BEX, CTD, CTS, EG, GAP1, GAP2, TLEV, TLEVC, TPB, TRD, TRG, TRS

L Model

L LEVEL=3(Resistor Model) is supported.

M Models: NMOS, PMOS

Levels 49, 53, and 54 are supported.

For level 49, UCB's STI model, WPE model, and Ig model from BSIM4 are not supported.
Unsupported parameters of level 49: ENCMODE, HSPVER, APWARN, BINFLAG, SFVTFLAG, VGSLIM, LITE, WPEMOD, IGCMOD, IGDMOD

Unsupported parameters of level 53: ENCMODE, HSPVER, APWARN, BINFLAG, SFVTFLAG

For level 54, JUNCAP models and Hspice junction diode models (ACM) are not supported.

Unsupported parameters of level 54: JUNCAP, ACM, TRD, TRS

Q Models: NPN, PNP

Level 1 is supported.

Substate node can only be specified in components and not through model parameter BULK(NSUB).

Unsupported parameters: BULK(NSUB), CBCP, CBEP, CCSP

R Model

Reference node for Wire RC is always the ground.

Unsupported parameters: BULK, RAC, VC1R, VC2R, NOISE

SP Model

The SP model has a limited support. It is supported only in the context of the W element tabular model. Formats that are irrelevant to the W element are not supported. Sweeping parameters of the SP model is not supported.

Unsupported HSPICE Parameters
DC - dc data included in the table is handled
INFINITY
INTERPOLATION - cubic splines are used internally
EXTRAPOLATION

W Model

Sweeping parameters of the W model is not supported.

Unsupported HSPICE Parameters
wp
Lognd
Rognd
Rsgnd

The parameter value MODELTYPE=FieldSolver is not supported.

Supported Statements

This section lists supported statements and their limitations.

.end

Syntax: `.end`

After `.end`, any remaining lines in the file are discarded.

.global

Syntax: `.global node1 node2 ...`

Use `.global` at the beginning of a netlist to declare global nodes.

.hdl

Syntax: `.hdl "model.va"`

Use `.hdl` to include a Verilog-A model.

.if, .elseif, .else, .endif

Syntax:

```
.if x < 0
...
.elseif x > 1
...
.else
...
.endif
```

Use `.if` blocks to conditionally include or exclude portions of the netlist.

.include

Syntax: `.include "path.sp"`

Use `.include` to incorporate an external netlist file.

.lib, .endl

Syntax:

```
.lib "library.sp" "section"
.lib "section"
```



```
...
.endl
```

Use `.libendl` to define sections inside a library file. Use `.lib` statements in your main netlist to incorporate a particular section from a library.

.macro, .eom

Syntax:

```
.macro name node1 node2 ... param1=val param2=val ...
...
.eom name
```

Use `.macroeom` to define a macro.

.model

Syntax: `.model name type param1=val param2=val ...`

See [Supported Models](#).

.option

Syntax: `.option opt1 opt2=val opt3 ...`

See [Supported Simulation Options](#).

.param

Syntax: `.param param1=val param2=val ...`

Use `.param` to define global variables, and to define default parameters for a subcircuit.

.subckt, .ends

Syntax:

```
.subckt name node1 node2 ... param1=val param2=val ...
...
.ends name
```

Use `.subcktends` to define a subcircuit.

.pat

Syntax: `.pat name=b_string_or_nested_pattern_structure r=val rb=val`

Use `.pat` to define patterns referred to from I or V Element PAT sources.

Supported Simulation Options

This section lists supported simulation options and their limitations.

scale

Syntax: `.option scale=1u`

Use `.option scale` to set a scale factor that is applied to all dimension parameters on elements.

search

Syntax: `.option search="path"`

Use `.option search` to specify additional paths to use when looking for included netlists or Verilog-A files.

tnom

Syntax: `.option tnom=25`

Use `.option tnom` to specify the nominal temperature for simulation.

General Limitations

Parameter scoping

Simulations in hpeesofsim are handled as if `.option parhier=local` was specified. The default HSPICE scoping (`.option parhier=global`) is not available.

Connections to global nodes and parameters

Although HSPICE netlists are case-insensitive, connections to a global node or references to a global variable from *outside* the HSPICE-language portion of the netlist must use the lowercase variant of the name. Connections or references within the HSPICE-language portion can use any case for the letters.

Administrative Tasks for HSPICE Compatibility

This page covers the topics of Configuring PDKs for HSPICE Simulation in ADS, and how to Encrypt HSPICE files for use in ADS.

Configuring PDKs for HSPICE Simulation in ADS

Currently there is no automated way of configuring an HSPICE PDK for usage within ADS. In order to use individual components contained within an HSPICE library file, you must manually create your symbols, and have them reference the appropriate subcircuit or model within your HSPICE library file. These ADS elements should then be placed into an ADS design kit, which can be redistributed with the HSPICE library file. For more information on how to create an ADS design kit, see *Design Kit Development* (dkarch). This remainder of this section will cover specialized items related to HSPICE compatibility.

Using the Netlist File Include for a PDK model file

The NetlistInclude component has been updated so it recognizes HSPICE library files. When the NetlistInclude is used with a PDK library file, you need to specify the location of the file, as well as the library section. During netlisting, a block in the following form is output

```
simulator lang=spice
.lib '<file>' section
simulator lang=ads
```

You can use the same file or multiple files in one NetlistInclude component, with as many sections as you like.

Creating a Process Include for a PDK model file

In the case of the HSPICE component, the only difference is in how the netlist callback function must output the file names and sections. For HSPICE, pre-processor directives for definitions and a #include should not be used. Instead, the appropriate .lib statement should be output.

```
defun mykit_process_netlist_cb(cbP, cbData, instH)
{
decl libSection;
decl netStrg;
decl parmH, parmName, parmFormName;
// Output a guard #ifdef so that the .lib statement
// is only processed once
netStrg = "#ifndef MYKIT_PROCESS\n#define MYKIT_PROCESS\n";
netStrg = "simulator lang=spice\n"
//--- corner case/resistance -----*/
parmH = db_first_parm(instH);
// This while loop will process all of the parameters
// for the include component. This example only has
```

HSPICE Compatibility

```
// one parameter, so it is for illustrative purposes here.
while (parmH != NULL)
{
    parmName = db_get_parm_attribute(parmH, PARM_NAME);
    if (parmName == "CornerCase")
    {
        netStrg = strcat(netStrg, "; corners\n");
        parmFormName = db_get_parm_attribute(parmH, PARM_FORM_NAME);
        if (parmFormName == "mykit_form_process_best")
            libSection="MYKIT_BEST_SECTION"
        else if (parmFormName == "mykit_form_process_worst")
            libSection="MYKIT_WORST_SECTION";
        else
            libSection="MYKIT_NOMINAL_SECTION";
        netStrg=strcat(netStrg, sprintf(".lib '%s/circuit/models/mykit_models.hsp' %s\n",
                                      MYKIT_PATH, libSection))
    }
    parmH = db_next_parm(parmH);
}
// Reset to ADS netlist parsing mode
netStrg = strcat(netStrg, "simulator lang=ads\n");
// Close the guarding #ifdef
netStrg = strcat(netStrg, "#endif\n");
// Return the output to the calling function
return(netStrg);
}
```

The key differences in this output function are the bracketing simulator `lang=spice/simulator lang=ads`, and the use of the `.lib` statement. If your file is not a library file, you can still include the file using the `.include` syntax, and omit a section. It is still recommended to use the guard `#ifdef/#endif` with the file to avoid conflicts with parameters if a file is included multiple times.

Configuring the item definitions for your ADS elements

HSPICE compatibility does not differentiate between ADS and HSPICE subcircuits or models after flattening. If you have an instance of an element netlisted in ADS format, it can reference an HSPICE subcircuit or model that is contained within an HSPICE library file. The convention that should be used is to make the ADS element netlist as the name of the HSPICE subcircuit or model you wish to use. This allows you to use the standard ADS percent strings for netlisting, as opposed to needing to create a custom netlisting function.

For example, if your library file has the following:

```
.LIB mySampleLib
.subckt sampleCircuit1 1 2 ...
.
.
.
.ends sampleCircuit1
```

You would ideally create an ADS component named `sampleCircuit1`. In the item definition field for the netlist string, you would use `ComponentNetlistFmt`. This would give you an instance netlist line that looks like this:

```
sampleCircuit1:I1 _net1 _net2 ...
```

Note that all parameters that are defined for the library elements will utilize ADS equation syntax on the instances placed in ADS.

Encrypting HSPICE Compatibility Components

This page describes how to encrypt an **HSPICE Compatibility Component (HCC)**. To encrypt an HSPICE netlist for ADS simulator, simply use the *HSPICE Compatibility Component Wizard* to create an HCC component, verify ADS simulation results of the HCC to be correct, and then use the menu item described below to create an Encrypted HCC component that can replace the original HCC.

Encrypt an HSPICE Compatibility Component

An *HSPICE Compatibility Component (HCC)* can be encrypted by selecting the **Tools > HSPICE Compatibility Component > Encrypt** menu item. If an HCC instance is already selected before the **Encrypt** menu item is clicked, that HCC component will be encrypted. If no instance was selected, the ADS schematic window prompt (at lower left corner) and mouse cursor will change until an HCC instance is selected to be encrypted or *End Command* is selected to cancel the encryption process.

The callback function of the Encrypt menu item will create a new **Encrypted HSPICE Compatibility Component** based on the selected *HCC* instance. ADS simulation results of a circuit using an HCC instance should remain the same after replacing the *HCC* instance with an instance of its corresponding Encrypted HCC component.



Note

Before trying to encrypt an HCC component, please make sure that its ADS simulation results are correct and the HSPICE netlist it includes is not encrypted.

The new Encrypted HCC component will have an identical symbol view as the original HCC component. It will have a schematic view that is the same as the original HCC component except that it includes a new netlist file with only two lines. The first line changes the simulator language to *ads* syntax; and the second line is an ADS **#uselib** statement referencing a new encrypted ADS library that was generated (under the current workspace) based on the HSPICE netlist file included by the original HCC component. Running ADS simulations using the encrypted ADS library should be equivalent to including the original plain HSPICE netlist.

The new encrypted ADS library is defined in the *data/ADSlibconfig* file under the current ADS Workspace. The second field of each line in an *ADSlibconfig* file must be the full path to the encrypted ADS library. Therefore, if the library is moved to a different directory or if it is sent to someone without access to the full path of your encrypted library, the path in the user's own *data/ADSlibconfig* file must be modified accordingly for ADS simulator to locate the library.

All files created for the new Encrypted HCC component will have a common file name suffix **_xhcc**. The ADS cell name of the new *Encrypted HCC* component will be that of the original *HCC* component plus a **_xhcc**; and the name of the new two-line netlist file will be the original HSPICE file name followed by **_xhcc** and then the same extension name.

Due to a wrapper subcircuit being added to the HSPICE netlist file before encryption, ADS simulator will issue a subcircuit redefinition warning, if the original HCC component requires a subcircuit header (parameter designRequiresSubcktHeader=yes). Please just ignore the warning, which reads like:

Warning detected by hpeesofsim during netlist parsing.

In file `flip_xhcc/flip' at, or just before, line 3.

Subcircuit `flip_xhcc' is redefined. Discarding the earlier definition

from `<string/GEMX netlist>:4'

Distribute an Encrypted HSPICE Compatibility Component

Once simulation results of the *Encrypted HCC* component are verified to be the same as those of its original *HCC*, the original *HCC* (and HSPICE netlist, if it is under the same workspace) can be deleted and the archive of the current ADS workspace can then be sent to your users without revealing the content of the original HSPICE netlist. The users just need to change the full path of the encrypted ADS library (which is created under the current workspace) in their *data/ADSlibconfig* file.

Debug Mode

If it failed to create an encrypted ADS library or if its simulation results proved to be different from those of the original *HCC* component, a debug mode can be turned on by entering the following AEL command in the ADS AEL Command Line (accessible by selecting **Tools > Command Line** menu item in the ADS Main window):

```
xhcc_debug();
```

Under debug mode, select the **Tools > HSPICE Compatibility Component > Encrypt** menu item to encrypt the original HCC again and some information dialog boxes will open in turn. Please select the OK button in each information dialog box to continue the debug process. Eventually, the *xhcc.log* debug log will open when the operation is done or if it failed and the timeout limit is exceeded. The timeout limit has a default value of 1000 seconds and can be changed by defining the **XHCC_TIMEOUT** (UNIX or Windows) environment variable with a desired number of seconds before starting ADS.

The most important message in *xhcc.log* debug log is the line beginning with *Command:.* The rest of that line is the system command sent from ADS with an AEL *system()* call. It invokes the **encrypt_spice.pyc** compiled Python script to parse and encrypt the HSPICE file into an encrypted ADS library. The script also generates other associated files as indicated in the log. This command can be issued directly from the (Windows CMD shell or UNIX shell) system prompt under the current workspace directory to see any error message returned by Python that is not visible inside ADS environment. The message may help in identifying the source of the issue or coming up with a workaround.

Note that a flattened copy of the original HSPICE netlist is created under the current workspace directory and used as the input to hpeesofencode program to generate the

encrypted ADS library. In debug mode, the input file is not removed after encryption.

The compiled Python script removes comments and blank lines in the flattened HSPICE netlist, then joins HSPICE continuation lines (beginning with a plus sign or ending with double backslashes) before encrypting it. This additional process can be disabled by entering the following command in the AEL Command Line:

```
xhcc_debug(2);
```

In both debug modes, all HSPICE .LIB and .INC statements are expanded.

HSPICE implicit include is not supported, please add a .INC statement to include any desired netlist file. HDL modules cannot be encrypted by the ADS encoder but can be replaced by their corresponding **cml** files. For questions about compiling Verilog-A modules, please contact EEsof Technical Support or Tiburon Design Automation.

On UNIX, LD_LIBRARY_PATH must be set properly to run hpeesofencode, which is invoked by the compiled Python script. One way of achieving that is to open an xterm window from ADS AEL Command Line by entering the following command:

```
system("xterm")
```

The xterm opened should inherit the required environment setups for the hpeesofencode program from ADS.

To end debug mode, enter the following AEL command in the AEL Command Line:

```
xhcc_quit_debug();
```