



**Agilent Technologies**

ADS 2008  
January 2008  
HSPICE Compatibility

## Advanced Design System 2008

© Agilent Technologies, Inc. 2000-2008

395 Page Mill Road, Palo Alto, CA 94304 U.S.A.

No part of this manual 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. Microsoft®, Windows®, MS Windows®, Windows NT®, and MS-DOS® 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. Java™ is a U.S. trademark of Sun Microsystems, Inc. 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.

**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 manual 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** If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

## Contents

- About HSPICE Compatibility
  - ◦ Understanding HSPICE Netlist Design Generation Wizard
    - Intended Audience
    - Supported Design Flows
      - - Flow 1: Self contained HSPICE simulation file simulated in ADS
      - - Flow 2: HSPICE core design file that requires source stimulation in ADS
- Creating HSPICE Compatible Designs
  - ◦ Creating a Component using the HSPICE Compatibility Wizard
    - - HSPICE Compatibility Component Wizard
    - Utilizing HSPICE PDK Components in a PDE Design
- Compatible Features and Limitations
  - ◦ Supported Elements
    - - C Element
    - - D Element
    - - E, F, G, H Element
    - - I Element
    - - J Element
    - - K Element
    - - L Element
    - - M Element
    - - P Element
    - - Q Element
    - - R Element
    - - T Element
    - - V Element
    - - X Element
  - Supported Models
    - - C Model
    - - D Model
    - - J Model
    - - L Model
    - - M Models: NMOS, PMOS
    - - Q Models: NPN, PNP
    - - R Model
  - Supported Statements
    - - .end
    - - .global
    - - .hdl
    - - .if, .elseif, .else, .endif
    - - .include
    - - .lib, .endl
    - - .macro, .eom
    - - .model
    - - .option
    - - .param
    - - .subckt, .ends

- Supported Simulation Options
- - scale
  - search
  - tnom
- General Limitations
- - Parameter scoping
  - Connections to global nodes and parameters
- Administrative Tasks for HSPICE Compatibility
- - Configuring PDKs for HSPICE Simulation in ADS
  - - Using the Netlist File Include for a PDK model file
    - Creating a Process Include for a PDK model file
    - Configuring the item definitions for your ADS elements

## 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 Netlist Design Generation 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, X)
- common syntax structures (.subckt, .include, .lib, .param)
- selected .options

HSPICE Compatibility does not support:

- B, J, S, U, and W 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](#).

### 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.

```

HSPICE File Viewer
* HSPICE Compatibility top-level design example
*
* This file is provided to demonstrate the HSPICE Compatibility Wizard dialog in
* a mode where there are elements that are not contained within a subcircuit.
*
* Created by Michael Burt, 2007

.include 'subcircuitHSPICEExample.sp'
.lib "HSPICELibraryFileExample.sp" tt

* Matching impedance parameters
.param Cseries=7pF
.param Lshunt=7.5nH

* Sinusoidal stimulus
V1_Tone _net18 _net17 SIN(0 .1 960M 0 0 0)

* Input impedance load
RR1 _net18 _net24 50

* DC Biasing source
VSRC1 _net17 0 3.95
VSRC2 _net25 0 5
VSRC3 _net26 0 2

* PA Subcircuit
X1 _net23 _net25 _net24 _net26 0 PA

* Matching circuitry
    
```

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.

```

subcircuitHSPICEExample.sp - WordPad
File Edit View Insert Format Help
* 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
For Help, press F1
    
```

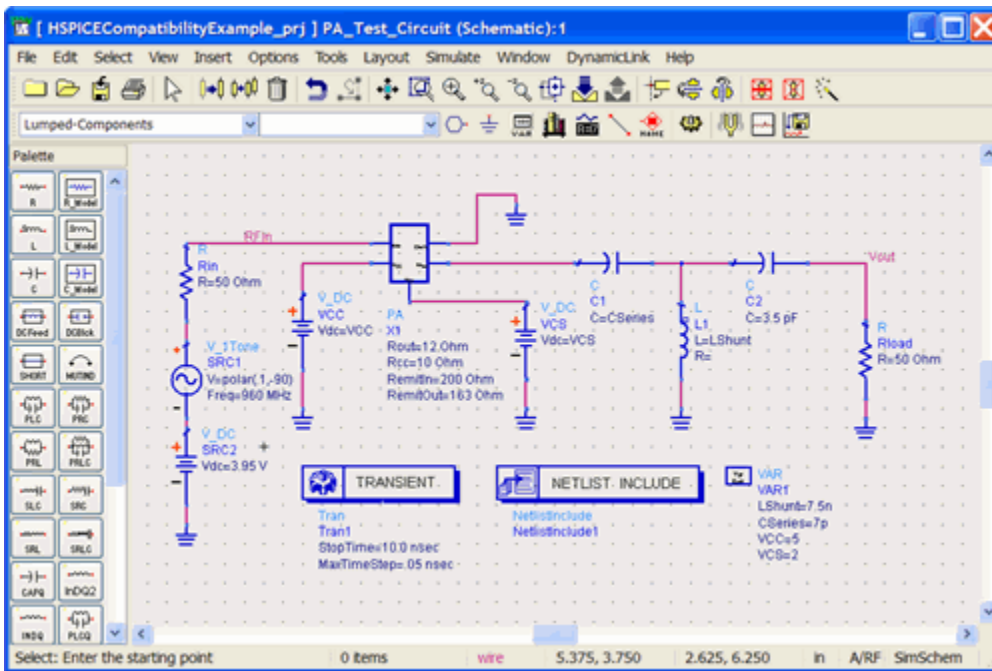
With HSPICE compatibility, the file is used directly with no import conversions or file changes necessary. To use the file within ADS, you will use the HSPICE Compatibility Wizard to generate a component that represents the HSPICE subcircuit. The wizard will allow you to choose one of the subcircuits 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.



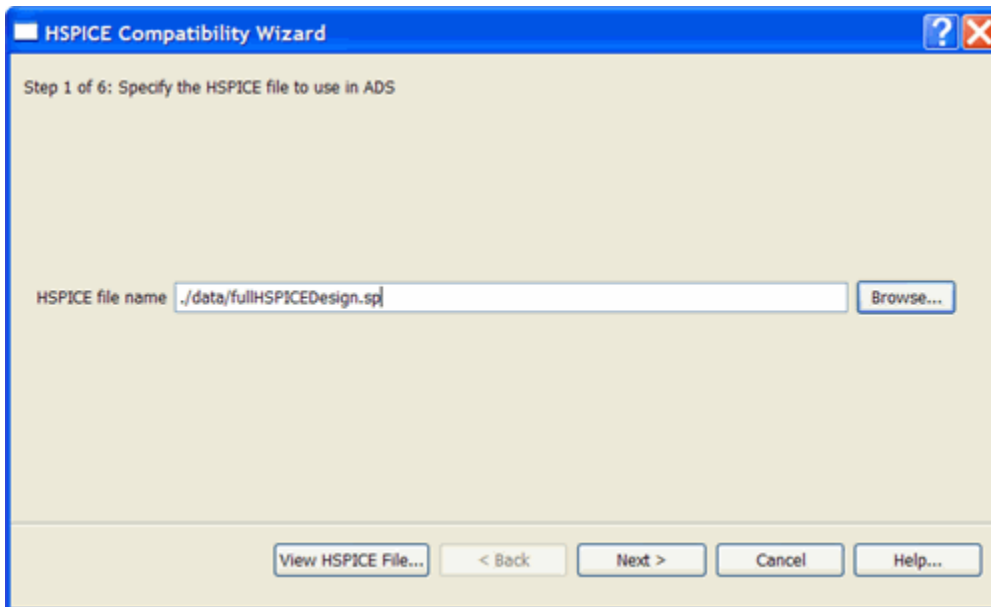
## Creating HSPICE Compatible Designs

This page describes how to create an HSPICE compatible design.

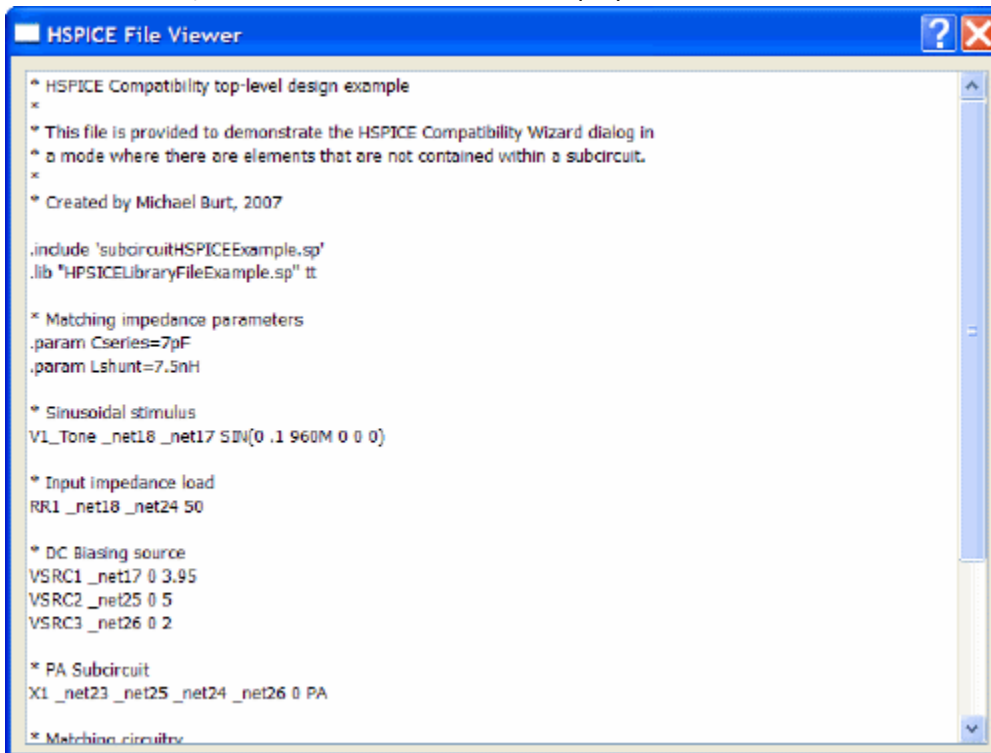
### Creating a Component using the HSPICE Compatibility Wizard

1. Open the project 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



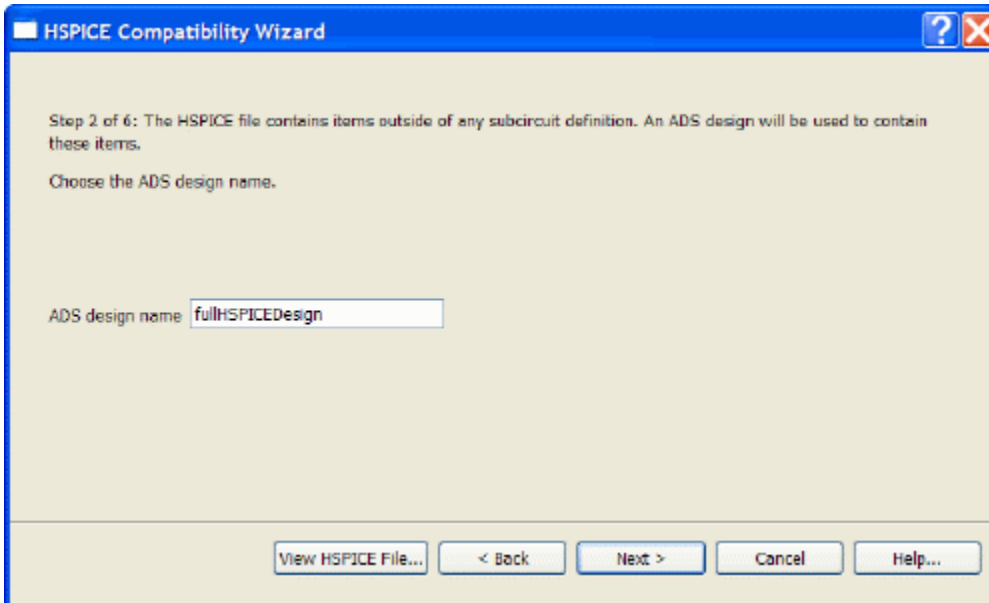
3. Select the HSPICE File that you will create a design for.  
You can manually enter a file, or use the browse button to browse for a file. By default, the dialog will browse the data directory of the current project. If the specified file does not exist, the View HSPICE File and Next buttons will remain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



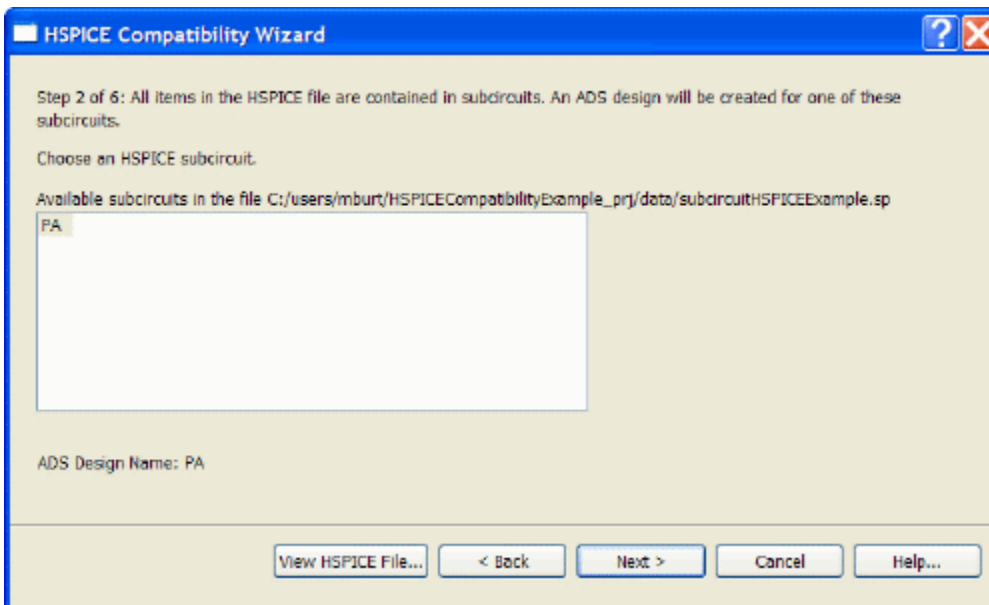
5. Click Next to analyze the file.  
Progress dialogs are shown. When finished, the dialog will either alert you to the fact that it was unable to use the specified file because it is not a valid type for the HSPICE Compatibility, or it will forward you to the

appropriate next page of the wizard.

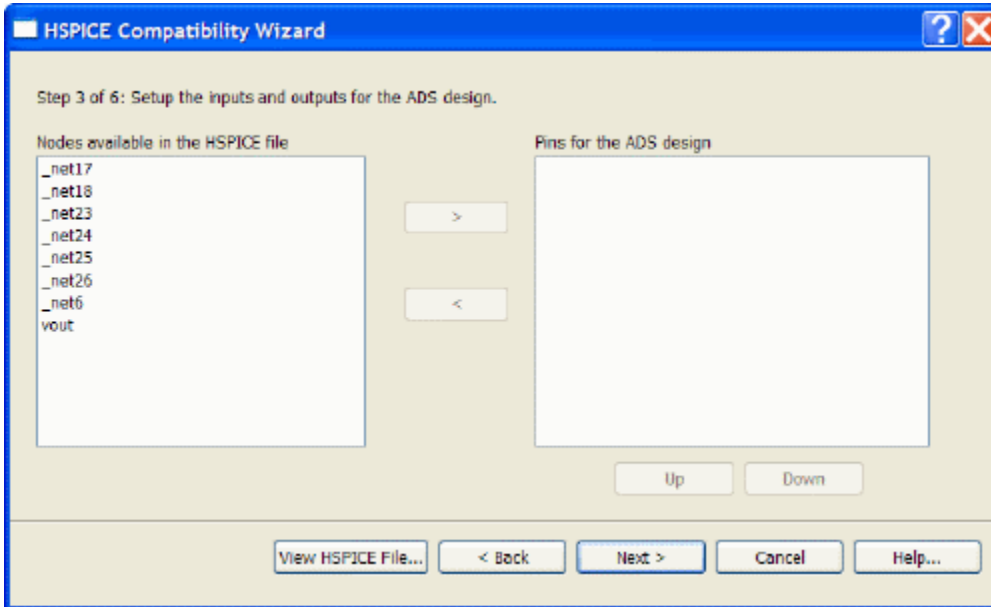
- If the file analyzer sees elements, parameters, or models that are not contained within a subcircuit, it is considered a top level file. In ADS, the netlist is bracketed with a new subcircuit header, so that the HSPICE netlist can be used with other ADS elements. Choose the name for the top-level design in ADS.



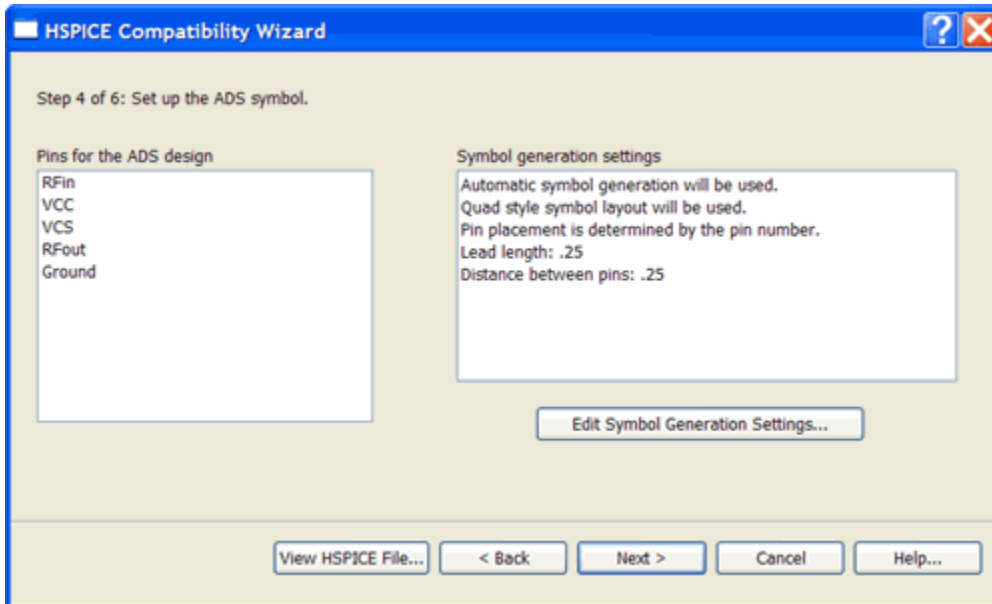
- If the file analyzer finds that all elements, models, and parameters are contained within subcircuits, it is considered a subcircuit file. The subcircuits in the file can be used directly, so an additional subcircuit header is not necessary. A list of the subcircuits available in the file is shown.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



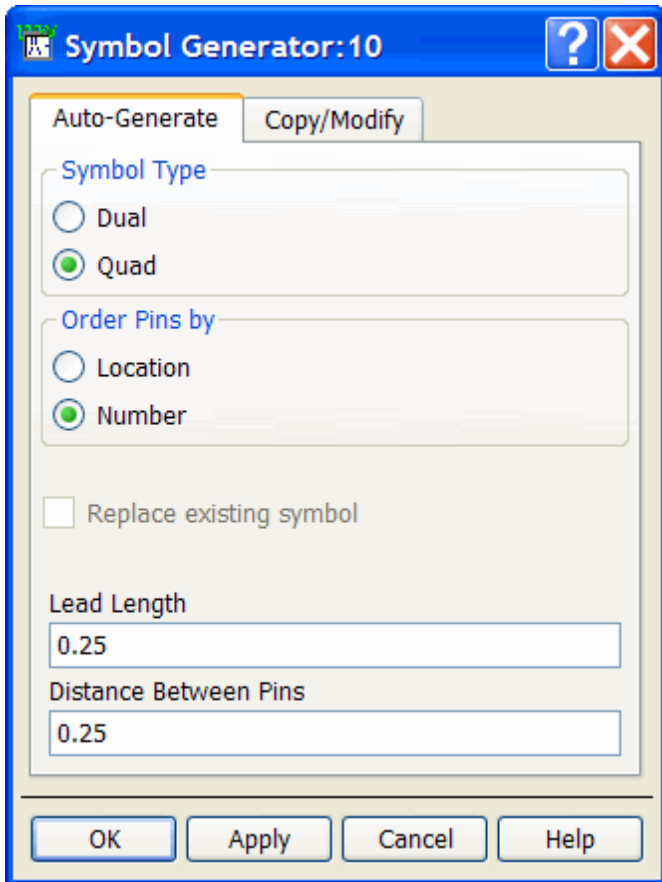
- If the analyzer determines that you have a top level design type, you will be shown a page where you can choose nodes in the top level that will be promoted to being I/O terminals. This allows you to connect inputs and outputs into your design so that the design can interact with other ADS elements (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.



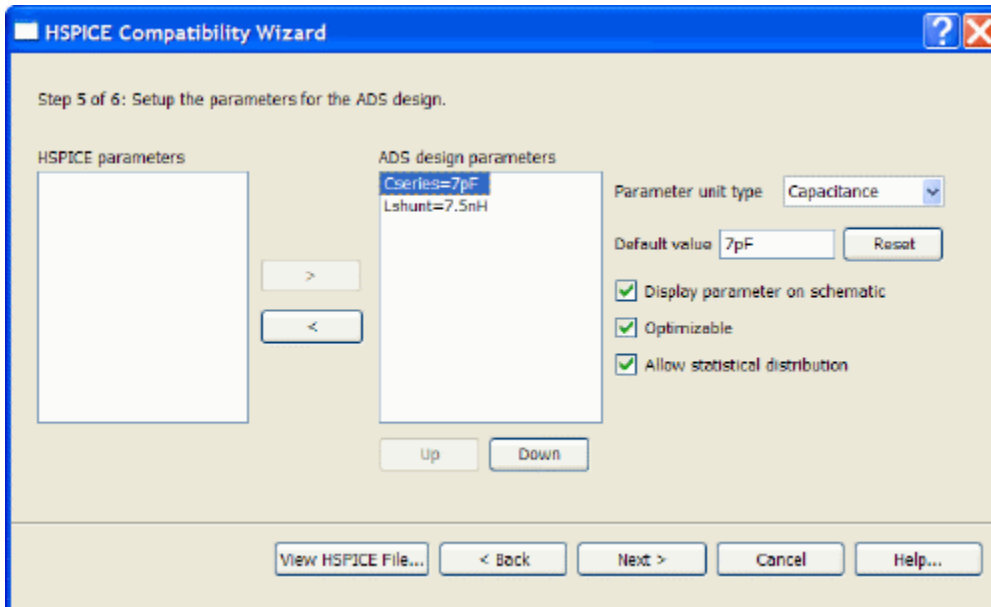
6. The next page allows you to choose the symbol generation parameters. Based on the terminal setup page (or on the .subckt terminal definition for the circuit that was chosen), the pins for the ADS design is populated. This is a non-editable field. To change the order or pins you must go back to the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



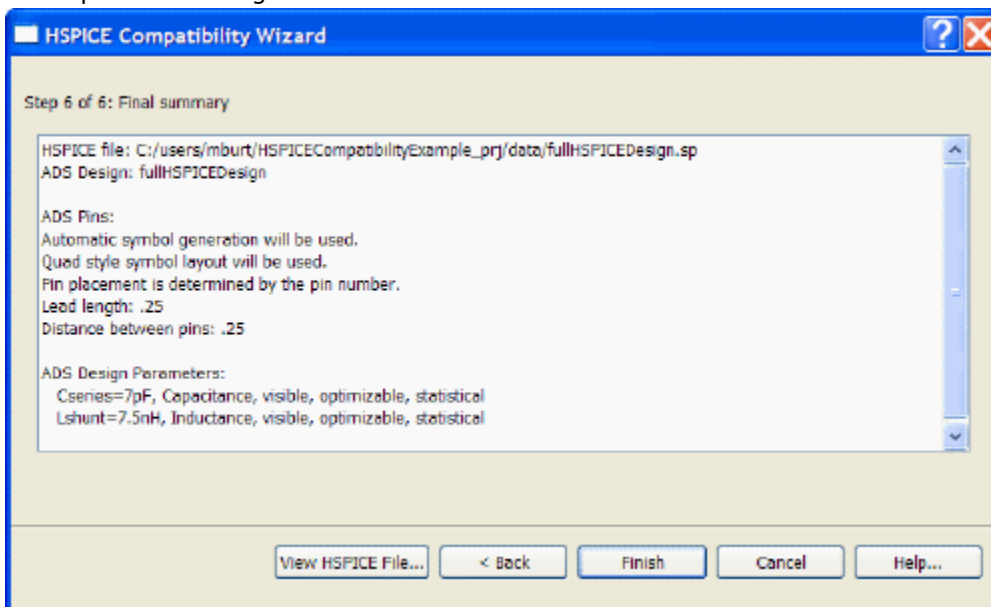
To change the settings, click Edit Symbol Generation Settings. This displays the standard ADS auto-generation dialog for symbols. When finished, click OK to update the summary. These settings will be used to generate the symbol when the wizard is finished.



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.



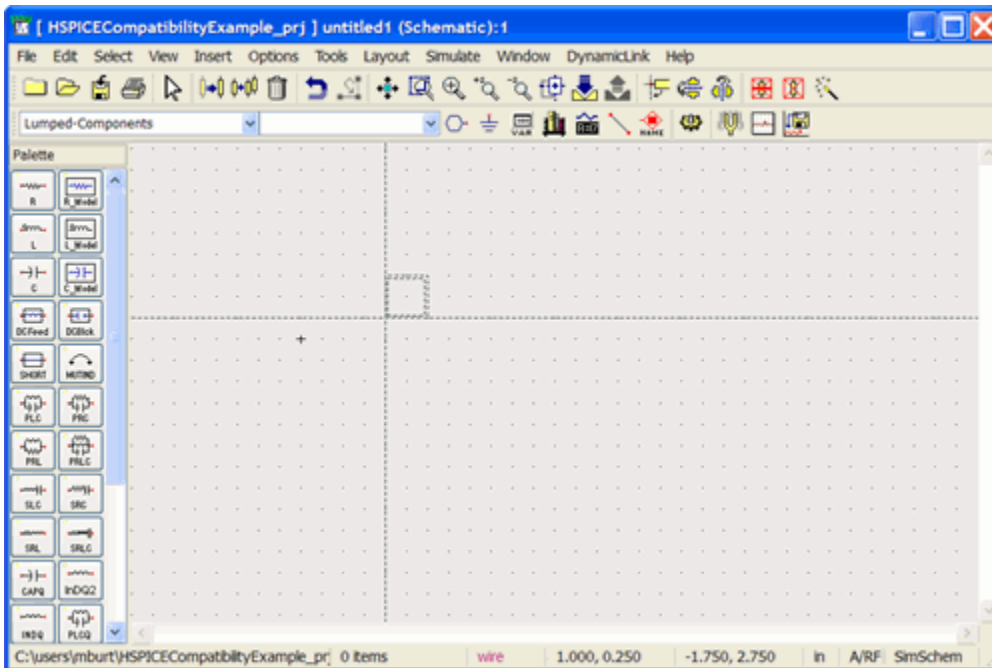
9. To complete the component, click Finish.  
 Designs are created in the current project. You can cancel out if a design in the current project will be overwritten. Note that if the dialog is brought up again within the same ADS session, the last settings are reused. Changing the file will delete the settings.  
 During the generation of the component, a new schematic window is opened. In this window, a schematic with ports is created, and a symbol is generated based on the symbol generation settings that were specified in the wizard. When the component has been created and saved, the window is closed.

## Utilizing HSPICE PDK Components in a PDE Design

## Advanced Design System 2008

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

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.



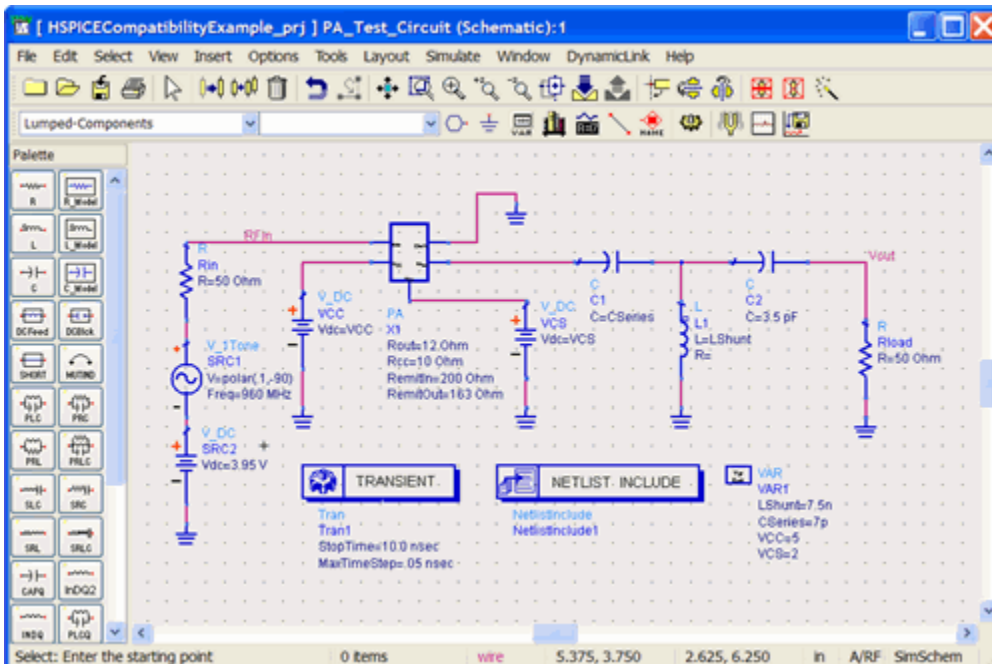
If you choose to push into the HSPICE design, instead of going into the schematic, a text editor window will be opened that allows you to edit the HSPICE file.

```

subcircuitHSPICEExample.sp - WordPad
File Edit View Insert Format Help
* 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

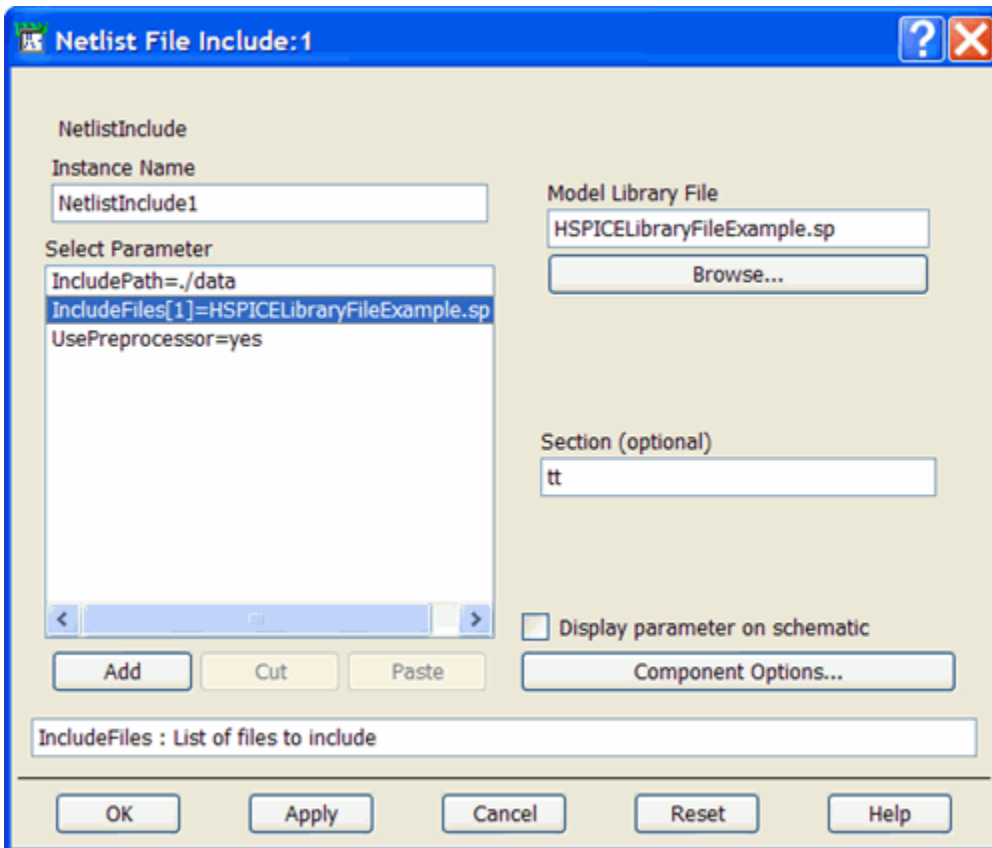
.subckt PA RfOut VCC RFin VCS Ground
.param Rout=12 Rcc=10 RemitIn=200 RemitOut=163
QBJT1 _net7 RFin _net4 Q37MOD
QBJT2 _net11 VCS _net1 Q34MOD
QBJT3 _net8 VCS _net14 Q34MOD
QBJT4 _net4 VCS _net10 Q34MOD
QBJT5 _net6 VCS _net13 Q34MOD
QBJT6 _net7 _net4 _net11 Q34MOD
QBJT7 _net7 _net4 _net8 Q34MOD
QBJT8 _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
    
```

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, PAT and tabular-form of PWL. For use model of the I element, refer to HSPICE Simulation and Analysis User Guide, Chapter 5: Sources and Stimuli.

| Unsupported HSPICE Keywords |
|-----------------------------|
| LFSR                        |
| PAT                         |
| PWL (tabular)               |

J Element

## Advanced Design System 2008

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, Chapter 5: 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, PAT and tabular-form of PWL. For use model of the V element, refer to HSPICE Simulation and Analysis User Guide, Chapter 5: Sources and Stimuli.

| Unsupported HSPICE Keywords |
|-----------------------------|
| LFSR                        |
| PAT                         |
| PWL (tabular)               |

## 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

## 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 `.lib ... endl` 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 `.macro ... .eom` 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 `.subckt ... .ends` to define a subcircuit.

## 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 the manuals regarding [Design Kit Development](#). 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

For an HSPICE PDK, the process component generation is similar to what is described in [Example Process Component with Forms and Formsets](#). 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 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.