HSPICE/SPICE Interface Reference

Product Version ICADVM20.1
October 2020

Contents

Preface

The Virtuoso design environment provides interfaces to the following products:

- The HSPICE® interface, which lets you run an HSPICE simulation in the Cadence environment.
- The SPICE® interface, which lets you run a SPICE simulation in the Cadence environment.

This manual describes how you can run simulations using the HSPICE and SPICE interfaces. This manual is intended for engineers and designers of integrated circuits and assumes that you are familiar with the HSPICE® simulator from Synopsys, Inc and the SPICE® simulator from the University of California at Berkeley.

This preface contains the following topics:

- Scope
- Licensing Requirements
- Related Documentation
- Additional Learning Resources
- Customer Support
- Feedback about Documentation
- Typographic and Syntax Conventions

Scope

Unless otherwise noted, the functionality described in this guide can be used in both mature node (for example, IC6.1.8) and advanced node and methodologies (for example, ICADVM20.1) releases.

Label	Meaning

(ICADVM20.1 Only)	Features supported only in the ICADVM20.1 advanced nodes and advanced methodologies release.
(IC6.1.8 Only)	Features supported only in mature node releases.

Licensing Requirements

- For information on licensing in the Virtuoso design environment, see *Virtuoso Software Licensing and Configuration Guide*.
- For information on licensing for the HSPICE simulator, contact Synopsys, Inc.
- For information on licensing for the SPICE simulator, contact the University of California at Berkeley.

Related Documentation

Installation, Environment, and Infrastructure

- *Cadence Installation Guide*
- *Virtuoso Design Environment User Guide*
- *Simulation Environment Help*

Additional Learning Resources

Video Library

The Video Library on the Cadence Online Support website provides a comprehensive list of videos on various Cadence products.

To view a list of videos related to a specific product, you can use the *Filter Results* feature available in the pane on the left. For example, click the *Virtuoso Layout Suite* product link to view a list of videos available for the product.

You can also save your product preferences in the Product Selection form, which opens when you click the *Edit* icon located next to *My Products*.

Virtuoso Videos Book

You can access certain videos directly from Cadence Help. To learn more about this feature and to access the list of available videos, see Virtuoso Videos.

Rapid Adoption Kits

Cadence provides a number of Rapid Adoption Kits that demonstrate how to use Virtuoso applications in your design flows. These kits contain design databases and instructions on how to run the design flow.

To explore the full range of training courses provided by Cadence in your region, visit Cadence Training or write to training_enroll@cadence.com.

Note: The links in this section open in a separate web browser window when clicked in Cadence Help.

Help and Support Facilities

Virtuoso offers several built-in features to let you access help and support directly from the software.

- The Virtuoso *Help* menu provides consistent help system access across Virtuoso tools and applications. The standard Virtuoso *Help* menu lets you access the most useful help and support resources from the Cadence support and corporate websites directly from the CIW or any Virtuoso application.
- The Virtuoso Welcome Page is a self-help launch pad offering access to a host of useful knowledge resources, including quick links to content available within the Virtuoso installation as well as to other popular online content.

  The Welcome Page is displayed by default when you open Cadence Help in standalone mode from a Virtuoso installation. You can also access it at any time by selecting *Help – Virtuoso Documentation Library* from any application window, or by clicking the *Home* button on the Cadence Help toolbar (provided you have not set a custom home page).

For more information, see Getting Help in *Virtuoso Design Environment User Guide.*

Customer Support

For assistance with Cadence products:

- Contact Cadence Customer Support

  Cadence is committed to keeping your design teams productive by providing answers to technical questions and to any queries about the latest software updates and training needs. For more information, visit https://www.cadence.com/support.

- Log on to Cadence Online Support

  Customers with a maintenance contract with Cadence can obtain the latest information about various tools at https://support.cadence.com.

Feedback about Documentation

You can contact Cadence Customer Support to open a service request if you:

- Find erroneous information in a product manual
- Cannot find in a product manual the information you are looking for
- Face an issue while accessing documentation by using Cadence Help

You can also submit feedback by using the following methods:

- In the Cadence Help window, click the *Feedback* button and follow instructions.
- On the Cadence Online Support Product Manuals page, select the required product and submit your feedback by using the *Provide Feedback* box.

Typographic and Syntax Conventions

The following typographic and syntax conventions are used in this manual.

text	Indicates names of manuals, menu commands, buttons, and fields.
`text`	Indicates text that you must type as presented. Typically used to denote command, function, routine, or argument names that must be typed literally.
`z_argument`	Indicates text that you must replace with an appropriate argument value. The prefix (in this example, *`z_`*) indicates the data type the argument can accept and must not be typed.
|	Separates a choice of options.
`{ }`	Encloses a list of choices, separated by vertical bars, from which you **must** choose one.
`[ ]`	Encloses an optional argument or a list of choices separated by vertical bars, from which you **may** choose one.
`[ ?argName` *`t_arg`* `]`	Denotes a *key argument*. The question mark and argument name must be typed as they appear in the syntax and must be followed by the required value for that argument.
`...`	Indicates that you can repeat the previous argument. Used with brackets to indicate that you can specify zero or more arguments. Used without brackets to indicate that you must specify at least one argument.
`,...`	Indicates that multiple arguments must be separated by commas.
`=>`	Indicates the values returned by a Cadence® SKILL® language function.
`/`	Separates the values that can be returned by a Cadence SKILL language function.

If a command-line or SKILL expression is too long to fit within the paragraph margins of this document, the remainder of the expression is moved to the next line and indented. In code excerpts, a backslash (\ ) indicates that the current line continues on to the next line.

1

Introduction

This chapter includes the following topics:

- Product Overview
- Simulation Flow

Note: Ensure that the following third party tools are installed before you use the HSPICE or SPICE interface:

- HSPICE® simulator from Synopsys, Inc
- SPICE® simulator from University of California at Berkeley

Product Overview

SPICE

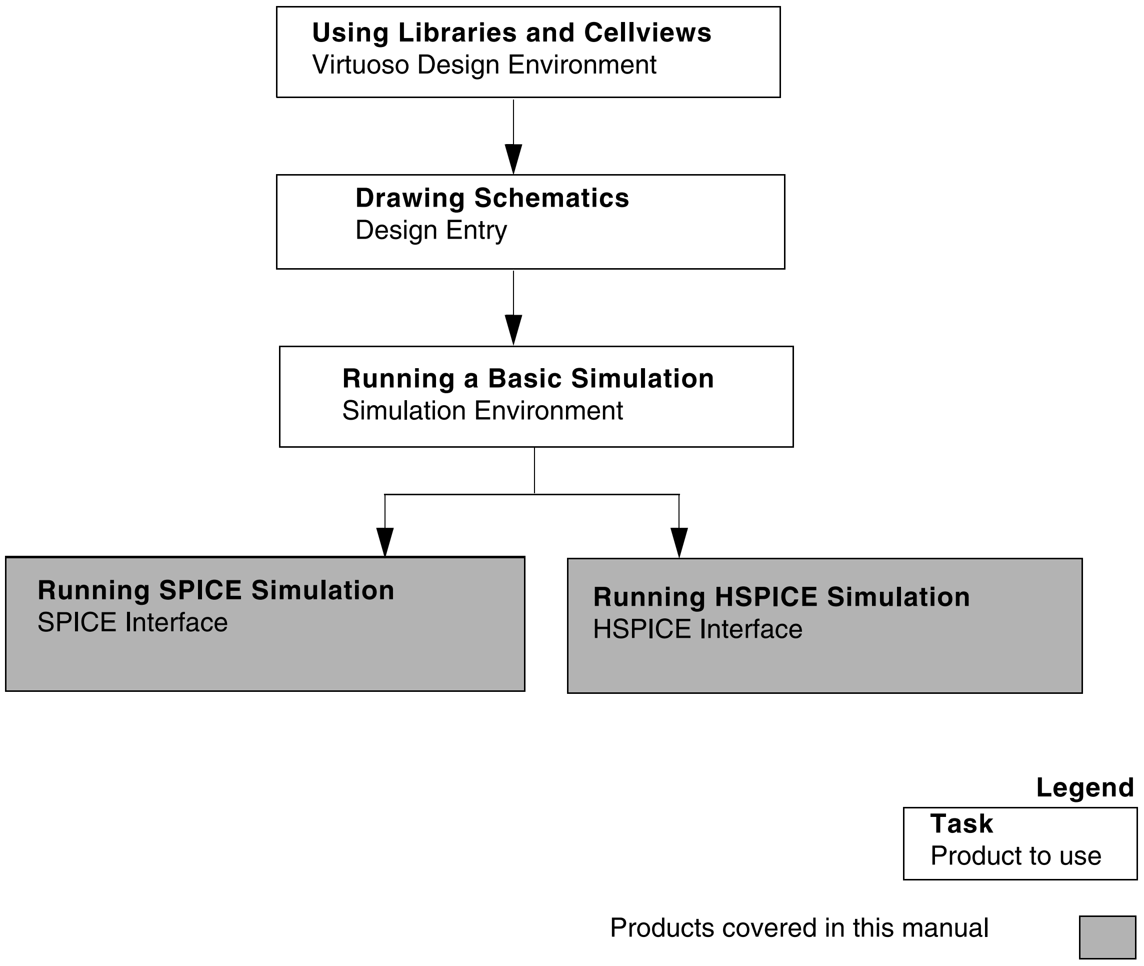
SPICE is a general-purpose circuit simulator developed by the University of California at Berkeley. It is used for nonlinear DC, nonlinear transient, and linear AC analysis. Cadence does not sell the SPICE program but supplies a copy of this public Domain program and its manual free of charge when you purchase the interface.

HSPICE

HSPICE is a general-purpose circuit simulator from Synopsys, Inc. It has an extensive set of built-in device models, including models for small geometry MOSFETs and MESFETs. The program is compatible with SPICE and MSING input formats. Cadence supports a library of primitives and a full interface for HSPICE.

Simulation Flow

The following chart shows the flow of the tasks involved in running a simulation using HSPICE. This manual covers the products in the shaded boxes.


Using Libraries and Cellviews
Virtuoso Design Environment
Drawing Schematics
Design Entry
Running a Basic Simulation
Simulation Environment
Running SPICE Simulation
SPICE Interface
Running HSPICE Simulation
HSPICE Interface
Legend
Task
Product to use
Products covered in this manual

2

SPICE Circuit Simulation Interface

This chapter contains the following topics:

Overview

To set up and run a basic simulation using menus and forms, refer to *Simulation Environment Help*. Here is information specific to running the SPICE simulator:

■ SPICE does not have a command to read an input file, so the *netlist* file must be specified in the *control* file with the simulation environment file inclusion function, *"[!netlist]"*; this tells the interface to include the *netlist* file in the file that is passed to SPICE. For an example of this, refer to the sample SPICE *control* file in "Example of a SPICE Simulation Run". Any other command or stimulus files you want to use as input to SPICE can be specified in the same way.

■ The template *control* file for SPICE includes two other files in addition to the *netlist* file, *spice.inp* and *spice.sim*. These files are for stimulus data and simulator commands. The *Initialize* command from the Simulation menu automatically creates templates for these files in a new run directory.

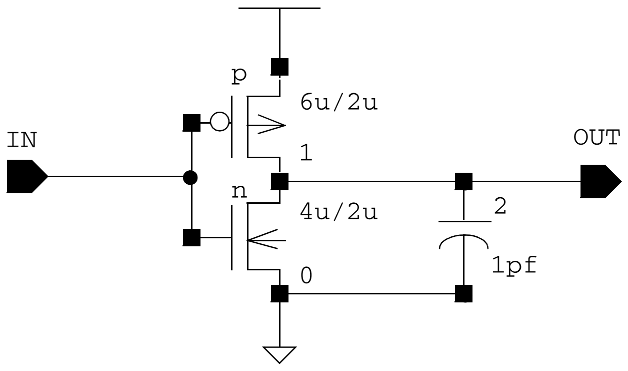
■ The waveform interface for SPICE does not handle multiple analysis simulation runs. To view waveforms, perform one analysis for each simulation run.

■ The Waveform Display program does not display results of AC small-signal analysis simulation runs. Use the SPICE .PRINT or .PLOT commands to generate SPICE text output, and read it using the view output command.

Example of a SPICE Simulation Run

This section shows an example of input and output files needed for running a SPICE simulation.

The following input and output files are for running the simulation on the inverter shown below:



- The *control* file is used as an input for running the simulation.
- The netlist file created by running the netlister on the design.
- The si.inp file is created by the simulation interface and is passed to SPICE. This file is formed by including the specified files in the control file and mapping the user-defined names to numbers suitable for SPICE.
- The spice.inp file is created by the *Initialize* command. Modify this file by adding the stimulus data.
- The *spice.sim* file is created by the *Initialize* command. Modify this file by adding the simulator commands.
- The *si.log* file is the log of the simulation run.
- The *si.out* file is the text output created by SPICE.

control file

```
* Spice control file

.options acct opts nopage limpts=1000
.width in=80 out=80
[!spice.inp]
[!netlist]
[!spice.sim]
.end
```

netlist **file**

```
* net 1 = vdd!
* net 0 = gnd!
* net 2 = /OUT
* net 3 = /IN
.MODEL Model1 pmos level=2 vto=-.7 kp=1.5e-05 gamma=.4
+lambda=.03 tox=6.e-07 xqc=.5
* pmos(0) = /1
M$#0 1 3 2 1 Model1 l=2u w=6u
* capacitor(1) = /2
C$#1 2 0 poly 1pf
.MODEL Model3 nmos level=2 vto=.7 kp=3.e-05 gamma=.2 +lambda=.02
+ tox=6.e-07 xqc=.5
* nmos(2) = /0
M$#2 2 3 0 0 Model3 l=2u w=4u
```

spice.inp **file**

```
vdd [#vdd!] [#gnd!] dc 5v
vin [#/IN] 0 pwl 0 0 100ns 5v 150ns 5v 250ns 0
```

spice.sim **file**

```
.tran 1ns 300ns
```

si.inp **file: Used as Input to SPICE**

```
* Spice control file

.options acct opts nopage limpts=1000
.width in=80 out=80
vdd 1 0 dc 5v
vin 3 0 pwl 0 0 100ns 5v 150ns 5v 250ns 0
* net 1 = vdd!
* net 0 = gnd!
* net 2 = /OUT
* net 3 = /IN
.MODEL Model1 pmos level=2 vto=-.7 kp=1.5e-05 gamma=.4
+lambda=.03 tox=6.e-07 xqc=.5
* pmos(0) = /1
M$#0 1 3 2 1 Model1 l=2u w=6u
* capacitor(1) = /2
C$#1 2 0 poly 1pf
.MODEL Model3 nmos level=2 vto=.7 kp=3.e-05 gamma=.2
    lambda=.02
+ tox=6.e-07 xqc=.5
* nmos(2) = /0
M$#2 2 3 0 0 Model3 l=2u w=4u
.tran 1ns 300ns

.end
```

si.log file: Produced from the Simulation Run

```
si version 4.0.55 Wed Apr 18 21:51:31 PDT 1990          (cds2082)
si: Loading user defined simulation run control file "~/.simrc".
si: Loading simulation environment file "/usr/mnt2/hpeter/4.0/group/spice/test2/
run1/si.env".
si: Loading simulation capabilities file "/usr/mnt2/hpeter/4.0/etc/skill/si/
simcap.ile".

Running simulation in directory: "/usr/mnt2/hpeter/4.0/group/spice/test2/run1".

Running netlist
Begin netlist:    Apr 26 11:58:45 1990
    simulation library path = ". ~"
    simulation library = testLib
               library configuration = default
               cell = spice.cct2
               view = schematic
    view list = ("spice" "cmos.sch" "schematic")
    stopping view list = ("spice")
End netlist:      Apr 26 11:58:54 1990

Running simin

Running runsim with simulator: "spice"

Begin simulation:    Apr 26 11:58:56 1990
End simulation:      Apr 26 12:02:38 1990

Running simout
Simulation completed successfully.
```

si.out file: Output of a SPICE Simulation Run

```
***4/26/90 ***** SPICE 2G.6 3/16/83 ****14:35:31***

* SPICE CONTROL FILE

**** INPUT LISTING  TEMPERATURE = 27.000 DEG C

*******************************************************
.OPTIONS ACCT OPTS NOPAGE LIMPTS=1000
.WIDTH IN=80 OUT=80
VDD vdd! gnd! DC 5V
VIN /IN gnd! PWL 0 0 100NS 5V 150NS 5V 250NS 0
* NET 1   =         VDD!
* NET 0   =         GND!
* NET 2   =         /OUT
* NET 3   =         /IN
.MODEL MODEL1 PMOS LEVEL=2      VTO=-.7 KP=1.5E-05
          GAMMA=.4
+LAMBDA=.03      TOX=6    .E-07 XQC=.5
* PMOS(0) = /1
M/1 vdd! /IN /OUT vdd! MODEL1 L=2U      W=6U
* CAPACITOR(1) = /2
C/2 /OUT gnd! POLY 1PF
.MODEL MODEL3 NMOS LEVEL=2 VTO=.7 KP=3.E-05 GAMMA=.2
+ LAMBDA=.02
+ TOX=6.E-07      XQC=.5
* NMOS(2) = /0
M/0 /OUT /IN gnd! gnd! MODEL3 L=2U      W=4U
.TRAN 1NS 300NS
.END
```

```
*****************************************************************
**** MOSFET MODEL PARAMETERS  TEMPERATURE = 27.000 DEG C
          MODEL1    MODEL3
TYPE      PMOS      NMOS
LEVEL     2.000     2.000
VTO       -0.700    0.700
KP        1.50D-05  3.00D-05
GAMMA     0.400     0.200
LAMBDA    3.00D-02  2.00D-02
TOX       6.00D-07  6.00D-07
XQC       0.500     0.500
**** OPTION SUMMARY TEMPERATURE = 27.000 DEG C
DC ANALYSIS -
GMIN      =         1.000D-12
RELTOL    =         1.000D-03
ABSTOL    =         1.000D-12
VNTOL     =         1.000D-06
LVLCOD    =         1
ITL1      =         100
ITL2      =         50
PIVTOL    =         1.000D-13
PIVREL    =         1.000D-03
TRANSIENT ANALYSIS -
METHOD    =         TRAP
MAXORD    =         2
CHGTOL    =         1.000D-14
TRTOL     =         7.000D+00
LVLTIM    =         2
MU        =         0.500
ITL3      =         4
ITL4      =         10
ITL5      =         5000
MISCELLANEOUS -
LIMPTS    =         1000
LIMTIM    =         2
CPTIME    =         100000000
NUMDGT    =         4
TNOM      =         27.000
DEFL      =         1.000D-04
DEFW      =         1.000D-04
DEFAD     =         0.000D+00
DEFAS     =         0.000D+00

*************************************************************
MOSFET MODEL PARAMETERS  TEMPERATURE = 27.000 DEG C
          MODEL1    MODEL3
TYPE      PMOS      NMOS
LEVEL     2.000     2.000
VTO       -0.700    0.700
KP        1.50D-05  3.00D-05
GAMMA     0.400     0.200
LAMBDA    3.00D-02  2.00D-02
TOX       6.00D-07  6.00D-07
XQC       0.500     0.500
**** OPTION SUMMARY TEMPERATURE = 27.000 DEG C
DC ANALYSIS -
GMIN      =         1.000D-12
RELTOL    =         1.000D-03
ABSTOL    =         1.000D-12
VNTOL     =         1.000D-06
LVLCOD    =         1
```

```
ITL1      =         100
ITL2      =         50
PIVTOL    =         1.000D-13
PIVREL    =         1.000D-03
TRANSIENT ANALYSIS -
METHOD    =         TRAP
MAXORD    =         2
CHGTOL    =         1.000D-14
TRTOL     =         7.000D+00
LVLTIM    =         2
MU        =         0.500
ITL3      =         4
ITL4      =         10
ITL5      =         5000
MISCELLANEOUS -
LIMPTS    =         1000
LIMTIM    =         2
CPTIME    =         100000000
NUMDGT    =         4
TNOM      =         27.000
DEFL      =         1.000D-04
DEFW      =         1.000D-04
DEFAD     =         0.000D+00
DEFAS     =         0.000D+00
********************************************************
INITIAL TRANSIENT SOLUTION  TEMPERATURE = 27.000 DEG C
NODE VOLTAGE NODE VOLTAGE NODE VOLTAGE
(vdd!) 5.0000 (/OUT) 5.0000 (/IN) 0.0000
VOLTAGE SOURCE CURRENTS
NAME      CURRENT
VDD       -6.933D-12
VIN       0.000D+00
TOTAL POWER DISSIPATION 3.47D-11 WATTS
********************************************************
OPERATING POINT INFORMATION  TEMPERATURE = 27.000 DEG C
**** MOSFETS
          M/1       M/0
MODEL     MODEL1    MODEL3
ID        6.80E-12  1.93E-12
VGS       -5.000    0.000
VDS       0.000     5.000
VBS       0.000     0.000
JOB CONCLUDED
**** JOB STATISTICS SUMMARY  TEMPERATURE = 27.000 DEG C
NUNODS   NCNODS   NUMNOD   NUMEL    DIODES   BJTS     JFETS    MFETS
4        4        4        5        0        0        0        2
NUMTEM   ICVFLG   JTRFLG   JACFLG   INOISE   IDIST    NOGO
1        0        301      0        0        0        0
NSTOP    NTTBR    NTTAR    IFILL    IOPS     PERSPA
6.       13.      13.      0.       21.      63.889
NUMTTP   NUMRTP   NUMNIT   MAXMEM   MEMUSE   COPYKNT
71.      4.       170.     249984   1576     15052.
READIN    0.42
SETUP     0.05
TRCURV    0.00      0.
DCAN      0.18      13.
DCDCMP    0.067     2.
DCSOL     0.067
ACAN      0.00      0.
TRANAN    3.63      170.
OUTPUT    0.00
```

```
LOAD      3.067
CODGEN    0.000     1fs       0.
CODEXC    0.000
MACINS    0.000
OVERHEAD  0.07
TOTAL JOB TIME  4.35
********************************************************
```

3

HSPICE Circuit Simulation Interface

This chapter contains the following topics:

- Overview
- Running HSPICE
- Hierarchical Netlisting

Overview

Using the HSPICE interface is similar to using the SPICE interface. Before reading this chapter, which contains information specific to running an HSPICE simulation, read the chapter on the SPICE interface.

- With the HSPICE interface you can
    - Generate flat or hierarchical netlists
    - Run local and remote simulations
    - Use element and model parameters supported by HSPICE
- HSPICE differs from SPICE in its handling of waveform output from multiple analysis simulation runs.

  The waveforms are written in the Parameter Storage Format (PSF) format if the *psf=1* option is set in your *control* file. The waveforms are stored in the *psf* directory in the simulation run directory and can be viewed using the Waveform Display program.

Running HSPICE

This section describes the following:

- Creating the schematic

- Generating the run directory
- Editing the control file
- Starting the analysis run
- Generating the netlist
- Creating the input file
- Running the simulation
- Translating the output
- Viewing the results

Most of this information is covered in greater detail in the Virtuoso Schematic Editor L and Simulation Environment Help. Here you will find only what is specific to HSPICE.

Creating the Schematic

Create schematics for HSPICE simulation in the same way as any other schematic in the Virtuoso Design Environment. The main difference is that each cell in the schematic (for example, transistors, resistors, capacitors) must have a *symbol* and an *hspice* view to be recognized by the HSPICE interface. For a list of the netlisting properties required for the *hspice* view, see "Primitive Cell Requirements".

Generating the Run Directory

The first time you run a simulation, select the *Initialize* command from the Simulation menu to generate a run directory. This run directory is where the *control, netlist*, and simulation input and output files are kept for the simulation being run. When you want to keep an older simulation instead of overwriting it, you can generate and use multiple simulation run directories. Refer to *Simulation Environment Help*.

Editing the Control File

When the system initially creates the run directory, it also creates a dummy *control* file.

Edit this file to include any information that is not in the schematic. Add the *psf* option on the *.options* line in the *control* file. Set *psf=1* or *psf=2*, which tells HSPICE to generate a waveform file in the binary or ASCII PSF format respectively, as required by the simulation environment. You can also specify just the `.option` keyword in the control file. To disable the PSF and other options in an hspiceD netlist specify just the `.option` keyword in the control file.

Note: From the IC 6.1.2 release, Cadence does not support the WSF waveform format. Because of this, do not add the *sda=2* option in the control file to enable HSPICE to create waveforms in the WSF format.

Edit the *hspice.inp* file to add stimulus data and edit the *hspice.sim* file to add simulator commands.

Starting the Analysis Run

With the schematic entered and the *control* file generated, you are ready to start the simulation. Select the *Netlist/Simulate* command from the Simulation menu. This command netlists the schematic and starts the simulation; simulation can be run in the background or foreground. Each step in the analysis process is described below.

Generating the Netlist

The simulation environment creates an HSPICE netlist for the desired schematic. This netlist contains the connectivity description in the proper format for HSPICE. See "Formatting Functions" for information about formats. You can generate a flat or a hierarchical netlist. Both have the same information, but their formats are different. You control the type of netlist generated by setting the *simNetlistHier* simulation environment variable. If this variable is set to *t,* a hierarchical netlist is created; otherwise, a flat netlist is created by default. Typically, you set this variable in your *.simrc* file.

Note the following:

- By default, parameter values are converted to engineering notation in HSPICE netlist. For example, the parameter `w=100u` is written as `W=100e-6` in the netlist. Set the `setEngNotation` environment variable to in your `.cdsenv` file to print the parameter values as is (without conversion to engineering notation) in the netlist.

  ```
  hspiceD.envOpts setEngNotation boolean nil
  ```

  To revert to the default behavior, set the variable to `t`. For example:

  ```
  hspiceD.envOpts setEngNotation boolean t
  ```

- For devices other than diodes and MOS transistors, the netlister netlists only a predefined list of parameters. If you add user defined parameters in the CDF Simulation Information (`simInfo`) of devices other than diodes and MOS transistors, those parameters will not be netlisted. For such devices, you can write a custom netlist procedure to netlist user defined parameters.

Creating the Input File

Once the netlist is generated, the HSPICE interface automatically translates the instance and net names in the *control*, *hspice.inp*, *hspice.sim*, and *netlist* files to legal names for HSPICE. When the translation completes, these files are assembled to create the HSPICE *si.inp* input file.

Running the Simulation

When the simulator input file (*si.inp*) is created, simulation starts. If the *simHost* variable is set to a different machine, the simulation runs remotely.

For more information about remote simulation, refer to *Simulation Environment Help*.

Translating the Output

When the simulation is finished, the simulation interface translates net and instance names that were previously translated back to the original user-assigned names. The resulting translated text output is stored in the *si.out* file*.*

Viewing the Results

The simulation environment notifies you when analysis is complete so you can view the outputs. Information about the simulation run is recorded in the *si.log* file when a background simulation is run.

Hierarchical Netlisting

The hierarchical netlister produces a netlist that is easier to read and understand than one that has been flattened. For more information about flat netlisting, refer to *Simulation Environment Help*. The following are primary features of a hierarchical netlist:

- The hierarchy of the netlist duplicates the hierarchy of your design. The netlister creates a separate subcircuit for each cell in your schematic. This can dramatically reduce the number of lines in the netlist, since the subcircuit definition is printed only once and all instances of the cell are netlisted as calls to the subcircuit.
- Unlike the flat netlister, which translates every instance and net name to a unique name, the hierarchical netlister translates only names that are illegal to HSPICE. To avoid naming conflicts, the hierarchical netlister makes every effort to keep the original user-assigned names in the netlist. When necessary, names are mapped, but the mapping is minimal. Characters that are illegal in HSPICE names are

  . , ( ) [ ] $ ' < >

  When any of these characters is found in a name, the character is automatically deleted. In some cases, the name is completely remapped. Usually this occurs when you have specified a name that is too long. HSPICE names are limited to eight characters. If a name is longer than eight characters, it is mapped by the interface.

  When HSPICE maps the entire name, it assigns a unique number preceded by an *n* for net names, an *i* for instance names, and an *m* for macro and model names.

  The first character of an element name in the netlist indicates the element type. The netlister automatically adds a prefix to all instance names. For example, MOSFET instance names are prefixed with *m***,** and resistor names with *r*.

Primitive Cell Requirements

A cell must have both a *symbol* and an *hspice* view to be recognized by the HSPICE interface. The *hspice* view for a cell must contain the same pins that exist in the *symbol* view for the cell. For netlisting, define the following properties in the *hspice* view:

- **NLPElementPostamble**

  Indicates to the flat netlister how to format the element cards for an instance of the cell.

- **NLPModelPreamble**

  Indicates to the flat netlister how to format the model card for an instance of the cell.

- **hnlHspiceFormatInst**

  Indicates to the hierarchical netlister what procedure to call to format and print the element cards for an instance of the cell.

- **hnlHspiceParamList**

  Indicates to the hierarchical netlister what parameters can be inherited. The value of this parameter must be the name of a Cadence SKILL™ language variable, whose value is the list of parameters that can be inherited. Any parameter that does not appear on this list cannot inherit its value and must be assigned fixed values.

- **hnlHspiceFormatModel**

  Indicates to the hierarchical netlister what procedure to call to format and print the model card for an instance of the cell.

Example

The following are the property values for the above netlisting properties in the *hspice* view of the *nmos* cell in the *sample* library:

```
NLPElementPostamble = nlpExpr("[@NLPElementComment:%\n]
    [@NLPnmosElementCard]")
NLPModelPreamble = nlpExpr("[@NLPmosfetModelCard]")
hnlHspiceFormatInst = "hnlHspicePrintNMOSfetElement()"
hnlHspiceParamList = "hnlHspiceMOSfetParamList"
hnlHspiceFormatModel = "hnlHspicePrintMOSfetModel()"
```

The flat netlister uses expressions defined in the *nlpglobals* cell to format elements and models. The first two properties, therefore, tell the netlister to format the element and model cards with the *NLPnmosElementCard* and *NLPmosfetModelCard* expressions, defined in the *hspice* view of the *nlpglobals* cell.

For hierarchical netlisting, the element and model cards are formatted using the procedures defined in the *hspice* formater. For the *nmos* transistor, these procedures are called *hnlHspicePrintNMOSfetElement* and *hnlHspicePrintMOSfetModel.*

4

HSPICE/SPICE Elements

This chapter contains the following topics:

- HSPICE/SPICE Elements and Corresponding Library Cells

  This section lists the HSPICE/SPICE elements and the corresponding cells in the sample library.

- HSPICE/SPICE Model and Element Parameters

  This section has information about the library cells, terminal names, element and model parameters, for each HSPICE/SPICE element.

- Formatting Functions

  This section has information about the formatting functions for each HSPICE/SPICE element.

HSPICE/SPICE Elements and Corresponding Library Cells

HSPICE/SPICE Element (Model Type)	Library Cell
R	res
R	resistor
C	cap
C	capacitor
C	pcapacitor
L	inductor
T	tline
G	soi.vc
G	vcisrc
E	sov.vc
E	vcvsrc
F	soi.ic
H	sov.ic
V	sov
V	vsrc
I	soi
I	isrc
D	diode
D	pdiode
Q(npn)	npn
Q(npn)	npns
Q(pnp)	pnp
Q(pnp)	pnps
J(njf)	njfet

HSPICE/SPICE Element (Model Type)	Library Cell
J(pjf)	pjfet
M(nmos)	ndepl
M(nmos)	nfet
M(nmos)	nmos
M(nmos)	nmosd
M(nmos)	nmose
M(nmos)	nsftn
M(nmos)	nxfr
M(pmos)	pdepl
M(pmos)	pfet
M(pmos)	pmos
M(pmos)	pmosd
M(pmos)	pmose
M(pmos)	psftn
M(pmos)	pxfr

HSPICE/SPICE Model and Element Parameters

Resistor

HSPICE/SPICE Element: Resistor
Element Name: R
Used by Library Cells: *res*, *resistor*

Terminal Name	Level of Simulation	Direction
A	Circuit	inputOutput
Y	Circuit	inputOutput

Element Parameter	Data Type	Units
r	float	ohms
tc	string	--

Capacitor

HSPICE/SPICE Element: Capacitor
Element Named: C
Used by Library Cells*: cap, capacitor*, *pcapacitor*

Terminal Name	Level of Simulation	Direction
Y	Circuit	inputOutput
gnd!	Circuit	inputOutput

Element Parameter	DataType	Units
c	float	farads
ic	string	--

Inductor

HSPICE/SPICE Element: Inductor
Element Name: L
Used by Library Cell*: inductor*

Terminal Name	Level of Simulation	Direction
PLUS	Circuit	inputOutput
MINUS	Circuit	inputOutput

Element Parameter	Data Type	Units
l	float	henrys
ic	string	--

Transmission Line

HSPICE/SPICE Element: Transmission Line
Element Name: T
Used by Library Cell*: tline*

Terminal Name	Level of Simulation	Direction
N1	Circuit	inputOutput
N2	Circuit	inputOutput
N3	Circuit	inputOutput
N4	Circuit	inputOutput

Element Parameter	Data Type	Units
z0	float	ohms
td	float	seconds
f	float	hertz
nl	float	unitless
ic	string	--

Diode

HSPICE/SPICE Element: Diode
Element Name: D
Model Type: D
Used by Library Cell*: diode*

Terminal Name	Level of Simulation	Direction
PLUS	Circuit	inputOutput
MINUS	Circuit	inputOutput

Element Parameter	Data Type	Units
area	float	unitless
off	string	“off”
ic	string	--

Model Parameter	Data Type	Units
is	float	amperes
rs	float	ohms
n	float	unitless
tt	float	seconds
cjo	float	farads
vj	float	volts
m	float	unitless
eg	float	electronvolts
xti	float	unitless
kf	float	unitless
af	float	unitless
fc	float	unitless

Model Parameter	Data Type	Units
ibv	float	amperes
bv	float	volts

BJT

HSPICE/SPICE Element: BJT
Element Name: Q
Model Type: NPN, PNP
Used by Library Cells*: npn, npns, pnp, pnps*

Terminal Name	Level of Simulation	Direction
C	Circuit	inputOutput
B	Circuit	inputOutput
E	Circuit	inputOutput
SUB	Circuit	inputOutput

Element Parameter	Data Type	Units
area	float	unitless
off	string	“off”
ic	string	--

Model Parameter	Data Type	Units
is	float	amperes
bf	float	unitless
nf	float	unitless
ise	float	amperes
ne	float	unitless
br	float	unitless

Model Parameter	Data Type	Units
nr	float	unitless
isc	float	amperes
nc	float	unitless
rb	float	ohms
rbm	float	ohms
re	float	ohms
rc	float	ohms
cje	float	farads
vje	float	volts
mje	float	unitless
tf	float	seconds
xtf	float	unitless
itf	float	amperes
ptf	float	degrees
cjc	float	farads
vjc	float	volts
mjc	float	unitless
xcjc	float	unitless
tr	float	seconds
cjs	float	farads
vjs	float	volts
mjs	float	unitless
xtb	float	unitless
eg	float	electronvolts
xti	float	unitless
kf	float	unitless
af	float	unitless

Model Parameter	Data Type	Units
fc	float	unitless
vtf	float	volts
irb	float	amperes
ikr	float	amperes
var	float	volts
vaf	float	volts
ikf	float	amperes

JFET

HSPICE/SPICE Element: JFET
Element Name: J
Model Type: NJF, PJF
Used by Libary Cells*: njfet, pjfet*

Terminal Name	Level of Simulation	Direction
D	Circuit	inputOutput
G	Circuit	inputOutput
S	Circuit	inputOutput

Element Parameter	Data Type	Units
area	float	unitless
off	string	“off”
ic	string	--

Model Parameter	Data Type	Units
vto	float	volts
beta	float	amperes/square volts

Model Parameter	Data Type	Units
lambda	float	1/volts
rd	float	ohms
rs	float	ohms
cgs	float	farads
cgd	float	farads
pb	float	volts
is	float	amperes
kf	float	unitless
af	float	unitless
fc	float	unitless

MOSFET

HSPICE/SPICE Element: MOSFET
Element Name: M
Model Type: NMOS, PMOS
Used by Library Cells*: ndepl, nfet, nmos, mosd, nmose, nsftn, nxfr, pdepl, pfet, pmos, pmos, pmose, psftn, pxfr*

Terminal Name	Level of Simulation	Direction
D	Circuit	inputOutput
G	Circuit	inputOutput
S	Circuit	inputOutput
B	Circuit	inputOutput

Element Parameter	Data Type	Units
l	float	meters
w	float	meters

Element Parameter	Data Type	Units
ad	float	square meters
as	float	square meters
pd	float	meters
ps	float	meters
nrd	float	unitless
nrs	float	unitless
off	string	“off”
ic	string	--

Model Parameter	Data Type	Units
level	integer	unitless
vto	float	volts
kp	float	amperes/volts squared
gamma	float	volts**0.5
phi	float	volts
lambda	float	1/volts
rd	float	ohms
rs	float	ohms
cbd	float	farads
cbs	float	farads
is	float	amperes
pb	float	volts
cgso	float	farads/meter
cgdo	float	farads/meter
cgbo	float	farads/meter
rsh	float	ohms/square

Model Parameter	Data Type	Units
cj	float	farads/square meter
mj	float	unitless
cjsw	float	farads/meter
mjsw	float	unitless
js	float	amperes/square meter
tox	float	meters
nsub	float	1/cubic centimeters
nss	float	1/square centimeters
nfs	float	1/square centimeters
tpg	integer	unitless
xj	float	meters
ld	float	meters
uo	float	square centimeters/volt seconds
ucrit	float	volts/centimeter
uexp	float	unitless
utra	float	unitless
vmax	float	meters/second
neff	float	unitless
xqc	float	unitless
kf	float	unitless
af	float	unitless
fc	float	unitless
delta	float	unitless
theta	float	1/volts
eta	float	unitless
kappa	float	unitless

Formatting Functions

This section describes the formatting functions included with the HSPICE/SPICE interface. The cells in the *sample* library that use these formatting functions are also listed. The syntax for these formatting functions is that of the *nlpglobal* functions, but the same parameters are also defined in the *hspice* formatter for hierarchical netlisting.

This section uses the following notations:

`<...>`	name in the brackets is an identifier
`{...}`	item(s) in braces can be repeated as many times as necessary
`[...]`	item(s) in brackets are optional
`@name`	value of property name is substituted

Element Formats

Below is the list of HSPICE/SPICE elements and their formats:

res

```
r<name><A> <Y> @r @ns @tc1 @tc2 @scale @rsh ac=@ac m=@m
```

The *res* cell in the *sample* library uses this format.

resistor

```
r<name> <PLUS> <MINUS> @r @ns @tc1 @tc2 @scale @rsh ac=@ac m=@m
```

The *resistor* cell in the *sample* library uses this format.

cap

```
c<name> <Y> gnd! @c @ns @tc1 @tc2 @scale @cj ic=@ic m=@m
```

The *cap* cell in the *sample* library uses this format.

capacitor

```
c<name><PLUS> <MINUS> @c @ns @tc1 @tc2 @scale @cj ic=@ic m=@m
```

The *capacitor* and *pcapacitor* cells in the *sample* library use this format.

inductor

```
l<name> <PLUS> <MINUS> @l  @tc1 @tc2 @nt ic=@ic
```

The *inductor* cell in the *sample* library uses this format.

transmission line

```
t<name> <N1> <N2> <N3> <N4> z0=@z0 td=@td f=@f nl=@nl ic=@ic
```

The *tline* cell in the *sample* library uses this format.

diode

```
d<name> <PLUS> <MINUS> <cellName> w=@w l=@l area=@area pj=@pj wp=@wp lp=@lp wm=@wm
     lm=@lm @off ic=@ic m=@m
```

The *diode* and *pdiode* cells in the *sample* library use this format.

BJT

```
q<name>  <C> <B> <E> <SUB> <cellName> area=@area @off ic=@ic m=@m
```

The *npns* and *pnps* cells in the *sample* library use this format.

JFET

```
j<name> <D> <G> <S> <cellName> area=@area w=@w l=@l @off ic=@ic m=@m
```

The *njfet* and *pjfet* cells in the *sample* library use this format.

MOSfet

```
m<name> <D> <G> <S> <B> <cellName> w=@w l=@l ad=@ad as=@as pd=@pd ps=@ps nrd=@nrd
     nrs=@nrs off" off ic=@ic m=@m
```

The *ndepl, nfet, nsftn, pdepl, pfet,* and *psftn* cells in the *sample* library use this format.

NMOSfet

```
m<name> <D> <G> <S> gnd! <cellName> w=@w l=@l ad=@ad as=@as pd=@pd ps=@ps nrd=@nrd
     nrs=@nrs off" off ic=@ic m=@m
```

The *nxfr, nmos, nmosd,* and *nmose* cells in the *sample* library use this format.

NPN

```
q<name> <C> <B> <E> vee! <cellName> area=@area off" off ic=@ic m=@m
```

The *npn* cell in the *sample* library uses this format.

PMOSfet

```
"m<name><D> <G> <S> vdd! <cellName> w=@w l=@l ad=@ad as=@as pd=@pd ps=@ps nrd=@nrd
     nrs=@nrs off" off ic=@ic m=@m
```

The *pxfr, pmos, pmosd,* and *pmose* cells in the *sample* library use this format.

PNP

```
"q<name> <C> <B> <E> vcc! <cellName> @area off" off ic=@ic m=@m
```

The *pnp* cell in the *sample* library uses this format.

Model Format

Below is the list of HSPICE/SPICE models and their model card format.

Diode

```
.model <cellName> d level=@level area=@area eg=@eg is=@is jsw=@jsw n=@n pj=@pj
     tlev=@tlev xti=@xti ibv=@ibv tcv=@tcv vb=@vb af=@af kf=@kf rs=@rs trs=@trs
     cjo=@cjo cjp=@cjp cta=@cta ctp=@ctp fc=@fc fcs=@fcs m=@m mjsw=@mjsw pb=@pb
     php=@php tt=@tt ef=@ef er=@er jf=@jf jr=@jr w=@w l=@l tox=@tox wm=@wm lm=@lm
     wp=@wp lp=@lp xm=@xm xp=@xp xoi=@xoi xom=@xom
```

The *diode* and *pdiode* cells in the *sample* library use this format.

BJT

```
.model <cellName> @modelType bf=@bf br=@br bulk=@bulk eg=@eg is=@is iss=@iss nf=@nf
     nr=@nr subs=@subs isc=@isc ise=@ise nc=@nc ne=@ne vaf=@vaf var=@var ikf=@ikf
     ikr=@ikr irb=@irb rb=@rb rbm=@rbm re=@re rc=@rc cjc=@cjc cje=@cje cjs=@cjs
     fc=@fc mjc=@mjc mje=@mje mjs=@mjs vjc=@vjc vje=@vje vjs=@vjs xcjc=@xcjc
     itf=@itf ptf=@ptf tf=@tf tr=@tr vtf=@vtf xtf=@xtf tlev=@tlev tre1=@tre1
     tre2=@tre2 trb1=@trb1 trb2=@trb2 trc1=@trc1 trc2=@trc2 trm1=@trm1 trm2=@trm2
     xtb=@xtb xti=@xti af=@af kf=@kf
```

The *npn, npns, pnp*, and *pnps* cells in the *sample* library use this format.

JFET

```
.model <cellName> @modelType level=%s" level a=@a alpha=@alpha beta=@beta d=@d
     gamds=@gamds lambda=@lambda w=@w l=@l wdel=@wdel ldel=@ldel tcv=@tcv vto=@vto
     eg=@eg gap1=@gap1 gap2=@gap2 is=@is n=@n ni=@ni xti=@xti af=@af kf=@kf rd=@rd
     rg=@rg rs=@rs trd=@trd trg=@trg trs=@trs cgd=@cgd cgs=@cgs fc=@fc m=@m pb=@pb
     capop=@capop ctd=@ctd cts=@cts tt=@tt bex=@bex lam1=@lam1 nchan=@nchan
     sat=@sat ucrit=@ucrit vbi=@vbi vgexp=@vgexp vp=@vp tlev=@tlev tlevc=@tlevc
     tpb=@tpb
```

The *njfet* and *pjfet* cells in the *sample* library use this format.

MOSfet

```
.model<cellName> @modelType level=@level vto=@vto nss=@nss tpg=@tpg phi=@phi
     gamma=@gamma nsub=@nsub bulk=@bulk bex=@bex kp=@kp lambda=@lambda
     ecrit=@ecrit neff=@neff nfs=@nfs ucrit=@ucrit uexp=@uexp uo=@uo utra=@utra
     vmax=@vmax xj=@xj ld=@ld theta=@theta clm=@clm dns=@dns fds=@fds mbl=@mbl
     mob=@mob nu=@nu nwe=@nwe nwm=@nwm scm=@scm tcv=@tcv ufds=@ufds vbo=@vbo
     vfds=@vfds vsh=@vsh wic=@wic f1=@f1 mob=@mob af=@af kf=@kf cgbo=@cgbo
     cgdo=@cgdo cgso=@cgso cox=@cox meto=@meto tox=@tox wd=@wd capop=@capop
     cf1=@cf1 cf2=@cf2 cf3=@cf3 cf4=@cf4 cf5=@cf5 cf6=@cf6 alpha=@alpha is=@is
     js=@js jsw=@jsw vcr=@vcr cbd=@cbd cbs=@cbs cj=@cj cjsw=@cjsw mj=@mj mjsw=@mjsw
     pb=@pb php=@php ldif=@ldif rd=@rd rs=@rs rsh=@rsh trd=@trd trs=@trs
     delta=@delta kappa=@kappa eta=@eta
```

The *ndepl, nfet, nmos, nmosd, nmose, nsftn, nxfr, pdepl, pfet, pmos, pmosd, pmose, psftn,* and *pxfr* cells in the *sample* library use this format.

Index