HSPICE Basics

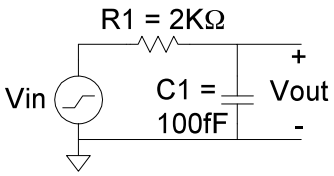
This tutorial is a quick start to the use of HSPICE.

Running a simple example simulation:

The environment for running hspice has been setup for you adding the location of the directory containing hspice executable files (/usr/synopsys/HSPICE/hspice/bin) to the PATH environment variable specified in the **`.bashrc`** file.

Create a working directory:

`mkdir simExample`



Change to the directory you just created

`cd simExample`

Using your favorite text editor (e.g. vim, gedit, emacs, nano) create the following spice input file and name it *example430.sp*:

```
* example430.sp
* CT 9/10/11
* Find the response of RC circuit to rising  input
*---------------------------------
* Parameters and models
*---------------------------------
.option post brief nomod
*---------------------------------
* Simulation Netlist
*---------------------------------
Vin in gnd PWL 0ps 0 100ps 0 150ps 1.0 1ns 1.0
R1 in out 2k
C1 out gnd 100f
*---------------------------------
* Analysis
*---------------------------------
.tran 20ps 1ns
.probe v(in) v(out)
.end
```

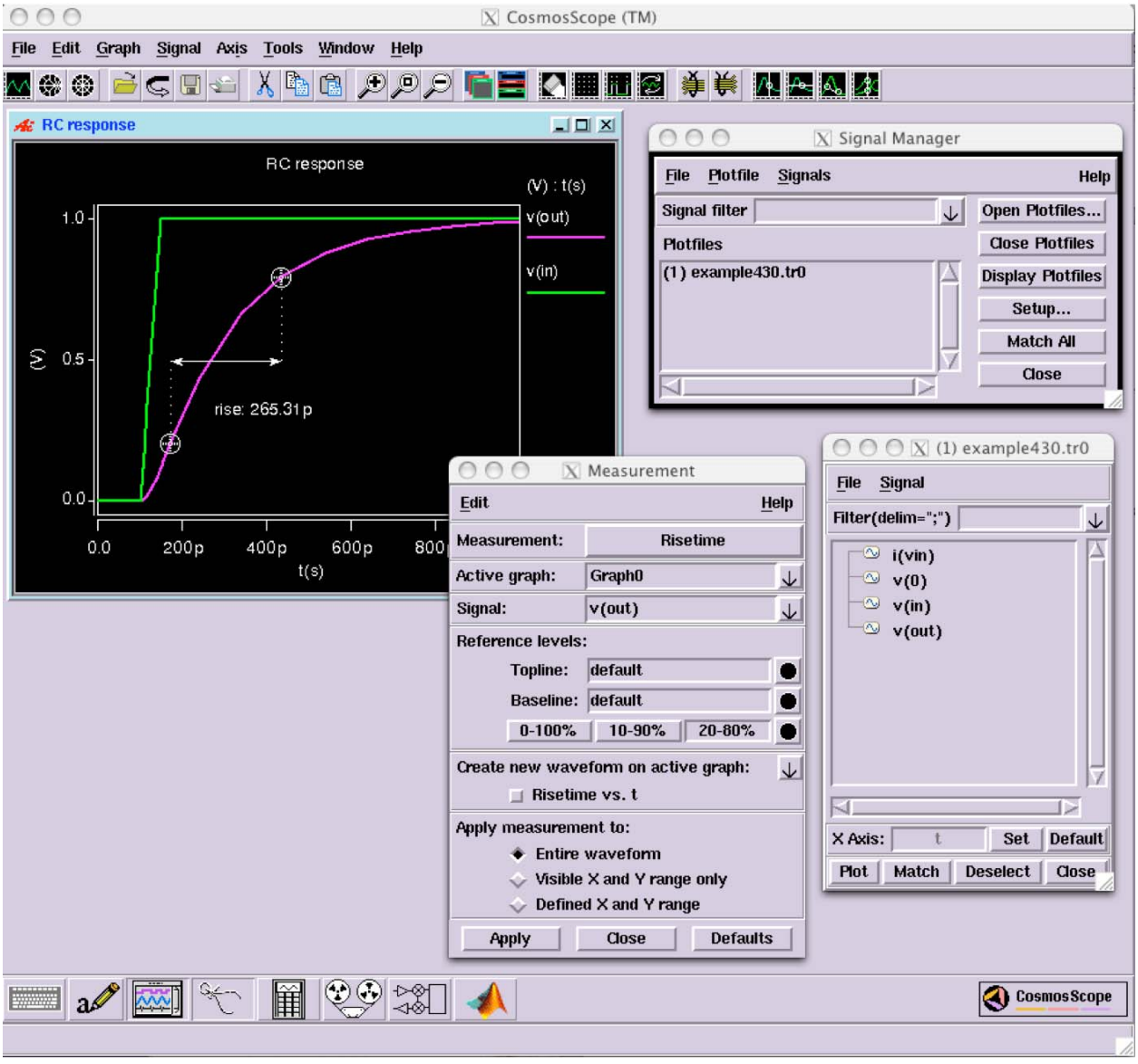
Run the HSPICE simulation

`hspice example430.sp > example430.lis`

NOTE: Make sure to always check the .lis file for Errors and Warnings.

To view the simulation results you can use either CosmosScope.

`cscope &`



or Synopsys Custom Explorer:

`sx &`

To learn more about hspice look up the official documentation:

`hspice -doc`

HSPICE Basics

SPICE (***S**imulation **P**rogram with **I**ntegrated **C**ircuits **E**mphasis*) is an analog circuit simulator (originally developed in the 1070s at Berkeley) capable of analyzing electric circuits in steady-state (DC), transient, and frequency (including small signal AC analysis) domains. There are a number of flavors of SPICE available. Synopsys HSPICE is without doubt the most popular. In particular, HSPICE is widely used in industry because of it offers excellent numerical convergence, supports the latest device and interconnect models, and has a large number of enhancements for measuring and optimizing circuits. Another popular flavor of SPICE is LTspice. LTSpice is not as robust and flexible as HSPICE, but it is available for free. Please, be aware that the various flavors of SPICE are not fully compatible among them. In this class we will use HSPICE.

The rest of this document provides basic information about HSPICE and its associated viewers (Cosmoscope and Custom Explorer), which should be sufficient for running simple simulations. For more details consult the "extensive" HSPICE Documentation Set.

The general procedure for simulating a circuit with HSPICE is to:

1. Use a text editor (vi, emacs, nano, gedit, etc.) to create an input file;
2. Run HSPICE to generate output graphs and data files
3. Run CosmosScope, Custom Explorer or MATLAB to view customize, and print the resulting graphs

All versions of SPICE read an input file and generate a list file with results, warnings, and error message. The input file is often called a SPICE netlist or a SPICE deck (this last name is due to the fact that it was once provided to a mainframe as a deck of punch cards). It is customary to save the input file with .sp extension. Once you have prepared your input file you can run the simulation issuing the following command:

hspice *filename.sp* > *filename.lis &*

The input file contains a netlist consisting of components and nodes, simulation options, analysis commands, output desired, device models and libraries to use.

NOTE: Make sure to always check the .lis file for Errors and Warnings.

HSPICE Input File

The hspice input file contains the following:

- Design netlist (with circuit elements, subcircuits and macros, power supplies, device models and so on)
- Statement naming the library to be used (optional)
- Specification of the analysis to be run (optional)
- Specification of the output desired (optional)

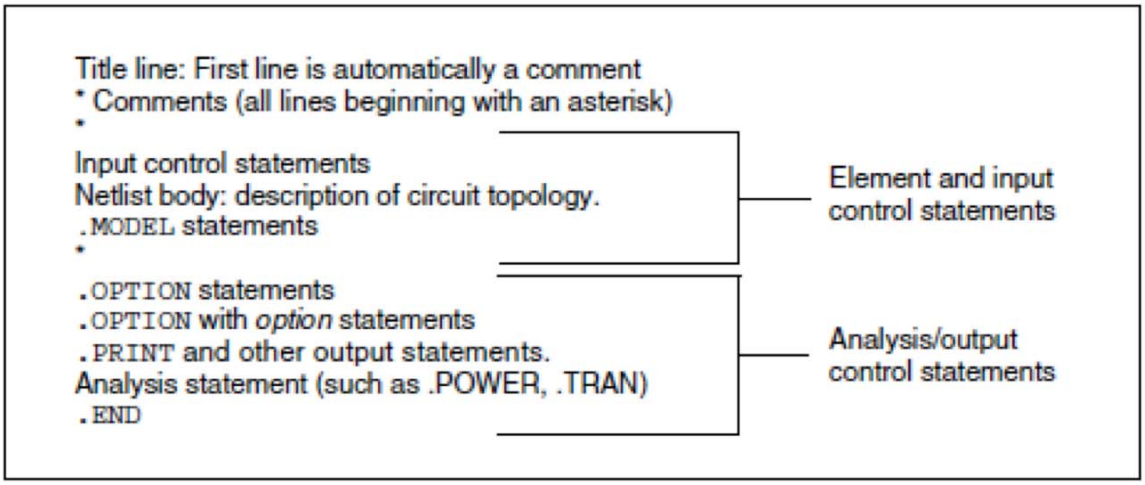


Figure 5 Basic Netlist Structure

Here is a summary of the most important formatting rules to remember:

- Statements in the input netlist file can be in any order, except the first line that is a title line, continuation lines (lines beginning with a +) must immediately follow the statement being continued, and the last .ALTER submodule must appear at the end of the file before the .END statement.
- Node 0, GND, GND!, and GROUND all refer to the global HSPICE ground.
- An input filename can be up to 1024 characters long (this is true for all platforms except PC Windows, which has a limitation of 256 characters).
- HSPICE has a limitation on the number of characters in a path name plus a file name of 1024 characters (except PC Windows, 256 characters or fewer).
- An input statement or equation is limited to 1024 characters per line.
- If the –case command-line switch is not invoked, hspice ignores differenced between upper and lower case, except in quoted filenames and or after the .INC or .LIB commands
- A statement may be continued on the next line by entering a plus (+) sign as the first nonnumeric, nonblank character in the next line.
- Each netlist line cannot exceed 1024 characters. The "+" line continuation character should be used to break up lines > 1024 characters in length. Otherwise, an error is generated.

- All statements, including quoted strings such as paths and algebraics, are continued with a backslash (\) or a double backslash (\\) at the end of the line to be continued. The single backslash preserves white space and the double backslash squeezes out any white space between the continued lines. The double backslash guarantees that path names are joined without interruption. Input lines can be 1024 characters long, so folding and continuing a line is generally only necessary to improve readability.
- Comments can be added at any place in the file. Lines beginning with an asterisk (*) are comments. Place a comment on the same line as input text by entering a dollar sign ($), preceded by one or more blanks, after the input text.

Table 3-3: Input Netlist File Statements and Elements

Statement or Element	Definition
Title	The first line is the input netlist file title.
* or $	Designates comments to describe the circuit
.OPTIONS	Sets conditions for simulation
Analysis statements and .TEMP	Statements to set sweep variables
.PRINT/.PLOT/.GRAPH/.PROBE	Statements to set print, plot and graph variables
.IC or .NODESET	Sets initial state; can also be put in subcircuits
Sources (I or V) and digital to analog inputs	Sets input stimuli
Netlist	Circuit
.LIB	Library
.INCLUDE	General include files
.PROTECT	Turns off output printback
.MODEL libraries	Element model descriptions
.UNPROTECT	Restores output printback
.DELETE LIB	Removes previous library selection
.ALTER	Sequence for in-line case analysis
.END	Required statement to terminate the simulation

Numbers

You can enter numbers as integer (e.g., 15 or -31) , floating point (3.1415), floating point with an integer exponent (31415e-4), or integer or floating point with one of the scale factors listed in the following table.

Table 13 Scale Factors

Scale Factor	Prefix	Symbol	Multiplying Factor
T	tera	T	1e+12
G	giga	G	1e+9
MEG or X	mega	M	1e+6
K	kilo	k	1e+3
MIL	n/a	none	25.4e-6
M	milli	m	1e-3

Table 13 Scale Factors (Continued)

Scale Factor	Prefix	Symbol	Multiplying Factor
U	micro	µ	1e-6
N	nano	n	1e-9
P	pico	p	1e-12
F	femto	f	1e-15
A	atto	a	1e-18

Numbers can use exponential format or engineering key letter format, but not both (1e-12 or 1p, but not 1e-6u).

To designate exponents use D or E.

Trailing alphabetic characters are interpreted as units comments.

Units comments are not checked.

HSPICE input file sections

Table 16 Input Netlist File Sections

Sections	Examples	Definition
Title	.TITLE	The first line in the netlist is the title of the input netlist file. (HSPICE)
Set-up	.OPTION .IC or .NODESET, .PARAM, .GLOBAL	Sets conditions for simulation. Initial values in circuit and subcircuit. Set parameter values in the netlist. Set node name globally in netlist.
Sources	Sources and digital inputs	Sets input stimuli (I or V element).
Netlist	Circuit elements .SUBKCT, .ENDS, or .MACRO, .EOM	Circuit for simulation. Subcircuit definitions.
Analysis	.DC, .TRAN, .AC, and so on. .SAVE and .LOAD .DATA, .TEMP	Statements to perform analyses. Save and load operating point information. (HSPICE) Create table for data-driven analysis. Set temperature analysis.
Output	.PRINT, .PROBE, .MEASURE	Statements to output variables. Statement to evaluate and report user-defined functions of a circuit.
Library, Model and File Inclusion	.INCLUDE	General include files.
	.MALIAS	Assigns an alias to a diode, BJT, JFET, or MOSFET.
	.MODEL	Element model descriptions.
	.LIB	Library.
	.OPTION SEARCH	Search path for libraries and included files. (HSPICE)

Table 16 Input Netlist File Sections (Continued)

Sections	Examples	Definition
	.OPTION BRIEF= 1 and .OPTION BRIEF= 0	Control printback to output listing. (HSPICE)
Alter blocks (HSPICE Only)	.ALIAS, .ALTER, .DEL LIB	Renames a previous model. Sequence for in-line case analysis. Removes previous library selection.
End of netlist	.END	Required statement; end of netlist.

HSPICE output files

HSPICE saves the simulation results requested in an output listing file and, if .OPTIONS POST is specified, in various data files that can be viewed using a waveform viewer like CosmosScope or Custom Explorer. All of the files associated with a particular design reside in one directory and are named by concatenating the design name and a particular suffix.

Table 2 HSPICE Output Files and Extensions

Output File Type	Extension
AC analysis measurement results	.ma#[1]
AC analysis results (from .POST statement)	.ac#
DC analysis measurement results	.ms#
DC analysis results (from .POST statement)	.sw#
Digital output	.a2d
FFT analysis graph data (from FFT statement)	.ft#
Hardcopy graph data (from meta.cfg PRTDEFAULT)	.gr#[2]
Operating point information (from .OPTION OPFILE statement)	.dp#
Operating point node voltages (initial conditions)	.ic#
Output listing	.lis, or user-specified
Output status	.st#
Output tables (from .DCMATCH OUTVAR statement)	.dm#

Table 2 HSPICE Output Files and Extensions

Output File Type	Extension
Subcircuit cross-listing	.pa#
Transient analysis measurement results	.mt#
Transient analysis results (from .POST statement)	.tr#
Waveform viewing files from .OPTION WDF argument for use with Synopsys WaveView/SX tools	*_wdf.tr#, *_wdf.sw#, or *_wdf.ac#

1. # can be either a sweep number or a hardcopy file number. For .ac#, .dp#, .dm#, .ic#, .st#, .sw#, and .tr# files, # is from 0 through 9999.

2. Requires a .GRAPH statement (obsolete), or a pointer to a file in the meta.cfg file. The Windows and Linux versions of HSPICE do not generate this file.

Defining Parameters

The .PARAM statement defines parameters. Parameters in HSPICE are names that have associated numeric values. **Note:** A .PARAM statement with no definition is illegal.

You can assign the following types of values to parameters:

- Constant real number
- Algebraic expression of real values
- Predefined function
- Function that you define
- Circuit value
- Model value

Limitation: if a parameter is defined as an expression containing output signals such as v(node) or i(element), this parameter only can be used in an element value expression directly, and can not be evaluated to another parameter.

For example, the following is correct:

.param a='2*sqrt(V(p,n))'

r1 p n '1k+a'

Although the following definition is syntactically correct, it generates an incorrect result.

.param a='2*sqrt(v(p,n))'

.param b='a+1'

r1 p n '1k+b'

It is best to use a user-defined function to replace the previous example, so that all of r1 and r2 are correct.

.param a(x)='2*sqrt(x)'

.param b(x)='a(x)+1'

r1 p n '1k+a(V(p,n))'

r2 p n '1k+b(V(p,n))'

Algebraic Expressions

In HSPICE, an algebraic expression, with single-quoted strings, can replace any parameter in the netlist.

Parameters:

.PARAM x='y+3'

Functions:

.PARAM rho(leff,weff)='2+*leff*weff-2u'

Algebra in elements:

R1 1 0 r='ABS(v(1)/i(m1))+10'

Algebra in .MEASURE statements:

.MEAS vmax MAX V(1)

.MEAS imax MAX I(q2)

.MEAS ivmax PARAM='vmax*imax'

Algebra in output statements:

.PRINT conductance=PAR('i(m1)/v(22)')

Hierarchical subcircuit algeaibric parameter passing:

.subckt inv in out wp=10u wn=5u qbar_ic=vdd

.ic qbar=qbar_ic

…

.ends

In addition to simple arithmetic operations (+, -, *, /), it is possible to use built-in functions.

Table 27 Synopsys HSPICE Built-in Functions

HSPICE Form	Function	Class	Description
sin(x)	sine	trig	Returns the sine of x (radians)
cos(x)	cosine	trig	Returns the cosine of x (radians)
tan(x)	tangent	trig	Returns the tangent of x (radians)
asin(x)	arc sine	trig	Returns the inverse sine of x (radians)
acos(x)	arc cosine	trig	Returns the inverse cosine of x (radians)
atan(x)	arc tangent	trig	Returns the inverse tangent of x (radians)
sinh(x)	hyperbolic sine	trig	Returns the hyperbolic sine of x (radians)
cosh(x)	hyperbolic cosine	trig	Returns the hyperbolic cosine of x (radians)
tanh(x)	hyperbolic tangent	trig	Returns the hyperbolic tangent of x (radians)

Table 27 Synopsys HSPICE Built-in Functions (Continued)

HSPICE Form	Function	Class	Description
abs(x)	absolute value	math	Returns the absolute value of x: |x|
sqrt(x)	square root	math	Returns the square root of the absolute value of x: sqrt(-x)=-sqrt(|x|)
pow(x,y)	absolute power	math	Returns the value of x raised to the integer part of y: $x^{(\text{Integer part of y})}$
pwr(x,y)	signed power	math	Returns the absolute value of x, raised to the y power, with the sign of x: (sign of x)$\|x\|^y$
x**y	power		If x<0, returns the value of x raised to the integer part of y. If x=0, returns 0. If x>0, returns the value of x raised to the y power.
log(x)	natural logarithm	math	Returns the natural logarithm of the absolute value of x, with the sign of x: (sign of x)log(|x|)
log10(x)	base 10 logarithm	math	Returns the base 10 logarithm of the absolute value of x, with the sign of x: (sign of x)$\log_{10}(\|x\|)$
exp(x)	exponential	math	Returns e, raised to the power x: e^x
db(x)	decibels	math	Returns the base 10 logarithm of the absolute value of x, multiplied by 20, with the sign of x: (sign of x)$20\log_{10}(\|x\|)$
int(x)	integer	math	Returns the integer portion of x. The fractional portion of the number is lost.
nint(x)	integer	math	Rounds x up or down, to the nearest integer.
sgn(x)	return sign	math	Returns -1 if x is less than 0. Returns 0 if x is equal to 0. Returns 1 if x is greater than 0

Table 27 Synopsys HSPICE Built-in Functions (Continued)

HSPICE Form	Function	Class	Description
sign(x,y)	transfer sign	math	Returns the absolute value of x, with the sign of y: (sign of y)|x|
def(x)	parameter defined	control	Returns 1 if parameter x is defined. Returns 0 if parameter x is not defined.
min(x,y)	smaller of two args	control	Returns the numeric minimum of x and y
max(x,y)	larger of two args	control	Returns the numeric maximum of x and y
val(*element*)	get value	various	Returns a parameter value for a specified element. For example, val(r1) returns the resistance value of the *r1* resistor.
val(*element. parameter*)	get value	various	Returns a value for a specified parameter of a specified element. For example, val(rload.temp) returns the value of the *temp* (temperature) parameter for the *rload* element.
val(*model_type: model_name. model_param*)	get value	various	Returns a value for a specified parameter of a specified model of a specific type. For example, val(nmos:mos1.rs) returns the value of the *rs* parameter for the *mos1* model, which is an nmos model type. Not supported for CMI models (Level 54 and greater). See Measuring the Value of MOSFET Model Card Parameters for an example and details.
valm (*elem_name. model_param*)	get value		Returns a value for a specified model parameter of a specified element. For example, valm(m1.vth0) returns the value of vth0 parameter of the model card that is used by m1. valm() is supported only by vth0, lmin, lmax, wmin, wmax, lref, wref, xl, dl, dell, xw, dw, delw, scalm, lmlt, wmlt and level54, level57 and level70. See Measuring the Value of MOSFET Model Card Parameters for an example and details.

Table 27 Synopsys HSPICE Built-in Functions (Continued)

HSPICE Form	Function	Class	Description
valp(*parameter*)	get value		Returns a value for a specified parameter. The parameter can only be a named parameter as defined in a subcircuit. For example: `.meas tran asdf param='valp(x1/zzz.p1)'` An expression is not permitted.
lv(Element) or lx(Element)	element templates	various	Returns various element values during simulation. See Element Template Output (HSPICE Only) on page 388 for more information.
v(Node), i(Element)...	circuit output variables	various	Returns various circuit values during simulation. See DC and Transient Output Variables on page 372 for more information.
cond ?x : y	ternary operator		Returns *x* if *cond* is not zero. Otherwise, returns *y*. .param z= 'condition ? x:y'
<	relational operator (less than)		Returns 1 if the left operand is less than the right operand. Otherwise, returns 0. .para x=y<z (y less than z)
<=	relational operator (less than or equal)		Returns 1 if the left operand is less than or equal to the right operand. Otherwise, returns 0. .para x=y<=z (y less than or equal to z)
>	relational operator (greater than)		Returns 1 if the left operand is greater than the right operand. Otherwise, returns 0. .para x=y>z (y greater than z)
>=	relational operator (greater than or equal)		Returns 1 if the left operand is greater than or equal to the right operand. Otherwise, returns 0. .para x=y>=z (y greater than or equal to z)
==	equality		Returns 1 if the operands are equal. Otherwise, returns 0. .para x=y==z (y equal to z)

Table 27 Synopsys HSPICE Built-in Functions (Continued)

HSPICE Form	Function	Class	Description
!=	inequality		Returns 1 if the operands are not equal. Otherwise, returns 0. .para x=y!=z (y not equal to z)
&&	Logical AND		Returns 1 if neither operand is zero. Otherwise, returns 0. .para x=y&&z (y AND z)
||	Logical OR		Returns 1 if either or both operands are not zero. Returns 0 only if both operands are zero. .para x=y||z (y OR z)

Table 28 Synopsys HSPICE Special Variables

HSPICE Form	Function	Class	Description
time	current simulation time	control	Uses parameters to define the current simulation time, during transient analysis.
temper	current circuit temperature	control	Uses parameters to define the current simulation temperature, during transient/temperature analysis. You can use the HSPICE simulation temperature in an equation by using the temper variable parameter. For example: .temp 20 50 100 .par x="temper/2" v0 1 0 1 r0 1 0 r=x

Table 28 Synopsys HSPICE Special Variables (Continued)

HSPICE Form	Function	Class	Description
hertz	current simulation frequency	control	Uses parameters to define the frequency, during AC analysis.

Summary of HSPICE Analysis

.AC	.DCSENS	.LIN	.PAT	.STATEYE
.ACMATCH	.DISTO	.LSTB	.PZ	.TEMP (or) .TEMPERATURE
.DC	.FFT	.NOISE	.SAMPLE	.TF
.DCMATCH	.FOUR	.OP	.SENS	.TRAN

Summary of output statements

Table 31 Output Statements

Output Statement	Description
.PRINT	Prints numeric analysis results in the output listing file (and post-processor data, if you specify .OPTION POST). See .PRINT.
.PROBE	Outputs data to post-processor output files, but not to the output listing (used with .OPTION PROBE, to limit output). See .PROBE.
.MEASURE	Prints the results of specific user-defined analyses (and post-processor data, if you specify .OPTION POST), to the output listing file or HSPICE RF. See .MEASURE (or) .MEAS.
.DOUT (HSPICE only)	Specifies the expected final state of an output signal. See .DOUT or Expected State of Digital Output Signal (.DOUT).

Table 31 Output Statements (Continued)

Output Statement	Description
.STIM (HSPICE only)	Specifies simulation results to transform to PWL, Data Card, or Digital Vector File format. See .STIM.

Names of Element Instances

The names of element instances begin with the element key letter (see Table), except in subcircuits where instance names begin with X. (Subcircuits are sometimes called macros or modules.)

Letter (First Char)	Element	Example Line
B	IBIS buffer	b_io_0 nd_pu0 nd_pd0 nd_out nd_in0 nd_en0 nd_outofin0 nd_pc0 nd_gc0
C	Capacitor	Cbypass 1 0 10pf
D	Diode	D7 3 9 D1
E	Voltage-controlled voltage source	Ea 1 2 3 4 K
F	Current-controlled current source	Fsub n1 n2 vin 2.0
G	Voltage-controlled current source	G12 4 0 3 0 10
H	Current-controlled voltage source	H3 4 5 Vout 2.0
I	Current source	I A 2 6 1e-6
J	JFET or MESFET	J1 7 2 3 GAASFET
K	Linear mutual inductor (general form)	K1 L1 L2 1
L	Linear inductor	LX a b 1e-9
M	MOS transistor	M834 1 2 3 4 N1
P	Port	`P1 in gnd port=1 z0=50`
Q	Bipolar transistor	Q5 3 6 7 8 pnp1
R	Resistor	R10 21 10 1000
S	S parameter element	S1 nd1 nd2 s_model2
V	Voltage source	V1 8 0 5
T,U,W	Transmission Line	W1 in1 0 out1 0 N=1 L=1

Letter (First Char)	Element	Example Line
X	Subcircuit call	X1 2 4 17 31 MULTI WN=100 LN=5

Common Sources and Stimuli

Independent Source Elements

Use independent source element statements to specify DC, AC, transient, and mixed independent voltage and current sources. Depending on the analysis performed, the associated analysis sources are used. The value of the DC source is overridden by the zero time value of the transient source when a transient operating point is calculated

```
Vxxx n+ n- [[DC=] dcval tranfun [AC=acmag acphase]]
Ixxx n+ n- [[DC=] dcval tranfun [AC=acmag acphase]] + [M=val]
```

Parameter	Description
Vxxx	Independent voltage source element name. Must begin with V, followed by up to 1023 alphanumeric characters.
Ixxx	Independent current source element name. Must begin with I, followed by up to 1023 alphanumeric characters.

Parameter	Description
n+	Positive node.
n-	Negative node.
DC=dcval	DC source keyword and value in volts. Used for the operating point calculation for all simulations except transient. For transient analysis, an additional operating point is calculated with the tranfun value at time zero. Default=0.0.
tranfun	Transient source function (one or more of: AM, DC, EXP, PAT, PE, PL, PU, PULSE, PWL, SFFM, SIN). The functions specify the characteristics of a time-varying source. See the individual functions for syntax.
AC	AC source keyword for use in AC small-signal analysis.
acmag	Magnitude (RMS) of the AC source, in volts.
acphase	Phase of the AC source, in degrees. Default=0.0.
M	Multiplier, to simulate multiple parallel current sources. HSPICE or HSPICE RF multiplies source current by M. Default=1.0.

DC Sources

For a DC source, you can specify the DC current or voltage in different ways:

V1 1 0 DC=5V

V1 1 0 5V

I1 1 0 DC=5mA

I1 1 0 5mA

AC Sources

AC current and voltage sources are impulse functions, used for an AC analysis. To specify the magnitude and phase of the impulse, use the AC keyword.

V1 1 0 AC=10V,90

VIN 1 0 AC 10V 90

Common Transient Source Functions

- Trapezoidal pulse (PULSE function)
- Sinusoidal (SIN function)
- Exponential (EXP function)
- Piecewise linear (PWL function)

Trapezoidal Pulse Function

```
Vxxx n+ n- PU[LSE] [()]v1 v2 [td [tr [tf [pw [per]]]]] [)]
+ [PERJITTER=val [SEED=val]]

Ixxx n+ n- PU[LSE] [()]v1 v2 [td [tr [tf [pw [per]]]]] [)]
+ [PERJITTER=val [SEED=val]]
```

Parameter	Description
Vxxx, Ixxx	Independent voltage source, which exhibits the pulse response.
PULSE	Keyword for a pulsed time-varying source. The short form is PU.
v1	Initial value of voltage or current before the pulse onset (units: volts/amps).
v2	Pulse plateau value (units of volts or amps).
td	Delay (propagation) time in seconds from the beginning of the transient interval to the first onset ramp. Default=0.0
tr	Duration of the onset ramp (in seconds) from the initial value to the pulse plateau value (reverse transit time). Default=TSTEP.
tf	Duration of the recovery ramp (in seconds) from the pulse plateau back to the initial value (forward transit time). Default=TSTEP.
pw	Pulse width (the width of the plateau portion of the pulse), in seconds.Default=TSTOP.
per	Pulse repetition period, in seconds. Default=TSTOP.
perjitter	RMS value for period jitter, adjusts the magnitude of the random time.
seed	Used to generate random number sequences with different seed value. The value is a negative integer, defaults to -1.

Table 19 Time-Value Relationship for a PULSE Source

Time	Value
0	v1
td	v1
td + tr	v2
td + tr + pw	v2
td + tr + pw + tf	v1
tstop	v1

Sinusoidal Source Function

HSPICE provides a damped sinusoidal source function, which is the product of a dying exponential with a sine wave. To apply this waveform, you must specify:

- Sine wave frequency
- Exponential decay constant
- Beginning phase
- Beginning time of the waveform

```
Vxxx n+ n- SIN [() vo va [freq [td [q [j]]]] ())]
+ [[PERJITTER=val] [SEED=val]]

Ixxx n+ n- SIN [() vo va [freq [td [q [j]]]] ())]
+ [[PERJITTER=val] [SEED=val]]
```

Parameter	Description
Vxxx, Ixxx	Independent voltage source that exhibits the sinusoidal response.
SIN	Keyword for a sinusoidal time-varying source.
vo	Voltage or current offset, in volts or amps.
va	Voltage or current peak value (vpeak), in volts or amps.
freq	Source frequency in Hz. Default=1/TSTOP.
td	Time (propagation) delay before beginning the sinusoidal variation, in seconds. Default=0.0. Response is 0 volts or amps, until HSPICE or HSPICE RF reaches the delay value, even with a non-zero DC voltage.
q	Damping factor, in units of 1/seconds. Default=0.0.
j	Phase delay, in units of degrees. Default=0.0.
perjitter	RMS value for period jitter, used to adjust the magnitude of the random time.
seed	Used to generate random number sequences with different seed value. The value is a negative integer, defaults to -1.

The following table of expressions defines the waveform shape:

Table 20 Waveform Shape Expressions

Time	Value
0 to td	$vo + va \cdot SIN\left(\frac{2 \cdot \Pi \cdot \varphi}{360}\right)$
td to tstop	$vo + va \cdot Exp[-(Time - td) \Rightarrow q] \cdot SIN\left\{2 \cdot \Pi \cdot \left[freq \cdot (time - td + x(t)) + \frac{j}{360}\right]\right\}$
	Where q and j are the damping factor and phase delay in the syntax.

In these expressions, TSTOP is the final time.

Example

```
*file: sin.sp sinusoidal source
.options post
.param v0=0 va=1 freq=100meg delay=2n theta=5e7 phase=0
v 1 0 sin(v0 va freq delay theta phase)
r 1 0 1
.tran .05n 50n
.end
```

This damped sinusoidal source connects between nodes 1 and 0. In this waveform:

- Peak value is 1 V.
- Offset is 0 V.
- Frequency is 100 MHz.
- Time delay is 2 ns.
- Damping factor is 5e7 (50 MHz).
- Phase delay is zero degrees.

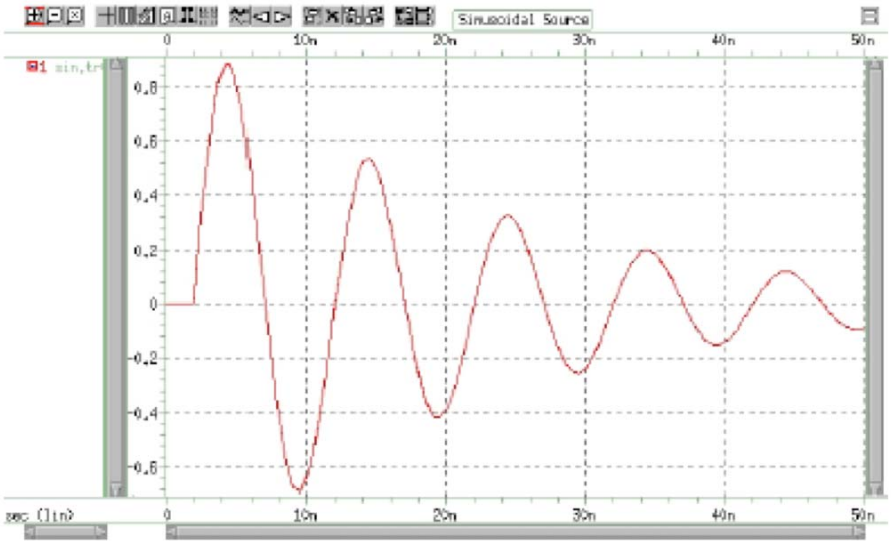


Figure 23 Sinusoidal Source Function

Exponential Source Function

```
Vxxx n + n- EXP [() v1 v2 [td1 [t1 [td2 [t2]]]] ())]
Ixxx n+ n- EXP [() v1 v2 [td1 [t1 [td2 [t2]]]] ())]
```

Parameter	Description
Vxxx, Ixxx	Independent voltage source, exhibiting an exponential response.
EXP	Keyword for an exponential time-varying source.
v1	Initial value of voltage or current, in volts or amps.
v2	Pulsed value of voltage or current, in volts or amps.
td1	Rise delay time, in seconds. Default=0.0.
td2	Fall delay time, in seconds. Default=td1+TSTEP.
t1	Rise time constant, in seconds. Default=TSTEP.
t2	Fall time constant, in seconds. Default=TSTEP.

TSTEP is the printing increment, and TSTOP is the final time.

The following table of expressions defines the waveform shape:

Table 22 Waveform Shape Definitions

Time	Value
0 to td1	v1
td1 to td2	$v1 + (v2 - v1) \cdot \left[1 - exp\left(-\frac{Time - td1}{\tau_1}\right)\right]$
td2 to tstop	$v1 + (v2 - v1) \cdot \left[1 - exp\left(-\frac{(Time - td1)}{\tau_1}\right)\right] + (v1 - v2) \cdot \left[1 - \exp\left(-\frac{(Time - td2)}{\tau_2}\right)\right]$

Example

```
*file: exp.sp exponential independant source
.options post
.param v0=-4 va=-1 td1=5n tau1=30n tau2=40n td2=80n
V 1 0 exp(v0 va td1 tau1 td2 tau2)
R 1 0 1
.tran .05n 200n
.end
```

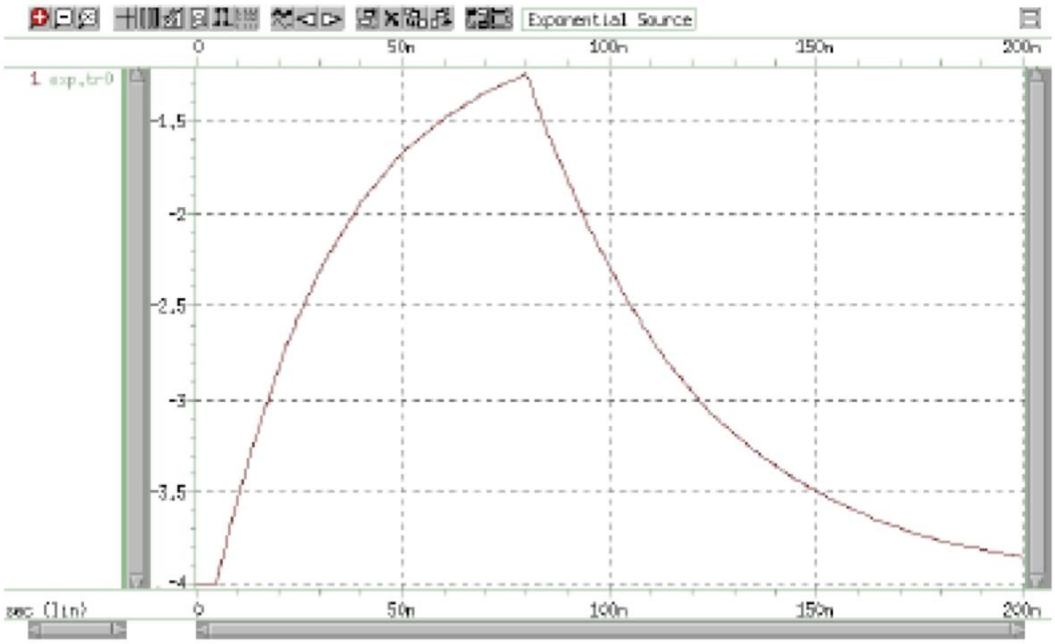


Figure 24 Exponential Source Function

The above example describes an exponential transient source, which connects between nodes 1 and 0. In this source:

- Initial t=0 voltage is -4 V.
- Final voltage is -1 V.
- Waveform rises exponentially from -4 V to -1 V with a time constant of 30 ns.
- At 80 ns, the waveform starts dropping to -4 V again, with a time constant of 40 ns.

General PWL Function

```
Vxxx n+ n- PWL [() v1 t1 [v2 t2 v3 t3…] [R= [repeat]] + [TD=delay] ())]
Ixxx n+ n- PWL [() t1 v1 [t2 v2 t3 v3…] [R= [repeat]] + [TD=delay] ())]
```

Parameter	Description
Vxxx, Ixxx	Independent voltage source; uses a piecewise linear response.
PWL	Keyword for a piecewise linear time-varying source.
v1 v2 ... vn	Current or voltage values at the corresponding timepoint.
t1 t2 ... tn	Timepoint values, where the corresponding current or voltage value is valid.

Parameter	Description
R=repeat	Keyword and time value to specify a repeating function. With no argument, the source repeats from the beginning of the function. *repeat* is the time, in units of seconds, which specifies the start point of the waveform to repeat. This time needs to be less than the greatest time point, *tn*.
TD=delay	Time, in units of seconds, which specifies the length of time to delay (propagation delay) the piecewise linear function.

Each pair of values (t1, v1) specifies that the value of the source is v1 (in volts or amps), at time t1. Linear interpolation between the time points determines the value of the source, at intermediate values of time.

Mixed Sources

Mixed sources specify source values for more than one type of analysis. For example, you can specify a DC source, an AC source, and a transient source, all of which connect to the same nodes. In this case, when you run specific analyses, HSPICE selects the appropriate DC, AC, or transient source. If the mixed source DC value is missing, the zero-time value of its transient source will be used as the DC value by default. Otherwise, for DC analysis, the DC source value is used by the program, and is selected for operating point calculation for all analysis except TRAN; for TRAN analysis, an additional operating point is calculation with the zero-time source transient value.

Example

VIN 13 2 0.5 AC 1 SIN (0 1 1MEG)

Where,

- DC source of 0.5 V
- AC source of 1 V
- Transient damped sinusoidal source
- Each source connects between nodes 13 and 2.

For DC analysis, the program uses its dc value 0.5v, and this operating point is selected for the coming AC analysis. For transient analysis, another operating point is calculated by using zero source value because the sinusoidal source is zero at time zero.

Frequently used Commands

For more detailed and complete information on HSPICE commands please consult:

Synopsys, *HSPICE® Reference Manual: Commands and Control Options*, Version E-2010.12, December 2010, pp.19-24.

.AC type np fstart fstop

Performs several types of AC analyses. For AC analysis, the input file must include at least one independent AC source element command (for example: VI INPUT GND AC 1V)

type Any of the following keywords:

- DEC – decade variation.
- OCT – octave variation.
- LIN – linear variation.
- POI – list of points.

np Number of points or points per decade or octave, depending on which keyword precedes it.

fstart Starting frequency. If you use POI (list of points) type variation, use a list of frequency values, not fstart fstop.

fstop Final frequency

Example

```
.AC DEC 10 1K 100MEG
```

This example performs a frequency sweep by 10 points per decade from 1kHz to 100MHz.

.ALTER *title_string*

Reruns an HSPICE simulation with selected changes. *title_string* is any string up to 80 characters. A .END or another .ALTER terminates the ALTER sequence

.DC *var1 start1 stop1 incr1* [var2 *start2 stop2 incr2*]

Performs several types of sweeps during DC analysis.

Example 1

Sweeps the value of the VIN voltage source from 0.25 volts to 5.0 volts in increments of 0.25 volts.

```
.DC VIN 0.25 5.0 0.25
```

Example 2

Sweeps the drain-to-source voltage from 0 to 10 v in 0.5 v increments atVGS values of 0, 1, 2, 3, 4, and 5 v.

```
.DC VDS 0 10 0.5 VGS 0 5 1
```

Example 3

Starts a DC analysis of the circuit from -55° C to 125° C in 10° Cincrements.

```
.DC TEMP -55 125 10
```

.ENDS subckt_name

Ends a subcircuit definition (.SUBCKT) in an HSPICE input netlist file.

.GLOBAL *node1 node2 node3* ...

node1, node2... Name of a global nodes, such as supply and clock names; overrides local subcircuit definitions.

.IC V(node1)=val1 V(node2)=val2 ... [subckt=sub_name]

val1 ... Specifies voltages. The significance of these voltages depends onwhether you specify the UIC parameter in the .TRAN command.

node1 ... Node numbers or names can include full paths or circuit numbers.

subckt=sub_name Initial condition is set to the specified node name(s) within all instances of the specified subcircuit name. This subckt setting is equivalent to placing the .IC statement within the subcircuit definition.

Use the .IC command command to set transient initial conditions in HSPICE. How it initializes depends on whether the .TRAN analysis command includes the UIC parameter. This command is less preferred compared to using the .NODESET command in many cases. If you do not specify the UIC parameter in the .TRAN command, HSPICE computes the DC operating point solution before the transient analysis. The node voltages that you specify in the .IC command are fixed to determine the DC operating point. They are used only in the first iteration to set an initial guess for the DC operating point analysis. The .IC command is equivalent to specifying the IC parameter on each element command, but is more convenient. The DC operating point for each .IC'd node should be very close to the voltage specified in the .IC command. When using the .IC command, “forcing

circuits" (i.e. fixed equivalent voltage sources) are connected to the .IC nodes for the duration of DC convergence. Forcing a node voltage applies a fixed equivalent voltage source during DC analysis. Transient analysis removes the voltage sources to calculate the second and later time points. Note that forcing a node value of the dc operating point may not satisfy KVL and KCL. If you specify the UIC parameter in the .TRAN command, HSPICE does not calculate the initial DC operating point, but directly enters transient analysis. When you use .TRAN UIC, the .TRAN node values (at time zero) are determined by searching for the first value found in this order: from .IC value, then IC parameter on an element command, then .NODESET value, otherwise use a voltage of zero.

Example 1

```
.IC V(11)=5 V(4)=-5 V(2)=2.2
```

Example 2

All settings in this statement are applied to subckt my_ff.

```
.IC V(in)=0.9 subckt=my_ff
```

.INCLUDE '*file_pathfile_name*'

Use this command (.INCLUDE (or) .INC (or) .INCL) to include another netlist in the current netlist. You can include a netlist as a subcircuit in one or more other netlists. You must enclose the file path and file name in single or double quotation marks.

Examples

```
.INCLUDE '/myhome/subcircuits/diode_circuit'
```

.LIB

Creates and reads libraries of commonly used commands, device models, subcircuit analyses, and commands.

Use the following syntax for library calls:

.LIB '[file_path] file_name' entry_name

Use the following syntax to define library files:

.LIB entry_name1

$ ANY VALID SET OF HSPICE STATEMENTS

…

.ENDL entry_name1

.LIB entry_name2.

$ ANY VALID SET OF HSPICE STATEMENTS

…

.ENDL entry_name2

.LIB entry_name3

$ ANY VALID SET OF HSPICE STATEMENTS

…

.ENDL entry_name3

Example 1

* Library call

.LIB '../models/ibm065/opconditions.lib' TT

Example 2

.LIB TT

$ Any valid set of HSPICE commands

…

.ENDL TT

.MACRO subckt_name n1 [n2 n3…] [parnam=val]

Use this command to define a subcircuit in your netlist (this is effectively the same as the .SUBCKT command). You can create a subcircuit description for a commonly used circuit and include one or more references to the subcircuit in your netlist. Use the .EOM command to terminate a .MACRO command.

subckt_nam reference name for the subcircuit model call.

n1 … Node numbers for external reference; cannot be the ground node (zero). Any element nodes that are in the subcircuit, but are not in this list strictly local with three exceptions:

- Ground node (zero).
- Nodes assigned using BULK=node in MOSFET or BJT models.
- Nodes assigned using the .GLOBAL command.

parnam Parameter name set to a value. Use only in the subcircuit. To override this value, assign it in the subcircuit call or set a value in a .PARAM command.

.SUBCKT subnam n1 n2 n3 ... [param=val]

Use this command to define a subcircuit in your netlist. You can create a subcircuit description for a commonly used circuit and include one or more references to the subcircuit in your netlist.

Argument	Description
subnam	Reference name for the subcircuit model call.
n1...	Node numbers for external reference; cannot be the ground node (0, gnd, ground, gnd!). Any element nodes that are in the subcircuit, but are not in this list are strictly local with three exceptions: ▪ Ground node (0, gnd, ground, gnd!). ▪ Nodes assigned using BULK=node in MOSFET or BJT models. ▪ Nodes assigned using the .GLOBAL command.
parnam	Parameter name set to a value. Use only in the subcircuit. To override this value, assign it in the subcircuit call or set a value in a .PARAM command.

Example 1

Defining two subcircuits: SUB1 and SUB2. These are resistor-divider networks, whose resistance values are parameters (variables). The X1, X2, and X3 commands call these subcircuits. Because the resistor values are different in each call, these three calls produce different subcircuits.

```
*FILE SUB.SP TEST OF SUBCIRCUITS
.OPTION LIST ACCT
V1 1 0 1
.PARAM P5=5 P2=10
.SUBCKT SUB1 1 2 P4=4
R1 1 0 P4
R2 2 0 P5
X1 1 2 SUB2 P6=7
X2 1 2 SUB2
.ENDS
*
.MACRO SUB2 1 2 P6=11
R1 1 2 P6
R2 2 0 P2
.EOM
```

X1 1 2 SUB1 P4=6

X2 3 4 SUB1 P6=15

X3 3 4 SUB2

…

.END

.MEASURE

The .MEASURE command prints user-defined electrical specifications of a circuit. Optimization uses .MEASURE commands extensively. You can shorten the command name to .MEAS. The specifications include:

- Propagation
- Delay
- Rise time
- Fall time
- Peak-to-peak voltage
- Minimum and maximum voltage over a specified period
- Other user-defined variables

You can use .MEASURE with either the error function (ERRfun) or GOAL parameter to optimize circuit component values, and to curve-fit measured data to model parameters. The .MEASURE command can use several different formats, depending on the application.

.MODEL *mname* type [level=*num*]
+ [*pname1=val1 pname2=val2* …]

Includes an instance of a predefined HSPICE model in an input netlist. See specific element type for supported model parameter information.

mname Model name reference. Elements must use this name to refer to the model

type Model type. Must be one of the following:

- AMP—operational amplifier model
- C—capacitor model
- CORE—magnetic core model
- D—diode model

- L—inductor model or magnetic core mutual inductor model
- NJF—n-channel JFET model
- NMOS—n-channel MOSFET model
- NPN—npn BJT model
- OPT—optimization model
- PJF—p-channel JFET model
- PLOTQ—plot model for the .GRAPH command (obsolete)
- PMOS—p-channel MOSFET model
- PNP—pnp BJT model
- R—resistor model
- U—lossy transmission line model (lumped)
- W—lossy transmission line model
- S—S-parameter

level Model level

pname1 ... Parameter name. Assign a model parameter name (pname1) from the parameter names for the appropriate model type. Each model section provides default values. For legibility, enclose the parameter assignment list in parentheses and use either blanks or commas to separate each assignment. Use a plus sign (+) to start a continuation line.

.OP

When you include an .OP statement in an input file, HSPICE calculates the DC operating point of the circuit. You can also use the .OP statement to produce an operating point, during a transient analysis. You can include only one .OP command in a simulation

.OPTION opt1 [opt2 opt3 ...]

.OPTION modifies various aspects of an HSPICE simulation. See the following list of common option commands:

.OPTION ACCURATE

Tighten integration tolerances to obtain more accurate results. This is useful for oscillators and high-gain analog circuits or when results seem fishy.

.OPTION AUTOSTOP (or) .OPTION AUTOST

Conclude simulation when all .measure results are obtained rather than continuing for the full duration of the .TRAN statement. This can substantially reduce simulation time.

.OPTION BRIEF

Stops echoing (printback) of data file to stdout

.OPTION INGOLD

Controls whether HSPICE prints *.lis file output in exponential form or engineering notation

.OPTION NOMOD

Suppresses the printout of model parameters

.OPTION POST

Saves simulation results for viewing by an interactive waveform viewer.

.OPTION PROBE

Limits post-analysis output to only variables specified in .PROBE and .PRINT commands

.OPTION SCALE

Sets the element scaling factor for HSPICE. The Syntax is .OPTION SCALE=x

Use this option to scale geometric element instance parameters whose default unit is meters. In HSPICE, the possible geometrical instance parameters include width, length, or area for both passive and active devices

.PARAM (or) .PARAMETER (or) .PARAMETERS

Use this command to define parameters. Parameters in HSPICE are names that have associated numeric values

Examples:

.PARAM Pi ='355/113'

.PARAM Pi2 ='2*Pi'

.PARAM npRatio =2.1

.PARAM nWidth =3u

.PARAM pWidth ='nWidth * npRatio'

.PARAM x=cos(2)+sin(2)

.PRINT DC v(3) gain=PAR('v(3)/v(2)')

+ PAR('V(4)/V(2)')

.PARAM MyFunc(x, y)='Sqrt((x*x)+(y*y))'

.PRINT antype ov1 [ov2 …]

Use this command to print the values of specified output variables. You can include wildcards in .PRINT commands.

antype Type of analysis for outputs. Can be one of the following types: DC, AC, TRAN, NOISE, or DISTO.

ov1 … Output variables to print. These are voltage, current, or element template variables from a DC, AC, TRAN, NOISE, or DISTO analysis

For a description of the output variables format see Ch. 11 of Synopsys, HSPICE User Guide: Simulation and Analysis, Version E-2010.12

.PROBE analysis_type ov1 [ov2 …]

.PROBE analysis_type v(inst_name.subckt_port_name)

Use this command to save output variables and print to interface and graph data files. Parameters can be node voltages, currents, elements, reasonable expressions, and node probe instances and ports. You can include wildcards in .PROBE commands.

Example1

Saves several node voltages and an expression.

.PROBE DC V(4) V(5) V(1) beta=PAR(`I1(Q1)/I2(Q1)')

Example 2

This syntax probes the voltage of the net connected with the Gate of XINST1.MN0.

.PROBE TRAN V2(XINST1.MN0)

Example 3

Illustrates saving derivative and integrative functions.

* Derivative function

.PROBE der=deriv('v(NodeX)')

* Integrate function

.PROBE int=integ('v(NodeX)')

.TEMP t1 [t2 t3 ...]

Use this command (.TEMP (or) .TEMPERATURE) to specify the circuit temperature for an HSPICE simulation.

.TF *ov srcnam*

ov small-signal output variable.

srcnam small-signal input source.

Use this command to calculate DC small-signal values for transfer functions (ratio of output variable to input source). The .TF command defines small-signal output and input for DC small-signal analysis.

When you use this command, HSPICE computes:

- DC small-signal value of the transfer function (output/input)
- Input resistance
- Output resistance

Examples

.TF V(5,3) VIN

.TF I(VLOAD) VIN

.TRAN tstep1 tstop1 [START=val] [UIC]

.TRAN tstep1 tstop1 [tstep2 tstop2]

+ [START=val] [UIC] [SWEEP var type np pstart pstop]

Starts a transient analysis that simulates a circuit at a specific time. The START value is the time when printing or plotting begins. Caution: If you use .TRAN with a .MEASURE command, a non-zero START time can cause incorrect .MEASURE results. Do not use non-zero START times in .TRAN commands when you also use .MEASURE

var Name of an independent voltage or current source, any element or model parameter, or the TEMP keyword (indicating a temperature sweep).

type Any of the following keywords:

- DEC - decade variation.
- OCT - octave variation (the value of the designated variable is eight times its previous value).
- LIN - linear variation.
- POI - list of points.

np Number of points or number of points per decade or octave, depending on what keyword precedes it.

pstart Starting voltage, current, or temperature; or any element or model parameter value. If you set the type variation to POI (list of points), use a list of parameter values, instead of pstart pstop.

pstop Final value: voltage, current, temperature; element or model param.

Example 1

Performs and prints the transient analysis every 1 ns for 100 ns.

.TRAN 1NS 100NS

Example 2

Performs the calculation every 0.1 ns for the first 25 ns; and then every 1 ns until 40 ns. Printing and plotting begin at 10 ns.

.TRAN .1NS 25NS 1NS 40NS START=10NS

Example 3

Does the calculation every 0.1 ns for 25 ns; and then every 1 ns for 40 ns; and then every 2 ns until 100 ns. Printing and plotting begin at 10 ns.

.TRAN .1NS 25NS 1NS 40NS 2NS 100NS START = 10NS

Example 4

This example performs the calculation every 10 ns for 1 μ s. This example bypasses the initial DC operating point calculation. It uses the nodal voltages specified in the .IC command (or by IC parameters in element commands) to calculate the initial conditions.

.TRAN 10NS 1US UIC

Example 5

This example increases the temperature by 10 degrees C through the range -55 degrees C to 75 degrees C. It also performs transient analysis for each temperature.

.TRAN 10NS 1US UIC SWEEP TEMP -55 75 10

Example 6

Analyzes each load parameter value at 1 pF, 5 pF, and 10 pF.

.TRAN 10NS 1US SWEEP load POI 3 1pf 5pf 10pf

Accuracy of HSPICE Simulations

One issue that can make a difference in your simulations is the accuracy that HSPICE uses in its internal numerical algorithms. By controlling the accuracy of numerical calculations, HSPICE results will be more trustworthy. If you specify relative and absolute accuracy for the circuit solution, then the simulator iteration algorithm will attempt to converge onto a solution that is within these set tolerances.

A few things you can do to make calculations more accurate:

- In the netlist, include a line:

  `.option accurate`

  This will tighten various parameters that control the accuracy of calculations.
- In the netlist, include a line:

  `.option delmax=tau`

  This sets the minimum time step used in transient analysis. If you are simulating a clocked circuit, a reasonable value for tau is about 1/100 of the clock period.
- Specify a time step in the .TRAN analysis statement.

  Note, however, that this time step will not necessarily be the time step used for calculations internally, so setting "delmax" as well is not at all redundant.
- Tighten specific accuracy parameters. The following table gives some control options and their default values:

Table. Absolute and Relative Accuracy Tolerances

Type	***Option***	***Default***
Nodal Voltage Tolerances	ABSVDC	50μV
	RELVDC	0.001
Current Element Tolerances	ABSI	1 nA
	RELI	0.01
	ABSMOS	1 μA
	RELMOS	0.05

For more information on controlling accuracy parameters, please refer to the HSPICE documentation set.

References

CAD Basics: EE114 and EE214, Stanford University, Department of Electrical Engineering, 2009

N.H.E. Weste and D.M. Harris*, CMOS VLSI Design. A Circuits and Systems Perspective*, 4/e, Addison-Wesley, 2011

HSPICE User Guide: Simulation and Analysis, version E-2010.12, Synopsys

HSPICE Reference Manual: Commands and Control Options, version E-2010.12, Synopsys

HSPICE Reference Manual: Elements and Device Models, version E-2010.12, Synopsys

HSPICE Reference Manual: MOSFET Models, version E-2010.12, Synopsys

CosmosScope User Guide, version E-2011.03, Synopsys

CosmosScope Reference Manual, version E-2011.03, Synopsys

CosmosScope Tools Reference, version E-2011.03

CustomExplorer Ultra and CustomExplorer User Guide, E-2011.03, Synposys

Custom WaveView User Guide, E-2011.03, Synopsys

Custom WaveView Quick Reference, E-2011.03, Synopsys

Analysis Command Environment (ACE) Reference Manual, E-2011.03, Synopsys

R. T. Howe and C. G. Sodini, *Microelectronics. An Integrated Approach*, Prentice Hall, 1997, pp.193-246