Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.012
Electronic Devices and Circuits
Fall 1995

Introduction to Circuit Simulation for 6.012

[Written by Rod Hinman, Fall 1993. Revised by Tracy Adams and Yakov Royter, Spring 1994, and Siva Narendra, Fall 1995]

HSPICE is an analysis tool to study circuits ranging from diode protection networks to bipolar transistor amplifiers to logic gates. HSPICE is a fairly straightforward simulation tool, provided you take the time to read the information presented in this package. The purpose of this handout is to introduce you to the programs you will be using.

In this handout you will find the following: an explanation of the basics of SPICE, a circuit simulation program developed at UC Berkeley in the early 1970s; example simulation files and an explanation of how to run them; a description of some of the enhancements of HSPICE, a commercial version of SPICE which we will be using; and information using HSPICE on ATHENA.

After you have read through the information here, go through the example circuit making sure that you understand how each circuit component is represented in the SPICE model. Simulate it yourself to get practice creating input files and using GSI, the graphical output package. If you have problems, be sure to see one of the TAs.

1 Generic SPICE

The input files to HSPICE can be created using EMACS (or VI). If you are not familiar with one of these editors, be sure to attend one of the ATHENA minicourses offered and/or get a hold of a manual for EMACS (or VI). SPICE was originally created when punchcards were the primary method of computer input. Each circuit element was described on a separate card, or sometimes on several adjacent cards. Because of this, individual lines in the input files are sometimes referred to as “cards,” and the whole file as the “SPICE deck.”

Each input file to HSPICE begins with a title line. The first line of the file is taken as the title whether you intended it to be or not! If a title is not given, the first line specifying your circuit will be assumed as the title and your simulation results will be erroneous (or more likely, the simulation won’t run).

Each input file should end with an `.END` statement. This is needed to tell HSPICE that the input file has ended. Without the `.END` statement, the simulation will not run. Be sure that there is a newline (or return) after that last `.END`, some SPICE programs need to have the final newline.

After giving a title line, the ordering of the remaining information needed within the file is fairly arbitrary (except for the `.END` statement). There is no convention for listing circuit components in any particular order within the file. One possible convention for listing information within a SPICE input file is shown in the example file included in this package. Note that SPICE is not case sensitive, so `.end` and `.END` are the same thing.

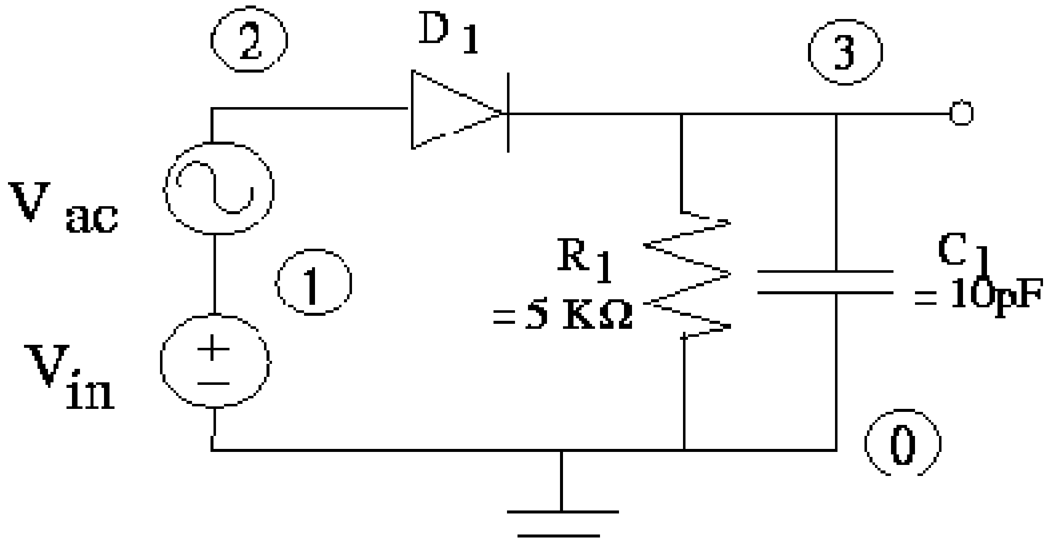

Table 1 Example Circuit

Before creating an input file for HSPICE, you should label each node of the circuit that you wish to simulate with a different number (see the example circuit above). The numbers given to each node will serve to specify the topology of the circuit by defining interconnections between different components. There are no set rules for how to define each node; HOWEVER, be sure that two separate nodes are not labeled with the same number as this will produce erroneous results in your simulation. The general convention is to label the ground node as NODE 0. After the nodes in the circuit have been labeled, the location of different circuit components can be specified by giving the two (or three, etc.) nodes which a particular component is attached to. For example, a resistor is specified by

```
Rxxxx N1 N2 Value
```

The `xxxx` can be any alphanumeric string. The `N1` and `N2` are the two nodes at which the resistor is connected and `Value` specifies the size of the resistance. For example (from the circuit above), R1 would be entered as:

```
R1  3  0   5K
```

The `R1` specifies that the component is a resistor, the `3 0` specifies that the resistor is connected between node 3 and node 0. and `5K` specifies that the resistor value is 5kΩ. Capacitors are entered in a similar way, their name must start with `C`.

Component values and parameters may be specified in three formats: a normal number, exponential notation, or with a scale factor. An example of exponential notation is 1.5e3 which is equivalent to 1500. You may also use a character following the number to represent multiplication by a power of ten. Thus 1.5k is also equivalent to 1500. The following table shows the different scale factors. Note that except for the combination `meg`, other characters after the multiplier character are ignored. Thus `5ns` is 5δ7 10^{-9} but does not explicitly represent seconds.

Abbrev.	Multiplier	Abbrev.	Multiplier
T	1e12	M	1e-3
G	1e9	U	1e-6
MEG	1e6	N	1e-9
K	1e3	P	1e-12
		F	1e-15

Some components used in a circuit require a `.model` specification (such as a diode or transistor). The `.model` statement sets the model parameters for a given device. As an example, the `.model` specification for the diode used in the circuit above is

```
.model diode1 D level=1 IS=1e-14 CJO=1e-12 M=.5 TT=3e-11 VJ=.783
+ VB=30
```

The `diode1` is the name for the model, while `D` specifies the device type. Other device types you will likely encounter include nmos, pmos, npn, and pnp transistors.

These specifications are followed by the device parameters, in any order. `level=1` specifies the particular equations to use when modeling the diode. Level 1 is the simplest, and corresponds closely to the equations we are using in class. There are many other models to take second order effects into account. `IS` (amps) is the reverse saturation current of the diodes, `CJO` (F) is the depletion capacitance at zero bias voltage. `M` is a parameter that HSPICE uses to calculate non-zero bias junction capacitance, and its dependence on the doping profile. In 6.012, this parameter is equal to 0.5. `TT` (sec) is the transit time, related to the diffusion capacitance, and `VJ` (Volts) is the built in potential. The first character of the next line is a `+` character, which means that the line is just a continuation of the previous one. You may use as many continuation lines as you need to complete a statement. The last of the parameter is `VB` (Volts), which is the reverse breakdown voltage.

The diode in the circuit above is specified by the line:

```
d1 2 3 diode1
```

The `d1` is the name of the diode. The `diode1` specifies to use the value for the model statement labeled `diode1`. The numbers `2 3` specify the positive and negative connections, respectively.

In 6.012 we will use three types of SPICE circuit analysis: DC analysis, AC small signal analysis, and transient analysis. The DC is self–explanatory; SPICE computes the DC operating points of all the nodes in the circuit. All inductors are shorted and capacitors are opened. You may also compute the DC operating point for a number of values of a voltage source, this is known as a DC sweep. For the AC small signal analysis, the DC operating point of the circuit is first determined then all the components on the circuit are linearized to their small signal model. The circuit is then simulated over a specified range of frequencies. For the transient analysis, the time varying signals within the circuit are simulated over a user-specified time interval. This analysis can be done after SPICE first calculates a DC bias point or with initial conditions the user specifies.

The original SPICE has two forms of output. The `.print` statement creates a table of output variable values for each value of the independent variable (time for a transient analysis, frequency for an AC small signal analysis, and the swept parameter for a DC sweep). The `.plot` statement creates a plot using characters (asterisks, dashes, and the like). This plot can be printed on a line printer, but the resolution is poor. The syntax of both statements is the same, and is of the form:

```
.print tran v(1) i(vin) i(D1)
```

The `tran` specifies that the `.print` applies to a transient simulation, and may be omitted if there is only one analysis called for in the input file. You may also use `ac` or `dc` for the corresponding types of analysis. Next come the output variables, you may print or plot as many output variables as you wish: `v(1)` prints the voltage on node 1, `i(vin)` and `i(D1)` print the current through the diode.

SPICE uses slightly different conventions than you were taught in 6.002. In particular, if a voltage source is supplying current to a circuit, then SPICE outputs a negative value. Similarly, the current is not defined as positive flowing into all the transistor leads. The current conventions for voltage sources and transistors are summarized in Figure [Ref: current] . Arrows point in the direction of positive current. A npn transistor is shown, but the pnp is similar. Positive current flows *into/* the collector, but *out of/* the emitter for both types of BJTs. Similarly, positive current flows *into/* the drain, but *out of/* the source for both types of MOSFETs.

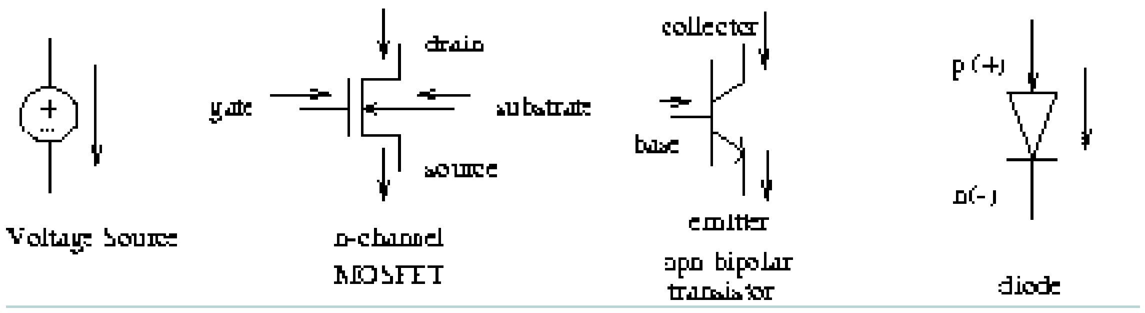

Table 2 SPICE Positive Current Conventions

2 Example

Next, to make the concepts clearer, let us look at an example designed for you to practice HSPICE simulation. The following HSPICE file defines the example circuit above and performs two analyses: a dc sweep of the circuit,and an ac sweep of the circuit.

```
6.012 hspice primer example1.sp
.model diode1 D level=1 IS=1e-14 CJO=1e-12 M=.5 TT=3e-11 VJ=.783
+ VB=30
d1 2 3  diode1
vin 1 0 1
vac 2 1 ac 1
r1 3 0 5k
c1 3 0 10p
* Analysis Control Starts Here:
.option TNOM=17

.dc vin 0 2 .05
.plot dc v(3) v(2,3) i(d1)
.ac dec 5 10k 10g
.plot ac v(3)
.options post
.end
```

Going through, step by step, what each line does:

line 1 `6.012 hspice primer example1.sp` — Simply gives the title of the file.

lines 3 & 4 `.model diode1 D ...` — Creates a diode model named `diode1`, and specifies the device parameters for it, as described above. This model is used for diode `d1`.

line 6 `d1 2 3 diode1 ...` — Specifies the node connections for diode `d1` and indicates that model diode1 is to be used to set the device parameters. Node 2 is the positive node and Node 3 is the negative node.

line 8 `vin 1 0 1` — Specifies an independent voltage source. After the title of the voltage source, `vin`, the node connections for the voltage source are given (nodes 1 & 0). The last value of line 8, `1`, specifies that the dc voltage of vin is 1 V from the positive polarity node to the negative polarity node. The node which is listed first is defined as the positive node.

line 9 `vac 2 1 ac 1` —This defines vac as an ac voltage source. The SPICE form for an AC Source is: `Vxxxx N+ N` AC ACMAG-. `N+` is the positive node of the source, while `N-` is the negative node. `AC` signifies an ac source, and `ACMAG` is the magnitude of the source.

line 11 `r1 3 0  5k` — This line specifies a resistor by name, gives the two nodes where the resistor is connected, then gives the value of the resistor.

line 13 `cl 3 0 10p` — This line specifies a capacitor by name, gives the two nodes where the capacitor is connected, then gives the value of the capacitance, 10 picofarads (basically the same form as for resistors).

line 16 `* Analysis Control Starts Here:` — This statement begins with an asterisk (`*`) which tells SPICE that the rest of the line is a comment. This line is ignored by the program but is useful for human readers.

line 18 `.option TNOM=17` — This specifies the operating temperature for all the devices in the circuit. To simulate kT/q as .025 Volts, which is the value used in 6.012, use a Temperature of 17 degrees Celsius.

line 21 `.dc vin 0 2 0.05` — This line invokes the dc sweep analysis option. Instead of using the dc value described above, voltage source `vin` will be varied from 0 to 2 volts in 0.05 volt increments. At each point the dc operating point will be computed.

line 22 `.plot dc v(3) v(2,3) i(d1)` — Specifies that the voltage at node 3, the voltage between nodes 2 and 3, and the current through the diode are to be plotted versus the dc sweep voltage (vin) over the range specified in the `.dc` statement.

line 23 `.ac dec 5 10k 10g`— Specifies an ac analysis. The SPICE form for AC analysis is `.ac Dec Nd Fstart Fstop`. `Dec` stands for decade variation, and `Nd` is the number of points per decade. `Fstart` is the starting frequency, and `Fstop` is the final frequency. SPICE will perform an ac analysis of the circuit over the specified frequency range after first determining the DC Operating point. During this part of the analysis, the voltage source vin will be 1 Volt.

line24 `.plot ac v(3)`— Specifies the voltage at node 3 is to be plotted versus the ac sweep frequency (vac) over the range specified in the `.ac` statement. Note that the output in this example will be the transfer function of the circuit.

line 26 `.options post` — This will create a file that can be used for advanced graphing using GSI (more to come).

line 27 `.end` — Indicates the end of the input file.

The following HSPICE files defines the same circuit, but performs a transient analysis using initial conditions.

```
6.012 hspice primer example2.sp
.model diode1 D level=1 IS=1e-14 CJO=1e-12 M=.5 TT=3e-11 VJ=.783
+ VB=30
d1 2 3  diode1
vin 1 0 1
vac 2 1 ac 1
r1 3 0 5k
c1 3 0 10p
.ic v(3)=10
* Analysis Control Starts Here:
.option TNOM=17
.tran 1ns 180ns UIC
.plot tran i(d1)
.options post
.end
```

The following lines are different in the two files:

line 14 `.ic v(3)=10` — This line specifies the initial voltage at node 3 to be 10 Volts. This initial voltage is used for transient analysis.

line 19 `.tran 1ns 180ns UIC` — This Specifies a transient analysis. The simulation will cover the first 180ns and the output will be given at 1ns intervals. `UIC`, or "Use Initial Conditions", tells SPICE to use the initial node voltages specified by the `.ic` commands. (Nodes not specified by voltage sources or `.ic` commands will start at 0.) A DC operating point is not found when `UIC` is used.

line 20 `.plot tran i(d1)` — Plots the current through the diode versus time (over the specified transient time period).

Note: When `UIC` is specified in a transient statement, the DC operating point is never computed. If AC and Transient simulations are specified in the same file and `UIC` is used, the initial conditions specified will be used for both the AC analysis and the Transient analysis. If you want to run an AC analysis at a DC bias point, you must do this in a separate simulation, as shown in these examples.

3 Running HSPICE on Athena

To run HPSICE on athena you will have to first add the HSPICE locker. You can do that by:

```
unix% add hspice
```

After this type (assuming that he file sample.sp is in /mit/yourid/hspice/):

```
unix% hspice /mit/yourid/hspice/sample.sp > sample.out
```

Look for errors in the `sample.out`. If there are no errors then you are ready to analyse the result. By default the files you will need to analyze will be created by hspice in your home directory.

4 HSPICE Enhancements

4.1 Using GSI

GSI is one of the tools that can be used to analyse HSPICE simulations. It lets you plot the simulated waveforms for analysis. To invoke GSI, add HSPICE locker, if you haven't already. Then type:

```
unix% gsi &
```

You can now load `*.sw*, and *.tr*` files into GSI (these files will be created when you have `.option post` line in the spice deck). Then select the waveforms interactively to plot it in GSI panels. One great advantage in GSI is, if you re-simulated the circuit with HSPICE you can update the plots in GSI by cliking "UPDATE" button in the waveform display window. Other features in GSI include:

- Multi-panel display (upto 6 panels), and multiple waveforms per panel.
- Measurement capability include 1st, and 2nd derivate, delta, intercept etc.
- Print selected panels to a printer or file.
- Selection of independent parameter for x-axis.

4.2 Automatic Measurements

One of the things that HSPICE will do is make "measurements" of various parameters of interest, such as rise and fall times, the voltage on one node when another node crosses a threshold, the maximum for a certain parameter during the transient simulation, or the power dissipation in a device or the whole circuit. This is done with the `.measure` statement. This section explains how to make one type of measurement, which will be useful in the upcoming design project, the frequency of the 3dB rolloff point in a transfer function.

As you noticed in the output file “example1.out”, at the end of the handout, there are a few statements in the transcription of the input file which were not yet explained. These are

```
.measure ac vmax max v(3)
.measure ac f3db when v(3)='vmax*.707' fall=1
```

These lines use the `.measure` statement, which performs a specified function on the data from the `.dc`, `.ac`, or `.tran` analysis. After the `.measure` statement, one has to specify the data from which type of analysis to process. In this case we specify `ac`. Next, we give the name to the output variable of `.measure`, in this case `vmax` or `f3db`. Next, we specify the type of `.measure` function we want. For example, as in the first line, `.measure analysis name max variable` finds a maximum of variable `variable` in the data from the analysis `analysis`, and names it `name`.

In the second line, the function finds when the variable `v(3)` falls for the first time below 0.707 of `vmax`, which was found in the previous statement. The syntax for this statement is `.measure analysis name variable=threshold fall=#`. This statement find the value of the input variable of the analysis `analysis` when the output variable `variable` falls below `threshold` the #th time. Notice that we are able to use values defined in other `.measure` statements, as well as algebraic expressions using these values. The algebraic expressions have to be enclosed in single quotation marks, as shown. The output of these statements can be seen at the end of the ac analysis in the “example1.out” file.

For a sample circuit

```
unix% add 6.012
unix% cd /mit/6.012/hspice/sample_not_gate_circuit
unix% more not1.sp
```

You have now learned all the SPICE you will need to know to get started. There are also books about SPICE simulation available in the library.