

HSPICE Tutorial for EE133

Prepared by Ben Mossawir

Introduction

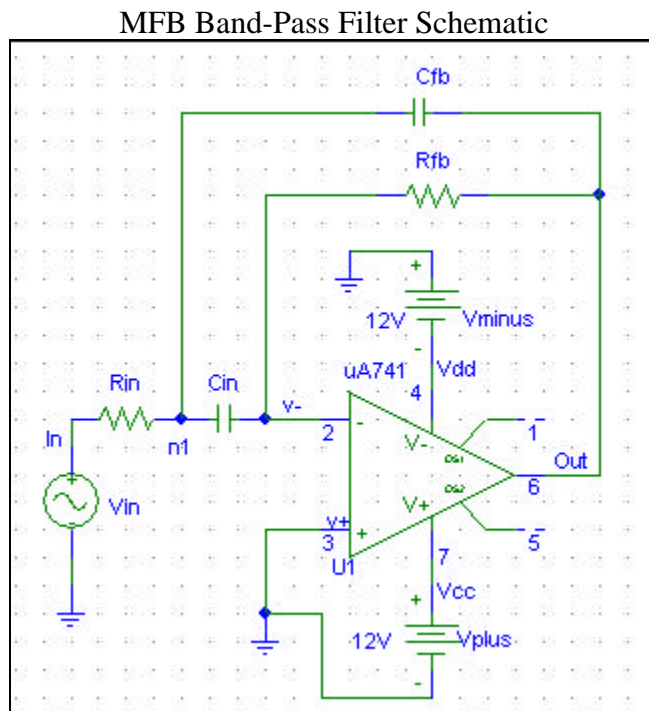
SPICE (Simulation Program for Integrated Circuits Emphasis) is a valuable tool for the rapid prototyping and design iteration that are hallmarks of EE133. Although SPICE comes in many flavors (you think that's a bad pun, keep reading), we strongly recommend that you utilize HSPICE for this course. HSPICE is a UNIX-based program that is one of the most powerful and widely used SPICE flavors.

HSPICE can be found on all the UNIX machines in Sweet Hall. You can also use the program remotely from home or the lab via telnet. However, to view the graphical outputs, you will need to run an X-server (e.g. X-win32).

In this tutorial, you will practice the steps necessary to perform simulation using HSPICE; namely:

1. Create a text **input file** that contains a description of your circuit and the commands you wish SPICE to perform.
2. Run the **HSPICE** compiler to generate text output files that contain the operating information about your circuit.
3. View the output files in **MWAVES**, which provides graphical interpretations of the data.

As an example, we will be simulating the performance of a MFB band-pass filter which all of you should have designed and built in EE122. The diagram of this circuit is below:



I. Input Files

A SPICE input file is known as a “spice deck” or simply “deck” because, at one time, several such files would be printed on cards and the whole ‘deck’ of cards would then be fed into the computer.

Getting Started

To get started from the UNIX prompt on the Sweet Hall machines, simply type `x` (lowercase) to run X-windows. Once the X-windows session loads, you can create a new window (or `xterm`) by right-clicking anywhere on the screen and selecting `Terminal` from the pop-up menu. Then, just pick your favorite color and another window will appear. It’s a good idea to use a secondary `xterm` for running your text editor.

To generate this file you are welcome to use your favorite UNIX text editor (`emacs`, `vi`, `pico`, etc.) but this handout will presume you are using `emacs` for simplicity. First, you will need to open a new file for this tutorial:

```
myth0:~>emacs tutorial.sp
```

The first lines of any deck should be a title for the file and a description of its contents. This will not only be helpful to the TA’s in deciphering your code but will help you to stay organized as you assemble a library of files. The ‘*’ is used to denote comments in a SPICE deck (friendly hint: I urge you to make liberal use of this character by commenting your deck extensively. In fact, the first line of your deck will be treated as a comment by the compiler regardless of what you put there, so be sure to only type comments on the first line.), so your header may be of the form:

```
* MFB Bandpass: Tutorial, v1.0  
* EE133 - Alex Tung  
*-----
```

Including Models

Then, much like in C, you will need to include any other files that will be referenced in your program. But, instead of libraries, your SPICE deck will reference models for circuit elements such as opamps, transistors, and diodes. For this class, all such models will be provided, so you need merely to link to them with a ‘`.include`’ command that looks like this:

```
.include 'UA741_model'
```

To avoid hassles, the model you reference should be in the same directory as the deck in order for HSPICE to find it. So, you should always make a local copy of the models we provide on the class webpage. If you need to link to a model that is not in the same directory as your deck, you should use:

```
.include '/usr/class/ee133/models/UA741_model'
```

Each model will have a specific syntax for invoking the circuit element it describes (see the ‘cheat sheet’), but if you are unsure you should always open the model deck and look at its contents before including one in your design.

Specifying the Netlist

Now, it’s time to tell SPICE what your circuit looks like. Almost all simulation programs expect this information in the form of a list of circuit elements and the nodes to which they are connected (called a “netlist”). For those of you who have used a graphical version of SPICE (e.g. PSPICE), this amounted to drawing out the circuit with a schematic editor and letting PSPICE convert that picture to an actual list of elements and nodes. But, as we all know, GUI’s are for whims. Besides, we often need to discover if the schematic capture tool has made an error (as in the infamous “I-thought-those-two-nodes-were-connected-by-the-wire-but-I-guess-it-wasn’t-touching” problem) and the best way to discover this is by examining the netlist. So, for HSPICE, you will type out the netlist by hand (oh boy!).

The first step to creating a netlist is to label the nodes in your circuit diagram with names that SPICE will accept (numbers or words are fine, but the number ‘0’ is always reserved for the ground node). We have chosen to use words to name the nodes as shown on page one.

Each circuit element has a specific syntax for how it should appear in the netlist (see the ‘cheat sheet’). For the MFB band-pass filter we are going to design in this tutorial, the netlist is as follows:

```

X1    0      v-    vcc      vdd      Out    UA741
      *This is the syntax for a UA741 opamp:
      * X1 in+ in- cc+ cc- out  UA741
Vplus vcc 0    12V
Vminus vdd 0   -12V
      * You could also specify this rail as:
      * Vminus 0 vdd 12V
      * Do you see why?
Vin   In    0    ac    1V
      * We want an 'ac' voltage source so
      * that we can examine
      * the frequency domain performance of
      * the filter
Rin   In    n1   500K
Cin   n1    v-   .016UF
Cfb   n1    Out  .016UF
Rfb   v-    Out  200K
      * Be careful with prefixes. "M" always
      * stands for "milli." Use "meg" or "X"
      * to denote "mega."

```

Control Statements

Once you have described the circuit, you just have to tell SPICE which domain you are interested in by specifying which type of analysis you want to run. The three most common analyses you will perform for this class are:

- DC analysis (.dc): For examining DC operation and bias points
- AC analysis (.ac): For examining frequency domain response
- Transient analysis (.tran): For examining time domain response

Each type of analysis requires you to specify a different set of parameters in the control statement (see the ‘cheat sheet’). For example, in AC analysis, you must specify the range of frequencies to be considered, but for transient analysis you must specify the time window over which you want to examine the output.

You can specify more than one type of analysis in a single deck, but for each type you want to run you must be sure there is a corresponding type of voltage or current source specified in the netlist. The types of sources that correspond to each analysis are:

- DC-type source: Vdc x y 5V (assumed by default)
- AC-type source: Vac x y ac 1V
- Transient-type source: Vsin x y sin(0 .1 24e6 0 16x) (or PWL, pulse, etc.)

You can even do mixed sources so that the same supply can be referenced in two different types of analysis, such as: V1 Vin 0 DC 1V AC 1V, 90 where the 90 means that the AC signal has a 90 degree phase shift.

For this deck, we are only concerned about the frequency response, so we have included an AC-type voltage source and will use the following control statement:

```
*-----  
* AC Response  
*-----  
.ac dec 20 1Hz 100kHz
```

Measure Statements

The last major piece of information we are missing is that we need to tell SPICE which parameters of the circuit we would like to measure. Often, this is just voltages and currents, but there are actually a whole set of measurement commands (see the ‘cheat sheet’) for everything from gain to slew rate.

For voltages and currents, we are often looking not for numbers but for plots of the value as a function of time or frequency. To obtain these, we use a ‘.print’ statement and list the quantities of interest. Each quantity is simply a measurement function applied to a particular node. To observe the voltage of the output as well as its magnitude and phase (i.e. a Bode Plot) we use the following command:

```
*-----
```

```
* Measurements
* -----

.print v(Out) vdb(Out) vp(Out)
```

If there are quantities whose value is simply a number, and therefore we do not need to see a plot, we can just measure them with a ‘.meas’ statement and the resulting value will appear in the output file. For instance, if we want to measure the peak gain of the filter (which should be at the center frequency) we can do so with the following statement and store the result in the variable `peak`:

```
.meas ac peak max v(Out) FROM=15kHz TO=45kHz
```

Lastly, we will almost always want to examine the operating point of our circuits. That is, we will want to know what the DC voltages and currents are during the transient or AC operation (this is especially useful for transistor circuits so that we can check our bias schemes). To obtain this information, we simply include the following line and then view the results in the output file:

```
.op
```

Finishing Touches

The last two lines should appear at the end of every SPICE deck. The first is to format the output file in such a way that it can be properly viewed with MWAVES. The latter is self-explanatory. But, be sure to leave a carriage return after the end command! Otherwise, HSPICE will not run and you will be hunting around for this problem for a long time:

```
.option post
.end
<cr>
```

II. HSPICE

Once you have completed and saved your deck, you can compile it using the following command from the UNIX prompt:

```
myth0:~> hspice tutorial.sp >! tutorial.lis &
```

This will run the HSPICE compiler and pipe the resulting output file into the file named `tutorial.lis`. It is suggested that you use the same name for the ‘.sp’ and ‘.lis’ files. The exclamation mark means that it will overwrite the existing version of that file without asking for confirmation. This is fine because you will be iterating through this process many times and do not want to accumulate a bunch of obsolete versions. In addition, HSPICE will generate several other output files with endings such as ‘.pa0’,

‘.st0’, and ‘.ic’. You can just ignore these and they will automatically be overwritten with each run through the compiler.

Don’t worry if it takes a couple seconds for HSPICE to run, but as long as the command completes successfully (you’ll know because it will say “***** hspice job concluded”, give you the time of execution, and return to the prompt), then you are golden!

Now, if you open up the tutorial.lst file in emacs, you’ll see the results of the simulation, including the values from any ‘.meas’ and ‘.op’ statements you included. Pretty nifty, eh?

III. MWAVES

The output file has lots of useful information in it and I encourage you to spend some time getting to know it (the structure is a little cryptic, so feel free to `grep` through it for the good stuff. If you don’t know what `grep` does, type `man grep` at the UNIX prompt). But, most of the time you will be interested in generating plots from your HSPICE results and for that you need to use MWAVES.

To view the results of the HSPICE compilation in MWAVES, type:

```
myth0:~> mwaves tutorial
```

This should open up MWAVES (also called Avant!Waves, or awaves, but don’t worry, it’s all the same thing) and you will be presented with a window called the ‘Results Browser.’ By clicking on the first line under the design field (which should look suspiciously like the first line of your spice deck) the remaining fields of the ‘Results Browser’ will be filled with the names of the nodes in your circuit. In addition, any quantities you specified with a ‘.print’ statement will be listed, as long as you also have included a ‘.option post’ at the end of your deck. If not, the ‘.print’ information will only appear in the output file.

Now you can select the variable you wish to examine by select the type from the ‘Types’ menu and then double-clicking on the node/measurement name or dragging it into the black rectangle onto the plotting window. The variable will be plotted against the independent variable for the type of analysis you have chosen, which is identified under ‘Current x-axis’ in the Results Browser (i.e. they will be plotted against frequency for our AC sweep). If you performed more than one type of sweep, you will be asked which set of results you want to view first when you load MWAVES. You can later go back and view the results of the other type of analysis.

Once you have the plot in the plotting window, you can right-click on the axes to adjust the scale or change to logarithmic progression, and right click on the black background to zoom in and out. In addition, you can plot other variables on the same axes (just drag their names in), add a second set of axes to the same window (look under the ‘Panels’ menu and select ‘Add’), or delete waves (click on the name of the wave under the ‘Wave List’ then go to ‘Delete Curves’ in the ‘Panels’ menu).

In fact, you can even plot the results of expressions such as $20\log(v(\text{Out}))$ (which of course gives the output voltage in dB) by using the expression editor. To do so, go to the ‘Tools’ menu and select ‘Expression’. Then, you can type in 20, select the * from the

‘Operators’ menu (double-click), select `log()` from the ‘Functions’ menu (double-click), and then drag the node named `Out` from the Results Browser to the ‘Expression’ field of the ‘Expression Builder.’ This will insert the name of the node in question into the argument of the log function. Now, name the resulting variable by typing `dBout` in the ‘Result’ field and hit ‘Apply’. The variable named `dBout` should now appear in the ‘Expressions’ menu and when you drag it into the panel on the plotting window – presto!

There is also a set of markers, or measurement cursors, you can use to measure things like the 3-dB bandwidth of your filter or the center frequency. First, you’ll need to configure the measuring options by selecting ‘Measure Label Options’ from the ‘Measure’ menu. In the window that pops up, turn off ‘Derivative’ and ‘Slope’, but enable the ‘Delta X’. Once you’re done, hit ‘OK.’ To take the bandwidth measurement, select ‘Point-to-Point’ from the ‘Measure’ menu in the plotting window. Your cursor is now a set of crosshairs whose position is constantly updated in the upper right-hand corner of the plotting window as you move it across the screen. Once you’ve located the first point that is -3 dB from the peak, click-and-drag the cursor to the corresponding 3 dB point on the other side of the passband and release. But, be advised, the resolution will only be as good as the number of frequency steps you took in the ‘.ac’ sweep. (Note: you can also do this using the ‘Anchor Cursors’ in the ‘Measure’ menu which perform a similar function).

Best of all, you can print out the results (it’s under the ‘Tools’ menu) and paste them into your lab notebooks or lab write-ups. That is a really nice touch, and it will be helpful for you as you get further along with the radio and forget what you’ve already done (trust me, it happens). Plus, it makes your TA’s happy ;)

IV. Above and Beyond

Well, that’s about it for this brief intro to HSPICE and MWAVES. For further information on this material, check out the cheat sheet as well as these online manuals:

Easy-to-navigate manual:

<http://www.ece.orst.edu/~moon/hspice98/>

Example deck with notes:

http://www.stanford.edu/class/ee214/Spice_reference/simple_amp.sp

Another online manual:

<http://www.stanford.edu/class/ee214/spice.htm>

And, of course, Happy Spicing!