UNIVERSITY OF CALIFORNIA AT BERKELEY
College of Engineering
Department of Electrical Engineering and Computer Sciences

EE105 Lab Experiments

Experiment 1: Introduction to SPICE

1 Objective

SPICE stands for **S**imulation **P**rogram with **I**ntegrated **C**ircuit **E**mphasis. It is the predominant tool used to simulate circuits and was developed at UC Berkeley in the 1970s. You will be using SPICE extensively in your circuit design courses, and this lab will teach you how to use a popular commercial package called HSPICE, one of many implementations based on the original Berkeley SPICE. You will learn how to specify and analyze a circuit with HSPICE and how to plot the results of your analysis with a tool called Avanwaves, or Awaves for short.

2 Materials

For this lab, all you need is a computer with HSPICE installed. You can use either the Windows or UNIX version, but you will have to read the appropriate documentation for whichever version you choose. The computers located in the EE105 lab in 353 Cory are Windows machines with the Windows version of HSPICE installed, so if you're working in lab you should use the Windows version. If you need to use HSPICE at home, you can either use Windows Remote Desktop to login to a Windows server and use the Windows version of HSPICE, or you can use SSH to login to a UNIX server and use the UNIX version of HSPICE.

3 Procedure

3.1 Transient Analysis

1. Write a netlist for the circuit in Figure 1, a simple RC low-pass filter. Let v_s be a 1 kHz square wave oscillating between 0 V and 5 V. You can model this square wave using the `PULSE` source type.

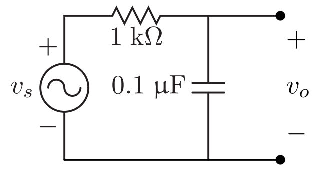

Figure 1: Low-pass filter

2. Run a transient analysis on this circuit from $t = 0$ ms to $t = 10$ ms.

3. Use Awaves to generate plots for v_s and v_o as functions of time. Print copies of these plots.

4. In Awaves, use the cursor to estimate how long it takes for the capacitor to charge from 0 V to 3.16 V (note that 3.16 is $5\,(1 - 1/e) \approx 0.63 \cdot 5$). Is this value close to what you'd expect?

5. Now, use a `.measure` statement to measure how long it takes for the capacitor to charge from 0 V to 3.16 V. How does this compare to your result obtained from the graph in Awaves? *Note: If you're using TRIG/TARG, you must use a non-zero value for the TRIG. If you want to measure from zero, simply use a very small, but non-zero, TRIG value.*

6. Adjust the step time used in your transient analysis (try making it larger and smaller). Does this variation change the result of your `.measure` statement?

3.2 DC Analysis

1. Figure 2(a) shows a transistor called an n-MOSFET, short for **n**-channel **m**etal-**o**xide-**s**emiconductor **f**ield **e**ffect **t**ransistor. We often call these NMOS or NFETs for short. Although you haven't learned how these work yet, you can still simulate them in SPICE. The terminals of the device are labeled for you: **D**rain, **G**ate, **S**ource, and **B**ody. In this case, the body is connected to the source, a configuration we'll see often in this course. In this configuration, transistors are often drawn as in Figure 2(b).

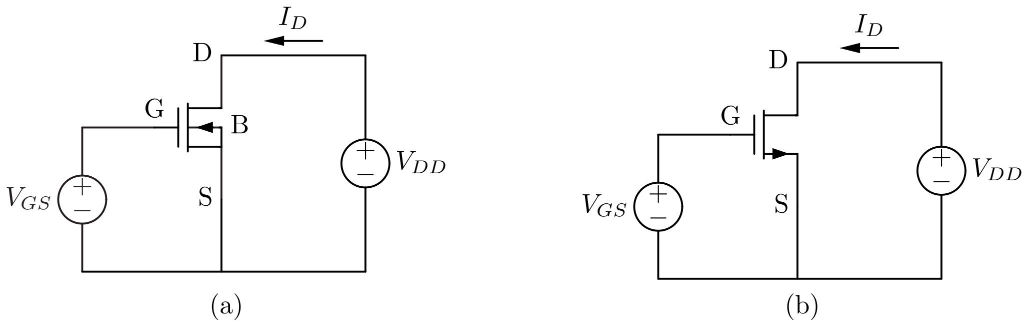

Figure 2: (a) NMOS transistor with body terminal explicity shown (b) NMOS transistor with body implicity tied to source

2. Write a netlist for the circuit in Figure 2 using the following parameters:
   - Step V_{GS} from 0 V to 5 V in increments of 1 V.
   - Sweep V_{DD} from 0 V to 5 V in increments of 0.1 V.
   - The model for the NMOS should have the following parameters: `kp=60e-6 vt0=1 lambda=0.05`
   - The dimensions of the transistor are $W/L = 4.5\ \mu\text{m}/1.5\ \mu\text{m}$

3. Use Awaves to plot I_D versus V_{DD} with all the values of V_{GS} you stepped in your DC analysis. *Note: You may have to invert I_D in Awaves—it should be positive and increasing with V_{DD} in your plot.* Everything should be shown on one graph. These are the I-V curves for a typical NMOS transistor.

3.3 TF Analysis

1. Figure 3 shows the small-signal model for a NPN BJT, short for **b**ipolar **j**unction **t**ransistor. We often model transistors with simpler, linear components in order to analyze them. Once again, we've labeled the terminals for you: **C**ollector, **B**ase, and **E**mitter.

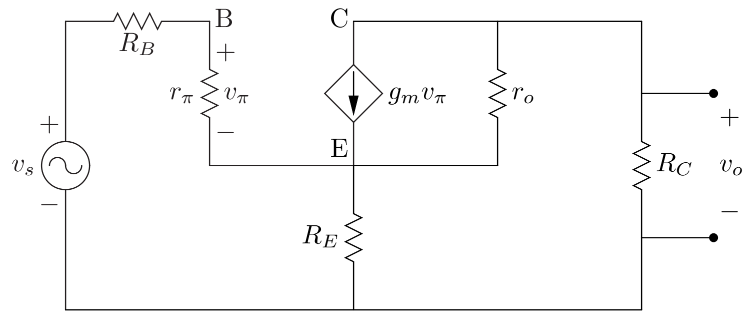

Figure 3: A small-signal model of a BJT

2. Write a netlist for the circuit in Figure 3 using the following parameters:
   - $R_B = 1$ kΩ, $r_\pi = 2.6$ kΩ, $R_C = 10$ kΩ, $R_E = 5$ kΩ, $r_o = 24$ kΩ, $g_m = 38$ mS
   - Use a `.tf` statement to find the voltage gain, $A_v = v_o/v_s$.

4 Appendix

4.1 Syntax Reference

This is the same syntax reference supplied in the HSPICE tutorial. Any bracketed labels must be replaced entirely (i.e. if you want a value of 5 V, you should replace `<value>` with `5V`).

- Independent voltage source
  `v<name> <+ terminal> <- terminal> <value>`
- Independent current source
  `i<name> <+ terminal> <- terminal> <value>`
- Voltage-controlled voltage source
  `E<name> <+ terminal> <- terminal> <+ control> <- control> <gain>`
- Current-controlled voltage source (`vcontrol` refers to the voltage source which the controlling current flows through)
  `H<name> <+ terminal> <- terminal> <vcontrol> <gain>`
- Voltage-controlled current source
  `G<name> <+ terminal> <- terminal> <+ control> <- control> <gain>`
- Current-controlled current source (`vcontrol` refers to the voltage source which the controlling current flows through)
  `F<name> <+ terminal> <- terminal> <vcontrol> <gain>`
- Sinusoidal source (used as a `<value>`)
  `sin(<offset> <amplitude> <frequency> <delay> <damping> <phase>)`
- Square wave source (used as a `<value>`)
  `pulse(<vmin> <vmax> <delay> <rise time> <fall time> <pulse width> <period>)`
- Piece-wise linear source (used as a `<value>`)
  `pwl(<t0> <v0> <t1> <v1> <t2> <v2> ...)`
- Resistor
  `r<name> <terminal 1> <terminal 2> <value>`
- Capacitor
  `c<name> <terminal 1> <terminal 2> <value>`
- Inductor
  `l<name> <terminal 1> <terminal 2> <value>`
- Model (type can be `nmos`, `pmos`, `NPN`, `PNP`, or `D` for diode)
  `.model <name> <type> (<parameter list>)`
- MOSFET (you can specify additional parameters, such as `W=<value> L=<value>`, in the parameter list)
  `m<name> <drain> <gate> <source> <body> <model> <parameter list>`
- BJT
  `q<name> <collector> <base> <emitter> <model> <parameter list>`

- Diode
  `d<name> <+ terminal> <- terminal> <model> <parameter list>`
- AC analysis (pick either `lin`, `dec`, or `oct`)
  `.ac <lin|dec|oct|> <number of samples> <freq start> <freq stop>`
- DC analysis
  `.dc <source> <start> <stop> <step>`
- Nested DC analysis (`source1` is swept, `source2` is stepped)
  `.dc <source1> <start1> <stop1> <step1> <source2> <start2> <stop2> <step2>`
- Transient analysis
  `.tran <t step> <t stop>`
- TF analysis
  `.tf v(<node>) <source>`
- PZ analysis
  `.pz v(<node>) <source>`

4.2 Using a GUI

There are many graphical editors for SPICE netlists available. In the HSPICE tutorial and in this lab, we've mostly concentrated on the actual syntax behind netlists. Now that you're familiar with netlists, the transition to a graphical editor should be quite straightforward. Although we aren't going to provide tutorials for these programs, we will point you to other documentation that should help you get started.

One SPICE GUI that is available on the lab computers is called Multisim. It is available under *Start → Programs → Electronics Workbench → Multisim 9 → Multisim 9.* Multisim is not available through Remote Desktop, so you must be in lab to use Multisim. A tutorial, written by B. Muthuswamy and B. Boser, is available here.

A free SPICE GUI called XCircuit is also available. It will allow you to draw a circuit, then export the drawing to a netlist. It will not handle analysis statements, so you will still have to edit the netlist by hand after exporting the netlist.