NGspice Primer
Within CppSim (Version 5) Framework

Michael H. Perrott
http://www.cppsim.com

October 12, 2013

Table of Contents

Introduction

CppSim is a general behavioral simulation framework that leverages the C++ language to achieve very fast simulation times, and a graphical framework to allow ease of design entry and modification. Users may freely use this package for either educational or industrial purposes without restriction. However, the package and all of its components come with ***no warranty or support***.

Starting with CppSim Version 4.1, NGspice is included as an auxiliary simulator within the CppSim framework. At this point in time, there is no direct interaction between NGspice and the CppSim simulator, but there are various conveniences provided for NGspice such as schematic entry using the Sue2 schematic editor, a simple GUI interface for running NGspice simulations, a waveform viewer using CppSimView, support for Python using the Ngspice Data module for Python, and Matlab/Octave using the Hspice Toolbox for Matlab/Octave. All of these packages are included within the CppSim installation file.

To install this package, a self-extracting installation file for Windows 7/Vista/XP/2000 machines is readily downloadable from the web at http://www.cppsim.com. Running this file automatically installs several sub-packages to perform the various tasks required:

1) Sue2: a free, open source, schematic capture program that is easy to use and has a similar look and feel as Cadence Composer,
2) CppSimView: a free waveform viewer that allows easy plotting of signals produced by CppSim and NGspice,
3) CppSim and Ngspice Data Modules for Python: a free, open source set of Python classes and routines to allow straightforward access to CppSim and Ngspice simulation results within Python.
4) Hspice Toolbox for Matlab/Octave: a free, open source set of Matlab/Octave routines to allow straightforward access to Hspice, NGspice, and CppSim simulation results within Matlab or Octave,
5) Emacs: a free, open source, text editor that is especially convenient for writing and editing simulation files used with NGspice,

This document is intended as a primer that covers basic use of NGspice in conjunction with Sue2, CppSimView, and Matlab. While this document covers enough information on running NGspice within the CppSim framework to get a good idea of its operation, a more full description of the capabilities and functionality of NGspice is provided in its manual available in **c:/CppSim/CppSimShared/NGspice/doc/ngspice23-manual.pdf** (Note that **c:/CppSim** should be replaced by the actual path you chose for CppSim during its installation). The Sue2, CppSim Data module for Python, and Hspice Toolbox manuals are provided in the files **sue2_manual.pdf, cppsimdata_for_python.pdf, and hspice_toolbox.pdf**, respectively, and are available in the **Doc** menu of Sue2. Note that there is no separate manual for CppSimView – this document contains a full description of CppSimView.

Setup (Windows 7/Vista/XP/2000)

Go to the web site http://www.cppsim.com/dowload, and then download the file **setup_cppsim5.exe**. To install, simply run **setup_cppsim5.exe** in Windows (i.e., double-click on **setup_cppsim5.exe** in Windows Explorer) and follow the instructions. To run Sue2 or CppSimView, click on their respective Windows icons once the installation process has completed and Windows has restarted.

Known Bugs

1) Some computers require installation of the Microsoft Visual C++ 2008 Redistributable Package (x86) in order to run NGspice. This is a small set of DLL files, and is easily downloaded and installed directly from Microsoft's website at: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=29

2) As in keeping with SPICE conventions, the first letter of a schematic element must be appropriately specified (i.e., **R** for resistor, **C** for capacitor, **V** for voltage source, etc.). CppSim automatically provides the correct value of this letter when creating a new instance – be sure not to change that first letter!

3) True library support is lacking in Sue2 right now (i.e., name clashing occurs between cells of the same name even though they may be in different libraries). This issue is taken care of by using the Import and Export tools of Sue2.

4) The undo command in Sue2 is broken.

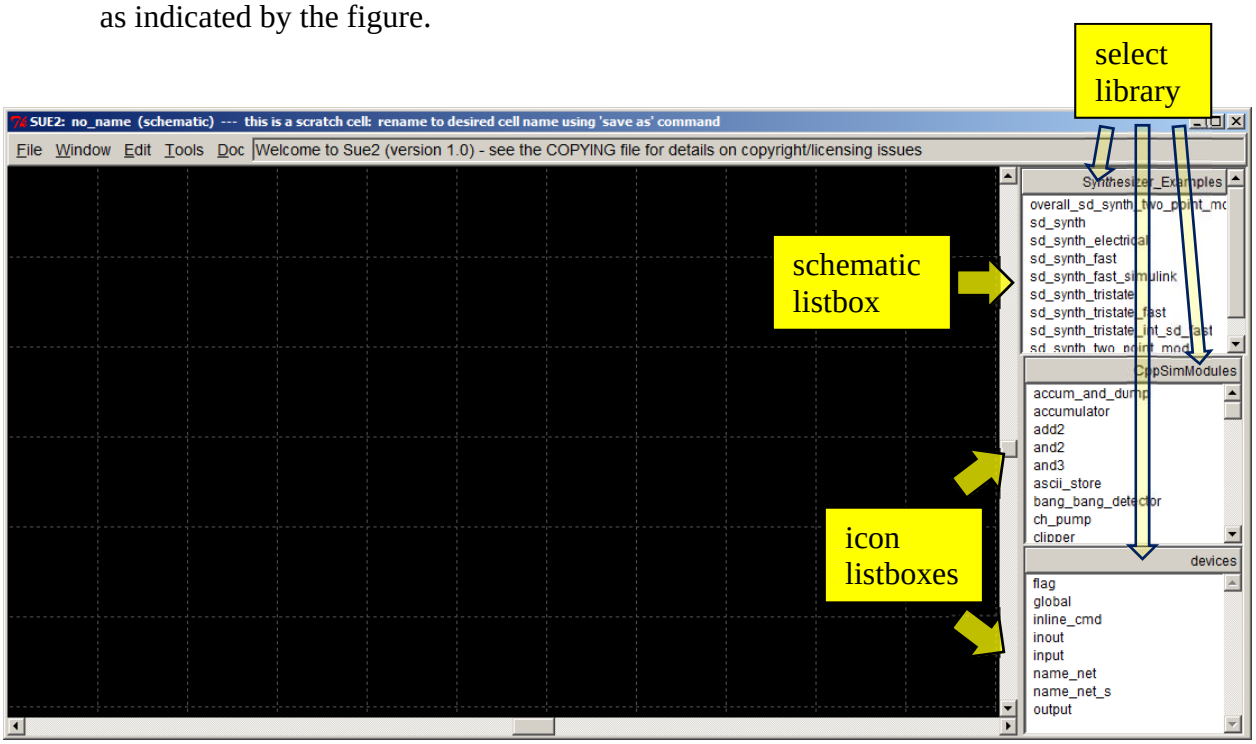
5) Sometimes the history file of CppSimView gets corrupted and does not allow CppSimView to start. If so, within Windows Explorer, go to the SimRuns directory associated with the current cell of Sue2, and then erase the file called **cppsimview_history.mat** within that directory. As an example, if Sue2 is currently displaying cell **sd_synth_fast** within library **Synthesizer_Examples**, then delete the file **cppsimview_history.mat** located within directory **c:/CppSim/SimRuns/Synthesizer_Examples/sd_synth_fast** (where **c:/CppSim** corresponds to the base directory location that CppSim was installed at, and may be different for different machines).

The Basics of Running NGspice Simulations

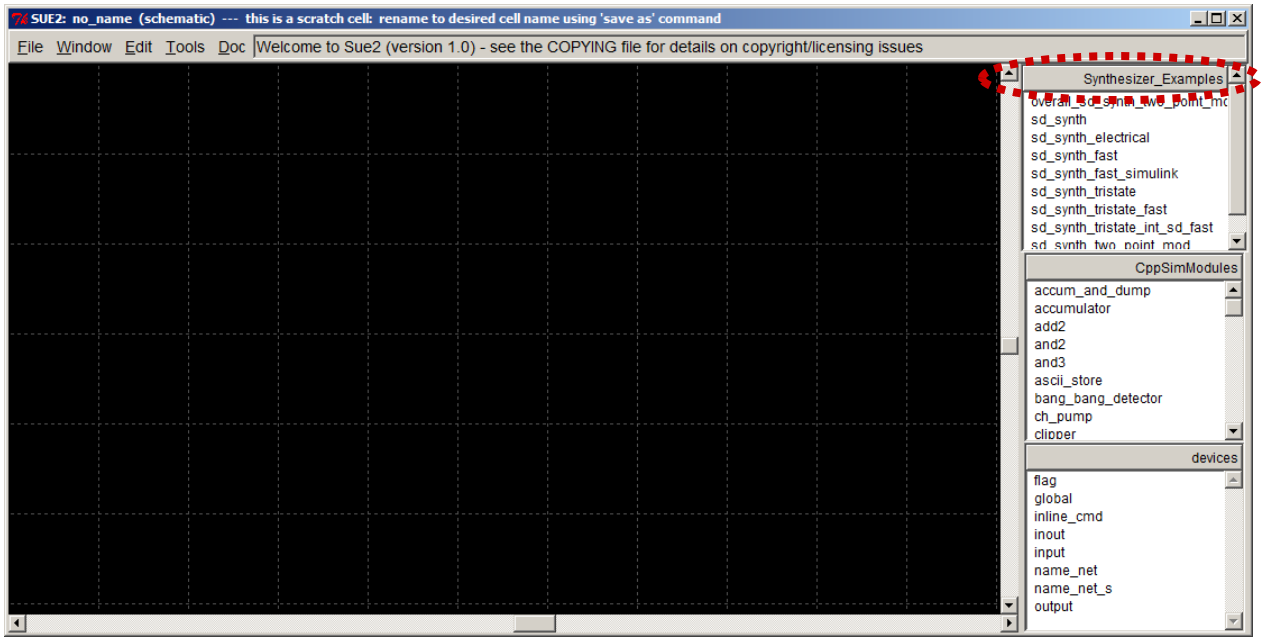
To explain the basic operation of running NGspice within the CppSim framework, let us now walk through an example using the **Sue2** as the schematic editor and **CppSimView** as the simulation viewer.

A. Running an NGspice Transient Simulation in Sue2

- Open up **Sue2** by clicking on its icon on the Windows Desktop. You should see a window similar to what is shown below. Note that there is one schematic listbox and two icon listboxes, each of which lists cells from the library that is selected by pushing their top button as indicated by the figure.

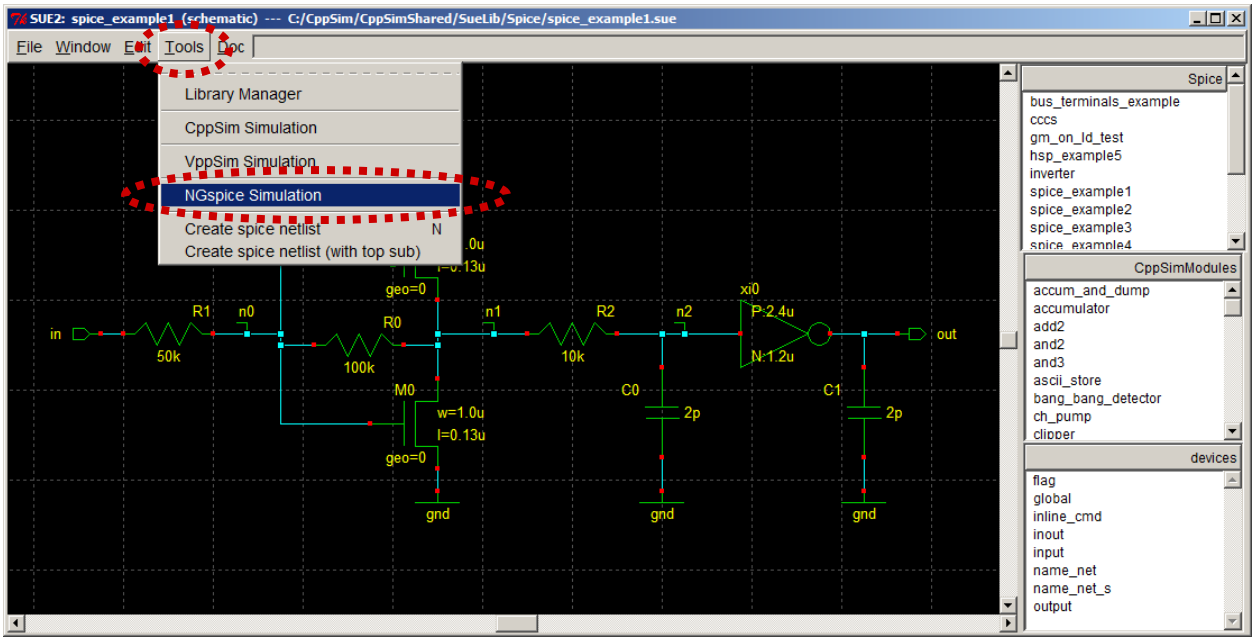


- Select the **Spice** library by clicking on it in the top portion of the **schematic listbox** as illustrated below

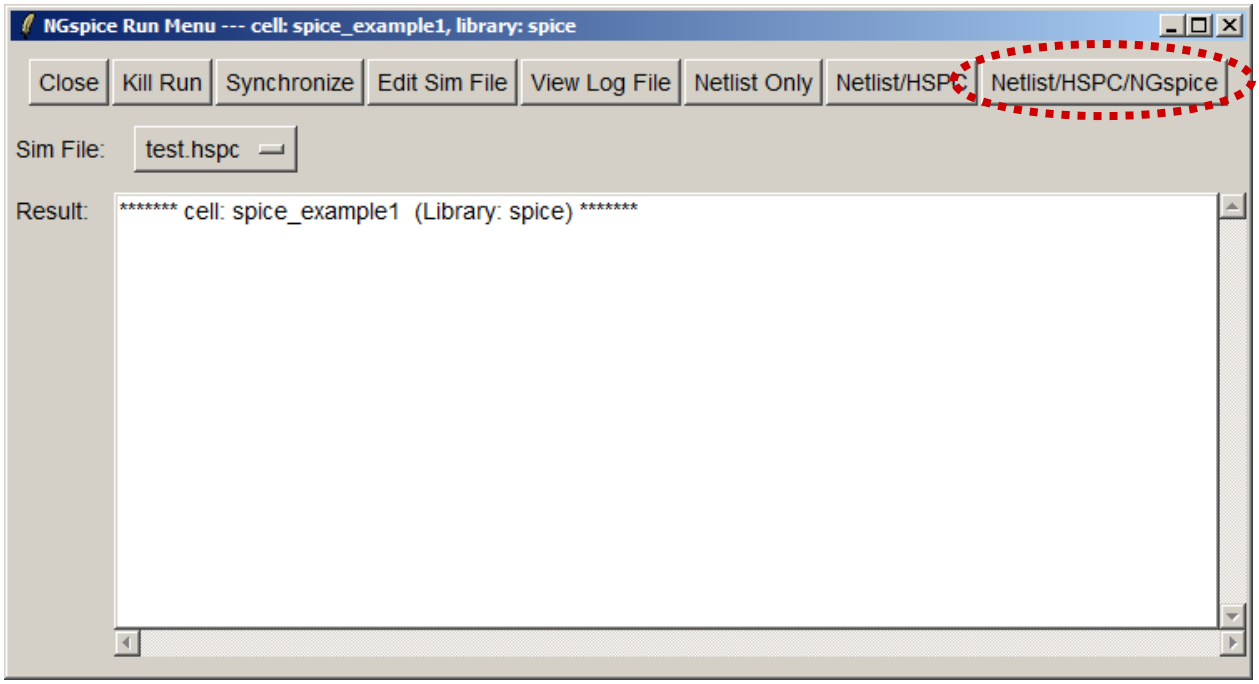


- Select the **spice_example1** schematic by clicking on it in the **schematic listbox**.

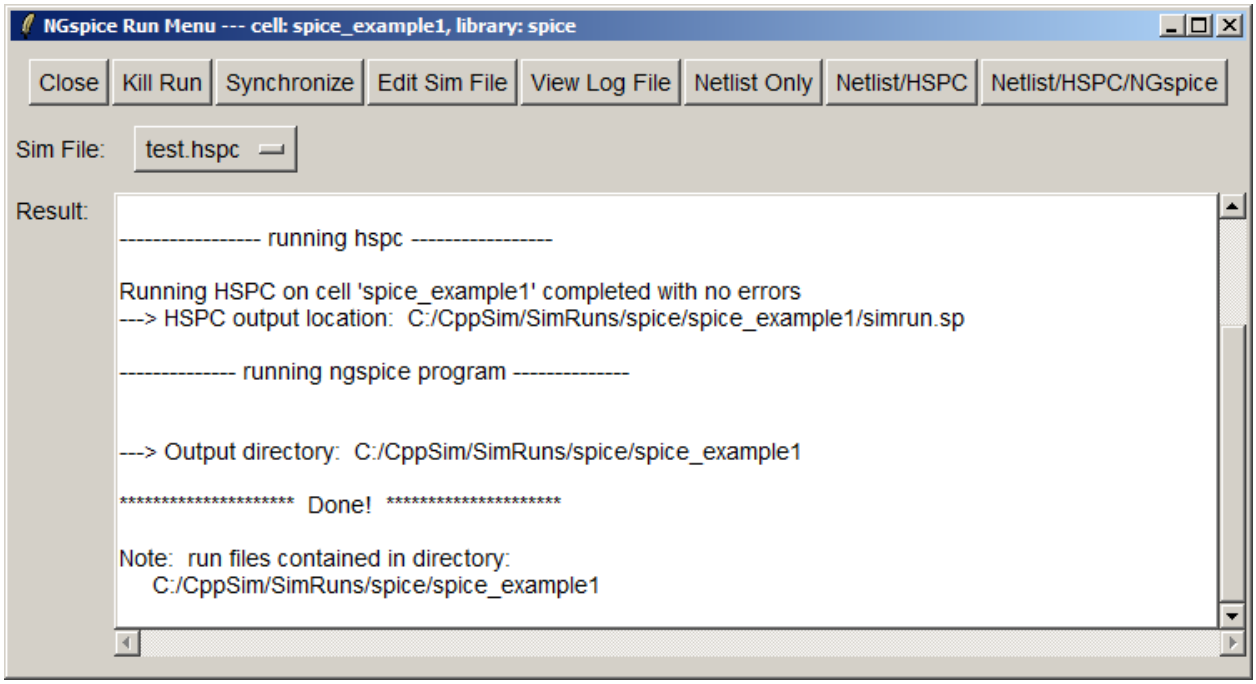
- Sue2 should now display the **spice_example1** schematic as shown below. Note that the **NGspice Simulation** menu item is obtained by clicking on **Tools** (circled in red below).



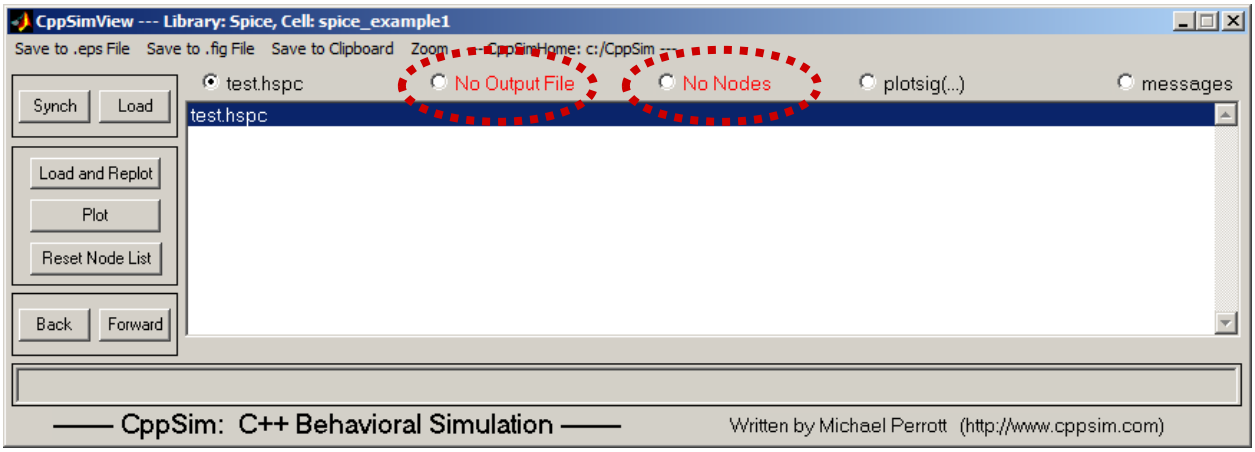
- Clicking on the **NGspice Simulation** menu item, as shown above, yields the **NGspice Run Menu** as shown below. Note the **Netlist/HSPC/NGspice** button, which is circled in red.



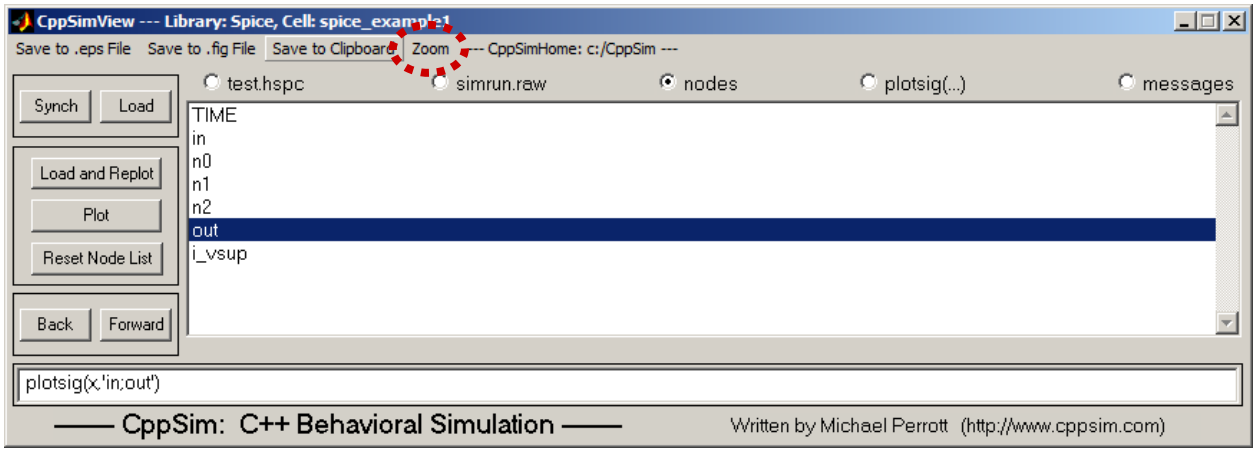
- Run an NGspice simulation on the **spice_example1** cell by clicking on the **Netlist/HSPC/NGspice** button shown in the above figure. You should see some messages in the window along with an additional simulation window that eventually closes, and then finally the window should appear as shown below.



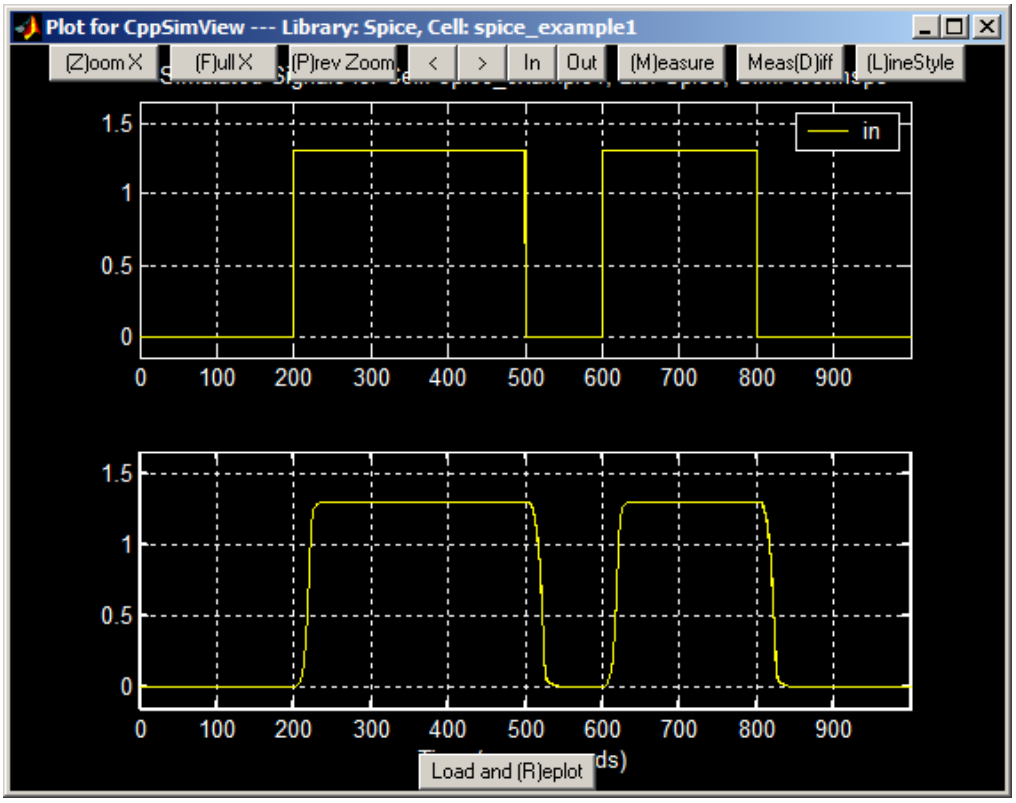
- To view results of the simulation, click on the **CppSimView** Icon on the Windows Desktop. You should see a new window appear as shown below.



- To view simulation results for a given signal within the **spice_example1** cell, you need to first choose an appropriate **Output File** and then the **Node** that the signal is associated with. In the window shown above, first click on the No Output File radio button, and choose **simrun.raw** as the output file. Next click on the No Nodes radio button, and then double-click first on node **in** and then on node **out**. The resulting CppSimView window should appear as shown below, and the Plot Window should show the corresponding signal waveforms.



- Now click on **Zoom** (circled above in red). The resulting Plot Window should appear as shown below.

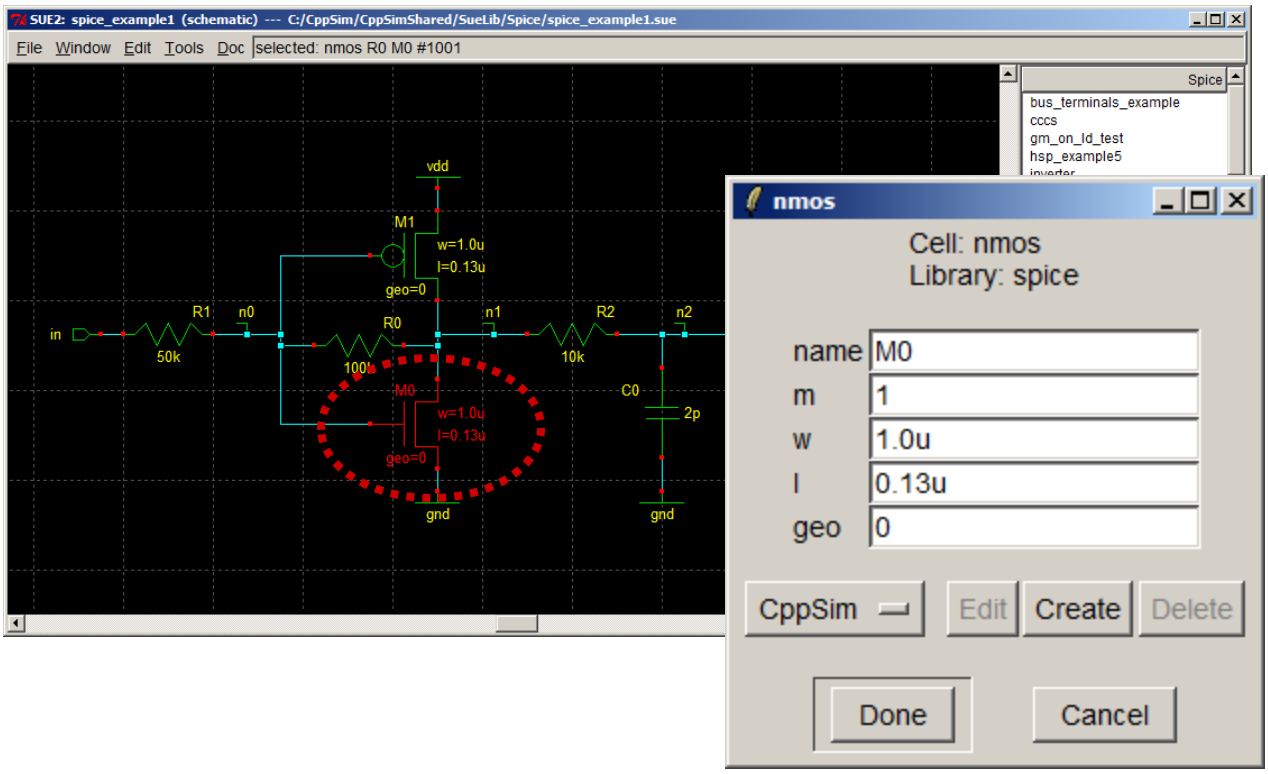


- Consider clicking on different buttons in the Plot Window to zoom into portions of the signals and perform various other operations. One convenient feature is the use of the arrow keys on your keyboard to zoom in, zoom out, and pan left and right. The **down** arrow key zooms in, the **up** arrow key zooms out, and the **left** and **right** arrow keys pan left and right, respectively.

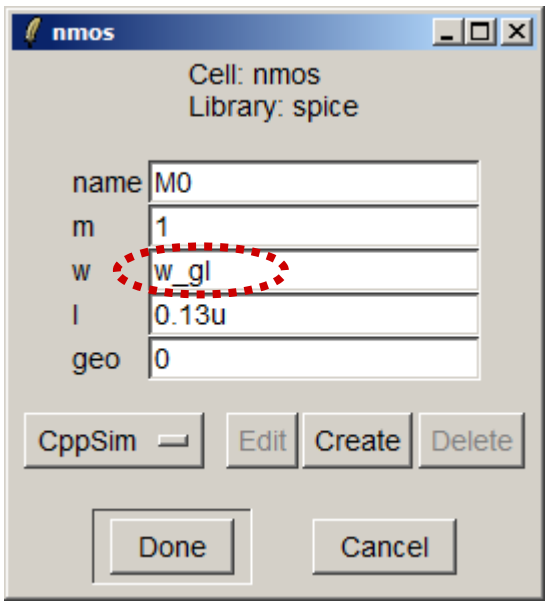
B. Editing Schematic Level Module Parameters

There are two approaches to specifying schematic level parameters when running NGspice within the CppSim framework – one can either specify parameter values in the schematic or within the simulation file (i.e., **test.hspc** for the example discussed above). We will discuss each of these approaches in this section.

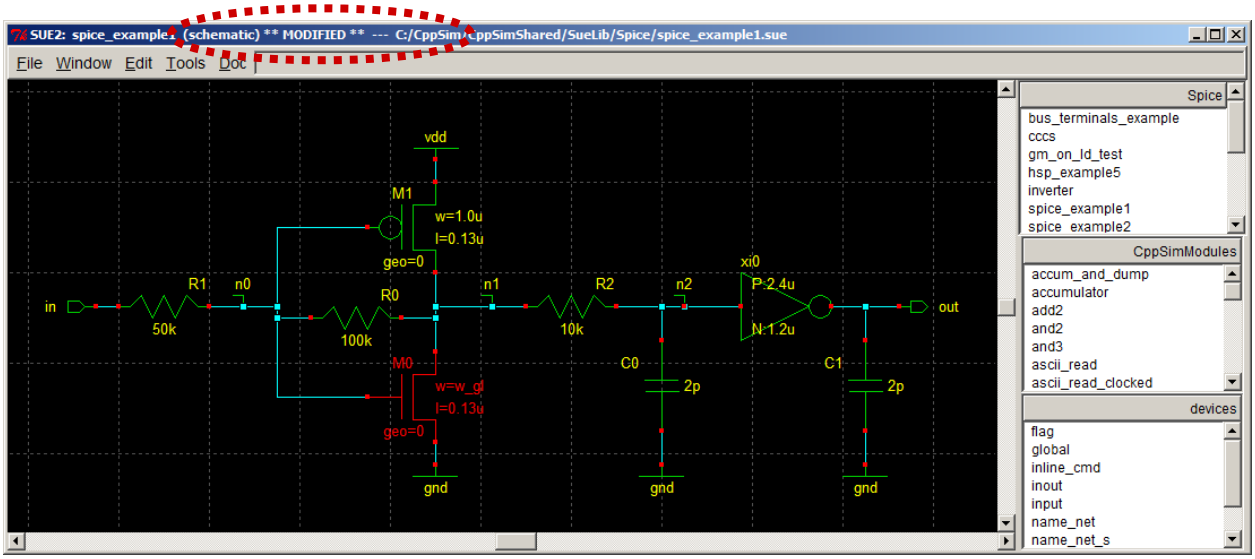
- Within the Sue2 schematic window, double-click on a given module such as **M0** (circled in red in the figure below). You may then edit the model parameters and then click on **Done** to save their new values.



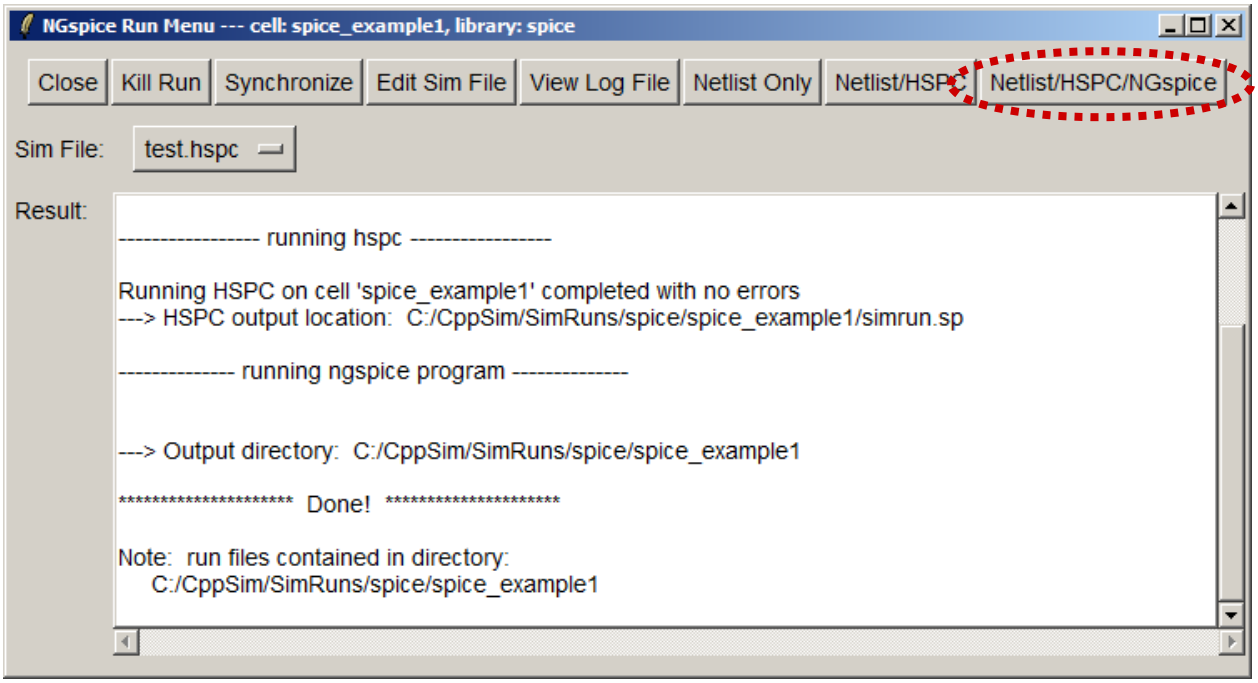
- To change parameters within the simulation file (i.e., **test.hspc**), first change the **w** parameter of transistor M0 to the value **w_gl** as shown below and then click on the **Done** button.



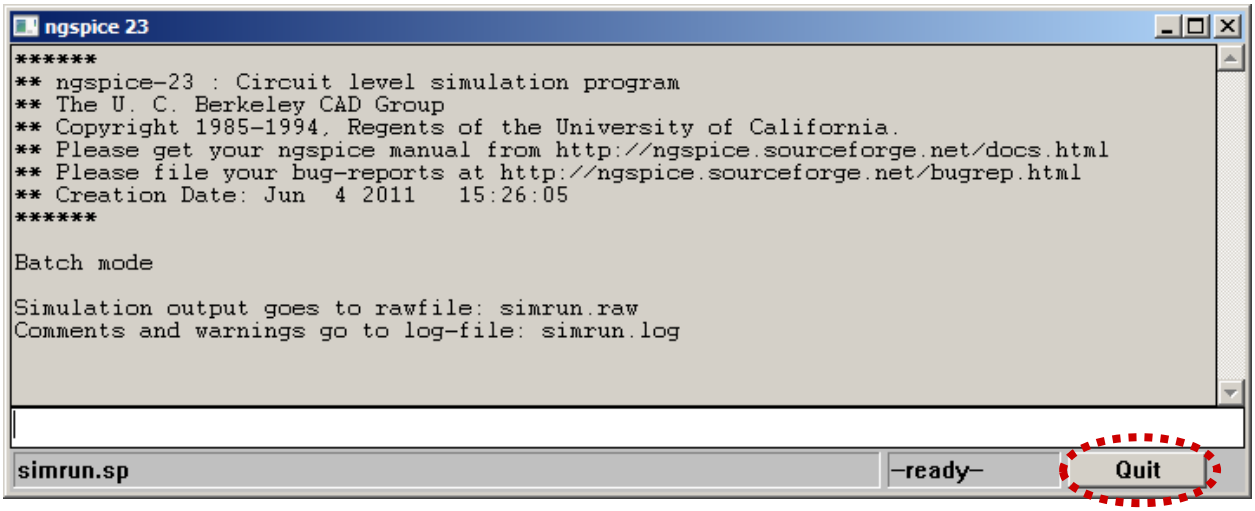
- Whenever you make changes to a schematic, the title bar will show **MODIFIED** as shown in the figure below. Be sure to save the schematic before running simulations by typing **Ctrl-s** or clicking on **Save** within the **File** menu, and verify by checking that the **MODIFIED** text goes away in the title bar.



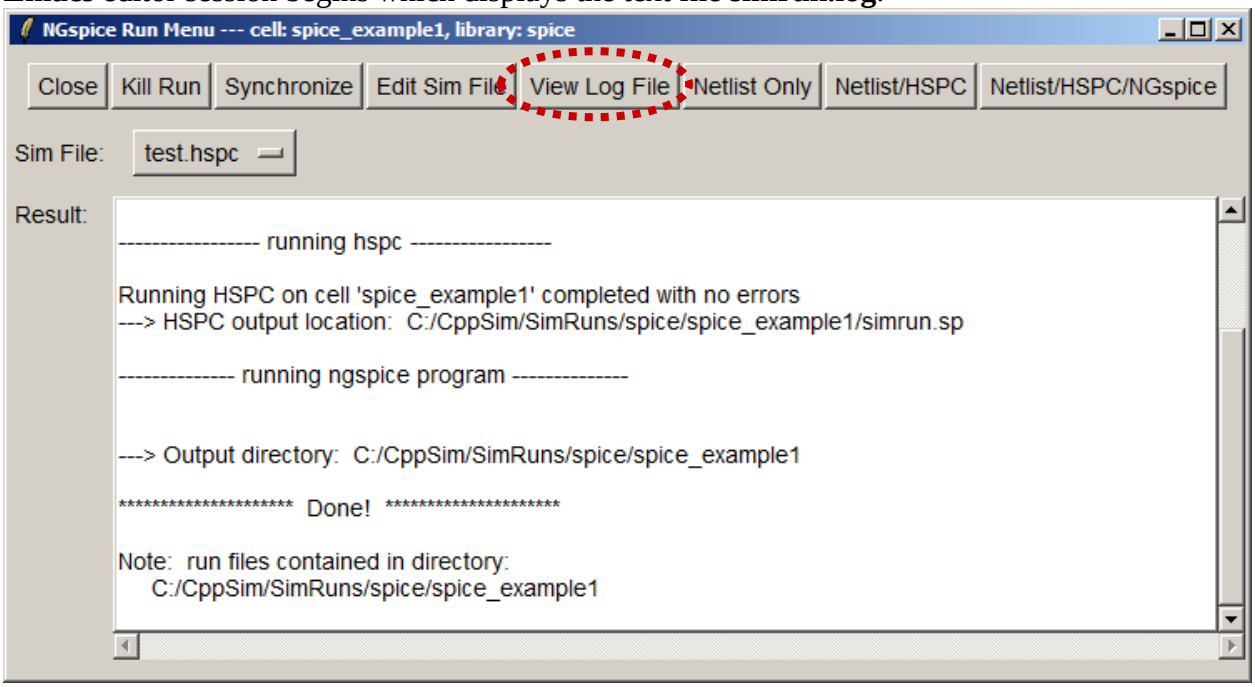
- Now click on **Netlist/HSPC/NGSpice** in the **NGspice Run Menu** as shown below. You will find that the **ngspice simulation window** appears but that the simulation does not go forward. Unfortunately, there is no prompt telling you this, so you must simply pay attention to this sort of issue happening. However, note that the NGspice Run Menu window will often alert you to error messages, so be sure to examine it after every run.



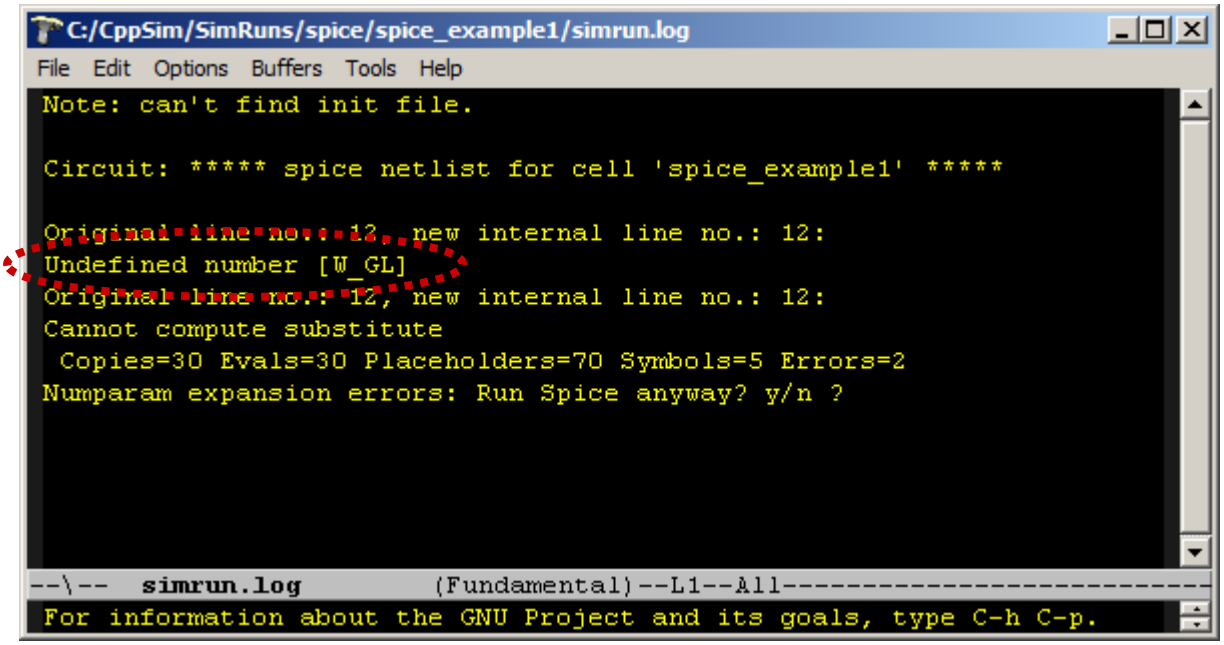
- Now click on **Quit** button within the **ngspice simulation window** shown below. This will stop the simulation and allow you to move forward in your investigation of what went wrong.



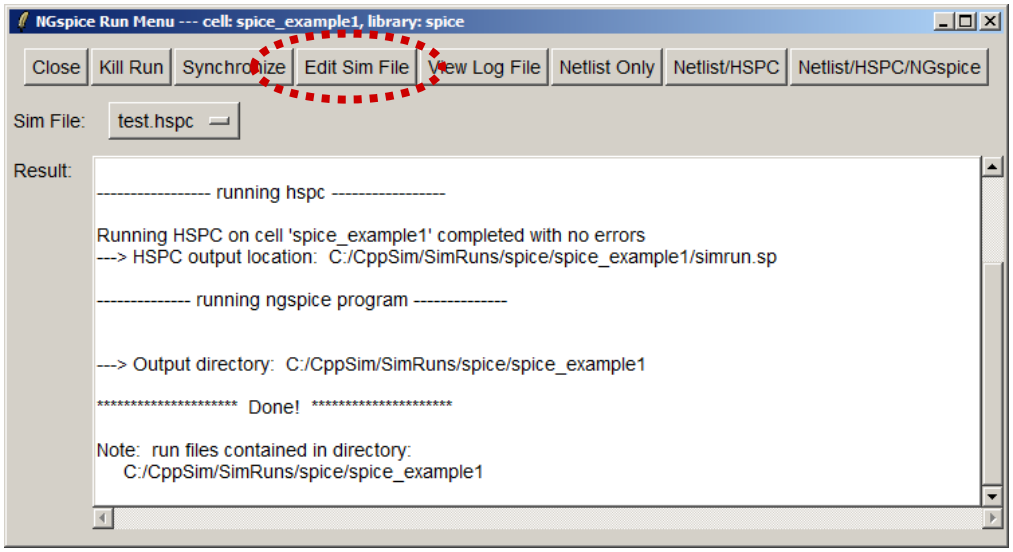
- Now click on **View Log File** in the **NGspice Run Menu** as shown below. You will find that an **Emacs** editor session begins which displays the text file **simrun.log**.



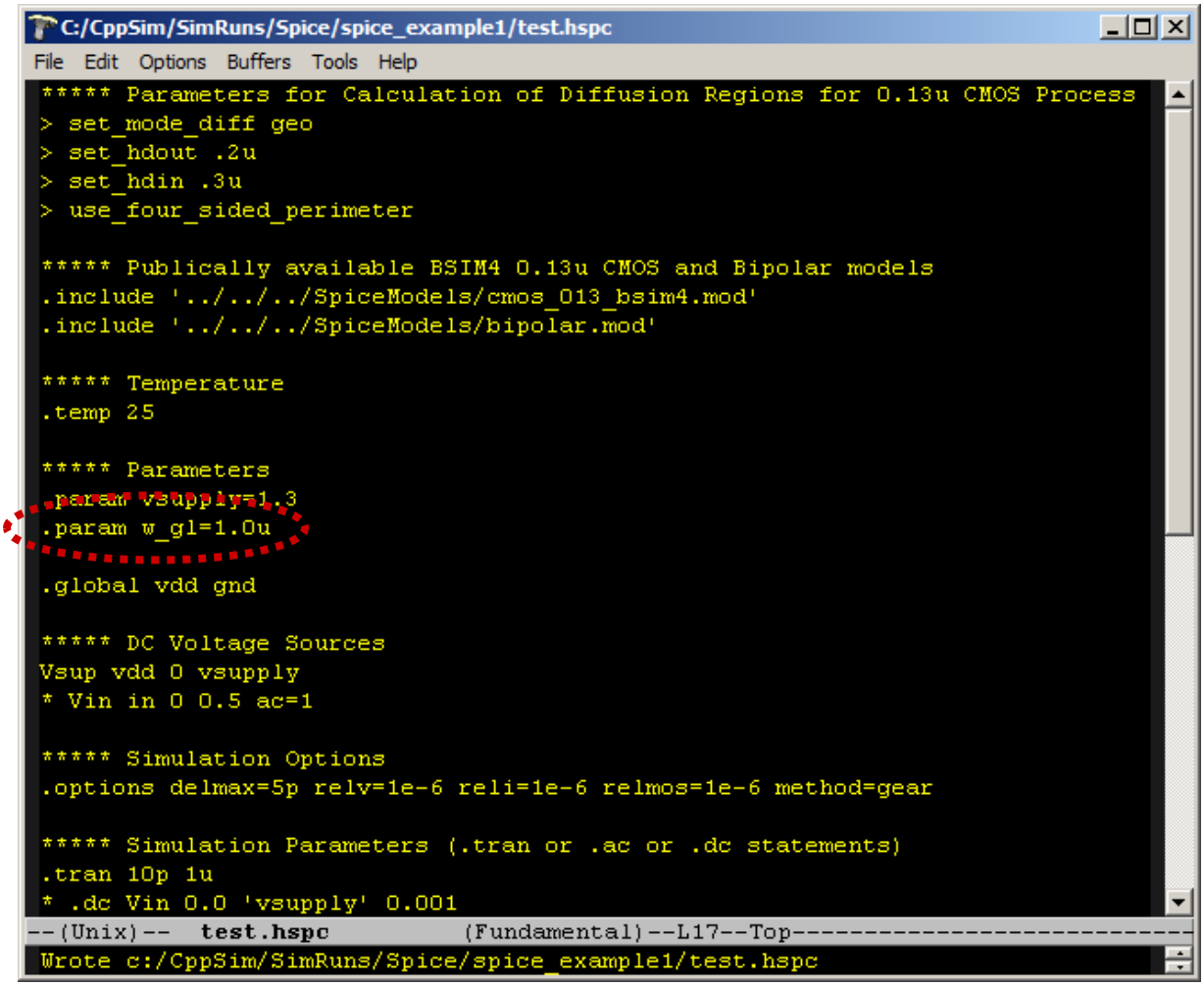
- In examining the **simrun.log** file shown below, we see that the error is that the parameter **W_GL** is undefined. This is the very same parameter **w_gl** that we entered into the schematic earlier, and the issue is that we never chose its value. We will do this by editing the **test.hspc** simulation file, as described in the next step.



- Now click on **Edit Sim File** in the **NGspice Run Menu** as shown below. An **Emacs** window that displays the file **test.hspc** simulation file will appear.



- Within the **Emacs** window as shown below, edit the **test.hspc** simulation file to include the line: **.param w_gl=1.0u**
  Make sure that you save the file. You have now specified the value of **w_gl**, and so click on **Netlist/HSPC/NGSpice** in the **NGspice Run Menu** to re-run the **NGspice** simulation. This time the simulation should complete and you can click **Load and Replot** within the **CppSimView** window to view the updated results (which should look identical to the previous results).



C. Editing Simulation Files (i.e., test.hspc files)

In the above exercise, you received some brief exposure to **Simulation Files**, but it will important for you to better understand the role of these files going forward. While the **Sue2** schematic describes the overall circuit topology that is to be simulated, additional simulation specifications must be provided to **NGspice** such as the type of analysis to be performed, the parameters of that analysis (such as the time duration of a transient simulation as performed in the previous example), the value of global parameters used within models in the schematic (as discussed in the previous section) as well as global nodes, and many other items which are described in further detail within the **NGspice**, **HSPC,** and transistor model documentation located under the **Doc** section of the Sue2 window as well as:

- **NGspice** Documentation: **c:/CppSim/CppSimShared/NGspice/doc/ngspice23-manual.pdf**
- **HSPC** Documentation: **c:/CppSim/CppSimShared/Doc/hspc.pdf**
- **Berkeley BSIM4 Model** Documentation: **c:/CppSim/CppSimShared/NGspice/doc/BSIM464_Manual.pdf**
- **ASU Predictive BSIM4 Model**: http://ptm.asu.edu/
    - Y. Cao, T. Sato, D. Sylvester, M. Orshansky, C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," pp. 201-204, CICC, 2000

In this section we will touch on key points related to the above documentation, but there are many important details in the above documents which will merit their examination. Note that the **CppSim** framework allows you to specify several simulation files (such as **test.hspc**, **test_ac.hspc**, etc.) in order to perform different types of analysis on a given schematic. However, in this document, we will focus on editing the same simulation file to perform different types of analysis.

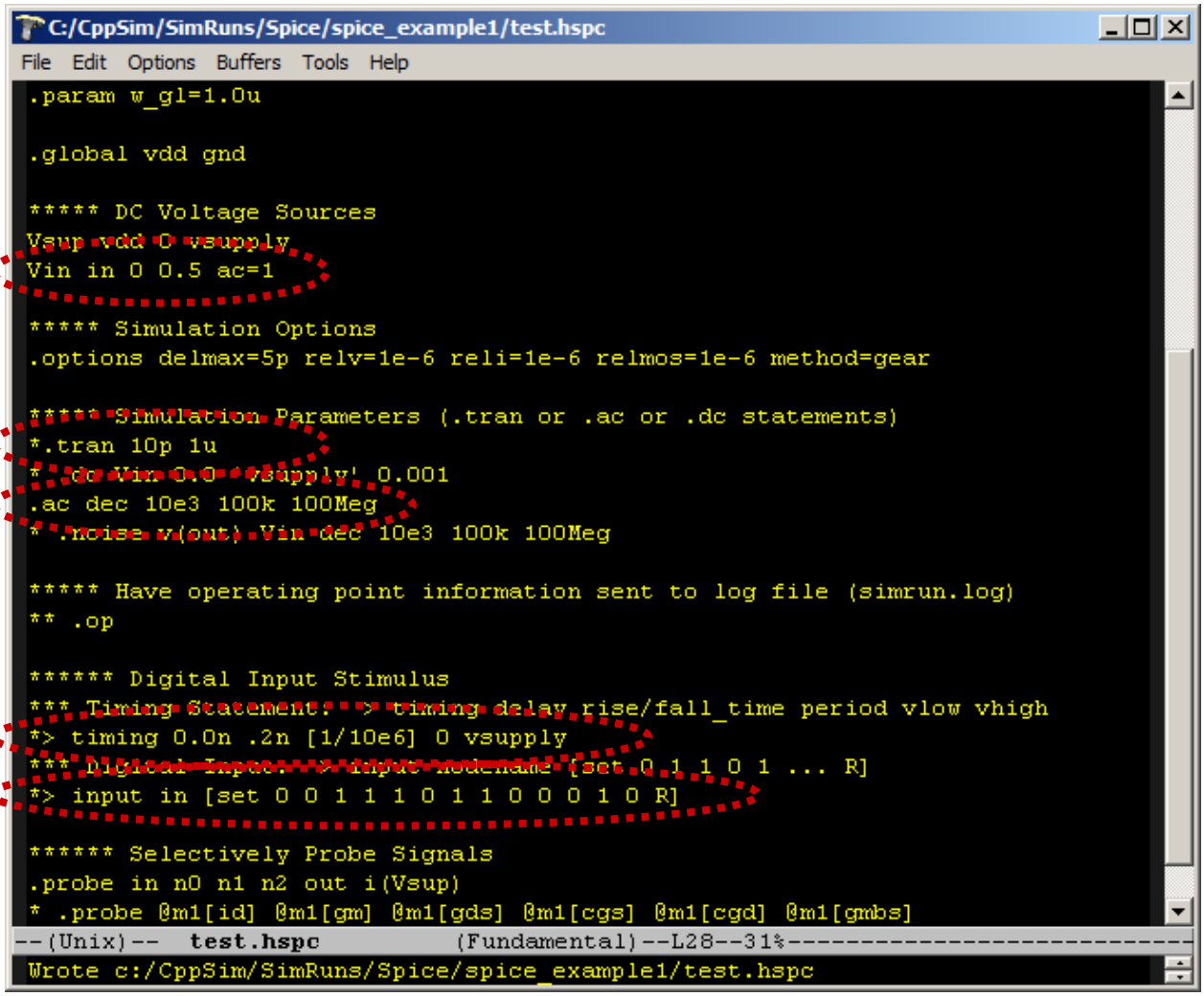
- Continuing with the example of the previous section, if you closed the **Emacs** window containing the **test.hspc** file, then click on the **Edit Sim File** button within the **NGspice Run Menu** window. The **test.hspc** simulation file should appear within **Emacs** as shown in the previous section. There are several key points to make of this file:
    - Lines that start with '*****' are comments, and are ignored by **NGspice**,
    - Lines that start with '**>**' are **HSPC** commands, and are used to augment the normal functions provided by **NGspice**. One should examine the **HSPC** documentation indicated above for further information on these commands.
    - Lines not falling under the above two categories are native **NGspice** commands. One should examine the **NGspice** documentation indicated above for further information on these commands.

D. AC Analysis Example

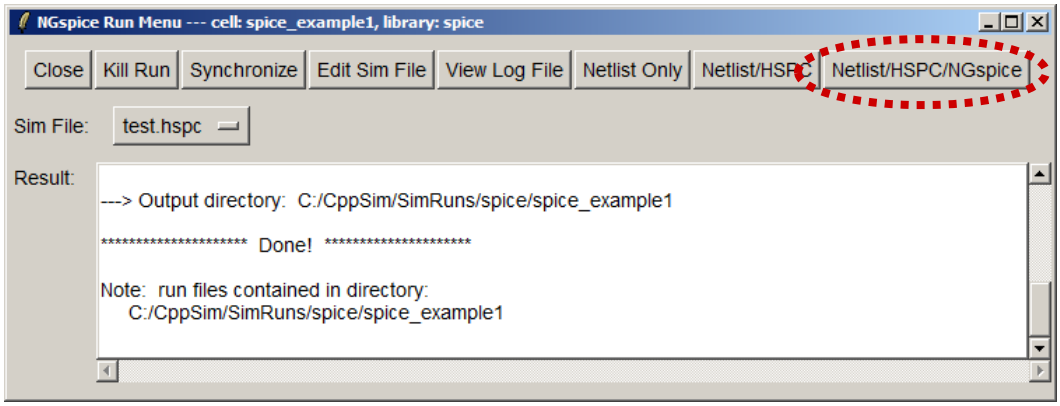
The previous sections illustrated the steps involved in performing transient simulations with **NGspice**. Continuing with the above example, we will now focus on performing AC analysis through modification of the **simrun.hspc** simulation file.

- Continuing with the example of the previous section, if you closed the **Emacs** window containing the **test.hspc** file, then click on the **Edit Sim File** button within the **NGspice Run Menu** window. Make the following modifications to the **test.hspc** file within the **Emacs** window (note that the final version of the file is shown below):
    - Uncomment the line: **Vin in 0 0.5 ac=1**
        - This specifies an input voltage source to the circuit with AC magnitude of 1
    - Comment the line: **.tran 10p 1u**
        - This removes transient analysis from being performed
    - Uncomment the line: **.ac dec 10e3 100k 100Meg**
        - This specifies that AC analysis should take place for 10,000 frequency points spaced logarithmically from 100kHz to 100MHz
    - Comment the line: **> timing 0.0n 0.2n [1/10e6] 0 vsupply**
        - This removes the HSPC timing information for the input signal that was used for the transient simulation
    - Comment the line: **> input in [set 0 0 1 1 1 0 1 1 0 0 0 1 0 R]**
        - This removes the HSPC input signal that was used for the transient simulation

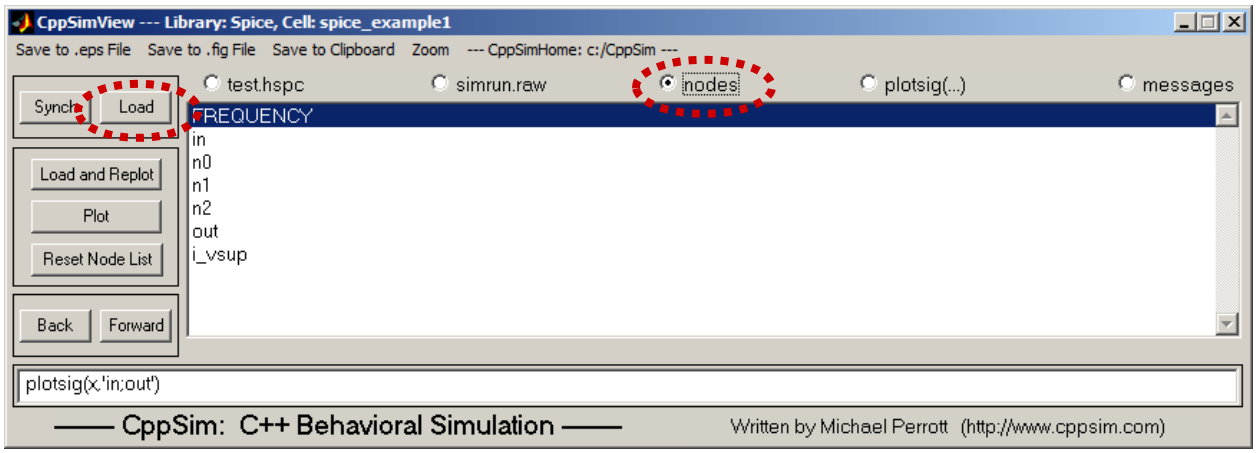
  Completion of the above commands yields the file shown below. Be sure to save this file.



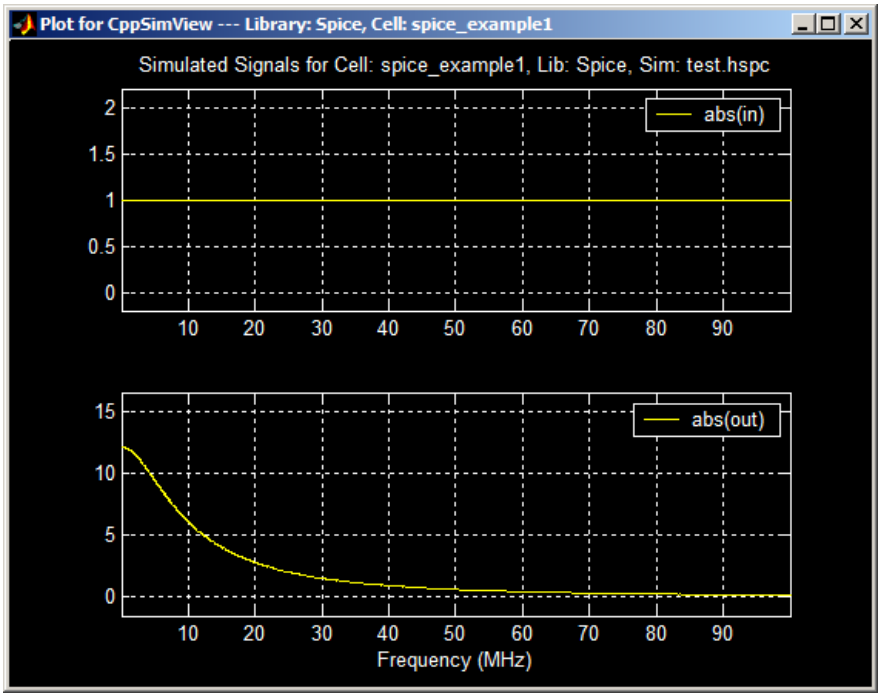
- Now click on **Netlist/HSPC/NGspice** in the **NGspice Run Menu** as shown below:



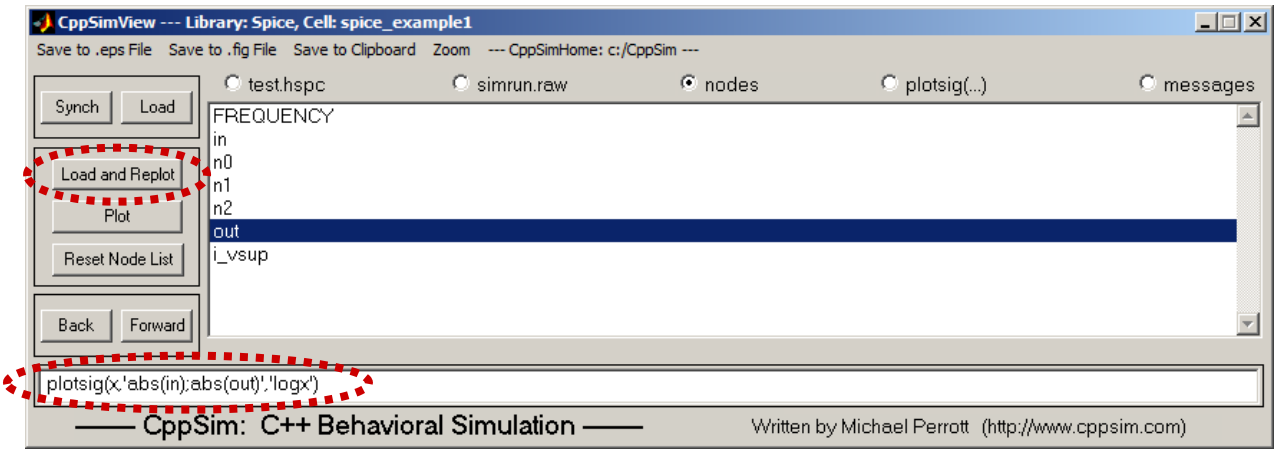
- If it is already running, exit out of **CppSimView**. Then restart **CppSimView** by clicking on the **CppSimView** icon on the Windows desktop. Click on either the **nodes** radio button (circled below in red) or on the **Load** button (whichever is appropriate) to load in the signals from this new run in **CppSimView**. You should see a list of node names as shown in the figure below.



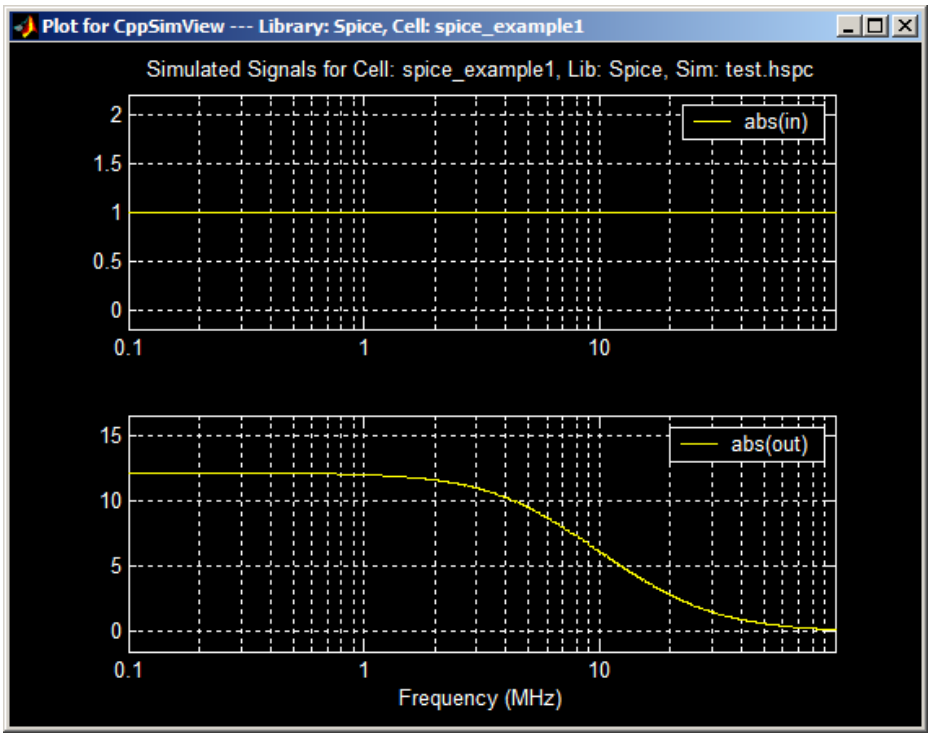
- In the **CppSimView** window, double-click on nodes **in** and **out**. You should see a figure similar to what is shown below (note that the **Zoom** button in the main **CppSimView** window shown above toggles on and off the buttons in the plot window shown below). Note that **CppSimView** automatically takes the magnitude of each signal (i.e., **abs(in)** and **abs(out)**) since they are complex signals in frequency domain. Note also that the magnitude of **in** is equal to 1 as defined in the Simulation File above (i.e., Vin in 0 0.5 **ac=1**)



- For AC analysis, it is often easier to view the frequency axis in a logarithmic fashion, and to plot magnitudes in terms of dB. In general, you can changing plotting functions from **abs(.)** to **ph(.)** (i.e., phase of the signal) or **db(.)** (i.e., magnitude in dB) by simply overwriting their value in the **CppSimView** plot expression section at the very bottom of the main window of **CppSimView**. To view the frequency axis in logarithmic fashion, you need to add a 'logx' specification. As an example, modify the **CppSimView** plot expression as shown below, and then push on the **Load and Replot** button to update the plot.



- After you complete the commands above, you should see a plot window similar to what is shown below.



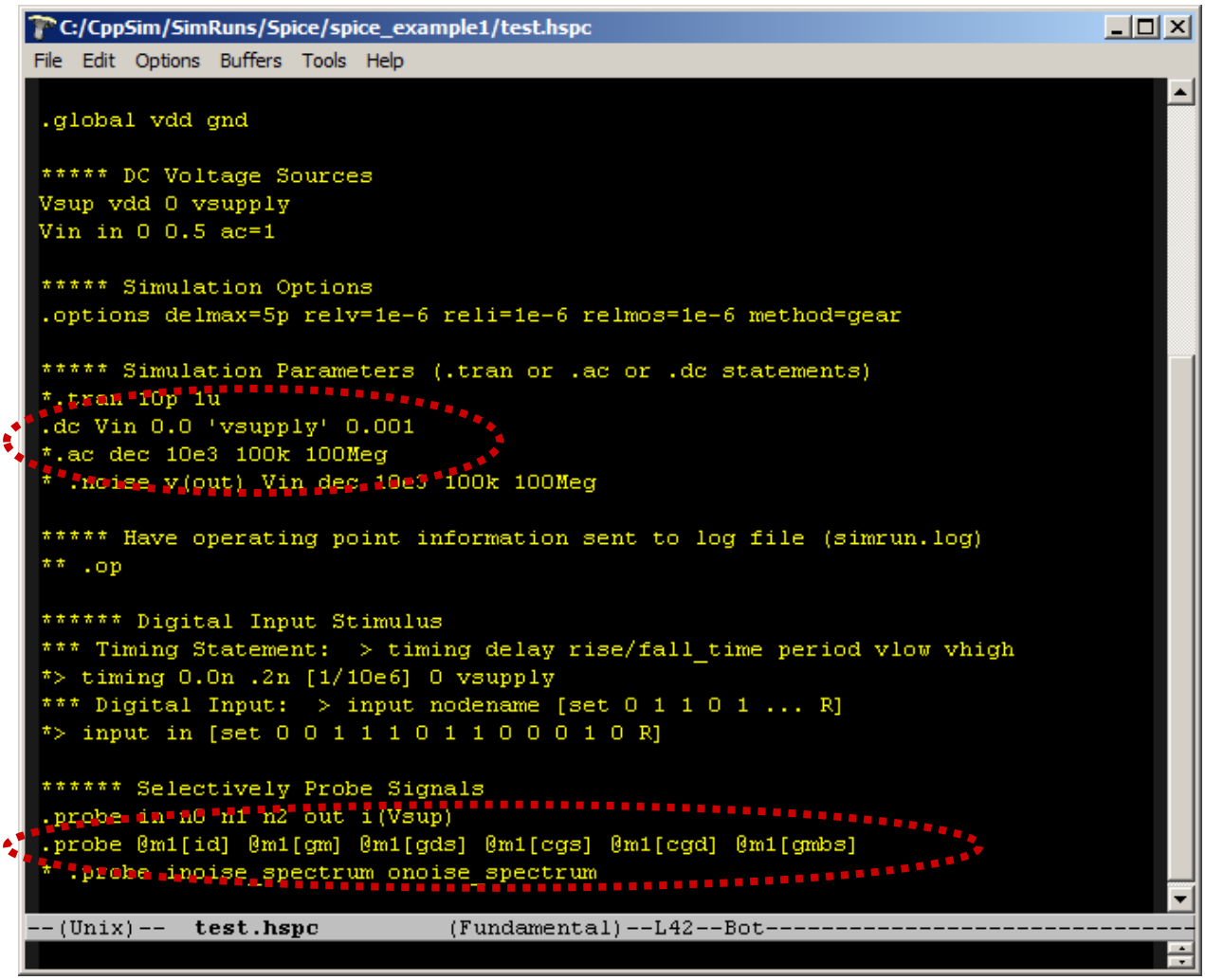
E. DC Analysis Example

The previous section illustrated the steps involved in performing AC analysis with **NGspice**. Continuing with the above example, we will now focus on performing DC analysis through modification of the **simrun.hspc** simulation file.

- Continuing with the example of the previous section, if you closed the **Emacs** window containing the **test.hspc** file, then click on the **Edit Sim File** button within the **NGspice Run Menu** window. Make the following modifications to the **test.hspc** file within the **Emacs** window (note that the final version of the file is shown below):

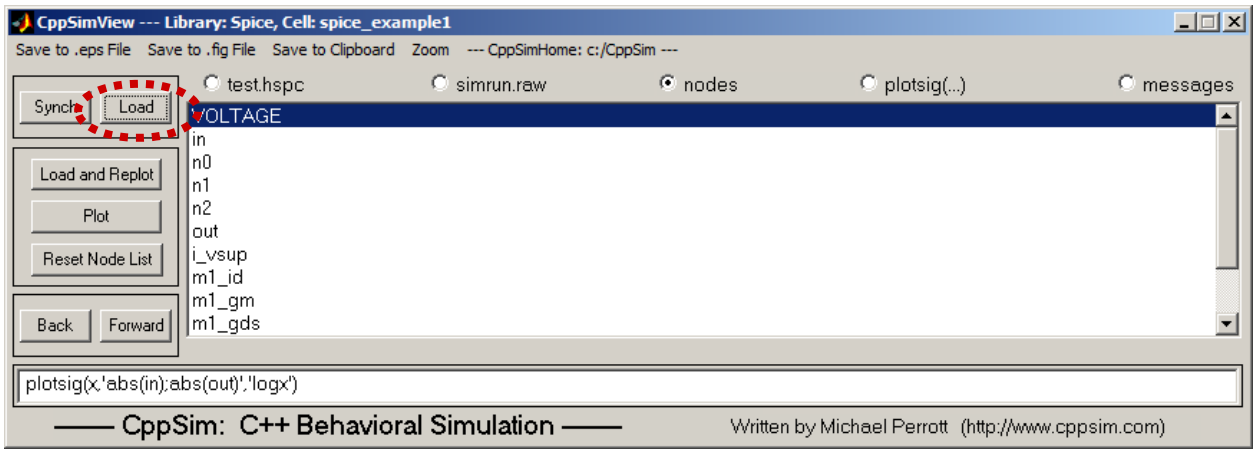
- Comment the line: **.ac dec 10e3 100k 100Meg**
  - This removes AC analysis from being performed
- Uncomment the line: **.dc Vin 0.0 'vsupply' 0.001**
  - This specifies that DC analysis should be performed by sweeping the **Vin** supply from 0 to **vsupply** (specified as 1.3V in the **.param vsupply=1.3** statement in the **test.hspc** file) in increments of 0.001V.
- Uncomment the line: **. probe @m1[id] @m1[gm] @m1[gds] @m1[cgs] @m1[cgd] @m1[gmbs]**
  - This specifies that additional signals should be saved for viewing in **CppSimView** which correspond to various parameters of CMOS transistor M1.

Completion of the above commands yields the file shown below. Be sure to save this file.

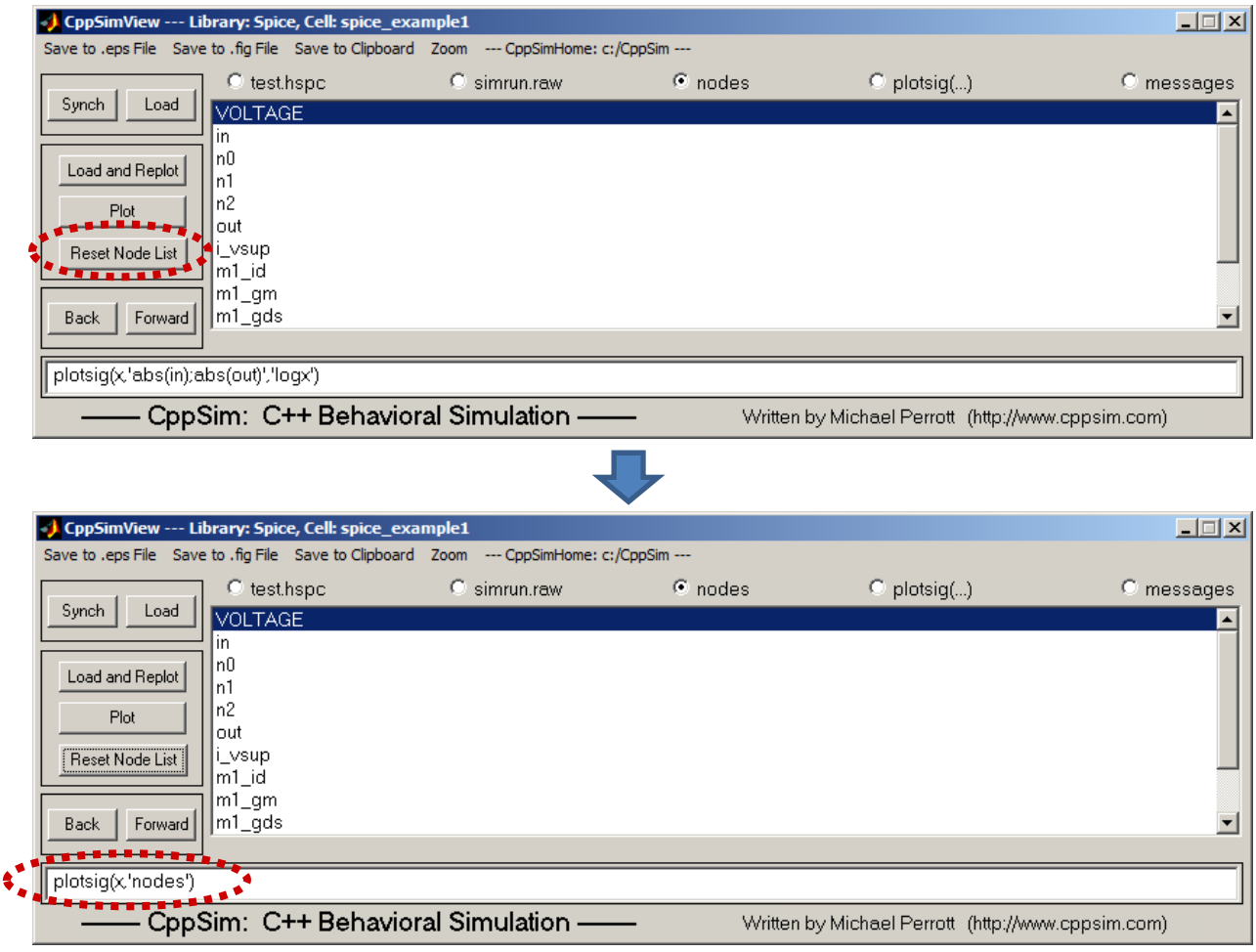


- Now click on **Netlist/HSPC/NGspice** in the **NGspice Run Menu** to run the **NGspice** simulation.

- Assuming CppSimView is still open from the previous section, click on the **Load** button to load in the signals from this new run in **CppSimView**. You should see a list of node names as shown in the figure below.

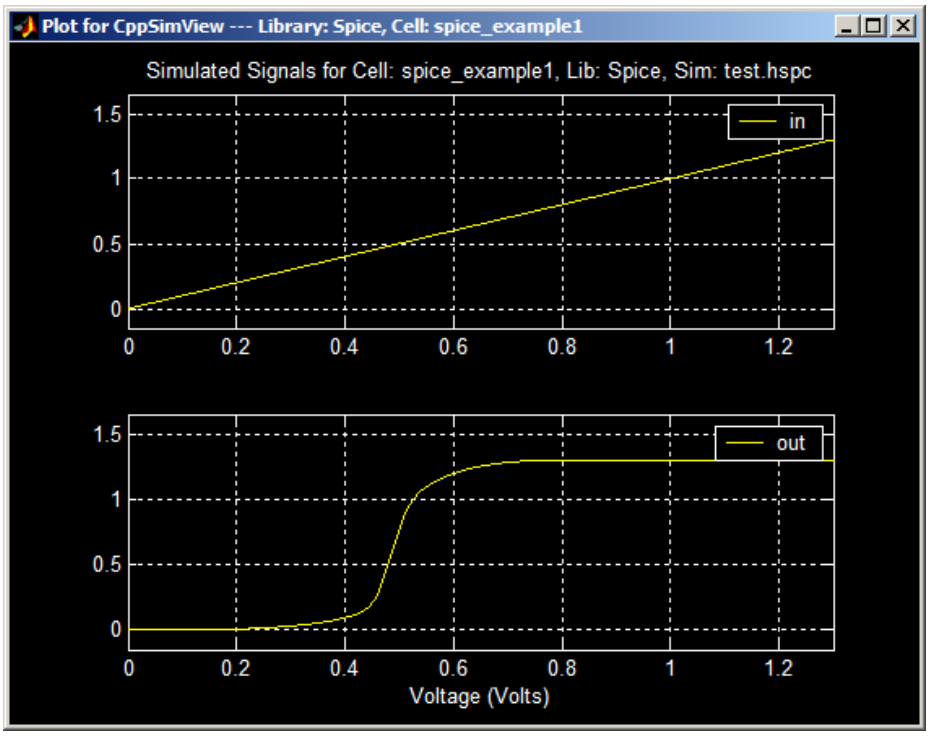


- In the **CppSimView** window, click on the **Plot** button indicated above. Unfortunately, the signals are plotted in db(.) formate with logarithmic x-axis, which is not appropriate for DC analysis. Click on the **Reset Node List** button to reset the plotting function as shown below.



- In the **CppSimView** window, double-click on nodes **in** and **out**. You should see a figure similar to what is shown below (note that the **Zoom** button in the main **CppSimView** window shown above toggles on and off the buttons in the plot window shown below). The x-axis

corresponds to the voltage sweep value, and the plots reveal how both **in** and **out** vary as a function of this voltage sweep. You might consider looking at other signals, as well.



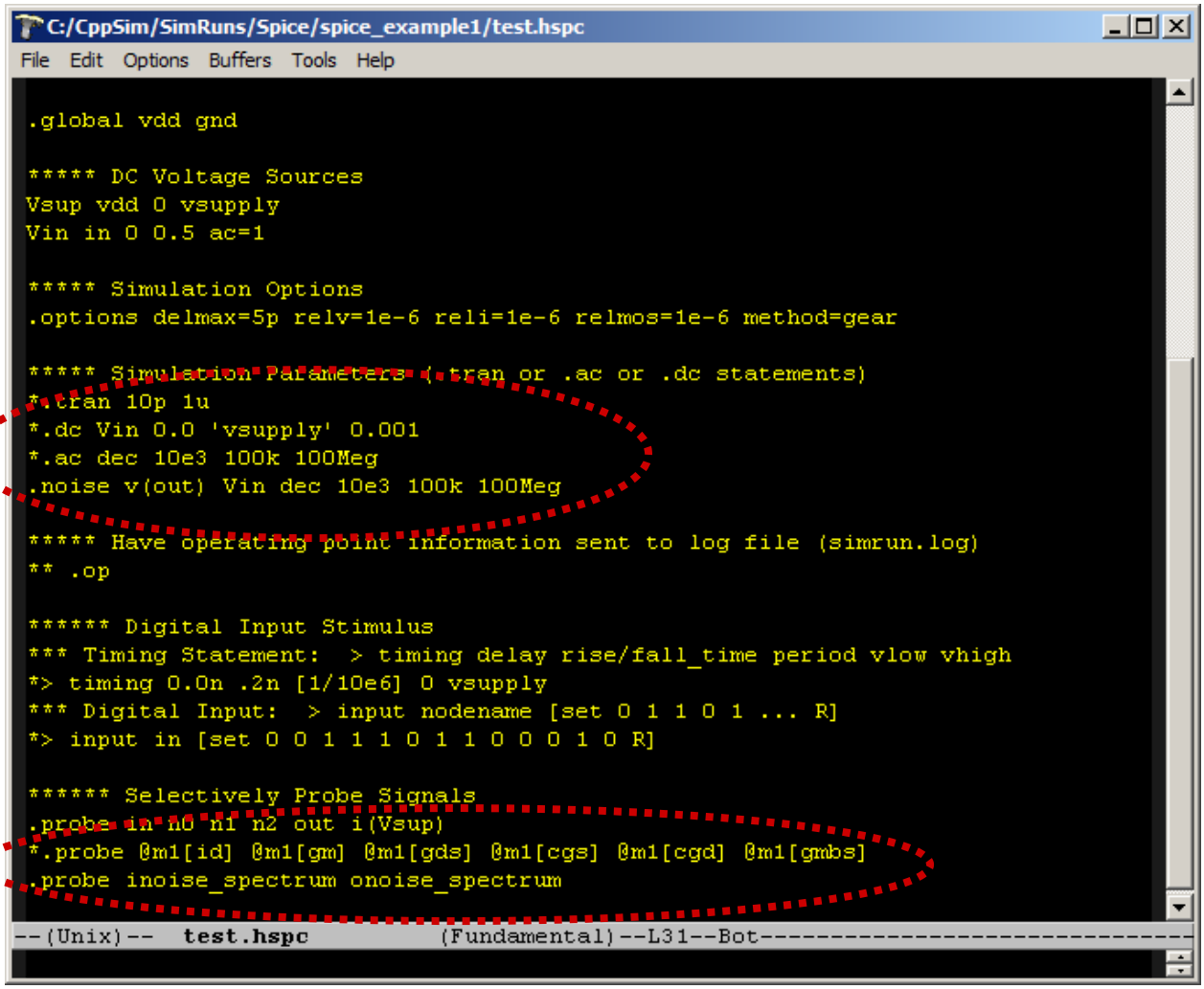
F. Noise Analysis Example

The previous section illustrated the steps involved in performing AC analysis with **NGspice**. Continuing with the above example, we will now focus on performing noise analysis through modification of the **simrun.hspc** simulation file.

- Continuing with the example of the previous section, if you closed the **Emacs** window containing the **test.hspc** file, then click on the **Edit Sim File** button within the **NGspice Run Menu** window. Make the following modifications to the **test.hspc** file within the **Emacs** window (note that the final version of the file is shown below):
  - Comment the line: **.dc Vin 0.0 'vsupply' 0.001**
    - This removes DC analysis from being performed
  - Uncomment the line: **.noise v(out) Vin dec 10e3 100k 100Meg**
    - This specifies that noise analysis should be performed by calculating the voltage noise spectrum (i.e., V^2/Hz) at the **out** node for 10,000 frequency points spaced logarithmically from 100kHz to 100MHz. The input-referred voltage noise spectrum should also be calculated as referenced to the output of signal source **Vin**. Note that **Vin** must be a signal source and not a node, whereas **out** corresponds to a node and not a signal source.
  - Comment the line: **. probe @m1[id] @m1[gm] @m1[gds] @m1[cgs] @m1[cgd] @m1[gmbs]**
    - This removes plotting for these signals.
  - Uncomment the line: **.probe inoise_spectrum onoise_spectrum**

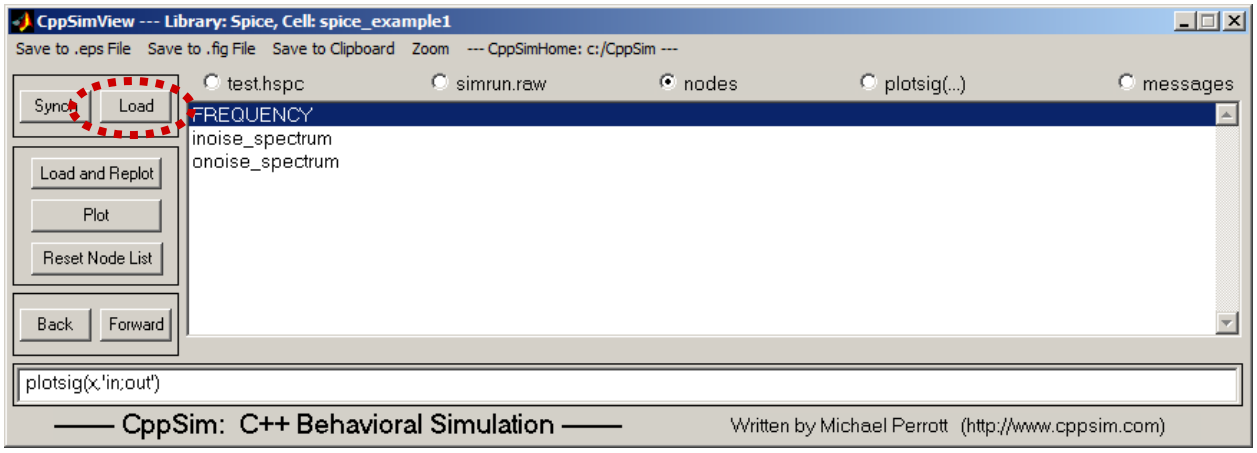
- This specifies the input-referred (at the output of signal source **Vin**) and output-referred (at node **out**) noise spectrums should be saved for plotting.

Completion of the above commands yields the file shown below. Be sure to save this file.

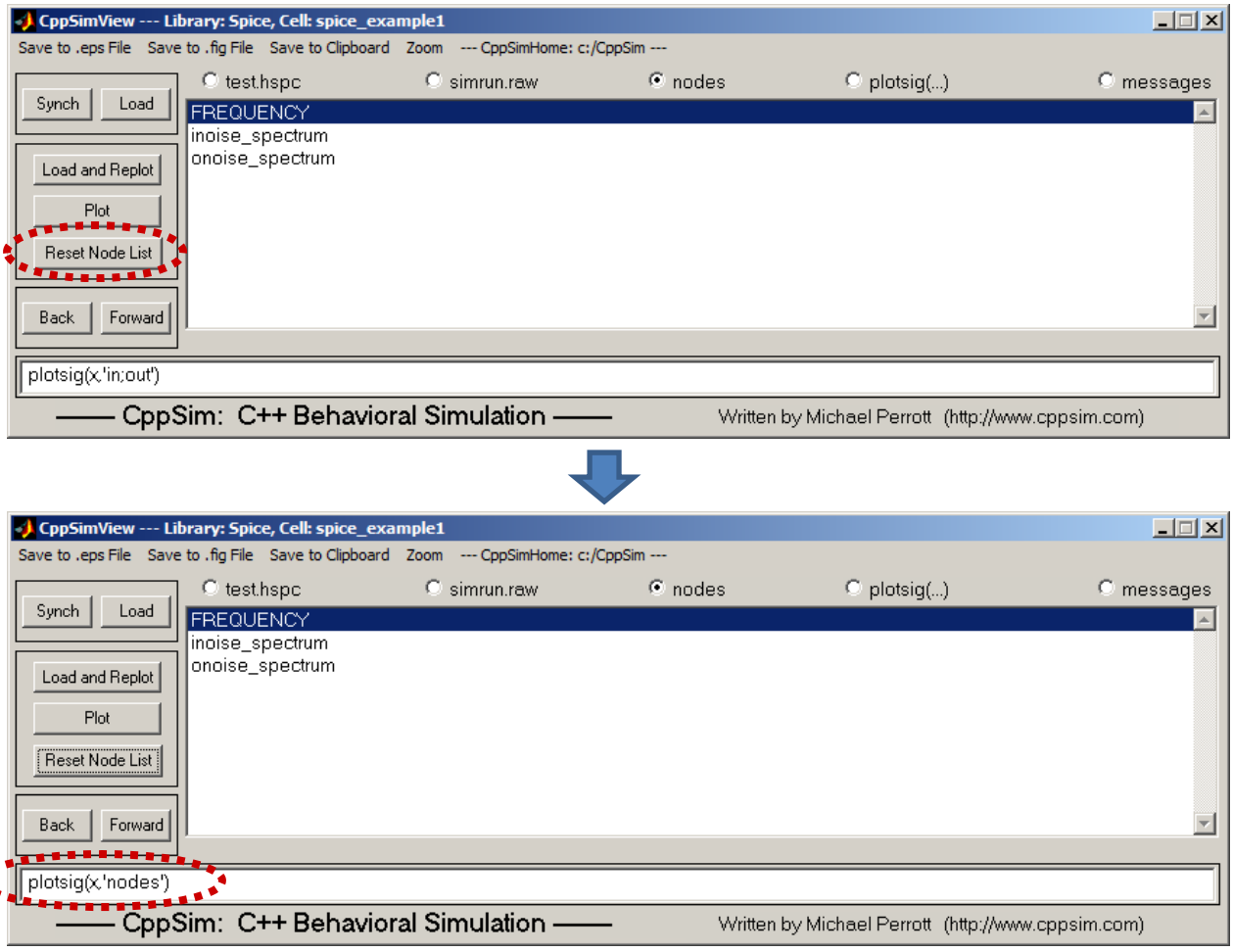


- Now click on **Netlist/HSPC/NGspice** in the **NGspice Run Menu** to run the **NGspice** simulation.

- Assuming CppSimView is still open from the previous section, click on the **Load** button to load in the signals from this new run in **CppSimView**. You should see a list of node names as shown in the figure below.

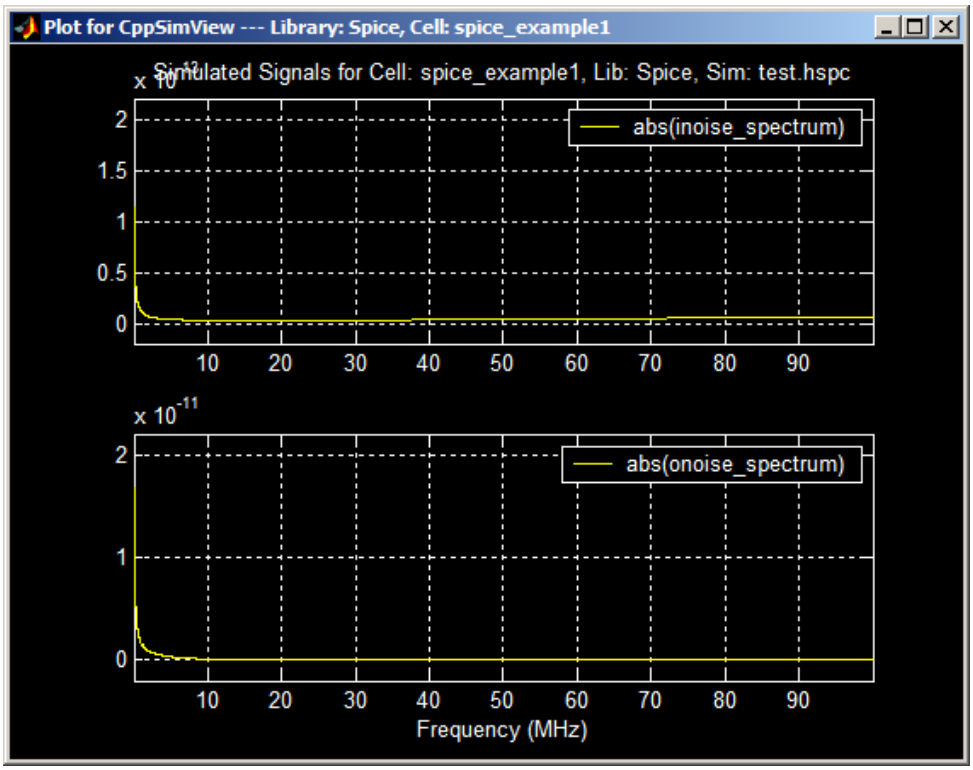


- In the **CppSimView** window, click on the **Plot** button indicated above. Unfortunately, an error occurs since the signals specified in the plot expression are no longer included for plotting (note that only **inoise_spectrum** and **onoise_spectrum** are valid signals for noise analysis with **NGspice**). As such, click on the **Reset Node List** button to reset the plotting function as shown below.

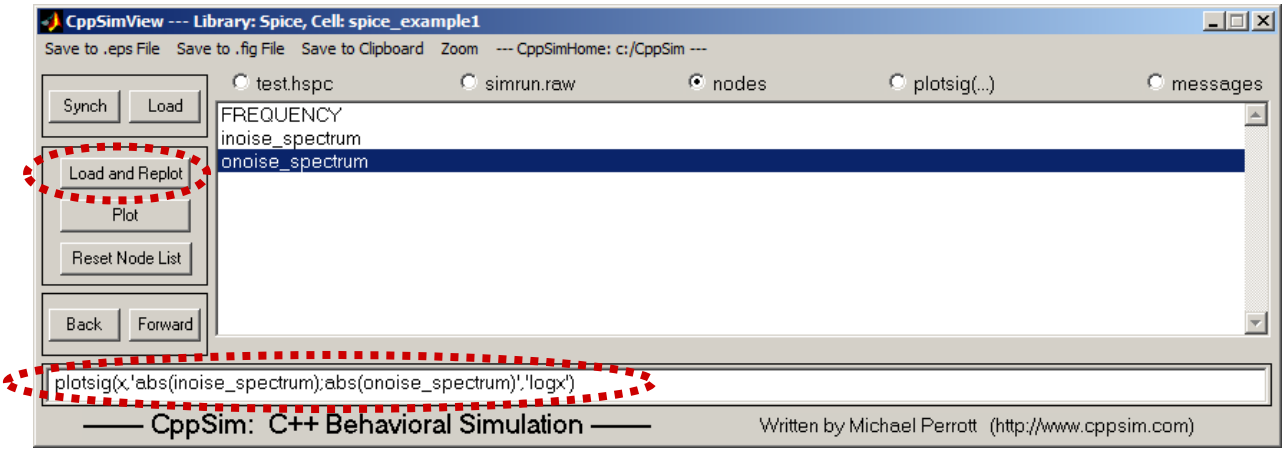


- In the **CppSimView** window, double-click on nodes **inoise_spectrum** and **onoise_spectrum**. You should see a figure similar to what is shown below (note that the **Zoom** button in the main

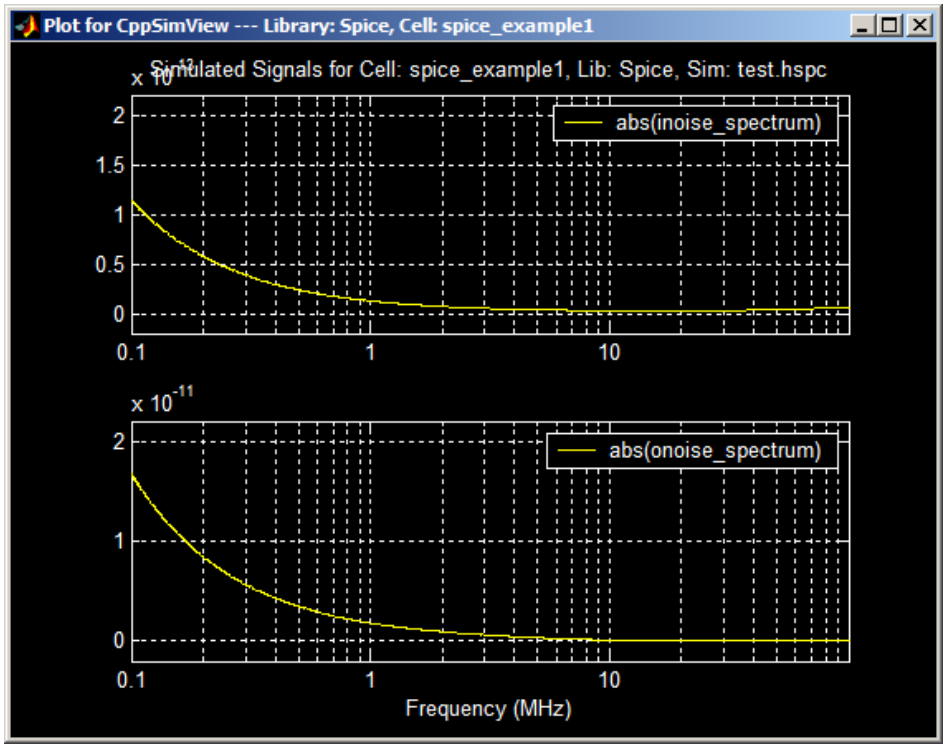
CppSimView window shown above toggles on and off the buttons in the plot window shown below).



- Just as for AC analysis, it is often easier to view the frequency axis in a logarithmic fashion for observing noise spectra. To view the frequency axis in logarithmic fashion, you need to add a 'logx' specification. As an example, modify the **CppSimView** plot expression as shown below, and then push on the **Load and Replot** button to update the plot.



- After you complete the commands above, you should see a plot window similar to what is shown below.



Using Matlab or Octave with NGspice

While using the NGspice Run Menu and CppSimView is a convenient interface for beginners, more advanced users may want to consider running their simulations and doing post-processing directly in Matlab or Octave. To use NGspice within Matlab or Octave, users simply need to add the Hspice Toolbox commands (which come with the standard CppSim installation) to the Matlab or Octave path. This operation is performed by typing the following in the Matlab or Octave command window

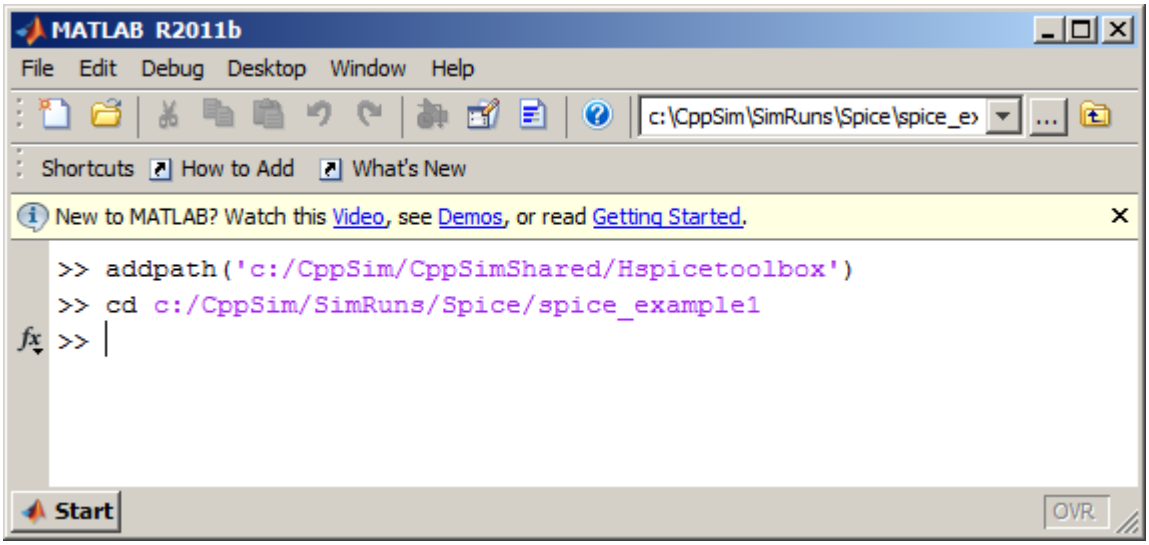
addpath('c:/CppSim/CppSimShared/HspiceToolbox')

Note that c:/CppSim should be replaced by the actual path you chose for CppSim during the installation.

A. Basic Operations

- As an example of running NGspice in Matlab, go to the simulation directory for the cell **spice_example1** by typing (in the Matlab command window):

cd c:/CppSim/SimRuns/Spice/spice_example1

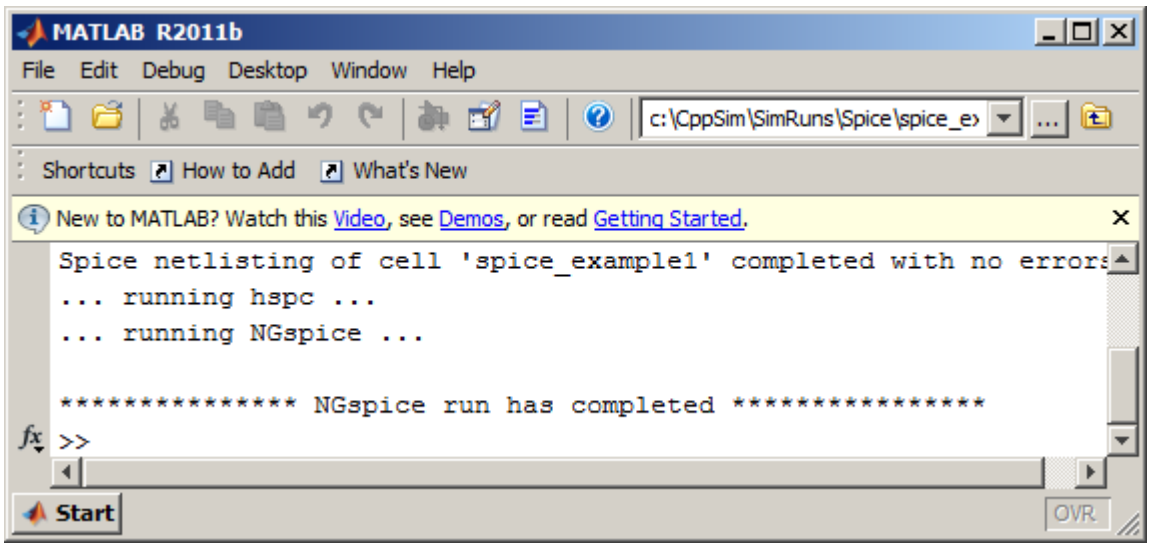


Again, you must substitute the proper path that you chose for CppSim in place of c:/CppSim. If you type **ls** at the Matlab prompt, you will see the many files produced by previous simulations. The simulation file, **test.hspc**, should and must be present in order for the steps that follow to work.

Once you are in the above directory, type

ngsim

at the Matlab prompt – this will run **HPSC** and then **NGspice** by default on the **test.hspc** file located in the current directory. The **ngsim** script will use the current directory information to determine the name of the cell and library (the current directory is the cell name (i.e., **spice_example1**), and the next directory up is the library name (i.e., **spice**)) and then use this information to automatically netlist the Sue2 cell and then run the simulation. When completed, the Matlab command window should appear similar to the following figure:



(Note: if one desires to run **NGspice** on a different simulation file, such as **test2.hspc** for instance, then type the following command at the Matlab prompt: **ngsim test2.hspc)**

Once the run has completed, load the signals in file **simrun.raw** into Matlab by typing

x = loadsig('simrun.raw');

You can then view the signals contained within this file by typing

lssig(x);

Finally, plot the signals **inoise_spectrum** and **onoise_spectrum** by typing

plotsig(x,'abs(inoise_spectrum);abs(onoise_spectrum)', 'logx');

- A key advantage of using Matlab or Octave is the greatly increased flexibility it offers for doing post-processing. In particular, one can create Matlab scripts to load in NGspice output files (i.e., simrun.raw) and then perform sophisticated processing on the signals they contain. To do so, one needs to turn signals embedded within NGspice output files into Matlab signals. This is achieved for signal **inoise_spectrum** in the above example by typing

  inoise = evalsig(x,'inoise_spectrum');

  in Matlab. The above operation allows one to now directly access the data values of **inoise_spectrum** in Matlab. For instance, to view the first ten samples of **inoise_spectrum**, simply type

  inoise_spectrum(1:10)

  in Matlab.

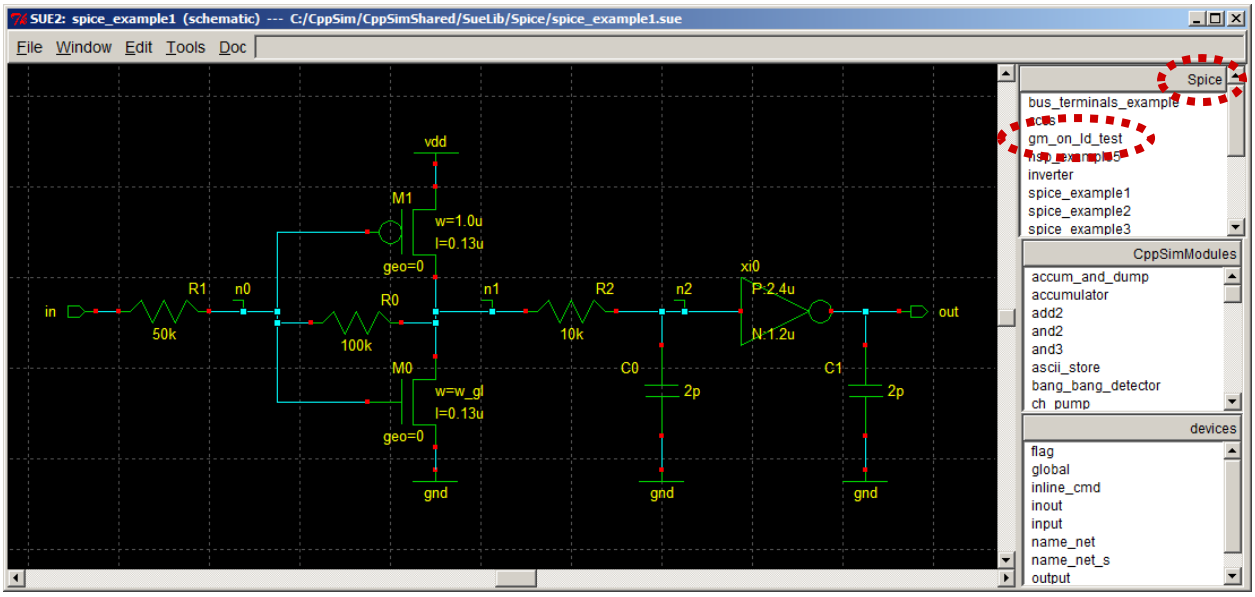
  It is worthwhile to examine the Hspice Toolbox manual, which is provided as the PDF document:
  - **c:/CppSim/CppSimShared/Doc/hspice_toolbox.pdf**

  for more information on the Matlab commands it offers related to viewing and post-processing.

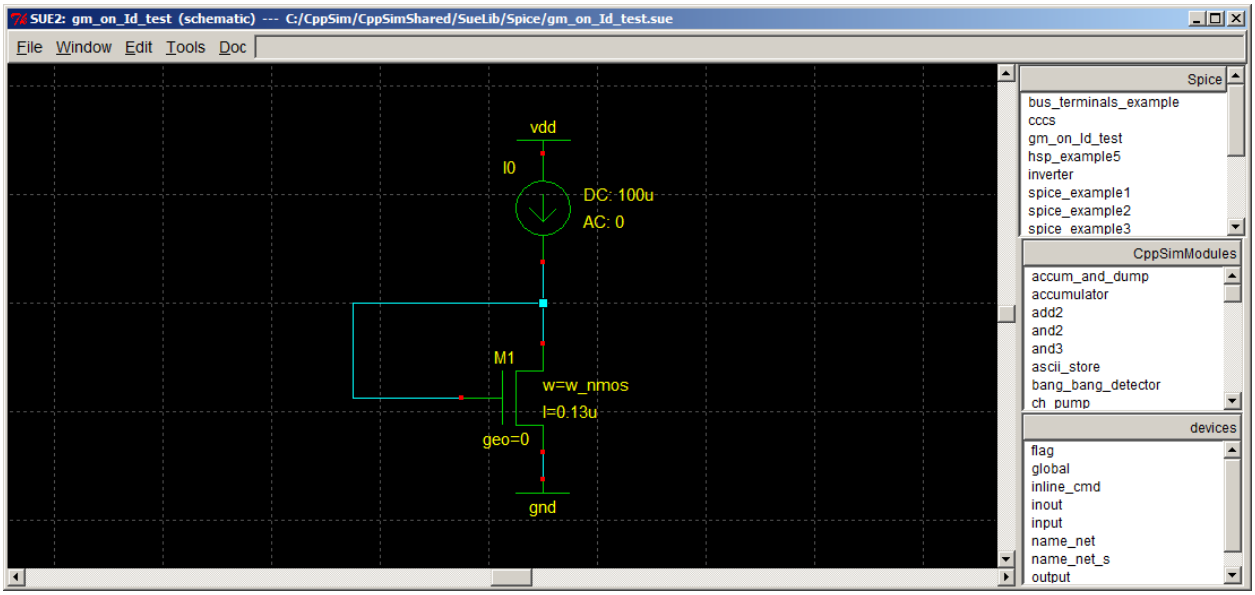
B. Running Parameter Sweeps using Matlab/Octave Scripting

We now consider an example of using a Matlab script to perform a parameter sweep by performing multiple Ngspice simulations from the script

- As a first step, go to the cell **gm_on_id_test** by clicking on its schematic from the **schematic listbox** menu in Sue2 as indicated below. Note that you need to make sure that the library for the schematic listbox is Spice as shown in the figure.



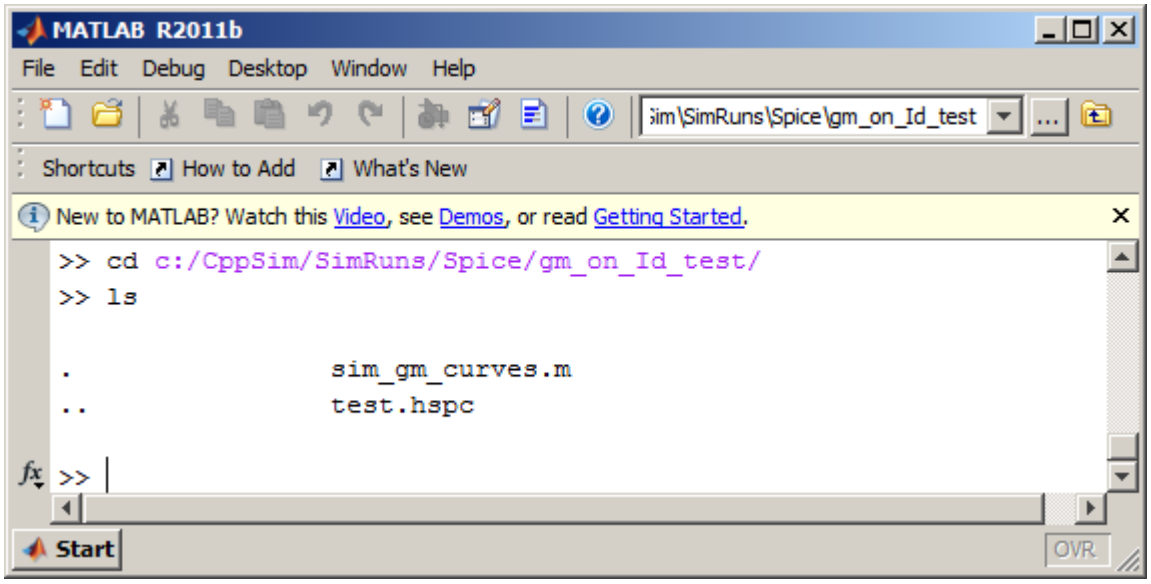
- You should now see the schematic shown below, which contains a diode-connected NMOS transistor fed by a current source.



- Now go to the simulation directory for the cell **gm_on_Id_test** by typing in the Matlab command window:

cd c:/CppSim/SimRuns/Spice/gm_on_Id_test

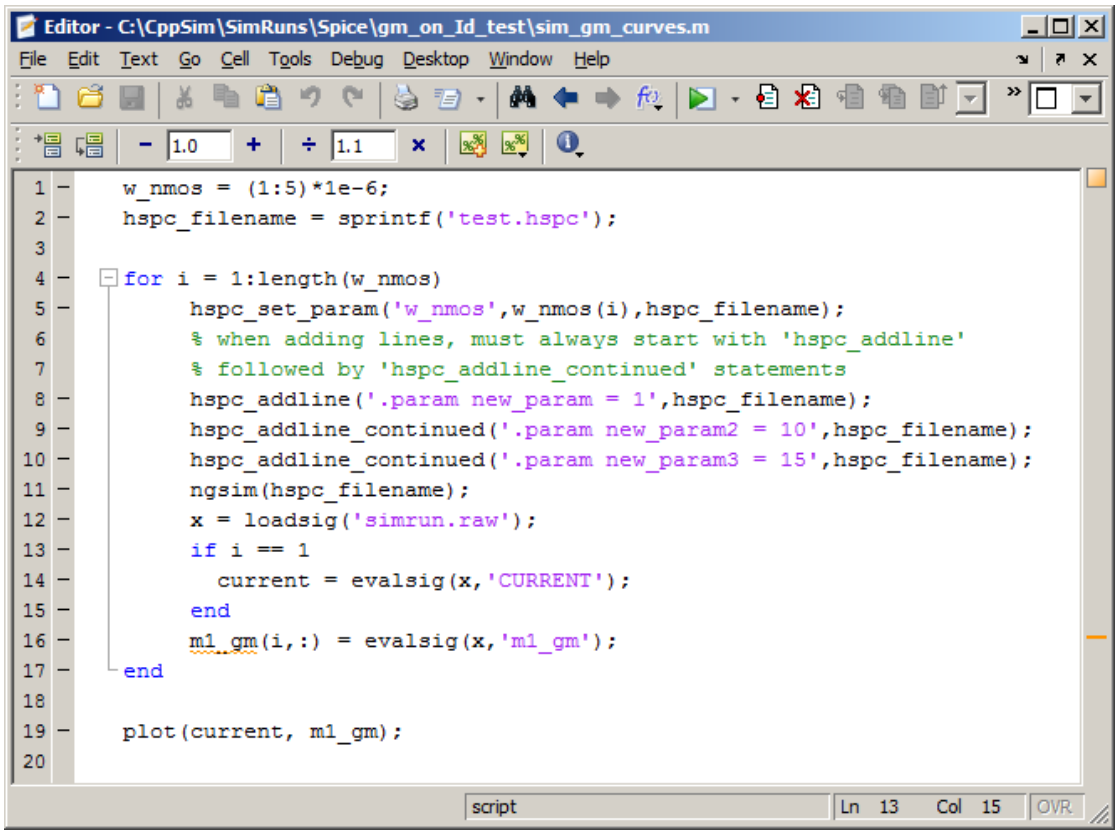
After typing ‘ls’ in the Matlab command window, you should see the following two files:



- We can examine the **sim_gm_curves.m** script by typing in the Matlab command window:

edit sim_gm_curves.m

The resulting editor window should appear as follows:



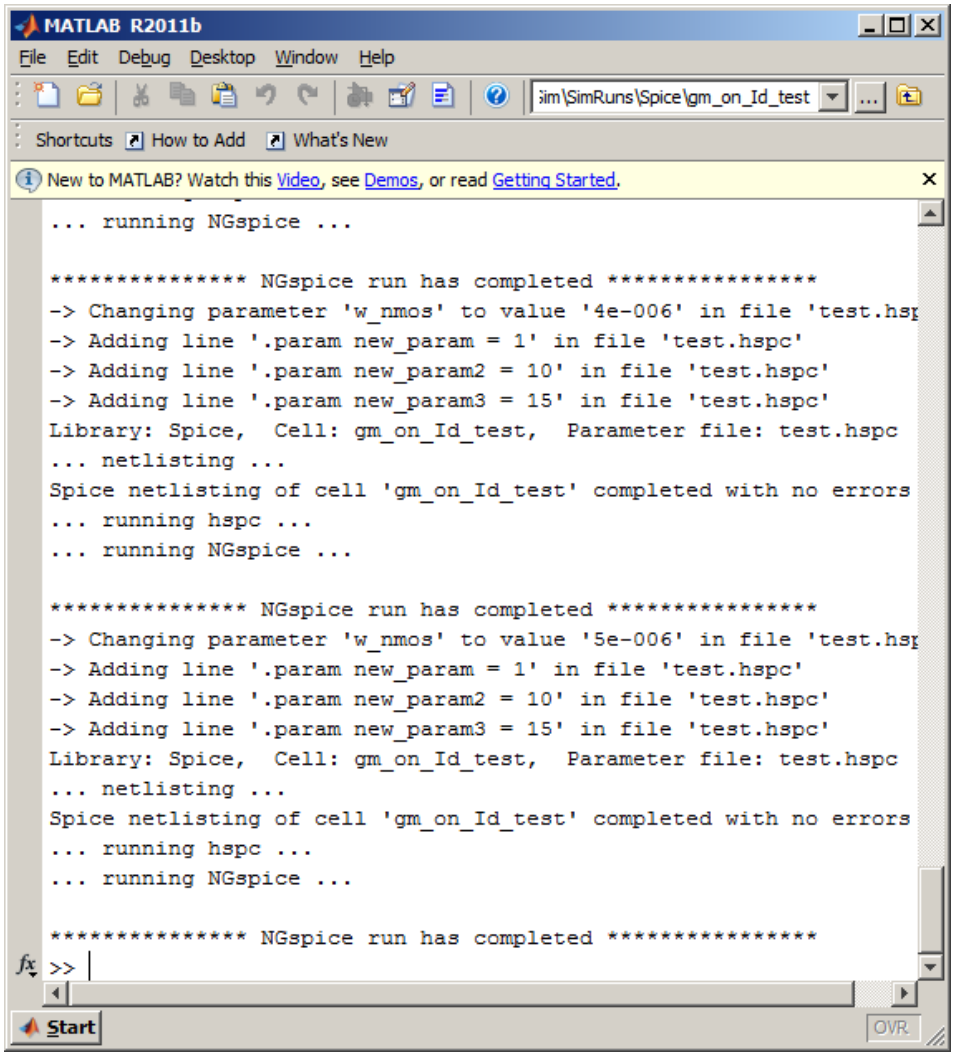
Key commands seen in the above script include:

- **ngsim(hspc_filename)**: runs Ngspice with the HSPC file indicated by **hspc_filename**.
- **hspc_set_param(param_name, new_value, hspc_filename)**: this command searches the HSPC file specified by **hspc_filename** for a parameter with name **param_name** and changes its value to **new_value**. For the above example, we change parameter **w_nmos** each time before running a new Ngpice simulation using the **ngsim** command.
- **hspc_addline(new_line,hspc_filename)**: this command adds a new line specified by **new_line** to the file indicated by **hspc_filename**. For the above example, we simply add the line **.param new_param = 1** to the **test.hspc** file. This is not useful for this example, but illustrates how the command is used.
- **hspc_addline_continued(new_line,hspc_filename)**: this command adds additional lines beyond the new line created by the **hspc_addline** command. As many such lines can added as desired, but the first one must always be **hspc_addline** with the rest being **hspc_addline_continued**. Again, the lines added in this example are not useful here, but illustrate how the command is used.

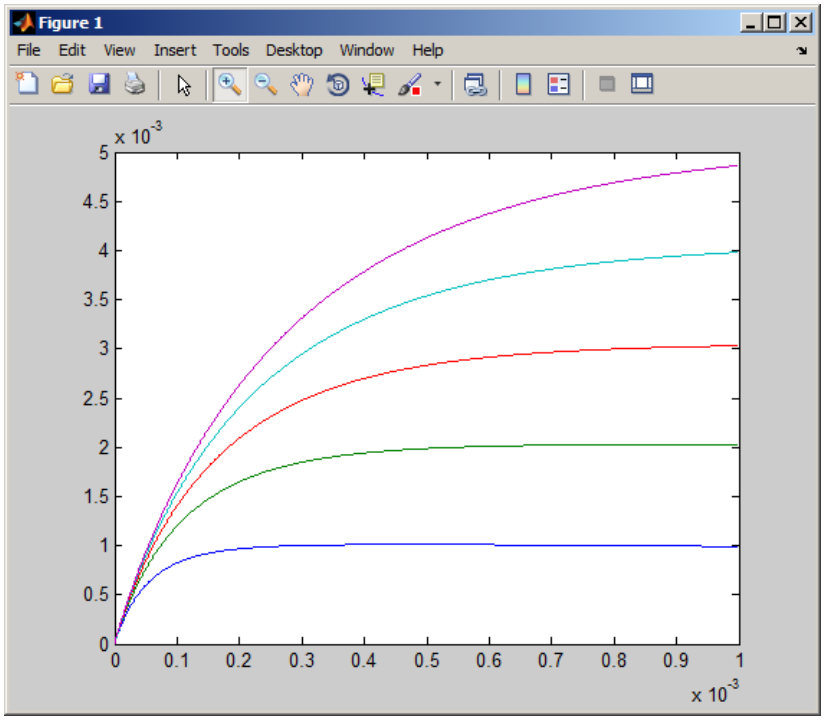
- Run the **sim_gm_curves.m** script by typing in the Matlab command window:

clf; sim_gm_curves

where **clf** clears the Matlab plot window and **sim_gm_curves** runs the script. You should see several Ngspice simulation windows pop up and the Matlab command line will appear as follows when the simulations have concluded:



- Further, the Matlab plot window should show the results of the 5 simulations as shown below:



Using Python with Ngspice

To use Ngspice within Python, you simply need to import the Ngspice Data module (which comes with the standard CppSim installation) by including the following lines in a given Python script:

```
# import ngspicedata module
import os
import sys
if sys.platform == 'darwin':
        home_dir = os.getenv("HOME")
        sys.path.append(home_dir + '/CppSim/CppSimShared/Python')
else:
        cppsimsharedhome = os.getenv("CPPSIMSHAREDHOME")
        sys.path.append(cppsimsharedhome + '/Python')
from ngspicedata import *
```

The Ngspice Data module provides a class called **NgspiceData** to allow easy loading of simulation data into Python, a function called **ngsim()** to run Ngspice simulations within Python, and some supporting functions for running parameterized sweeps. While we will show some simple examples below, one should read the manual **CppSim and Ngspice Data Modules for Python** that is available in the **Doc** menu of Sue2 for further details.

For the Python examples below, it is highly recommended that you download and install the Express (i.e., free) version of the Enthought Canopy distribution of Python available at:

https://www.enthought.com/products/epd/free/

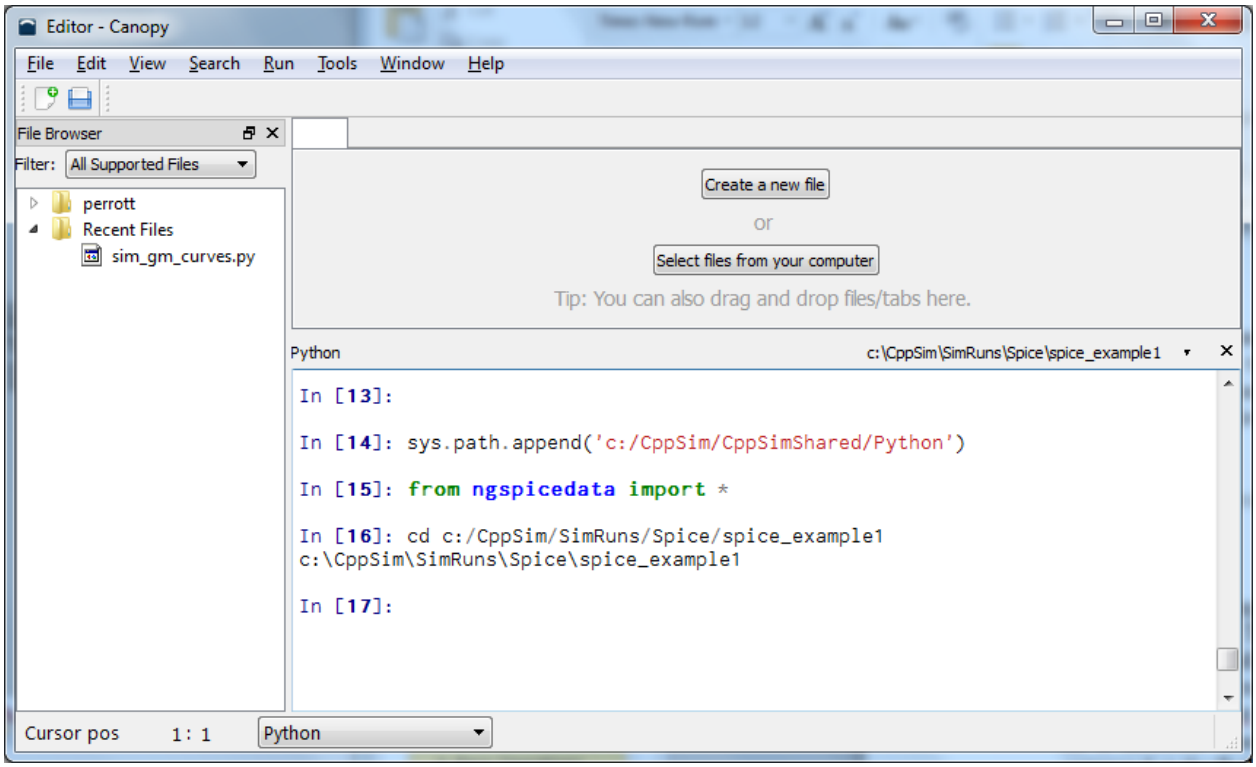
Note that for Windows platforms, you should download the 32-bit version of Canopy. For Mac platforms (assumed to be 64-bit), you should download the 64-bit version of Canopy. For Linux platforms, you should download the version that corresponds to your Linux operating system.

A. Basic Operations

- As an example of running Ngspice in Python, go to the simulation directory for the cell **spice_example1** by typing the following command in the Canopy Python Editor window (which we will refer to as the Python prompt):

  cd c:/CppSim/SimRuns/Spice/spice_example1

  For the above command, you must substitute the proper path for CppSim in place of c:/CppSim. If you type **ls** at the Python prompt, you may see various files produced by previous simulations. The simulation file, **test.hspc**, should and must be present in order for the steps that follow to work. Also, you must have imported the **ngspicedata** module as discussed above. The Canopy editor window below summarizes these operations:

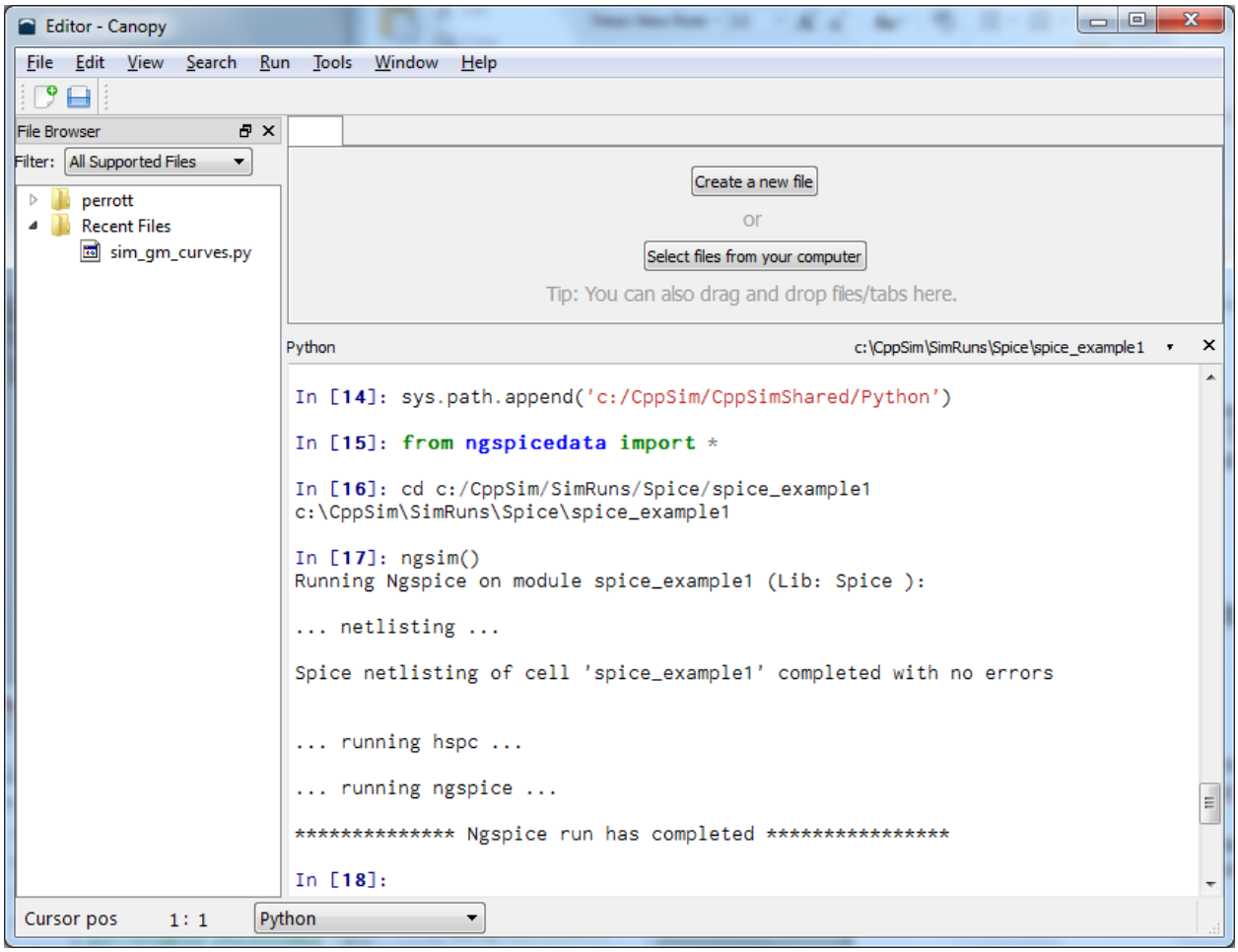


- Once you are in the above directory, type

  ngsim()

  at the Python prompt – this will run Ngspice by default on the **test.hspc** file located in the current directory. The **ngsim()** script will use the current directory information to determine the name of the cell and library (the current directory is the cell name (i.e., **spice_example1**), and the next directory up is the library name (i.e., **Spice**)) and then use this information to

automatically netlist the Sue2 cell and then run the simulation. The Canopy editor window displays the result of running **ngsim()** as shown below:



Note that if one desires to run Ngspice on a different simulation file, such as **test2.hspc** for instance, then type the following command at the Python prompt instead of the above:

ngsim('test2.hspc')

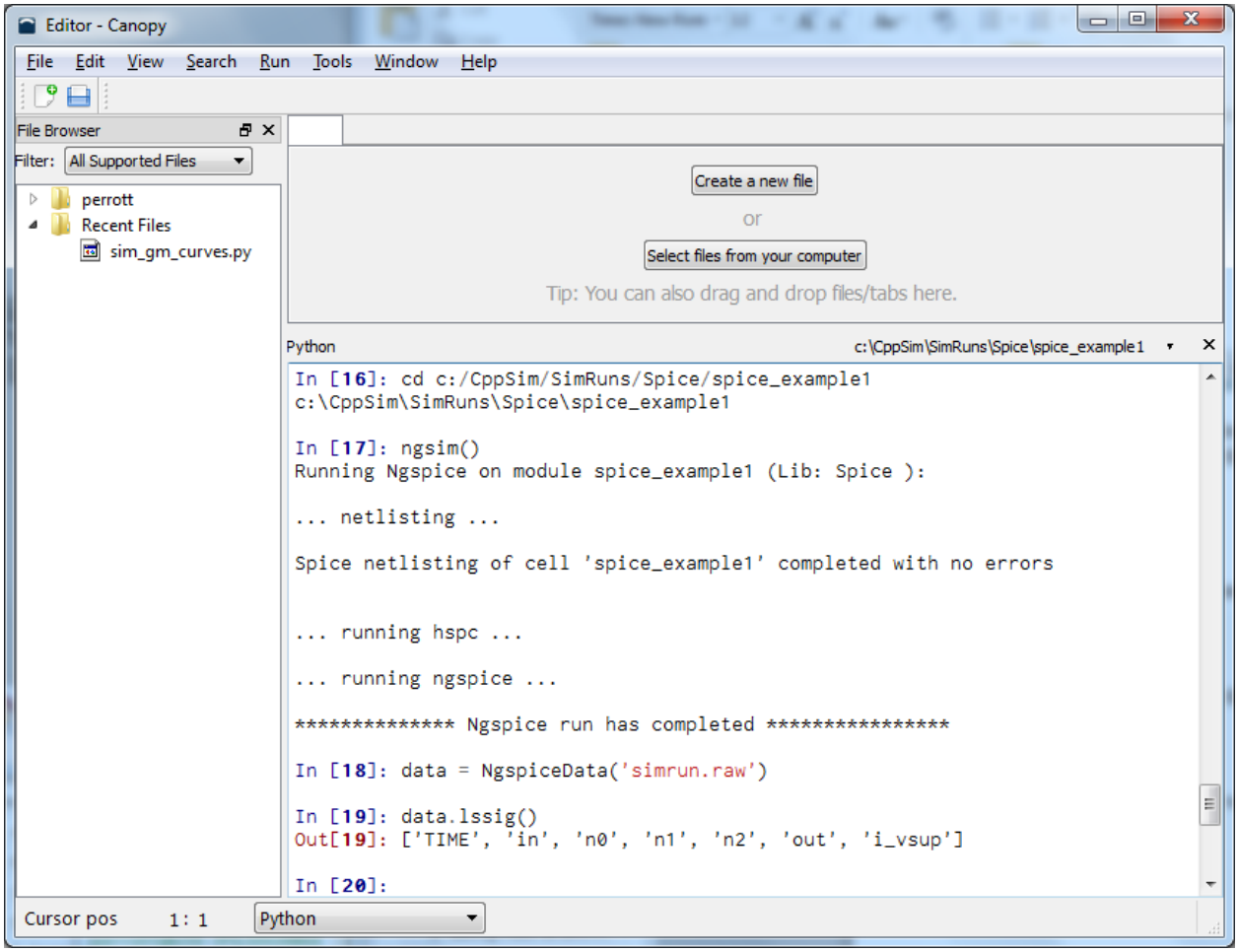
- Once the run has completed, load the signals in file **simrun.raw** into Python by typing

data = NgspiceData('simrun.raw')

You can then view the signal names contained within this file by typing

data.lssig()

The Canopy editor window displays the results of running these commands as shown below:



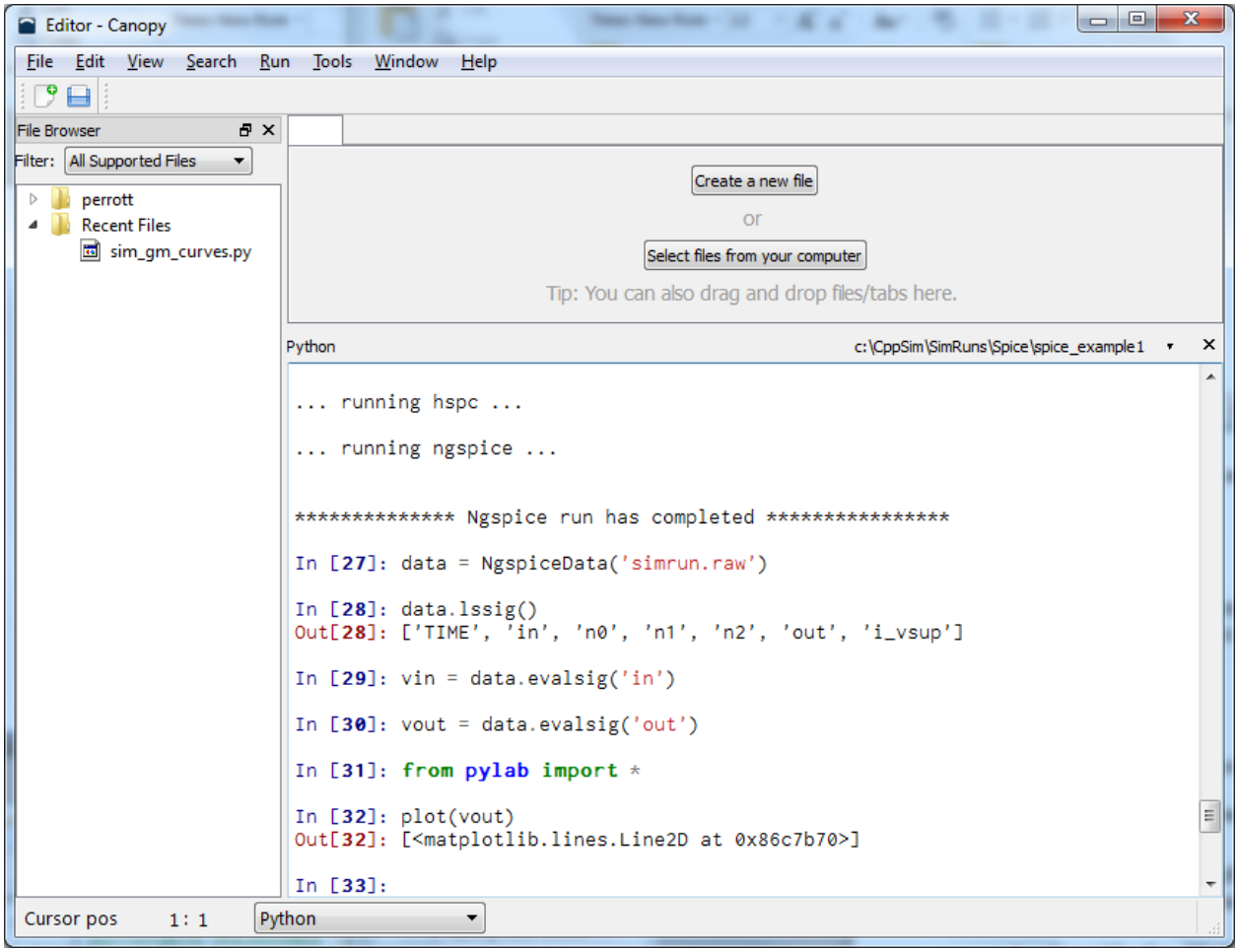
- The signals **in** and **out** are loaded into corresponding Python Numpy arrays as follows:

vin = data.evalsig('in')
vout = data.evalsig('out')

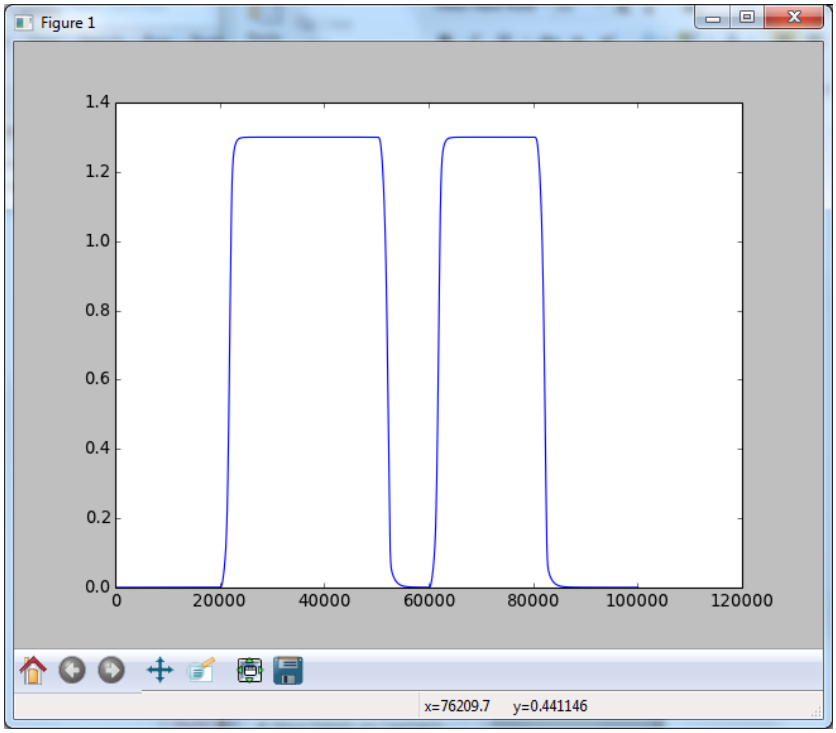
One can then perform post-processing or plotting of the above signals in Python as desired. As an example, one can plot the **out** signal by using the following commands:

from pylab import *
plot(vout)

These commands are also shown in the Canopy editor window below:



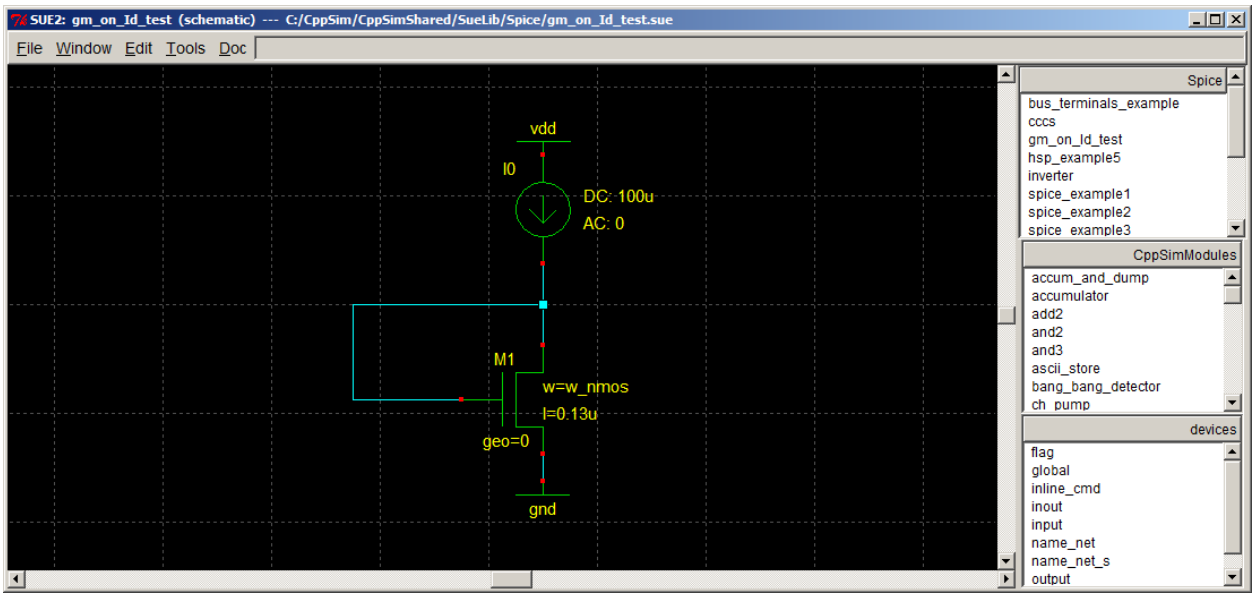
Also, the resulting plot is shown below:



B. Running Parameter Sweeps using Python Scripting

We now consider an example of using a Python script to perform a parameter sweep by performing multiple Ngspice simulations from the script

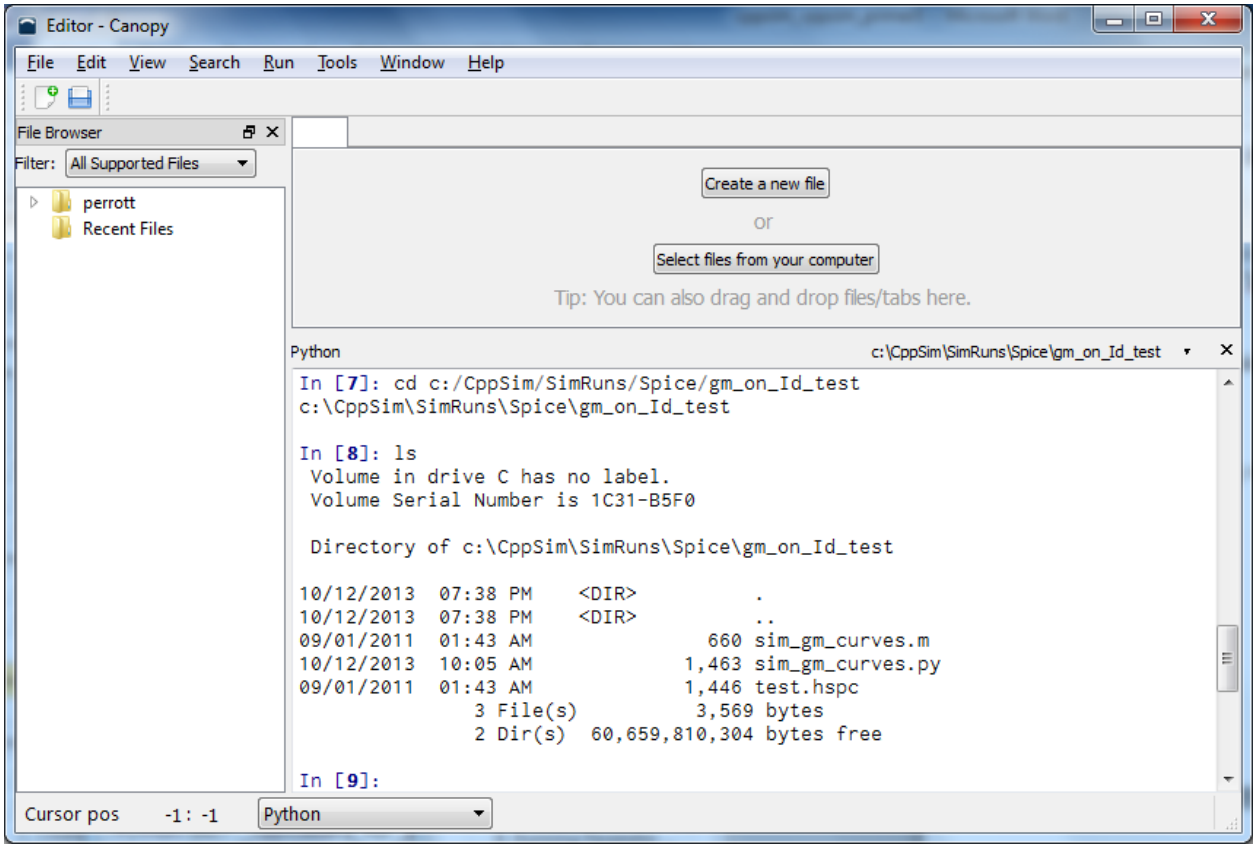
- As with the Matlab example in the previous section of this document, go to the cell **gm_on_id_test** by clicking on its schematic from the **schematic listbox** menu in Sue2 as indicated below. Note that you need to make sure that the library for the schematic listbox is **Spice** as shown in the figure. You should see the schematic shown below, which contains a diode-connected NMOS transistor fed by a current source.



- Now go to the simulation directory for the cell **gm_on_Id_test** by typing the following at the Python prompt:

cd c:/CppSim/SimRuns/Spice/gm_on_Id_test

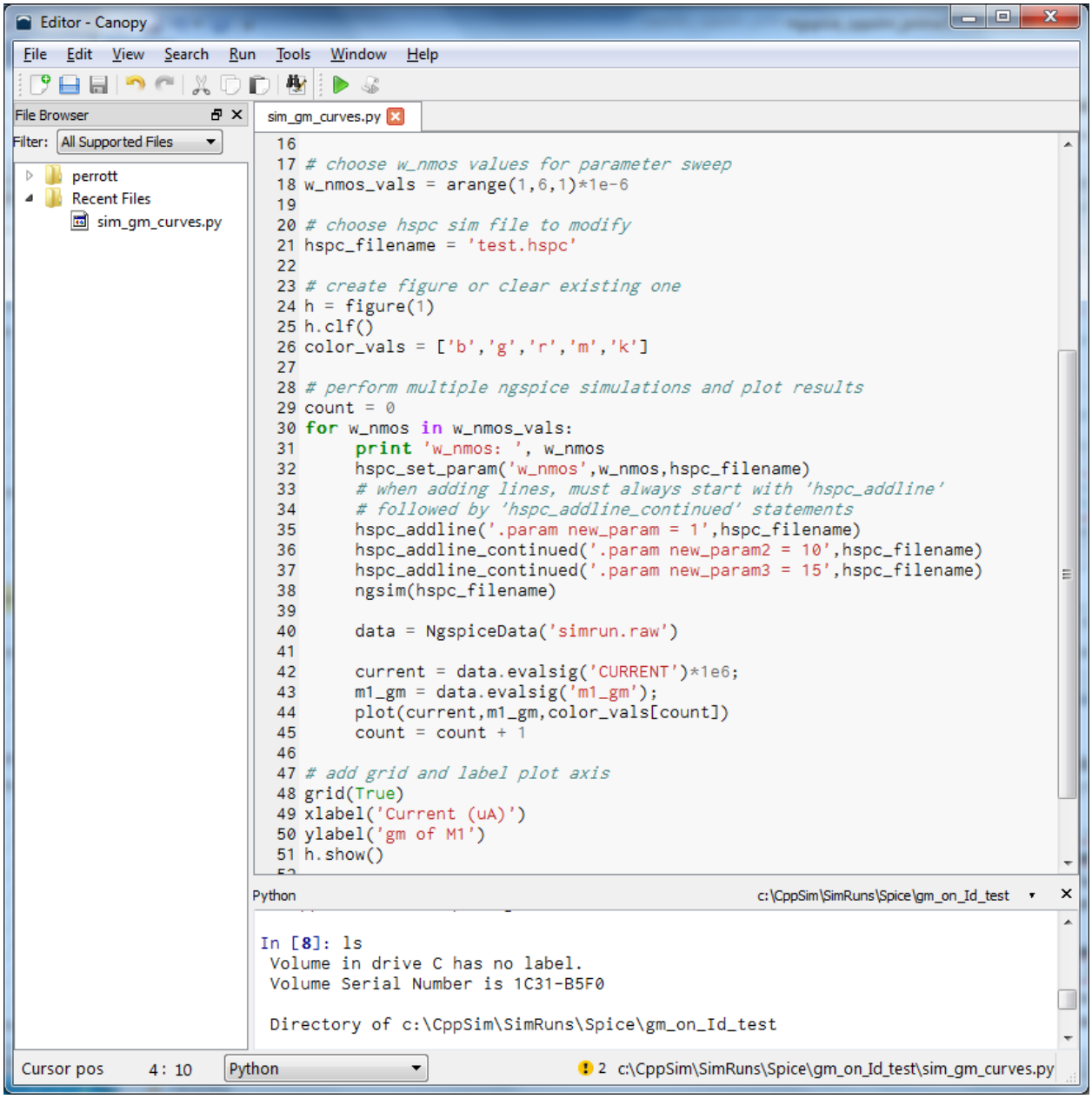
After typing 'ls' at the Python prompt, you should see **sim_gm_curves.py** as one of the files:



- We can examine the **sim_gm_curves.py** script by typing at the Python prompt:

edit sim_gm_curves.py

The resulting editor window should appear as follows:



Key commands seen in the above script include:

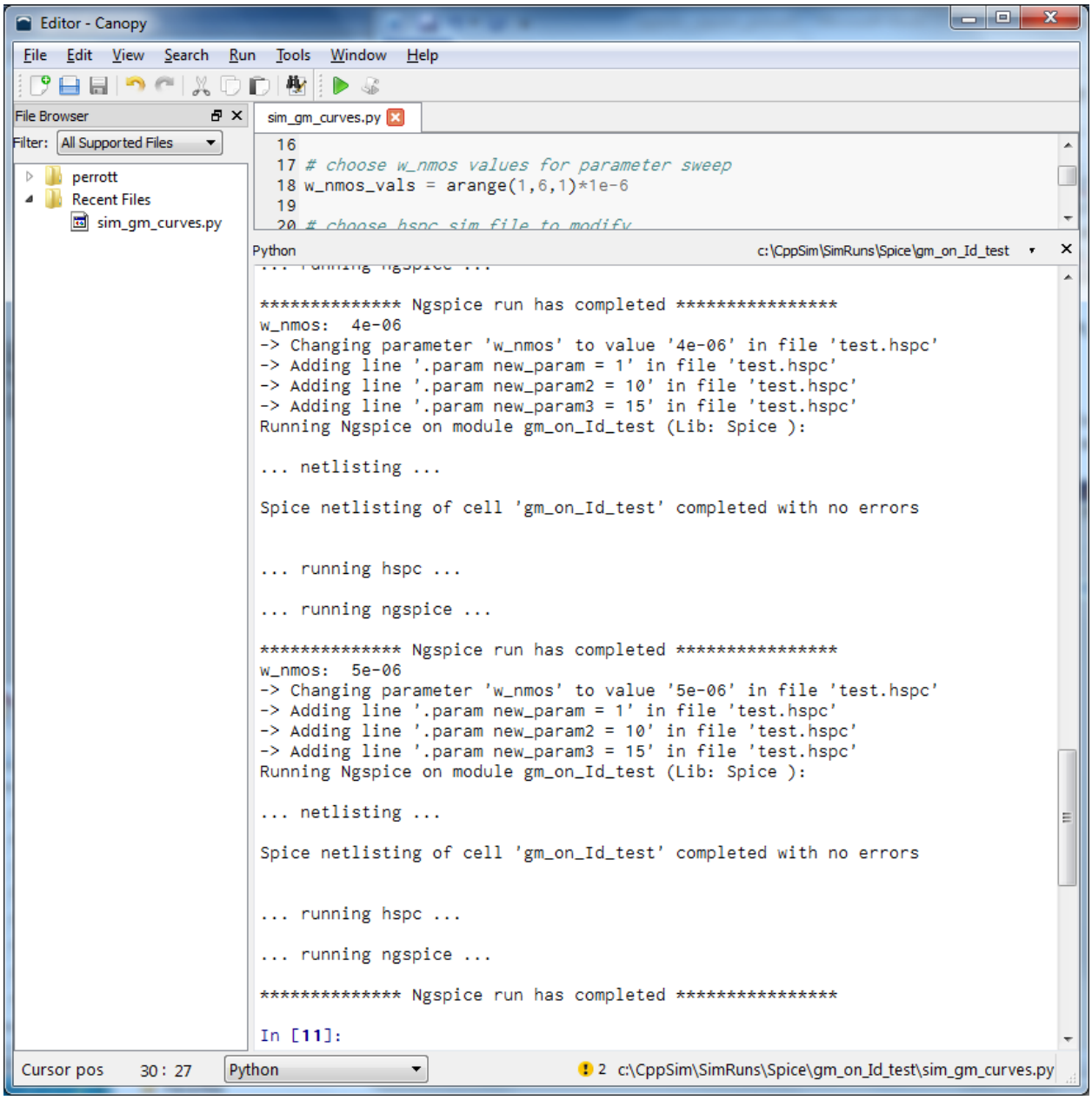
- **ngsim(hspc_filename)**: runs Ngspice with the HSPC file indicated by **hspc_filename**.
- **hspc_set_param(param_name, new_value, hspc_filename)**: this command searches the HSPC file specified by **hspc_filename** for a parameter with name **param_name** and changes its value to **new_value**. For the above example, we change parameter **w_nmos** each time before running a new Ngpice simulation using the **ngsim** command.
- **hspc_addline(new_line,hspc_filename)**: this command adds a new line specified by **new_line** to the file indicated by **hspc_filename**. For the above example, we simply add the line **.param new_param = 1** to the **test.hspc** file. This is not useful for this example, but illustrates how the command is used.

- **hspc_addline_continued(new_line,hspc_filename)**: this command adds additional lines beyond the new line created by the **hspc_addline** command. As many such lines can added as desired, but the first one must always be **hspc_addline** with the rest being **hspc_addline_continued**. Again, the lines added in this example are not useful here, but illustrate how the command is used.

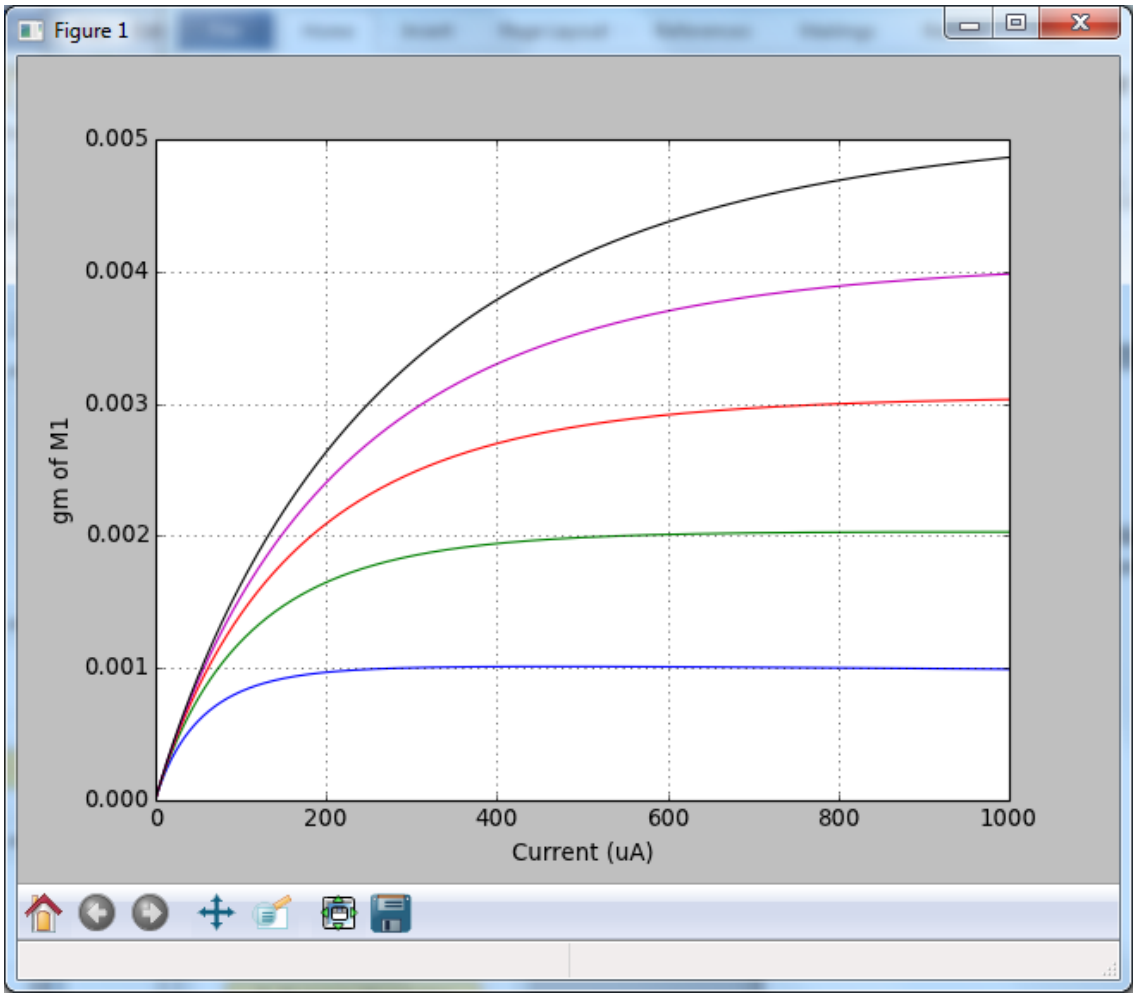
- Run the **sim_gm_curves.py** script by typing at the Python prompt:

%run sim_gm_curves

You should see several Ngspice simulation windows pop up and the Python editor window will appear as follows when the simulations have concluded:

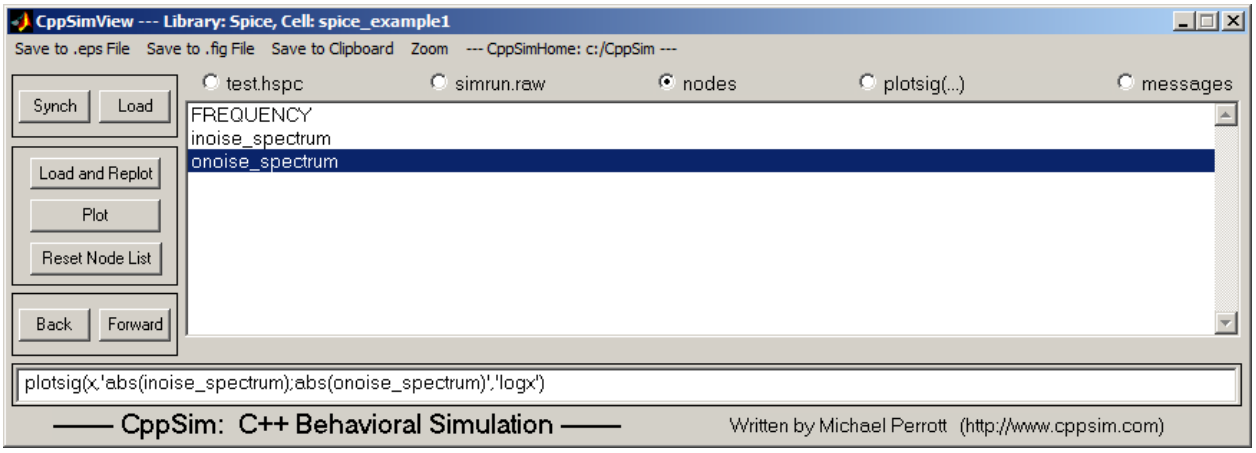


- Further, the Python plot window will show the results of the 5 simulations as shown below:



More Details on CppSimView

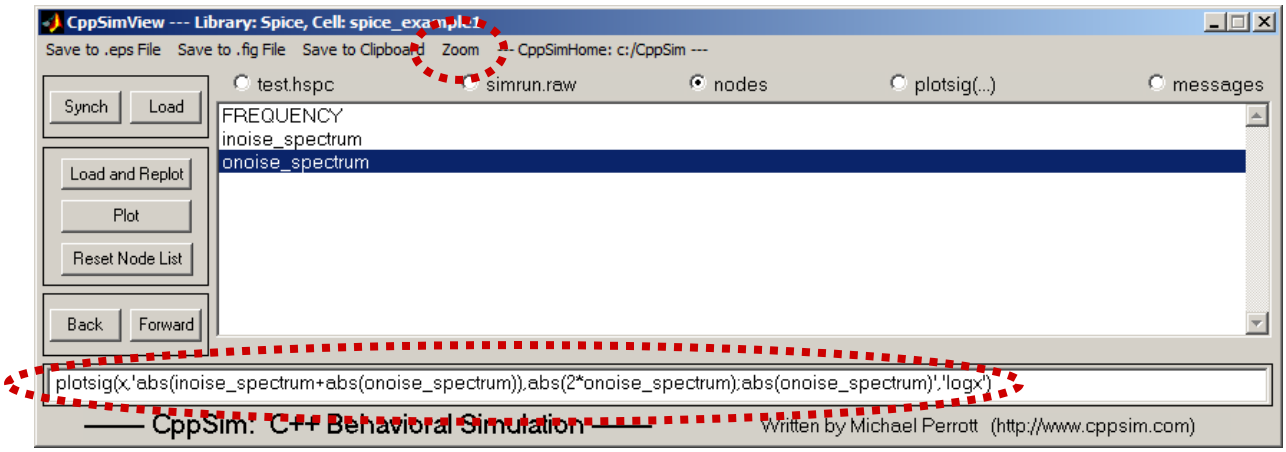
In this section, we will examine more details related to using CppSimView to view simulation results from NGspice. We will so by continuing the example from the previous section. As such, we will assume that the starting point of CppSimView is as shown below.



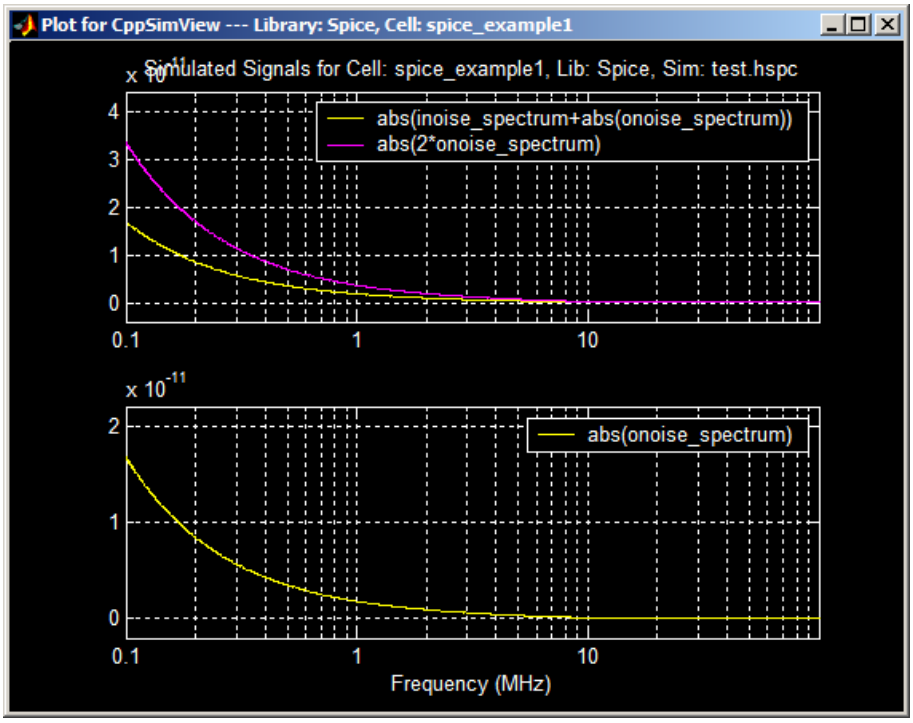
A. Basic Plotting and Zoom Methods

We have already discussed how to load in signals from an NGspice simulation and choose a logarithmic scaled x-axis as shown in the CppSimView plot window below. We will now provide further details on forming plot expressions, which are also documented in the Hspice Toolbox for Matlab manual (i.e., c**:/CppSim/CppSimShared/Doc/hspice_toolbox.pdf**) and zoom methods for interactive viewing of plots.

- As described in the Hspice Toolbox for Matlab manual, the **plotsig**() function used in CppSimView view supports mathematical operations in the plot expression. As an example, consider modifying the expression in the above CppSimView window such that we compare an individual noise spectrum to the addition of two spectra. To do so, we simply modify the expression as shown below, and use a comma separator (i.e., ‘,’) to plot within the same subplot (note that a semicolon separator (i.e., ‘;’) is used to generate separate subplots as shown below). For future reference, note also the Zoom button circled below.

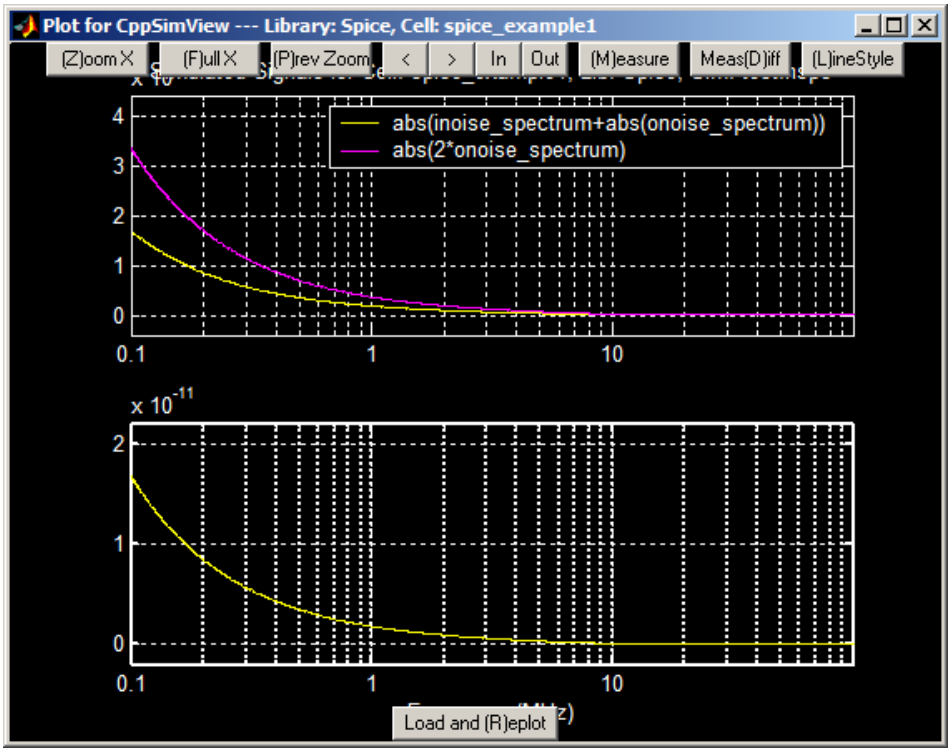


- Plotting of the above expression (by pushing the **Plot** or **Load and Replot** button in CppSimView) yields the figure shown below.



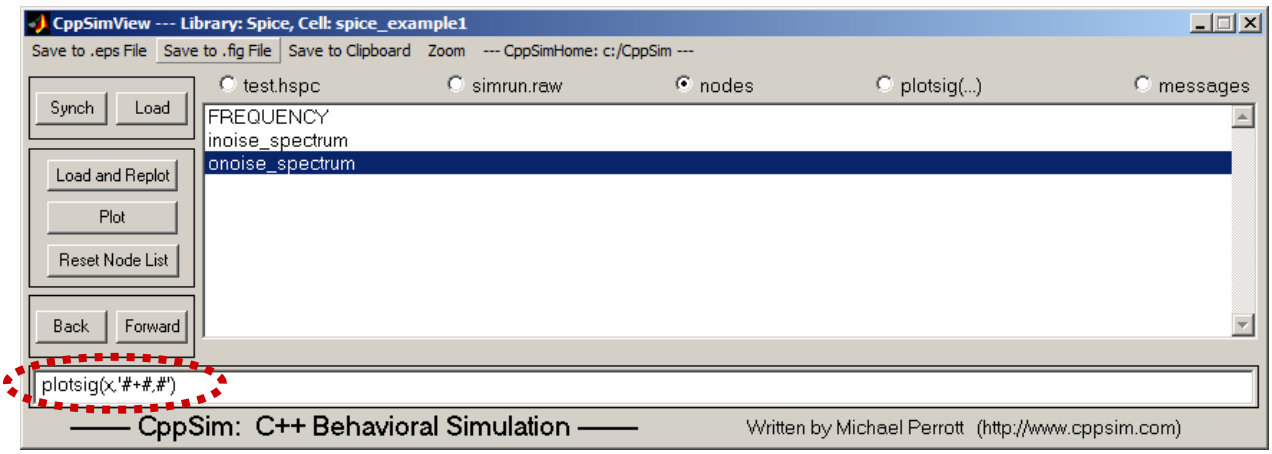
- Now click on the **Zoom** button (circled in the CppSimView window shown above) to bring up zoom controls on the plot as shown below. Each of the zoom keys has a respective hot key indicated by the parenthesis in each word. For instance, pressing **z** (upper or lowercase), allows one to zoom into a subportion of the x-axis of the current plot. Exceptions to this rule are the zoom **In** and **Out** buttons, whose hotkeys are the up and down arrow keys. Also, the left and right pan keys, < and >, are hot-keyed to the left and right arrow keys.

    - Note that no Y zoom functions are currently implemented since they are generally unnecessary since the Y-axis gets adjusted automatically during X-zoom operations.



- Press the **m** key to begin measuring a signal. Press the left mouse button repeatedly until you are satisfied with the point selected. Then press the right mouse button to complete the measurement.
- Press the **d** key to begin a difference measurement. Press the left mouse button repeatedly until you are satisfied with the first point to be selected. Then press the right mouse button repeatedly until you are satisfied with the second point to be selected. Press the left mouse button to complete the measurement.
    - Note that you can combine the **Measure** and **MeasDiff** commands. First, perform a measurement command by pressing the **m** key as described above. Upon completion of this command, press the **d** key to begin a **MeasDiff** command. However, instead of pressing the left button, press the right one. The first point will remain that selected by the **Measure** command, and the second can now be set where desired. Press the left mouse button to complete the **MeasDiff** operation. The advantage offered by this option is that you can zoom into a particular part of the waveform and select an initial point using the **Measure** command. You can then zoom into a different portion of the waveform, and then left-click on **MeasDiff** to determine the difference from the last point to a new point in the current zoom location by using this technique.
- Press the **l** key (i.e. lowercase L) to display the actual sample values from the simulation (as indicated by circles). Press the **l** key again to return to solid lines for the plot.
- Press the **p** key to return to the previous zoom value (i.e., the last achieved through use of the **Zoom X** button). Note that if you just used the **Zoom X** function without doing any other zoom or pan operations, you will see no change in the plot.
- Press the **Zoom** button again on the CppSimView main window (as circled in the figure above) to remove the zoom buttons from the plot window.

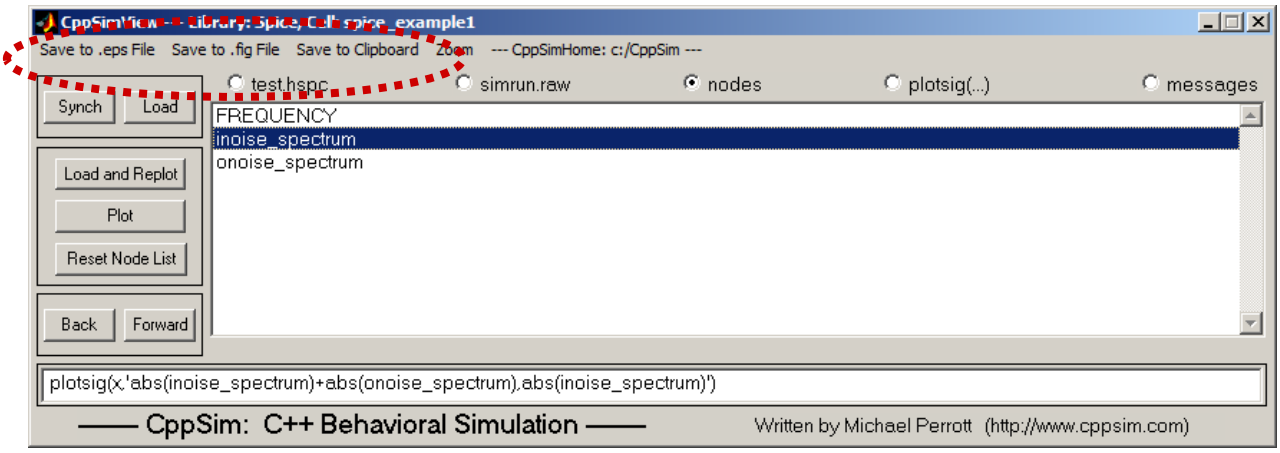
D. Advanced Plotting Methods

- There are actually five ways to perform plotting with CppSimView.
  - The first is to left-click on the **Plot** button once an expression is entered into the bottom command line (as demonstrated above).
  - The second is to double-click on a node in the listbox as covered in previous sections of this document. The plot expression currently selected on the plot radio button (i.e., **plotsig(…)** in the figure below) will then be filled with the selected node, and additional subplots are created as you continue to double-click on signals. To reset the number of subplots to one, press the **Reset Node List** button – it then turns back to **Plot** and additional double-clicks on signals start the process over.
  - The third is to enter a plot expression directly into the command line and then press the **Enter** key to produce the corresponding plot. One can also modify an existing expression created, for instance, by the second method. The latter method often proves convenient – simply double-click on the desired signals to produce various subplots, and then modify the resulting command line expression to implement functions on the various signals or to position them on the same subplot (using a comma separator rather than a semicolon).
  - The fourth is to enter a plot expression in the command line, but insert **#** characters into the expression where you would like to have signal names. Once you have completed the expression, double-click on node names and observe that the # characters are substituted from left to right with the signal names. Once the last # character has been filled in, a plot of the expression will be produced. Note that it is useful to click on the **Reset Node List** button first to clear the plot expression.
  - 

  - The fifth method is to use the **Back** and **Forward** buttons to scroll through a history of previous plotting expressions. Once a desired plotting expression is encountered, left-click on the **Plot** button to replot it or perform alterations of the expression in the command line as desired and then press the **Enter** key. Note that the history commands are specific to the selected simulation file and output file (i.e., **test.hspc** and **simrun.raw**, for example, in the figure below). The history keeps track of the last 400 commands used on a given cellview (i.e., for **spice_example1**, as an example), and it is shared among the various simulation and output files for that cellview.

E. Saving Plots to EPS files, FIG files, or the Windows Clipboard

- To save plots to an eps file, fig file, or to the clipboard, press either **Save to .eps File, Save to .fig File,** or **Save to Clipboard**, respectively, in the CppSimView main window. When saving to the clipboard, the plots can then be pasted into other Windows applications such as Word or PowerPoint.



More Details on Sue2

Sue2 provides a convenient graphical interface for creating and modifying circuit schematics, and is designed to have many similarities to professional schematic captures tools such as Cadence Composer so that IC designers can easily alternate between these tools as they iteratively perform system and circuit level design. A more complete manual is available for Sue2 as the PDF document:

- **c:/CppSim/CppSimShared/Doc/sue_manual.pdf**

but we will cover enough of its operation here for users to get a good feel of this package.

Before we begin, there are two important things to keep in mind when you use Sue2:

- Always pay attention to the Help Message Window, which is to the right of the menu at the top of the main canvas, during command operations – it provides information for bindkeys activated while a given command is in effect

- To break out of any given command mode, hit the **Esc** key. This is very important to remember – if Sue2 ever seems to lock up, hit the **Esc** key! (The other reason Sue2 may appear to lock up is if an entry form was opened but not completed – in such case, be sure to find the entry form among the Windows applications and close it to continue with Sue2).

A. Using Navigation and Edit Commands

Sue2 allows its bind-keys to be changed according to user preference by editing of the file c:/CppSim/Sue2/.suerc. That being said, the default values of common navigation and edit bind-keys are listed here.

- Sue2 navigation commands:

- **f** – fit view to the window size
- **z** – zoom in
- **Z** – zoom out
- Zooming can also be accomplished by pressing the right mouse button and dragging the mouse over the region to be zoomed into
- Panning is done by either hitting the arrow keys or by holding the **Ctrl** key and then dragging the mouse while the left mouse button is held down.
- **e** - descend into hierarchy of selected cell.
- **Ctrl+e** – Return to higher level of hierarchy.

- Sue2 editing commands:

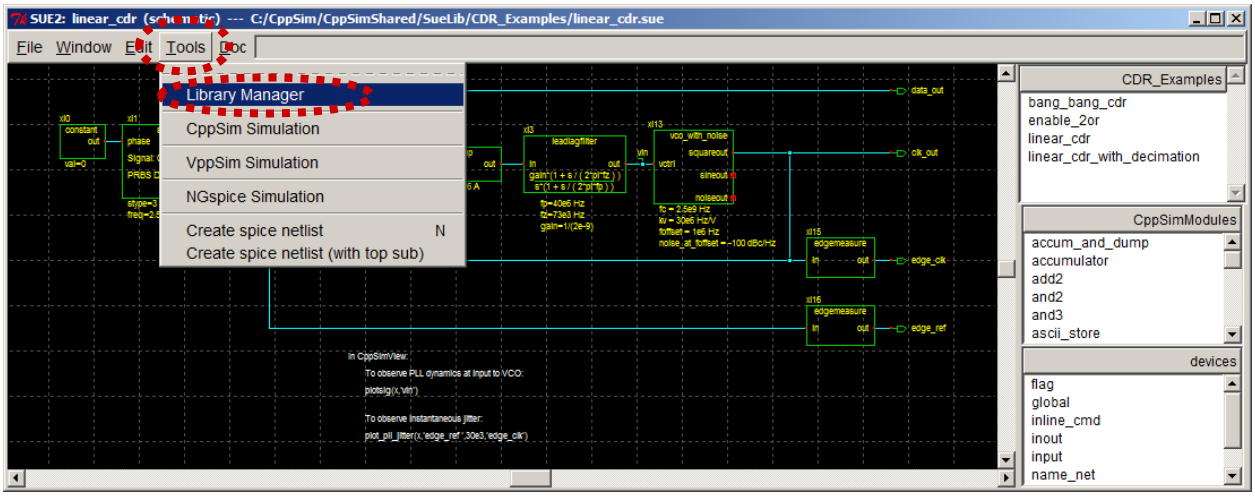
  - Modify the parameters of a cell within a schematic by double-clicking on the cell. A listbox will appear that displays the cell parameters and allows their modification.
  - Move cells by pressing and holding the left mouse button on the desired cell and then dragging the mouse.
  - Select multiple items by holding the left mouse button and dragging the mouse over the items to be selected. Additional items can be added to the current selection by holding the **shift** key and then progressively clicking the left mouse button on the items of interest.

B. Creating a New Schematic

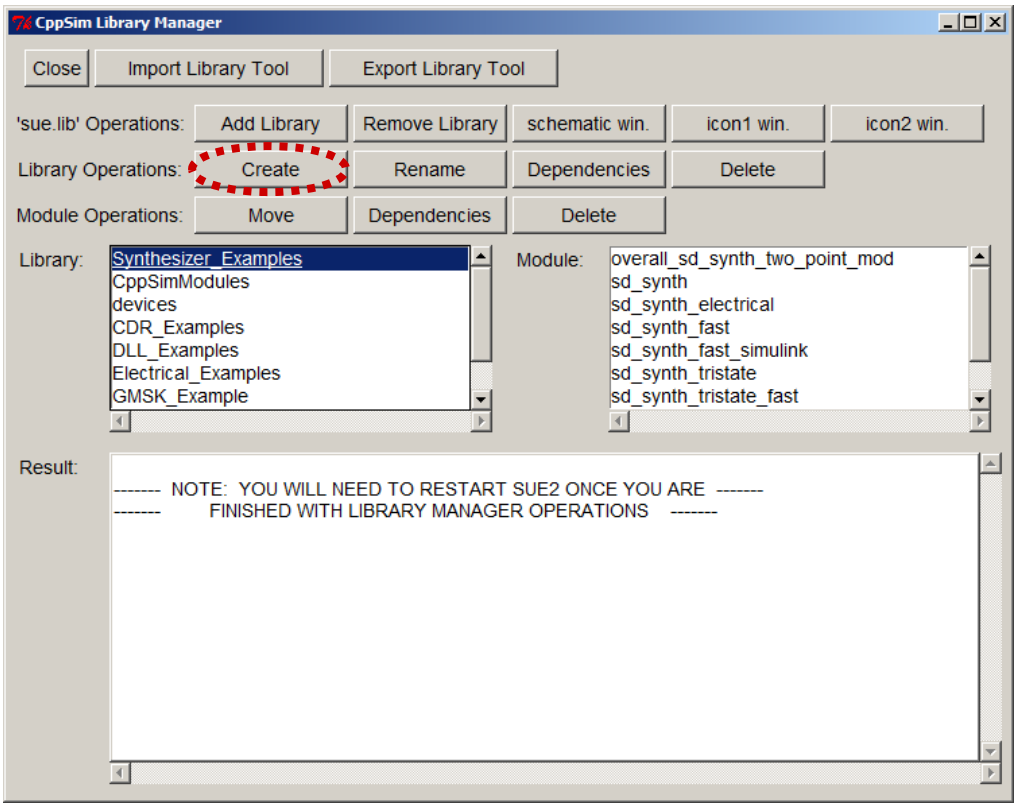
Let us now walk through an example to see how to create a new Sue2 schematic. Note that this will be done in the context of creating a CppSim simulation, but the ideas carry directly over to NGspice. There is no need to understand how CppSim works for this example.

For our example, we will create a pseudo-random bit stream (PRBS), pass it into a lowpass filter, and then view the results both as time domain signals and in the form of an eye diagram.

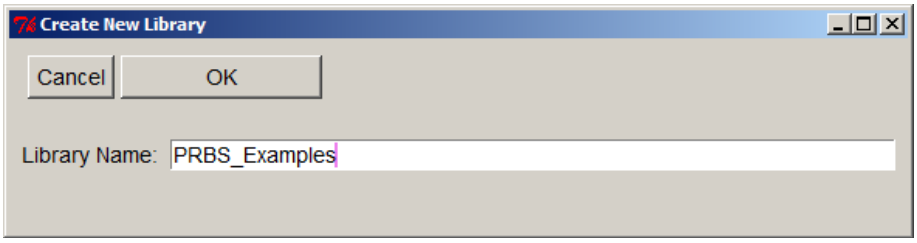
- We will first create a new library called **PRBS_Examples**
  - In Sue2, click on the **Library Manager** menu item under the **Tools** menu bar item as shown in the figure below.



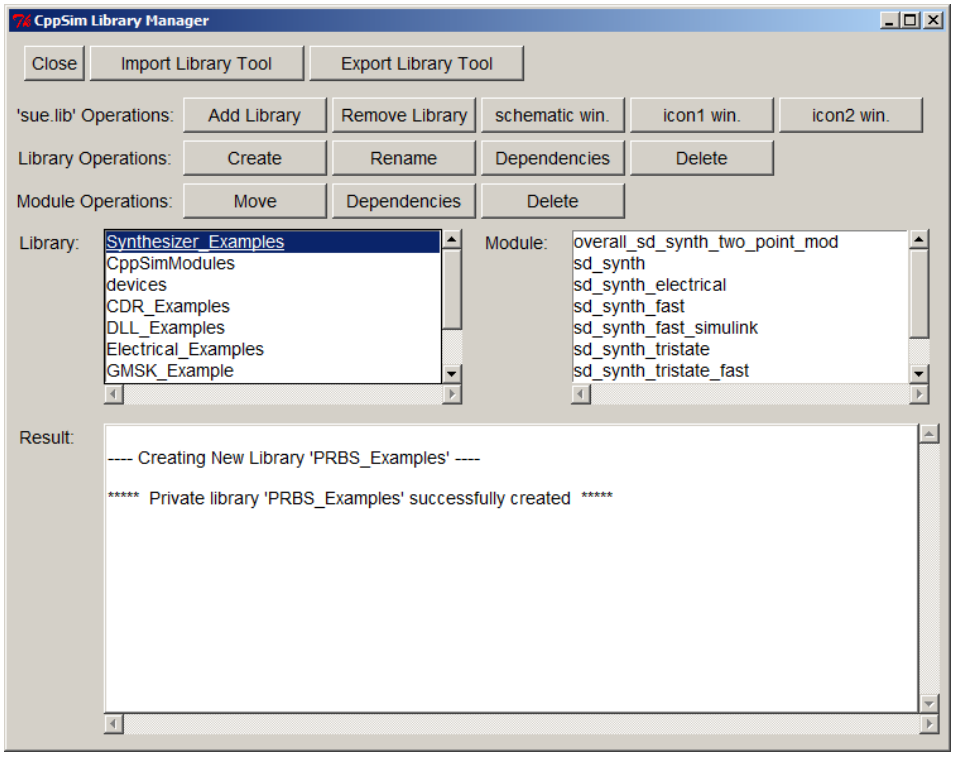
- In the **Library Manager** window that appears, click on **Create** Library as shown below.



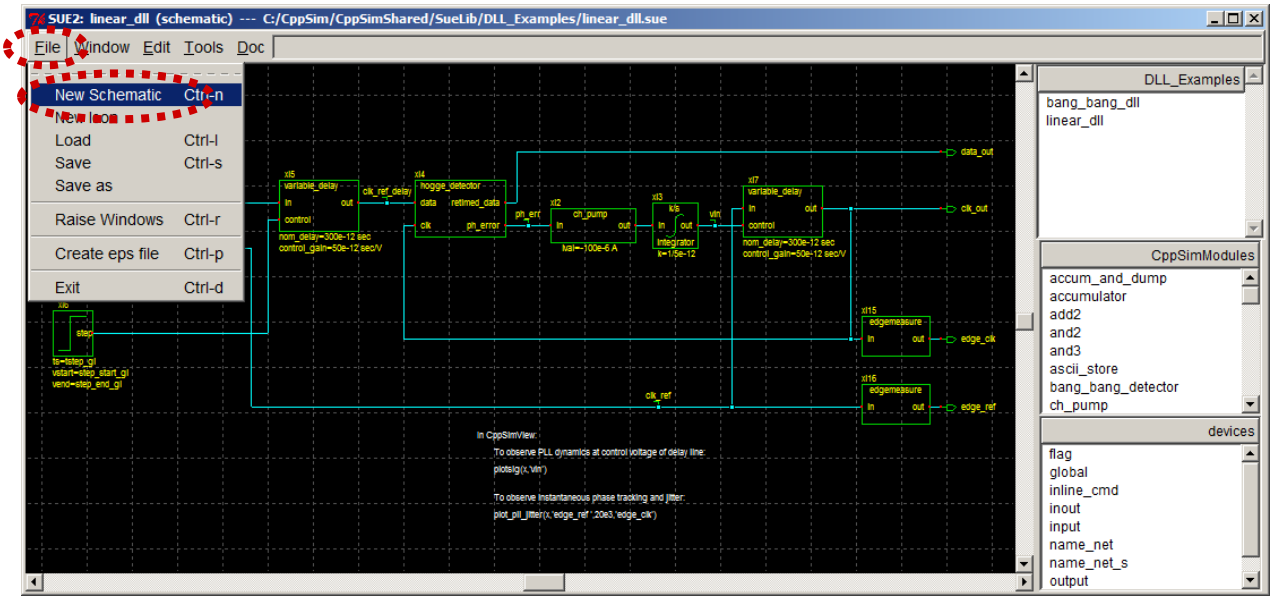
- Choose the new library name as **PRBS_Examples** and then press **OK.**



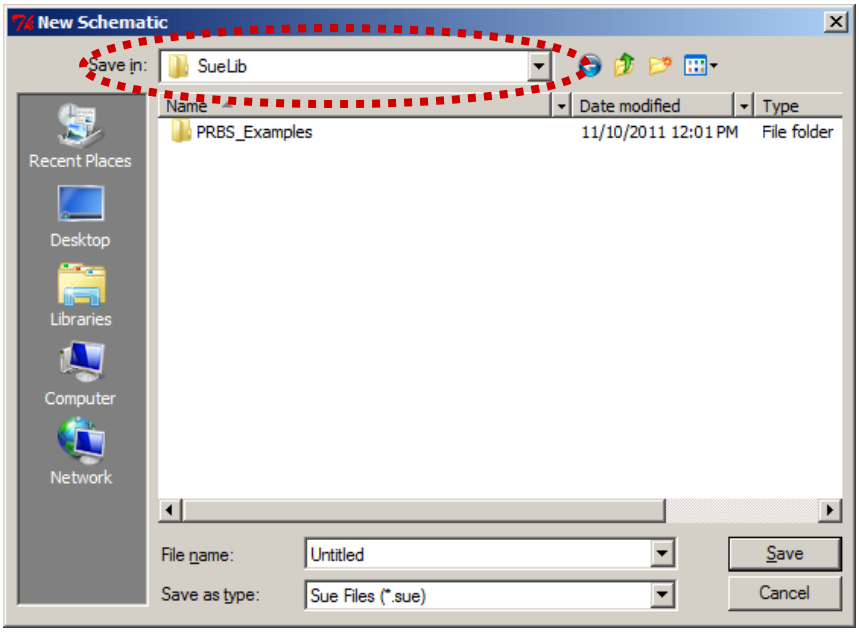
- You should see a confirmation window in the **CppSim Library Manager** window as show below. You should then **Close** this window.



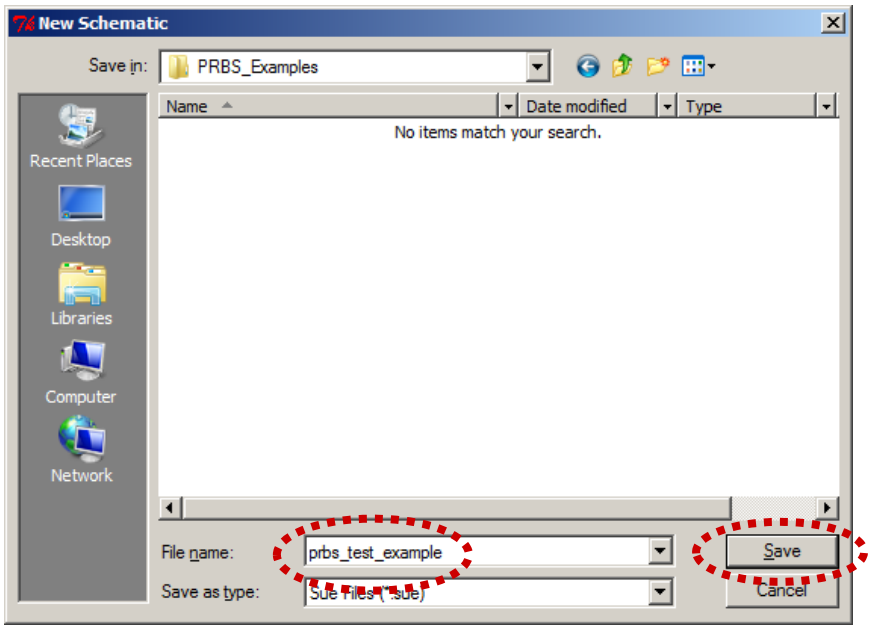
- In Sue2, create a new schematic cell as follows:
  - Select **File -> New** Schematic as shown below.



  - A **New Schematic** window opens as shown below. Within the **Save in:** section, select the current path to be **c:/CppSim/SueLib**. You should see the **PRBS_Examples** directory as shown in the figure below.

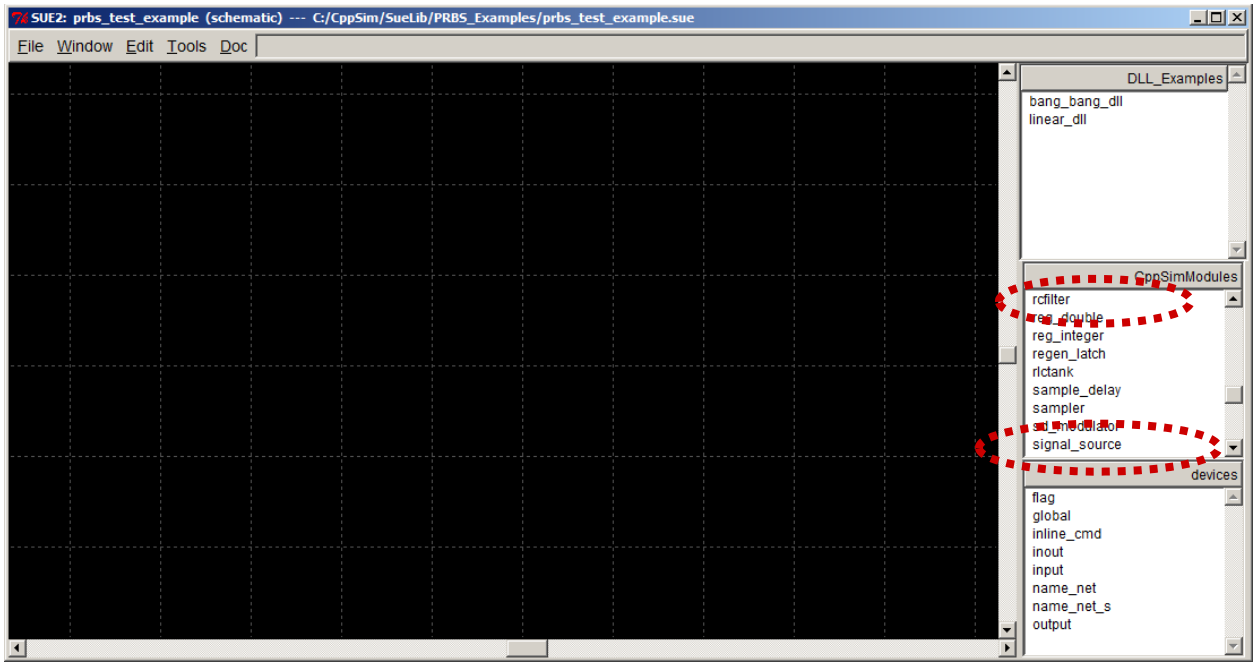


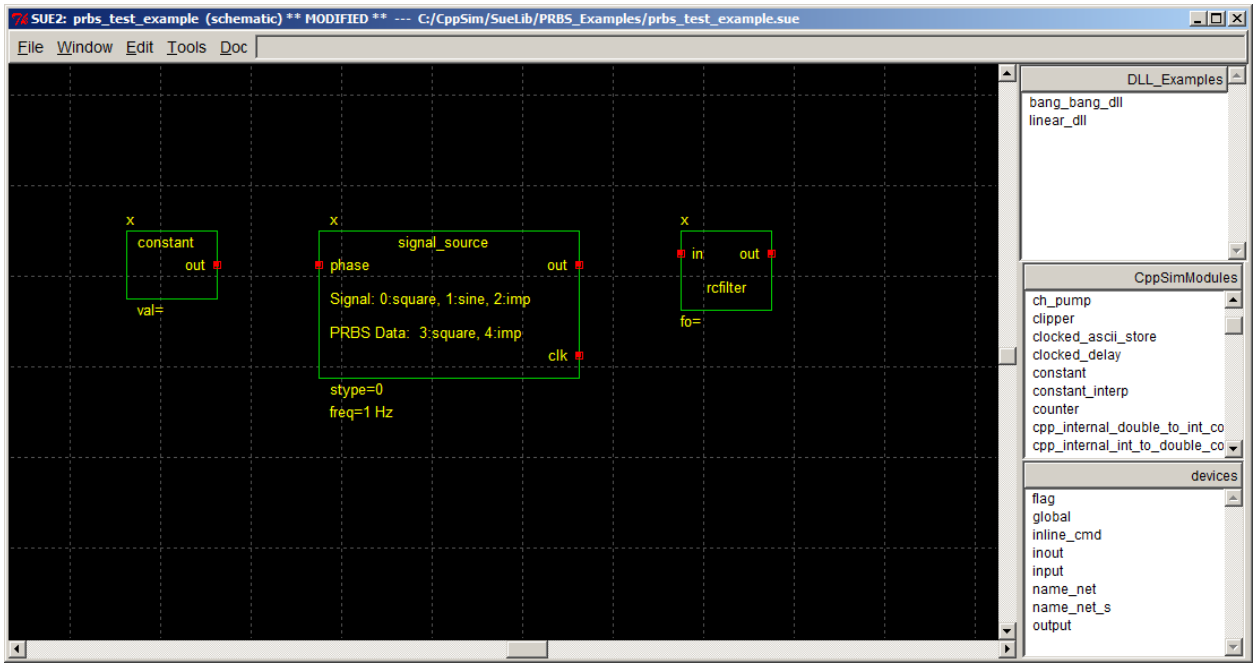
- Click on the **PRBS_Examples** folder icon, and then specify the **File name** as **prbs_test_example** as circled below. Left-click on the **Save** button, as also circled below, to complete the creation of the new schematic. You should now see the top banner of the schematic window state that the new schematic is **C:/CppSim/SueLib/PRBS_Examples/prbs_test_example.sue.** In case this point is not clear, please view the schematic window shown as a figure on the next page in this document to see how this information is displayed.



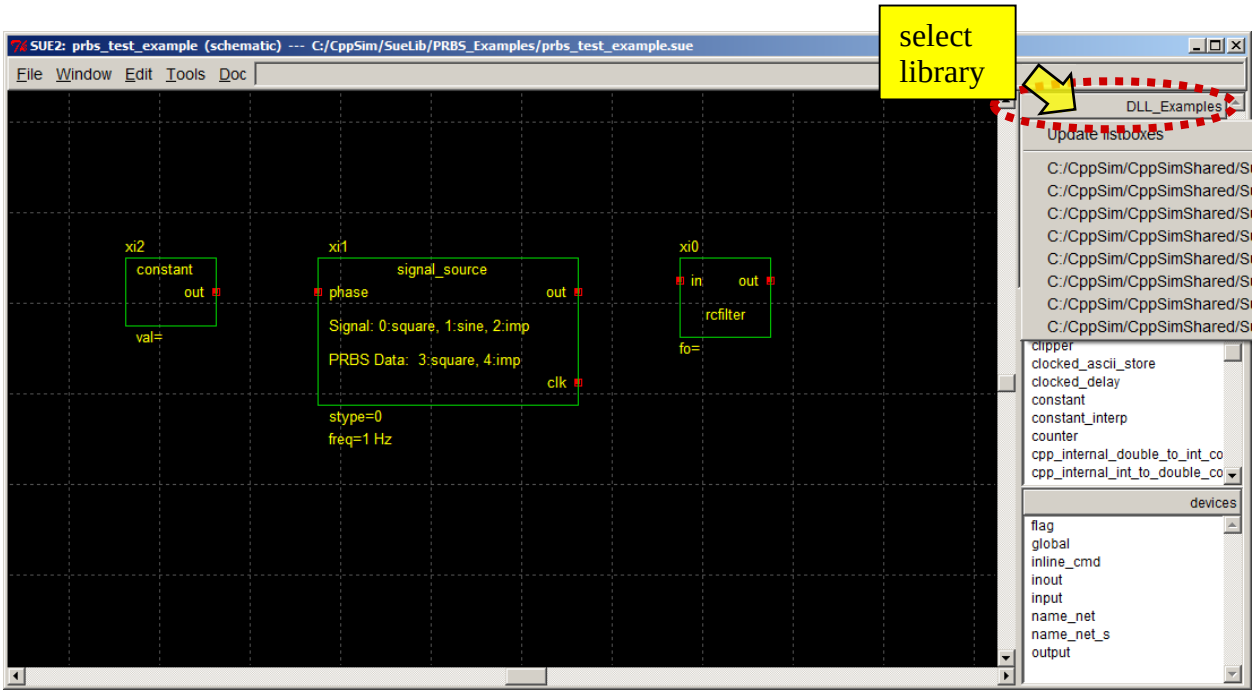
- In the Sue2 **icons1 listbox**, as shown below, select the **signal_source** icon and then move the cursor into the main Sue2 schematic window. Click on the mouse to place the icon at an appropriate place. Then select the **rcfilter** icon, as circled below, and again move the mouse into the main Sue2 schematic window to place this icon to the right of **signal_source** cell.

Finally, select the **constant** icon from the **icons1 listbox** (you must use the scroll button to see this icon name) and then place it to the left of the **signal_source** cell. The main Sue2 schematic window should now appear similar to the figure displayed below.

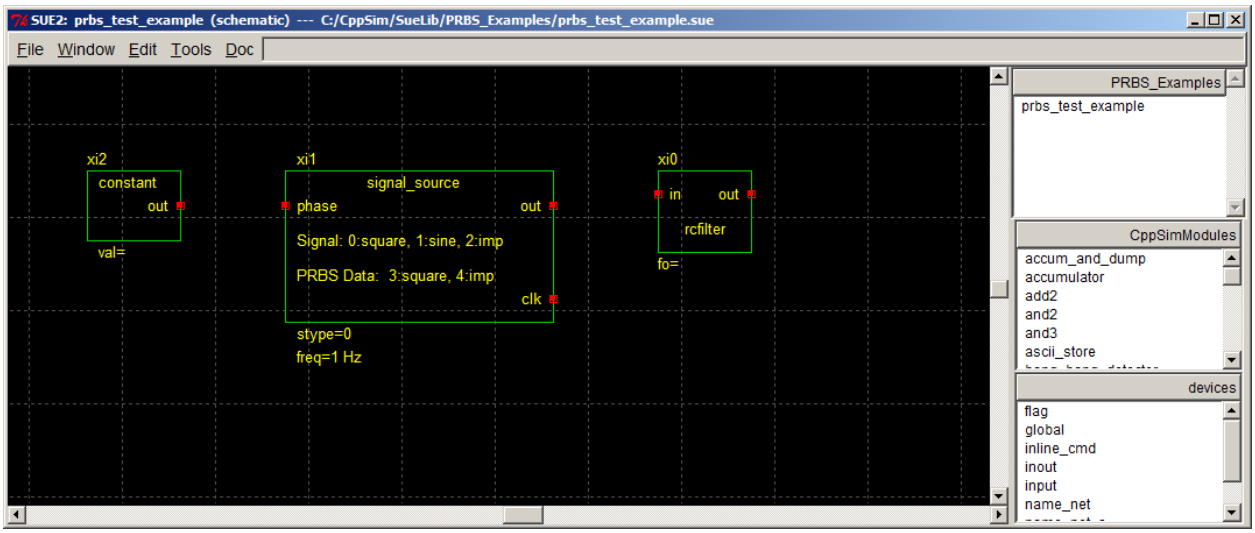




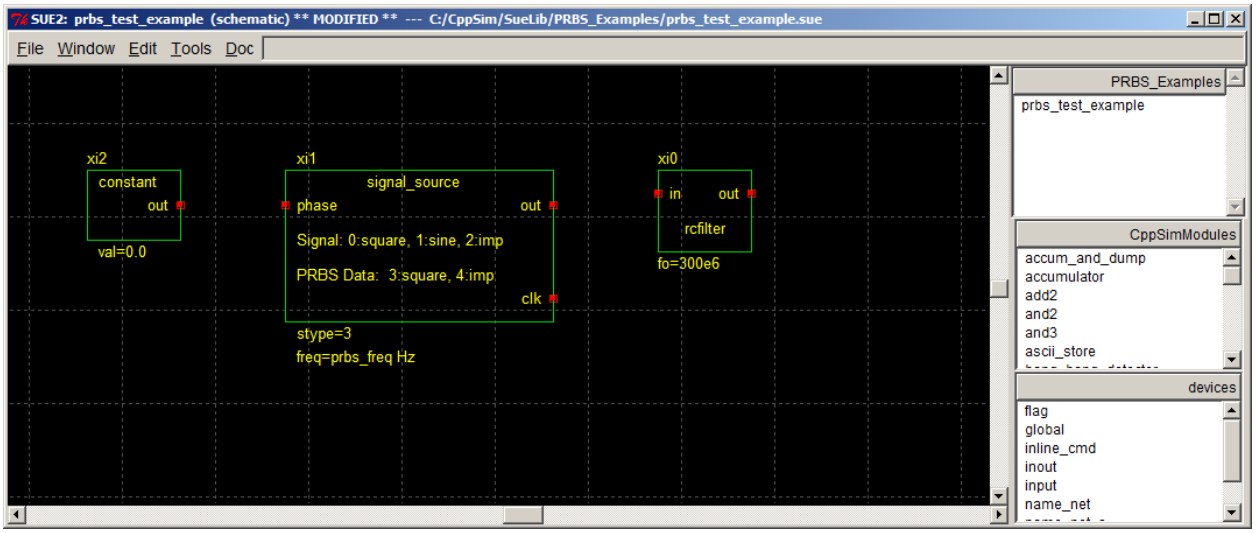
- Save the schematic view by clicking on **File->Save** (or hold the **Ctrl** key and then press the **s** key). If you now click on the top portion of the **schematics listbox**, as circled below, you'll notice that the library **PRBS_Examples** does not show up.



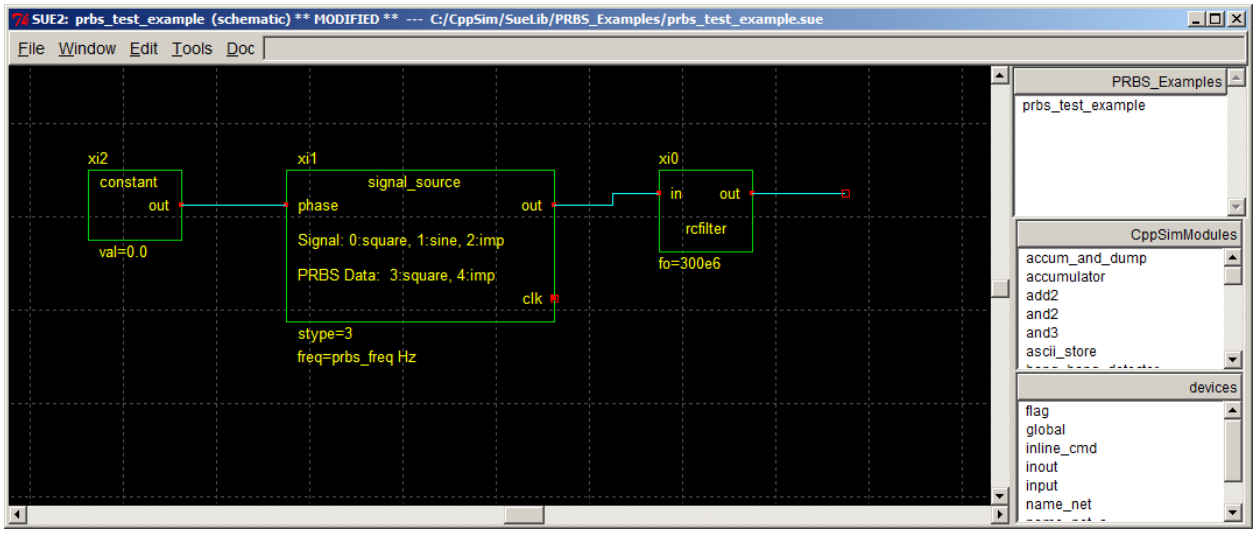
- The issue of **PRBS_Examples** not showing up as a library in Sue2 occurred since there were previously no cells in this library. Now that you have created a cell for this library, you can correct this issue by exiting Sue2 and then restarting it again. After doing so, you should then click on the top portion of the **schematics listbox**. Now the set of libraries will include **PRBS_Examples,** which should be selected. You should then choose schematic **prbs_test_examples** to re-obtain the same schematic shown above.



- Select parameter values for each of the cells in the above schematic by double-clicking on each of them and setting them as follows:
  - **constant cell**: consval = 0.0
  - **signal_source** cell: stype = 3, freq = prbs_freq
  - **rcfilter** cell: fo = 300e6



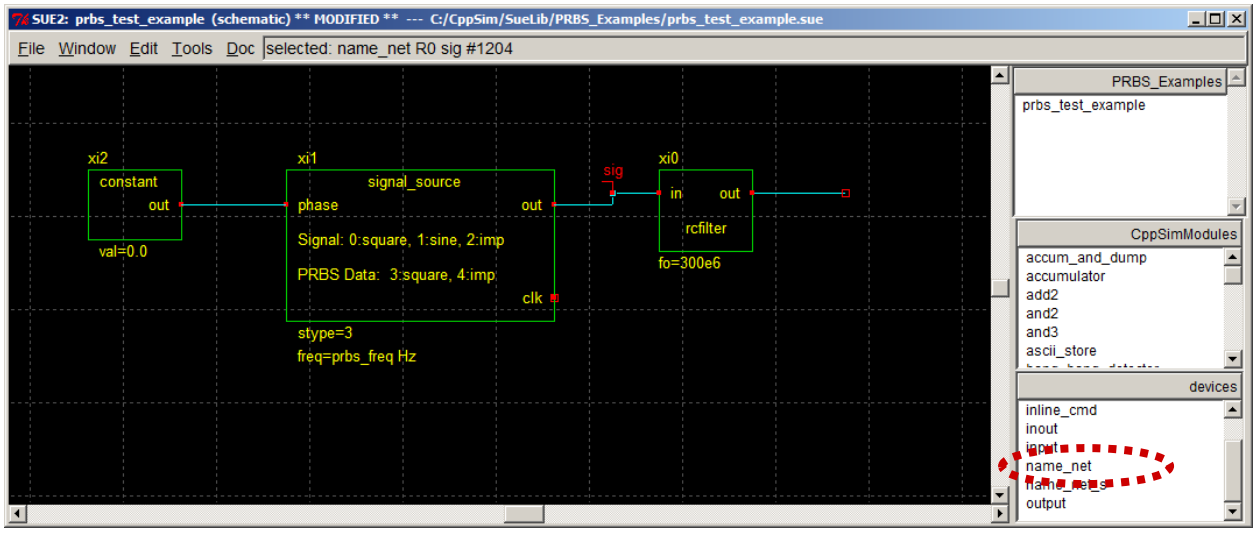
- To connect the cells, we need to add wires. You enter wire-create mode by typing **w** in the main Sue2 schematic window. To start a wire, left-click at the desired starting point (usually at the terminal of a cell). Place the cursor at the end of the desired wire segment, and then left-click to create a new segment. A wire is completed when it is connected to a cell or pin terminal, though double-clicking the left mouse button (or single-clicking the right mouse button) will force the end of a wire at any point in the schematic. Note that you must push the **Esc** key to end wire mode. Given this information, complete wiring for the schematic as shown below.



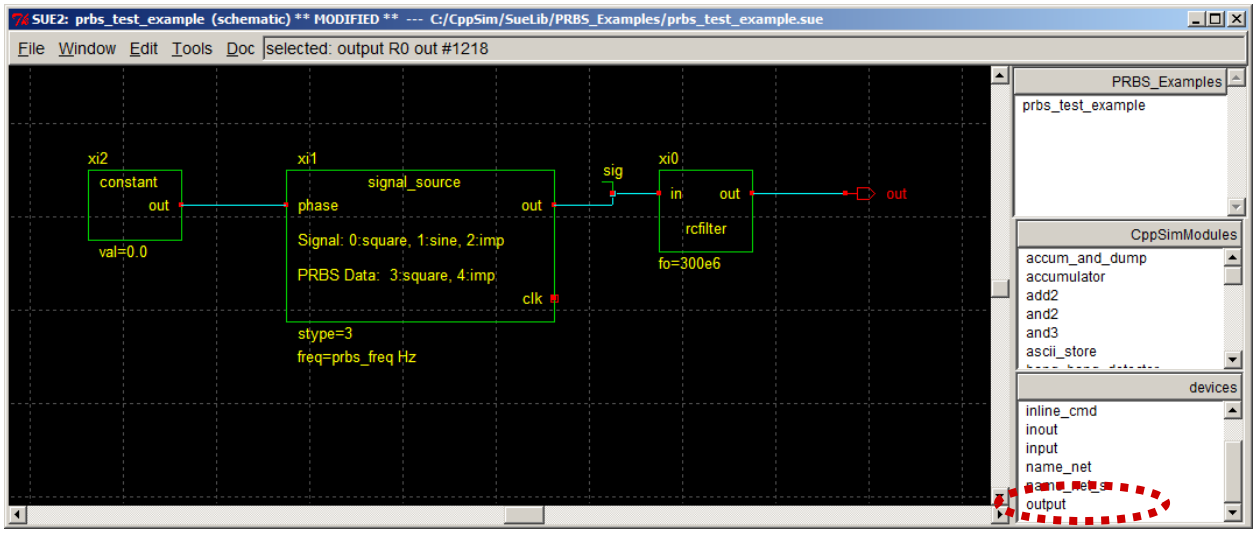
- To probe signals produced in the CppSim simulation of the schematic, we need to label all signals of interest. We also should add pins to any nodes that we might want to bring up to the next level of hierarchy.

    - For this example, let us label the output node of the **signal_source** cell as **sig**. To do so, click on **name_net** of the **icons2 listbox** (as circled below), move the mouse cursor into the main schematic window, and then place the **name_net** icon on the wire connected to terminal **out** of **signal_source**. Double-click on the **name_net** icon once it is placed, and set its name to **sig**. The schematic figure below illustrates how the

name_net icon should look within the schematic once these operations are completed. Note that you can also use the **name_net_s** icon instead of **name_net** to name nodes – the only difference between them is their appearance.

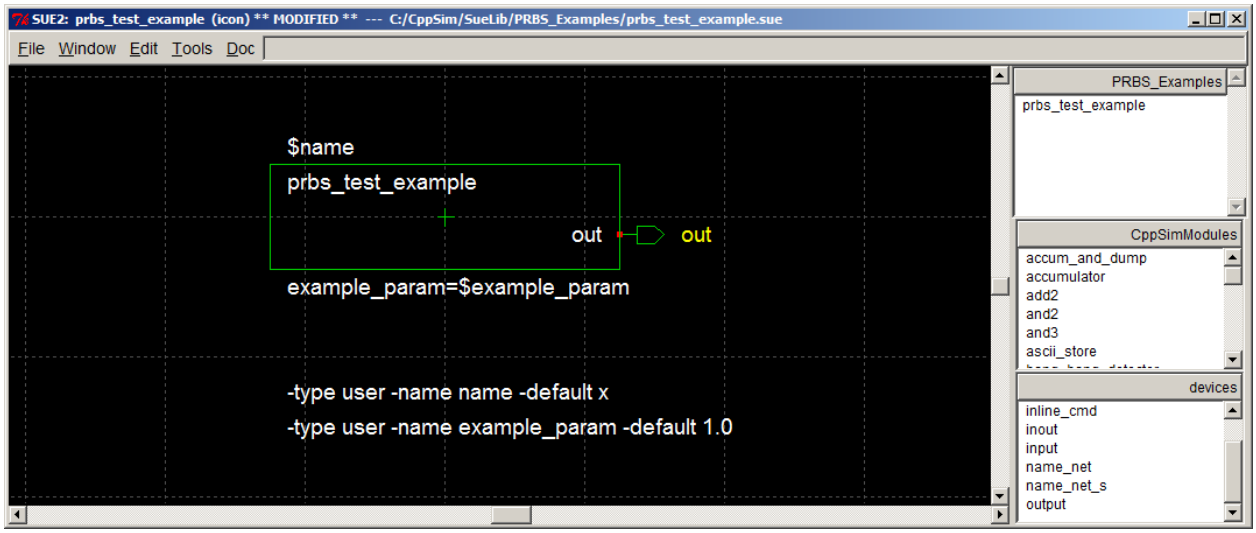


- Add an output pin to the schematic by clicking on **output** in the **icons2 listbox** (as circled above), moving the mouse cursor into the main schematic window, and then placing the pin at the output of the rightmost wire in the schematic as shown below. Once the output pin has been placed, double-click on it to change its name to **out**. Be sure to save the schematic at the completion of these operations.



C. Creating an Icon View (And Associated Parameters) For A Given Schematic

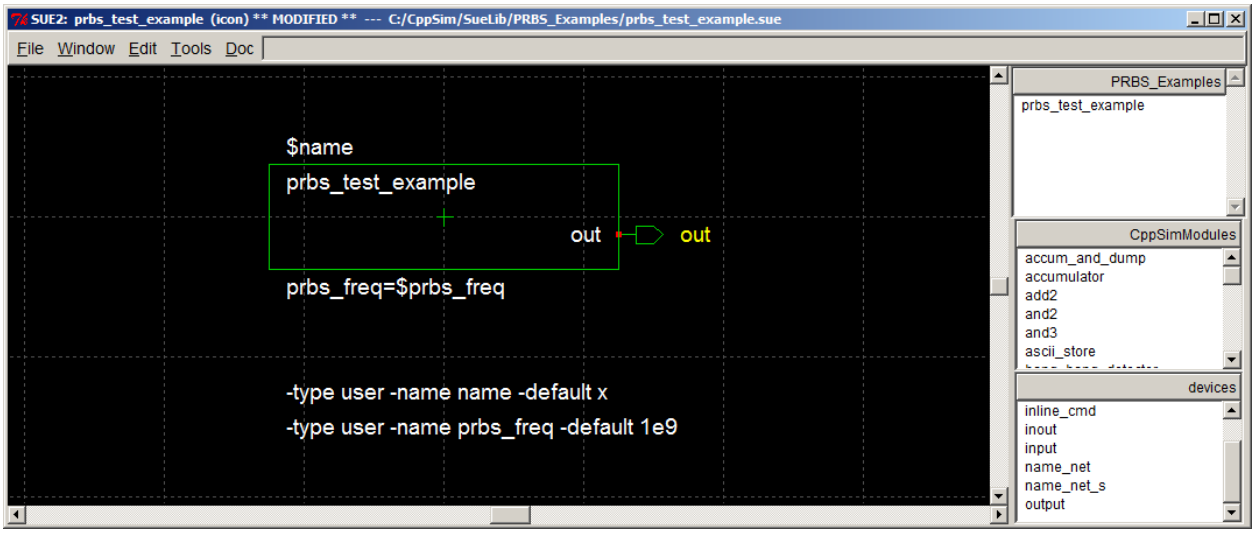
- Assuming you are currently in the schematic shown above, creation of an associated icon is straightforward. Simply click on **Window->make icon**, or press its associated bindkey, **K**. The resulting icon view should appear as shown below. Be sure to click on **File->Save** to save this new icon view.



- The newly created icon view is intended to be a template for the actual icon view desired. We will now change its default parameter, **example_param**, and explain how to alter its rectangle box.

    - The two statements involving **example_param** are intended as a template for creating parameters, and should either be removed or modified to reflect a parameter name of interest. The top statement specifies how the parameter and its value will be displayed when the icon is instantiated within a schematic. The bottom statement declares the parameter and provides its default value.

      In this case, our schematic has one parameter, **prbs_freq**, that we would like to implement. To do so, double click on the two statements involved **example_param** (one at a time), and replace **example_param** with **prbs_freq**. Select the default value of **prbs_freq** to be 1e9, and add units of **Hz** to the top statement. Hit the **Enter** key each time you complete the changes for a given statement. The figure below indicates how the icon view should look upon the completion of these changes.



- To add more parameters, you would simply add more statements in similar fashion to the two you just modified. Statements can be added by either copying a current statement (click-left on a statement of interest to select it, press **c**, left-click again, then left-click once more to place the copy) and then modifying the copy, or by clicking on **Edit->add text** (bindkey is **t**) and directly entering a new text statement.

- To change the size of the icon rectangle (i.e., the green rectangle shown in the above icon view), double-click on the rectangle and solid boxes will appear at its corners. Left-click on one of the corner boxes and then move the mouse – the associated corner of the rectangle will change in accordance with the mouse movements. Release the left mouse key to retain the current position of the given rectangle corner.

- You have several options for creating shapes for icons in Sue2:

  - Create a line by pushing the **l** (as in **l**ine) key followed by the left mouse button, and then double-clicking on the left mouse button (or single-clicking the right mouse button) at a different point on the canvas. Multi-segment lines are created by single rather than double-clicking on the left mouse button at each desired breakpoint of the line, with a double-click of the left mouse button (or single-click of the right mouse button) to end the line. Press the **shift** key to limit the drawing of line segments to either the vertical or the horizontal plane. Once a line is created, its various line segments can be modified by first double-clicking on the line, and then pressing and holding the left mouse button over the given breakpoint followed by dragging of the mouse to the new desired location.
  - Create an arc by pressing the **a** button followed by pressing (not holding) the left mouse button, moving the mouse until the appropriate size and shape for the arc is achieved, and then pressing the left mouse button.
  - As mentioned above, create text by pushing the **t** key followed by the left mouse button at the desired location for the text. Modify text by double-clicking on it with the left mouse button and then performing edits. Only three sizes of text are available – the size of the given text segment may be varied while in text mode by holding the **Shift** key and then pressing either the left, middle or right mouse button to select the desired size. Also, the text can be changed to either left, middle or right justified by holding the **Ctrl** key then pressing either the left, middle or right mouse button.

- Save the icon view after you have completed the desired changes. The icon is ready to add to other schematics, and can be accessed in one of the **icons listboxes** by selecting **PRBS_Examples** as the library for a given **icons listbox**.