

# Tutorial 1: Getting Started

## 1 Background on Cadence Virtuoso

Cadence Design System, Inc. is an American electronic design automation software and engineering services company. Cadence produces software and hardware for designing integrated circuit, system on Chip and printed circuit boards. Cadence Virtuoso is a large, sophisticated software tool for designing and manipulating designing full-custom integrated circuits; includes schematic entry, behavioral modeling (Verilog-AMS), circuit simulation, custom layout, physical verification, extraction and back-annotation. Used mainly for analog, mixed-signal, RF, and standard-cell designs, but also memory and FPGA designs projects. For this reason, it has many moving parts and can be more than a little intimidating at first glance. Additionally, years of iterative change in an environment driven by many different strategies of circuit design and the expectation that customers will be large companies with full-time support staff have rendered the overall interface design of Cadence somewhat... eccentric.

This guide will take you on a tour of the major functioning parts of Cadence Virtuoso, providing you with a platform from which to start designing and simulating circuits with confidence.

### 1.1 Before starting (important!) [ ECE Linux Server ]

Make sure that your EE account is set to C-Shell `/bin/tcsh`. To verify this setting, go to the UWEE computing page and select **Manage Your EE Account**. Once you are logged in, ensure that `/bin/tcsh` is highlighted in the field **Login Shell**, and select **Submit these changes**.

### 1.2 On terminal command examples

Terminal commands are prefixed with `$` to denote a command that is typed into the shell. Terminal output is sometimes shown for the sake of demonstration or clarity, and will not be prefixed with the `'$'`. For instance, the output of `'pwd'` is sometimes shown to indicate the correct directory in which to copy a file, or execute a command.

### 1.3 Folder Organization

- 476 Folder ECE Linux server

```
/home/projects/ee476.2022aut
```

- 476 Folder ECE Hyak

```
/gscratch/ece/courses/476_aut2022
```

## 2 Setup Linux environment

(Note: The steps in this section only need to be performed once!)

### 2.1 Set tcsh as default shell.

Create/modify the file `~/.bash_profile` with following content:

**Be sure to have EXACT same content:**

```
# .bash_profile
```

```
WHICH_TCSH=`which tcsh`

if [ $WHICH_TCSH != "" ] ; then
    echo "tcsh found - switching..."
    exec tcsh -l
fi
```

Close your ssh session and open a new session, when you login again your default shell is going to be `tcsh`.

## 2.2 Enable EDA tools in your environment.

Create/modify the file `~/.cshrc` with the following content:

```
source /<476_FOLDER>/common/476_cshrc.csh
```

This command allow you to use the EDA tools and configure the FreePDK45 in your environment.

## 3 Setup for Cadence Virtuoso

In this section, we will walk through the set up for Cadence Virtuoso and the FreePDK45 development environment. Make a directory for Cadence, and your project.

### 3.1 Cadence Directory

**Note: The steps in this section only need to be performed once!**

Cadence generates many annoying files and logs, so we will create a single directory from which it should always be run from:

```
mkdir <476_FOLDER>/<NETID>/cadence
```

This will create a new folder named **cadence** in the directory `/<476_FOLDER>/<NETID>/`. The **cadence** directory will be used to hold the libraries for all your projects (allowing you to re-use designs from older CAD assignments should you choose).

Run `cadence_setup/set_freepdk45_cadence.csh` (while in cadence directory)

- ECE Linux servers:

```
/home/projects/ee476.2022aut/common/cadence_setup/set_freepdk45_cadence.csh
```

- HYAK servers:

```
/gscratch/ece/courses/476_aut2022/common/cadence_setup/set_freepdk45_cadence.csh
```

Your cadence directory should have the following content:

```
[cadence] ls
CAD0
CDS.log
CDS.log.1
CDS.log.1.cdslck
CDS.log.cdslck
SKILL
cad0
calibreDRC.rul
calibreLVS.rul
calibrexRC.rul
cds.lib
display.drf
freepdk45.1
general.il
layoutSpecs.il
```

```
leSchBindKeys.il
lib.defs
libManager.log
libManager.log.1
libManager.log.1.cdslck
libManager.log.cdslck
netLister.il
postProcessAbstract.il
schBindKeys.il
technology.tf
```

You should also create a directory inside your `<NETID>` directory for each project or tutorial. For this tutorial, we will create a folder named `cad0`.

```
cd /<476_FOLDER>/<NETID>/
mkdir cad0
```

You can also create a soft-link to the cadence directory to make navigation easier:

```
cd /homes/projects/ee476/<NETID>/cad0
ln -s ../cadence ./cadence
```

[OPTIONAL] To make it easier to navigate between your `/<476_FOLDER>/<NETID>` directory and your home, go to your home directory (`cd ~`) and create a soft-link to your ee476 directory:

```
cd ~
ln -s <476_FOLDER>/<NETID> ./ee476
```

Now, after opening a new terminal (which starts in the home directory) you can enter `cd ee476`, instead of a longer path, such as `cd /homes/projects/ee476/<NETID>`. To see what a soft-link points to, try `ls -al`.

## 3.2 Start Cadence Virtuoso

Once the tasks above are complete, the following command will start Cadence:

```
pwd
/<476_FOLDER>/<NETID>/cadence
virtuoso &
```

Be sure to always start Cadence from the above directory, or it will not see your files.

# 4 Overview of Cadence Virtuoso

## 4.1 CIW (Command Interpreter Window)

The Command Interpreter Window (CIW) is the main window of Cadence Virtuoso. You can think of it as the “shell” for a Cadence Virtuoso session. Advanced users may invoke commands in the CIW directly (using a language called SKILL), but this class will not involve the direct use of SKILL commands. The CIW also prints out error reports and logs command output. For the sake of this class, it can be used to launch the Library Manager (under the Tools menu), and to set user preferences.

## 4.2 The Library Manager

The Library Manager is the central file organization utility in Cadence. From the Library Manager, you will access and navigate all of the different design files you create and edit in Virtuoso. Files in Cadence are organized into a hierarchy of libraries, cells, and views:

Abstractly, a **Library** is a collection of cells (a.k.a. designs). In Virtuoso, all of the design files for a library are kept in a folder of the same name. For instance, if a library named ‘cad0’ is created, there will actually be a folder named ‘cad0’ in the ‘cadence’ directory that contains the data files for the cells in the ‘cad0’ library. There are two main types of libraries: technology libraries, and design libraries. A technology library is provided by the foundry, and usually contains a large number of basic building blocks (or “primitives”). Users of the technology library can use

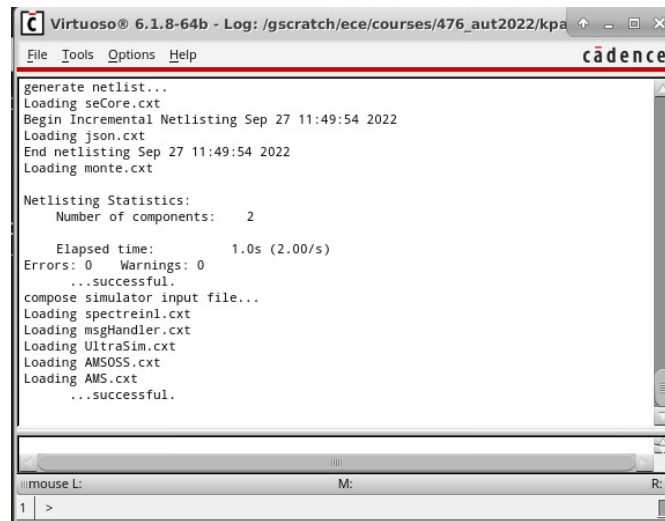


Figure 1: Virtuoso - Command Interpreter Window

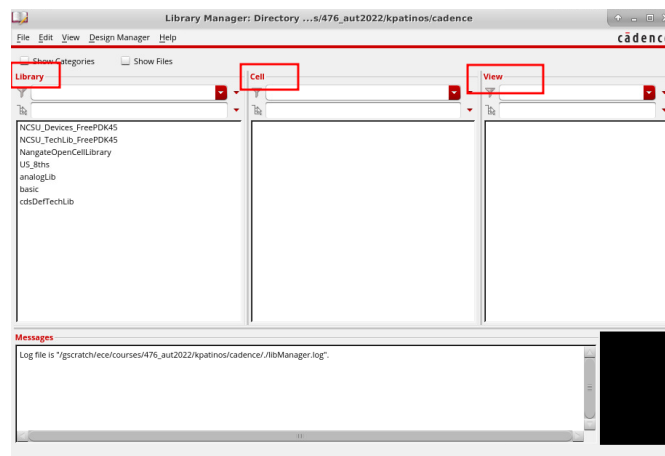


Figure 2: Virtuoso - Library Manager

the primitives to build their own designs, which are stored in a design library. This distinction is entirely conceptual, and designs/cells can be moved or copied across libraries, and designs can freely instantiate components from other libraries.

A **Cell** is a single circuit. As with libraries, the data for a Cell can actually be found in a single folder within the library. However, circuits can have different representations, which in Virtuoso are called Views.

An abstract representation of a Cell is called a **View**. Examples of a representation include a symbol, schematic, or mask layout. In Library Manager, the rightmost column lists all the views for the selected Cell.

### 4.3 Editors

This course will utilize 3 editors, which are used to design the views for **Symbols**, **Schematics**, and **Layouts**.

The inverter gate **TINV\_X1** from the **NandgateOpenCellLibrary** library will be used to introduce the three editors.

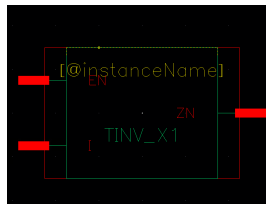


Figure 3: The symbol view for the TINV\_X1 cell

A **Symbol** is a view of a cell that can be instantiated in schematics. The symbol for the TINV\_X1 gate is shown above in Figure 3. (You can find this symbol in **NandgateOpenCellLibrary** library, under **TINV\_X1** -> symbol)

A **schematic** is a combination of symbols, wires, and pins. For the TINV\_X1 gate example, the original schematic is shown in the Figure 4. (You can find this schematic view from **NandgateOpenCellLibrary** -> **TINV\_X1** -> **schematic**.)

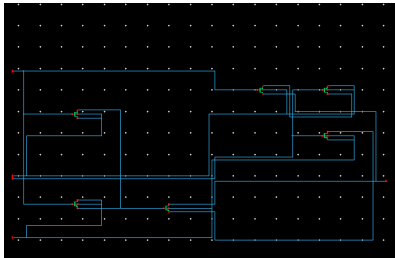


Figure 4: A automatic gate schematic

The schematic in Figure 4 looks garbled because it was automatically generated, but a user-designed schematic would look like that in Figure 5.

A layout drawing (or “mask layout”) is the collection of polygon shapes that will be used to create a set of photolithographic masks. The geometry information in the layout drawings is the data that is required by the foundry to create a set of masks, and actually manufacture an IC. As with a schematic, a layout can contain instances of other layouts. To open the window shown in Figure 6, open the view at **NandgateOpenCellLibrary** -> **TINV\_X1** -> **layout**.

### 4.4 Analog Design

Circuit netlist generation is performed in Analog Design Environment (ADE). Simulation profiles can be constructed to run simulations for transient analysis, DC analysis, and more. Some of the simulation types are especially suited for specific circuits, for instance oscillators or power amplifiers. However, for this class we will use ADE primarily for netlist generation, and use HSPICE to perform circuit simulation.

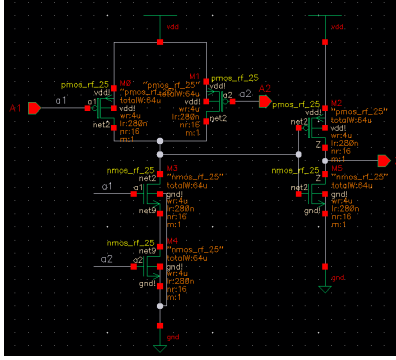


Figure 5: A human-readable gate schematic

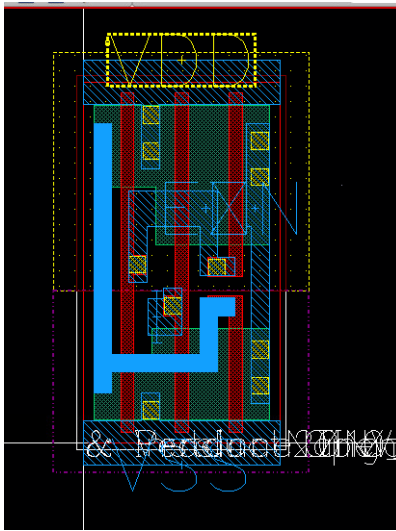


Figure 6: Gate layout view

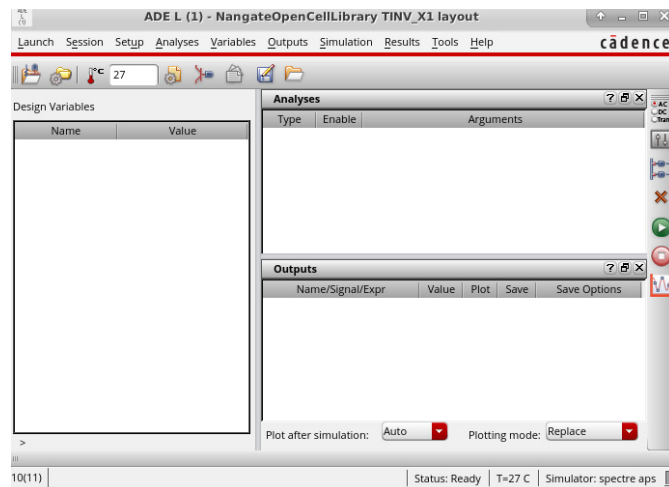


Figure 7: Analog Design Environment

## 4.5 The Command-Line Underbelly

Many of the key functions in Cadence have command line counterparts, or are command line tools that are launched from Cadence's GUI. This might be an academic point, if the command line wasn't so powerful.

Historically, both in academia and industry, Cadence and other IC development tools have been augmented by significant amounts of text-based, command-line tool operations. Being effective in integrated circuit design requires use of these tools.

In particular, there are great advantages to using command-line tools for simulation, wave display, and post-layout verification.

## 5 Basic Library Management

In this part of the tutorial, we will create a new library named 'cad0'. For course work and research, you may want to create other libraries to store new circuits, or to keep subsystems of a larger integrated circuit project separate from each other. If you set up Cadence Virtuoso properly, the Library Manager window will show up automatically when you open Virtuoso.

### 5.1 Open Library Manager

To open Library Manager from the CIW, select Tools -> Library Manager.

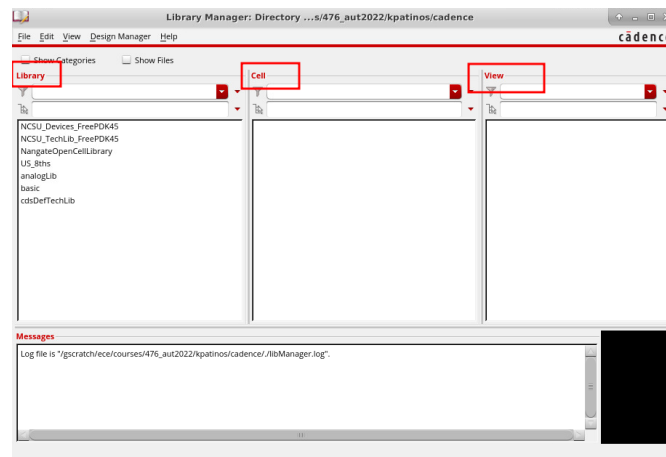


Figure 8: Virtuoso - Library Manager

By default, creating a new library creates a folder for that library inside 'cadence' (assuming you started Virtuoso from inside 'cadence'). All actions you perform on this library will appear in some form inside that folder. (Note: the libraries shown in your Library Manager may not match those shown in Figure 8)

### 5.2 Create a new library

In the main Library Manager window, select File -> New -> Library, and enter the name for the new library (for instance, 'cad0'). Select OK.

From the options in the window that appears (Figure 10), select Attach to an existing technology library and then OK.

From the dropdown menu, select **NandgateOpenCellLibrary** and hit OK.

## 6 Schematic Creation

Now that we have a new library, we will create a cell named 'rc\_series', and later use it to demonstrate some simulation techniques with HSPICE.

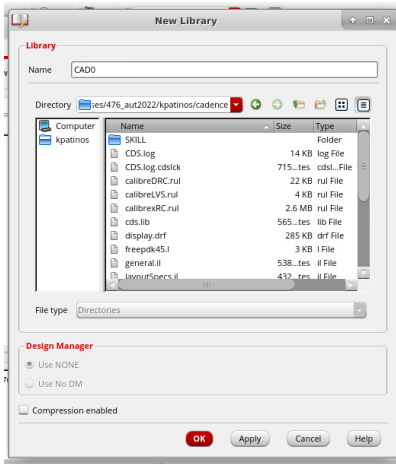


Figure 9: Virtuoso - Create New Library

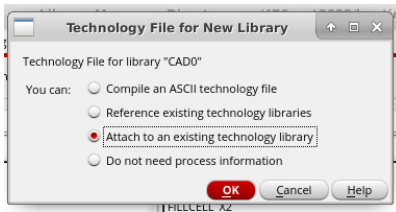


Figure 10: Virtuoso - Create New Library II

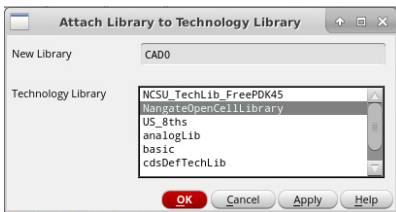


Figure 11: Virtuoso - Create New Library III



## 6.1 Create a Cell View

First, we will create a cell view. Make sure that 'cad0' is selected under **Library**, and select **File -> New Cellview**. Make sure **Schematic** is selected under Type.



Figure 12: Virtuoso - Create New Cell View

After typing the cell name, select **OK**. The schematic editor, with the new cell, should appear (Figure 13). If you get an warning about the proper license not being available (shown below), click **Always**.

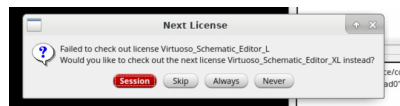


Figure 13: Virtuoso - warning about licenses

## 6.2 Create instances

First, insert the resistor, capacitor and ground into the schematic. You can do this in three ways: 1. Press the hot-key **i** 2. Select **Create -> Instance**

Option 1 is by far the best choice; while it may not be apparent at the moment, the use of hotkeys saves tremendous amounts of time.

In the window that appears after pressing **i**, select **Browse**, choose the **analogLib** library, and look for the cell named **res**. The lower portion of the Add Instance window shows the properties of the component. After selecting the resistor and filling in the field as per Figure 14, click **Hide**.

Place the resistor into the cell view, and use the **r** hotkey to rotate the resistor 90 degrees.

After placing the resistor symbol, place a capacitor and ground as shown in Figure 15 (found under the names **cap** and **vss**, in **analogLib**).

## 6.3 Connect wires

The next step is to connect the components in the schematic. You can use the hot-key 'w'.

## 6.4 Insert pins

The last step is to create pins for the circuit. Press the hot-key **p**, or select **Create-> Pin**.

Make sure that the value under **Direction** is set correctly! Circuits that swap inputs or outputs will not function correctly.

Repeat this process with **Vi**, but with **Direction** set to **input**. After placing the two pins and creating the pin connections, the circuit should look like that shown below (Figure 18).

(Note: make sure to use check and save, instead of just save)

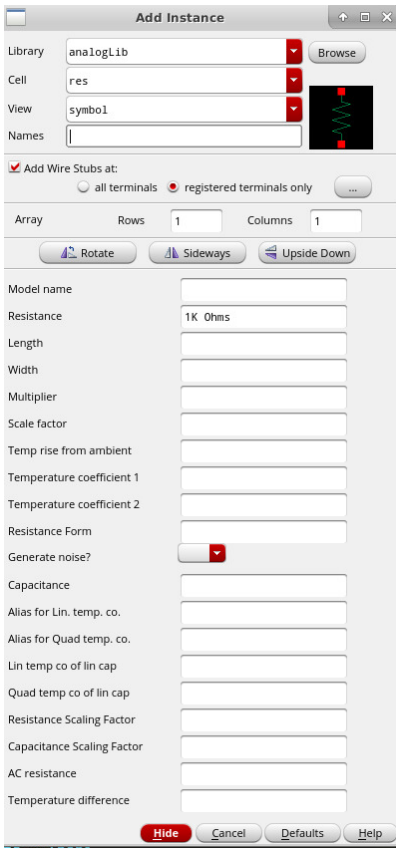


Figure 14: Virtuoso - “Add Instance” Window

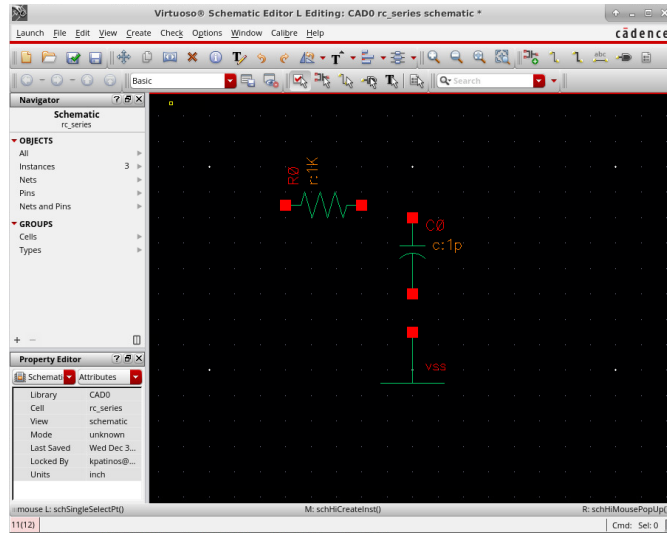


Figure 15: Virtuoso - Instance in Cell View

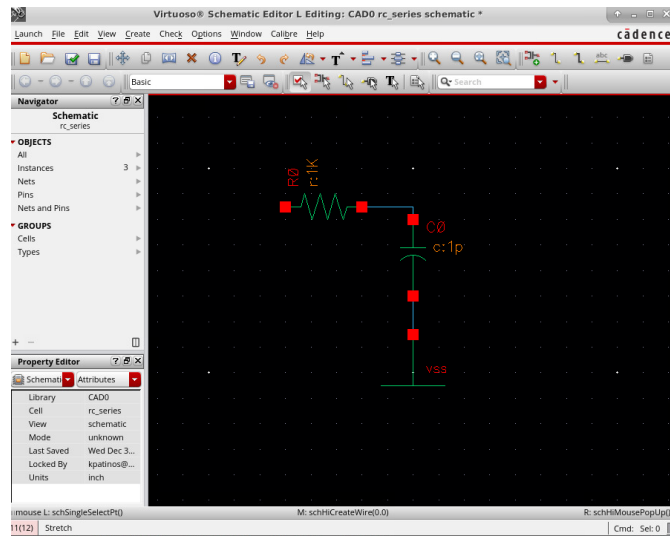


Figure 16: Virtuoso - Wire connections

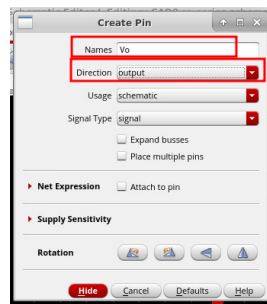


Figure 17: Virtuoso - Pin Window

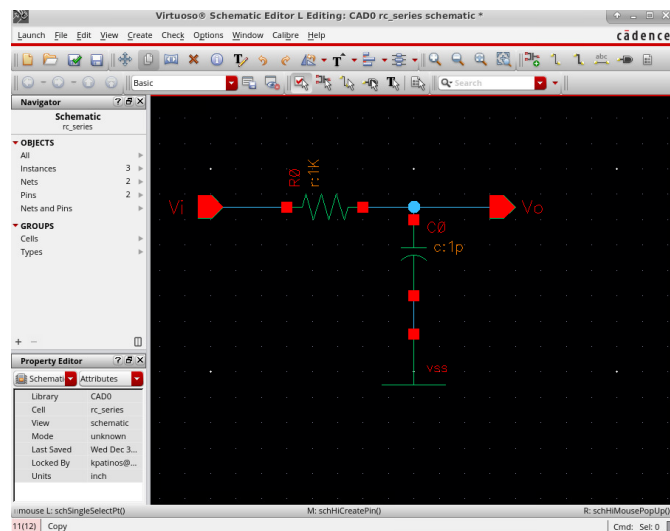


Figure 18: Virtuoso - The completed RC circuit

## 7 Simulation with HSPICE

### 7.1 Create simulation directories

It is good practice to create a new directory to contain all of the simulation files for a given circuit. For this tutorial, we will use the directory 'spice' to contain circuit netlists, and to contain the directory for simulations on the 'rc\_series' circuit.

```
$ pwd
<476_FOLDER>/<NETID>/cad0
$ mkdir spice
$ mkdir spice/netlists
$ mkdir spice/rc_series
```

### 7.2 Generate the circuit netlist (.ckt file)

From the schematic editor window, click **Launch** -> **ADE L**. In Analog Design Environment window that appears, select **Setup** -> **Simulator/Directory/Host/**. The simulator should be changed from 'spectre' to 'hspiceD'. For Project Directory, browse for the 'spice/netlists' directory we created above.

Now, select **Simulation**->**Netlist**, and select **Create** -> **Netlist**. If the netlist creation was successful, a file named 'input.ckt' will be generated at `spice/netlists/rc_series/hspiceD/schematic/netlist/input.ckt`.

To avoid having to move or copy the netlist every time the schematic is changed, create a soft-link from the netlist to the simulation directory:

```
$ pwd
/<476_FOLDER>/<NETID>/cad0/spice/rc_series
$ ln -s ../netlists/rc_series/hspiceD/schematic/netlist/input.ckt ./rc_series.ckt
```

### 7.3 Get the example control file

After generating the netlist we are ready to perform simulations. There are a variety of ways to organize an HSPICE simulation, but the recommended flow for the class involves using a single control file and one or more netlist files. A control file contains commands to specify the type of simulation (transient/ac/dc sweep etc.), create test stimulus, and take automatic measurements.

For this tutorial we have created a control file for you. Copy it from the common directory with:

```
$ pwd
/<476_FOLDER>/<NETID>/cad0/spice/rc_series
$ cp /homes/projects/ee476/common/spice/tutorial_1/rc_series.ct1 ./.
```

### 7.4 Run HSPICE

At this point, the netlist and the control file should both be in the simulation directory that was created for the 'rc\_series' circuit.

```
$ pwd
<476_FOLDER>/<NETID>/cad0/spice/rc_series
$ ls
rc_series.ckt rc_series.ct1
```

You can now run HSPICE with the following:

```
hspice rc_series.ct1
```

The command will generate many lines of output at the terminal; to redirect the output to a new file, we can run `hspice rc_series.ct1 -o rc_series.lis`

You should see **hspice concluded** at the command prompt. If for any reason HSPICE fails, search the terminal output (or log file) for any lines containing the word **Error**.

## 8 Viewing Waveforms in Silicon Explorer

### 8.1 Open Silicon Explorer (SX)

Running HSPICE will have generated about a half-dozen files, which are used to record different simulation results or logs. To open the transient waveform file, run the following:

```
sx -w rc_series.tr0 &
```

A window similar to that in Figure 19 should appear. In the top left-hand panel, expand the signal list, select `toplevel`, and drag-and-drop signals to the waveform viewer.

We have run commands followed by `&` several times. The ampersand is a special command that tells the OS to run the application independently (in the “background”), so that the terminal can still be used while the application is running.

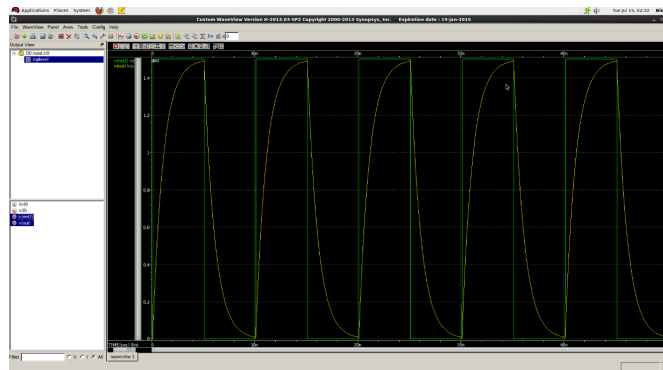



Figure 19: SX - RC waveform

### 8.2 Add waveform cursors

Cursors can be added into the waveform viewer by clicking the button. The value of a signal where it intersects with the cursor is displayed on the left hand side of the SX window. You may need to resize the portion of the window containing the signals, in order to see the signal names and numeric values.

### 8.3 Add measurement tools

Various types of signal measurements can be added to the waveform viewer by selecting the measurement tool () , which is located in a toolbar near the top of the waveform viewer. The window that appears should look like that in Figure 20. The measurement tool is very handy – experiment with its various functions liberally.

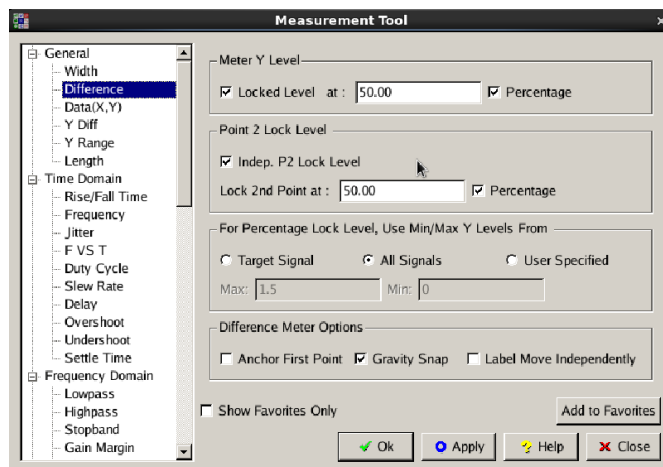


Figure 20: SX - Measurement Tool Interface

To measure the delay time of the 'rc\_series' circuit, select 'Difference' in the left panel, and mark the checkboxes for 'Locked level at' and 'Indep. P2 Lock Level'. Enter '50.00' for both values and mark the 'Percentage' boxes. Select 'Ok', you should be able to drag the measurement tool and snap two points onto the 50% level of two different signals. The delta indicates the time difference between the two snapping points.

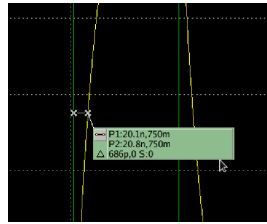


Figure 21: SX - Delay Measurement

## 8.4 Build equations

Clicking the Function button will open the window shown in Figure 22. This tool, called the Equation Builder, can be used to perform various mathematical functions on signals to aid in on-the-fly analysis.

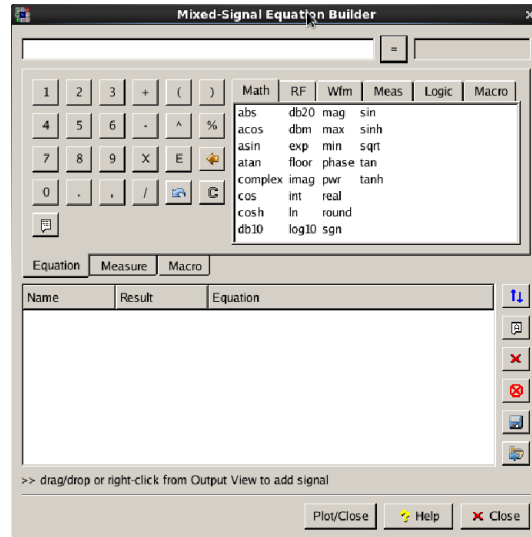


Figure 22: SX - Measurement Tool Interface

For example, to measure the total charge delivered or received over some time period, you could integrate current with the integral function. Click the tab labeled Wfm, and select the integral function. In the space for equation entry near the top of the Equation Builder window, type `integral(i(v0))`. Then select **Plot/Close**. Note that, `i(v0)` is the current through voltage source `v0`, and the expression may be different if you define your voltage source differently. Alternatively, try dragging a signal from the bottom left panel of Silicon Explorer into the space for equation entry.

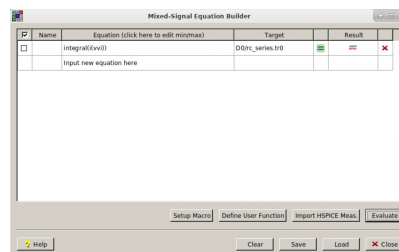


Figure 23: SX - Measurement Tool Interface III

For the integral operation above, the equation builder should create a new signal in the waveform viewer that corresponds to the result of the function.

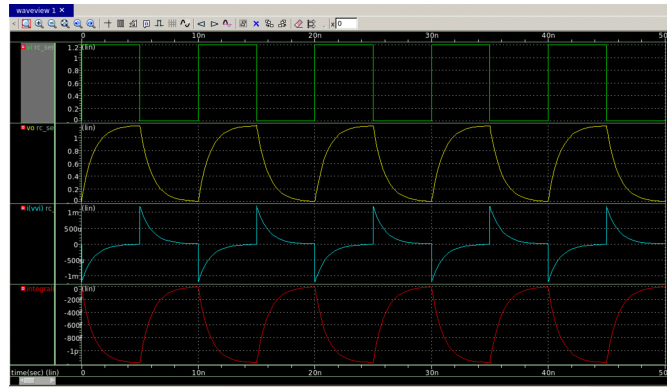


Figure 24: SX - Tutorial 1 Waveform results