LUND
UNIVERSITY

Electrical and Information Technology

Integrated A/D and D/A Converters (ETIN55)

Lab2 Manual

Martin Anderson
Pietro Andreani

February/October 2021

Lab 2 – Analog Modeling with Verilog-A

This lab will introduce the procedure of modelling and simulating analog circuits using the hardware describing language **Verilog-A**. Your behavioral code will be simulated using the analog circuit simulator *spectre*, which is the same simulator that is used for simulating analog circuits in Cadence.

Laboratory report

Include all plotted simulation results, the schematics and answers to all questions in your laboratory report.

Homework exercises

1. Go through the slides from the lab seminar.
2. Read carefully the tasks to be performed in this lab.
3. Download the VerilogA manual from /usr/local-eit/cad2/cadence/ic616/doc/verilogamsref/verilogamsref.pdf, and read the sections relevant to this lab.
4. A parallel RLC resonator circuit has R = 10 kΩ, C = 10 pF and L = 100nH. At what frequency will the impedance be peaking?
5. Write down the truth table of a NAND gate.

Laboratory exercises

Part 1: Model and simulate an RLC resonance circuit

A parallel resonance circuit consists of a resistor, a capacitor and an inductor in parallel. In this part, you shall model the behavior (impedance) of this circuit using Verilog-A.

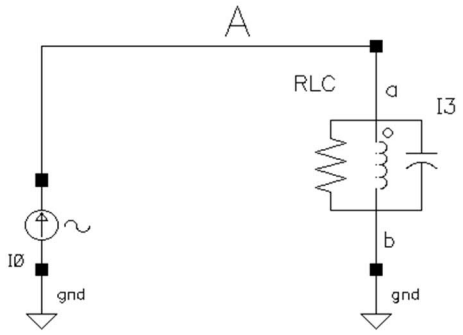

Figure 1. RLC resonator test bench.

1. Preliminary steps:
    a. Log into a UNIX workstation.
    b. Create a Cadence directory in your home directory, for example CDSLAB, and descend into it:

```
> mkdir CDSLAB
> cd CDSLAB
```

    c. Download the zip file VerilogALab.zip from the course homepage, and unzip it in your Cadence directory:

```
> unzip VerilogALab.zip
```

    d. If you are working on a UNIX workstation in the undergraduate lab or in the research lab: while in your Cadence directory, start **Cadence**:

```
> inittde adda21
> virtuoso &
```

    e. If a window appears asking "**Would you like to try checking out the Licence Virtuoso_Schematic_Editor_XL**" (or something analogous), click on **Yes** (repeatedly, if necessary).
    f. Select *Tools > Library Manager...* in the **Virtuoso** window to open the *Library Manager* window if it did not start automatically. Add the **VerilogALab** library to your library path in the *Library Manager* with *Edit > Library Path...* Save your changes to the library path when done.
2. Create a **Verilog-A** model of an **RLC circuit**:
    a. Create the **RLC instance** in the **VerilogALab** library by selecting *File > New > Cell view...* in the **Library Manager**. Name your circuit **RLC** and select **veriloga** in the **View** field by choosing **verilogA** in the scroll-down window for **Type**. Then click **OK**.
    b. In the **emacs** window that pops up, enter the **veriloga** code for the RLC resonance circuit:

```
// VerilogA for VerilogALabSol, RLC, veriloga
```

```
‘include "constants.vams”
‘include "disciplines.vams"
module RLC(a,b);
inout a,b;
electrical a,b;
parameter real R = 10k;
parameter real L = 0.1u;
parameter real C = 10p;
// Model the behavior here !!!
endmodule
```

c. To compile the code, save it and close the **emacs** window. If there are errors, you will be asked if you want to correct them. Answer *yes*.

d. After editing and correcting all the errors, save the file and quit. A message “*Cellview RLC symbol does not exist*” will appear. Choose *yes* to generate the symbol view of instance **RLC**. Specify the pins ***a*** and ***b*** in the new window (Figure 2) and generate the symbol. Click **OK**.

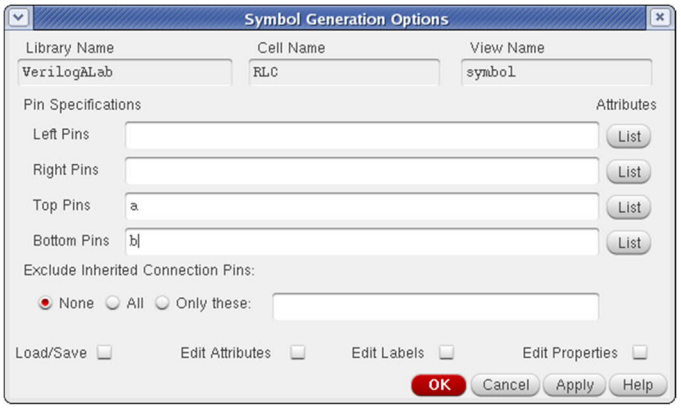

Figure 2. The **Symbol Generation Options** window.

e. In the **Symbol Editor** view, you can modify the symbol as you like. For example, you can reuse already existing symbols by choosing *Create* > *Import Symbol...* and selecting the symbol of the cell you want to use. For the **RLC** circuit, combine the symbol views of the **res**, **cap** and **ind** cells of the **analogLib** library. Remove all pins, *cds parameters* etc (i.e., everything except the drawings in green) from the imported symbols. Be sure to use your own pins, ***a*** and ***b***. Save your symbol and quit.

3. Simulate the **RLC** circuit and verify the operation of your model. Since it is a *Verilog-A* module, *spectre* can be used to simulate it:

a. To simulate the **veriloga** description of the **RLC** circuit, a test bench shall be used. Create a new cell view **RLC_Tb** and create a **schematic** view of it. Instantiate a current source **isin** and a ground **gnd** from the library **analogLib**. Instantiate your **RLC** circuit from your own design library. Set the **AC Magnitude** of the current source to 1 (A). The resulting test bench is shown in Figure 1.

b. In the **Schematic** window, select *Launch* > *ADE L*. If a **Next License** window appears, click on **Yes** (repeatedly, if necessary). In the **Virtuoso Analog Design Environment**

(**ADE**) window, make sure that **verilоga** is in the **Switch View List** by clicking *Setup* > *Environment...* and adding **verilоga** in the **Switch View List**, if needed.

c. Click on *Analysis* > *Choose...* and select an **ac** analysis from 1Hz to 300MHz according to the following: the *Sweep Variable* is **Frequency** (default); the *Sweep Range* is 1 (**Start**) to 300M (**Stop**); choose *Sweep Type* > *Linear* and set the **Number of Steps** to 1000 (double-check that you did not select **Step Size** instead of **Number of Steps**).

d. In **ADE**, select *Outputs* > *To be plotted* > *Select on Schematic*. Select the **A** net (and not the **a** node!) in the **schematic**. Does your **ADE** look like the one below?

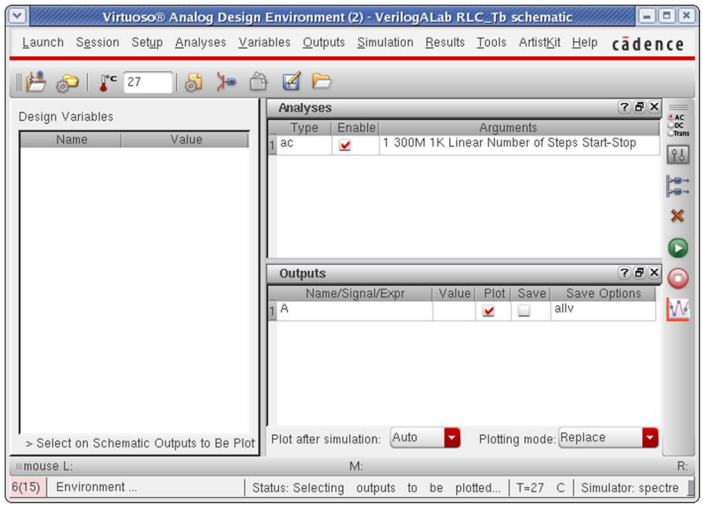

Figure 3. The ADE (Analog Design Environment) window for AC simulation of the RLC circuit.

e. Click *Simulation* > *Netlist and Run* (green button on the right margin of the ADE window) to run the simulation. The results should look like in Figure 4.

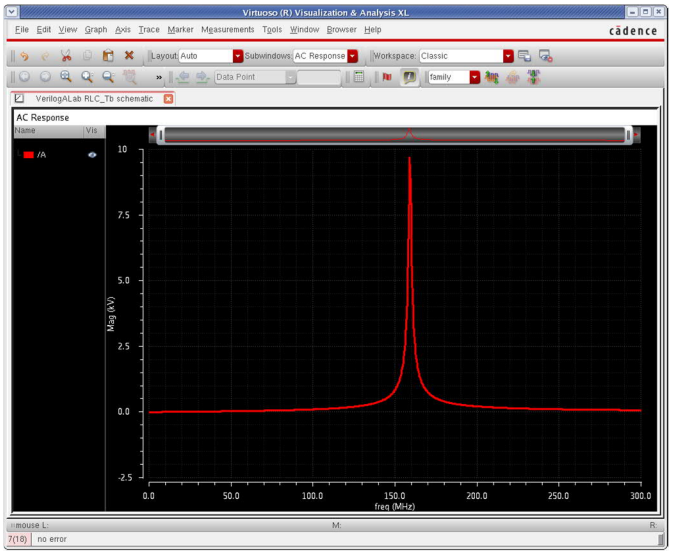

Figure 4. Simulation result for the RLC circuit.

f. Add a 10Ω parasitic resistance in series with the capacitor and 5Ω in series with the inductor by changing your **verilоga** code, introducing two extra nodes and writing KCL equations for them as well. Rerun the simulation. Include the simulation results in your report.

Part 2: Modeling a digital NAND gate

Digital functionality can also be modeled using **Verilog-A**. In this lab module, you will model a digital **NAND** gate with the possibility of specifying delay time, rise time, and fall time.

The NAND gate shall have the two inputs **A** and **B** and the output **Y**. Use two integer variables, **logica** and **logicb**, to save the state of the **A** and **B** inputs. When **V(A)** or **V(B)** cross the threshold on a rising edge, **logica** or **logicb = 1**. When **V(A)** or **V(B)** cross the threshold on a falling edge, **logica** or **logicb = 0**.

Recall that the function **@(cross(V(A) - thresh, +1))** detects when the voltage on **A** crosses the threshold **thresh** on a rising edge. If the event is reached, the action is executed.

Use the **transition** statement to describe the output **Y**, which has delay time **td**, rise time **tr**, and fall time **tf**. The defaults shall be **td=2n**, **tr=1n**, **tf=1n**.

4. Create the **veriloga** view of cell **NAND1**:
    a. Click *File > New > Cellview* in the Cadence **Virtuoso** window. Select **NAND1** as the cell name and **veriloga** as view name. Cancel the **Invoke Modelwriter** form if it appears.
    b. Write the code for the NAND gate. Let the code below guide you.

```
// VerilogA for VerilogALab, NAND1, veriloga

`include "constants.vams"
`include "disciplines.vams"

module NAND1(A,B,Y);

inout A,B;
output Y;
electrical A,B,Y;

parameter real vlogic_high = 1.0;
parameter real vlogic_low = 0.0;
parameter real td = 2n, tr=1n, tf=1n;

Write the veriloga code modeling the NAND here!

endmodule
```

    c. Save the changes and close the text editor. If there are no errors, a window appears and asks you if you want to create the associated symbol. Click **Yes**.
    d. Modify the **NAND1** symbol if you are not satisfied by the default. Save the symbol and exit the symbol editor.

5. Create a test bench for your design:
    e. Create the cell **NAND1_Tb**, view **schematic**.
    f. A blank schematic window appears. Instantiate your **NAND1** cell, an output pin, and two **vpwlf** voltage generators and a **gnd** from **analogLib**. Add the wire names **Ain** and **Bin**. The schematic should look like the one in Figure 5.
    g. To set the properties of the **NAND1**: click on the instance and press **q** on the keyboard and select **veriloga** for **CDF Parameter of view** close to the bottom of the window (the default may be **Use Tools Filter**). Set **td=4ns**, **tr=5ns**, **tf=10ns**.
    h. For the **V0** voltage generator, write **Apwl** in the field **Pwl file name**. For **V1**, write **Bpwl** in the same field. The files Apwl and Bpwl are found in the **stim/** directory (which was contained in VerilogALab.zip).

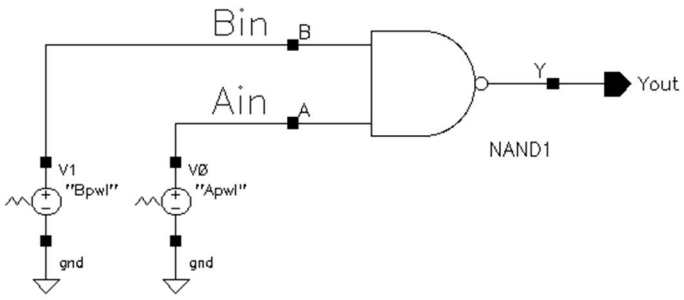

Figure 5. Test bench schematic for **NAND1**.

6. Simulate your test bench:
    a. Start the **ADE** from your test bench schematic.
    b. Set up a **tran** (transient) analysis with **Stop Time** of **6us**, with **moderate** accuracy.
    c. Set up a relative path to the stimulus files by selecting *Setup* > *Simulation Files* ... in the **ADE** and adding **./stim** to the **Include Path** field.
    d. Set up the outputs to be plotted with *Outputs* > *To Be Plotted* > *Select On Schematic*. Select the **Ain**, **Bin** and **Yout** nets. Press *ESC* while cursor is still inside the schematic.
    e. Select *Simulation* > *Netlist* and *Run* to start the simulation.
    f. When the simulation finishes, the transient signals at **Ain**, **Bin** and **Yout** are plotted. Verify that the operation of **NAND1** is correct, including delays and rise/fall times. Repeat the simulation with **td=40ns** and **tr=tf=100ns**, and check that the plots change as expected. Save the result for your lab report.

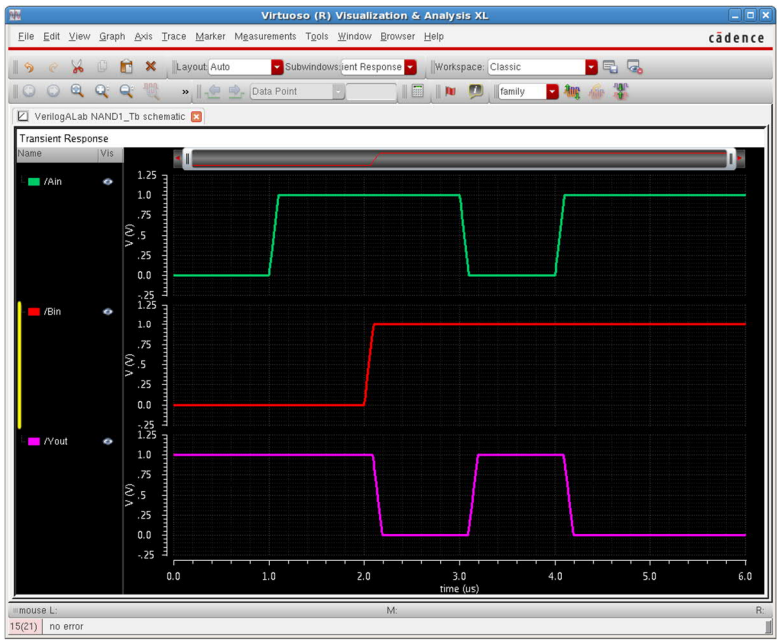

Figure 6. Simulation results for **NAND1** (use *Graph* > *Split Current Strip* in the **Visualization** window to plot the three waveforms as above – by default, they share the same y-axis)

Part 3: Creating a frequency measuring device

In the following, you will create a circuit module that measures the frequency of a single-tone input signal. You will plot the results and write the value of the measured frequency to a file. Furthermore, you will use this module to measure the frequency of a VCO.

Your module shall have three terminals: **vp**, **vn** and **fout**. The frequency of the signal between **vp** and **vn** will be measured and sent to **fout**. Make this measurement by detecting the times at which the last two zero crossings occurred. You may want to use some real variables inside your **veriloga** description.

For a frequency meter, you could just output a voltage proportional to the frequency by measuring the period of the input signal and inverting the value. However, to show how **natures** and **disciplines** work, in this lab you will use a new **nature** *Frequency*, and a new **discipline** *freq_current*. **Verilog-A** uses access functions to output information. Normally, those are **V** for voltage and **I** for current. *Frequency* has to have its own access function, which is called **FF** in the definitions below. These definitions could be added to the *disciplines.vams* file, but in this lab they will be added within the module itself.

```
nature Frequency
        abstol = 1m;
        access = FF;
        units = “Hz”;
endnature

discipline freq_current
        potential Frequency;
        flow Current;
enddiscipline
```

In addition, your module will use an integer flag, **log_to_file**, to determine when to write the output data to a file. When **log_to_file=0**, no data log file is written. When **log_to_file=1**, the output time and frequency data are written to the file “**%C:r.dat**”, which is
~/simulation/FreqMeter_Tb/spectre/schematic/netlist/input.dat

Insert a header at the top of your file, such as “**#Time/Freq data from module ‘%M’**”. Insert also a closing line at the end of it, and close it at the end of your circuit module.

More specifically, do the following:

7. Define the **FreqMeter veriloga** module:
    a. Create a **veriloga** view for a cell named **FreqMete**r.
    b. Use and modify the veriloga code below by filling in the comments in **boldface** with suitable code.

```
// VerilogA for VerilogALab, FreqMeter, veriloga
 ‘include "constants.vams”
 ‘include "disciplines.vams"
nature Frequency
    abstol = 1m;
    access = FF;
    units = “Hz”;
endnature
```

```
discipline freq_current
    potential Frequency;
    flow Current;
enddiscipline

module FreqMeter(vp, vn, fout);
    electrical vp,vn;
    freq_current fout;
    parameter integer log_to_file = 0;
    // define local variables here !!!
 analog begin
        @(initial_step) begin
            if (log_to_file) begin
            // create the open file statements here !!!
            end
        end
        @(cross(V(vp,vn),+1)) begin
       // setup the crossing statements here !!!
            if (log_to_file) begin
            // log the output to your file here !!!
            // use the function $fstrobe(file,"formats",variables)
            // E.g., format %-.10g for each variable gives a
            // precision of 10 digits
            end
         end
    // contribute the output to fout here !!!
 end
endmodule
```

c. Save your code and exit the text editor. Iterate until the **Verilog-A** syntax check is ok.

d. When all errors are fixed, create the symbol with **vp** and **vn** to the left and **fout** to the right. Save the symbol and close the symbol editor.

8. Create a test bench for the frequency meter:

   a. Create a new cell called **FreqMeter_Tb** with view name schematic.

   b. Instantiate the **FreqMeter** cell you just created and the **SweptSine** component from the current **VerilogALab** library; build your schematic as in Figure 7.

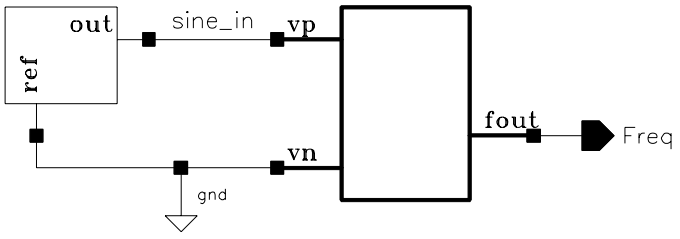

Figure 7. Test bench schematic for **FreqMeter**.

c. Set **log_to_file=1** for the **FreqMeter** (click on the instance, press **q** and select **veriloga** for **CDF Parameter of view)**.

d. Set **StartFreq=1k** and **HertzPerSecond= 10k** for the **SweptSine** instance.

e. *Check and Save* the schematic.

9. Simulate the frequency meter:

a. Start the **ADE**, and set up a transient analysis of **100ms**.

b. Set up the outputs to be plotted by clicking *Outputs > To Be Plotted > Select On Schematic* and select the **sine_in** and **Freq** signals.

c. Select *Simulation > Netlist* and *Run* to run the simulation.

d. In the **Visualization** window, check that the frequency at **100ms** is approximately **2 kHz**, as expected. Check that all data has been written correctly in *~/simulation/FreqMeter_Tb/spectre/schematic/netlist/input.dat*.
The file should contain two columns, with simulation time and calculated frequency, respectively.

e. Save your results for your lab report.

Part 4: Create a VCO Verilog-A model

Open the **VCO** (voltage-controlled oscillator) cell and understand the algorithm described by the veriloga code.

10. Build a test bench to test the VCO:
    a. Create a new cell **VCO_Tb** with a **schematic** view.
    b. Instantiate the **VCO** and the **FreqMeter** from your lab library as well as one **vpwl** voltage generator and a **gnd** symbol from the **analogLib** library. The schematic should look like in Figure 8.

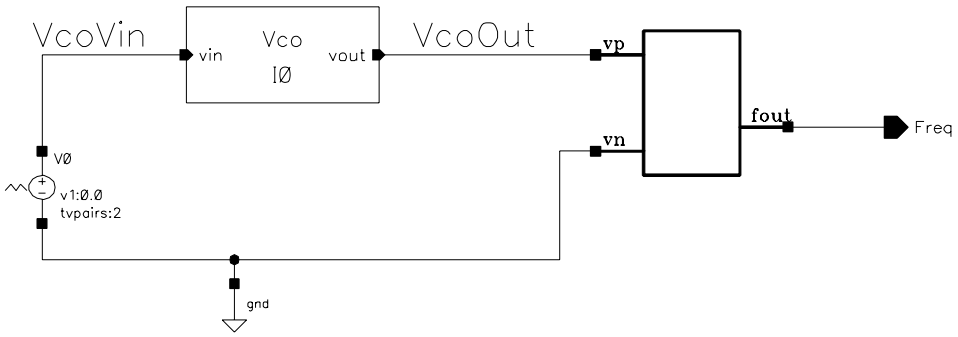

Figure 8. Test bench schematic for VCO.

    c. Set the properties of the **vpwl** source: **Number of pairs of points** is **2**, **Time 1** is **0**, **Voltage 1** is **-1.0**, **Time 2** is **1**, and **Voltage 2** is **1.0**.
    d. Set the properties of the VCO instance: **vco_amp=1.0**, **vco_cf=1.5k**, **vco_gain=1.0k** and **vco_ppc=40**.
    e. Set the properties of the **FreqMeter**: **log_to_file=0**.

11. Simulate the VCO:
    a. Start the **ADE** and set up a transient analysis with **Stop Time = 1s**, with **Accuracy** set to **conservative**.
    b. Click *Outputs > To Be Plotted > Select on Schematic*. Select the **VcoVin**, **VcoOut** and **Freq** nets. Use the **ESC** key to end the selection process.
    f. Select *Simulation > Netlist* and *Run* to start the simulation.
    g. In the waveform window, check the frequency as you did in the previous lab. It should measure **500Hz** at **-1.0V** and **2500Hz** at **+1.0V** and look like in Figure 9 (to visualize three distinct waveforms, use the command *Graph > Split Current Strip* in the **Visualization** window).

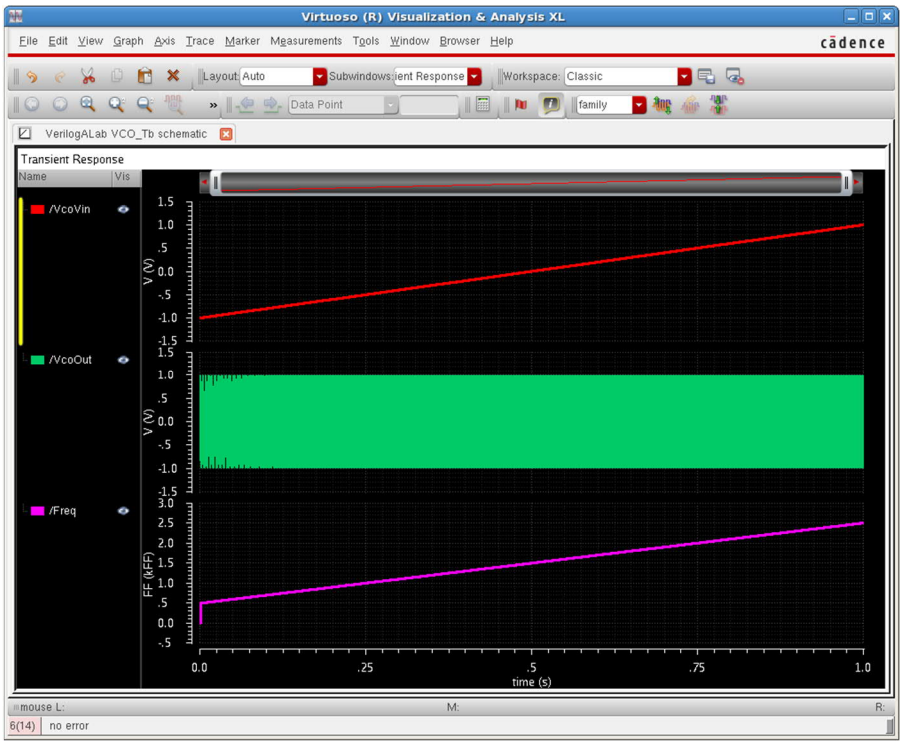

Figure 9. Simulation results for **VCO**.

Part 5: Create and simulate a look-up table model of a diode

In this part, you will extract a look-up table and write a veriloga look-up table based model of a diode. The table model will then be compared to the diode from the **cmos065** library.

12. Create the diode test bench:
    a. Create a new cell view called **Diode_Tb** with a schematic view. Instantiate a **vdc** from **analogLib** and a **dnlvtgp** N+/Pwell diode from the library **cmos065**. Set both **Length** and **Width** of the diode to **1**um. It should look like in Figure 10.

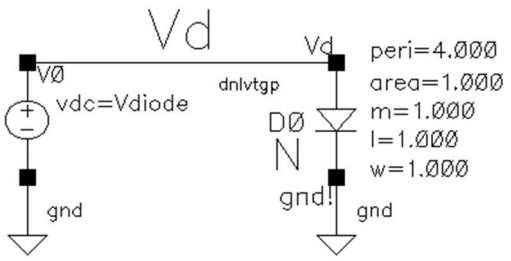

Figure 10. Test bench schematic for the **dnlvtgp** diode table extraction.

    b. Set the **DC Voltage** of the voltage source to a variable **Vdiode** that you can sweep in the simulation.

13. Simulate test bench to extract the table model:
    a. Start the **ADE** and import the design variables by clicking on *Variables* > *Copy from Cellview*. Set **Vdiode** to **0mV**.
    b. Setup a **dc analysis**. Click on **Design Variable** and type **Vdiode** in the **Variable Name** field. Sweep **Vdiode** from **0V** to **1V**; choose **Sweep Type** as **Linear** and set **Number of Steps** to **100**.
    c. Choose *Outputs* > *To Be Plotted* > *Select from Schematic* and select the net **Vd**, as well as the current entering the diode by clicking on the diode anode terminal.
    d. In the **ADE**, select *ArtistKit* > *Setup Corners…*; in the window that appears, find the **Select All** field, where there are three scroll-down menus. Under **GLOBAL VARIATIONS**, choose **<all typical>**; under **LOCAL VARIATIONS**, choose **no**; under **LIBRARY**, choose **DK**. Click now on the **Save Model File** button close to the top of the window, after which you close the window.
    e. Run the dc analysis with *Simulation* > *Netlist* and *Run*.
    f. Open the calculator by clicking *Tools* > *Calculator* ... in the **ADE**.
    g. Referring to Figure 11: Select the **wave** field and thereafter click on the diode current waveform in the **Visualization** window to import it into the Calculator; click on the field indicated in Figure 11 (red arrow on the right) to create a look-up table with the values of the diode current vs. the input voltage. The result of this operation is shown in Figure 12.
    h. Save the look-up table to a file with *File* > *Export* choosing *SPECTRE* as file type.

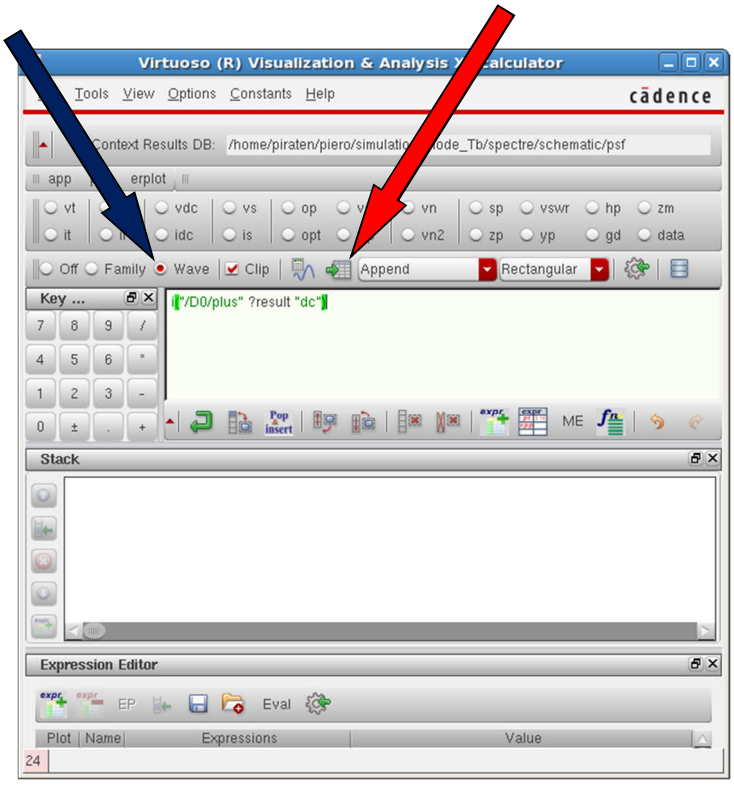

Figure 11. The **Calculator** window.

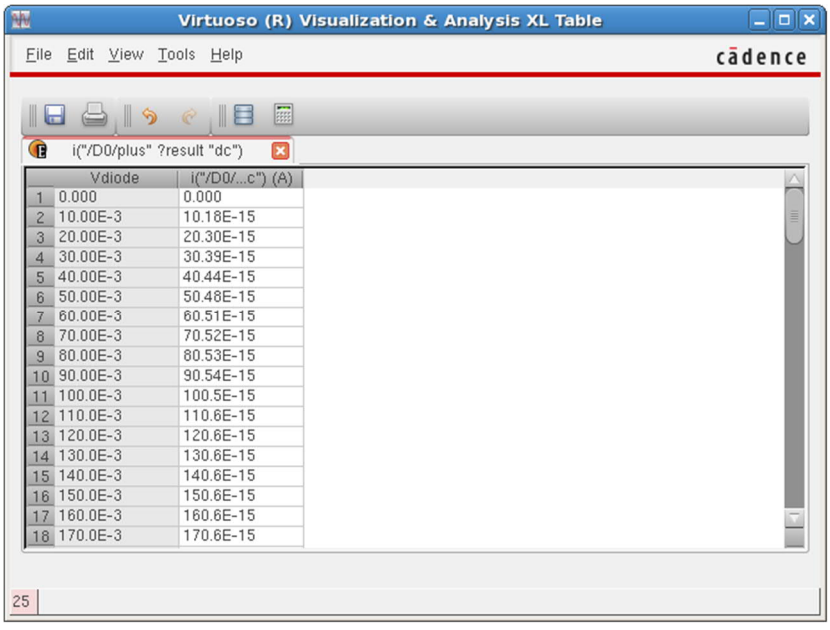

Figure 12. The **Table** window.

14. Write the **veriloga** diode model:
    a. Create a new cell called **MyDiode** with a **veriloga** view.
    b. Edit your **veriloga** code starting from the following module:

```
// VerilogA for VerilogALabSol, MyDiode, veriloga
```

```
`include "constants.vams"
`include "disciplines.vams"

module MyDiode(p,n);
inout p,n;
electrical p,n;
    analog begin
        // Model Id(Vd) here !!!
    end
endmodule.
```

c. Create a symbol by importing the **dnlvtgp** symbol from **cmos065** and edit it.

15. Update the test bench:

a. Open **Diode_Tb** and instantiate your **MyDiode** module into the test bench in parallel with **dnlvtgp**. The test bench should now look like in Figure 13.

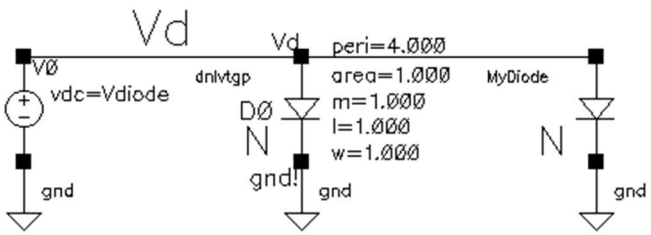

Figure 13. Test bench schematic for MyDiode.

16. Simulate the look-up table model:

a. Change the dc **Sweep Range** to **Automatic** (instead of **100 points**). The simulator will now interpolate between the values in the table. However, since the number of points in the table is relatively large, you will not see a large difference.

b. Select the anode current of **MyDiode** to be plotted as well.

c. Click on *Simulation > Netlist and Run* to run the simulation.

d. Check that the **dnlvtgp** and **MyDiode** currents are the same. They should look something like the plots in Figure 14 (the two curves are basically coincident).

e. Extend the dc sweep to **1.05V**, and use different ways of interpolating the current curve for the values beyond the look-up table limits (possible interpolation methods are **clamp**, **linear**, and three different **spline curves**). Compare the results.

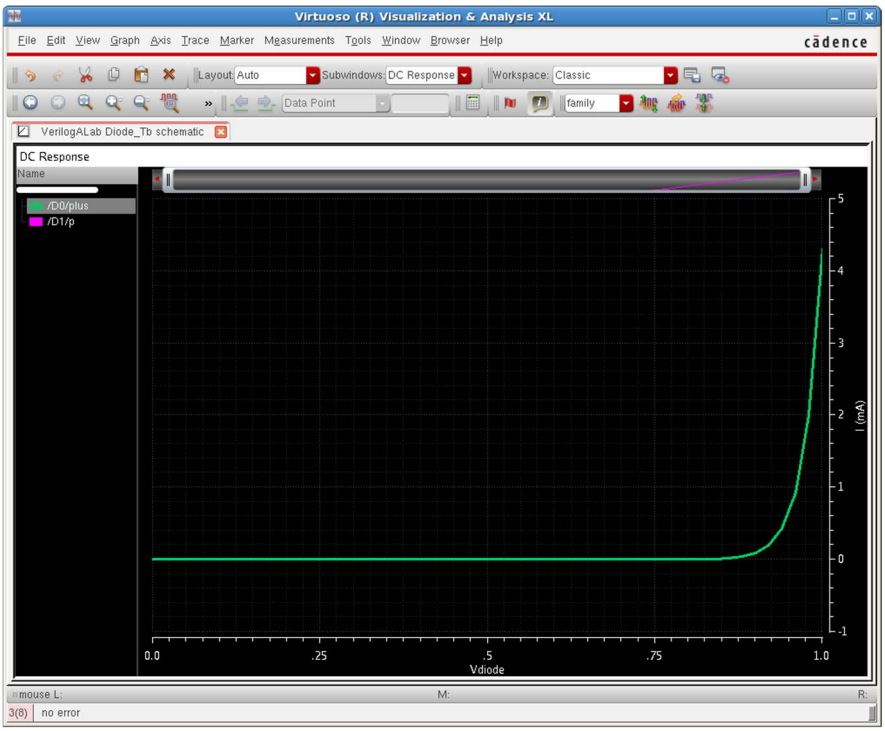

Figure 14. Simulation result for **dnlvtgp** and **MyDiode**.