Electronic circuit simulation with gEDA and NG-Spice by Example

Andreas Fester

May 25, 2004

Abstract

This article describes how to simulate electronic circuits using the open source packages gEDA (GNU Electronic Design Automation) and NG-Spice. It was written after I spent some time to get involved with these packages, and especially NG-Spice needed some effort until the first circuit was ready for simulation. The article is meant to be a tutorial-by-example, not a reference manual; there is a detailed reference manual available in the NG-spice distribution.

1 Some SPICE history

SPICE stands for Simulation Program with Integrated Circuit Emphasis. Originally, SPICE was implemented at the University of Berkeley. Other spice flavours have been derived from this implementation, and today there are several of them, for example winspice for the MS-Windows™ operating system, pspice, ltspice or tclspice. This article is about NG-spice, which has the goal to be a complete rewrite of the berkeley spice implementation. Currently it still contains code from berkeley spice, but has fixed a lot of bugs from the original code base. It is important to understand that each spice flavour might behave different in various areas; some of them are more compatible, others are less compatible. It is always necessary to read the documentation which came with the specific spice implementation. The NG-spice distribution comes with a detailed reference manual, but this article might help to learn where to start. It might be a good idea to have the NG-spice reference manual available when reading this article so that specific commands can be looked up in more detail.

2 Downloading and installing NG-Spice

For the gEDA package, several binary packages are available. Many distributions already contain those binary packages. For example, for Debian installing gEDA is as simple as **apt-get install geda**. One of the packages from testing or unstable should be used, because they are much newer then the package in stable.

The NG-spice package can easily be installed by compiling it from sources. After downloading the tarball, build a package for your specific distribution:

```
% tar xvzf ng-spice-rework-15.tgz
% cd ng-spice-rework-15
% ./configure --with-readline=yes
% make
% checkinstall
```

Make sure that the GNU readline library is enabled, this makes using the command line interface much more comfortable.

More information on how to compile NG-spice can be found in the NG-spice user manual which is part of the NG-spice distribution.

3 Step by Step example

This section contains a very simple step-by-step example to show how to simulate a circuit. Basically, the circuit will be drawn with **gschem** and then be simulated interactively with **ngspice**.

3.1 Drawing the circuit with gschem

The circuit we want to simulate is a simple RC filter. This example was choosen because it contains a very limited and therefore manageable number of components: a voltage source, a resistor and a capacitor.

Start **gschem** and draw the circuit shown in Figure 1. Using the schematics editor can be a littlebit confusing for the first time. Unlike other drawing applications, schematic editors tend to use an approach where the command has to be choosen first, and then the object on which the command is executed must be selected. Other graphics applications usually us an approach where the object is choosen and then the command to execute on these objects is selected.

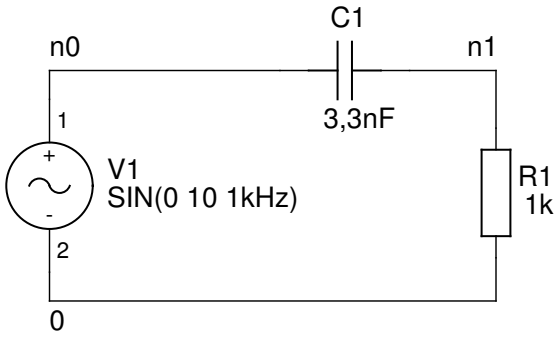

Figure 1: A simple RC filter.

The values which are shown in each corner are the *netnames*. They can be set by providing a value for the attribute `netname` for each net. A net can basically have any name, with one exception: There must be one net which is called "0". This is the reference (ground) net.

V1 is an *independant voltage source*. Its value defines the wave form (SIN), the voltage offset (0), the amplitude (10V) and the frequency (1kHz).

Save the schematic as `rc.sch`.

3.2 Creating the netlist

The input for the spice simulation is a *netlist*. The netlist can be created from the schematics file using the command **gnetlist**. Create the netlist from the schematic file with the following command:

```
$ gnetlist -g spice -o rc.net rc.sch
gEDA/gnetlist version 20040111
gEDA/gnetlist comes with ABSOLUTELY NO WARRANTY; see COPYING for more details.
This is free software, and you are welcome to redistribute it under certain
conditions; please see the COPYING file for more details.

Loading schematic [rc.sch]
```

The netlist contains the following lines:

```
* Spice netlister for gnetlist
* Spice backend written by Bas Gieltjes
V1 n0 0 SIN(0 10 1kHz)
C1 n1 n0 3,3nF
R1 0 n1 1k
.END
```

The format of a netlist is quite simple: each line of the netlist file contains one device of the circuit. The first column in each line contains the name of the device, and the subsequent columns contain the netnames each pin is connected to and the value of the device. In our example, the third line contains the voltage source, which is connected to net `n0` with pin 0 and to net `0` with pin 1. Its value is `SIN(0 10 1kHz)`. The same applies to the capacitor and to the resistor.

The first and the last line are special for NG-spice: when reading the netlist, NG-spice treats the first line as title or description of the circuit. The last line must contain the .END token.

As the format of the netlist files is quite simple, it would even be possible to create those files manually with a text editor. Some articles about spice simulation even start with this approach, but I dont think that this is a suitable approach in general. Using a schematics editor, it is much easier to modify the circuit, and the schematic can also be easily printed or embedded into some documentation. For larger circuits it is really cumbersome to create the netlist manually. Furthermore, I like integrating tools in a tool chain. This simplifies work in most cases.

3.3 Simulating the circuit

We are now ready to simulate the circuit. First, we need to decide which kind of analysis to run. SPICE can be used for different simulations, like *transient*, *frequency (AC)* and *parameterized*:

- A transient simulation is a time simulation. The result shows how the circuit behaves over time.
- An AC simulation is a frequency simulation. The behaviour of the circuit with varying frequency is described.
- A parameterized simulation can be both a transient or an AC simulation, but varies one or more values of the circuit. An example would be to simulate how a circuit behaves with a varying value for a specific capacitor.

For the very first step, we want to see how the input voltage of our circuit behaves over time. We want to perform a transient analysis of the circuit and show the voltage between the nets 0 and n0. To start the simulation, launch **ngspice**:

```
$ ngspice
Note: can't find init file.
******
** ngspice-15 : Circuit level simulation program
** The U. C. Berkeley CAD Group
** Copyright 1985-1994, Regents of the University of California.
** Please submit bug-reports to: ngspice-devel@lists.sourceforge.net
** Creation Date: Sun May  9 19:55:31 CEST 2004
******
ngspice 6 ->
```

We now need to load the netlist:

```
ngspice 6 -> source rc.net

Circuit: * Spice netlister for gnetlist

ngspice 7 ->
```

Since we have defined a frequency of 1 kHz for the input voltage, the time period is 1 ms. We want to see how the input voltage behaves during the first 5 ms. The simulation is started with the following command:

```
ngspice 9 -> tran 0.01ms 5ms
Doing analysis at TEMP = 300.150000 and TNOM = 300.150000
Warning: v1: no DC value, transient time 0 value used

Initial Transient Solution
--------------------------

Node                                   Voltage
----                                   -------
n0                                           0
n1                                           0
v1#branch                                    0


No. of Data Rows : 512
ngspice 10 ->
```

The first parameter to **tran** determines the step to use, the second parameter is the end value. If no third parameter is given, the start time is 0, otherwise the start time is given by this third parameter.

Well - thats all ;-) The simulation is now finished. What is still missing is to make the result of the simulation visible.

3.4 Viewing the results

Spice has now generated tables with all calculated values (512 values for each of the nodes as shown above). To make the result visible simply type the following command:

```
ngspice 10 -> plot n0
```

This plots the voltage transient of net n0. Note that plots always refer to two nets, that means the difference voltage between two nets is measured. If only one net name is given, the other net is automatically the reference net 0. This results in the following window opened:

You should see the diagram with inverted colors on your monitor, i.e. what is white here is black in your image and vice versa. To make the image more readable and to save ink, I switched the colors before I made the screenshot. Close the plot window and enter the following commands in NG-spice:

```
ngspice 11 -> set color0=rgb:f/f/f
ngspice 12 -> set color1=rgb:0/0/0
ngspice 13 -> plot n0
```

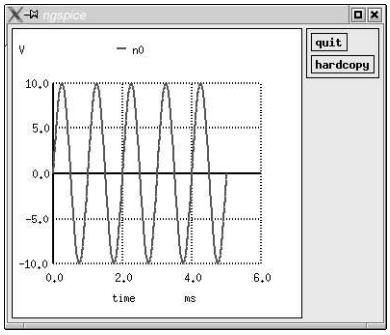

Figure 2: Screenshot of the plot window

The diagram shows the waveform of the input signal as we expected it, because we defined it so. But now, we are interested in the voltage across the resistor; additionally, we would like to compare the two signals! This can be simply achieved by adding the net name of another net to the plot command:

```
ngspice 13 -> plot n0 n1
```

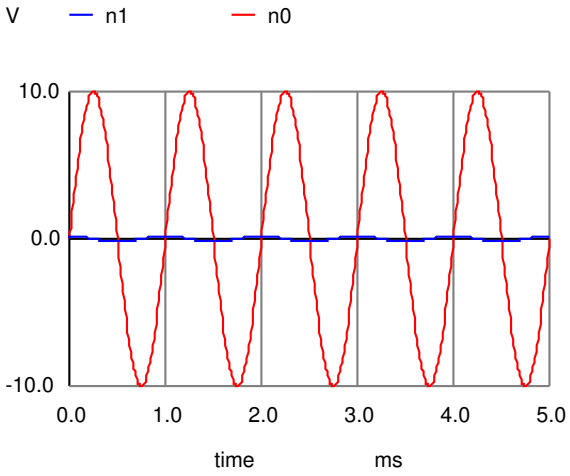

Figure 3: Input and output of the RC filter.

There is almost no signal remaining at the resistor. The question now is: Are there frequencies which can pass the filter better? To check this, we will now perform an AC simulation. The general command to perform an AC simulation is **ac (DEC | OCT | LIN) N FStart FEnd**. FStart and FEnd are the start and the end frequency. The optional parameter DEC, OCT or LIN describes whether to vary the frequency linear, per decade or per octave. In the case of linear variation, N is the number of frequencies to generate. If octave or decade variation is choosen, N is the number of frequencies per decade or per octave. To perform the AC analysis, the voltage source must be changed: currently it is defined as a sine wave with an amplitude of 10V and a frequency of 1kHz. For the AC analysis, it must be an AC voltage source. Go back to gschem, load the circuit and modify the value of the voltage source to AC:

Recreate the netlist, load the netlist into NG-spice and enter the following command to perform the AC analysis:

```
ngspice 65 -> ac lin 1000 0.1 250kHz
Doing analysis at TEMP = 300.150000 and TNOM = 300.150000
```

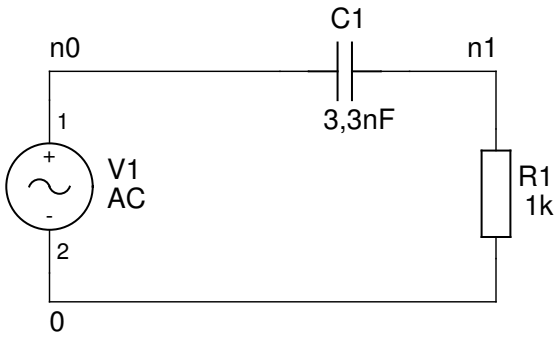

Figure 4: Filter for AC simulation.

```
Warning: v1: has no value, DC 0 assumed

No. of Data Rows : 1000
```

This command performs a linear frequency analysis from (almost) 0 Hz to 250kHz. The result can be viewed by plotting both the voltage source and the voltage on R1:

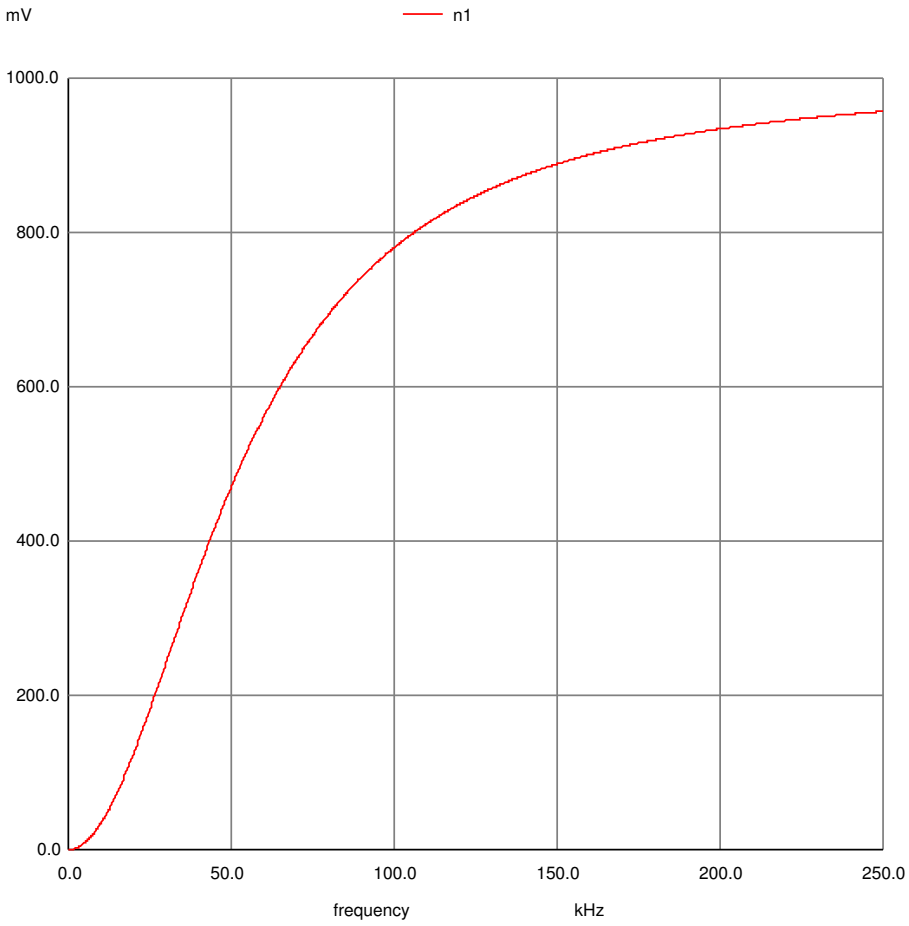

Figure 5: Frequency analysis of the RC filter.

In the following sections, we are going to simulate more circuits to get better involved with NG-spice.

4 Simulating a full wave rectifier

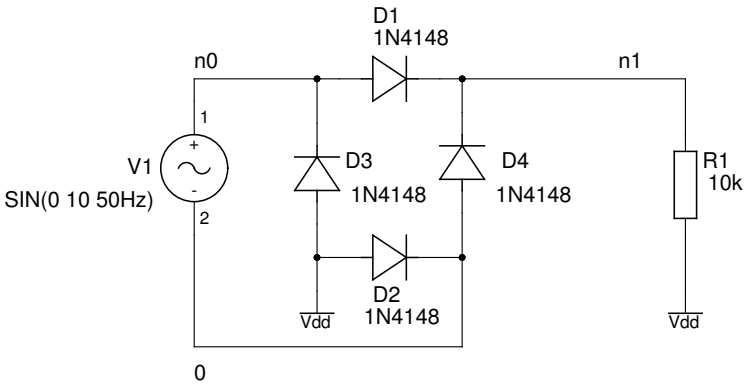

Figure 6: A full wave rectifier.

Figure 6 shows the typical full wave rectifier circuit. Both half waves of the input signal can be used, i.e. the negative half wave is mirrored upwards. This can be seen in Figure 7 which shows the spice simulation of the c

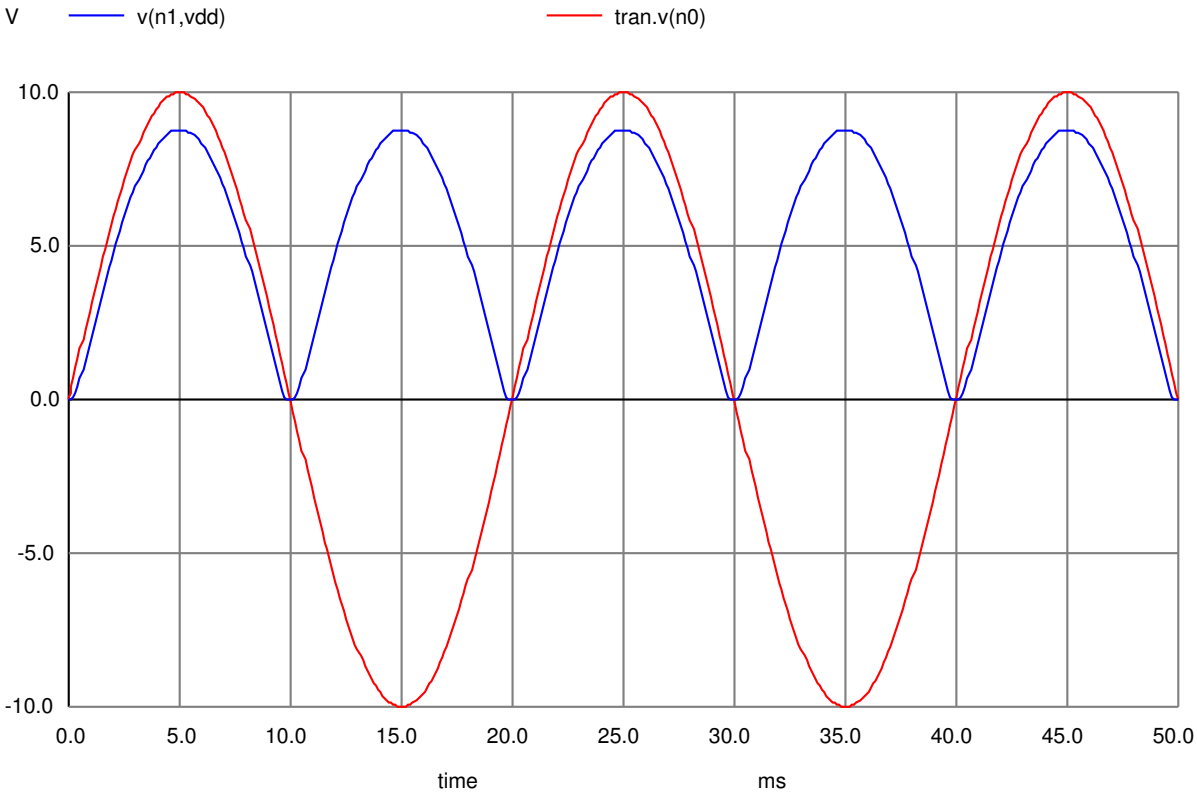

Figure 7: Source voltage transient and voltage across R1

Links to useful resources

URLs

[1] , *TBD* <http://www.ecircuitcenter.com/> .

[2] , *GPL Electronic Design Automation* <http://geda.seul.org> .

[3] , *TBD* <gnucap> .

[4] , *TBD* <gwave> .

[5] , *NG-SPICE: The free circuit simulator* <http://www.ngspice.org> .