Qucs

A Tutorial

Component, compact device and circuit modelling using symbolic equations

Mike Brinson

Introduction

Qucs releases 0.0.11 and 0.0.12 mark a turning point in the development of the Qucs component and circuit modelling facilities. Release 0.0.11 introduced component values defined by equations and for the first time allowed subcircuits with parameters. Release 0.0.12 extends these features to add model development using symbolic equations that are similar to compact device code written in the Verilog-A modelling language. In designing the latest Qucs modelling features the Qucs team has made a central focus of their work the need to provide the package with an interactive and easy to use modelling system which allows fast model prototype construction. Much of these new aspects have up to now been undocumented and are likely to be very new to most Qucs users. The aim of this tutorial note is to outline the background to these important package extensions and to provide real help to Qucs users who are interested in writing and experimenting with their own models. The text includes a number of illustrative examples for readers to try and experiment with.

Qucs electronic device and circuit modelling

Circuit simulation packages are complex software systems which often take years to mature to a stage where they are capable of analysing the current generation of integrated and discrete electronic circuits. Most circuit simulators have a number of common basic attributes; firstly circuits are represented by a textual netlist or a schematic diagram which contains all the information required by a simulator to analyse the performance of a circuit, and secondly a simulation engine which undertakes the calculation of circuit performance in one or more different circuit domains such as DC, AC or transient, and thirdly a post simulation processing system which structures and displays the simulation data in both tabular and graphical forms. All circuit simulators have one other important attribute, namely that they represent individual electronic components by a model, or abstraction, in a way that can be understood and analysed by the simulation engine when undertaking a simulation task. Without component models the science of circuit simulation would not have developed to the stage it has today. From a users point of view component models are the key to simulator productivity; the greater the number of different models the easier it becomes to analyse mixed analogue and digital electronic systems.

Shown in Fig. 1 is a block diagram of the analogue component modelling and simulation facilities currently provided by the Qucs package. The diagram is structured as a flow chart which emphasises the different device modelling routes. When Qucs

was first released only two of these were available for users to develop new device models. The first of these has been used extensively by the package developers to construct the built-in models that are distributed with each Qucs release. This fundamental route involves hand coding the C++ code for a new model[1], its compilation and linking with the core Qucs C++ code. Obviously, this does require a specialised knowledge of the Qucs model programming interface[2], the necessary C++ skills, including a good working knowledge of the Trolltech Qt toolkit[3]. At the time of writing these notes the latest device to be added to Qucs using this approach is the exponential pulse source[4]. Models based on hand written C++ code are normally restricted to basic devices that form the fundamental component core of a simulator - particularly where simulation computational efficiency is important. One disadvantage of this approach, is the obvious one, in that the time to implement a new model increases disproportionately with increasing model complexity. For most Qucs users this route would not be the most natural to use when developing new models. However, for the specialist who spends a significant amount of time researching new device models this has always in the past, been the route of choice. Unfortunately, modern semiconductor device models are becoming so complex that the model development time can stretch into months or even years and requires typically thousands of lines of C or C++ code to characterise a model[5]. With the more complex models the problem of finding bugs in the model code also acts as a limit to fast model development.

For the average Qucs user their first introduction to the software is probably through constructing circuit schematics made entirely from the standard component models built into the package and the testing of their performance by launching the simulator from one of the Qucs simulation icons.[6] The next natural stage in the Qucs modelling and simulation learning curve is the use of subcircuits where groups of built-in components are collected together to form a higher level circuit block. These blocks are often arranged with a common theme, forming

[1]The technical details of the built-in models are described in: Qucs Technical Papers, Stefan Jahn, Michael Margraf, Vincent Habchi and Raimund Jacob, `http://qucs.sourceforge.net/technical.html`.

[2]Writing the documentation for the Qucs model programming interface is on the to do list and will be completed, when time allows, sometime in the future.

[3]Qt is a registered trademark of Trolltech, Norway; `http://www.trolltech.com/copyright`.

[4]Added by Gunther Kraut on 15 April 2007. This device has been added for compatibility with SPICE.

[5]A good introduction to writing compact device models is given in "How to (and how not to) write a compact model in Verilog-A", Geoffrey J. Coram, 2004, Proc. 2004 IEEE International Behavioral Modeling and Simulation Conference (BMAS 2004), pp 97- 106.

[6]The "Getting Started with Qucs" tutorial by Stefan Jahn outlines a number of basic simulation techniques; `http://qucs.sourceforge.net/docs.html`.

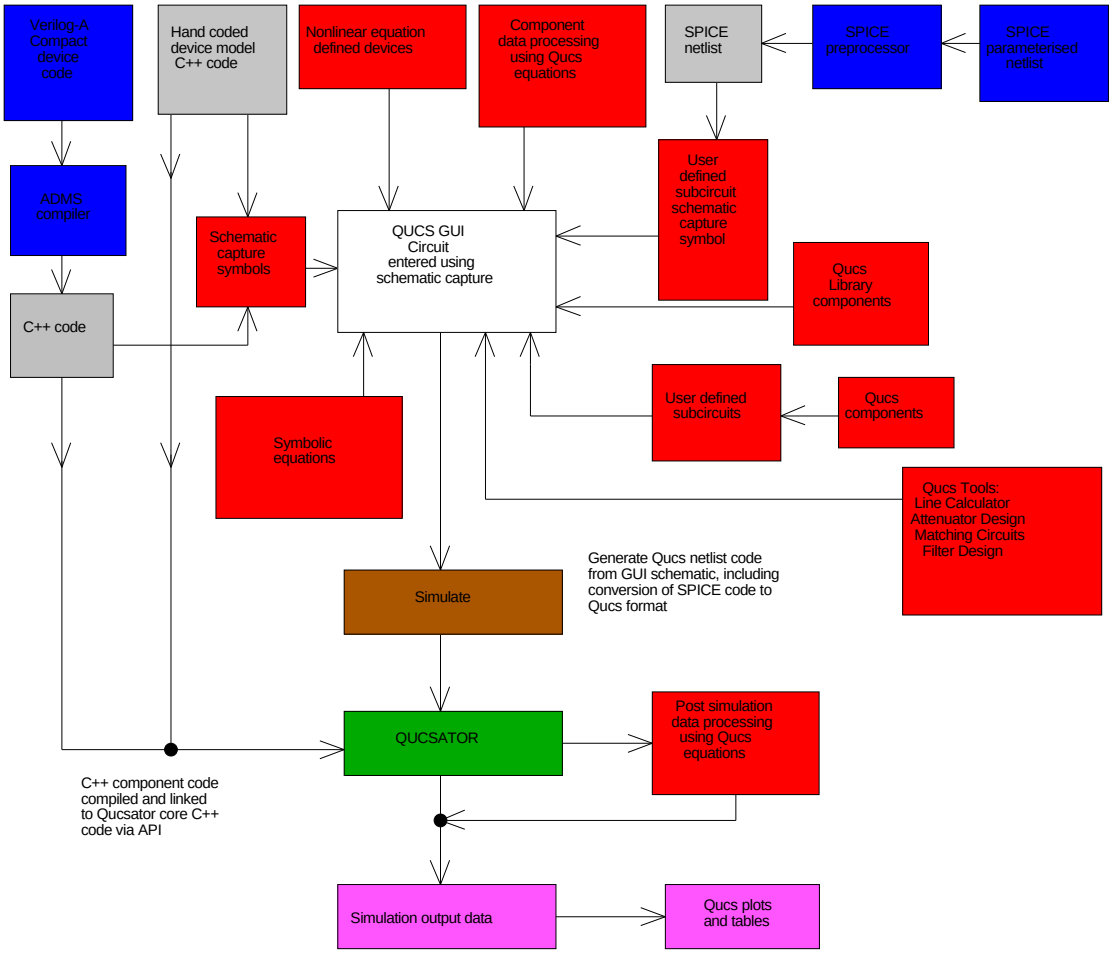

Figure 1: Qucs analogue component modelling and simulation block diagram (not including optimisation)

a Qucs library. The process of modelling new devices/circuits is normally done by connecting existing component models and user defined subcircuits. With this type of modelling higher level functional models can only be constructed from existing fundamental components or previously constructed subcircuits. Engineers often call this approach to modelling, macromodelling. Qucs releases up to 0.0.10 relied on macromodelling for functional model development via the Qucs schematic interface. This route remains popular amongst most Qucs users because it is easy to understand, is fully interactive and allows straight forward testing of new models. One feature that is common to all components included in Qucs releases up to 0.0.10 may not be immediately obvious to readers, namely that, with the exception of sweep variables, component values could only be numbers, for example R1 = 1k, and were not allowed to be represented by algebraic expressions like R1 = Value1, where Value1 = $100.0+50 \cdot X$. Its also worth pointing out at this point that during simulation, again performed by Qucs releases up to 0.0.10, component values were required to remain constant and could not be a function of the circuit variables such as voltage, current or charge.

One way to remove the component value restrictions imposed by early Qucs releases is to model devices and circuits using preprocessor extended forms of the SPICE netlist language. Circuit design equations can then be embedded in SPICE netlists and the calculation of component values completed by the SPICE preprocessor. Both the SPICE to Qucs and OP AMP tutorials[7] outline in detail the steps required to merge circuit design and simulation in this way. This modelling route is a very important and powerful model development tool. So much so that ongoing tests to identify how compatible Qucs is with the industrial standard SPICE 2g6 and 3f5 syntax are currently being undertaken as part of the Qucs development schedule[8]. Although perfectly viable as a model development tool the use of an extended SPICE netlist language has a number of serious disadvantages, namely that not all the Qucs built-in component models have equivalent SPICE models and secondly text netlists are the only entry medium for describing models.

The previous paragraphs give a brief statement of the different component modelling routes that were available up to release 0.0.11. Qucs 0.0.11 is very much a modelling water shed in that symbolic equations were introduced for the calculation of component values, previously equations were only allowed when structuring simulation output data for post simulation listing or plotting. Release 0.0.11 allows the following types of variable;

[7]Qucs simulation of SPICE netlists and Modelling Operational Amplifiers, Mike Brinson, `http://qucs.sourceforge.net/docs.html`.

[8]Qucs: Report Book; SPICE to Qucs test reports, Mike Brinson, `http://qucs.sourceforge.net/docs.html`.

1. sweep variables,
2. equations left hand side,
3. component parameter's left hand side (e.g. R1.R),
4. subcircuit parameters and
5. simulation output data.

With each Qucs release the number of analysis functions, and other data processing features, included in the Qucs equation set continues to expand[9]. From release 0.0.11 parameters are also allowed with subcircuits so that data can be passed to a model. This allows generalised subcircuit/macromodels to be developed for popular devices such as operational amplifiers. Through the use of embedded design equations within subcircuits and parameter passing it became possible to construct powerful models that mix both circuit design procedures and the calculation of individual component values. Qucs 0.0.11 still imposed the restriction that equations could not be functions of voltage, current or charge.

With the release of Qucs 0.0.12 the voltage, current and charge restrictions imposed on equations will finally be relaxed. The introduction of a new device modelling component called the equation defined device (EDD) allows firstly device current to be formulated as a function of voltage, and secondly device charge to be calculated as a function of voltage and current. The syntax adopted for the new model borrows heavily on the compact device modelling approach taken by the Verilog-A modelling language.

Some readers will probably have noted that so far these notes make no reference to the ADMS model development route illustrated in Fig. 1. ADMS stands for Automated device model synthesizer[10] and includes a Verilog-A to C/C++ compiler. It allows compact device models to be described in the Verilog-A language then compiled to C/C++ and the resulting code linked with the Qucs core simulation code[11]. Model development using ADMS is similar to the fundamental hand coded C++ model development route except that model development is greatly simplified by the power of the high level Verilog-A language. A strong relationship exists between the ADMS and EDD modelling procedures in that EDD can

[9]See Measurement Expressions Reference Manual, Gunther Kraut and Stefan Jahn, `http://qucs.sourceforge.net/docs.html`.

[10]L.Lemaitre, C.C. McAndrew, and S. Hamm, ADMS - Automated Device Model Synthesizer, Proc. IEEE CICC, 2002.

[11]For more details see, Qucs Description: Verilog-AMS interface, Stefan Jahn and Hélène Parruitte, `http://qucs.sourceforge.net/docs.html`.

be considered a fast interactive model prototyping method whose equations can easily be expressed in Verilog-A and compiled into C/C++ code for permanent inclusion in the Qucs simulator[12].

The opening paragraphs attempt to outline the available device modelling techniques that are central to the functioning of the Qucs package. The remaining sections of this tutorial note are devoted to illustrating the power of Qucs modelling through the introduction of a number of illustrative examples. Initially these start from a simple, and hopefully familiar, point and then proceed to more complex examples which present many of the concepts lightly touched upon in the opening text.

Extending circuit simulation capabilities with equations

Just adding component value calculations, via equations, to a circuit simulator immediately increases the underlying design and simulation capabilities way beyond that found in earlier generation simulators. Consider the simple RC circuit shown in Fig. 2. Capacitor *Cap* is stepped from 0.1µF to 1.1µF and the small signal AC response of the network calculated. In this example the values for both *R1* and *Cap* are given as numeric values. The simulation test shows the effect of stepping the value of one component through a series of values and recording the effect of component changes on circuit performance. In other words this is a classical circuit analysis use of a circuit simulator. In a real design situation different data is often required. Most designers would prefer to find the value of *Cap* that gives a specific RC cut-off frequency (f_c) for a specified value of *R1*. This is the type of investigative problem where adding equations into the simulation process generates more informative results. Shown in Fig. 3 is a similar RC network to that illustrated in Fig. 2.
Capacitor voltage $VCap$ is given by:

$$V_{Cap} = \frac{V_1}{\sqrt{1 + \omega^2 \cdot C_1^2 \cdot R_1^2}} \tag{1}$$

where the cut-off frequency in the voltage transfer function is

$$f_c = \frac{1}{2\pi \cdot R_1 \cdot C_1} \tag{2}$$

Hence, by expressing *Cap* as a function of f_c and stepping f_c through a range of frequencies, the effect of capacitance changes on the voltage transfer function can

[12]Appendix A gives an operator and function comparison table for Qucs and Verilog-A.

be found. More importantly a nomogram of *Cap* values against f_c can be plotted giving the circuit designer a visual aid for determing the value of *Cap* required for given values of *R1* and f_c. Although the circuits shown in Figs. 2 and 3 are very basic they do demonstrate how much more powerful a circuit simulator becomes when component values are calculated using equations.

Low pass active filter design with embedded design equations

In this section a more advanced circuit design example is introduced to illustrate the power of embedded design equations in a Qucs simulation schematic. A second order Sallen-Key low pass filter is employed for this task because it is so well known and most readers are likely to have met it's design in the past. A second order low pass filter is represented by the voltage transfer function:

$$A(S) = \frac{V_{out}}{V_{in}} = \frac{A0}{(1 + a_2 \cdot S + b_2 \cdot S^2)} \tag{3}$$

where $A0$ is the passband DC gain and coefficients a_2, b_2 are for Bessel, Butterworth, Tschebyscheff or similar polynomials.

The following list[13] gives the second order coefficients for the Bessel $\rightarrow$ 1.3617, 0.618; Butterworth $\rightarrow$ 1.4142, 1.000; and 3dB ripple Tschebyscheff $\rightarrow$ 1.065, 1.9305, polynomials. The second order Sallen-Key low pass filter circuit is shown in Fig. 4. This circuit has a voltage gain transfer function given by:

$$A(S) = \frac{A0}{1 + \omega_c \cdot \left[C_1 \cdot (R_1 + R_2) + (1 - A0) \cdot R_1 \cdot C_2\right] \cdot S + \omega_c^2 \cdot R_1 \cdot R_2 \cdot C_1 \cdot C_2 \cdot S^2} \tag{4}$$

where

$$A0 = 1 + \frac{R_3}{R_4} \tag{5}$$

This can be simplified by letting $R_1 = R_2 = R$ and $C_1 = C_2 = C$; the transfer function then becomes:

$$A(S) = \frac{A0}{1 + [\omega_c \cdot R \cdot C \cdot (3 - A0)] \cdot S + \left[(\omega_c \cdot R \cdot C)^2\right] \cdot S^2}. \tag{6}$$

[13] See OP Amps for everyone, Chapter 16: Active filter design technology, Texas Instruments, August 2002, SL0D006B, PP 16.1,16.63.

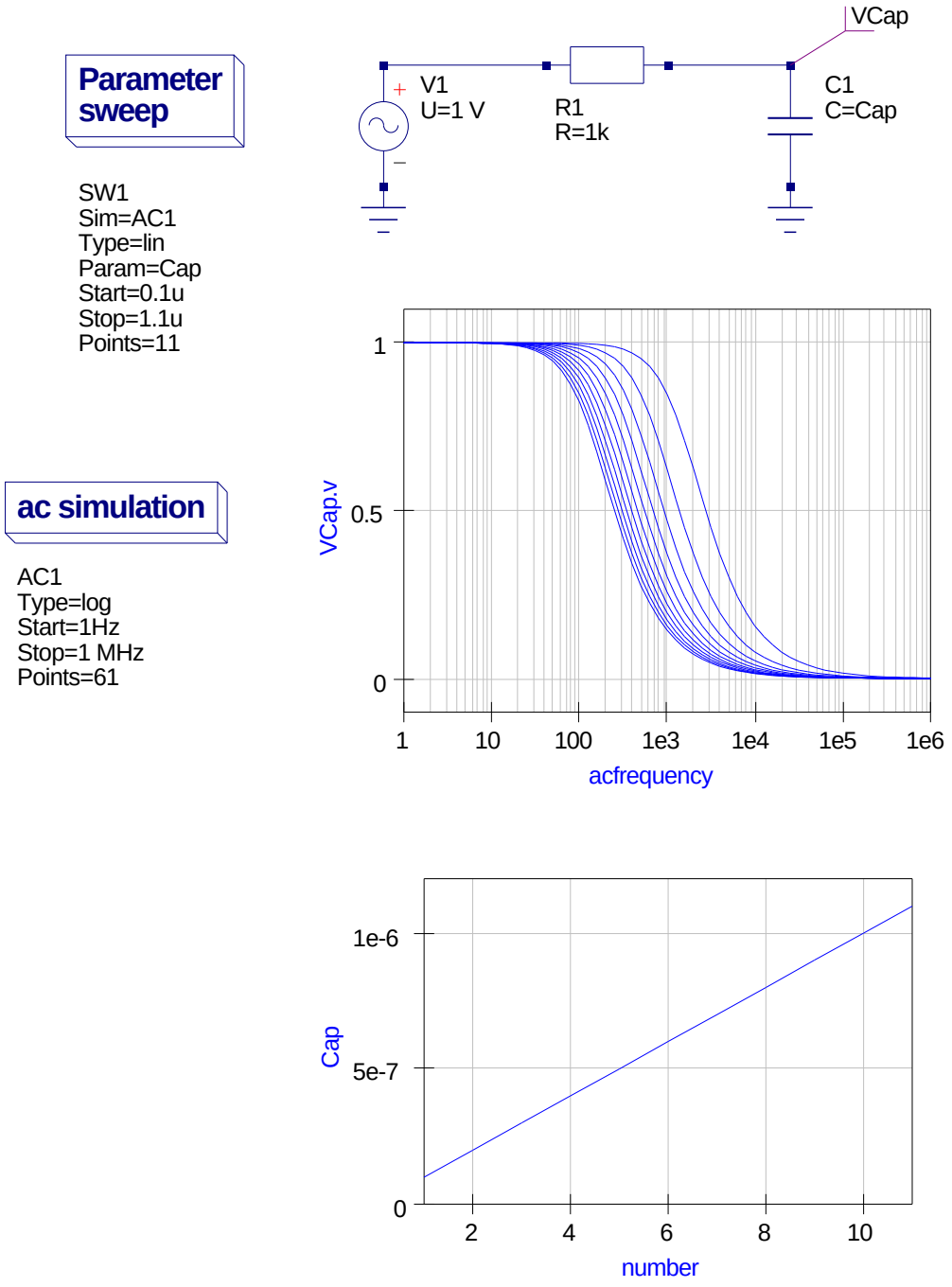

Figure 2: A simple RC circuit simulation using numerical component values

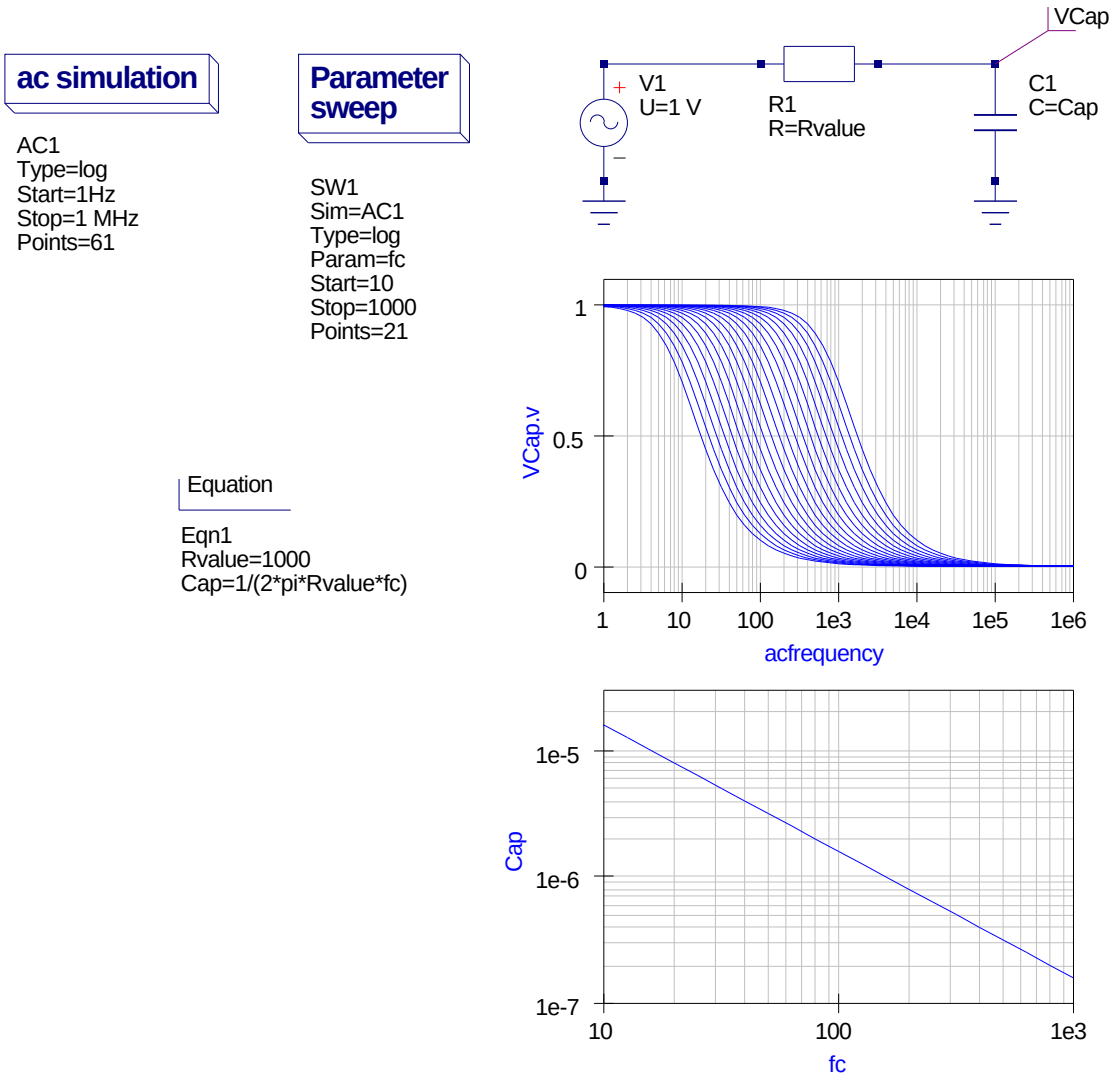

Figure 3: A simple RC circuit simulation employing equation determined component values

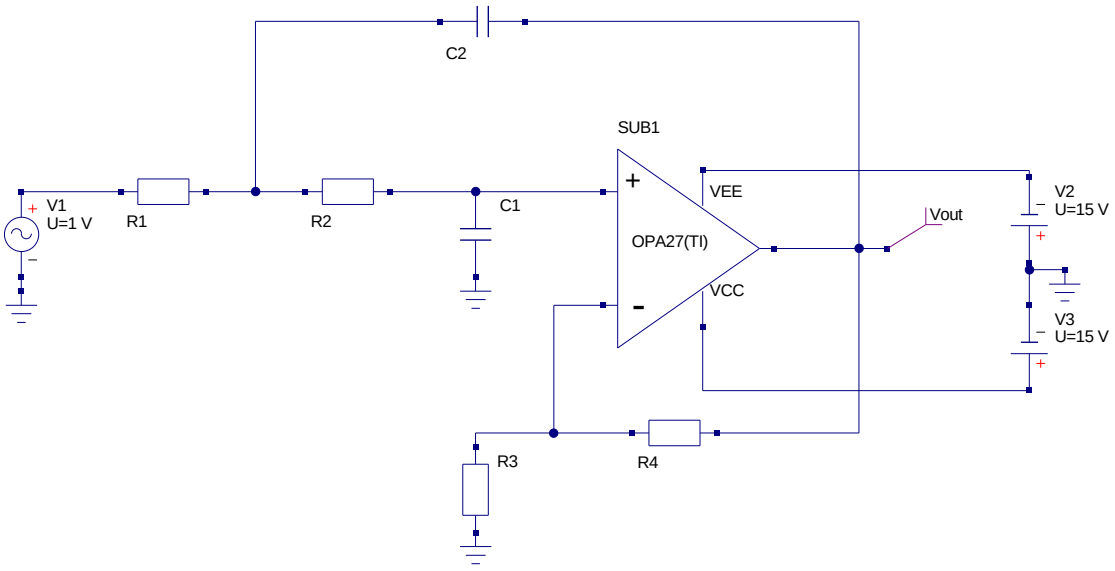

Figure 4: The Sallen-Key lowpass active filter circuit

By comparison

$$a_2 = \omega_c \cdot R \cdot C \cdot (3 - A0) \tag{7}$$

and

$$b_2 = (\omega_c \cdot R \cdot C)^2 \tag{8}$$

Fixing C and solving for R and $A0$, yields

$$R = \frac{\sqrt{b_2}}{\omega_c \cdot C} \text{ , and } A0 = 3 - \frac{a_2}{\sqrt{b_2}}. \tag{9}$$

Also once A0 is known the value for $R4$ can be calculated using equation

$$A0 = 1 + \frac{R3}{R4}. \tag{10}$$

Hence by providing values for C and $R3$ the values for R and $A0$, and of course $R4$, can be determined for a specified cut off frequency fc. Figure 5 shows the final design schematic and the simulation results for this example. A number of important observations can be made from Fig. 5:

1. One or more equation blocks hold both design and post simulation data processing equations plus assignments for named items: C, fc and $R3$ are given numerical values, the a and b polynomial coefficients are set to the values introduced in the text, and finally the design equations for R, $A0$ and $R4$ calculations are listed.

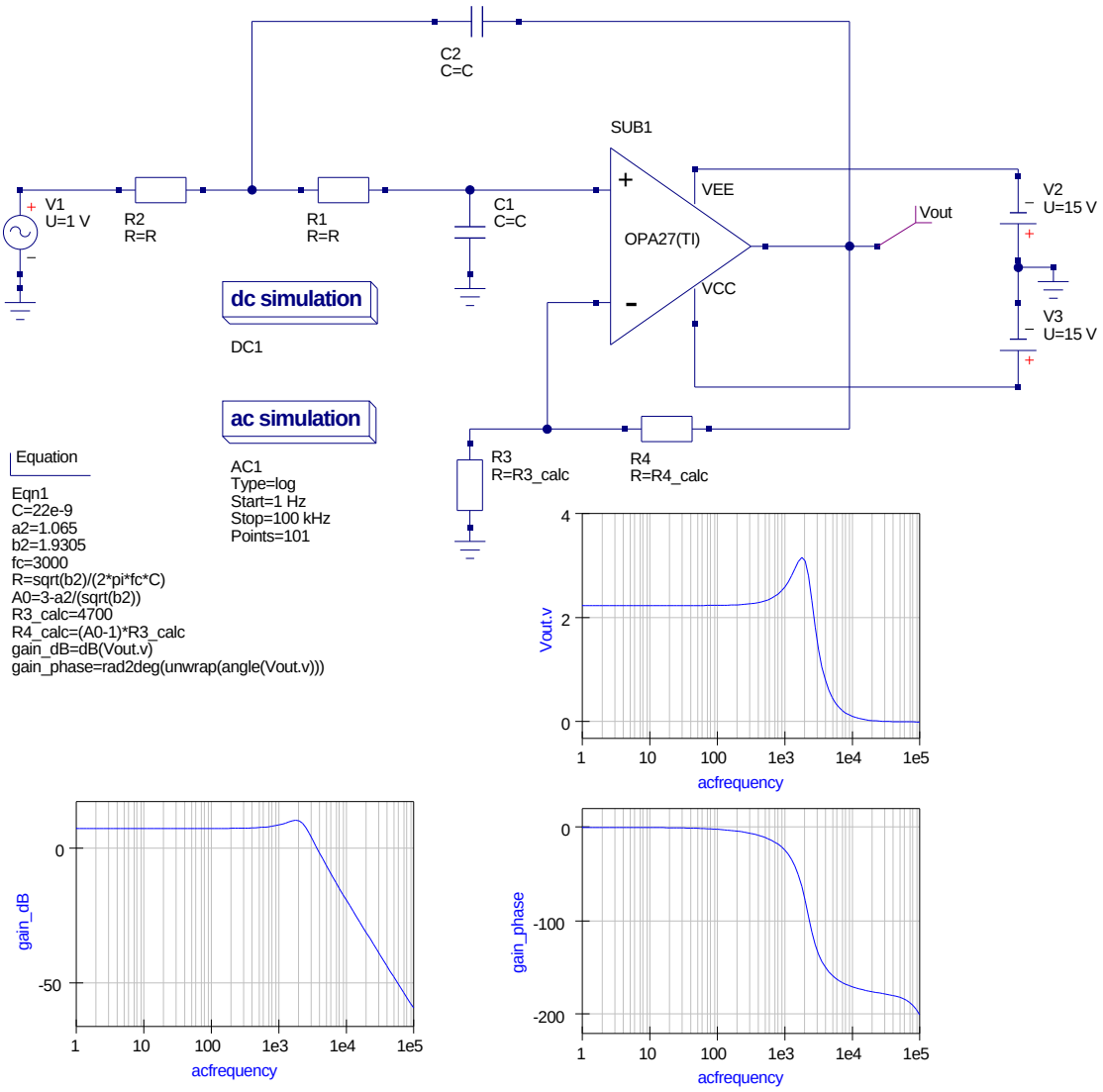

Figure 5: The Sallen-Key lowpass active filter schematic with embedded design equations

2. The order of entries in equation blocks is not important because Qucs automatically sorts out the data it requires when calculating equations.

3. The lefthand quantities in the assignment entries in the equation blocks are linked to the component values in the schematic, see for example C and R.

4. The OP27 operational amplifier model is from the modified Qucs 0.0.11 OPAMP library. This model was generated using the SPICE to Qucs modelling route.

5. To design and simulate a Sallen-Key low pass filter with a different cut off frequency[14] simply change the value of fc and rerun the Qucs simulator.

6. On completion of a simulation, pressing key F5 (Show last messages) causes the simulation log to be displayed. This includes the calculated values of the components and the netlist for the circuit, see Fig. 6.

7. One final point of significance that some readers may have noticed - all numerical values in equation blocks must be specified in scientific notation; electronic notation like 1k or 3nF is not allowed[15].

[14] If the design calculations result in impractical values for the filter components then the value of C should be changed and the simulation repeated.

[15] In long term it is expected that electronic notation will be allowed. The changes for this are on the to do list but at the moment the work has a low priority.

```
Output:
_______

netlist content
  13 R instances
   5 C instances
   2 VCCS instances
   5 CCCS instances
   2 VCVS instances
   1 CCVS instances
   8 Vdc instances
   1 Idc instances
   1 Vac instances
   4 Diode instances
   2 BJT instances
   1 DC instances
   1 AC instances
creating netlist...
checker notice, variable 'Vout.v' in equation 'gain_dB' not yet defined
checker notice, variable 'Vout.v' in equation 'gain_phase' not yet defined
kB = 1.38065e-23
e = 2.71828
pi = 3.14159
C = 2.2e-08
a2 = 1.065
b2 = 1.9305
fc = 3000
R = 3350.51
A0 = 2.2335
R3_calc = 4700
R4_calc = 5797.43
kB = 1.38065e-23
e = 2.71828
pi = 3.14159
kB = 1.38065e-23
e = 2.71828
pi = 3.14159
kB = 1.38065e-23
e = 2.71828
pi = 3.14159
```

Figure 6: Message output log for the simulation of the Sallen-key low pass circuit: for brevity only the component value section is given

Introduction to Qucs subcircuit parameters

Subcircuits are a concept that has been part of the simulation scene for a long time. All circuit simulators based on SPICE have subcircuits as part of their basic device compliment. This is not surprising because they form a natural way of breaking an electronic system down into a number of smaller self contained functional blocks. What is surprising however, is the fact that a significant number of simulators, including SPICE 2g6 and 3f5[16], do not allow parameters to be passed to a subcircuit. Parameter passing appears to have been first introduced when a number of the popular commercial circuit simulators were being developed[17]. Qucs releases up to version 0.0.10 are similar to SPICE in that they also did not allow parameters with subcircuits.

This very important limitation has been removed with release 0.0.11, which allows parameters to be attached to component symbols and used in subcircuit equation calculations. Shown in Fig. 7 are the circuit schematic and user generated symbol for a simple harmonic generator with a fundamental and three harmonic sinusoidal components. Parameters *f1* to *f4* determine these frequency components. Notice that an equation block, at the circuit schematic level, is used to calculate the harmonic frequencies. Parameters *ph1* to *ph4* set the phase of the individual sinusoidal oscillators. The process of attaching parameters, and their default values, to a subcircuit symbol is straightforward; simply right click on the symbol subcircuit name, *SUB1* in Fig. 7, and an Edit Subcircuit Properties dialog box appears allowing parameter names and their default values to be entered[18]. Subcircuit parameters and their values are normally displayed as a list underneath the subcircuit name. Changing parameter values is done in a similar fashion to changing the values of the standard built-in components. The diagram and simulation results illustrated in Fig. 8 show a waveform formed from a fundamental and two harmonics.

An equation block is employed to calculate and plot the amplitude and power spectral densities of the harmonic waveform. By changing the fundamental frequency, signal amplitudes and phases different wave shapes can be generated by resimulating the circuit. In this example transient analysis is used to generate the harmonic waveform with the run time set to 10ms and the number of points equal to 500[19].

[16] One of the reasons SPICE preprocessors were developed was to allow parameter passing to subroutines, for more details see Qucs Tutorial: Qucs simulation of SPICE netlists, Mike Brinson, `http://qucs.sourceforge.net/`.

[17] See, for example, the extended netlist format originally designed by the MicroSim Corporation for the PSpice circuit simulator.

[18] See Appendix B for a more detailed description of the procedure.

[19] Qucs function length() determines the correct data length in equation block Eqn1 calcula-

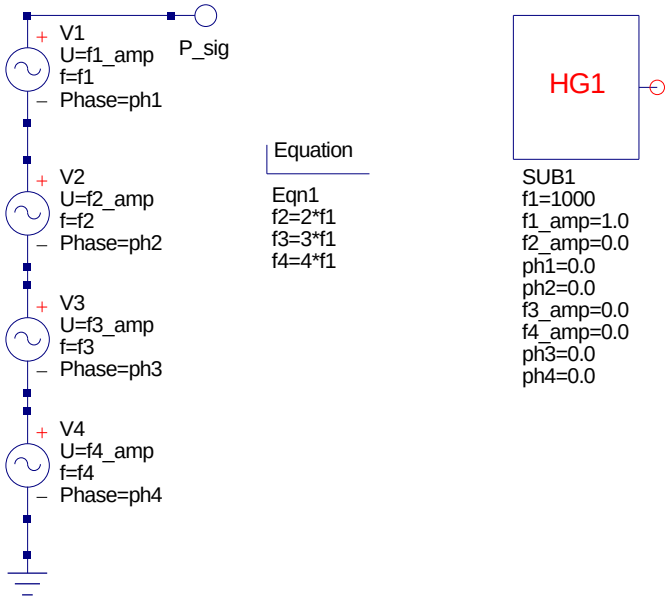

Figure 7: Harmonic generator subcircuit schematic and symbol

This gives a sampling time of 20µs and a sampling frequency of 50kHz. Equation block *Eqn1* demonstrates how the Qucs functions[20] can be used to postprocess simulation generated data - in this example they are used to compute the DFT of the harmonic generator waveform, convert the resulting spectra from double sided to single sided form, compute and plot the amplitude and power spectral densities.

tions.

[20]If you have used a program like Octave, or indeed Matlab, many of these functions should be familiar to you. These functions provide Qucs with powerful numerical resource which significantly extends the range of problems that Qucs can analyse.

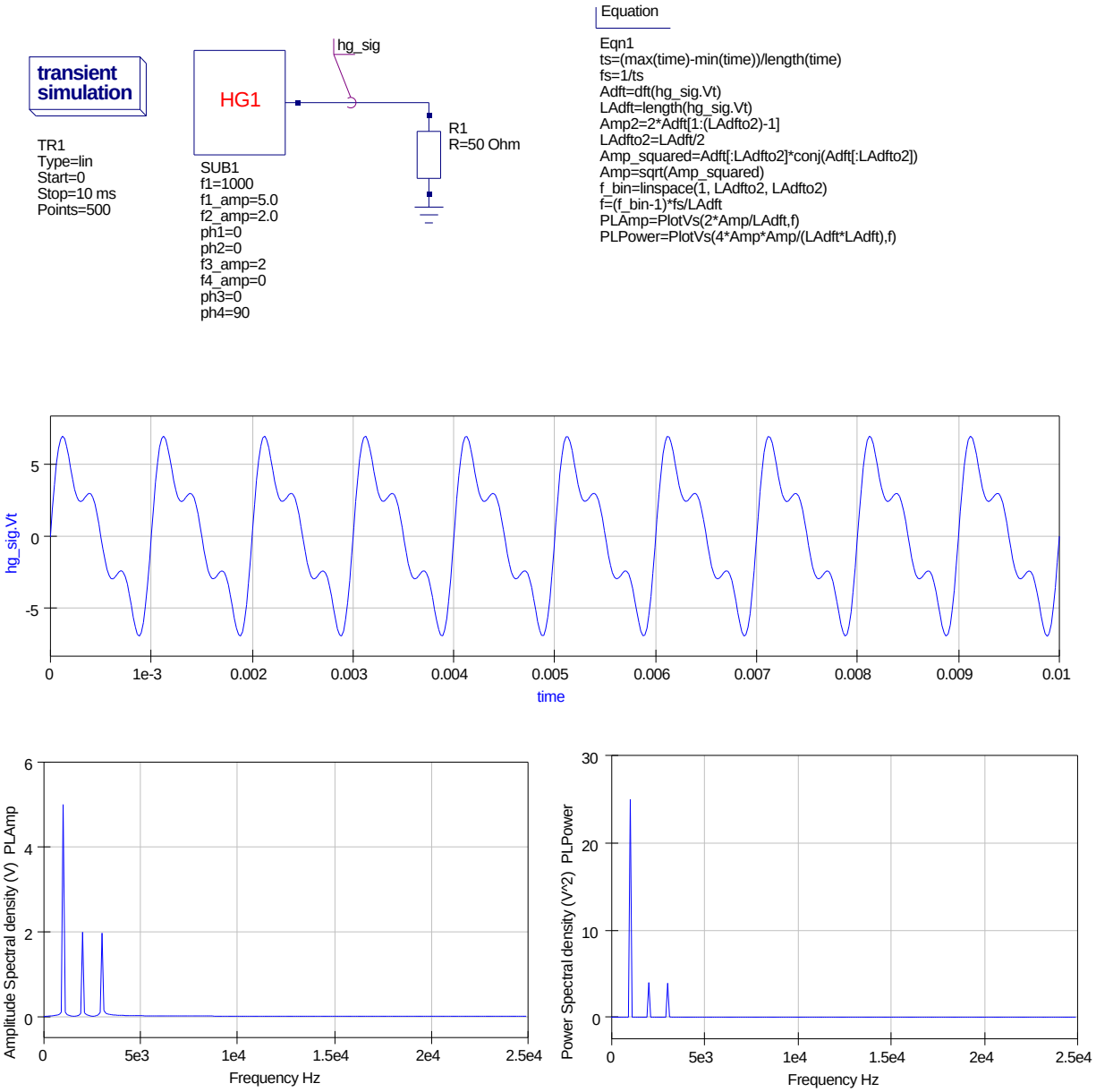

Figure 8: Harmonic generator subcircuit test circuit and simulation waveforms

Building universal macromodels using subcircuits and parameters

Passing parameters to subcircuits allows universal macromodels to be built. One obvious application of this technique is the modelling of operational amplifiers (OP AMP) and other integrated circuits. The approach adopted is similar to that outlined in the last section. However, because of the complexity of the models it is advisable to break a model into a series of smaller blocks. These are then combined to form a complete subcircuit macromodel. Two techniques are possible when partitioning models, these are demonstrated next. Shown in Fig. 9 is a simple AC OP AMP model[21] consisting of an input stage, an intermediate gain stage and an output stage[22]. An equation block, if needed, is associated with each stage. These blocks contain the equations for calculating the component values in a given stage. A single schematic symbol represents the model. This has a list of parameters attached. The flow of information into a macromodel starts with parameters passed into a subcircuit, via a schematic symbol, then onto the equation blocks, where it is finally used to calculate the component values. Hence, by simply changing the subcircuit parameters different OP AMPs can be simulated using a single generalised macromodel. However, please note that different OP AMP circuit structures, or indeed technologies, naturally result in a series of generalised subcircuit macromodels to cover all possible types in a given device family. The second technique involves breaking a model down into smaller blocks and associating subcircuit symbols with each block. This approach is illustrated in Fig. 10. Again parameters are passed from the top level symbol (called AC in the schematic) to the inner subcircuits. These pass their own parameters down a subcircuit level where the component calculations are completed. The second technique results in two levels of subcircuit, accounting for the change in parameter name when passing a parameter from top to lower hierarchy. A second more detailed example showing how to construct nested subcircuits is presented later in these notes.

In reality the macromodel for a typical OP AMP that models DC, AC and transient domains is much more complex than the model given in Fig. 9. The schematic for a typical multi-domain OP AMP modular macromodel is shown in Fig. 11, where

[21]The term AC here refers to the fact that the OP AMP model chosen for demonstration purposes is a simplified version of a multi-domain OP AMP model. It only models small signal AC parameters and device input stage bias and offset properties.

[22]The schematic shown in Fig. 9 forms part of a modular OP AMP macromodel. A detailed description of the function of individual networks and the derivation of the component equations is given in Qucs tutorial Modelling Operational Amplifiers, Mike Brinson, `http://qucs.sourceforge.net/docs.html`.

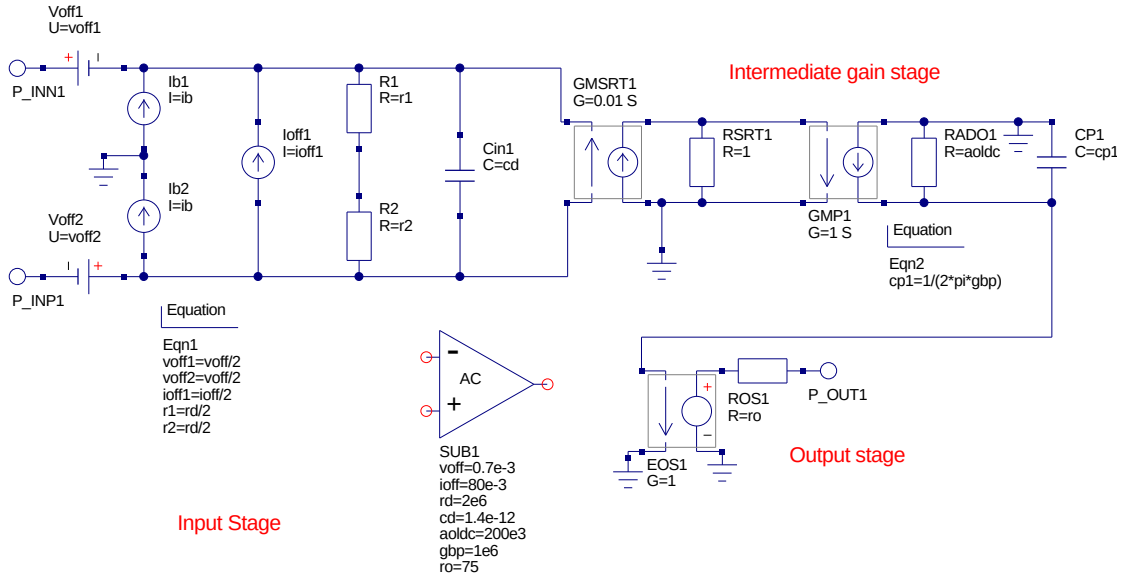

Figure 9: Expanded AC OP AMP model showing circuitry and equation blocks

each section of the macromodel is represented, if needed, by it's own equation block.

The test schematics shown in Figures. 12 and 13 show two OP AMPs with different subcircuit parameters. In Fig. 12 the small signal characteristics of unity gain closed loop amplifiers clearly show the difference in performance of the OP AMPs. Figure 13 is particularly interesting in that it illustrates how Qucs can be used to determine the effect of amplifier offset voltage on integrator DC saturation by stepping resister *rp* through a series of values. The low offset voltage of the OP27 makes this device much more suitable for integrator circuits when compared to the popular UA741. These results can be confirmed by a simple calculation: the offset voltage for the UA741 is set at 0.7 mV and the amplifier open loop DC gain at roughly $200,000$. The UA741 goes into saturation when *rp* is approximately 20 MΩ. In saturation the OP AMP gain becomes open loop giving a DC output voltage of roughly $0.7\text{e-}3 \cdot 2\text{e}5$ or 14 V, which agrees with the Qucs simulation results.

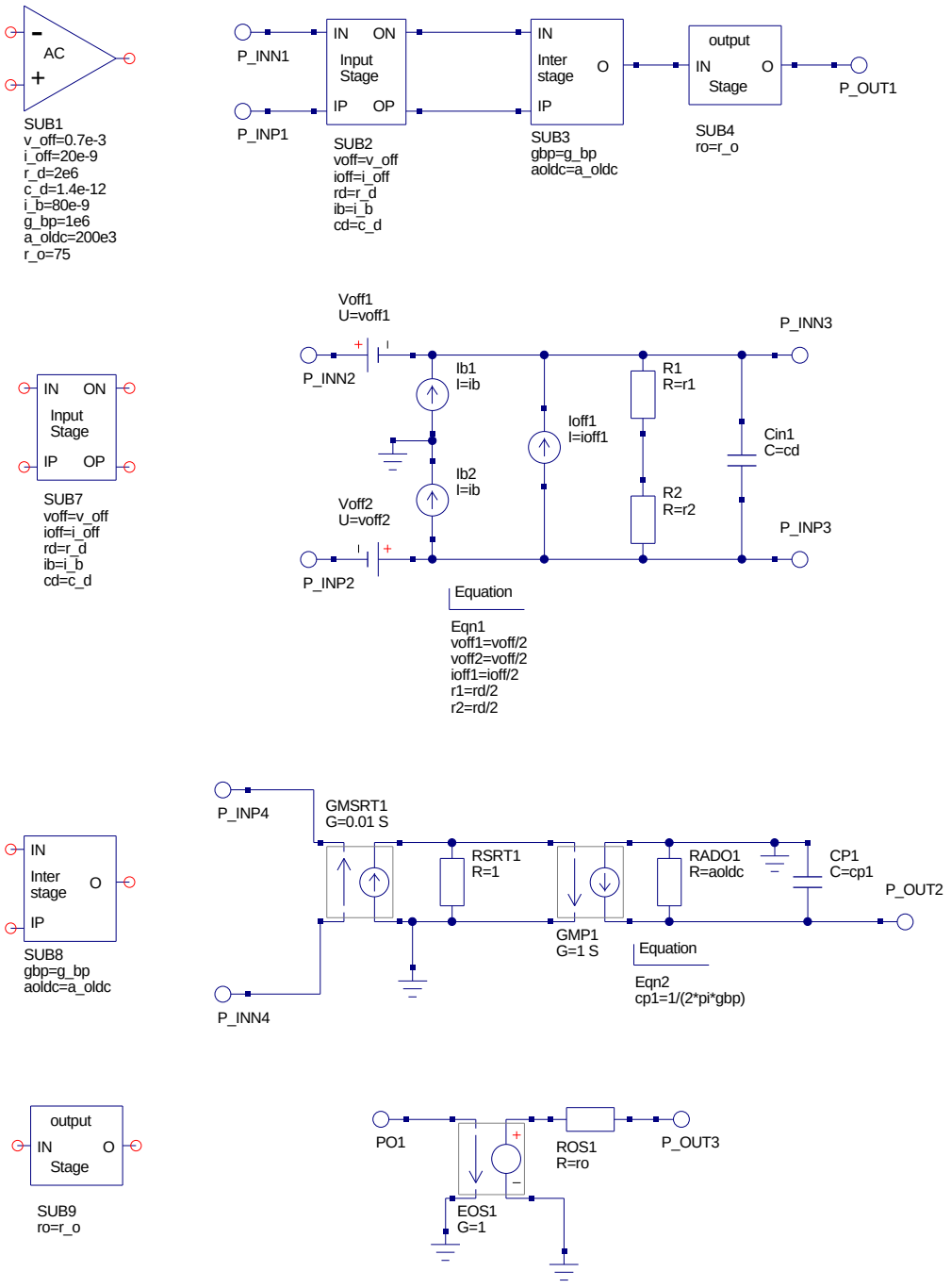

Figure 10: Modular AC OP AMP model showing subcircuits

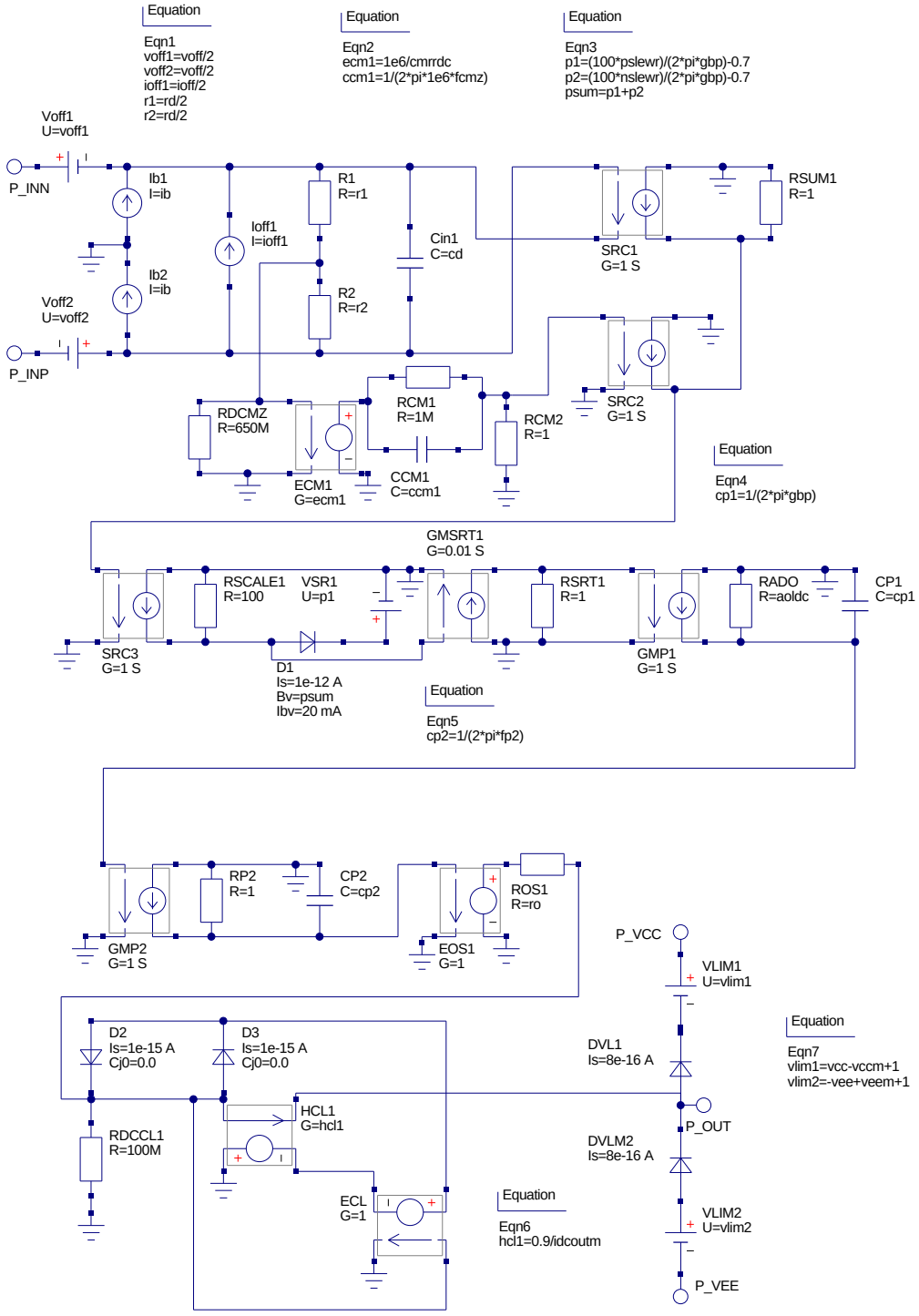

Figure 11: Modular OP AMP subcircuit schematic with embedded component calculation equations

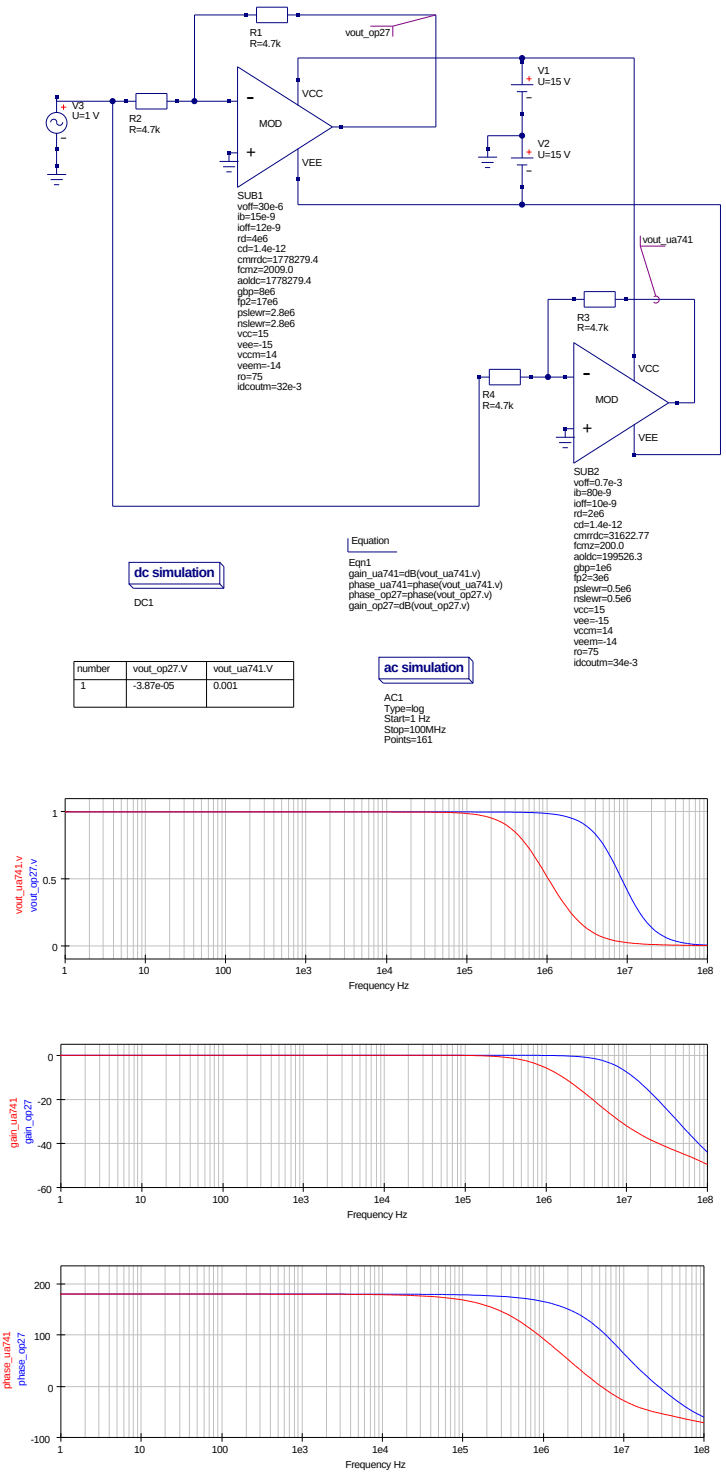

Figure 12: Unity gain OP AMP test circuit and waveforms

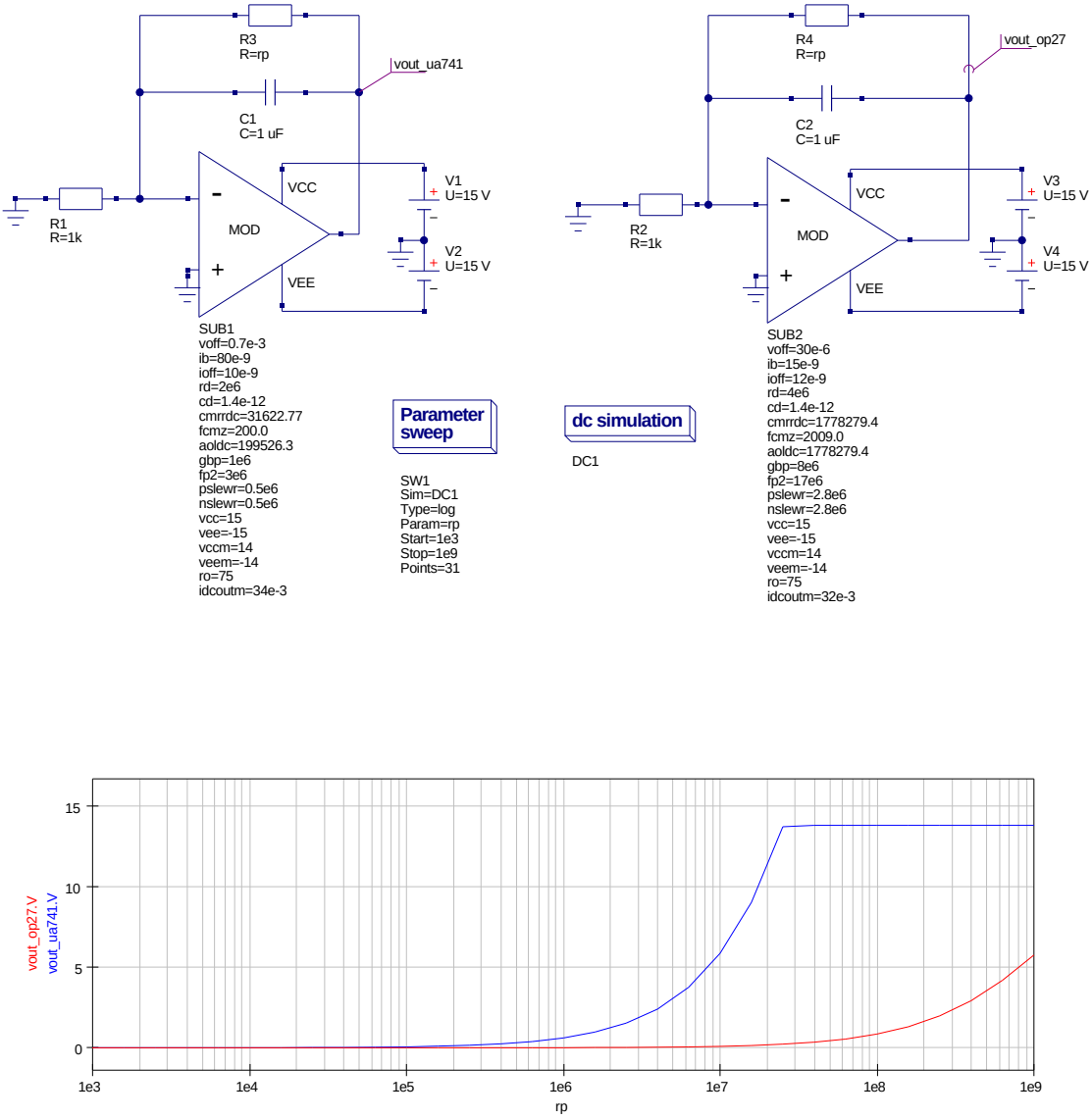

Figure 13: Integrator test circuits for determining DC saturation

More complex nested subcircuit models

In the previous two sections the example circuits only included subcircuits nested to one or two levels. Qucs does however, allow subcircuits to be nested to an arbitrary level and parameters can be passed down the nested chain to any depth required. Some care is needed when setting up the parameter passing sequence. Shown in Fig. 14 is a top level subcircuit with temperature swept between 10 and 110 centigrade. A simple resistor voltage divider network is at the bottom of a series of linked subcircuits, three levels down. R2 in the divider is a function of temperature. A schematic representation of the coupled subcircuits parameter passing sequence is also given in the right hand side of Fig. 14. Each level passes the value of temperature to it's next lower member in the hierarchy. The Qucs generated netlist given in Fig. 15 clearly shows the parameter passing mechanism employed by Qucs. The ability to nest subcircuits and pass parameters down a hierarchy is an important feature in Qucs because it allows both circuit design and device data to be passed to different sections of the circuit/system being simulated. These parameters can, of course, be at different levels in a problem hierarchy providing a very flexible and powerful design/analysis tool.

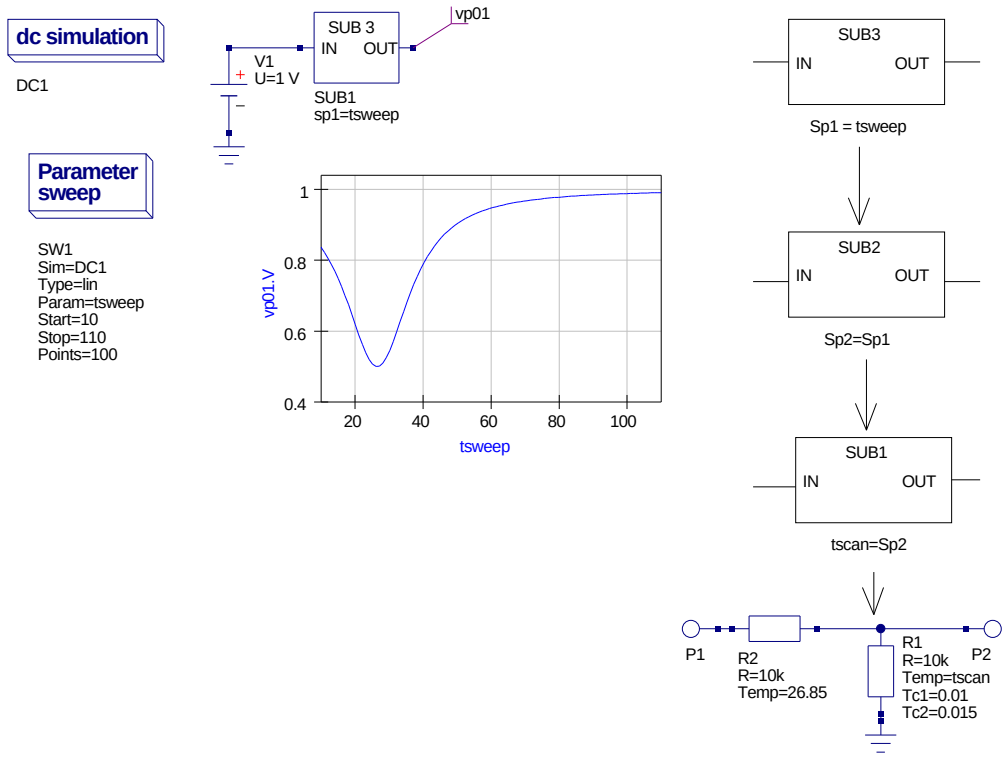

Figure 14: A nested subcircuit showing parameter passing sequence

```
# Qucs 0.0.12  /media/hda2/Qucs_equation_modelling_prj/rdiv_test_tsweep_3l.sch
.Def:rdiv_sub1_temp _net1 _net0 tscan="27"
R:R2 gnd _net0 R="10k" Temp="tscan" Tc1="0.01" Tc2="0.015" Tnom="26.85"
R:R1 _net1 _net0 R="10k" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
.Def:End

.Def:rdiv_test6_temp _net1 _net0 sp2="27"
Sub:SUB1 _net1 _net0 Type="rdiv_sub1_temp" tscan="sp2"
.Def:End

.Def:rdiv_sub3_temp _net0 _net1 sp1="27"
Sub:SUB1 _net0 _net1 Type="rdiv_test6_temp" sp2="sp1"
.Def:End

Vdc:V1 _net0 gnd U="1 V"
.DC:DC1 Temp="26.85" reltol="0.001" abstol="1 pA" vntol="1 uV"
saveOPs="no" MaxIter="150" saveAll="no" convHelper="none" Solver="CroutLU"
.SW:SW1 Sim="DC1" Type="lin" Param="tsweep" Start="10" Stop="110" Points="100"
Sub:SUB1 _net0 vp01 Type="rdiv_sub3_temp" sp1="tsweep"
```

Figure 15: Qucs netlist for nested subcircuit showing parameter passing sequence

Introduction to equation defined devices (EDD)

Although adding symbolic equations to a simulator merges circuit design and analysis, it is by making these equations functions of circuit variables that the real power of modern circuit simulator is fully exploited. Equations that are functions of voltage, current and charge have to be continuously evaluated as a simulation progresses. This is in contrast to the type of equations previously introduced, which are only evaluated at the start of a simulation sequence. When component properties are functions of circuit variables considerable complexity is added to a simulation engine and as a result most simulators restrict such properties to a small number of component types, the most common being controlled current and voltage generators[23]. Qucs version 0.0.12 introduces an equation defined device (EDD) which allows it's terminal currents to be functions of voltage, and it's stored charge to be functions of voltage and current. The EDD is similar, but more

[23]Probably the most well known non-linear controlled generators are the SPICE 2g6 and 3f5 forms, see A. Vladimirescu, Kaihe Zhang, A.R. Newton, D.O. Pederson and A. Sangiovanni-Vincentelli, SPICE Version 2G User's Guide, 1981, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Ca. 94720, section 11, Appendix B: Non-linear dependent sources., and B. Johnson, T. Quarles, A.R. Newton, D.O. Pederson and A. Sangiovanni-Vincentelli, SPICE3 Version f User's Manual, 1992, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Ca. 94720, section 3.2.2.4, Non-linear dependent sources.

advanced, to the B type controlled source implemented in SPICE 3f5. It is capable of realising the same models as the SPICE B type device plus an extensive range of more complex compact device models. At this stage in Qucs development only the explicit form of EDD is implemented[24]. EDD is an advanced component that allows Qucs users to construct their own device models from a set of equations derived from the physical properties that characterise a device. The explicit form of EDD can only be used to develop models for devices where their defining equations can be transformed into the explicit analysis form required by Qucs[25]. A range of functions similar to those defined in the Verilog-A compact device modelling language are provided by Qucs, making the equation modelling language easy to use and powerful. The ternary ? : form of the C language if statement has also been implemented to allow selection of model equations that change with differing device voltage, current and charge conditions. Before introducing the EDD symbol and it's properties consider the following circuit simulation modelling problem: a model for a device is required where the output voltage is a function of two input voltages VIN_1 and VIN_2, such that

$$V_{out}\left(VIN_1, VIN_2\right) = VIN_1 \cdot VIN_2, \tag{11}$$

where VIN_1 and VIN_2 can be arbitrary varying voltages.

This type of model is difficult to simulate at functional level[26] using the pre-version 0.0.12 built-in devices. A linear voltage controlled voltage source can be used to multiply a voltage by a constant. Multiplying by a second voltage is not possible with the linear controlled sources. Qucs AM modulated and PM modulated sources are the nearest that Qucs has to the source defined above. These sources however, only allow sinusoidal carrier signals. Illustrated in Fig. 16 is a four quadrant multiplier EDD which allows multiplication of two varying signals[27]. The EDD device generates current $I1 = V2 \cdot V3$. This in turn is transformed to the output voltage by a unity gain current controlled voltage source SRC1. An EDD device can consist of up to 8 branches. The branches have currents, I1 to I8, voltages V1 to V8 and internal charges Q1 to Q8 respectively. Overall the total device current depends how these branches are connected. A similar comment applies to the total device charge. In Fig. 16 currents I2 and I3 are set to zero, charges Q2 and Q3 are

[24]See Qucs Technical Papers, Section 10.7: Equation defined models, Stefan Jahn, Michael Margraf, Vincent Habchi and Raimund Jacob, `http://qucs.sourceforge.net/technical.html`.

[25]The Y parameters of the device being modelled must also exist for the explicit form of the EDD to be valid.

[26]It is, of course, possible to model the multiplier operation at discrete component level e.g. using a Gilbert cell mixer circuit.

[27]This model is based on an idea suggested by Stefan Jahn, during the EDD development phase.

also zero, and voltages $V2 = VIN_1$ and $V3 = VIN_2$. Hence current I1 becomes the multiplication of VIN_1 and VIN_2. The fact that currents I2 and I3 are set to zero implies that the terminals connected to the external input voltages have high impedance and act as voltage probes. The test circuit in Fig. 16 is shown with signal inputs generated by sinusoidal oscillators; V1 acts as a modulating signal and V2 as a carrier signal. The bottom right hand corner of Fig. 16 includes a second graph which illustrates the effect of changing signal V2 to a square wave source with 0.05ms period.

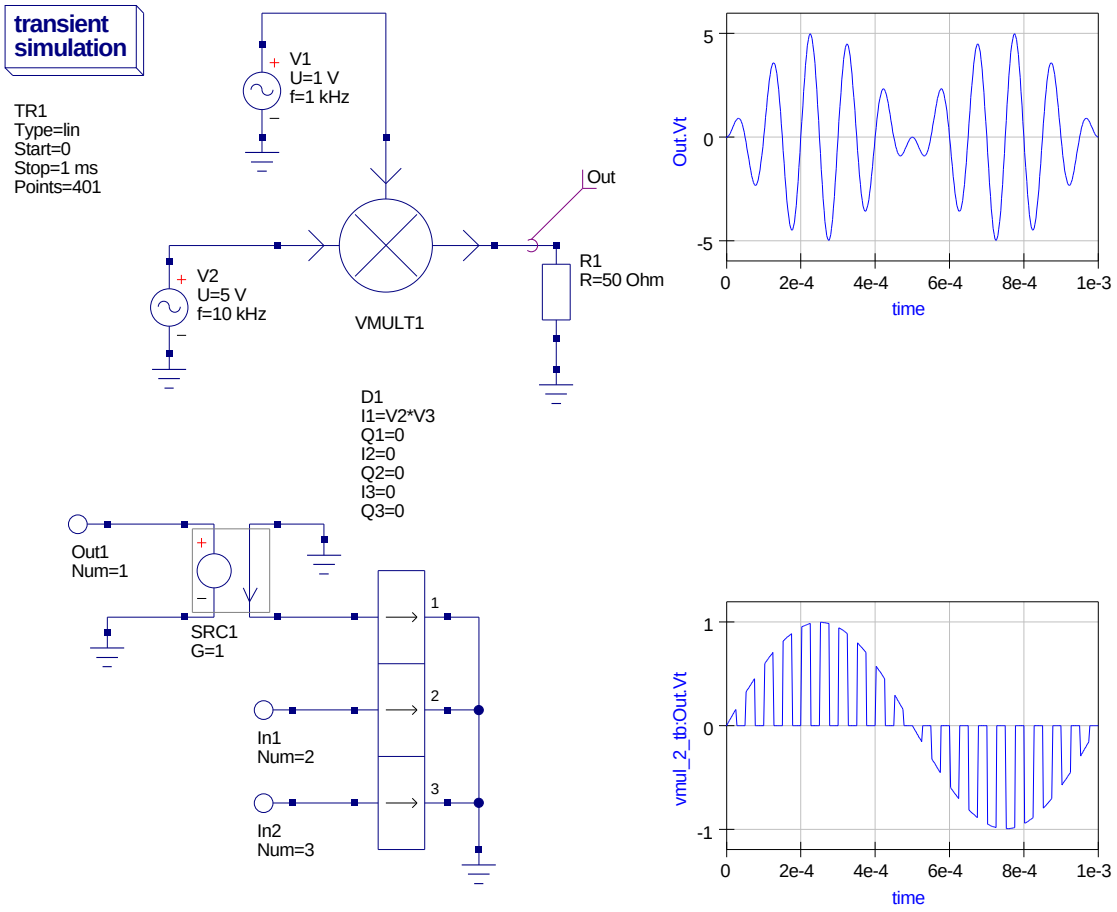

Figure 16: Qucs EDD four quadrent multiplier model and test circuit

The Qucs EDD component

A two terminal model for a universal non-linear component with resistive, capacitive and inductive parallel branches is shown in Fig. 17. All three branches have

elements that can be functions of either voltage or current or charge[28]. The Qucs EDD component can be used to model this nonlinear device. One EDD element is needed to model the resistive and capacitive branches. A second EDD device, plus a gyrator, models the inductive branch. The total terminal current is the sum of the individual branch currents. Equations for the three branch currents are given by the following equations:

$$I = I1 + IC + IL, \tag{12}$$

where

$$I1 = f(V), \quad IC = C(V, I) \cdot \frac{dV1}{dt} = \frac{dQ1}{dt} \tag{13}$$

Also

$$V1 = i2, \quad V2 = -IL, \quad i2 = -L(I) \cdot \frac{dV2}{dt}, \quad V1 = L(I) \cdot \frac{dIL}{dt} \tag{14}$$

Giving

$$IL = \frac{1}{L(I)} \cdot \int V2 \cdot dt \tag{15}$$

and

$$VL = V2 = V1 = \frac{d\Phi}{dt} \tag{16}$$

Hence

$$I = f(V) + C(V, I) \cdot \frac{dV1}{dt} + \frac{1}{L(I)} \cdot \int V1 \cdot dt \tag{17}$$

The EDD is characterised by eight parallel branches each comprising a current component In and a charge component Qn, where n ranges from 1 to 8. The currents may be constants or defined by equations that are functions of the EDD branch voltages (these are designated $V1$ to $V8$). This form of the EDD component is known as the explicit EDD model. Please note, EDD currents cannot be functions of current. However, with release 0.0.12 implementation of the explicit EDD the device charge can be a function of either voltage or current[29]. The current in the resistive branch being a function of EDD voltage allows a range of two terminal[30] devices to be modelled, allowing, for example, nonlinear resistors and diode models to be easily developed. Similarly, the fact that the EDD charge can be a function of voltage or current extends the range of allowed Qucs capacitor

[28]Each branch can be a function of one or more of these circuit variables but not necessarily all three at the same time.

[29]This allows modelling of semiconductor capacitive effects where the amount of stored charge is either a function of voltage (depletion layer capacitance), or a function of current (diffusion capacitance).

[30]The number of device terminals can be increased to model transistors and other devices.

types opening new areas of application. The same comments apply to the nonlinear inductors where components that have inductance values which are functions of current allow modelling of nonlinear transformer and coupled inductor effects. This was not possible with earlier Qucs releases. The EDD current and charge values may be defined by symbolic equations that include the operators and functions listed in the "Short description of mathematical functions" entry in the Qucs help index[31].

[31] The Qucs operators and functions are a superset of those defined in the Verilog-A language manual. However, in some cases the name of the operator or function differs slightly. For example Verilog-A uses $pow(x, y)$ for the power function whilst Qucs uses $\wedge$ to denote x^y. An example of differing function names are the inverse trigonometric functions. A list of the available functions is given in Appendix A.

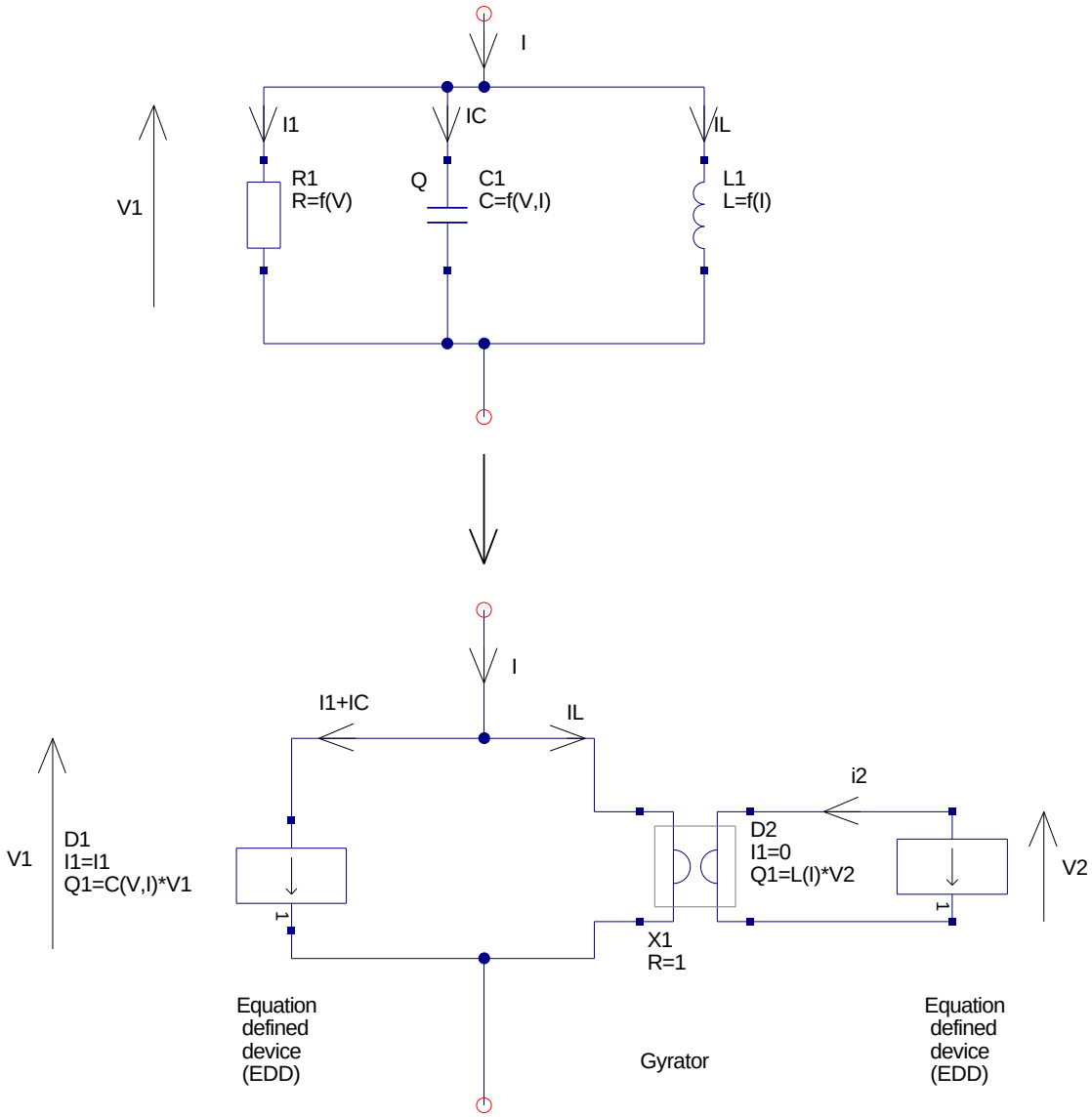

Figure 17: A non-linear two terminal branch with parallel resistive, capacitive and inductive components

Modelling nonlinear resistors

In many measurement applications a transducer is employed to transform changing values of a physical quantity to, say, changes in resistance. Often the resistive characterstics of these devices are nonlinear. To demonstrate how the EDD can be used to model a nonlinear resistance the example shown in Fig. 18 is introduced. In this schematic an EDD represents a resistance that is a function of the applied voltage across it's terminals. This example deliberately shows an extreme case where the resistance changes in a resistive pulse like fashion as the terminal voltage increases. The example also introduces for the first time the ternary ? : operator and illustrates how it can be nested to give an "if then else" structure to define the component properties. A point of note with these very nonlinear devices centres around the fact that it is possible to define components that have discontinuities in their I-V characteristics[32]. The EDD current equation defines how the resistance of this device changes with changing terminal voltage. This equation is given by

```
I1=V1/((V1<1.0) ? 1000 : (V1<2.0)
                ? 1000+4000*(V1-1) : (V1<5.0)
                ? 5000 : ((V1 >=5.0) &&  (V1<6.0))
                ? 5000-4500*(V1-5.0) : 500)
```

Which in terms of an "if then else" type statement is equivalent to:

```
I1 = V1/( if (V1 < 1.0) then 1000
          else if (V1 <  2.0) then 1000 + 4000*(V1-1)
          else if (V1 <  5.0) then 5000
          else if ((V1 >= 5.0) && (V1 < 6.0)) then 5000 - 4500*(V1-5.0)
          else 500 )
```

[32]One effect of such a discontinuity is the introduction of rapidly changing circuit conditions which can cause the simulator difficulties in converging to a correct solution. Sometimes, if this happens, simulation run times may be dramatically increased or simulation fails altogether.

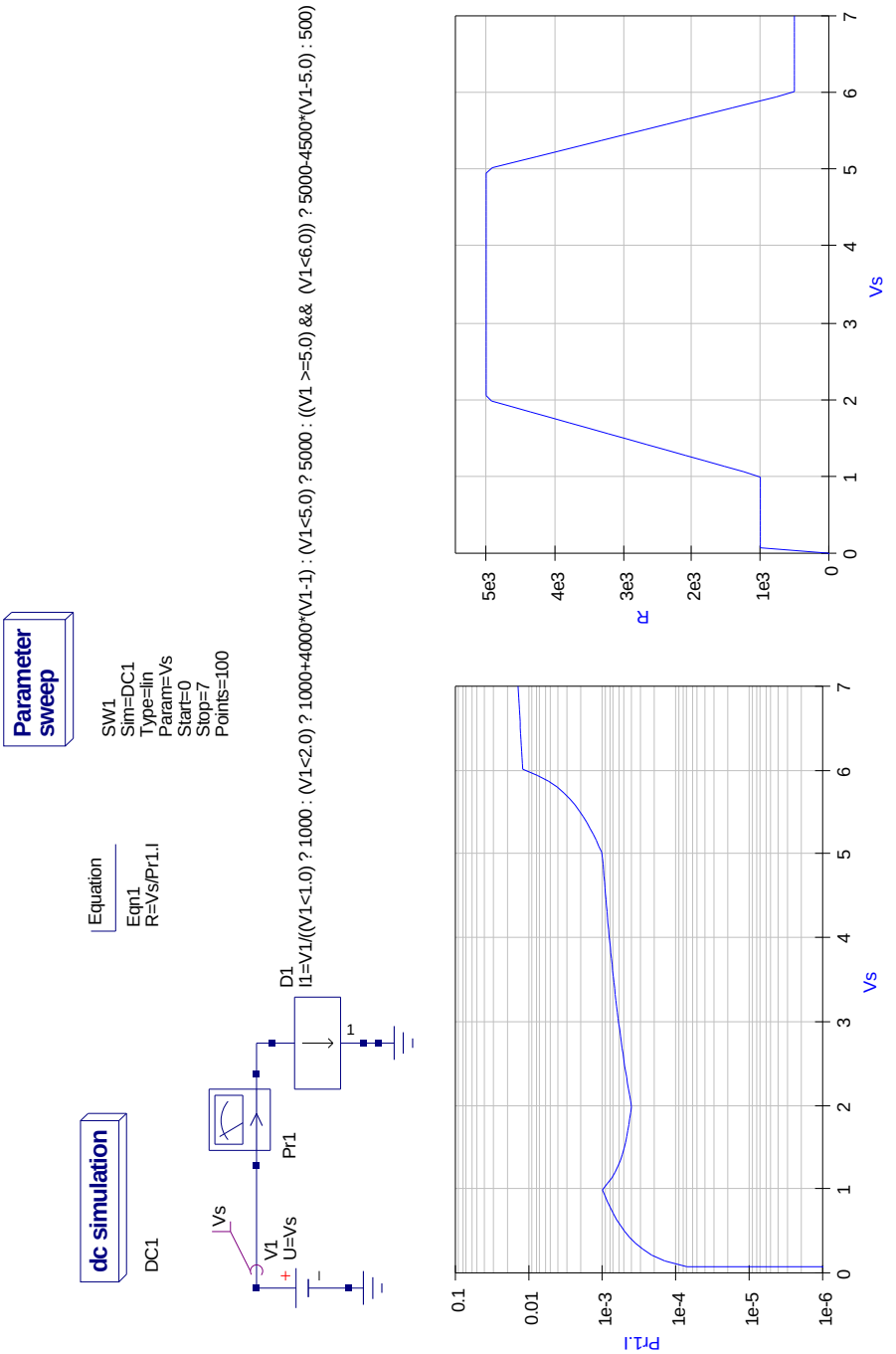

Figure 18: Qucs nonlinear resistor model

Modelling nonlinear capacitors and inductors

Nonlinear capacitors, who's C value is a function of terminal voltage, and nonlinear inductors, who's L value is a function of terminal current, commonly act as control elements in electronic systems. SPICE 2g6 includes a nonlinear symbolic polynomial form of C and L[33]. The schematic shown in Fig. 19 illustrates how a nonlinear capacitor can be modelled by an EDD. This model is based on a SPICE like polynomial function with four coefficients; C0, C1, C2 and C3[34]. The test circuit is a simple RC network with nominally identical R and C component values to those shown in Fig. 2. Increasing the value of DC source V1 also increases C which in turn decreases the RC low pass filter -3dB frequency. This effect is very visible in Fig. 19. The nonlinear changes in C are also clearly illustrated in the output voltage and phase curves. The schematic symbol for the nonlinear capacitor is shown in Fig. 19 with a red ring drawn around the normal capacitor symbol. This denotes an EDD based component. An alternative convention is to use red lettering within a symbol. The test circuit and simulation results for a nonlinear inductance are shown in Fig. 20. The EDD model is similar to the SPICE 2g6 nonlinear inductance model with four coefficients. This number can be increased, if required, by extending the EDD polynomial expression. A gyrator is employed with the EDD to model the nonlinear inductance. The effect of nonlinear inductance on the inductance current is shown by the difference between probe currents *Pr1* and *Pr2*.

[33]The details of these polynomial functions are presented in Test Reports 4 and 5 of the SPICE to Qucs testing Series, Mike Brinson, `http://qucs.sourceforge.net/docs.html`.

[34]SPICE 2g6 allows up to twenty coefficients. Simply add more higher order terms to the Qucs polynomial if required.

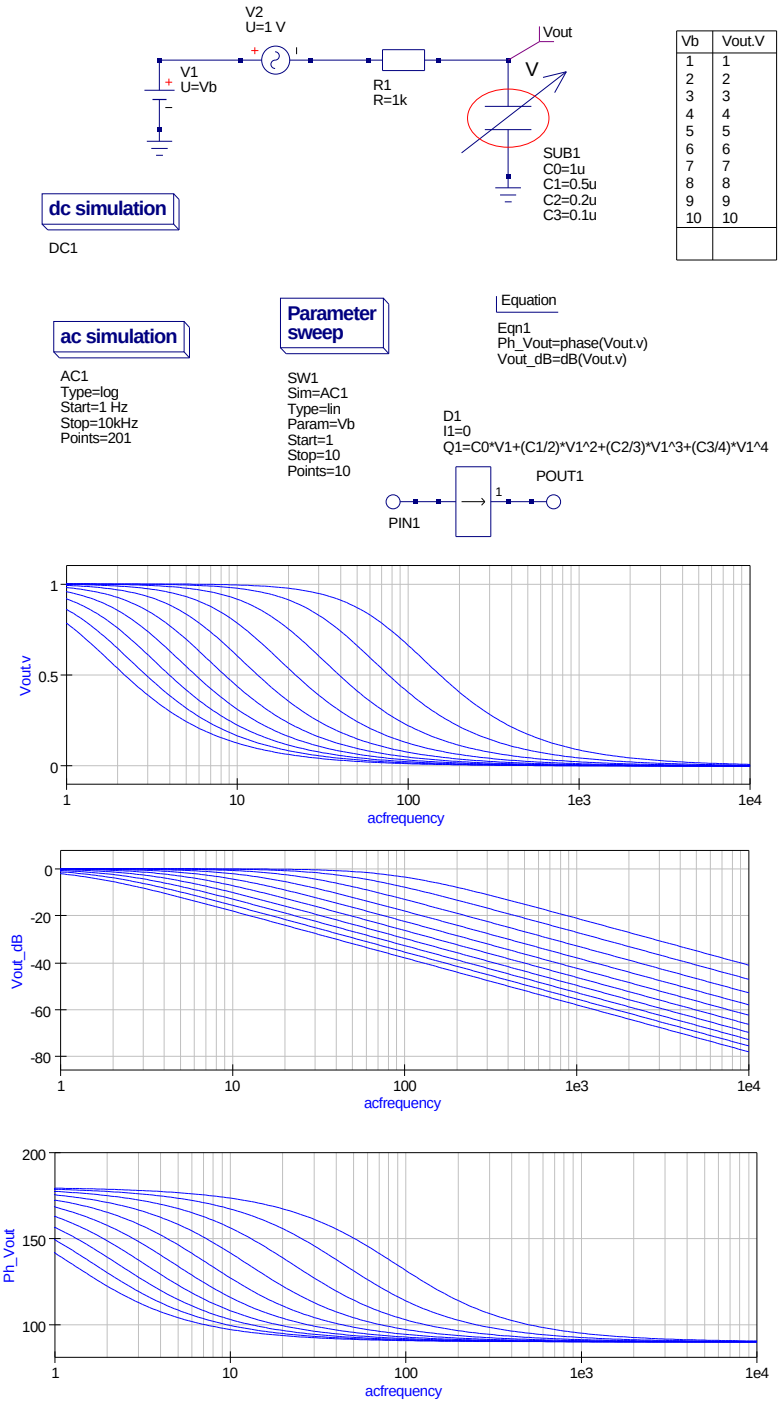

Figure 19: Qucs nonlinear capacitor model

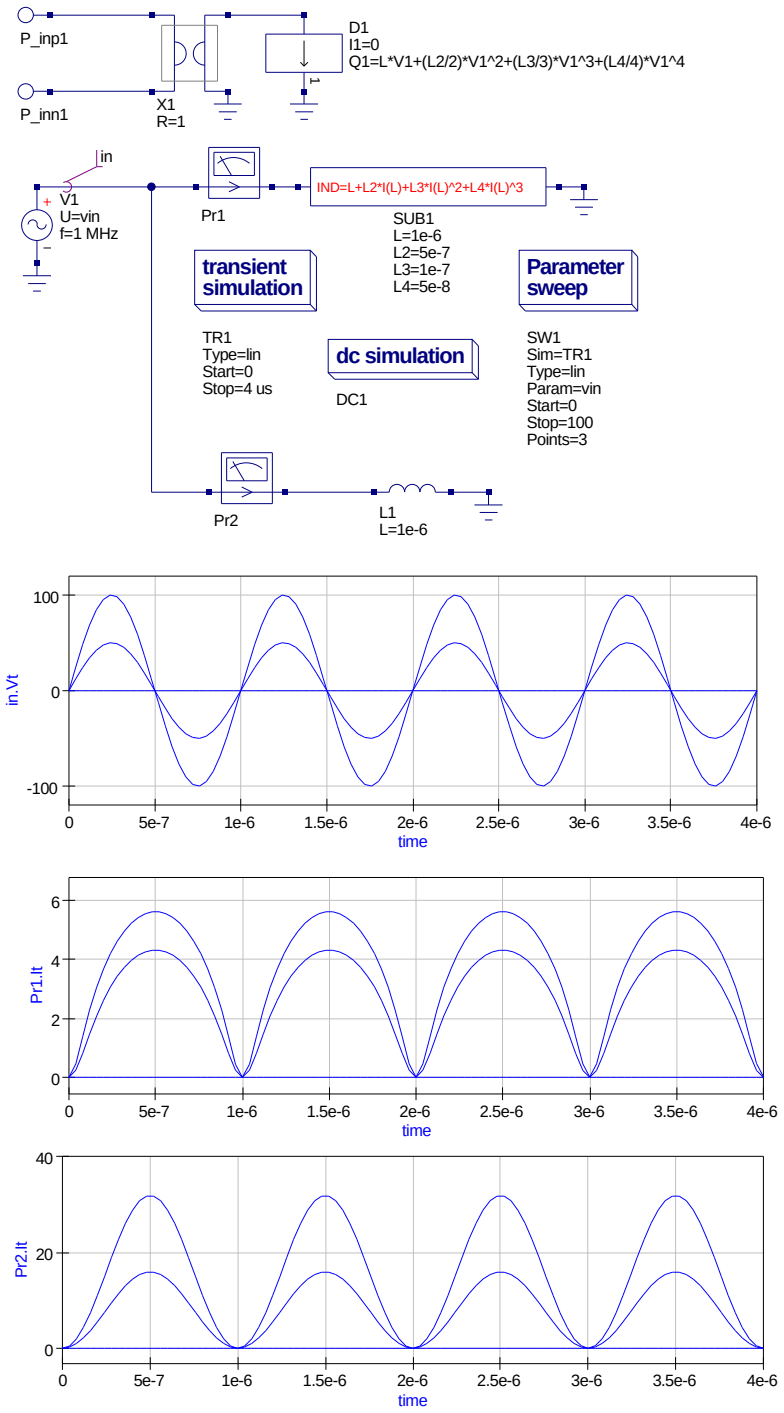

Figure 20: Qucs nonlinear inductor model

Compact device modelling using EDD

Semiconductor device models are a corner stone of all circuit simulators. Often they are characterised by the same parameters as those found in the SPICE 2g6 and 3f5 diode, BJT, FET and MOS models.[35]. Since the original SPICE semiconductor device models where first developed many new extensions to these models have been proposed. Unfortunately, adding such models to a circuit simulator is a complex process, being both time consuming and requiring specialised knowledge. For the average Qucs user the hand coded C++ model generation route is one that they would not contemplate attempting because of the depth of knowledge and specialised skills required. The Qucs EDD was devised to promote fast, and straight forward, prototyping of semiconductor compact models, allowing a wider Qucs population the opportunity to try their hand at device model construction. To demonstrate the stages needed to generate an EDD model of a semiconductor device a compact model of a diode is introduced in this section[36].

The DC diode current I_d is given by the following functions of diode voltage V_d[37].

$$I_d = I_s \cdot \left(\exp\left(V_d/(n \cdot Vt) - 1\right) + V_d \cdot GMIN, \quad \forall \left(-5 \cdot n \cdot Vt \leq V_d\right)\right. \tag{18}$$

$$I_d = -I_s + V_d \cdot GMIN, \quad \forall \left(-BV < V_d\right) \quad \text{and} \quad \left(V_d < -5 \cdot n \cdot Vt \leq V_d\right) \tag{19}$$

$$I_d = -IBV, \quad \forall \left(V_d = -BV\right) \tag{20}$$

$$I_d = -I_s \cdot \left(\exp\left(-(BV + V_d)/Vt\right) - 1 + BV/Vt\right), \quad \forall \left(V_d < -BV\right). \tag{21}$$

In these equations:

- I_s = the saturation current.
- n = the emission coefficient.
- $GMIN$ = a small conductance in parallel with the diode[38]

[35]The SPICE 2g6 and 3f5 device parameters are a subset of those commonly provided with current generation of circuit simulators, including Qucs.

[36]A second three terminal MESFET transistor example is available for downloading from the Qucs Web site.

[37]These equations are for the SPICE 2g6 diode model, see Giuseppe Massobrio, Chapter 1, Pn-junction diode and Schottky diode, Semiconductor device modeling with SPICE, Edited by Paolo Antognetti, Giuseppe Massobrio, 1988, McGraw-Hill,Inc, ISBN 0-07-002107-4.

[38]GMIN is added to help Qucs DC convergence. The SPICE default value is 1e-12S.

- $Vt = kB \cdot T/q$, where T is the diode temperature in Kelvin, kB is Boltzmann's constant and q the charge on the electron.
- BV = reverse breakdown voltage (positive number)
- IBV = reverse breakdown current (positive number).

Figure 21 gives the EDD model for the experimental semiconductor diode. The ternary operator ?: is used to select the correct equation for each diode operating region. The diode current I_d is the sum of EDD branch currents $I1$ to $I4$, where $I1$ represents the diode forward bias region, $I2$ the reverse bias region and $I3$ plus $I4$ the diode reverse bias breakdown region. When calculating diode current a special form of the exponential function exp(), called limexp(), is employed to assist Qucs to converge to a solution during DC and transient large signal analysis. The function limexp() linearises the exponential function at large argument values minimising the possibility of floating point overflow and generation of software exceptions. The $I_d - V_d$ characteristic curves shown in Fig. 21 are for the forward bias region with series resistance rs set to 0.01Ω. For completeness the simulation data for the Qucs built-in diode are also given. Clearly the two sets of results are very similar. The DC simulation results for the diode reverse breakdown region of operation are shown in Fig. 22. Again for comparison an $I_d - V_d$ plot for the Qucs built-in diode is also provided. In this region of operation some slight differences are apparent: although for both devices the reverse breakdown is very close to 100V the slope of the $Id - Vd$ curve at negative voltages beyond -BV is different, emphasising that the SPICE diode model does not model breakdown or zener effects well[39].

The next stage in the development of the diode model is to add capacitance effects: depletion layer capacitance for the reverse bias region and diffusion capacitance for the forward bias region. Diode capacitance is given by:

- Depletion layer capacitance

$$C_{dep} = \frac{dQ_{dep}}{dV_d} = Area \cdot Cj0 \left(1 - \frac{V_d}{V_j}\right)^{-m} \tag{22}$$

- Diffusion capacitance

$$C_{diff} = \frac{dQ_{diff}}{dV_d} = tt \cdot \frac{dI_d}{dV_d} \tag{23}$$

[39]See Steven M. Sandler, SPICE subcircuit accurately models zener characteristics, Personal Engineering, November 1998, pp 45-48 for more information on this subject.

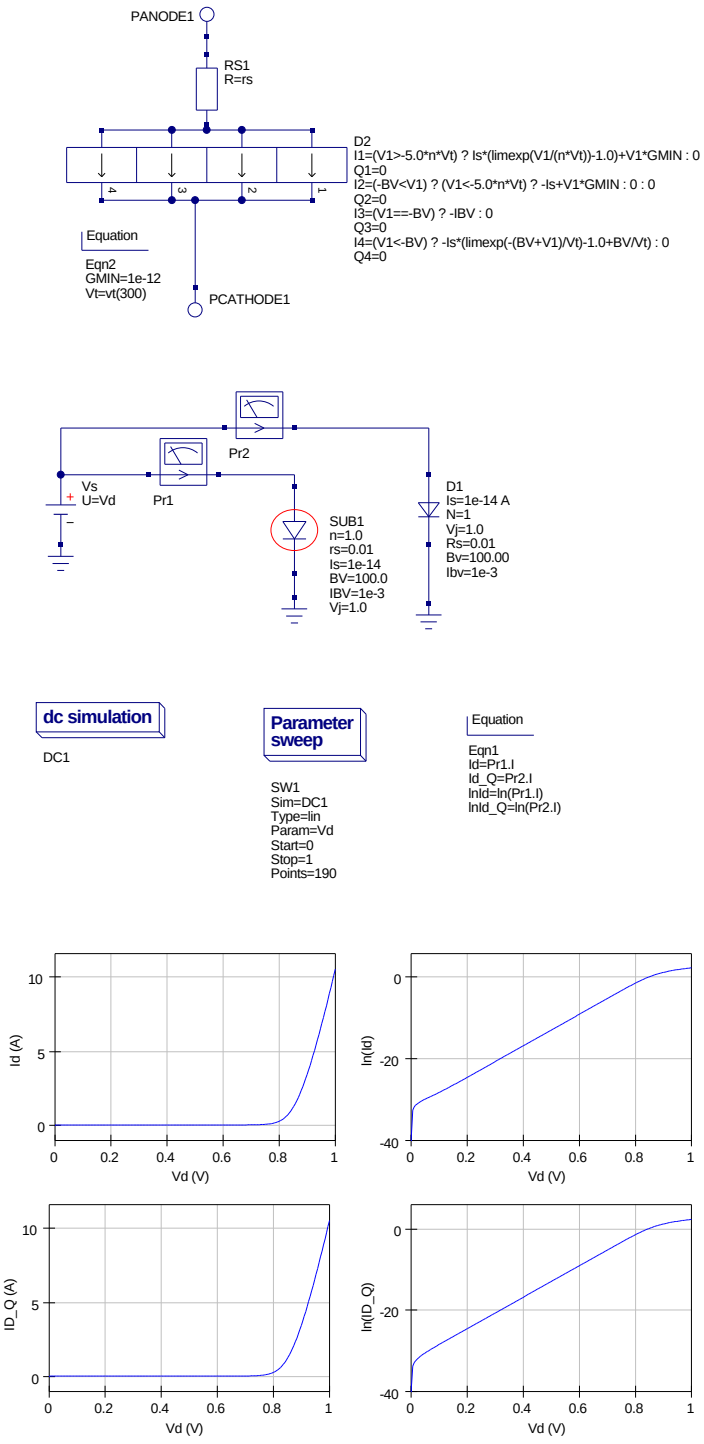

Figure 21: Compact diode model DC test circuit and simulation results: SUB1 is the EDD diode model and D1 the Qucs diode model with the same parameters as SUB1.

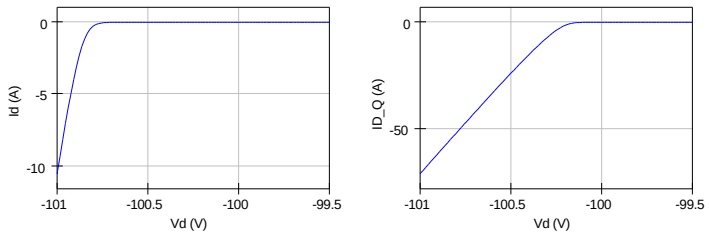

Figure 22: Compact diode model DC simulation results for the reverse breakdown region of operation

Where the total stored charge $Q_d = Q_{dep} + Q_{diff}$. Using the same notation as the SPICE diode model:

$$Q_{diff} = tt \cdot I_d \tag{24}$$

$$Q_{dep} = Area \cdot Cj0 \int_0^{V_d} \left(1 - \frac{V_d}{V_j}\right)^{-m} dV, \quad \forall \, (V_d <= FC \cdot V_j) \tag{25}$$

Using integration formula $\int (ax+b)^n dx = \dfrac{1}{a}\dfrac{(ax+b)^{1+n}}{1+n}$ and simplifying yields:

$$Q_{dep} = \frac{Area \cdot Cj0 \cdot V_j}{1-m}\left[1 - \left(1 - \frac{V_d}{V_j}\right)^{1-m}\right] \tag{26}$$

Also, in the forward bias region

$$Q_{dep} = Area \cdot Cj0 \cdot F1 + \frac{Area \cdot Cj0}{F2} \int_{FC \cdot V_j}^{V_d} \left(F3 + \frac{m \cdot V_d}{V_j}\right) dV, \quad \forall \, (V_d >= FC \cdot V_j) \tag{27}$$

On integrating

$$Q_{dep} = Area \cdot Cj0 \left[F1 + \left(\frac{1}{F2}\right) \cdot \left\{F3 \cdot (V_d - FC \cdot V_j) + \left(\frac{m}{2 \cdot V_j}\right) \cdot \left(V_d^2 + (FC \cdot V_j)^2\right)\right\}\right] \tag{28}$$

Where

$$F1 = \frac{V_j}{1-m}\left[1 - (1-FC)^{1-m}\right\}, F2 = (1-FC)^{1+m}, F3 = 1 - FC \cdot (1+m) \tag{29}$$

In these equations:

- FC = Coefficient for forward-bias depletion capacitance.
- m = Grading coefficient.
- tt = Transit time.
- $Area$ = Device area.
- $Cj0$ = Zero-bias junction capacitance.

Figure 23 shows the extended diode model. The C_{dep} and C_{diff} components of the device capacitance have been included in the EDD model as stored charge Q1 and Q2. Again the ternary operator ?: is employed to select the correct equation for each section of the diode DC operating range. An equation block is used to simplify the charge equations through the use of factors F1, F2 and F3.[40]. An area factor has also been added to the EDD model in Fig. 23. This is introduced to allow simulation of two or more equivalent parallel devices. The diode variables scaled by area are:

$$I_s(A) = I_s \cdot Area, \quad Cj0(A) = Cj0 \cdot Area, \quad \text{and} \quad r_s(A) = rs/Area. \tag{30}$$

The test circuit shown in Fig. 23 illustrates how device capacitance and resistance can be determined as a function of diode bias voltage. Firstly, the diode S parameters are determined at a given bias voltage, secondly these are converted to Y parameters and the diode capacitance (Cap) and resistance (RD) extracted from $Y[1,1]$, and finally the variation of Cap and RD with diode voltage V_d plotted using the Qucs plotting function PlotVs. Notice that the value of Cap at $V_d = 0V$ agrees with the value of $Cj0$.

To complete the demonstration EDD diode model all that remains to do is to add temperature dependence to the current and capacitance equations. Circuit simulators normally use two temperatures to determine device temperature dependence; the first called Tnom represents the temperature that the device parameters were measured, and the second called Temp represents the current device temperature. A high percentage of the diode parameters are temperature dependent. However, to simplify the demonstration diode model only the temperature dependence of parameters I_s, Vj and $Cj0$ will be included in the model. Adding extra temperature dependence to the diode model is left to readers as an exercise[41]. One of the

[40] In complex current and charge expressions precalculating subexpressions in equation blocks ensures that they are only calculated once at the beginning of a simulation, ensuring minimum run times for an EDD model.

[41] For example, parameters m and BV are both temperature dependent.

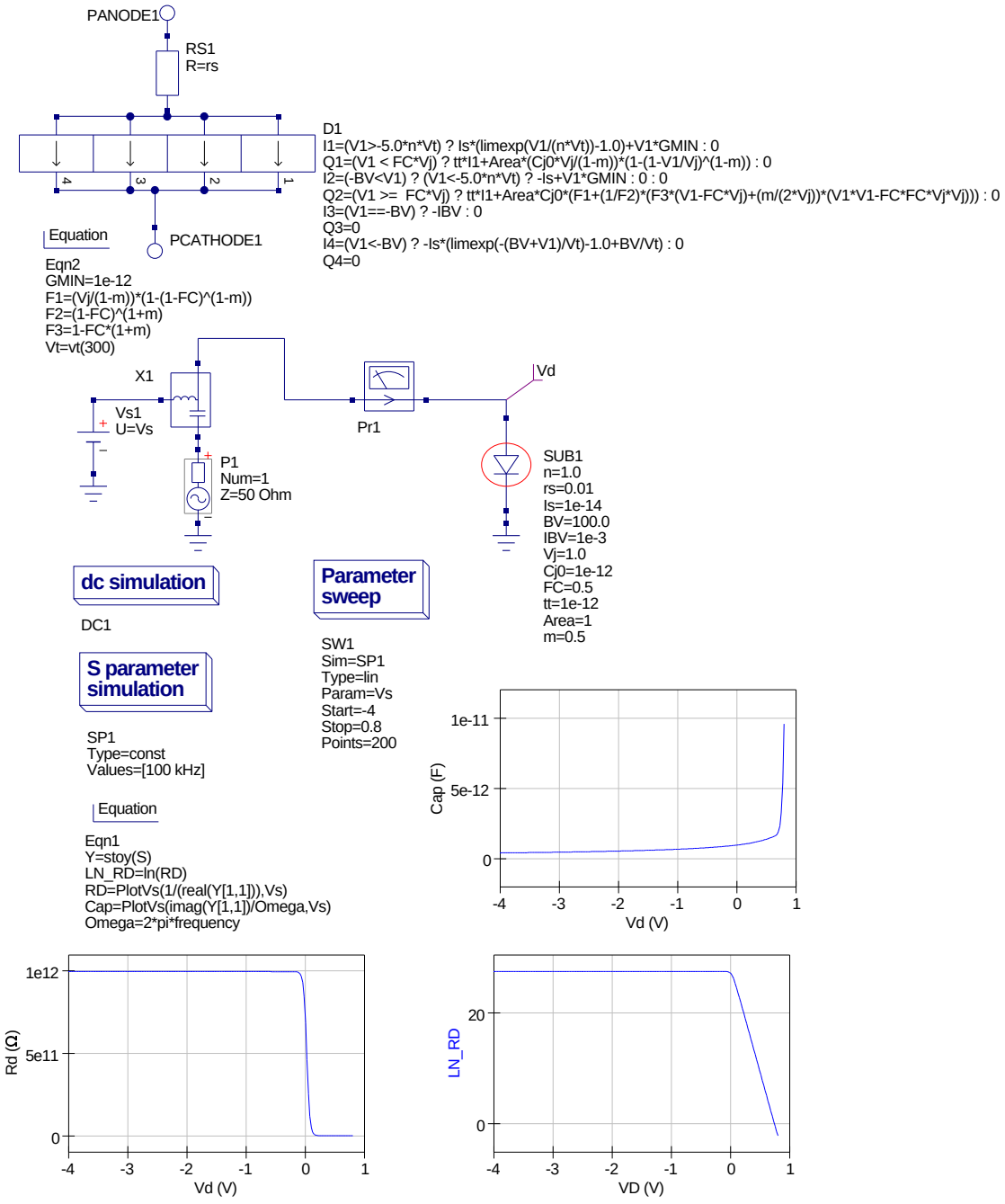

Figure 23: Compact diode model capacitance and resistance simulation

great advantages of the EDD style of modelling is that it is interactive allowing easy experimentation with models to any given level. The following equations list the temperature dependence of I_s, Vj and $Cj0$.

Let T1 = Tnom and T2 = Temp, then

$$I_s(T2) = I_s(T1) \left\{ \frac{T2}{T1} \right\}^{\frac{XTI}{n}} \exp \left[\frac{-q \cdot Eg(300)}{kB \cdot T2} \left(1 - \frac{T2}{T1} \right) \right] \tag{31}$$

$$Vj(T2) = \frac{T2}{T1} \cdot Vj(T1) - \frac{2 \cdot kB \cdot T2}{q} \ln \left(\frac{T2}{T1} \right)^{1.5} - \left[\frac{T2}{T1} \cdot Eg(T1) - Eg(T2) \right] \tag{32}$$

$$Cj0(T2) = Cj0(T1) \left[1 + m \left\{ 400 \cdot 10^{-6} \left(T2 - T1 \right) - \frac{Vj(T2) - Vj(T1)}{Vj(T1)} \right\} \right] \tag{33}$$

In these equations:

- XTI = Saturation current temperature exponent.
- $Eg(T) = EG(0) - \dfrac{7.02e - 4 \cdot T^2}{1108 + T}$, the energy gap.

Figure 24 shows the extended EDD for the experimental diode model. Again the limexp() function is used in preference to the standard exp() function in the temperature calculations listed in equations block Eqn2. The test circuit in Fig. 24 sweeps the device temperature from 20 to 80 degrees Centigrade. The graph inlay illustrates the experimental diode current I_d plotted as a function of temperature. The temperature of the built-in Qucs diode is held constant, at room temperature, and it's current Id_Q plotted as an overlay. The two curves cross at room temperature, indicating identical currents at this temperature.

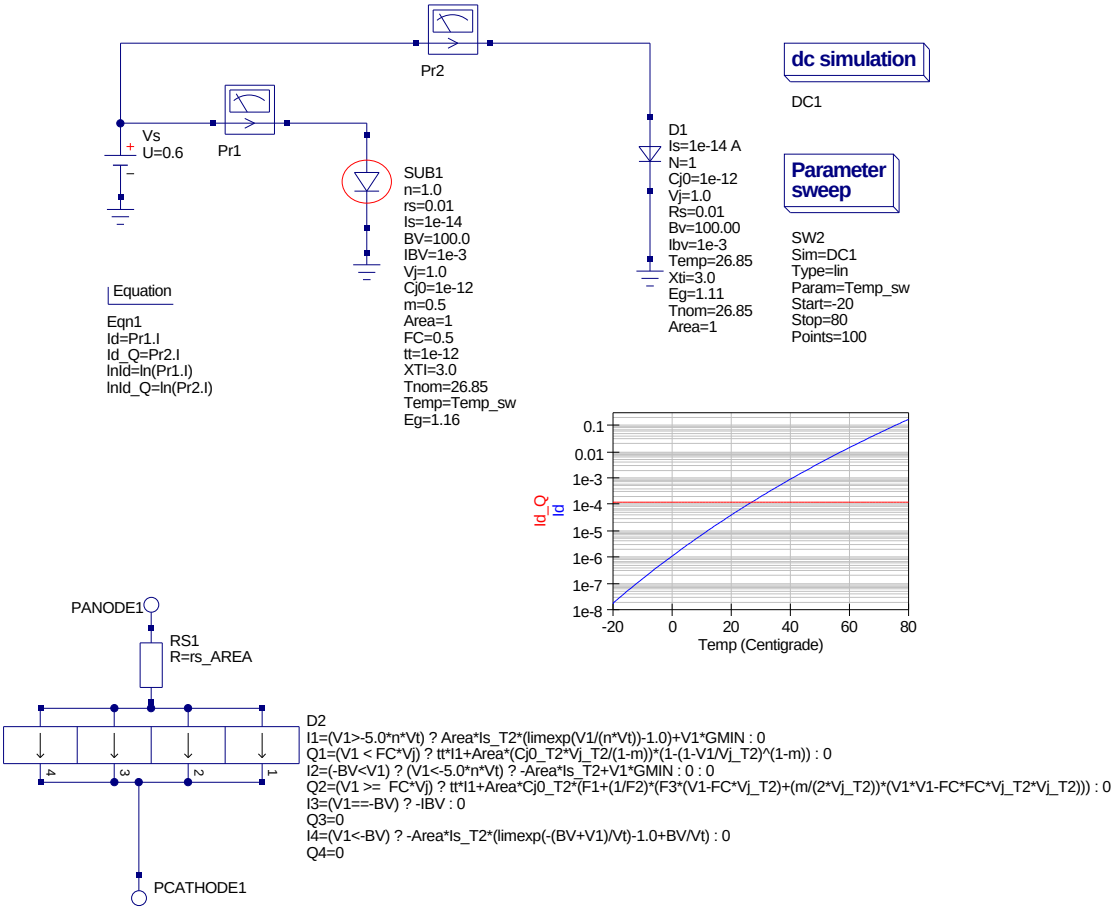

Figure 24: Compact diode model with temperature dependence

Constructing EDD compact device models and circuit macromodels

Component equations, subcircuits with parameters and EDD models are major developments for the Qucs circuit simulator. They provide advanced modelling capabilities with enough power and flexibility to allow a much greater range of models to be developed than the ones currently provided with each Qucs release. In the future it is proposed to add new models to the Qucs Web site. The Qucs team is very keen to encourage all Qucs users to support the modelling effort. If you have constructed a new model and would like to share it with other Qucs users please post your model on the qucs-devel or qucs-help mailing lists. Both the model schematic file and a brief outline of its operation and specification are requested. An example model specification for the Curtice MESFET device can be found on the Qucs Web site. Please use the same format when writing model descriptions.

End Note

This tutorial note introduces a large number of new modelling concepts and shows how equations, subcircuits with parameters and the new equation defined device perform a central role in constructing Qucs models. The EDD approach to modelling makes possible, for the first time, the construction of equation defined compact device models and circuit macromodels using the Qucs schematic capture facilities as an interactive modelling medium. This is a major step forward for Qucs. Once again these notes are very much a record of work in progress: much still remains to be done in the future to improve the modelling capabilities provided by Qucs. A major short term task will be the development of additional models covering as wide a range of applications as possible. If Qucs is to fulfill it's mission to become a truly universal circuit simulator then it must be supported by models. Some readers will have noticed that these notes include very little information about the ADMS-Verlog-A and hand coded C++ model development routes. This was a deliberate decision on my part. Sometime in the future I intend to return to these subjects and update the tutorial. A very special thank you must go to Stefan Jahn for all his hard work, skill, and dedication during the period he has worked on programming the amazing modelling capabilities now embedded in Qucs.

Appendix A: Qucs constants, operators and functions

This appendix lists the constants, operators and a number of functions that are available for constructing Qucs equations. Items in [...] indicate the equivalent object in the Verilog-A language. The functions listed are common to Qucs and Verilog-A. A number of other functions have been implemented in Qucs. The full list can be found in the Qucs help system; "Short Description of mathematical Functions" or in the Qucs "Measurement Expression Reference Manual" by Gunther Kraut and Stefan Jahn, `http://qucs.sourceforge.net/docs.html`.

- Constants
  1. pi = 3.141593...
  2. e = 2.718282...
  3. kB = 1.380651e-23 J/K
  4. -q = -1.602177e-19 C
- Operators
  1. +x unary plus
  2. -x unary minus
  3. x+y addition
  4. x-y subtraction
  5. x*y multiplication
  6. x/y division
  7. x%y modulo (remainder)
  8. x^y power [pow(x,y)]
  9. ?: ternary (condition) ? (expression if true) : (expression if false)
  10. || logical or
  11. && logical and
  12. == equal
  13. < less than
  14. <= less than or equal to
  15. > greater than
  16. >= greater than or equal to
  17. != not equal to
  18. ( ) brackets

- Functions

  1. ln(x) natural logarithm
  2. log10(x) decimal logarithm [log(x)]
  3. exp(x) exponential function base e
  4. sqrt(x) square root
  5. min(x,y) minimum
  6. max(x,y) maximum
  7. abs(x) absolute value
  8. sin(x) sine
  9. cos(x) cosine
  10. tan(x) tangent
  11. arcsin(x) inverse sine [asin(x)]
  12. arccos(x) inverse cosine [acos(x)]
  13. arctan(x[,y]) inverse tangent [atan2(x,y)]
  14. sinh(x) hyperbolic sine
  15. cosh(x) hyperbolic cosine
  16. tanh(x) hyperbolic tangent
  17. arsinh(x) inverse hyperbolic sine [asinh(x)]
  18. arcosh(x) inverse hyperbolic cosine [acosh(x)]
  19. artanh(x0 inverse hyperbolic tangent [atanh(x)]
  20. limexp(x) argument limited exponential function
  21. hypot(x,y) Euclidean distance function

Appendix B: Constructing subcircuits with parameters

In this appendix a series of screen dumps illustrate the sequence needed to construct a subcircuit with parameters. A simple series resonance circuit has been chosen for the demonstration.

Enter the series resonance circuit and add input and output pins

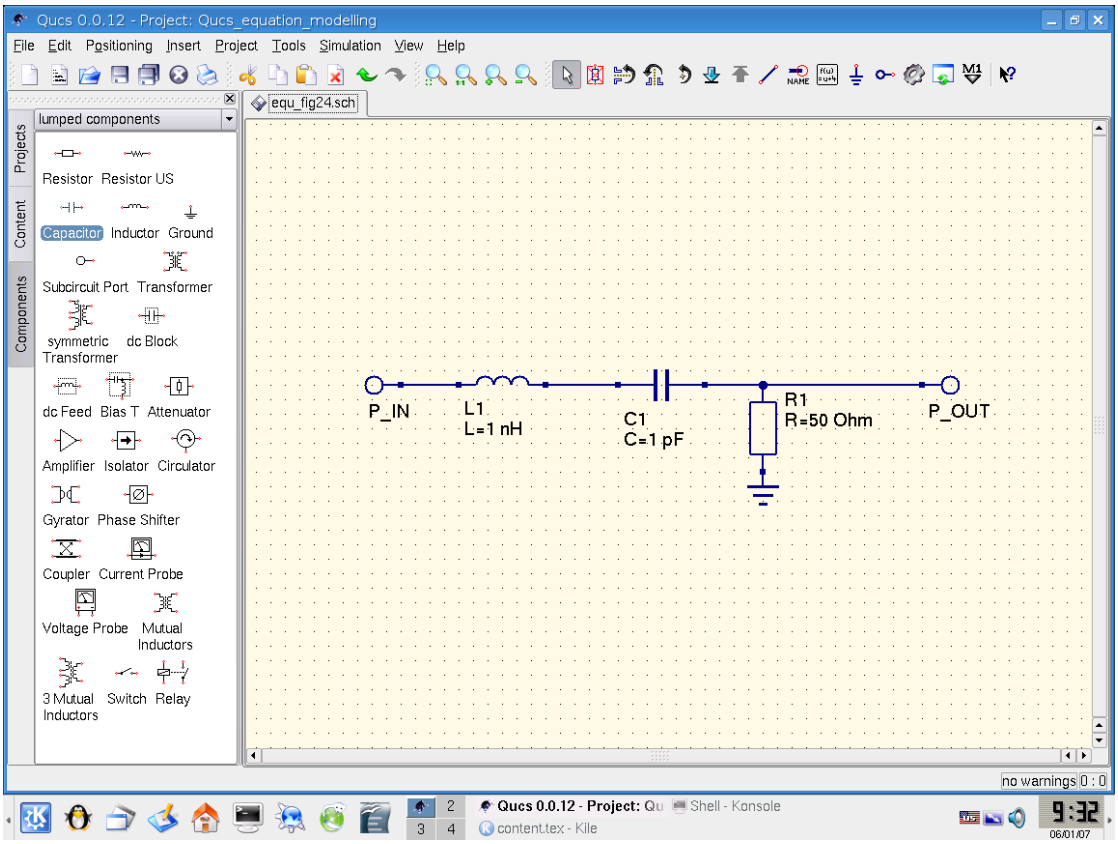

Figure 25: Stage 1: screen dump showing LCR circuit

Change the component names to Ls, Cs and Rs

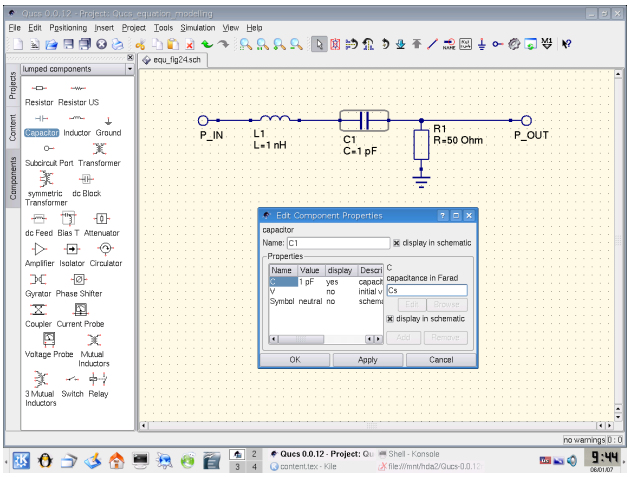

Figure 26: Stage 2: screen dump showing LRC circuit

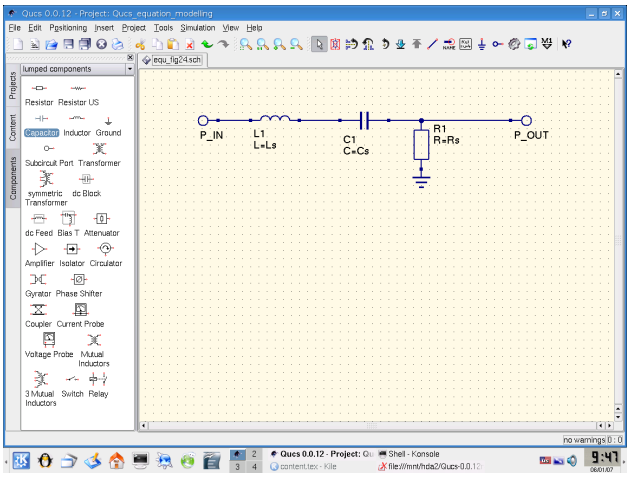

Figure 27: Stage 2: screen dump after name changes

Construct symbol for new subcircuit

Right click on the Qucs drawing area and select Edit Circuit symbol or press key F9. Edit the drawing symbol to give the design shown in Fig. 28.

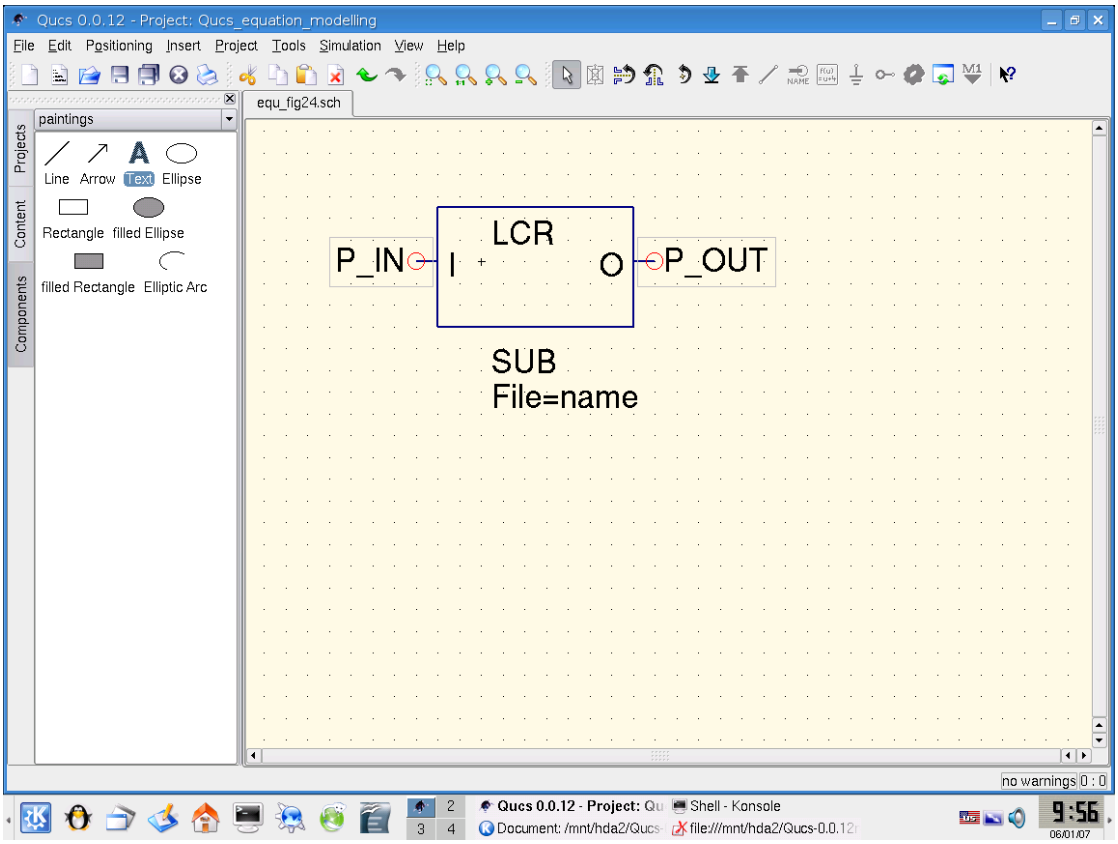

Figure 28: Stage 3: the subcircuit symbol

Add the names of the subcircuit parameters to the LCR symbol

Right click on the SUB / File=name caption and enter names of subcircuit parameters with their default values.

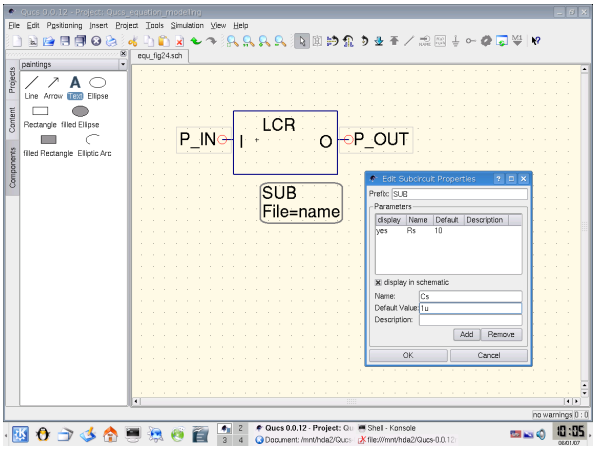

Figure 29: Stage 4: entering subcircuit parameter names and default values

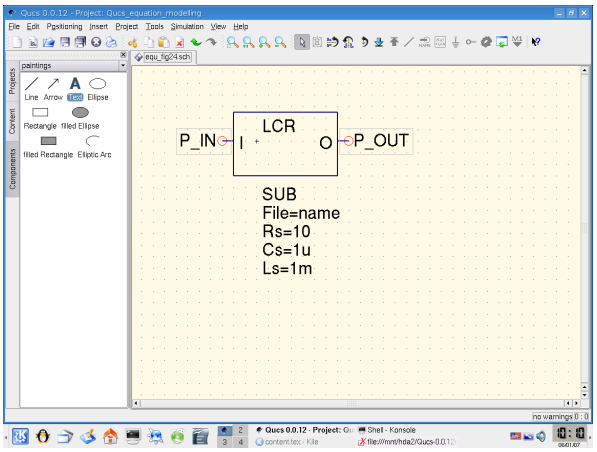

Figure 30: Stage 4: resulting subcircuit and parameter list with default values

Test the LCR subcircuit

Figure 31 gives a simple AC transfer function test circuit and resulting waveforms. Parameter *R_SW* is swept over the range 1Ω to 10Ω and the AC transfer function recorded and plotted.

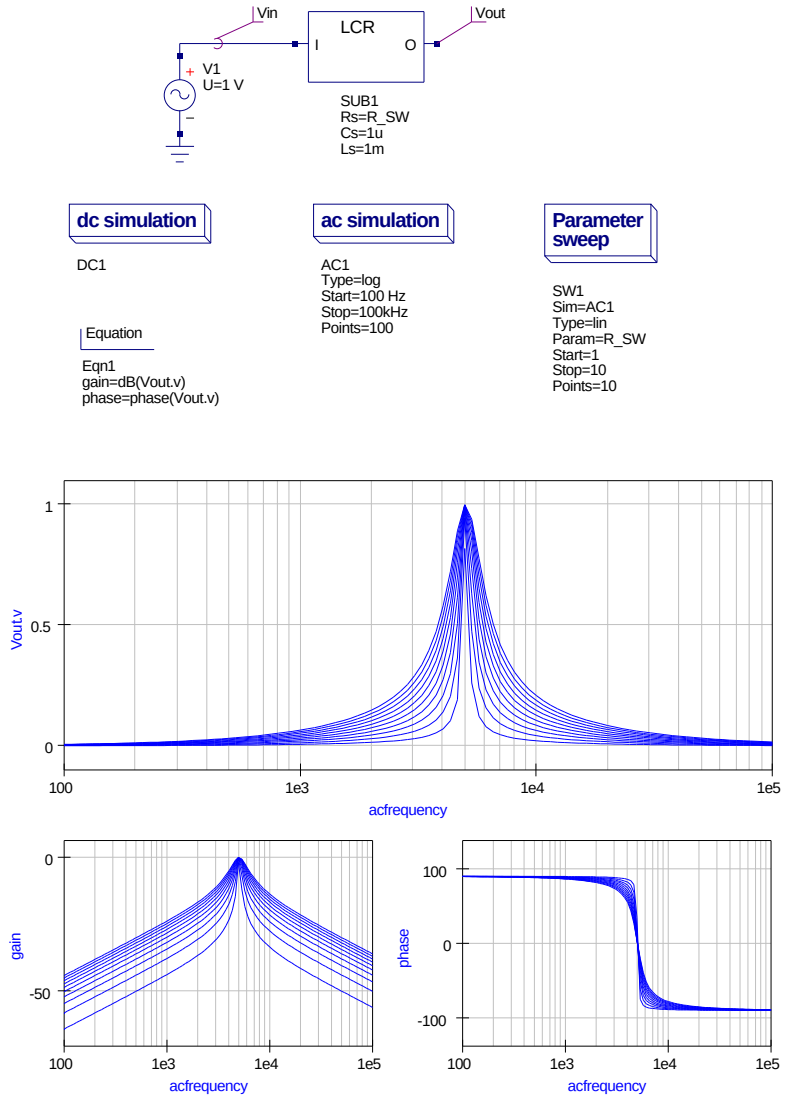

Figure 31: Stage 5: Subcircuit test circuit and output waveforms