Qucs: improvements and new directions in GPL compact device modelling and circuit simulation tools

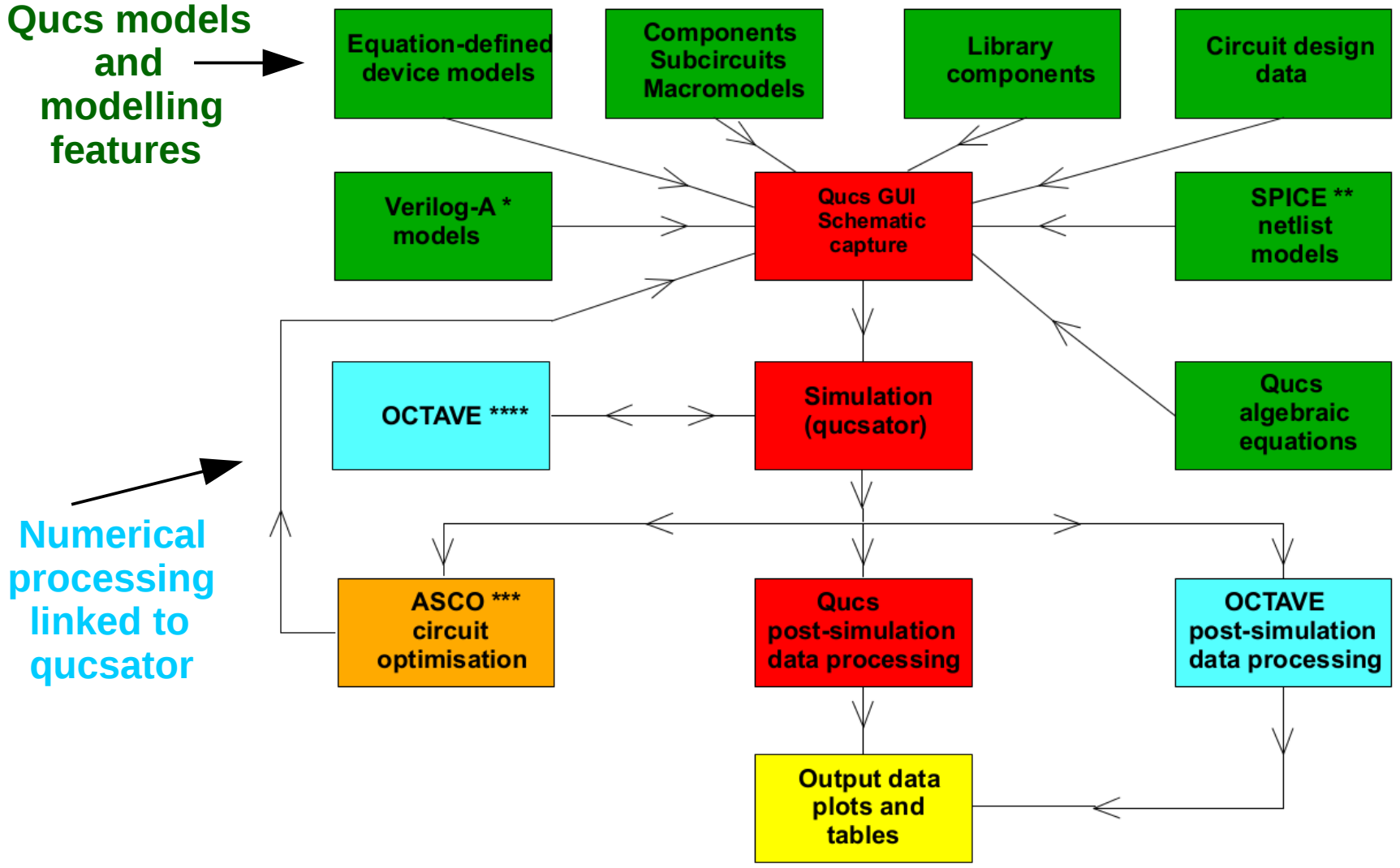
Mike Brinson, Centre for Communications Technology, London Metropolitan University, UK, mbrin72043@yahoo.co.uk.
Richard Crozier, The University of Edinburgh, UK, richard.crozier@yahoo.co.uk.
Vadim Kuznetsov, Bauman Moscow Technical University, Russia, ra3xdh@gmail.com.
Clemens Novak, Qucs Developer, clemens@familie-novak.net.
Bastien Roucaries, Laboratoire SATIE – CNRS UMR 8929, Université de Cergy-Pontoise, ENS Cachan, FR, bastien.roucaries@satie.ens-cauchan.fr.
Frans Schreuder, Nikhef, Amsterdam, NL, fransschreuder@gmail.com.
Guilherme Brondani Torri, Qucs Developer, guitorri@gmail.com.

Plus contributions from members of the Qucs user community.

- **From Qucs-0.0.18 to Qucs-0.0.19: the way forward**
- **Octave support for Qucs circuit simulation**
- **Expanded compact device modelling capabilities with the Berkeley Model and Algorithm Prototyping Platform (MAPP)**
- **Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools**
- **Linking ngspice Xyce to Qucs**
- **New circuit design aids**
- **Improvements to Qucs documentation and code control features**
- **Possible directions for Qucs development after release 0.0.19**

Presented at the MOS-AK Spring Workshop at DATE, Grenoble, France, March 12, 2015.

From Qucs-0.0.18 to Qucs-0.0.19: the way forward
• Part 1 - Qucs-0.0.18



GPL software used by Qucs-0.0.18

* ADMS – Automatic model synthesiser, http://sourceforge.net/projects/mot-adms

** PS2SP – PSPICE to SPICE preprocessor, http://members.acon.at/fschmid7/

*** ASCO – SPICE circuit optimiser, http://asco.sourceforge.net/

**** OCTAVE – Numerical analysis package, https://www.gnu.org/software/octave/

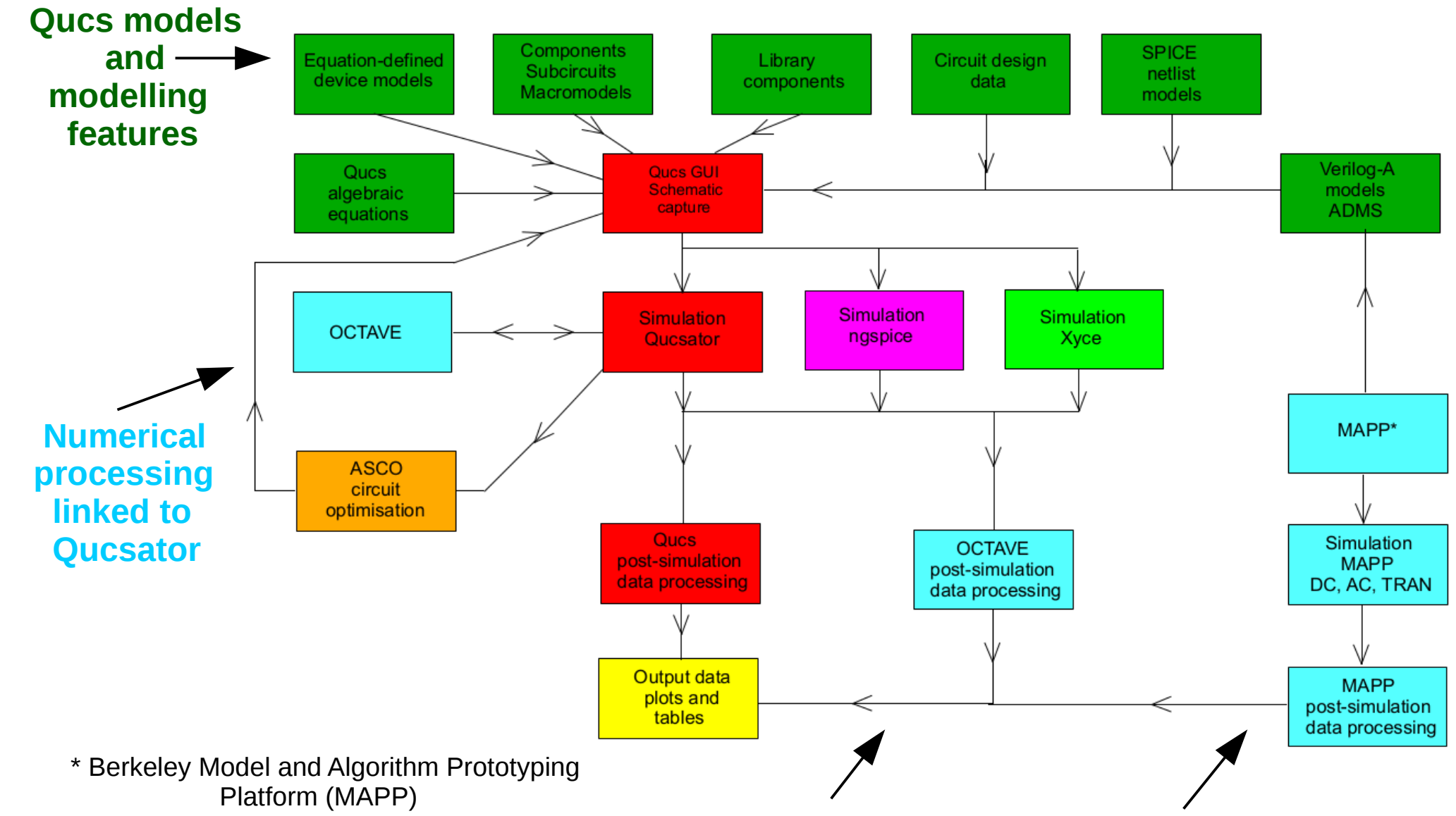
From Qucs-0.0.18 to Qucs-0.0.19: the way forward
• Part 2 - Qucs-0.0.18 download statistics

Qucs-0.0.18 Download statistics between 1 August 2014 to 15 Jan 2015

Downloads 74,922 Top Country United States 15% Top OS Windows 78%

	Country	Android	BSD	Linux	Macintosh	Solaris	Unknown	Windows	Total
1.	United States	272	13	520	1,392	0	1,572	7,741	11,510
2.	India	172	1	445	89	0	322	4,189	5,218
3.	Germany	23	4	349	443	0	1,784	2,520	5,123
4.	France	21	7	178	223	0	26	2,929	3,384
5.	Spain	19	0	204	347	0	204	2,398	3,172
6.	Russia	34	3	182	130	0	115	2,624	3,088
7.	Italy	34	0	287	281	0	14	2,405	3,021
8.	Japan	12	8	73	184	0	19	2,439	2,735
9.	United Kingdom	28	0	112	258	4	122	1,973	2,497
10.	Brazil	21	1	155	91	0	33	1,634	1,935
11.	Mexico	9	0	92	333	0	6	1,402	1,842
12.	Canada	7	2	65	240	0	15	1,380	1,709
13.	Argentina	3	0	77	38	0	9	1,502	1,629
14.	Poland	9	6	73	40	0	5	1,361	1,494
15.	Colombia	6	0	55	116	0	7	1,020	1,204
16.	Indonesia	11	0	49	32	0	9	971	1,072
17.	Australia	9	4	44	138	0	6	843	1,044
18.	Portugal	8	0	30	77	0	3	910	1,028
19.	Netherlands	6	3	55	105	0	111	737	1,017
20.	South Africa	18	0	23	31	0	22	905	999

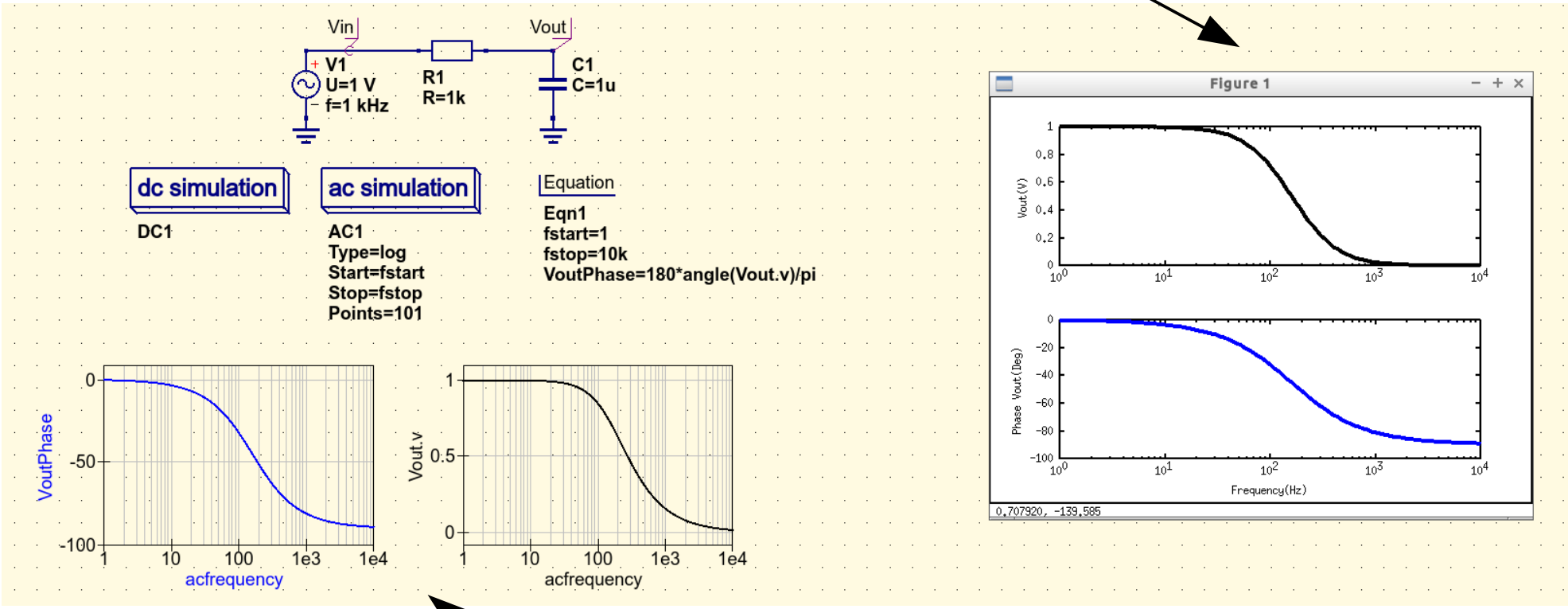
From Qucs-0.0.18 to Qucs-0.0.19: the way forward
• Part 3 - Introducing the Qucs-0.0.19 structure



* Berkeley Model and Algorithm Prototyping Platform (MAPP)
http://draco.eecs.berkeley.edu:8765/jr/MAPP/wikis/home

Octave support for Qucs circuit simulation
Part 1 – post simulation data processing

Octave Post-simulation data processing

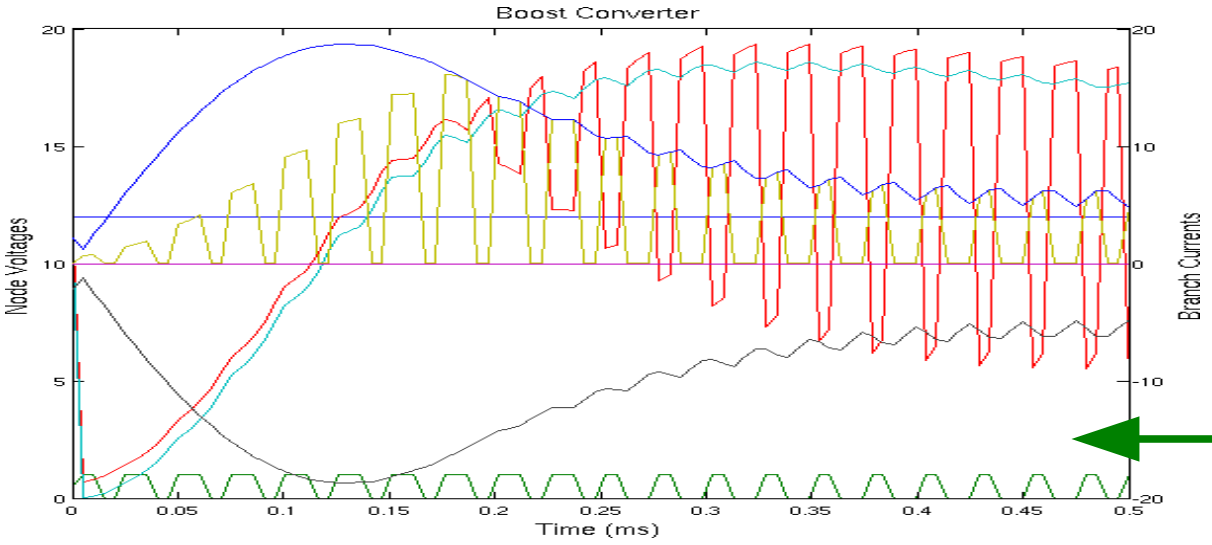


Qucs Post-simulation data processing

Octave support for Qucs circuit simulation
Part 2 – Qucs-Octave linked transient simulation

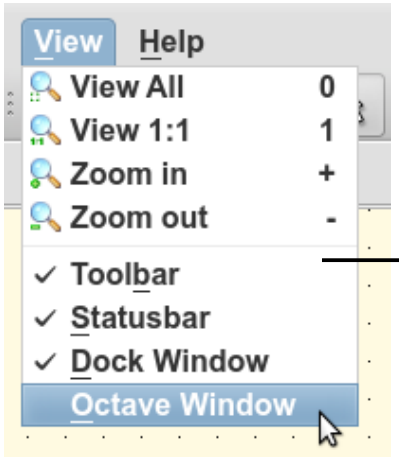
```
# Qucs 0.0.18  C:/Documents and Settings/s0237326/My Documents/Temp/boostconverter.sch
# boostconverter.net
L:L1 _net0 dio L="47uH" I="0"
Vdc:V2 _net0 gnd U="12V"
Eqn:Eqn1 Tmax="Bperiod*20" Tstep="Bperiod/1000" Export="yes"
.ETR:ETR1 IntegrationMethod="Trapezoidal" Order="2" InitialStep="1 ns" MinStep="Tstep"
    MaxIter="150" reltol="0.001" abstol="1 pA" vntol="1 uV" Temp="26.85" LTEreltol="1e-3"
    LTEabstol="1e-6" LTEfactor="1" Solver="CroutLU" relaxTSR="no" initialDC="yes" MaxStep="Tstep"
Eqn:Eqn2 Bfreq="40k" Bperiod="1/Bfreq" Bduty="50" Ton="Bperiod*Bduty/100" Toff="Bperiod-Ton"
Export="yes"
Relais:S1 ctrl dio gnd gnd Vt="0.5 V" Vh="0.1 V" Ron="1" Roff="1e12" Temp="26.85"
Vrect:V1 ctrl gnd U="1V" TH="Ton" TL="Toff" Tr="1 ns" Tf="1 ns" Td="0 ns"
Diode:D1 out dio Is="1e-12 A" N="1" Cj0="10 fF" M="0.5" Vj="0.7 V" Fc="0.5" Cp="0.0 fF" Isr="0.0" Nr="2.0"
   Rs="0.0 Ohm" Tt="0.0 ps" Ikf="0" Kf="0.0" Af="1.0" Ffe="1.0" Bv="0" Ibv="1 mA" Temp="26.85" Xti="3.0"
   Eg="1.11" Tbv="0.0" Trs="0.0" Ttt1="0.0" Ttt2="0.0" Tm1="0.0" Tm2="0.0" Tnom="26.85" Area="1.0"
C:C1 out gnd C="100u" V="0"
R:R1 gnd out R="5" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
```

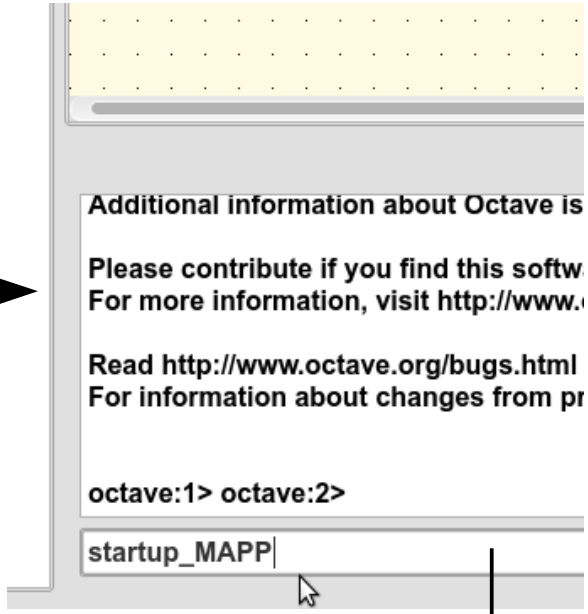
Qucs netlist: boostconverter.net




```
% boost_converter_example.m
%
%
cd(fileparts(which('asynchronous_boost_converter_example.m')));
Tstart = 0; n = 100;  tend = 5e-4;
% fixed-step synchonous solver test
clear qtr_async1
% create a new asynchronous solver object from a netlist
qtr_async1 = asynctrcircuit('boostconverter.net');
% initialist the simulation
qtr_async1.init(tstart, (tend - tstart) / (10 * n));
% get the number of nodes
N = qtr_async1.getn;
% get the number of voltage sources
M = qtr_async1.getm;
% choose some time points
T1 = linspace(tstart, tend, n)';
% initialise storage for the solution
Y1 = zeros(numel(T1), M+N);
% get the initial solution
Y1(1, 1:(N+M)) = qtr_async1.getsolution();
% step through time solving the circuit
for ind = 2:numel(T1)
    % accept the step into the solution history
    qtr_async1.acceptstep(T1(ind));
    % get the node voltages and currents at the current time
    Y1(ind, 1:(N+M)) = qtr_async1.getsolution();
end
% plot the node voltages and branch currents
figure;
AX = plotyy(T1 * 1000, Y1(:,1:N), T1 * 1000, Y1(:,((N+1):(N+M))));
title ('Boost Converter', 'FontSize', 14)
xlabel('Time (ms)', 'FontSize', 14)
set(get(AX(1),'Ylabel'),'String','Node Voltages', 'FontSize', 14)
set(get(AX(2),'Ylabel'),'String','Branch Currents', 'FontSize', 14)
```

Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools
Part 1: Launching MAPP fom Qucs

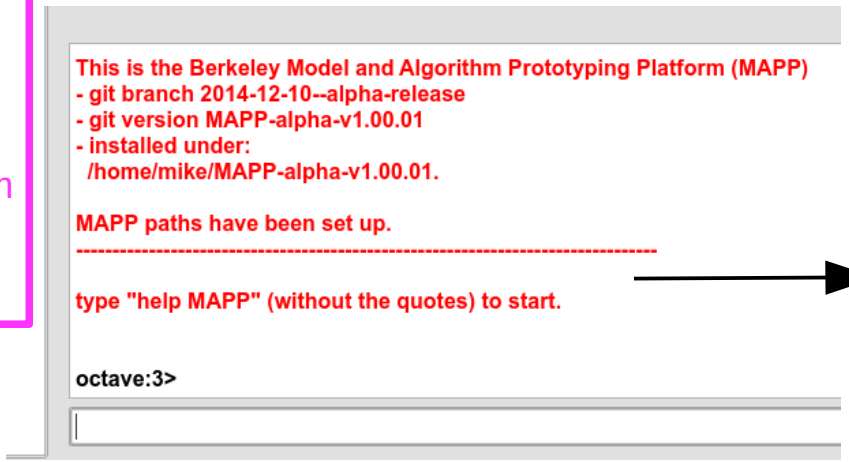




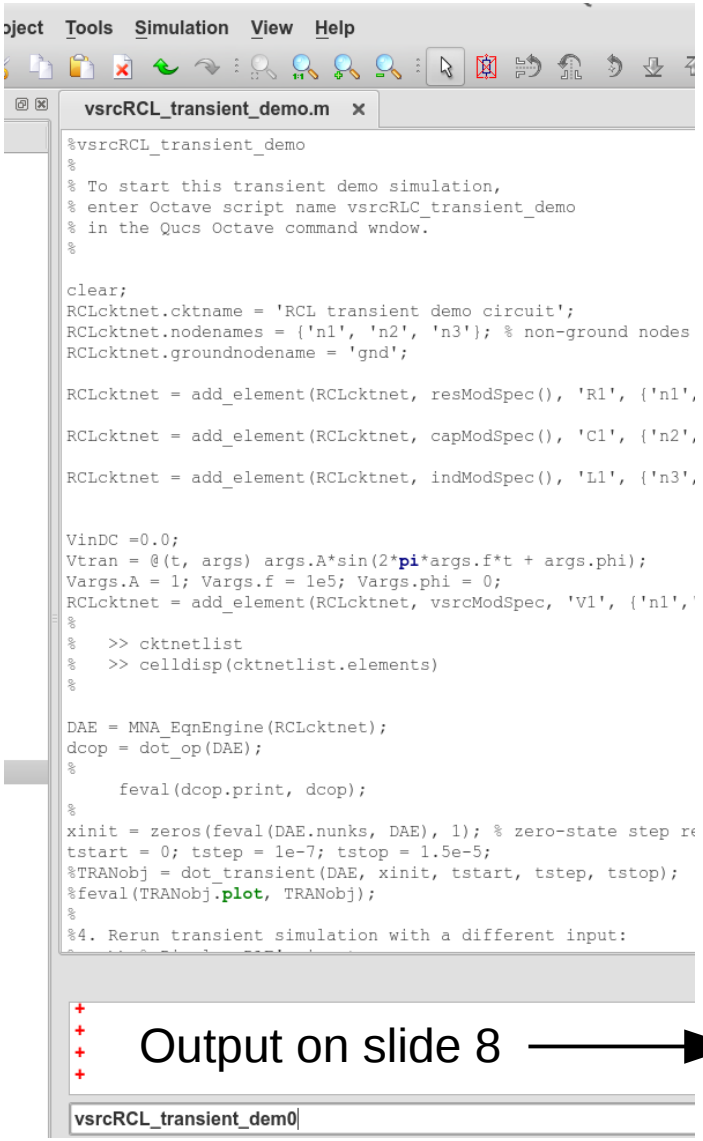
Qucs project:

MAPPExamples_prj

Startup_MAPP.m
vsrcRLC_transient_demo.m
..
..



MAPP code in Qucs edit window

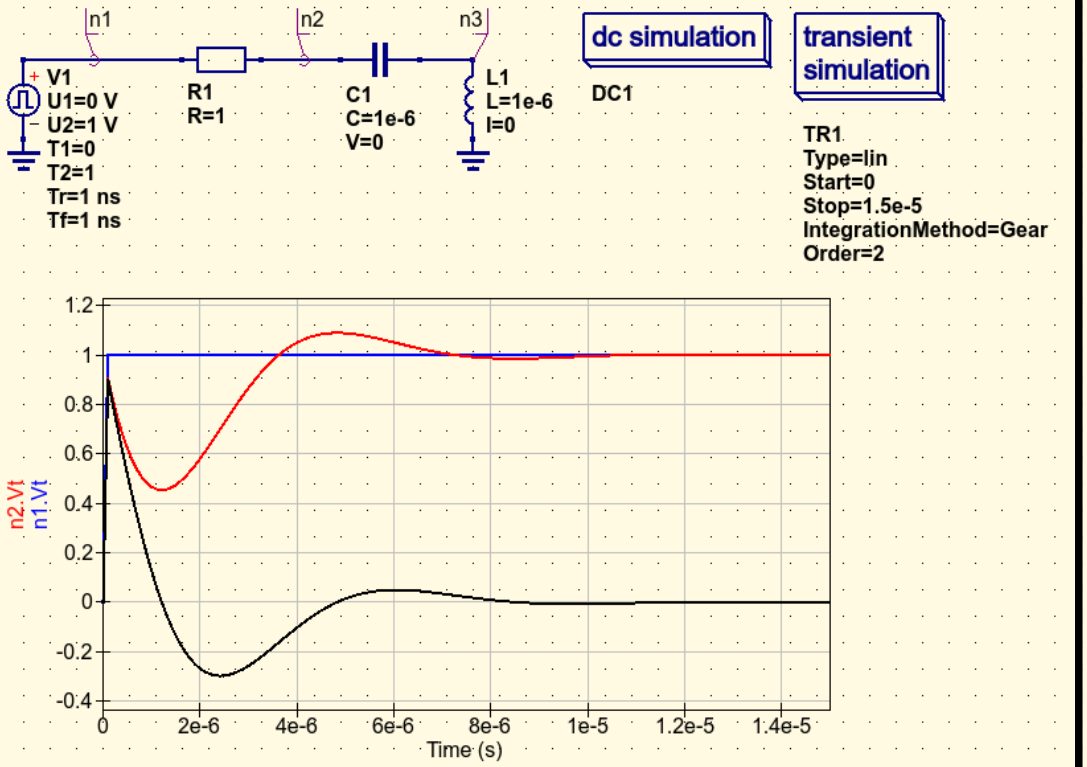


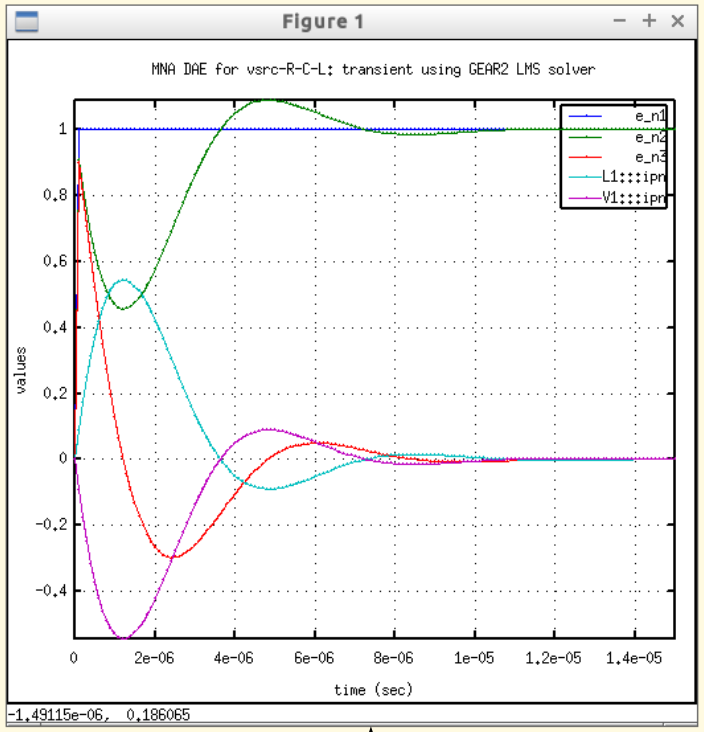
Background to MAPP:

http://www.mos-ak.org/berkeley_2014/presentations/07_Jaijeet_Roychowdhury_MOS-AK_Berkeley_2014.pdf
http://www.mos-ak.org/berkeley_2014/presentations/07_Tianshi_Wang_MOS-AK_Berkeley_2014.pdf

Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools
Part 2: Qucs/MAPP output

Qucs circuit schematic and output data





From slide 7

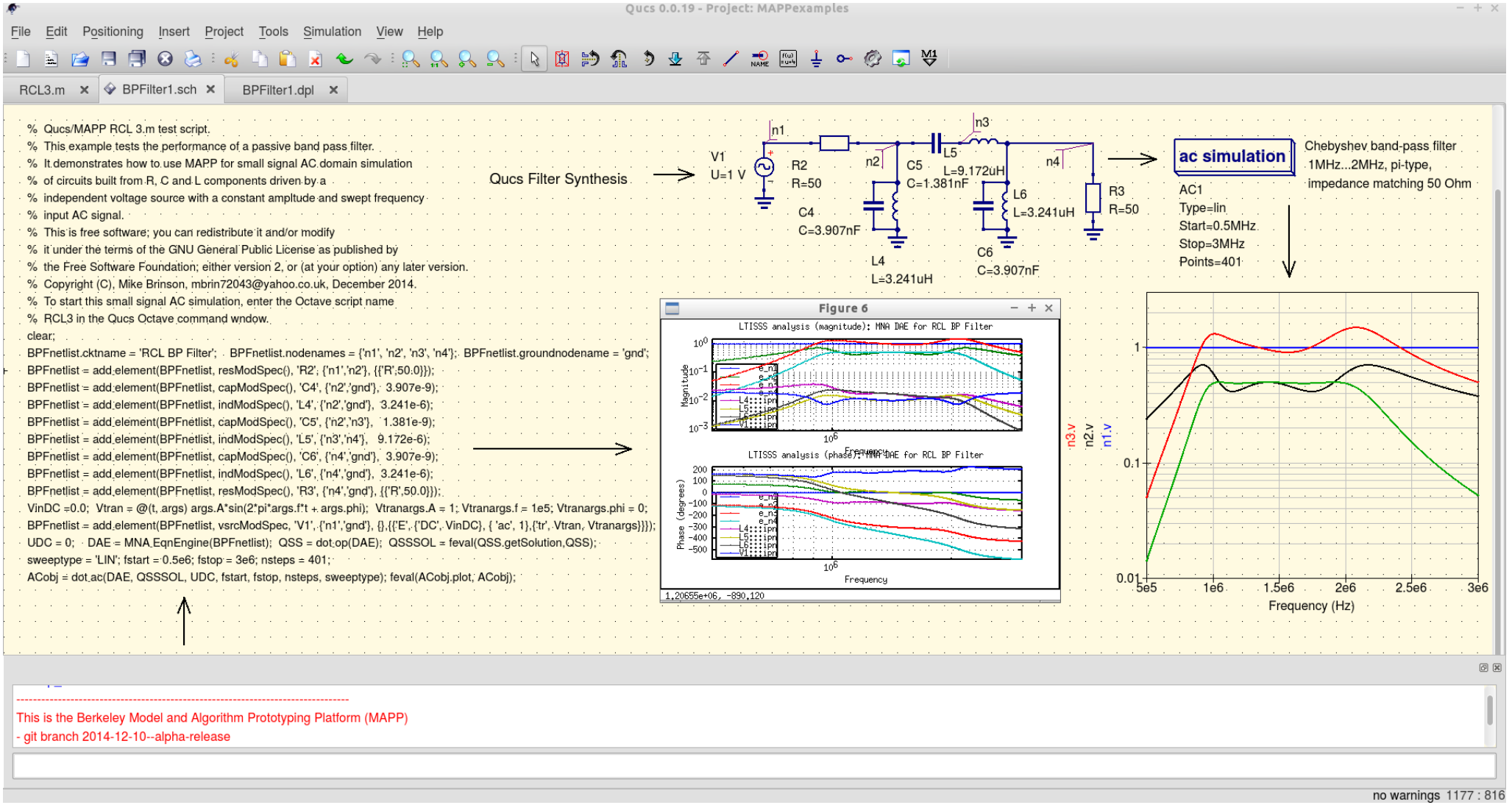
MAPP output data

Background to MAPP:

http://draco.eecs.berkeley.edu/dracotiki/tiki-index.php?page=MAPPfeatures

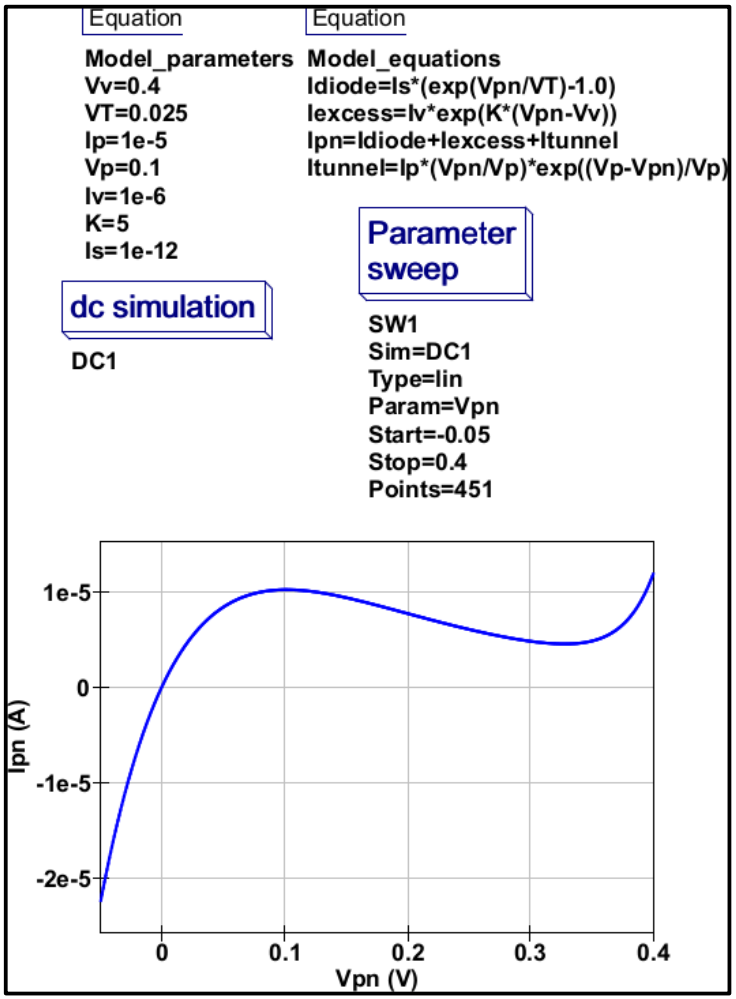
Jaijeet Roychowdhury, Numerical simulation and modelling of electronic and biochemical systems, Foundations and Trends in Electronic Design Automation 3:2-3, NOW, 2009

Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools
Part 3: From Qucs circuit synthesis to MAPP output



Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools
Part 4: Compact device modelling; the tunnel diode model revisited

Qucs evaluation of tunnel diode model equations




```
function plotIV_tunnelDiode_ModSpec_wrapper()
    MOD = tunnelDiode_ModSpec_wrapper();

    vs = -0.05:0.001:0.4;
    is = zeros(size(vs));

    S = ee_model_parm2struct(MOD);
    for idx = 1:1:size(is,2)
        S.vpn = vs(1,idx);
        is(1,idx) = MOD.fe_of_S(S);
    end
```

MAPP tunnel diode test wrapper

```
    figure();

    plot([min(vs), max(vs)], [0, 0], 'Color', 'red', ...
      'LineWidth', 1.25, 'LineStyle', '--');
    hold on;
    plot([0, 0], [min(is), max(is)]*1e6, 'Color', ...
       'red', 'LineWidth', 1.25, 'LineStyle', '--');
    h = plot(vs, is*1e6, 'Color', 'blue', 'LineWidth', 1.75);

        axis tight;
        box on;
        grid on;
        set(gca,'FontName','Times New Roman','FontSize', ...
           15,'FontWeight','bold');
        xlabel('vpn (V)','FontName','Times New Roman',...
           'FontSize',18,'FontWeight','bold');
        ylabel('ipn (uA)','FontName','Times New Roman',...
           'FontSize',18,'FontWeight','bold');
        title(['I/V curve of a tunnel diode'],'FontName',...
            'Times New Roman','FontSize',18,'FontWeight','bold');
        set(gcf,'color','white');
end
```

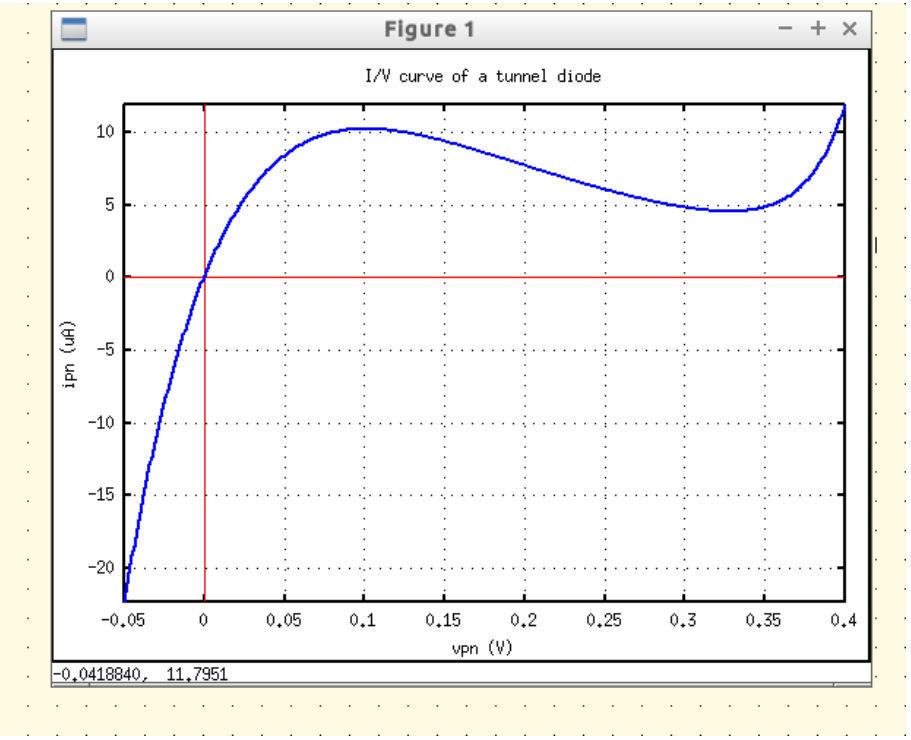

S. Jahn and M.E. Brinson, Interactive compact device modelling using Qucs equation-defined devices, Int. J. Numer. Model. 2008; 21:335-349. http://onlinelibrary.wiley.com/doi/10.1002/jnm.676/abstract
http://www.mos-ak.org/berkeley_2014/presentations/07_Tianshi_Wang_MOS-AK_Berkeley_2014.pdf

Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools
Part 4: Compact device modelling; the tunnel diode model continued

MAPP tunnel diode compact device model

```
function MOD = tunnelDiode_ModSpec_wrapper()
    MOD = ee_model();
    MOD = add_to_ee_model(MOD, 'external_nodes', {'p', 'n'});
    MOD = add_to_ee_model(MOD, 'explicit_outs', {'ipn'});
    MOD = add_to_ee_model(MOD, 'parms', {'Is', 1e-12, 'VT', 0.025});
    MOD = add_to_ee_model(MOD, 'parms', {'Ip', 1e-5, 'Vp', 0.1});
    MOD = add_to_ee_model(MOD, 'parms', {'Iv', 1e-6, 'Vv', 0.4, 'K', 5});
    MOD = add_to_ee_model(MOD, 'parms', {'C', 0});
    MOD = add_to_ee_model(MOD, 'f', @f);
    MOD = add_to_ee_model(MOD, 'q', @q);
    MOD = finish_ee_model(MOD);
end

function out = f(S)
    v2struct(S);
    I_diode = Is*(exp(vpn/VT)-1);
    I_excess = Iv * exp(K * (vpn - Vv));
    I_tunnel = (Ip/Vp) * vpn * exp(-1/Vp * (vpn - Vp));
    out = I_diode + I_tunnel + I_excess;
end

function out = q(S)
    v2struct(S);
    out = C*vpn;
end
```

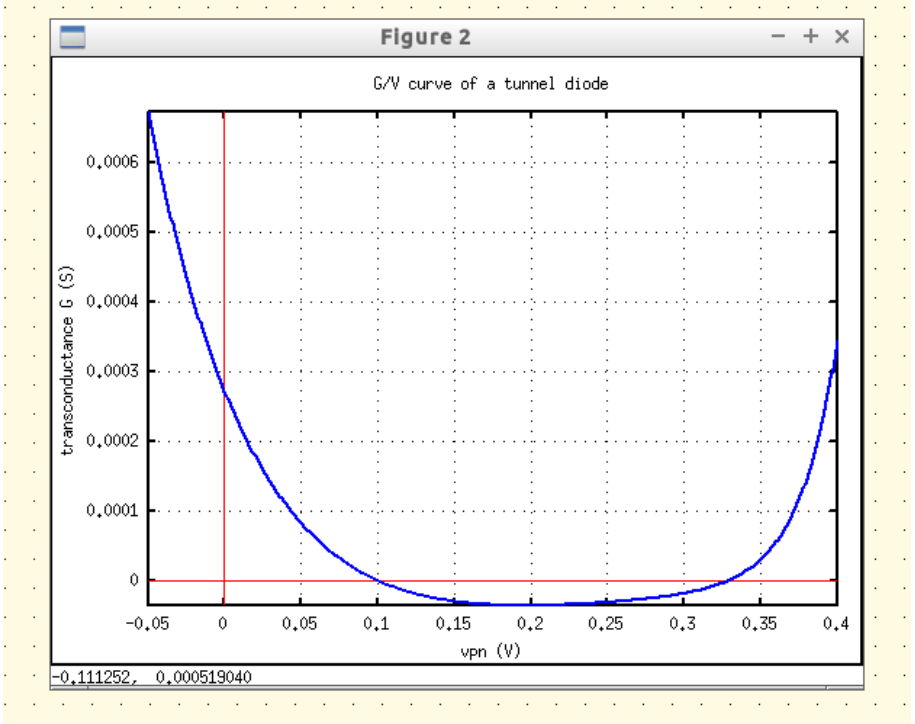
Tunnel diode I/V curve



MAPP post-simulation output

Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools
Part 5: Compact device modelling; the tunnel diode model continued

```
function out = plotGV_tunnelDiode_ModSpec_wrapper()
   MOD = tunnelDiode_ModSpec_wrapper();

   vs = -0.05:0.001:0.4;
   gs = zeros(size(vs));

   for idx = 1:1:size(gs,2)
      gs(1,idx) = MOD.dfe_dvecX(vs(idx), [], [], [], MOD);
   end

   figure();

   plot([min(vs), max(vs)], [0, 0], 'Color', 'red', 'LineWidth', …
      1.25, 'LineStyle', '--');
   hold on;
   plot([0, 0], [min(gs), max(gs)], 'Color', 'red', 'LineWidth', …
     1.25, 'LineStyle', '--');
   h = plot(vs, gs, 'Color', 'blue', 'LineWidth', 1.75);

         axis tight;
         box on;
         grid on;
         set(gca,'FontName','Times New Roman','FontSize',...
           15,'FontWeight','bold');
         xlabel('vpn (V)','FontName','Times New Roman','FontSize',...
           18,'FontWeight','bold');
         ylabel('transconductance G (S)','FontName','Times New Roman',...
           'FontSize',18,'FontWeight','bold');
         title(['G/V curve of a tunnel diode'],'FontName','Times New Roman', …
        'FontSize',18,'FontWeight','bold');
        set(gcf,'color','white');

end
```



Tunnel diode G/V curve

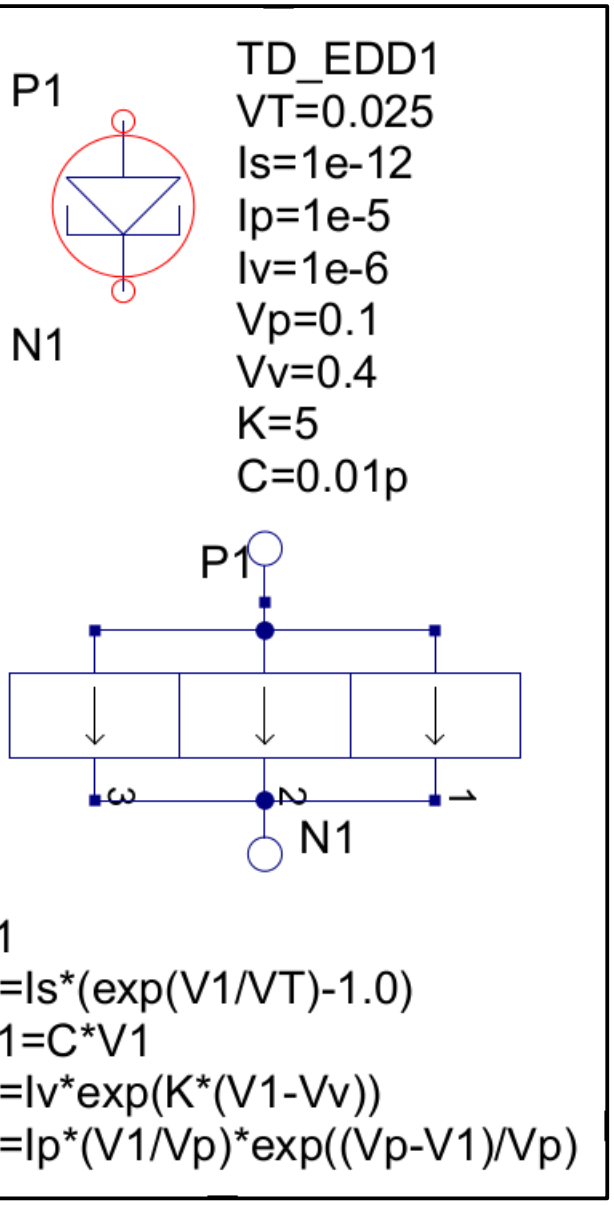
Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools
Part 6: Tunnel diode EDD model and test circuit

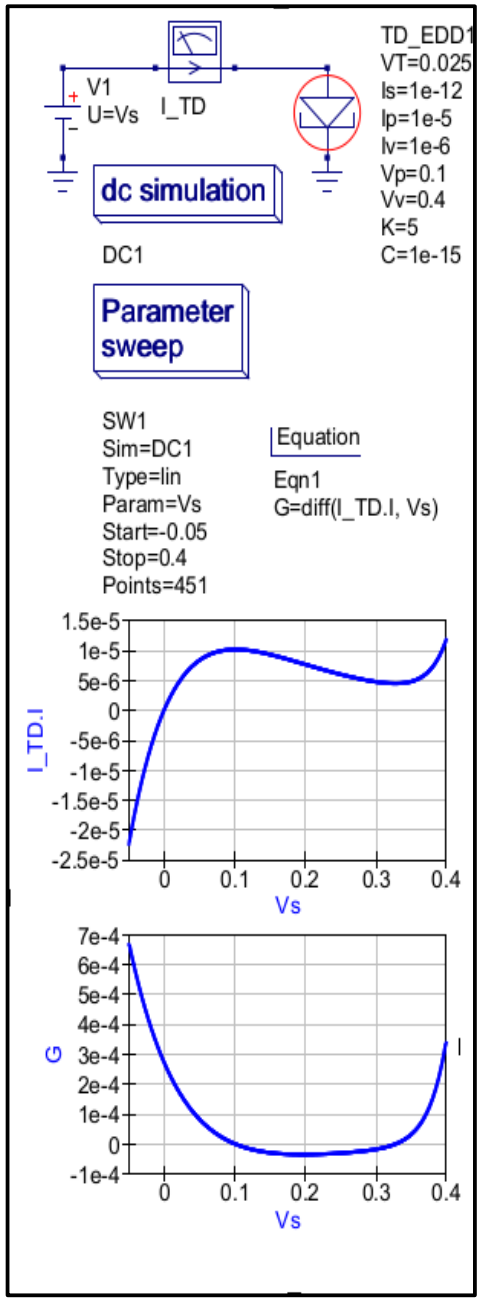
Equation

Model_parameters
Vv=0.4
VT=0.025
Ip=1e-5
Vp=0.1
Iv=1e-6
K=5
Is=1e-12

Equation

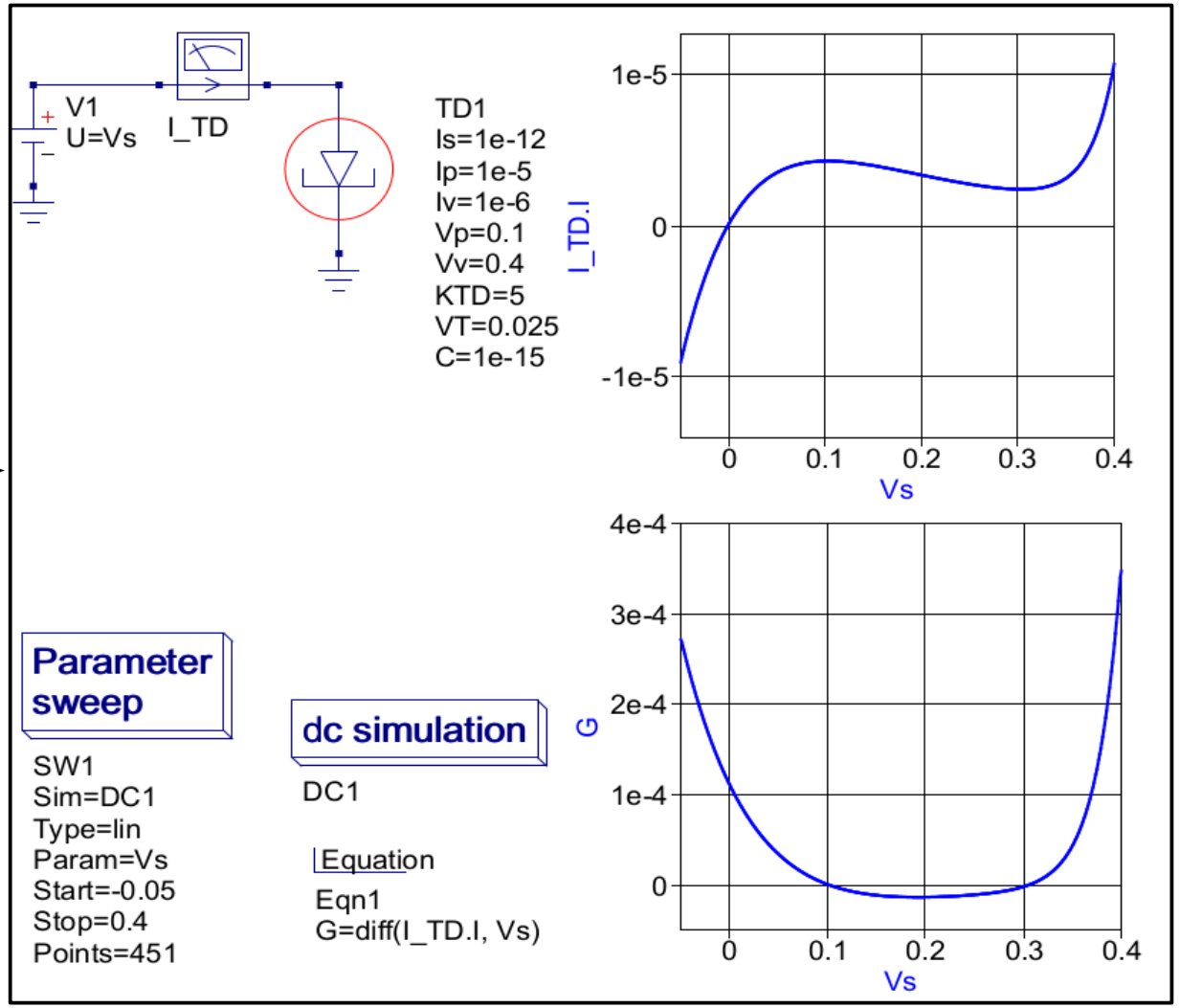
Model_equations
Idiode=Is*(exp(Vpn/VT)-1.0)
Iexcess=Iv*exp(K*(Vpn-Vv))
Ipn=Idiode+Iexcess+Itunnel
Itunnel=Ip*(Vpn/Vp)*exp((Vp-Vpn)/Vp)





Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools
Part 7: Tunnel diode Verilog-A model and test circuit

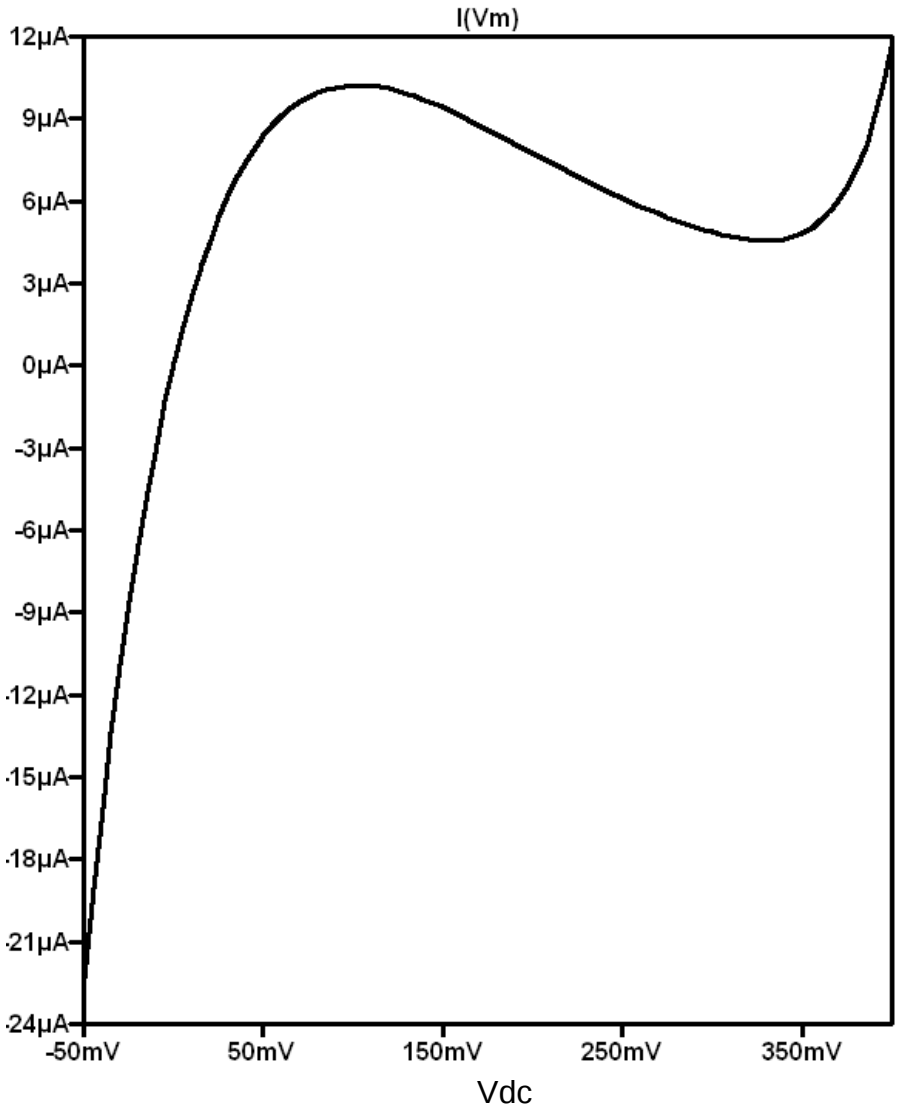
```
// Tunnel diode Verilog-A compact device model.
// Verilog-A code translated, by hand, from
// MAPP model.
//
`include "disciplines.vams"
`include "constants.vams"
//
module TD(p,n);
inout p,n;
electrical p,n;
//
parameter Is=1e-12 from [1e-20 : inf);
parameter Ip=1e-5  from [1e-20 : inf);
parameter Iv=1e-6  from [1e-20 : inf);
parameter Vp=0.1   from [1e-20 : inf);
parameter Vv=0.4   from [1e-20 : inf);
parameter KTD = 5    from [1e-20 : inf);
parameter VT=0.025 from [1e-20 : inf);
parameter C=1e-15  from [1e-20 : inf);
//
real Idiode, Iexcess, Itunnel;
//
analog begin
 Idiode = Is*(exp(V(p,n)/VT)-1.0);
 Iexcess = Iv*exp(KTD*(V(p,n)-Vv));
 Itunnel = Ip*(V(p,n)/Vp)*exp(Vp-V(p,n)/Vp);
 I(p,n) <+ Idiode+Iexcess+Itunnel;
 I(p,n) <+ ddt(C*V(p,n));
end
endmodule
```



Compact device modelling with Qucs EDD models, SPICE B models, MAPP DAE models, and Qucs ADMS/Verilog-A “turn-key” tools Part 8: Tunnel diode SPICE B model and test circuit

Typical ngspice, Xyce and LTspice tunnel diode netlist constructed with a subcircuit and B type non-linear current sources

```
*  Test of TD_SPICE model
*
*
.subckt tdspice p1 n1 Vv=0.4 VT=0.025 Ip=1e-5 Vp=0.1 Iv=1e-6 K=5 Is=1e-12
Bdiode    p1 n1 I={Is}*(exp( (v(p1)-v(n1))/{VT}) - 1.0)
Bexcess   p1 n1 I={Iv}*exp( {K}*(v(p1)-v(n1)-{Vv} ) )
Btunnel   p1 n1 I={Ip}*(  (v(p1) -v(n1))/{Vp})*exp(  ({Vp}-(v(p1)-v(n1) ) )/{Vp})
Ctd       p1 n1 1e-15
.ends
*
Vdc 1 0  DC 0
Vm  1 2 DC 0
Xtd  2  0  tdspice  Vv=0.4  VT=0.025  Ip=1e-5  Vp=0.1  Iv=1e-6  K=5  Is=1e-12
*
.dc Vdc -0.05 0.4 5e-3
.plot DC I(Vm)
*
.end
```



The tunnel diode conductance = diff(I(Vm), Vdc).

SPICE equivalent of diff appears not to be implemented ?

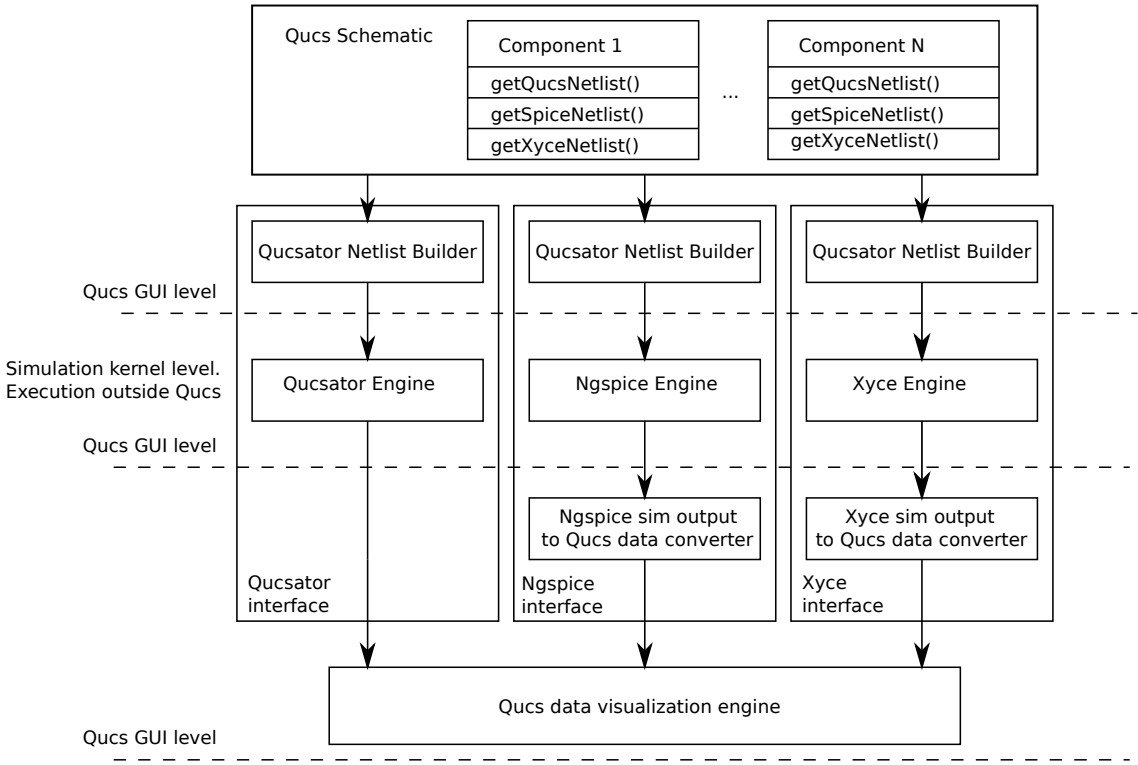
Linking ngspice and Xyce to Qucs: Part 1 - Background

Concept

The primary purpose of the proposed integration is (1) to provide a facility which allows Qucs schematics to be simlulated with GPL SPICE compatible simulation engines, in particular ngspice (http://ngspice.sourceforge.net) and Xyce (https://xyce.sandia.gov/), and (2) to check compact device model performance and accuracy across different simulators.

Current state of Work:

- Implemented Ngspice/Xyce netlist builder
- Implemented Ngspice/Xyce simulation output to Qucs XML-dataset converter
- Supported components:
  RCL components
  Nonlinear devices (Diode, BJT, MOSFET, JFET)
  AC, DC, pulse voltage sources
  Controlled sources
  Relay
- Supported simulations
  DC simulation
  Transient simulation
  AC simulation
  Harmonic Balance (Xyce)
- Qucs library components support (ngspice)



Source code available from https://github.com/ra3xdh/qucs/tree/spice4qucs

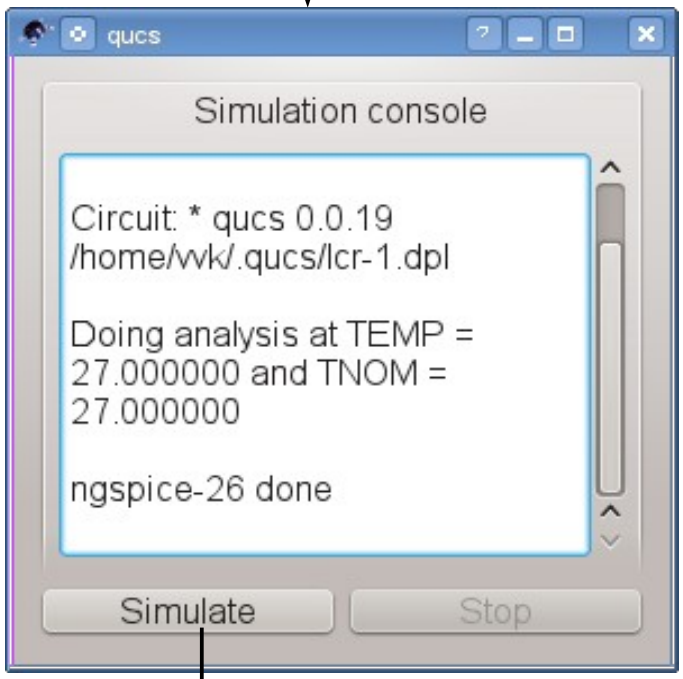
Linking ngspice and Xyce to Qucs: Part 2 - A simple RC circuit example

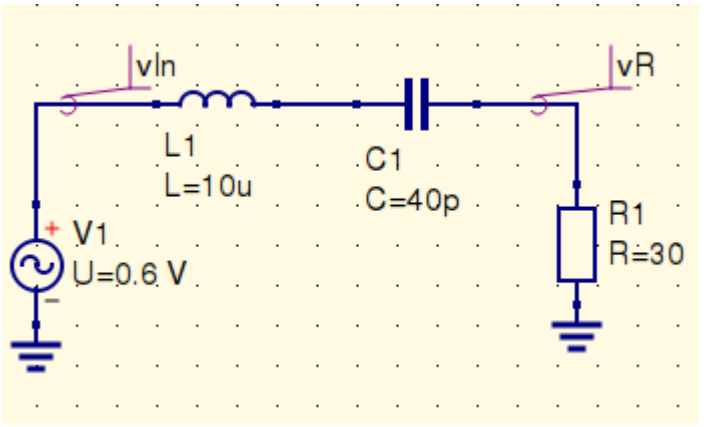
Qucs menu Simulate (F2)

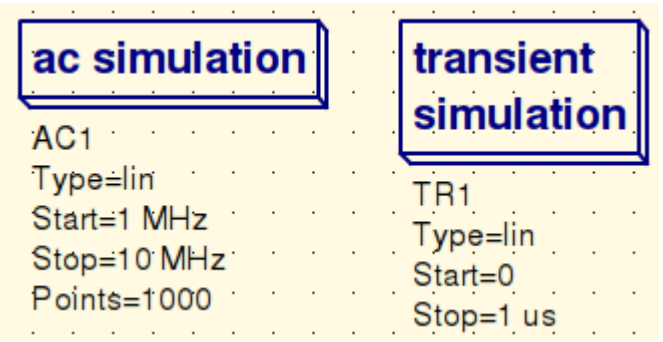
↓

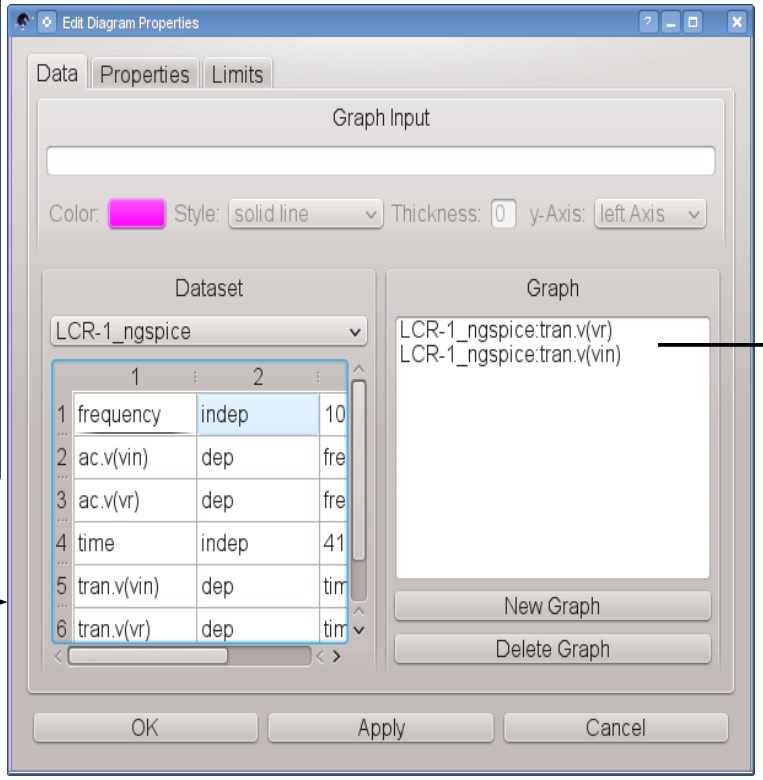
Simulate with SPICE
(ngspice or Xyce)

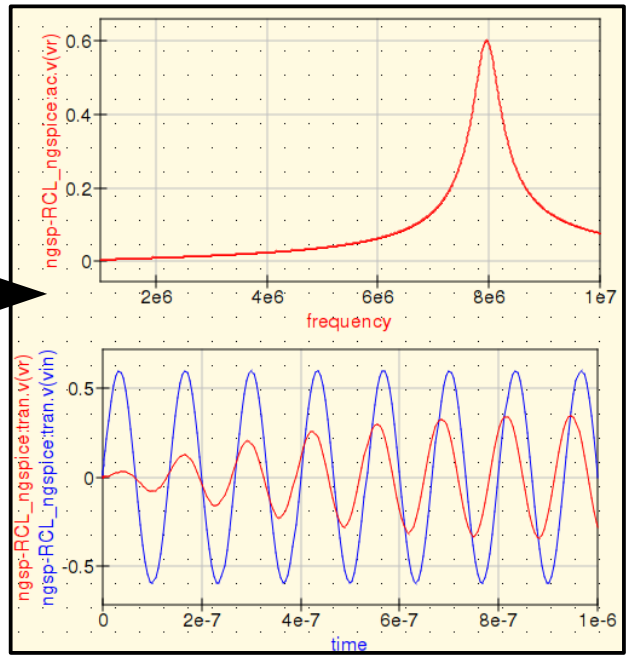
↓





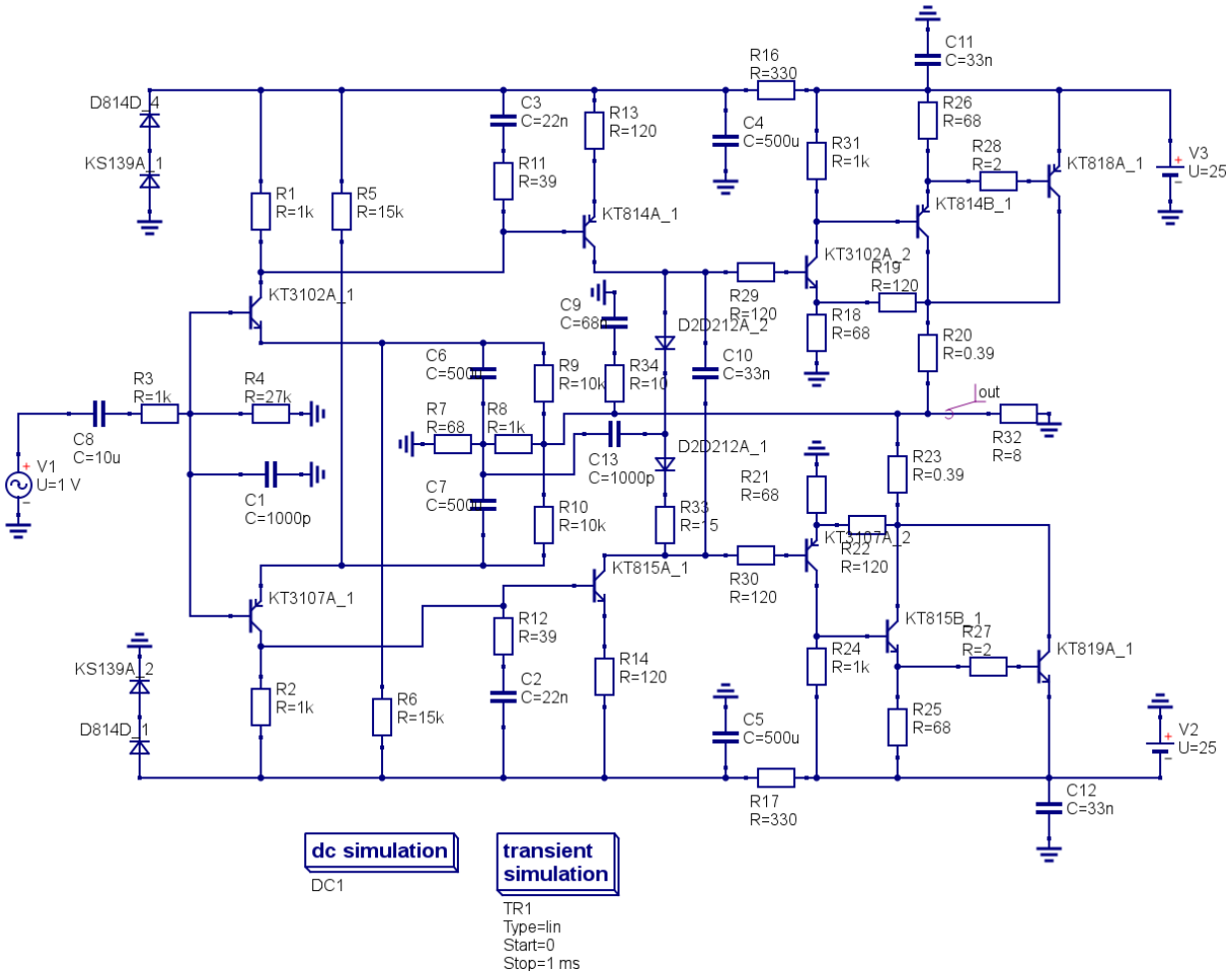




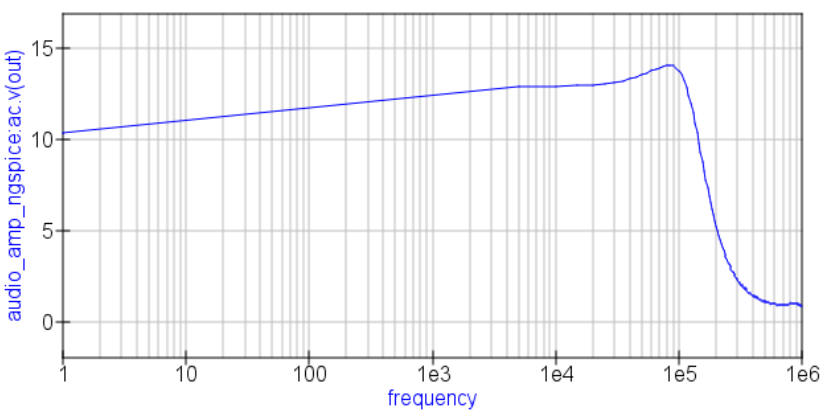


Linking ngspice and Xyce to Qucs: Part 3 - A larger circuit example

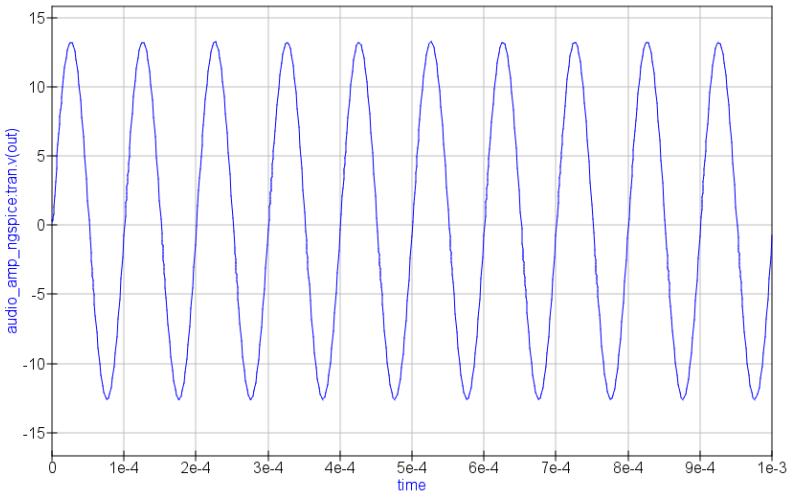
Audio amplifier schematic



AC gain against frequency



Transient output voltage against time

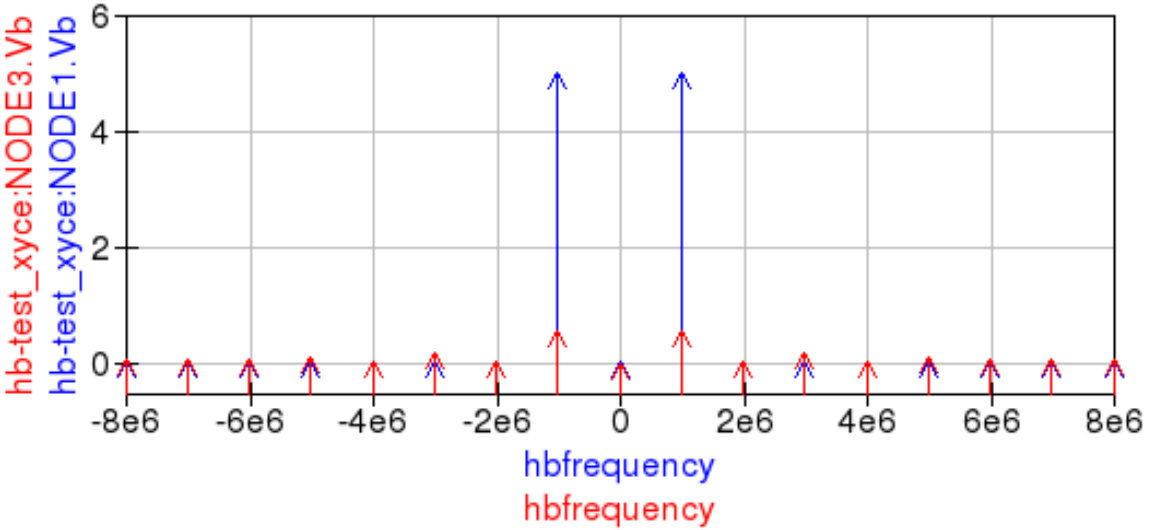


Ngspice and Xyce audio amplifier test circuit and typical simulation results

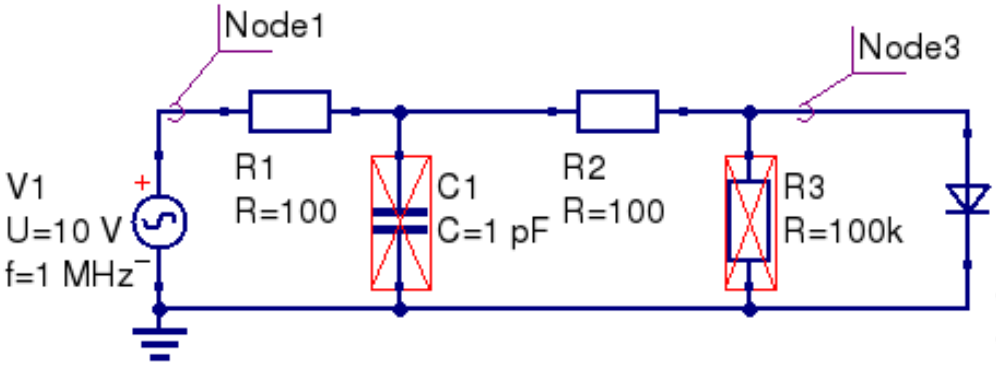
Linking ngspice and Xyce to Qucs: Part 4 – Comparing Qucs and Xyce Harmonic Balance simulation

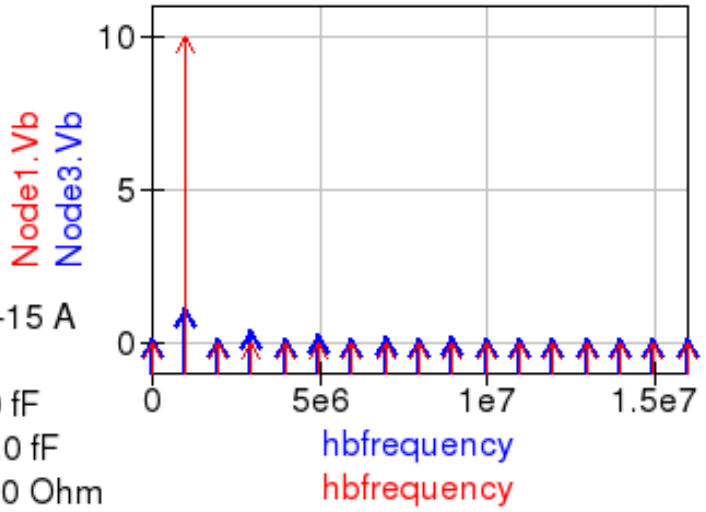
Harmonic balance simulation

HB1
f=1 MHz
n=17
iabstol=1 pA
vabstol=1 uV
reltol=0.001



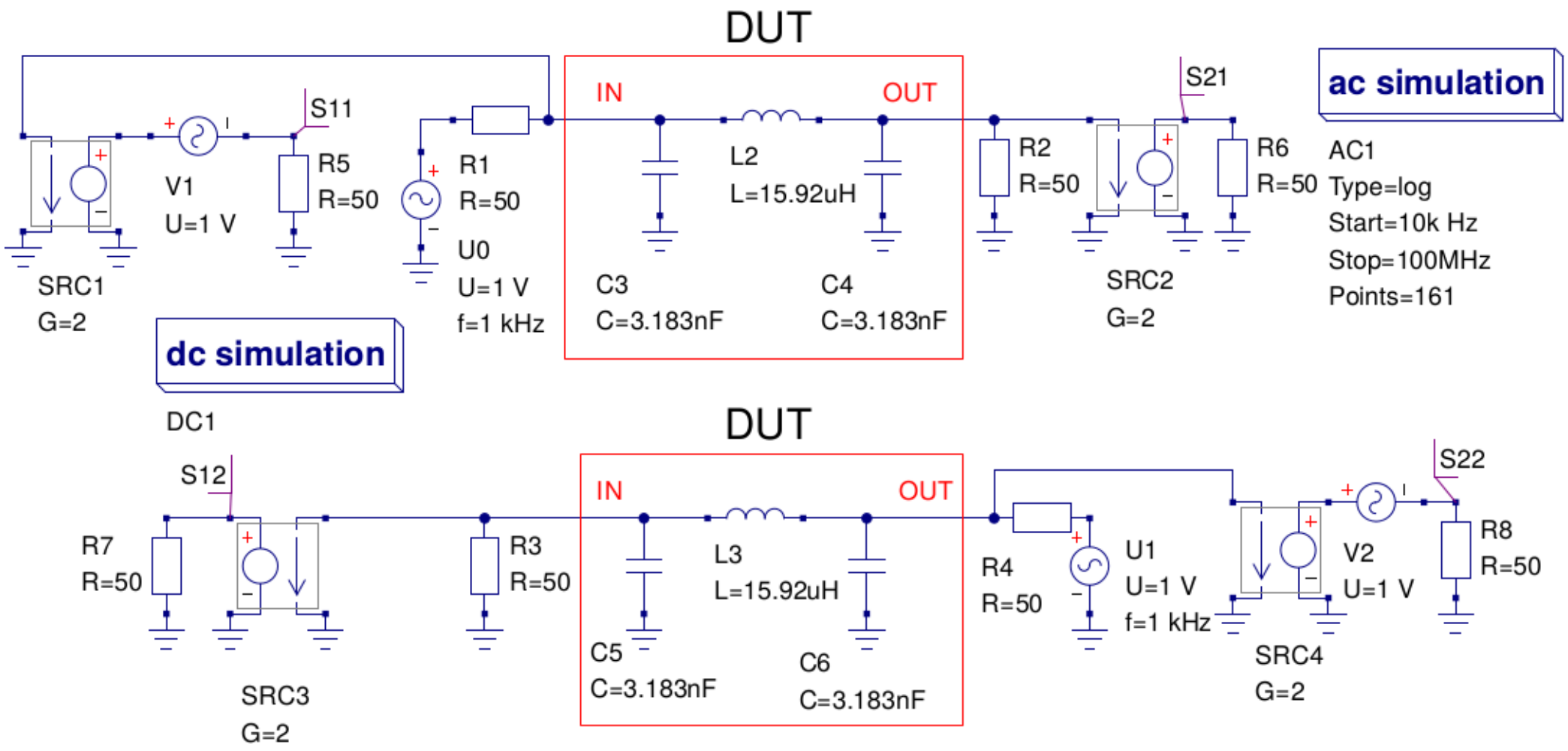
Xyce



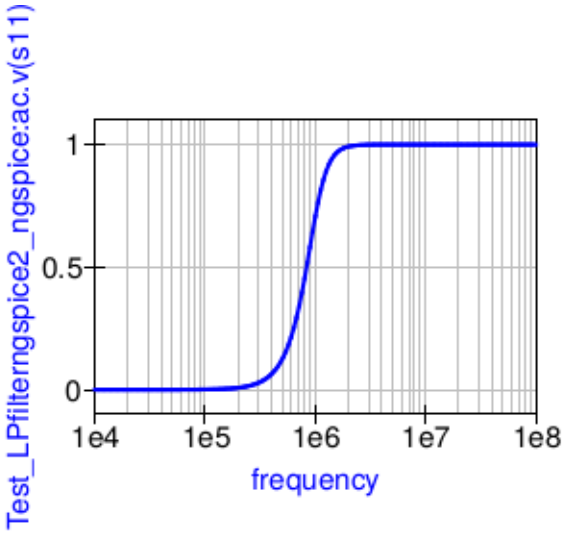


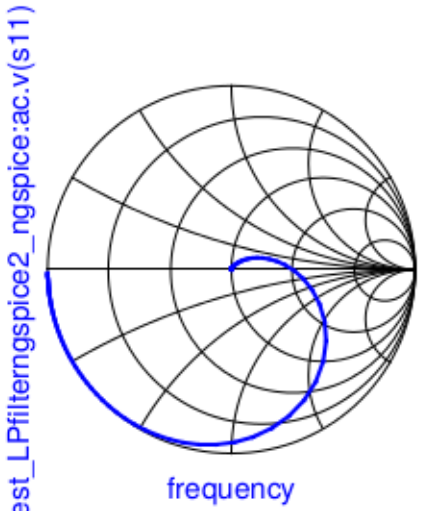
Qucs

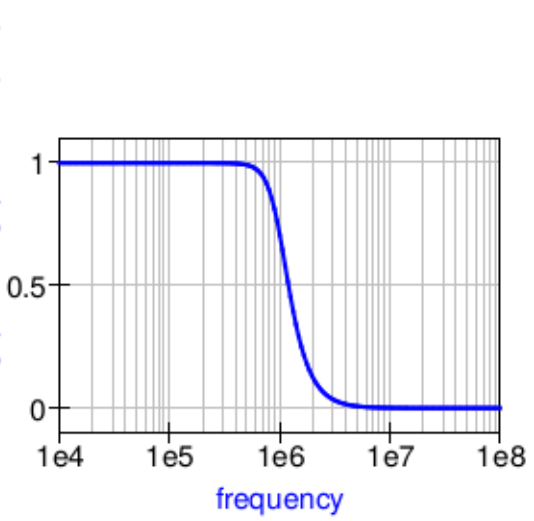
Linking ngspice and Xyce to Qucs: Part 4 – Small signal S parameter simulation with ngspice and Qucs

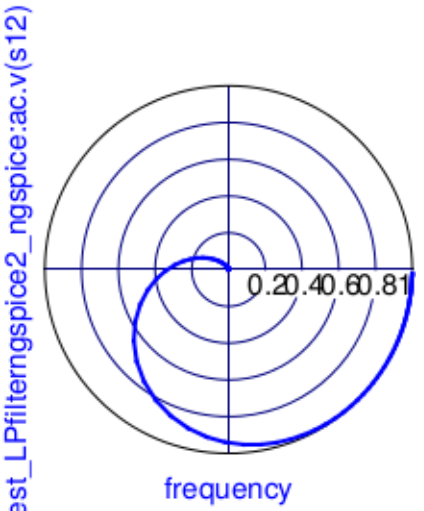


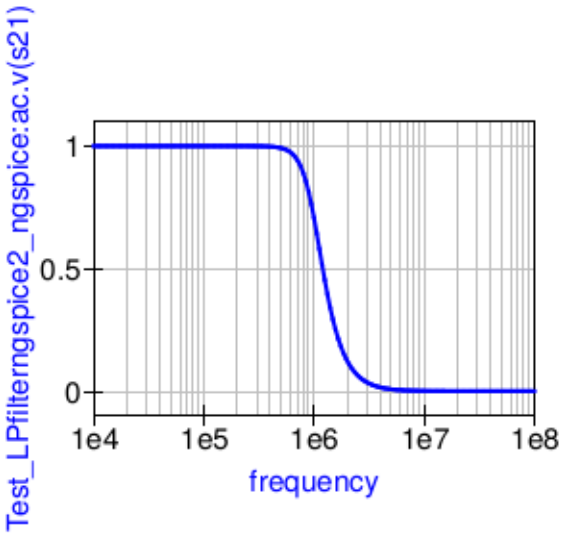
Linking ngspice and Xyce to Qucs: Part 4 – Small signal S parameter simulation with ngspice and Qucs continued

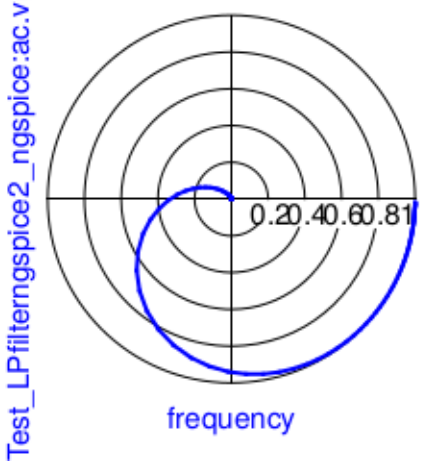


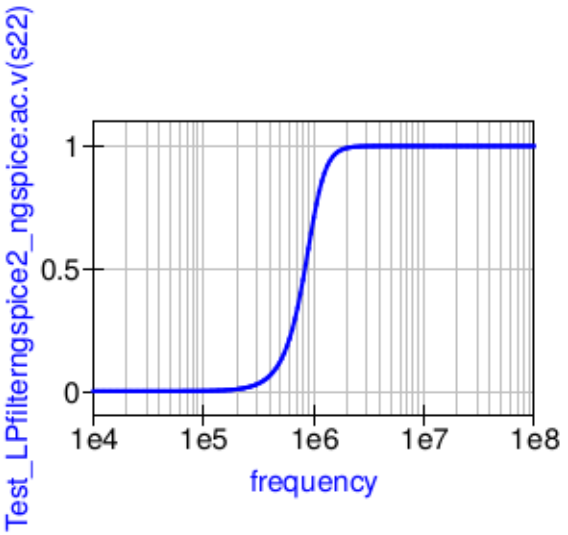


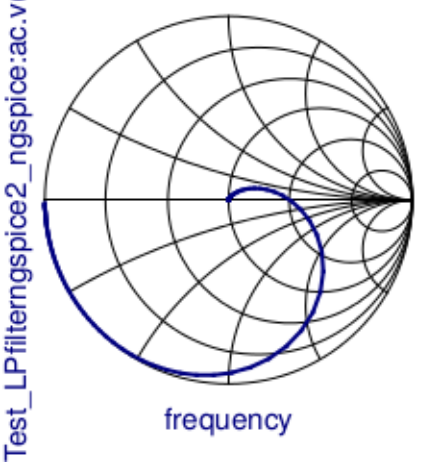








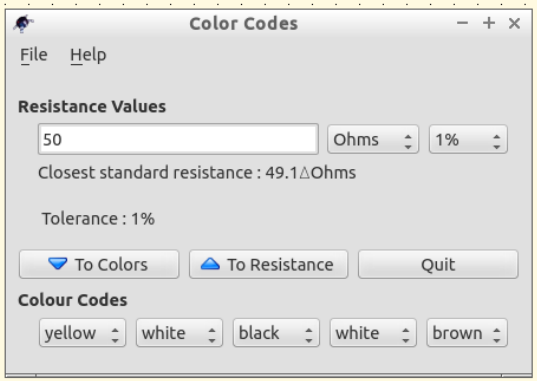


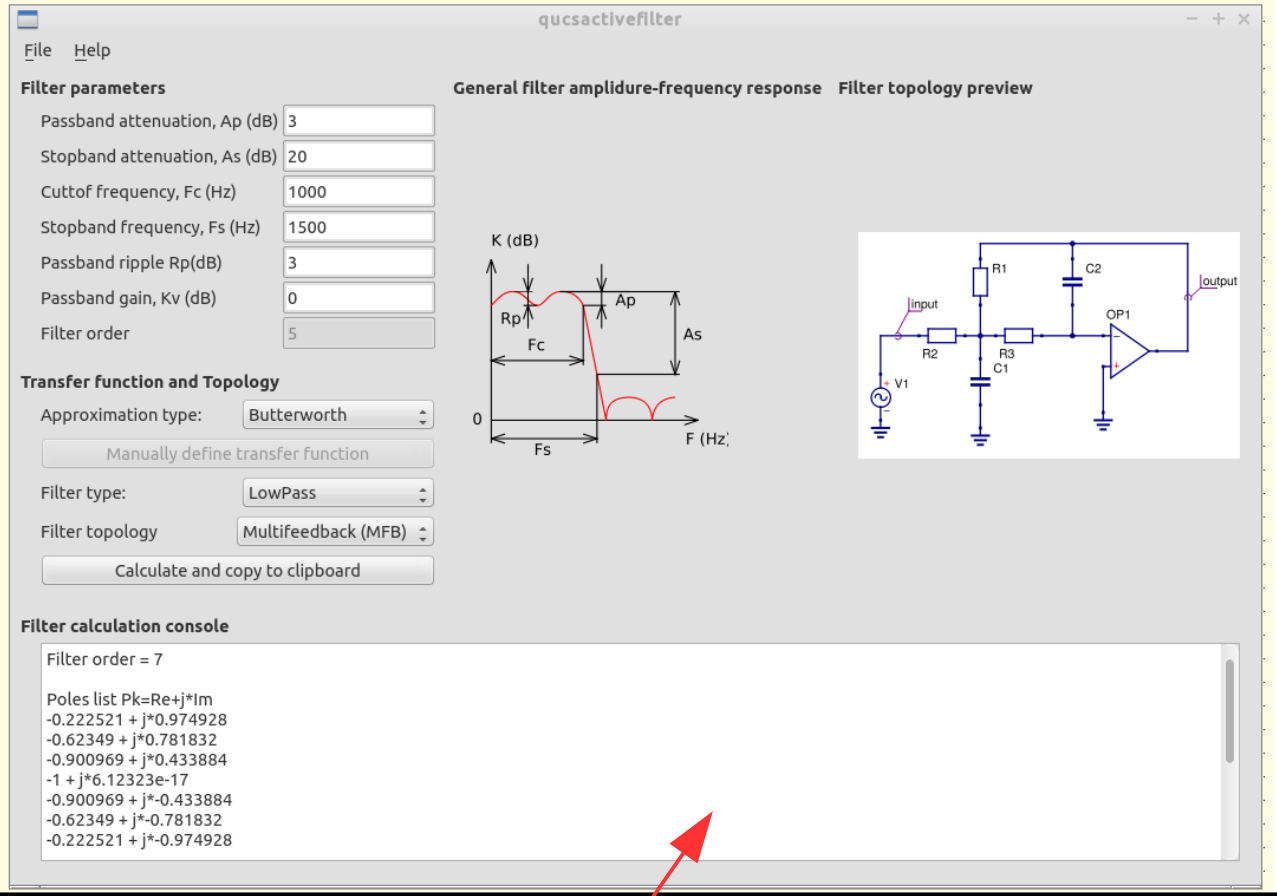


Qucs: new circuit design aids

Active filter synthesis

Resistor colour codes



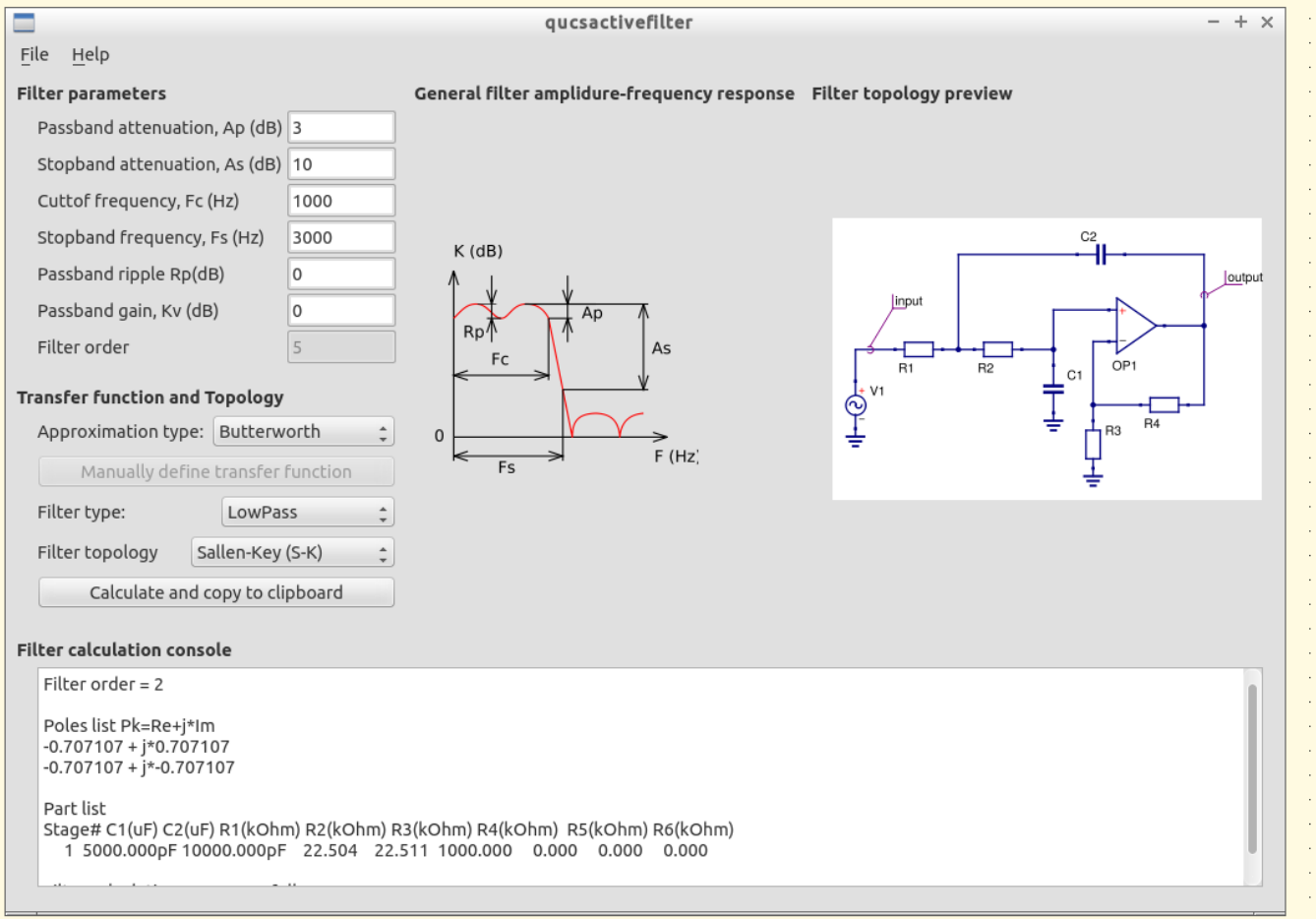


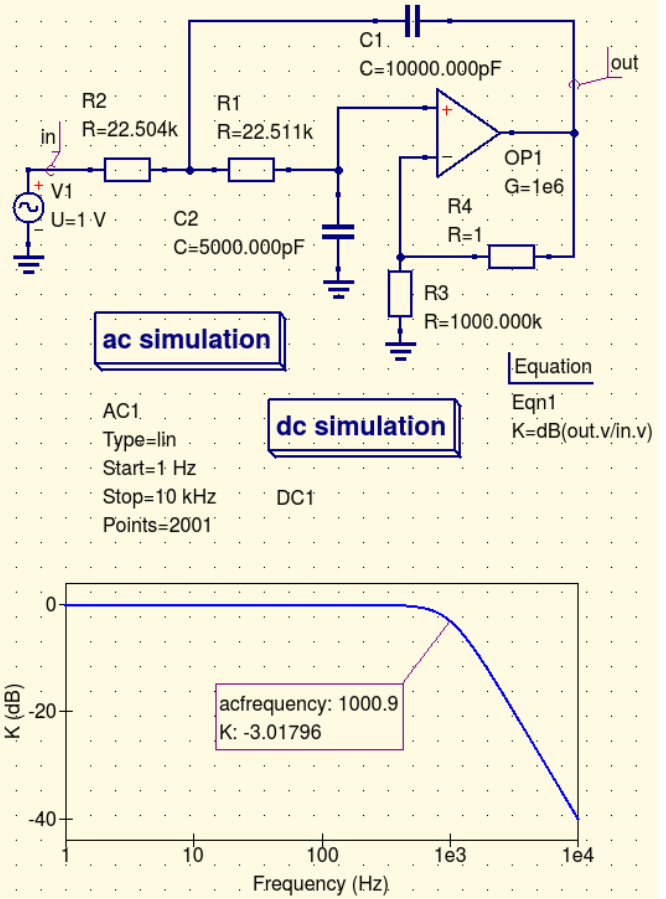
Main features

- Main window of the Qucs-active filter design tool.
- Butterworth, Chebyshev (Type I and II), Bessel, and Cauer low-pass, high-pass, band-pass, and band-stop active filter design capabilities.
- Sallen-Key, Multifeedback and Cauer filter section topologies are available.
- User-defined filter transfer functions.
- Full Qucs integration, including a copy-and-paste interface.
- To be released with Qucs-0.0.19.

Qucs: Actave filter design example

Butterworth Low Pass Active filter design





Qucs: Actave filter design algorithms - 1

Filter transfer functions

- For odd order filters without transfer function zeros (Butterworth, Chebyshev Type-I and Bessel):

$$H(s) = \frac{N(s)}{D(s)} = H_1(s) \prod_{i=0}^{N_2} H_2(s) = H_0 \frac{1}{s + C_N} \prod_{i=0}^{N_2} \frac{C_i}{s^2 + B_i s + C_i} \tag{1}$$

- For odd order filter with transfer function zeros (Cauer and Chebyshev-Type-II)

$$H(s) = H_1(s) \prod_{i=0}^{N_2} H_2(s) = H_0 \frac{1}{s + C_N} \prod_{i=0}^{N_2} \frac{s^2 + A_i}{s^2 + B_i s + C_i} \tag{2}$$

- For even order filter without transfer function zeros:

$$H(s) = \prod_{i=0}^{N_2} H_2(s) = H_0 \prod_{i=0}^{N_2} \frac{C_i}{s^2 + B_i s + C_i} \tag{3}$$

- For even order filter with transfer function zeros:

$$H(s) = \prod_{i=0}^{N_2} H_2(s) = H_0 \prod_{i=0}^{N_2} \frac{s^2 + A_i}{s^2 + B_i s + C_i} \tag{4}$$

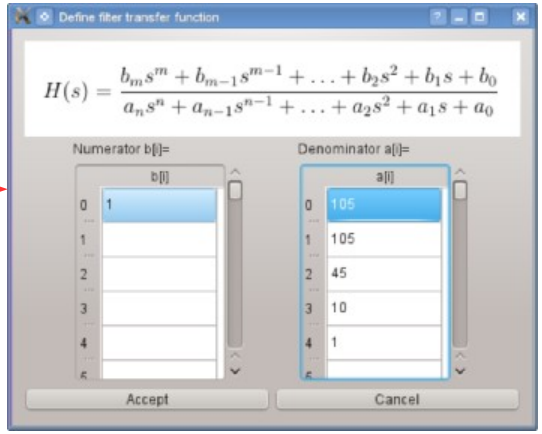
Qucs: Actave filter design algorithms - 2

- Input magnitude response parameters:
  - Cutoff frequency
  - Stopband frequency
  - Passband attenuation
  - Stopband attenuation
  - Passband ripple (for Chebyshev and Cauer filters only)
  - Passband gain
- User defined transfer function representation

$$H(s) = \frac{N(s)}{D(s)} = \frac{b_m s^m + \ldots + b_1 s + b_0}{a_n s^n + \ldots + a_1 s + a_0} \qquad (5)$$

Input parameters:

- Numerator coefficients b_m
- Denominator coefficients a_n



Transfer function definition window

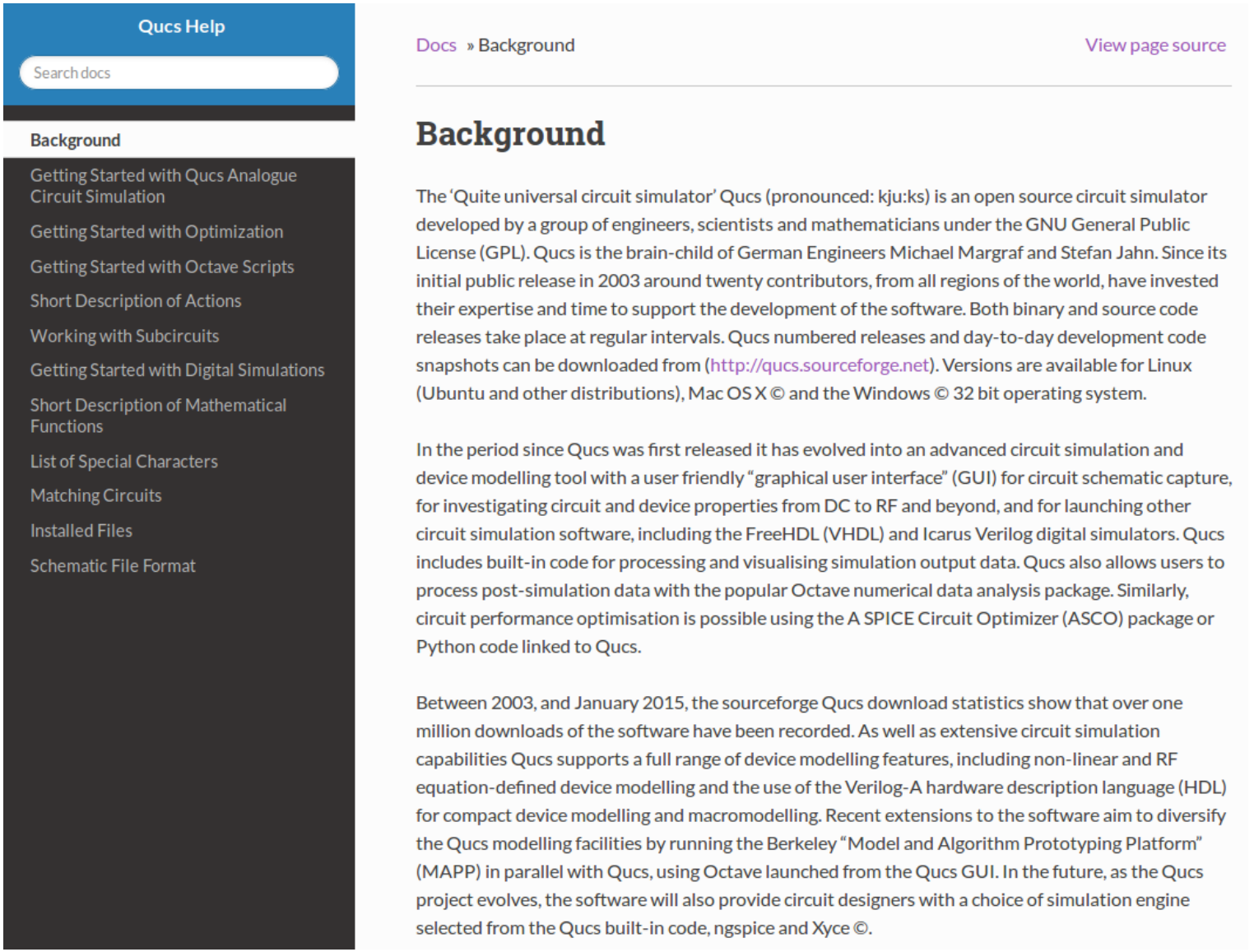
- Filter circuit building algorithm

Data: Filter Magnitude response parameters or transfer function numerator and denominator coefficients
Result: Active filter circuit in Qucs XML format in system clipboard

begin
  if *UserDefinedTransferFunction* **then**
    SolveEquiations $N(s) = 0$, $D(s) = 0$;
    Poles,Zeros ← FindPolesAndZeros();
  else
    SelectTransferFunctionApproximation();
    DetermineFilterOrder();
    Poles,Zeros ← CalculatePolesAndZeros();
  end
  SectionsCount ← EvaluateFilterSectionsCount();
end
for $i \leftarrow 1$ **to** *SectionsCount* **do**
  FindABCcoefficients(Poles[i],Zeros[i]);
  CalculateRCElementsValues();
end
BuildFilterCircuit();

Improvements to Qucs documentation and code control features - 1

1. New style HTML “Qucs-Help” documentation



Improvements to Qucs documentation and code control features - 2

2. New style HTML Qucs Tutorial/Report documents

Docs » Subcircuit and Verilog-A RF Circuit Models for Axial and Surface Mounted Resistors

View page source

Subcircuit and Verilog-A RF Circuit Models for Axial and Surface Mounted Resistors

Mike Brinson

Introduction

Resistors are one of the fundamental building blocks in electronic circuit design. In most instances conventional resistor circuit simulation models are characterized by I/V characteristics specified by Ohm's law. In reality the impedance of RF resistors is frequency dependent, being determined by component physical properties, component manufacturing technology and how components are connected in a circuit. At low frequencies fixed resistors have a nominal value at roomtemperature and can be modelled accurately by Ohm's law. At RF frequencies the fact that a resistor acts more like an inductance or a capacitance can play a crucial role in determining whether or not a circuit operates as designed. Similarly, if a resistor is modelled as an ideal component at a frequency where it exhibits significant reactive properties then the resulting simulation data are likely to be incorrect. The subcircuit and Verilog-A compact resistor models introduced in this Qucs note are designed to give good performance from low frequencies to RF frequencies not greater than a few GHz.

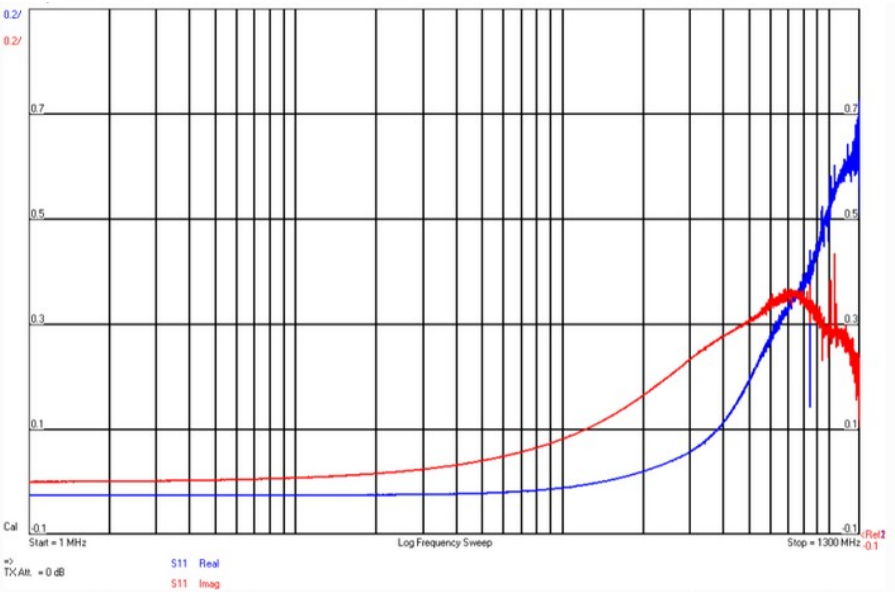


Figure 7 - DGSAQ Vector Network Analyser S parameter measurements for a 47 Ω axial RF resistor.

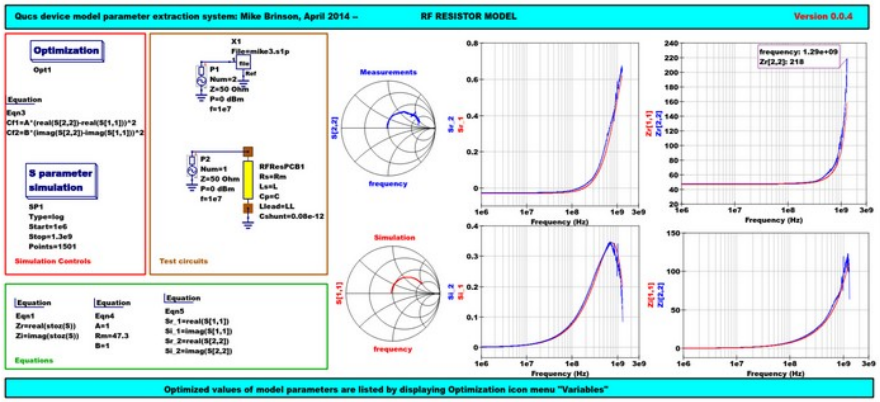
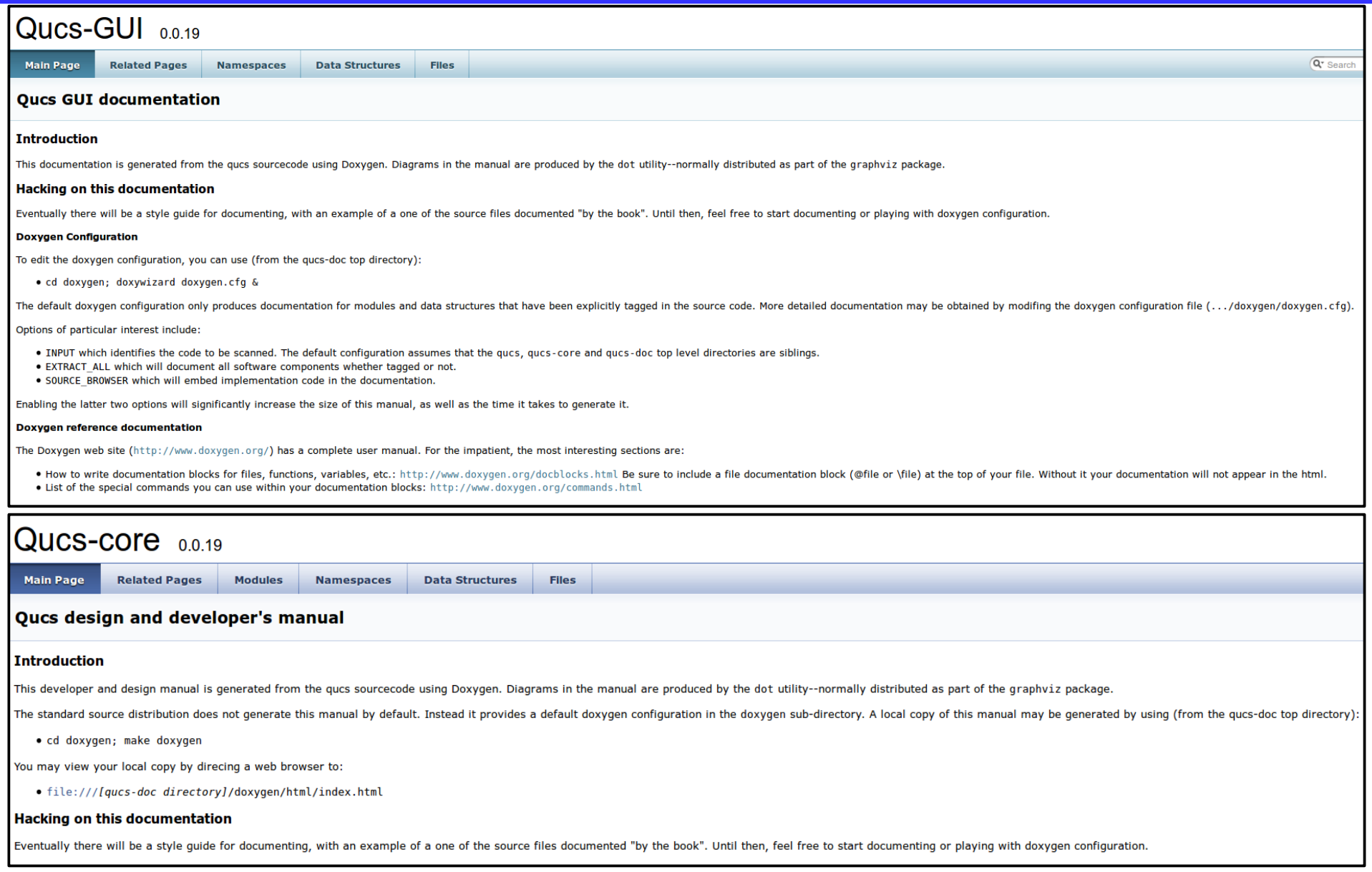


Figure 8 - Qucs device model parameter extraction system applied to a nominal 47 Ω resistor represented by the subcircuit model illustrated in Figure 2 (c). Fixed model parameter values: **Rs = Rm = 47.3 Ω, CShunt = 0.08pF**; Optimised values: **Ls = L = 10.43nH, Llead = LL = 1.47nH, Cp = C = 0.69pF**. To reduce simulation time the ASCO cost variance was set to 1e-3. The ASCO method was set to

Improvements to Qucs documentation and code control features - 3



Qucs-GUI 0.0.19

Main Page | Related Pages | Namespaces | Data Structures | Files | Search

Qucs GUI documentation

Introduction

This documentation is generated from the qucs sourcecode using Doxygen. Diagrams in the manual are produced by the dot utility--normally distributed as part of the graphviz package.

Hacking on this documentation

Eventually there will be a style guide for documenting, with an example of a one of the source files documented "by the book". Until then, feel free to start documenting or playing with doxygen configuration.

Doxygen Configuration

To edit the doxygen configuration, you can use (from the qucs-doc top directory):

- `cd doxygen; doxywizard doxygen.cfg &`

The default doxygen configuration only produces documentation for modules and data structures that have been explicitly tagged in the source code. More detailed documentation may be obtained by modifing the doxygen configuration file (`.../doxygen/doxygen.cfg`).

Options of particular interest include:

- `INPUT` which identifies the code to be scanned. The default configuration assumes that the `qucs`, `qucs-core` and `qucs-doc` top level directories are siblings.
- `EXTRACT_ALL` which will document all software components whether tagged or not.
- `SOURCE_BROWSER` which will embed implementation code in the documentation.

Enabling the latter two options will significantly increase the size of this manual, as well as the time it takes to generate it.

Doxygen reference documentation

The Doxygen web site (http://www.doxygen.org/) has a complete user manual. For the impatient, the most interesting sections are:

- How to write documentation blocks for files, functions, variables, etc.: http://www.doxygen.org/docblocks.html Be sure to include a file documentation block (@file or \file) at the top of your file. Without it your documentation will not appear in the html.
- List of the special commands you can use within your documentation blocks: http://www.doxygen.org/commands.html

Qucs-core 0.0.19

Main Page | Related Pages | Modules | Namespaces | Data Structures | Files

Qucs design and developer's manual

Introduction

This developer and design manual is generated from the qucs sourcecode using Doxygen. Diagrams in the manual are produced by the dot utility--normally distributed as part of the graphviz package.

The standard source distribution does not generate this manual by default. Instead it provides a default doxygen configuration in the doxygen sub-directory. A local copy of this manual may be generated by using (from the qucs-doc top directory):

- `cd doxygen; make doxygen`

You may view your local copy by direcing a web browser to:

- `file:///[qucs-doc directory]/doxygen/html/index.html`

Hacking on this documentation

Eventually there will be a style guide for documenting, with an example of a one of the source files documented "by the book". Until then, feel free to start documenting or playing with doxygen configuration.

3. Source code documentation generated with Doxygen

Improvements to Qucs documentation and code control features - 4

Concept

With each release of Qucs the program code becomes more complex and the number of built in device models increases. In an attempt to check that new device models and code changes do not, inadvertently, introduce bugs or simulation errors a set of software tests are under development. Eventually, these tests will exercise all Qucs passive component and active device models across relevant simulation domains.

Current state of work:

- Tests operation of Qucs GUI (qucs) and simulator engine (qucsator)
- Test process uses Python, testing Qucs projects held in a “testsuit” directory ($python run.py --prefix /home/user/local/qucs-master/bin/ --skip skip.txt --qucsator)
- Projects which are known to fail can be skipped
- Test options:
- `--qucs` runs the schematic to netlist conversion.
- `--qucsator` runs the simulator.
- `--add-test [schematic].sch` adds a schematic as a test-project into the test-set.
- `--skip [file]` will skip the projects listed in the [file].
- `--project [project directory from testset]` will run a single test from the set.
- `--compare-qucsator [prefix/one prefix/two prefix/three]` runs multipe qucsator simulators.
- `--verbose [0|1]` increase verbosity: 0 = progress and errors, 1 = all info.

Typical Qucs simulation report for passed tests

Project	Schem. Version	Sim. Runtime
AC_SW_resonance_prj	0.0.4	0.016968
AC_SW_swr_meter_prj	0.0.4	0.017098
AC_bandpass_prj	0.0.17	0.115111
AC_groupdelay_ac_prj	0.0.12	0.034698
DC_AC_SP_stab_prj	0.0.4	0.067629
DC_AC_active_bp_prj	0.0.5	0.016422
DC_AC_active_lp_prj	0.0.5	0.017316
DC_AC_bbv_prj	0.0.5	0.032346
DC_AC_gain_prj	0.0.5	0.031954
DC_AC_gyrator_prj	0.0.4	0.008243
DC_AC_notch_prj	0.0.5	0.016785
DC_AC_selective_amp_prj	0.0.5	0.017569
DC_SP_LPF-Balun2_prj	0.0.15	0.216681
DC_SP_LPF-Balun3_prj	0.0.15	0.165655
DC_SP_opamp_gyrator_prj	0.0.5	0.018379
DC_SW_bridge_prj	0.0.3	0.034325
DC_SW_bsim4v30nMOS_Ids_Vgs_prj	0.0.18	0.217675
DC_SW_bsim4v30pMOS_Ids_Vgs_prj	0.0.18	0.167487
DC_SW_charac_prj	0.0.4	0.117094
DC_SW_diff1_prj	0.0.5	0.167375
DC_SW_diode_prj	0.0.18	0.421467
DC_SW_fgummel_prj	0.0.10	0.034314
DC_SW_preregulator_prj	0.0.15	0.033987
DC_SW_rgummel_prj	0.0.10	0.065789
DC_TR_SW_spice_BFR520_prj	0.0.18	0.218216
DC_TR_active_mixer_prj	0.0.4	0.181998
SP_SmithChartTest_prj	0.0.12	0.065689
SP_Sparam_diagrams_prj	0.0.18	0.066238
SP_bpf_10Ghz_prj	0.0.4	0.065355
SP_chebyshev1_5th_prj	0.0.3	0.034585
SP_elliptic_5th_prj	0.0.3	0.065700
SP_fet_noise_prj	0.0.5	0.034303

Possible directions for Qucs development after release 0.0.19

Summary and possible future directions

- Qucs 0.0.19 is likely to be a major release of the circuit simulator package with many of the features introduced in this presentation included. At this stage in the Qucs development cycle it is difficult to say what the final structure of the software will be, or indeed how complete the new features will be. However, as a minimum Qucs 0.0.19 will have benefited from the significant amount of work done by the Development Team to remove bugs, restructure the software, port the GUI from Qt3 to Qt4, improve the performance of qucsator, add new circuit design and modelling features and make the Qucs GUI more user friendly and productive.

MUCH WORK STILL NEEDS TO BE DONE BEFORE FURTHER NEW FEATURES ARE ADDED TO THE SOFTWARE CODE.

- Medium to long term possible improvements to Qucs have been published in the
- revised Qucs Wiki roadmap. This can be found at

https://github.com/Qucs/qucs/wiki/Roadmap

Stable and development versions of Qucs and MAPP can be downloaded from:
1. Qucs : http://qucs.sourceforge.net/ and https://github.com/Qucs/qucs/
2. MAPP : http://draco.eecs.berkeley.edu/dracotiki/tiki-index.php?page=MAPPgplDownloads