

Qucs: An introduction to the new simulation and compact device modelling features implemented in release 0.0.19/0.0.19S of the popular GPL circuit simulator

Mike Brinson ¹, mbrin72043@yahoo.co.uk.

Richard Crozier ², richard.crozier@yahoo.co.uk

Vadim Kuznetsov ³, ra3xdh@gmail.com

Clemens Novak ⁴, clemens@familie-novak.net

Bastien Roucaries ⁵, bastien.roucaries@satie.ens-cauchan.fr

Frans Schreuder ⁶, fransschreuder@gmail.com

Guilherme Brondani Torri ⁴, guitarri@gmail.com

¹Centre for Communications Technology, London Metropolitan University, UK

²The University of Edinburgh, UK

³Bauman Moscow Technical University, Russia

⁴Qucs Developer

⁵Laboratoire SATIE — CNRS UMR 8929, Université de Cergy-Pontoise, ENS Cachan, FR

⁶Nikhef, Amsterdam, NL

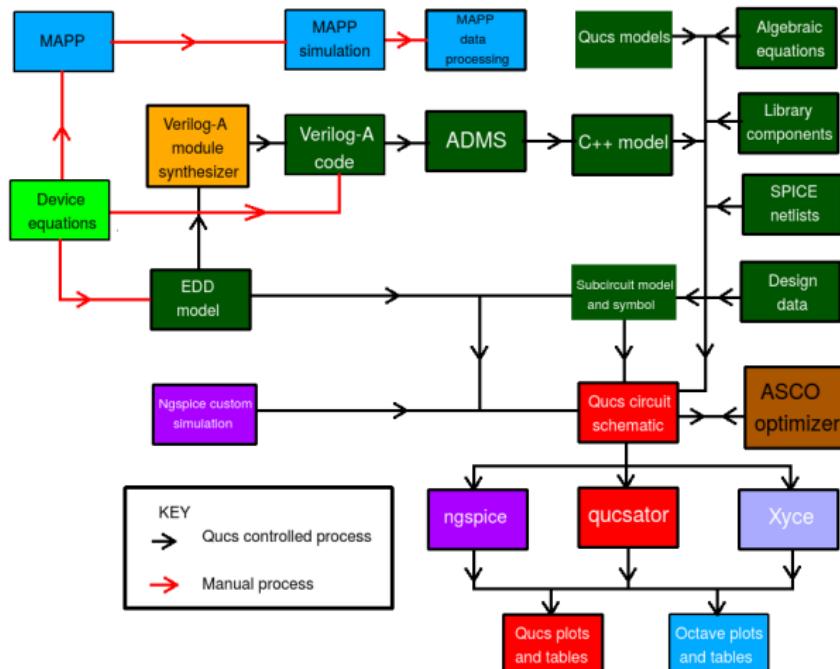


Qucs: An introduction to the new simulation and compact device modelling features implemented in release 0.0.19/0.0.19S of the popular GPL circuit simulator

- Qucs-0.0.19/S structure: overview, spice4qucs initiative tasks and main features
- Ngspice and Xyce applications: legacy Qucs circuit simulation, larger analogue circuits, power electronics and qucs2spice netlist converter
- Compact modelling with Qucs, ngspice, and Xyce
 - EDD support: Current and charge equations
 - XSPICE macromodel support: capacitance probes
 - B-type SPICE sources
 - Harmonic balance simulation with Xyce and Qucs compact models
- New components implemented by spice4qucs
 - Behavioural, modulated and noise sources: B-type, PWL, AM, SFFM and time domain noise
 - Transmission lines: TLINE, LTRA and UDRCTL
 - Full SPICE specification for semiconductor Diode, BJT, JFET, MOSFET and MESFET models
- Parametrization features and ngnutmeg scripting introduced with spice4qucs
- New simulation types implemented by spice4qucs: .FOUR, .NOISE, .DISTO and ngspice “Custom simulation”
- New tools for active and passive filter synthesis
- Introduction to the Qucs subcircuit to Verilog-A module synthesizer
- Plans for future



Qucs-0.0.19/S structure diagram for simulation and compact device modelling



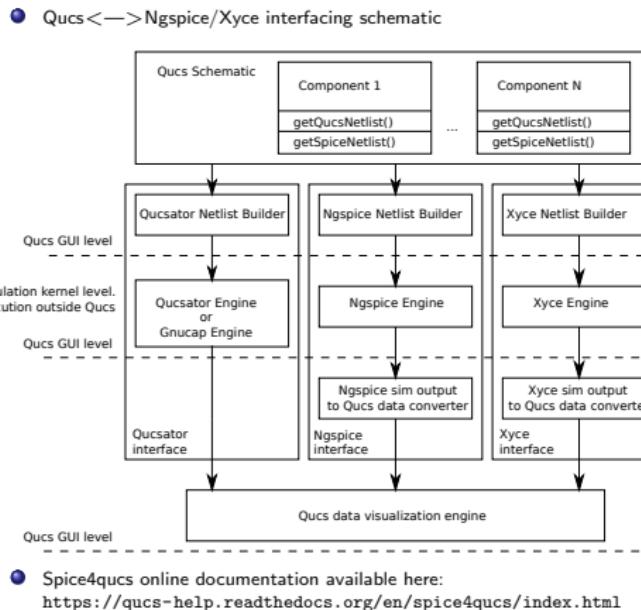
Overview of spice4qucs structure: Part I – spice4qucs initiative tasks

Spice4qucs initiative tasks:

- Correct known weaknesses observed with the current Qucs simulation engine qucsator
- Provide Qucs users with a choice of simulator selected from qucsator, ngspice and Xyce
- Extend Qucs subcircuit, EDD, RFEDD and Verilog-A device modelling capabilities
- Access to the additional simulation tools and extra component and device models provided by ngspice and Xyce
- Mixed-mode analogue-digital circuit simulation capability using Qucs/ngspice/XSPICE simulation

Currently implemented in Qucs-0.0.19/S:

- Ngspice, Xyce (both serial and parallel) support
- Basic simulations support (.DC, .AC, .TRAN)
- Advanced simulation support (.FOUR, .DISTO, .NOISE, .HB)
- Semiconductor devices with full SPICE specifications
- Qucs equations, parametrization (.PARAM), and ngnutmeg script support
- Custom ngspice simulation – User controlled simulation based on ngnutmeg scripts
- Qucs subcircuit to Verilog-A module synthesizer support

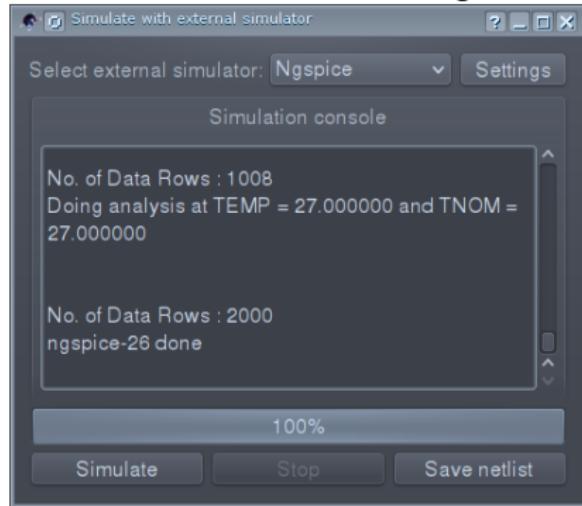


Overview of spice4qucs structure: Part II – New simulation features available with spice4qucs

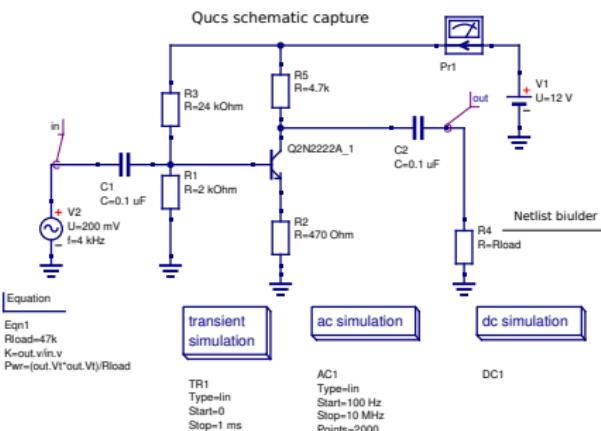
The list of supported simulations:

- Qucsator, ngspice, and Xyce;
 - DC sweep analysis
 - AC small signal analysis
 - Transient analysis
 - Single parameter sweep
- Qucsator and ngspice: Parameter sweep in nested loops
- Quescator and Xyce only; Harmonic balance (HB)
- Ngspice and Xyce: Fourier analysis
- Ngspice only:
 - Distortion analysis
 - Noise analysis
 - Custom simulation – ngnutmeg scripts embedded in Qucs schematics

New "SPICE simulation" dialogue:



Ngspice and Xyce simulation techniques: Part I – Legacy Qucs circuit simulation with ngspice and Xyce



Spice netlist

```

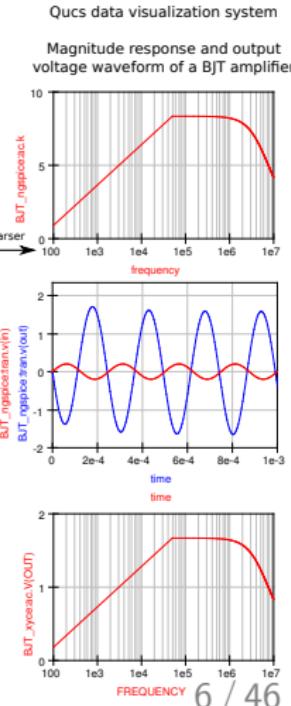
* Qucs 0.0.19 /home/vvk/.qucs/BJT.sch
.PARAM Rload=47k
.QN2222A_1 _net1 _net0 _net2
.MODEL QN2222A_1 ARRE=1 TEMP=26.85
.RBw=0.1k Rb=0.137 Re=0.343 Rd=1.37
.Cje=2.95e-11 Cj=0.225 Vaf=113 Var=24
.Ise=1.06e-11 Ne=2 Isc=0 Nc=2 Bf=205 Br=4
.Rbb=0.1k Rc=0.137 Re=0.343 Rd=1.37
.Cje=2.95e-11 Vje=0.75 Mje=0.33 Cjc=1.52e-11
.Vje=0.75 Mjc=0.33 Xjc=1 Cjs=0.75
.Mjj=0 Fc=0.5 Tf=3.97e-10 Xtf=0 Vtf=0 Itf=0
.Tr=8.5e-08 Kf=1 Af=1 Ptf=0 Xtb=1.5 Xti=3
.Eg=1.11 Tnom=26.85
R1 _net0 2K
R2 _net0 470
C1 _net0 0.1u
R3 _net0 _net3 24K
C2 _net1 out 0.1u
R5 _net1 _net3 4.7K
V2 0 DC 0 SIN(0 200M 4K 0 0) AC 200M
R4 0 out (RLOAD)
VPri _net4 _net3 0 AC 0
V1 _net4 0 DC 12
.control
set filetype=ascii
echo "> spice4qucs.cir.noise
let Rload=47k
TRAN 1e-06 0.001 0
let Pwr=(V(out)*V(out))/Rload
write BJT_tran.txt VPri#branch v(in) v(out) Pwr
destroy all
reset

AC LIN 2000 100 10MEG
let K=V(out)/V(in)
write BJT_ac.txt VPri#branch v(in) v(out) K
destroy all
reset

.exit
.endc
.END

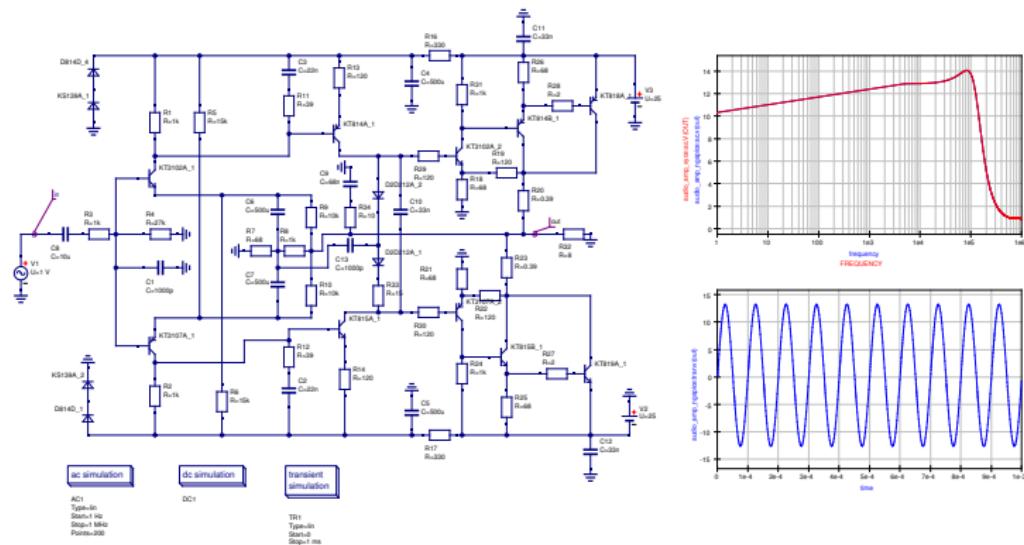
```

Netlist builder



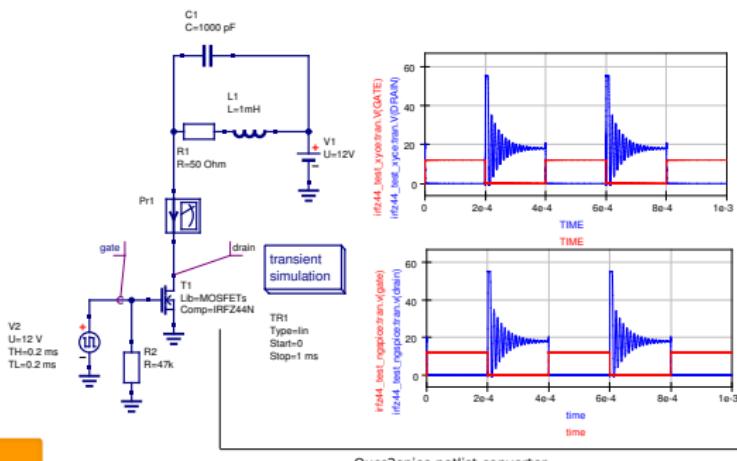
Ngspice and Xyce simulation techniques: Part II – Larger circuit simulation with ngspice and Xyce

This example illustrates an ngspice simulation of a larger analogue circuit: the BJT audio amplifier simulation data, for both the frequency and time domains, are given on the slide:



Ngspice and Xyce simulation techniques: Part III – A power electronics simulation example

- A MOSFET switch circuit with an inductive load is shown simulated by ngspice and Xyce: this example introduces a support feature for Qucs library components introduced with current implementation of spice4qucs.
- In this simulation a SPICE model of the power MOSFET is synthesized from a Qucs library model using a qucs2spice subsystem included with spice4qucs:



IRFZ44 MOSFET model converted from Qucs netlist

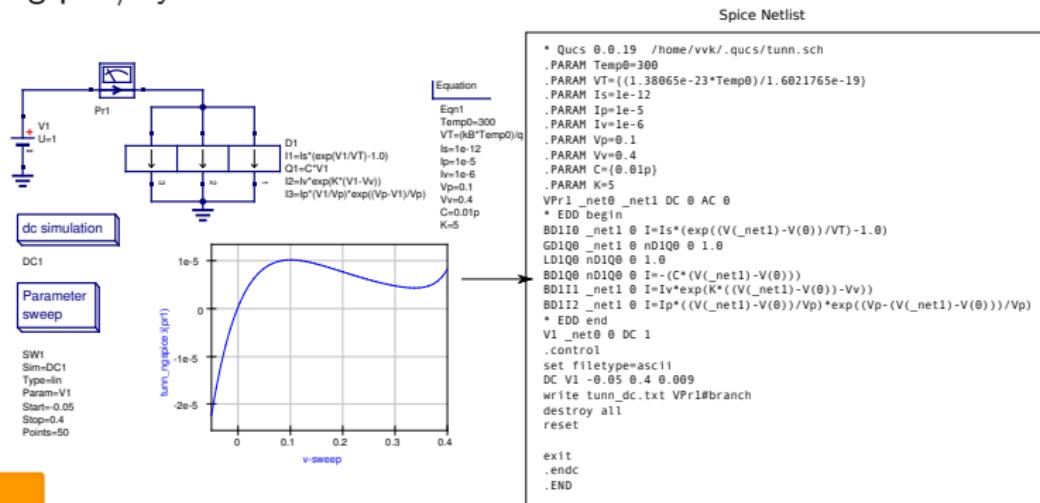
```
.SUBCKT MOSFETs_IRFZ44N gnd _net2 _net1 _net3
MM1 _net9 _net7 _net8 MMOD_M1 L=100u W=100u
.MODEL MMOD_M1 NMOS (Is=1e-32 VT0=3.56214 Lambda=0
+ Kp=39.3974 Cgs=1.25255e-05 Cgd=2.2826e-07
+ Rs=0 Rd=0 Ld=0 Cbs=0 Cgbo=0 Gamma=0 Phi=0.6
RRS _net8 _net3 0.0133305
DD1 _net3 _net1 DMOD_D1
.MODEL DMOD_D1 D(Is=9.64635e-13 Rs=0.00967689
+ N=1.01377 Bv=55 Ibv=0.00025 Eg=1.08658 Xti=2.9994
+ Tt=1e-07 Cjo=1.39353e-09 Vj=0.5 M=0.42532 Fc=0.5)
RRD5 _net3 _net1 2.2e+06
RRD _net9 _net1 0.0001
RRG _net12 _net2 2.20235
DD2 _net4 _net5 DMOD_D2
.MODEL DMOD_D2 D(Is=1e-32 N=50 Cjo=1.52875e-09 Vj=0.5
+ M=0.584414 Fc=1e-08 Rs=0 Eg=1.11 Xti=3 Tt=0 Bv=0
+ Ibv=1mA)
DD3 gnd _net5 DMOD_D3
.MODEL DMOD_D3 D(Is=1e-10 N=0.408752 Rs=3e-06
+ Eg=1.11 Xti=3 Tt=0 Cjo=0 Bv=0 Ibv=1mA M=0.5 Vj=0.7)
RRL _net5 _net11 1
FF12 _net7 _net9 FF12 -1
FF12 _net4 gnd DC 0
EEV16 _net10 gnd _net9 _net7 1
CCAP _net10 _net10 2.06741e-09
FF11 _net7 _net9 FF11 -1
FF11 _net11 _net6 DC 0
RRCAP _net6 _net10 1
DD4 gnd _net6 DMOD_D4
.MODEL DMOD_D4 D(Is=1e-10 N=0.408752 Eg=1.11 Xti=3
+ Tt=0 Cjo=0 Rs=0 Bv=0 Ibv=1mA M=0.5 Vj=0.7)
.ENDS
```

Compact modeling with Qucs and ngspice/Xyce: Part I – Current equation support

Consider tunnel diode model represented by

$$I = I_s \left(e^{\frac{V}{\varphi_T}} - 1 \right) + I_v e^{k(V - V_v)} + I_p \cdot \frac{V}{V_p} e^{\frac{V_p - V}{V_p}} \quad (1)$$

With spice4qucs, Qucs EDD charge components can be represented by B-type ngspice/Xyce current sources:



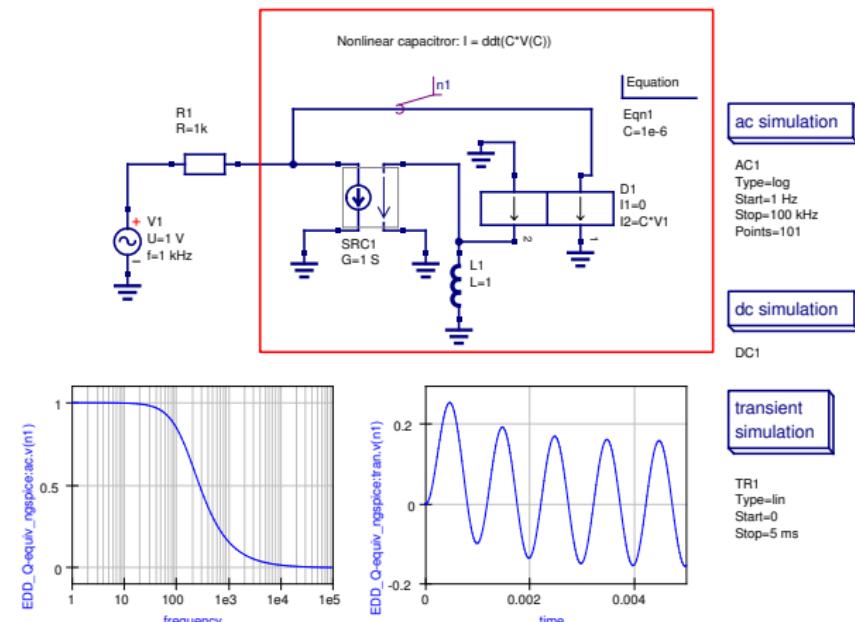
Compact modelling with Qucs and ngspice/Xyce: Part II – Charge equation approach

Nonlinear capacitance current expressed as a function of device voltage can be written as:

$$I = \frac{dQ}{dt} = \frac{d}{dt} CV \quad (2)$$

As Xyce and ngspice appear not to support the diff() operator an electrical equivalent circuit is needed to model capacitor charge equations:

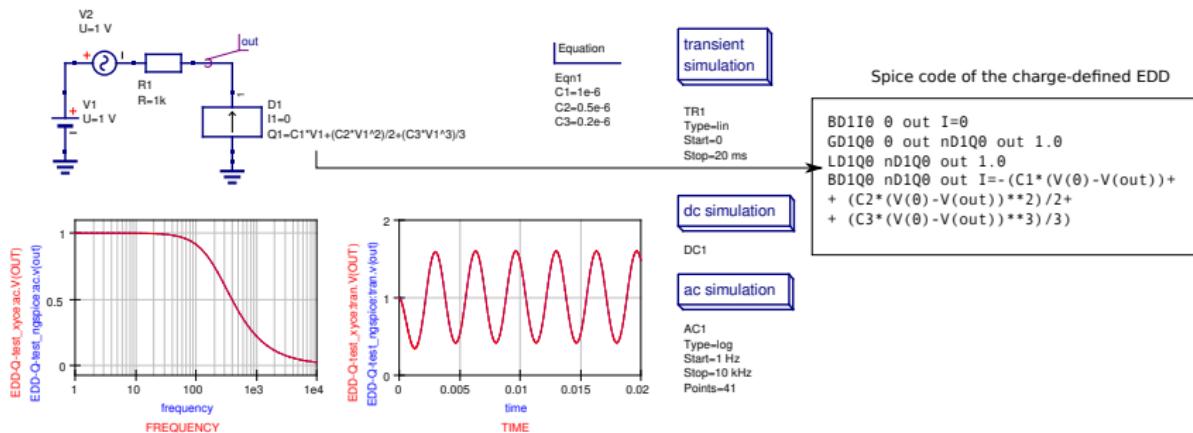
- Nonlinear capacitance equivalent circuit:



Compact modelling with Qucs and ngspice/Xyce: Part III – Charge equations usage example

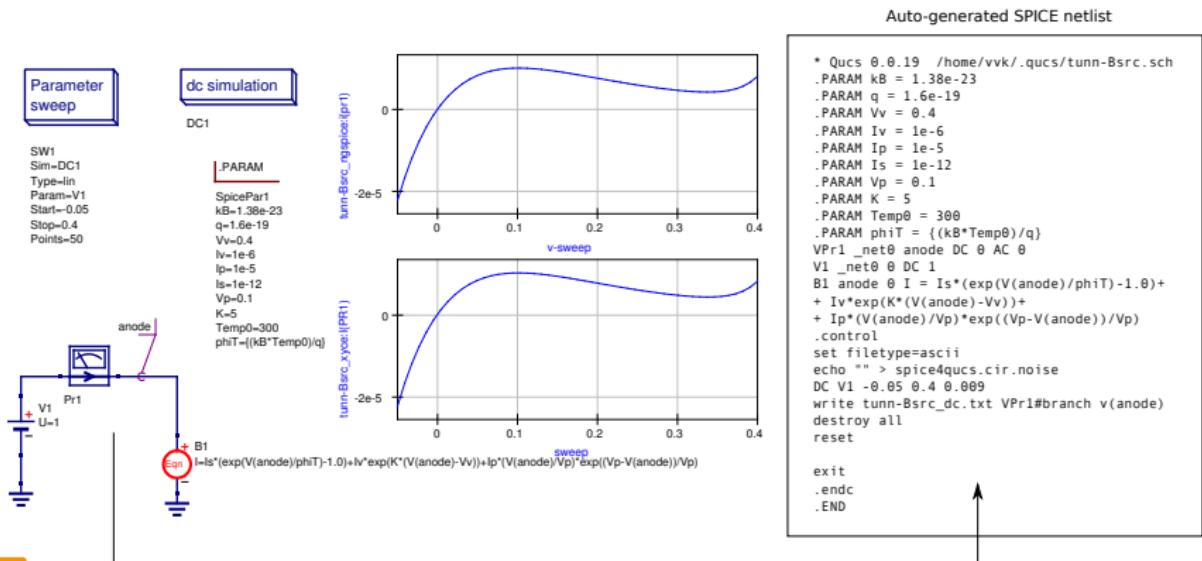
- In this example a nonlinear capacitance is simulated with ngspice and Xyce:

$$Q = C_1 V + \frac{C_2 V^2}{2} + \frac{C_3 V^3}{3} + \dots + \frac{C_N V^N}{N} \quad (3)$$



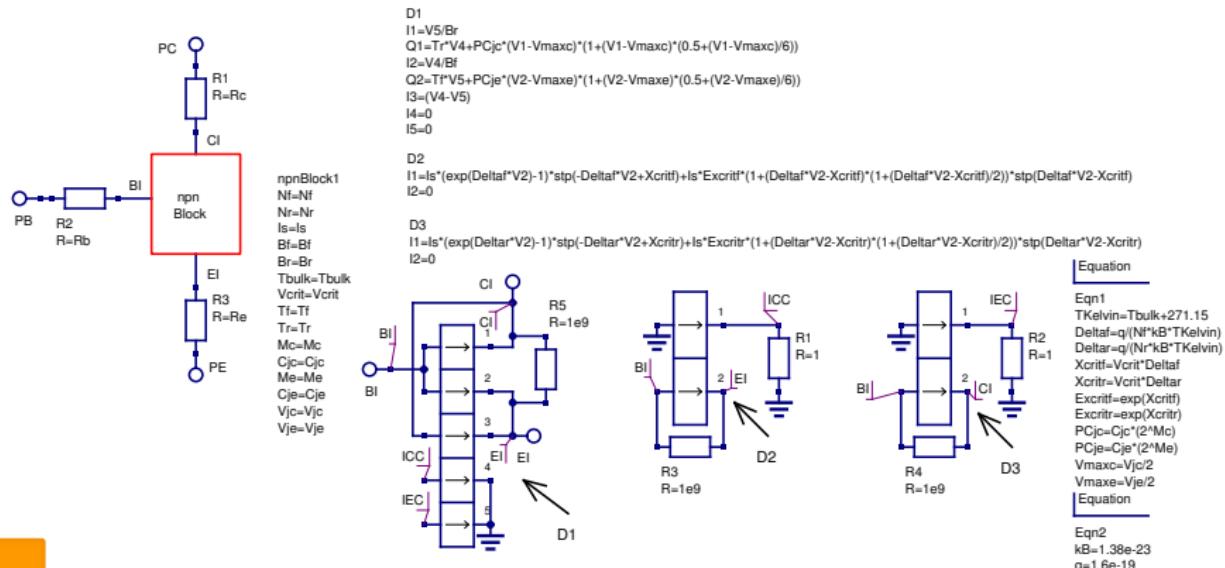
Compact modeling with Qucs and ngspice/Xyce: Part IV – B-type source usage for compact modelling

- Qucs 0.0.19/S introduces a new component: SPICE-compatible equation defined voltage or current sources (SPICE B-type source). The B-type sources allow straight forward construction of compact device models:



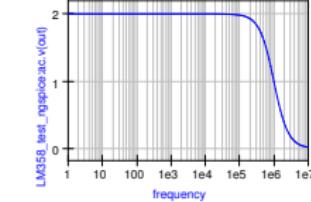
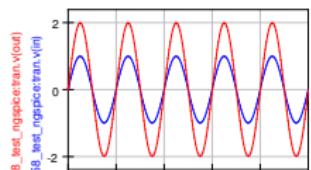
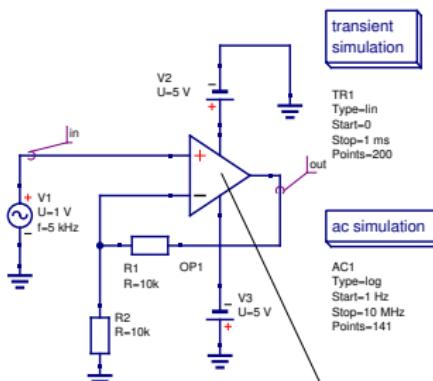
Compact modeling with Qucs and ngspice/Xyce: Part V – NPN BJT compact model used for Harmonic balance analysis of a one-stage BJT amplifier

- Spice4qucs and Xyce allow large signal steady state AC Harmonic Balance simulation, for example the simulation of an experimental NPN BJT compact macromodel:



Compact modelling with Qucs and ngspice/Xyce: Part VII – XSPICE macromodels usage

- Qucs-0.0.19/S allows embedding of SPICE netlist models in Qucs libraries
- An example application of this feature is show below
 - Direct simulation of SPICE defined components
 - XSPICE macromodel usage
- LM358 XSPICE macromodel usage example (noninverting amplifier):

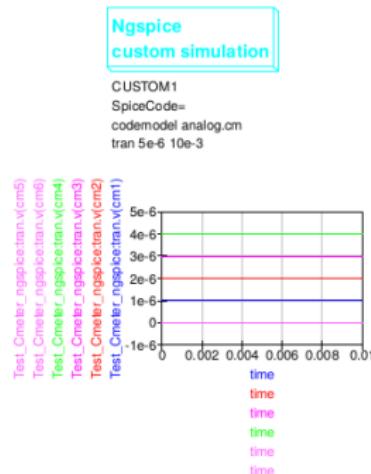
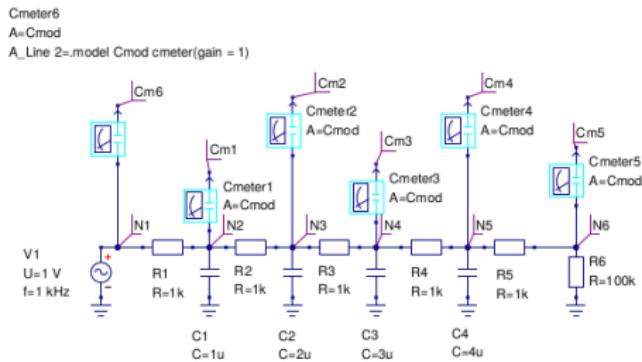


LM358 XSPICE macromodel

```
.SUBCKT LM358 1 2 3 4 5
*
C1 11 12 5.544E-12
C2 6 7 20.00E-12
DC 5 53 DX
DE 54 5 DX
DLP 90 91 DX
DLN 92 90 DX
DP 4 3 DX
EGND 99 0 POLY(2) (3.0) (4.0) 0 .5 .5
FB 7 99 POLY(5) VB VC VE VLP VLN 0 15.91E6
+ -20E6 20E6 20E6 -20E6
GA 6 0 11 12 125.7E-6
GCM 0 6 10 99 7.067E-9
IEE 3 10 DC 10.04E-6
HLM 90 0 VLIM 1K
Q1 11 2 13 QX
Q2 12 1 14 QX
R2 6 9 100.0E3
RC1 4 11 7.957E3
RC2 4 12 7.957E3
RE1 13 10 2.773E3
RE2 14 10 2.773E3
REE 10 99 19.92E6
R01 8 5 50
R02 7 99 50
RP 3 4 30.31E3
VB 9 0 DC 0
VC 3 53 DC 2.100
VE 54 4 DC .6
VLIM 7 8 DC 0
VLP 91 0 DC 40
VLN 0 92 DC 40
.MODEL DX D(I5=800.0E-18)
.MODEL QX PNP(I5=800.0E-18 BF=250)
.ENDS
```

Compact modeling with Qucs and ngspice/Xyce: Part VIII – XSPICE probe usage for circuit node capacitance extraction

- Capacitance probe component (XSPICE CMeter) introduced with Qucs release 0.0.19/S
- It allows extraction of the capacitance connected to a circuit node drawn on a Qucs schematic and simulated by ngspice:

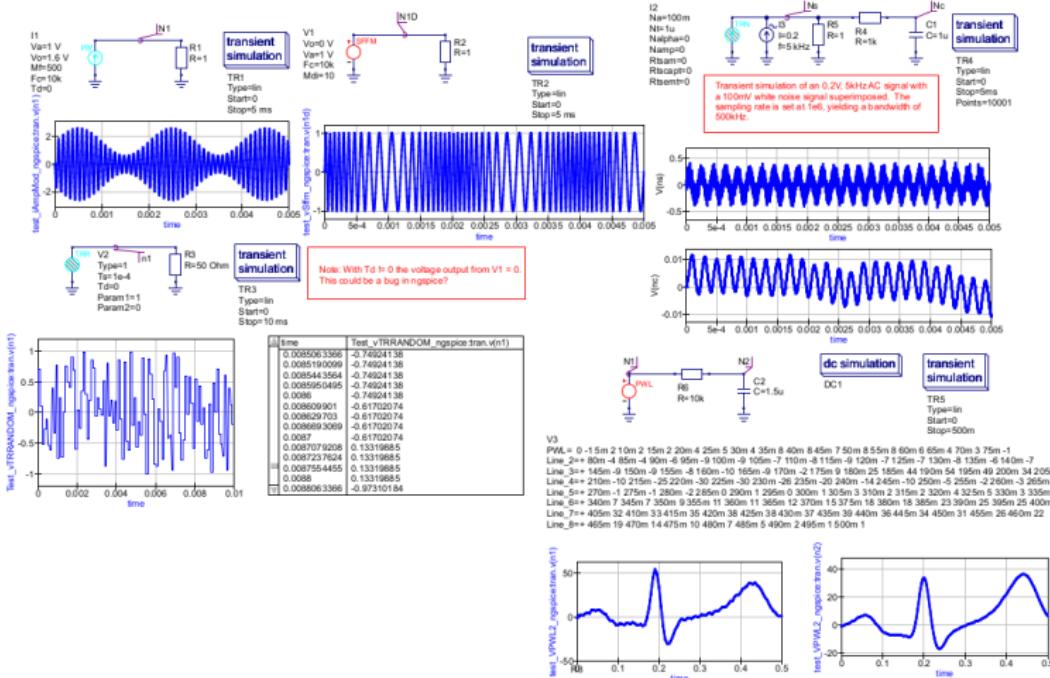


Overview of new components introduced in Qucs-0.0.19/S: Part I – A short components list

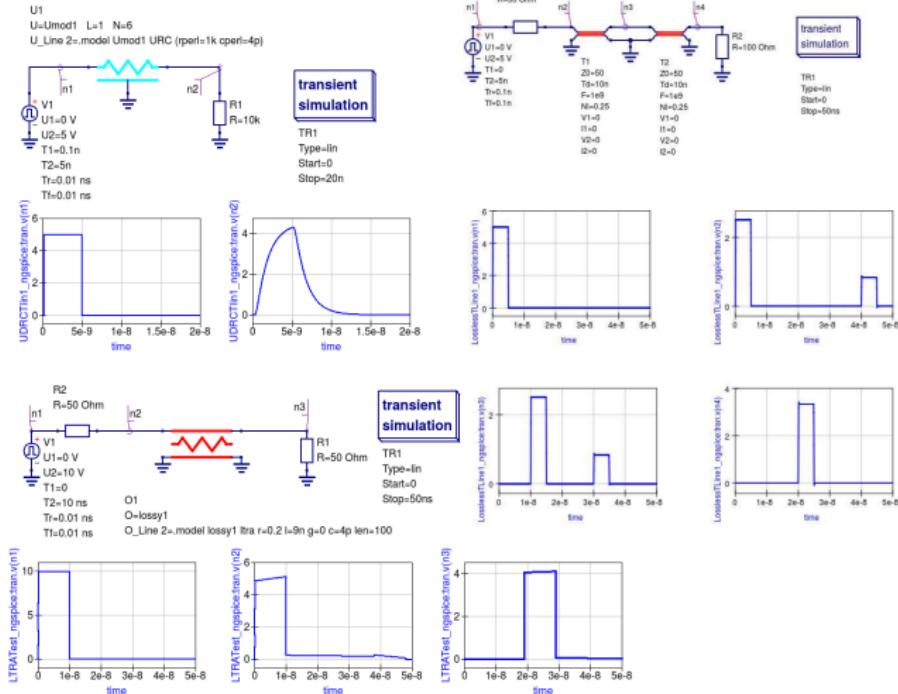
- New SPICE components introduced with Qucs-0.0.19/S:
 - Sources: B-type source, SPICE AC voltage source, SPICE large signal noise sources;
 - Modulated sources: AM, SFFM;
 - Piecewise source: PWL;
 - Passive components: SPICE RCL, inductive coupling;
 - Transmission lines: LTL, LTRA, UDRCTL;
 - Semiconductor devices: diode, BJT, JFET, MOSFET, MESFET;
- SPICE netlist sections: .PARAM, .GLOBAL_PARAM, .IC, .NODESET, .OPTIONS, Nutmeg equation
- Additional SPICE specific simulation routines: .FOUR, .NOISE, .DISTO, and ngspice “Custom ngnutmeg script” simulation



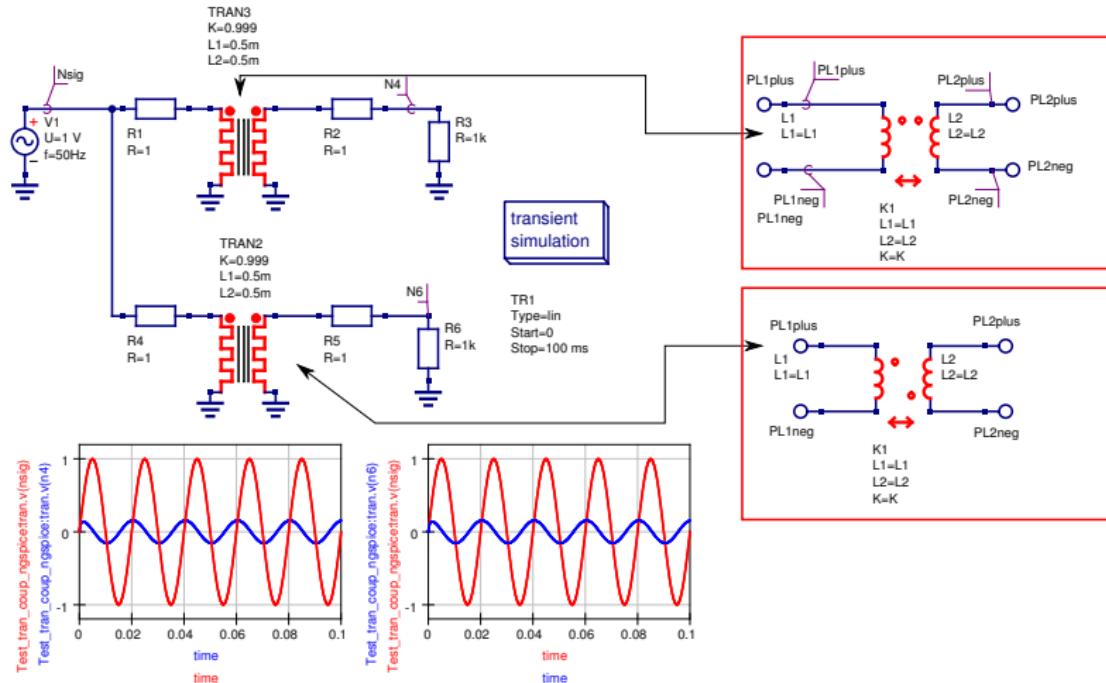
Overview of new SPICE components introduced in Qucs-0.0.19/S: Part II – New SPICE compatible sources



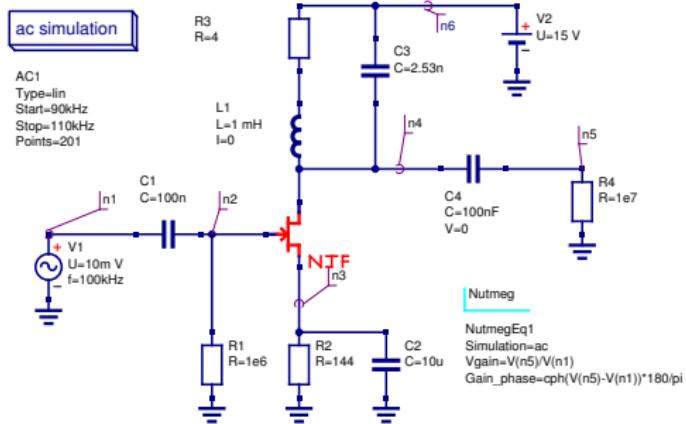
Overview of new SPICE components introduced in Qucs-0.0.19/S: Part III – New transmission line models - these work in the time domain



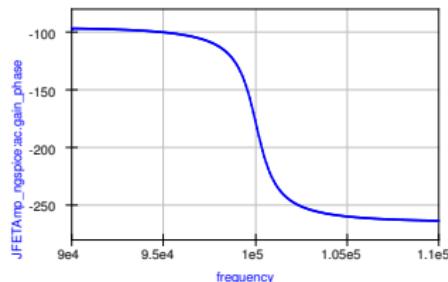
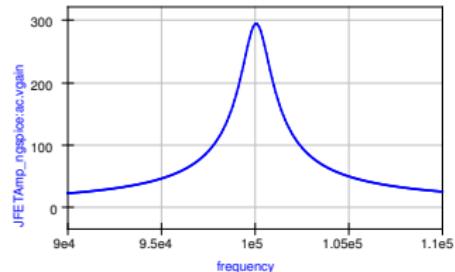
Overview of new SPICE components introduced in Qucs-0.0.19/S: Part IV – Ideal coupled inductor models



Overview of new SPICE components introduced in Qucs-0.0.19S: Part V – Semiconductor devices with full SPICE specification



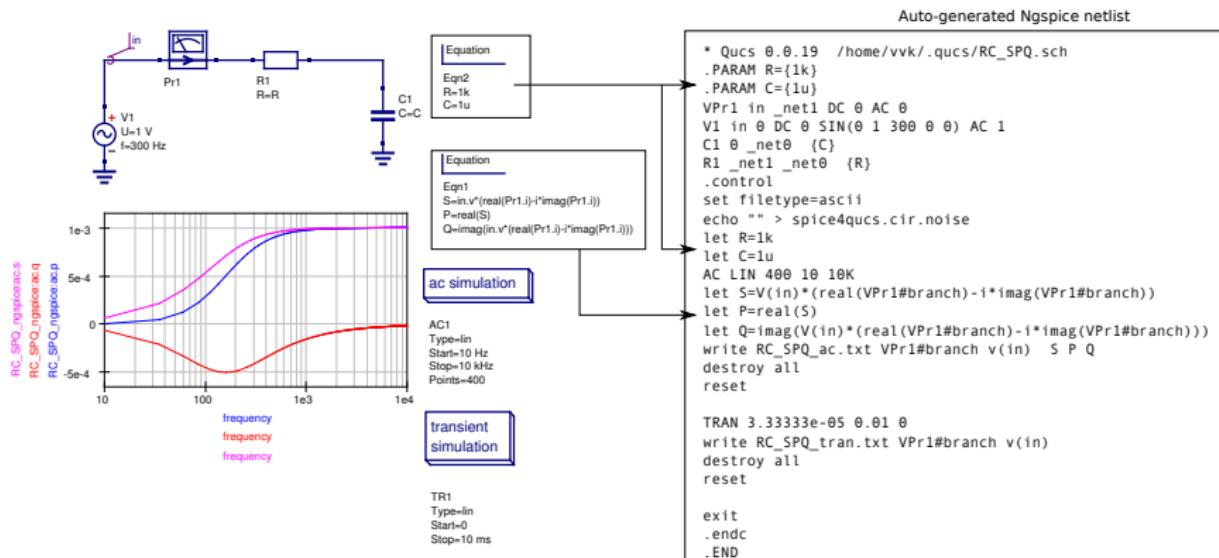
```
J1
J=J2N3819
J_Line 2+=model J2N3819 NJF(Beta=2.0m Betate=-.5 Rd=1 Rs=1 Lambda=2.25m Vto=-3
J_Line 3+=Vtoto=-2.5m Is=33.57f Isr=322.4f N=1 Nr=2 Xti=3 Alpha=311.7u
J_Line 4+=Vg=243.6 Cgd=1.6p M=.3622 Pb=1 Fc=.5 Cgs=2.414p Kf=9.882E-18
J_Line 5+=Af=1)
```



Qucs equation support in spice4qucs

An example for evaluating the total S , active P , and reactive Q power in an RC passive electrical network:

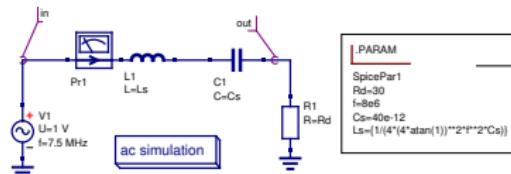
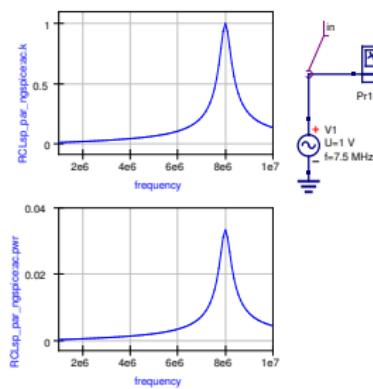
$$S = \text{abs}(U \cdot \bar{I}) \quad P = \Re[U \cdot \bar{I}] \quad Q = \Im[U \cdot \bar{I}] \quad (4)$$



SPICE style parametrization and ngnutmeg postprocessor usage implemented by spice4qucs

The following Qucs "equation" style icons introduce model parametrization and simulation data postprocessing:

- SPICE .PARAM section icon
- ngnutmeg equation icon

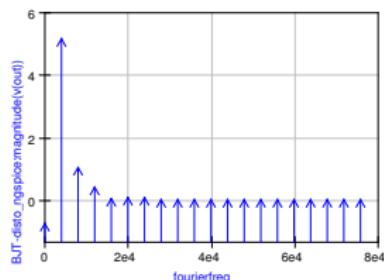
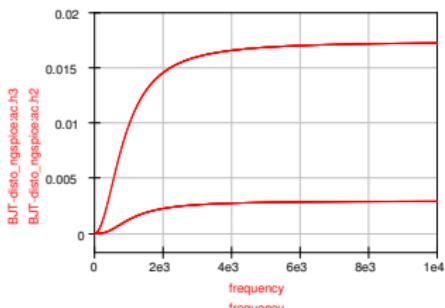
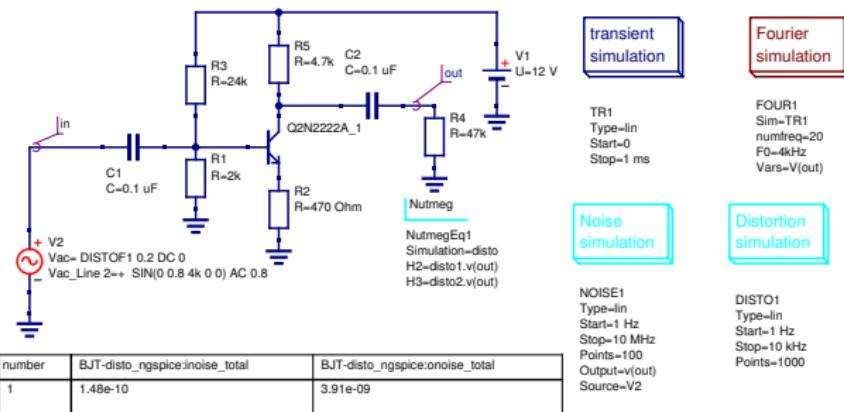


```
Auto-generated Ngspice netlist
* Qucs 0.8.19 /home/vvk/.qucs/RCLsp_par.sch
.PARAM Rd = 30
.PARAM f = 8e6
.PARAM Cs = 40e-12
.PARAM Ls = {1/(4*(4*atan(1))**2*f**2*Cs)}
L1 _net0 _net1 {Ls}
C1 _net1 out {Cs}
VPr1 in _net0 DC 0 AC 0
V1 in _net0 DC 0 SIN(0 1 7.5MEG 0 0) AC 1
R1 0 out {RD}
.control
set filetype=ascii

AC LIN 500 1MEG 10MEG
let K = v(out)/v(in)
let Pwr = v(in)*VPr1#branch
write RCLsp_par_ac.txt v(in) v(out) K Pwr
destroy all
reset

exit
.endc
.END
```

New analysis-simulation types implemented with spice4qucs: SPICE small signal distortion, SPICE small signal AC domain and large signal time domain noise, and SPICE Fourier analysis

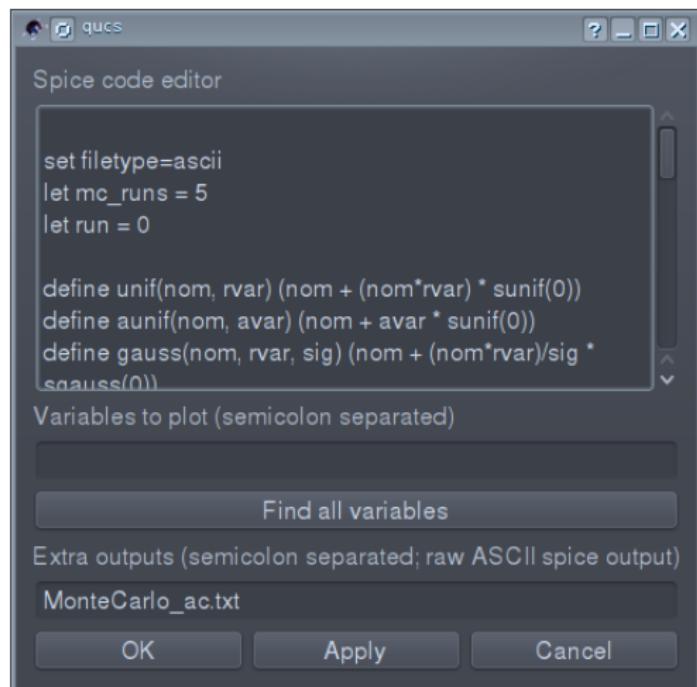


Ngspice custom simulation techniques: Part I – Main features

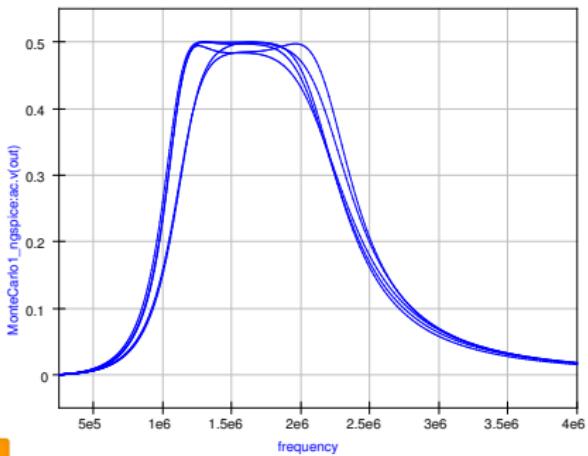
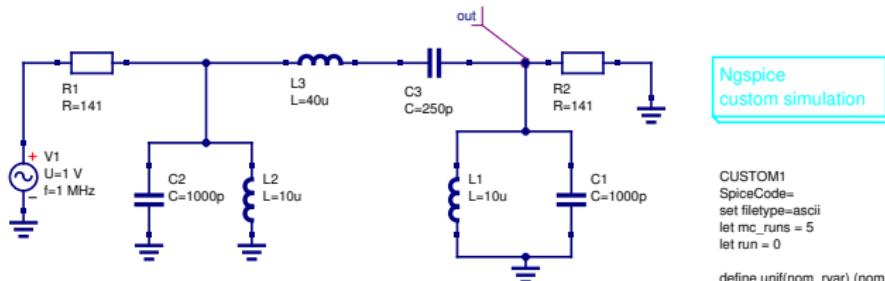
Main features:

- Embedding user defined ngnutmeg scripts in a Qucs schematic
- Full ngnutmeg operator and function support
- User defined variables for plotting simulation data
- User defined raw ASCII SPICE3f5 style output

- Ngnutmeg script editing dialogue:



Ngspice custom simulation technique: Part II – Application example: Monte-Carlo simulation controlled via a ngnutmeg script



```
CUSTOM1
SpiceCode=
set filetype=ascii
let mc_runs = 5
let run = 0

define unif(nom, rvar) (nom + (nom*rvar) * sunif(0))
define aumif(nom, avar) (nom + avar * sunif(0))
define gauss(nom, rvar, sig) (nom + (nom*rvar)/sig * sgauss(0))
define agauss(nom, avar, sig) (nom + avar/sig * sgauss(0))
define limit(nom, avar) (nom + ((sgauss(0) >= 0) ? avar : -avar))

dowhile run < mc_runs $ loop starts here
    *
    *
    alter c1 = unif(1e-09, 0.1)
    alter i1 = unif(1e-06, 0.1)
    alter c2 = unif(1e-09, 0.1)
    alter i2 = unif(1e-06, 0.1)
    alter l3 = unif(40e-06, 0.1)
    alter c3 = limit(250e-12, 25e-12)
    *
    ac oct 100 250K 4Meg

set run ="$&run" $ create a variable from the vector

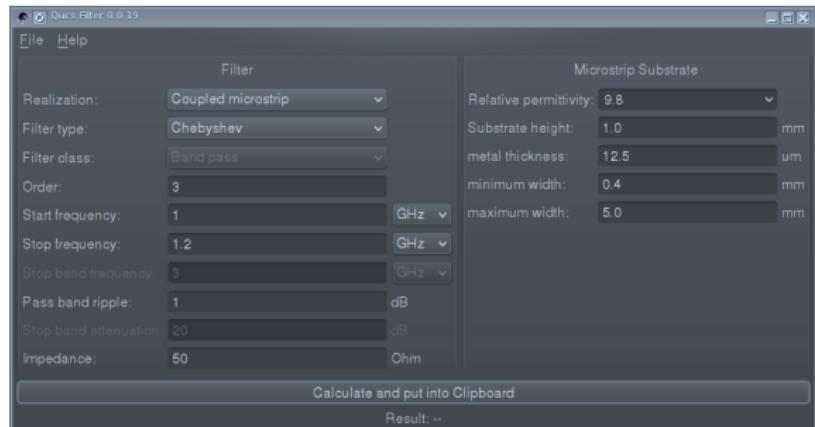
let K = db(v(out))
write MonteCarlo_ac.txt v(out) K
set appendwrite
let run = run + 1
end $ loop ends here
```

Extended passive filter synthesis tool Qucsfilter

Filter topologies added
in Qucs-0.0.19/S:

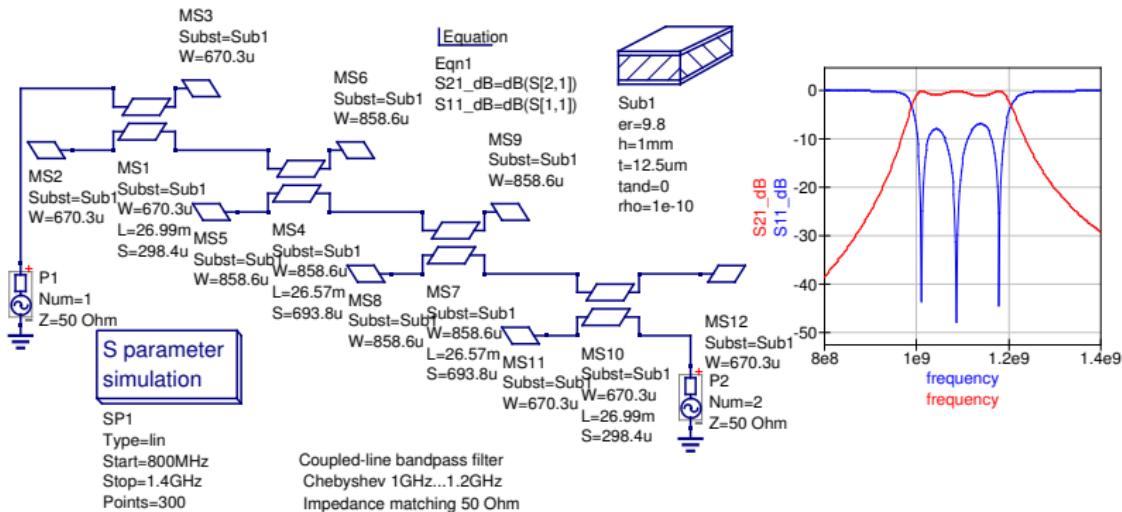
- C-coupled transmission lines
- Coupled microstrip
- Coupled transmission lines
- End-coupled microstrip
- Stepped impedance
- Stepped impedance microstrip

- Qucsfilter utility main window



- An example of a Qucs synthesized filter topology is presented in the next slide

Auto synthesized microstrip filter topology and simulated S parameter frequency response

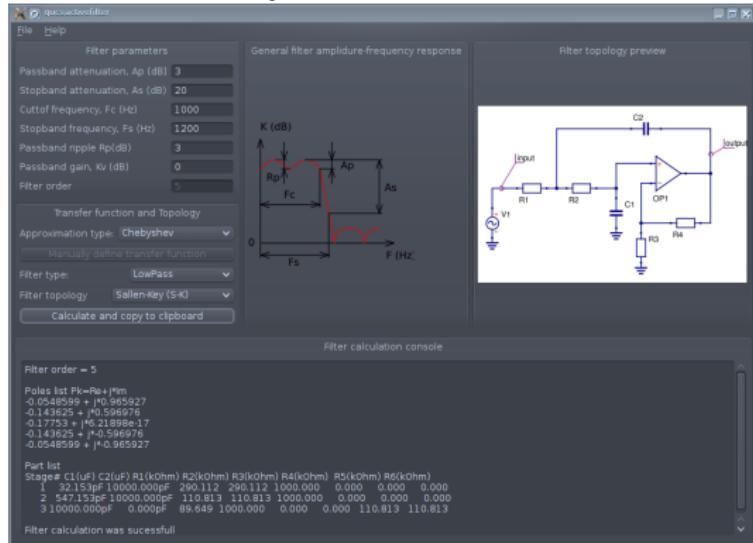


A new Qucs design feature for Active filter synthesis

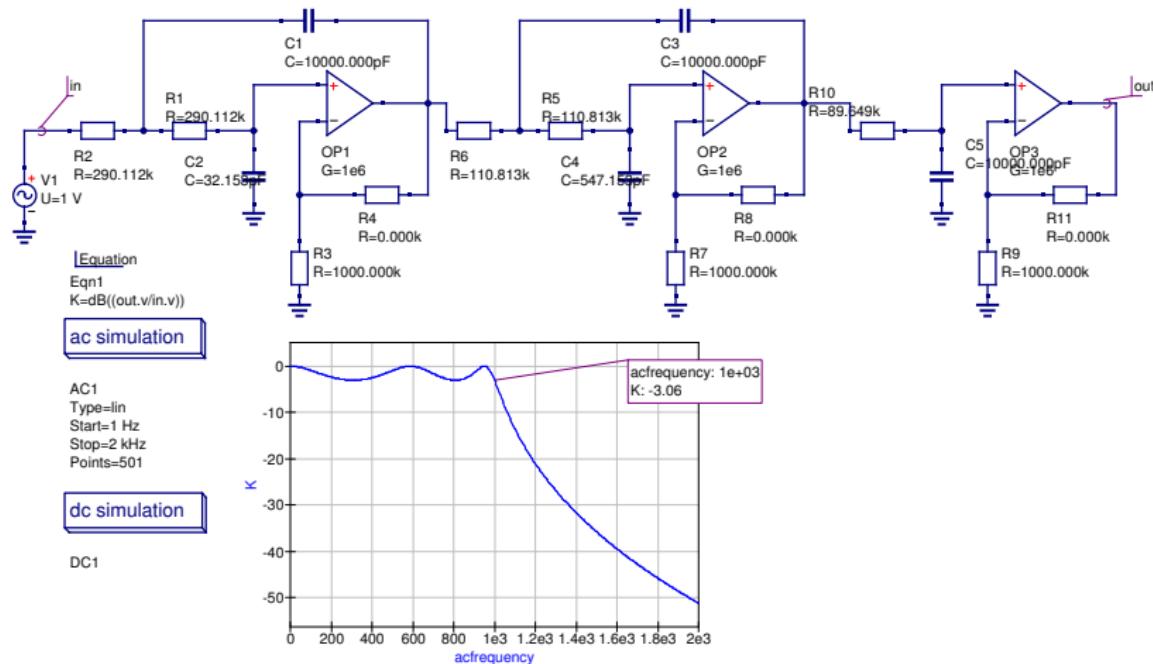
Main features

- Butterworth, Chebyshev (Type I and II), Bessel, and Cauer low-pass, high-pass, band-pass, and band-stop active filters design
- Sallen-Key, Multifeedback, and Cauer filter section topologies are available
- User-defined filter transfer functions
- Full Qucs integration via copy-paste interface with Qucs GUI

Main window of Qucsactive filter:

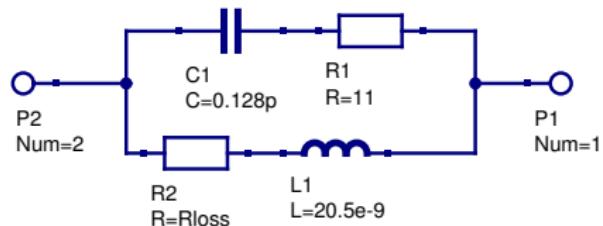


Auto synthesized Chebyshev 5th-order filter and its magnitude response



RFEDD support in spice4qucs: Part I – The problem

- Consider an inductor with frequency dependent losses:



- Frequency dependent resistance losses are given by

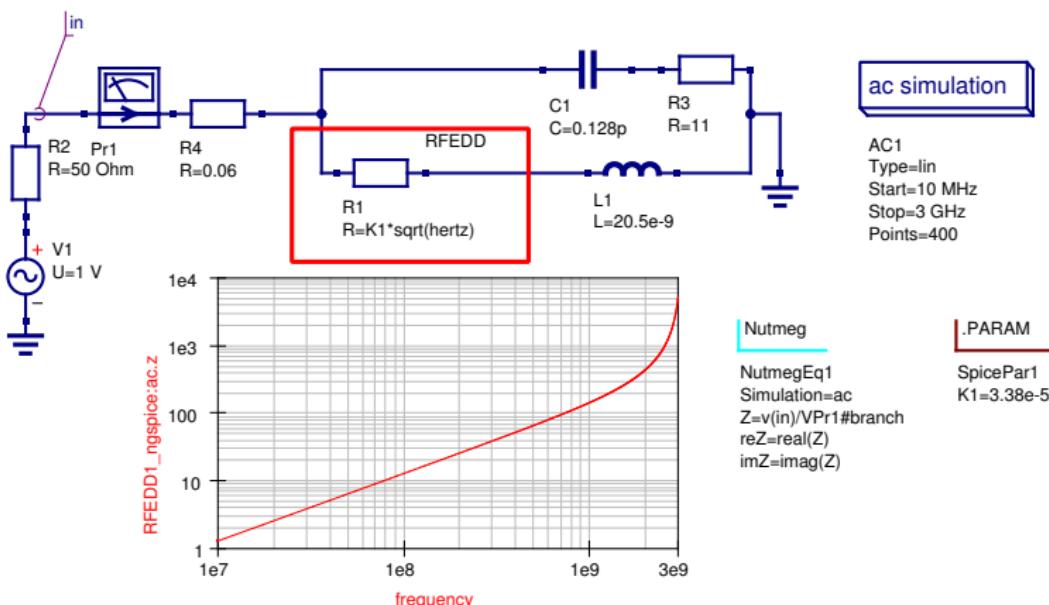
$$R_{loss}(f) = K_1 \sqrt{F} \quad (5)$$

- With an equivalent Z-parameter matrix:

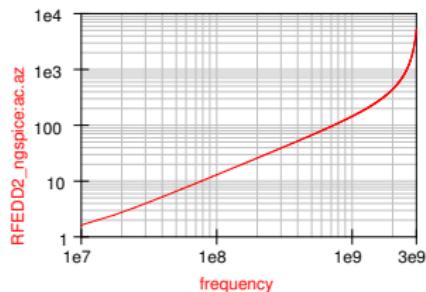
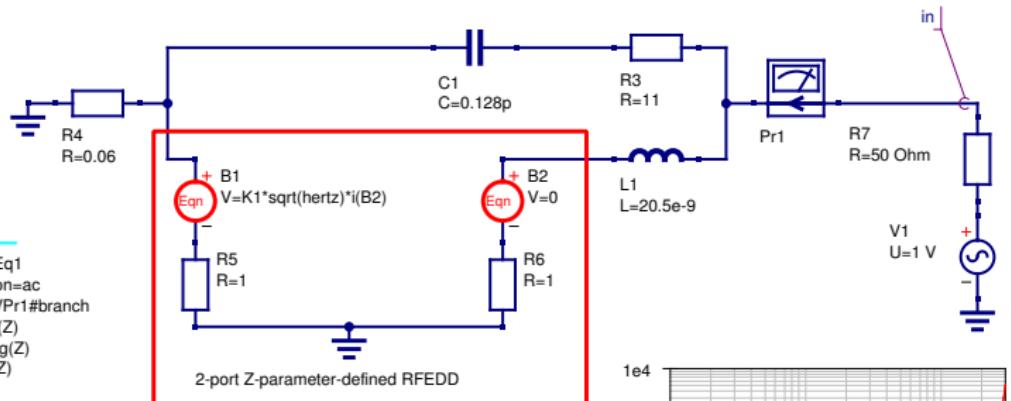
$$Z = \begin{bmatrix} 1 & K_1 \sqrt{F} \\ 1 & 0 \end{bmatrix} \quad (6)$$

RFEDD support in spice4qucs: Part II – ngspice approach

- The ngspice “hertz” frequency variable can be used in algebraic expressions to represent passive component (RCL) frequency dependence:



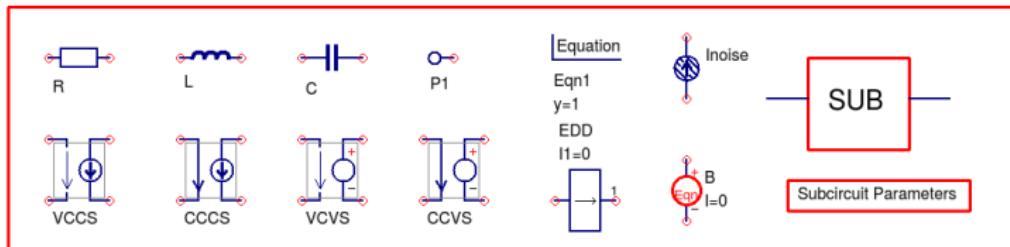
RFEDD support in spice4qucs: Part III – A Qucs RFEDD equivalent circuit for resistance frequency dependent losses



Introduction to the Qucs GPL Verilog-A module synthesizer: Part I

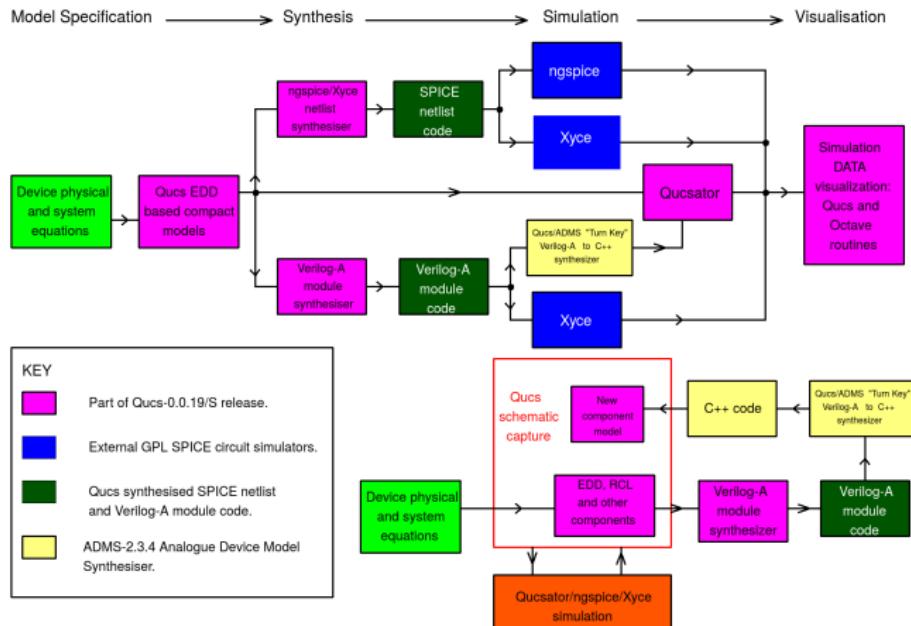
Qucs-0.0.19S includes the first release of a GPL Verilog-A synthesis tool for compact device modelling.

- The Qucs-0.0.19S Verilog-A synthesizer is a basic working version of this new open source ECAD tool.
- It is for test purposes: bugs are likely and it may not be very stable.
- Generated synthesized Verilog-A code is relatively basic and has to be optimized manually for speed. However, it is expected that in the future its operation will improve as development of the Qucs synthesizer progresses.
- Circuits and Verilog-A synthesized models can be constructed from the following Qucs/SPICE built in components:



Introduction to the Qucs GPL Verilog-A module synthesizer: Part II

Structure:

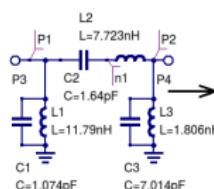


Introduction to the Qucs GPL Verilog-A module synthesizer: Part III

Data flow through the Qucs GPL compact device modelling tool set.

QUCS FILTER SYNTHESIS

Realization : LC ladder (pi-type)
 Type: Bessel
 Class: Bandpass
 Order: 3
 Fstart: 1 GHz
 Fstop: 2 GHz
 Impedance: 50 Ohm



VERILOG-A MODEL SYNTHESIS

```
'include "disciplines.vams"
'include "constants.vams"
module BPF2(P1, P2);
inout P1, P2;
electrical P1, _net0L1, n1, P2, _net0L2, _net0L3;
analog begin
@(initial_model)
begin
end
I(_net0L1) <+ ddif(V(_net0L1));
I(_net0L1) <+ (-V(P1));
I(P1) <+ V(_net0L1)/(11.79n+1e-20);
I(P1) <+ ddif( (-V(P1)) * 1.074p );
I(_net0L2) <+ ddif(V(_net0L2));
I(_net0L2) <+ V(n1,P2);
I(_net0L2) <+ V(1,P2);
I(n1,P2) <+ ddif(V(_net0L2))/(7.723n+1e-20);
I(P1,n1) <+ ddif( V(P1,n1) * 1.64p );
I(_net0L3) <+ ddif(V(_net0L3));
I(_net0L3) <+ (-V(P2));
I(P2) <+ V(_net0L3)/(1.806n+1e-20);
I(P2) <+ ddif( (-V(P2)) * 7.014p );
end
endmodule
```

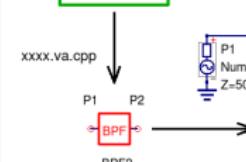
Build Verilog-A module from subcircuit

QUCS/ADMS VERILOG-A "TURN KEY" COMPILER



xxxx.va.cpp

ADMS



DEVELOP TEST CIRCUIT, SIMULATE, AND EVALUATE OUTPUT DATA

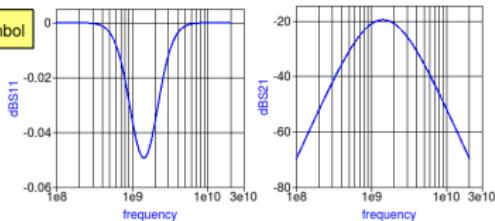
Create circuit schematic and simulate

dc simulation

S parameter simulation

SP1
 Type=log
 Start=100MHz
 Stop=20GHz
 Points=201

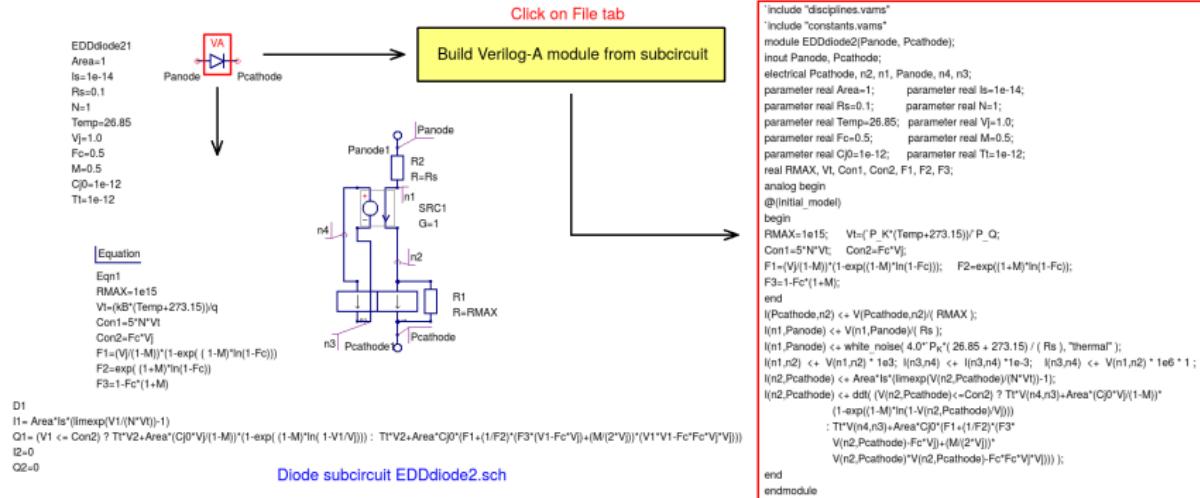
Edit text symbol



Plotted and tabulated simulation data

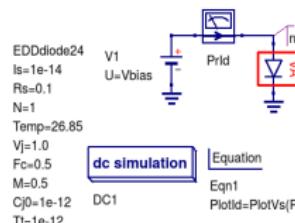
Introduction to the Qucs GPL Verilog-A module synthesizer: Part IV

Synthesis of a SPICE like compact semiconductor diode model: static I_d and dynamic capacitance model plus synthesized Verilog-A module code.



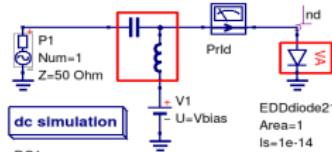
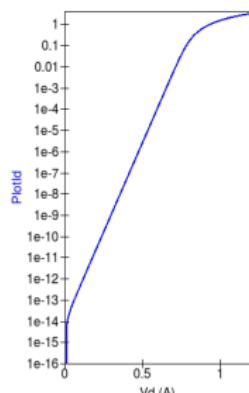
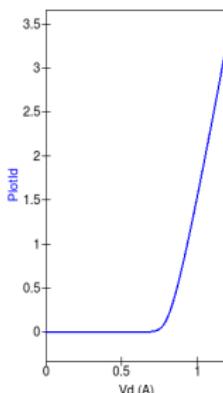
Introduction to the Qucs GPL Verilog-A module synthesizer: Part V

Synthesis of a SPICE like semiconductor diode model: simulated static and dynamic characteristics.



Parameter sweep

SW1
Sim=DC1
Type=lin
Param=Vbias
Start=0
Stop=1.2
Points=101



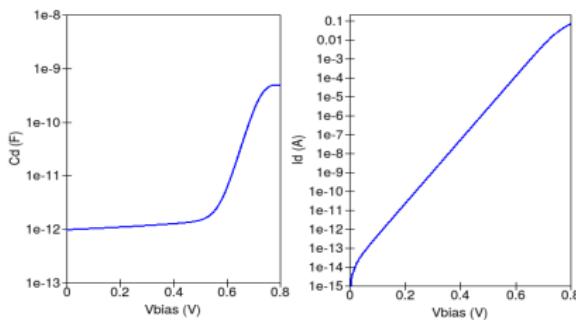
S parameter simulation

SP1
Type=const
Values=[10e6]

Parameter sweep

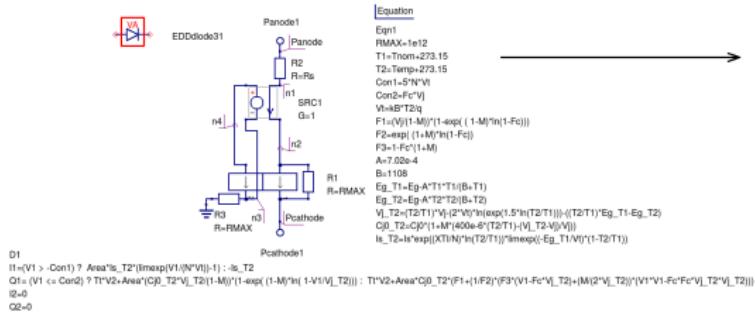
SW1
Sim=DC1
Type=lin
Param=Vbias
Start=0
Stop=1.2
Points=101

Equation
Eqn1
y=stoy(S)
Cd=PlotVs(imag(y[1,1])/Omega,Vbias)
Omega=2*pi*frequency



Introduction to the Qucs GPL Verilog-A module synthesizer: Part VI

Verilog-A synthesis of a SPICE like semiconductor diode model: temperature effects



Qucs EDD diode model with temperature effects

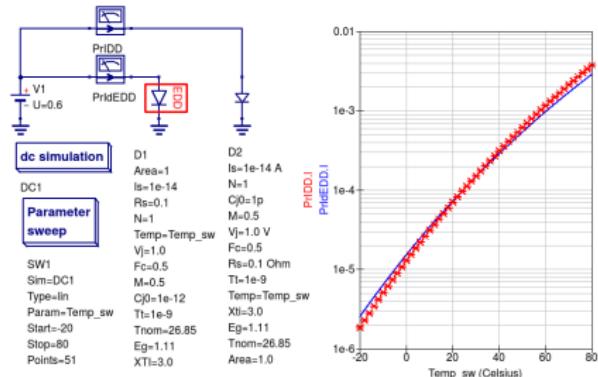
```

include "disciplines.vams"
include "constants.vams"
module EDDdiode3(Pnnode, Pnnode);
  input Pnnode, Pnnode;
  output Pnnode, Pnnode;
  parameter real Area=1; parameter real Is=1e-14; parameter real Rs=0.1; parameter real N=1;
  parameter real Temp=26.85; parameter real Vt=1.0; parameter real Fc=0.5; parameter real M=0.5;
  parameter real C0=1e-12; parameter real Rd=1e-9; parameter real Thoms=26.85; parameter real Eg=1.11;
  parameter real XTI=3.0;
  real RMAX, T1, T2, Con1, Con2, F1, F2, F3, A, B, Eg1, Eg2, Vt2, C0=0.2, Isr2;
  analog begin
    @(!initial_model)
    begin
      RMAX=Xti*T2; T1=Thoms-273.15; T2=Temp-273.15; Con1=5*N*Vt; Con2=Fc*T2/Pt; Vt=k*T2/Pt;
      F1=(Vt/(1-M))^(1-exp(-(1-M)*ln(1-Fc))); F2=exp(-(1-M)*ln(1-Fc)); F3=1-Fc*(1+M); A=7.02e-4; B=1108;
      Eg=A*T1*T1/(B+T1); Eg2=Ep*A*T2*T2/(B+T2);
      Vt_2=(T2/T1)*Vt^(2*Vt)/(exp(1.5*ln(T2/T1))-(T2/T1)*Eg_1-Eg_T2);
      C0_T2=C0^(1+M^(400e-6*(T2/T1)-[Vt_2*T2/Vt]));
      Is2=Is*exp((XTI/N*ln(T2/T1))/Imexp(-Eg_1/Vt_2)*(1-T2/T1));
      Isr2=Is*exp((XTI/N*ln(T2/T1))/Imexp(-Eg_1/Vt_1)*(1-T2/T1));
    end
    Pnnode <-> V(Pnnode,n2)/RMAX;
    Pnnode <-> white_noise(4.0*Pw*(26.85 + 273.15) / (RMAX), "thermal");
    !Int(Pnnode) <-> V(n1,Pnnode)/Rs;
    !Int(Pnnode) <-> white_noise(4.0*Pw*(26.85 + 273.15) / (RMAX), "thermal");
    Int(Pnnode) <-> V(n1,Pnnode)/Rs;
    Int(Pnnode) <-> white_noise(4.0*Pw*(26.85 + 273.15) / (RMAX), "thermal");
    Int(Pnnode) <-> V(n3,Pnnode)-Con1?Area*Is2*Imexp(Vn2,Pnnode)/(N*Vt)-1:Is2;
    Int(Pnnode) <-> ddt((Vn2,Pnnode)-(Con2*T1)Vn1/[n1,n2]+Area*(C0*2*Vt_2*(1-M)^2*1-exp(-(1-M)^2*Vt_2)));
    Int(-Vn2,Pnnode)/Vj(2));
    T1*Vt_2*Area*C0*2*(F1+[1/F2]*[F3*(Vn2,Pnnode)-Fc*Fc*Vt_2]+M^2*Vt_2);
    (Vn2,Pnnode)/Vn2,Pnnode)*Fc*Fc*Vt_2*Vt_2);
    Is3 <-> (-V(n3))/RMAX;
    Is3 <-> white_noise(4.0*Pw*(26.85 + 273.15) / (RMAX), "thermal");
  end
endmodule

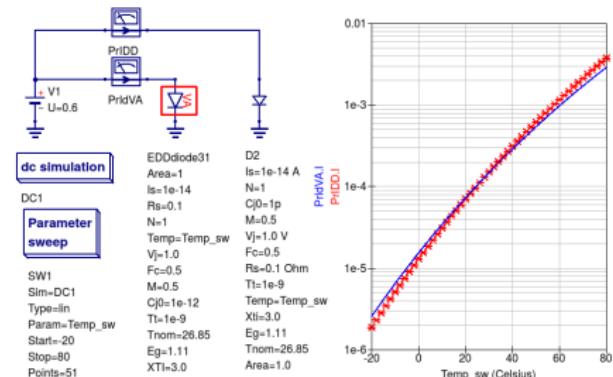
```

Introduction to the Qucs GPL Verilog-A module synthesizer: Part VII

Verilog-A synthesis of a SPICE like semiconductor diode model: simulated $I_d - V_d$ temperature effects.



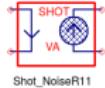
Simulation data for
Qucs EDD model and built-in diode model



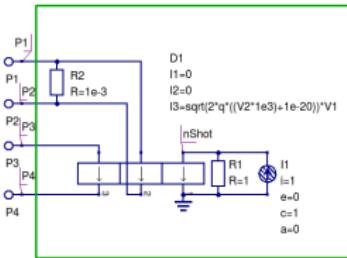
Simulation data for
Verilog-A model and built-in diode model

Introduction to the Qucs GPL Verilog-A module synthesizer: Part VIII

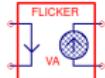
Verilog-A synthesis of semiconductor device shot and flicker noise: EDD models and Verilog-A module code.



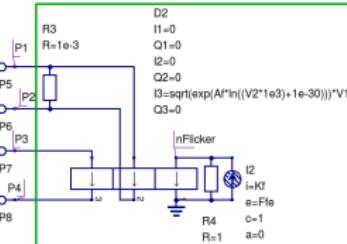
Noise model symbols



```
'include "disciplines.vams"
'include "constants.vams"
module Shot_NoiseR1(P1, P2, P3, P4);
inout P1, P2, P3, P4;
electrical nShot, P2, P1, P3, P4;
analog begin
@(initial_model)
begin
end
I(nShot) <= (-V(nShot))/( 1 );
I(nShot) <+ white_noise(1,"shot");
I(P2,P1) <= V(P2,P1)/( 1e-3 );
I(P3,P4) <= sqrt(2^*P_o^*((V(P1,P2)^*1e3)+1e-20))*V(nShot);
end
endmodule
```



Flicker_NoiseR1
 $Kf=1e-16$
 $Fle=1$
 $Af=1$

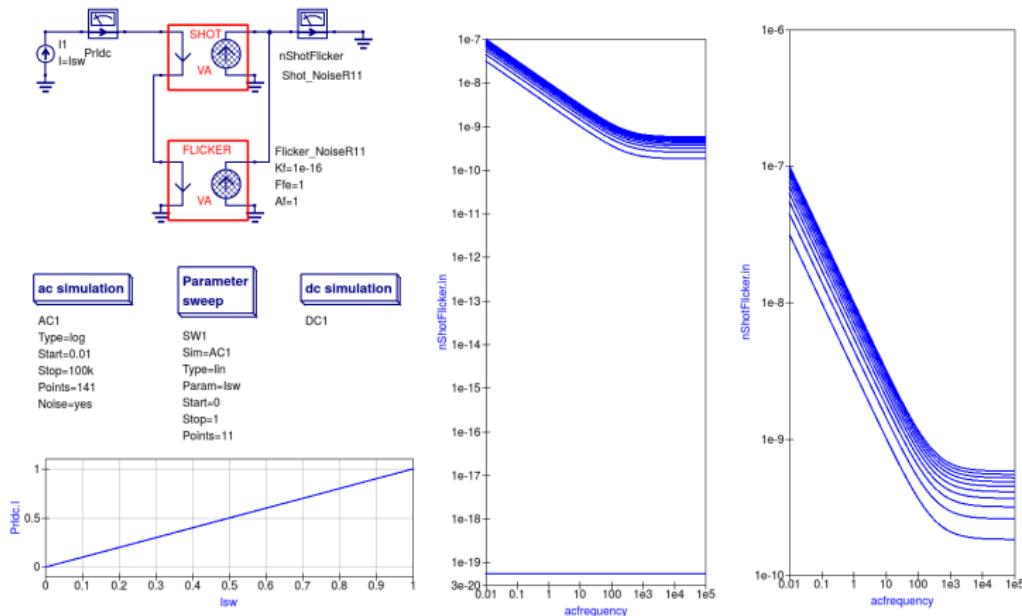


Synthesized Verilog-A module code

```
'include "disciplines.vams"
'include "constants.vams"
module Flicker_NoiseR1(P1, P2, P3, P4);
inout P1, P2, P3, P4;
electrical P2, P1, nFlicker, P3, P4;
parameter real Kf=1e-12;
parameter real Fle=1;
parameter real Af=1;
analog begin
@(initial_model)
begin
end
I(P2,P1) <= V(P2,P1)/( 1e-3 );
I(nFlicker) <+ flicker_noise(Kf, Fle, "flicker");
I(nFlicker) <= (-V(nFlicker))/( 1 );
(P3,P4) <= sqrt(exp(Af^*\ln((V(P1,P2)^*1e3)+1e-30)))*V(nFlicker);
end
endmodule
```

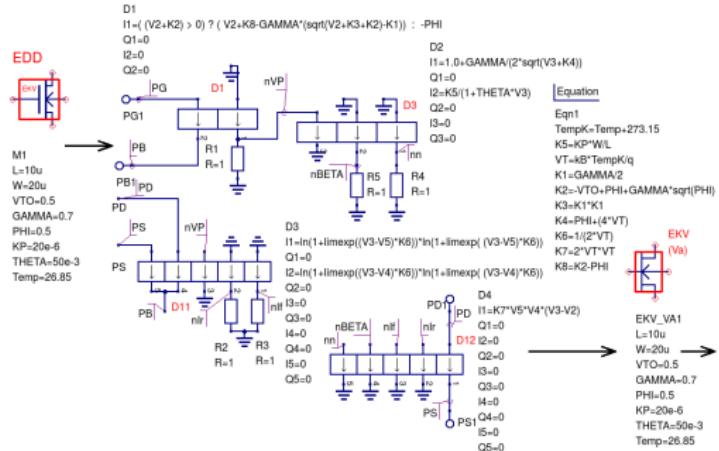
Introduction to the Qucs GPL Verilog-A module synthesizer: Part IX

Verilog-A synthesis of semiconductor device shot and flicker noise: small signal AC domain simulation data.



Introduction to the Qucs GPL Verilog-A module synthesizer: Part X

Verilog-A synthesis of multi-EDD models: EKV2p6 nMOS
 $I_{ds} = f(V_d, V_g, V_s, V_b)$ model for a transistor operating in long channel mode.



```

'include "disciplines.vams"
'include "constants.vams"
module EKV_VA(PB, PG, PD, PS);
input PB, PG, PD, PS;
electrical PD, PS, ntr, nlf, nBeta, nn, nVP, PG, PB;
parameter real L=10u; parameter real W=20u; parameter real VTO=0.5;
parameter real GAMMA=0.7; parameter real PHI=0.5; parameter real KP=20e-6;
parameter real THETA=50e-3; parameter real Temp=26.85;
real TempK, K5, VT, K1, K2, K3, K4, K6, K7, K8;
analog begin
@initial model
begin
TempK=Temp+273.15; K5=KP*W/L; VT=P_K*TempK/P_Q;
K1=GAMMA/2; K2=VTO*PHI+GAMMA*sqrt(q)/PHI;
K3=K1*K1; K4=PHI*(4*VT);
K6=1/(2*VT); K7=2*VT*VT; KB=K2*PHI;
end
I(PD,PS) <+ K7*V(ntr)*(V(nBeta)*(V(nlf)-V(nn)));
I(nVP) <+ ((V(PG,PB)+K2)>0)?(V(PG,PB)+K8*GAMMA*(sqrt(V(PG,PB)+K3+K2)-K1)):PHI);
I(nn) <+ V(nVP)*V(1);
I(nBeta) <+ (-K5*(V(nlf)-V(nn)));
I(nlf) <+ ((K5*(1+THETA)*(V(nVP))));;
I(nn) <+ (-V(nlf));
I(nlf) <+ ((V(nlf)-V(1)));
I(nlf) <+ (-ln(1+ilimexp(V(nVP)-V(PG,PB))*K6))/ln(1+ilimexp(-V(nVP)-V(PG,PB))*K6));
I(nn) <+ (-ln(1+ilimexp(V(nVP)-V(PD,PS))*K6))/ln(1+ilimexp(-V(nVP)-V(PD,PS))*K6));
end
endmodule

```

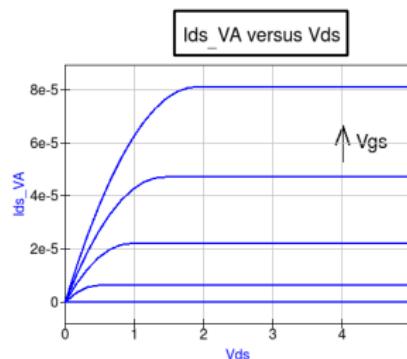
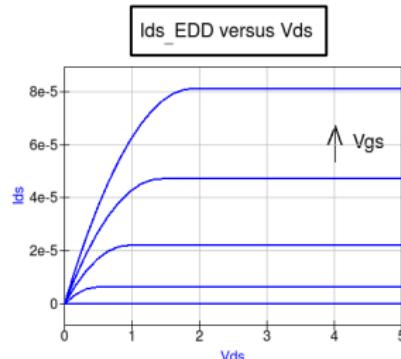
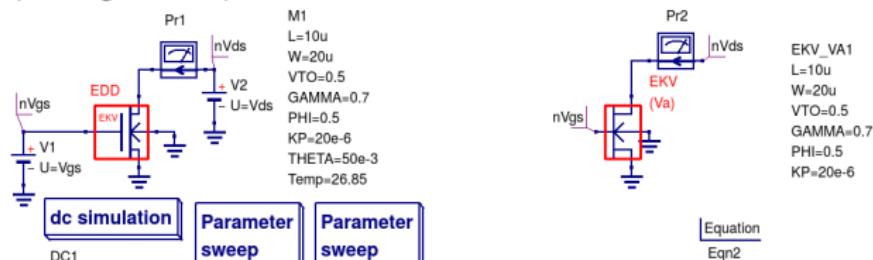
Qucs EDD EKV2p6 $I_{ds}=f(V_d, V_g, V_s, V_b)$ model

Synthesized EKV2p6 $I_{ds}=f(V_d, V_g, V_s, V_b)$ Verilog-A code

Introduction to the Qucs GPL Verilog-A module synthesizer: Part XI

Verilog-A synthesis of multi-EDD models: EKV2p6 nMOS

$I_{ds} = f(V_d, V_g, V_s, V_b)$ swept DC simulation data.



Introduction to the Qucs GPL Verilog-A module synthesizer: Part XII

Verilog-A synthesis of multi-EDD models: Optimization of Qucs synthesized Verilog-A module code for speed.

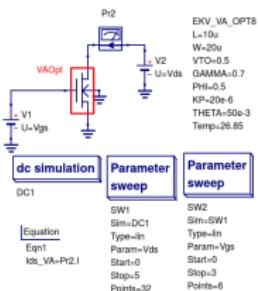
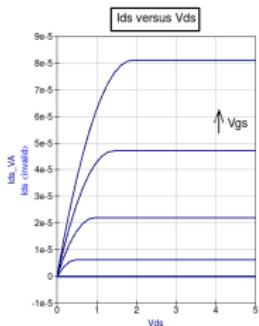
```

#include "disciplines.vams"
#include "constants.vams"
module EKV_VA_OPT(PB, PG, PD, PS);
  input PB, PG, PD, PS;
  electrical PD, PS, PG, PB;
  parameter real L=10e-3; parameter real W=20e-3;
  parameter real VT0=-0.5; parameter real GAMMAA=0.7;
  parameter real PH1=-0.5; parameter real KP=20e-6;
  parameter real THETA=50e-3; parameter real Temp=-26.85;
  parameter K1, K2, K3, K4, K5, K7, KB;
  real Vg, Vs, Vd, nVP, nBeta, nn, nlf, nlr;
  analog begin
    @([initial] model)
  begin
    TempC=Temp+273.15; K5=KP/W/L;
    VT= P_ KTempK/P_Q; K1=+GAMMA/2;
    K2=-VT0+PH1+GAMMA*sqrt(PH1); K3=K1'K1;
    K4=PH1*(4*VT); K6=1/(2^2*VT); K7=2*VT*VT;
  end
  Vg = VP(G,PB); Vs = VP(S,PB); Vd = VP(D,PB);
  nVP = ((Vg-K2)>0)?(Vg-K2+PH1+GAMMA*sqrt((Vg-K2)*K3)-K1):-PH1;
  nBeta = K5*(1+THETA*VP);
  nlf = 1.0+GAMMA/(2*sqrt(VP*K4));
  nn = ln[1+lnexp((nVP-Vd)/K6)]*ln[1+lnexp((nVP-Vs)/K6)];
  nlr = ln[1+lnexp((nVP-Vd)/K6)]*ln[1+lnexp((nVP-Vs)/K6)];
  nLP,PS <- K7*nBeta*(nn-nlr);
  end
endmodule

```

TEST MODULE

NOTES



A comment on the Qucs simulation process:

Simple simulation run time tests indicate that the optimized EKV2p6 Verilog-A model simulation speed is at least 30X faster than the Interactive EDD model.

- At this stage in the development of the Qucs synthesizer optimized Verilog-A module code is done manually.
- General procedure:
 - Reduce current contribution statements to a minimum. This can be done by representing model equation quantities by real variables rather than internal node voltages.
[one l(a) <+ in the EKV nMOS example]
 - Eliminate as many as possible internal model nodes and remove current to voltage one Ohm conversion resistors.
[zero left in EKV nMOS example]

Conclusion

Summary:

- Version 0.0.19 is a major release of the Qucs circuit simulator, updating the popular RF package while simultaneously adding a new software tool, Qucs 0.0.19S, which provides Qucs users with an experimental software package that links legacy Qucs with ngspice and Xyce GPL SPICE.

In the future the main Qucs development directions are likely to be:

- Further integration of Qucs with ngspice and Xyce: including improvement of the existing ngnutmeg support, an RFEDD synthesizer implementation, additional analysis support for SPICE .SENS and .PZ etc, and a range of new SPICE compatible components, for example magnetic core models.
- Improvements to the Verilog-A module synthesizer.
- Implementation of mixed signal simulation with ngspice/XSPICE and Xyce.

Qucs-0.0.19S-RC3 from

<https://github.com/ra3xdh/qucs/releases/download/0.0.19S-rc3/qucs-0.0.19Src3.tar.gz> (linux source)
<https://github.com/ra3xdh/qucs/releases/download/0.0.19S-rc3/qucs-0.0.19Src3-setup.zip> (Windows installer)

