

An outline of Qucs-S compact device modelling: History and capabilities: Part 2 XSPICE Code Models; basic properties to model synthesis, and beyond

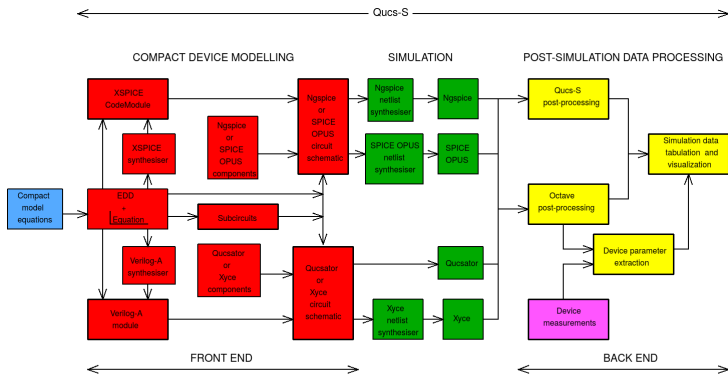
Mike Brinson ¹, mbrin72043@yahoo.co.uk.
Vadim Kuznetsov ², ra3xdh@gmail.com

¹Centre for Communications Technology, London Metropolitan University,
UK

²Bauman Moscow Technical University, Russia



A flow chart showing Qucs-S compact modelling facilities and data movement



NOTES:

1. Qucs-S allows the selection of the simulation engine to use.
2. Available simulation components depends on the simulation engine chosen.
3. Users may select either Qucs-S or Octave post-processing of simulator data.

Qucs-S binary packages

- Debian packages are available here:
<http://download.opensuse.org/repositories/home:/ra3xdh/>
- Windows Installer: <https://github.com/ra3xdh/qucs/releases/download/0.0.19S/qucs-0.0.19S-setup.zip>

The screenshot shows the openSUSE Build Service web interface for the project 'home:ra3xdh > Qucs-S'. The page includes navigation tabs for Overview, Repositories, Revisions, Requests, Users, and Advanced. A 'Download package' button is visible. The 'Source Files' section contains a table with columns for Filename, Size, Changed, and Actions. The 'Build Results' section shows a list of successful builds for various architectures (i586, x86_64) across different operating systems (Debian, xUbuntu).

Filename	Size	Changed	Actions
debian.changelog	131 Bytes	28 days ago	
debian.compat	1 Byte	26 days ago	
debian.control	578 Bytes	28 days ago	
debian.rules	844 Bytes	3 months ago	
packageName.dsc	301 Bytes	28 days ago	
qucs-0.0.19S.tar.gz	10.7 MB	28 days ago	

Showing 1 to 6 of 6 entries

Latest Revision
Vadim Kuznetsov (ra3xdh) committed 29 days ago (revision 26)
[Files changed](#) [Browse Source](#)

Comments for home:ra3xdh (0)



Qucs-S a maturing GPL software package: Qucs-S Help documentation

Search docs

- Chapter 1. Introduction
- Chapter 2. Basic Ngspice, Xyce and SPICE OPUS simulation
- Chapter 3. Spice4qucs subcircuits, macromodels and device libraries
- Chapter 4. Device and component modelling with algebraic equations
- Chapter 5. More advanced circuit simulation techniques.
- Chapter 6. Ngspice, Xyce and SPICE OPUS post-simulation data processing with Qucs-S and Octave
- Chapter 7. Qucs and SPICE simulation models that work with ngspice, Xyce and SPICE OPUS
- Chapter 8. Ngspice custom simulation technology
- Chapter 9. XSPICE standard components and library
- Chapter 10. XSPICE user written device models and library
- Chapter 11. Introduction to mixed analogue/digital simulation
- Chapter 12. Verilog-A compact semiconductor device modelling
- Chapter 13. RF simulation with Ngspice, Xyce and SPICE OPUS
- Chapter 14. Spice4qucs subcircuits



Qucs-S Help documentation

User Manual and Reference Material

Authors Mike Brinson (mbrin72043@yahoo.co.uk) and Vadim Kusnetsov (ra3xdh@gmail.com)

Copyright 2015, 2016

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation. A copy of the license is included in the section entitled "GNU Free Documentation License".

Contents:

- [Chapter 1. Introduction](#)
- [Chapter 2. Basic Ngspice, Xyce and SPICE OPUS simulation](#)
- [Chapter 3. Spice4qucs subcircuits, macromodels and device libraries](#)
- [Chapter 4. Device and component modelling with algebraic equations](#)
- [Chapter 5. More advanced circuit simulation techniques.](#)
- [Chapter 6. Ngspice, Xyce and SPICE OPUS post-simulation data processing with Qucs-S and Octave](#)



Qucs-S a maturing GPL software package: Ngspice, Xyce and SPICEOPUS built in components

R2
 $R=50 \text{ Ohm}$

R3
 $R=50 \text{ Ohm}$

C2
 $C=1 \text{ pF}$

L2
 $L=1 \text{ nH}$

R1
 $R=$

L1
 $L=$

K1
 $\text{Ind1}=L1$
 $\text{Ind2}=L2$
 $K=0.1$

Pr1
Pr2
 $\text{Num}=1$

S1
 $S=$

W1
 $W=$

S2

ICOUPLE1
 $A=$

V1
 $U=1 \text{ V}$

V2
 $U=1 \text{ V}$

SRC1
 $G=1 \text{ S}$

SRC2
 $G=1$

SRC3
 $G=1$

SRC4
 $G=1 \text{ Ohm}$

V3
 $U1=0 \text{ V}$
 $U2=1 \text{ V}$
 $T1=0$
 $T2=1 \text{ ms}$

V4
 $U=1 \text{ V}$
 $\text{TH}=1 \text{ ms}$
 $\text{TL}=1 \text{ ms}$

I1
 $I=1 \text{ mA}$

I2
 $I=1 \text{ mA}$

I3
 $I1=0$
 $I2=1 \text{ A}$
 $T1=0$
 $T2=1 \text{ ms}$

I4
 $U1=1 \text{ mA}$
 $\text{TH}=1 \text{ ms}$
 $\text{TL}=1 \text{ ms}$

I5
 $I1=0$
 $I2=1 \text{ A}$
 $T1=0$
 $T2=1 \text{ ms}$

V6
 $V=$

I6
 $I=$

B3
 $V=$

Eqn1
B1
 $I=$

PWL
V7
 $\text{PWL}=$

Eqn2
B2
 $I=$

SFFM
I7
 $I0=0$
 $Ia=1$
 $\text{Fc}=1$
 $\text{Mdi}=10$
 $\text{Fs}=500$

SFFM
I8
 $I0=0$
 $Ia=1$
 $\text{Fc}=1$
 $\text{Mdi}=10$
 $\text{Fs}=500$

ENL
E1
 $E=$

GNL
G1
 $G=$

V8
 $Va=1$
 $Vo=0$
 $\text{Mf}=500$
 $\text{Fc}=10\text{k}$
 $\text{Td}=0$

V5
 $U1=0 \text{ V}$
 $U2=1 \text{ V}$
 $T1=0$
 $T2=1 \text{ ms}$

Exp
I5
 $I1=0$
 $I2=1 \text{ A}$
 $T1=0$
 $T2=1 \text{ ms}$

V10
 $\text{Type}=2$
 $\text{Ts}=1\text{m}$
 $\text{Td}=0$
 $\text{Param1}=1$
 $\text{Param2}=0$

V11
 $\text{Vac}=$

O1
 $O=$

T1
 $Z0=50$
 $\text{Td}=0.25\text{sn}$
 $F=1\text{e9}$
 $\text{NI}=0.25$
 $V1=0$
 $I1=0$
 $V2=0$
 $I2=0$

D1
 $D=$

Q1
 $Q=$

Q2
 $Q=$

VTRN
V9
 $\text{Na}=20\text{n}$
 $\text{Nt}=0.5\text{n}$
 $\text{Nalpha}=1.1$
 $\text{Namp}=12\text{p}$
 $\text{Rtsam}=0$
 $\text{Rtscapt}=0$
 $\text{Rtsemt}=0$

X1
 $\text{File}=$
 spice

X2
 $\text{NPins}=3$
 $\text{Letter}=Z$

X3
 D2
 $\text{I1}=0$

OP1
 $G=1\text{e6}$

J1
 $J=$

J2
 $J=$

J3
 $J=$

J4
 $J=$

M1
 $M=$

M2
 $M=$

Z1
 $Z=$

Z2
 $Z=$

Z3
 $Z=$

sub
SUB1

lib

Precompiled CM-library
 XSP_CMlib1
 $\text{File}=\text{home/user/library.cm}$

XSPICE CodeModel
 XSP_CMod1
 $\text{File}=\text{sfunc.mod}$
 $\text{File}=\text{ifspec.ifs}$

S Domain Transfer Function
SDF1
 $A=A_XSDFT\text{mod}$

PWL controlled voltage source
XAPWL1
 $A=A_PWL\text{mod}$

CORE1
 $A=$

Qucs-S a maturing GPL software package: Available semiconductor device models

Type	Description	Level	Ngspice	Xyce	SPICEOPUS	
D	Legacy	1	X	X	X	
		2		X		
		3			X	
BJT	Legacy VBIC FBH HBI_X MEXTRAM	1	X	X	X	
		10,11,12		X		
		23		X		
		504,505			X	
JFET		1	X	X	X	
		2	X	X	X	
MESFET		1 (Statz)	X	X	X	
		2 (Ytterdel)	X			
MOSFET Legacy		1	X	X	X	
		2	X		X	
		3	X	X	X	
		4(BSIM1)	X		X	
		5(BSIM2)	X		X	
		6	X	X	X	
		BSIM3v2	47			X
			8	X		
			9		X	
		BSIM3v3	53			X
			49	X		
		BSIM4	60			X
			14	X	X	
			54	X		
		BSIM3SOIv1	55			X
		BSIMOlv2	56	X		X
			58,55,57	X		
			10	X	X	
		STAGSOI3	57			X
		UFSOI	58			X
SOI3	60	X				
UFET	7			X		
EKVv2p6	44	X		X		
HISIM2	61,68	X				
HISIM_HV	62,63	X				
VDMOS	18		X			
BSIM 6p1	77		X			
PSP 103p1	103		X			

Qucs-S a maturing GPL software package: Simulation control icons

dc simulation

DC1

transient simulation

TR1
Type=lin
Start=0
Stop=1 ms

ac simulation

AC1
Type=lin
Start=1 GHz
Stop=10 GHz
Points=19

Harmonic balance simulation

HB1
n=4

Fourier simulation

FOUR1
Sim=TR1
numfreq=10
F0=1 kHz
Vars=V(1)

Distortion simulation

DISTO1
Type=lin
Start=1 Hz
Stop=10 kHz
Points=100

Nutmeg script

CUSTOM1
SpiceCode=
AC LIN 2000 100 10MEG
let K=V(1)/V(2)

XYCE script

XYCESCR1
SpiceCode=
.AC LIN 2000 100 10MEG
.PRINT AC format=raw file=ac.txt V(1)

Nutmeg

NutmegEq1
Simulation=ac
y=1

Pole-Zero simulation

PZ1
Input=in 0
Output=out 0
TF_type=vol
PZ_mode=pz

._NODESET

Nodeset1
v(node1)=1

Equation

Eqn1
y=1

._INCLUDE

SpiceInclude1
File=~/.home/user/library.inc

._GLOBAL PARAM

SpGlobPar1
y=1

._PARAM

SpicePar1
y=1

._MODEL

SpiceModel1
Line_1 =.MODEL DIODE1 D(BF=50 Is=1e-13 Vbf = 50)

._OPTIONS

SpiceOptions1
GMIN=1e-12

Parameter sweep

SW1
Sim=
Type=lin
Param=R1
Start=5 Ohm
Stop=50 Ohm
Points=20

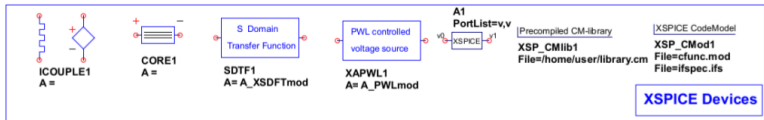
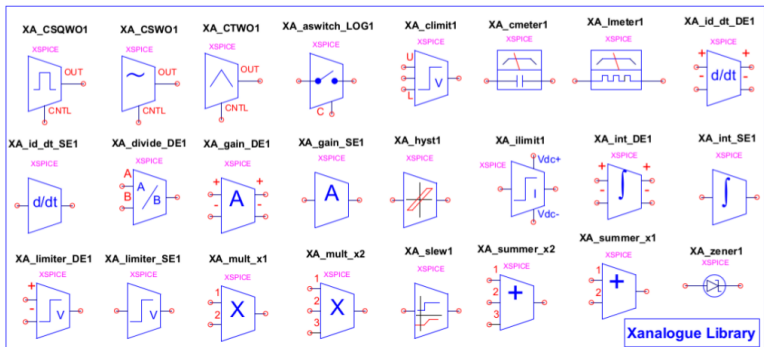
._IC

SpiceIC1
v(node1)=1

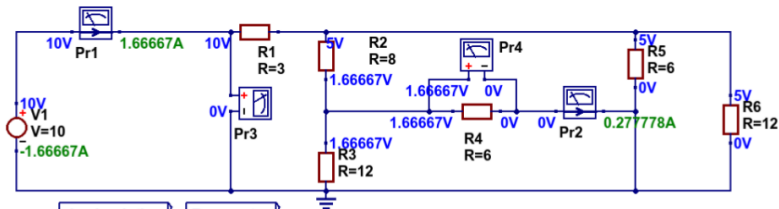
Noise simulation

NOISE1
Type=lin
Start=1 Hz
Stop=10 kHz
Points=100
Output=v(node1)
Source=V1

Qucs-S a maturing GPL software package: XSPICE analogue component models



Qucs-S a maturing GPL software package: Qucs-S extended circuit simulation Part 1. SPICE .OP to visual DC by pressing key F8



dc simulation

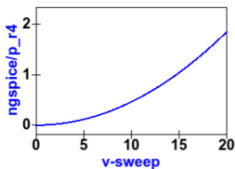
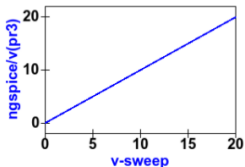
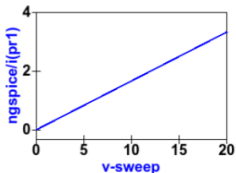
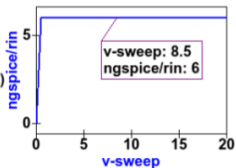
DC1

Parameter sweep

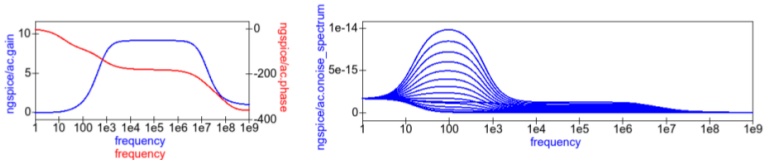
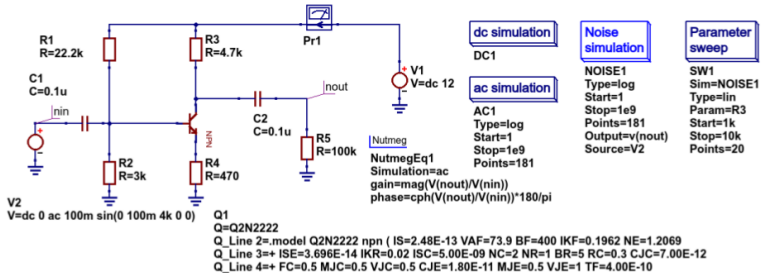
SW1
 Sim=DC1
 Type=lin
 Param=V1
 Start=0
 Stop=20
 Points=40

Nutmeg

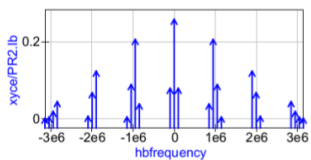
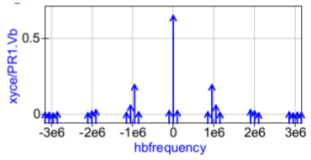
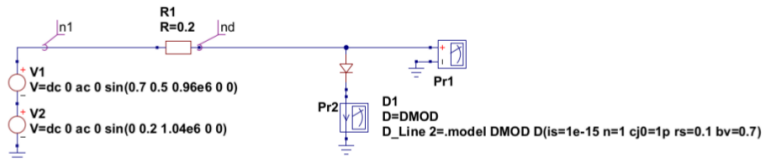
NutmegEq1
 Simulation=dc
 $Rin = V(pr3) / (vpr1\#branch + 1p)$
 $P_R4 = V(pr4) * vpr2\#branch$



Qucs-S a maturing GPL software package: Qucs-S extended circuit simulation Part 2. Noise spectral response

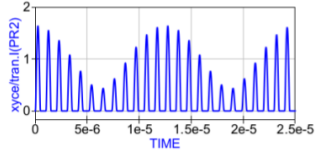
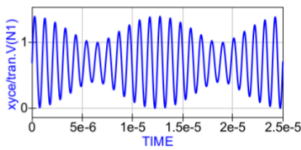


Qucs-S a maturing GPL software package: Qucs-S extended circuit simulation Part 3. Multi-tone Harmonic balance analysis



Harmonic balance simulation

HB1
 $f=0.95e6\ 1.05e6$
 $n=3,3$



transient simulation

TR1
 Type=lin
 Start=0
 Stop=25 us



Qucs-S a maturing GPL software package: Qucs-S extended circuit simulation Part 4. Direct support for SPICE libraries

The screenshot displays the Qucs-S interface with three main panels:

- Left Panel (Libraries):** A tree view under 'Libraries' with 'User Libraries' expanded to show 'ad822' selected.
- Center Panel (Code):** A text editor showing SPICE code for an AD822 op-amp circuit. The code includes component definitions, gain settings, and model parameters.
- Right Panel (Circuit):** A schematic diagram of an AD822 op-amp circuit. It features a 12V supply (U=12), a 1V supply (U=1), and a 10k resistor (R1). The output is shown as 1.99919V. A second op-amp (X2) is also present, with its output at 1.999595V.

Text Annotations:

- A red arrow points from the 'ad822' entry in the library to the SPICE code.
- A red arrow points from the SPICE code to a small schematic of the AD822 component.
- A red arrow points from the small schematic to the main circuit diagram.
- Red text at the bottom center reads: "Qucs-S now includes direct support for SPICE libraries".

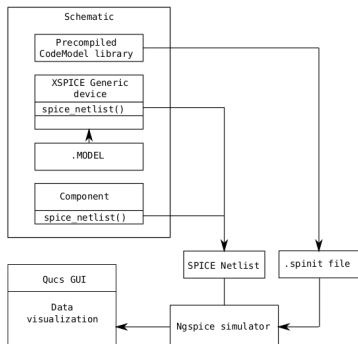
Qucs-S a maturing GPL software package: Modelling tool additions and new features

- Qucs-S includes for the first time a turn-key XSPICE "code level modelling" package for use with the Ngspice and SPICE OPUS circuit simulators,
- Qucs-S has also been extended to include a new Qucs/Octave integrated tool set for compact device model and circuit macromodel parameter extraction. The technique employed is based on data fitting and optimization using measured, or manufacturers published device data, compared against simulated circuit data.



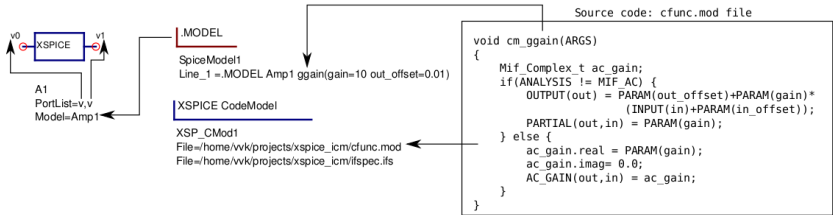
Qucs-S a maturing GPL software package: XSPICE Code Model support subsystem

- The XSPICE generic device component is the foundation for
 - Precompiled XSPICE device (*.cm) library support, and
 - Dynamic XSPICE "Code Model" compilation system which allows Code Model sources to be attached to a schematic and compiled automatically at simulation time.

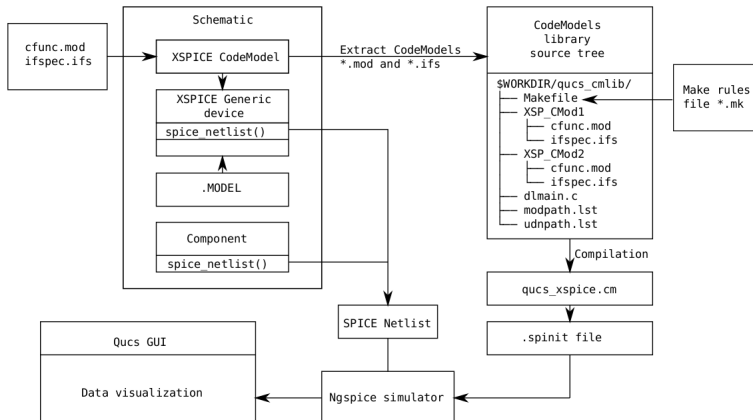


Qucs-S a maturing GPL software package: XSPICE Turn-Key Model Generation; compiler system building blocks

- The XSPICE generic device component is a building block for the construction of user-defined A-devices. It is defined by a comma separated port list, plus XSPICE port designators. These are attached to a SPICE .MODEL statement



Qucs-S a maturing GPL software package: XSPICE Turn-Key Model Generation; compiler system dataflow diagram

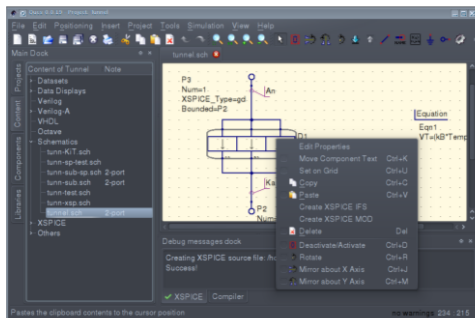


Main features of the XSPICE CodeModel synthesizer

Main features:

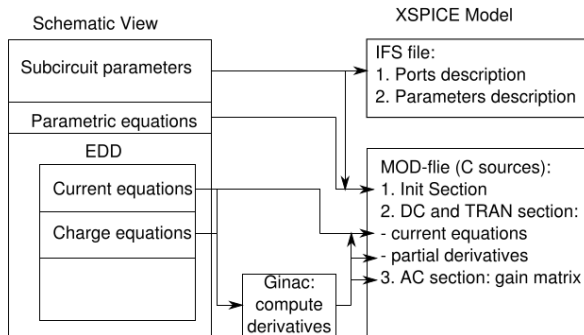
- Synthesize XSPICE C-code and interface description from EDD schematic view;
- Access to code synthesizer from right-click on the EDD component;
- Synthesizer generates a pair of MOD and IFS files from a single EDD;
- Automatic recognition of model parameters and dependent variables;
- Automatic symbolic computation of partial derivatives and AC gain matrix using Ginac embedded CAS library;

• XSPICE synthesizer context menu



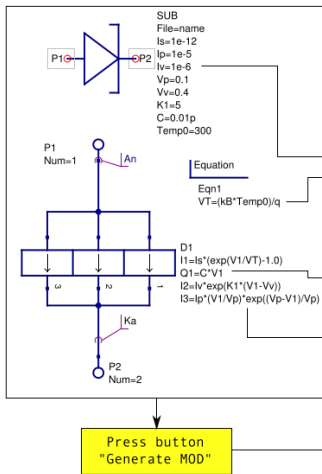
The structure of the XSPICE model synthesizer

- Ginac <http://www.ginac.de/> library is used for symbolic computation of partial derivatives and AC gain matrix;
- Interface description file (*.IFS) is generated from subcircuit symbol or from the EDD and attached equations;
- Model description (C-code *.MOD) is generated from individual EDDs;



Tunnel diode XSPICE model

Schematic view



XSPICE sources

```

/* XSPICE codemodel tunnel auto-generated template */
#include <math.h>
#include "xspice_mathfunc.h"

void cm_tunnel(ARG5)
{
  Complex_t ac_gain00;
  static double Is,Iv,K1,Vv,Ip,Vp,C,Temp0;
  static double VT;
  static double V1,V1_old;
  double Q0, cQ0; double delta_t;

  if(INIT) {
    Is = PARAM(is); Iv = PARAM(iv);
    K1 = PARAM(k1); Vv = PARAM(vv);
    Ip = PARAM(ip); Vp = PARAM(vp); C = PARAM(c);
    Temp0 = PARAM(temp0);
    VT=8.6173402243760290e-05*Temp0;
  }
  if (ANALYSIS != AC) {
    if (TIME == 0) {
      V1_old = V1 = INPUT(An_Ka);
      Q0=0.0; cQ0=0.0;
    } else {
      V1 = INPUT(An_Ka);
      delta_t=TIME-T(1);
      Q0 = (C)*(V1-V1_old)/(delta_t+1e-20);
      cQ0 = (C)/(delta_t+1e-20);
      V1_old = V1;
    }
    OUTPUT(An_Ka) = Is*( exp(1.0/VT*V1)-1.0)+
      exp(- ( Vv-V1)*K1)*Iv+exp(( Vp-V1)/Vp)*Ip/Vp*V1 + Q0;
    PARTIAL(An_Ka,An_Ka) = Ip*exp(1.0/Vp*(Vp-V1))/Vp+
      1.0/VT*exp(V1/VT)*Is-Ip*exp(1.0/Vp*(Vp-V1))/(Vp*Vp)*V1+
      Iv*K1*exp((V1-Vv)*K1) + cQ0;
  } else {
    ac_gain00.real = Ip*exp(1.0/Vp*(Vp-V1))/Vp+
      1.0/VT*exp(V1/VT)*Is-Ip*exp(1.0/Vp*(Vp-V1))/(Vp*Vp)*V1+
      Iv*K1*exp((V1-Vv)*K1);
    ac_gain00.imag = (C)*RAD_FREQ;
    AC_GAIN(An_Ka,An_Ka) = ac_gain00;
  }
}
    
```

Press button "Generate MOD"

Two-port testbench model: vacuum triode

- Triode is one of the simplest possible compact models. Triode equations:

$$I_{grid} = 0 \quad (11)$$

$$I_{plate} = \frac{1}{K_g} \left(V_{grid} + \frac{V_{plate}}{\mu} \right)^{1.5} \quad (12)$$

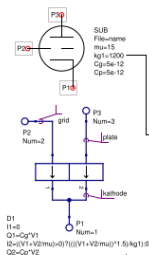
- Model parameters are: μ , K_g , C_{grid} , and C_{plate} ;
- Additional equations are required to implement XSPICE model (two partial derivatives and AC gain matrix):

$$g_{plate} = \frac{\partial I_{plate}}{\partial V_{plate}} = \frac{1.5}{\mu K_g} \sqrt{\frac{V_{plate}}{\mu} + V_{grid}} \quad (13)$$

$$g_{p.k.} = \frac{\partial I_{plate}}{\partial V_{grid}} = \frac{1.5}{K_g} \sqrt{\frac{V_{plate}}{\mu} + V_{grid}} \quad (14)$$

$$(G_{AC}) = \begin{pmatrix} j\omega C_g & g_{p.k.} \\ 0 & g_{plate} + j\omega C_{plate} \end{pmatrix} \quad (15)$$

Triode EDD implementation and auto-generated XSPICE Code



```

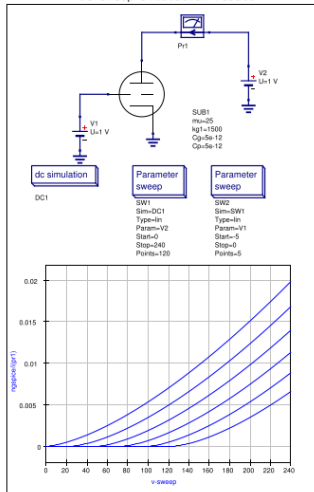
/* XSPICE codemodel triode auto-generated template */
#include <math.h>
#include "xspice_mathfunc.h"

void cn_triode(ARG5)
{
    Complex_t ac_gain00, ac_gain01, ac_gain10, ac_gain11;
    static double Cg,mu,kg1,Cp, V1,V2,V1_old,V2_old;
    double Q0, cQ0, Q1, cQ1; double delta_t;

    if (INIT) {
        Cg = PARAM(cg); mu = PARAM(mu); kg1 = PARAM(kg1); Cp = PARAM(cp);

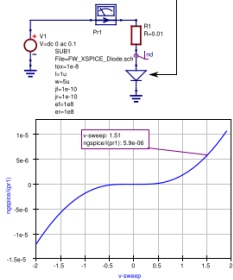
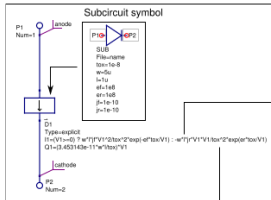
    if (ANALYSIS != AC) {
        if (TIME == 0) {
            V1_old = V1 = INPUT(grid_kathode); V2_old = V2 = INPUT(plate_kathode);
            Q0=0.0; cQ0=0.0; Q1=0.0; cQ1=0.0;
        } else {
            V1 = INPUT(grid_kathode); V2 = INPUT(plate_kathode);
            delta_t=TIME-T(1);
            Q0 = (Cg)*(V1-V1_old)/(delta_t+1e-20);
            cQ0 = (Cg)/(delta_t+1e-20);
            Q1 = (Cp)*(V2-V2_old)/(delta_t+1e-20);
            cQ1 = (Cp)/(delta_t+1e-20);
            V1_old = V1; V2_old = V2;
        }
        OUTPUT(grid_kathode) = 0.0 + Q0;
        OUTPUT(plate_kathode) = ((V1+V2/mu)>0.7)*pow(V1+V2/mu,1.50)/kg1/0.0 + Q1;
        PARTIAL(grid_kathode,grid_kathode) = 0.0 + cQ0;
        PARTIAL(plate_kathode,grid_kathode) =
            ((V1+V2/mu)>0.7)*1.50*Xpow(V1+V2/mu,0.50)/kg1/0.0;
        PARTIAL(plate_kathode,plate_kathode) =
            ((V1+V2/mu)>0.7)*1.50*Xpow(V2/mu+V1,0.50)/mu/kg1/0.0 + cQ1;
        } else {
            ac_gain00.real = 0.0;
            ac_gain00.imag = (Cg)*RAD_FREQ;
            AC_GAIN(grid_kathode,grid_kathode) = ac_gain00;
            ac_gain01.real = 0.0; ac_gain01.imag = 0.0;
            AC_GAIN(grid_kathode,plate_kathode) = ac_gain01;
            ac_gain10.real = ((V1+V2/mu)>0.7)*1.50*Xpow(V1+V2/mu,0.5)/kg1/0.0;
            ac_gain10.imag = 0.0;
            AC_GAIN(plate_kathode,grid_kathode) = ac_gain10;
            ac_gain11.real = ((V1+V2/mu)>0.7)*1.50*Xpow(V2/mu+V1,0.5)/mu/kg1/0.0;
            ac_gain11.imag = (Cp)*RAD_FREQ;
            AC_GAIN(plate_kathode,plate_kathode) = ac_gain11;
        }
    }
}
    
```

DC sweep simulation result



Fowler-Nordheim diode model

Fowler-Nordheim diode



Auto-synthesized XSPICE code

```

/* XSPICE codemodel fw_diode auto-generated template */
#include <math.h>
#define D_0_step(x) (0)
#define step(x) ((x)>0.071.0;(((x)+=0)70.5:0.8))
#define Xpow(x,p) pow(fabs(x),.p)
void ch_fw_diode(ARGS)
{
  Complex_t ac_gain#0;
  static double w,l,jf,tox,ef,jr,er;
  static double V1,V1_old;
  double Q0,cQ0;
  double delta_t;

  if(INIT) {
    w = PARAM(w);
    l = PARAM(l);
    jr = PARAM(jr);
    tox = PARAM(tox);
    ef = PARAM(ef);
    jr = PARAM(jr);
    er = PARAM(er);
  }
  if (ANALYSIS != AC) {
    if (TIME == 0) {
      V1_old = V1 = INPUT(anode_cathode);
      Q0=0;
      cQ0=0;
    } else {
      V1 = INPUT(anode_cathode);
      delta_t=TIME-T(1);
      Q0 = (3.45314388980000011*(l/tox*w)*(V1-V1_old)/(delta_t*1e-20));
      cQ0 = (3.45314388980000011*(l/tox*w)/(delta_t*1e-20));
      V1_old = V1;
    }
    OUTPUT(anode_cathode) = (V1==0)?
      jf*(V1*V1)*exp(-ef/V1*tox)/(tox*tox)*l:
      -jr*exp(er/V1*tox)*(V1*V1)*w/(tox*tox)*l + Q0;
    PARTIAL(anode_cathode,anode_cathode) = (V1==0)?
      2.0*1.0/(tox*tox)*l*exp(-tox*ef/V1)*jf*V1*w*1.0/tox*1*exp(-tox*ef/V1)*jf*ef*w:
      -2.0*1.0/(tox*tox)*l*jr*V1*w*exp(tox*er/V1)+1.0/tox*1*jr*er*w*exp(tox*er/V1) + cQ0;
  } else {
    ac_gain#0.real = (V1==0)?
      2.0*1.0/(tox*tox)*l*exp(-tox*ef/V1)*jf*V1*w*1.0/tox*1*exp(-tox*ef/V1)*jf*ef*w:
      -2.0*1.0/(tox*tox)*l*jr*V1*w*exp(tox*er/V1)+1.0/tox*1*jr*er*w*exp(tox*er/V1);
    AC_gain#0.imag = (3.45314388980000011*(l/tox*w)*RAD_FREQ);
    AC_GAIN(anode_cathode,anode_cathode) = ac_gain#0;
  }
}
    
```

Qucs-S : .FUNC entry: user-defined SPICE functions 1

- .FUNC pseudo-component is placed at the "SPICE specific section group"
- .FUNC entries prepend components description in the auto-generated netlist

The screenshot displays the Qucs-S software interface. On the left is a component library panel with sections for "SPICE specific sections", "PARAM S...", "GLOBAL...", "OPTI...", "IC Section", "NOESET...", "Nutmeg Eq...", "MODEL S...", ".INCLUDE...", and ".FUNC new... Include script". The main workspace shows a circuit diagram with an AC voltage source V1 (U=1 V, f=1000 kHz), a resistor R1 (R=R_s), and a capacitor C1 (C=C_p). The circuit has input terminals "in" and output terminals "out".

Below the circuit, there are two simulation configuration boxes:

- ac simulation**
AC1
Type=log
Start=100 Hz
Stop=1000 kHz
Points=121
- transient simulation**
TR1
Type=lin
Start=0
Stop=10u
Points=200

To the right of the circuit, there is a definition for a user-defined function:

.FUNC
amul(x,y)=(x*y)

Equation
Eqn1
Cp=1000p
Rs=amul(0.5,2k)
Kv=out.v/in.v

At the bottom right, a graph plots the magnitude of the transfer function |ngspice/hz| against frequency. The y-axis ranges from 0 to 1, and the x-axis ranges from 0 to 1e6 Hz. The curve shows a low-pass filter response, starting at 1.0 at 0 Hz and decaying towards 0 as frequency increases.

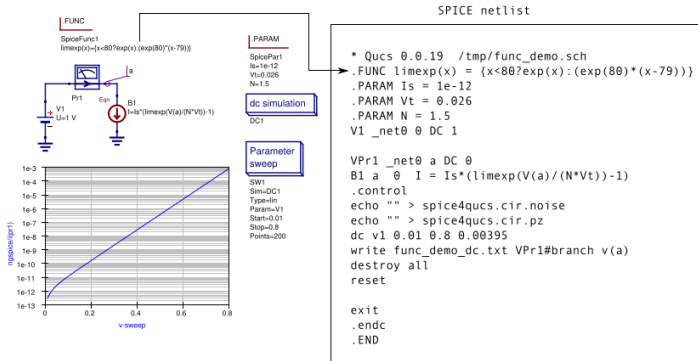
At the bottom of the window, the status bar reads "Ngspice no warnings 392 347".

Qucs-S : .FUNC entry: user-defined SPICE functions 2

- Diode model implementation with Ngspice and limexp() function:

$$I = I_s \left(\exp \left(\frac{V}{N V_t} \right) - 1 \right)$$

- Schematic and auto-generated SPICE netlist:



Include scripts: run some SPICE code before components initialized

- Include scripts allow to place some custom SPICE code (parameters, options, function definitions) before components initialization and edit this code manually.

The screenshot displays a SPICE simulation environment with a circuit diagram, a simulation configuration window, and a SPICE code editor.

Circuit Diagram: A simple RC circuit with an AC voltage source V_1 (1V, 1000 kHz), a resistor R_1 ($R=Rs$), and a capacitor C_1 ($C=Cp$). The input is labeled "in" and the output is labeled "out".

Simulation Configuration: The "Equation" window shows the following configuration:

- Eqn1: $Kv=out.vin.v$
- INCLUDE SCRIPT
- INCLSCR1
- SpiceCode=
- PARAM Cp=1000p
- PARAM Rs=2k

The "Component: Include script before simulation" dialog box is open, showing the "SPICE code editor" with the following code:

```
PARAM Cp=1000p
PARAM Rs=2k
```

SPICE Code Editor: The code editor shows the following SPICE code:

```
.include /home/xxx/spice/PS2.sch
PARAM Cp=1000p
PARAM Rs=2k
AC 1 0 DC 0 SIN(0 1 1000k 0 0) AC 1
IN out (R1)
OUT out (C2)
.control
echo "" > spice@ps2.cir.maine
echo "" > spice@ps2.cir.g1
ae dec 20 100 1000
let g1(out1)/f1(m)
write R1.g1.txt v(in) v(out) %t
destroy all
reset
ic00 0e-06 1e-05 0
write R2.txt I(R1) v(in) v(out)
destroy all
reset
.exit
END
```

Simulation Results: The "ac simulation" and "transient simulation" results are displayed below the circuit diagram.

ac simulation:

- AC1
- Type=log
- Start=100 Hz
- Stop=1000 kHz
- Points=121

transient simulation:

- TR1
- Type=in
- Start=0
- Stop=10u
- Points=200

Post-simulation data processing : 1. using Qucs

Equation blocks + simulation data sets → Data processing → Tables and plots

Constants: i, j, pi, e, kB, q

Immediate: 2.5, 1.4+j5.1, [1, 3, 4, 5, 7], [11, 12; 21, 22]

Ranges: Lo:Hi, :Hi, Lo:, :

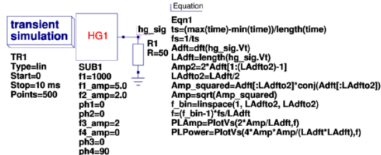
Logical operators: !x, x&& y, x||y, x^^y, x?y:z, x==y, x!=y, x<y, x<=y, x>y, x>=y

Number suffixes: E, P, T, G, M, k, m, u, n, p, f, a

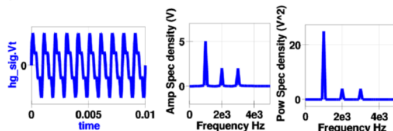
Matrices: M, M[2,3], M[:,3]

Arithmetic operators: +x, -x, x+y, x-y, x*y, x/y, x%/y, x^y

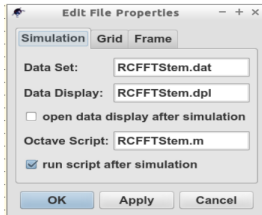
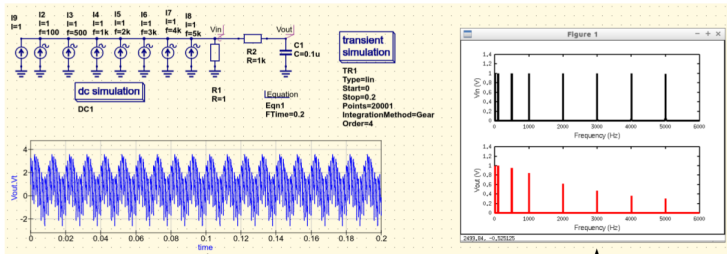
abs adjoint angle arccos arccosec arccot arcosech arcosh arccoth arcsec arcsin arctan arg arsech arsinh artanh
 avg besseli0 besselj bessely ceil conj cos cosec cosech cosh cot coth cumavg cumprod cumsum dB dbm dbm2w
 deg2rad det dft diff erf erfc erfcinv erfinv exp eye fft fix floor Freq2Time GaCircle GpCircle hypot idft ifft imag
 integrate interpolate inverse kbd limexp linspace ln log10 log2 logspace mag max min Mu Mu2 NoiseCircle norm
 phase PlotVs polar prod rad2deg random real rms Rollet round rtoswr rtoyr rtoz runavg sec sech sign sin sinc sinh
 sqr sqrt random StabCircleL StabCircleS StabFactor StabMeasure stdev step stos stoy stoz sum tan tanh
 Time2Freq transpose twoport unwrap variance vt w2dbm xvalue ytor ytos ytoz yvalue ztor ztos ztoy



Limitations: NO user defined functions or control loops



Post-simulation data processing : 2. using Octave



```

% test RC transient simulation plus FFT of output.
clear;
Data = "RCFFTstem.dat";
qdsset = loadQucsDataSet(Data);
[time] = getQucsVariable(qdsset, "time");
[Vin_Vt] = getQucsVariable(qdsset, "Vin.Vt");
[Vout_Vt] = getQucsVariable(qdsset, "Vout.Vt");
[FTime] = getQucsVariable(qdsset, "FTime");
showQucsDataSet(qdsset);
[Y1 , Y2, freq] = plotFFT2V("Stem", "Frequency (Hz)", 0, 6000,
    Vin_Vt, "Vin (V)", "black",
    Vout_Vt, "Vout (V)", "red", 4, FTime);
    
```



Post-simulation data processing : 3. using Python

```
# Basic Python script to demonstrate Qucs simulation with Qucsator
#
import subprocess
import parse_result as pr
import numpy as np
import matplotlib.pyplot as plt
#
from string import Template
#
def runSim():
    netfile = open("RC.net", 'r')
    outfile = open("sim_result.dat", 'w')
    process = subprocess.Popen("qucsator", stdin = netfile, stdout = outfile)
    process.wait()
    netfile.close()
    outfile.close()
#
# Undertake simulation and plot output results
#
runSim()
data = pr.parse_file("sim_result.dat")
x = data["acfrequency"]
y = np.abs(data["out.v"])
plt.loglog(x, y, '-kD')
plt.grid()
plt.title("RC voltage transfer function")
plt.xlabel("Frequency (Hz)")
plt.ylabel("Vout (V)")
plt.show()
```

Python script

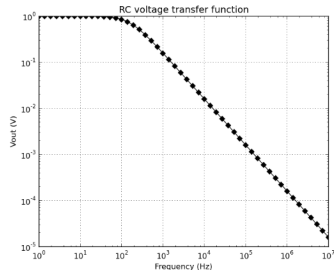
```
# File RC.net
#
Vac:V1 in gnd U = "1V" f = "1 kHz"
RR:R1 out in R = "1 k"
C:C1 out gnd C = "1 u"
.AC:AC1 Start = "1 Hz" Stop = "10 MHz" Points = "50" Type = "log"
```

Qucs netlist

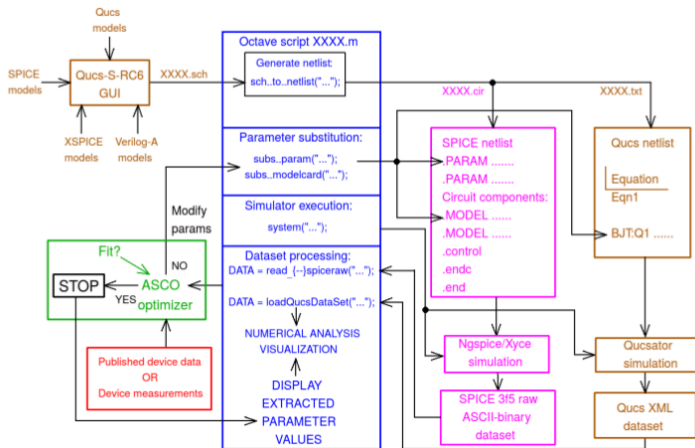
Qucsator

Parse Qucsator output data
in Python format

Plot with matplotlib



Qucs-S/Ngspice/Xyce Circuit Analysis and Compact Device Parameter Extraction from Manufacturers Data or Measurements Controlled by Octave Script Files: Part I Structure Diagram



Qucs-S/Ngspice/Xyce Circuit Analysis and Compact Device Parameter Extraction from Manufacturers Data or Measurements Controlled by Octave Script Files: Part II Octave Package

- The main purposes of Octave integration are:
 - Parameter substitution in Qucs and SPICE netlists,
 - Simulation process control from Octave,
 - Simulator output dataset (SPICE3f5-raw and Qucs XML) loading into Octave matrix structures,
 - Compact model parameter extraction from simulation and manufacturer's, or measured, data using curve fitting and optimization with the ASCO package.
- Example Octave package functions:
 - *subs_spice_netlist(FILE, PARAM, VALUE),*
 - *subs_qucs_netlist(FILE, PARAM, VALUE),*
 - *subs_spice_model_netlist(FILE, MODEL, PARAM, VALUE),*
 - *DATA = read_spiceraw(FILE).*

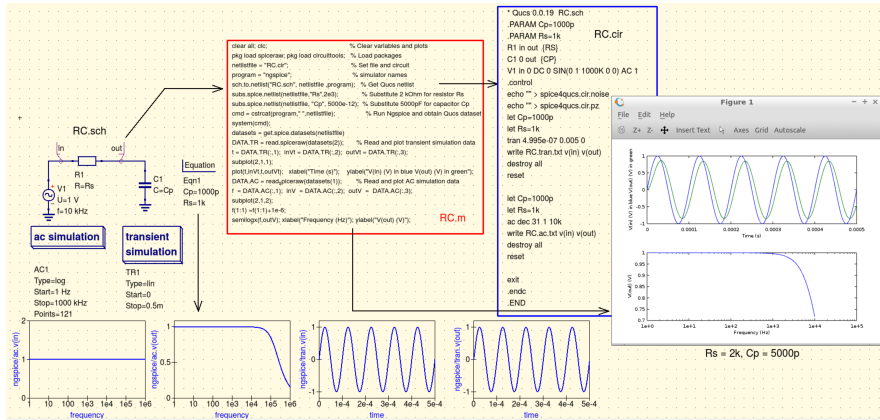
Where FILE – represents SPICE or Qucs netlist files

PARAM – represents a SPICE or Qucs variable or .MODEL parameter name

VALUE – represents a parameter value to replace its original quantity



Qucs-S/Ngspice/Xyce Circuit Analysis and Compact Device Parameter Extraction from Manufacturers Data or Measurements Controlled by Octave Script Files: Part III Simple Ngspice example



Qucs-S/Ngspice/Xyce Circuit Analysis and Compact Device Parameter Extraction from Manufacturers Data or Measurements Controlled by Octave Script Files: Part IV Simple Qucs example

transient simulation

TR1
Type=in
Start=0
Stop=0.5 ms

Equation
Eqn1
Cp=1000p
Rs=1k

```

clear all; clc;
pkg load spicewar; pkg load circuitools;
netlistfile = "RCQ.cir";
program = "qucs";
sch.to.netlist("RCQ.sch", netlistfile_program);
subs.qucs.netlist(netlistfile, "Rs",2e3);
subs.qucs.netlist(netlistfile, "Cp", 5000e-12);
run.qucsator(netlistfile, "RCQ.dat");
DATA = loadQucsDataSet("RCQ.dat");
showQucsDataSet(DATA);
t = DATA(3).data; inVt=DATA(5).data; outVt=DATA(6).data;
figure(1);
plot(inVt,outVt); xlabel("Time (s)"); ylabel("V(in) (V) in blue V(out) (V) in green");
                    
```

RCQ.m

```

# Qucs 0.0.19 RCQ.sch
R:R1 in out R="Rs" Temp="26.85" Tc1="0.0" Tc2="0.0" Tnom="26.85"
Eqn:Eqn1 Cp="1000p" Rs="1k" Exports="yes"
Vac:V1 in grnd U="1 V" f="10 kHz" Phase="0" Theta="0"
.TR:TR1 Type="in" Start="0" Stop="0.5 ms" Points="1001" IntegrationMethod="Trapezoida" Order="2"
InitialStep="1 ns" MinStep="1e-16" MaxIter="150" reitola="0.001" abstol="1 pA" vntol="1 uV"
Temp="26.85" LTEitola="1e-3" LTEabstola="1e-6" LTEfactor="1" Solvers="CircuitLU"
relaxTSR="no" InitialDC="yes" MaxStep="0"
C:C1 grnd out C="Cp" V="0"
                    
```

RCQ.cir

Rs = 2k
Cp = 5000p

```

RCQ
qucs -e -f "RCQ.sch" -o "RCQ.cir"
qucsator -f "RCQ.cir" -o "RCQ.dat"
DATA =
cell struct array containing the fields:
    name
    nameDep
    dep
    data
    len
Contents of data set:
Variable    Dependency    Length
-----
Rs          -              1
Cp          -              1
time       time           1001
V1.it      time           1001
in.Vt      time           1001
out.Vt     time           1001
                    
```

Octave Log

Qucs-0.0.19S : Qucs with SPICE 1

Qucs-S: Qucs with SPICE

Download links

The latest stable release is Qucs-0.0.19S. It is based on stable Qucs-0.0.19:

- Documentation at readthedocs.io
- Source tarball qucs-0.0.19S.tar.gz
- Debian repository (32 and 64 bit), built with openSUSE OBS:
 - Debian 8 "Jessie" at download.opensuse.org
 - Debian 7 "Wheezy" at download.opensuse.org
- Windows installer (Zipped EXE): qucs-0.0.19S-setup.zip

[\[Installation instructions...\]](#)

Contribution guide

Qucs-S is open for everyone's contribution. See [here](#) for contribution guide.

What is Qucs-S?

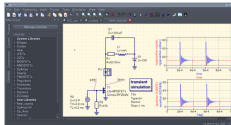
Qucs-S is a spin-off of the [Qucs](#) cross-platform circuit simulator. "S" letter indicates SPICE. The purpose of the Qucs-S subproject is to use free SPICE circuit simulation kernels with the Qucs GUI. It merges the power of SPICE and the simplicity of the Qucs GUI. Qucs intentionally uses its own SPICE incompatible simulation kernel Qucsator. It has advanced RF and AC domain simulation features, but the most of existing industrial SPICE models are incompatible with it. Qucs-S is not a simulator by itself, but it requires to use a simulation backend with it. The schematic document format of Qucs and Qucs-S are fully compatible. Qucs-S allows to use the following simulation kernels with it:

- [Ngspice](#) is recommended to use. Ngspice is powerful mixed-level/mixed-signal circuit simulator. The most of industrial SPICE models are compatible with Ngspice. It has an excellent performance for time-domain simulation of switching circuits and powerful postprocessor.
- [Xyce](#) is a new SPICE-compatible circuit simulator written by Sandia from the scratch. It supports basic SPICE simulation types and has an advanced RF simulation features such as Harmonic balance simulation.
- [Spice3-gnu](#) is developed by the Faculty of Electrical Engineering of the Ljubljana University. It is based on the SPICE-3F5 code
- Qucsator as backward compatible

News

- January 26, 2017 Qucs-S 0.0.19 is released! The first stable release. [Release announcement](#)
- November 8, 2016 Qucs-S RC8 released. [Release notes and download link](#)
- September 3, 2016 Qucs-S RC7 released. [Release notes and download link](#)
- May 15, 2016 Qucs-S RC6 released. [Release notes and download link](#)
- March 23, 2016 Qucs-S RC5 released. [Release notes and download link](#)
- January 31, 2016 Qucs-S RC4 released. [Release notes and download link](#)
- August 29, 2015 Qucs-S RC3 released.
- July 28, 2015 Qucs-S RC2 released.
- July 25, 2015 Qucs-S RC1 released.

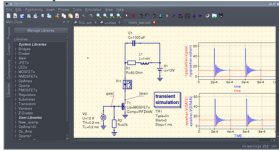
Simulation example with Qucs-S and Ngspice



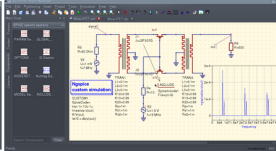
[\[More screenshots...\]](#)

Qucs-0.0.19S : Qucs with SPICE 2

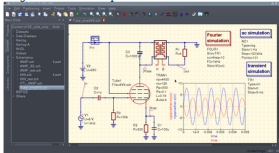
• MOSFET power switch



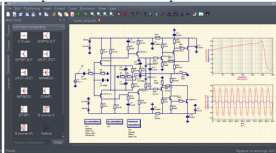
• JFET mixer



• Single-ended tube amplifier



• Hi-Fi bipolar transistor audio amplifier



[\(to top...\)](#)

Main features

- Backward compatible with Qucs by the component types and simulations
- Direct support of SPICE models from components datasheets. SPICE model could be added to schematic without any adaptation.
- Basic SPICE components: RCL, BJT, MOSFET, JFET, MESFET, switches;
- Advanced SPICE components: Equation-defined sources and RCLs, transmission lines;
- Direct support of SPICE Modelcards, SPICE sections (.IC, .NODESET);
- Parametric circuits (.PARAM) and SPICE postprocessor (Nutmeg)
- Basic SPICE simulations: DC, AC, TRAN;
- Advanced SPICE simulation: DISTO, NOISE, Spectrum analysis;
- Single-tone and Multitone Harmonic balance analysis with XYCE backend;
- Nutmeg script simulation: direct access to the SPICE code and construct your own simulation;
- XYCE script simulation type;

Plans for the next Qucs-S 0.0.20 release:

- Include an XSPICE code synthesizer;
- The support for .FUNC and Include scripts;
- Improvements in XYCE support: new components and .SENS analysis;
- Ngspice digital library;
- Synchronize code base with mainline Qucs and bugfixing;
- Release date scheduled: Summer 2017;

Source code available at:

- Stable and release candidates:
<https://github.com/ra3xdh/qucs/tree/qucs-s-stable>
- Development branch:
https://github.com/ra3xdh/qucs/tree/spice4qucs_current



Qucs-S : Plans for the future 2; more software features – Possible new directions for XSPICE synthesizer development

- Synthesize more complex XSPICE models: for example GaN HEMT, photodiodes and photoBJTs;
- Introduce a new generation of source based components, where their C code can be dynamically synthesized and compiled before simulation;
- Allow the linking of symbolic computation libraries to the XSPICE kernel to introduce symbolic computations at simulation time;
- Extend the number of devices in the pre-synthesized XSPICE libraries which are shipped with Qucs-S (currently very limited);

Qucs-S : Plans for the future 3; Adding Open Source Hardware to Qucs-S – current low cost open source measurement hardware choices

Labrador (Espotek)
\$29



Oscilloscope (2 channel, 750ksps)
Arbitrary Waveform Generator (2 channel, 1MSPS)
Power Supply (4.5 to 15V, 1.5W max, closed-loop)
Logic Analyzer (2 channel, 3MSPS per channel, with serial decoding)
Multimeter (V/I/R/C)
Software compatible with Windows, OSX, and Linux

ADALM1000 (Analog Devices)
£38



USB powered learning tool
Measuring and sourcing current (- 200 to +200mA) and voltage (0 to 5V) simultaneously on same pin
Oscilloscope (100 kSPS), function generator (100 kSPS)
PixelPulse 2 (open source) supports Windows, Linux, OS X
16-bit (0.05%) basic measure accuracy with 4 digit resolution
Source and sink (2-quadrant) operation
C, C++ and Python bindings
MATLAB Data Acquisition Toolbox Support
Open source hardware
ADALP2000 Analog Parts Kit offers wide selection of analog components and allows solderless breadboarding



Qucs-S : Plans for the future 4; Adding Open Source Hardware to Qucs-S – current low cost open source measurement hardware choices 2

adalm2000 (Analog Devices)
\$99



Two-channel USB digital oscilloscope

Two-channel arbitrary function generator

16-channel digital logic analyzer (3.3V CMOS and 1.8V or 5V tolerant, 100MS/s)

16-channel pattern generator (3.3V CMOS, 100MS/s)

16-channel virtual digital I/O

Two input/output digital trigger signals for linking multiple instruments (3.3V CMOS)

Single channel voltmeter (AC, DC, $\pm 20V$)

Network analyzer – Bode, Nyquist, Nichols transfer diagrams of a circuit.
Range: 1Hz to 10MHz

Spectrum Analyzer – power spectrum and spectral measurements (noise floor, SFDR, SNR, THD, etc.)

Digital Bus Analyzers (SPI, I²C, UART, Parallel)

Two programmable power supplies (0...+5V, 0...-5V)



Qucs-S : Plans for the future 5; Adding Open Source Hardware to Qucs-S – current low cost open source measurement hardware choices 3

Analog Discovery 2 (Digilent)
\$279



- Two-channel USB digital oscilloscope (1 M Ω , \pm 25 V, differential, 14-bit, 100 MSPS, 30 MHz+ bandwidth - with the Analog Discovery BNC adapter board, 1286-1073-ND sold separately)
- Two-channel arbitrary function generator (\pm 5 V, 14-bit, 100 MSPS, 20 MHz+ bandwidth - with the Analog Discovery BNC adapter board, 1286-1073-ND sold separately)
- Stereo audio amplifier to drive external headphones or speakers with replicated AWG signals
- 16-channel digital logic analyzer (3.3 V CMOS, 100 MSPS)
- 16-channel pattern generator (3.3 V CMOS, 100 MSPS)
- 16-channel virtual digital I/O including buttons, switches, and LEDs which are perfect for logic training applications
- Two input/output digital trigger signals for linking multiple instruments (3.3 V CMOS)

